



UNIVERSITY OF
LIVERPOOL

Computational Prediction of Gel Properties

Jack Simpson

2022

Thesis submitted in accordance with the requirements of the University of
Liverpool for the degree of Doctor in Philosophy by Jack Simpson

June 2022

**PGR Policy on Plagiarism and Dishonest Use of Data
PGR CoP Appendix 4 Annexe 1**

PGR DECLARATION OF ACADEMIC HONESTY

NAME (Print)	JACK SIMPSON
STUDENT NUMBER	201127505
SCHOOL/INSTITUTE	Department of Chemistry
TITLE OF WORK	Computational Prediction of Gel Properties

This form should be completed by the student and appended to any piece of work that is submitted for examination. Submission by the student of the form by electronic means constitutes their confirmation of the terms of the declaration.

Students should familiarise themselves with Appendix 4 of the PGR Code of Practice: PGR Policy on Plagiarism and Dishonest Use of Data, which provides the definitions of academic malpractice and the policies and procedures that apply to the investigation of alleged incidents.

Students found to have committed academic malpractice will receive penalties in accordance with the Policy, which in the most severe cases might include termination of studies.

STUDENT DECLARATION

I confirm that:

- I have read and understood the University's PGR Policy on Plagiarism and Dishonest Use of Data.
- I have acted honestly, ethically and professionally in conduct leading to assessment for the programme of study.
- I have not copied material from another source nor committed plagiarism nor fabricated, falsified or embellished data when completing the attached material.
- I have not copied material from another source, nor colluded with any other student in the preparation and production of this material.
- If an allegation of suspected academic malpractice is made, I give permission to the University to use source-matching software to ensure that the submitted material is all my own work.

SIGNATURE..........

DATE.....01/06/2022.....

Acknowledgements

My deepest thanks goes to the people who helped and supported me throughout the undertaking of this thesis. In particular, my thanks go out to Prof Neil Berry for his continual guidance and support throughout the entire PhD thesis and for the help in writing this thesis. His support throughout the PhD programme empowered me to forge my own path in this project, allowing me to explore interesting research areas which has allowed me to develop skills to support in my later career.

My thanks also go to Prof Dave Adams for his help and guidance with the gels background and for his feedback on chapters in this thesis. For the synthesis and rheology testing of the molecules in chapters 3 and 4 I would like to thank Lisa Thomson for her time and effort in carrying this out. For her help in the early stages of my work as a computational chemist I would like to thank Charmaine Chu for her support and guidance.

In preparation of Chapter 1, I would like to thank Chris Woodley for taking the time to give his thoughts and feedback.

My deepest thanks go to my family and friends for their support and encouragement throughout my studies. Finally, I am eternally indebted to my incredible fiancée Fatouma. Without her unwavering support and patience dealing with me at my most stressed I would not have been able to finish writing this thesis.

Abstract

Low molecular weight gels are an interesting class of soft solids formed when an external trigger causes a molecule to self-assemble into fibres. Interactions between molecules during the self-assembly are driven by non-covalent interactions such as hydrophobic interactions and hydrogen bonding. The fibres can entangle to immobilise the solvent in a fibrous matrix forming a soft solid. Since gel formation is driven by non-covalent interactions, the process of gelation is reversible meaning that removal or reversal of the trigger results in the solid returning to solution. The reversibility of this process leads to numerous potential applications in drug delivery, pollution clean-up and 3D printing.

However, issues persist around the design and synthesis of new gels. Although there are reports of machine learning models that can predict whether a molecule can form a gel, these models were built with hard to interpret descriptors.

Within this thesis Chapter 2 focuses on building classification models that show comparable performance to models presented in the literature but are built on interpretable descriptors. In this work, both Random Forest and Partial Least Squares show excellent performance across measures including Kappa, Balanced Accuracy and H-Measure on the test and validation sets using both Pipeline Pilot derived and RDKit derived interpretable descriptors.

In Chapter 3 work moves towards predicting the properties of those molecules that do form gels. In particular, the models are trained to predict the storage and loss moduli of the gels. Our Random Forest and k-Nearest Neighbours approaches show good performance in terms of R^2 and RMSE on the test and validation sets but show reduced predictive ability on an external set of molecules.

Following this, Chapter 4 uses Bayesian Additive Regression Tree Models (BART) models trained on storage and loss moduli that show comparable performance to the Random Forest model presented in Chapter 3. However, the BART models have the benefit of displaying uncertainty of the quantitative predictions through the predictive posterior distribution.

Finally, Chapter 5 deploys the Random Forest classifier and BART regression models in a De-novo generative approach to identify potential low molecular weight gelators with preferred rheological properties. Both an evolutionary algorithm and recurrent neural network approach are explored which generate molecules with user defined storage and loss moduli. However, these approaches require improvement to increase the diversity of the proposed molecules.

List of Abbreviations

<i>AI</i>	Artificial Intelligence
<i>AUC</i>	Area under the curve
<i>BA</i>	Balanced Accuracy
<i>C4.5</i>	C4.5 Classification Algorithm
<i>C5.0</i>	C5.0 Classification Algorithm
<i>ComFA</i>	Comparative molecular field analysis
<i>CoMSIA</i>	Comparative molecular similarity indices analysis
<i>DBC</i>	<i>N,N'</i> -dibenzoyl-L-cystine
<i>ECFP</i>	Extended Connectivity Fingerprint
<i>ECM</i>	Extracellular Matrix
<i>ED₅₀</i>	Effective dose that produces desired pharmacological effect in 50% of the population
<i>FCFP</i>	Functional Class Fingerprint
<i>Fmoc</i>	Fluorenylmethoxycarbonyl
<i>G'</i>	Storage Modulus
<i>G''</i>	Loss Modulus
<i>GLM</i>	Generalised Linear Model
<i>IUPAC</i>	International Union of Pure and Applied Chemistry
<i>kNN</i>	k-Nearest Neighbours
<i>k-NNG</i>	k-Nearest Neighbours Graph
<i>KRR</i>	Kernel Ridge Regression
<i>KS</i>	Kennard-Stone
<i>LMWG</i>	Low Molecular Weight Gel
<i>LSH</i>	Locality Sensitivity Hashing
<i>LVR</i>	Linear Viscoelastic Region
<i>MD</i>	Molecular Dynamics
<i>ML</i>	Machine Learning
<i>MST</i>	Minimum Spanning Tree
<i>NB</i>	Naïve Bayes
<i>NN</i>	Neural Network
<i>OECD</i>	The Organisation for Economic Co-operation and Development
<i>PCA</i>	Principal Component Analysis

<i>PES</i>	Potential Energy Surface
<i>PLS-DA</i>	Partial Least Squares – Discriminant Analysis
<i>QSAR</i>	Quantitative Structure Activity Relationships
<i>R²</i>	Squared correlation coefficient
<i>RF</i>	Random Forest
<i>RMSE</i>	Root mean squared error
<i>ROC</i>	Receiver Operating Characteristic curve
<i>SHAP</i>	Shapley Additive Explanations
<i>SMILES</i>	Simplified Molecular Input Line Entry Specification
<i>SVM</i>	Support Vector Machine
<i>TMAP</i>	Tree Manifold Approximation
<i>TNR</i>	True Negative Rate
<i>TPR</i>	True Positive Rate

Table of Contents

Acknowledgements	3
Abstract	4
List of Abbreviations.....	5
Chapter 1 - Introduction to Gels and Quantitative Structure Activity Relationships	11
1.1 Molecular Gels	12
1.1.1 Gel Formation.....	12
1.1.2 Applications of gels.....	14
1.2 Difficulties in finding new gels	18
1.3 Rheology.....	23
1.3.1 Mechanical Properties of Low Molecular Weight Gels	23
1.3.2 Rheology impacted applications.....	25
1.3.3 Process impact on mechanical properties	26
1.4 Computational Investigations for Gels	28
1.5 Machine Learning.....	30
1.5.1 Background to Machine Learning.....	30
1.5.2 Artificial Intelligence in Chemistry – Quantitative Structure-Activity Relationships	31
1.5.3 Background to QSAR.....	33
1.6 Modelling Algorithms	42
1.6.1 Bayesian Generalised Linear Models	43
1.6.2 k-Nearest Neighbours.....	44
1.6.3 Naïve Bayes.....	45
1.6.4 Partial Least Squares – Discriminant Analysis.....	45
1.6.5 Random Forest	46
1.6.6 C5.0	47
1.6.7 Neural Networks	47
1.6.8 Support Vector Machine.....	49
1.8 Performance metrics.....	50
1.8.1 Classification Performance Metrics	50
1.8.2 Regression performance metrics.....	52
1.9 Applicability domain	53
1.10 Y-randomisation.....	55
1.11 Model Interpretability	55
1.12 Thesis overview.....	57
1.13 References	58

Chapter 2 - Interpretable Classification Models to Predict Gelation State of Low Molecular Weight Peptides	66
2.1 Introduction.....	67
2.1.1 Importance of predictive models for gels	67
2.2 Experimental	67
2.2.1 Splitting of data.....	68
2.2.2 Software	68
2.2.3 Descriptors.....	69
2.2.4 Data Pre-processing	70
2.2.5 Descriptor Visualisation.....	71
2.2.6 Model selection	71
2.2.7 Model building.....	71
2.2.8 Model validation	72
2.2.9 Applicability domain.....	72
2.2.10 Model interpretation.....	73
2.3 Results and Discussion	74
2.3.1 Fingerprints as bits.....	74
2.3.2 Fingerprints as features.....	82
2.3.3 Python Classification Models.....	100
2.4 Conclusions.....	109
2.5 References.....	110
2.6 Appendix	113
2.6.1.1 Molecules in this dataset.....	113
2.6.2 <i>fp_as_bits</i> descriptors	118
2.6.2 <i>fp_as_features</i> descriptors.....	118
2.6.3 <i>py_class</i> descriptors	119
Chapter 3 – Rheology Regression Models	120
3.1 Introduction.....	121
3.2 Experimental.....	124
3.2.1 Dataset	124
3.2.2 Descriptor calculation	124
3.2.3 Data Splitting	127
3.2.4 PreProcessing.....	128
3.2.5 Model Building.....	129
3.2.6 Model Validation.....	129
3.2.7 Applicability domain.....	130

3.2.8 Model Interpretation	130
3.3 Results and Discussion	131
3.3.1 G' Models.....	131
3.3.2 G'' model based on G' model	183
3.4 Conclusions.....	197
3.5 References.....	198
3.6 Appendix.....	200
3.6.1 Dataset	200
Chapter 4 – Bayesian Regression Models.....	205
4.1 Introduction.....	206
4.1.1 Uncertainty and Probability.....	206
4.1.2 Conditional Probability.....	206
4.1.3 Bayes theorem.....	207
4.1.4 Posterior sampling	209
4.1.5 Bayesian Models	211
4.1.6 Chapter Overview	214
4.2 Experimental	214
4.2.1 Dataset	214
4.2.2 Descriptors.....	214
4.2.3 Processing.....	215
4.2.4 Model Building.....	215
4.2.5 Model Validation.....	216
4.2.6 Applicability Domain	217
4.2.7 Model Interpretation	217
4.3 Results and Discussion	218
4.3.1 G' BART model	218
4.3.2 G'' BART model	242
4.4 Conclusions.....	253
4.5 References.....	254
Chapter 5 – De-Novo Generative Models.	257
5.1 Introduction.....	258
5.1.1 DeNovo design.....	258
5.2 Experimental.....	267
5.2.1 CReM.....	267
5.2.1.1 Software used	267
5.2.2 Reinvent – Reinforcement Learning.....	270

5.3	Results and Discussion	272
5.3.1	Exploration of virtual Library	272
5.3.2	CReM	274
5.3.3	Reinvent RL	289
5.4	Conclusions	303
5.5	Outlook & Future Work	304
5.6	References	306
5.7	Appendix	310
5.7.1	Virtual Library Structures	310
	List of Supporting Information	315

Chapter 1 - Introduction to Gels and Quantitative Structure Activity Relationships

1.1 Molecular Gels

Gels are an incredibly interesting class of soft solids that are widely found throughout society. They consist of a three-dimensional network that encompasses a liquid. For hydrogels, this liquid is water. There has been much argument over the years about how to best define a gel but the IUPAC defines a gel as a “Non-fluid colloidal network or polymer network that is expanded throughout its whole volume by a fluid.”¹ Although these materials are predominantly covalently bonded polymer networks, they can also be formed using supramolecular networks. Figure 1.1 gives a general overview of supramolecular assembly.

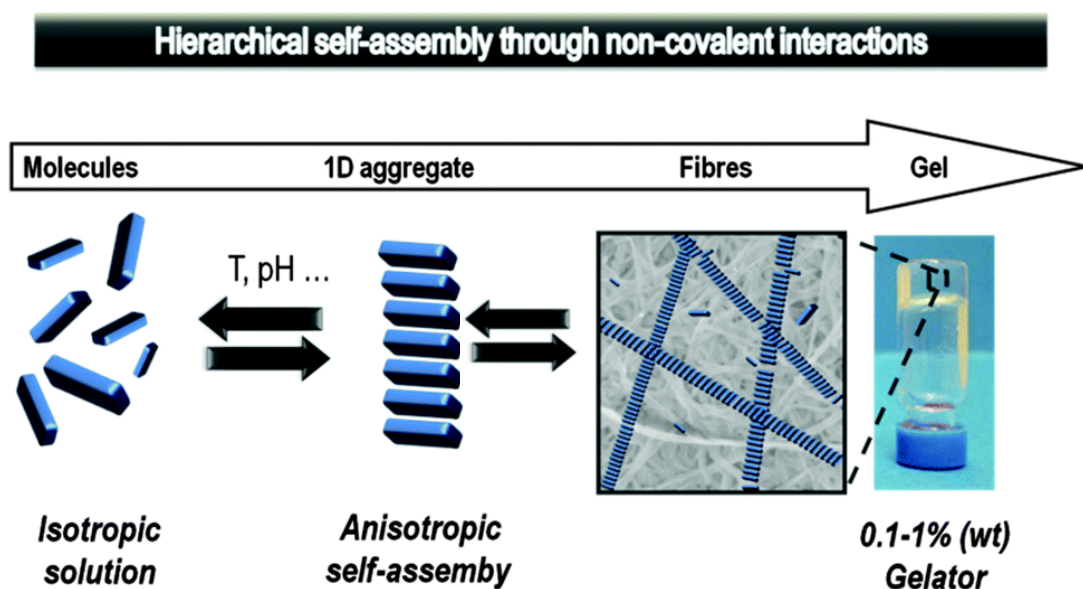


Figure 1.1: Example of molecular self-assembly in supramolecular gels adapted from Okesola et al.²

1.1.1 Gel Formation

Supramolecular gels, or Low Molecular Weight Gels (LMWGs), are similar to polymer gels except that it takes comparatively little effort to destroy the networks encompassing the fluid than it does in a polymer gel due to the non-covalent nature of the interactions in these systems.³ LMWGs are formed, initially, by the dissolution of the gelator (which usually for hydrogels take the form of amphiphilic molecules) in a suitable solvent (*i.e.* water). Common precursors include functionalised di/tripeptides (Figure 1.2) - typically a hydrophobic aromatic group bound to a dipeptide or tripeptide of either hydrophobic or hydrophilic amino acids. The hydrophilic portions of the molecule aid in solubility of the system and act as a barrier to crystal formation by retaining some solubility.⁴ Hydrophobic portions, particularly the aromatic groups, can aid in the self-assembly through hydrophobic

interactions with water and through π - π interactions with other precursor molecules. In water, hydrophobic interactions are the biggest driving force governing self-assembly in proteins.⁵

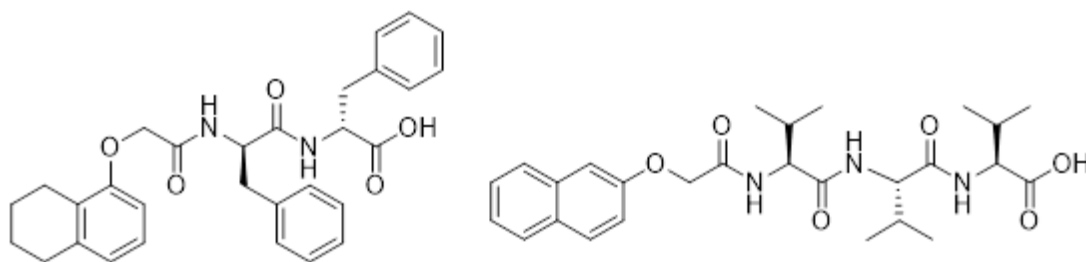


Figure 1.2: Example of dipeptide and tripeptide small molecules that successfully form gels.

Once in solution, an external trigger is applied to the solution, such that the solubility of the gelator is reduced in solution triggering self-assembly. Triggers that can result in gelation include: a temperature jump⁶, addition of metal ion⁷ or a pH change.⁸ Once the trigger is applied, the solubility of the precursor in solution is reduced until aggregation of the molecules becomes more favoured to dissolution. For example, gels containing peptide chains as shown in Figure 1.2 are typically prepared by first adding the molecule to water with the addition of a base such as NaOH which acts to deprotonate the terminal carboxylic acids of the peptide chain and increase the solubility in water to allow dissolution.⁹ Addition of glucono- δ -lactone (Figure 1.3), which slowly hydrolyses to gluconic acid, results in a uniform drop in pH, re-protonating the carboxylic acids and decreasing the solubility in water, driving self-assembly.

10

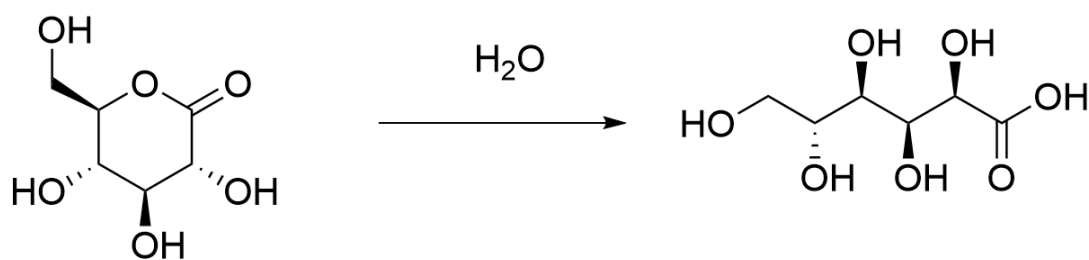


Figure 1.3: Hydrolysis of glucono- δ -lactone to gluconic acid

Once self-assembly is triggered, the molecules organise into one dimensional fibres that are typically on the scale of 1 μm in length. Once organised into fibres, these fibres can then act to immobilise the solvent in a fibrous matrix via branching or cross-linking. Surface tension between the solvent and the fibres helps to immobilise the solvent and results in the formation of a gel.¹¹

1.1.2 Applications of gels

Since the formation of the gel state is facilitated by non-covalent interactions, removal or reversal of the trigger that caused gelation will cause a breakdown in the fibrous matrix as the solubility of the constituent molecules increases, and the molecules return to solution. This reversibility has been leveraged as the key to a range of potential interesting applications of these materials.

One of the most promising applications of these materials is within biology and medicine, particularly in drug delivery and cell culturing. In cell culturing, cells are typically grown on top of a stiff (sometimes glass) film which only allow cells to receive signals and differentiate in two-dimensions when the cells are sat on top of the plate.¹² However, in the body, cells can differentiate in the extracellular matrix (ECM) in three-dimensions and groups have utilised gels as a potential mimetic for the ECM for cell growth. Alakpa *et al.*¹³ designed a range of Fmoc based supramolecular gels (molecules shown in Figure 1.4) of varying stiffness and compared the differentiation of stem cells with a glass control. Interestingly, they find that the cells differentiated via osteogenic, neuronal and chondrogenic pathways depending on the medium within which they were cultured.

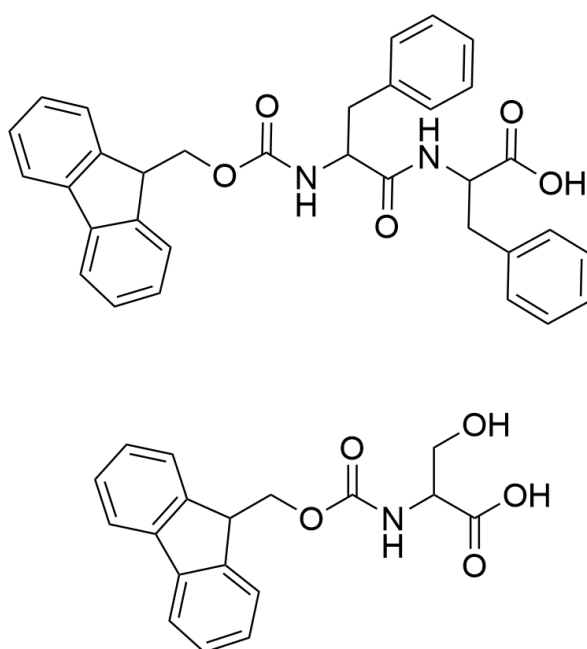


Figure 1.4: Fmoc based gels exhibiting varying stiffness in the cell differentiation work by Alakpa *et al.*¹³

Another approach¹⁴ with similar results uses functional groups within the hydrogel to control differentiation of stem cells down targeted pathways through careful consideration for the nature of the functional group.

Not only can these gels be loaded with stem cells for cell culturing, but they have also been loaded with drugs or a gel formed from the drug for selective release of therapies at the site of action. Nandi *et al.*¹⁵ synthesised a range of peptides with long alkyl chains (Figure 1.5) and tested their antimicrobial activity against both gram positive and gram negative bacteria and found some peptide based hydrogels that exhibit intrinsic antimicrobial activity. Others have attempted to repurpose commercially available drugs as gels.

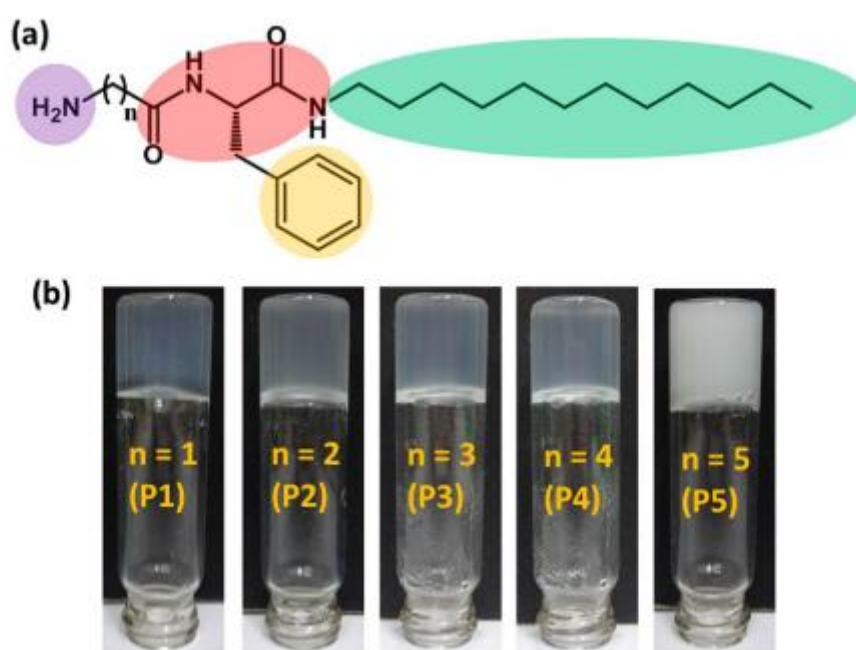


Figure 1.5: The long alkyl chain gels screened for antibacterial activity in Nandi *et al.* Figure adapted from [15]

Vancomycin is one such example of a drug that has been repurposed as a gel.¹⁶ An important antibiotic, Vancomycin can be gelled in water by structurally modifying the drug with a pyrene residue at the C-terminus (Figure 1.6), causing the drug to become a hydrogelator. The gel version of the antibiotic was found to be incredibly potent (between 0.125–2 µg.mL) which is 8- to 11-fold dilutions lower than free vancomycin. Other important drugs modified to form gels, but still retain some activity include Ibuprofen. Addition of a glycine-glycine dipeptide transformed ibuprofen into a gelator which could be cleaved enzymatically to return ibuprofen at the site of action.¹⁷ Another example is paracetamol which can be converted to a hydrogel pro-drug which can be loaded with drugs or stem cells or cleaved to re-form the active drug.¹⁸

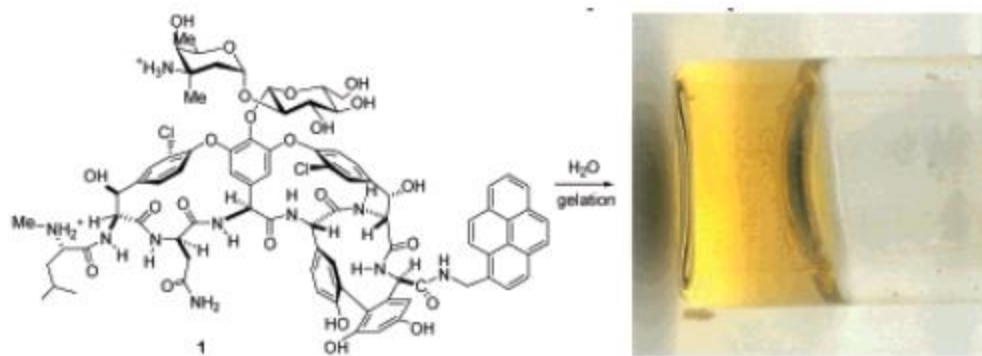


Figure 1.6: Pyrene modified Vancomycin as reported in Xing *et al.*¹⁶ Figure adapted from [16]

Sáez *et al.*¹⁹ have utilised the versatility of supramolecular hydrogels by attaching a drug candidate to a self-assembly fragment via a spacer (Figure 1.7). Utilising a lysine-based gelling moiety, enzymatic cleavage of the *p*-aminobenzyloxycarbonyl linker releases the phenylethylamine (model drug in Figure 1.7) and benzylamine (self-immolative linker in Figure 1.7) from the lysine gel. Following this, self-immolation of the phenylethylamine and benzylamine moiety allows for a sustained release of the drug into the site of action.

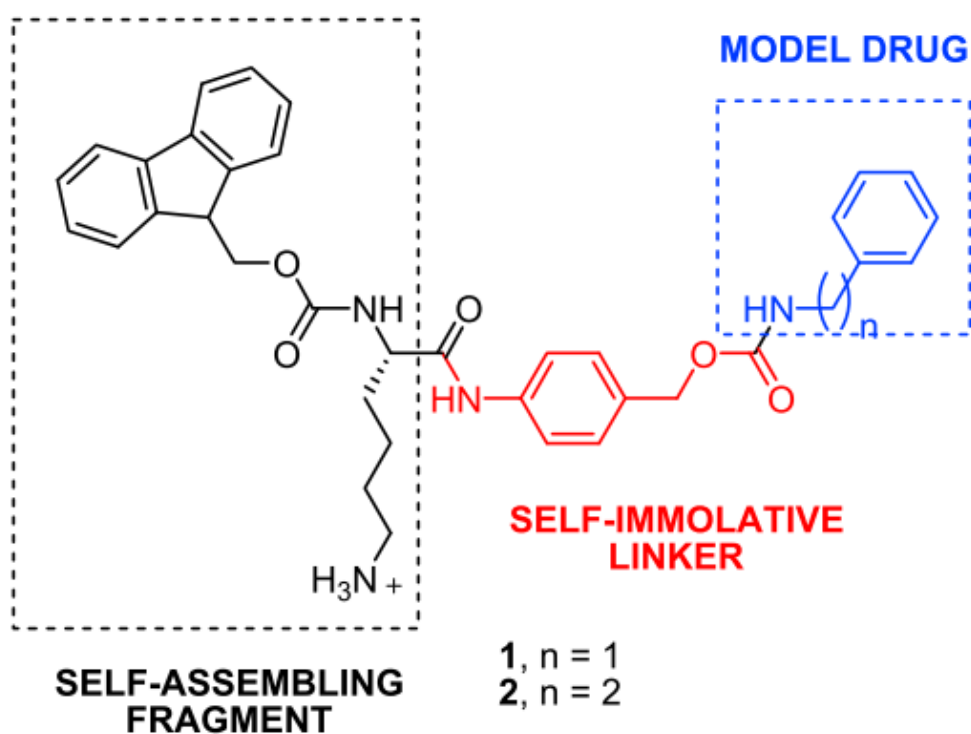


Figure 1.7: Lysine based gel reported by Sáez *et al.*¹⁹ for sustained release of benzylamine and phenethylamine. Figure adapted from [19]

Li *et al.*²⁰ modified olsalazine (Figure 1.8), an anti-inflammatory prodrug of 5-aminosalicylic acid, by appending a tripeptide moiety which forms a supramolecular hydrogel. They found that reduction of the hydrogel via a pH change afforded the 5-aminosalicylic acid drug in solution.

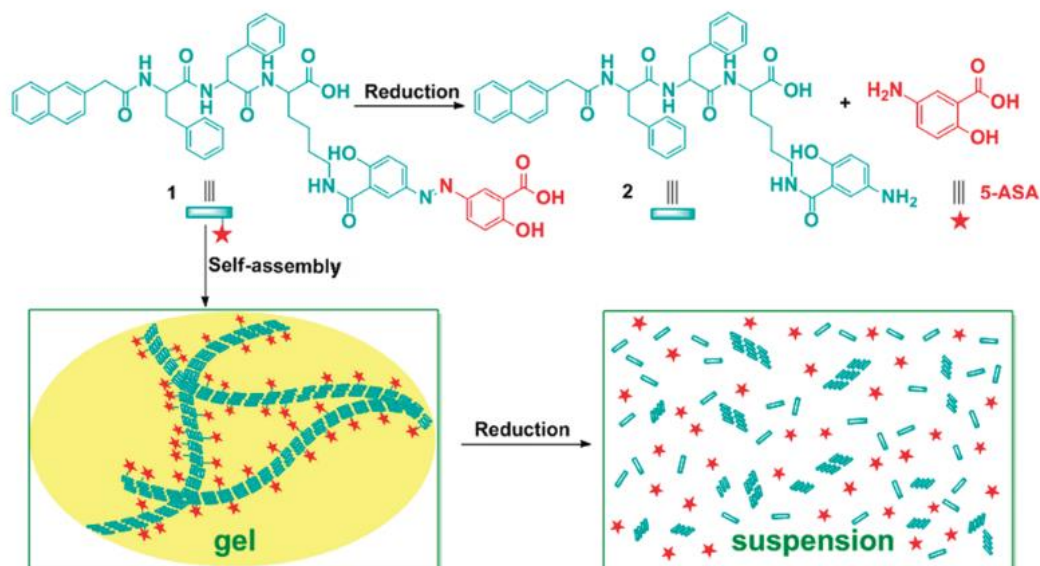


Figure 1.8: Figure from Li *et al.*²⁰ showing the olsalazine tripeptide moiety and the subsequent reduction to release olsalazine in solution. Figure adapted from [20]

In the case of vancomycin, not only has it been successfully converted into a hydrogelator, it has also been loaded into polymeric niosomes which were found to have a 2.5 fold increase in activity against MRSA than the free solution.²¹ Not only have drugs been loaded into polymer gels, they have successfully been loaded into supramolecular hydrogels also.

LMWGs have been investigated for their ability to act as carrier systems for drug delivery. Friggeri *et al.*²² loaded *N,N'*-dibenzoyl-L-cystine (DBC, Figure 1.9) LMWGs with both 8-aminoquinoline (AQ) and 2-hydroxyquinoline (HQ) and investigated the kinetics of their release from a cysteine based hydrogel. They found that hydroxyquinoline was released 7x faster than aminoquinoline – attributing this to stronger interactions between drug and gelator, hypothesising that fine-tuning of drug-gelator interactions can help achieve optimal release of a drug from a gel.

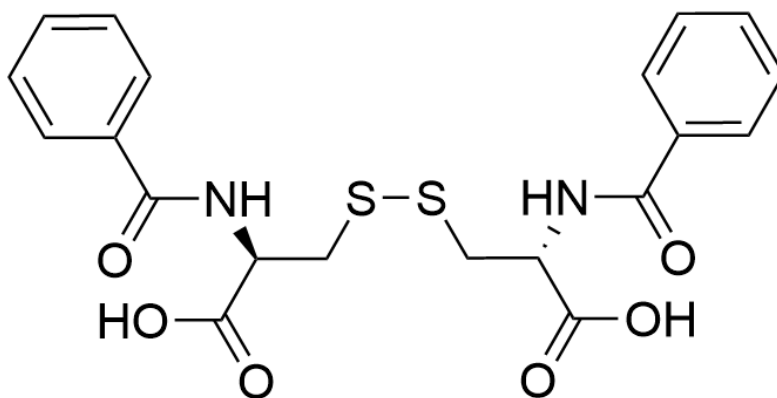


Figure 1.9: DBC molecule used to form the gels loaded with AQ and HQ in Friggeri *et al.*²²

The potential applications of these systems are not limited to the medical field. Groups have leveraged the stimuli responsive nature of these systems in far reaching applications, such as pollution removal. Gels have been developed to selectively gelate to remediate oil spills^{2,23} and remove ions and dyes within wastewater.²⁴

The potential versatility of these materials is further evident in their potential uses as sensors, to selectively gelate in the presence of glucose to detect glucose levels²⁵ or, similar to cell culturing, act as the environment for enzymatic sensing using, for example, Ca probes or pH probes.²⁶

1.2 Difficulties in finding new gels

The plethora of potential applications for these materials has caused them to receive increasing interest. However, the formation of gels has been described as a serendipitous process, and no design rules exist to successfully form gels. Although, as described above, the inclusion of amphiphilic moieties with aromatic anchors are ever present in those that successfully gel, the ability of a molecule to gelate is finely balanced, and small modifications to structures and solvent systems can dramatically alter their ability to gelate.

Work by Gronwald *et al.*²⁷ focused on incorporating groups into gelators that are known to promote self-assembly, namely sugars.²⁸ In the work, a range of Methyl benzylidene sugars were prepared and their gelation ability probed in a range of solvents. Interestingly, they find that the same molecules can form a gel in one solvent and forms a precipitate in structurally similar solvents. For example, the sugar

in Figure 1.10 successfully forms a gel in methylcyclohexane. However, in the structurally similar solvent cyclohexane – the attempt to gelate results in formation of a precipitate.

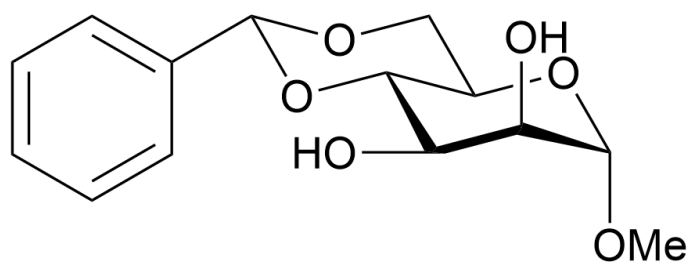


Figure 1.10: Methyl benzylidene sugar that successfully forms a gel in methylcyclohexane but produces a precipitate in cyclohexane

Other work has also shown the effects solvent can have on the formation of gelation in organic solvents. Here²⁹, a dendritic precursor (Figure 1.11) successfully gels in 1,2-dichloroethane but remains in solution when in 1,1,2,2-tetrachloroethane. These results suggest that solvents play an important role in the molecular aggregation that facilitates gel formation and needs to be carefully considered when considering the composition of the gelation system.

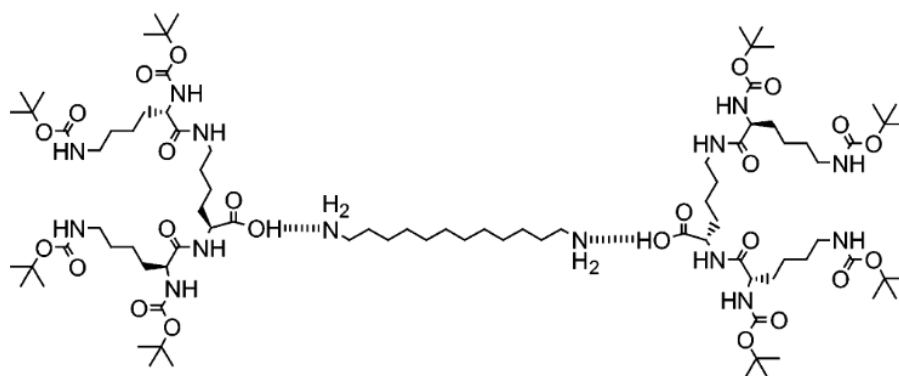


Figure 1.11: Dendritic precursor which shows different gelation ability in similar solvents. Figure adapted from [28]

Although using water as the solvent for a hydrogel system reduces this complexity somewhat, complexity remains in the form of structural modification and stereoisomerism. The similarity principle states that similar molecules exhibit similar properties.³⁰ However, in the case of gels, even minor modifications to a molecule can prevent molecules forming gels.

This can be seen in the work by Awhida *et al.*³¹ which probed the effect of backbone modifications on the gelation ability for a range of dipeptides. They found that modification of a single R group from H to CH₃ in the peptide (Figure 1.12) was enough of a structural modification to cause the molecule to no longer gel and instead, precipitate, irrespective of the gelation trigger used.

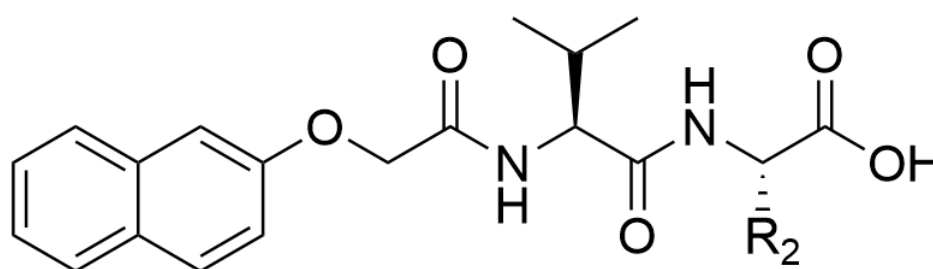


Figure 1.12: Modification of R₂ from H to CH₃ causes the molecule to no longer gel in both a pH and dimethyl sulfoxide trigger.

From within the study of the cyclohexane solvents above²⁸, anomers of the same molecule (Figure 1.13) show differing gelation ability, even in the same solvent. This effect has also been reported elsewhere³², where stereoisomers of a F₂Phe moiety (Figure 1.14) in water were subject to a pH trigger. The results were a solution remaining for the 2S,5R isomer in Figure 1.14 yet a gel formed for the 2R,5S isomer.

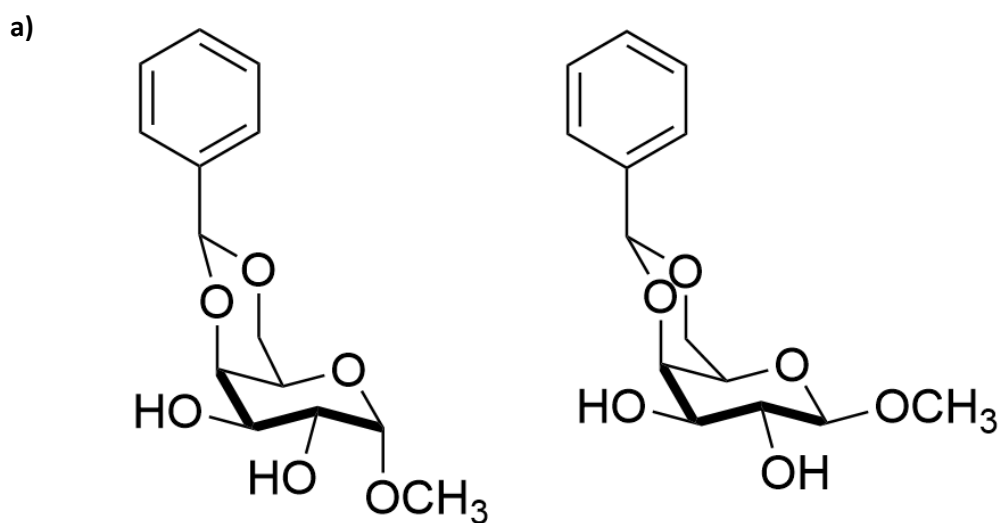


Figure 1.13: Anomers presented in [27] that show differing gelation ability in diethyl ether

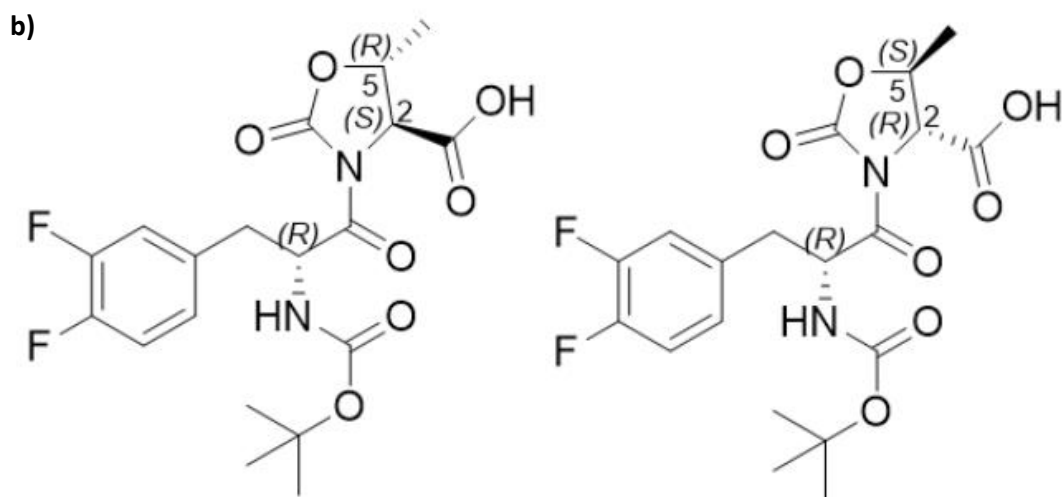


Figure 1.14: Stereoisomers that show differing gelation ability in water using a pH trigger in [32]. Left isomer (2S, 5R) forms a solution whereas right (2R, 5S) successfully gels.

The fine balance of non-covalent interactions that govern the ability of molecules to form gels demonstrated above has made it difficult to iteratively modify known gelators in the search for new examples. However, the issue is further complicated by the type of trigger used to initiate gelation. Molecules have been shown to form a gel using one trigger and fail to gelate, or form gels with different properties, when another trigger is applied.³¹

For example, the gelator 2NapFF (Figure 1.15a) can form a gel using a pH trigger, a metal trigger or through the addition of water to a solution of the gelator in DMSO. However, the molecule fails to form a gel from a simple temperature jump.³ This has been shown elsewhere with other naphthalene based

dipeptides such as the naphthalene based molecule in Figure 1.15b, where the analogues studied all formed a gel using a pH trigger, but some fail to form gels in the formation of certain metal triggers.³³

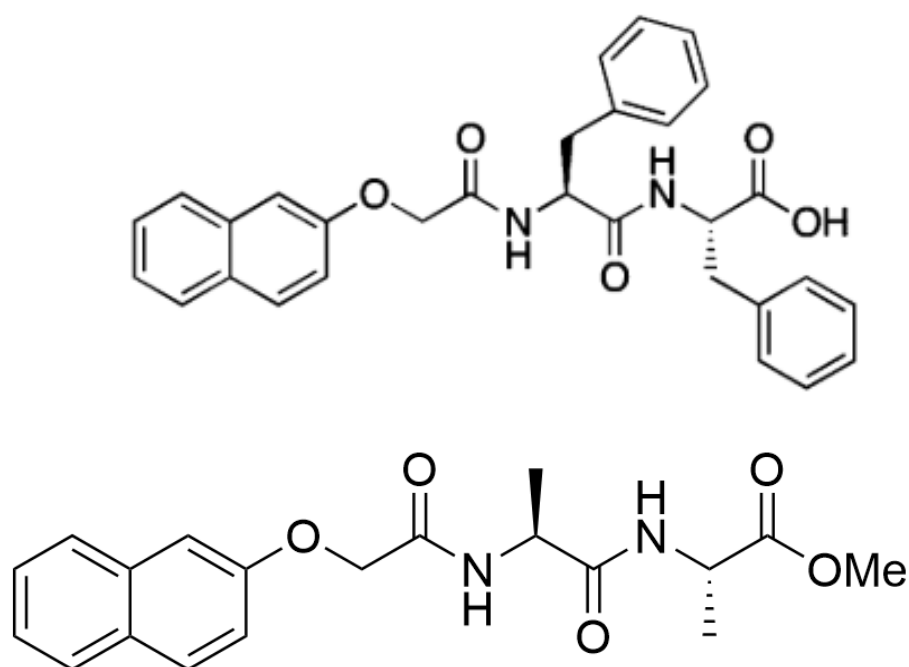


Figure 1.15: a) 2NapFF gelator that forms a gel under a pH trigger, metal trigger and water addition but doesn't with a temperature jump b) Naphthalene based peptide that forms a gel using a pH trigger but fails to gel using calcium nitrate

1.3 Rheology

1.3.1 Mechanical Properties of Low Molecular Weight Gels

In order to determine whether the material you have created after applying the trigger is a gel and not a very viscous liquid, rheology measurements can be carried out on the material to assess its viscoelastic properties. Oscillatory rheology is the investigation into the flow properties of a material and is used with gels to quantify the elastic nature of the gel, the storage modulus (G') – how much energy the material stores during deformation and the viscous nature, the loss modulus (G'') – the measure of the resistance to flow of the material.

Rheological results for a material are typically reported at strain values within the viscoelastic region (LVR) of the material. The LVR is where the viscoelastic properties of a material are independent of the applied strain. Provided the strain applied is within the LVR, the applied strain will be insufficient to break the fibrous network. Therefore, strain sweeps are used to determine the critical strain, the strain above which the fibrous network in the gel would begin to fail.

When carrying out a strain sweep, the sample is sandwiched between two parallel plates (Figure 1.16). A shear strain is applied to the top plate by oscillation of the top plate at a constant frequency while keeping the bottom plate stationary. The displacement of the material caused by the top plate rotation is applied as a percentage of the total strain of the material (where 100% would result in a material displacement equal to the width of the sample) from 0.01% strain up to 1000%.

The applied deformation of the sample causes a resultant stress in the material as it opposes the applied deformation. The stress induced by an applied strain is measured by a transducer in the rheometer as the torque applied by the material to the stationary bottom plate.

For a purely elastic material, there is no phase difference between the sinusoidal strain applied and the measured stress in the material. However, in a purely viscous material, there is exactly a 90° phase difference between the applied strain and the measured stress.

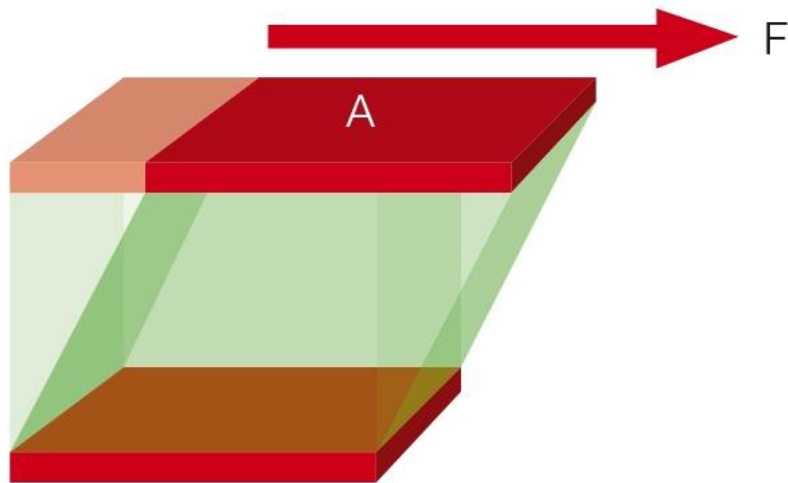


Figure 1.16: Example of a shear strain being applied from the top plate and the resultant stress of the material (Taken from: <https://wiki.anton-paar.com/en/basics-of-rheology/>)

Since gels are viscoelastic materials, the stress is measured somewhere between the two extremes. We can calculate the shear storage and loss modulus from the stress (σ) and strain (ϵ) using equations **1.1** and **1.2**.

$$G^* = \frac{\sigma}{\epsilon} \quad (1.1)$$

Where G^* is the complex modulus, the resistance to deformation of the material, and can also be expressed in terms of G' and G'' as:

$$G^* = \sqrt{G'^2 + G''^2} \quad (1.2)$$

Where G' and G'' can be described using equations **1.3** and **1.4**

$$G' = \frac{\sigma}{\epsilon} \cos(\delta) \quad (1.3)$$

$$G'' = \frac{\sigma}{\epsilon} \sin(\delta) \quad (1.4)$$

Where δ is the phase lag between the sinusoidal stress and sinusoidal strain. G' and G'' are inherently linked properties of the material as they both contribute to the complex modulus, the overall stiffness of the material.

For a gel, the storage modulus is typically an order of magnitude greater than the loss modulus.³⁴ Indeed, we can calculate the ratio of the loss modulus to the storage modulus, the $\tan(\delta)$, and when $\tan(\delta) < 0.1$, the material can be considered a gel.

1.3.2 Rheology impacted applications

Assessing the resistance to deformation (stiffness) and resistance to flow (viscosity) of a gel is imperative as it has been shown to be closely linked to the potential applications of the gel. For example, in the above example of the cell culturing work carried out in a range of gels of varying stiffness, it was shown that the stiffness of the gel controls the differentiation pathway.¹³

Also, the stiffness of the ECM is known to vary throughout the body, ranging from around 0.2-4 kPa in lung, liver and kidney tissue to 100 kPa in bone³⁵. Therefore, the stiffness of a gel system acting as a mimetic for the ECM for these tissues must have a stiffness that is compatible for cell growth.

Another mechanical property dependent application of gels is their use as injectable biomaterials. This is because the process of injection and forcing the gel through a needle is itself applying a shear strain to the gel. Therefore, the gel needs to exhibit the correct mechanical profile to withstand the shear strain of injection and reform its solid-like structure.

Wang *et al.*³⁶ reported a nucleoside based gel 2-FA (Figure 1.17) which exhibited a storage modulus of 1 MPa which they claim is the strongest gelator built from a monomer with MW < 300. Shear thinning experiments of the 2-FA through injection, showed that the initially formed gel could successfully be injected and at 5 wt%, immediately reformed into a gel post injection. In rat studies, the gel was administered via a subcutaneous injection and the degradation of the gel followed. Although no characterisation of the gel could be performed post-injection, a solid-like mass remained post injection that degraded within 6 hours. The injected gel showed no adverse reactions and even displayed antibacterial activity against both gram positive and gram negative bacteria.

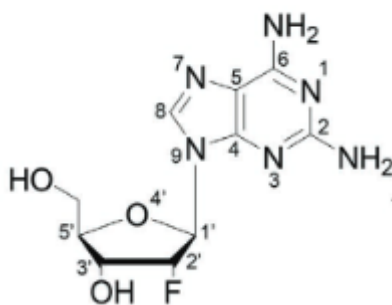


Figure 1.17: 2-FaFA from Wang *et al.*³⁶ a bioactive, biocompatible and injectable low molecular weight hydrogel

Other examples of small molecule injectable hydrogels have been reported by Xie *et al.*³⁷ where a peptide based small molecule (Figure 1.18) could be injected whilst re-forming the gel state and showed excellent inhibitory activity, particularly against *B. Subtilis* bacteria, where only 2.1% of the bacteria remained one hour after treatment with the hydrogel.

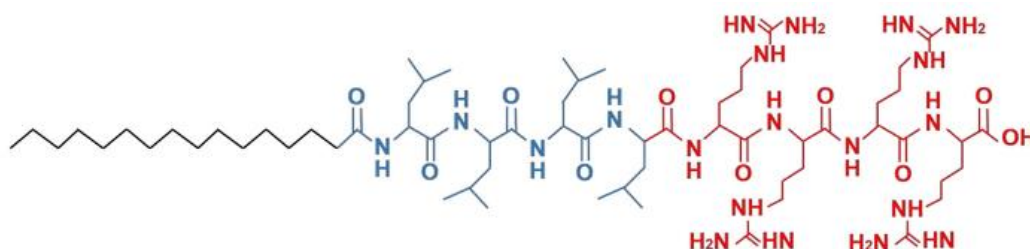


Figure 1.18: Peptide based hydrogel demonstrating excellent inhibitory activity against *B. Subtilis* in Xie *et al.*³⁷

Other work, including those containing polymeric gels, have also been reported highlighting the ability of gels to display preferential mechanical properties for injection.^{38,39} Therefore, prior knowledge of the mechanical profile of a candidate gel is useful in gauging its suitability for any particular application.

1.3.3 Process impact on mechanical properties

However, like in the formation of the gels, the process, solvent and trigger utilised to create the gel are responsible for controlling the mechanical properties of the solids. In work by Nolan *et al.*⁴⁰ they gelate two molecules (1 and 2 in Figure 1.19) for 3D printing using both solvent and pH trigger methods and found that the gels formed from the solvent trigger provided a more continuous print than the same molecular gel formed from a pH trigger.

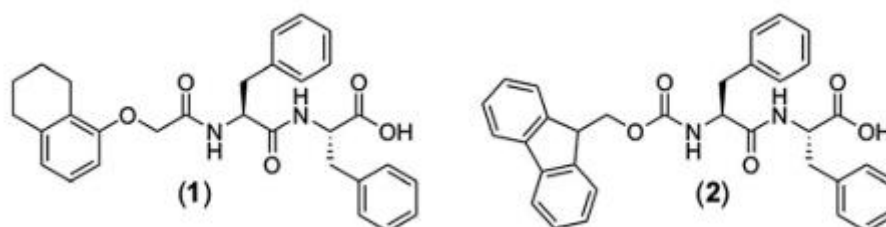


Figure 1.19: Molecules that are capable of forming gels for 3D printing displaying a more even print from a solvent trigger when compared with those formed through a pH trigger in Nolan *et al.*⁴⁰

Moreover, comparison of the rheology of Fmoc based gels (Figure 1.20) formed by both Ca^{2+} , Cu^{2+} and Na^+ metal triggers show that for the same molecule, Ca^{2+} and Cu^{2+} triggers result in gels that are an order of magnitude stiffer than the gels formed by the Na^+ trigger. Moreover, the metal triggered gels were all stiffer than the same molecules formed using a pH trigger.⁴¹

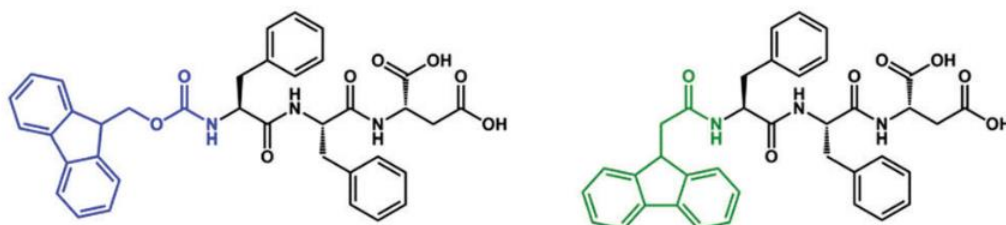


Figure 1.20: Fmoc based molecules whose gels are stiffer from Ca and Cu triggers compared with those formed from Na trigger. Figure adapted from [41]

Other instances of trigger controlled mechanical properties were reported by Adams *et al.*¹⁰ which showed that the process of pH change to trigger gelation had an impact on the resulting stiffness of the gels. Gels of an Fmoc based dipeptide (Figure 1.21) prepared using HCl as the pH change, resulted in gels with a G' of 5 kPa. However, forming gels from the same pre-cursor using the glucono- δ -lactone trigger resulted in gels with a G' of 184 kPa. These results further highlight the complexity governing gel formation and its subsequent properties.

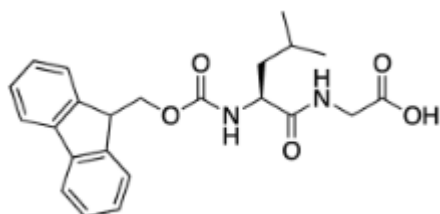


Figure 1.21: Fmoc based gelator from Adams *et al.*¹⁰ displaying pH trigger dependent rheology

1.4 Computational Investigations for Gels

Given the sheer complexity involved in the discovery of new gels, research groups have explored predictive tools to explore this vast chemical space and attempt to streamline the discovery of new gels. The simplest of these approaches are attempts to correlate solubility parameters, δ_d (the energy from dispersion forces between the molecules), δ_p (the dipolar intermolecular force energy between the molecules) and δ_h (the energy from hydrogen bonds between molecules) with gelation.

One such approach involves plotting the Hansen solubility parameters for THF/water mixtures known to cause or disrupt gelation for a particular gelator. Spheres of gelation can be constructed for a solvent (Figure 1.22a) which show, depending on the Hansen solubility parameters for a new gelator (Figure 1.22b), if the molecule is likely to form a gel in that particular solvent mixture.⁴² The idea of correlating molecular gelation to solvent properties has been further explored elsewhere.⁴³

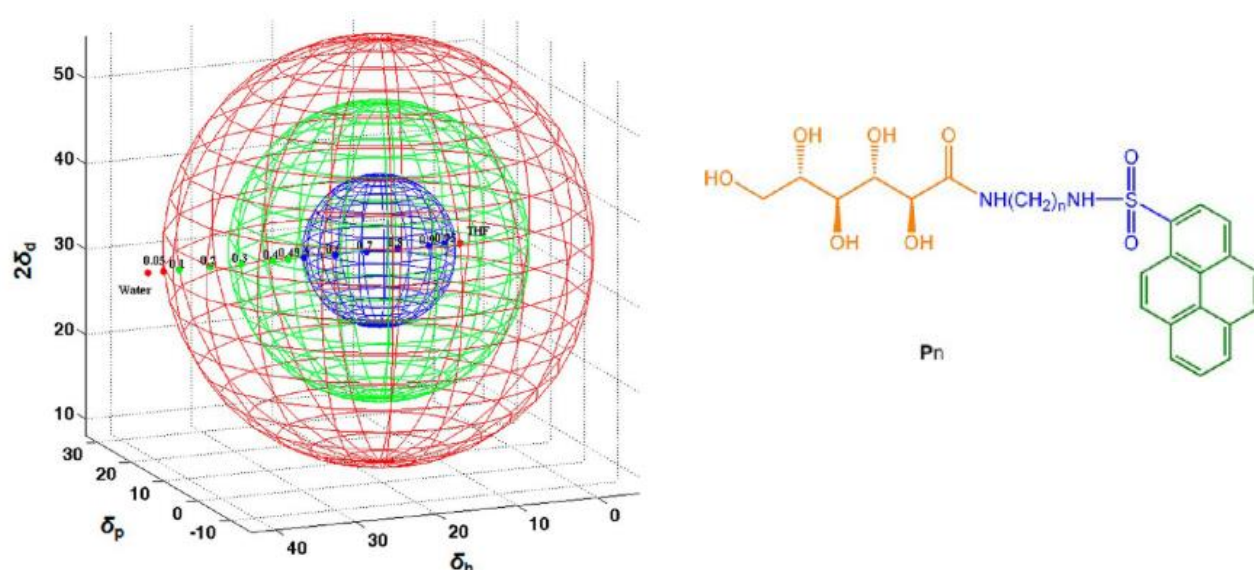


Figure 1.22: a) Hansen solubility sphere for molecule P7 from Yan *et al.* where red = insoluble, blue = soluble and green = gel. b) Molecule P7 ($n = 7$). Figure adapted from [42]

More complex computational approaches have also been explored to try and rationalise and understand the process of gelation. Density functional theory-based approaches have been explored to investigate the relationship between gel fibres and solvents to rationalise gel formation. Meng *et al.*⁴⁴ investigated the interactions between urea based hydrogelator (Figure 1.23) and water and found that significant interactions between water and urea at short distances were evenly distributed across the molecular backbone – hinting that the gelation ability of this hydrogelator is linked to its strong uniform interaction with the water phase.

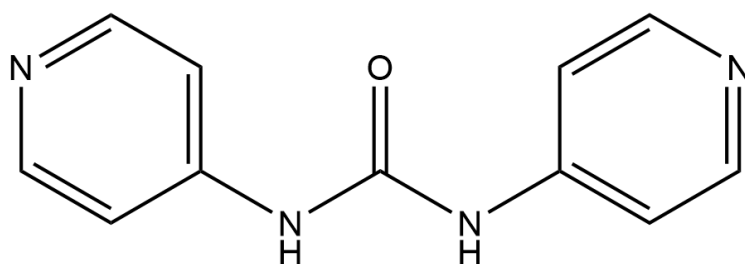


Figure 1.23: Urea based gelator from Meng *et al.*⁴⁴

Mu *et al.*⁴⁵ further investigated the self-assembly of gelators, this time utilising molecular dynamics (MD) simulations. Here, they utilise MD simulations of a Fmoc-AA (Figure 1.24) dipeptide in water and find that the Fmoc moiety forms hydrogen bonds with the water – disagreeing with an earlier hypothesis that Fmoc was buried within the fibre and suggesting it plays an important role in the self-assembly in this system.

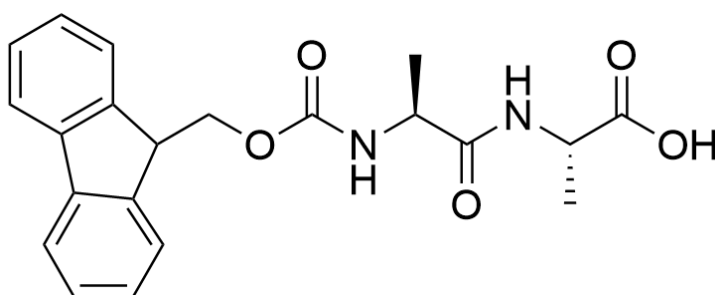


Figure 1.24: Chemical structure of the gelator FmocAA used by Mu *et al.*⁴⁵

MD simulations have also been utilised in tripeptides by Moreira *et al.*⁴⁶ Here, they use coarse-grained simulations to computationally screen potential dipeptides and investigate the likelihood that they form a multi-component gelation system with a tripeptide, H-aspartyl-phenylalanyl-phenylalanine-OH (DFF) (Figure 1.25), and find dipeptides that produce better multicomponent gels than DFF alone.

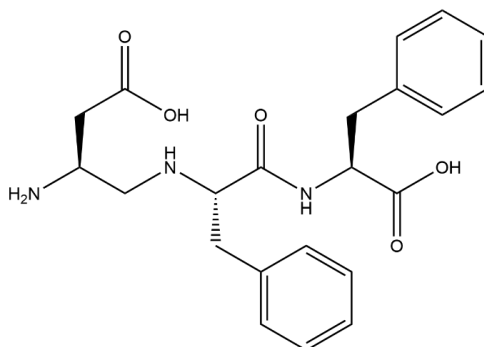


Figure 1.25: Structure of DFF that forms part of the multicomponent system with improved gelation in Moreira *et al.*⁴⁶

Although work investigating the relationship of the gelator-solvent system are invaluable, they have the drawback that they are not predictive and thus cannot be used to quickly screen new molecules and generate new potential gelators. To this end, groups have turned to the improvements in machine learning (ML) that can act as a rapid screening tool for the gelation ability of molecules under certain conditions.

The first such report of a ML model that was a predictive tool for the gelation ability of molecules was from Gupta *et al.*⁴⁷ Here, they presented a range of ML models that are able to predict the gelation ability of a molecule using a glucono- δ -lactone trigger – purely from a text representation of the molecule. This approach allows for the rapid screening of molecules, through the presented webpage, of molecules to accelerate potential gel discovery. Other groups have presented similar ML approaches⁴⁸ and others have even combined MD simulations as the descriptors for the ML model to learn from.⁴⁹

The examples outlined above outline the power of computational approaches and their increased prevalence in materials chemistry. Machine learning in particular is becoming much more widely used within chemistry.⁵⁰

1.5 Machine Learning

1.5.1 Background to Machine Learning

Improvements in computation ability over the last 40 years have revolutionised the way we live our lives and possibly the most impactful of these has been the advances in machine learning. Machine learning involves utilizing computers to detect patterns in data which can then be used to form predictions on new, unseen data.⁵¹ To do this, the computer is shown data with known values of interest and it attempts to find relationships between the data and the values of interest, using algorithms that differ in terms of both methodology and complexity, in what is known as the “training” phase of the predictive model.⁵² We can then use the relationships deciphered during the training phase to make predictions on data where the true value is unknown.

“Artificially intelligent” systems built using this training and testing approach are already ingrained in everyday life. From self-driving cars to search engine autocomplete, machine learning systems can advance all walks of life, with their application seen in more and more diverse disciplines. The field of chemistry is not unique in this, having seen rapid growth in the use of machine learning with the practice penetrating more and more fields within chemistry.

1.5.2 Artificial Intelligence in Chemistry – Quantitative Structure-Activity Relationships

Models built within a chemistry context aim to relate chemical structures to properties of interest. Building models using this approach exploits the similarity principle, which is that similar molecules should exhibit similar properties.³⁰ This allows for the development of models dependent on molecular structure. One of the earliest examples of such a model is the Hansch equation⁵³ (equation 1.3), which was applied to a series of phenoxyacetic acids acting as plant hormones. The equation utilises the octanol-water partition (π) and the Hammett constant (σ) of a series of analogues to predict the biological activity, c , in terms of the concentration of the phenoxyacetic acids which induces 10% growth in the *Avena coleoptiles* plant assay.

They found that Equation 1.5 correlates very closely with observed activity and is credited as being the first example of mathematically linking molecular structures with a property of interest. This work has also been credited as being the foundation of an entire scope of molecular modelling; Quantitative Structure-Activity Relationships (QSAR).

$$\log\left(\frac{1}{c}\right) = 4.08\pi - 2.14\pi^2 + 2.78\sigma + 3.36 \quad (1.5)$$

The process of building predictive models using computers is widely deployed in the field of drug discovery. Predictive models, within the context of drug discovery, have the potential to speed up hit-to-lead optimisation⁵⁴ by deploying models that can accurately predict ADMET properties.⁵⁵ The ability to predict properties such as toxicities for a range of compounds simultaneously prevents the unnecessary time spent synthesising molecules with undesirable toxicity profiles, not to mention the cost-associated benefits of such a tool.⁵⁶

Therefore, much of the advancement in QSAR and molecular modelling have been driven by this desire to speed up lead optimisation. One of the earliest examples of a QSAR model being utilised in drug development is the case of the antibiotic, norfloxacin, active against gram-negative bacteria that cause urinary tract infections. Here, they use a Hansch equation to find analogues of commercially available nalidixic acid with improved activity.⁵⁷ After QSAR analysis, the analogue that would be marketed as Noroxin (Figure 1.26) was synthesised and shown to have a 500 fold improvement in minimum inhibitory concentration over nalidixic acid.⁵⁷

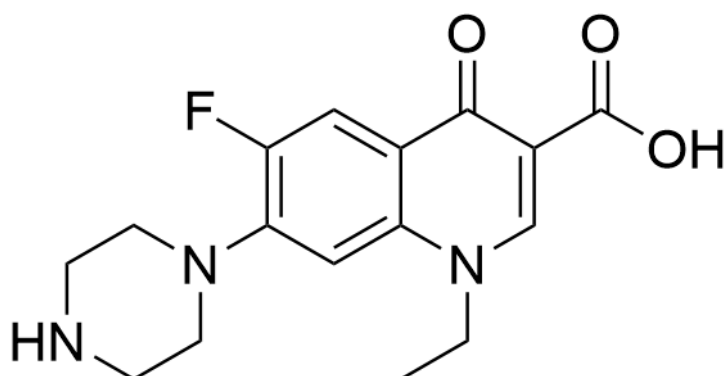


Figure 1.26: Structure of Noroxin that shows 500-fold improvement in MIC over Nalidixic acid.

Another example of the early implementation of QSAR that resulted in commercial deployment of a molecule is that of the herbicide Metamitron (Figure 1.27) where QSAR analysis linked the octanol-water partition coefficient to photosynthesis inhibition activity for a range of analogues. Each set of analogues investigated the effects of a particular substituent on biological activity and determined how each R-group affected photosynthesis inhibition. They produce a Hansch equation for the predicted IC_{50} based upon the octanol-water partition coefficient with good correlation coefficient ($R=0.95$). Then, this equation is coupled with a lipophilicity term to give an overall equation for the $\log(1/ED_{50})$ to accurately predict the photosynthesis inhibition ability of analogues against cotton and wheat.⁵⁸

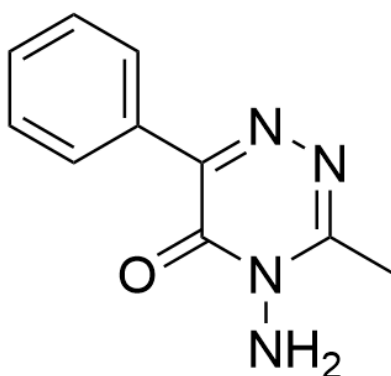


Figure 1.27: Chemical structure of the herbicide Metamitron from an early implementation of QSAR in [58]

These are just some examples of early successes of mathematically linking properties of molecules to a property of interest to synthesise more effective molecules, quicker and cheaper. Although these examples are simple, they form the basis for the more complex models that are built today. An increase in the quality and amount of data available has allowed for the generation of more complex models using more complicated and computationally intensive algorithms.

Although QSAR modelling has its roots in drug discovery, it has become commonplace in the field of materials chemistry. Machine learning has been used by groups investigating green energy to: predict the band gap of perovskites for their use in applications such as photovoltaics⁵⁹, utilise machine learning for the high-throughput screening of new catalysts⁶⁰ and repurpose failed experiments to predict future reaction success for new material discovery.⁶¹ In the work to predict the band gap of perovskites, they train a kernel ridge regression (KRR) model using a dataset of double perovskites from the Computational Materials Repository. Using 90% of their dataset the KRR is trained and predictions generated using the remaining 10%. They achieve R^2 between calculated and predicted band gaps of 0.96 on the training set and 0.90 on the test set, showing good predictive performance.⁵⁹

1.5.3 Background to QSAR

1.5.3.1 QSAR Process

QSAR models are generally built using the same four major steps and the process is summarised in Figure 1.28. First, the data used to build the models has to be generated. Historically this was limited to having to synthesise a range of molecules, measuring the experimental endpoint and build the models on this relatively limited dataset. However, today, there are numerous datasets containing upwards of millions of molecules that can be used as molecules for QSAR modelling. Examples of such datasets are the ZINC database, a database of over 230 million commercially available compounds⁶² or ChEMBL, a database of over 2 million bioactive molecules with drug-like properties.⁶³ However, these do not always contain molecules within the chemical space of interest or have the experimental endpoint that you are trying to model.

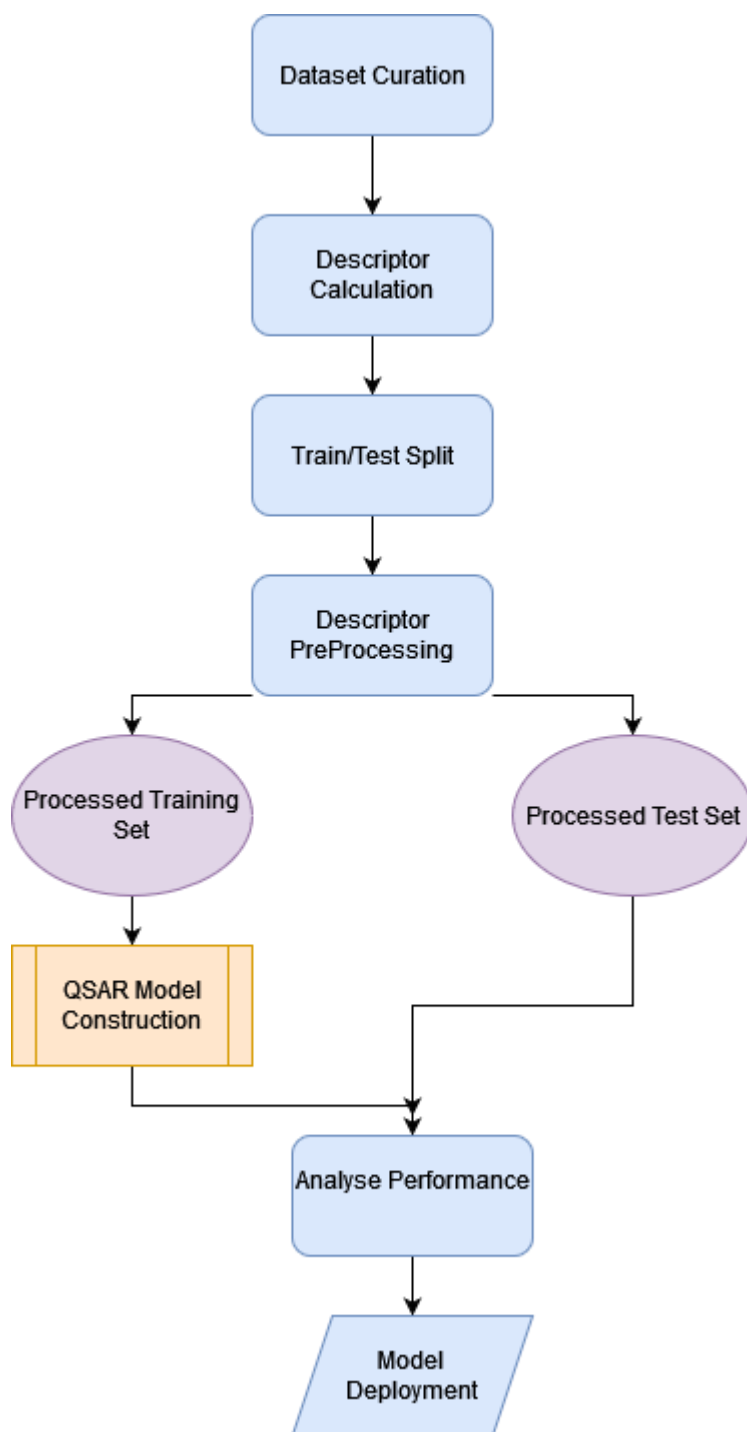


Figure 1.28: Generic overview of the QSAR process

Once a dataset is obtained, descriptor representations of the molecules (*see 1.5.3.2 Molecular Descriptors*) need to be calculated for input into the modelling algorithm. To do so, the molecule must be represented in a computationally readable format. Usually, the basis for calculating descriptors is the Simplified Molecular-Input Line-Entry System, or SMILES⁶⁴, representation of the molecule. The SMILES representation is a text-based representation of the molecular structure that allows the specification of atoms using characters, such as C for carbon, O for oxygen *etc.* and bonds as special characters such as '-' for a single bond, '=' for a double bond and ':' for an aromatic bond. SMILES can be interconverted between the text representation and 2D/3D representations using various software packages and make for an attractive representation for molecules, owing to their simplicity.

1.5.3.2 Molecular Descriptors

Once in a computationally readable format, molecular descriptors can be calculated. The ability to build models for use in the range of applications described above has been spearheaded by the advancement in chemical descriptors used to encode the molecules. The performance of any model hinges on the ability of the descriptors to accurately represent the molecules and their properties - if the descriptors do a poor job of describing the molecule, then performance is likely to suffer. Therefore, it is imperative that appropriate molecular descriptors are used for a given context in model building and, thanks to advances in the field, there are numerous types of descriptors to choose from – differing in complexity to calculate.

Molecular descriptors are diverse, spanning the range of 0 dimensional (0-D) descriptors, to 4-D descriptors. 0-D descriptors are molecular descriptors that do not take into account molecular structure or atom connectivity and are counts of a particular property, such as the number of rings, the number of hydrogen bonds or the molecular weight.⁶⁵

Increasing in complexity, the 1-D descriptors consider substructures within the molecule and are generally a binary representation of the absence or presence of particular substructures in the molecule. These are generally fingerprint descriptors and are discussed in further detail below. LogP, the measure of hydrophobicity of a molecule, and the number of hydrogen-bond donor and acceptor atoms are also considered 1D descriptors.

Increasing in the complexity from 1D descriptors, 2D descriptors also consider the connectivity within the molecule as a molecular graph.⁶⁶ 2D topological descriptors include adjacency matrices⁶⁷ that encode information about the molecular graph and topological indices which apply a mathematical operation to the molecular graph to represent them as a numeric value, or set of values.⁶⁸

3D and 4D descriptors further increase the complexity by considering the molecule as a geometric shape in space, expressing the location of atoms as their Cartesian coordinates in x-y-z space. An example of a 3D descriptor is the 3D-MoRSE descriptor⁶⁹ which represents 3D molecular representations based on electron diffraction transformations.⁷⁰ 4D descriptors take it even further by considering interactions between the molecule in 3D space with either a receptor or another molecule.⁶⁵ Examples of 4D descriptors include Comparative Molecular Field Analysis (CoMFA) and Comparative Molecular Similarity Indices Analysis (CoMSIA).⁷¹

However, it is 0-D and 1-D descriptors that are the most used in QSAR model building due, in part, to their ease of calculation. The most common type of 1D descriptor used in QSAR is the fingerprint descriptor. These descriptors give an insight into the presence or absence of certain molecular fragments. One common fingerprint descriptor is the circular Extended Connectivity Fingerprint (ECFP)⁷² which calculates the presence or absence of molecular fragments within a given radius.

To calculate the fragments in the molecule, initially, each atom in the molecule is assigned an integer which is placed into an array (Figure 1.29). Then, each atom collects this identifier, and creates a new identifier taking into account the identifier of its immediate neighbours and the bond order between the atoms. These numbers are then converted back to a single number using a hash function – this is the new identifier which is appended to the array.

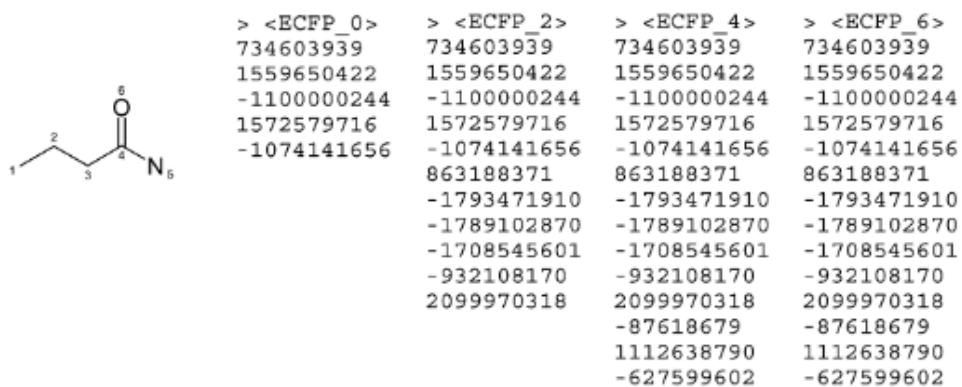


Figure 1.29: Example of iterative fingerprint generation for the ECFP algorithm. Figure adapted from [72]

Each iteration of this process “grows” the fragment from the starting atom, giving more information as to the local environment of each atom. This algorithm is repeated for a set radius around each atom, usually a radius of 4 (Figure 1.30). This hash key array can then be encoded into a bit string of desired length, most commonly a bit string of 2048-bits.

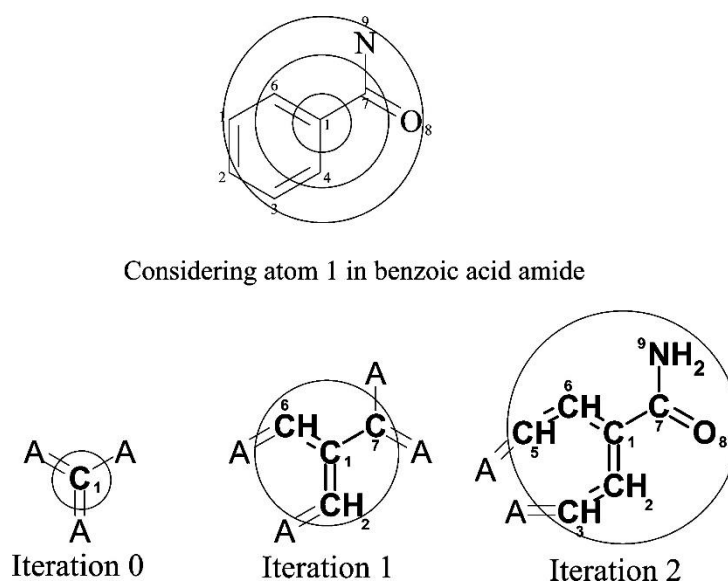


Figure 1.30: Example of the iterative updating of the local molecular environment of a given radius around an atom. Figure adapted from [72]

A similar type of fingerprint to the ECFP is the Functional Class fingerprint (FCFP) - the FCFP is a more generic form of fingerprint as it considers the presence and absence of different pharmacophore functional groups, such as halogens, hydrogen bond acceptors/donors *etc.*⁷²

1.5.3.3 Dimensionality Reduction

Once descriptors are calculated, a common next step in QSAR model building is the visualisation of the data. To visualise the dataset, the dimensionality of the set has to be reduced to typically 2 or 3 dimensions and since the number of descriptors is typically much larger than 3 dimensions, there is inevitably some loss of information during the reduction.

However, many methods for dimensionality reduction attempt to capture as much of the information in the dataset in the reduced dimension as possible, like Principal Component Analysis (PCA).⁷³ PCA is a popular dimensionality reduction algorithm widely used as a visualisation tool⁷⁴⁻⁷⁶ and as part of pre-processing for machine learning⁷⁷⁻⁷⁹ (*see 1.5.3.5 Data Preprocessing for other examples*).

The aim of PCA is to construct principal components that capture as much of the variance within the dataset as possible in each component. To do this, PCA first centres and scales the dataset on each feature mean and constructs the covariance matrix for the dataset. From the matrix, the first principal component is constructed as a linear combination of the features that have the largest possible variation, subject to a constraint that prevents all features being chosen. Then subsequent principal components are calculated, again as linear combinations, but such that they are uncorrelated to previous principal components.⁸⁰

It is common for visualisation approaches, to calculate the first 2-3 principal components for a dataset which can then easily be visualised as a 2D or 3D graph whilst retaining as much information as possible from the original dimensions.

Another dimensionality reduction algorithm is the Tree Manifold Approximation (TMAP) algorithm used to visualise large chemical datasets.⁸¹ TMAP has been used to visualise the fragments in the GDB-17 fragment library⁸², large chemical datasets used to predict reaction yield⁸³ (Figure 1.31) and in fragment libraries in the potential discovery of new drugs for COVID-19.⁸⁴

gram-scale

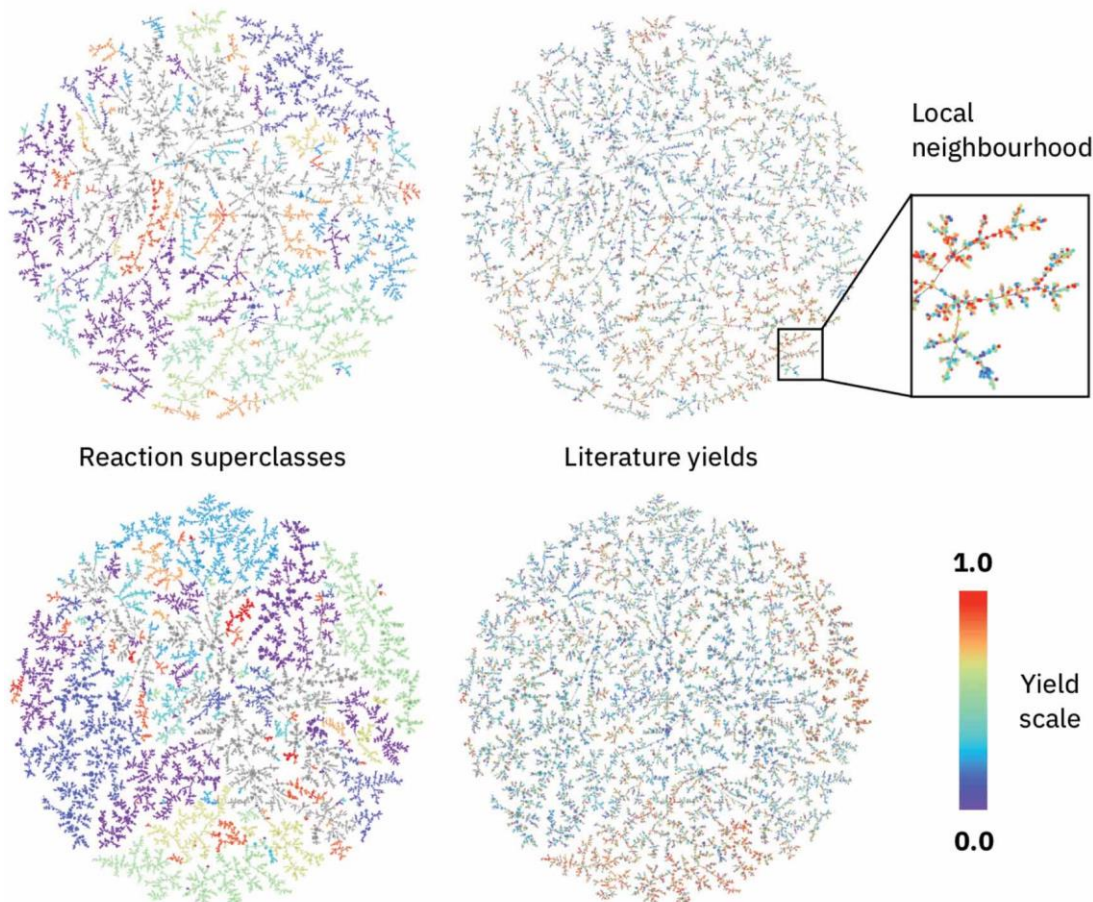


Figure 1.31: TMAP plots showing clusters that represent different types of reaction. On the left each colour represents a type of reaction and the colour on the right is the literature yields of the reaction. Top refers to reactions on a gram scale and bottom is reactions on the subgram-scale. Figure adapted from [83]

It reduces the dimensions of a dataset through four stages, Locality Sensitivity Hashing (LSH) forest indexing, and construction of a nearest neighbour graph, calculation of a minimum spanning tree and generation of the layout for the minimum spanning tree.

LSH forest indexing uses a range of locally sensitive hashing procedures to generate variable length labels for each data point such that more similar labels equate to more similar points. For more information, the reader is directed to the work of *Bawa et al.*⁸⁵ Once indexed, a k-nearest-neighbour graph (k-NNG) is generated for the indexes and the minimum spanning tree for the k-NNG is calculated and represented in Euclidean space. In the MST, branches within clusters represent relationships within clusters of the data.

1.5.3.4 Data Splitting

With the descriptors calculated, the whole dataset needs to be split into the training and testing subsets described earlier. Typically, the dataset is split into at least two sets, though this is not necessarily always the case due to a lack of available data. The dataset can be split into a training set to try and find the relationship between descriptors (or a subset of them) and the endpoint, a test set to either tune the mathematical model describing the relationship or as a test of the models predictive ability and a validation set to expose the model to unseen data and further gauge its true predictive ability.

There are numerous methods utilised to carry out this data set split, each with their own advantages and disadvantages. The simplest such approach is the random split, where the dataset is randomly partitioned into sets of a predetermined size, usually 80% train, 20% test. One issue with a random split is that there is no guarantee that the split results in a training set that covers the entire range of the dataset.⁸⁶ One way of overcoming this is what is known as a stratified split which splits the data in such a way that all splits are representations of the entire range of the endpoint to be predicted.⁸⁷

Groups have presented more complicated methods of data splitting such as the Kennard-Stone algorithm⁸⁸ which selects two points that are furthest apart in Euclidean space and then selects points subsequently that maximise the distance from already selected points. The KS algorithm can select a fraction of the dataset that smoothly fills the data space for model training.⁸⁹ Another is the Sphere exclusion algorithm⁹⁰ (summarised in Figure 1.32) which select, initially, the point with the highest activity value and places it into the training set. Then a sphere is constructed around this point and points within the sphere are placed into the test set. A new molecule is selected that is most or least similar to the point used to derive the sphere and the process is repeated.⁹¹

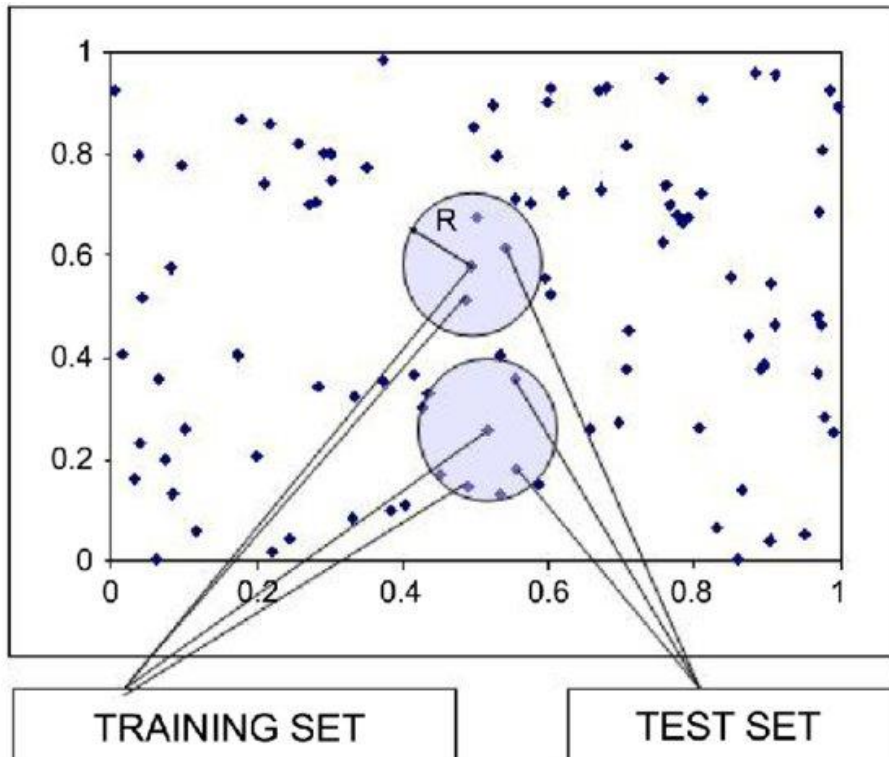


Figure 1.32: Example of the sphere exclusion taken from [91]. In both cases, the central point in the sphere is the basis for train and test splitting. Each point within the sphere is considered and placed alternatively between the training and test sets.

However, a powerful data splitting procedure is the time-split procedure.⁹² In a time split procedure, the data is split based on when it was generated, the earlier data forms the basis of the training set, and the more recent data forms the testing set. This is a useful splitting approach as it is more indicative of the real-world applications of any model derived from your data.

1.5.3.5 Data Preprocessing

Once split, the data is almost ready to be applied to a machine learning algorithm. However, there is a final processing step, which may or may not be required – depending on the type of algorithm used. It is good practice, to carry out checks on the training data to remove descriptors that are highly correlated with one another – using two highly correlated variables in the same model can, in certain circumstances, lead to no new extra information being added to the model and makes it difficult to separate the effects of each variable on the model whilst simultaneously increasing model complexity.⁹³

Moreover, the training set also needs to be screened for any low variance descriptors, typically where a single value makes up 95% of the entire descriptors values. For example if there are 100 data points in the dataset, and for a particular descriptor 95 of these data points have the value 1 – then we would consider that descriptor uninformative and thus it would be removed.⁹⁴ Finally, some algorithms are sensitive to the scale of the descriptors used to learn them and as such, it is useful to centre and scale each column so that each column has a mean of 0.⁹⁵

The choice of modelling algorithm will depend on the nature of the problem: classification or regression. Classification models are utilised when the endpoint being modelled is a binary value that can only take on discrete values. The most common class of classification is a two-class classification which typically predicts True/False or Class/Not Class for the data. If the endpoint is a continuous value, then a regression model is required.

Whether the problem is classification or regression there are a range of potential algorithms to model the problem. Some of the algorithms are specific to classification or regression and others are applicable to both.

1.6 Modelling Algorithms

Modelling algorithms, whether they are classification or regression algorithms can broadly be classified as linear, non-linear and decision tree based models. Linear models are the simplest of the machine learning models where the algorithm attempts to model the relationship between the descriptors and the endpoint as a linear combination of the descriptors.⁹⁶ In a regression setting, the linear combination of the descriptors are used directly to predict the endpoint value⁹⁶ whereas in classification, the endpoint is generally predicted between 0 and 1 with a threshold, such as 0.5, used as the boundary to classify points.⁹⁷

Non-linear models are models where the descriptors and endpoint are not linearly related and utilise approaches such as projections to higher dimensions⁹⁸ (see 1.6.8 SVM) or weights in neural networks⁹⁹ to map the descriptors to the endpoint. Non-linear models also encompass decision trees which partition data based on thresholds set on the descriptor values.¹⁰⁰ After partitioning, the descriptors are mapped to the endpoint by averaging the values of other points within that partition.

Below, a range of models encompassing linear, non-linear and tree-based models are introduced as they are used throughout this thesis to build our QSAR models.

1.6.1 Bayesian Generalised Linear Models

Generalised linear models (GLM) are a form of linear regression (Equation 1.6) that applies a function to the output from the regression. In a simple linear regression, the regression equation takes the form of

$$y = X\beta + \varepsilon \quad (1.6)$$

where ε is the error value, X is a matrix consisting of N descriptor features and β is a matrix consisting of the regression coefficients, a measure of how much influence each feature has on the predicted y value. In a generalised linear model, a function, called the link function can be applied to y such as taking the log values. However, the function applied can be an identity function where y is returned unaltered.

The Bayes aspect of the GLM utilises Bayes theorem (introduced in detail in Chapter 4) to estimate the coefficient values by setting a prior probability on the values they are likely to take.

Bayesian regression models have been successfully used in chemistry to model NO_2 exposure in Europe¹⁰¹ and in the prediction of ground level ozone concentrations using a generalised linear model coupled with a Bayesian maximum likelihood approximation.¹⁰² During training the model achieves R^2 of 0.67 but does not report any external set prediction.

1.6.2 k-Nearest Neighbours

kNN is an example of a modelling algorithm influenced by the scale of the descriptor values. In kNN the k-nearest neighbours are found for each data point in the descriptor space and are used as the k points to predict values for the point in question (Figure 1.33). In classification, the most common class within the k-nearest points are used as the predicted class and in regression, the average value of the nearest points is used as the points prediction.

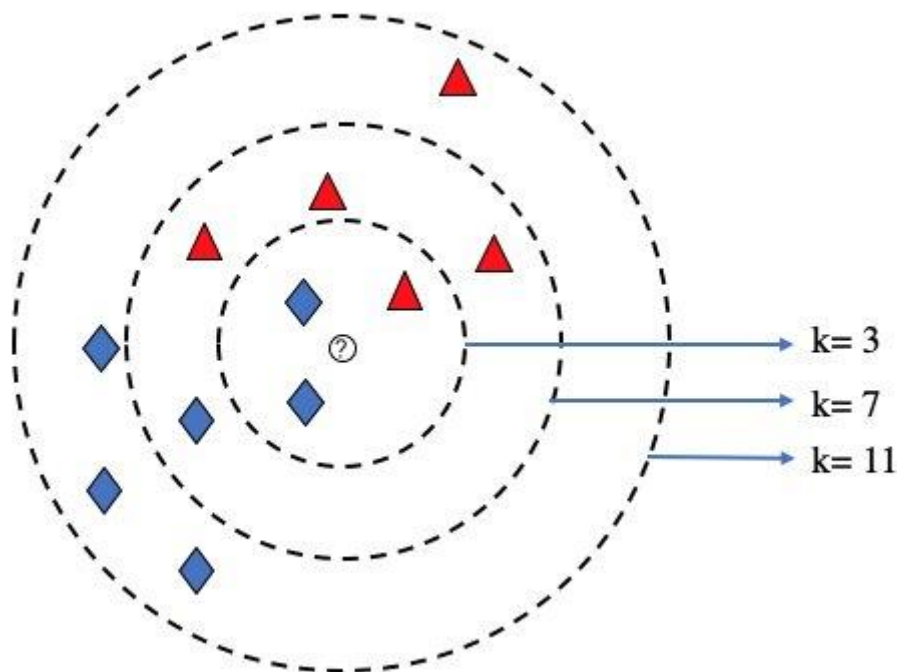


Figure 1.33: Example of a kNN model, the unknown point will take its prediction from the average class (red/blue) of the nearest k neighbours. Taken from [103]

Although a relatively simple model, the kNN model has been successfully utilised to tackle both classification and regression problems.¹⁰³ For classification, the kNN algorithm has been used to predict liver-related adverse drug effects¹⁰⁴, classification of lung cancer subtypes¹⁰⁵ and in the classification of defects in the manufacturing of semiconductors.¹⁰⁶ In a regression setting, kNN algorithms have successfully been used to monitor water quality at a wastewater treatment plant¹⁰⁷ and in the prediction of the compressive strength of concrete mixed with fly ash and damaged ceramics to repurpose these waste materials for more sustainable concrete.¹⁰⁸

1.6.3 Naïve Bayes

Naïve Bayes is a probabilistic method based upon Bayes Theorem (discussed in detail in Chapter 4) which in a classification usage, uses the prevalence of each variable for a particular class $P(\text{Variable} | \text{Class})$, the prevalence of each class $P(\text{Class})$ and the prevalence of each variable $P(\text{Variable})$ to predict the probability of class membership given a particular set of variables. The Naïve assumption in Naïve Bayes is the assumption that no features are related and all features are completely independent from one another.¹⁰⁹

Naïve Bayes has been deployed as a classifier in the prediction of Ames mutagenicity¹¹⁰, respiratory toxicity¹¹¹ and classification of molecules in terms of their cell cytotoxicity.¹¹² In chemistry, NB classifiers have been used by Solov'ev *et al.* to classify organic molecules as weak or strong binders to a range of rare-earth metals, such as Yb^{3+} and Er^{3+} . Their models show good training performance with balanced accuracy scores above 0.7 in training for each metal ion. On an external test set, the performance is equally as good, with the lowest performing model possessing a balanced accuracy of 0.714.

1.6.4 Partial Least Squares – Discriminant Analysis

Partial Least Squares – Discriminant Analysis (PLS-DA) is the application of a partial least squares regression problem to a classification problem. To apply PLS-DA to a classification problem, the endpoint is converted from a string or factor into a numeric value (the dummy variable), usually 0 and 1. The descriptors are dimensionally reduced to construct the PLS components such that the variance of the data described in the components is maximised and they correlate strongly with the endpoint variable. The model will then create a regression model between the dummy variable and the PLS components.¹¹³

PLS-DA has been used in a range of settings, including in the identification as to the source of oil spills¹¹⁴, classify Raman spectra of serum samples as cancerous or non-cancerous with 99% sensitivity and specificity¹¹⁵ and as a classification tool in reduced dimensions to assess the quality of catalysts in a manufacturing plant during quality control.¹¹⁶

1.6.5 Random Forest

The Random Forest is an example of an ensemble decision tree-based model. A random forest is made up of several individual decision trees, each of which is built upon a bagging (bootstrap aggregation) approach where each tree is generated upon a bootstrap of the dataset utilising a subset of predictors at each node to select the best possible split. A bootstrap is a random sample taken from the dataset without replacement.¹¹⁷ The trees are split until there are no more possible splits, the child nodes contain homogenous labels, or a predetermined depth has been reached. Each tree is validated internally utilising the rest of the dataset that was not included in the bootstrap used to grow the decision tree.

For a classification problem, in a single tree a data point is assigned a class based upon the predominant class of the node it is present in. The overall prediction for all trees is the most common prediction assigned by all trees. For a regression problem, each node is assigned the mean value of the data points that terminate at that node and the average of all trees is used to assign the prediction to a new data point. Figure 1.34¹¹⁸ shows a generic overview of the process of generating the trees in a random forest model.

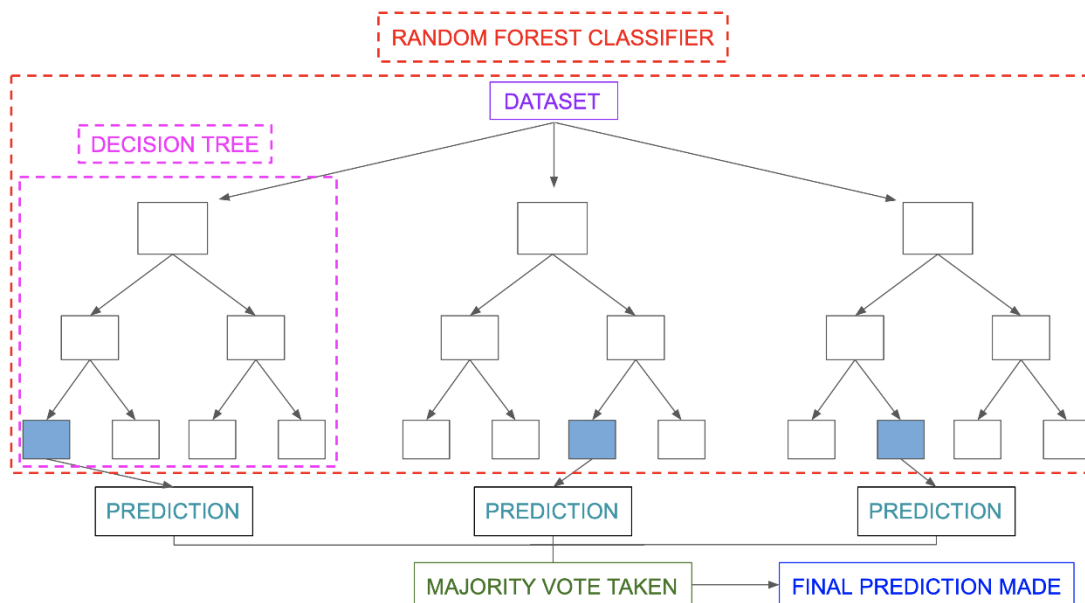


Figure 1.34: A generic example of a Random forest taken from [118]

Random forest (RF) is a widely used modelling algorithm and has a range of examples of its successful application in materials chemistry. Carrete *et al.*¹¹⁹ train a Random Forest model on a dataset of 32 semiconductors to predict the lattice thermal conductivity with a Spearman's rank correlation coefficient of 0.74 on the training set. Their model is tested on 4 external points which show good agreement between experiment and prediction.

Nagasawa *et al.*¹²⁰ successfully used a RF model to model the power conversion efficiency of conjugated polymers. The experimental and predicted agreement is given by a correlation coefficient of 0.62, indicating moderately good relationship. Kwak *et al.*¹²¹ use a Random Forest regressor to predict the tensile strength, elongation and other mechanical properties of γ -TiAl alloys with high R^2 (> 0.9) between prediction and experiment.

1.6.6 C5.0

C5.0 is another example of a tree-based method and is an improvement of C4.5. In C4.5, decision trees are built on the training set variables, splitting the variables such that the information gain or difference in entropy is maximised. The dataset is split using the maximum information gain until no further splits are possible. If the later splits do not contribute significantly to the accuracy of the model, the trees are pruned to remove these later splits.¹²²

C5.0 builds upon C4.5 but generates smaller and simpler trees.⁹⁵ Although there is very little literature about C5.0, most understanding has come from investigating the source code of C5.0. One improvement of C5.0 over C4.5 is that C5.0 "combines non-occurring conditions for splits with several categories" and an improved global pruning procedure and both of these result in smaller and simpler trees.⁹⁵

1.6.7 Neural Networks

Neural networks (Figure 1.35) are a class of methods that have derived their name from the functions of the brain. The simplest neural network is a single layer perceptron (SLP). In a SLP, the input data is fed forward into a single hidden layer which contains a number of nodes with an assigned weight, the value of the weight is initially set to 1 but is optimised upon the presence of data using a range of algorithms.^{123–125} The descriptor matrix is multiplied by the weight matrix and a bias term is added before being passed to the output node. The output node applies a function, the activation function, to the values and the type of function utilised depends on the nature of the problem. In a classification

problem, a logistic function (1.7) can be applied to scale the output between 0 and 1 and then a decision boundary is decided to classify the points.

$$\text{logistic function} = \frac{e^x}{e^x + 1} \quad (1.7)$$

In a regression setting, the type of output function can be a linear function (1.8), a ReLu function (1.9) or a range of other functions.

$$\text{linear function } f(x) = x \quad (1.8)$$

$$\text{ReLU function } f(x) = \max(0, X) \quad (1.9)$$

Neural networks (Figure 1.35)¹²⁶ are a popular machine learning algorithm, used extensively within the field of materials chemistry. Behler *et al.*¹²⁷ utilised a NN to calculate potential energy surfaces (PES) of bulk silicon from calculated energies of silicon in varying configurations. The NN calculated PES maps closely with those calculated by DFT and other methods.

Other work using neural networks include predicting the surface tension of organic molecules as a function of temperature – showing good agreement between experimental values and NN predictions.¹²⁸ Montavon *et al.*¹²⁹ build a range of NN models based on *ab initio* calculations of organic molecules to predict a range of electronic properties including HOMO and LUMO.

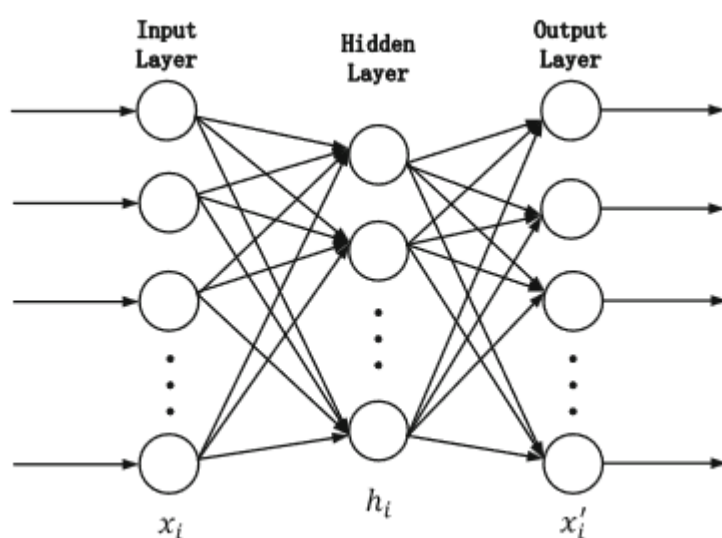


Figure 1.35: An example of a neural network. The data (x) is inputted into a hidden layer and weights applied. The data is fed forward from the hidden layer to the output layer where the models prediction is returned. Taken from [126]

1.6.8 Support Vector Machine

A support vector machine (SVM) shown in Figure 1.36 is an example of an algorithm that is sensitive to the scale of the data. The input data is projected into a multidimensional space by a radial basis kernel and the algorithm attempts to draw a hyperplane in the multidimensional space that, in a classification problem, distinguishes between each class.¹³⁰ There will inevitably exist a trade-off between the complexity of the hyperplane and the number of incorrectly classified points.

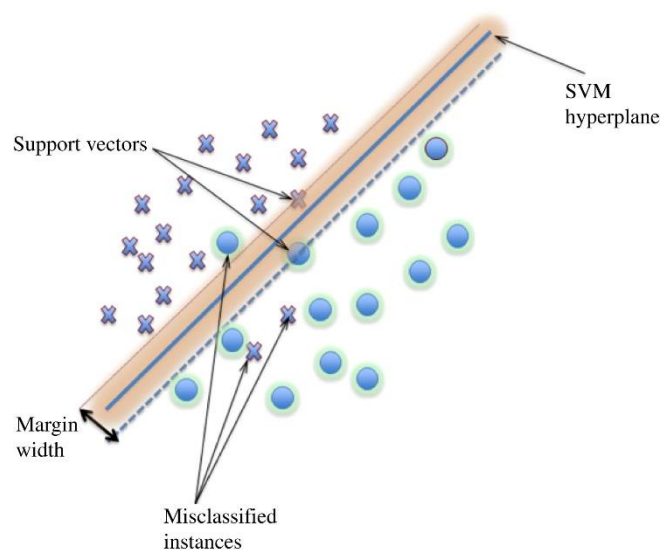


Figure 1.36: An example of an SVM hyperplane classifying data. The points within the margin width are the support vectors responsible for constructing the hyperplane Taken from [98]

The hyperparameters that control this complexity are C , the cost, and γ . C applies a cost to each misclassified point for a particular hyperplane and can be set to a value that allows (or restricts) the line complexity by applying a low (or high) cost to misclassification. γ controls the radius at which the model selects points as support vectors for constructing the hyperplane, low values of γ mean that points far away can influence the hyperplane.¹³¹

In a regression setting, the data is still projected into a higher dimensional space where a regression function between the descriptors X and dependent variable Y is derived such that it minimises the prediction errors. The parameter ϵ , controls the distance from the regression line that the points contribute linearly to the prediction error. Outside of this zone, the errors are penalised by the cost.¹³²

SVM has been used in both a classification and regression setting in materials chemistry. In crystal structure prediction, SVM models have been built to classify the type of structure obtained by AB inorganic solids. On an external validation set, the model performed well with sensitivity, specificity and accuracy of 94.2%, 92.7% and 99.6% respectively.¹³³

As for regression, SVM has been utilised to predict the mechanical properties of molybdenum diselenide (MoSe₂). Wang *et al*¹³⁴ train an SVM regressor on the output of simulations for 700 instances of MoSe₂ and build models to predict the strain, tensile strength and modulus of the material with mean squared errors of 1.9×10^{-5} , 2.3×10^{-2} and 2 which they assess as good as it is 1 to 4 orders of magnitude below the target values.

1.8 Performance metrics

1.8.1 Classification Performance Metrics

The ability of a model to correctly classify points can be assessed in numerous ways, most of which can be derived from a confusion matrix (Table 1.1) which collates all the predictions based on if they were predicted correctly and if not, which type of misclassification was seen (positive predicted as negative *etc.*).

Table 1.1: An example confusion matrix

		Observation	
		Positive	Negative
Prediction	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Common measures derived from the table include the True Positive Rate (TPR) or the sensitivity (equation 1.10) and the True Negative Rate (TNR), the specificity (equation 1.11).

$$TPR = \frac{TP}{TP+FN} \quad (1.10)$$

$$TNR = \frac{TN}{TN+FP} \quad (1.11)$$

These two measures combine to form the balanced accuracy (1.12), an important measure of the accuracy of the model prediction – accounting for any class imbalances that may be present in the data.

$$\text{Balanced Accuracy} = \frac{\text{TPR} + \text{TNR}}{2} \quad (1.12)$$

1.8.1.1 ROC and H measure

Another common performance metric for a classification model is the receiver operating characteristic plot (Figure 1.37) which is a plot of the sensitivity against 1-specificity. The curve¹³⁵ plots these values for different thresholds of the classifier, taking into account all potential cut-off values used to determine the decision boundary. When interpreting the curve, the area under the curve (AUC) is calculated and used as a measure of classifier performance. An AUC of 1 indicates perfect classification and an AUC of 0.5 indicates a classifier that is no better than chance.

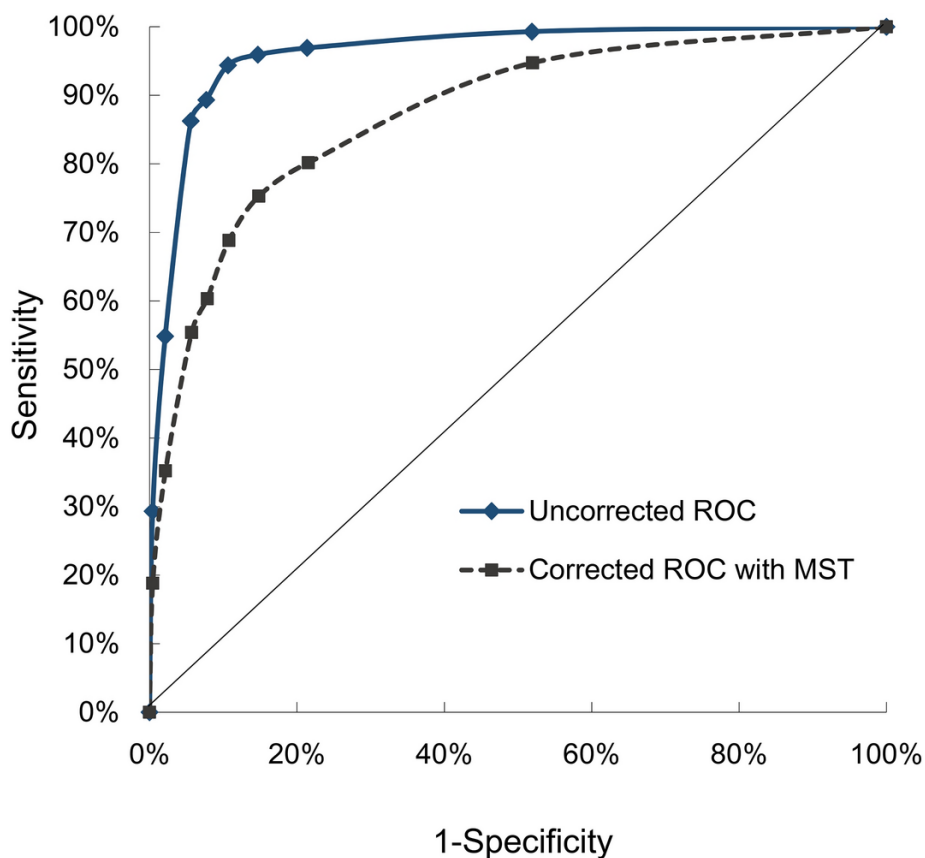


Figure 1.37: An example of a ROC curve showing the trade-off between sensitivity and specificity as a function of classification threshold. Taken from [135]

However, although the ROC AUC considers different decision boundary thresholds, it fails to consider the cost of a misclassification which may be important, depending on the application. Misclassifying a patient as cancer-free when they do in fact have cancer does not carry the same cost as the reverse. To overcome this, Hand¹³⁶ presented the H-Measure score which is similar to the ROC AUC except that it also takes into account different costs of misclassification. Here, a loss score is calculated for all thresholds for a distribution of cost values and the H score is given by the minimum of the loss value, normalised by the maximum possible loss for that classifier (1.13)

$$H = 1 - \frac{L}{L_{max}} \quad (1.13)$$

1.8.1.2 Cohens Kappa

Cohens Kappa (1.14) is a similar metric to the balanced accuracy but which also considers the agreement that would be expected due to chance.¹³⁷ It takes values between 0 and 1 and values above 0.6 are considered a good agreement.¹³⁸

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \quad (1.14)$$

Where p_0 is the TP + TN rate and p_e is the expected chance agreement.

1.8.2 Regression performance metrics

When considering the performance of a regression model, we can utilise the difference between the known value and the predicted value (the residual) to quantify the performance of the model overall.

Two common approaches for this are the Root Mean Squared Error (RMSE) (1.15) and the squared correlation coefficient (R^2). (1.16)

$$RMSE = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_i - \hat{y}_i)^2} \quad (1.15)$$

$$R^2 = \frac{\sum (y - \hat{y})^2}{\sum (y - \bar{y})^2} \quad (1.16)$$

The RMSE is a measure of the residuals of the model and the metric depends on the scale of the endpoint values, but a calculated metric closer to 0 indicates better prediction. The R^2 is a measure of

the percentage variance in the outcome that is explained by the model and is calculated using the output from the model prediction, \hat{y} , and the mean of the outcome, \bar{y} . It typically takes values between 0 and 1 with values close to 1 indicating good predictive performance.

1.9 Applicability domain

The Organisation for Economic Co-operation and Development (OECD)¹³⁹, an organisation that aims to establish international standards, met in 2004 to establish a range of guidelines for the validation of QSAR models.¹⁴⁰

Section 3 of the OECD principles for validation of a QSAR models states that all models should have a “defined domain of applicability”.¹⁴¹ The applicability domain of a model (Figure 1.38) is the chemical space in which the model makes predictions with a particular reliability.¹⁴² It is important to define the applicability domain of any model so that any validation of models takes place within the same chemical space in which the model was built.¹⁴³



Figure 1.38: A simple illustration of applying an applicability domain to a model. The rightmost plot shows that the applicability domain for this model is between the values of a and b and any point outside of this range is considered out of domain for the model. Taken from [143]

There are numerous ways of defining the applicability domain with no real consensus on the best approach. One common approach is a box approach of the chemical descriptors. In the box approach, each descriptor is checked for its minimum and maximum value in the training set. These minimum and maximum values are used as thresholds for the values of unseen data, not in the training set. If a descriptor value for an unseen data value falls outside of the thresholds from the training data, the point is considered out of domain.¹⁴⁴ This is summarised in Figure 1.39.¹⁴⁵ One drawback of this

min/max approach is that it fails to identify any gaps in the chemical space where predictions are likely to be poor.

Other approaches to define the applicability domain of a model include dimensional reduction of the data using Principal Component Analysis before applying a bounding box of the principal components or a Convex Hull approach which defines the smallest convex area containing the entire training set.¹⁴⁶ Neither of these approaches address the issue of regions with low density of training set molecules.

One approach that attempts to consider the density of training set points around a training set point is a similarity-based measure (Figure 1.39), built upon a kNN approach. In this approach, the average distance between data points in the descriptor space of the training set is calculated and then the distance between a new data point and its nearest training set point is calculated. If this value exceeds the average distance between training set molecules, the point is considered too dissimilar to the training set and is considered out of domain.¹⁴⁴

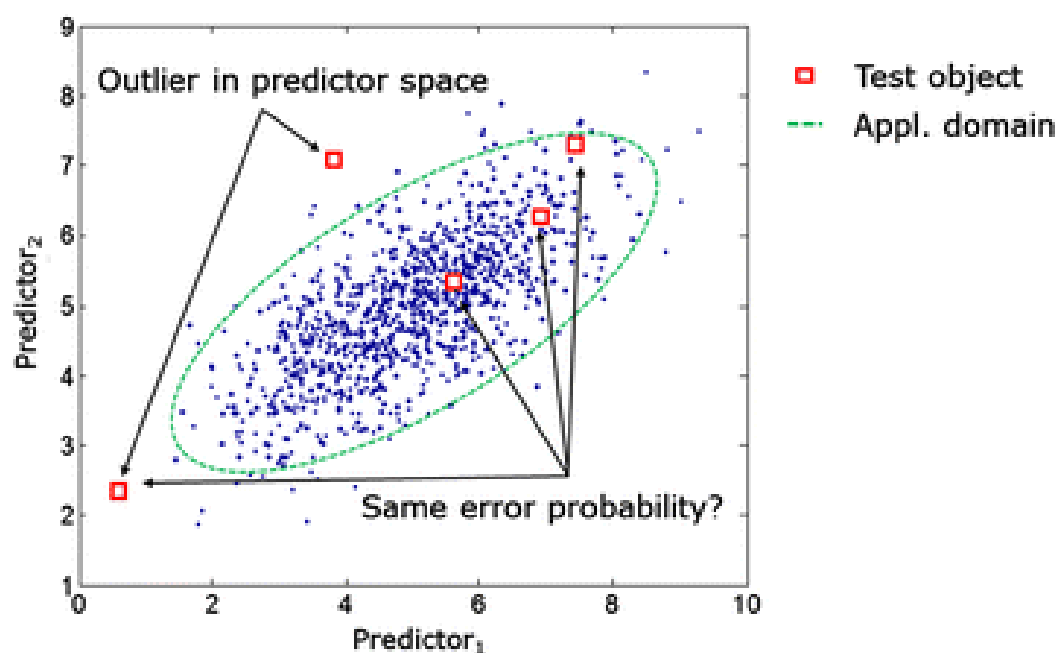


Figure 1.39: Example of the range check of a test set point between two predictors. The green area defines the descriptor space for predictors 1 and 2 with red test set points outside this space considered out of domain. Figure adapted from [145]

Combinations of a similarity measure and bounding box approach can help ensure that the molecules in an external set are both within the range of descriptors used to train the model and within a region of chemical space with sufficient training set density such that we can be reasonably confident in the ability of the external point to validate the model.

1.10 Y-randomisation

One final step in model validation is a form of hypothesis testing called Y-randomisation. The idea behind a hypothesis test is the desire to reject a null hypothesis by getting enough evidence to suggest that the hypothesis is false. In a QSAR approach, the null hypothesis is that the model was found by a chance correlation, and we wish to prove our alternative hypothesis, that the relationship found within the model is real, is true.

To do this, we can perform a shuffling of the endpoint values such that the features no longer necessarily are a true reflection of the endpoint being described. We can then train a number of models using the same approach as carried out in the true model and plot a distribution of their performance metrics and compare it to the performance metric calculated for the true model.

We can quantify our belief about whether our true model performance metric was drawn from the random model distribution or if it exists in a distinct and different distribution by the Z score (1.17).

$$Z = \frac{x - \text{average}(\tilde{x})}{\text{standard deviation}(\tilde{x})} \quad (1.17)$$

Where \tilde{x} is a performance metric value for a randomised model and x is the same performance metric of the true model. The Z score is a measure of the number of standard deviations the true model's metric is from the mean of the randomised distribution. Shen *et al.*¹⁴⁷ provides a table of significance values corresponding to the percentage chance of a chance correlation given a particular Z score.

1.11 Model Interpretability

As well as requiring a domain of applicability, the OECD guidelines also suggest a mechanistic interpretation, if possible.¹⁴¹ The idea of model interpretability is important, particular in social acceptance of machine learning models as they become more commonplace in society. It is important to know why a model is making a particular prediction, for example in a self-driving car, if the car is

consistently running over a cyclist – being able to interpret and assess the model to ascertain the important features it uses to define a cyclist could be useful to fine tune the model.¹⁴⁸

Machine learning algorithms can be interpreted using two broad approaches. Firstly, if the model is inherently interpretable then there are inbuilt methods to interpret the importance of features. The idea of a mechanistic interpretation is straightforward in simple linear models such as regression where the regression coefficients can act as the variable importance measure.

Other modelling algorithms that contain inherent interpretability are decision tree models such as the Random Forest which utilises the Gini importance, which for a feature calculates the average gain in purity for child nodes when split on this variable.¹⁴⁹ Another method with inherent interpretability is Naïve Bayes where the probability assigned to each feature and its contribution to the class prediction are measures of the importance.¹⁴⁸

However, as machine learning algorithms become non-linear and more complex such as in a neural network, these require model agnostic approaches to interpret. Model agnostic approaches can be classified as either global or local methods. If the approach is global, it describes the average behaviour of the model whereas the local approach describes each model prediction.¹⁴⁸

One interesting model agnostic approach is the Shapley Additive Values (SHAP) (Figure 1.40) based upon the idea of game theory. SHAP is interesting as it can both describe the global effects of the model by aggregating SHAP values, calculated from local interpretation of individual predictions.

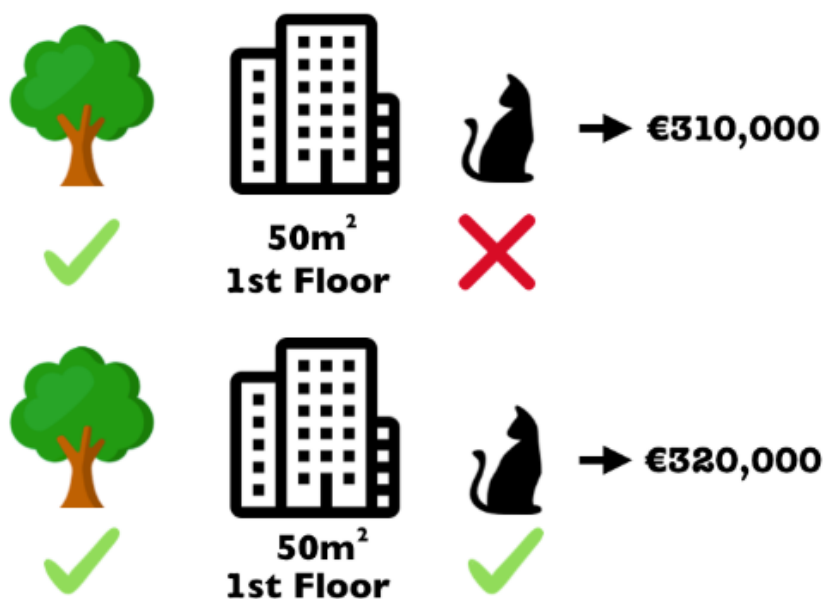


Figure 1.40: Toy example from Molnar showing the variation in prediction of house prices by allowing pets or not. Pets being allowed would be assigned a "payout" of 10,000 to the prediction. Figure adapted from [148]

SHAP values consider the descriptors as coalitions working together to a common goal. In this case, the SHAP values are considered in small coalitions that are tasked with predicting the endpoint. The difference between the total feature prediction and the coalition's prediction is noted and the difference is distributed between the members of the coalition as a "payout". This is repeated for many feature coalitions and each feature has its contribution averaged among all coalitions.¹⁵⁰

The SHAP approach has been used within QSAR over the past few years to interpret machine learning models. Rodriguez-Perez *et al.*¹⁵¹ utilised the SHAP approach to highlight important fragments in the prediction of molecular activity against 10 targets using a RF, SVM and NN model. Zhong *et al.*¹⁵² also use SHAP to interpret their XGBoost, deep neural network ensemble model which they built to predict HO• radical rate constants and Ye *et al.*¹⁵³ used SHAP to interpret their LightGBM model used in predicting the binding-free energy of pharmaceutical formulations.

1.12 Thesis overview

In this thesis, chapter 2 will build upon the classification models presented from within the group by Gupta *et al.* and address the concerns of model interpretability given by the choice of molecular descriptors used within the work. Moreover, using new descriptors will allow us to explore the model interpretability and try to rationalise why the model is making the predictions it does and attempt to elucidate any chemical reasoning underpinning the model's predictions.

Chapter 3 will begin to look at models for the prediction of the storage and loss moduli of gels created under the same reaction conditions of those used in the models of chapter 2. Since it is known to play an influential role in the potential applications of any gels, an *a priori* prediction of the likely stiffness and viscosity of a potential gel is an invaluable tool to prevent wastage in time and resources synthesising gels with unfavourable rheology.

Since we have described the difficulty in finding new gels, the work in chapter 3 has been built, by necessity, utilising a relatively "small data" approach which brings with it issues surrounding model uncertainty which will be discussed as part of chapter 4. In chapter 4 – we present models based on the same data in chapter 3 but the models in chapter 4 will utilise Bayesian inference to quantify the uncertainty in our predictions.

Finally, chapter 5 will leverage both our classification and regression models from chapters 2 and 4 utilising both an evolutionary algorithm and a generative neural network with reinforcement learning to generate new predicted hydrogelators with requested rheology parameters with these predictions compared to the measured rheology properties.

1.13 References

- 1 J. Alemán, A. V. Chadwick, J. He, M. Hess, K. Horie, R. G. Jones, P. Kratochvíl, I. Meisel, I. Mita, G. Moad, S. Penczek and R. F. T. Stepto, *Pure Appl. Chem.*, 2007, **79**, 1801–1829.
- 2 B. O. Okesola and D. K. Smith, *Chem. Soc. Rev.*, 2016, **45**, 4226–4251.
- 3 C. Colquhoun, E. R. Draper, R. Schweins, M. Marcello, D. Vadukul, L. C. Serpell and D. J. Adams, *Soft Matter*, 2017, **13**, 1914–1919.
- 4 L. A. Estroff and A. D. Hamilton, *Chem. Rev.*, 2004, **104**, 1201–1218.
- 5 J. N. Israelachvili, in *Intermolecular and Surface Forces*, Elsevier, 3rd edn., 2011, pp. 341–380.
- 6 L. Wang, X. Shi and J. Wang, *Soft Matter*, 2018, **14**, 3090–3095.
- 7 L. Qin, P. Duan, F. Xie, L. Zhang and M. Liu, *Chem. Commun.*, 2013, **49**, 10823–10825.
- 8 W. Deng and D. H. Thompson, *Soft Matter*, 2010, **6**, 1884–1887.
- 9 E. R. Draper and D. J. Adams, *Chem*, 2017, **3**, 390–410.
- 10 D. J. Adams, M. F. Butler, W. J. Frith, M. Kirkland, L. Mullen and P. Sanderson, *Soft Matter*, 2009, **5**, 1856–1862.
- 11 P. Terech and R. G. Weiss, *Chem. Rev.*, 1997, **97**, 3133–3159.
- 12 S. R. Caliari and J. A. Burdick, *Nat. Methods*, 2016, **13**, 405–414.
- 13 E. V. Alakpa, V. Jayawarna, A. Lampel, K. V. Burgess, C. C. West, S. C. J. Bakker, S. Roy, N. Javid, S. Fleming, D. A. Lamprou, J. Yang, A. Miller, A. J. Urquhart, P. W. J. M. Frederix, N. T. Hunt, B. Péault, R. V. Ulijn and M. J. Dalby, *Chem*, 2016, **1**, 298–319.
- 14 D. S. W. Benoit, M. P. Schwartz, A. R. Durney and K. S. Anseth, *Nat. Mater.*, 2008, **7**, 816–823.
- 15 N. Nandi, K. Gayen, S. Ghosh, D. Bhunia, S. Kirkham, S. K. Sen, S. Ghosh, I. W. Hamley and A. Banerjee, *Biomacromolecules*, 2017, **18**, 3621–3629.
- 16 B. Xing, C. W. Yu, K. H. Chow, P. L. Ho, D. Fu and B. Xu, *J. Am. Chem. Soc.*, 2002, **124**, 14846–14847.
- 17 S. Bhuniya, Y. J. Seo and B. H. Kim, *Tetrahedron Lett.*, 2006, **47**, 7153–7156.
- 18 P. K. Vemula, G. A. Cruikshank, J. M. Karp and G. John, *Biomaterials*, 2009, **30**, 383–393.
- 19 J. A. Sáez, B. Escuder and J. F. Miravet, *Tetrahedron*, 2010, **66**, 2614–2618.
- 20 X. Li, J. Li, Y. Gao, Y. Kuang, J. Shi and B. Xu, *J. Am. Chem. Soc.*, 2010, **132**, 17707–17709.

- 21 A. Allam, M. A. El-Mokhtar and M. Elsabahy, *J. Pharm. Pharmacol.*, 2019, **71**, 1209–1221.
- 22 A. Friggeri, B. L. Feringa and J. Van Esch, *J. Control. Release*, 2004, **97**, 241–248.
- 23 L. Yan, G. Li, Z. Ye, F. Tian and S. Zhang, *Chem. Commun.*, 2014, **50**, 14839–14842.
- 24 M. Mohar and T. Das, *Soft Mater.*, 2019, **17**, 328–341.
- 25 X. D. Xu, B. B. Lin, J. Feng, Y. Wang, S. X. Cheng, X. Z. Zhang and R. X. Zhuo, *Macromol. Rapid Commun.*, 2012, **33**, 426–431.
- 26 I. Yoshimura, Y. Miyahara, N. Kasagi, H. Yamane, A. Ojida and I. Hamachi, *Langmuir*, 2014, **30**, 12204–12205.
- 27 O. Gronwald and S. Shinkai, *Chem. - A Eur. J.*, 2001, **7**, 4328–4334.
- 28 O. Gronwald and S. Shinkai, *ChemInform*, 2010, **33**, 4329–4334.
- 29 A. R. Hirst and D. K. Smith, *Langmuir*, 2004, **20**, 10851–10857.
- 30 G. Gini, *Found. Chem.*, 2020, **22**, 383–402.
- 31 S. Awhida, E. R. Draper, T. O. McDonald and D. J. Adams, *J. Colloid Interface Sci.*, 2015, **455**, 24–31.
- 32 P. Ravarino, D. Giuri, D. Faccio and C. Tomasini, *Gels*, 2021, **7**, 1–10.
- 33 L. Chen, G. Pont, K. Morris, G. Lotze, A. Squires, L. C. Serpell and D. J. Adams, *Chem. Commun.*, 2011, **47**, 12071–12073.
- 34 A. Dawn and H. Kumari, *Chem. - A Eur. J.*, 2018, **24**, 762–776.
- 35 P. A. Janmey and R. T. Miller, *J. Cell Sci.*, 2011, **124**, 9–18.
- 36 Z. Wang, Y. Zhang, Y. Yin, J. Liu, P. Li, Y. Zhao, D. Bai, H. Zhao, X. Han and Q. Chen, *Adv. Mater.*, 2022, **34**, 1–20.
- 37 Y. Y. Xie, X. Q. Wang, M. Y. Sun, X. T. Qin, X. F. Su, X. F. Ma, X. Z. Liu, C. Zhong and S. R. Jia, *J. Mater. Sci.*, 2022, **57**, 5198–5209.
- 38 O. M. Benavides, A. R. Brooks, S. K. Cho, J. Petsche Connell, R. Ruano and J. G. Jacot, *J. Biomed. Mater. Res. - Part A*, 2015, **103**, 2645–2653.
- 39 R. Coronado, S. Pekerar, A. T. Lorenzo and M. A. Sabino, *Polym. Bull.*, 2011, **67**, 101–124.
- 40 M. C. Nolan, A. M. Fuentes Caparrós, B. Dietrich, M. Barrow, E. R. Cross, M. Bleuel, S. M. King and D. J. Adams, *Soft Matter*, 2017, **13**, 8426–8432.
- 41 H. Mcewen, E. Y. Du, J. P. Mata, P. Thordarson and A. D. Martin, *J. Mater. Chem. B*, 2017, **5**, 9412–9417.

- 42 N. Yan, Z. Xu, K. K. Diehn, S. R. Raghavan, Y. Fang and R. G. Weiss, *J. Am. Chem. Soc.*, 2013, **135**, 8989–8999.
- 43 Y. Lan, M. G. Corradini, R. G. Weiss, S. R. Raghavan and M. A. Rogers, *Chem. Soc. Rev.*, 2015, **44**, 6035–6058.
- 44 S. Meng, W. Li, X. Yin and J. Xie, *Comput. Theor. Chem.*, 2013, **1006**, 76–84.
- 45 X. Mu, K. M. Eckes, M. M. Nguyen, L. J. Suggs, P. Ren and S. V Sambasivarao, *Biomacromolecules*, 2013, **18**, 1199–1216.
- 46 I. P. Moreira, G. G. Scott, R. V. Ulijn and T. Tuttle, *Mol. Phys.*, 2019, **117**, 1151–1163.
- 47 J. K. Gupta, D. J. Adams and N. G. Berry, *Chem. Sci.*, 2016, **7**, 4713–4719.
- 48 F. Li, J. Han, T. Cao, W. Lam, B. Fan, W. Tang, S. Chen, K. L. Fok and L. Li, *Proc. Natl. Acad. Sci.*, 2019, **116**, 11259–11264.
- 49 R. Van Lommel, J. Zhao, W. M. De Borggraeve, F. De Proft and M. Alonso, *Chem. Sci.*, 2020, **11**, 4226–4238.
- 50 A. Y. T. Wang, R. J. Murdock, S. K. Kauwe, A. O. Oliynyk, A. Gurlo, J. Brgoch, K. A. Persson and T. D. Sparks, *Chem. Mater.*, 2020, **32**, 4954–4965.
- 51 R. Calo, *Univ. Bol. Law Rev.*, 2018, **3**, 180–218.
- 52 J. Burrell, *Big Data Soc.*, 2016, **3**, 1–12.
- 53 C. Hansch, P. P. Maloney, T. Fujita and R. M. Muir, *Nat. 1962 1944824*, 1962, **194**, 178–180.
- 54 S. Ekins, J. S. Freundlich, J. V. Hobrath, E. Lucile White and R. C. Reynolds, *Pharm. Res.*, 2014, **31**, 414–435.
- 55 E. L. Cáceres, M. Tudor and A. C. Cheng, *Future Med. Chem.*, 2020, **12**, 1995–1999.
- 56 S. Ekins, A. C. Puhl, K. M. Zorn, T. R. Lane, D. P. Russo, J. J. Klein, A. J. Hickey and A. M. Clark, *Nat. Mater.*, 2019, **18**, 435–441.
- 57 H. Koga, A. Itoh, S. Murayama, S. Suzue and T. Irikura, *Prog. Drug Res*, 1980, **23**, 105562.
- 58 W. Draber and C. Fedtke, in *Advances in Pesticide Science*, Pergamon, 1979, pp. IV–15.
- 59 G. Pilania, A. Mannodi-Kanakkithodi, B. P. Uberuaga, R. Ramprasad, J. E. Gubernatis and T. Lookman, *Sci. Rep.*, 2016, **6**, 1–10.
- 60 Z. Li, S. Wang, W. S. Chin, L. E. Achenie and H. Xin, *J. Mater. Chem. A*, 2017, **5**, 24131–24138.
- 61 P. Raccuglia, K. C. Elbert, P. D. F. Adler, C. Falk, M. B. Wenny, A. Mollo, M. Zeller, S. A. Friedler, J. Schrier and A. J. Norquist, *Nature*, 2016, **533**, 73.
- 62 J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad and R. G. Coleman, *J. Chem. Inf. Model.*,

- 2012, 52, 1757–1768.
- 63 A. Gaulton, A. Hersey, M. -I Nowotka, A. Patrícia Bento, J. Chambers, D. Mendez, P. Mutowo, F. Atkinson, L. J. Bellis, E. Cibrí an-Uhalte, M. Davies, N. Dedman, A. Karlsson, M. Paula Magariños Magari, J. P. Overington, G. Papadatos, I. Smit and A. R. Leach, *Nucleic Acids Res.*, 2016, **45**, 945–954.
- 64 D. Weininger, *J. Chem. Inf. Comput. Sci.*, 1988, **28**, 31–36.
- 65 O. Nicolotti, *Computational Toxicology Methods and Protocols*, Springer, 2018.
- 66 A. Mauri, V. Consonni and R. Todeschini, *Handb. Comput. Chem.*, 2017, 2065–2093.
- 67 A. Gribov, V. A. Dementiev, I. V Mikhailov and L. A. Gribov, *J. Struct. Chem.*, 2008, **49**, 197–200.
- 68 R. King, ed. K. Humbel, *Analytica Chimica Acta*, 1987, p. 294.
- 69 O. Devinyak, D. Havrylyuk and R. Lesyk, *J. Mol. Graph. Model.*, 2014, **54**, 194–203.
- 70 J. Gasteiger, J. Sadowski, J. Schuur, P. Selzer, L. Steinhauer and V. Steinhauer, *J. Chem. Inf. Comput. Sci.*, 1996, **36**, 1030–1037.
- 71 X. Zhao, M. Chen, B. Huang, H. Ji and M. Yuan, *Int. J. Mol. Sci.*, 2011, **12**, 7022–7037.
- 72 D. Rogers and M. Hahn, *J. Chem. Inf. Model.*, 2010, **50**, 742–754.
- 73 I. T. Jolliffe and J. Cadima, *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.*, 2016, 374.
- 74 L. C. Blum, R. Van Deursen and J. L. Reymond, *J. Comput. Aided. Mol. Des.*, 2011, **25**, 637–647.
- 75 X. F. Cadet, O. Lo-Thong, S. Bureau, R. Dehak and M. Bessafi, *Sci. Rep.*, 2019, **9**, 1–12.
- 76 C. Reynès, H. Host, A. C. Camproux, G. Laconde, F. Leroux, A. Mazars, B. Deprez, R. Fahraeus, B. O. Villoutreix and O. Sperandio, *PLoS Comput. Biol.*, 2010, **6**, 1–15.
- 77 S. E. Rensi and R. B. Altman, *J. Chem. Inf. Model.*, 2017, **57**, 1859–1867.
- 78 M. A. H. Khan, B. Thomson, R. Debnath, A. Motayed and M. V. Rao, *IEEE Sens. J.*, 2020, **20**, 6020–6028.
- 79 J. Ma and A. Motsinger-Reif, *BioData Min.*, 2021, **14**, 1–15.
- 80 R. D. Drennan, in *Interdisciplinary Contributions to Archaeology*, 2009, pp. 299–307.
- 81 D. Probst and J. L. Reymond, *J. Cheminform.*, 2020, **12**, 1–13.
- 82 J. Pujol-Giménez, M. Poirier, S. Bühlmann, C. Schuppisser, R. Bhardwaj, M. Awale, R. Visini, S. Javor, M. A. Hediger and J. L. Reymond, *ChemMedChem*, 2021, **16**, 3306–3314.
- 83 P. Schwaller, A. C. Vaucher, T. Laino and J. L. Reymond, *Mach. Learn. Sci. Technol.*, 2021, **2**, 1–9.

- 84 A. L. Chávez-Hernández, N. Sánchez-Cruz and J. L. Medina-Franco, *Biomolecules*, 2020, **10**, 1–16.
- 85 M. Bawa, T. Condie and P. Ganesan, *Proc. 14th Int. Conf. World Wide Web - WWW '05*, 2005, 651.
- 86 J. Šafránková, Annual Conference of Doctoral Students (19 2010.06.01-04 Prague), WDS'10 (19 2010.06.01-04 Prague) and Week of Doctoral Students 2010 (19 2010.06.01-04 Prague), 2010, 31–36.
- 87 R. J. May, H. R. Maier and G. C. Dandy, *Neural Networks*, 2010, **23**, 283–294.
- 88 A. R. W. Kennard and L. A. Stone, 2016, **11**, 137–148.
- 89 A. Saptorio, M. O. Tadé and H. Vuthaluru, *Chem. Prod. Process Model.*, 2012, **7**, 1–17.
- 90 T. M. Martin, P. Harten, D. M. Young, E. N. Muratov, A. Golbraikh, H. Zhu and A. Tropsha, *J. Chem. Inf. Model.*, 2012, **52**, 2570–2578.
- 91 A. Tropsha, A. Golbraikh and W. J. Cho, *Bull. Korean Chem. Soc.*, 2011, **32**, 2397–2404.
- 92 R. P. Sheridan, *J. Chem. Inf. Model.*, 2013, **53**, 783–790.
- 93 A. O. Sykes, *Coase-Sandor Inst. Law Econ. Work. Pap.*, 1993, **20**, 1–34.
- 94 S. García, J. Luengo and F. Herrera, *Intell. Syst. Ref. Libr.*, 2015, **72**, 163–193.
- 95 M. Kuhn and K. Johnson, *Applied predictive modeling*, Springer, 2013.
- 96 S. Kavitha, S. Varuna and R. Ramya, *Proc. 2016 Online Int. Conf. Green Eng. Technol.*, 2017, 1–5.
- 97 M. P. LaValley, *Circulation*, 2008, **117**, 2395–2399.
- 98 G. Mountrakis, J. Im and C. Ogole, *ISPRS J. Photogramm. Remote Sens.*, 2011, **66**, 247–259.
- 99 Z. R. Yang and Z. Yang, in *Comprehensive Biomedical Physics*, 2014, vol. 6, pp. 1–17.
- 100 L. Breiman, *Mach. Learn.*, 2001, 5–32.
- 101 A. Beloconi and P. Vounatsou, *Environ. Int.*, 2020, **138**, 105578.
- 102 Y. Mei, J. Li, D. Xiang and J. Zhang, *Remote Sens.*, 2021, **13**, 1–15.
- 103 W. Zhang, X. Chen, Y. Liu and Q. Xi, *IEEE Access*, 2020, **8**, 50118–50130.
- 104 A. D. Rodgers, H. Zhu, D. Fourches, I. Rusyn and A. Tropsha, *Chem. Res. Toxicol.*, 2010, **23**, 724–732.
- 105 C. Wang, Y. Long, W. Li, W. Dai, S. Xie, Y. Liu, Y. Zhang, M. Liu, Y. Tian, Q. Li and Y. Duan, *Sci. Rep.*, 2020, **10**, 1–12.

- 106 Q. P. He and J. Wang, *IEEE Trans. Semicond. Manuf.*, 2007, **20**, 345–354.
- 107 M. Kim, Y. Kim, H. Kim, W. Piao and C. Kim, *Front. Environ. Sci. Eng.*, 2016, **10**, 299–310.
- 108 K. J. T. Elevado, J. G. Galupino and R. S. Gallardo, *Int. J. GEOMATE*, 2018, **15**, 169–174.
- 109 H. Zhang, *Am. Assoc. Artif. Intell.*, 2004, 1–6.
- 110 H. Zhang, Y. L. Kang, Y. Y. Zhu, K. X. Zhao, J. Y. Liang, L. Ding, T. G. Zhang and J. Zhang, *Toxicol. Vitro.*, 2017, **41**, 56–63.
- 111 H. Zhang, J. X. Ma, C. T. Liu, J. X. Ren and L. Ding, *Food Chem. Toxicol.*, 2018, **121**, 593–603.
- 112 A. L. Perryman, J. S. Patel, R. Russo, E. Singleton, N. Connell, S. Ekins and J. S. Freundlich, *Pharm. Res.*, 2018, **35**, 1–10.
- 113 R. G. Brereton and G. R. Lloyd, *J. Chemom.*, 2014, **28**, 213–225.
- 114 M. P. Gómez-Carracedo, J. Ferré, J. M. Andrade, R. Fernández-Varela and R. Boqué, *Anal. Bioanal. Chem.*, 2012, **403**, 2027–2037.
- 115 S. Akbar, M. I. Majeed, H. Nawaz, N. Rashid, A. Tariq, W. Hameed, S. Shakeel, G. Dastgir, R. Z. A. Bari, M. Iqbal, A. Nawaz and M. Akram, *Anal. Lett.*, 2021, **55**, 1588–1604.
- 116 M. Joswiak, Y. Peng, I. Castillo and L. H. Chiang, *Control Eng. Pract.*, 2019, **93**, 104189.
- 117 L. Breiman, *Mach. Learn.*, 1996, **24**, 123–140.
- 118 Machine Learning- Decision Trees and Random Forest Classifiers | by Karan Kashyap | Analytics Vidhya | Medium, <https://medium.com/analytics-vidhya/machine-learning-decision-trees-and-random-forest-classifiers-81422887a544>, (accessed 14 April 2022).
- 119 J. Carrete, W. Li, N. Mingo, S. Wang and S. Curtarolo, *Phys. Rev. X*, 2014, **4**, 1–9.
- 120 S. Nagasawa, E. Al-Naamani and A. Saeki, *J. Phys. Chem. Lett.*, 2018, **9**, 2639–2646.
- 121 S. Kwak, J. Kim, H. Ding, X. Xu, R. Chen, J. Guo and H. Fu, *J. Mater. Res. Technol.*, 2022, **18**, 520–530.
- 122 S. L. Salzberg, *Mach. Learn.*, 1994, **16**, 235–240.
- 123 R. Fletcher, *Comput. J.*, 1970, **13**, 317–322.
- 124 D. Goldfarb, *Math. Comput.*, 2006, **24**, 23.
- 125 D. F. Shanno, *Math. Comput.*, 2006, **24**, 647.
- 126 R. Fei, J. Sha, Q. Xu, B. Hu, K. Wang and S. Li, *Eurasip J. Wirel. Commun. Netw.*, 2020, **2020**, 1–25.
- 127 J. Behler and M. Parrinello, *Phys. Rev. Lett.*, 2007, **98**, 1–4.

- 128 A. Roosta, P. Setoodeh and A. Jahanmiri, *Ind. Eng. Chem. Res.*, 2012, **51**, 561–566.
- 129 G. Montavon, M. Rupp, V. Gobre, A. Vazquez-Mayagoitia, K. Hansen, A. Tkatchenko, K. R. Müller and O. Anatole Von Lilienfeld, *New J. Phys.*, 2013, **15**, 1–17.
- 130 C. Cortes and V. Vapnik, *Mach. Learn.*, 1995, **20**, 273–297.
- 131 X. Ding, J. Liu, F. Yang and J. Cao, *J. Franklin Inst.*, 2021, **358**, 10121–10140.
- 132 R. M. Balabin and E. I. Lomakina, *Analyst*, 2011, **136**, 1703–1712.
- 133 A. O. Oliynyk, L. A. Adutwum, J. J. Harynuk and A. Mar, *Chem. Mater.*, 2016, **28**, 6672–6681.
- 134 X. Wang, Y. Hong, M. Wang, G. Xin, Y. Yue and J. Zhang, *Phys. Chem. Chem. Phys.*, **21**, 9159.
- 135 H. H. Jen, W. J. Chang, C. Y. Hsu, A. M. F. Yen, A. Auvinen, T. H. H. Chen and S. L. S. Chen, *Sci. Rep.*, 2020, **10**, 1–8.
- 136 C. Anagnostopoulos, D. J. Hand and N. M. Adams, *Measuring classification performance: the hmeasure package .*, 2012.
- 137 J. Cohen, *Educ. Psychol. Meas.*, 1960, **20**, 37–46.
- 138 P. Czodrowski, *J. Comput. Aided. Mol. Des.*, 2014, **28**, 1049–1055.
- 139 Home page - OECD, <https://www.oecd.org/>, (accessed 7 May 2022).
- 140 European Commission Environment Directorate General, 2007, 1–154.
- 141 O. for E. C. and Development, *The report from the expert group on (Quantitative) StructureActivity Relationships[(Q)SARs] on the principles for the validation of (Q)SARs*, 2004.
- 142 J. S. Jaworska, M. Comber, C. Auer and C. J. Van Leeuwen, *Environ. Health Perspect.*, 2003, **111**, 1358–1360.
- 143 C. Sutton, M. Boley, L. M. Ghiringhelli, M. Rupp, J. Vreeken and M. Scheffler, *Nat. Commun.*, 2020, **11**, 1–9.
- 144 F. Sahigara, K. Mansouri, D. Ballabio, A. Mauri, V. Consonni and R. Todeschini, *Molecules*, 2012, **17**, 4791–4810.
- 145 W. Klingspohn, M. Mathea, A. Ter Laak, N. Heinrich and K. Baumann, *J. Cheminform.*, 2017, **9**, 1–17.
- 146 F. P. Preparata and M. I. Shamos, 1985, 95–148.
- 147 M. Shen, A. LeTiran, Y. Xiao, A. Golbraikh, H. Kohn and A. Tropsha, *J. Med. Chem.*, 2002, **45**, 2811–2823.
- 148 C. Molnar, *Interpretable Machine Learning*, 2nd edn., 2022.

- 149 B. H. Menze, B. M. Kelm, R. Masuch, U. Himmelreich, P. Bachert, W. Petrich and F. A. Hamprecht, *BMC Bioinformatics*, , DOI:10.1186/1471-2105-10-213.
- 150 S. M. Lundberg and S. I. Lee, *Adv. Neural Inf. Process. Syst.*, 2017, **2017-Decem**, 4766–4775.
- 151 R. Rodríguez-Pérez and J. Bajorath, *J. Med. Chem.*, 2020, **63**, 8761–8777.
- 152 S. Zhong, K. Zhang, D. Wang and H. Zhang, *Chem. Eng. J.*, 2021, **405**, 1–12.
- 153 Z. Ye, W. Yang, Y. Yang and D. Ouyang, *Food Front.*, 2021, **2**, 195–207.

Chapter 2 - Interpretable Classification Models to Predict Gelation State of Low Molecular Weight Peptides

2.1 Introduction

2.1.1 Importance of predictive models for gels

As alluded to in Chapter 1, low molecular weight gels are a versatile class of material, with a wide range of potential interesting interactions in medicine¹⁻³, cell culturing^{4,5}, 3D printing^{6,7}, sensing⁸ and pollution clean-up.^{9,10} However, the art of producing a low molecular weight gel has been referred to as a serendipitous process.¹¹ Work has been reported within the group¹² and elsewhere^{13,14} presenting predictive models for the gelation ability of small molecules. However, these works lack an interpretation element that may help rationalise gel formation and help assist in the generation of new gels.

In this chapter, we will expand upon the classification models presented in Gupta *et al.*¹² Initially, we will aim to replicate the performance of the reported models using the same modelling algorithms reported within (2.3.1 *Fingerprints as bits*). Then, we will build upon these models by calculating different descriptor sets that allow for the mapping of a chemical fingerprint back on the molecule. We will investigate the difference in performance calculating the descriptors as binary absence or presence (2.3.2 *Fingerprints as features*).

We will then carry out a similar model building procedure in Python and assess the similarity of performance with the best of our R models. We chose to build models in Python due to the wide-ranging use and open-source nature of Python. This allows us to be much more flexible in the deployment of our models and allows us to combine it with the work in Chapter 5 for generation of new molecules. Moreover, the descriptors calculated in Python allow for direct comparison with those from Gupta whilst maintaining model interpretability (2.3.4 *Python Classifier*).

2.2 Experimental

Our dataset for model building was curated from the Dave Adams group at the University of Glasgow. It is the same dataset as presented in the models from within the group presented in Gupta *et al.*¹² (see Appendix 2.6.1 for structures and gel state of the dataset). Although this leaves us with a relatively small dataset of 62 molecules, since the process of gelation is so important in determining whether a molecule will form a gel or not, the dataset was curated to ensure that the same gelation process was carried out for each dataset. This is vital in ensuring the quality of the data used for model building. Gels were prepared using 5mg/mL of the precursor and the gelation triggered using the glucono- δ -

lactone trigger.¹⁵ A molecule was considered a gel if it retained its self-supporting structure 18 hours after inversion. The dataset contains functionalised amino acids and peptide mimetics.

2.2.1 Splitting of data

The dataset was split using a time split procedure and the molecules were placed into training, test and validation sets based on when they were synthesised. This resulted in a training set of 34 molecules (17 gelators and 17 non-gelators), 21 test set molecule (4 gelators and 17 non-gelators) and a validation set of 7 molecules (2 gelators and 5 non-gelators). Balanced training sets are preferable for a machine learning algorithm to learn to decipher the characteristics of molecules that gel and those that don't.¹⁶ The nature of the test and validation sets being heavily non-gelator is a fair reflection of the chemical space where more systems are non-gels than gels.

2.2.2 Software

Our modelling workflow utilises a range of software packages including Pipeline Pilot, the statistical programming software R and the open-source programming language Python. Pipeline Pilot is a workflow software that contains modules for chemistry molecular descriptor calculation, data manipulation, machine learning and data visualisation.¹⁷ R is a free software package for statistical modelling and visualisation, it contains packages for machine learning such as the **Classification And Regression Training** (caret) package¹⁸.

Python is an open-source programming language popular in the fields of data science and machine learning¹⁹. It contains a myriad of packages that make it a good choice to carry out machine learning projects. Packages used include the open source cheminformatics package RDKit²⁰ for descriptor calculation, Pandas^{21,22} for dataset manipulation, NumPy²³ for mathematical operations, scikit-learn²⁴ for data processing, machine learning and model evaluation and Matplotlib²⁵ and Seaborn²⁶ for data visualisation.

2.2.3 Descriptors

2.2.3.1 R Model Descriptors

Since we are building models using both caret in R and scikit-learn in Python, two different descriptor calculation procedures were undertaken. For the R models, we utilise Pipeline Pilot (version 16.5.0.143)¹⁷ to calculate the molecular descriptors for our models. Molecular properties:

- *AlogP*: The octanol-water partition coefficient, a measure of the hydrophobicity of a molecule
- *Molecular Weight*: Mass of the molecule, summation of the atomic mass of all atoms present.
- *Number of Atoms*: Count of the total number of atoms in the molecule
- *Number of Rotatable Bonds*: Number of bonds that allow full rotation around themselves, defined as single bonds bonded to a non-terminal heavy atom.
- *Number of Rings*: Total number of complete ring systems present in the molecule.
- *Number of Hydrogen-Bond Acceptors*: Count of the number of possible hydrogen bond acceptors (O, N, S, P with a lone pair).
- *Number of Hydrogen-Bond Donors*: Count of the number of possible hydrogen bond donors (O, N, S, P with an attached hydrogen atom).
- *Number of Aromatic Rings*: Similar to number of rings but with planar rings with π -bonding resonance forms.
- *Molecular Surface Area*: Considering the atoms and their van der Waals radius as spheres, the sum of the surface area of each sphere in a molecule²⁷
- *Molecular Polar Surface Area*: Total surface area of the polar atoms within a molecule. Calculated through the sum of the surface contributions of the polar elements of the molecule. The surface contributions of a polar element are determined through a least-squares fitting to a single conformer 3D PSA for a set of drug-like molecules.²⁷
- *Molecular Polar Solvent Accessible Surface Area*: Surface area of the polar molecules within a molecule accessible by solvent molecules, an implementation of the Shrake method.²⁸
- *Molecular Solubility*: Ability of the molecule to form a solution with water calculated using E-State indices.²⁹
- *LogD*: Octanol-water coefficient of the molecule with consideration for the ionisation states of the molecule.

were calculated along with both the Extended Connectivity Fingerprint (ECFP) with radius 4 and the Functional Class Fingerprint (FCFP) also with radius 4.³⁰

The R models differ based upon how the ECFP and FCFP fingerprints are represented. They are summarised as follows:

- *FP_as_bits (2.3.1)*: Both fingerprints were converted to a 2048 bit string which is large enough to minimise potential bit collisions for our dataset and has been shown to perform well in machine learning tasks.^{31,32} The fingerprint bit strings were presented as a binary 1 and 0 to represent present or absent in the molecule
- *FP_as_features (2.3.2)*: No conversion of the generated fingerprints and the total number of fingerprints calculated was the total number of possible unique fingerprints in the dataset. The fingerprints were presented as a binary 1 and 0.

2.2.3.2 Python Descriptors

In our Python (*version 3.8.5*) models, where we use RDKit (*version 2020.09.1*) as the descriptor generating tool, only LogD, Num_Rings and AlogP were calculated – as these were most frequently present after pre-processing (*see 2.2.4 Data Pre-processing*) of the R model descriptors (*see 2.2.3.1 R Model Descriptors*). Molecular Solubility was calculated using an implementation of Delaney’s ESOL in RDKit.³³ This was to attempt to keep the descriptors as similar as possible between R and Python. Molecular_PolarSASA was also present after pre-processing of the R models but is only available in the Linux distribution of RDKit and therefore it was omitted from the descriptor set.

Both the ECFP and FCFP were calculated using a radius 2 (which is analogous to the radius of 4 in Pipeline Pilot) and were converted to a 2048-bit string. RDKit contains functionality to access the bit from the converted bit string and so conversion was carried out. There were no issues visualising fragments using RDKit. We call this dataset *Python_Classifier (2.3.3)*.

2.2.4 Data Pre-processing

For R models, pre-processing, model training and evaluation were carried out using the caret package³⁴ (*version 6.0-86*) in R (*version 3.6.2*). Firstly, on our training set we use the `caret::nearZeroVar()` function on the descriptors to remove all descriptors with near zero variance, where 95% of the columns values were set by a single value. Next, the pairwise correlation of the remaining descriptors was found using `caret::findCorrelation()` to find features where their correlation exceeds 0.9 and those within the pair with a larger average correlation to the rest of the feature set were removed.

Also, individual pre-processing was carried out for the *fp_as_features dataset*. We manually removed Num_RotatableBonds from this dataset so that the molecular properties match exactly with those remaining in the *fp_as_bits* dataset so that the dataset only differed in the fingerprints we are comparing.

For the Python classifier, the same approach was taken using the VarianceThreshold function in scikit-learn (*version 0.23.2*), again removing those with less than 5% variance. To find the correlation, the corr() function in Python was used and any descriptor with a correlation with another above 0.9 was removed.

In both cases, the features of the test and validation set were curated to match those features that remained after processing of the training set features.

2.2.5 Descriptor Visualisation

We visualised all datasets descriptors using PCA in both R and Python. We also carry out a comparative visualisation using a Tree Manifold Approximation (TMAP) approach using the TMAP package (*version 1.0.6*) in Python (*version 3.7.11*) in an Ubuntu (*version 20.04.2*) Windows Subsystem for Linux environment.

2.2.6 Model selection

For our R models, we train Random Forest, Neural Network, Support Vector Machine, Naïve Bayes, k-Nearest Neighbours, C5.0 and Partial Least Squares models to allow direct comparison of models with the models published in Gupta et al.¹² For our Python model, we use the performance of our R models to determine which method to use and owing to the best performance across the descriptor sets of the Random Forest model, it was chosen as the method to deploy in the *Python_Classifier* (see 2.3.3 *Python_Classifier*) work .

2.2.7 Model building

R models were trained using the caret::train() functionality using custom parameter search grids and 10-fold cross validation, repeated 10 times. Part of the train() call carries out a centre and scaling of the data for all models. In total, we learnt 10 models for each method, leading to 70 models in total

(from 10 models for 7 methods). We then predicted the gelation state of the test and validation sets using our trained models.

In Python, our RF model was trained using the GridSearchCV function for hyperparameter tuning in scikit-learn (*version 0.23.2*). Our search grid was as follows:

- Number of trees = [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]
- Max depth = 10 to 50 in intervals of 5 and No max
- Minimum samples required to split a node = 2,5 and 10
- Minimum samples required in a leaf node = 1,2 and 4

2.2.8 Model validation

To evaluate the performance of our models, we assessed the performance of our models to correctly predict the class labels using three approaches using the balanced accuracy, the Cohen's kappa and the H-Measure. We employed the same measure of good prediction from the Gupta paper¹² and state that models are good if the Balanced Accuracy > 0.7, Hmeasure > 0.6 and Kappa > 0.4.

As a further validation step, the Z-score is calculated (*see Chapter 1: 1.10 Y-randomisation equation 1.17*) for the H-Measure statistic on the in-domain test set. Once the Z-score is calculated an α value can be calculated which gives a percentage confidence that the metric is not drawn from the random distribution. To calculate α , the cumulative distribution of a normal distribution with mean 0 and variance 1 is used to calculate the probability that the Z-score is not drawn from the random distribution.

2.2.9 Applicability domain

We define the domain of applicability for our models using the same approach as the work in Gupta *et al.* The applicability domain was defined by a range check of the molecular descriptors of external molecules against the minimum and maximum values of the training set descriptors. Moreover, the same approach is applied to the loadings of the first 10 principal components of the PCA space of the molecular descriptors. If any of the molecular descriptors or PCA loadings fall outside of the minimum or maximum range, the point was considered out of domain.

2.2.10 Model interpretation

2.2.10.1 R Model Interpretation

We investigated the interpretability of our models using two approaches. Firstly, we used the `caret::VarImp()` function, which uses an inherent variable importance metric if it exists, or defaults to `caret::filterVarImp()` if it does not. The `caret::filterVarImp()` function carries out a ROC curve analysis and the AUC is calculated. The variable importance for a variable using this approach is a measure of how well the data can be separated into classes, based on that variable. SHAP values³⁵ were also calculated for the R models, using the R wrapper (*version 0.1.3*) of the Shap package in Python.

2.2.10.2 Python Model Interpretation

For the Python model, we use the intrinsic variable importance measure within the Random Forest and compare the results with the Python Shap package (*version 0.39.0*). It is worth noting, that the caret variable importance measures the impact of the training descriptors whereas in SHAP we utilise both the joint test and validation set descriptors.

2.3 Results and Discussion

2.3.1 Fingerprints as bits

2.3.1.1 Descriptors and Visualisation

After descriptor calculation, there were 4109 columns (2048 ECFP, 2048 FCFP and 13 physicochemical descriptors). After preprocessing, there were 54 descriptors remaining comprising of AlogP, Num_Rings, Molecular_PolarSASA and Molecular_Solubility as the physicochemical descriptors. Of the remaining 50 descriptors, 25 were ECFP bits and 25 were FCFP bits (*see 2.6.2 fp_as_bits fingerprint bits for a list of the bits*).

Visualisation of the dataset in reduced dimensions can be seen in Figure 2.1. Figure 2.1a shows the points of the training, test and validation set represented by their first two principal components. It is worth noting that the first two principal components only account for 37% of the data (PCA-1 = 21%, PCA2 = 16%) and so variances between points are not completely captured in the figure. 87% of the variance in the dataset is explained by the first 10 principal components. However, with that caveat, within the PCA space, there is the highest density of training set points below 0 for the PC2 axis and few training set points in the PC2 0-5 range where most of the test set points are located.

Furthermore, the TMAP representation of the dataset in Figure 2.1b, presents the dataset as a single branch (as defined by the minimum spanning tree generated) with a series of sub-branches. Given that our dataset is small and all molecules within are peptide based, this is not very surprising. Also, the distance between points on the plot is a measure of how dissimilar they are – molecules further apart are more dissimilar.

However, the most interesting aspect of the plot in Figure 2.1b is the sub-branch that appears to only contain test set molecules. Given that there are no training set molecules within this “cluster” it would be interesting to investigate whether these points are incorrectly classified or if their probability of forming a gel is less certain than those with training points in the same cluster (*see 2.3.2.5 Relating performance to descriptor visualisation*).

PCA and TMAP plots of the *fp_as_bits* dataset

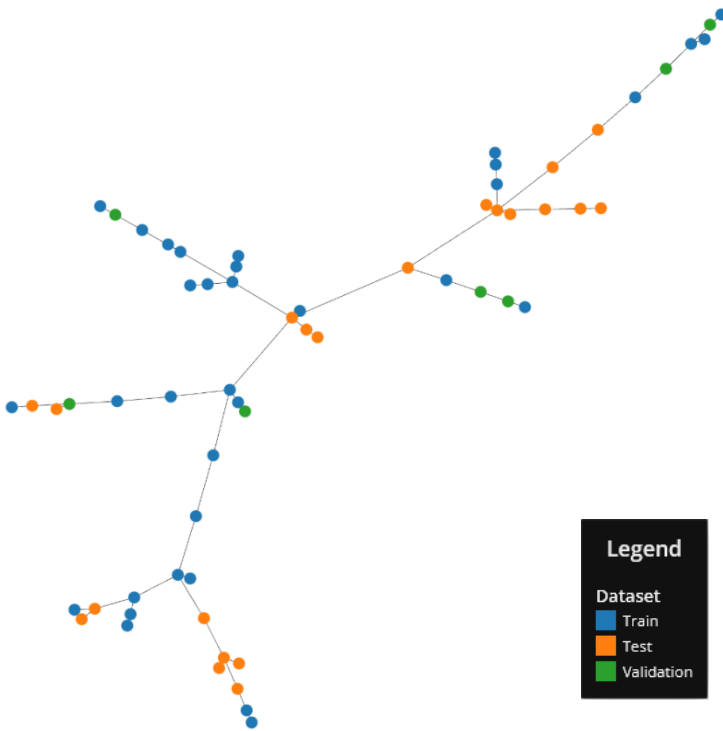
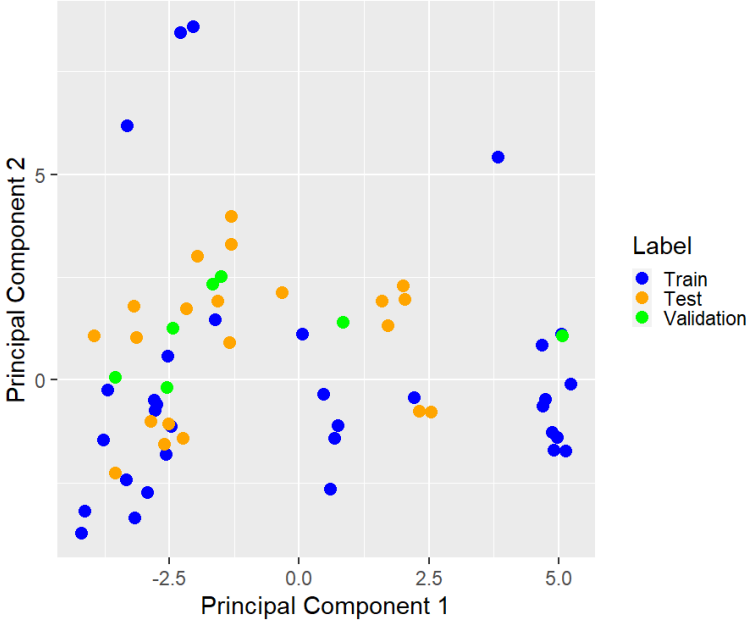


Figure 2.1 Visualisation of the first two principal components of the *fp_as_bits* dataset and b) Visualisation of the dataset using TMAP

2.3.1.2 Training Performance

All models trained with hyperparameter tuning via cross validation, regardless of method, showed good predictive performance on the training data. Table 2.1 highlights the performance of the models on the training set, and the average performance of the 10 models summarised in the quality of predictions column. In Table 2.1, the “Quality of Predictions” is a qualitative label assigned to the average predictions based on the thresholds in Gupta *et al.*¹² which were that the models satisfy all of: H-Measure > 0.6, Balanced Accuracy > 0.7 and Kappa > 0.4

Table 2.1: Summary of the average (\pm standard deviation) performance for each modelling method on the training set.

Method	Kappa	Balanced Accuracy	H-Measure	Quality of Predictions
RF	1.00 (\pm 0.00)	1.00 (\pm 0.00)	1.00 (\pm 0.00)	Good
SVM	0.92 (\pm 0.04)	0.96 (\pm 0.02)	0.91 (\pm 0.00)	Good
kNN	0.83 (\pm 0.02)	0.91 (\pm 0.01)	1.00 (\pm 0.00)	Good
NN	0.96 (\pm 0.03)	0.98 (\pm 0.01)	0.95 (\pm 0.05)	Good
NB	0.76 (\pm 0.00)	0.88 (\pm 0.00)	0.76 (\pm 0.00)	Good
PLS	0.73 (\pm 0.22)	0.86 (\pm 0.11)	0.85 (\pm 0.10)	Good
C5.0	1.00 (\pm 0.00)	1.00 (\pm 0.00)	1.00 (\pm 0.00)	Good

All models meet the minimum requirements for good for both the balanced accuracy and the H-Measure with all models showing strong training performance. All models average a Kappa above 0.73, a Balanced Accuracy above 0.86 and a H-Measure above 0.76. For the tree-based RF and C5.0 algorithms, both have perfect classification during training with scores for Kappa, Balanced Accuracy and H-Measure of 1.00 (\pm 0.00) for all three metrics. This highlights that there is potential for over-fitting for these models which will be assessed during testing. The Naïve Bayes and Partial Least Squares methods perform comparatively poorly in training with the NB scoring 0.76 (\pm 0.00) for Kappa, 0.88 (\pm 0.01) for Balanced Accuracy and 0.76 (\pm 0.00) for H-Measure and the PLS method scoring 0.73 (\pm 0.22), 0.86 (\pm 0.11) and 0.85 (\pm 0.10) for the same metrics.

2.3.1.3 Test set performance

In 2.2.9 *Applicability Domain* we defined the applicability domain of our models as points in the test and validation set that were within the range check of the descriptor and PCA values. Of the 21 points in the test set 16 of these points were within the range check of the descriptor and PCA space (4 gels, 12 non-gels).

Table 2.2 shows the results on the 16 test set molecules that pass the range check AD filter. Overall, the performance on the test set is naturally below that of the training set. Of the good models during training, kNN, NB, PLS and C5.0 fail to perform well across the performance metrics on the test set.

This confirms the degree of overfitting of the C5.0 model discussed above. For the kNN and NB models, we see a reduction in performance when exposed to the test set. For both, the biggest reduction is in the Kappa, which drops from 0.83 (± 0.02) to 0.29 (± 0.09) for the kNN and 0.76 (± 0.00) to 0.33 (± 0.00) for the NB. The PLS model only fails the check for good for Kappa, dropping below 0.40 with a score of 0.37 (± 0.28) meaning some of the PLS models will pass the threshold for good.

Like during training, RF and NN models perform well – with the highest BA (0.79 and 0.79) and high H-Measure scores (0.70 and 0.74) of the models tested. We see slightly worse performance in our SVM and PLS models, but they still perform well with high balanced accuracy (0.79 and 0.72) and high H-measure (0.73 and 0.60).

Table 2.2: Average performance of our 10 models for each method on the test set within the range check of the training set.

Method	Kappa	Balanced Accuracy	H-Measure	Quality of Predictions
RF	0.57 (± 0.15)	0.79 (± 0.06)	0.70 (± 0.07)	Good
SVM	0.52 (± 0.05)	0.79 (± 0.03)	0.73 (± 0.02)	Good
kNN	0.29 (± 0.09)	0.72 (± 0.07)	0.73 (± 0.05)	Bad
NN	0.49 (± 0.15)	0.82 (± 0.06)	0.74 (± 0.19)	Good
NB	0.33 (± 0.00)	0.75 (± 0.00)	0.71 (± 0.00)	Bad
PLS	0.37 (± 0.28)	0.72 (± 0.17)	0.60 (± 0.25)	Bad
C5.0	-0.02 (± 0.03)	0.49 (± 0.02)	0.39 (± 0.06)	Bad

2.3.1.4 Comparison with results of Gupta et al.

In the work by Gupta¹², the applicability domain is defined using the descriptor and PCA range check that we use as our initial filter. Therefore, our results from Table 2.2 are directly comparable to those reported. In their work, 12 of the 21 molecules pass the applicability domain threshold compared with 16 in our work. The discrepancy is likely due to differing approaches in calculating the PCA components (this work utilised Python for range checking and PCA calculation whereas Gupta *et al.* used Pipeline Pilot) and in defining the range checking procedure (we use Python CIM tools package (*version 4.0.8*) and Gupta et al used Pipeline Pilot). Python was chosen to carry out this approach as the process was much more user friendly.

Table 2.3 shows the results reported in the work by Gupta. The results are similar in terms of which models perform well with RF, SVM and NN passing the threshold in both cases. For the performance of the RF, SVM and NN performance is slightly better overall in Gupta particularly for H-Measure with values of 1.00, 0.70 and 1.00 in Gupta compared to 0.70, 0.73 and 0.74 in this work but the models presented here are still strong.

Table 2.3: Reports on the reduced test set from Gupta et al.

Method	Kappa	Balanced Accuracy	H-Measure	Quality of Predictions
RF	0.76	0.96	1.00	Good
SVM	0.42	0.71	0.70	Good
kNN	0.29	0.79	0.31	Bad
NN	0.46	0.88	1.00	Good
NB	0.29	0.79	0.53	Bad
PLS	0.18	0.63	0.53	Bad
C5.0	0.10	0.58	0.33	Bad

2.3.1.5 Validation set performance

In the validation set, of the 7 points in this set, 4 (1 gel, 3 non-gels) are within the range check of the training set. Since it is not feasible to calculate performance metrics or assess the performance of a model using a set of 4 points, we consider all 7 points in the validation set with the caveat that some would be considered out of domain. Even with this, all RF and SVM models predict the full validation set flawlessly.

2.3.1.6 Pareto-Optimal Model

Given the arbitrary nature of the cut-offs used to determine whether a model can be considered good, we now explore the pareto optimal models which aims to find the set of models where no further improvement in one metric can be found without detriment for another metric. These pareto-optimal models will aim to maximise Kappa, Balanced Accuracy and H-Measure for the range-domain test set molecules.

When calculating the pareto front for the models on the test set. There are five models in the pareto front and 4 of these are Neural Network models, confirming the good performance seen above. However, interestingly the 5th model in the pareto front is a PLS model. Since the PLS model failed the threshold for good on kappa (with a large variance in kappa), the results here show that a model that would have been overlooked based on the average performance (model 6) would require further inspection as it is potentially a useful model.

Looking at models within the 2nd and 3rd pareto fronts, Random Forest models make up 3 out of the 4 models within the first 2nd and 3rd pareto fronts. The remaining model is in the 3rd pareto front and is a Support Vector Machine model. The results of the pareto optimisation suggest that the Neural Network models show the best performance in the *fingerprints_as_bits* set of models.

2.3.1.7 Chemical Reasoning behind misclassification

Given the large number of models trained during this section, understanding the trends in misclassified molecules could allow for some interesting conclusions to be drawn about the performance of the models. Of the 16 molecules in the in-domain test set, the molecules misclassified the most overall are shown in Figure 2.2.

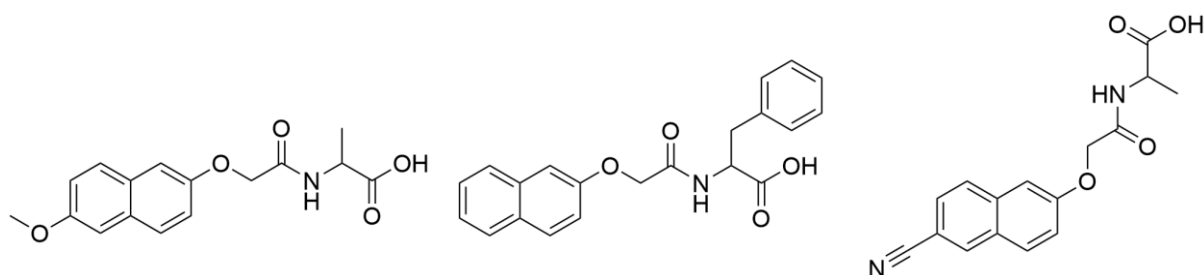


Figure 2.2: Molecules 1, 2 and 3 that are most misclassified by our *fp_as_bits* classifiers.

These molecules are misclassified 49, 41 and 40 times respectively. Given that there are 70 total predictions made for each molecule, these molecules are misclassified 70%, 59% and 57% of the time respectively. Suggesting that molecule 1 in particular is universally difficult to predict, regardless of the modelling algorithm. Moreover, when predicting molecules 2 and 3, the models do worse than guessing the gelation state of the molecule.

All three molecules are non-gels suggesting that the models are trained on data that suggests molecules like these form gels. However, when calculating the average Tanimoto similarity between the three molecules to the gels and non-gels in the training set, the similarities to both are similar. For molecule 1, the average similarity to gels in the training set is 0.44 compared to 0.42 to the non-gels in the training set. The difference is slightly more pronounced for molecule 2 with similarities of 0.58 to the gels and 0.45 to the non-gels but molecule 3 follows a similar pattern to molecule 1 with similarities of 0.41 and 0.38. These similarities suggest there are other reasons as to why the models predict these points poorly.

Conversely, there are 7 test set molecules that are perfectly classified by all 70 models, these are shown in Figure 2.3. These molecules are mostly functionalised amino acids with a single functionalised dipeptide. It is surprising that the models perform well on the functionalised amino acids given that only 6 of the 34 molecules in the training set are functionalised amino acids.

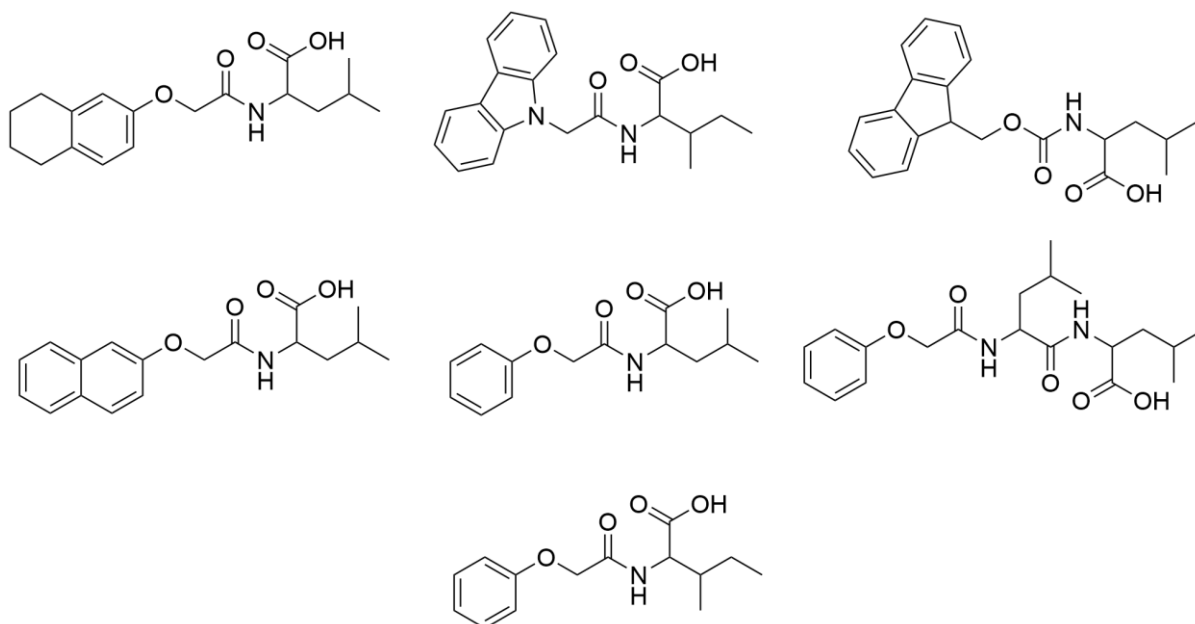


Figure 2.3 Test set molecules that are perfectly predicted by all 70 classification models.

2.3.1.8 Y-Randomisation

We have successfully replicated the results of those reported in Gupta *et al.* and we could use the new models as the basis of our comparison of new descriptor sets moving forward. However, one last check of the models' performance would be the results of models trained on randomised training data on the test set. Table 2.4 shows the average test set prediction of our RF, SVM and NN models on the in-domain test set. The dramatic fall in performance on the test set for the y-randomised models gives us good confidence that the predictions made on the test sets on the 10 normal models are utilising true relationships between the descriptors and gel state to make accurate predictions on unseen molecules.

Table 2.4: Performance of our randomised models on the in-domain test set.

Method	Kappa	Balanced Accuracy	H-Measure	Quality of Predictions
RF	-0.08 (\pm 0.24)	0.43 (\pm 0.18)	0.39 (\pm 0.19)	Bad
SVM	0.00 (\pm 0.25)	0.49 (\pm 0.15)	0.26 (\pm 0.12)	Bad
NN	-0.16 (\pm 0.24)	0.40 (\pm 0.14)	0.33 (\pm 0.15)	Bad
PLS	-0.05 (\pm 0.23)	0.47 (\pm 0.15)	0.39 (\pm 0.19)	Bad

As a further measure in our confidence as to the reliability of our models, we can calculate the Z score (see section Chapter 1: 1.10 Y-randomisation) for the H-Measure values calculated for all three models with respect to the average score obtained by these models on our true dataset. We take the average H-Measure as the score to compare and define the distribution in terms of the mean and standard deviation of the y-randomised H-Measure.

The RF Z-Score was calculated to be 1.62 meaning that the true average H-Measure is 1.62 standard deviations away from the y-randomised distribution. This results in a 5.24% chance that the true models are drawn from the same distribution as the randomised models.

For the SVM the Z-score is 3.82 giving 0.006% chance that the true model is drawn from the randomised SVM model distribution. For the NN this value is 2.77, resulting in 0.28% chance the model is random. The PLS model has a Z-score of 1.83 which corresponds to a randomised model chance of 3.38%. We have successfully recreated the results of Gupta *et al.* and can use these models as our baseline models as we explore new models with interpretability.

2.3.2 Fingerprints as features

Now that we have successfully replicated the work in Gupta *et al.* we use this as a baseline to compare the performance of models moving forward. One of the limitations of the work of Gupta *et al.* was the lack of interpretation resulting from loss of chemical information following fingerprint folding and we attempt to address this in the models that follow.

2.3.2.1 Comparison with *fp_as_bits* dataset

After descriptor calculation, there were 253 descriptors (152 ECFP descriptors, 88 FCFCP descriptors and 13 physicochemical). After preprocessing of the *fp_as_features* descriptors, there were 54 descriptors remaining (see Appendix 2.6.3 for a list of descriptors), including AlogP, Num_Rings, MolecularPolarSASA and Molecular_Solubility which were also the physicochemical descriptors of the *fp_as_bits* dataset. However, in the *fp_as_features* dataset, Number_RotatableBonds was present after pre-processing but the decision was taken to remove it from the dataset so that the *fp_as_bits* and *fp_as_features* datasets differed only in the different fingerprints we were using, allowing for a more direct comparison into the influence the different fingerprint representation had on model performance. Of the remaining 49 descriptors, 5 were FCFCP descriptors and 44 were ECFP descriptors.

2.3.2.2 Descriptor Visualisation

Compared to the analogous plots in Fingerprints as bits, the Figures in 2.4a and 2.4b show the PCA and TMAP plots for the *fp_as_features* dataset. In the PCA plot, if the training set points were to define a bounding box in the first 2 principal components, only one point has a PC1 loading value marginally larger than the maximum value in the training set and there are no test or validation set values with loading values above or below the minimum and maximum value of the training set molecules in PC2. We would expect this point to fall outside the applicability domain.

In the TMAP plot, again there appears to be a sub-branch that consists of 4 test set points as the terminal points within a branch – suggesting that these share little similarity to training set points and raising questions about their applicability within our model's domain, as was shown for the Figure in 2.1b for the *fp_as_bits* dataset. Interestingly these are not all the same, but the final 2 points in that sub-branch are the same test set molecules, shown in Figure 2.5, and highlights that whilst different, the descriptors are maintaining some consistent relationships between molecules.

PCA and TMAP plots of the *fp_as_features* dataset

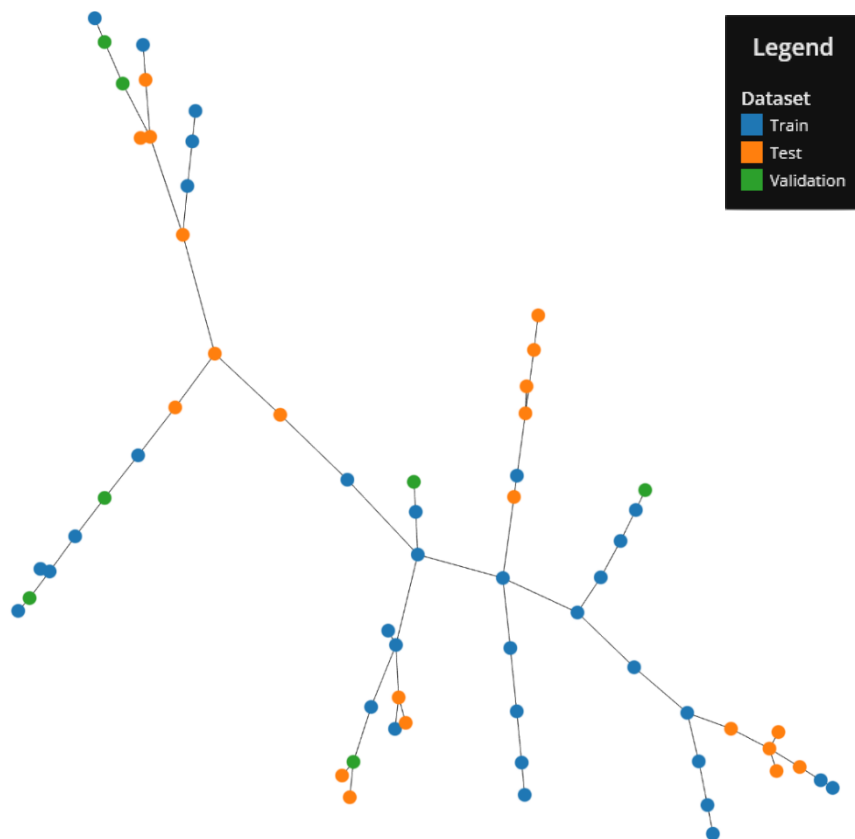
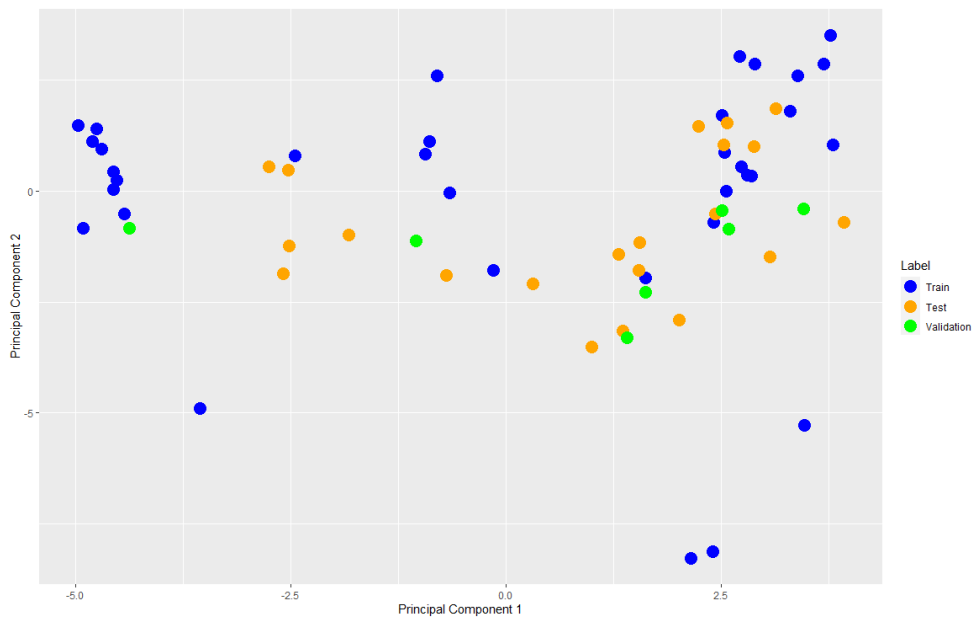


Figure 2.4: a) Plot of our *fp_as_features* dataset in PCA space and b) TMAP plot of our *fp_as_features* dataset.

Chemical representations of test molecules 20 and 21.

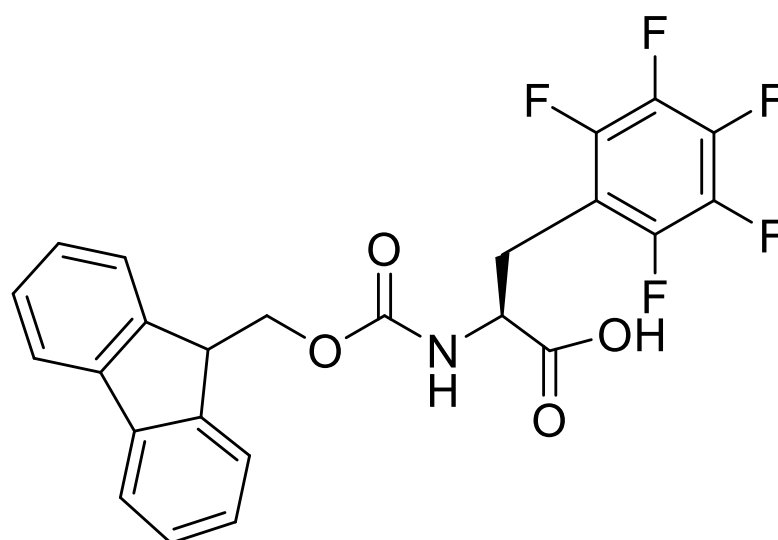
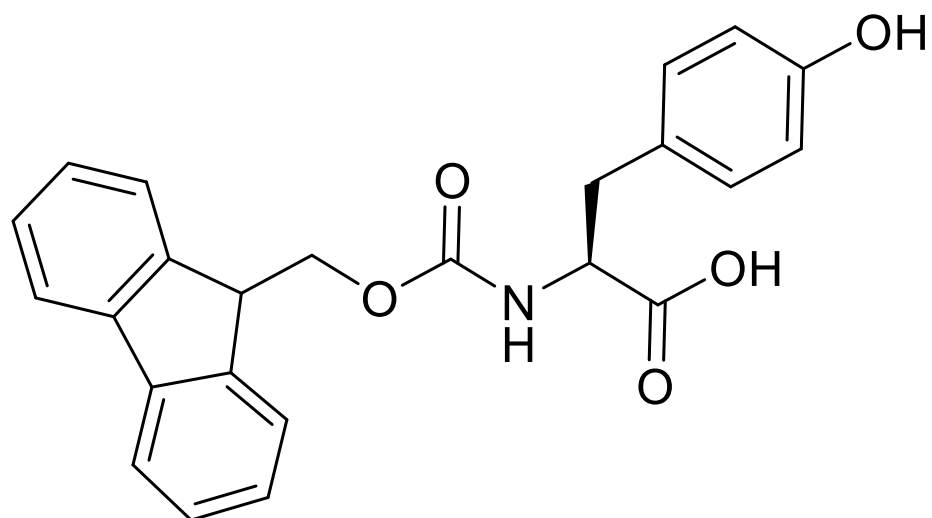


Figure 2.5: Molecules test_20 and test_21 whose TMAP representation remains consistent across descriptors

2.3.2.3 Training Performance

Our *fp_as_features* models show superior overall performance during training when compared to the *fp_as_bits* models, with the Kappa, Balanced Accuracy and H-Measure values in Table 2.5 showing an improved performance for the *fp_as_features* models here when compared to the values in Table 2.1 for the *fp_as_bits* dataset. All 7 modelling methods, on average, produce models that we consider good by the thresholds set. Overall, C5.0 shows the best performance in training, with perfect classification in all 10 models. However, it is likely these models overfit, despite the 10-fold cross validation repeated 10 times we carry out, but prediction on the test set will ascertain whether the C5.0 models are generalizable or not.

Like in the *fp_as_bits* models RF, SVM and NN models perform well with Kappa values of $0.99 (\pm 0.02)$, $0.96 (\pm 0.03)$ and $0.96 (\pm 0.03)$ and H-Measure scores of 1.00, $0.95 (\pm 0.05)$ and $0.93 (\pm 0.05)$ in Table 2.5. The performance remains good for models like the NB and PLS models which show similar performance to the *fp_as_bits* models with scores of $0.88 (\pm 0.00)$ and $0.76 (\pm 0.00)$ for balanced accuracy. Here, their scores are similar, with values of $0.85 (\pm 0.00)$ and $0.91 (\pm 0.07)$.

Table 2.5: Performance during training for our *fp_as_features* dataset

Method	Kappa	Balanced Accuracy	H-Measure	Quality of Predictions
RF	$0.99 (\pm 0.02)$	$0.99 (\pm 0.01)$	$1.00 (\pm 0.00)$	Good
SVM	$0.96 (\pm 0.03)$	$0.98 (\pm 0.02)$	$0.95 (\pm 0.05)$	Good
kNN	$0.84 (\pm 0.02)$	$0.92 (\pm 0.01)$	$1.00 (\pm 0.00)$	Good
NN	$0.96 (\pm 0.03)$	$0.98 (\pm 0.01)$	$0.93 (\pm 0.05)$	Good
NB	$0.71 (\pm 0.00)$	$0.85 (\pm 0.00)$	$0.76 (\pm 0.00)$	Good
PLS	$0.83 (\pm 0.13)$	$0.91 (\pm 0.07)$	$0.87 (\pm 0.08)$	Good
C5.0	$1.00 (\pm 0.00)$	$1.00 (\pm 0.00)$	$1.00 (\pm 0.00)$	Good

2.3.2.4 Test and validation set performance

Whether this improved performance in training translated into predictive performance was ascertained by predicting on the in-domain test set. Again, our applicability domain is defined by the range checking in both the descriptor space and PCA space. After range-checking, 16 molecules are again present in the reduced dataset (3 gels and 13 non-gels) and in the validation set this is 5 molecules (2 gels and 3 non-gels)

Performance on the reduced test set (Table 2.6) for the *fp_as_features* models is slightly improved compared to the performance of the reduced *fp_as_bits* set (Table 2.2). Even though in the *fp_as_bits* data we only considered the RF, SVM, NN and PLS models, the PLS model passes the kappa threshold again with an increased average kappa from 0.37 to 0.49. The RF performance is worse in terms of all performance metrics with kappa (0.57 for *fp_as_bits* vs 0.56 here), the same H-Measure of 0.70 and BA increasing slightly from 0.79 vs 0.82. For the NN it is on the boundary to pass the kappa threshold for good, having an average kappa of exactly 0.40 whilst demonstrating solid performance in terms of balanced accuracy (0.82) and H-Measure (0.79). The SVM model here falls below the threshold for good in terms of kappa, with the performance metrics decreasing from 0.52 to 0.38 for kappa.

Table 2.6: Performance of the reduced test set for all modelling algorithms explored

Method	Kappa	Balanced Accuracy	H-Measure	Quality of Predictions
RF	0.56 (\pm 0.17)	0.82 (\pm 0.09)	0.70 (\pm 0.06)	Good
SVM	0.38 (\pm 0.33)	0.75 (\pm 0.21)	0.79 (\pm 0.27)	Bad
kNN	0.35 (\pm 0.06)	0.79 (\pm 0.03)	0.81 (\pm 0.10)	Bad
NN	0.40 (\pm 0.04)	0.82 (\pm 0.02)	0.79 (\pm 0.24)	Good
NB	0.19 (\pm 0.00)	0.69 (\pm 0.00)	1.00 (\pm 0.00)	Bad
PLS	0.49 (\pm 0.28)	0.83 (\pm 0.18)	0.77 (\pm 0.27)	Good
C5.0	0.43 (\pm 0.09)	0.74 (\pm 0.05)	0.30 (\pm 0.02)	Bad

For the validation set, there are only 5 molecules remaining in the set after filtering (2 gels and 3 non-gels) and so with this caveat we predict the performance of the RF and PLS models on the full validation set. Again, the classification for all models on the full validation set is flawless.

2.3.2.5 Relating performance to descriptor visualisation

We re-plot Figure 2.1a in Figure 2.6 with the points sized as to the distance from the correct classification. To do this, we consider the thresholds of the models trained which were probability > 0.5 = gel and probability < 0.5 = non-gel. With this, we calculate the distance from 100% confidence in the prediction classification by the model depending on the true label of the point. For example, if a non-gel is predicted with a probability of 0.7 to be a gel it would be assigned a score of 0.7. If a gel was predicted to form a gel with the same probability, it was assigned a score of 0.3.

To carry this out, we chose the model that performs best on the applicability domain filtered test set for the RF and PLS models. The models are *rf_fit_3* and *pls_fit_1* which are the 3rd, and 1st model learnt for the respective models.

The results from these scores are displayed in Figures 2.6a and 2.6b where the TMAP plots are scaled by how far the probability is from perfect for the true class (distance from 0% predicted gel probability for non-gels and 100% predicted gel for gels) – with larger points illustrating points being closer to the decision boundary. There is a similar trend present in both plots with the PLS model having a generally larger score. The largest scores are reserved for the test set points at point 1 in the figure that are single point sub-branches and for the sub-branch containing test set points we discussed previously. This gives some indication that the TMAP can highlight potentially difficult predictions for the test and validation sets, but further work investigating whether this is widely applicable is required before solid conclusions are drawn.

Although the sub-branch of test set points does not have the largest residuals in this approach, the residuals are still large when compared to molecules that are within the main branch of the TMAP plot. This indicates that there is some difficulty predicting the points isolated from training set points in the TMAP plots.

TMAP representation of the *fp_as_features* dataset sized by distance to perfect classification

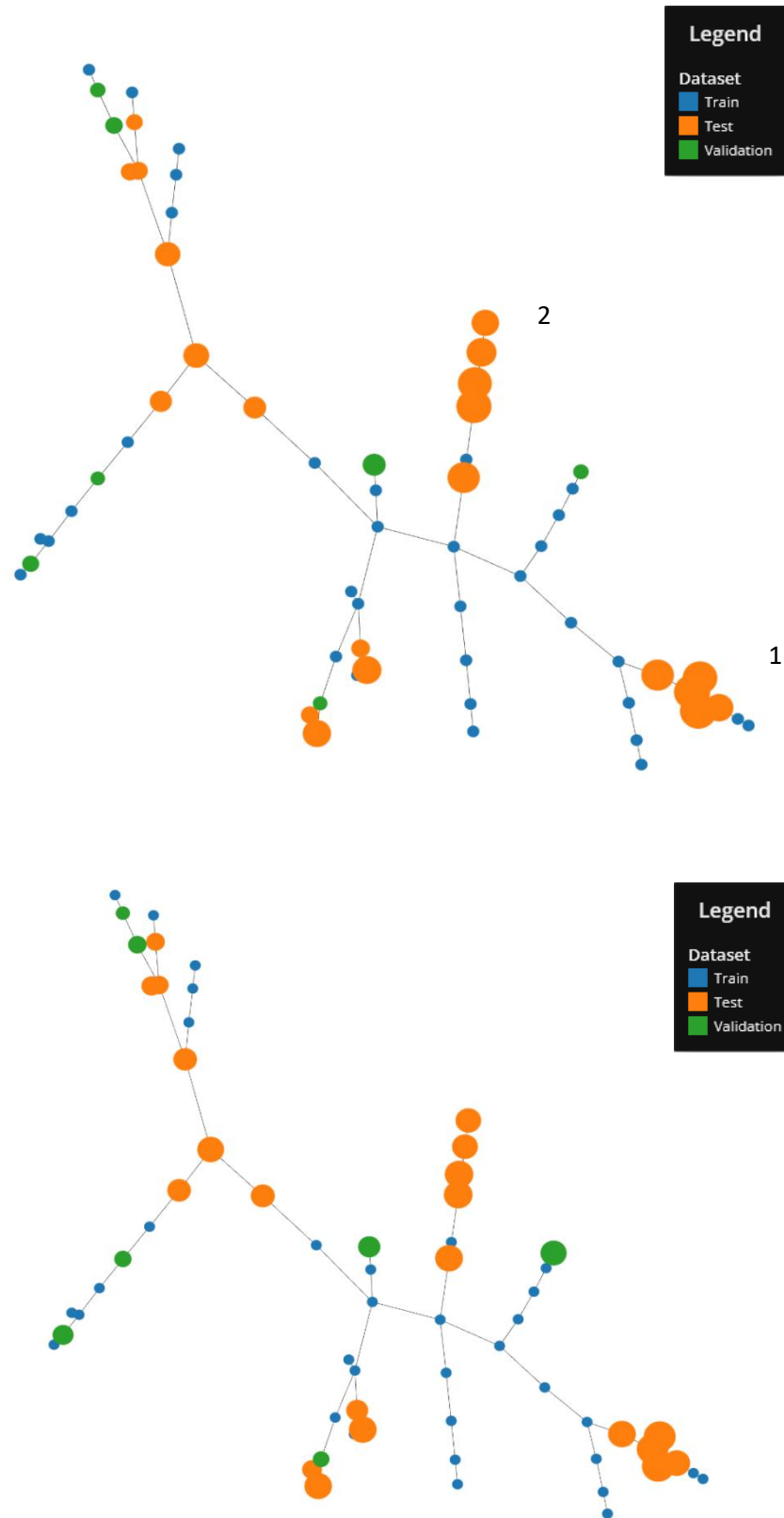


Figure 2.6: TMAP representations of the *fp_as_features* dataset sized by the distance from a perfect correct classification for a) PLS b) RF

2.3.2.6 MPO – Model performance

Again, as in the *fp_as_bits* work we calculate the pareto optimal model in terms of maximising the Balanced Accuracy, Kappa and H-Measure. There are 9 models in total that form the first pareto front. There are 3 PLS models, 4 SVM models and 2 RF models. Interestingly, the variation in performance seen in the *fp_as_bits* for the PLS models is seen here. PLS models account for the 2 optimal models but also is the least optimal model highlighting the variance in the performance of these models.

Again, the performance of the RF and SVM mirrors that seen in the *fp_as_bits* closely with them forming 7 of the 10 most optimal models. Unlike in the *fp_as_bits*, the NN models perform poorly here, with the best NN model forming part of the third pareto front.

2.3.2.7 Chemical similarity in misclassification

Figure 2.7 shows the molecules that are most misclassified by the *fp_as_features* models.

Interestingly, it is the same 3 molecules that are difficult for the models to predict. This may indicate that the feature fingerprint sets are very similar to the bit fingerprints and the models are failing to predict these molecules for similar reasons to the *fp_as_bits* models.

However, in these models, these points are misclassified 59, 53 and 43 times resulting in misclassification 84%, 76% and 61% of the time. Suggesting that the *fp_as_features* descriptors may do a poorer job of representing the models.

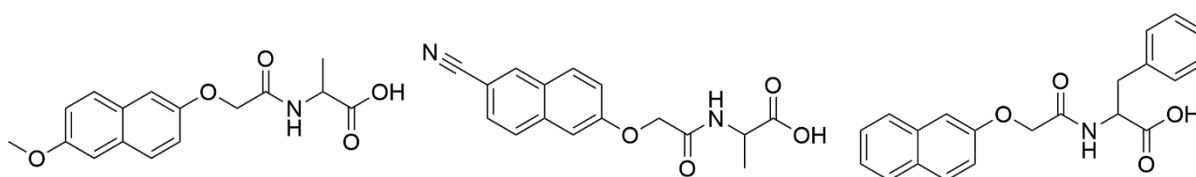


Figure 2.7 The 3 most misclassified molecules in the *fp_as_features* model set.

2.3.2.8 Y-Randomisation

Since the *fp_as_features* RF and PLS model show promising predictive performance, y-randomisation is again used as a tool to further increase our belief in the predictive ability of these models. It is clear from the results in Table 2.7 that the predictive performance of our y-randomised models is poor. This is further demonstrated by the H-Measure Z scores for the two models. For the RF models, the Z score is 1.52 giving 6.4% chance that the true model is drawn from the same distribution as the randomised

models. For the PLS model, this value decreases to 1.40 resulting in an 8.1% chance interval that the true model is a sample drawn from the same distribution as the y-randomised models.

Table 2.7: Performance of the y-randomised models on the in domain test set

Method	Kappa	Balanced Accuracy	H-Measure	Quality of Predictions
RF	-0.01 (± 0.27)	0.47 (± 0.24)	0.38 (± 0.21)	Bad
PLS	0.02 (± 0.31)	0.52 (± 0.24)	0.42 (± 0.25)	Bad

2.3.2.9 Variable Importance

Since it appears we have generated good models with the *fp_as_features* data, the benefit of this descriptor set can now be exploited with the investigation of variable importance and the mapping of fingerprint fragments back on to the molecules for rationalisation of the importance.

For model interpretability, we investigate the variable importance for the RF and PLS model as these models perform best overall. For each of the 10 models, for RF, we use the built-in `caret::varImp()` which returns the difference in prediction accuracy between the descriptors as inputted into the model and the descriptors after permutation (random shuffling of the features in a descriptor column) and average the score across all 10 models to get a general idea of variable importance.

For the PLS models, we carry out the same approach but `caret` defaults to the `filterVarImp()` (see 2.2.10.1 R Model Interpretation) function due to the lack of inherent importance measure. This outputs a ROC score between 0 and 1 based on how well the points can be classified into their correct classes based purely on a descriptor, with scores close to 1 showing good separation between classes for that descriptor. This means that the scores in the importance across models are not necessarily directly comparable, but the order of the most important descriptors may provide some insight.

For the SHAP variable importance, the SHAP values are calculated for the applicability domain filtered test and validation sets, 16 data points in total (12 test and 4 validation). Due to the different scale of the physicochemical descriptors, these have been scaled between 0 and 1 so that they are on the same scale as the fingerprint descriptors whilst retaining the relative size of the descriptor within the set. The values for the SHAP plot are directly comparable across modelling algorithms and will be compared to the R `caret` variable importance for any similarities.

As our dataset comprises both physicochemical and fingerprint descriptors (which can be mapped back onto a molecule), we can relate the relative importance of the R `caret` and SHAP importance to

experimental findings. To aid in this, the molecules within our dataset are generalised using the same composition as those in Fleming *et al.*³⁶ as being composed of four components (Figure 2.8), the aromatic N terminus group – in our case this is predominantly Fmoc or Naphthalene. The linker between the aromatic group and the peptide, the peptide (which itself is made up by the peptide backbone and the R groups of the amino acids) and any modification of the C terminus.

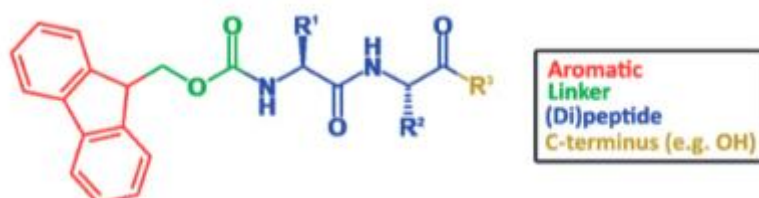


Figure 2.8: Composition of our molecules for variable importance mapping taken from [35]

The four regions have all been shown to play different roles within self-assembly of peptides, outlined in detail in [35]. These include increased π - π stacking of the aromatic region, the linker controlling planarity of the system and the peptide providing an amphiphilic region to drive self-assembly.

2.3.2.9.1 R Variable Importance

2.3.2.9.1.1 RF Importance

The results of the average variable importance during training for the 10 RF models is summarised in Figure 2.9. For the RF models, on average Molecular Solubility is the most important descriptor, suggesting that solubility plays a key role in differentiating between molecules that form gels and those that do not. Given that solubility is the driving force it is plausible that this is in fact an important descriptor, however, it is also a problem that molecular solubility is heavily correlated to a number of descriptors removed in the pre-processing of the data and could be acting as a surrogate descriptor for these fragments in the molecule.

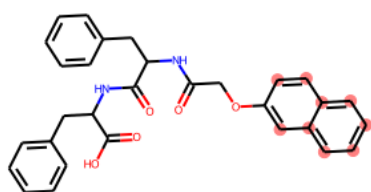
ALogP, Molecular_PolarSASA and the Num_Rings are the 2nd, 3rd and 5th most important descriptors overall according to this approach. Although this approach doesn't specify how important the value for each descriptor is, further analysis using the SHAP approach will allow us to draw conclusions about this.

In a similar vein, of the ECFP and FCFP fragments considered important by the RF models, the 1st, 2nd, 5th and 6th most important fingerprints out of the 6 present in the top 10 are the aromatic N terminal

moiety, the 3rd (6th overall most important) most important being the linker with the peptide backbone and the 4th (8th overall) being the side chain of a isoleucine amino acid.

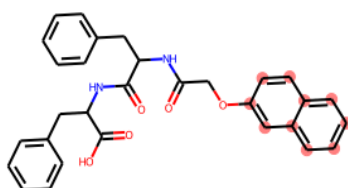
R variable importance for the Random Forest model

	Average Variable Importance
Molecular_Solubility	1.94
ALogP	1.78
Molecular_PolarSASA	0.76
CM.ECFP_4.717474525	0.47
Num_Rings	0.44
CM.ECFP_4..755462605	0.35
CM.ECFP_4..1194163736	0.34
CM.FCFP_4..1272798659	0.31
CM.ECFP_4..178525456	0.30
CM.ECFP_4..2019199918	0.30



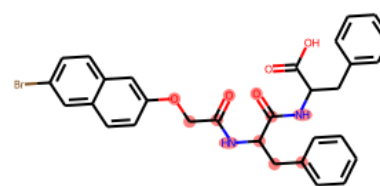
ECFP_4.717474525

RF Average Importance = 0.47



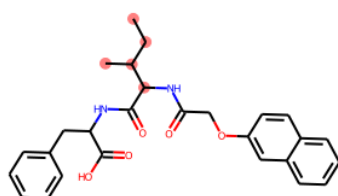
ECFP_4.-755462605

RF Average Importance = 0.35



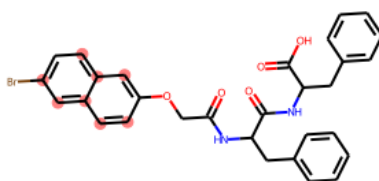
ECFP_4.-1194163736

RF Average Importance = 0.34



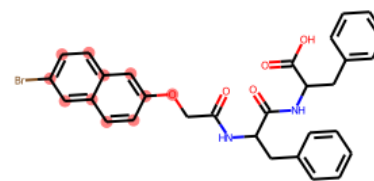
FCFP_4.1272798659

RF Average Importance = 0.32



ECFP_4.-178525456

RF Average Importance = 0.30



ECFP_4.-2019199918

RF Average Importance = 0.30

Figure 2.9: Mean Accuracy Decrease of each descriptor for our RF model with the corresponding fragment highlighted on a training set molecule where appropriate.

2.3.2.9.1.2 PLS importance

For the PLS important descriptors, in Figure 2.10, none of the physicochemical descriptors are in the most important 10 descriptors, whereas all 4 were within the top 5 of the RF caret importance. Of the 9 most important fragments shown in Figure 2.10, there is a wide range of the regions we defined in Figure 2.8 in our molecules represented in these fragments. The most important fragment is the phenyl ring of a phenylalanine amino acid. The side chain of an isoleucine is again present in the top fragments, being the 8th most important by the model. The linker between the aromatic region and the peptide is the second most important fragment from the ROC approach. The entire peptide makes up the fragments thought to be the 3rd, 4th, 5th and 9th most important by our PLS model, with the 6th and 7th most important fragments being examples of naphthalene aromatic moieties only.

Although we have gained a general idea about the importance both the RF and PLS models place on different types of molecule fragments, we still have no knowledge about how the values of these fragments (whether they are absent or present) affect the probability of gel formation (does the presence of a fragment increase or decrease the probability of gel formation?). The following SHAP analysis should provide some insights as to these relationships within our models.

R variable importance for the Partial Least Squares – Discriminant Analysis model

	Average Variable Importance
CM.ECFP_4_244977436	0.020
CM.ECFP_4_1205575950	0.020
CM.ECFP_4_1307307440	0.019
CM.ECFP_4_642810091	0.019
CM.ECFP_4_1380908375	0.019
CM.ECFP_4_1886441421	0.018
CM.ECFP_4_717474525	0.018
CM.FCFP_4_1043339860	0.018
CM.ECFP_4_944269100	0.018
CM.ECFP_4_1731135544	0.018

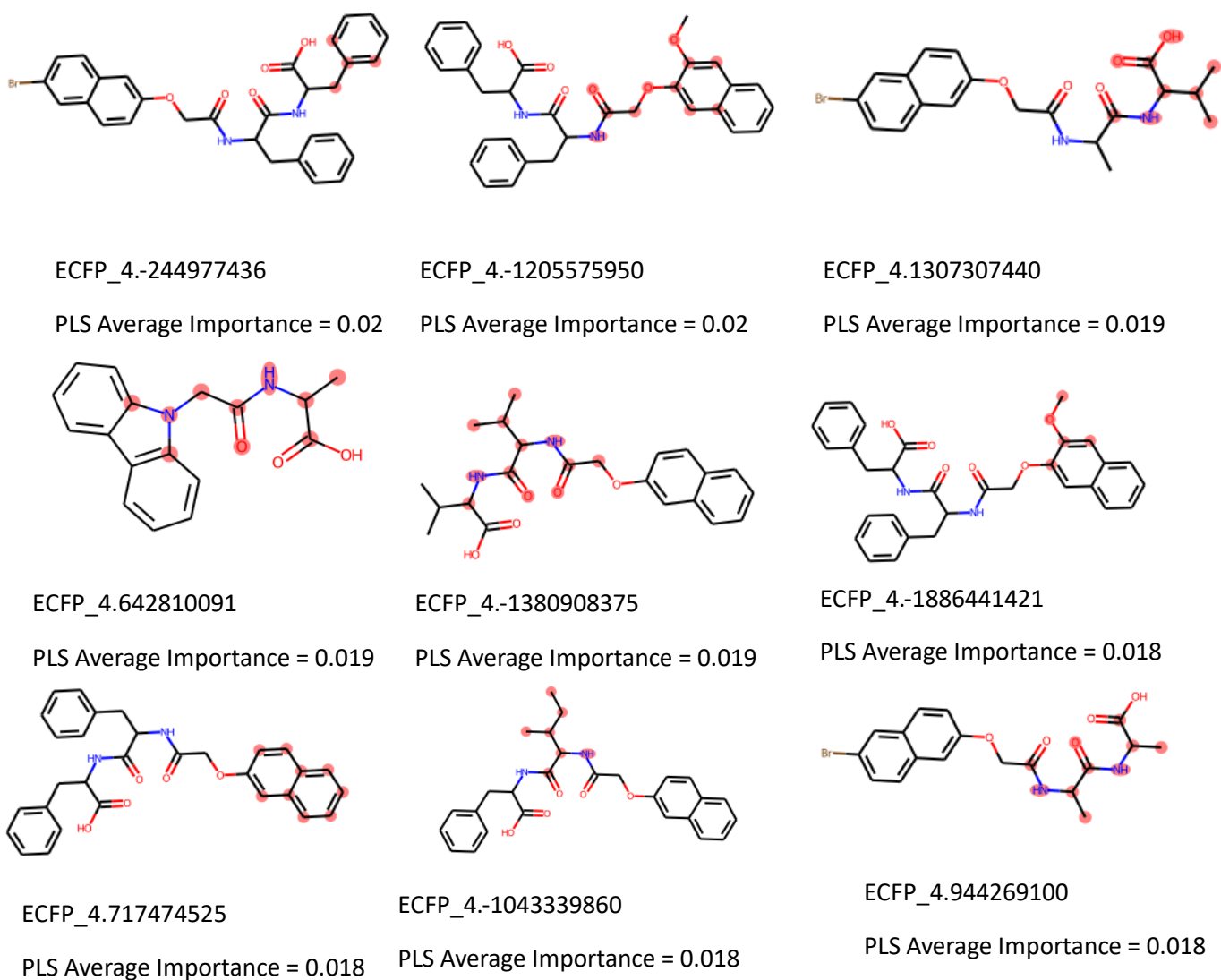


Figure 2.10: ROC filter score for the descriptors for our PLS model, again highlighted as red circles on a training set molecule where appropriate.

2.3.2.9.2 R SHAP Importance

2.3.2.9.2.1 RF Shap Importance

The beeswarm plot in Figure 2.11 illustrates the importance of each molecule in the combined test and validation set as a point on the graph, coloured by descriptor value. The position of each point on the beeswarm plot is the average SHAP value for that descriptor, for that molecule, across all coalitions that descriptor was present in. The rows are ordered by the absolute mean SHAP value for a descriptor, meaning that the most influential descriptors (regardless of whether the importance is positive or negative) are presented first. We take the top 10 descriptors and plot them in the beeswarm in Figure 2.11.

Again, Molecular Solubility is seen as the most important descriptor, followed by the Number of Rings. We can see from Figure 2.11 that there are clear value dependencies on the prediction effect, showing that increased molecular solubility results in a more negative SHAP attribution for the descriptor, meaning it has a negative effect on the probability of gel formation. This corroborates experimental findings that show that increased solubility of peptides reduces their propensity for aggregation.^{37,38}

For the number of rings, we see as the number of rings present in the molecule increases, the probability of forming a gel also increases. This is likely because ring systems, in particular aromatic rings are popular methods of introducing hydrophobicity into the system to increase hydrophobic interactions. For example, in Fmoc based gels, fluorescence spectra show significant packing of the aromatic Fmoc protecting group.³⁹

The only other physicochemical descriptor present in the top 10 most important is the molecular polar solvent accessible surface area, which shows that an increase in the surface area corresponds with a modest increase in gel probability. However, in all but one case the change in probability is still negative, suggesting that an increase in PolarSASA still results in a decrease in gel probability – meaning this descriptor does a poor job of differentiating between gels and non-gels.

SHAP variable importance for the Random Forest model

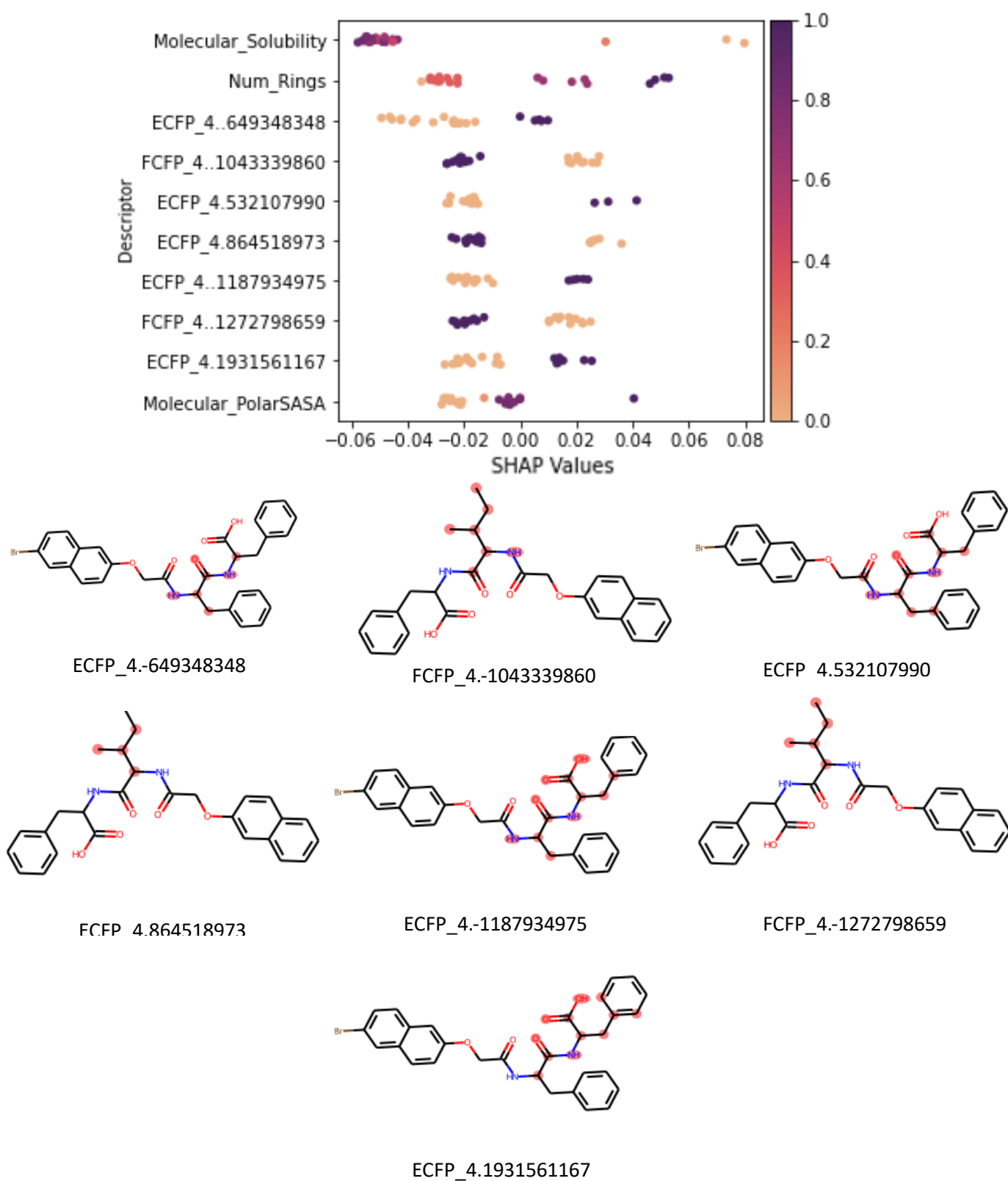


Figure 2.11 Beeswarm plot of the RF model and the corresponding molecular fragments highlighted as red circles on a training set molecule.

For the fingerprint fragments in the top 10, although in Figure 2.11 they look like very similar fragments, there is some variation within them. The most important fingerprint ECFP_4.-649348348 ([*]NC(=O)C([*])[*]) has a positive relationship between the presence of this fragment and gelation. This fragment is only present in molecules where in the Linker-AA-COOH arrangement, the first amino acid contains a side chain, such as phenylalanine (F), isoleucine (I) or valine (V). There are examples of Aromatic-Phe, Aromatic -Val, Aromatic -Ile and Aromatic -Leu (where the aromatic is either phenyl and naphthalene) containing just the one amino acid, in these cases the bit is set to 0, highlighting that the model is identifying the importance of the dipeptide arrangement in allowing gelation. Although there are reports^{40,41,42} of protected single amino acid gels, they are much less common than their dipeptide alternatives.⁴³⁻⁴⁵

The second most important fingerprint descriptor is FCFP_4.-1043339860 ([*]CC(C)C([*])[*]) and is set by molecules containing valine, leucine and isoleucine, regardless of whether they are as a single amino acid or a dipeptide. Interestingly, the presence of this fragment results in a decrease in the gel probability. This is somewhat surprising, as in work comparing the gelation state of Ile-Phe and Val-Phe based gels, the change of Val to Ile and the introduction of the extra methyl group promotes gelation in Ile-Phe whereas Val-Phe fails to gel. This is posited to be caused by an increased hydrophobicity in the amino acid side chain promoting hydrophobic interactions that drive self-assembly.⁴⁶

However, our datasets contain diphenylalanine (FF) based gels, and the 11 molecules within the entire training, test and validation sets with the highest gel probability are those that contain an FF moiety. This could mean that the presence of the valine, leucine and isoleucine decreases the gel probability not because they are bad for gelation but that they are worse than phenylalanine, which the model correctly considers an important fragment for gelation, as it has been shown that phenylalanine can self-assemble as a functionalised amino acid.⁴⁷

The remaining fragments in the SHAP importance are slight modifications of the first two fragments in the plot, they either explicitly include the phenyl ring in the fragment for phenylalanine and show a positive correlation between the presence of the fragment and the SHAP value or could be set by a valine, leucine or isoleucine with the same negative correlation between presence and SHAP value. It is interesting that the model interpretation is able to highlight the importance of the phenylalanine amino acid for self-assembly, corroborating experimental findings.

2.3.2.9.2.2 PLS Shap Importance

For our PLS models (Figure 2.12), the most important molecular fragment is again the ECFP_4-649348348 ([*]NC(=O)C[*][*]) which is the most important fragment overall. It follows the same relationship between descriptor value and SHAP value, showing that the PLS model highlights the same importance of the dipeptides. Molecular solubility is the second most important descriptor and again, the relationship is the same here for the RF SHAP.

Although there are examples of isoleucine (ECFP_4.-742649778) and phenylalanine (ECFP_4.532107990) and these show the same relationships as the RF SHAP, surprisingly there is a heavy importance placed on the aromatic linker with 4 examples of naphthalene-based fragments in the most important 10. The aromatic moiety is introduced to provide the hydrophobic region in the molecule to increase hydrophobic interactions and increase π - π stacking and naphthalene is widely used to promote gelation^{45,48,49}.

However, the PLS model places a relatively high negative SHAP on the presence of this fragment. This is a surprise as the PLS model suggesting that the presence of the fragment is decreasing the probability of a gel forming even though it has been thought experimentally to be an excellent group at promoting self-assembly. This may make the RF model more appealing for future deployment as interpretation of the model predictions corroborate more closely with experiment.

SHAP variable importance for the Partial Least Squares – Discriminant Analysis model

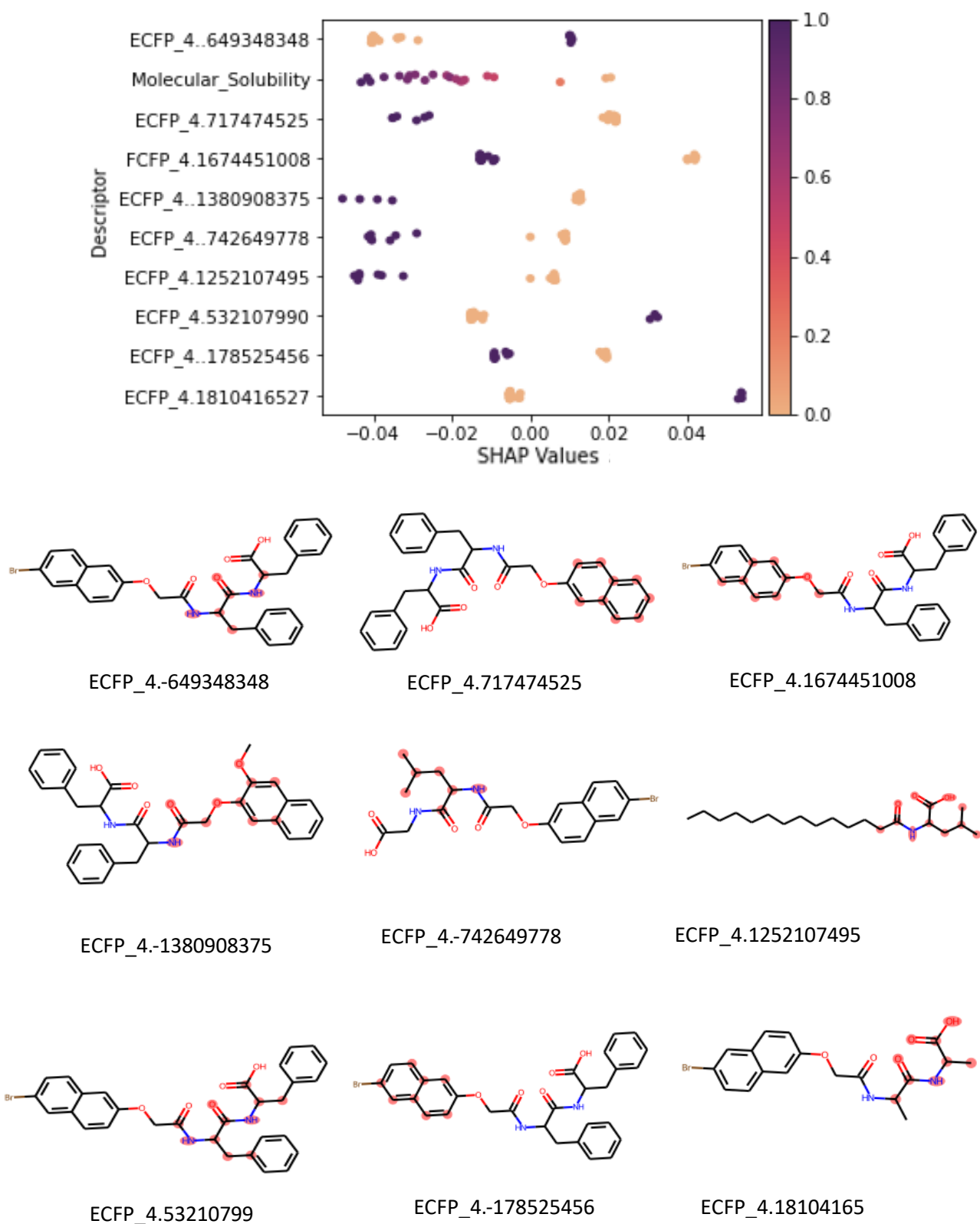


Figure 2.12: Beeswarm plot of the PLS model and the corresponding molecular fragments highlighted on a training set molecule.

2.3.3 Python Classification Models

Later in this work, when it became clear (Chapter 5) that we require a classifier in Python to interact with the work introduced there, we wished to use the RDKit functionality that allows mapping of fingerprint bits back onto the molecule to allow for direct comparison with the Gupta *et al.* models but with descriptors and modelling methods from other software packages and to allow for model interpretation that is comparable with the *fp_as_features* models.

However, given that across all the descriptor sets investigated thus far, the RF consistently performed well, and from the results of the model interpretation, we decided to focus on learning a RF in sci-kit learn²⁴ (*version 0.23.2*) as it also allows for the direct Gini importance variable importance metric to compare with the SHAP approach.

2.3.3.1 Comparison with *fp_as_bits* Pipeline Pilot descriptors

After descriptor calculation, there were 4100 descriptors (4 physicochemical, 2048 ECFP and 2048 FCFP) and after descriptor preprocessing there were 58 descriptors (*see Appendix 2.6.4 for a list of descriptors*), including Number of Rings, AlogP and Molecular Solubility. The remaining 55 descriptors were 21 ECFP and 34 FCFP fragments. Work has been carried out comparing the implementation of the fingerprint descriptor algorithms between Pipeline Pilot and RDKit showing that they are generally very similar with difference being “mostly aromatic” (Figure 2.13).⁵⁰ However, when comparing the *python_classifier* fingerprints to the *fp_as_bits* Pipeline Pilot descriptors, there were 25 ECFP and 25 FCFP fragments in the *fp_as_bits* data creating variation within our datasets.

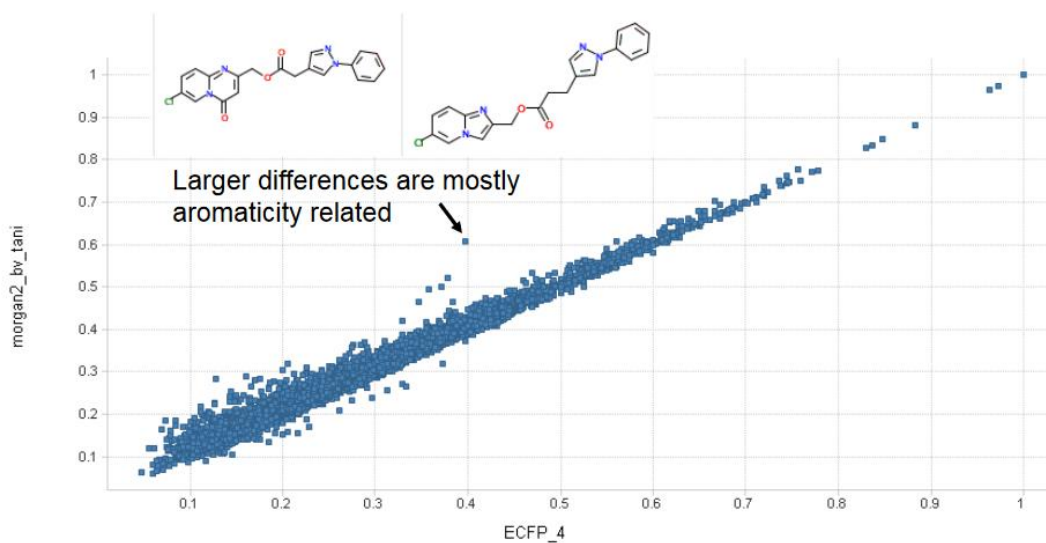


Figure 2.13: Difference between the Pipeline Pilot ECFP (ECFP_4) and the analogous RDKit implementation (morgan2) Taken from [47]

2.3.3.2 Descriptor Visualisation

The first two principal components only explain 11.55% and 8.85% (20.40% in total) of the variance in the dataset meaning that the PCA plot in Figure 2.14a may not fully capture the descriptor set. However, with this, the PCA plot shows good coverage of the PCA space by the training set and so we would expect good domain coverage of the test and validation sets, resulting in the test and validation sets being encompassed by the training set points if they were to create a 2D bounding box in this PCA space.

In the TMAP plot (Figure 2.14b) there are 6 of the 21 data points that are terminal points of a sub-branch. Again, there is a sub-branch with 4 terminal test set point but again they are not the same 4 points as those in the corresponding *fp_as_bits* and *fp_as_features* plots (Figures 2.1b and 2.4b).

PCA and TMAP plots of the *Python_Classifier* dataset

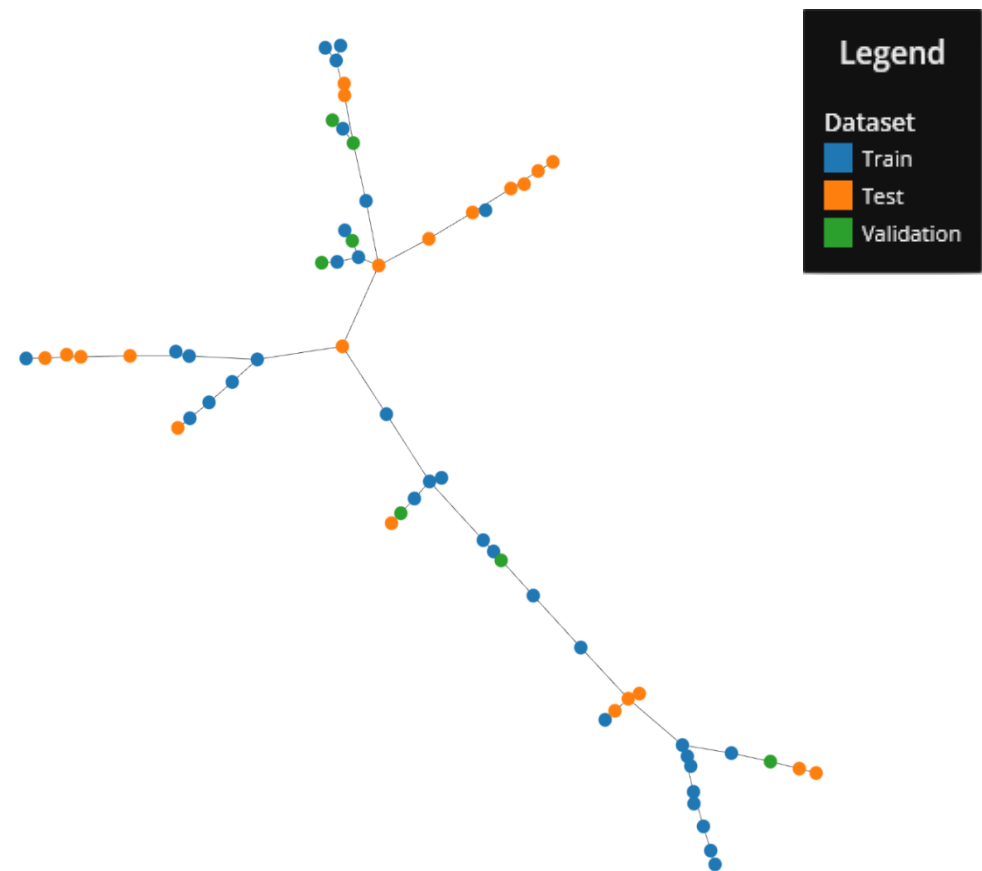
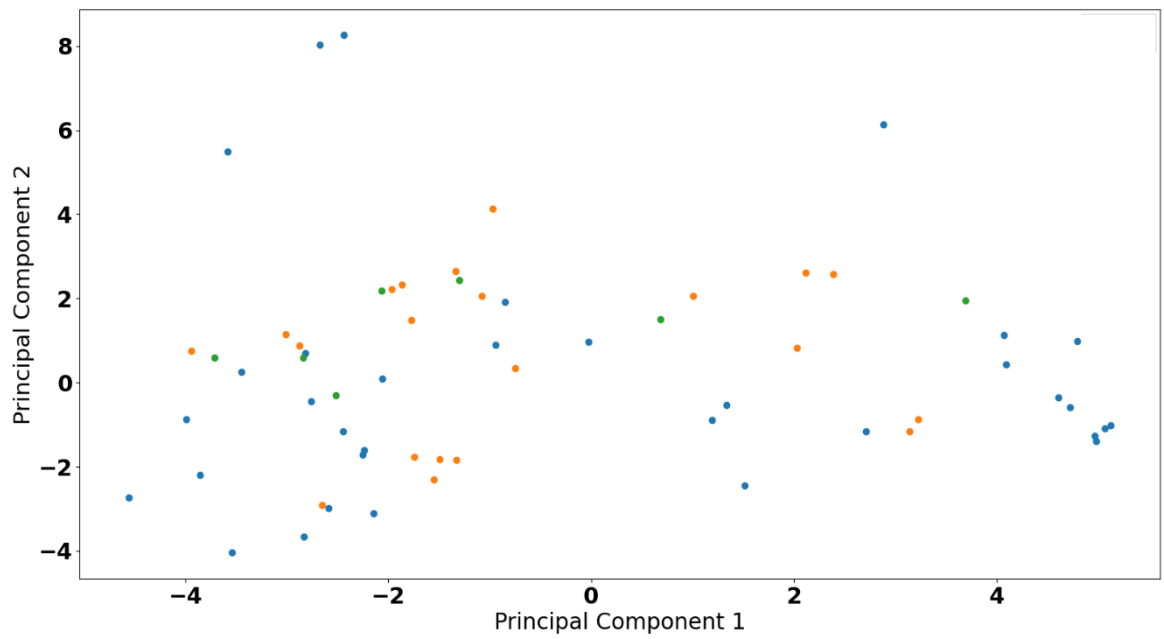


Figure 2.14 a) Visualisation of the *Python_Classifier* dataset in PCA space and b) TMAP visualisation of the *Python_Classifier* space

2.3.3.3 Training set performance

We can see good training performance for our Python classifier (Table 2.8), in fact we see decreased performance than the *fp_as_bits* R model across all metrics. For Kappa, the score decreases from 1.00 to 0.94 here and we see similar decreases for Balanced Accuracy (1.00 to 0.97) and H-Measure (1.00 to 0.92). Also, there is comparable performance to the *fp_as_features* models across kappa (0.99 vs 0.94 here), Balanced accuracy (0.99 vs 0.97 here) and H-Measure (1.00 vs 0.92) here.

Table 2.8: Training set performance of our sklearn RF model using the python_classifier descriptors

Method	Kappa	Balanced Accuracy	H-Measure	Quality of Predictions
RF	0.94	0.97	0.92	Good

2.3.3.4 Test and validation set performance

Again, we implemented an applicability domain filter for our test and validation set points based on the range check of the descriptors and first 10 principal components for direct comparison with the work in Gupta et al¹² and our *fp_as_bits* work. Filtering of the test set reduces the dataset from 21 molecules to 14 molecules (4 gels and 10 non-gels) and for the validation set it reduces to just 4 (all non-gels). Predictions of our RF model on the reduced test set are shown in Table 2.9. Comparing our Python_Classifier performance to the *fp_as_bits* models, we see the same kappa (0.57 v 0.57) and modest increases in performance for balanced accuracy (0.85 v 0.79) and H-measure (0.74 v 0.70).

Compared to the Gupta models, we again find slightly decreased performance across the kappa (0.57 v 0.76), balanced accuracy (0.85 v 0.96) and H-measure (0.74 v 1) which is similar to what we found for our *fp_as_features* models. Comparing with the *fp_as_features* data shows very similar performance across the kappa (0.57 v 0.56), balanced accuracy (0.85 v 0.82) and H-measure (0.74 v 0.70). Since the in-domain validation set contains no gels, we instead predict on the full validation set and see perfect classification on the 7 molecules.

Table 2.9: Performance of the test set on the range-check filtered test set

Method	Kappa	Balanced Accuracy	H-Measure	Quality of Predictions
RF	0.57	0.85	0.74	Good

2.3.3.5 Chemical Similarity

In total, the Random Forest misclassifies 3 molecules in the in-domain test set. When comparing to the most misclassified points in the R models, it is only molecule 2 that is again misclassified in our python models. This adds further evidence to this molecule being difficult for models to predict based on the training set used to train the model. However, the Python model does correctly predict the difficult to predict molecules in the R models hinting that the python model is making predictions based on fragments that can differentiate between gel and non-gel for these difficult to predict molecules.

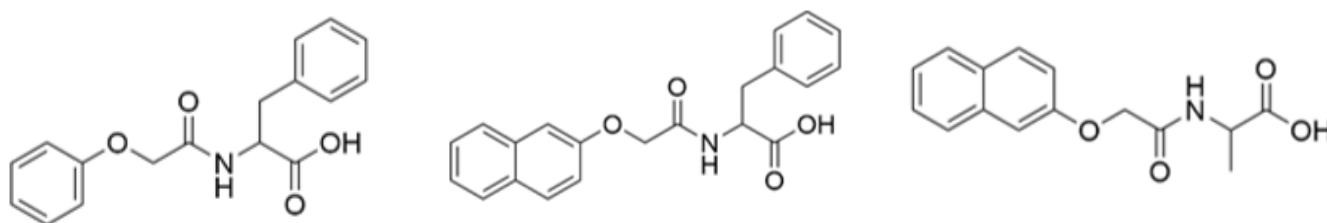


Figure 2.15: Molecules misclassified by the Python_Classifier Random Forest model

2.3.3.6 Y-randomisation

For our Python classifier models, we take an extra step in our Y-randomisation and carry out a calculation of the Z score for the H-Measure statistic for 50 randomised models and compare it to the value for our true model.

The average performance of the 50 y-randomised models for the H-Measure can be found in Figure 2.16. The distribution shown in Figure 2.16 is the range of H-Measure scores obtained from the 50 y-randomised models on the in-domain test set. From the figure we see a drastic difference in the predictive ability of the model on the in-domain test set. This is encouraging given that the labels have been shuffled and so the features are no longer reflective of the gel state they correspond to. This gives greater weight to the idea that the relationships found between the descriptors and gel label in the true models is capturing real relationships governing gel formation.

We can quantify the difference in prediction on the in-domain test set between the true model (solid vertical line in Figure 2.16) and the distribution of random model H-measure values through the Z score. We calculate the Z score between the true in-domain H-measure and the average and standard deviation of the y-rand H-Measure as 3.50 which results in 0.03% chance that our true model is a model drawn from the y-randomised model distribution.

Distribution of the H-Measure statistic on the y-randomised models

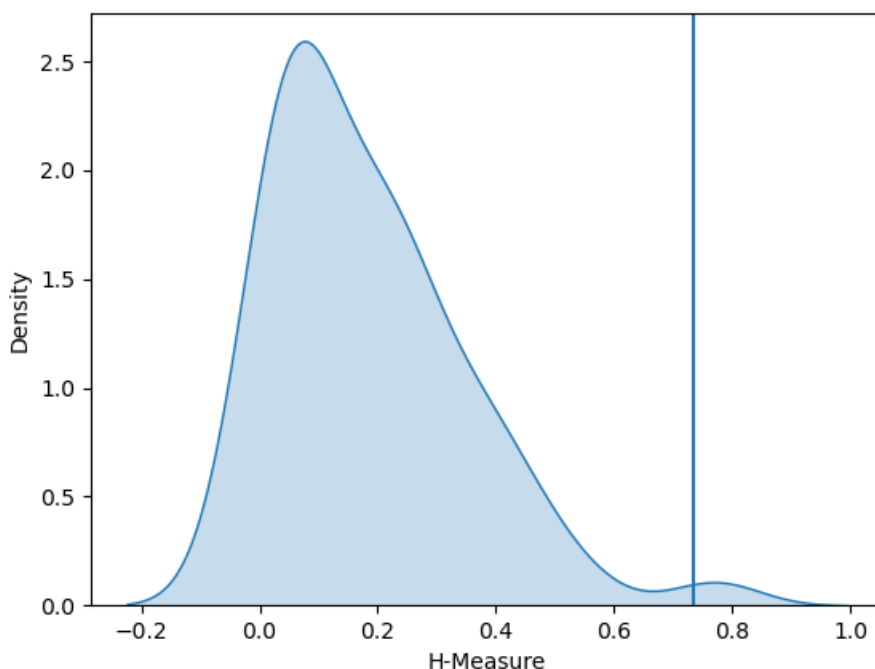


Figure 2.16: Distribution of H-Measure values on the in-domain test set for our 50 y-randomised models. The H-Measure of our true model on the in-domain test set is shown as a solid vertical line.

2.3.3.7 Variable importance

2.3.3.7.1 RF importance

Since it appears we have successfully created a predictive classifier in python for our dataset we can interpret the model and compare the importance to those calculated for our R models. For variable importance investigations of our model, we again turn to the in-built importance measure within RF and a Shap approach within Python. For the Python RF, importance is calculated differently than in R using the mean and standard deviation of the total impurity decrease within each tree for that descriptor. This results in descriptors with high variable importance being more preferential splits in the tree as they result in more homogenous labels in the child node. It is worth noting that the absolute values are not comparable to the R models, but the ordering will be compared.

From the variable importance (Figure 2.17), it appears that the python model interestingly places very similar importance to structurally similar fragments to the R models. We again see molecular solubility and number of rings as important descriptors. For the fragment descriptors, there is again a large proportion of similar fragments to the R variable importance with the phenylalanine fragments and the isoleucine, leucine and valine setting fragments also seen to be important.

Gini importance of the Python Random Forest model

Descriptor	Variable Importance
mol_solu	0.07
num_rings	0.06
FCFP_1566	0.05
FCFP_1668	0.05
FCFP_792	0.05
ECFP_2016	0.04
alosp	0.04
FCFP_468	0.04
ECFP_283	0.04
FCFP_1085	0.03

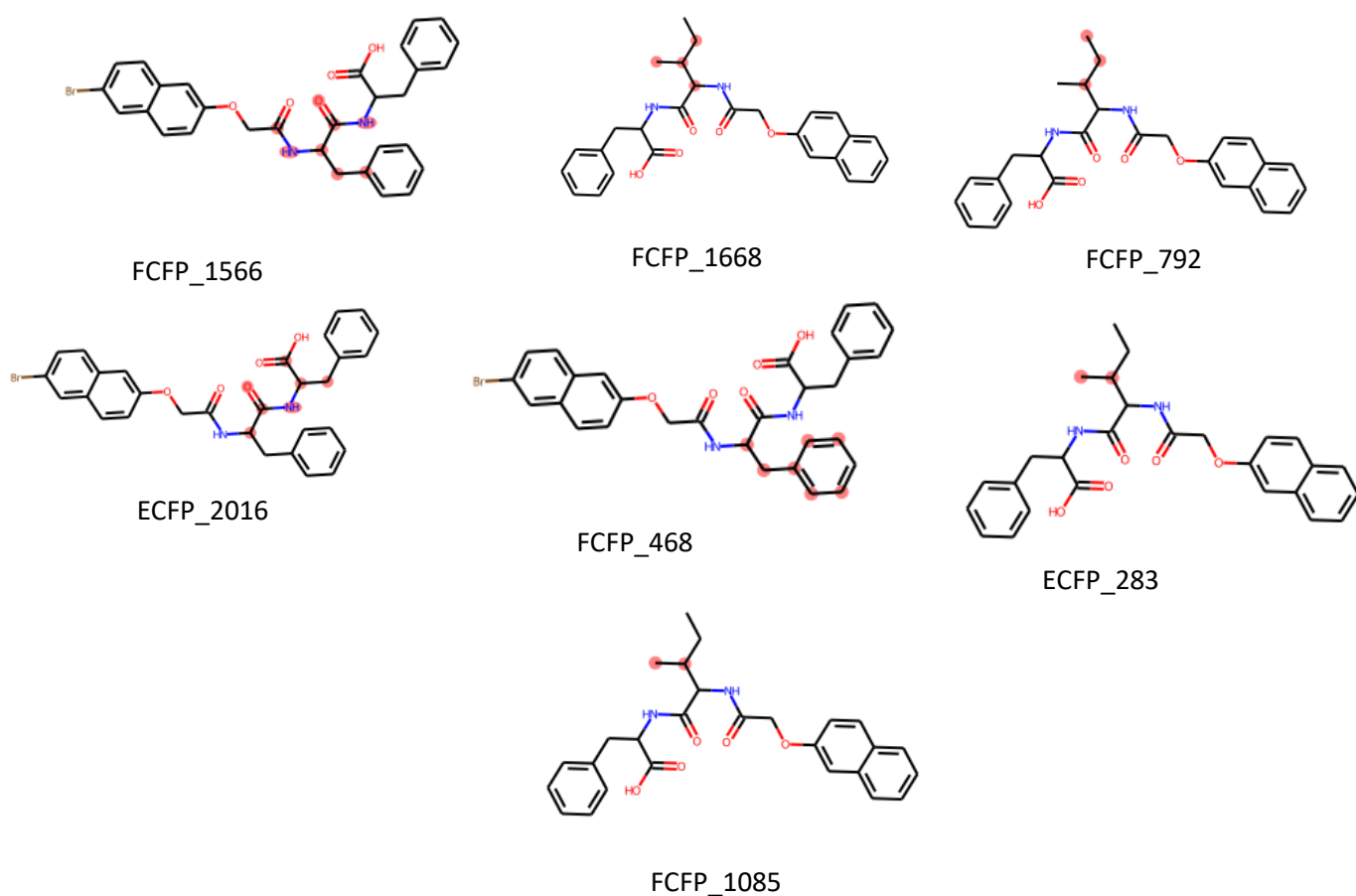


Figure 2.17: Mean Gini decrease ranked descriptors for our Python Classifier RF model

2.3.3.7.2 SHAP importance

Again, just as in the R models Molecular Solubility is an important descriptor in Figure 2.18, reaffirming our belief in how the balance between gelation and solution is one of the most important factors determining gel formation. Num_Rings, also, is a very important descriptor, illustrating that these are important in molecular self-assembly and thus gel formation. For the fingerprint fragments, we will utilise the Shap approach to determine the nature of the importance of the descriptors and determine if the Python models follow the same relationships as the R models.

Again, the Shap interpretation in Figure 2.18 shows remarkably similar conclusions to the R RF model. Here, there is again a high emphasis on the phenylalanine side chain and its positive impact on gelation and the potential negative impact of replacing the phenylalanine with leucine, valine or isoleucine resulting in a reduction in gel probability. It is reassuring that across descriptor sets and software packages not only does it appear, from the types of fragments present in the variable importance measures, that both datasets contain very similar fragments, and the same types of fragments are important in making accurate predictions.

SHAP importance of the Random Forest Python model

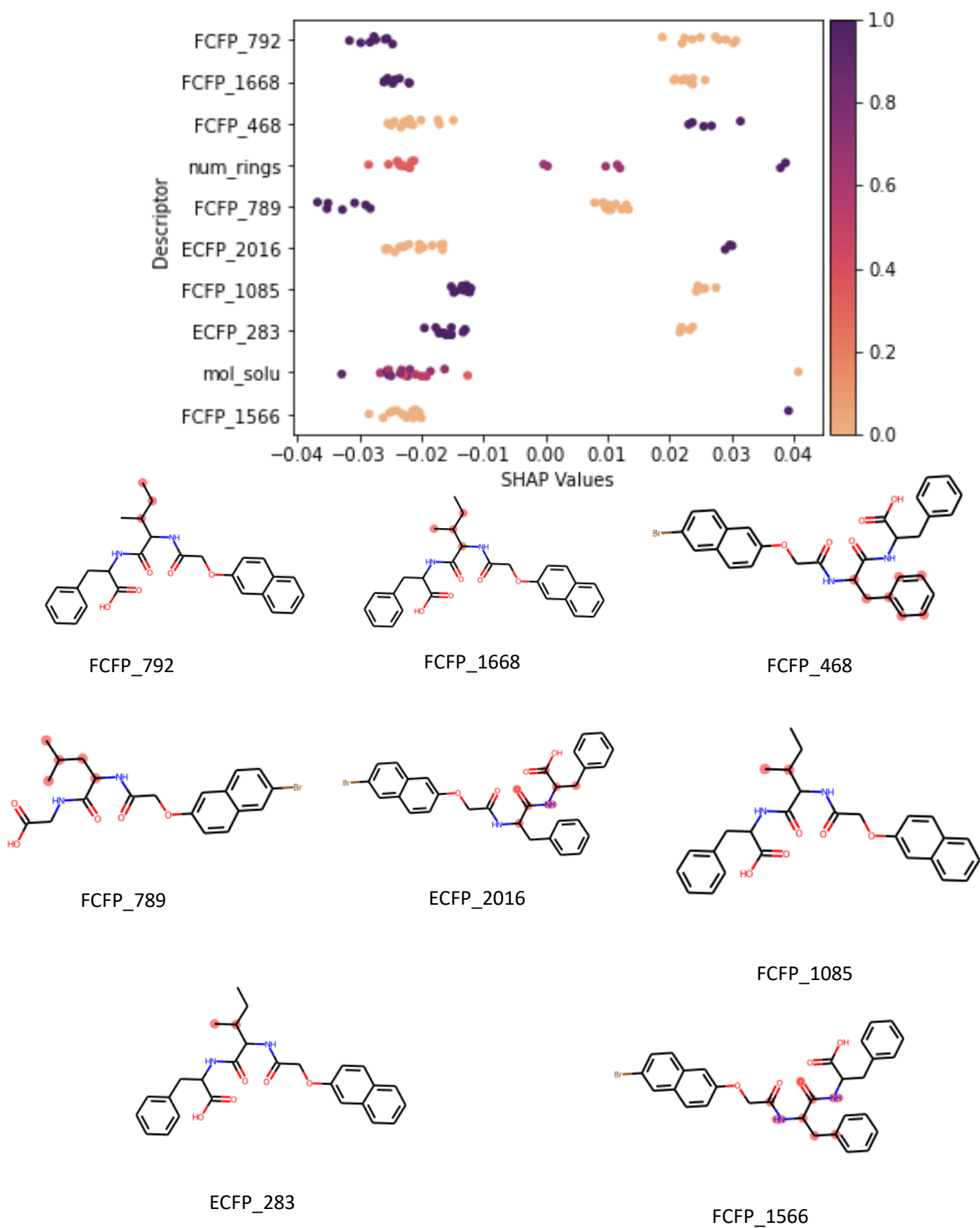


Figure 2.18: Beeswarm plot showing how the descriptor value in our Python_Classifier dataset impacts the SHAP value.

2.4 Conclusions

We have successfully replicated the models with similar performance to those in the literature. Importantly, we have built models where descriptor can be interpreted and that, for Random Forest and PLS, show excellent performance across Kappa, Balanced Accuracy and H-Measure performance on the test set and the in-domain test set points.

Moreover, we have investigated the model interpretation using both caret and SHAP methods and found trends, such as the importance of phenylalanine, that are consistent with those expected for the RF models but contradictory trends in our PLS, such as the negative correlation for the presence of naphthalene, reducing our confidence in this model. Our Python RF model shows comparable performance to the fp_as_features RF models and again, on fingerprints as a 2048 bit string, shows similar trends to the fp_as_features model.

Moving forward, we have a range of robust and interpretable models to predict whether a molecule will form a gel, work will continue into building regression models to predict important properties of gels, such as the storage and loss moduli of peptide-based gels (Chapters 3 and 4) to work in tandem with the models introduced here to build generative models with desired property profiles (Chapter 5).

2.5 References

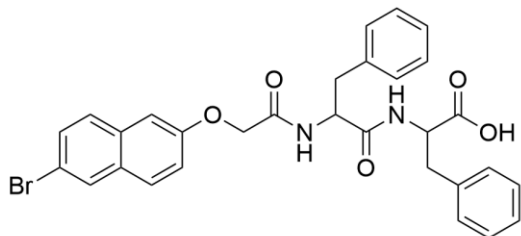
- 1 B. Xing, C. W. Yu, K. H. Chow, P. L. Ho, D. Fu and B. Xu, *J. Am. Chem. Soc.*, 2002, **124**, 14846–14847.
- 2 P. K. Vemula, G. A. Cruikshank, J. M. Karp and G. John, *Biomaterials*, 2009, **30**, 383–393.
- 3 J. J. Schmidt, J. Rowley and J. K. Hyun, *J. Biomed. Mater. Res. - Part A*, 2008, **87**, 1113–1122.
- 4 D. S. W. Benoit, M. P. Schwartz, A. R. Durney and K. S. Anseth, *Nat. Mater.*, 2008, **7**, 816–823.
- 5 S. Sundelacruz and D. L. Kaplan, *Semin. Cell Dev. Biol.*, 2009, **20**, 646–655.
- 6 M. C. Nolan, A. M. Fuentes Caparrós, B. Dietrich, M. Barrow, E. R. Cross, M. Bleuel, S. M. King and D. J. Adams, *Soft Matter*, 2017, **13**, 8426–8432.
- 7 C. B. Highley, C. B. Rodell and J. A. Burdick, *Adv. Mater.*, 2015, **27**, 5075–5079.
- 8 G. K. Veits, K. K. Carter, S. J. Cox and A. J. McNeil, *J. Am. Chem. Soc.*, 2016, **138**, 12228–12233.
- 9 B. O. Okesola and D. K. Smith, *Chem. Soc. Rev.*, 2016, **45**, 4226–4251.
- 10 M. Mohar and T. Das, *Soft Matter*, 2019, **17**, 328–341.
- 11 R. G. Weiss, *J. Am. Chem. Soc.*, 2014, **136**, 7519–7530.
- 12 J. K. Gupta, D. J. Adams and N. G. Berry, *Chem. Sci.*, 2016, **7**, 4713–4719.
- 13 R. Van Lommel, J. Zhao, W. M. De Borggraeve, F. De Proft and M. Alonso, *Chem. Sci.*, 2020, **11**, 4226–4238.
- 14 I. P. Moreira, G. G. Scott, R. V. Ulijn and T. Tuttle, *Mol. Phys.*, 2019, **117**, 1151–1163.
- 15 D. J. Adams, M. F. Butler, W. J. Frith, M. Kirkland, L. Mullen and P. Sanderson, *Soft Matter*, 2009, **5**, 1856–1862.
- 16 S. M. Abd Elrahman and A. Abraham, *J. Netw. Innov. Comput.*, 2013, **1**, 332–340.
- 17 BIOVIA, Dassault Systèmes, Pipeline Pilot, version 16.5.0.143, San Diego: Dassault Systèmes, 2017.
- 18 M. Kuhn, J. Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, the R. C. Team, M. Benesty, R. Lescarbeau, A. Ziem and L. Scrucca., *R Packag. version*, 2015, 6.0, 81.
- 19 G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*, CreateSpace, Scotts Valley, CA, 2009.
- 20 RDKit: Open-source cheminformatics. <https://www.rdkit.org>
- 21 W. McKinney, in *{P}roceedings of the 9th {P}ython in {S}cience {C}onference*, eds. S. van der Walt and J. Millman, 2010, pp. 56–61.
- 22 Data structures for statistical computing in python, McKinney, Proceedings of the 9th Python in Science Conference, Volume 445, 2010.
- 23 C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W.

- Weckesser, H. Abbasi, C. Gohlke and T. E. Oliphant, *Nature*, 2020, **585**, 357–362.
- 24 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, *J. Mach. Learn. Res.*, 2011, **12**, 2825–2830.
- 25 J. D. Hunter, *Comput. Sci. Eng.*, 2007, **9**, 90–95.
- 26 M. L. Waskom, *J. Open Source Softw.*, 2021, **6**, 3021.
- 27 P. Ertl, B. Rohde and P. Selzer, *J. Med. Chem.*, 2000, **43**, 3714–3717.
- 28 A. Shrake and J. A. Rupley, *J. Mol. Biol.*, 1973, **79**, 361–371.
- 29 I. V Tetko, V. Y. Tanchuk, T. N. Kasheva and A. E. P. Villa, *J. Chem. Inf. Comput. Sci.*, 2001, **41**, 1488–1493.
- 30 D. Rogers and M. Hahn, *J. Chem. Inf. Model.*, 2010, **50**, 742–754.
- 31 R. C. Braga, V. M. Alves, E. N. Muratov, J. Strickland, N. Kleinstreuer, A. Tropsha and C. H. Andrade, *J. Chem. Inf. Model.*, 2017, **57**, 1013–1017.
- 32 S. Zheng, Y. Wang, H. Liu, W. Chang, Y. Xu and F. Lin, *Chem. Res. Toxicol.*, 2019, **32**, 1014–1026.
- 33 J. S. Delaney*, *J. Chem. Inf. Comput. Sci.*, 2004, **44**, 1000–1005.
- 34 M. Kuhn, J. Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, the R. C. Team, M. Benesty, R. Lescarbeau, A. Ziem and L. Scrucca., *R Packag. version*, 2015, 6.0, 81.
- 35 S. M. Lundberg and S. I. Lee, *Adv. Neural Inf. Process. Syst.*, 2017, **2017-Decem**, 4766–4775.
- 36 S. Fleming and R. V. Ulijn, *Chem. Soc. Rev.*, 2014, **43**, 8150–8177.
- 37 M. Cao, C. Cao, L. Zhang, D. Xia and H. Xu, *J. Colloid Interface Sci.*, 2013, **407**, 287–295.
- 38 B. Sarkar, L. E. R. O'leary and J. D. Hartgerink, *J. Am. Chem. Soc.*, 2014, **136**, 14417–14424.
- 39 E. R. Draper, K. L. Morris, M. A. Little, J. Raeburn, C. Colquhoun, E. R. Cross, T. O. McDonald, L. C. Serpell and D. J. Adams, *CrystEngComm*, 2015, **17**, 8047–8057.
- 40 Z. Yang and B. Xu, *Chem. Commun.*, 2004, **1**, 2424–2425.
- 41 S. Misra, S. Mukherjee, A. Ghosh, P. Singh, S. Mondal, D. Ray, G. Bhattacharya, D. Ganguly, A. Ghosh, V. K. Aswal, A. K. Mahapatra, B. Satpati and J. Nanda, *Chem. - A Eur. J.*, 2021, **27**, 16744–16753.
- 42 W. J. Frith, A. M. Donald, D. J. Adams and A. Aufderhorst-Roberts, *J. Nonnewton. Fluid Mech.*, 2015, **222**, 104–111.
- 43 L. Chen, G. Pont, K. Morris, G. Lotze, A. Squires, L. C. Serpell and D. J. Adams, *Chem. Commun.*, 2011, **47**, 12071–12073.
- 44 L. Chen, S. Revel, K. Morris, L. C. Serpell and D. J. Adams, *Langmuir*, 2010, **26**, 13466–13471.
- 45 H. Shao and J. R. Parquette, *Chem. Commun.*, 2010, **46**, 4285–4287.
- 46 N. S. De Groot, T. Parella, F. X. Aviles, J. Vendrell and S. Ventura, *Biophys. J.*, 2007, **92**, 1732–1741.
- 47 S. Perween, B. Chandanshive, H. C. Kotamarthi and D. Khushalani, *Soft Matter*, 2013, **9**,

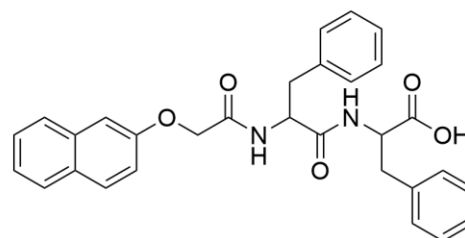
- 10141–10145.
- 48 S. Awhida, E. R. Draper, T. O. McDonald and D. J. Adams, *J. Colloid Interface Sci.*, 2015, **455**, 24–31.
- 49 S. S. Babu, V. K. Praveen and A. Ajayaghosh, *Chem. Rev.*, 2014, **114**, 1973–2129.
- 50 G. Landrum, *RDKit UGM 2012 Fingerprints RDKit*, 2012, 1–23.

2.6 Appendix

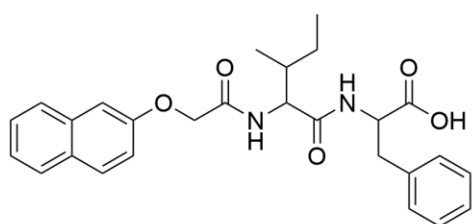
2.6.1 Molecules in this dataset



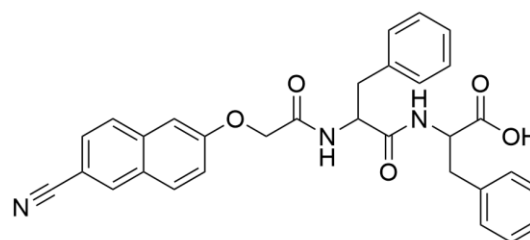
training1 - Gel



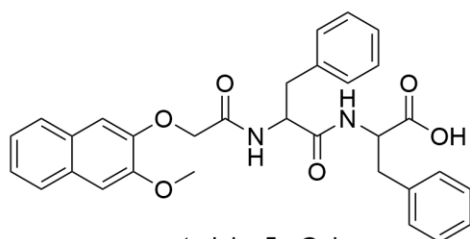
training2 - Gel



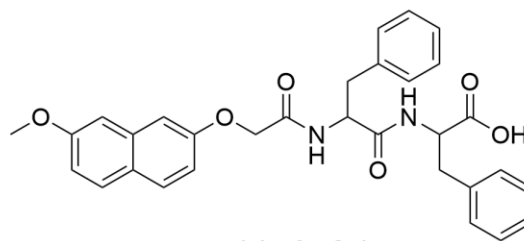
training3 - Gel



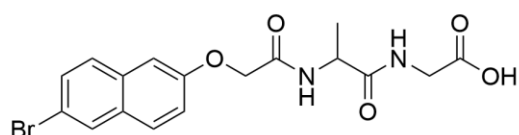
training4 - Gel



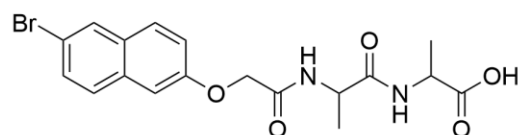
training5 - Gel



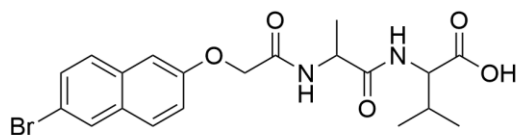
training6 - Gel



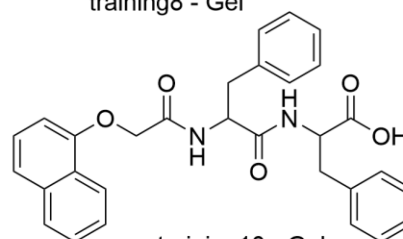
training7 - Gel



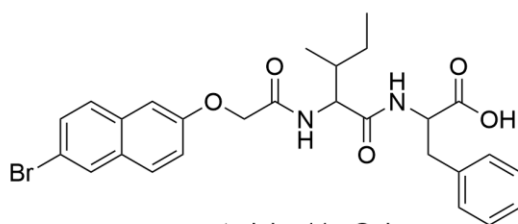
training8 - Gel



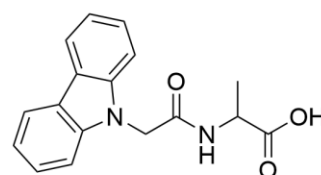
training9 - Gel



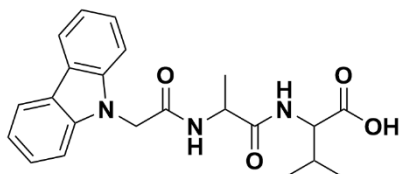
training10 - Gel



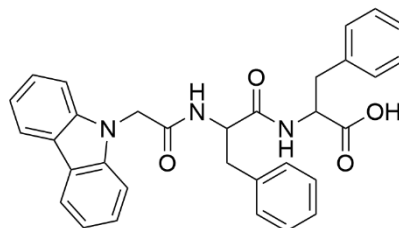
training11 - Gel



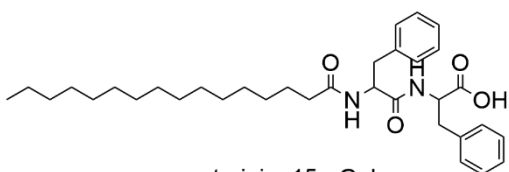
training12 - Gel



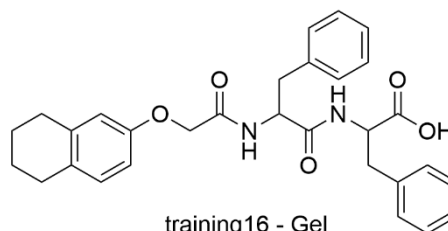
training13 - Gel



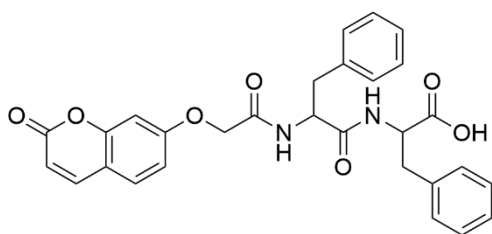
training14 - Gel



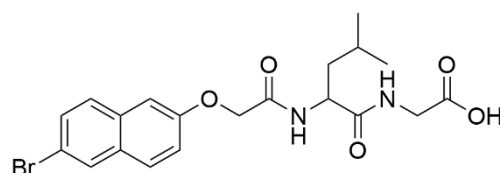
training15 - Gel



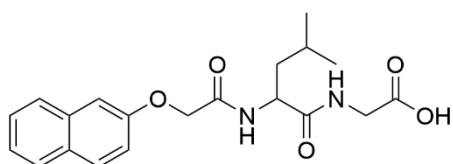
training16 - Gel



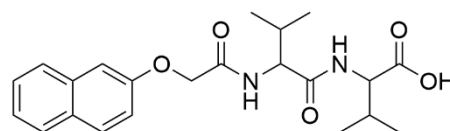
training17 - Gel



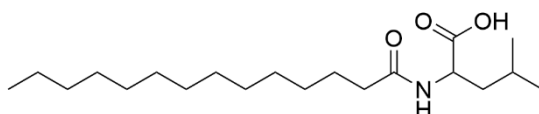
training18 - Non-Gel



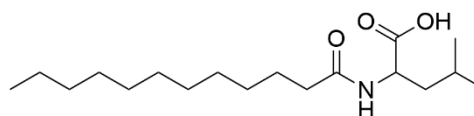
training19 - Non-Gel



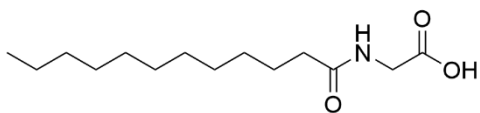
training20 - Non-Gel



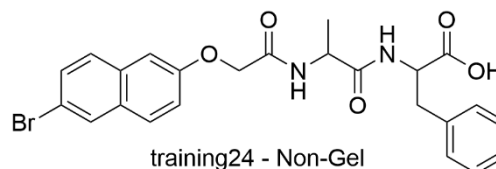
training21 - Non-Gel



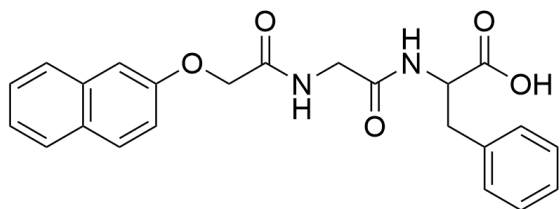
training22 - Non-Gel



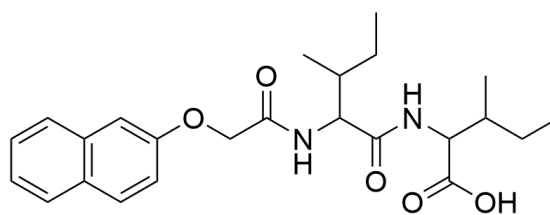
training23 - Non-Gel



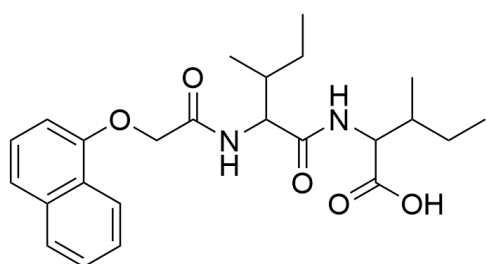
training24 - Non-Gel



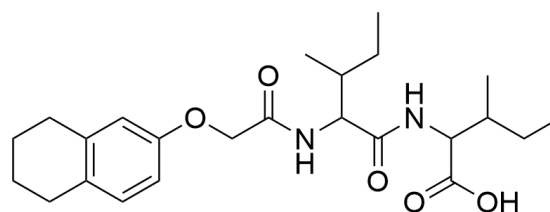
training25 - Non-Gel



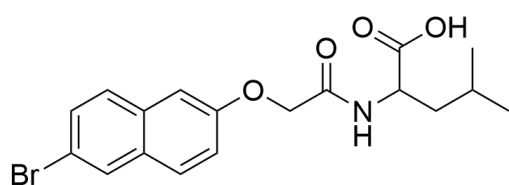
training26 - Non-Gel



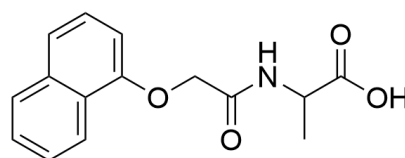
training27 - Non-Gel



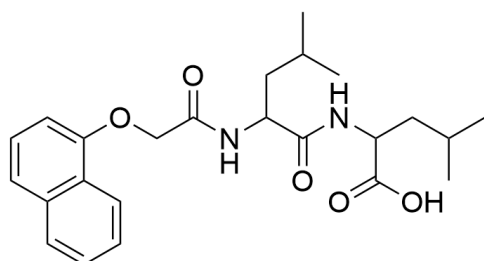
training28 - Non-Gel



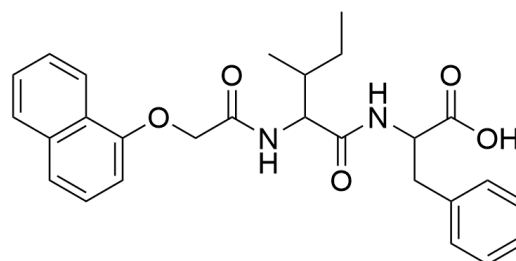
training29 - Non-Gel



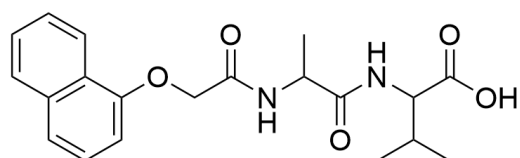
training30 - Non-Gel



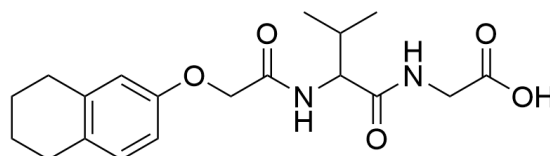
training31 - Non-Gel



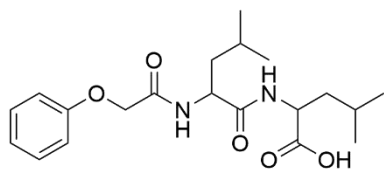
training32 - Non-Gel



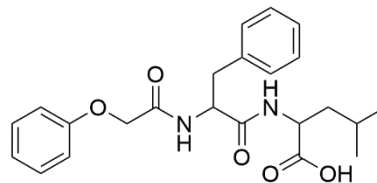
training33 - Non-Gel



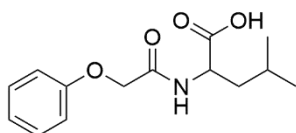
training34 - Non-Gel



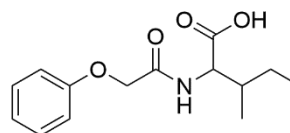
test1 - Non-Gel



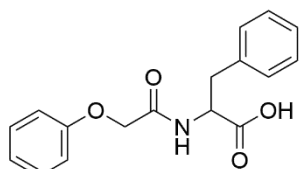
test2 - Non-Gel



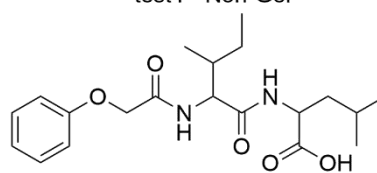
test3 - Non-Gel



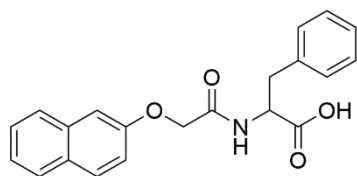
test4 - Non-Gel



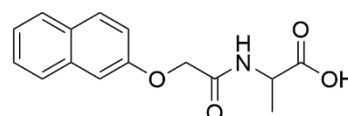
test5 - Non-Gel



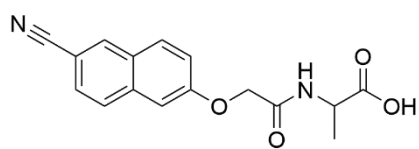
test6 - Non-Gel



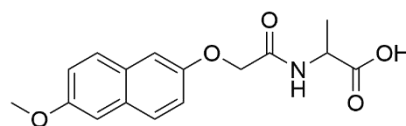
test7 - Non-Gel



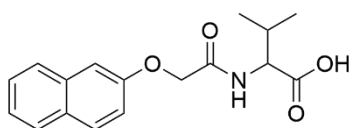
test8 - Non-Gel



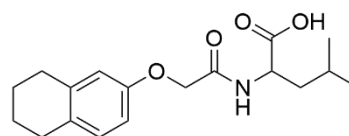
test9 - Non-Gel



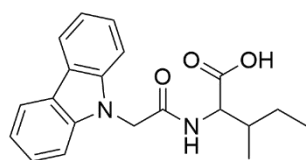
test10 - Non-Gel



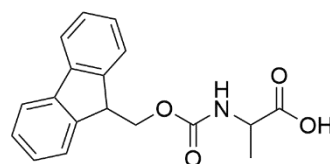
test11 - Non-Gel



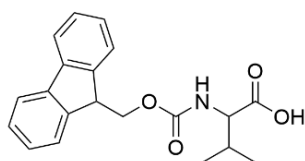
test12 - Non-Gel



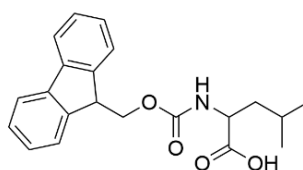
test13 - Non-Gel



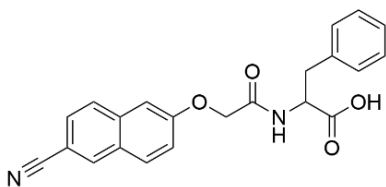
test14 - Non-Gel



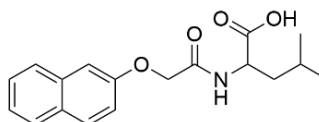
test15 - Non-Gel



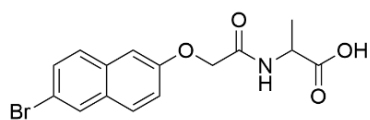
test16 - Non-Gel



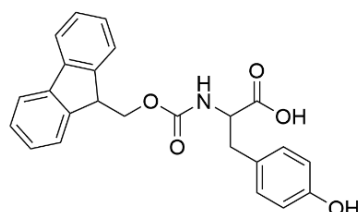
test17 - Gel



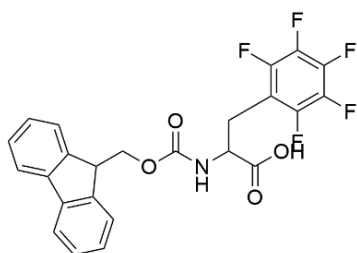
test18 - Non-Gel



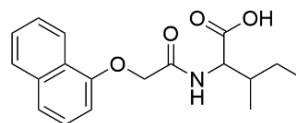
test19 - Gel



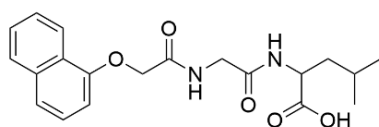
test20 - Gel



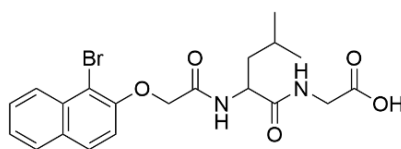
test21 - Gel



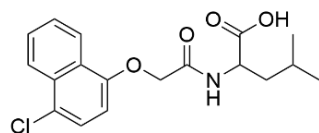
val1 - Non-Gel



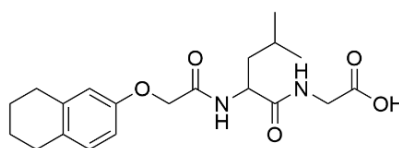
val2 - Non-Gel



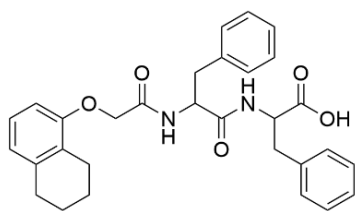
val3 - Non-Gel



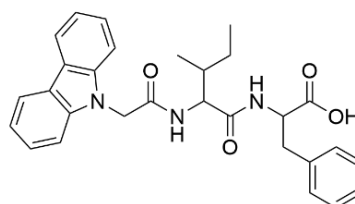
val4 - Non-Gel



val5 - Non-Gel



val6 - Gel



val7 - Gel

2.6.2 *fp_as_bits* descriptors

ALogP, Num_Rings, Molecular_PolarSASA, Molecular_Solubility, ECFP_4.7, ECFP_4.10, ECFP_4.241, ECFP_4.489, ECFP_4.676, ECFP_4.727, ECFP_4.734, ECFP_4.830 ECFP_4.947, ECFP_4.979, ECFP_4.1069, ECFP_4.1107, ECFP_4.1282, ECFP_4.1342, ECFP_4.1412, ECFP_4.1428, ECFP_4.1485, ECFP_4.1488, ECFP_4.1512, ECFP_4.1588, ECFP_4.1749, ECFP_4.1779, ECFP_4.1837, ECFP_4.1893, ECFP_4.1951, FCFP_4.65, FCFP_4.214, FCFP_4.271, FCFP_4.378, FCFP_4.464, FCFP_4.466, FCFP_4.574 FCFP_4.614, FCFP_4.805, FCFP_4.922, FCFP_4.943, FCFP_4.1056, FCFP_4.1152, FCFP_4.1194, FCFP_4.1343, FCFP_4.1453, FCFP_4.1479, FCFP_4.1485, FCFP_4.1496, FCFP_4.1565, FCFP_4.1732, FCFP_4.1828, FCFP_4.1871, FCFP_4.1873, FCFP_4.1883, FCFP_4.1961, FCFP_4.1972, FCFP_4.2007

2.6.3 *fp_as_features* descriptors

ALogP, Num_Rings, Molecular_PolarSASA, Molecular_Solubility, CM.FCFP_4.-453677277, CM.FCFP_4.136597326, CM.FCFP_4.1674451008, CM.FCFP_4.-1043339860, CM.FCFP_4.-1272798659, CM.ECFP_4.532107990 CM.ECFP_4.-244977436, CM.ECFP_4.-1187934975, CM.ECFP_4.1931561167, CM.ECFP_4.1307307440, CM.ECFP_4.1370095678, CM.ECFP_4.-649348348, CM.ECFP_4.-1194163736, CM.ECFP_4.1564392544, CM.ECFP_4.944269100, CM.ECFP_4.1336540477, CM.ECFP_4.-1145977934, CM.ECFP_4.-2019199918, CM.ECFP_4.1708805596, CM.ECFP_4.642810091, CM.ECFP_4.1813259826 CM.ECFP_4.1640927238, CM.ECFP_4.1810416527, CM.ECFP_4.-206566761, CM.ECFP_4.734603939, CM.ECFP_4.99898855, CM.ECFP_4.709582886, CM.ECFP_4.-178525456, CM.ECFP_4.1639858918, CM.ECFP_4.-755462605, CM.ECFP_4.-1886441421, CM.ECFP_4.711018196, CM.ECFP_4.-1096757101, CM.ECFP_4.721225968, CM.ECFP_4.-2049718428, CM.ECFP_4.-1205575950, CM.ECFP_4.497523368, CM.ECFP_4.-1421541878, CM.ECFP_4.706739587 CM.ECFP_4.1252107495, CM.ECFP_4.-742649778, CM.ECFP_4.716689417, CM.ECFP_4.-2123658813, CM.ECFP_4.391801793, CM.ECFP_4.-1380908375, CM.ECFP_4.863188371, CM.ECFP_4.717474525, CM.ECFP_4.864518973, CM.ECFP_4.1731135544

2.6.4 *py_class* descriptors

num_rings, alogp, mol_solu, FCFP_4, FCFP_6, FCFP_8, FCFP_21, FCFP_25, FCFP_71, FCFP_124, FCFP_147, FCFP_148, FCFP_208, FCFP_224, FCFP_397, FCFP_403, FCFP_416, FCFP_428, FCFP_468, FCFP_548, FCFP_598, FCFP_646, FCFP_764, FCFP_789, FCFP_792, FCFP_1063, FCFP_1085, FCFP_1157, FCFP_1336, FCFP_1395, FCFP_1566, FCFP_1668, FCFP_1706, FCFP_1727, FCFP_1797, FCFP_1863, FCFP_1907, ECFP_173, ECFP_197, ECFP_203, ECFP_283, ECFP_294, ECFP_322, ECFP_507, ECFP_598, ECFP_736, ECFP_875, ECFP_895, ECFP_989, ECFP_1039, ECFP_1057, ECFP_1087, ECFP_1107, ECFP_1118, ECFP_1575, ECFP_1623, ECFP_1624, ECFP_2016

Chapter 3 – Rheology Regression Models

3.1 Introduction

Although the classification models reported in the literature and built upon in chapter 2 are a positive step of whether or not a small molecule will form a gel under the glucono- δ -lactone trigger. Most gels are synthesised with an application in mind and there have been numerous reports of rheology dependent application, as discussed in Chapter 1. For example, the Alakpa *et al.*¹ work demonstrated that for three molecular gels with a range of stiffness, stem cells would differentiate preferentially down neuronal, chondrogenic or osteogenic pathways depending on the stiffness of the gel (Figure 3.1).

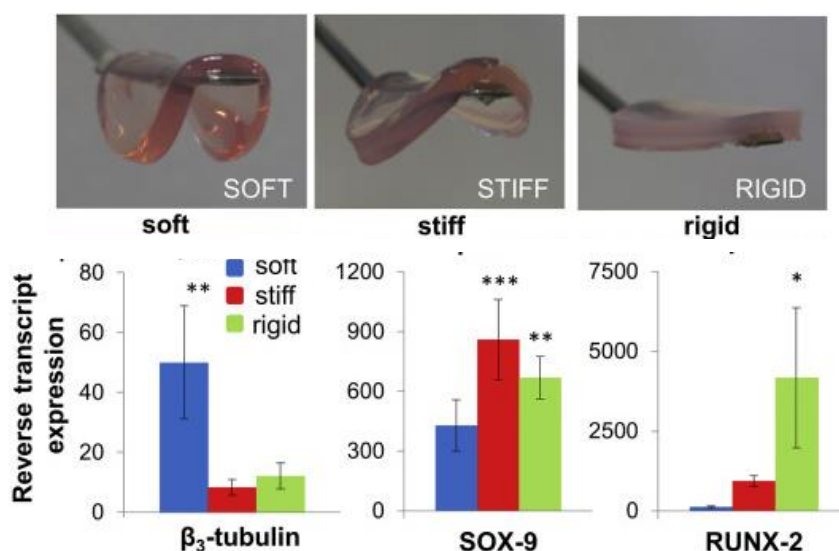


Figure 3.1: (top) the different stiffness of gel used by Alakpa *et al.* (bottom) the different expression of the neural, chondrocyte and osteoblast cells respectively for each gel. Figure adapted from [1]

The importance of the mechanical profile of gels for cell culturing has been further highlighted in work with peptide-based hydrogels.² Here, a peptide of 8 amino acids is modified by Arg-Gly-Asp to alter the concentration of peptide in solution and this is shown to alter the stiffness of the gel. Moreover, as in Alakpa¹, they show that the stiffness of the gel is important in determining the differentiation pathway of cells.

Also, the stiffness of the gel is influential if the gel is to be used as a drug-delivery medium.³ For example, if the drug is to be delivered via injection, the gel must possess the mechanical properties to become a low viscous gel upon the application of a shear stress (*see Chapter 1.3 - Rheology*) before returning into the solid-like gel once the shear stress has stopped.⁴

There are a range of reports of injectable peptide-based hydrogels. For example, Thota *et al.*⁵ report a leucine-phenylalanine dipeptide based gel which they successfully show shear thinning and recovering post injection. Moreover, they load the gel with both hydrophilic and hydrophobic drugs such as Neomycin (Figure 3.2a) and Rifampicin (Figure 3.2b) that demonstrate slow and consistent release from the gel.

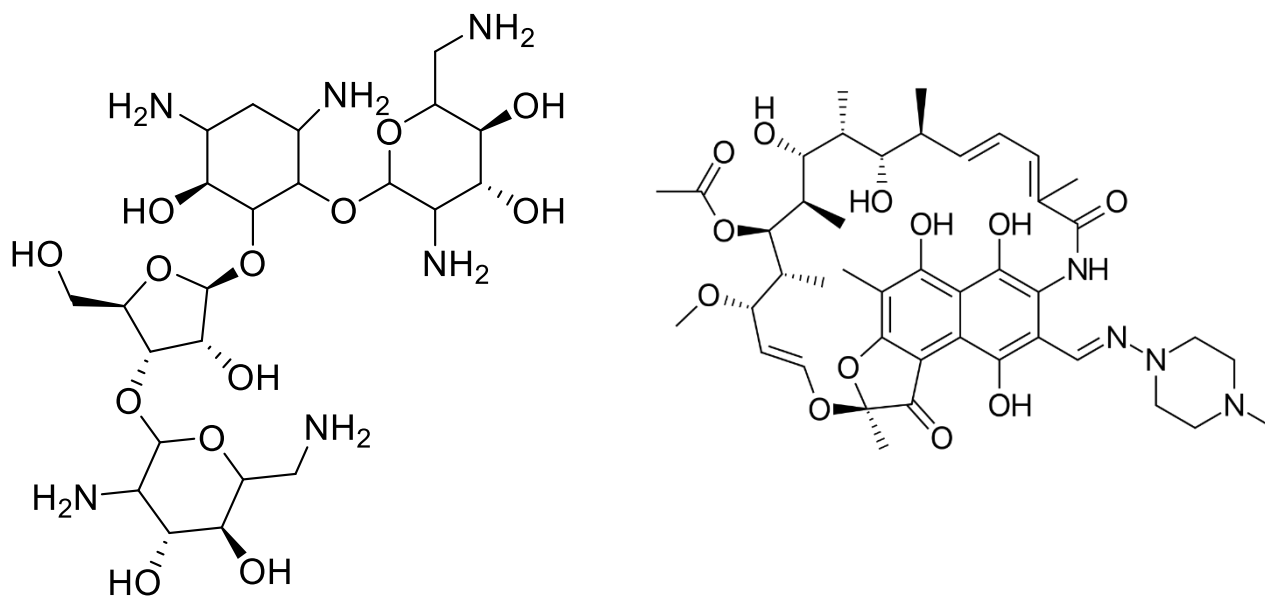


Figure 3.2: a) Neomycin and b) Rifampicin

Other examples of reported injectable hydrogels include a diphenylalanine based gel loaded with 5-Aminolevulinic acid (5-ALA) (Figure 3.3) a prodrug used to highlight tumour cells during surgery.⁶ In mice studies they show controlled and sustained release of 5-ALA after injection allowing for sustained visualisation of the tumour in mice.

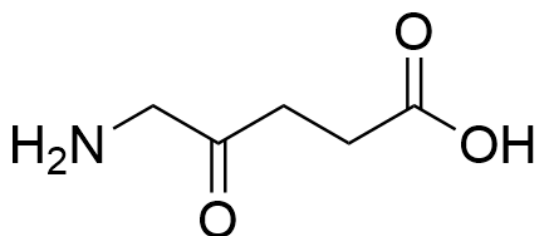


Figure 3.3: 5-Aminolevulinic acid

Another interesting potential application for these materials that is dependent on possessing the ability to become viscous upon the application of a shear stress before returning to a solid is as materials for 3D printing.⁷ Here they present a range of peptide-based gels (Figure 3.4a) that possess mechanical properties conducive to printing and produce printed gels (Figure 3.4b). The range of interesting potential applications highlight the benefits that profiling the mechanical properties before synthesis could produce for a candidate gel molecule.

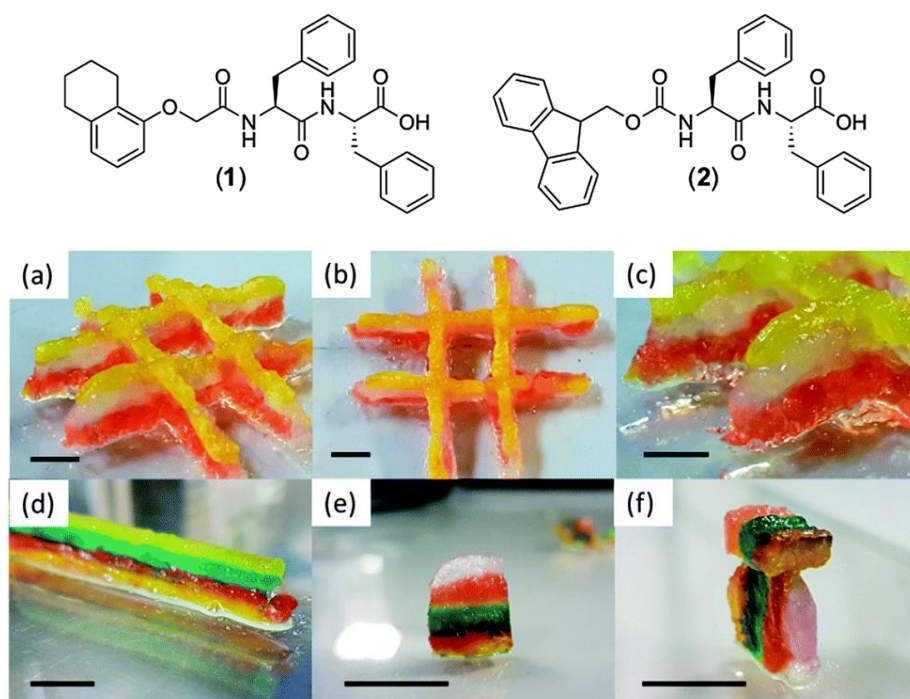


Figure 3.4: a) Gels used as the materials for 3D printing and b) Results of 3D prints showing stability in a range of conformations. Figures adapted from [7]

As introduced in *Chapter 1.3 - Rheology*, the mechanical properties of gels are determined by applying a shear stress to the gel and measuring the ensuing strain. The stiffness of the gel is quantified as the storage modulus (G') and the viscosity the loss modulus (G''). Therefore, in this chapter we produce predictive QSAR models to predict both the G' and G'' of peptide-based hydrogels. Our models are built on a gel dataset that is different to the one used to build the classification models of Chapter 2 but formed using the same glucono- δ -lactone trigger as the molecules in the classification. Structures and their respective G' and G'' can be found in the *Appendix 3.6.1*.

To do this, we attempt to use the same modelling approaches as Chapter 2, where appropriate. With this in mind, we examined PLS, kNN, Random Forest and Support Vector Machine modelling

algorithms. Moreover, we also looked at a linear model, the Bayesian Generalised Linear Model as a baseline model.

3.2 Experimental

3.2.1 Dataset

Our modelling dataset for our regression models was prepared by the Adams group in Glasgow using the same gelation approach as the molecules in Chapter 2 (*see 2.2 Experimental*) - the glucono- δ -lactone trigger method. Our dataset consists of 23 unique molecules (Figure 3.5), 22 of these molecules are present at 2.5, 5 and 10 mg/mL concentrations and a single molecule is present only at 2.5 mg/mL this results in a dataset of 67 $((22 * 3) + 1)$ points with unique G' and G'' values (*see Appendix 3.6.1 for values*). Our G' values span the range of 1624 Pa to 580333 Pa and our G'' values span the range 192 Pa to 52995 Pa, giving a range of three orders of magnitude for the entire G' and G'' space for our models.

3.2.2 Descriptor calculation

Our molecular descriptors were calculated using the same approach as the fp_as_features models in the previous chapter (*see 2.2.3.1 R Model Descriptors*). Molecular properties (AlogP, Molecular Weight, Num_Atoms, Num_RotatableBonds, Num_Rings, Num_H_Acceptors, Num_H_Donors, Num_AromaticRings, Molecular_SurfaceArea, Molecular_PolarSurfaceArea, MolecularPolarSASA, Molecular_Solubility and LogD) were calculated along with both the Extended Connectivity Fingerprint Count (ECFC) with radius 4 and the Functional Class Fingerprint Count (FCFC) also with radius 4.

Our fingerprints were kept as features, meaning no conversion to a bit string and the fingerprints are indexed by their hash values generated during calculation. The fingerprints are expressed as counts, giving the total number of times the fragment is present in the molecule.

Given that we have the same molecules present in the dataset at different concentrations, we need to carry out some feature engineering to make the calculated descriptors viable for model building. We engineered our features by multiplying all calculated descriptors by the concentration of the starting gelator. We do this as it has been shown that G' and G'' for functionalised dipeptides are linearly dependent on concentration⁸ and so we can utilise this knowledge to transform our descriptors to make them suitable for model building (Table 3.1).

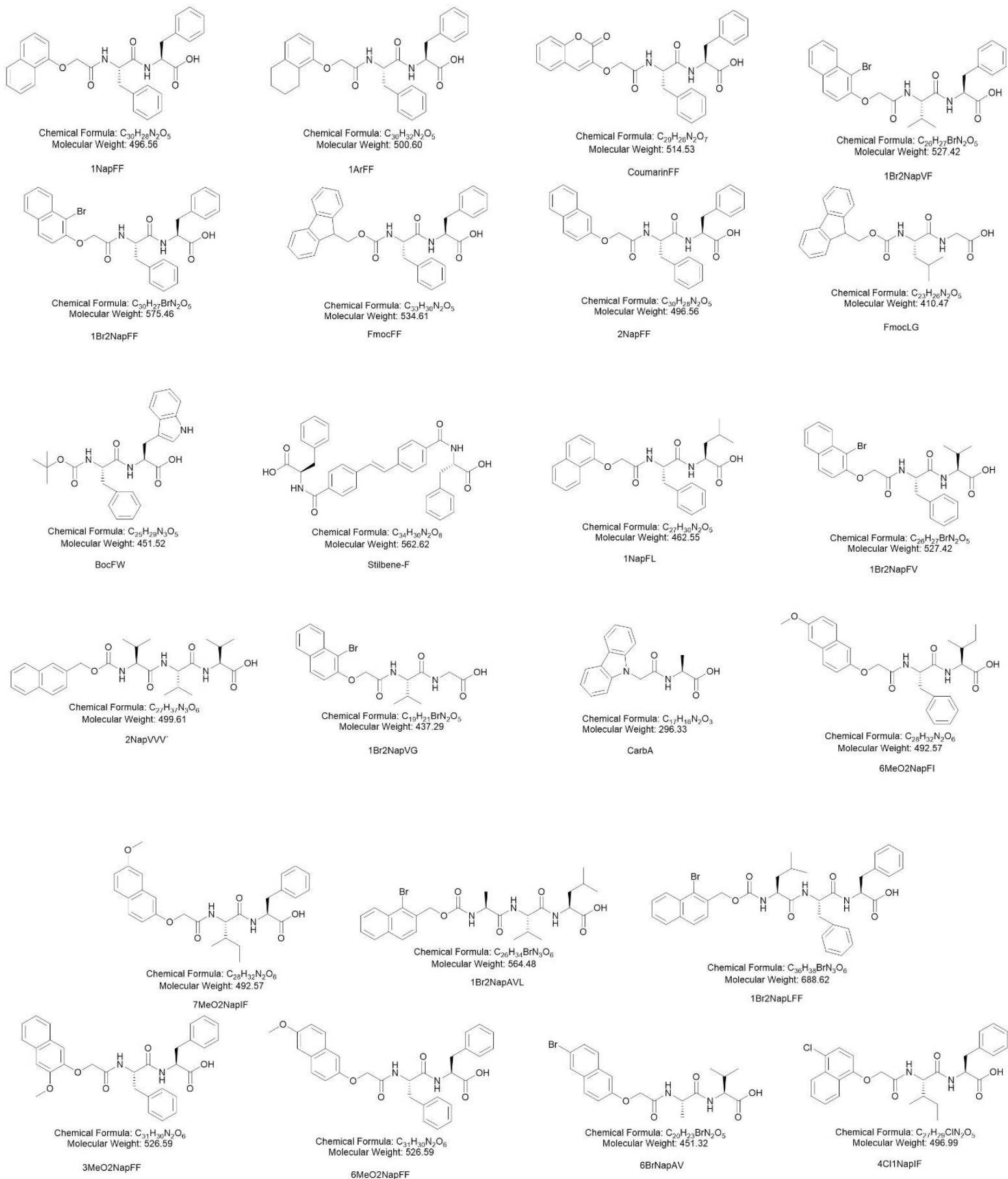


Figure 3.5: Chemical structures of the 23 molecules in the rheology dataset.

Table 3.1: Toy example showing the feature engineering of our descriptors

Starting Gelator Concentration (mg/mL)	Fingerprint 1 (pre-conversion)	Fingerprint 1 (Post-conversion)
2.5	2	5
5	2	10
10	1	10

3.2.3 Data Splitting

Given that our dataset is relatively small, we were mindful that the performance of any model could be heavily dependent on the split carried out on the data into training, test and validation sets. Therefore, we carried out a repeated data splitting procedure to gauge the extent the data split impacts model predictive performance. The overall splitting procedure is displayed graphically in Figure 3.6 and can be summarised as follows:

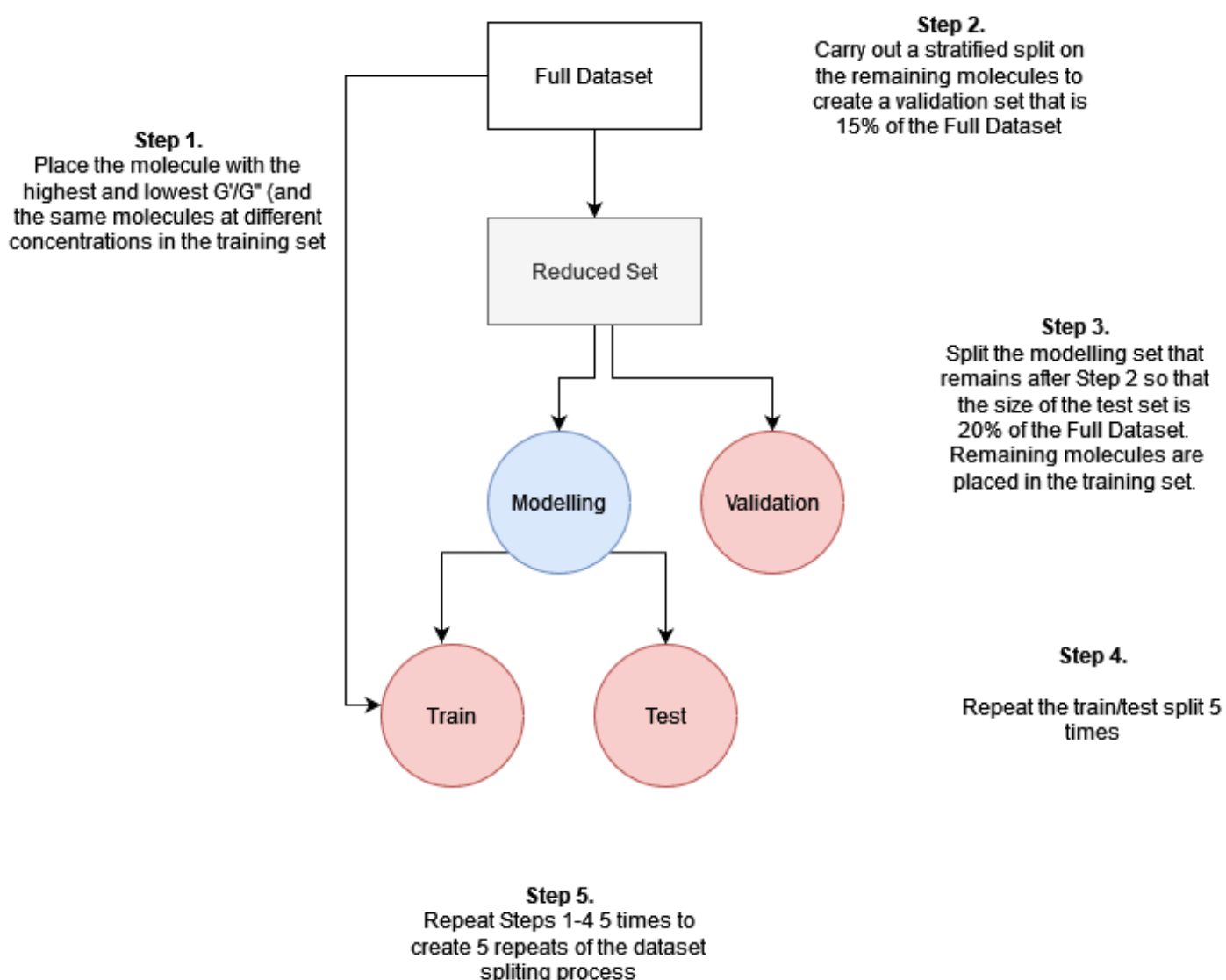


Figure 3.6: Flowchart demonstrating the data splitting procedure carried out in this chapter

From the entire dataset, molecules were placed with the smallest and largest endpoint values into the training set. All other instances of this molecule at different concentrations were also placed in the training set. The placement of the highest and lowest values into the training set ensures that no points in the test and validation set can be out-of-domain with respect to the G' or G'' . Placing all other concentrations in the training set addresses the potential that model performance could be artificially enhanced by having predictions of points in the training set present in the test and/or validation set, albeit at different concentrations.

From the remaining molecules, we split the data using `caret::CreateDataPartition()` data splitting procedure with 15% of molecules selected. It is worth noting that if the same molecule was picked more than once (for example a molecule could be picked at both 2.5 and 5 mg/mL in the initial split) in the original `caret::CreateDataPartition()`, then the size of the dataset could be larger or smaller than 15%. This 15% would form the validation set and the remaining 85% the modelling set. We repeat this procedure 5 times to obtain 5 modelling sets with a corresponding 5 validation sets.

From the modelling set molecules that were not already assigned to the training set, we split using `caret::CreateDataPartition()` so that the test set is approximately 20% of the entire dataset. We repeated this train/test split 5 times so that for each repeat we have 5 training sets, 5 test sets and a single unique validation set. Overall, this resulted in 25 training sets, 25 test sets and 5 validation sets.

3.2.4 PreProcessing

For each training set, we used the `caret` package to carry out the preprocessing. Firstly, we used the `caret::nearZeroVar()` function to remove any descriptors where 95% of the values are the same value. Of the remaining descriptors, we calculated the correlation between them using the `caret::findcorrelation()` with a cut-off of 0.9. We removed the descriptor in the pair that has the highest correlation with the other descriptors in the feature set. Each test set and validation set descriptor files were cleaned to match those remaining in the test set after processing.

Before model building, we examined the G' and G'' values based upon the range of their values. For G' , the value at 580333 Pa is a large outlier, with no data points with similar G' values. We can use the Z-Score (introduced in 1.10 – *Y-randomisation*) to categorise a point as an outlier if its Z-score is greater than 2.5⁹. Our point's Z-score of 5.49 clearly exceeds the limit and as such is an outlier.

To compensate for this, we normalise the G' and G'' values by taking the \log_{10} of all the G' and G'' values, bringing the entire dataset onto the same scale.¹⁰

3.2.5 Model Building

All model building was carried out using the `caret::train()` function. Centring and scaling of the data was carried out during training and is automatically applied to the data for any new predictions made by the model. During model building, we carried out 10-fold cross-validation repeated 10 times to optimise each model's hyperparameters.

Initially, we learnt a model using a simple linear regression model, namely a BayesGLM. We then investigated more complex models including a kNN, PLS, SVM and RF models. For each training set, we build 10 models for each algorithm to gauge an average performance for each model.

3.2.6 Model Validation

The performance of each model was assessed on the test set initially, and if the performance of the test set was deemed acceptable it was then exposed to the molecules in the validation set. The performance of the models were assessed using both the RMSE (Equation 3.1) and R^2 (Equation 3.2) between the experimental values and the model's prediction.

As a measure of good prediction, we utilise the framework for R^2 and RMSE proposed by Alexander *et al.*¹¹ which states that models can be considered predictive if they have:

- 3 High R^2 ($R^2 > 0.6$) on the test set
- 4 Low RMSE

Given that the value of RMSE depends on the scale of the predictions, for our best model we calculate RMSE and check if the RMSE is "low". Given the RMSE is dependent on the scale of the endpoint value, setting thresholds for "low" is difficult. However, normalising the RMSE between the minimum and maximum values of our training set and assigning a threshold relative to this value. 0.1 (10%) has been used as a threshold for good for the normalised RMSE¹². Given that our data roughly covers 2.6 log units ($G' = 3.2$ to $G' = 5.8$), we consider RMSE of 10% this range (0.3 or below) as "low". We consider both $R^2 > 0.6$ and $RMSE < 0.3$ as our thresholds for good.

$$RMSE = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_i - \hat{y}_i)^2} \quad (3.1)$$

$$R^2 = \frac{\sum (y - \hat{y})^2}{\sum (y - \bar{y})^2} \quad (3.2)$$

We also subject any good models to a Y-Randomisation approach (*see 1.10 - Y-randomisation*) by generating 50 randomised models using the same modelling approach on shuffled data. We calculated the R² and RMSE for each of these models and calculate the mean of their distribution. Calculation of a Z score (equation 3.3) allows us to calculate how many standard deviations away our true R² (RMSE) are from the randomised mean R² (RMSE) and gives an idea of our confidence as to whether the true model was drawn from the randomised model distribution or not.

$$Z = \frac{x - \text{average}(\tilde{x})}{\text{standard deviation}(\tilde{x})} \quad (3.3)$$

Where x is the RMSE or R² value and \tilde{x} is the same metric for a randomised model.

For our y-randomisation approach, 50 models are trained on shuffled data using the best performing modelling algorithm. Using these models, we predict on the test set and compare the average R² and RMSE to the values obtained from the best model.

3.2.7 Applicability domain

For our applicability domain, we again carry out the range check approach on the descriptors used in Chapter 2 (*see 2.2.9 – Applicability Domain*). We compared the minimum and maximum values of the descriptors for each descriptor in the training set, plus the minimum and maximum values of the first 10 principal components and compare the values of the test and validation set descriptors against these boundaries. If any descriptor falls outside of these bounds, the molecule is considered out-of-domain

3.2.8 Model Interpretation

As with the classification models, we investigated model interpretation of any good model using both the `caret::varImp()` function which will utilise any built-in model interpretation metric, if it existed or defaulted to a ROC analysis if not and with the Shap game theory approach to investigate important molecular properties utilising the R wrapper (*version 0.1.3*) of the Python Shap package.¹³

3.3 Results and Discussion

After the data splitting, we were left with 25 training sets to use for model building. For ease, we will refer to each set with a numeric identifier. For example, 1_1 is the first (of 5) training set, from the first repeat (of 5) and 2_3 is the third training set for the second repeat and so on.

3.3.1 G' Models

3.3.1.1 BayesGLM

3.3.1.1.1 Training Performance

For the BayesGLM models, all 10 models for each of the 25 training sets produces the same model, *i.e.* all performance metrics for the 10 models built on training set 1_1 have 0 variance. This leaves us with only 25 unique models and the performance of these models on the training set is summarised in Figure 3.7.

Figures 3.7a and 3.7b give an overview of the general performance of all 250 models over the 25 training sets. The plots are organised such that, for example, the R^2 values for Repeat 2 Set 1 are the R^2 values for the models learnt using training set 2_1 *etc.*

Regardless of the train/test/validation split carried out, all models appear to perform similarly during training. All models are clearly above the 0.6 threshold for R^2 on the training set with the highest R^2 of 0.80 (± 0.00) for the 5_2 and 5_3 datasets and the lowest R^2 of 0.75 (± 0.00) for the 4_5 dataset highlighting the low variation in R^2 values across the different training sets.

As for the RMSE, these values are much closer to the threshold of 0.3 than the models were to the threshold for R^2 . The largest RMSE on the training set was for the 4_5 dataset showing that the 4_5 dataset shows the worst performance during training. The smallest RMSE was 0.26 (± 0.00) for both the 5_3 and 4_1 datasets showing that the 5_3 performs best overall in training. The variance in the RMSE values is also low ranging from 0.26 (± 0.00) to 0.30 (± 0.00), highlighting that these models are potentially very similar.

Performance of the BayesGLM model on the training set in terms of R^2 and RMSE

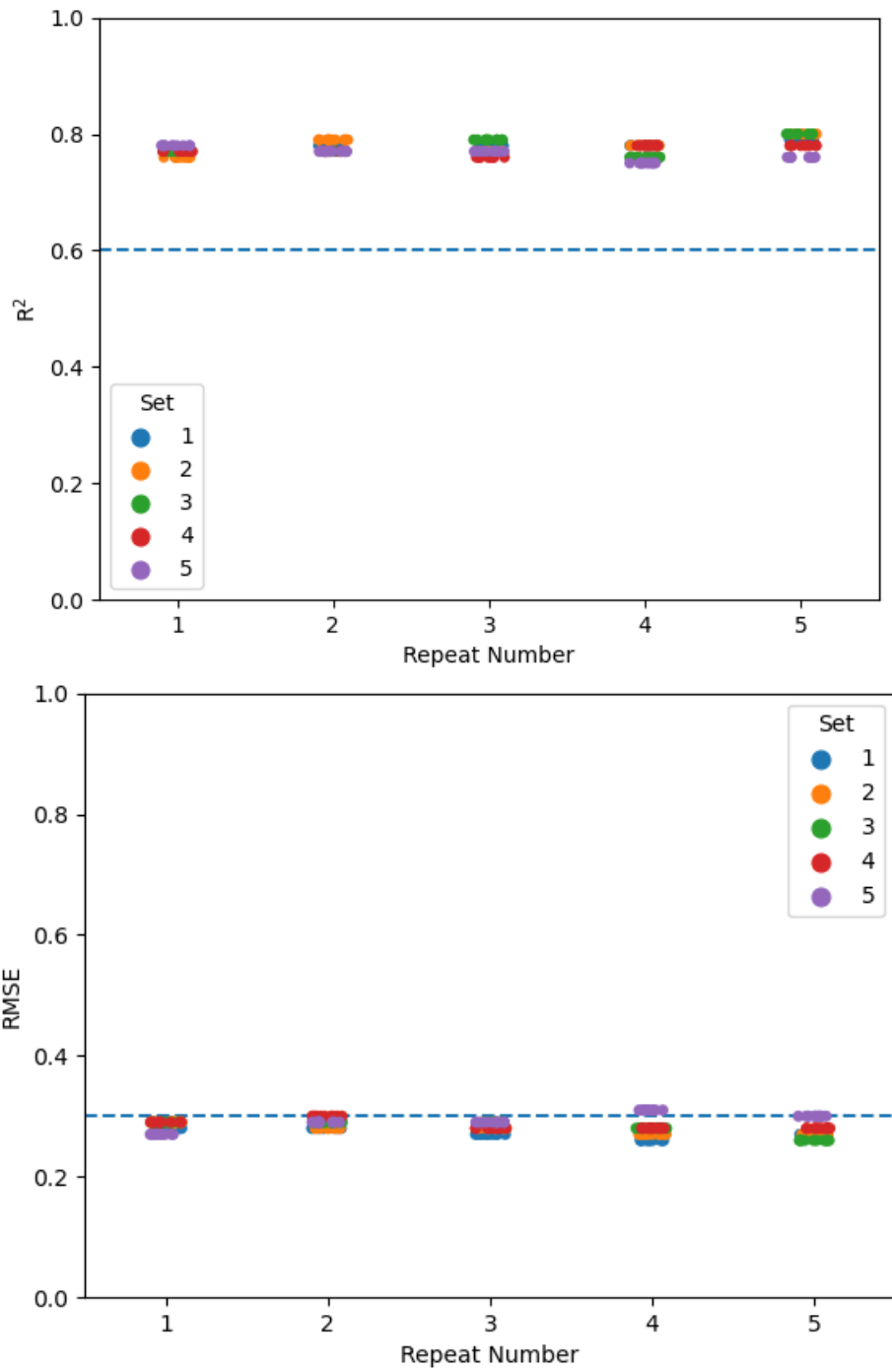


Figure 3.7 a) R^2 and b) RMSE of the 10 BayesGLM models learnt from each training set generated from the data splitting procedure

3.3.1.1.2 Test set performance

The performance of any model is only knowable after exposing it to data points it has not yet seen. Figures 3.8a and 3.8b show equivalent plots as 3.7a and 3.7b but on each data split's corresponding test set.

When considering the R^2 cut-off of 0.6, only three of the datasets produce BayesGLM models that exceed that threshold, 3_2, 3_3 and 3_4. These models have R^2 values of 0.68 (± 0.00), 0.78 (± 0.00) and 0.65 (± 0.00) respectively. However, for these datasets when considering the RMSE values for these 3 datasets, none of them produce models with test RMSE below our 0.3 threshold. The 3_4 model has an RMSE value of 0.69 (± 0.00) which is very high and is thus considered to be a poor model. The 3_2 models' RMSE are 0.42 (± 0.00) and for 3_3 the value is 0.43 (± 0.00). Overall, even though these fail to meet our threshold, given their high R^2 values they are exposed to the validation set to gauge whether they are useful models.

3.3.1.1.3 Validation set performance

As a final prediction, we exposed the models in repeat 3 to the validation set. It is worth remembering that for 3_2 and 3_3 the validation set is identical, so these predictions are a direct comparison of their predictive performance. Figures 3.9a and 3.9b show the performance of the repeat 3 set on the validation set which although close, fails to reach the R^2 threshold in both cases. The R^2 for the 3_2 dataset is 0.54 (± 0.00) with an RMSE of 0.4. For the 3_3 dataset the values are 0.56 (± 0.00) and 0.58 (± 0.00) respectively. Overall, we can conclude that these models fail to accurately model the G' values of our dataset.

Performance of the BayesGLM model on the test set in terms of R^2 and RMSE

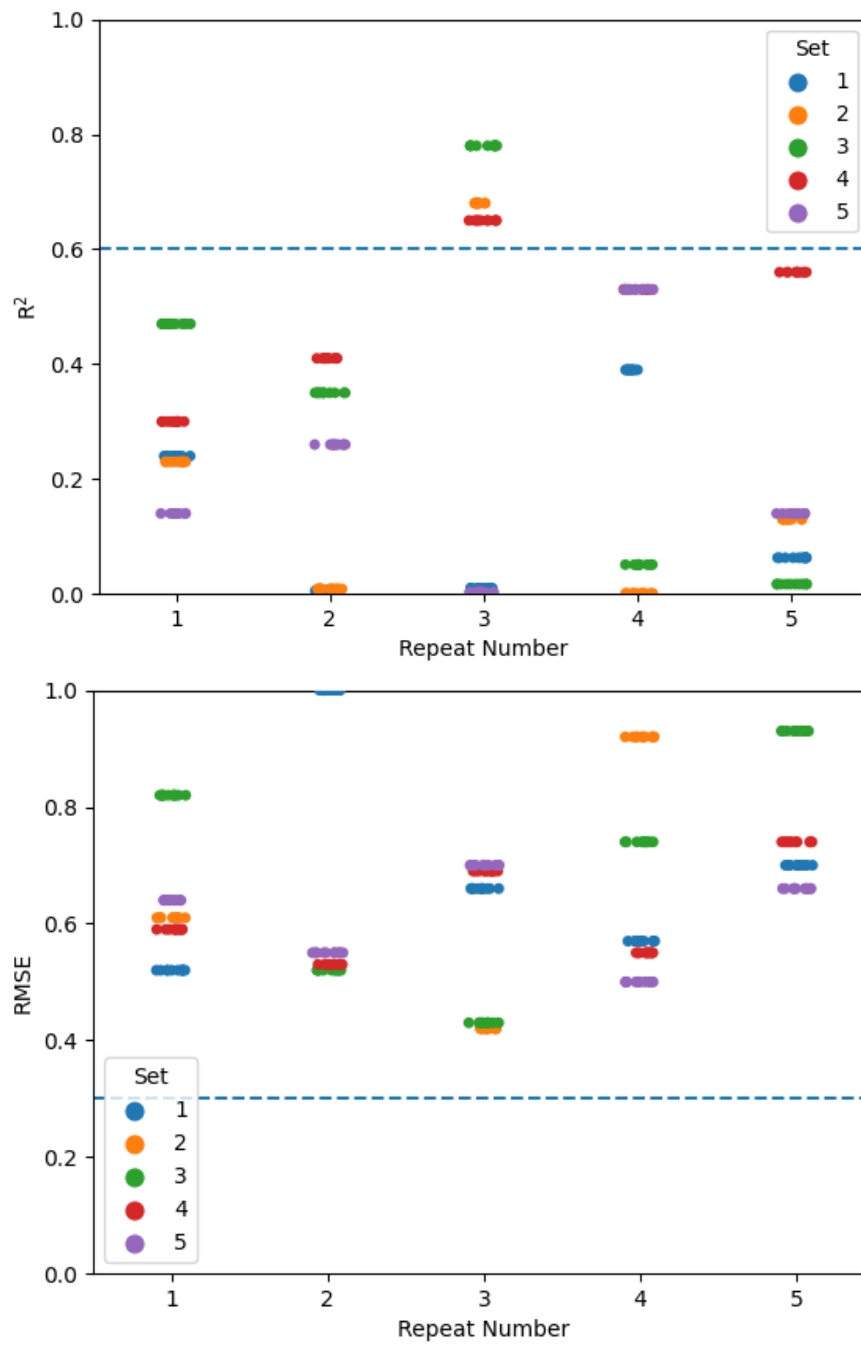


Figure 3.8a) R^2 and b) RMSE of the 10 BayesGLM models on the test set from the test set generated with each training set.

Performance of the BayesGLM model on the validation set in terms of R^2 and RMSE

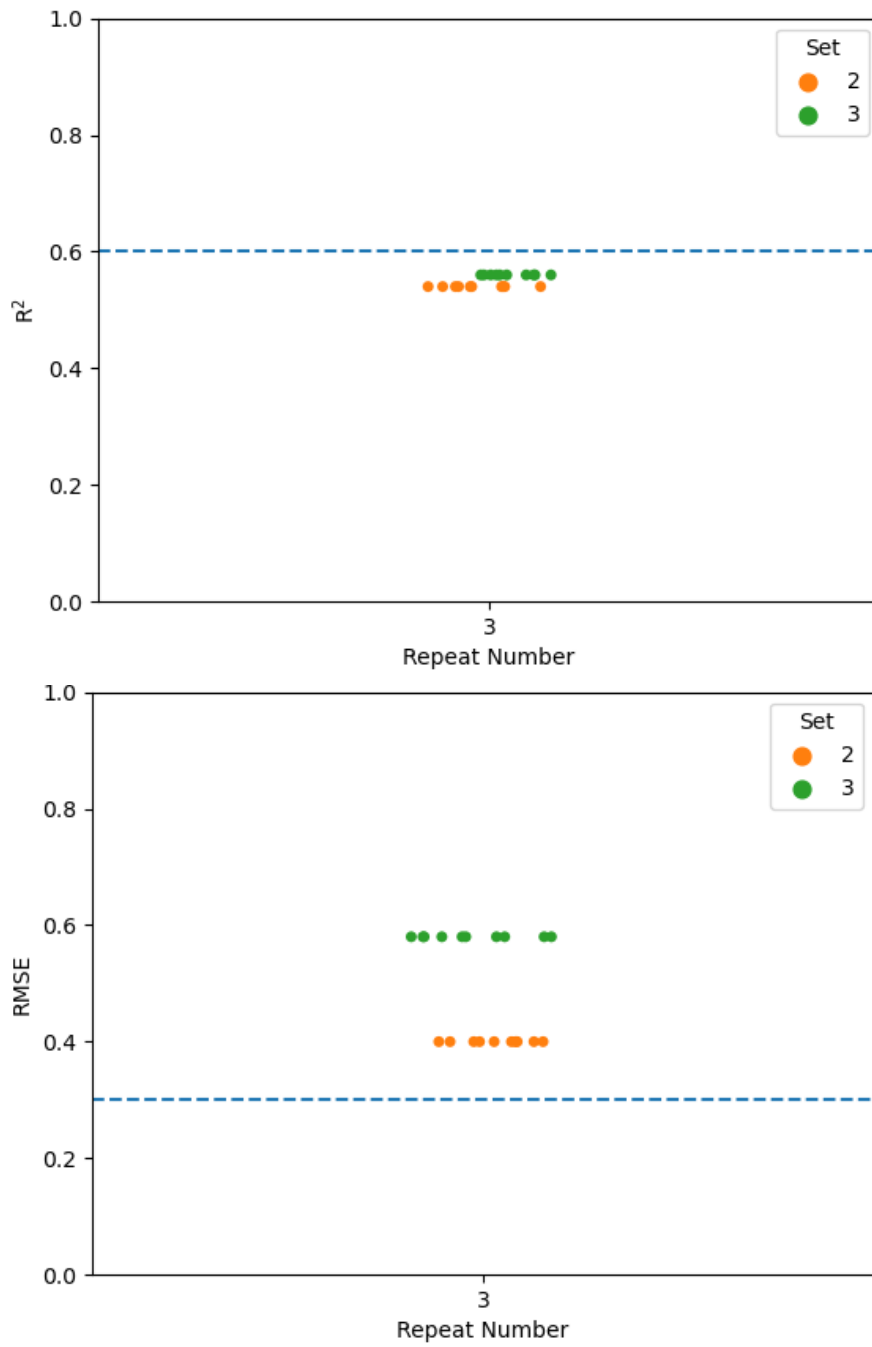


Figure 3.9: Performance of the 3_2 and 3_3 datasets on the validation set.

3.3.1.1.4 Best Performing Model

Although neither the 3_2 or 3_3 models pass the threshold for good on either the test and validation set, all models should be inspected visually before conclusions about a model's predictive performance can be drawn.¹⁴ Figure 3.10 shows a plot of the experimentally determined G' values against the values predicted by the 3_2 BayesGLM model as it exhibits lower RMSE than the 3_3 model.

The plot in Figure 3.10 confirms visually the idea that the model (model 1) fails to accurately model the G' for the test and validation sets. We see generally visually poor prediction on the validation set which corroborates the low R^2 of 0.54. Exploring more complex models may allow for models that can deduce more complicated relationships between the descriptors and G' .

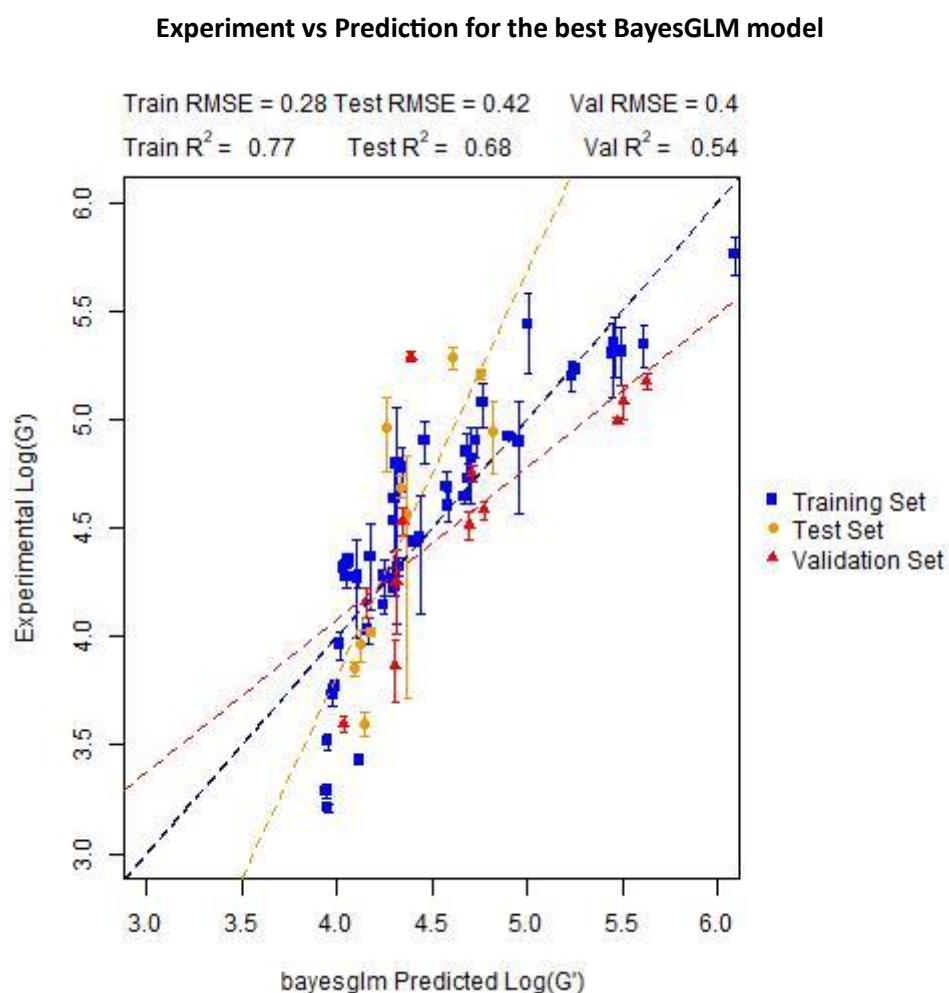


Figure 3.10: Experimental vs Predicted plot of the 3_2 BayesGLM model (model 1) on the train, test and validation sets with lines of best fit for each dataset.

In Figure 3.10, the three largest outliers in the model are a validation, training, and test set points respectively. The validation set point, 4ClNapIF and test set point 6BrNapAV are both functionalised dipeptides. Given that 30 of the 45 training set molecules are also functionalised dipeptides, it would hint that the model should be trained in the chemical space of these molecules suggesting that poor predictive performance is driven by other factors. Surprisingly, we see a large residual on the training set point Stilbene-F. However, when investigating the nature of the molecules in the training set, Stilbene-F and CarbA are the only molecules in the training set that aren't either functionalised dipeptides or tripeptides suggesting that they are too dissimilar to other molecules in the training set.

3.3.1.2 k-Nearest Neighbours

3.3.1.2.1 Training Performance

Like in the BayesGLM models, there is very little variance within the 10 models for a particular training set in Figures 3.11a and 3.11b, although there is slightly more variance in both the R^2 and RMSE. For the R^2 , overall the performance is good, with all models built having training R^2 over 0.7. The lowest R^2 for the training set is 0.78 (± 0.02) for the 3_2 dataset and the highest is 0.99 (± 0.00) for the 4_1 and 4_4 dataset. The R^2 of 0.99 suggests that there could be an element of overfitting during training for these models, but this should be spotted in the test set performance for these models.

In terms of RMSE, the 4_1 dataset is the smallest, with an RMSE of 0.066 (± 0.00) further indicating potential overfitting. The RMSE for the 4_1 model is 0.066 also further hinting at overfitting for these models too. The largest RMSE during training is 0.36 (± 0.02) for the 3_2 dataset meaning this model is performing worst in training across both R^2 and RMSE. There are a range of datasets that fail to show an RMSE below 0.3 even during training. Overall, 1_1, 1_5, 2_4, 3_2, 3_4 5_2 and 5_3 all have RMSE above 0.3 during training.

Performance of the kNN model on the training set in terms of R^2 and RMSE

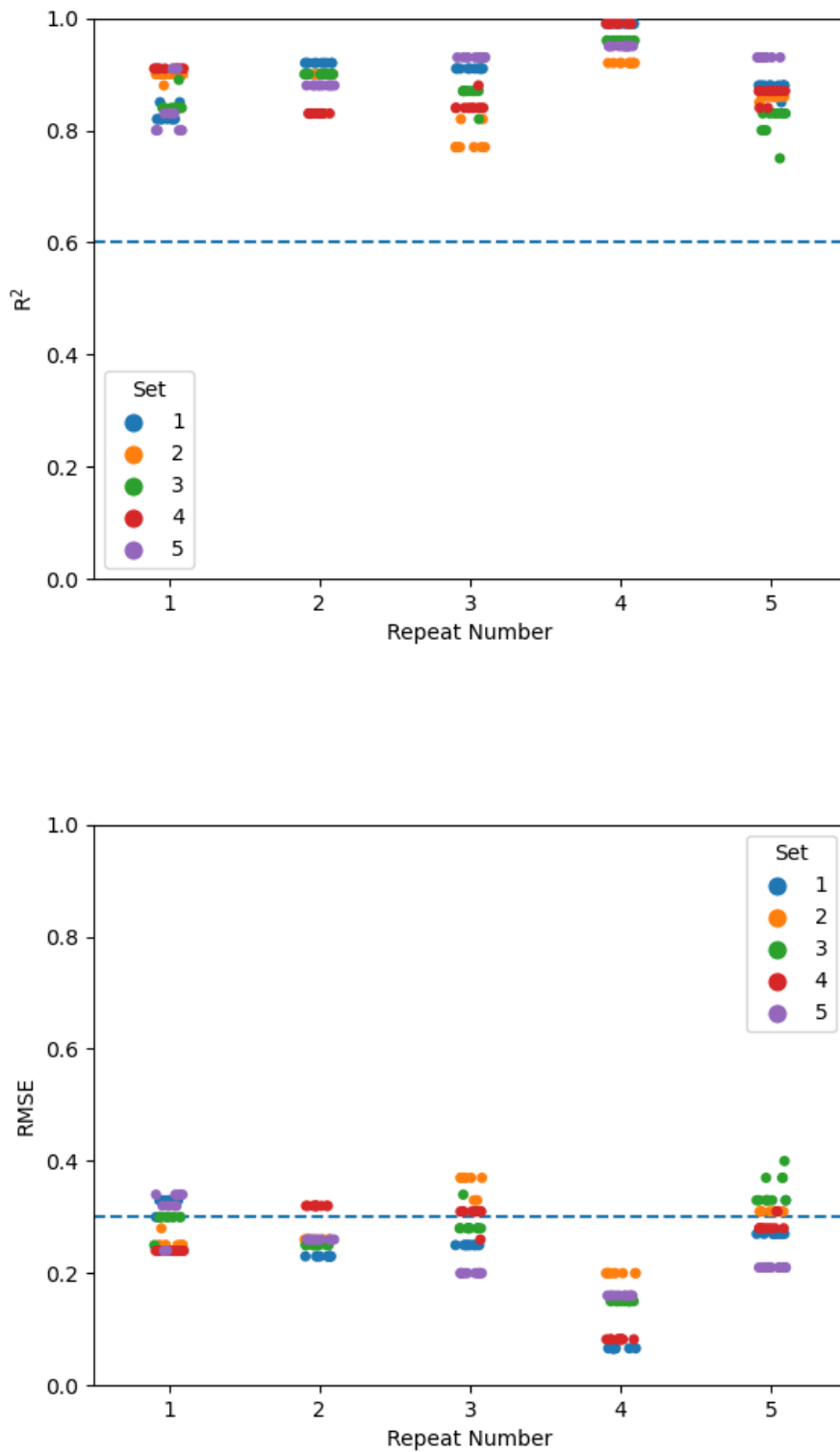


Figure 3.11: Training set performance of our 25 training sets using a k -Nearest Neighbours model

3.3.1.2.2 Test Performance

Performance of all models on the test set is shown in Figures 3.12a and 3.12b. For the R^2 , only 4 sets have models with values above 0.6, 1_2, 2_5, 3_2 and 4_4. For the RMSE, of these sets, 1_2 has an RMSE of 0.39 (± 0.00) which is slightly high but further investigation is warranted. 2_5 and 3_2 also have RMSE values 0.4 or below with 2_5 showing the best performance on the test set with R^2 of 0.8 (± 0.00) and RMSE of 0.25 (± 0.00). Therefore, we exposed, 1_2, 2_5 and 3_2 to the validation sets and assess their performance on another hold-out set.

3.3.1.2.3 Validation Performance

Performance of the 1_2, 2_5 and 3_2 datasets on the validation set can be seen in Figures 3.13a and 3.13b. Unfortunately, the only set of models that show moderate performance on the validation set is the 3_2 dataset. For the 1_2 model, the RMSE of 0.48 (± 0.00) coupled with the R^2 of 0.31 (± 0.00) suggests that performance is poor. For the 2_5 model the values are 0.55 (± 0.00) and 0.18 (± 0.00) respectively showing even worse performance. However, there are 2 models in the 3_2 dataset in particular that show promising performance. Both models are identical to each other and have R^2 of 0.63 and RMSE of 0.35. The RMSE is slightly above the threshold for good but close enough that visual inspection is warranted.

Performance of the kNN model on the test set in terms of R^2 and RMSE

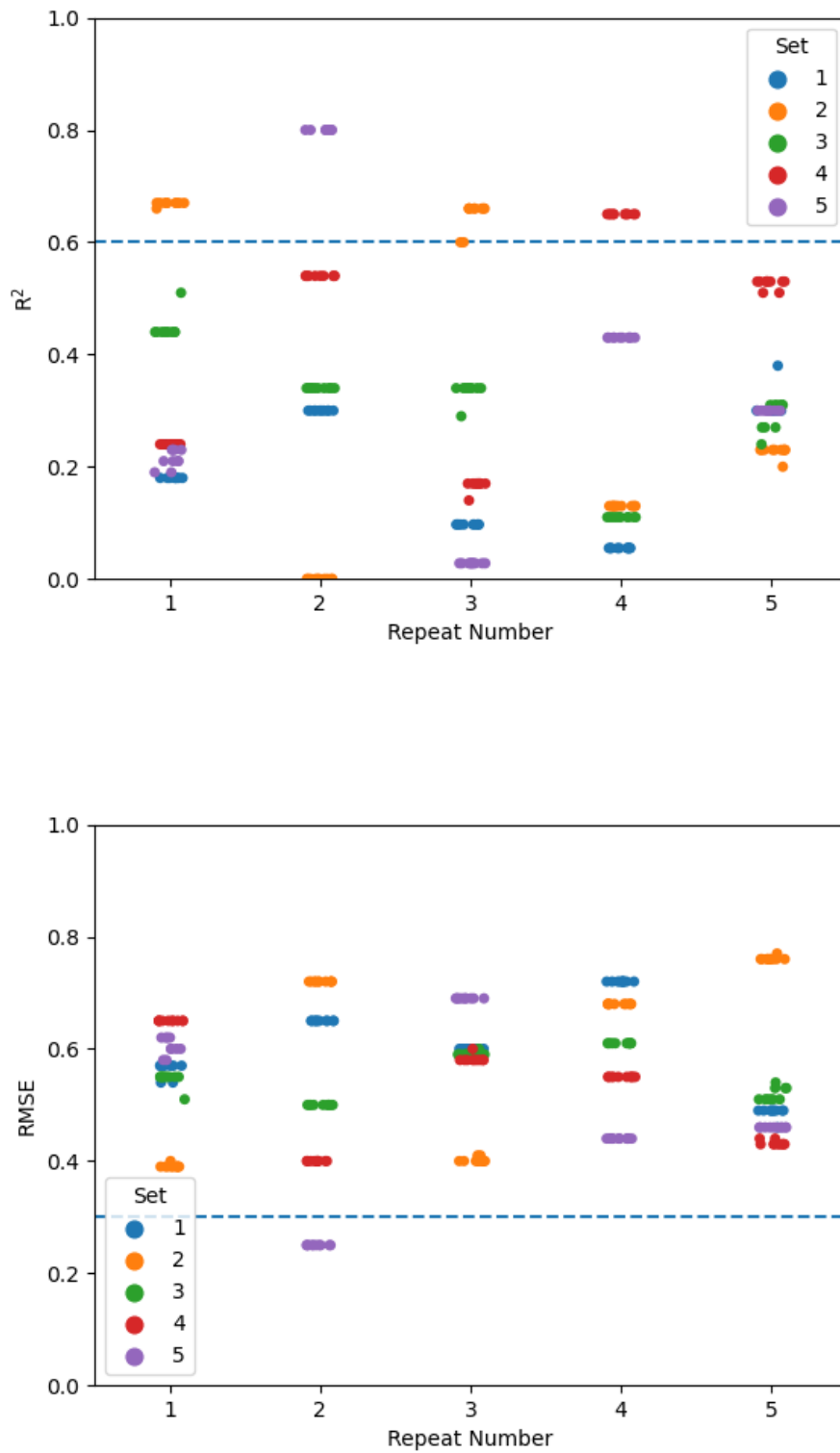


Figure 3.12: Performance of the kNN model on the test set

Performance of the kNN model on the validation set in terms of R² and RMSE

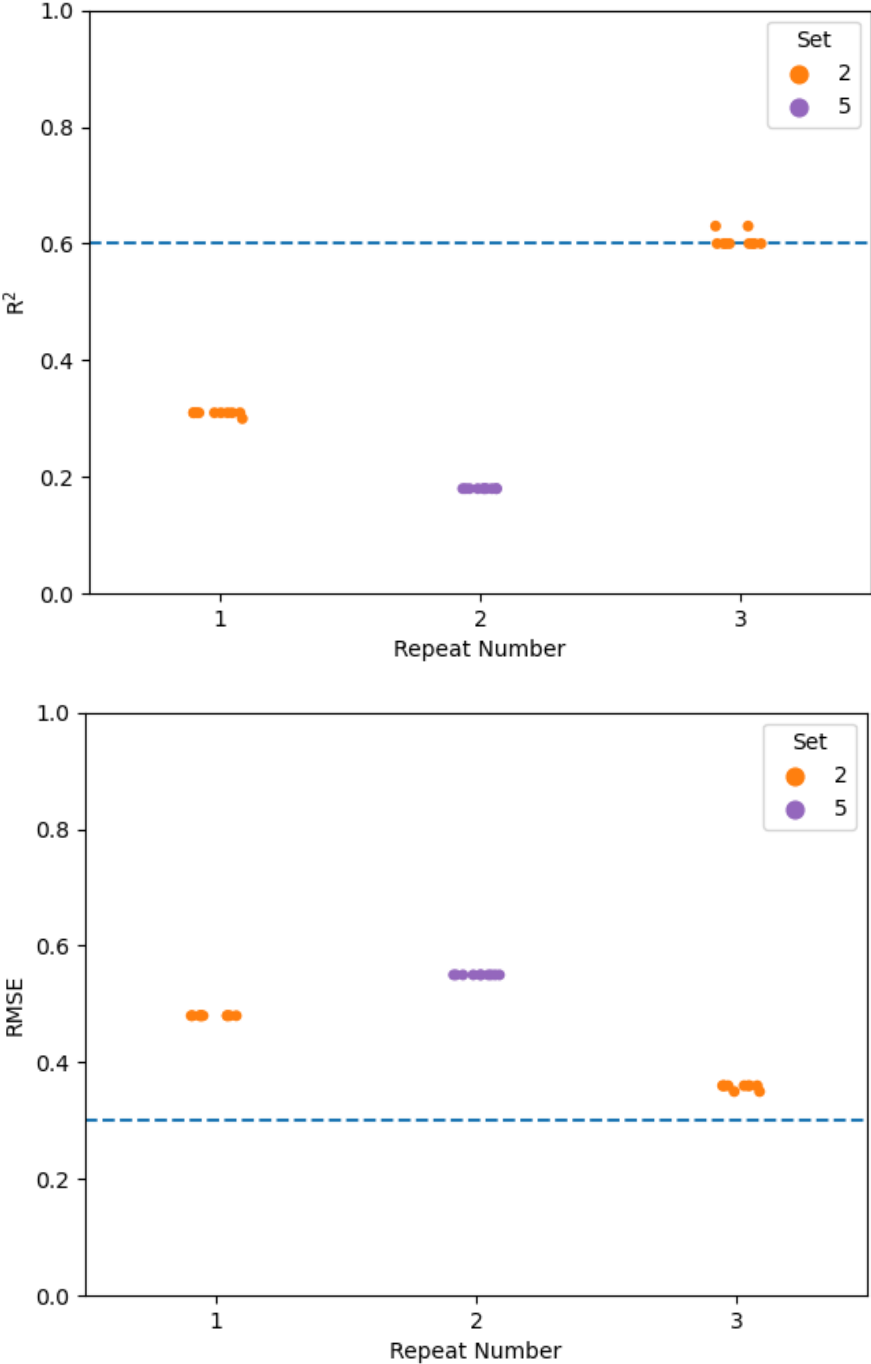


Figure 3.13: Performance of the 1_2, 2_5 and 3_2 dataset kNN models on the validation set

3.3.1.2.4 Best Performing Model

Visually, the performance of the kNN model (model 1) built using 3_2 shows overall good performance on the train, test and validation set (Figure 3.14) and is consistent with the good statistical performance we have found on all three sets. Interestingly, 3_2 has now been the best performing dataset for both the BayesGLM model and the k-Nearest Neighbours.

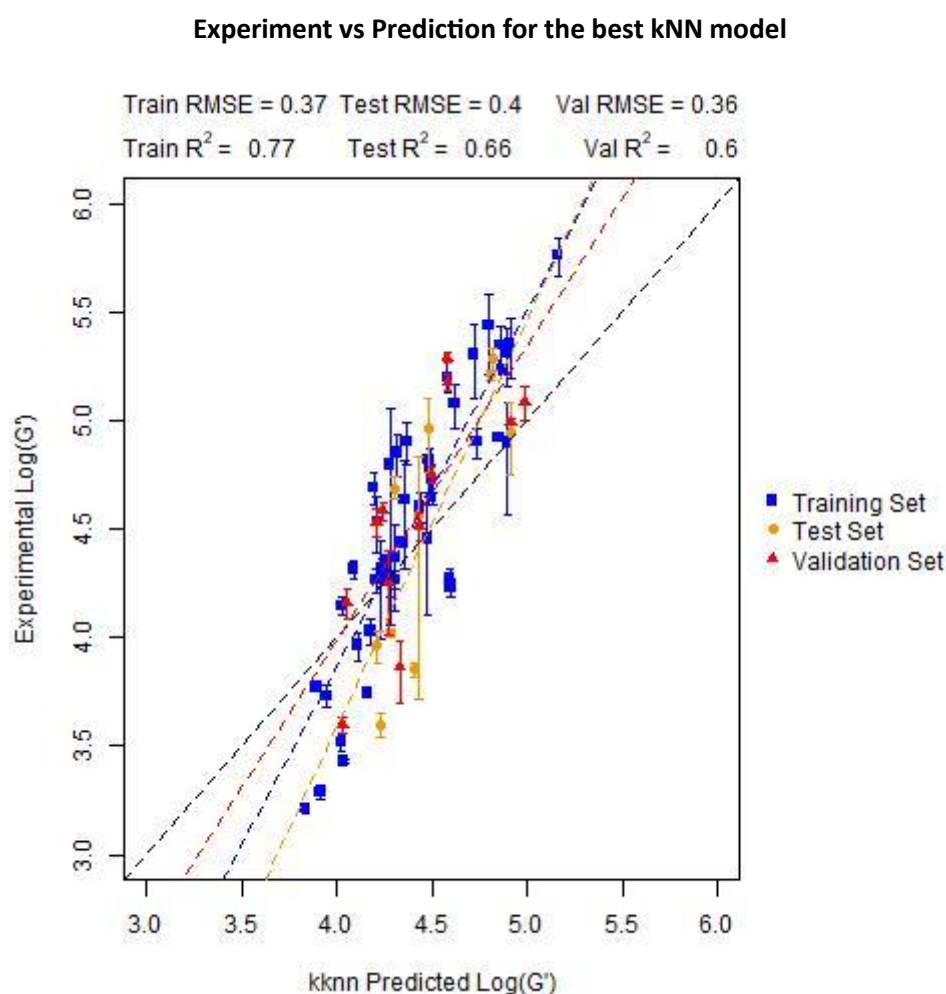


Figure 3.14: Performance of the best 3_2 kNN model (model 1) on the training, test and validation sets with lines of best fit

For the outliers in Figure 3.14, validation set point 4C1NapIF again features as a poorly predicted point further emphasizing the difficulties models have at predicting this molecule. The other two molecules with the highest residuals are the training set point 2NapVVV and the test set point 6MeONapFF. 2NapVVV forms part of the 9 tripeptides that make up the 45 molecules in the training set. This suggests that the training set requires some balancing between dipeptides and tripeptides.

3.3.1.3 PLS model

3.3.1.3.1 Training Performance

Next, we investigated the range of performances using a Partial Least squares model, a more complex model than the BayesGLM and kNN that we have already investigated. Figures 3.15a and 3.15b show a rather varied performance for R^2 even on the training set. Although variation within the 10 repeats is small, the variation between training sets is rather large.

Remarkably, we obtained a range of models that showed poor performance, in terms of R^2 , even on the training set – with some training set R^2 dropping below 0.4. The worst performance during training in terms of R^2 is found in the 4_4 dataset with an R^2 of 0.41 (± 0.00). The RMSE of the 4_4 dataset is also large at 0.46 (± 0.00). The best performing model in terms of R^2 is the 5_3 dataset which has R^2 values of 0.79 for all 10 repeats.

In terms of RMSE, the best performing model is from the 5_3 dataset with an RMSE of 0.27 (± 0.00), confirming it as the best performer during training. The worst performing dataset is the 4_4 dataset with RMSE = 0.46 (± 0.00). The 4_4 dataset performs worst during training for both RMSE and R^2 . Of the models learnt using the 3_2 dataset, all models have good performance in training, with R^2 of 0.75 (± 0.01) and RMSE of 0.30 (± 0.01) continuing the good performance of this dataset.

Performance of the PLS model on the training set in terms of R^2 and RMSE

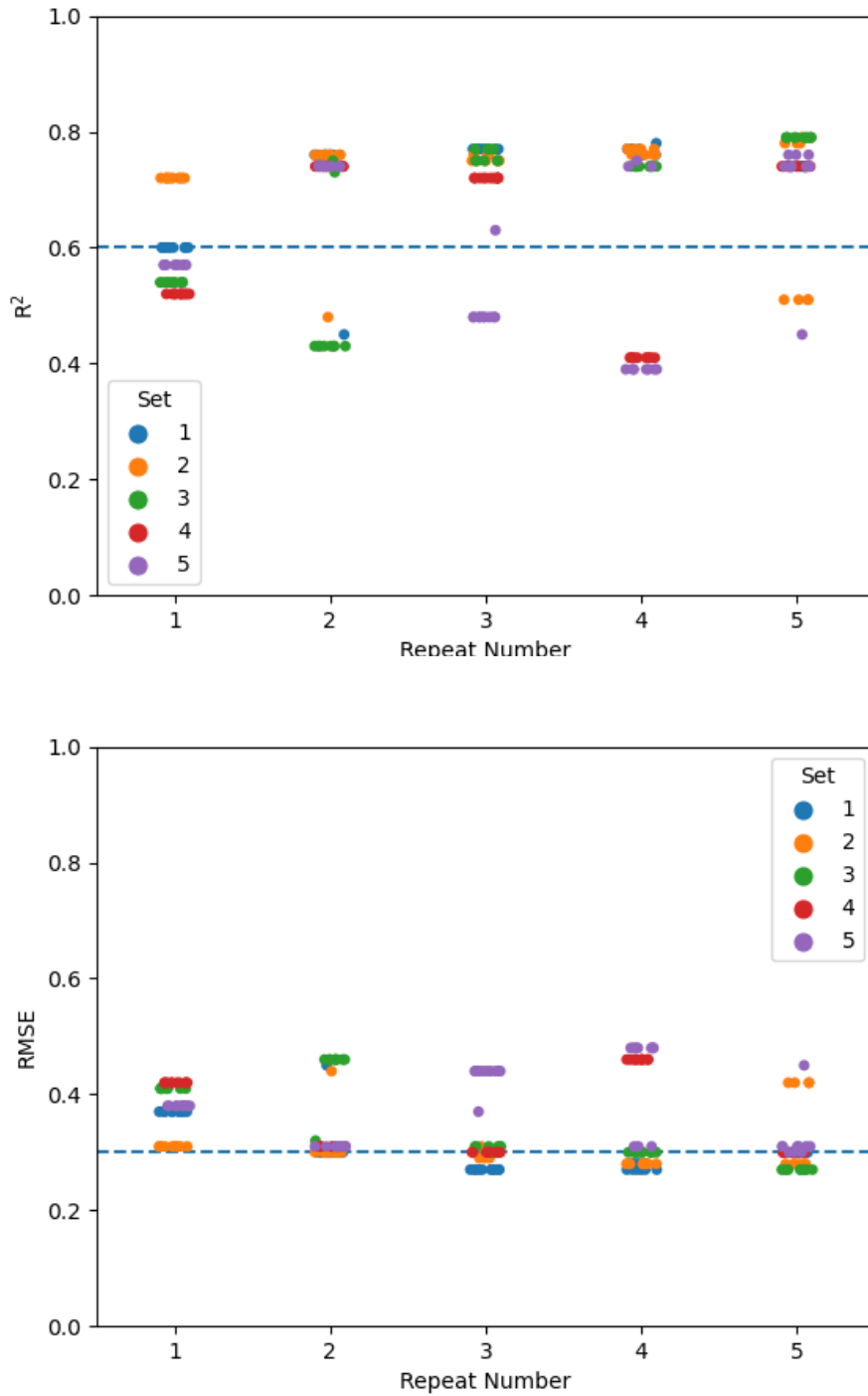


Figure 3.15: Performance of the training sets for the Partial Least Squares Models

3.3.1.3.2 Test Performance

For the test set predictions, Figure 3.16 shows that a range of models pass our R^2 threshold of 0.6 – or at least come close to it. In terms of R^2 , models built using 1_2, 1_4, 3_2, 3_3, and 3_4 all show R^2 values that are at least close to our 0.6 threshold. For the RMSE, the RMSE values for 1_2, 3_2, 3_3 are relatively low at 0.41 (± 0.00), 0.41 (± 0.01) and 0.30 (± 0.02). The R^2 of 0.8 (± 0.00) for the 3_3 model coupled with the RMSE show that it performs best overall on the test set. We will consider the 1_2, 3_2 and 3_3 datasets on the validation set.

3.3.1.3.3 Validation Performance

The performance of the 1_2 models on the validation set (Figure 3.17) is rather poor, with R^2 of 0.37 and RMSE of 0.43. We see comparably better performance on both the 3_2 and 3_3 datasets. For the 3_2 dataset, the RMSE is 0.42 (± 0.01) which is slightly high given our threshold of 0.3. The R^2 of 0.46 (± 0.01) however is low meaning that this model is also poor. However, the performance of the 3_3 dataset is promising with RMSE of 0.46 (± 0.01) being above our threshold but its high R^2 of 0.67 (± 0.00) meaning the 3_3 derived PLS models warrant closer inspection.

Performance of the PLS model on the test set in terms of R^2 and RMSE

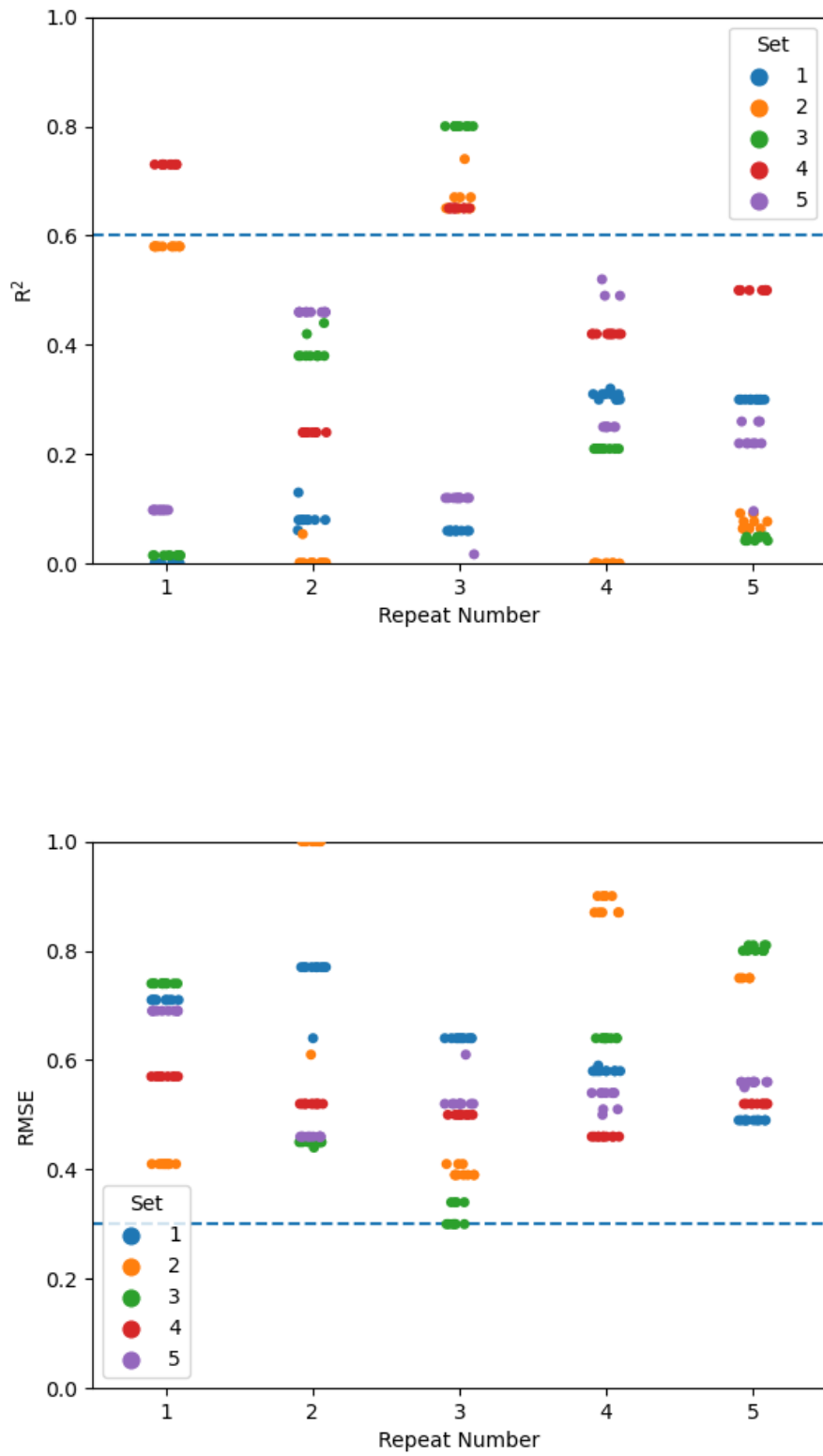


Figure 3.16: Performance for all PLS model datasets on their test set.

Performance of the PLS model on the validation set in terms of R^2 and RMSE

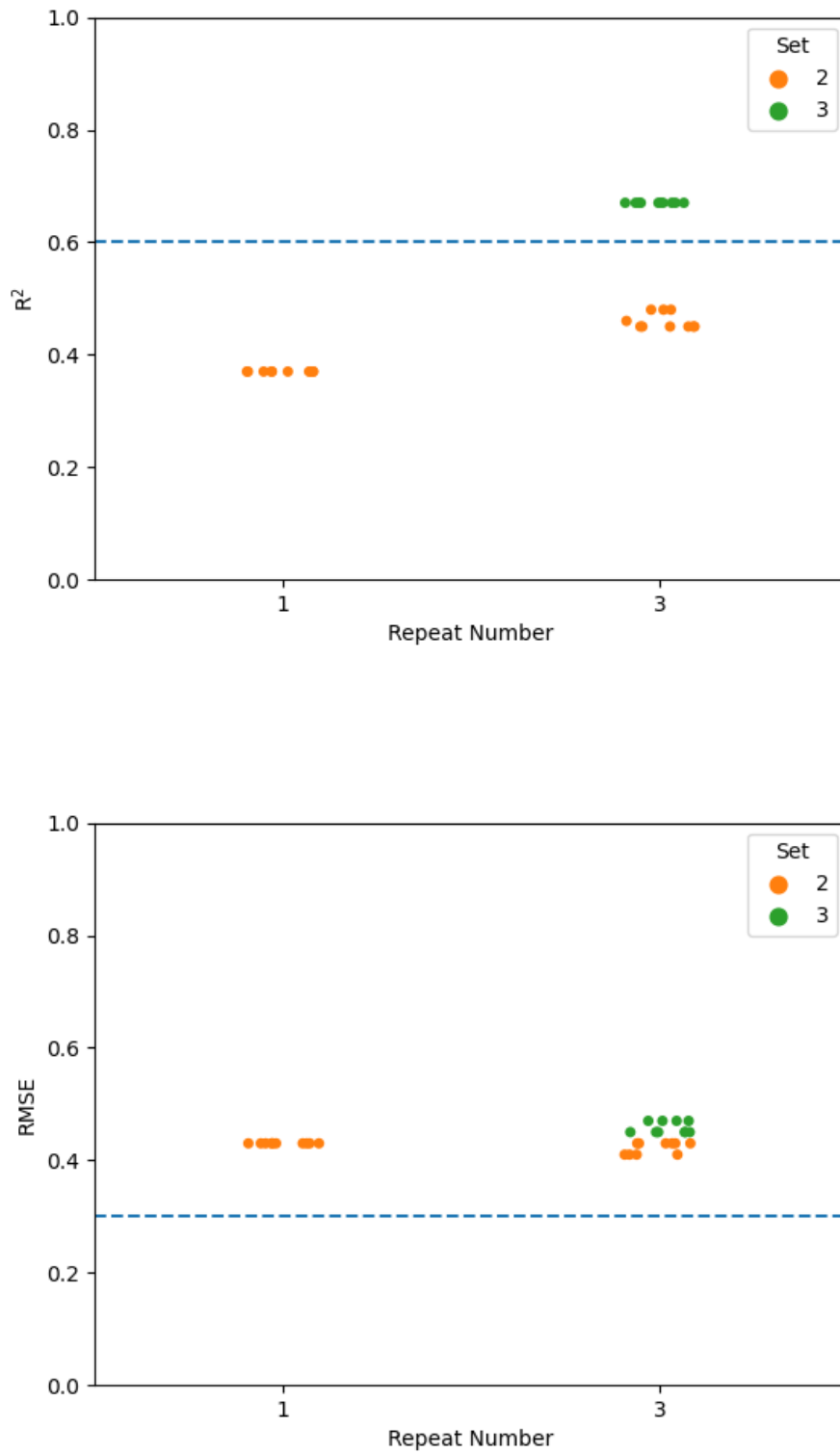


Figure 3.17: Performance of the 1_2, 3_2 and 3_3 datasets on the validation set for the PLS models.

3.3.1.3.4 Best Performing Model

The plot of the PLS model (Figure 3.18) based on dataset 3_3 R^2 shows that although performance metrics indicate a potentially good model, the apparent curvature of points at high predicted G' values suggest that model performance is likely to suffer in future prediction of points in this range. Overall, this model performs satisfactorily even with this caveat.

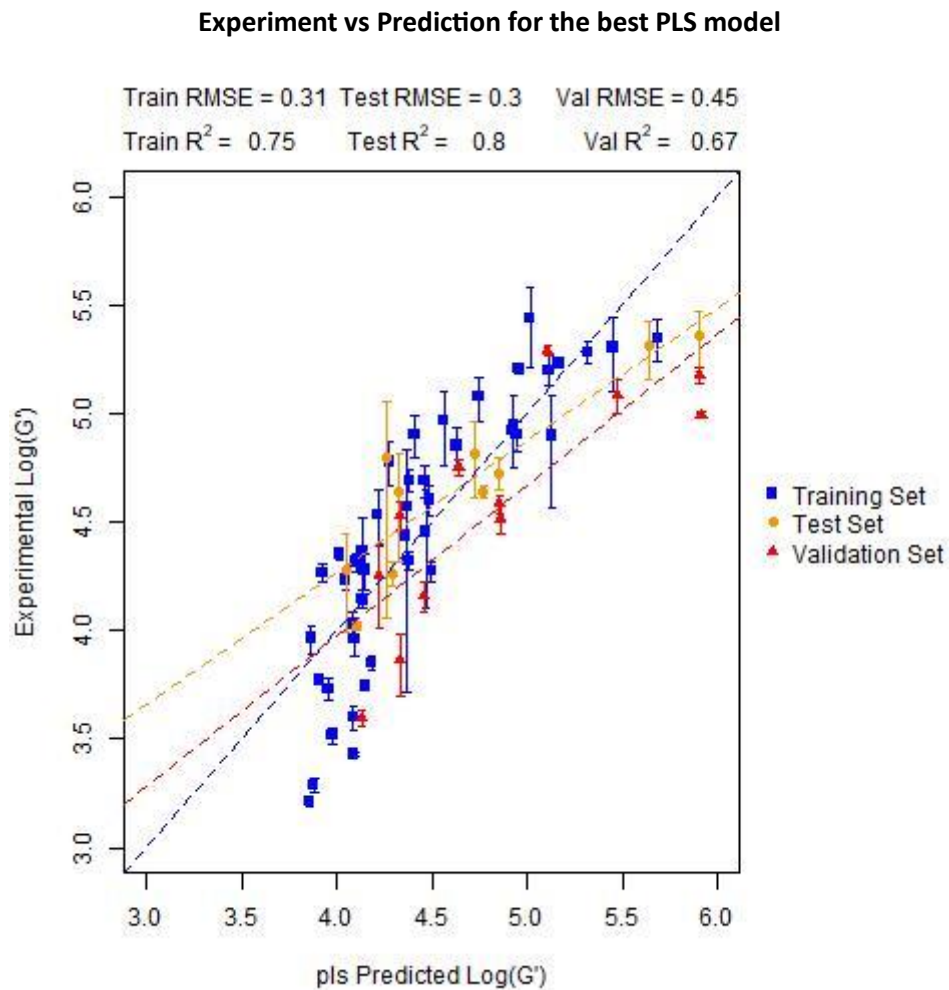


Figure 3.18: Performance of the best 3_3 PLS model (model 4) on the training, test and validation sets with lines of best fit

For Figure 3.18, all three of the molecules with the highest residuals are dipeptides, two validation set points 6MeO2NapFI and FmocFF along with the training set point 1Br2NAPVF are poorly predicted by the model. Given the weighting towards dipeptides in the training set, it is surprising that the model appears to perform poorly here. However, the two validation set points are the points at high $\log(G')$ values where there is a low density of training set points in this range, causing the model to underpredict these values.

3.3.1.4 SVM

3.3.1.4.1 Training Performance

Although performance so far had been adequate, there was definite room for improvement and so our datasets were used to learn SVM regression models. Here, like in the BayesGLM, kNN and PLS models we learn 10 SVM models to gauge an average performance for each training set. Figures 3.19a and 3.19b show the performance of the training set in terms of R^2 and RMSE. The Figures presenting results previously were presented as stripplot as the zero variance within the 10 models built for a training set meant calculation of the density of these performance metrics was impossible. Given the variation in R^2 and RMSE within the 10 models, the plots in Figures 3.19a and 3.19b are shown as a kernel density estimate for their distribution.

Overall, training performance is again good with the lowest R^2 in training of 0.92 (± 0.03) for dataset 1_3 and the highest average training R^2 of for dataset 1_4 of 0.98 (± 0.01). In terms of RMSE, dataset 5_3 produces models with the lowest average RMSE at 0.11 (± 0.04) and dataset 2_5 produces models with the highest average RMSE at 0.18 (± 0.02). Overall, there is little variation in performance during training across the datasets both in terms of R^2 and RMSE.

Performance of the SVM model on the training set in terms of R^2 and RMSE

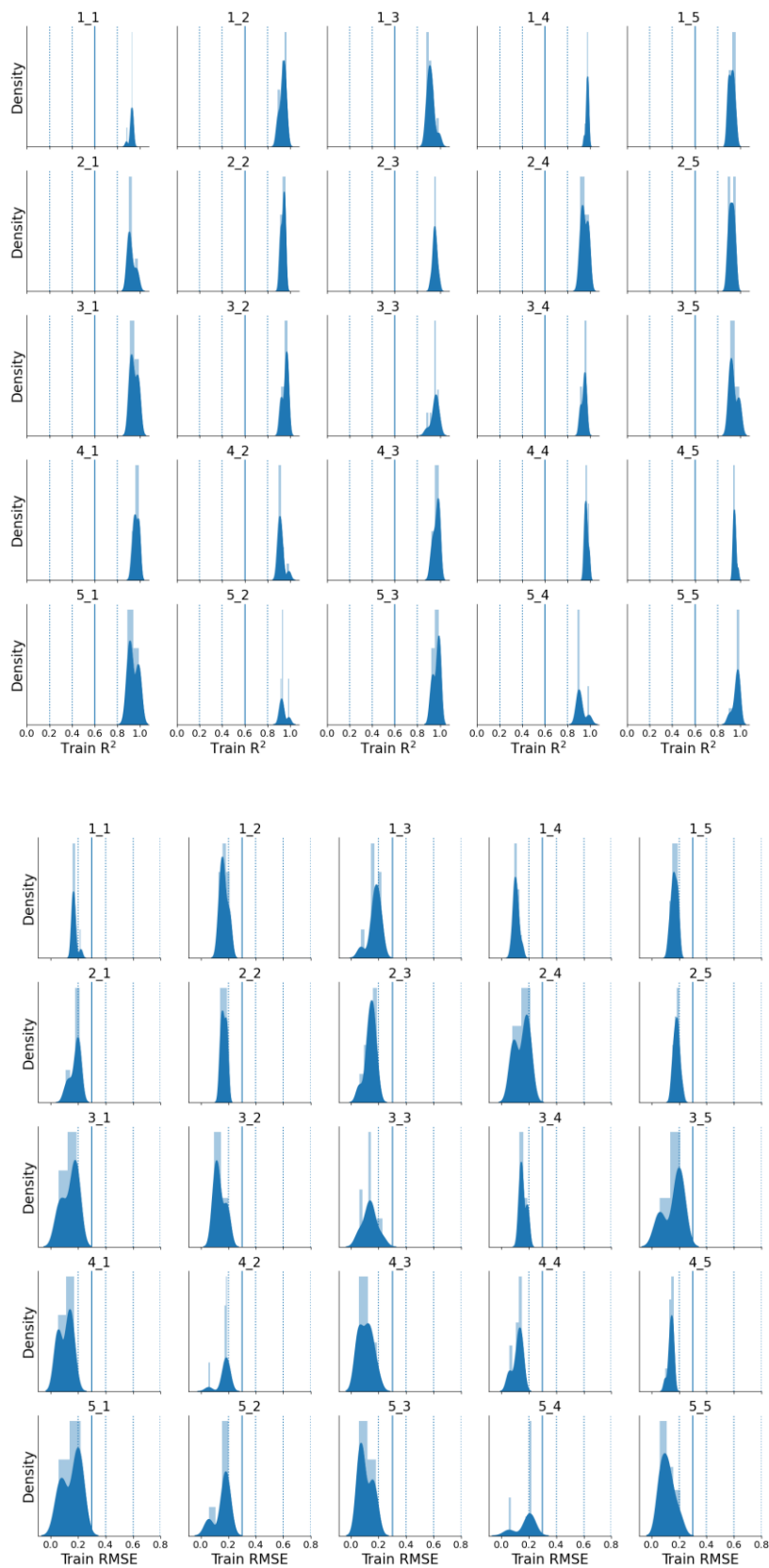


Figure 3.19: Distribution of performance of each SVM model on the training set. The solid vertical line on a) is the threshold for R^2 and in b) the threshold for RMSE.

3.3.1.4.2 Test Performance

Given that there are no issues with training performance, all models were predicted on the test set and the results shown in Figures 3.20a and 3.20b. For our SVM models on the training set, only three of the datasets produce models with an average R^2 above 0.6. These models are 5_1 with R^2 of 0.79 (± 0.02), 1_2 with R^2 of 0.78 (± 0.01) and 4_4 whose R^2 is 0.65 (± 0.06). Model 2_5 has an R^2 of 0.59 (± 0.05) making it borderline and its RMSE will be considered before determining whether the model is good by our thresholds.

During training, 1_4 and 5_3 had the best performing R^2 and RMSE respectively but their performing on the test set is poor. The average R^2 for the datasets on the test set are 0.45 (± 0.02) and 0.41 (± 0.12) respectively. Considering this and the RMSE of 0.55 (± 0.02) and 0.48 (± 0.03) for these datasets suggest that these models overfit on the training data.

For the 1_2, 4_4 and 5_1 datasets that show good R^2 on the training set, none have average RMSE below 0.3 but 5_1 and 1_2 in particular have average RMSE close to 0.3. The average RMSE for the 1_2 dataset is 0.31 (± 0.00) and for the 5_1 dataset it is 0.30 (± 0.02). For the 4_4 dataset the RMSE is a little higher at 0.43 (± 0.03) meaning that it fails to meet the criteria for both R^2 and RMSE.

3.3.1.4.3 Validation Performance

Of the 1_2 and 5_1 models that show promising performance on the test set, the 5_1 models perform much better on their validation set than 1_2 (shown in Figure 3.21). The average R^2 for the 1_2 dataset was 0.31 (± 0.02) compared with 0.68 (± 0.04) for the 5_1 dataset. In terms of RMSE, 1_2 has an average RMSE of 0.46 (± 0.00) and 5_1 has an average RMSE of 0.28 (± 0.01). Overall, 5_1 shows improved performance in terms of R^2 and RMSE over the 1_2 model and therefore we will visually inspect these models.

Performance of the SVM model on the test set in terms of R^2 and RMSE

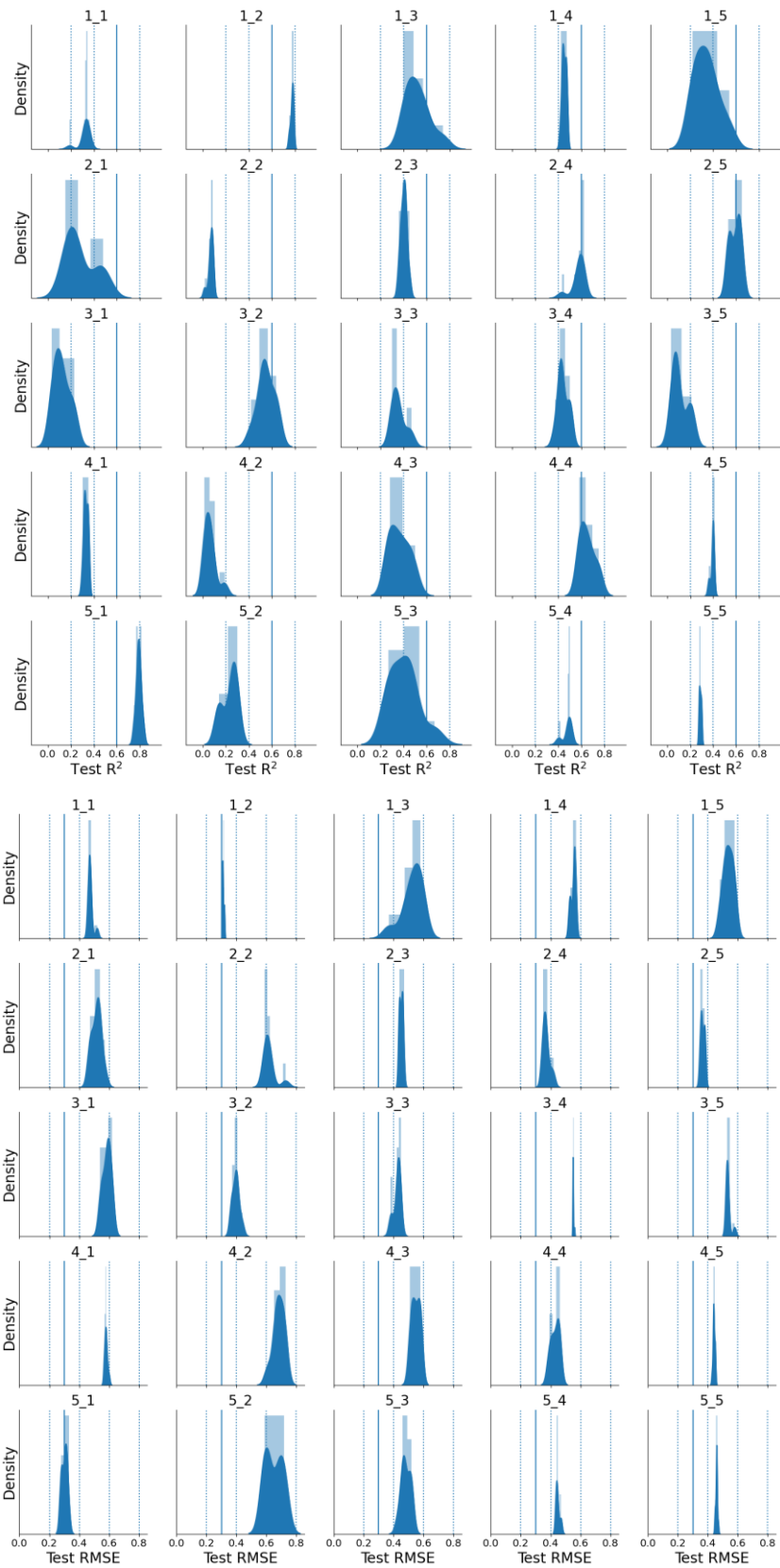


Figure 3.20: Performance of all SVM models on their datasets test set.

Performance of the SVM model on the validation set in terms of R^2 and RMSE

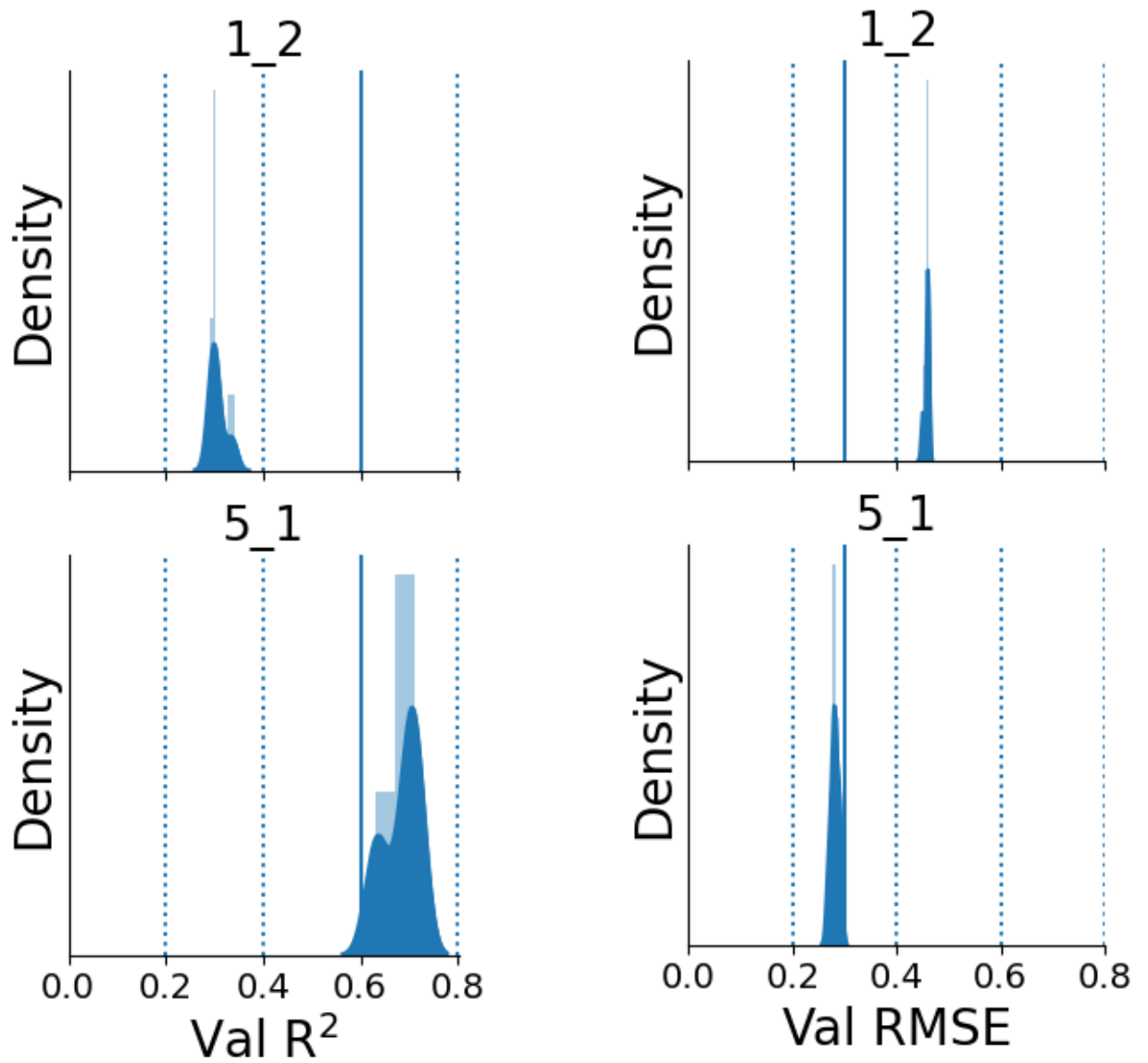


Figure 3.21: Performance of the 1_2 and 5_1 datasets on their validation sets a) R^2 and b) RMSE

3.3.1.4.4 Best Performing Models

Of the 10 models for the 5_1 dataset, we select the model with the lowest RMSE for both the test and validation set (Model 4 out of 10). Visual inspection of the model shows that the good statistical R^2 and RMSE performance is real as the predictions across all three sets are centred around the diagonal line of perfect fit (dashed line in Figure 3.22). Overall, it appears that the SVM model has produced the best model overall thus far but there are still RF models which performed best overall in the classification work (chapter 2.3 – *Results and Discussion*).

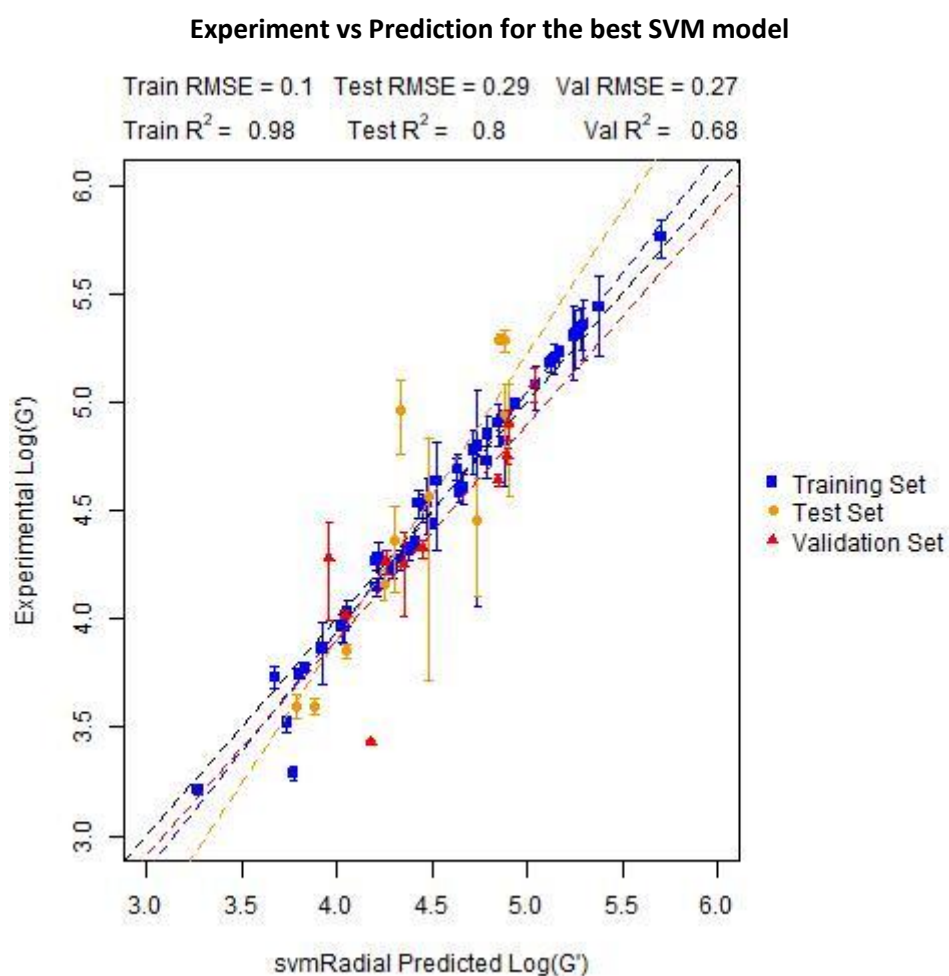


Figure 3.22: SVM Model 4 (out of 10 repeats) that shows the best R^2 and RMSE performance with lines of best fit

CarbA, one of only two protein mimetics in the dataset, is considered one of the biggest outliers in Figure 3.22. The training set point, at low $\log(G')$, performs poorly even as a training set point as it is not only in a region of $\log(G')$ space with low density of molecules, it is dissimilar to the dipeptides and

tripeptides that constitute the rest of the dataset. The remaining molecules that constitute the top 3 points with the highest residuals are validation set point 1Br2NapVF which is again in this low density region of $\log(G')$ space and test set point 6BrNapAV. 6BrNapAV does appear to be a true outlier as the dipeptide class of molecules are well represented in the training set.

3.3.1.5 RF Models

3.3.1.5.1 Training Performance

Having identified a good model using an SVM as well as models for the BayesGLM, kNN and PLS models attention turns to the RF model. Again, as in the SVM work the performance across the train, test and validation sets for R^2 and RMSE are presented as kernel density estimates. Figures 3.23a and 3.23b show the performance during training across all of the datasets. It is worth pointing out in both Figures 3.23a and 3.23b that in some cases, the variance in a performance metric for a dataset was 0 and this makes density estimation impossible. In these cases the data is presented below as a histogram.

Again, training performance is good in terms of R^2 and RMSE across the board. The highest average R^2 during training is 0.94 (± 0.00) for the 3_2 dataset and is lowest at 0.87 (± 0.02) for the 1_3 dataset. In terms of RMSE, it is lowest on average for the 1_1 dataset at 0.17 (± 0.00) and highest for the 5_4 dataset at 0.25 (± 0.03).

Performance of the RF model on the training set in terms of R^2 and RMSE

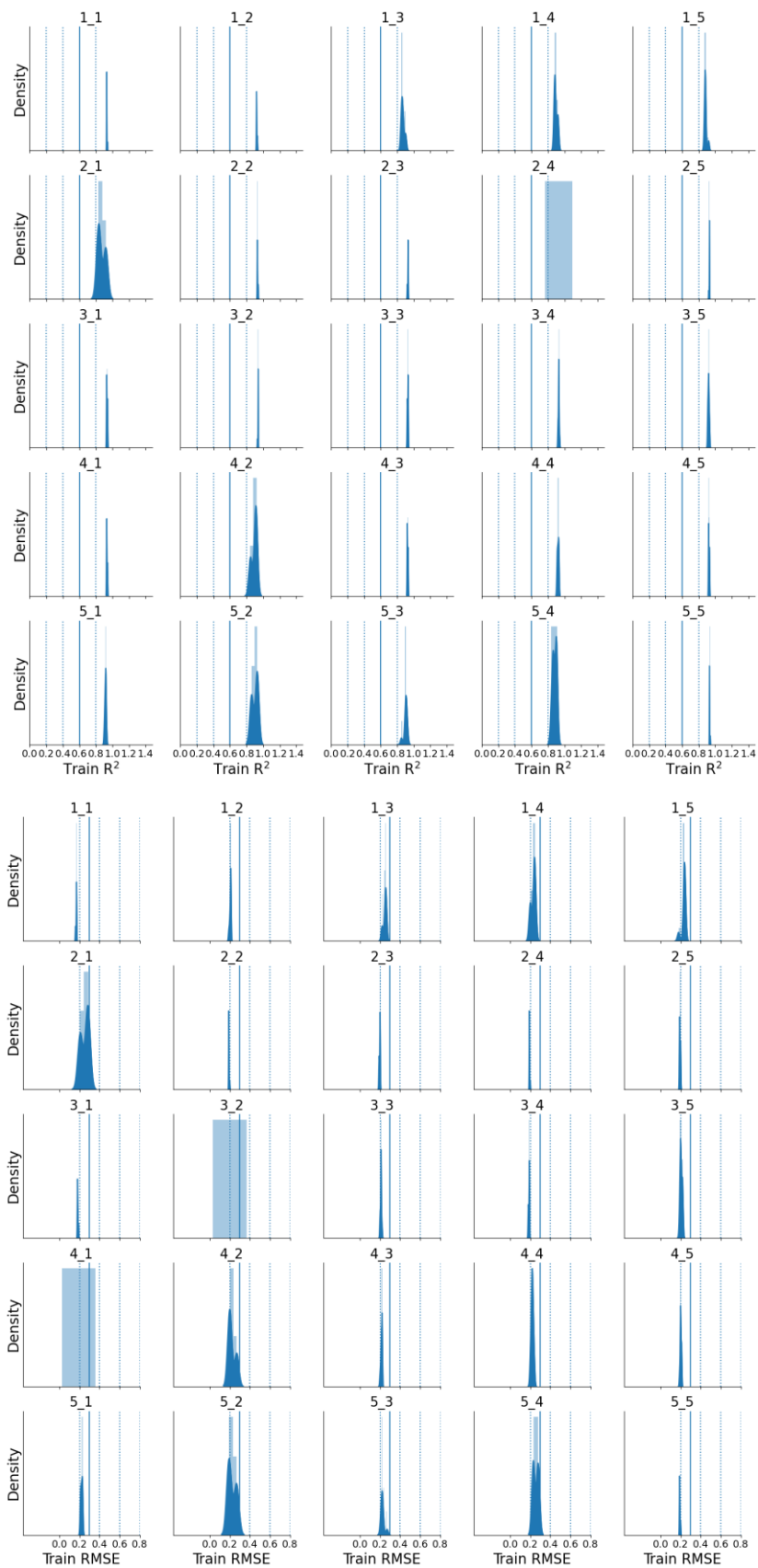


Figure 3.23: Performance of the training sets during RF model training for a) R^2 and b) RMSE

3.3.1.5.2 Test Performance

Figures 3.24a and 3.24b show the performance of all models on their test sets. Considering the R^2 of these models, 5 datasets are on average above the 0.6 threshold and 1 dataset is borderline. The 5 datasets are 1_2, 1_4, 2_5, 3_2 and 5_1. Their average R^2 values are 0.70 (± 0.01), 0.60 (± 0.02), 0.65 (± 0.02), 0.64 (± 0.01) and 0.79 (± 0.02). The borderline dataset is 4_4 with an R^2 average of 0.59 (± 0.02)

Of these 6 datasets, only 1_2, 2_5, 3_2 and 5_1 have average RMSE values around the threshold of 0.3. Their RMSE average values are 0.35 (± 0.01), 0.33 (± 0.01), 0.36 (± 0.01) and 0.35 (± 0.01). Although they are all above our 0.3 threshold they are still close and thus are considered for validation set prediction.

3.3.1.5.3 Validation Performance

In Figures 3.25a and 3.25b, the results on the validation set for the 4 best performing datasets on their test set are shown. For R^2 only the 5_1 dataset passes the threshold with R^2 of 0.69 (± 0.02) with the next best dataset being the 3_2 dataset with an average R^2 of 0.49 (± 0.02). For the 5_1 dataset, the average RMSE is 0.31 (± 0.01) slightly above the threshold but overall shows promising performance across test and validation sets.

Performance of the RF model on the test set in terms of R^2 and RMSE

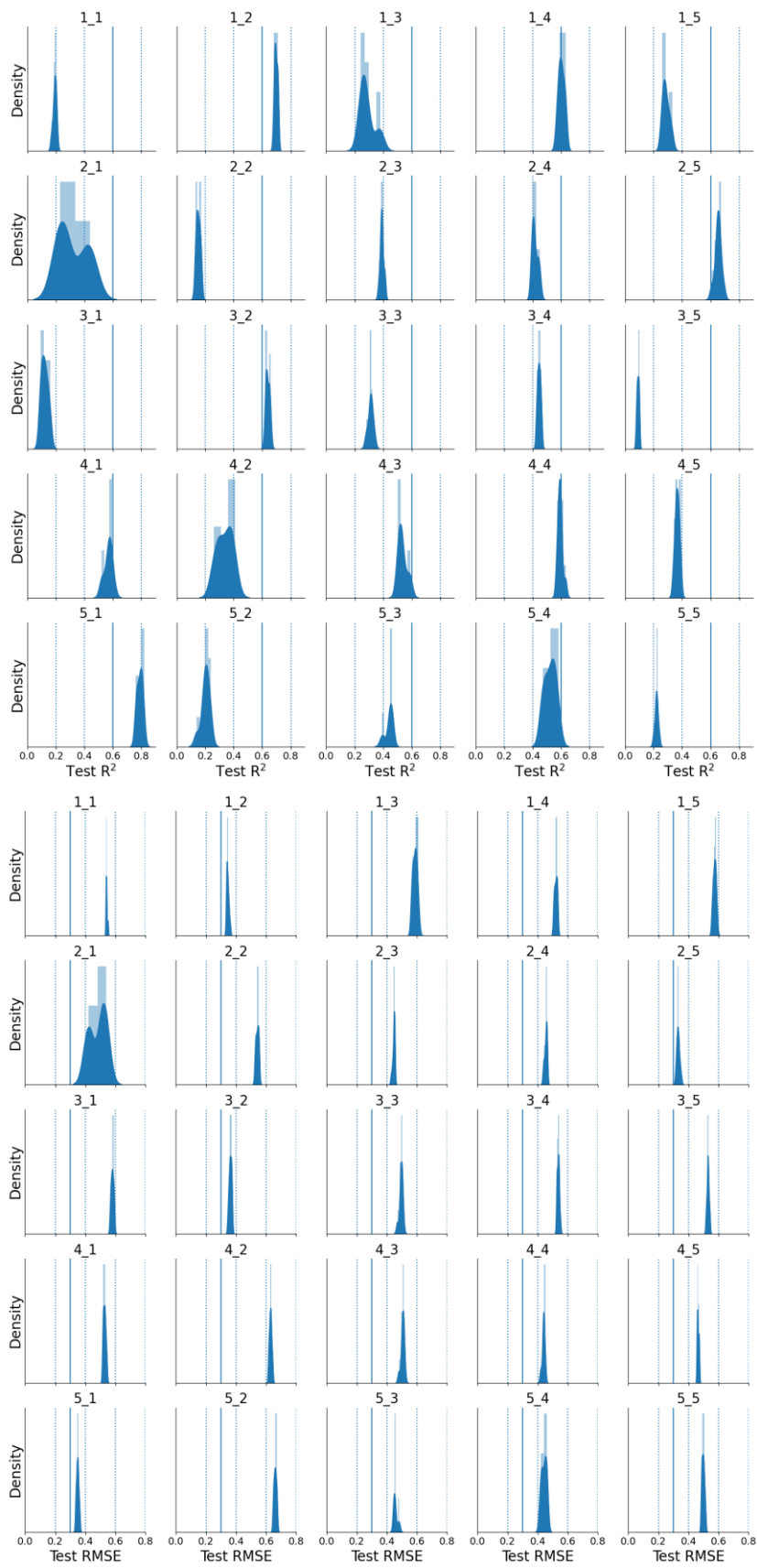


Figure 3.24: Performance of the RF models on their test set in terms of a) R^2 and b) RMSE

Performance of the RF model on the validation set in terms of R^2 and RMSE

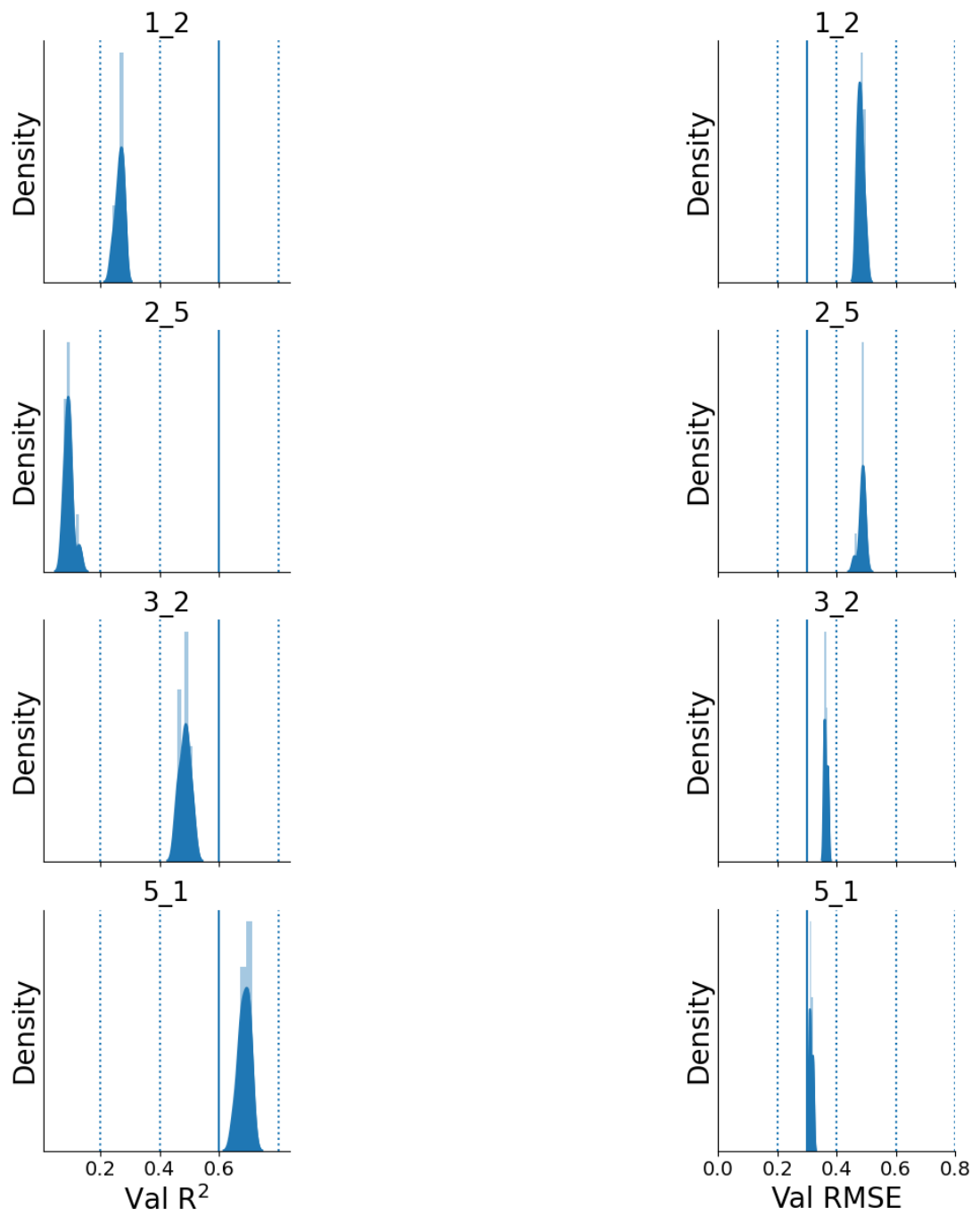


Figure 3.25: Performance of the top datasets on the validation set. Left: R^2 and Right: RMSE

3.3.1.5.4 Best Performing Model

Visual inspection of the RF model built using the 5_1 dataset (Figure 3.26) confirms that this is a good model. We see less overfitting on the training set compared to the SVM model as the difference between the performance metrics (particularly the RMSE) is not as stark.

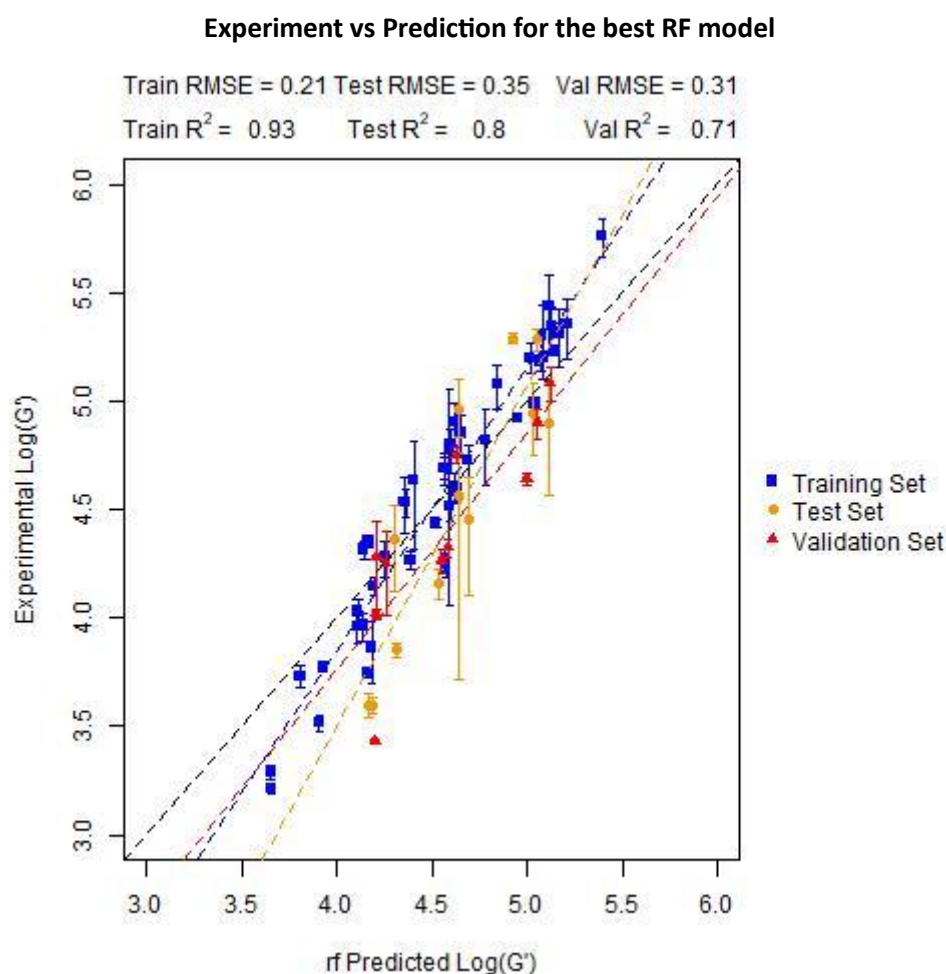


Figure 3.26: RF model (1 of 10) from the 5_1 dataset that shows the best performance across test and validation sets.

Like in the SVM models, the RF model struggles to predict the validation set point 1Br2NapVF. Moreover, the test set point 4Cl1NapIF which struggled in the BayesGLM and kNN models is also an outlier here. Test set point 6MeONapFF is the final outlier in the RF model which was also present in the kNN model as an outlier.

3.3.1.6 Overview of best performing models

Table 3.2 summarises the results of the best models found throughout all modelling algorithms examined. We have found promising models for all algorithms searched, with the RF and SVM models performing best overall with the SVM performing slightly better.

However, even though there are good models, we wish to investigate why some data splits result in good models and others do not. We hope to remove any doubt as to the predictive ability of the models and that there is a rational explanation to the performance measured and we haven't found good models based purely on the split.

Table 3.2: Summary of the best performing model for each algorithm investigated

Algorithm	Dataset	Train		Test		Val	
		R ²	RMSE	R ²	RMSE	R ²	RMSE
BayesGLM	3_2	0.77	0.28	0.68	0.42	0.54	0.40
kNN	3_2	0.82	0.33	0.60	0.41	0.63	0.35
PLS	3_3	0.72	0.30	0.65	0.50	0.62	0.34
SVM	5_1	0.98	0.10	0.80	0.29	0.68	0.27
RF	5_1	0.93	0.21	0.80	0.35	0.71	0.31

3.3.1.7 Applicability domain filtered performance.

The performance of the best models is further investigated through the application of the applicability domain filter to both the test and validation sets (Figure 3.27a and 3.27b). For the BayesGLM and kNN models both built on the 3_2 dataset, applicability domain filtering on the 3_2 dataset reduces the test set from 10 molecules down to 6 and the validation set from 12 to 9 molecules. For the BayesGLM model, the performance of the in-domain test set is good with R² = 0.86 and RMSE of 0.35 showing improvement on the full test set (R² = 0.68 and RMSE = 0.42). For the validation set, the R² is 0.76 and the RMSE is 0.27 both again showing improvement compared to the full validation set (R² = 0.54 and RMSE = 0.40).

On the same 3_2 dataset, the kNN performance on the reduced test set is $R^2 = 0.73$ and RMSE is 0.38 compared with 0.60 and 0.41 for the full test set. For the validation set, the R^2 on the reduced set is 0.59 and the RMSE = 0.28, again showing slightly improved performance in both test and validation.

BayesGLM and kNN experiment vs prediction for the in-domain test and validation sets

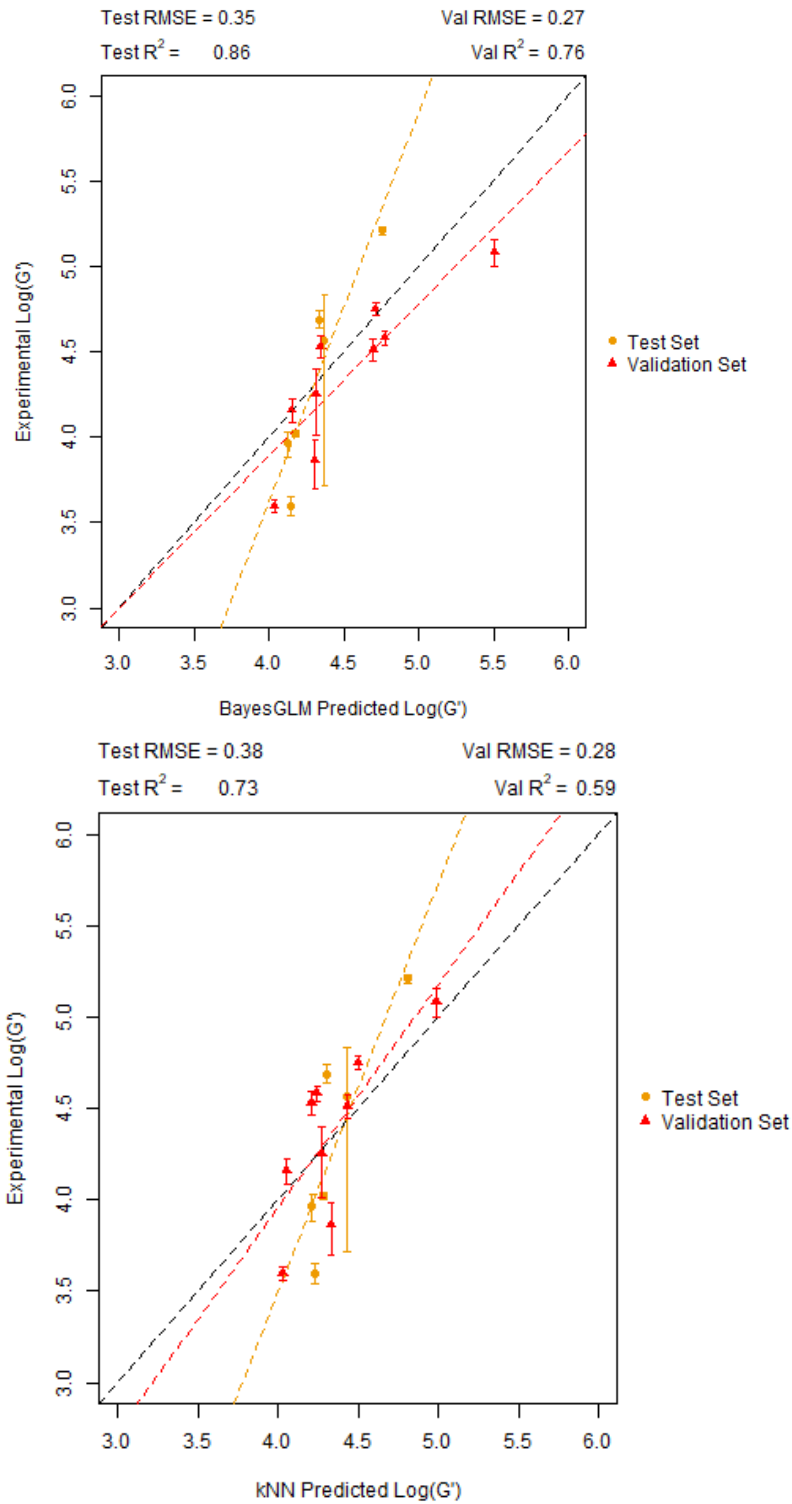


Figure 3.27: Performance of a) BayesGLM and b) kNN models on the range check in domain test and validation set. Both plots built on 3_2 dataset.

For the 3_3 dataset used to build the PLS model, of the 10 test points 2 are out of domain dropping the in-domain test set to 8 points and the validation set has 1 out of domain point reducing the set from 12 to 11 molecules. Figure 3.28 shows the performance of the PLS model on the in-domain test and validation set. For the test set most points are very well predicted but the single point with experimental G' above 5.0 is predicted to have a G' of 6.0 and is quite poorly predicted. Otherwise, the performance of the points is generally good with RMSE of 0.31 close to our threshold and high R^2 of 0.74. On the validation set shows slightly poorer prediction and this is reflected in the higher RMSE of 0.41. The R^2 for this set is still good however at 0.63.

PLS experiment vs prediction for the in-domain test and validation sets

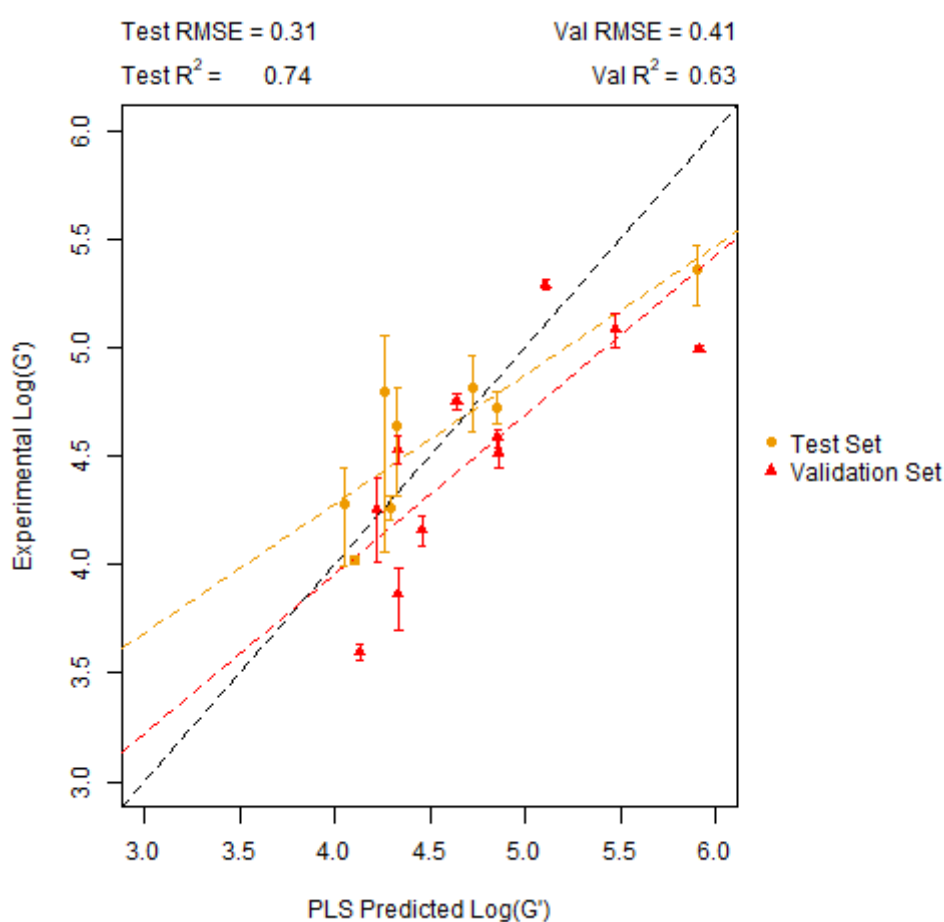


Figure 3.28: Performance of the PLS model on the in-domain test and validation sets.

The final dataset to apply an applicability domain filter to is the 5_1 dataset for the SVM and RF models. For this set, 3 of the test set are out of domain bringing this set down from 12 to 9 and a single point is out of domain for the validation set dropping the number of molecules from this set from 10 to 9.

Performance on the in-domain test and validation sets (Figures 3.29a and 3.29b) is good for the SVM model with the RMSE below 0.3 for both the test and validation set (0.21 and 0.28 respectively). R^2 is also high with R^2 values of 0.89 and 0.67 for each set above our threshold and continuing the strong performance of the SVM model that was seen on the full test and validation sets. As for the RF model, again performance is strong in terms of R^2 with values of 0.81 and 0.73 on the test and validation sets. Although the RMSE is not below the 0.3 threshold for the test (0.34) or validation (0.30) sets, the values are still low and the visual agreement between experiment and prediction is good, although the values at low G' appear to be the worst predictions in the test and validation set.

SVM and RF experiment vs prediction for the in-domain test and validation sets

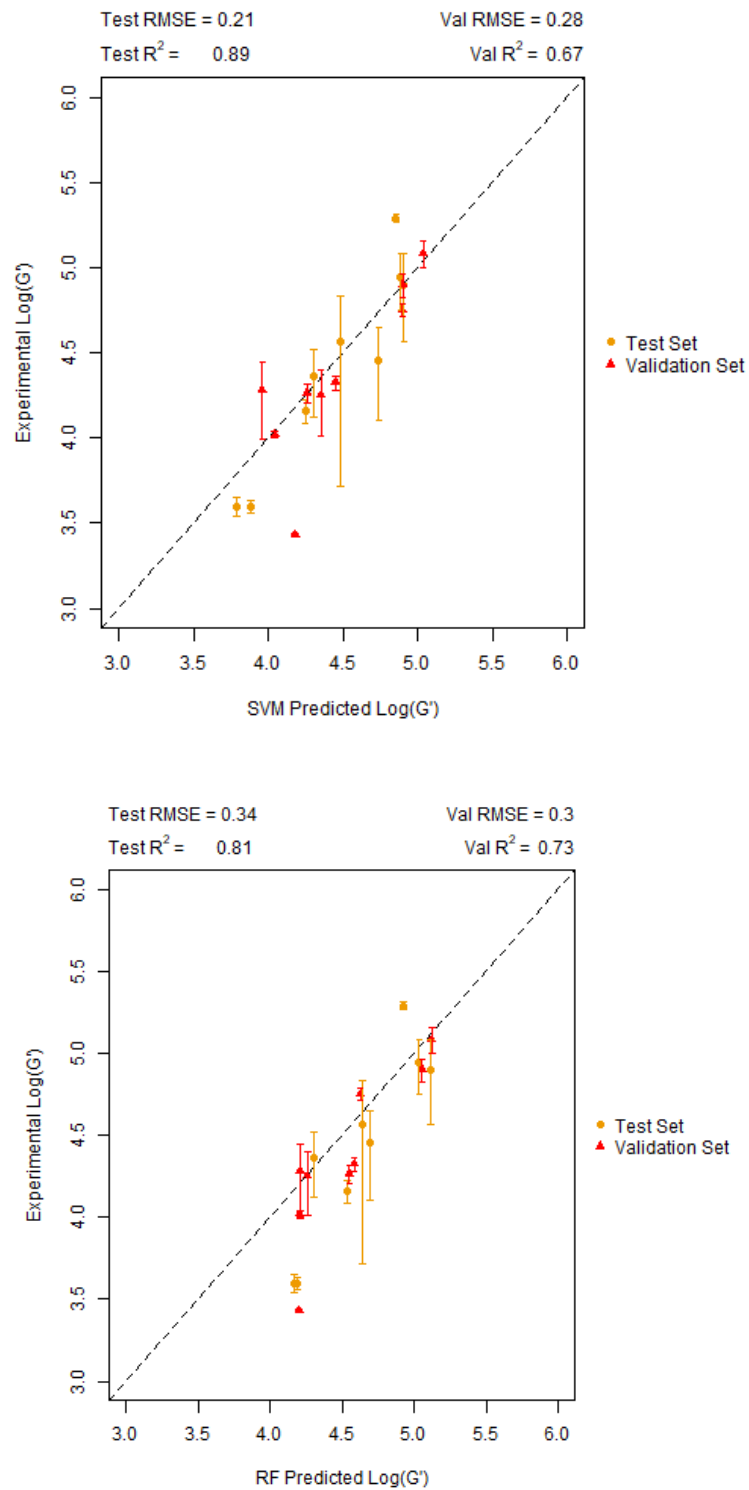


Figure 3.29: Performance of the a) SVM and b) RF models on the in-domain test and validation sets

3.3.1.8 Y-randomisation validation on top performing models

Table 3.3 shows the Z score and corresponding α value of confidence for the performance metrics on the in-domain test set. The α value represents our confidence interval and is the percentage confidence that our true model is not drawn from the same distribution as the random models.¹⁵

From Table 3.3, the α (R^2) and α (RMSE) for all models is generally high. The PLS model built on the 3_3 dataset is the worst performer on the y-randomisation with 16.10% and 21.51% that the true model R^2 and RMSE values are drawn from the same distribution than the y-randomised models. Compare this to the y-randomisation performance for the two best performing models the SVM and RF, which shows improved confidence for both R^2 and RMSE.

Again, the SVM and RF are the best performing models, this time by the α metric for both R^2 and RMSE. The α values for the SVM model are 3.33% for the R^2 and 0.01% for the RMSE giving good confidence that the true SVM model is drawn from the same distribution of models as the y-randomised models. This is also the case for the RF model which has α values of 0.17% for the R^2 and 0.07% for the RMSE, giving this model high confidence that it also finds a real relationship between the descriptors and the G' value.

Table 3.3: Y-randomisation results for the best model for each algorithm.

Algorithm	Dataset	Z-Score (R^2)	α (R^2)	Z-Score (RMSE)	α (RMSE)
BayesGLM	3_2	1.82	3.41%	2.06	1.97%
kNN	3_2	1.15	12.40%	2.14	1.61%
PLS	3_3	0.99	16.10%	0.79	21.51%
SVM	5_1	1.84	3.33%	5.64	0.01%
RF	5_1	2.94	0.17%	3.18	0.07%

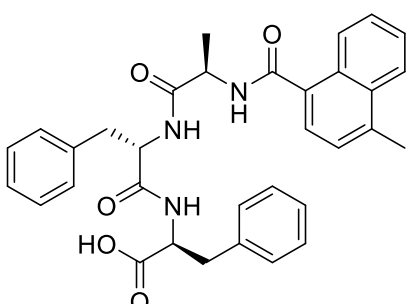
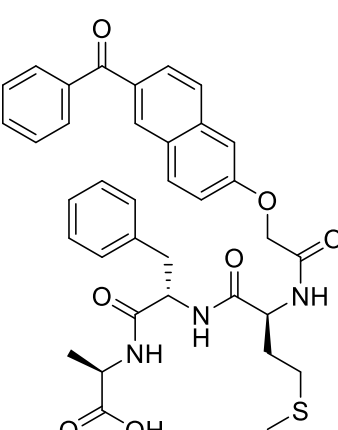
3.3.1.9 External Predictive Performance of Best Models

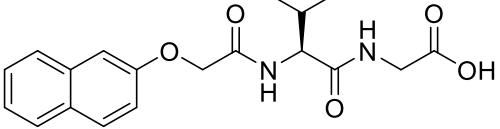
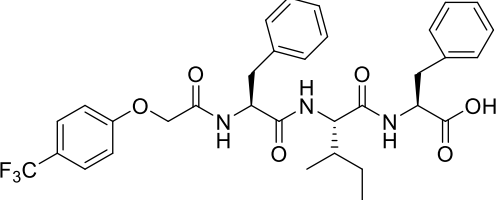
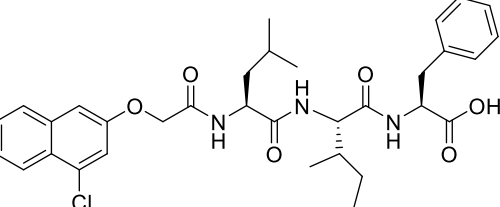
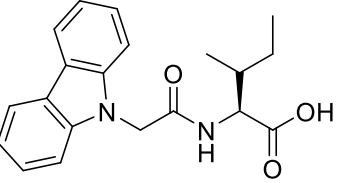
Given that so many data splits have been carried out, as a final validation step, the best performing models are exposed to an external dataset to check that the models are generalizable and do not just capture their test and validation sets well. After the γ -randomisation results, the PLS model from the 3_3 dataset is the weakest of the models and will be excluded from the analysis.

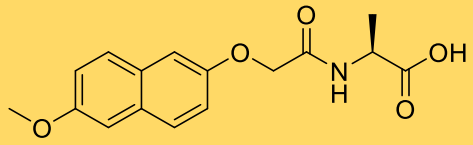
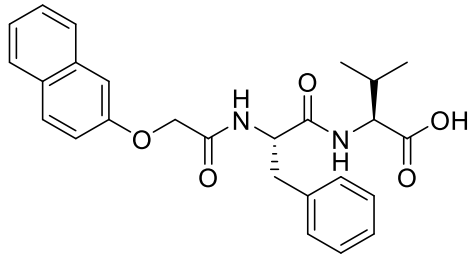
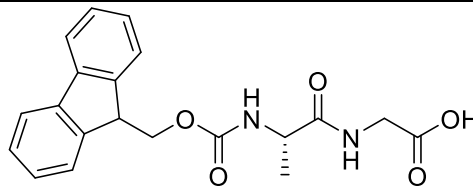
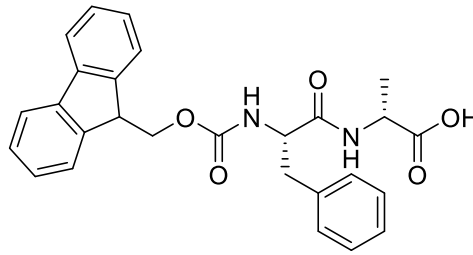
To investigate whether the models are generalizable and are truly predictive, through our collaborators (Professor Dave Adams' group at the University of Glasgow) 10 new molecules were synthesised and their G' values measured. The structures and corresponding G' values can be found in Table 3.4. These molecules are 10 distinct molecules that are separate from the data set that were used to train, test and validate the models described above.

The relationship between the external molecules and the train, test and validation sets for both the 3_2 and 5_1 datasets can be viewed in the PCA plots of the entire dataset (using the same molecular descriptors used for model building) in Figures 3.30a and 3.30b. For the 3_2 dataset, the first two components capture 57% of the variance of the dataset and for the 5_1 dataset it is 55%. Both Figures 3.30a and b show that, at least in the first two principal components, the external points are occupying the same region of space as the train, test and validation sets as no external points fall outside the minimum or maximum PCA1 or PCA2 loading of the training set suggesting that these new molecules aren't too dissimilar to the datasets used to train the model.

Table 3.4: Molecules that form the external set provided with their experimentally derived G' and G'' . The values in brackets are the $\log_{10}(G')$ and $\log_{10}(G'')$ respectively. Gel 7 highlighted in yellow is considered out of domain by both the 3_2 and 5_1 datasets.

Gel	Concentration (mg/mL)	Structure	G'	G' error	G''	G'' error
1 – Molecule 6	5.0		41600 (4.62)	1670	7460 (3.87)	577
2 – Molecule 15	5.0		24500 (4.39)	941	4630 (3.67)	230

3 – 2NapVG	5.0		18200 (4.26)	1040	3630 (3.59)	344
4 - CF3PhFIF	5.0		25700 (4.41)	1800	2770 (3.44)	106
5 - 4ClNapLIF	5.0		21800 (4.34)	1360	3150 (3.50)	182
6 - Carbl	5.0		12300 (4.10)	983	454 (2.66)	39

7 - 6MeONapA	5.0		3750 (3.57)	116	187 (2.27)	32
8 - 2NapFV	5.0		33600 (3.53)	4040	3460 (3.54)	326
9 - FmocAG	5.0		5520 (3.74)	705	966 (2.98)	87
10 - FmocFA	5.0		17200 (4.24)	2970	2590 (3.41)	621

PCA visualisations of the 3_2 and 5_1 with the external set of molecules

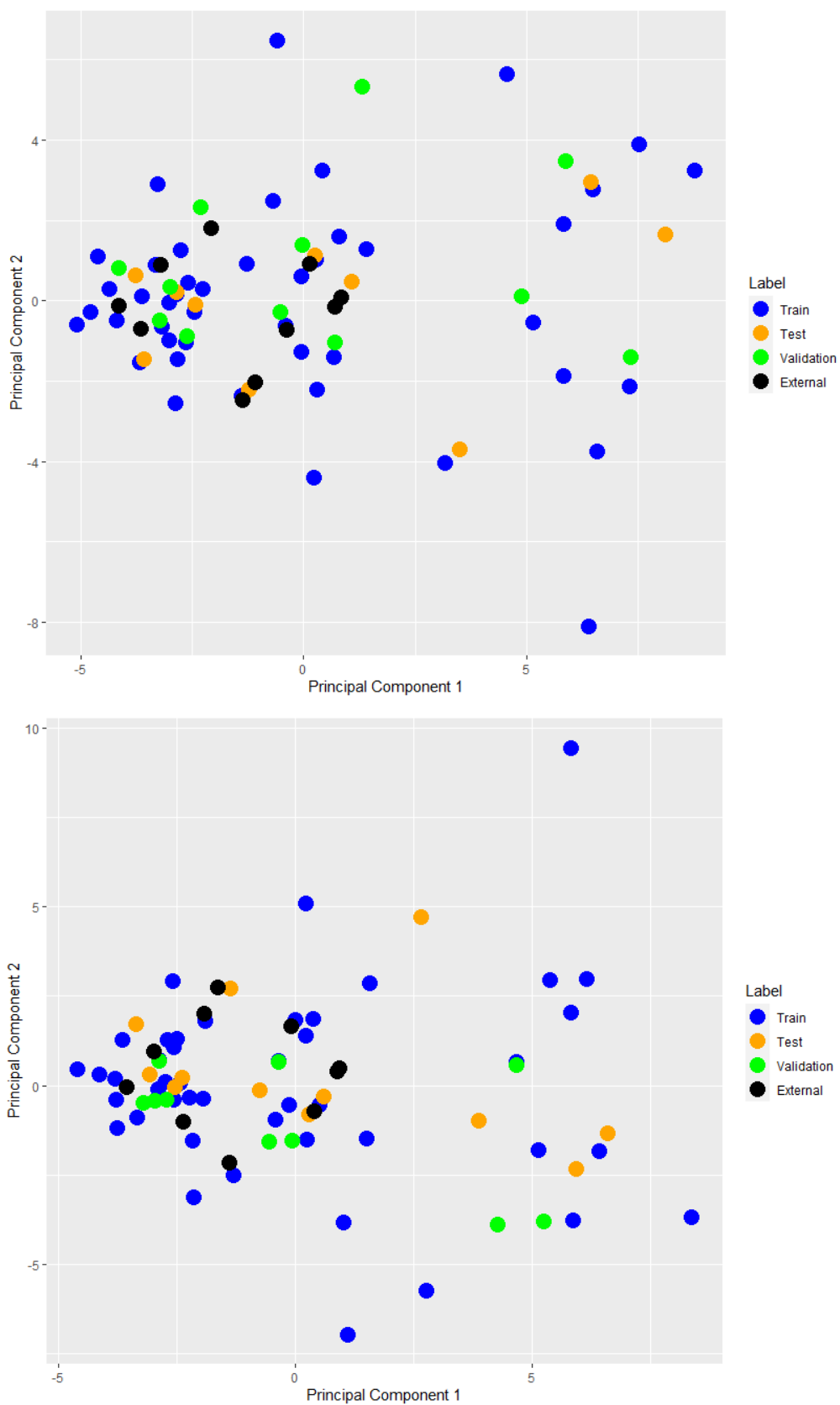


Figure 3.30: a) PCA visualisation of the 3_2 dataset with the new external plots and b) PCA plot for the 5_1 dataset.

We predicted the G' values for the best performing BayesGLM, kNN, SVM and RF models and the plot of the predictions agreement with experiment is shown in Figure 3.31a-d. From the plots in Figure 3.31 no models show any visual agreement between experiment and prediction. In terms of R^2 all models show low R^2 at BayesGLM = 0.04, kNN = 0.00, RF = 0.00 and SVM = 0.04. Clearly all of these are well below the 0.6 threshold for our models.

However, these models should not be discarded. RMSE values are admittedly very high for the BayesGLM and SVM (0.64 and 0.62) way above the 0.3 threshold used throughout this chapter. However, for the kNN and RF models, the RMSE values are much closer to the 0.3 threshold at 0.41 and 0.49.

Furthermore, the point highlighted in red is considered out-of-domain by all models (highlighted in Table 3.4) and so removing this point and re-calculating the RMSE on only in-domain points sees the RMSE improve from 0.41 to 0.36 for the kNN model and from 0.49 to 0.45 for the RF model. There is no obvious reason to state why this molecule is chemically different from any present in the training set.

Although both are still above the threshold, the visual agreement and the presence of a single outlier which is significantly influencing the RMSE for the models. In terms of the RMSE, removal of this point improves the RMSE for the kNN from 0.36 to 0.23 and for the RF from 0.45 to 0.31 showing much better agreement.

However, looking at the range of G' values in the external set, there is not full coverage of the training G' range due to lack of experimental time due to COVID-19. This is exacerbated at low G' for the out-of-domain point and the prediction outlier. However, with this caveat, we investigate the kNN and RF models due to their performance on the test and validation sets, and their interpretability to ascertain what is underpinning the model's predictions.

Experiment vs prediction for each best model on the external set

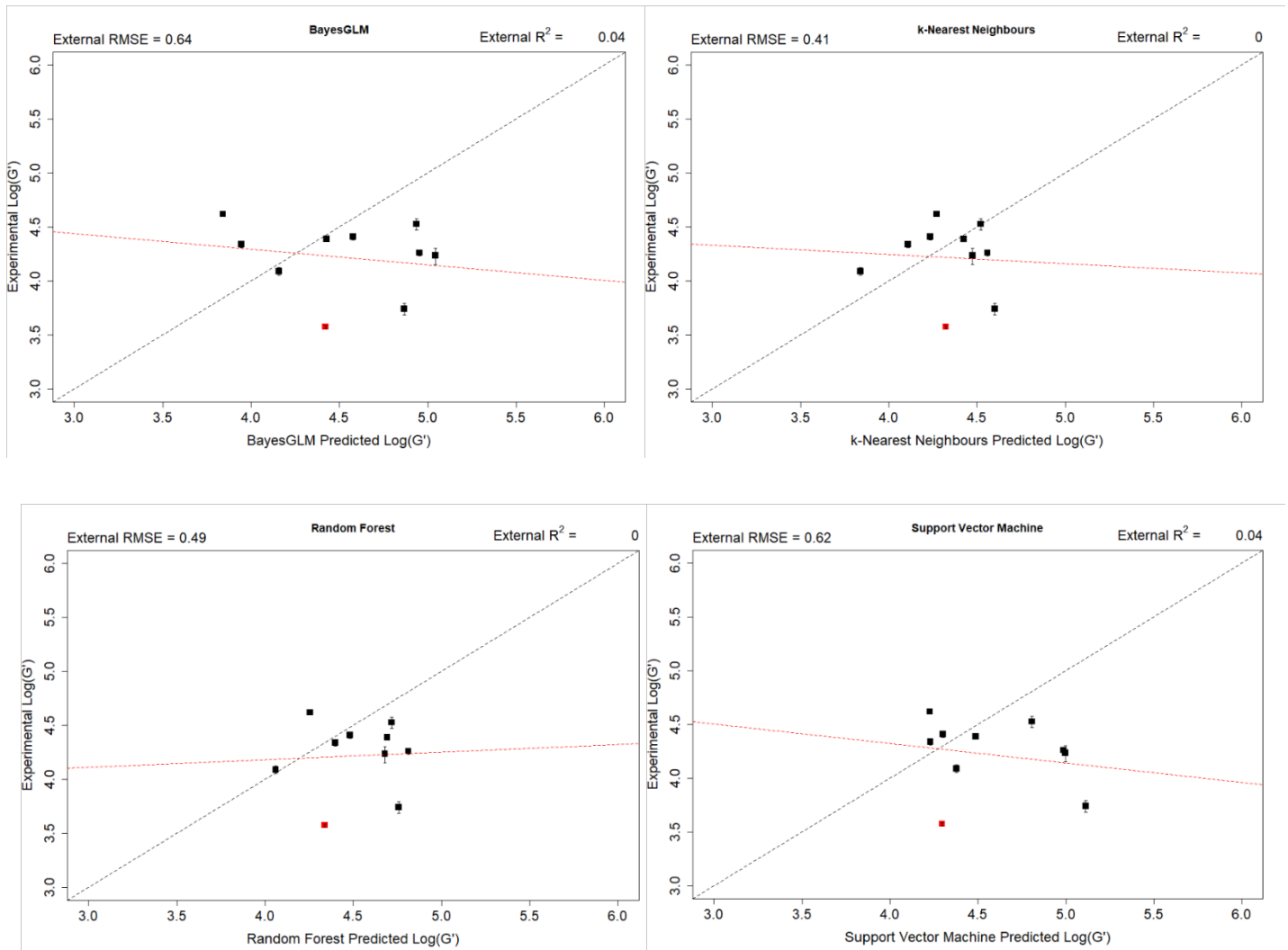


Figure 3.31a-d Performance of the bayesglm, kNN, RF and SVM models on the full external set. Point highlighted in red is considered out of domain (highlighted in yellow in table 4). The red line of best fit is the best fit for all 10 points.

3.3.1.10 Variable importance

3.3.1.10.1 k-Nearest Neighbours

Given the relatively good performance of our kNN model, we used a SHAP approach which considers coalitions of different descriptors and assesses the difference in the coalitions predicted value and the value predicted by the full descriptor set. The difference is then assigned to the members of the coalition and is averaged over numerous coalitions to give an idea of descriptor importance (see *Chapter 1.11 – Model Interpretation*). SHAP was used to investigate the molecular descriptors in the combined in-domain test and validation sets to ascertain which are important in generating our good predictions.

The SHAP results are again presented as a beeswarm plot, where the descriptors are presented in order of their mean absolute SHAP value – meaning more important points are presented first regardless of whether this importance has a positive or negative influence on the prediction. Each point in a row of a beeswarm plot is a single molecule and is positioned along the x axis by the descriptor contribution for that molecule. Each point is coloured by the normalised value of the descriptor between 0 and 1.

Given that we have molecules present in the dataset at 2.5, 5 and 10 mg/mL the beeswarm will be spliced into different concentration plots so that the comparisons are strictly between molecules at the same concentration. However, when considering the in-domain test and validation sets, as there are only 2 in-domain molecules at 10mg/mL which would make drawing conclusions here difficult. Therefore, only 2.5 and 5 mg/mL molecules are considered in the analysis. The beeswarm plot is shown in Figures 3.32a and 3.32b with the corresponding fragments shown highlighted in red on training set molecules in Figure 3.33.

The most important fragment is the ECFC_4.717474525 ([*]:[cH]:[c]1:[cH]:[cH]:[cH]:[*]:[c]:1:[*]) fragment which corresponds to an aromatic substituted ring that maps to both naphthalene rings separately. Interestingly, the ring differentiates between substitutions on the benzene rings. If the naphthalene is functionalized, the bit is only set if the naphthalene is substituted on the non-terminal benzene ring. If the non-terminal benzene in the naphthalene is substituted para, then the ring is also not set. If the ring is substituted ortho then the bit is set to a count of 1 and if the ring is unsubstituted the bit is set to a count of 2.

Beeswarm plot of the SHAP importance for the kNN model

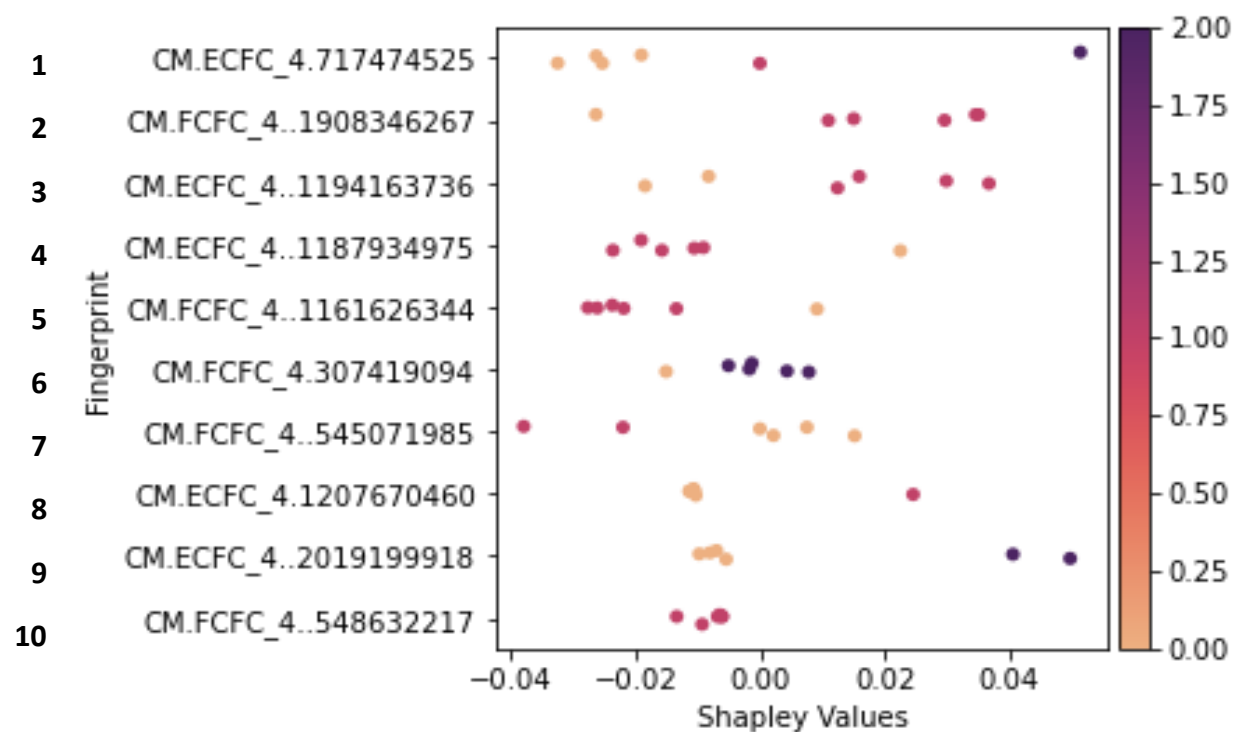
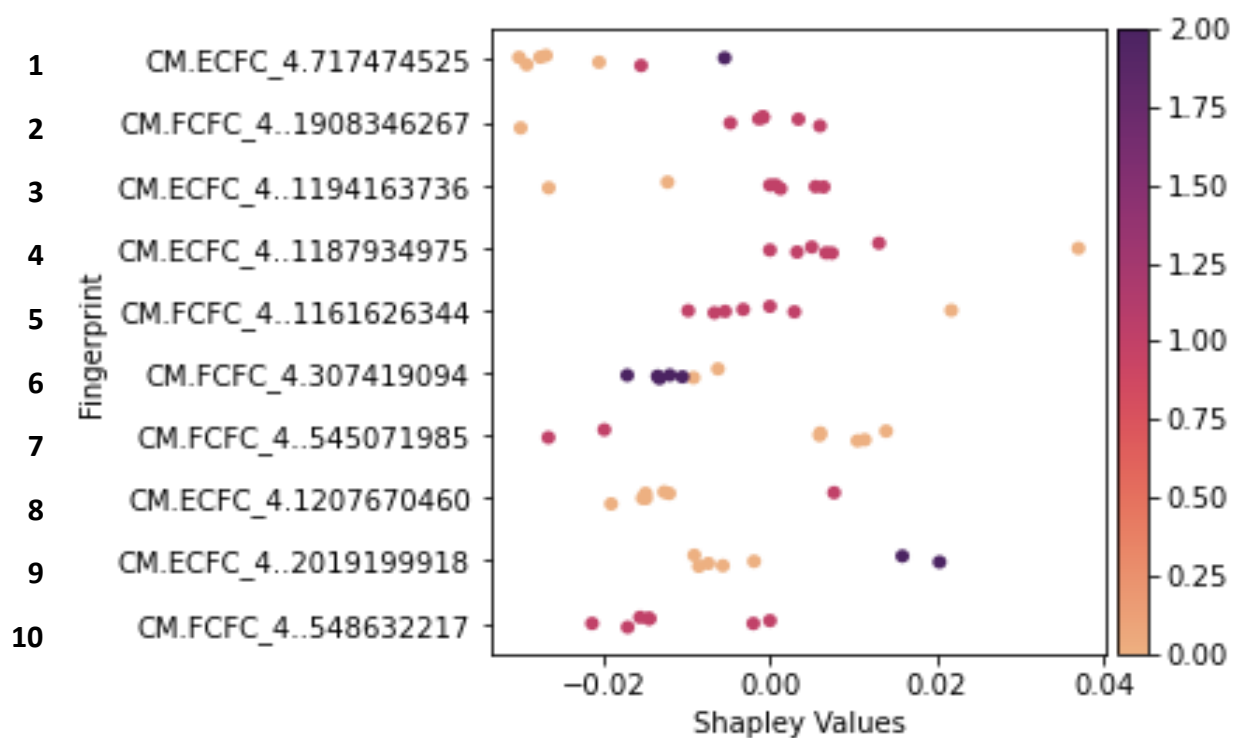
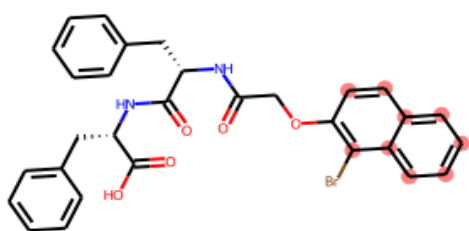
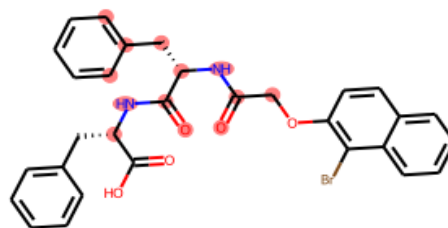


Figure 3.32: SHAP Beeswarm plots for the in-domain test and domain points a) molecules at 2.5 mg/mL and b) Molecules at 5 mg/mL

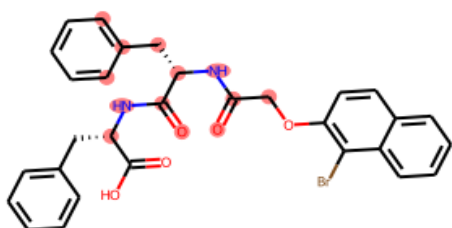
Chemical depictions of the most important fingerprints



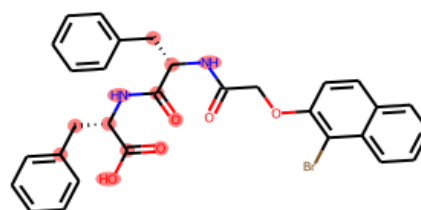
1 – ECFC_4.717474525



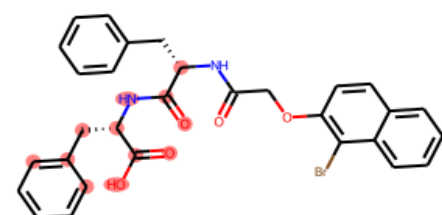
2 – FCFC_4.-1908346267



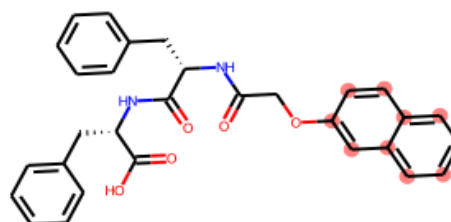
3 – ECFC_4.-1194163736



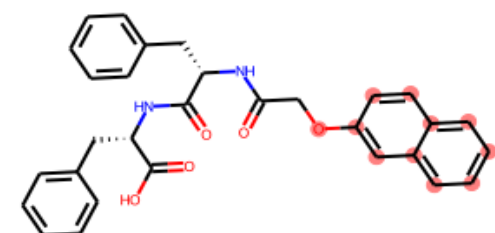
4 – ECFC_4.-1187934975



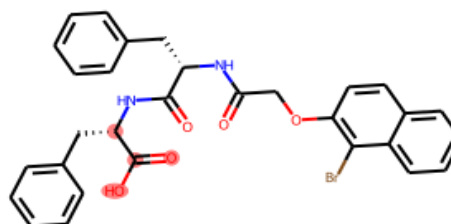
5 – FCFC_4.-1161626344



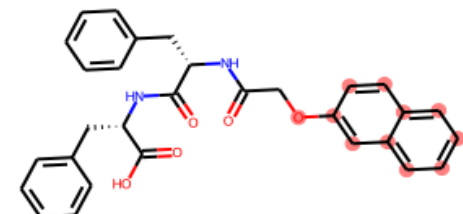
6 – FCFC_4.307419094



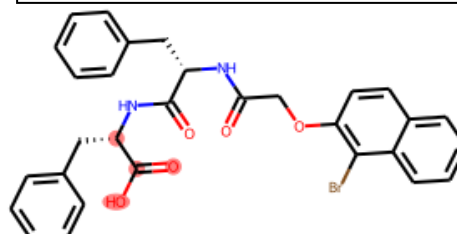
7 – FCFC_4.-545071985



8 – ECFC_4.1207670460



9 – ECFC_4.-2019199918



10 – FCFC_4.-548632217

Figure 3.33: Fragments in the top 10 most important kNN descriptors highlighted on a training set molecule. Note that the fragments are promiscuous and can be set by other similar fragments that are not shown here.

This implies that the G' is generally predicted to be larger if the naphthalene is unsubstituted. Substitution of the naphthalene ring is used as a tool to increase the hydrophobicity of the molecule to improve gelation ability of molecules.¹⁶ Since the presence of halogens on the naphthalene are supposed to improve gelation by promoting self-assembly, it is surprising that SHAP assigns a negative attribution to the substituted fragment. However, this fragment isn't fully characterizing naphthalene and so it may be other factors at play here.

Moreover, in the top 10 the 6th (FCFC_4.307419094), 7th (FCFC_4.-545071985) and 9th (ECFC_4.-2019199918) most important fragments are also naphthalene-based fragments. The FCFC_4.307419094 fragment ([*][c](:[*]):[c](:[cH]:[*]):[c](:[*]):[*]) shows no correlation with value but there are contradictory correlations with 7th and 9th. The 7th most important fragment FCFC_4.-545071985 ([*][c]1:[*]:[cH]:[c]2:[cH]:[*]:[cH]:[cH]:[c]:2:[c]:1Br) is only set by ortho-substituted naphthalene and shows that this fragment again is causing a decrease in the predicted G' as the presence of this fragment increases. However, the 9th most important fragment ECFC_4.-2019199918 ([*][c]1:[*]:[cH]:[c]2:[cH]:[*]:[cH]:[cH]:[c]:2:[cH]:1) shows the opposite trend and is set to 2 in our dataset by methoxy substitution on either ring and bromo substitution on the terminal naphthalene. The bit can also be set to 1 by the unsubstituted naphthalene.

This suggests that the model is placing an increased preference on the bromo substitution with the SHAP analysis. Moreover, although the most important fragment (ECFC_4.717474525) is set by naphthalene – since it is set to 2 the fragment only describes each ring in naphthalene whereas the bit here (ECFC_4.-2019199918) sets unsubstituted naphthalene to 1 this fragment describes the naphthalene in full. Therefore, the relationship between SHAP value and fragment count follows what would be expected and has been posited by Adams *et al.*¹⁶ the fact that the bromo substituted naphthalene gel (Figure 3.34) exhibits a stiffer rheology than the non-bromo substituted version could be due to this hydrophobic effect. This increase in rheology could also be driven by electron withdrawing groups stabilizing the π - π stacking¹⁷ and explains the preference for methoxy or bromo substitution over the unsubstituted naphthalene

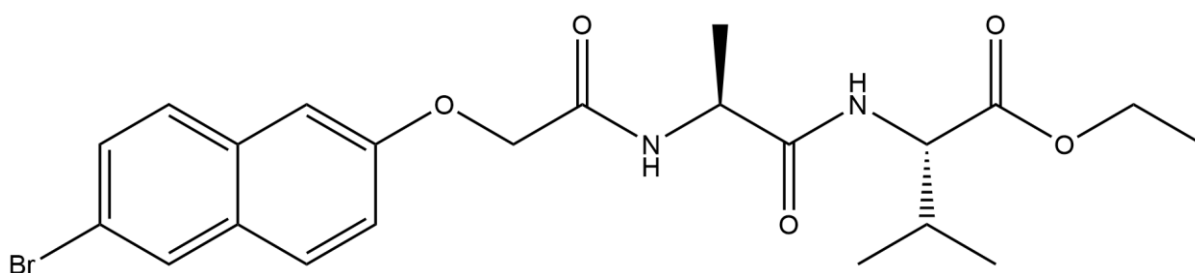


Figure 3.34: Functionalised naphthalene with bromo naphthalene substitution that shows stiffer mechanical properties than the non-brominated version.

The remaining fragments (except fragment 10 (FCFC_4.-548632217) which show no correlation with value) all contain phenylalanine fragments. In Chapter 2, the SHAP analysis investigated the importance to self-assembly the phenylalanine amino acid is (see Chapter 2. 2.3.2.7.2.1 RF Shap Importance). The 2nd, 3rd, 4th and 5th most important fragments in the kNN SHAP analysis can be mapped onto phenylalanine. However, the 2nd and 3rd fragments have a positive relationship and show that an increase in these fragments results in predictions of stiffer gels. For the 4th and 5th, the opposite trend is seen.

The 2nd (FCFC_4.-1908346267 ([*]NC(=O)[C@H](C[c](:[*]):[*])NC(=[*])[*])) and 3rd (ECFC_4.-1194163736 ([*]C[C@H](NC(=O)C[*])C(=[*])[*])) fragments are both set by phenylalanine with the 2nd fragment set by phenylalanine as the amino acid closest to the N-terminus and the 3rd fragment only set if that N-terminus is naphthalene. In both cases, the phenylalanine moiety shows an increase in SHAP attribution as the fragments increase in number. This can likely be attributed to the phenyl ring increasing hydrophobicity for increased self-assembly and for acting as a source of π - π stacking between molecules in the fibres, stabilizing the fibres that form.

However, the 4th and 5th most important fragments, both also map onto phenylalanine but show negative correlation with the presence of this fragment. The 4th fragment (ECFC_4.-1187934975 ([*]C[C@H](NC(=O)C([*])[*])C(=[*])[*])) is only set if the phenylalanine is the amino acid at the c terminus and the 5th fragment (FCFC_4.-1161626344 ([*]C(=[*])N[C@@H](C[c](:[*]):[*])C(=O)O)) is set by benzene rings on molecules not containing phenylalanine so attribution of this fragment to phenylalanine is not possible. Fragment 4 does suggest that the position of the phenylalanine impacts the rheology of the gels.

In fact, it has been shown that Boc-Phe-Aib-OH (Figure 3.35) forms a gel, but when the Phe is the terminal amino acid, Boc-Aib-Phe-OH, the molecule no longer gels¹⁸ suggesting that the position of the amino acids is important in governing their self-assembly and since the rheology of gels are determined by the macroscopic structures formed by the aggregates from self-assembly¹⁹, it is interesting that the model places a negative attribution to specifically terminal phenylalanine fragments.

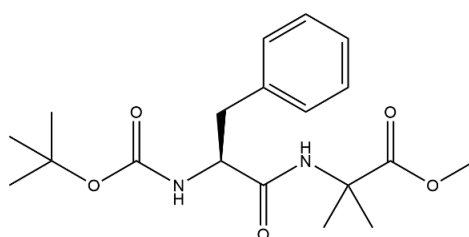


Figure 3.35: Boc-Phe-Aib-OH

3.3.1.10.2 Random Forest

The beeswarm plot for the RF descriptors in Figure 3.36 shows the descriptor value trends in SHAP attribution with the fingerprints visualised in Figure 3.37. Of the 10 most important RF descriptors, the 5th (ECFC_4.717474525 ([*]:[cH]:[c]1:[cH]:[cH]:[cH]:[*]:[c]:1:[*])) and 9th (FCFC_4.-548632217 ([*]C(=[*])O)) are also in the top 10 of the kNN model (at 1st and 10th respectively) and follow the same trends in the RF as they did in the kNN.

The most important fragments in the RF model, FCFC_4.-1272768868 ([*]OCC(=[*])[*]) and ECFC_4.-1059365320 ([*]O[*]) are both set by the methoxy linkers but the ECFC_4.-1059365320 fragment is also set by methoxy substitution on the naphthalene. Although there is no fragment value dependency on the methoxy linker it is known to be important for gelation. In a range of naphthalene analogues, using the methoxy acid linker was shown to form a gel whereas the simple alkyl linkers failed to form gels. Computational simulations show that the methoxy linker allows for a more planar arrangement of the molecule for self-assembly and can also participate in hydrogen bonding.²⁰ The preference for the methoxy linker over an alkyl linker has also been shown in Fmoc gels.²¹ However, it should be noted that the importance is negative at 2.5 mg/mL and positive at 10 mg/mL. This is potentially the concentration scaled descriptors showing their effect on the prediction as the value for this fragment is smaller at low concentration as is the G'.

As in the kNN model, fragments that can be set by phenylalanine (F) are again shown to be important by the model. The 3rd, 7th and 10th most important fragments can all be set by F. However, there are no obvious trends within this set as the in-domain validation set has these fragments all set to 1, but their continued importance is reassuring as it corroborates the experimental findings introduced in 3.3.1.9.1 *k-NN*.

The remaining fragments in the top 10 are FCFC_4.136597326 ([*]C([*])C) (4th) and FCFC_4.-1272798659 ([*]CCC[*]) (8th). The FCFC_4.136597326 (4th) fragment is set by alkyl R groups on amino acids such as valine, leucine, isoleucine and alanine. These fragments show a positive correlation between their number of times present in a molecule and the SHAP value which could be explained by the increased hydrophobicity that an increased number of these fragments can bring. What reinforces this idea, is that leucine has this bit set twice, isoleucine twice, valine twice and alanine once and the hydrophobicity increases from alanine to isoleucine.

Beeswarm plot of the SHAP importance for the RF model

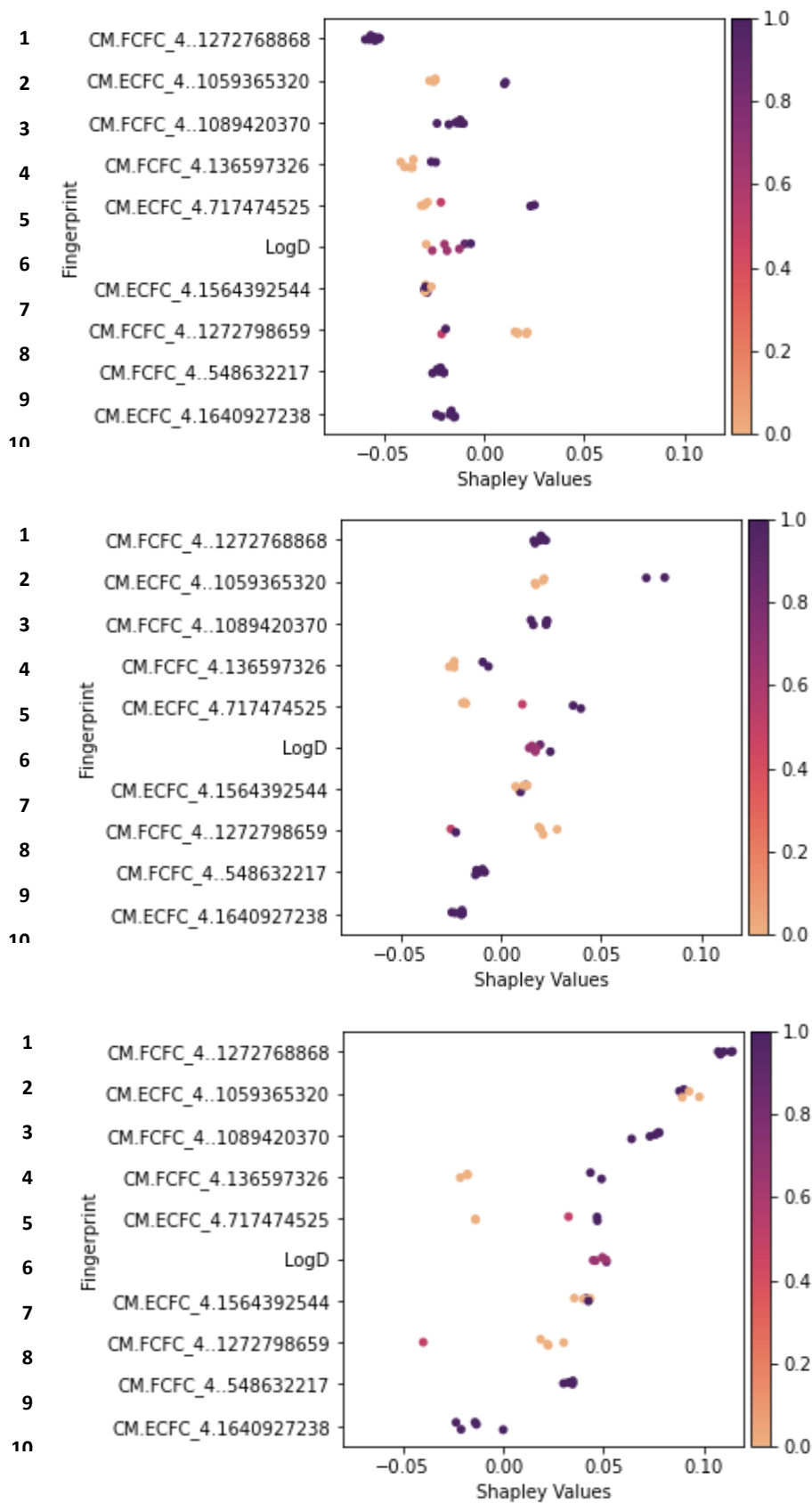
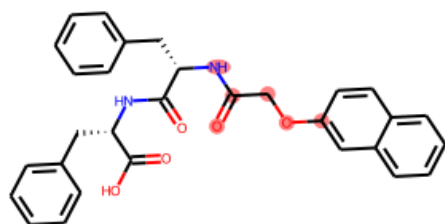
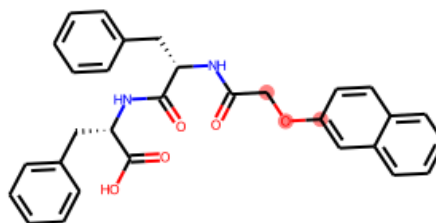


Figure 3.36: Beeswarm plot for the most important RF descriptors. a) 2.5mg/mL b) 5mg/mL and c) 10mg/mL

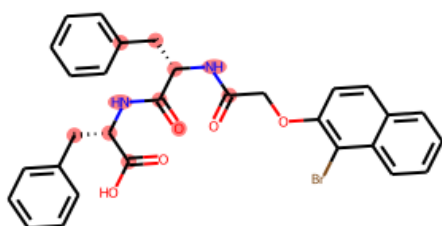
Chemical depictions of the most important fingerprints



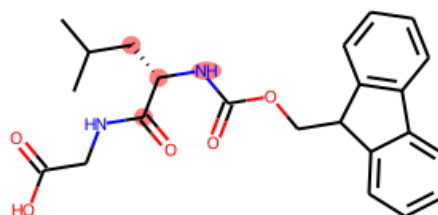
1 – FCFC_4.-1272768868



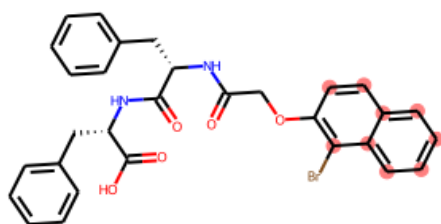
2 – ECFC_4.-1059365320



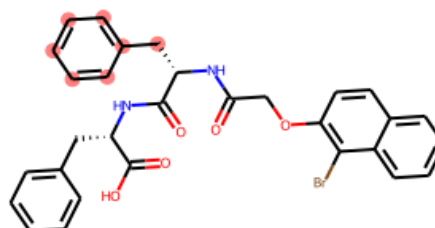
3 – FCFC_4.-1089420370



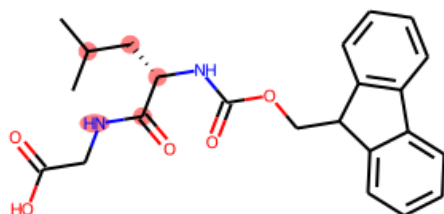
4 – FCFC_4.136597326



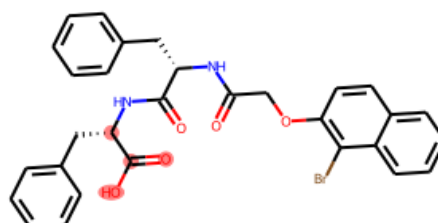
5 – ECFC_4.717474525



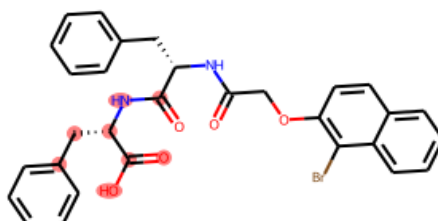
7 – ECFC_4.1564392544



8 – FCFC_4.-1272798659



9 – FCFC_4.-548632217



10 – ECFC_4.1640927238

Figure 3.37: Fragments for the important RF descriptors highlighted on a training set molecule.

FCFC_4.-1272798659 (8th) is set by leucine, isoleucine and cyclohexane. It is set once by the amino acids and twice by the cyclohexane. The negative correlation seen suggests that the cyclohexane has a negative impact on G' as it generally is present in our molecules in place of one of the benzene rings in naphthalene meaning it does not facilitate π - π stacking and this is being picked up by our model. However, the negative impact of the leucine and isoleucine in our validation set contradicts the findings from the FCFC_4.136597326 (4th) fragment suggesting more investigation is required to the relationship between alkyl amino acids and G' .

The final important fragment to be discussed is LogD which shows an increase in the G' value as the LogD value increases. This lends more support to the FCFC_4.136597326 (4th) fragment conclusions that increased hydrophobicity increases self-assembly and ultimately G' .

Across both the kNN and RF model interpretation trends are seen that generally corroborate experimental findings. It is seen that naphthalene, phenylalanine and hydrophobic amino acids (L, A, V and I) are thought of as important for the G' model predictions. However, there are examples of contradictory findings in the kNN and RF model interpretation which calls into question how the model is making predictions.

However, overall in this section, "good" quantitative models for G' have been developed and validated and shown to be predictive. Given the good performance of these kNN and RF models, their datasets will be used to build models for G'' in the following section.

3.3.2 G'' model based on G' model

Given the good performance of the kNN and RF models on the dataset to predict G', we use the same datasets to learn models for the loss modulus, G'', focusing on the 3_2 and 5_1 datasets due to their superior performance in the G' work.

3.3.2.1 kNN models

3.3.2.1.1 Model Performance

Although we learn 10 kNN models for the G'' using the 3_2 dataset as before, all 10 produce identical models and as such we present only the results of the first one. Figure 3.38 shows the combined performance of the train, test and validation set as a single plot. R² on the training set is good, giving an R² of 0.71 coupled with an RMSE of 0.39 which is slightly high when compared to the scores in the training set of the G' models (0.33) but the model performs suitably well in training to assess it on the test set.

An R² of 0.61 on the test set is greater than the 0.6 threshold we outline above and we can consider performance satisfactory thus far. The RMSE of the test set is 0.42 and is very similar to the 0.41 of the G' test set. R² is high on the validation set at 0.69 and the RMSE is low at 0.30. We get slightly better performance on the validation set than the test set, mirroring the G' results. However, a visual inspection of the kNN model shows that the predictions deviate from the dashed line (the line of perfect agreement) and shows that it under predicts at high G'' and over predicts at low G''

The applicability domain filter results in the same test and validation set as the G' set (as the descriptors are identical). This results in 6 remaining test set points and 9 validation points (down from 10 and 12). For the in-domain test set the R² increases from 0.61 to 0.70 and the RMSE from 0.42 to 0.37 (Figure 3.39). On the validation set, the scores improve from R² of 0.69 to 0.72 and the RMSE from 0.30 to 0.25, both indicating that the model performs well statistically on the in-domain test and validation sets.

For the kNN model in Figure 3.38, two of the top three molecules with the highest residuals are training set molecules FmocLG and 2NapFF. These molecules are both at low log(G'') space further emphasising the need for more molecules in these regions to improve the robustness of the model. The final molecule is test set molecule 6MeONapFF which is again seen to be difficult to predict for G'' which is not surprising given the relationship between G' and G''.

Experiment vs prediction plot for the kNN G'' model

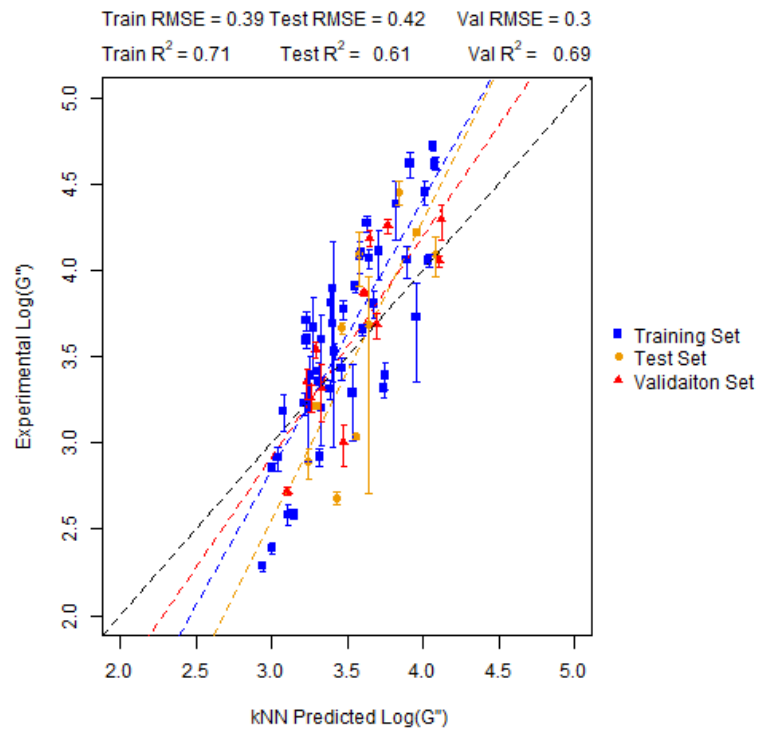


Figure 3.38: Performance of the kNN model on 3_2 dataset for G'' for the training, test and validation sets

Experiment vs prediction plot for the in-domain test and validation set

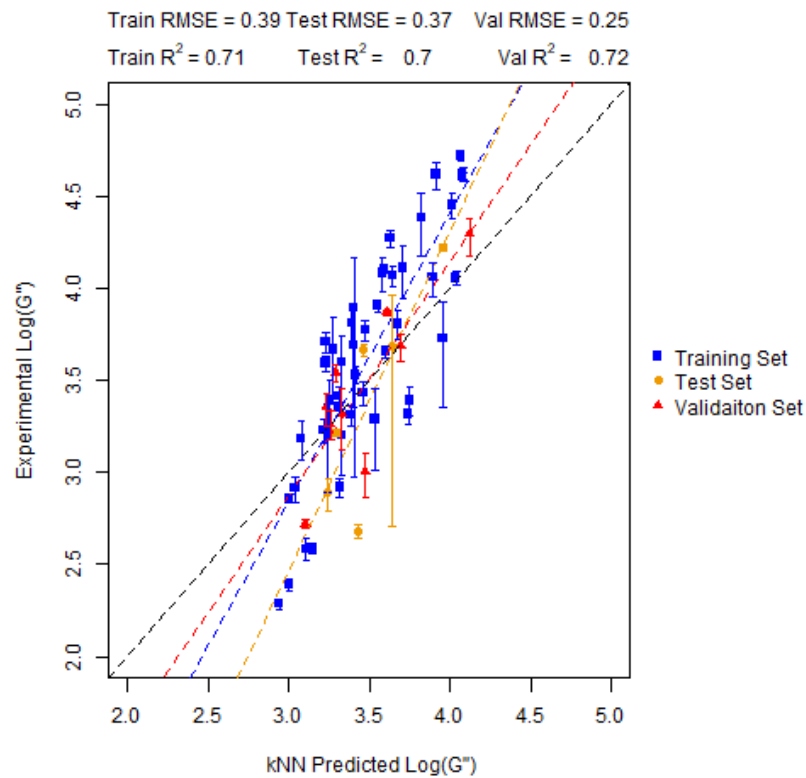


Figure 3.39: Performance of the kNN model on the training and in domain test and validation sets.

3.2.2.1.2 External Performance

Given the promising performance on the train, test and validation sets we expose the G'' model to the external set to gauge if it is truly predictive. From the same external validation set used for G' of 10 molecules (Table 3.4). Of the 10 molecules, the same molecule is again considered out-of-domain and is highlighted in red in the Figure 3.40 below.

Although, the R^2 for this dataset is much lower than the 0.6 threshold at 0.06, we can see that visually the model predicts the majority of the points correctly, although the RMSE of 0.51 on the external set is high. When calculating the performance metrics on the 8 in-domain data points only the R^2 improves to 0.25 and the RMSE to 0.31 – much closer to the 0.3 threshold.

We can conclude that the model has shown sufficient ability to predict for us to consider the model good. A final Y-Randomisation check is carried out to add further evidence as to the predictive ability of our model

Performance of the kNN G'' model on the external set

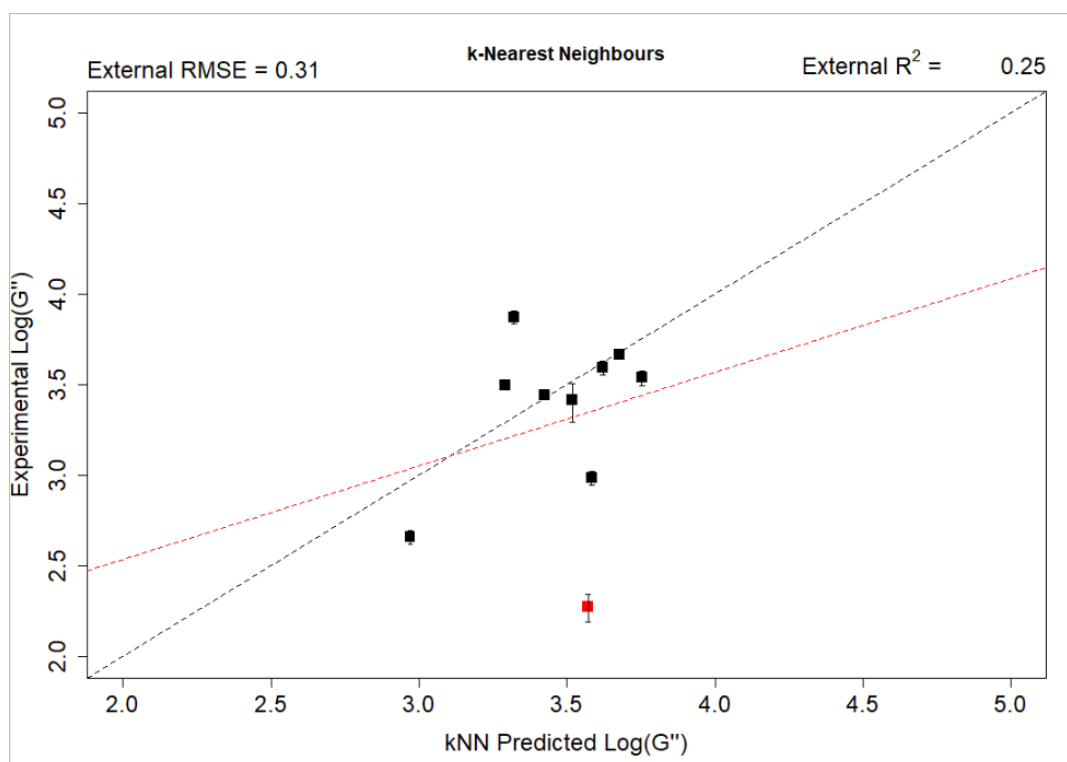


Figure 3.40: Performance of the kNN G'' model on the external dataset. Out-of-domain point is highlighted here in red. The red line of best fit is the best fit for all 10 points.

3.3.2.1.3 Y-Randomisation

Table 3.5 show the performance of the randomised models at predicting the in-domain test set in terms of the calculated Z score and the percentage confidence that the true kNN model is not drawn from the same distribution of models as the randomised model. The Z-scores and corresponding α values are very similar between the G'' model here and the G' values reported earlier. The confidence is slightly lower for our G'' model for R^2 14.41% here and 13.40% for the G' model and for RMSE which is 2.03% here and 1.61% for G' . The RMSE values still give good confidence that the models are distinctly different and the true model is not drawn from the same distribution as the randomised ones.

Table 3.5: Z-Score and confidence interval of the true model compared with the 50 y-randomised models.

Model	Dataset	Z-Score (R^2)	α (R^2)	Z-Score (RMSE)	α (RMSE)
kNN	3_2	1.06	14.41%	2.05	2.03%

3.3.2.2 RF models

3.3.2.2.1 Model Performance

The 10 models learnt using the RF on the 5_1 dataset show some variance in their training set performance. R^2 for all ten models on the test set are good, with all models showing R^2 above our 0.6 threshold and the average R^2 is 0.91 (± 0.01). Moreover, the RMSE values for these models is below the threshold of 0.3. The G'' average RMSE is 0.22 (± 0.01)

On the test set, R^2 is high at 0.69 (± 0.01), above the threshold we set. In terms of RMSE, the value is slightly high at 0.41 (± 0.01), showing similarity to the G' performance. Performance is again good on the validation set, with high R^2 above 0.7 (0.76 (± 0.01)) and low RMSE below the 0.3 threshold (0.26 (± 0.00)).

Therefore, we take the best performing model in terms of the test and validation RMSE which is model 3 (out of 10) as it has the lowest test and validation RMSE. Visual inspection of this model (Figure 3.41)

Applicability domain filtering of the test and validation sets results in the same test set reduction (12 to 9) and validation set (10 to 9) in the in-domain sets. For R^2 on the test set, applicability domain filtering results in an increase in R^2 from 0.71 to 0.75 but an increase in RMSE from 0.41 to 0.43. On the validation set, we find similar performance with R^2 staying at 0.77 and RMSE at 0.26. Figure 3.42 shows the performance of the training set and the in-domain test and validation set points as a comparison.

As in the G' RF model, the G'' model struggles with 1Br2NapVF and 6MeONapFF. However, in the G'' model, 3MeONapFF is also a large outlier in the test set. Given its similarity to 6MeONapFF it is unsurprising that the model will also struggle with this molecule.

Experiment vs prediction plot for the RF G'' model

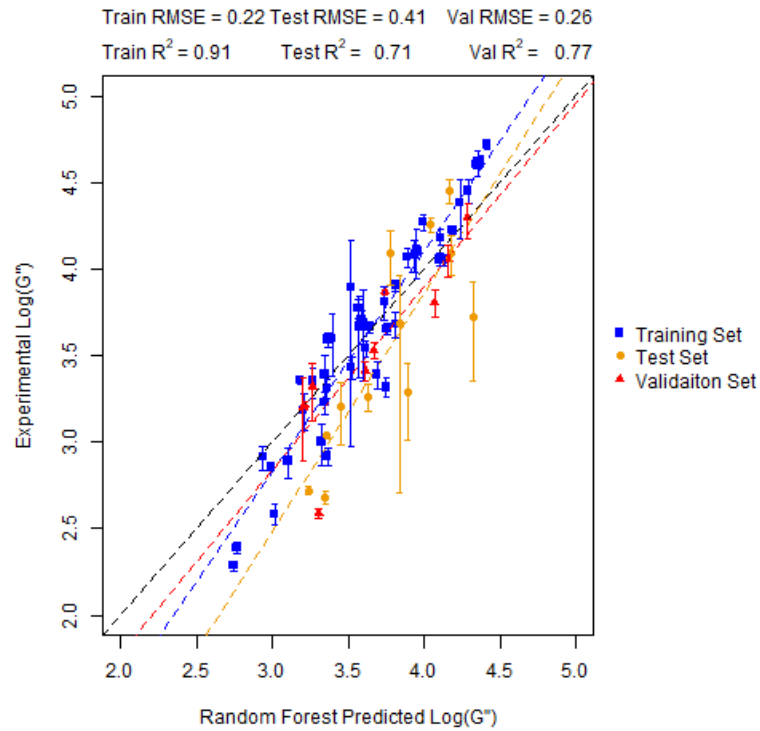


Figure 3.41: Performance of the third Random Forest model on the train, test and validation sets.

Experiment vs prediction plot for the in-domain test and validation set

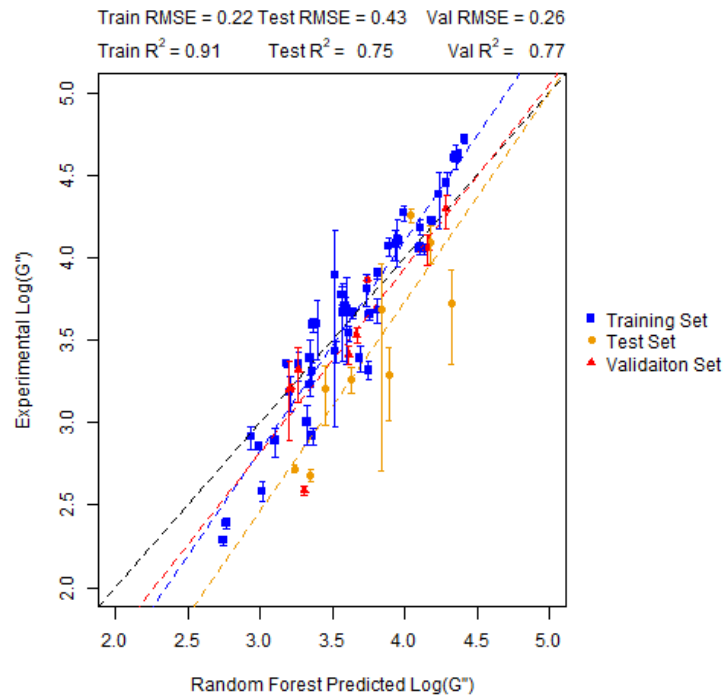


Figure 3.42: Performance of the third Random Forest model on the training set and the in-domain test and validation sets.

3.3.2.2.3 Validation on external set.

Performance of the random forest model on the external set is shown in Figure 3.43 with the out-of-domain again highlighted in red. Overall, like in the G' R^2 is low at 0.09 and RMSE is relatively high at 0.59 – much higher than the 0.3 threshold. When accounting for the out-of-domain point the RMSE improves to 0.45 a much more reasonable value and the R^2 remains low at 0.08. Although statistically poor, given the results on the test, and validation sets coupled with the visual agreement shown in Figure 3.43 means that this model could still be a useful model.

Performance of the RF G'' model on the external set

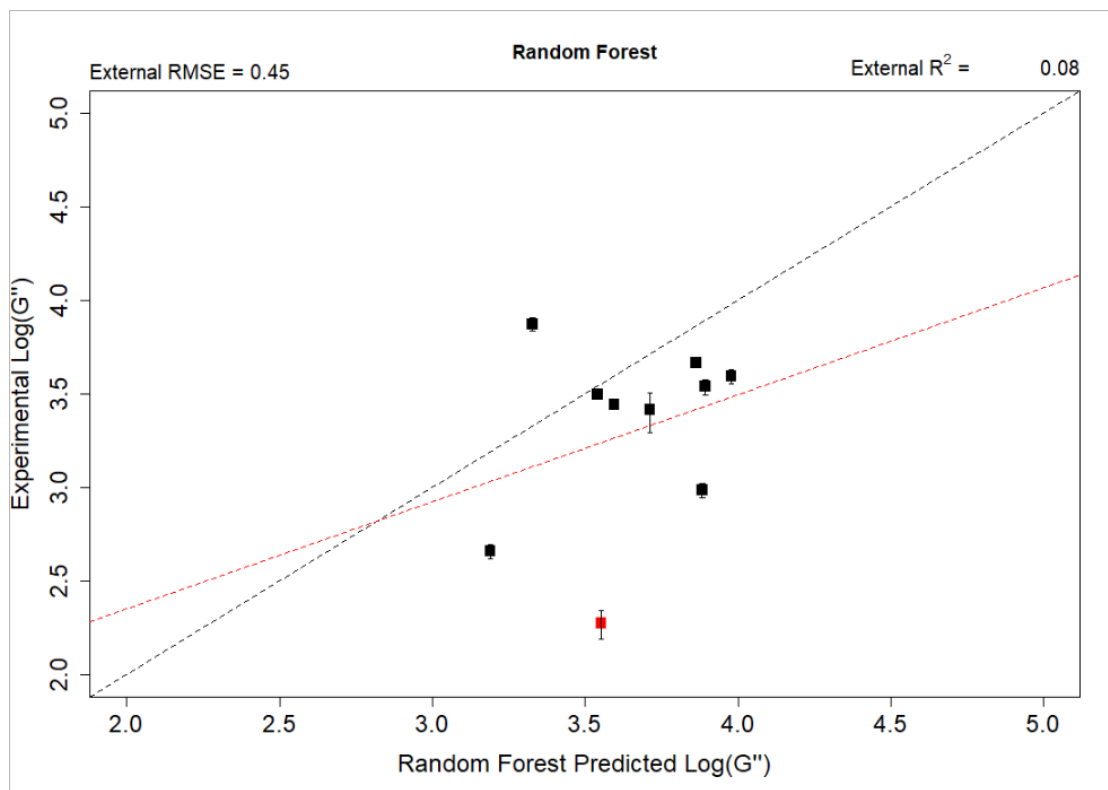


Figure 3.43: Performance of the Random Forest on the external set. Red line of best fit is the best fit for all 10 points.

3.3.2.2.4 Y-Randomisation

As a final check, a y-randomisation approach is again deployed as a validation tool. Table 3.6 shows the results of the y-randomisation approach and it becomes clear that the randomised models do a much worse job of predicting the in-domain test set than the true model in terms of both R^2 and RMSE. The R^2 confidence is 1.33% and the RMSE is 1.68% giving high confidence in both metrics that the true model is finding a relationship between the descriptors and G'' and the relationship is not being found by chance.

Table 3.6: Y-randomisation results for our Random Forest model on the in-domain dataset

Model	Dataset	Z-Score (R^2)	α (R^2)	Z-Score (RMSE)	α (RMSE)
RF	5_1	2.22	1.33%	2.13	1.68%

Overall, although performance could be improved for the RF model in terms of the RMSE on the external set, the model has still shown satisfactory performance across the different datasets and y-randomised check to have confidence in the ability to predict the G'' of dipeptide-based gels.

Given that both the kNN and RF model show promising performance on G'' the models are interpreted and comparisons are drawn between these models and their equivalent G' models.

3.3.2.1 Variable Importance

The same approach is carried out here as in (3.3.1.9 - Variable importance) where the SHAP values are calculated on the joint in-domain test and validation sets. The importance is again presented as a beeswarm plot (Figure 3.44) with the fragments in the beeswarm plot presented after the beeswarm in Figure 3.45.

3.3.2.2.1 k-Nearest Neighbours

There is some overlap between the fragments thought important in the G' kNN as well as here with the SHAP. Fragments ECFC_4.717474525 and FCFC_4.-1908346267 are both the 1st and 2nd most important fragment in the G'' model just as they were in the G' model. They also follow the same value dependence on SHAP with the SHAP attribution increasing as the value increases.

Of the remaining fragments that are present here as well as in the G' kNN SHAP, FCFC_4.307419094 ([*][c](:[*]):[c](:[cH]:[*]):[c](:[*]):[*]) is 3rd here and 6th in the G', ECFC_4.-1194163736 ([*]C[C@H](NC(=O)C[*])C(=[*])[*]) is 5th here and 3rd in G' and ECFC_4.-1187934975 ([*]C[C@H](NC(=O)C[*])[*])C(=[*])[*]) is 8th here and 4th in the G'. Of these fragments only ECFC_4.-1194163736 shows different value dependence, unlike in the G' increased counts of this fragment also result in an increase in G''. The relationship was explored in detail in *section 3.3.1.9.1 – k Nearest Neighbours*

Although not present in the kNN SHAP, fragments ECFC_4.1564392544 ([*][c]1:[*]:[cH]:[cH]:[cH]:[cH]:1) and FCFC_4.-1272768868 ([*]OCC(=[*])[*]) are found in the RF G' SHAP. ECFC_4.1564392544 is 6th here and 7th in the RF G' and FCFC_4.-1272768868 is 10th here but is the most important descriptor in the RF SHAP. Neither show value dependence on the SHAP score which is consistent with what was found for the RF G' SHAP. The relationship was explored in detail in *section 3.3.1.9.2 – Random Forest*

In total there are 3 fragments here that have not been seen to be important previously and all three map onto naphthalene. FCFC_4.346218766 ([*]CO[c]1:[cH]:[cH]:[*]:[c](:[*]):[cH]:1) is the 3rd most important fragment and maps to the naphthalene with a methoxy linker – showing an increased SHAP when this fragment increases in number. This is consistent with the methoxy linker investigated in *section 3.3.1.9.2 – Random Forest* and further emphasizes the potential impact it has on the mechanical properties through the planarity it introduces.²⁰

FCFC_4.-105186863 ([*][c]1:[*]:[cH]:[cH]:[c]2:[cH]:[cH]:[*]:[cH]:[c]:1:2) is a purely naphthalene fragment and is the 7th most important here showing a modest increase in SHAP as the fragment increases in count. This could be attributed to the π - π stacking and hydrophobicity introduced by naphthalene being picked up by the model and assigned high importance.

The final fragment is FCFC_4.-1977359400 ([*]C(=[*])CO[c](:[cH]:[*]):[c]([*]):[*]) which maps the naphthalene with the methoxy linker. The modest increase in the SHAP by the increase in the number of this fragment is consistent with the findings of the methoxy linker found in *3.3.1.9.2 – Random Forest*.

Overall, the findings for the G'' SHAP for the kNN model are mostly in line with the findings discussed within *3.3.1.10 Variable Importance* showing that the model is generally using similar fragments in its prediction of both G' and G''.

Beeswarm plot of the SHAP importance for the kNN G'' model

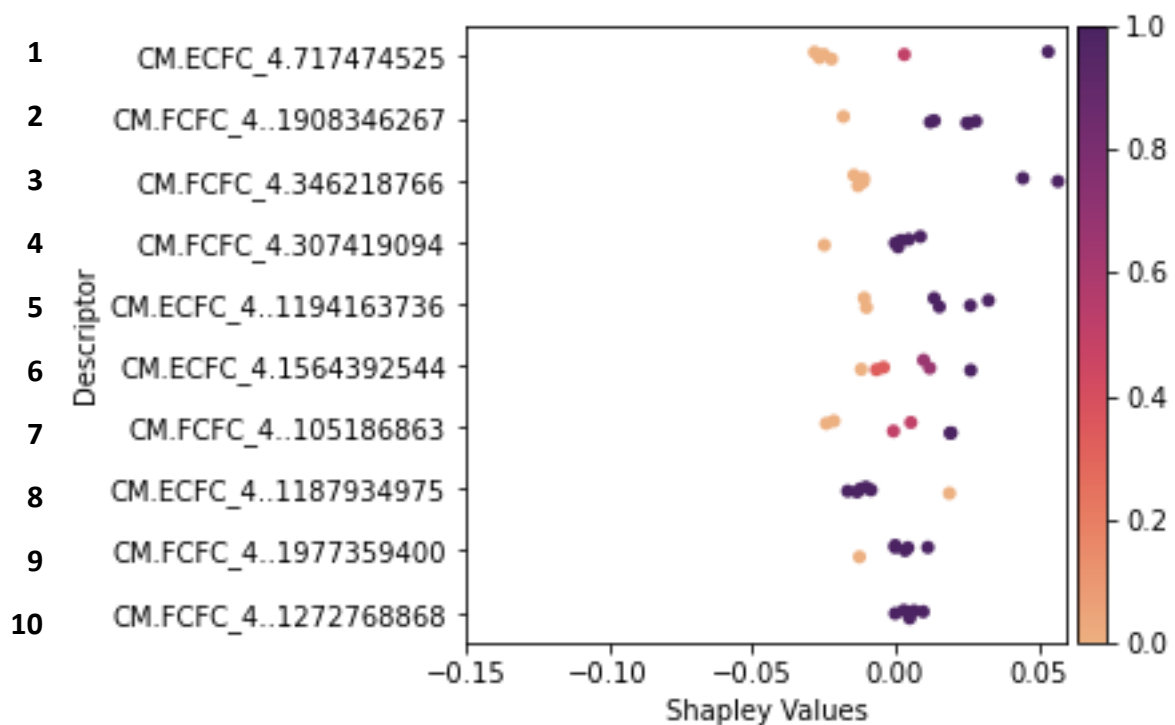
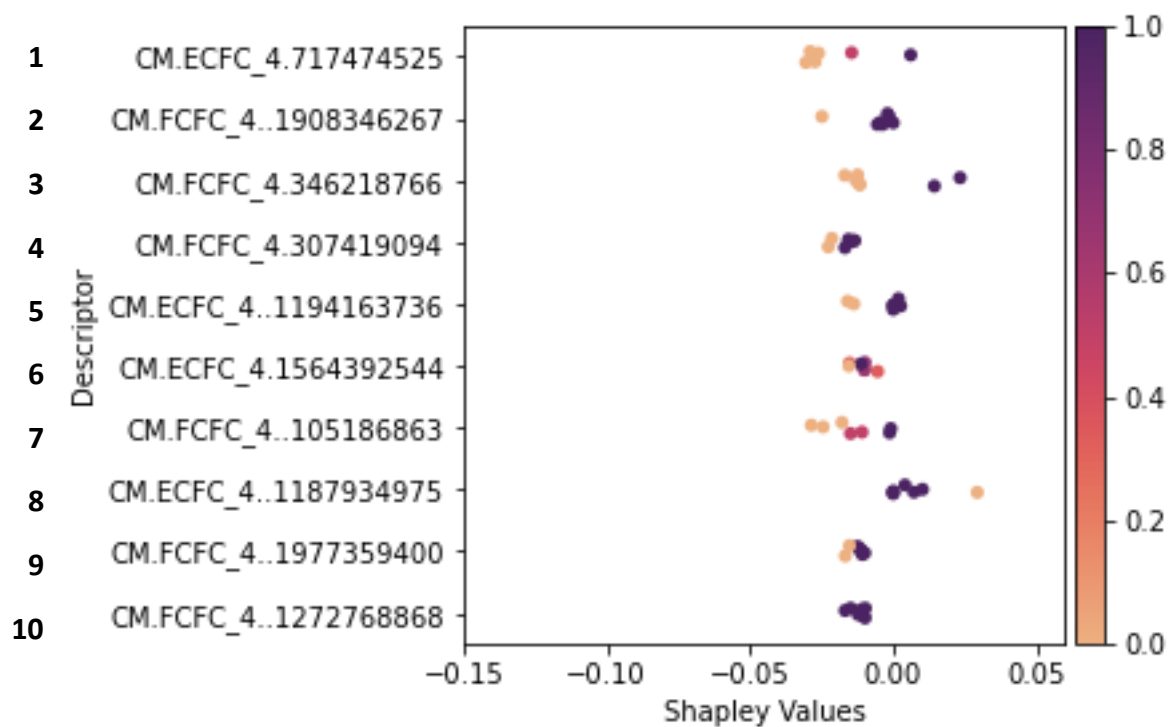
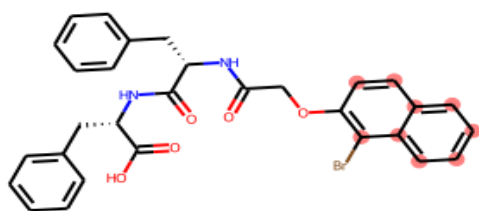
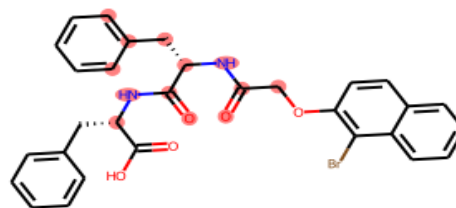


Figure 3.44: SHAP beeswarm plots for the 2.5 and 5 mg/mL molecules in the in-domain test and validation sets.

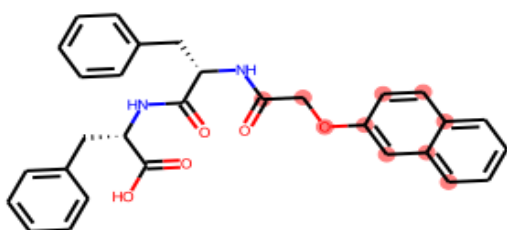
Chemical depictions of the most important fingerprints



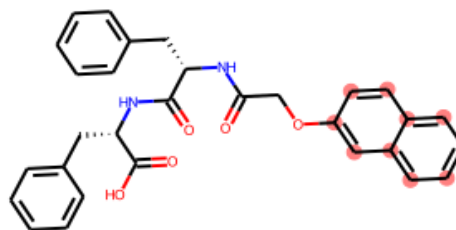
1 – ECFC_4.717474525



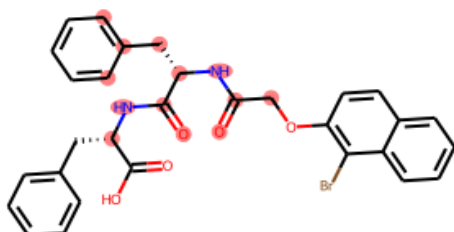
2 – FCFC_4.-1908346267



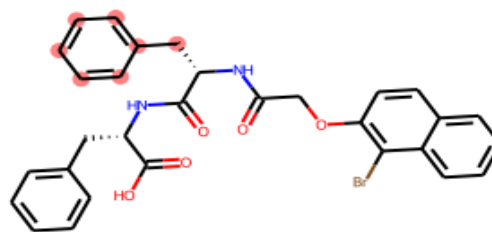
3 – FCFC_4.346218766



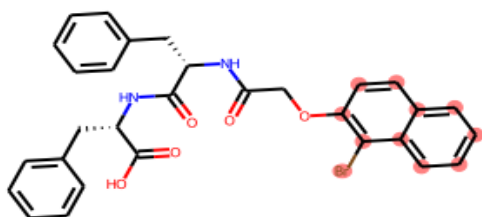
4 – FCFC_4.307419094



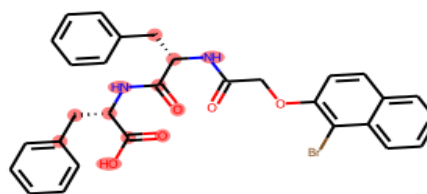
5 – ECFC_4.-1194163736



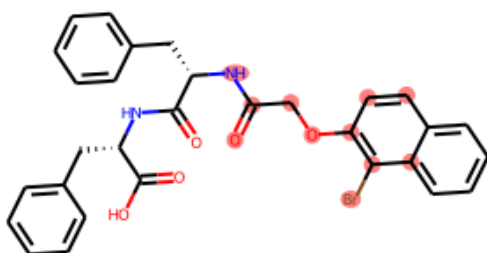
6 – ECFC_4.1564392544



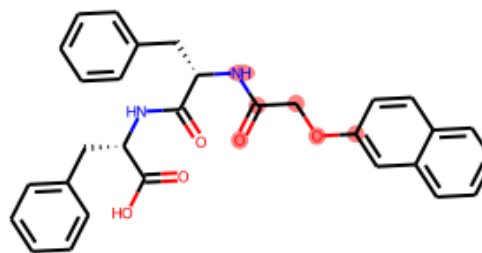
7 – FCFC_4.-105186863



8 – ECFC_4.-1187934975



9 – FCFC_4.-1977359400



10 – FCFC_4.-1272768868

Figure 3.45: Fingerprint fragments in the top 10 of the kNN G'' presented on a training set molecule.

3.3.2.3.2 Random Forest

Finally for this chapter, the SHAP importance for the descriptors used to build the G'' Random Forest are calculated. Of the 10 fragments important for the G'' RF model, 7 out of the 10 were also present in the top 10 for the G' RF model. These are the top 5 fragments plus the 7th and 9th fragment. The relationship between the fragment value and the SHAP score are the same but the order is slightly different and shown in Figures 3.46 with the structure of the fragments shown in Figure 3.47.

FCFC_4.-1272768868 ([*]OCC(=[*])[*]) and ECFC_4.-1059365320 ([*]O[*]) are 1st and 2nd respectively both in the G' and G'' SHAP. FCFC_4.-548632217 ([*]C(=[*])O) is 3rd here and 9th in the G' SHAP. The 4th and 5th fragments here ECFC_4.717474525 ([*]:[cH]:[c]1:[cH]:[cH]:[cH]:[*]:[c]:1:[*]) and FCFC_4.-1089420370 ([*]C[C@H](N[*])C(=O)NC([*])[*]) are 5th and 3rd respectively in the corresponding G' work. ECFC_4.1564392544 ([*][c]1:[*]:[cH]:[cH]:[cH]:[cH]:1) is 7th in both works and FCFC_4.136597326 ([*]C([*])C) is 9th here and 4th for G'

Of the three remaining fragments important in this set, one FCFC_4.-105186863 ([*][c]1:[*]:[cH]:[cH]:[c]2:[cH]:[cH]:[*]:[cH]:[c]:1:2) is found in the G'' kNN model (6th here and 7th there) leaving only 2 completely new fragments.

These fragments are ECFC_4..1559650422 ([*]C[*]) the 8th most important and FCFC_4.551279842 ([*]C[C@H](NC(=O)C[*])C(=[*])[*]) the 10th most important. Neither fragment possesses any fingerprint count dependence on the SHAP value and are thus rather uninformative.

Overall, it is clear that the G'' model is using the same fragments as the G' model to make its prediction and the trends are the same in both models suggesting a degree of a relationship between G' and G'', since gels are viscoelastic and exhibit both elastic and viscous properties, this relationship is unsurprising.

Beeswarm plot of the SHAP importance for the RF G'' model

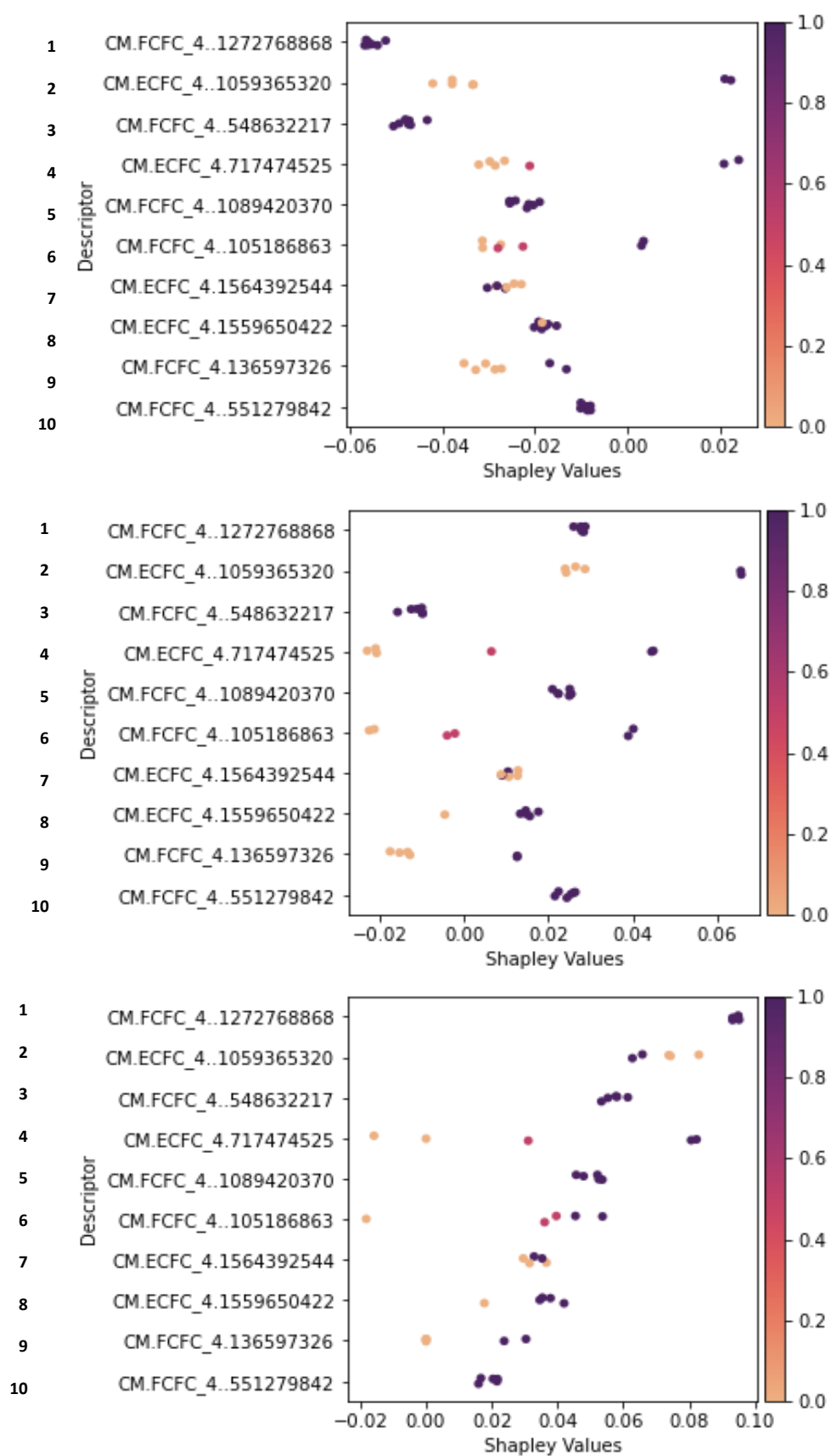


Figure 3.46: Beeswarm plot for the RF G'' model for 2.5, 5 and 10 mg/mL

Chemical depictions of the most important fingerprints

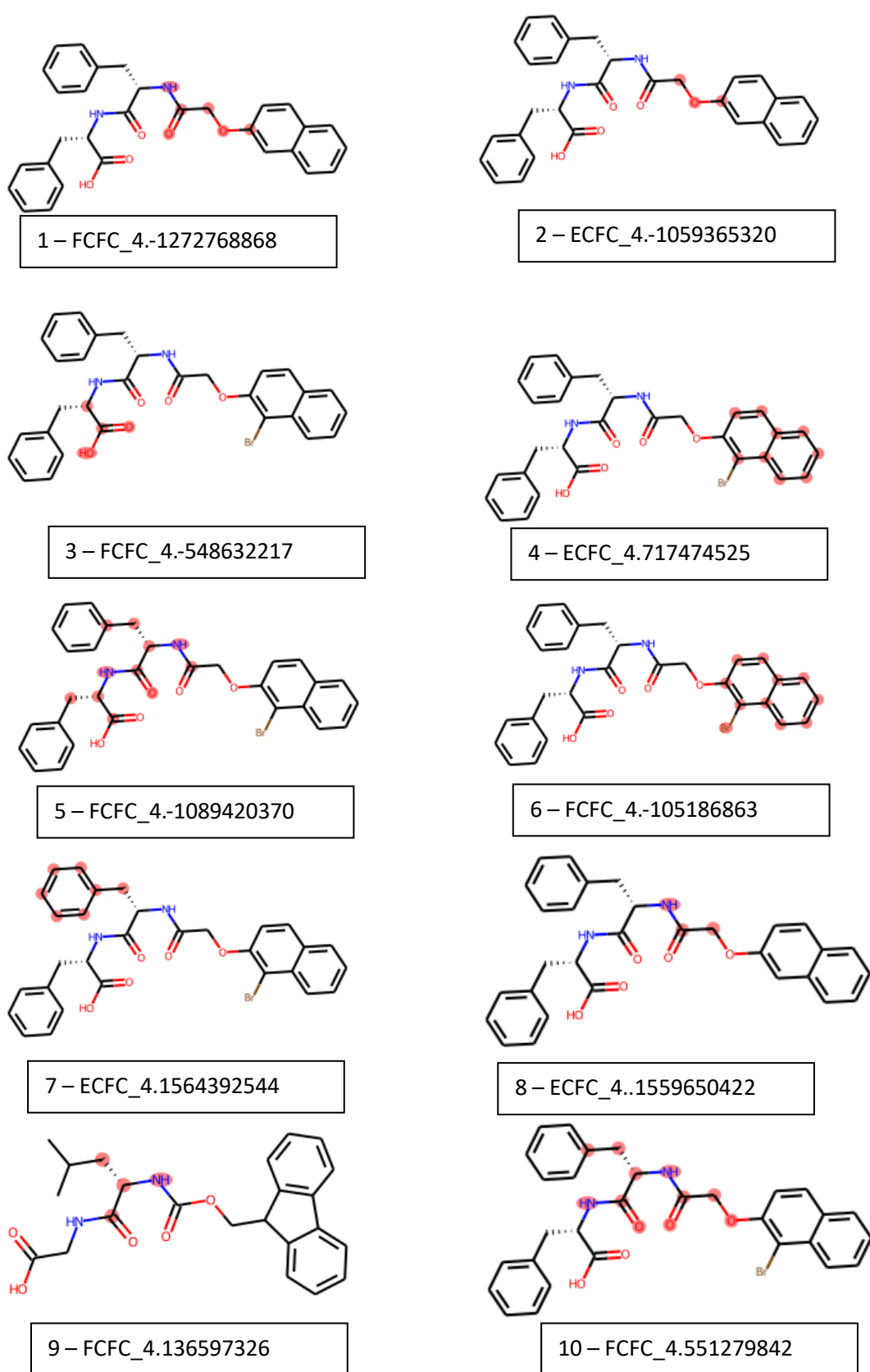


Figure 3.47: SHAP important fragments for the G'' RF Model highlighted on training set molecules

3.4 Conclusions

In this chapter we have used our experimental dataset to build predictive models for both the storage (G') and loss modulus (G''). Through an extensive splitting procedure, we found models that showed promising performance for G' for all 5 of our modelling algorithms investigated (BayesGLM, kNN, PLS, RF and SVM). However, we were mindful that due to our splitting approach, chance correlations were a potential problem. To combat this, we exposed our models to an external dataset and carried out a y -randomisation check. In both cases, the kNN model built on dataset 3_2 and the RF built on 5_1 showed good external predictive performance for G' . Owing to their good G' performance, both datasets were used to build models for G'' , showing similarly good performance during training, testing and validation.

However, due to time-restraints in our collaborator's lab, we have been unable to assess the G'/G'' models over a suitable range of values meaning that although we are confident performance is good, complete validation will have to be carried out in the future. All models have been interpreted and the importance values shown have been rationalised.

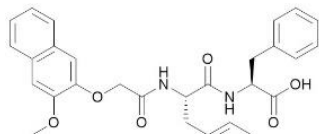
3.5 References

- 1 E. V. Alakpa, V. Jayawarna, A. Lampel, K. V. Burgess, C. C. West, S. C. J. Bakker, S. Roy, N. Javid, S. Fleming, D. A. Lamprou, J. Yang, A. Miller, A. J. Urquhart, P. W. J. M. Frederix, N. T. Hunt, B. Péault, R. V. Ulijn and M. J. Dalby, *Chem*, 2016, **1**, 298–319.
- 2 N. J. Hoglebe, J. W. Reinhardt, N. K. Tram, A. C. Debski, G. Agarwal, M. A. Reilly and K. J. Gooch, *Acta Biomater.*, 2018, **70**, 110–119.
- 3 K. Basu, A. Baral, S. Basak, A. Dehsorkhi, J. Nanda, D. Bhunia, S. Ghosh, V. Castelletto, I. W. Hamley and A. Banerjee, *Chem. Commun.*, 2016, **52**, 5045–5048.
- 4 L. Haines-Butterick, K. Rajagopal, M. Branco, D. Salick, R. Rughani, M. Pilarz, M. S. Lamm, D. J. Pochan and J. P. Schneider, *Proc. Natl. Acad. Sci. U. S. A.*, 2007, **104**, 7791–7796.
- 5 C. K. Thota, N. Yadav and V. S. Chauhan, *Sci. Rep.*, 2016, **6**, 1–12.
- 6 Q. Zou, R. Chang, R. Xing, C. Yuan and X. Yan, *J. Control. Release*, 2020, **319**, 344–351.
- 7 M. C. Nolan, A. M. Fuentes Caparrós, B. Dietrich, M. Barrow, E. R. Cross, M. Bleuel, S. M. King and D. J. Adams, *Soft Matter*, 2017, **13**, 8426–8432.
- 8 A. M. Fuentes-Caparrós, K. McAulay, S. E. Rogers, R. M. Dalgliesh and D. J. Adams, *Molecules*, 2019, **24**, 1–10.
- 9 P. J. Rousseeuw and M. Hubert, *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 2011, **1**, 73–79.
- 10 F. Montanari, L. Kuhnke, A. Ter Laak and D. A. Clevert, *Molecules*, 2020, **25**, 1–13.
- 11 D. L. J. Alexander, A. Tropsha and D. A. Winkler, *J. Chem. Inf. Model.*, 2015, **55**, 1316–1322.
- 12 S. A. Otto, M. Kadin, M. Casini, M. A. Torres and T. Blenckner, *Ecol. Indic.*, 2018, **84**, 619–631.
- 13 S. M. Lundberg and S. I. Lee, *Adv. Neural Inf. Process. Syst.*, 2017, **2017-Decem**, 4766–4775.
- 14 N. Chirico and P. Gramatica, *J. Chem. Inf. Model.*, 2012, **52**, 2044–2058.
- 15 A. Hazra, *J. Thorac. Dis.*, 2017, **9**, 4125–4130.
- 16 L. Chen, K. Morris, A. Laybourn, D. Elias, M. R. Hicks, A. Rodger, L. Serpell and D. J. Adams, *Langmuir*, 2010, **26**, 5232–5242.

- 17 S. E. Wheeler, *Acc. Chem. Res.*, 2013, **46**, 1029–1038.
- 18 S. K. Nandi, K. Maji and D. Haldar, *ACS Omega*, 2018, **3**, 3744–3751.
- 19 E. R. Draper and D. J. Adams, *Chem*, 2017, **3**, 390–410.
- 20 Z. Yang, G. Liang, M. Ma, Y. Gao and B. Xu, *J. Mater. Chem.*, 2007, **17**, 850–854.
- 21 S. Fleming, D. Sisir, P. W. J. M. Frederix, T. Tuttle and R. V. Ulijn, *Chem. Commun.*, 2013, **49**, 10587–10589.

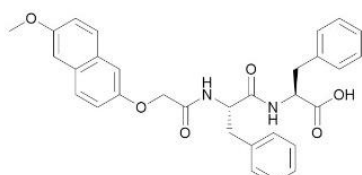
3.6 Appendix

3.6.1 Dataset



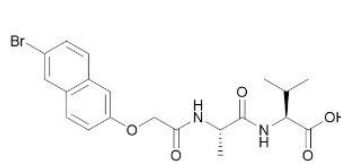
Chemical Formula: $C_{31}H_{30}N_2O_6$
Molecular Weight: 526.59

3MeO2NapFF



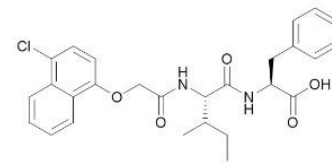
Chemical Formula: $C_{31}H_{30}N_2O_6$
Molecular Weight: 526.59

6MeO2NapFF



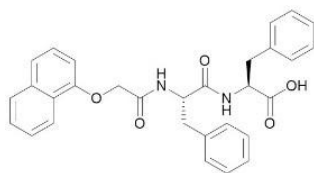
Chemical Formula: $C_{20}H_{23}BrN_2O_5$
Molecular Weight: 451.32

6BrNapAV



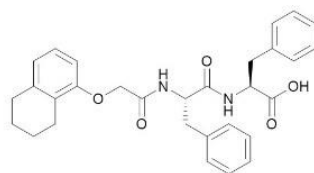
Chemical Formula: $C_{27}H_{31}ClN_2O_5$
Molecular Weight: 496.99

4Cl1NapIF



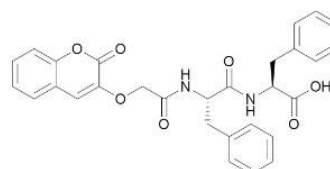
Chemical Formula: $C_{30}H_{28}N_2O_5$
Molecular Weight: 496.56

1NapFF



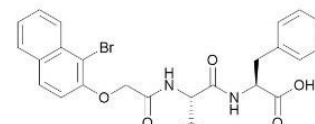
Chemical Formula: $C_{30}H_{32}N_2O_5$
Molecular Weight: 500.60

1ArFF



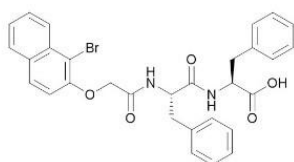
Chemical Formula: $C_{29}H_{30}N_2O_7$
Molecular Weight: 514.53

CoumarinFF



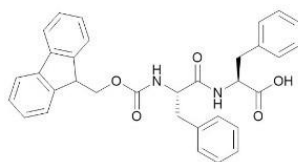
Chemical Formula: $C_{26}H_{27}BrN_2O_5$
Molecular Weight: 527.42

1Br2NapVF



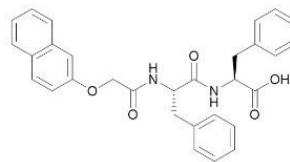
Chemical Formula: $C_{30}H_{27}BrN_2O_5$
Molecular Weight: 575.46

1Br2NapFF



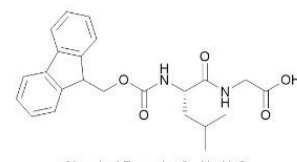
Chemical Formula: $C_{33}H_{30}N_2O_5$
Molecular Weight: 534.61

FmocFF



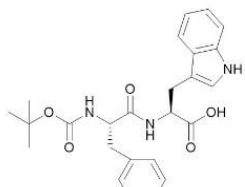
Chemical Formula: $C_{33}H_{30}N_2O_5$
Molecular Weight: 496.56

2NapFF



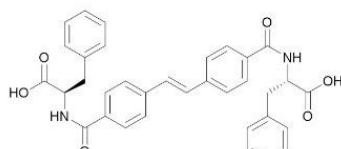
Chemical Formula: $C_{23}H_{26}N_2O_5$
Molecular Weight: 410.47

FmocLG



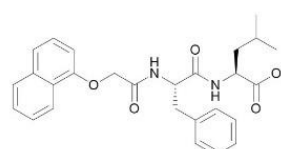
Chemical Formula: $C_{25}H_{29}N_3O_5$
Molecular Weight: 451.52

BocFW



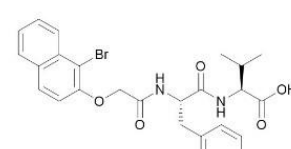
Chemical Formula: $C_{32}H_{30}N_2O_6$
Molecular Weight: 562.62

Stilbene-F



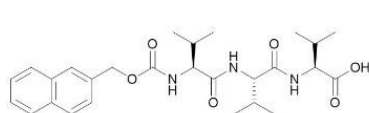
Chemical Formula: $C_{27}H_{30}N_2O_5$
Molecular Weight: 462.55

1NapFL



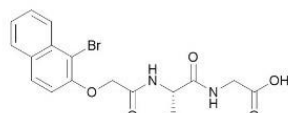
Chemical Formula: $C_{26}H_{27}BrN_2O_5$
Molecular Weight: 527.42

1Br2NapFV



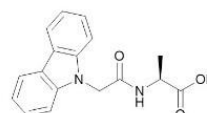
Chemical Formula: $C_{27}H_{37}N_3O_6$
Molecular Weight: 499.61

2NapVVV'



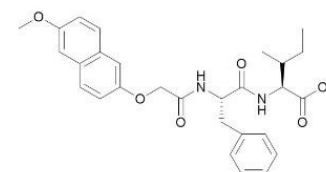
Chemical Formula: $C_{18}H_{27}BrN_2O_5$
Molecular Weight: 437.29

1Br2NapVG



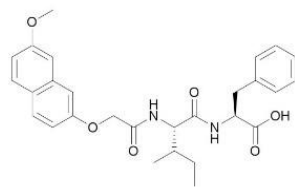
Chemical Formula: $C_{17}H_{18}N_2O_3$
Molecular Weight: 296.33

CarbA



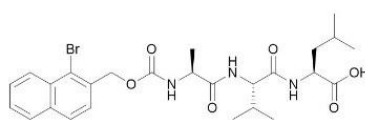
Chemical Formula: $C_{28}H_{32}N_2O_6$
Molecular Weight: 492.57

6MeO2NapFI



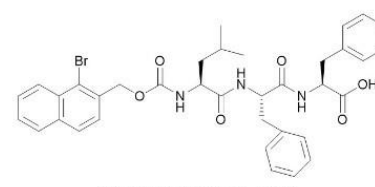
Chemical Formula: $C_{28}H_{32}N_2O_6$
Molecular Weight: 492.57

7MeO2NapIF



Chemical Formula: $C_{26}H_{34}BrN_3O_6$
Molecular Weight: 564.48

1Br2NapAVL



Chemical Formula: $C_{36}H_{38}BrN_3O_6$
Molecular Weight: 688.62

1Br2NapLFF

LMWG	Concentration (mg/mL)	G'	G' error	G''	G'' error
1Br2NapFF	2.5	9160	1520	778	156
1Br2NapFF	5	49200	5620	4630	321
1Br2NapFF	10	162000	8430	16900	762
1NapFF	2.5	17800	7450	2110	784
1NapFF	5	56700	4230	7400	187
1NapFF	10	122000	21800	19700	4580
FmocFF	2.5	34100	4940	2260	453
FmocFF	5	38500	3920	3510	377
FmocFF	10	152000	13700	15300	1620
3MeO2NapFF	2.5	23300	9900	1590	621
3MeO2NapFF	5	28500	15800	1950	920
3MeO2NapFF	10	79800	43100	5360	3070
6MeO2NapFF	2.5	3970	501	480	44
6MeO2NapFF	5	37100	31900	4920	4410
6MeO2NapFF	10	89200	32300	12500	3300
1ArFF	2.5	19000	9110	1570	790
1ArFF	5	18400	2280	2600	353
1ArFF	10	44000	3130	6440	1200
6Br2NapAV	2.5	7130	511	1090	34
6Br2NapAV	5	92900	35100	12400	4270
6Br2NapAV	10	193000	22400	28700	4780
2NapFF	2.5	19100	3640	7850	6890
2NapFF	5	40400	6470	11700	1440
2NapFF	10	171000	4500	53000	3710

FmocLG	2.5	60200	13600	5930	739
FmocLG	5	120000	28700	18800	1920
FmocLG	10	224000	49000	41700	6800
BocFW	2.5	14000	1420	1540	378
BocFW	5	49400	8110	5140	685
BocFW	10	159000	25100	12800	617
Stilbene-F	2.5	1620	76	193	11.7
Stilbene-F	5	5900	268	717	34.9
Stilbene-F	10	22700	1640	2290	116
CoumarinFF	2.5	10500	388	1640	61.2
1NapFL	2.5	34500	9900	2460	739
1NapFL	5	71200	16100	6530	1420
1NapFL	10	202000	74500	24200	9160
1Br2NapFV	2.5	43500	22800	3970	1540
1Br2NapFV	5	53300	8960	8100	615
1Br2NapFV	10	228000	69700	28600	4780
2NapVVV	2.5	80600	17500	4930	2690
2NapVVV	5	275000	110000	13000	4180
2NapVVV	10	580000	115000	42400	3030
1Br2NapVG	2.5	63100	51700	4680	2320
1Br2NapVG	5	66300	25300	12200	2510
1Br2NapVG	10	207000	62100	40700	2390
CarbA	2.5	1940	147	244	18.2
CarbA	5	5410	639	819	126
CarbA	10	20800	2000	3940	380
1Br2NapVF	2.5	2690	51.7	386	23.2
1Br2NapVF	5	21100	2070	3380	364
1Br2NapVF	10	80400	12700	11500	2420
4Cl1NapIF	2.5	3940	331	522	27.8
4Cl1NapIF	5	14400	2290	1830	331

4Cl1NapIF	10	196000	9820	18100	1830
6MeO2NapFI	2.5	7320	2320	1010	272
6MeO2NapFI	5	32700	4760	4830	852
6MeO2NapFI	10	98700	2810	11400	920
7MeONapIF	2.5	10700	1430	2070	266
7MeONapIF	5	27400	1200	4540	303
7MeONapIF	10	83800	2570	11400	1010
BrNapAVL	2.5	9230	1390	1700	263
BrNapAVL	5	18600	1630	2730	420
BrNapAVL	10	17200	1650	2080	268
BrNapLFF	2.5	3300	275	384	51.6
BrNapLFF	5	5550	273	830	96.3
BrNapLFF	10	18800	1960	2470	440

Chapter 4 – Bayesian Regression Models

4.1 Introduction

In chapter 3, promising models for both the G' and G'' have been presented using both a k-Nearest Neighbours and Random Forest algorithm. However, when presenting the results of a QSAR model, they are typically presented as a single value (the prediction). This in turn assumes a great deal of certainty in that prediction – attributing all the information contained within the descriptors to a single value. Also, when building models on datasets that are small, since there are fewer molecules present to generate the model, it is often more difficult to have a high degree of certainty in a single value. Therefore, using modelling approaches that can capture any uncertainty in the predicted value is more informative for any future deployment of the model.

4.1.1 Uncertainty and Probability

Uncertainty is a measure of our lack of knowledge about an event or random variable and we can express the range of our belief in a value as a probability. Probabilities are the likelihood of events occurring or of a random variable's potential values. They must be positive and the total area under the probability density for a continuous variable must be 1.¹ For a particular random variable, the probability of that variable possessing a range of values is given by the area under the probability density curve between those two values. It then follows that the probability of a continuous variable assuming a single value is zero and as such, when we present weights in a neural network or a predicted value as a single value – the probability that the prediction takes that value is zero and this is where probability distributions can represent our uncertainty in predictions.

4.1.2 Conditional Probability

The probability of two events A and B occurring ($A \cap B$) is given by the probability of A, $P(A)$, and the conditional probability of B given that A has occurred $P(B|A)$. The conditional probability of A given that B has occurred is $P(A|B)$.¹ It follows that this can also be expressed as $P(B)$ and the $P(A|B)$,

$$P(A \cap B) = P(B|A) * P(A)$$

$$P(B \cap A) = P(A|B) * P(B)$$

These two terms are equivalent and can be set equal

$$P(A|B) * P(B) = P(B|A) * P(A)$$

Which can be rearranged to give:

$$P(B | A) = \frac{P(A|B) * P(B)}{P(A)}$$

Which is formally known as Bayes Theorem.²

4.1.3 Bayes theorem

Bayes theorem relies on the idea of updating your belief of an event based on the available evidence. It is made up of the *prior* $P(B)$, our belief about B before we see any evidence. $P(A|B)$, the *likelihood* or the evidence. $P(B|A)$ is the *posterior*, the updated belief about B given that A has occurred. $P(A)$ is a normalising factor to ensure that the probabilities integrate to 1.³

Although the prior and evidence are well defined in most instances, direct calculation of the posterior distribution is tricky due to the marginal likelihood $P(A)$ term. If we apply Bayes theorem in a QSAR setting, we can re-write Bayes theorem as

$$P(y | X) = \frac{P(X|y) * P(y)}{P(X)}$$

Where X is our n-dimensional descriptor set and y is an endpoint of interest. The $P(X)$ term is the probability of X for all possible values of y and calculating this term is computationally intensive as it involves multi-dimensional integration. Since this term is a normalising factor, we can define our posterior distribution up to a constant

$$P(B | A) \propto P(A|B) * P(B)$$

Uncertainty has been captured in QSAR models in the form of Bayesian machine learning. Williams *et al.*⁴ created a Bayesian QSAR model for the prediction of drug induced liver injury (DILI). Here they use 96 drugs to build a log-odds regression model to predict a severity of the DILI between 0 and 1. The model then decides, from the data, thresholds for the DILI as low, medium and high severity. The model presents the DILI as a distribution between 0 and 1 and the posterior median is used to classify based on the thresholds. The model has a balanced accuracy of 86% during training and their model has been deployed within AstraZeneca since 2017 and would have flagged a melanin concentrating hormone receptor 1 antagonist, AZD197 (Figure 4.1), which showed liver safety biomarker elevation in human trials.

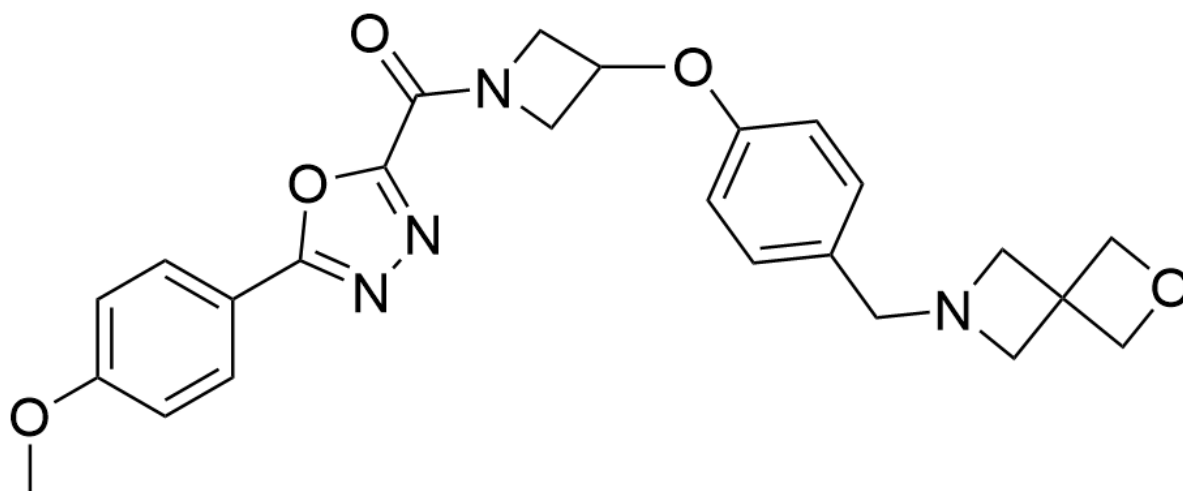


Figure 4.1: AZD 1979 which showed increased liver safety biomarkers in the work by Williams *et al*⁴

Hung *et al.*⁵ use Bayesian neural networks for the prediction of the mutagenicity of a molecule. Here they use the graph-based representation of the molecule to build a convolutional neural network to classify a molecule as a mutagen or non-mutagen. During training, the model possesses an accuracy of 0.84 and upon application of the model to an external test set, the accuracy remains high at 0.84. Other metrics such as the ROC area under the curve also show good performance in both training and testing.

Another example of a Bayesian neural network is within the field of organic photovoltaics. Meftahi *et al.*⁶ built a Bayesian regularised artificial neural network to predict the percentage conversion efficiency, the short circuit current and the open circuit voltage for the organic photovoltaics. These models exhibit high R^2 for the PCE in particular of 0.72 on the training set (280 molecules) and 0.78 for the test set (70 molecules). Moreover, the application of the Naïve Bayes model introduced in Chapter 1 (1.6.3 Naïve Bayes) are further examples of Bayes theorem in QSAR.

However, although the full Bayes theorem cannot be calculated in closed form, many methods have been developed to take samples from the true posterior distribution without having to calculate it.

4.1.4 Posterior sampling

4.1.4.1 MCMC Methods

The most common methods used to sample from intractable posterior distributions are a class of methods called Markov Chain Monte Carlo (MCMC) methods. A Monte Carlo simulation is a computational algorithm that relies on repeated independent random sampling from a distribution.⁷ A series of samples from a distribution are a Markov chain only if the next sample depends completely on the current sample and any previous samples have no importance in choosing the next sample.⁸

4.1.4.2 Metropolis Hastings

An example of a MCMC method used for Bayesian inference of the posterior is the Metropolis-Hastings (MH) algorithm. In MH, a surrogate probability distribution is used to draw samples from and samples from this distribution are accepted or rejected as representative samples from the true distribution with respect to an acceptance criterion.⁹

As an example, say we have our true posterior distribution $p(x|data)$ that we cannot directly sample from and so we define our surrogate distribution q . We carry out a MCMC sample from q , starting from a previous sample, x and accept this new sample, y , from q , with acceptance ratio, r :

$$r = \frac{p(y|data)q(x|y)}{p(x|data)q(y|x)}$$

Since we know the distribution $q(x)$ and $q(y)$ – the only unknown is p , the posterior distribution. We can substitute the RHS of Bayes theorem into the equation and write the ratio fully as

$$r = \frac{\frac{p(data|y)p(y)}{p(data)}q(x|y)}{\frac{p(data|x)p(x)}{p(data)}q(y|x)}$$

We can see in the ratio, the intractable calculation $p(data)$ cancels and therefore we can sample from the true posterior distribution from the likelihood and prior only.

$$r = \frac{p(data|y)p(y)q(x|y)}{p(data|x)p(x)q(y|x)}$$

We accept our sample y from q with probability

$$a = \min(1, r)$$

And begin the next sample from the accepted sample y . However, other acceptance cut offs can be used such as if $r > U_{\text{draw}}$ where U_{draw} is a draw from a uniform distribution $U(0,1)$ to ensure that the sampling doesn't get stuck in a local maximum and efficiently samples the entire posterior space. Figure 4.2 shows examples of a MH sample to simulate a normal distribution with mean 0 and standard deviation 1. As you can see, even after only 500 samples the algorithm does a good job at generating a normal distribution. The mean of the samples is -0.17 with a standard deviation of 0.79 which is close to the 0 and 1 of the true distribution. However, increasing the number of samples to 1000 results in increased performance as the mean becomes -0.03 with a standard deviation of 1.04, much closer to the true distribution.

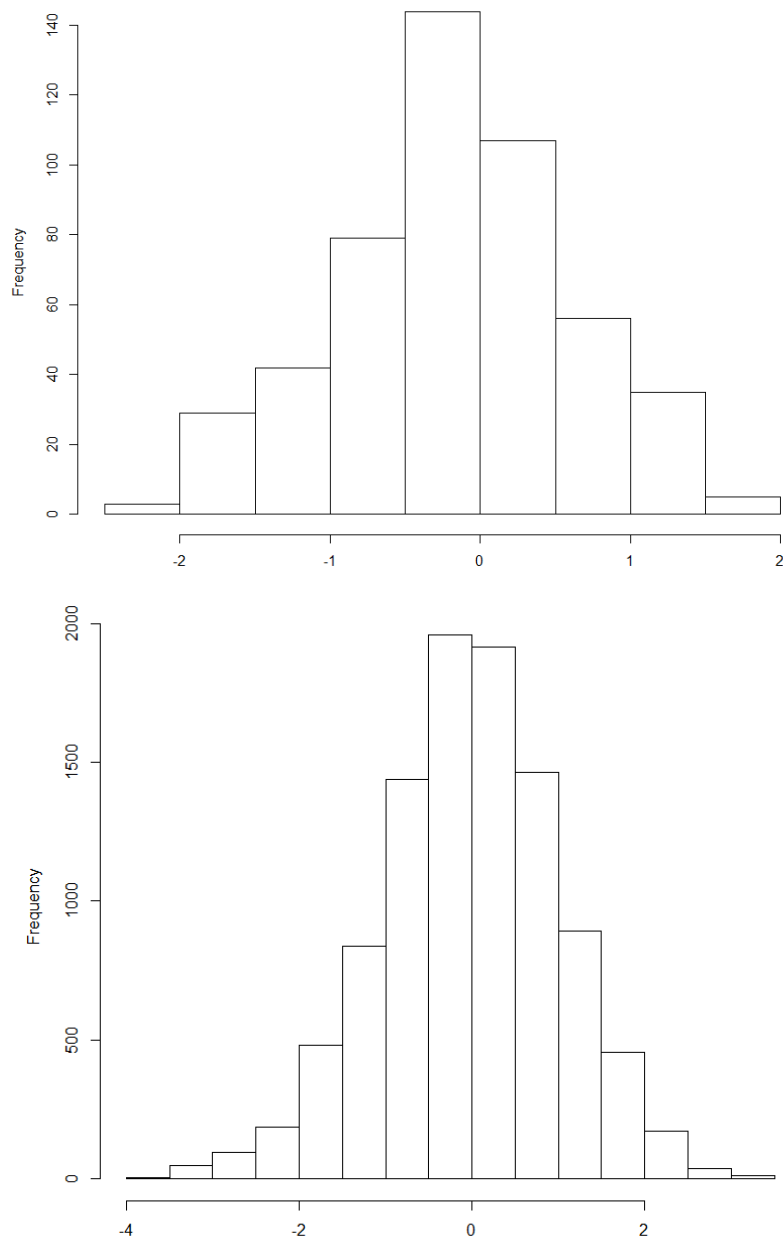


Figure 4.2: a) Distribution of 500 samples, x , from the MH sampling of a Normal distribution, $N(0,1)$ and b) 1000 samples

Once able to sample from the posterior, Bayesian inference allows for the generation of posterior samples that can be presented as a distribution. For example, in a Bayesian neural network (see 1.6.4 *Neural Networks*), we can set a prior probability distribution on the value of the weights in the hidden layer. Carrying out Bayesian inference when subjecting the neural network to our data will give a posterior distribution for the weights which when sampled will give a prediction distribution for each prediction, derived from a different draw from the posterior distribution of the weights.

When dealing with Bayesian models, comparing predictions to known values leaves decisions to the modeller on how best to present the distribution of predicted values. Although it is always best to present the entire distribution of a prediction, estimates of the posterior are required to point-wise compare our predictions to known values. Common approaches include, taking the mean of the distribution, the mode of the distribution (the maximum a posteriori) or the median. However, care needs to be taken as a single value will not describe the nature of the distribution and the distribution should always be explored.

4.1.5 Bayesian Models

Bayesian adaptations to popular algorithms introduced in Chapter 1 have been successfully used to measure prediction uncertainty. Bayesian neural networks have successfully been deployed in varying fields such as load forecasting for energy grid management¹⁰, repeat purchases modelling in a marketing setting¹¹ and predicting car crashes.¹²

4.1.5.1 Bayesian Additive Regression Trees

As tree-based machine learning algorithms are so popular, a Bayesian decision tree methodology such as a Bayesian Additive Regression Tree (BART) is an attractive proposition. A BART model is an additive regression tree where each tree works in an additive fashion to each explain small sections of the data.¹³ BART can be presented as sum of trees model (Equation 4.1)

$$Y = g(x; T, M) + e \quad (4.1)$$

Where T is the set of regression trees and M the set of node parameters and x the data.

To prevent overfitting, the trees are kept shallow so that no tree is too influential in the prediction. This is where the model is Bayesian. A prior distribution is set on the depth of the trees such that the highest probability is given to trees of 2 to 3 nodes depth. It is worth noting however, that the trees

can grow as long as they desire as long as the data requires it. The trees are grown iteratively, sampling from the posterior distribution using a modified MH approach that proceeds as follows.

In a BART model for 500 iterations of 5 trees, initiate empty nodes. Sequentially, starting with the first tree, draw from a uniform distribution both a feature and feature value. Carry out the split and then draw from the node parameters the value to be assigned to that node. The Y value for the next tree in the same iteration is set to the residual of the data not explained by the first tree so that each tree attempts to explain part of the unexplained data. At each step, the tree can be grown, pruned, the decision rule change or the split direction altered, all with predetermined probability. This is shown in Figure 4.3

The updated tree is then accepted or rejected with respect to a transition ratio (R_T) based on the ratio of the probabilities of the updated tree and the previous tree. This ratio is compared to a draw from a uniform distribution $U(0,1)$ and if R_T is greater than the draw from U , the change is accepted.

Sampling from a distribution requires a burn-in period to calibrate the samples with the true distribution. The burn-in period is the name referred to samples at the very start of the sampling process that are discarded from the set of final samples but are used to allow the sampling procedure time to find samples in the correct distribution.¹⁴ At the end of each iteration after the burn-in period, a prediction is made for a data point from the sum of the individual decision trees. The result would be 500 predictions for each data point which are the predictions from the trees as they existed at the end of each iteration. Bayesian additive models have been successfully used in predicting phenotypes based upon genotypes¹⁵, in credit risk modelling¹⁶ and in individual treatment regimens in personalised medicine.¹⁷

The BART model has also been deployed in QSAR and has been assessed in its ability to produce QSAR models for the activity values in a range of molecular datasets. These datasets include the CYP450 inhibition activity, LogD values and binding to angiotensin-II receptor datasets.¹⁸ Feng *et al.* use 30 of the datasets described in [18] to build QSAR models and compared their performance to RF and XGBoost (another form of decision tree model). They find that the BART model performs on average better than the RF in terms of the R^2 between the experiment and prediction on the test sets. BART has also been used to classify molecules as tyrosine kinase inhibitors and showed comparable performance to RF, SVM and C5.0 in terms of the balanced accuracy.¹⁹

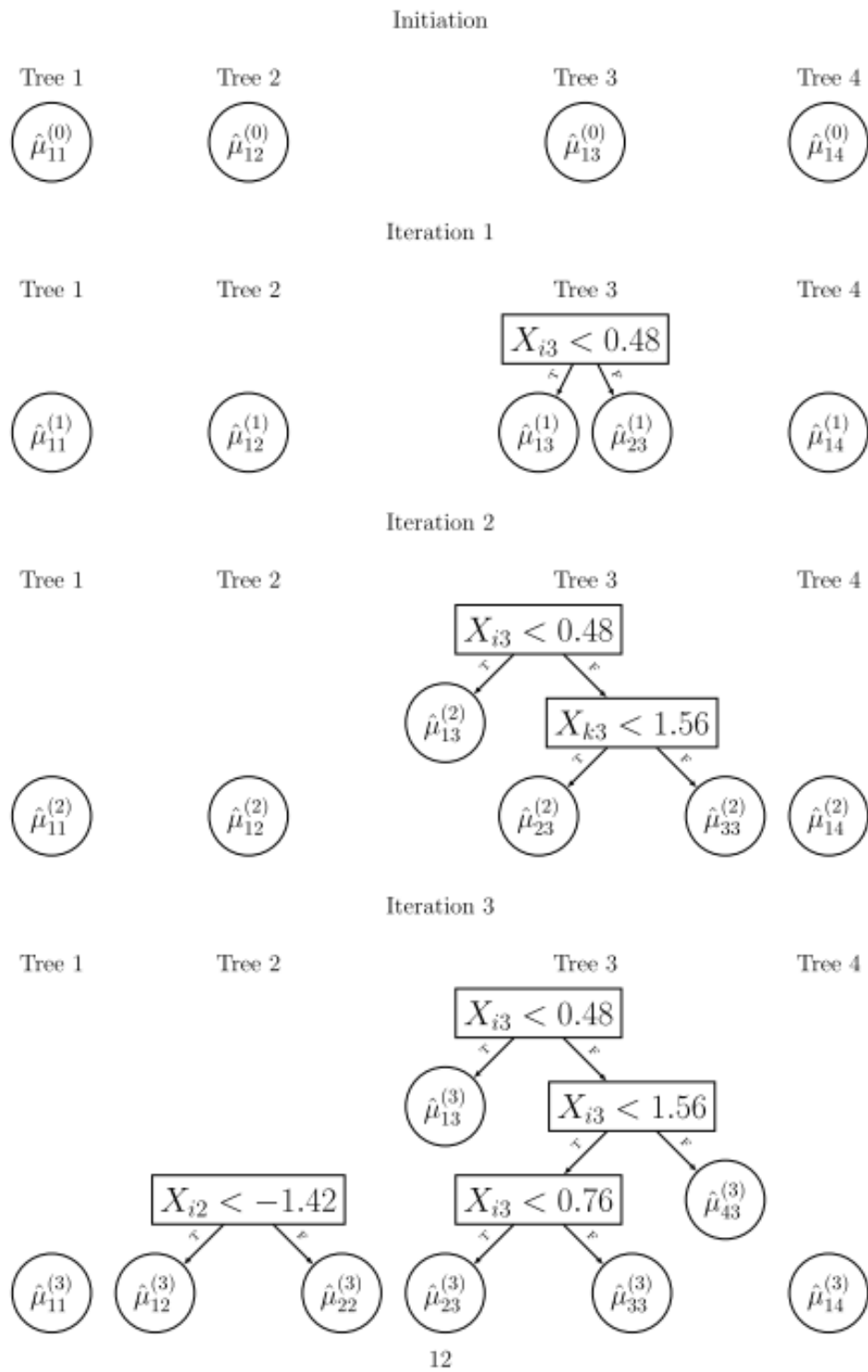


Figure 4.3: An example of the iterative tree building during sampling for a BART model. Figure adapted from Tan and Roy²⁰

4.1.6 Chapter Overview

In this chapter, we will present Bayesian regression models using the dataset with the best performance in Chapter 3. Since we have trained our rheology models using a small dataset, any certainty in the point predictions returned by the kNN or RF models is small and as such, we believe it would be informative to highlight this uncertainty using a Bayesian approach.

We used a Bayesian Additive Regression Tree (BART) model in Python and compared its performance to our previous RF models. This is to ensure that the performance is comparable to the models presented in chapter 3 and the RF model was chosen as it is also a decision tree based approach. We investigate the posterior distributions of our predictions and compare the model interpretation to investigate any commonality between the non-Bayesian RF models and the BART model.

4.2 Experimental

4.2.1 Dataset

Given that the BART model is a decision tree-based model, it is much more directly comparable to the Random Forest models from Chapter 3 and therefore, we have decided to utilise the same 5_1 dataset that was used to build the RF models in the previous chapter (*see Chapter 3: 3.3.1.5.4 Best Performing Model*) to build the BART models presented here. Although multiple datasets were created in chapter 3, time constraints meant that there was only time to investigate one dataset. Therefore, 5_1 was chosen as it performed best overall on the RF models built in chapter 3.

4.2.2 Descriptors

We attempted to mirror the descriptors used in chapter 3 as closely as possible. However, these models were built in Python, meaning we therefore used RDKit for the majority of our descriptor calculations. We calculated AlogP, Number of Rings and the ECFP and FCFP binary fingerprints with radius 2 (radius = 2 is synonymous with the radius = 4 from Pipeline Pilot calculated in earlier chapters) and converted them to a 2048 bit string. Molecular solubility was calculated using the RDKit implementation of Delaney's ESOL.²¹

4.2.3 Processing

Since the dataset has been used for model building previously, each molecule already contained a train, test or validation label. Therefore, Pandas²² (*version 1.1.3*) was used to split the dataset into train, test and validation based on the pre-assigned label. After splitting, the training set was processed to remove descriptors with less than 5% variance using the sklearn²³ (*version 0.23.2*) VarianceThreshold() function and remove all descriptors that were heavily correlated ($r > 0.9$) with other descriptors in the dataset. The test and validation set files were curated to match the descriptors remaining in the training set after processing. All datasets were centred and scaled with respect to the training set using the StandardScaler() module in sklearn.

4.2.4 Model Building

BART model building was carried out for both G' and G'' using the PyMC3 (*version 3.11.4*) package in Python using the default hyperparameters for the BART implementation (Figure 4.4) – except for the number of trees which was reduced to 50 trees due to the relatively small size of the dataset. We model the endpoint ($\log(G')$ or $\log(G'')$), y , as a Normal distribution with mean, μ , and standard deviation, σ . μ is the predicted value from the sum of the additive regression trees from our BART model of 50 trees, and the standard deviation, σ , of the predicted y value is initialised as a HalfNormal distribution with mean σ and standard deviation of 1. Sampling from σ is done via a No-U-Turn sampler (NUTS) from PyMC3 and sampling from μ is through the PyMC3 PGBART sampler. Each sampling procedure is carried out twice resulting in two sampling chains and each are run for 2000 samples.

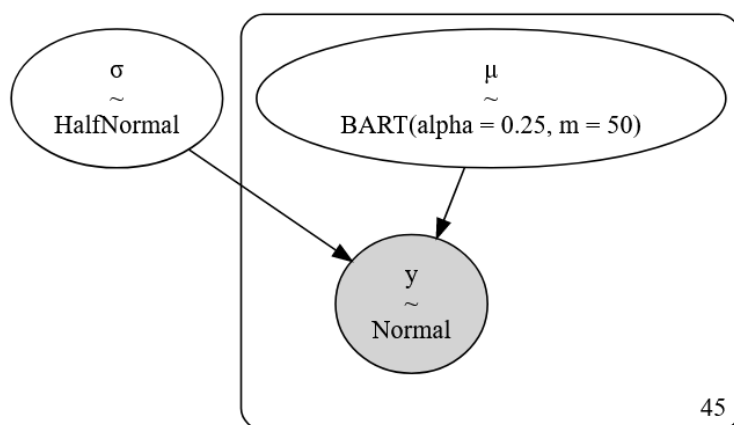


Figure 4.4: Schematic overview of the BART model

4.2.5 Model Validation

We validate the convergence of the sampling to the true posterior distribution for both the μ and σ using a trace plot from the Arviz package²⁴ in python. We assess the ability of the models to correctly predict G' (G'') by comparing the mean of the y distribution with the experimental G' (G'') values. Model performance is quantified using the same RMSE (Equation 4.2) and R^2 (Equation 4.3) thresholds from *Alexander et al.*²⁵ as chapter 3 which states that models can be considered predictive if they have:

- High R^2 ($R^2 > 0.6$) on the test set
- Low RMSE

Given that the value of RMSE depends on the scale of the predictions, for our best model we calculate RMSE and check if the RMSE is low. Given that our data roughly covers 2.6 log units ($G' = 3.2$ to $G' = 5.8$), we consider RMSE of 10% this range (0.3 or below) as low. This is the same validation criteria used in chapter 3 (see 3.2.6 Model Validation)

$$RMSE = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_i - \hat{y}_i)^2} \quad (4.2)$$

$$R^2 = \frac{\sum(y - \hat{y})^2}{\sum(y - \bar{y})^2} \quad (4.3)$$

Again, we use the same Y-Randomisation approach introduced earlier where we built 50 models on data with a shuffled $\log(G')$ ($\log(G'')$) value and assess the difference in model performance metric significance through calculation of the Z score (Equation 4.4) and comparing the Z score calculated with the threshold values for various significance values.

$$Z = \frac{x - \text{average}(\tilde{x})}{\text{standard deviation}(\tilde{x})} \quad (4.4)$$

Where x is the RMSE or R^2 value and \tilde{x} is the same metric for a randomised model.

4.2.6 Applicability Domain

The applicability domain of our Bayesian models is assessed through the descriptor range checking approach again in terms of the descriptor values and the loadings of the first 10 principal components for the training set (*see 2.2.9 Applicability domain*).

4.2.7 Model Interpretation

We calculate the same SHAP values for our BART model using the Shap package in Python (*version 0.40.0*) and compare the trends in descriptor importance with the results found for our RF model in chapter 3 (*see 3.3.10.2 Random Forest*).

4.3 Results and Discussion

4.3.1 G' BART model

4.3.1.1 Sampling Traceplot

Figure 4.5 shows the traceplot of the sampling from the BART μ and σ distributions. The traceplot contains four components each given a pane in Figure 4.5 and for each pane there are two plots for each parameter value (easily seen in the bottom left pane) which is due to the fact that we carry out the sampling using two chains. The left pane for both μ and σ is a density plot of all samples taken from the posterior distribution for each parameter and the right pane for both is a plot showing the value sampled (on the y axis) against the sample number from the 1st sample drawn to the 2000th (on the x axis). Since each μ represents a predicted G' from our model, there are multiple overlapping plots here (each one assigned a different colour) – one for each molecule in the training set. The right panes for both μ and σ gives an indication to the convergence of the sampling to the true distribution.

BART G' traceplot

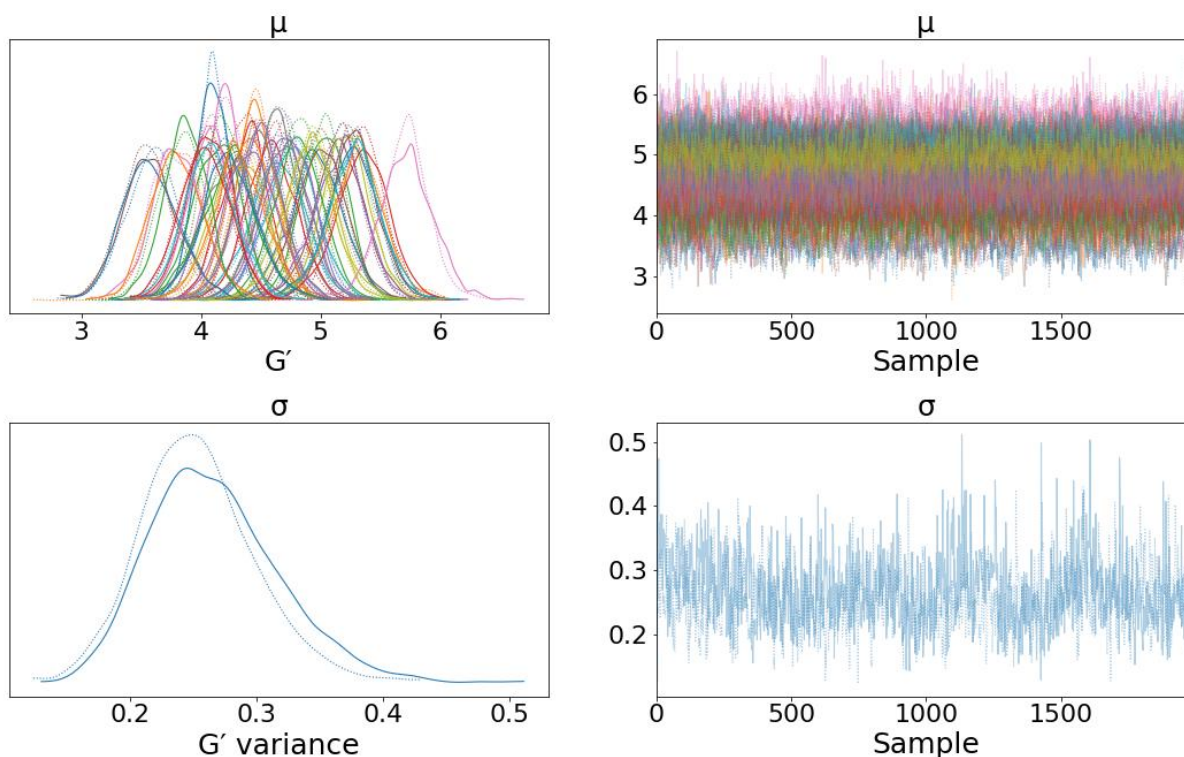


Figure 4.5: Traceplot of the MCMC sampling for the BART G' model.

To assess the convergence of the sampling of the μ and σ distributions the Gelman-Rubin (R) statistic²⁶ can be used (Equation 4.5). W is the within chain variance and B is the between chain variance meaning that R is a ratio that compares the variation within a chain to the variation between chains.

$$R = \frac{\frac{L-1}{L}W + \frac{1}{L}B}{W} \quad (4.5)$$

As B approaches 0 (as the chains become more similar) R tends towards 1. Thus, convergence is said to occur when the R statistic is close to 1, typically less than 1.1.²⁷ For all of the μ values sampled and the σ parameter, the R value is between 1.00 and 1.01 giving good confidence that the sampling has converged.

4.3.1.2 Model Performance

A pre-requisite of the BART model is that the experimental $\log(G')$ used to train the model should be normally distributed and the plot in Figure 4.6a shows that the data is moderately normal with a slight skew. To assess the normality of the $\log(G')$ distribution the Shapiro-Wilk test is used to determine the likelihood that the training $\log(G')$ was drawn from a normal distribution.

The Shapiro-Wilk calculates a statistic, W ²⁸ (Equation 4.6) with the i^{th} smallest value x_i multiplied by the Shapiro-Wilk constant a_i as the numerator and the sum-of-squares as the denominator. If the data is normally distributed the statistic is 1.

$$W = \frac{(\sum_{i=1}^n a_i x_{(i)})^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad (4.6)$$

The Shapiro-Wilk gives a score of 0.98 with a p-value of 0.65 which is larger than the the p-value threshold of 0.05 resulting in a failure to reject the null hypothesis (that the data is normally distributed).

Figure 4.6b shows the visualisation of our descriptor space in a TMAP²⁹ plot. The TMAP plots displays the similarity of molecules as sub-branches within the TMAP plot and the points in the plot have been sized by their G' value. It becomes clear that the datapoints are not easily distinguishable in TMAP space by their G' values and thus it calls for a more complicated procedure to model the relationship between the descriptors and G' .

In the TMAP plot, each sub-branch can be considered as a sub cluster within the dataset. Therefore, any test or validation set points in a sub-branch with few to no training set points are molecules that

may not have a close neighbour in the training set. There are only two test set points that are the terminal points within a branch and one in the validation set. Even in these cases, only the validation set point doesn't exist in a sub-branch with any training set points. Overall, this shows that all points in the test and validation set, bar this validation set point, has a similar molecule in the training set and should lead to good predictive performance for the model.

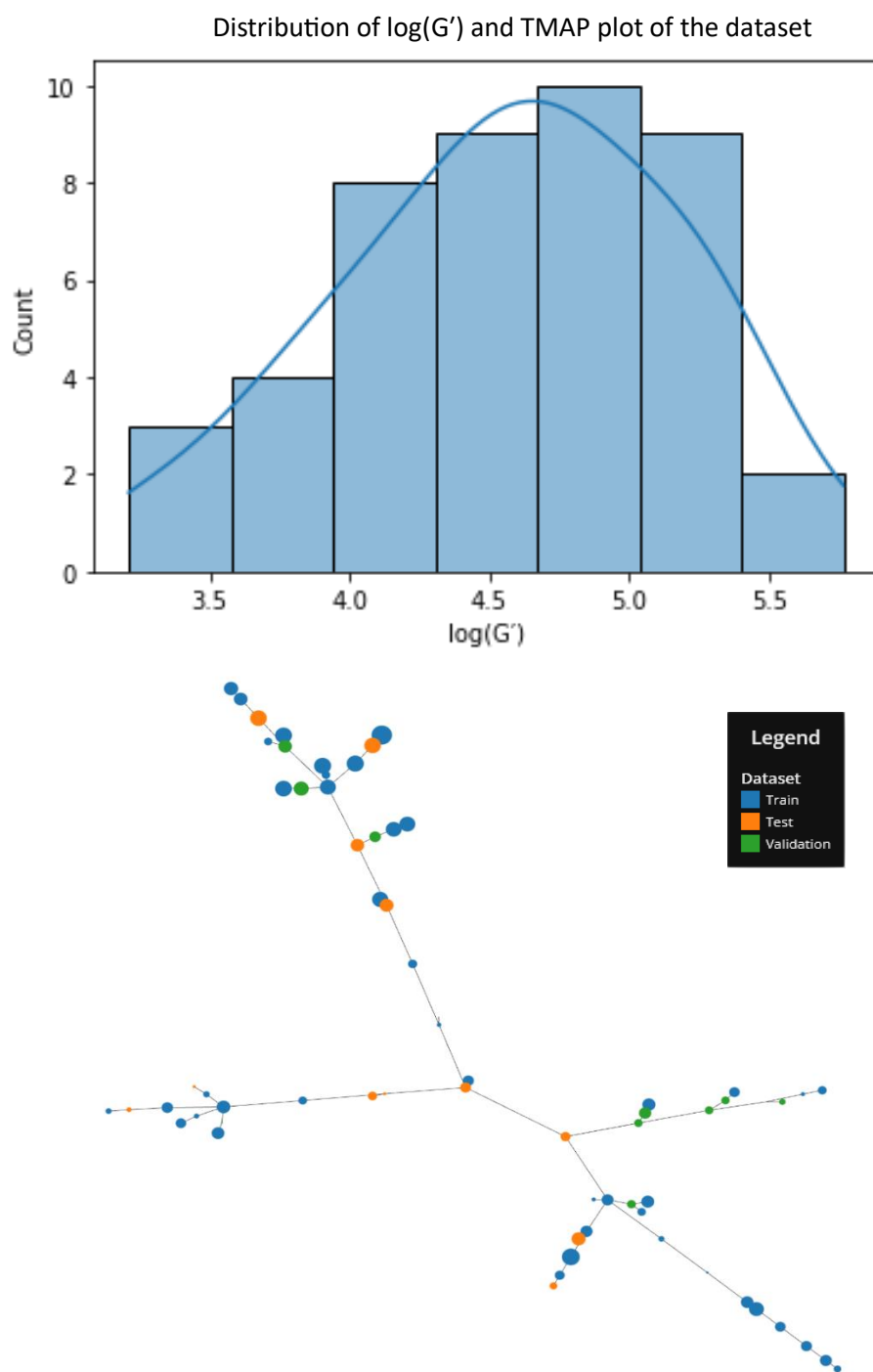


Figure 4.6a) Distribution of the $\log(G')$ values in the training set. b) TMAP representation of the 5_1 dataset with Python descriptors.

Figure 4.7a shows the combined performance of our BART model on the train, test and validation sets and plots the mean of the y posterior against the experimentally derived values. The error bars for the predictions derived from the posterior distribution are provided later in the chapter (see 4.3.1.5 *Test and validation set prediction uncertainty*). Visually, performance across all 3 datasets is good and Table 4.1 contains the performance metrics for all datasets in terms of RMSE and R^2

Performance of our BART model on the test set is good, with the R^2 of 0.64 surpassing our threshold of 0.6. However, the RMSE appears to be slightly higher than the threshold of 0.3 at 0.35 but is reasonably close to the threshold we set but visual inspection shows that performance is still adequate.

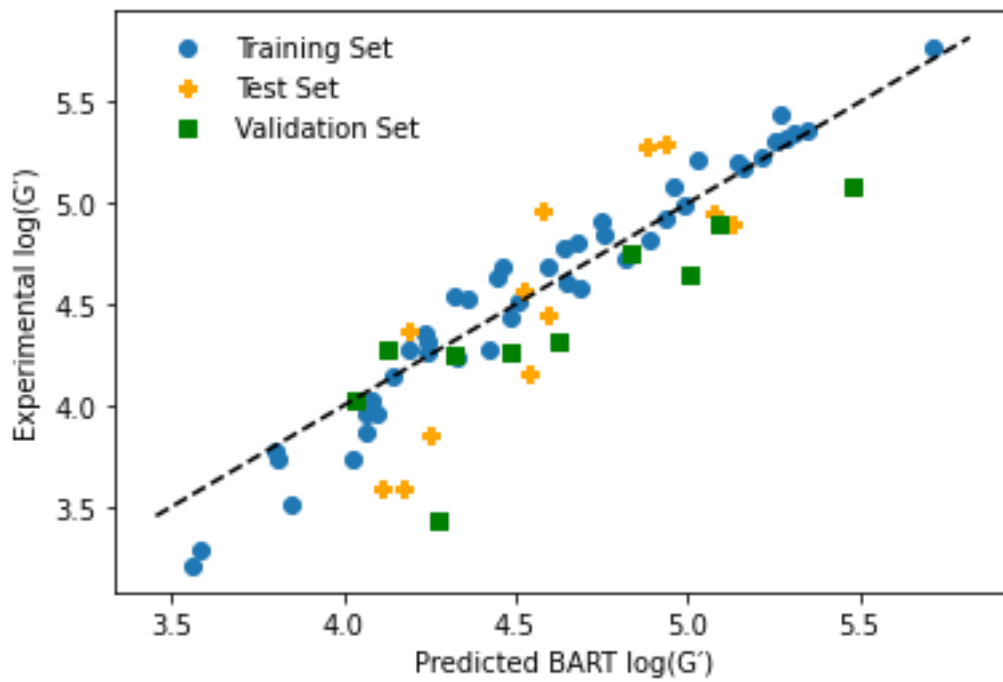
On the validation set, the RMSE shows the same performance as the test set, but the R^2 of 0.41 falls below our 0.6 threshold. On visual inspection, it is likely that the outlier at low G' contributes significantly to this low R^2 . Removal of this point causes the R^2 to improve to 0.51 – still below our threshold.

Table 4.1: R^2 and RMSE values for the train, test and validation sets.

Set	R^2	RMSE
Training	0.94	0.14
Test	0.64	0.35
Validation	0.41	0.35

Figure 4.7b is a replicate of figure 4.6b, but this time the points in the TMAP plot have been scaled by their absolute residual from the BART model. Generally, residuals are larger on test set points that exist within sub-branches of our TMAP plot. Moreover, the points that are terminal points in the TMAP plot do have the highest residuals in the test set (points 1 and 2), suggesting that this could potentially be hampering the predictions.

Experiment vs prediction for the BART G'



TMAP plot sized by residual

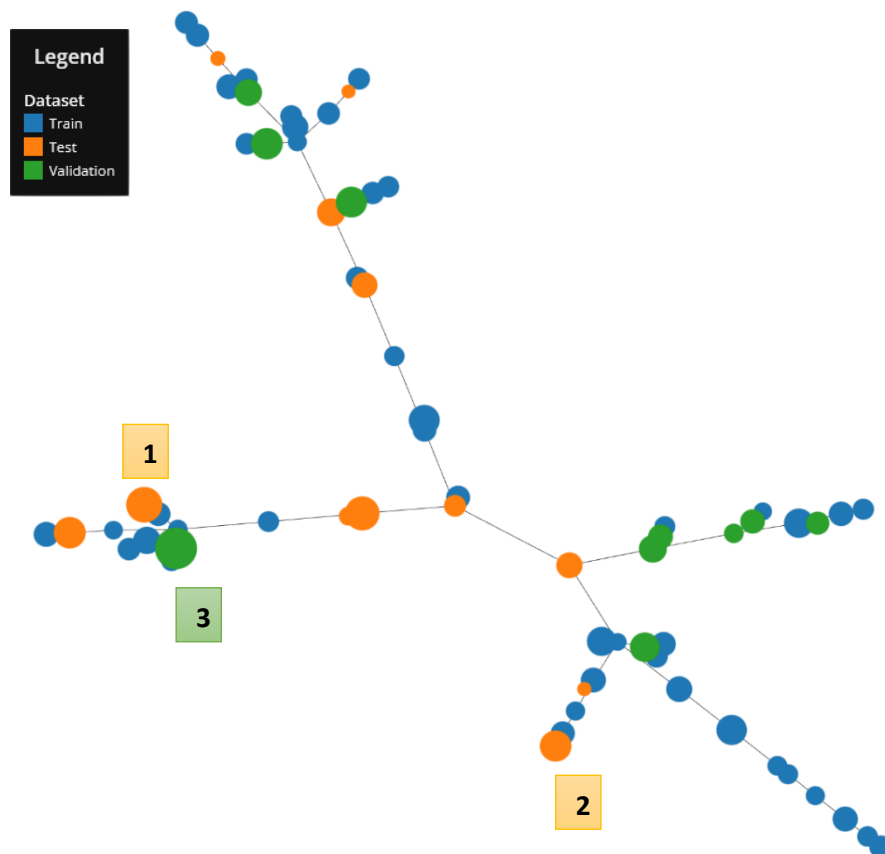


Figure 4.7 a) Performance of the model on the train, test and validation sets. b) TMAP plot for the dataset scaled by the prediction residuals.

However, in the validation set, the large outlier in Figure 4.7a exists in a sub-branch with two training set molecules and these points could potentially be influential in determining the predictions for this point. These points are shown in Figure 8 and are labelled Validation, Train_Term and Train where the Train_Term point is the terminal point in the sub-branch.

It is immediately clear in Figure 4.8 that visually these points are relatively similar, all containing a valine residue (V), but Train and Validation differ in the phenylalanine (F) replacement for one of the valines in the tripeptide Train. Between Validation and Train_Term, the difference in the peptides is a replacement of phenylalanine for glycine in Train_Term. Indeed, the Tanimoto similarity (discussed in detail in 5.3.2.1.4 *Molecular variation within outputs.*) between Validation and Train_Term is high at 0.7 but drops to 0.5 between Validation and Train. Moreover, when comparing the experimental G' values between the similar Train_term and Validation points they are 4.80 and 3.43 respectively which indicates that the poor prediction here is likely due to the existence of an activity cliff³⁰ (where similar molecules exhibit dissimilar properties) and this is potentially the cause for the poor prediction. For reference the average Tanimoto similarity in the combined train, test and validation sets is $0.45 (\pm 0.18)$ showing that Validation and Train_Term are very similar in the context of the dataset.

Chemical depiction of the outlier molecule and TMAP neighbours

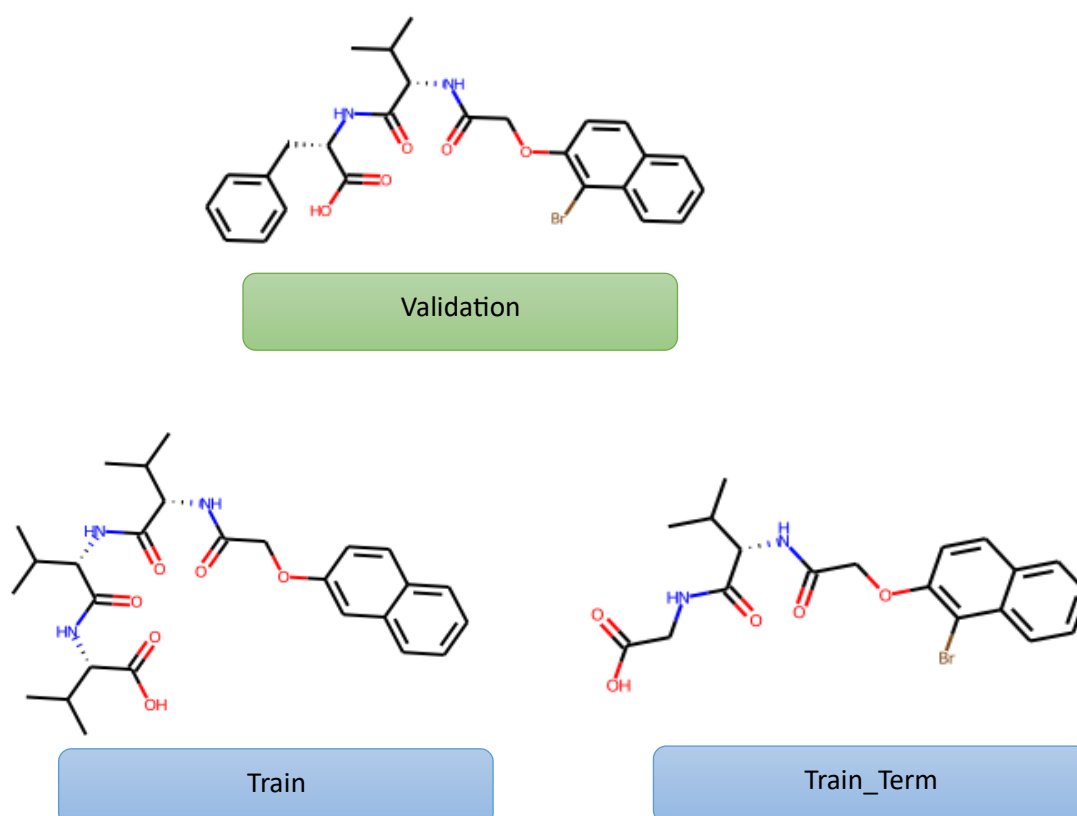


Figure 4.8: Validation set point with the highest residual and the only two training set points in the same sub-branch. Train_Term is the terminal point of the branch.

4.3.1.3 Comparison with our R Random Forest

Figure 4.9 is a reminder of the performance of our R Random Forest³¹ model (see Chapter 3: 3.3.1.5.4 Best Performing Model) for comparison with the BART model. In terms of RMSE, there is very little difference in the performance of the test and validation sets between Random Forest and BART and it is within the R^2 where our BART model appears to suffer in its performance. We see a drop in R^2 from 0.8 to 0.64 on the test set and from 0.71 to 0.41 on the validation set. The RMSE values are generally similar between the R Random Forest and BART model with the validation set RMSE in the R model being slightly lower (0.31) than the BART model (0.35).

However, the benefits of the BART model include an inherent measure in the uncertainty contained within the model's prediction posterior. Therefore, we can investigate the posterior distributions of the predictions and gauge a degree of uncertainty in the model predictions.

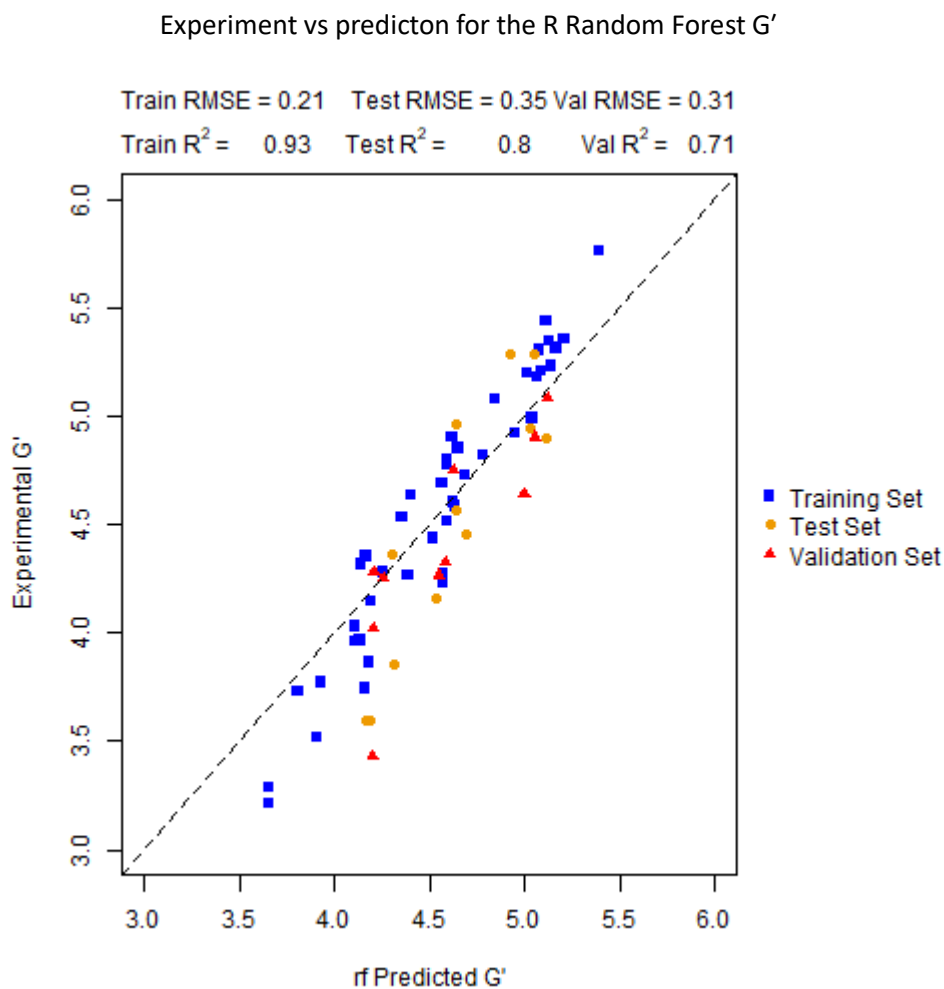


Figure 4.9: Best performing R Random Forest model from Chapter 3.

4.3.1.4 Posterior predictive checks

An example of this extra information can be seen in Figure 4.10 which shows the complete posterior prediction for the highlighted test set points. In this case, the light blue vertical bars indicate the 89% Bayesian credible interval which is the default credible interval for the ArviZ package used here.²⁴ You can see in the posterior plots, in this case the model is much more certain in the more accurate prediction (prediction A) as it shows a much narrower posterior distribution than the more poorly predicted point (prediction B). This information is useful when predicting unknown points to gauge how confident we can be in the model's certainty in the prediction.

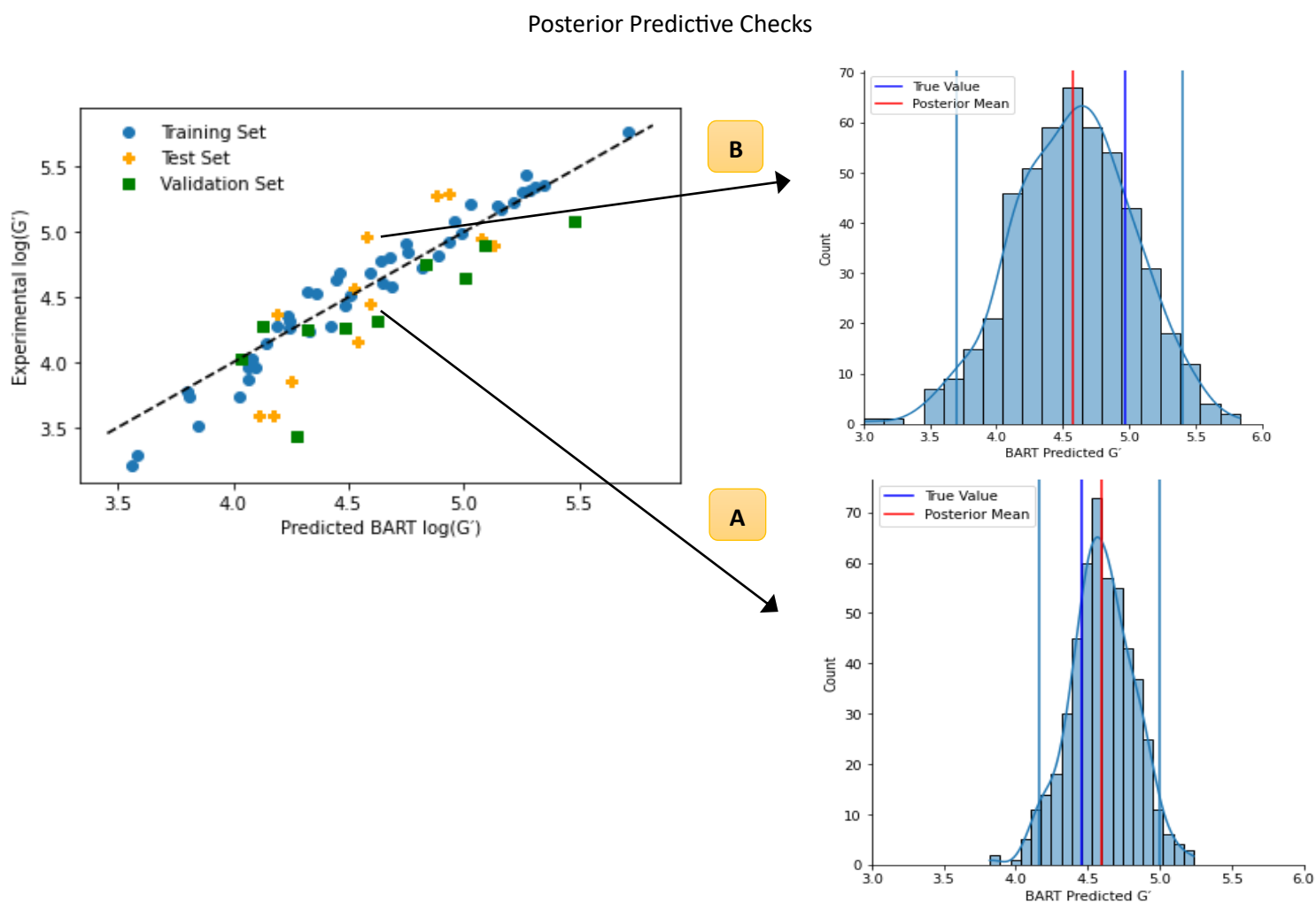


Figure 4.10: Posterior predictive checking for the test set molecules. The light blue vertical lines indicate the 89% Bayesian credible interval for the prediction.

4.3.1.5 Test and validation set prediction uncertainty

We can summarise this certainty in terms of the Bayesian credible interval and update our plots to reflect this. Figures 4.11a and 4.11b show the isolated test and validation set predictions with the 89% credible interval as the error bars. All points bar one in the test set are considered within the range checking of the descriptors and PCA components of the training set. The point out of domain is highlighted in black in Figure 4.11a. Considering only in-domain points, the test RMSE remains 0.35 and the R^2 drops slightly from 0.64 to 0.60. When considering the credible interval, we can gauge more about how close the model is to the true value when accounting for uncertainty.

On the test set, the model struggles to accurately predict at low G' values, where there are three points (labelled 1,2 and 3 in Figure 4.11a and shown visually in Figure 4.12) that fail to overlap with the $y=x$ line even when considering the 89% credible interval. This is due to the inherent nature of the BART model. For each prediction, it starts with the assumption that the predicted value is the mean of the training set and then it uses the descriptors to “pull” the prediction away from the mean. Therefore, if there is an under-represented region of the G' space there is less evidence for predictions to be pulled to low values hence the over-prediction of these points.

The Random Forest in Figure 4.9 also failed to accurately predict these points. Given that these points are all present in the test set at 5 mg/mL and 10 mg/mL and these are predicted moderately well – there is unlikely to be any chemical reasoning underpinning the poor prediction here. It is much more likely that the poor prediction is due to a lack of training data with low G' values in the dataset.

In the validation set all points are considered in-domain and only one point's 89% credible interval fails to overlap with the $y=x$ line in Figure 4.11b. This point is the point shown in Figure 4.8 and the poor prediction here has been attributed to the presence of an activity cliff but the lack of training data at low G' is likely also a contributing factor. These results, coupled with the non-Bayesian models suggest that model performance needs to be improved at low values of G' by including more data within that range in the training set.

Test and validation performance with confidence intervals

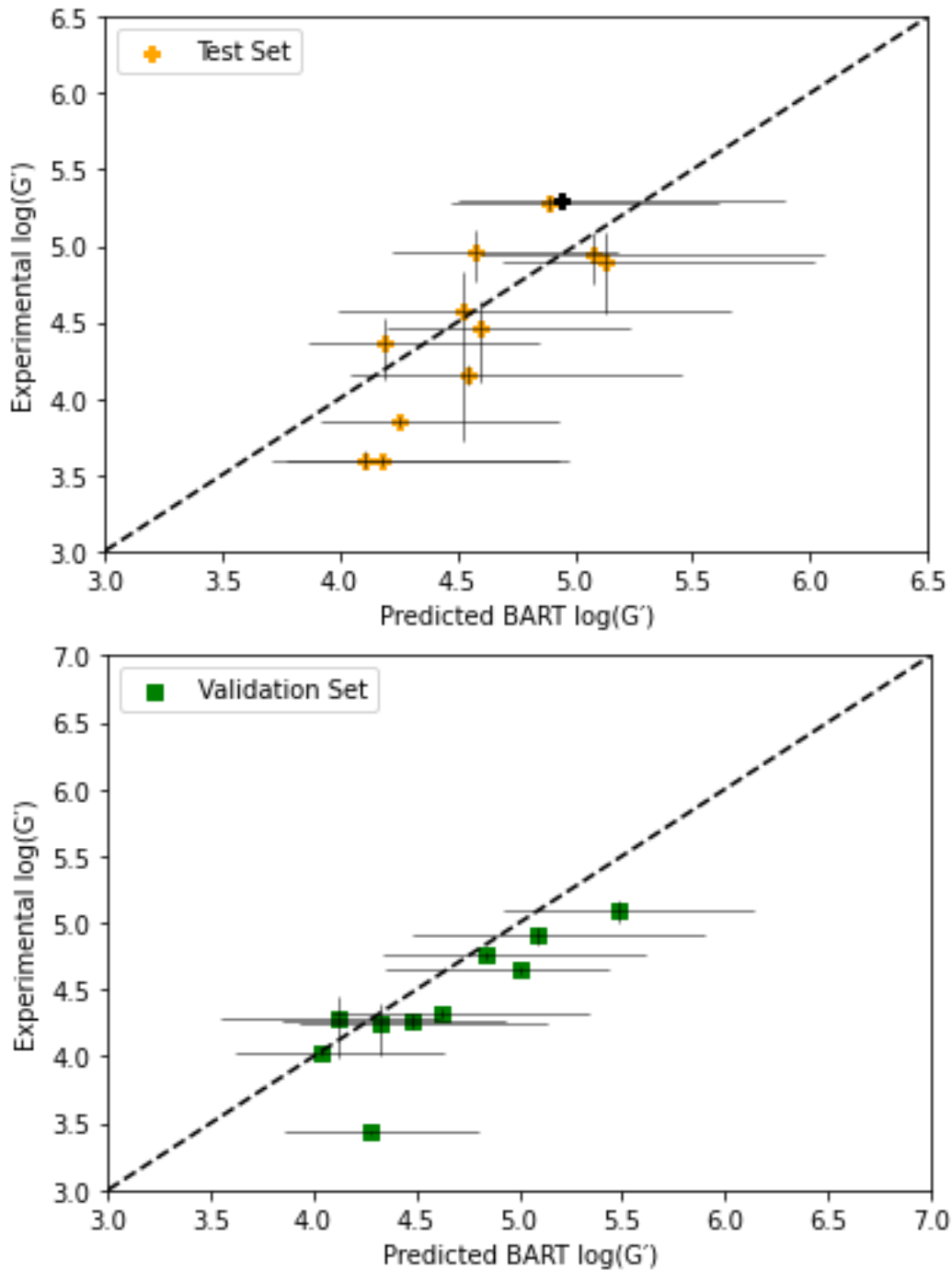
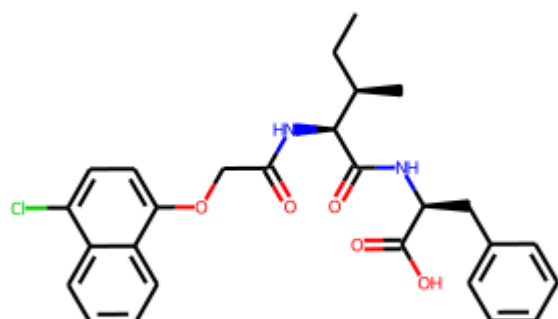
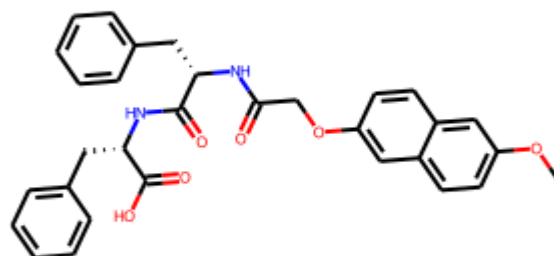


Figure 4.11: a) Test and b) validation set predictions with the 89% Bayesian credible interval shown as error bars on each point. The horizontal error bars correspond with the 89% credible interval. The vertical error bars are the experimental errors.

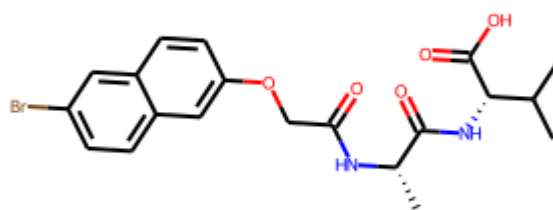
Chemical depictions of the poorly predicted test points



Test_1



Test_2



Test_3

Figure 4.12: Molecules in the test set whose 89% credible interval fails to overlap with the $y=x$ line

4.3.1.6 Model validation via a Y-randomisation check

Table 4.2 shows the result of the y-randomisation approach. A y-randomisation test is carried out to check that the patterns found within the training data of our model were not found by chance and that the same performance couldn't be achieved using any combination of descriptors and G' , this is more likely to occur when using small datasets. This is assessed on the model's ability to accurately predict the in-domain test and validation sets and how much it differs to the true model.

The α values for the R^2 and RMSE are high for both the test and validation sets. On the test set there is 0.49% confidence that the true model is drawn from the same distribution as the random models based on the R^2 and this is 0.13% when considering the test RMSE. Both values give high confidence that our model is finding a true relationship between the molecular descriptors and G' . On the validation set the confidence is lower for both R^2 and RMSE at 3.56% and 2.28% but confidence is still high.

Table 4.2: Y-randomisation results

Set	Z-Score (R^2)	α (R^2)	Z-Score (RMSE)	α (RMSE)
Test	2.58	99.51%	3.00	99.87%
Validation	1.80	96.44%	2.00	97.72%

An example of a model built using the y-randomisation approach is shown in Figure 4.13. It is clear from Figure 4.13 that there is no relationship between experiment and prediction across all three of the train, test and validation sets. This aligns closely with the high confidence obtained from Table 4.2 and gives confidence to predict the external set using this model.

Experiment vs prediction for a randomised model

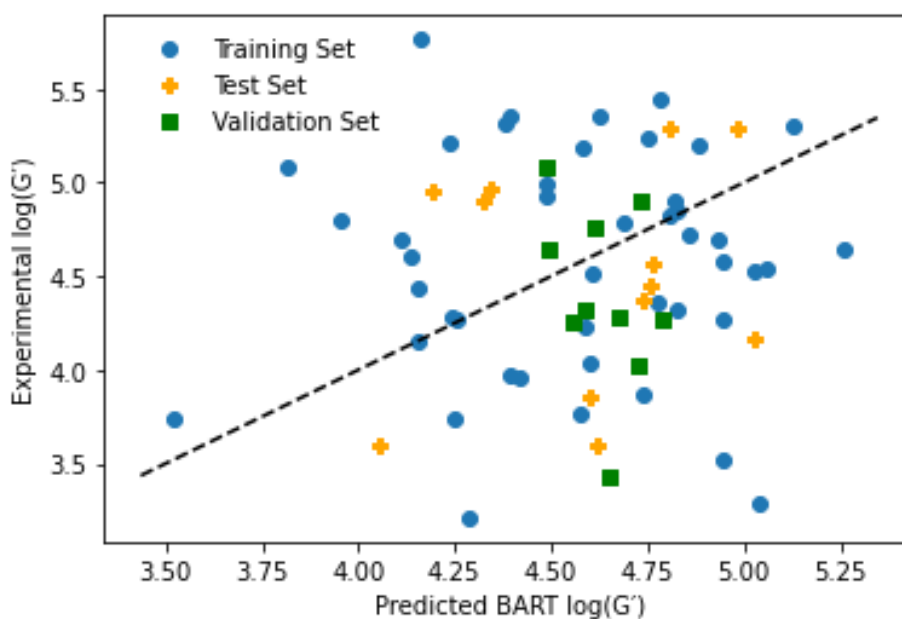


Figure 4.13: Performance of a randomly chosen single randomised model of the 50 trained on the train, test and validation set.

4.3.1.7 External set prediction

All of the points in the external set (see Table 4.3) are within the applicability domain of the training set. The performance of the model on this set is shown in Figure 4.14 and performance is mixed. Statistically, the RMSE is relatively high at 0.55 and the R^2 is -1.99 suggesting that the predictions on this set are worse than merely predicting each point to be the mean value of the set. This is concerning and the first time that the predictions have been this poor.

However, when accounting for the 89% credible interval – these intervals are large, particularly compared to the plots in 4.11a and 4.11b on the test and validation sets. Therefore, this suggests an element of uncertainty in the model for these predictions and it is reassuring that the poor prediction is accompanied by large uncertainty.

Of the molecules in the external set, only two fail to reach the $y=x$ line when considering the 89% credible interval and these are shown in Figure 4.15 as ext_2 and ext_8. Ext_2 is structurally very similar to the Train_Term molecule above, being the non-brominated analogue of this molecule. When comparing the experimental G' values of Ext_2 and Train_Term they are 4.26 and 4.82 (for the 5 mg/mL Train_Term) which makes the prediction of Ext_2 of 5.19 rather surprising and making this point an outlier.

Experiment vs prediction on the external set

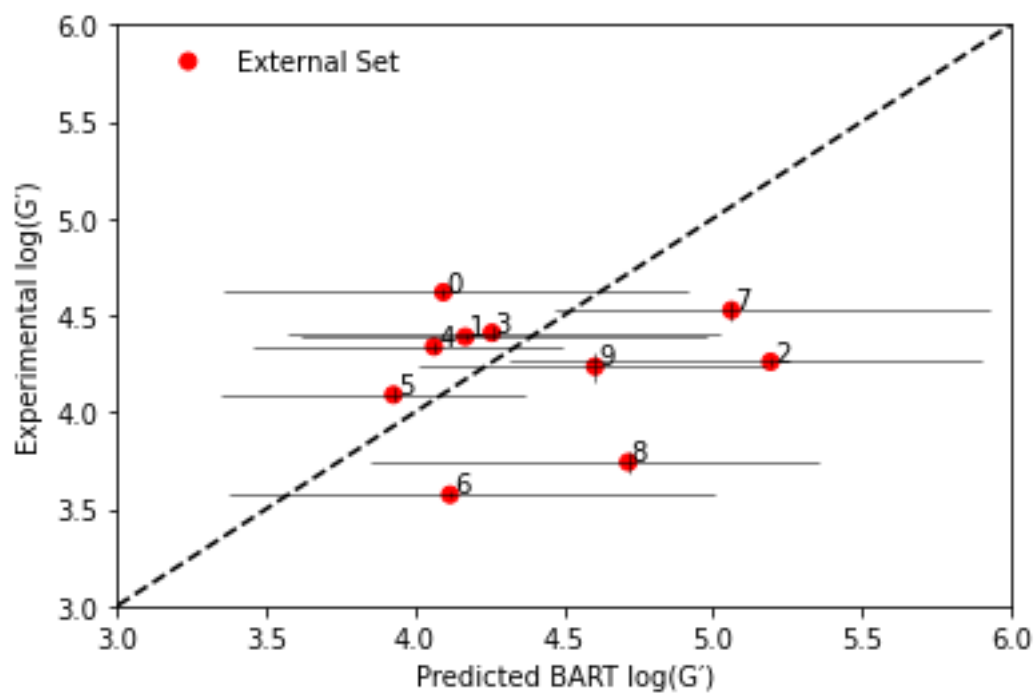


Figure 4.14: Performance of the BART G' model on the external set of molecules. The horizontal error bars correspond with the 89% credible interval. The vertical error bars are the experimental errors.

Chemical depictions of the poorly predicted external points and their TMAP neighbours

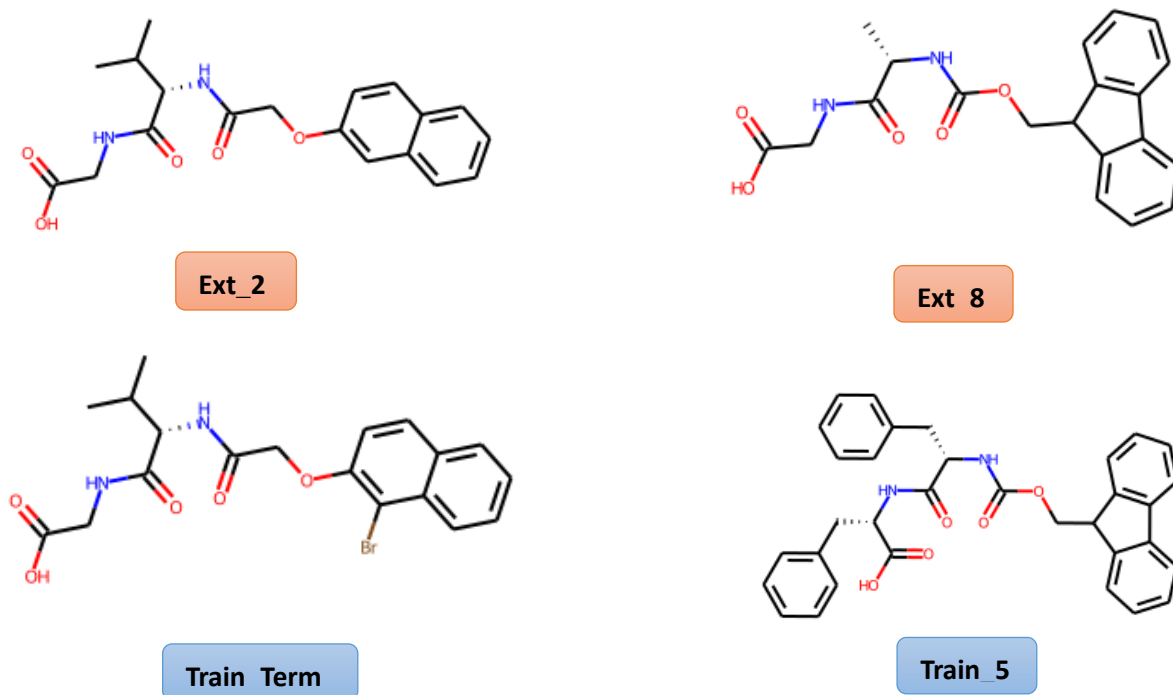


Figure 4.15: External molecules 1 and 2 that fail to overlap with the $y=x$ line with the 89% Bayesian credible interval and the nearest training set point to both points (Train_Term and Train_5).

Figure 4.16 shows the TMAP representation of the dataset with the external set points added. Ext_8 exists as part of three molecules the end of a branch with two other validation set points (the only other two Fmoc gels in the external set) and is the central of the three points. The nearest training set point is another Fmoc based gel but the Train_5 is an FmocFF gel whereas Ext_8 is FmocAG. The Tanimoto similarity for these points is somewhat low at 0.54 close to the similarity threshold of 0.5 that is defined in Chapter 5: 5.3.2.1.4 *Molecular variation within outputs*. Moreover, the experimental G' for the Train_5 point is 4.58 whereas Ext_8 has an experimental G' of 3.74 indicating an activity cliff here.

TMAP plot sized by external residuals

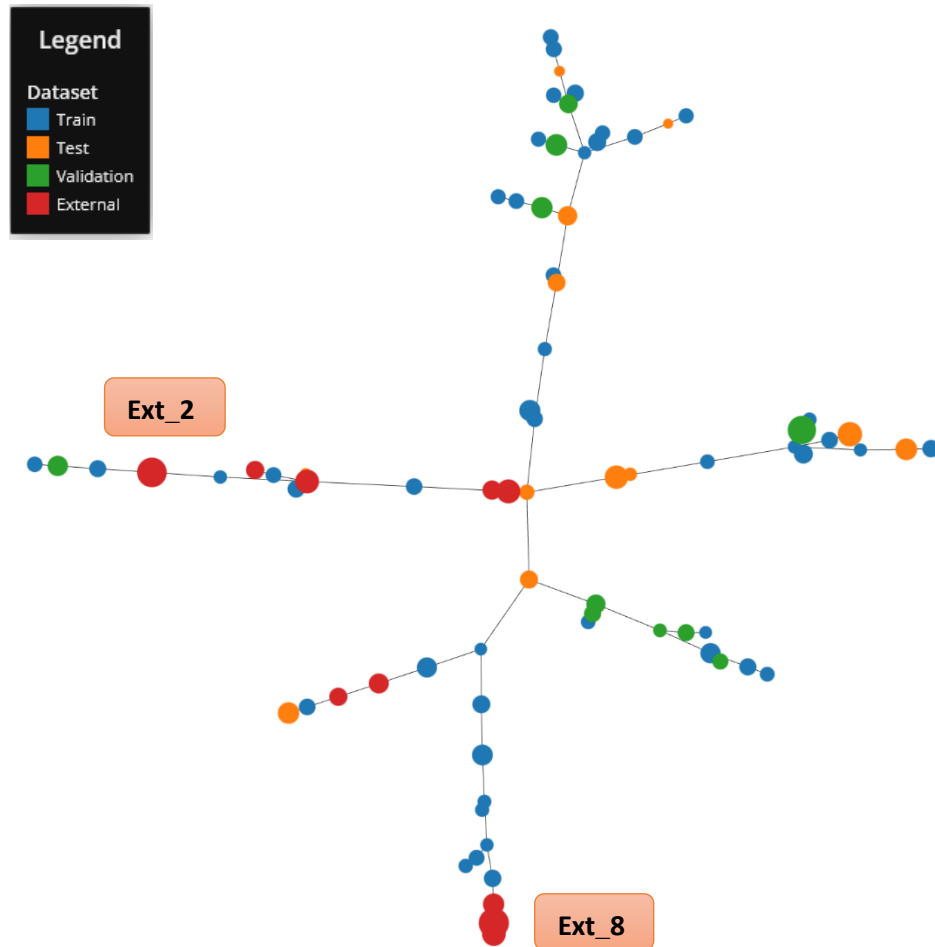
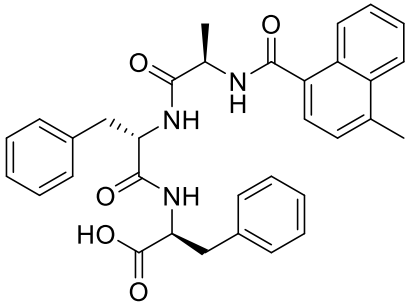
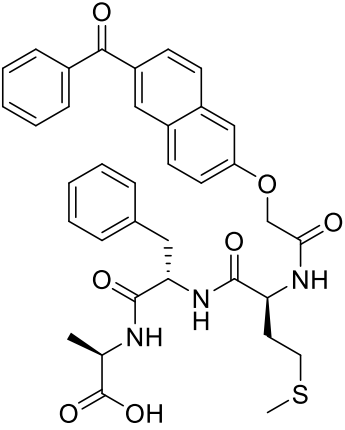
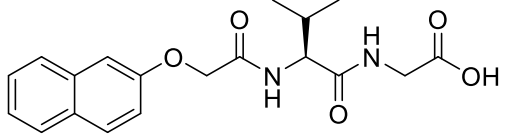
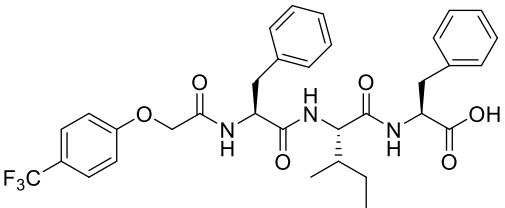
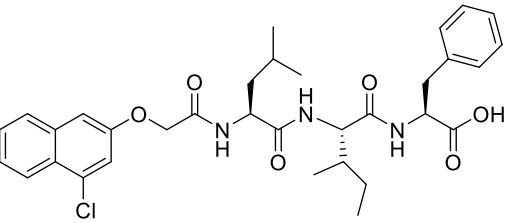
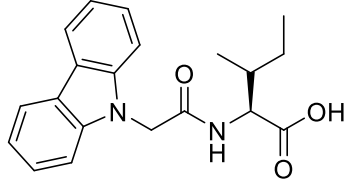
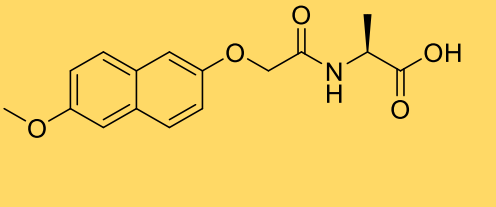
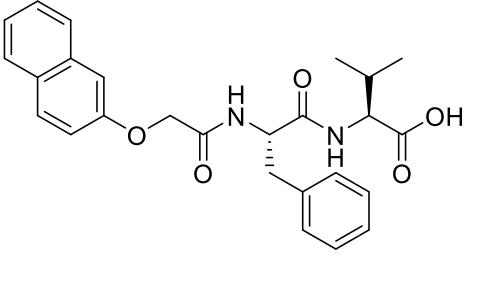
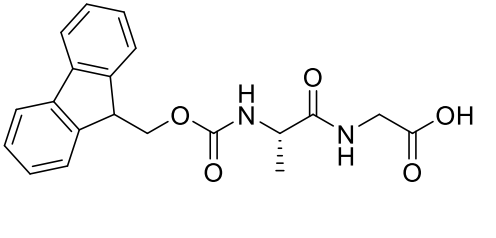
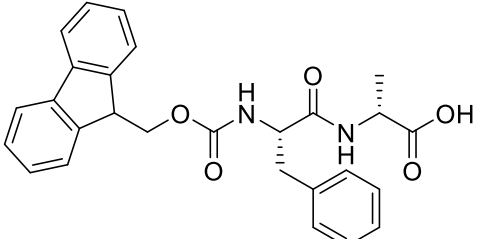


Figure 4.16: TMAP plot of the train, test and validation sets with the external set added in red.

Table 4.3: Molecules comprising the external set. Note these are the same molecules as those shown in Table 4 in Chapter 3.

Gel	Concentration (mg/mL)	Structure	G'	G' error	G''	G'' error
1 – Molecule 6	5.0		41600 (4.62)	1670	7460 (3.87)	577
2 – Molecule 15	5.0		24500 (4.39)	941	4630 (3.67)	230

3 - 2NapV G	5.0		18200 (4.26)	1040	3630 (3.59)	344
4 - CF3PhF IF	5.0		25700 (4.41)	1800	2770 (3.44)	106
5 - 4ClNap LIF	5.0		21800 (4.34)	1360	3150 (3.50)	182
6 - Carbl	5.0		12300 (4.10)	983	454 (2.66)	39

7 - 6MeON apA	5.0		3750 (3.57)	116	187 (2.27)	32
8 - 2NapFV	5.0		33600 (3.53)	4040	3460 (3.54)	326
9 - FmocA G	5.0		5520 (3.74)	705	966 (2.98)	87
10 - FmocFA	5.0		17200 (4.24)	2970	2590 (3.41)	621

4.3.1.8 Model Interpretation

Since the model has shown promising performance thus far, the model will be interpreted using the SHAP approach³² (see 2.3.2.9.2.1 *RF Shap Importance*). The SHAP approach considers descriptors in batches as a “coalition” and predicts the G' value on this reduced dataset and compares the difference to the prediction with the full dataset. The coalition is assigned a “score” which is the difference between the full set prediction and the coalition’s prediction. This is averaged over coalitions to give an idea of how much each descriptor, on average, impacts the predicted value.

The results of the SHAP approach are shown in Figure 4.17 below as a Beeswarm plot. The plot is divided into 2.5, 5 and 10 mg/mL plots respectively. The beeswarm y axis is ordered by the most important descriptor by average absolute SHAP value. Each point on the plot in each “row” in the y-axis is a single point in the in-domain test and validation sets. It is situated along the X axis by its SHAP value for that descriptor for each molecule. Each point is coloured by the descriptor value, scaled between 0 and 1 where 1 is the maximum value that descriptor takes in the test and validation sets with 0 being the minimum. Each fragment is presented in Figure 4.18 as red circles on a training set molecule.

Beeswarm plot of the BART G' model

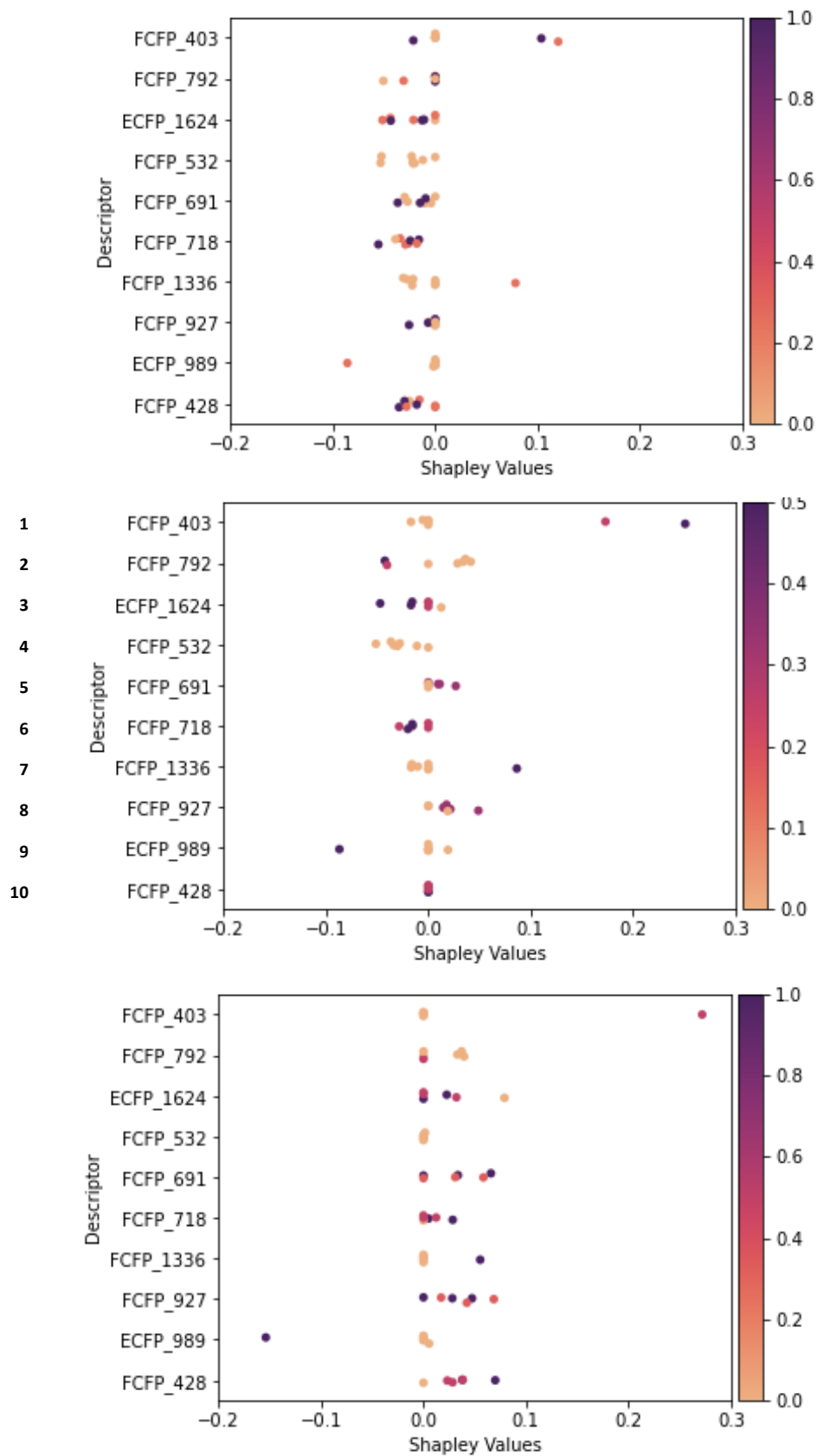
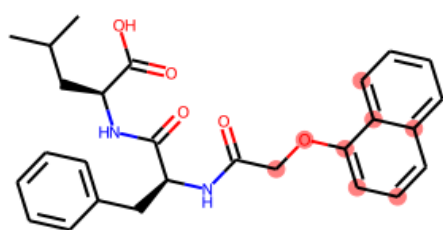
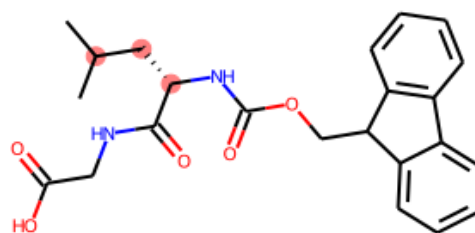


Figure 4.17: Beeswarm plot of the SHAP values for the G' model at a) 2.5 mg/mL b) 5 mg/mL and c) 10 mg/mL

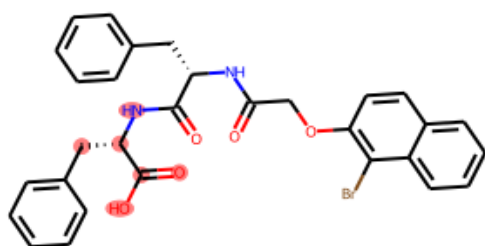
Chemical depictions of the most important fragments



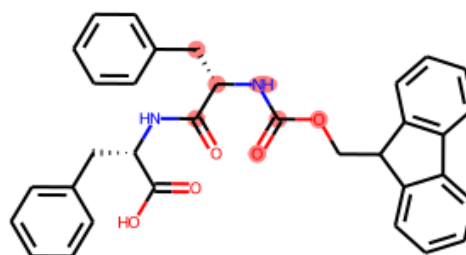
FCFP_403



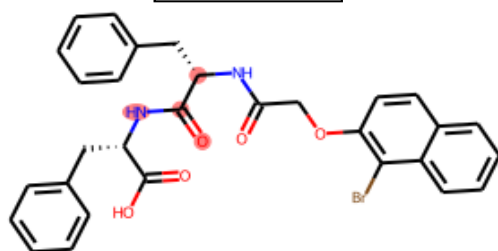
FCFP_792



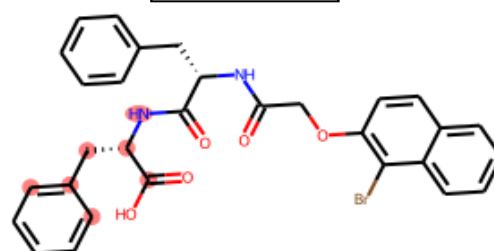
ECFP_1624



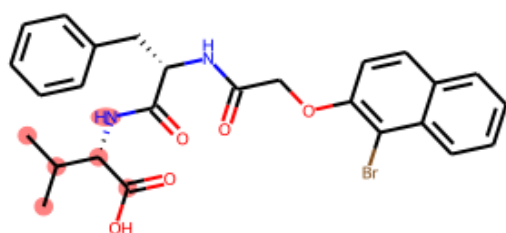
FCFP_532



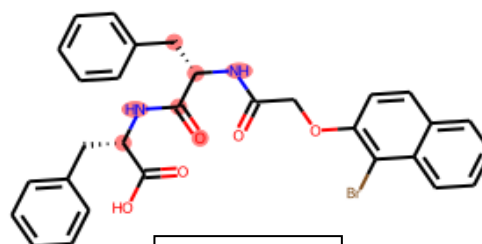
FCFP_691



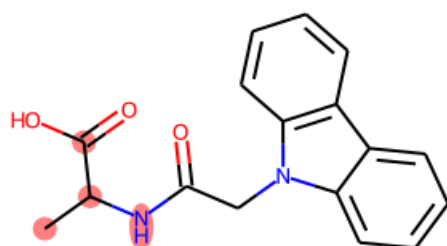
FCFP_718



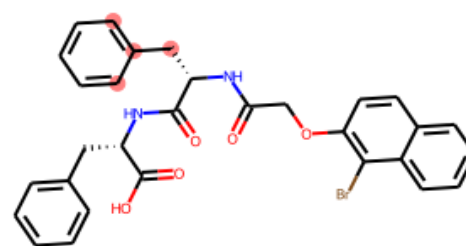
FCFP_1336



FCFP_927



ECFP_989



FCFP_428

Figure 4.18: Fragment representation shown on a training set molecule

It is immediately clear in the Figure that the model quickly learns to assign negative attribution to the points at 2.5 mg/mL and positive to 10 mg/mL. The hypothesis here is that since BART initially predicts the mean of the training set and then uses the descriptors to pull away from this value, the model quickly learns that most 2.5 mg/mL points are below the mean and most 10 mg/mL are above. Indeed, the mean G' in the training set is 4.57 and the highest G' in the test and validation set at 2.5 mg/mL is 4.36. For 10 mg/mL the minimum value is 4.90. As for 5 mg/mL this concentration has values both above and below the mean and so information about the importance of descriptors can be extracted from the beeswarm plot (Figure 17b) of this concentration.

The representations of the important fragments are highlighted on training set molecules in Figure 18. Again, as found in the non-Bayesian RF SHAP (see 3.3.1.9.2 *Random Forest*) fragments set by phenylalanine are ever-present in the top 10. ECFP_1624 (3rd), FCFP_532 (4th), FCFP_718 (6th) FCFP_927 (8th) and FCFP_428 (10th) can all be set by phenylalanine. Of these fragments, only ECFP_1624 has a value dependence on the SHAP score with an increase in this fragment causing a negative SHAP score and a decrease in the predicted G' . This bit is set by a phenylalanine at the C-terminus and this was also seen in the R- kNN model (see 3.3.1.9.1 *k-Nearest Neighbours*) and again shows that the importance being placed on the relative position of the phenylalanine, reinforcing the idea that it is less beneficial to have a terminal phenylalanine, as seen in the Boc-Phe-Aib-OH (Figure 4.19) work³³ discussed in 3.3.1.10.1.

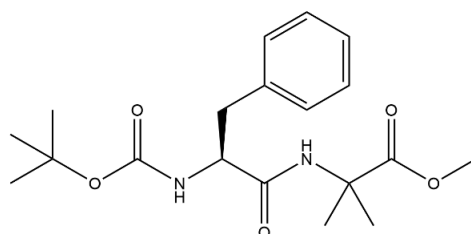


Figure 4.19: Boc-Phe-Aib-OH

Of the remaining fragments, the most important overall fragment, FCFP_403 is set by a naphthalene with a methoxy linker. In the R- Random Forest models (see 3.3.1.10.2 *Random Forest*) the methoxy linker was again seen to be important but there was no value dependence on the SHAP value. Here, there is a clear positive correlation with higher values of the FCFP_403 fragment resulting in a larger predicted G' . This would be expected based upon the experimental results introduced in 3.3.1.10.2 which showed that the methoxy linker promoted gelation over alkyl linkers³⁴ and the methoxy linker results in more beneficial bond angles for self-assembly.³⁵

There are three fragments in the top 10 set by hydrophobic alkyl chains, FCFP_792 (2nd) and FCFP_1336 (7th) are both set by valine and ECFP_989 is set by alanine. Of these, only FCFP_792 has a SHAP-value dependence with it showing a negative correlation, even though hydrophobicity is known to increase self-assembly in hydrogels³⁶ thus suggesting that the dependence here is more nuanced.

Now the analogous work is carried out for the G'' work on the same dataset and the performance compared to the best non-Bayesian model reported in Chapter 3 - 3.3.2.2 *RF models*

4.3.2 G'' BART model

4.3.2.1 Model Convergence and Assumptions

Again, the PyMC3 BART model must be checked for sufficient convergence of the MCMC sampling. Figure 4.20 shows the traceplot of the G'' BART model and the Gelman-Rubin statistic²⁶ for each parameter value sampled is again between 1.00 and 1.01 giving good confidence that the sampling chains have converged to the true posterior distribution.

The assumption of normality for the training G'' values is again assessed via the Shapiro-Wilk²⁸ normality and the statistic is 0.98 with a p-value of 0.60 giving high confidence that the data is normally distributed. The distribution can be seen in Figure 4.21.

BART G'' traceplot

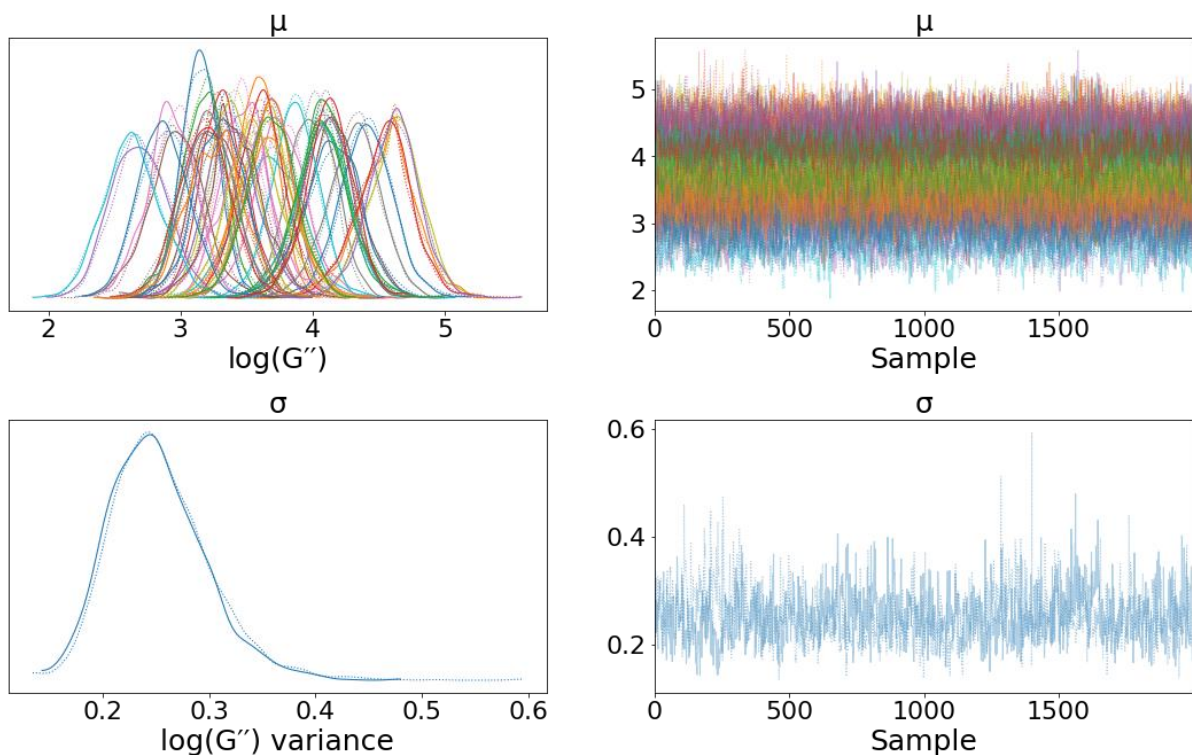


Figure 4.20: Convergence of the BART G'' model

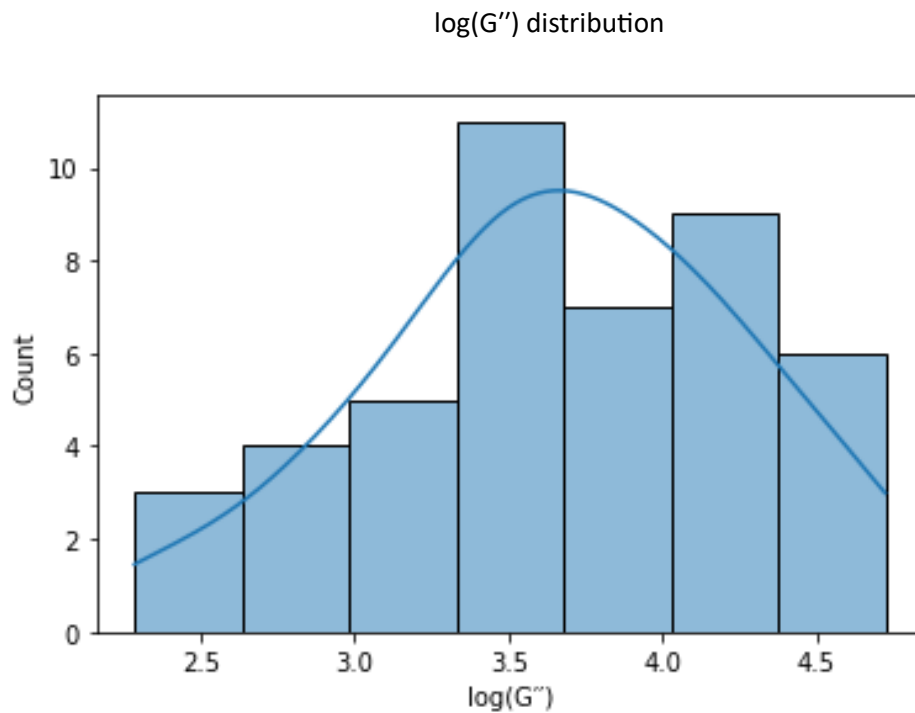


Figure 4.21: Distribution of $\log(G'')$ values in the training set

4.3.2.2 Model training and performance

The performance of the G'' model on the train, test and validation set is shown in Figure 4.22. The plot in Figure 4.22 shows overall good agreement between experimental and predicted values. Again, as in the G' work there are outliers at low values of G'' which are the same points as the validation and test set points highlighted in Figures 4.8 and 4.12 further emphasising the need for further model building with more data in this region. The test set predictions appear to be well correlated with the experimental values, but the majority of the test set points seem to be scattered around the $x=y$ dashed line.

Table 4.4 highlights the performance of the model in terms of R^2 and RMSE and shows overall similar performance to the G' model but much better performance on the validation set. For the test set, R^2 is below the 0.6 threshold at 0.5 and the RMSE is high at 0.40 compared to the threshold set of 0.30 but visually performance is good. For the validation set, the dataset performs very well even accounting for the known point that is poorly predicted. R^2 is above the threshold at 0.73 and RMSE is low at 0.23, comfortably below the 0.3 threshold.

Experiment vs prediction for the BART G'' model

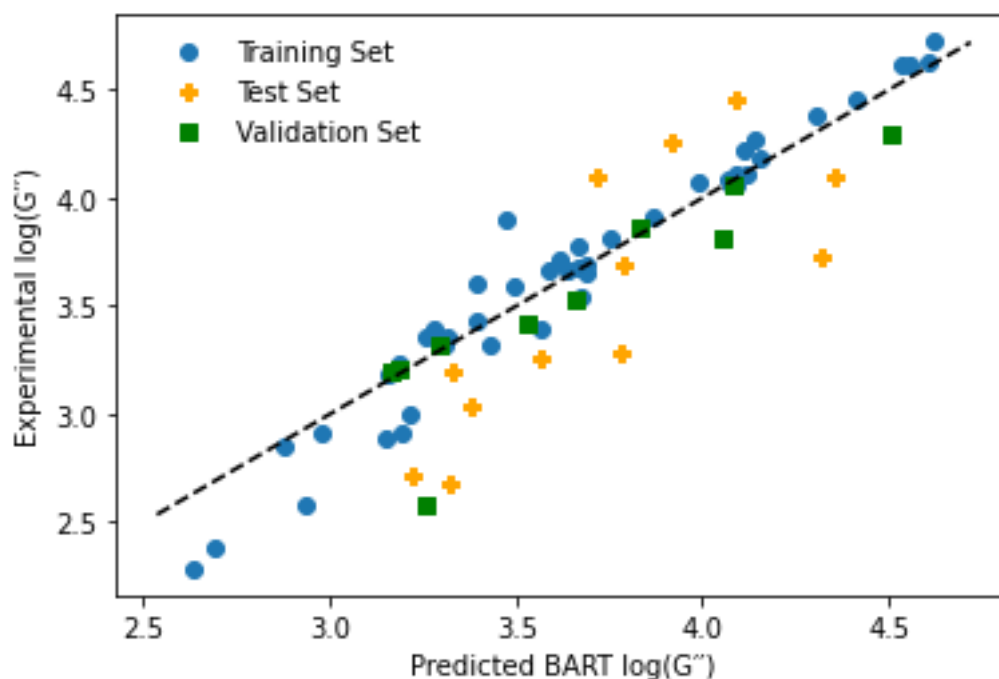


Figure 4.22: Performance of the G'' model during training, test and validation.

Table 4.4: Performance of the G'' model on the train, test and validation sets.

Set	R^2	RMSE
Training	0.94	0.15
Test	0.50	0.40
Validation	0.73	0.24

4.3.2.3 Test and validation plots with confidence interval bars

Since the G'' dataset is identical to the G' one, there is again only a single test set point out of domain, this is the black point in the confidence interval plots in Figure 4.23a and 4.23b. The confidence interval plots for the G'' test and validation set mostly follows the same pattern as the G' model, with the low G'' values being poorly predicted and still failing to encompass the true value in the 89% credible interval range. However in the G'' plot there is an additional point whose 89% credible interval fails to overlap with the $y=x$ line (test_4 - 3MeONapFF) which is shown in Figure 4.24.

From the TMAP plot (Figure 4.25) 3MeONapFF (Figure 4.24) has three connected neighbours, but only one of these is present in the training set. The point, shown in Figure 4.26 is 1BrNapFF which shows

moderate similarity in terms of Tanimoto similarity at 0.66. The 1BrNapFF G'' is 4.22 and it is clear in Figure 24a that the model is predicting 3MeONapFF to be closer to its structurally similar neighbour.

Test and validation performance with confidence intervals

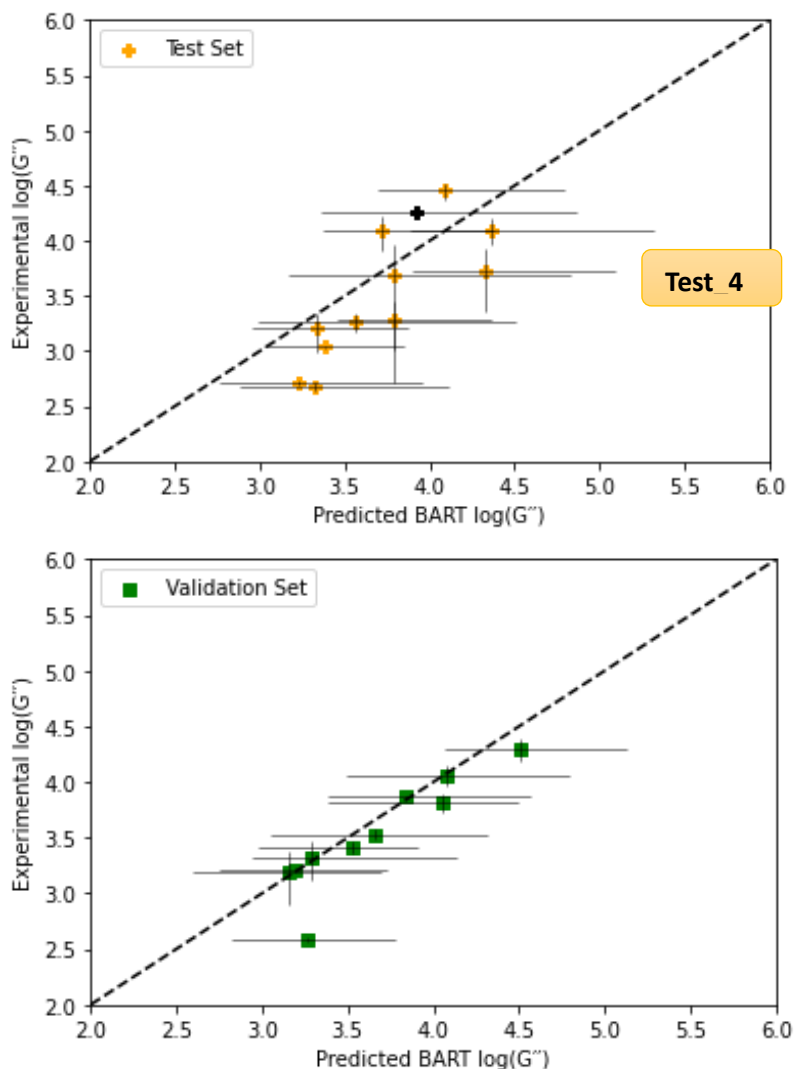


Figure 4.23: a) Test and b) validation set predictions. The horizontal error bars correspond with the 89% credible interval. The vertical error bars are the experimental errors. The single out-of-domain point is highlighted in black.

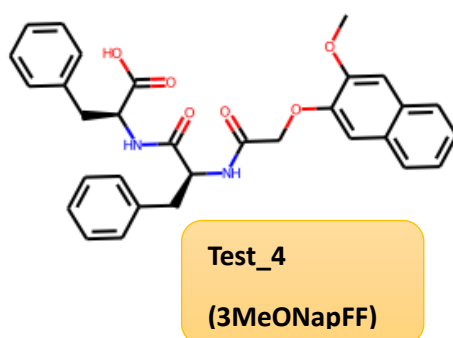


Figure 4.24: Test point 4 that fails to overlap with the $y=x$ line

TMAP plot of the G'' dataset

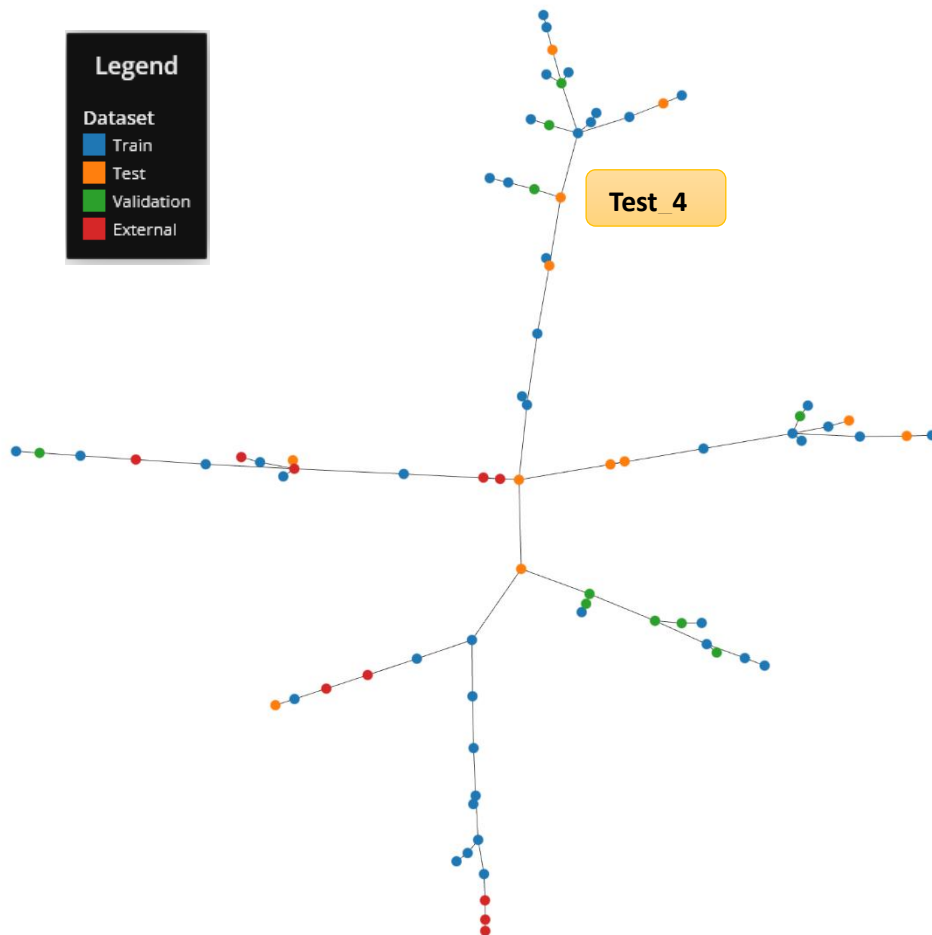


Figure 4.25: Replication of the TMAP plot of Figure 16 with Test_4 highlighted. Note the nearest neighbour below Test_4 is the orange test point and not the blue train point.

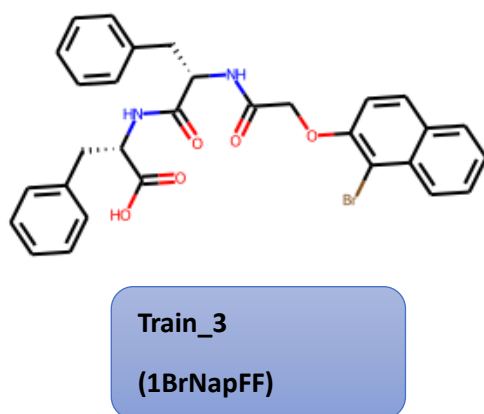


Figure 4.26: Training set point which is the nearest neighbour to Test_4 (3MeONapFF)

4.3.2.4 Validation via Y-Randomisation

The Y-randomisation process was also carried out for the G'' model and we obtain a similar result again to the G' model. The Z-score in Table 4.5 for the G'' model on the in-domain test set is 2.38 for the R^2 and 2.89 for the RMSE resulting in 0.85% and 0.19% that the models are a chance correlation. More confidence is seen on the validation set as the Z-Scores are 2.72 (R^2) and 3.75 (RMSE) giving 0.33% and 0.01% confidence from both that the true model is very likely drawn from a different distribution than the y-randomisation models. From this, we can conclude again that the model is finding a true relationship between the descriptors and G'' and hasn't happened-upon a chance correlation. Figure 4.27 shows an example of the y-randomised model and the poor visual correlation between experiment and prediction is evident in the Figure. The black dashed line indicates the $y = x$ diagonal.

Table 4.5: Y-randomisation results for the G'' model.

Set	Z-Score (R^2)	α (R^2)	Z-Score (RMSE)	α (RMSE)
Test	2.38	99.15%	2.89	99.81%
Validation	2.72	99.67	3.75	99.99%

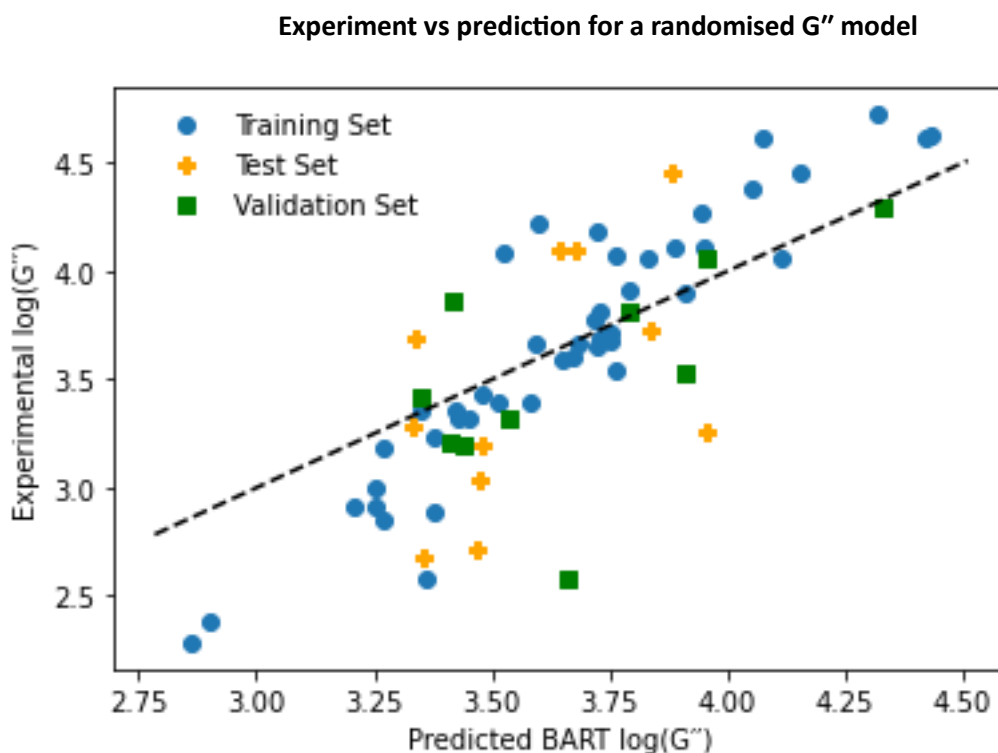


Figure 4.27: First randomised model showing no real visual correlation between experiment and prediction.

4.3.2.5 External set prediction

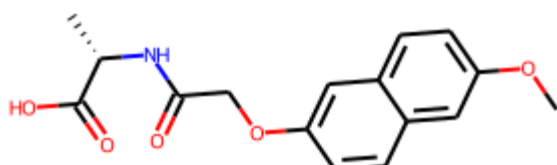
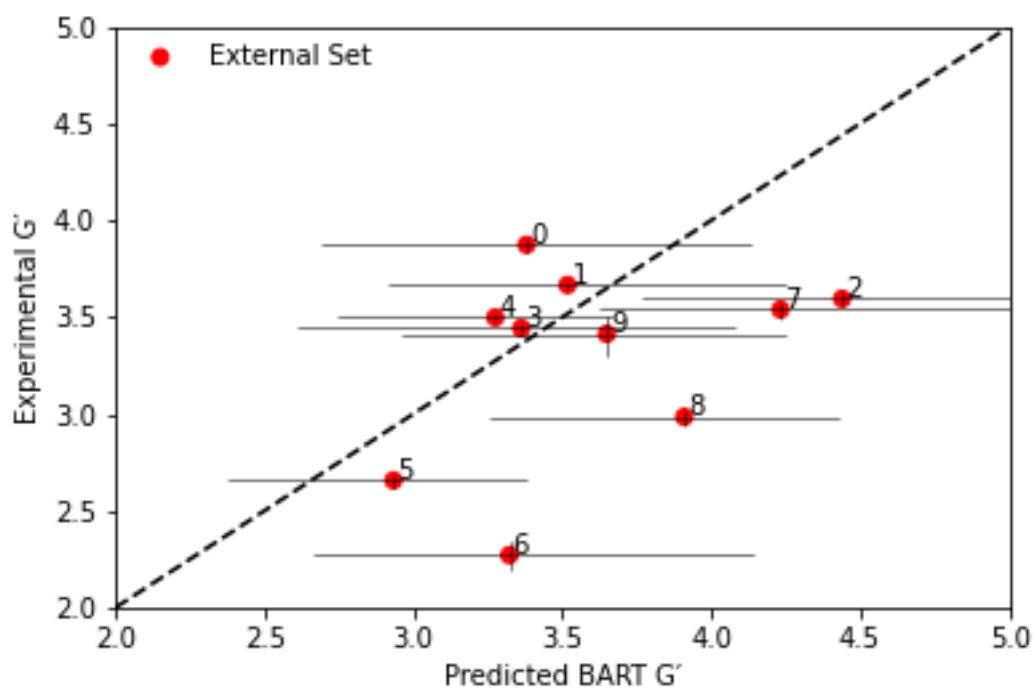
Performance of our G'' model on the external set (Figure 4.28a) is similar to that of the G' with an R^2 of -0.60 and an RMSE of 0.60, both of these are far from the thresholds we set. 6 of the 10 points overlap the $y=x$ line with the 89% confidence interval considered. Of the remaining 4 that don't, points 2 and 8 also failed to overlap the $y=x$ line in the G' model and the other two points 6 and 7 barely overlap with the $y=x$ in the G' model.

Molecules 6 and 7 are shown in Figure 4.28b. The TMAP plot in Figure 4.29a shows the relationship of molecules 6 and 7 to the training set. Molecule 6 (6MeONapA) has no close neighbours in the training set and this could explain the poor prediction here and further reinforces the need for more data at low G'' . Molecule 7 (2NapFV) closest neighbours are a test set point and forms two further sub-branches with two training molecules. It is also connected on the minimum spanning tree branch by two further training set points but the distance to these points is large. The training set point that forms a sub-branch containing only this point is formed by train_19 - 1Br2NapFV (Figure 4.29b) and the training set point that is part of the sub-branch containing a further external set point is train_28 - 6MeO2NapFI (Figure 4.29b)

The G'' values for 1Br2NapFV and 6MeO2NapFI are 3.91 and 3.68 respectively, not dissimilar to the 3.53 of 2NapFV. Moreover, this doesn't account for why the model predicts the value above these points at 4.23.

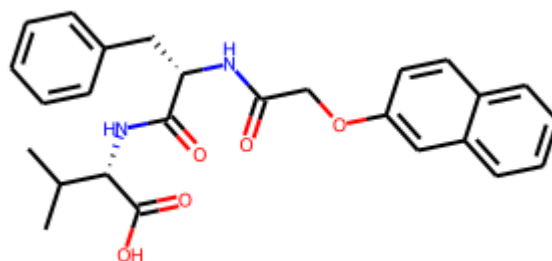
Returning to the two training set points that were on the same spanning tree but considered "far" – the Tanimoto similarity between 2NapFV and these two points, 2NapFF and 2NapVVV, is largest for the entire training set at 0.78 for both. These molecules at the same concentration have G'' values of 4.07 and 4.11 and are therefore more likely to be more influential in the prediction for 2NapFV.

Experiment vs prediction on the external set



Ext_6

6MeONapA



Ext_7

2NapFV

Figure 4.28: a) Performance of the G'' model on the external set. The horizontal error bars correspond with the 89% credible interval. The vertical error bars are the experimental errors. b) Molecules 6 and 7 who don't overlap with the $y=x$ line in the G'' model

External molecules 6 and 7 highlighted on the TMAP plot

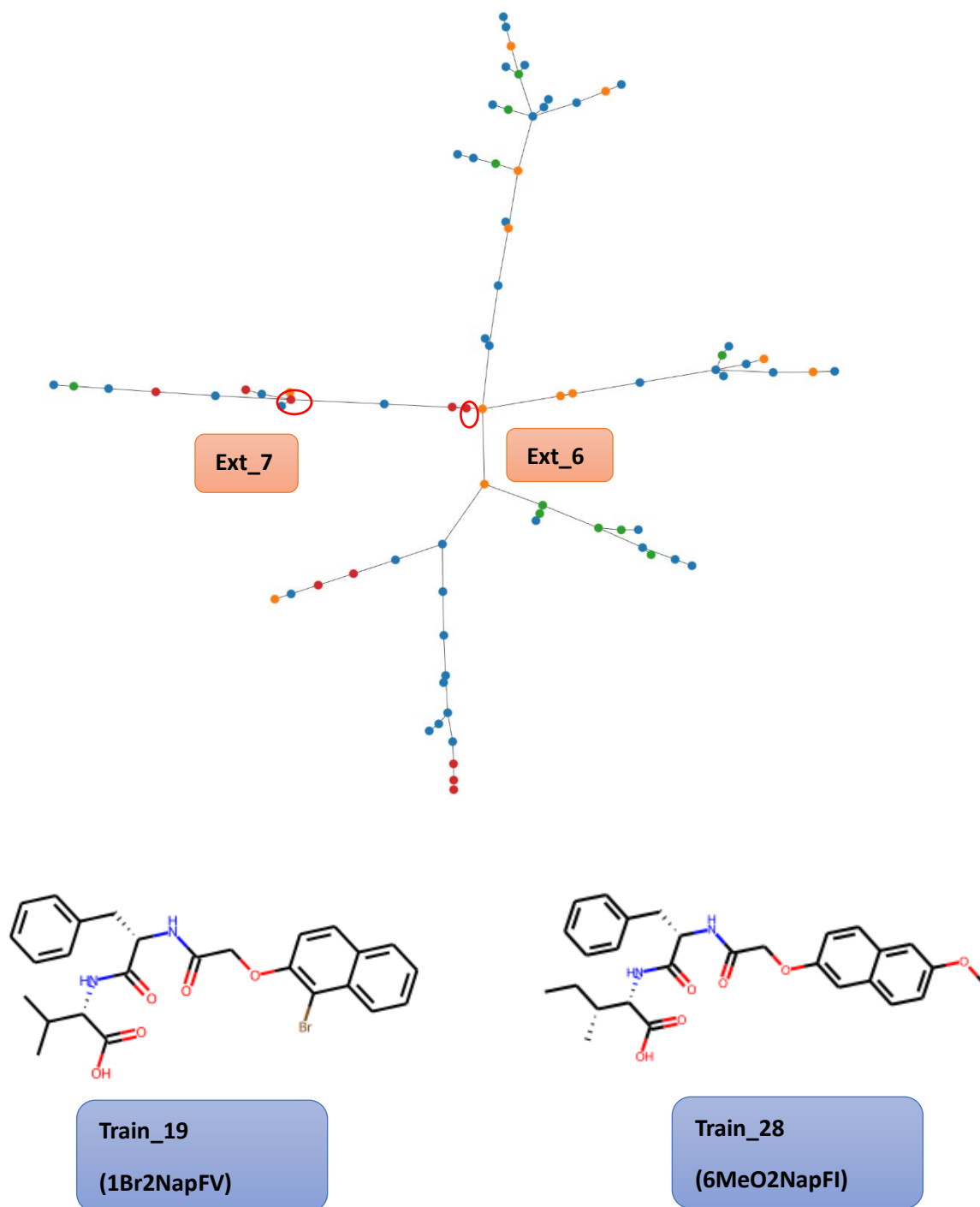


Figure 4.29: a) TMAP plot of the dataset with the external set and b) The two training set molecules closest to external_7 (2NapFV)

4.3.2.6 SHAP Interpretation

As in the G' SHAP, the 2.5 and 10 mg/mL beeswarm plots are uninformative and Figure 4.30 shows only the 5 mg/mL beeswarm plot. Of the top 10 fragments shown in Figure 4.31, in the G'' model, 4 are also present in the G' top 10. These are FCFP_403, FCFP_691, FCFP_927 and FCFP_428. Three of the top 10 fragments can be set by naphthalene, again underlying its importance. Fragments FCFP_148, FCFP_1395 and ECFP_1087 can all be set by naphthalene and FCFP_148 (1st) and ECFP_1087 (10th) both show the positive correlation between SHAP value and descriptor value. Two of the top 10 fragments, FCFP_25 (which can be set by phenylalanine) and FCFP_660 (which is the C-terminus) show no correlation with the descriptor value.

The final fragment in the top 10 is ECFP_507 which can be set by valine. Comparing ECFP_507 to FCFP_792 which can both be set to valine, ECFP_507 here has a positive correlation between SHAP and descriptor value whereas in FCFP_792 the relationship was negative. The SHAP results here corroborate the idea that valine increases the hydrophobicity and increases the tendency to self-assemble.

BART G'' beeswarm plot

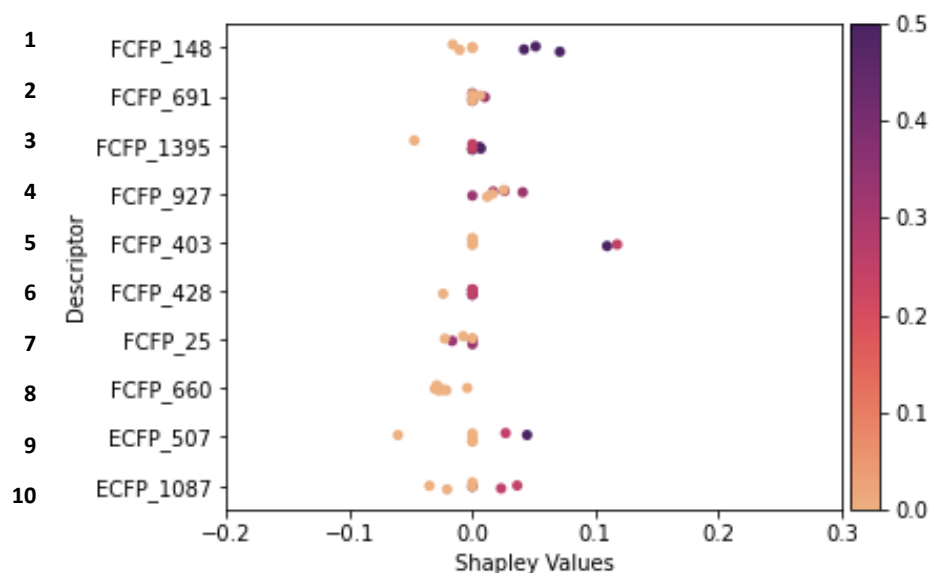
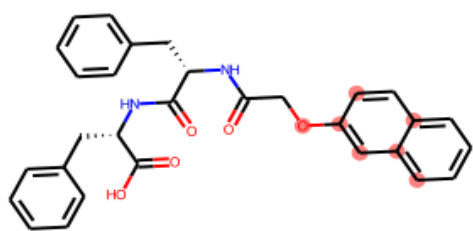
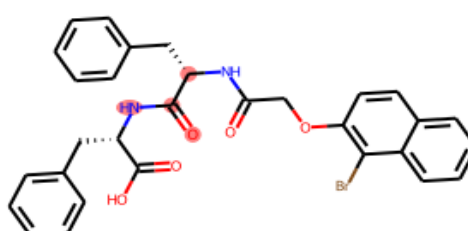


Figure 4.30: Beeswarm plot for the G'' model at 5 mg/mL

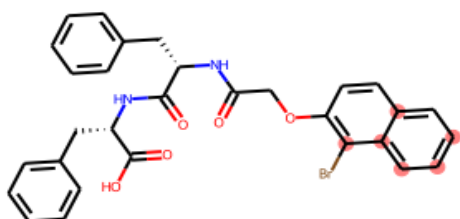
Chemical depictions of the most important fragments



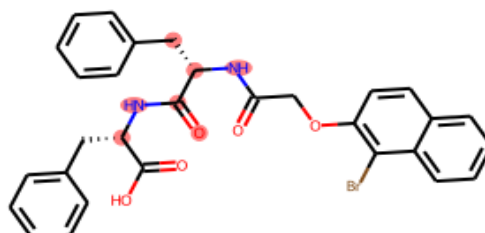
FCFP_148



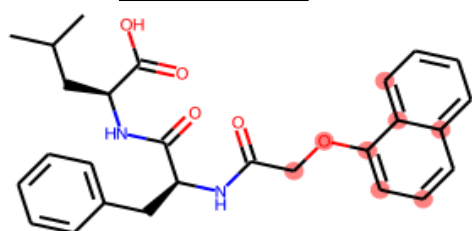
FCFP_691



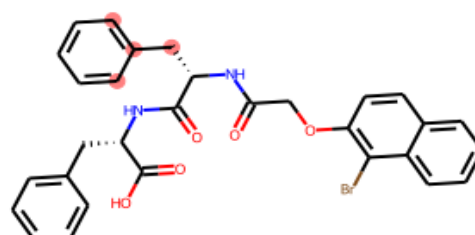
FCFP_1395



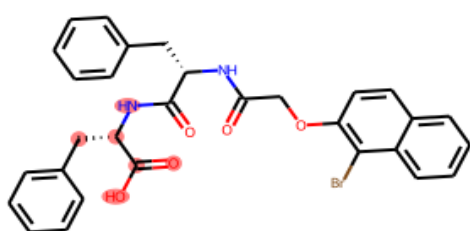
FCFP_927



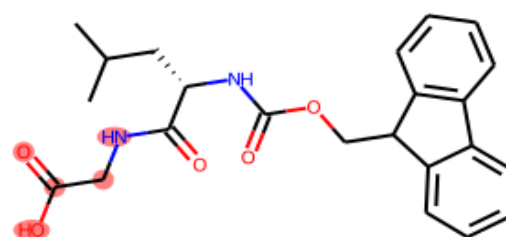
FCFP_403



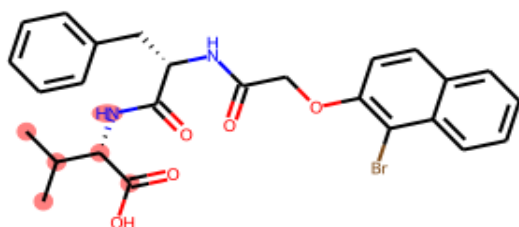
FCFP_428



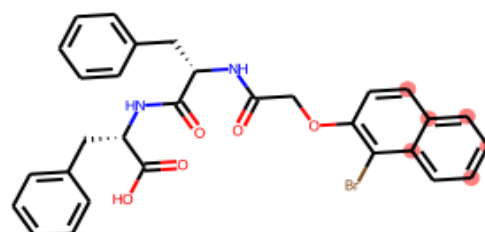
FCFP_25



FCFP_660



ECFP_507



ECFP_1087

Figure 4.31: Molecular representations of the 10 most important G'' fragments highlighted on training set molecules

4.4 Conclusions

Here we present two BART models to predict the rheological properties of low molecular weight hydrogels and compare their performance to the non-Bayesian models presented in chapter 3. Our Bayesian models show good visual correlation between experiment and prediction with R^2 and RMSE values that corroborate the predictive nature of our models. Although performance is slightly worse than the non-Bayesian models, the extra information given by the posterior of the predictions is valuable information when considering that the models were built on small data. We have validated the models using a y-randomisation approach and using Shapley values, ascertained which molecular fingerprints contribute most to the model's prediction. We hope these models can form the basis to accelerate the synthesis of new gels with a desired rheological profile, which could then be used to update the models and improve the certainty in the model's predictions.

These models, along with the Python classifier model from chapter 2 will be utilised in the De-Novo design of new gels in Chapter 5.

4.5 References

- 1 D. P. Bertsekas and J. N. Tsitsiklis, *Introduction to Probability*, Athena Scientific, Belmont, Massachusetts, 2nd edn.
- 2 J. H. Carey, L. A. Ball, A. W. Vermilyea, B. M. Voelker, D. Sea, A. Godrant, M. Furnas, T. D. Waite, E. A. Webb, T. D. Waite, J. W. Moffett, B. M. Peake, L. E. Richard, S. D. Nodder, R. Harris, Y. Shaked, A. J. Milligan, D. W. King, M. Morel, M. De Salas, T. Oda, G. Hallegraeff, D. Tchernov, A. Katsir, Y. Shaked, B. M. Voelker, M. Hansel, H. D. Easter and B. M. Voelker, 2013, **340**, 1177–1179.
- 3 M. Niewiadomska-Bugaj and R. Bartoszynski, in *Probability and Statistical Inference*, 2021, pp. 521–543.
- 4 D. P. Williams, S. E. Lazic, A. J. Foster, E. Semenova and P. Morgan, *Chem. Res. Toxicol.*, 2020, **33**, 239–248.
- 5 C. Hung and G. Gini, *Mol. Divers.*, 2021, **25**, 1283–1299.
- 6 N. Meftahi, M. Klymenko, A. J. Christofferson, U. Bach, D. A. Winkler and S. P. Russo, *npj Comput. Mater.*, 2020, **6**, 1–8.
- 7 B. A. Berg, *Markov Chain Monte Carlo Innov. Appl.*, 2005, **9**, 1–52.
- 8 W. R. Gilks, S. Richardson and D. J. Spiegelhalter, 1995, 45–47.
- 9 C. P. Robert, in *Wiley StatsRef: Statistics Reference Online*, 2015, pp. 1–15.
- 10 P. Lauret, E. Fock, R. N. Randrianarivony and J. F. Manicom-Ramsamy, *Energy Convers. Manag.*, 2008, **49**, 1156–1166.
- 11 B. Baesens, S. Viaene, D. Van Den Poel, J. Vanthienen and G. Dedene, *Eur. J. Oper. Res.*, 2002, **138**, 191–211.
- 12 Y. Xie, D. Lord and Y. Zhang, *Accid. Anal. Prev.*, 2007, **39**, 922–933.
- 13 H. A. Chipman, E. I. George and R. E. McCulloch, *Ann. Appl. Stat.*, 2012, **6**, 266–298.
- 14 A. M. Johansen, in *International Encyclopedia of Education*, Elsevier, 3rd edn., 2010, pp. 245–252.
- 15 P. Waldmann, *Genet. Sel. Evol.*, 2016, **48**, 1–12.

- 16 J. L. Zhang and W. K. Härdle, *Comput. Stat. Data Anal.*, 2010, **54**, 1197–1205.
- 17 B. R. Logan, R. Sparapani, R. E. McCulloch and P. W. Laud, *Stat. Methods Med. Res.*, 2019, **28**, 1079–1093.
- 18 J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl and V. Svetnik, *J. Chem. Inf. Model.*, 2015, **55**, 263–274.
- 19 R. Ancuceanu, B. Tamba, C. S. Stoicescu and M. Dinu, *Int. J. Mol. Sci.*, , DOI:10.3390/ijms21010019.
- 20 Y. V. Tan and J. Roy, *Stat. Med.*, 2019, **38**, 5048–5069.
- 21 J. S. Delaney*, *J. Chem. Inf. Comput. Sci.*, 2004, **44**, 1000–1005.
- 22 Data structures for statistical computing in python, McKinney, Proceedings of the 9th Python in Science Conference, Volume 445, 2010.
- 23 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, *J. Mach. Learn. Res.*, 2011, **12**, 2825–2830.
- 24 R. Kumar, C. Carroll, A. Hartikainen and O. Martin, *J. Open Source Softw.*, 2019, **4**, 1143.
- 25 D. L. J. Alexander, A. Tropsha and D. A. Winkler, *J. Chem. Inf. Model.*, 2015, **55**, 1316–1322.
- 26 A. Gelman, D. B. Rubin, A. Gelman and D. B. Rubin, 1992, **7**, 457–472.
- 27 D. Vats and C. Knudson, *Stat. Sci.*, 2021, **36**, 518–529.
- 28 S. S. Shapiro and M. B. Wilk, *Biometrika*, 1965, **52**, 591.
- 29 D. Probst and J. L. Reymond, *J. Cheminform.*, 2020, **12**, 1–13.
- 30 D. Stumpfe, H. Hu and J. Bajorath, *ACS Omega*, 2019, **4**, 14360–14368.
- 31 L. Breiman, *Mach. Learn.*, 2001, 5–32.
- 32 S. M. Lundberg and S. I. Lee, *Adv. Neural Inf. Process. Syst.*, 2017, **2017-Decem**, 4766–4775.
- 33 S. K. Nandi, K. Maji and D. Haldar, *ACS Omega*, 2018, **3**, 3744–3751.
- 34 Z. Yang, G. Liang, M. Ma, Y. Gao and B. Xu, *J. Mater. Chem.*, 2007, **17**, 850–854.
- 35 S. Fleming, D. Sisir, P. W. J. M. Frederix, T. Tuttle and R. V. Ulijn, *Chem. Commun.*, 2013, **49**, 10587–10589.

36 L. A. Estroff and A. D. Hamilton, *Chem. Rev.*, 2004, **104**, 1201–1217.

Chapter 5 – De-Novo Generative Models.

5.1 Introduction

Chapter 2 built upon the published models of Gupta *et al.*¹ to provide interpretable models with comparable performance for the prediction of whether a functionalised dipeptide forms a gel. Then, Chapters 3 and 4 introduced regression models for prediction of the mechanical properties, the storage modulus (G') and loss modulus (G'') including a Bayesian approach to capture the model's prediction uncertainties. Given that these models were generated to aid in the development of new gelators, attention now turns to the generation of new molecules through computational methods.

5.1.1 DeNovo design

Chemical space is vast, the size of potentially pharmacologically active molecules alone is thought to be on the order of 10^{60} . Since the factors impacting gelation are poorly understood, knowledge of the size of the potential chemical space for gelators isn't known.² Noting this, exploring the theoretical space exhaustively for potential new gelators is essentially impossible and as such, generative models have become an appealing approach to generating novel molecules. De-Novo design of molecules is the act of generating new molecules with a desired profile of some kind.³

Historically, a virtual screening approach has been deployed to screen for new molecules, even in the gels field, groups have generated libraries of molecules to screen their gelation ability⁴. Here, they synthesised a range of molecules analogous to those shown previously to gel (Figure 5.1). Through individual triggering through both the pH trigger via glucono- δ -lactone⁵ and a solvent trigger, they discover a range of molecules that form gels.

However, this approach is time consuming, expensive and has no guarantee of success. Therefore, De-Novo approaches to generate candidates and QSAR models to assess virtually molecules in order to obtain compounds with a desired profile, although with no guarantee of success, has the potential to be a streamlined way to expand the chemical space of gelators with desired properties.

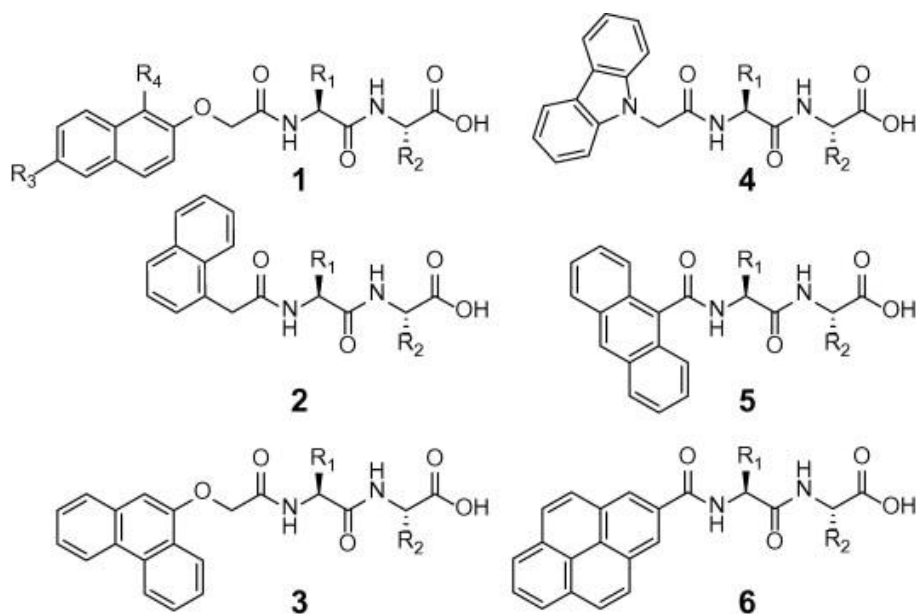


Figure 5.1: The dipeptides known to gel previously used to derive new analogues that also form gels. Figure adapted from [4]

5.1.1.1 Genetic Algorithms

Genetic Algorithms (GA) are named as such due to their similarity with evolution.⁶ In a genetic algorithm, members of an initial population have their fitness calculated with respect to a desired characteristic. Only those in the population that have desirable characteristics are taken to the next iteration to “breed”.⁷ Variations of GA which do not involve direct breeding of the population are called evolutionary algorithms. This process continues until some criteria is met or a desired number of generations has been reached.⁸ The general process of a GA is summarised in Figure 5.2⁹

This approach has found use in De-Novo molecule design in chemistry by considering molecules as a graph where molecules are nodes and bonds are the edges¹⁰. Globus *et al*¹¹ proposed a graph based genetic algorithm that twice compares two molecules at random, the best of each comparison is termed father and mother, respectively. The mother and father molecule are randomly fragmented and a fragment from each forms a new molecule, the son. This is repeated to produce a daughter. The comparison of two molecules is repeated and the worst in each comparison is replaced with the son and daughter. The “goodness” of a molecule at each comparison was governed by a fitness function that considers the atom pairs shortest path similarity between the molecule and a target molecule.¹²

Another graph based GA was presented by Jensen¹³ to produce molecules with desirable hydrophobicity in terms of LogP. Here, mutations are made in a probabilistic fashion, with the type of change to the molecular graph having roughly equal probability, meaning that changes to bond order

or atoms are equally likely. The bond/atom used to make the replacement is decided by the relative prevalence of the bond/atom in the ZINC database.¹⁴ Part of the optimisation function of this GA considers the synthetic accessibility of the new molecules and so optimisation should lead to chemically reasonable molecules.

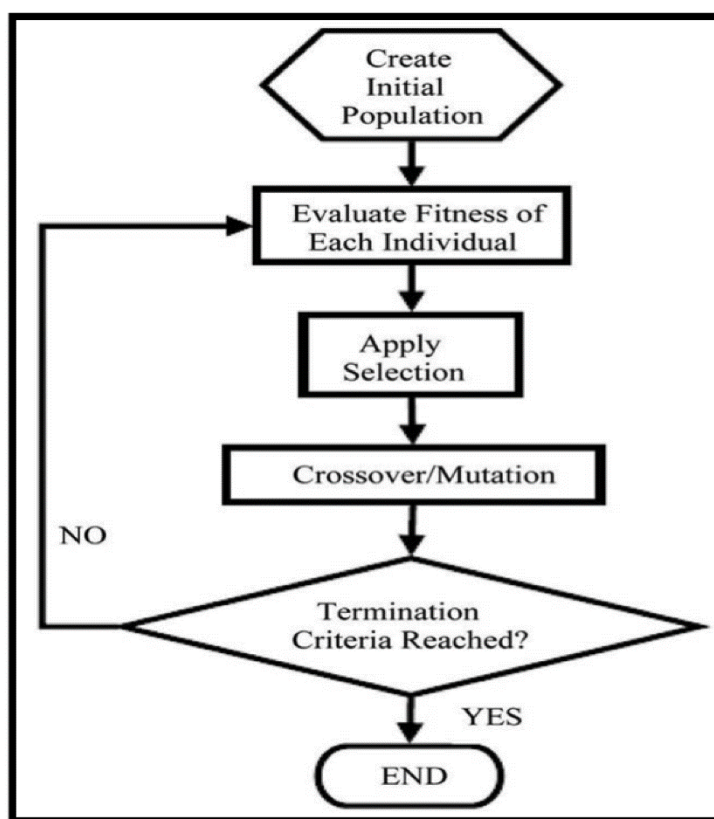


Figure 5.2: General workflow of a GA. Figure adapted from [9]

Given the issue of chemically reasonable mutations, work was carried out to define a methodology that produced mutated molecules with reasonable chemistry. The Chemically Reasonable Mutations (CReM) package in Python was presented by Polishchuck *et al*¹⁵. CReM works by first creating a database of fragments by fragmenting molecules from a user defined set of molecules. The default sets are versions of the ChEMBL database¹⁶. The fragmentation is done considering the local molecular context, within a given bond radius, of the molecules so that fragments can only be replaced if they are in the same chemical context. This is summarised in Figure 5.3.

When carrying out the mutation on a molecule, the molecule is fragmented based on the radius context and the database is searched for all fragments with the same chemical context. All possible replacements are made and the mutated molecules returned.

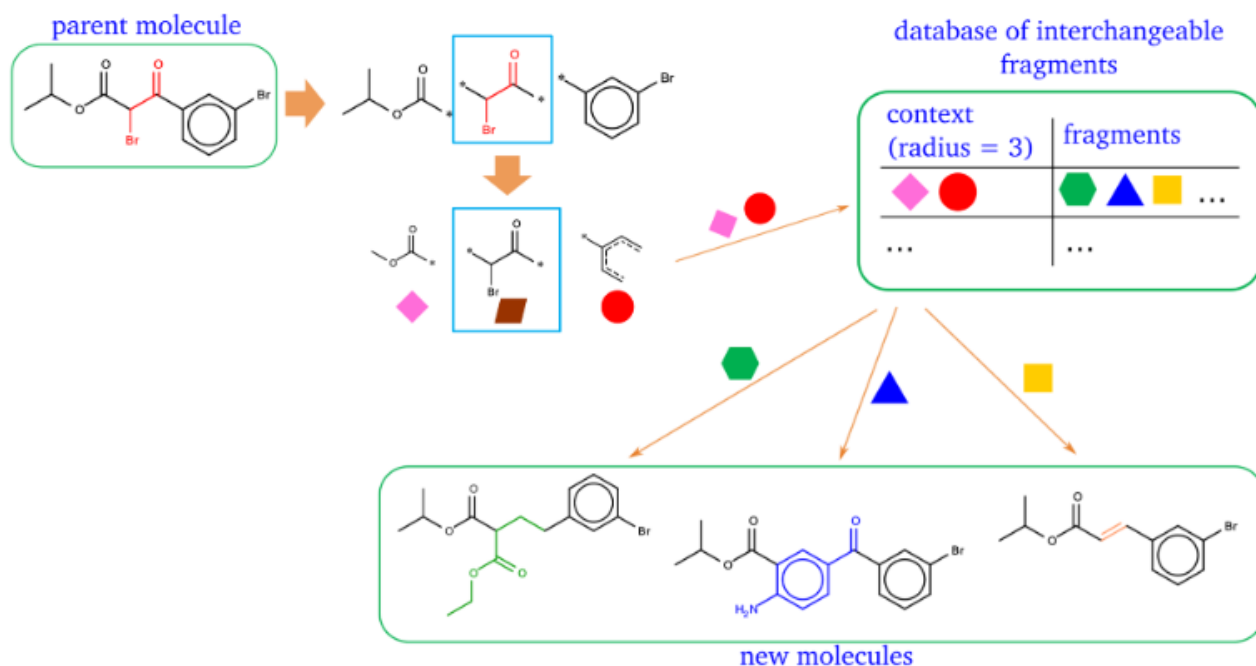


Figure 5.3: Example of the CReM mutation of a molecule. The brown fragment is replaced and the context radius of the pink and red fragments are checked in the database to find compatible fragments. Once found, the green, blue and yellow fragments replace the brown one in the parent molecule to form new child molecules. Figure adapted from [15]

5.1.1.2 Recurrent Neural Networks

Another popular approach for de novo generation is based upon neural networks¹⁷, more specifically a Recurrent Neural Network (RNN). A RNN (Figure 5.4) is designed for time series or sequential data, where data is passed to the model sequentially and the current data termed \mathbf{x} .¹⁸ The information outputted from the network from the previous time steps, the hidden state (\mathbf{h}_{-1}), is inputted along with the next input in the sequence (\mathbf{x}) for the current time step (\mathbf{t}) and it outputs an updated hidden state (\mathbf{h}). This updated hidden state also forms part of the input for the \mathbf{t}_{+1} and allows the RNNs to use information from data it has already seen to influence the current prediction.¹⁹ This allows the network to remember and RNNs are said to possess “memory”. This is visualised in Figure 5.4.²⁰ Due to this memory, these methods have been heavily utilised in Natural Language Processing²¹, language translation²² and the generation of music²³.

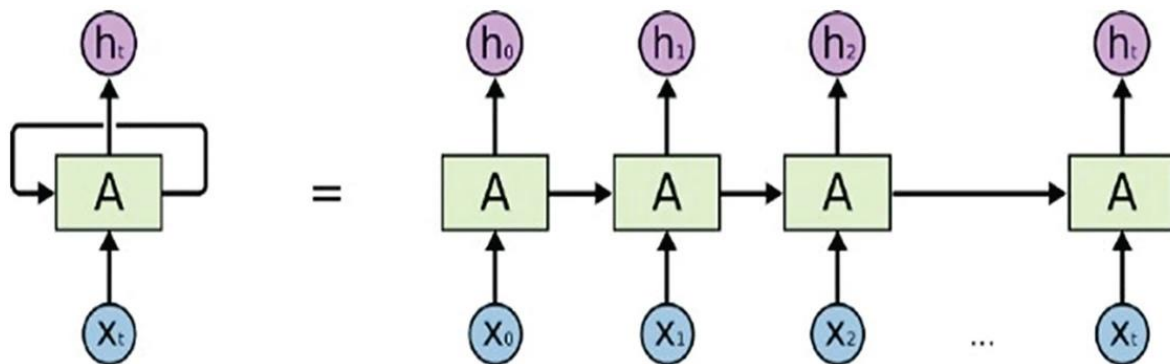


Figure 5.4: An example of a RNN where the information from the initial input X_0 at time step t_0 is passed through to the hidden layer for X_1 and influences the output. Figure adapted from [19]

However, one drawback of a standard RNN is that long-term memory, that may be necessary for context say in a text generation use, will be lost due to the vanishing gradient phenomenon.²⁴ When the error in the prediction is back-propagated through the network, the further back in the network the error propagates the smaller the gradient becomes and thus, the less that section of the network influences the current hidden state.²⁴ To overcome this issue, two new types of RNN layers are utilised, the Long Short-Term Memory (LSTM) layer²⁵ and the Gated Recurrent Unit (GRU) layer²⁶. These layers are visualised in Figure 5.5.²⁷

LSTM layers were developed to allow for important context early in the sequence to be propagated through the network. They do this through an extra state called the cell state, which contains more long-term memory than the hidden state.²⁵ At each time-step, t , the neuron receives the new character for this time step, the hidden state h_{t-1} and the cell state from the previous step C_{t-1}

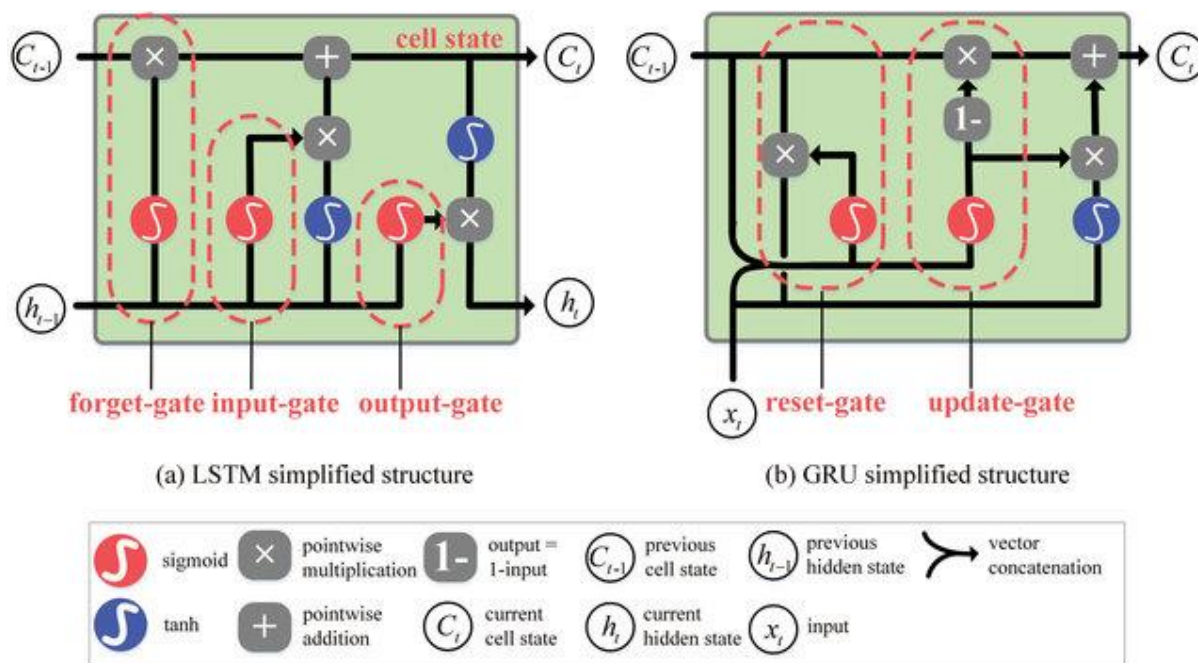


Figure 5.5: An example of an LSTM and GRU layer highlighting the different gates which control which information is passed to the next layer of the RNN. Figure adapted from [26]

LSTM models utilise three gates, which themselves apply functions to weight vectors and thus can be considered simple neural networks.²⁸ The first gate, the forget gate, decides which information from the cell state can be discarded based on the current input and the previous hidden state (h_{t-1}). Next, the input gate updates the cell state using the information from the previous hidden state and the current input. Next the output gate decides on the new hidden state should be for $t+1$.²⁹

On the other hand, the GRU layer doesn't utilise a cell state and transfers long term information through the hidden state. A GRU layer utilises two gates, the reset gate and the update gate. The Reset gate determines which past information to forget, and the update gate decides which current information to forget, and which new information gets added to the hidden state.³⁰

In a chemistry context, RNNs can be utilised by taking advantage of the SMILES representation of molecules³¹. Since SMILES are a string-based representation, we can use an RNN as a character generating model to iteratively build a molecule atom by atom. This is carried out by splitting the SMILES string into characters and then treating each character as a time series – utilising the previous atoms present in the string to generate new chemically reasonable characters to append to our molecule. The Long-Term memory of the LSTM and GRU layers are important in generating SMILES as we need a long term dependency to teach the model to close an aromatic ring it had previously opened. An example molecule generation using an RNN is shown in Figure 5.6.³²

There are reports of RNNs alone being used to generate molecules. For example Bjerrum *et al.*³³ downloaded the clean fragment and drug like subset from ZINC12¹⁴ as SMILES and used these as the molecules for an RNN to learn the molecular syntax of SMILES. They sampled a SMILES character at each time step in the RNN and added a sampling temperature parameter to modify how conservative the model became when sampling new molecules. They found that the generated models possessed similar profiles of physicochemical properties (LogD, Molecular Weight *etc.*) as the ZINC database. The similarity of the generated molecules to the training molecules can be tweaked with the temperature parameter but this leads to an increase in invalid SMILES strings.

However, RNNs are most typically deployed with a reinforcement learning aspect to guide molecular generation to molecules with a desirable property or activity against a particular target.

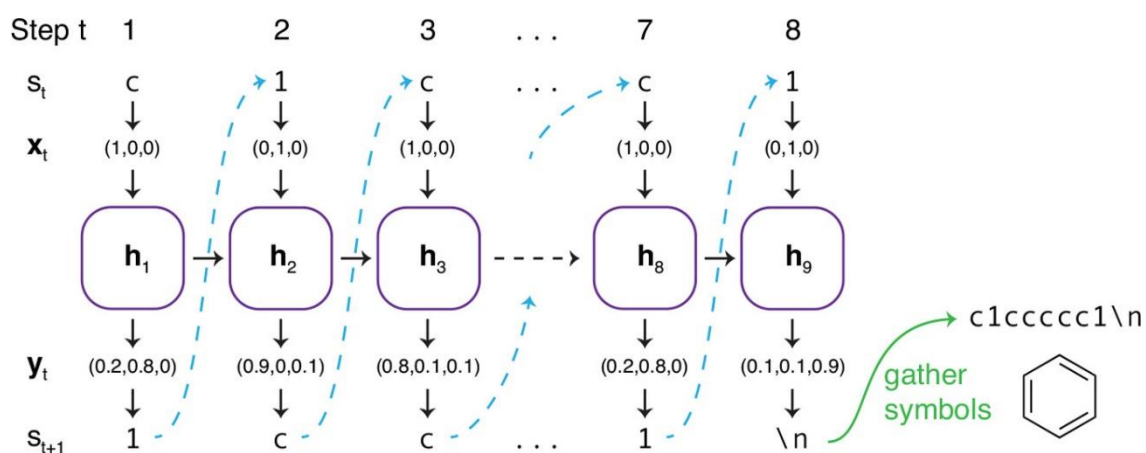


Figure 5.6: An example of generating benzene using an RNN. A carbon atom is chosen as the starting atom and each new sample (S_{t+1}) influences the next SMILES character generated. Note that the RNN has remembered to close the benzene ring by sampling a 1 at the end of the sequence. Figure adapted from [31]

5.1.1.3 Reinforcement learning

Reinforcement learning (RL) is based around the idea of rewarding a machine learning model based on the actions taken by it within an environment.³⁴ It is a form of unsupervised learning which allows machine learning models to explore environments via the use of an agent. The agent decides on a move to make in the environment and the agent is rewarded if the move is a “positive” move. The agent will attempt to make moves that maximise its reward.³⁵

Reinforcement learning rose to fame in 2015 when it became the first computer model to defeat a human player in the ancient Chinese game of Go. The program, AlphaGo, defeated the current European champion and later in 2016 beat 18 time world GO champion Lee Sedol³⁶. Researchers at DeepMind trained a deep neural network with reinforcement learning how to play Go using a neural

network to select the next move and another to predict the winner of the game. Over time AlphaGo became skilled enough to be the best Go player in the world.

Applications of reinforcement learning within chemistry to produce molecules with desired properties including the ReLeaSE approach³⁷ proposed by Popova *et al.* Here they utilise an RNN in the form of a stack-RNN with a LSTM based RNN which acts as a QSAR model for predicting molecular properties.

They trained their RNN with 1.5 million molecules from ChEMBL to learn the syntax of SMILES. Interestingly, they generate 1 million molecules, 0.1% of which were in the training set. Again, they generate molecules towards optimising logP below 5 but also generate molecules that minimise or maximise JAK2 activity, using the QSAR model.

Other Reinforcement learning approaches have been successfully used to generate molecules with good drug likeliness³⁸, show good inhibitory activity against DDR1 kinases³⁹ and generate analogues of commercially available drugs⁴⁰. The models built to generate drug analogues in [40], is the REINVENT approach and contains a predefined GRU based RNN architecture with customisable scoring functions⁴⁰, both of which we will utilise in this chapter for our RNN-RL.

The REINVENT approach utilises two RNN networks, the Prior and Agent network. Initially, the RNN is trained on a set of SMILES so that the model can learn the syntax of SMILES. Then, a scoring function is defined that scores a molecule between 0 and 1 based on a desired fitness function. The prior is then used as the starting point to train the agent network which is responsible for generating new molecules. The molecules are sampled from the agent subject to the augmented likelihood probability of the molecule based on both the molecules in the prior model and the score from the scoring function. The general process of the REINVENT process is summarised in Figure 5.7.

Other generative approach used within chemistry include the Generative Adversarial Network (GAN) set of models.⁴¹ Training a GAN model involves simultaneous training of two models, the generative model, G, and the discriminative model, D, form a zero-sum game. The game involves G creating fake input data and D receiving both real data, and the fake data created by G and is tasked with discriminating between the real and fake input data with the generator hoping it can fool the discriminator.⁴² Once trained, the discriminator is discarded and the generator can be used as a generative model.

GAN models have been used within chemistry as a molecular generation tool. Mol-CycleGAN was introduced by Maziarka *et al.* as a tool to generate structurally similar molecules to a reference molecule with optimised molecular properties (such as logP) for drug-like molecules.⁴³

Another example of a generative approach used within chemistry are diffusion-based models. Diffusion models are probabilistic models whose training procedure involves systematically adding noise to an image to “destroy it”. The model is then tasked with removing the noise and this process allows the model to learn the underlying distribution of the training data.⁴⁴ Once the underlying distribution is known, noise can be used as the input to the model to generate an image that is representative of the original training set.

Diffusion models have been used within chemistry to generate drug-like ligands in protein pockets. Here they create a diffusion-based model trained on the CrossDocked dataset to generate molecular point clouds within a protein binding site. They then assess the binding affinity of the generated molecules with the binding site and show that the generated molecules possess similar binding affinities to related generative approaches.⁴⁵

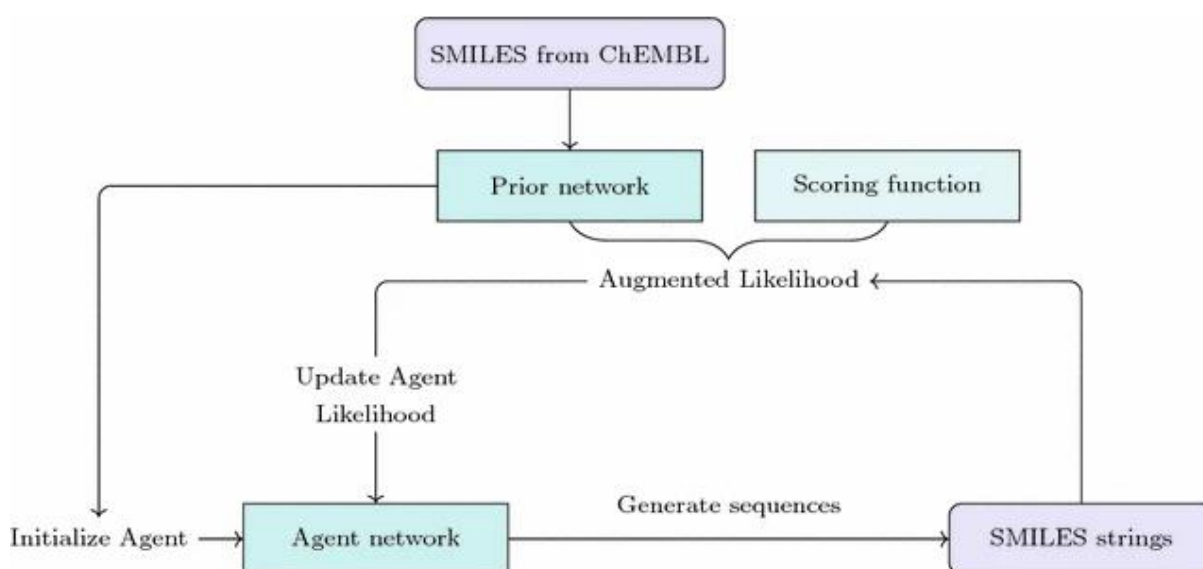


Figure 5.7: Summary of the REINVENT process. Figure adapted from [40]

In this chapter, generative models will be built based on both a genetic algorithm and an RNN. Initially, CRem will be used as the mutation operation on the molecules and score the molecules by how close they are to a desired G' value. Comparisons between this method and the RNN learnt via the REINVENT model with a reinforcement learning aspect which contains a similar scoring function to the genetic algorithm which assesses the closeness to a G' value between 0 and 1.

5.2 Experimental

5.2.1 CReM

5.2.1.1 Software used

For our Genetic Algorithm (GA) based approach to De-Novo design, all code is written within Python⁴⁶ (*version 3.7.9*). Mutations to our molecules is carried out using the CReM package (*version 0.2.9*)¹⁵ in Python. Within CReM, we utilise the ChEMBL library v22¹⁶ with maximum synthetic accessibility score of 2 as the fragment database used to mutate our virtual library. This dataset was chosen as it contains molecules that have been synthesised and should help to keep mutated molecules in a reasonable chemical space that is synthesisable.

5.2.1.2 Workflow

As our initial pool of “parent” compounds for the GA we utilise a virtual library we have enumerated. Enumeration was achieved using an in-house Python script which utilises the RDKit⁴⁷ reaction functionality to react each dipeptide/tripeptide in *Appendix 5.7.1* with an R group in the correct position. Reactions are carried out so that each possible molecule from the list of R groups in *Appendix 5.7.1* is formed. The enumerated virtual library consists of 529,200 unique compounds, for structures and R groups see *Appendix 5.7.1*. The GA process is summarised in Figure 5.8 and works as follows.

The desired G' value is chosen and the 1000 molecules in the virtual library with the closest predicted G' to this value is chosen from the initial pool. Once the initial pool is reduced, we carry out all possible CReM mutations on a user-defined number of (from 2 to 100) randomly selected molecules in the initial pool and carry out an applicability domain check for the classification models and the rheology models. If the molecule is considered in-domain, which here is defined only by the k-NN similarity to the nearest training set point (*see Chapter 1: 1.9 Applicability Domain*), we predict the gel state of the mutated molecules using our Python Classifier from Chapter 2 (*2.3.4 – Python Classification Models*) and the G' values using our PyMC3 BART rheology model from Chapter 4 (*4.3.1 - G' BART model*).

CReM Process

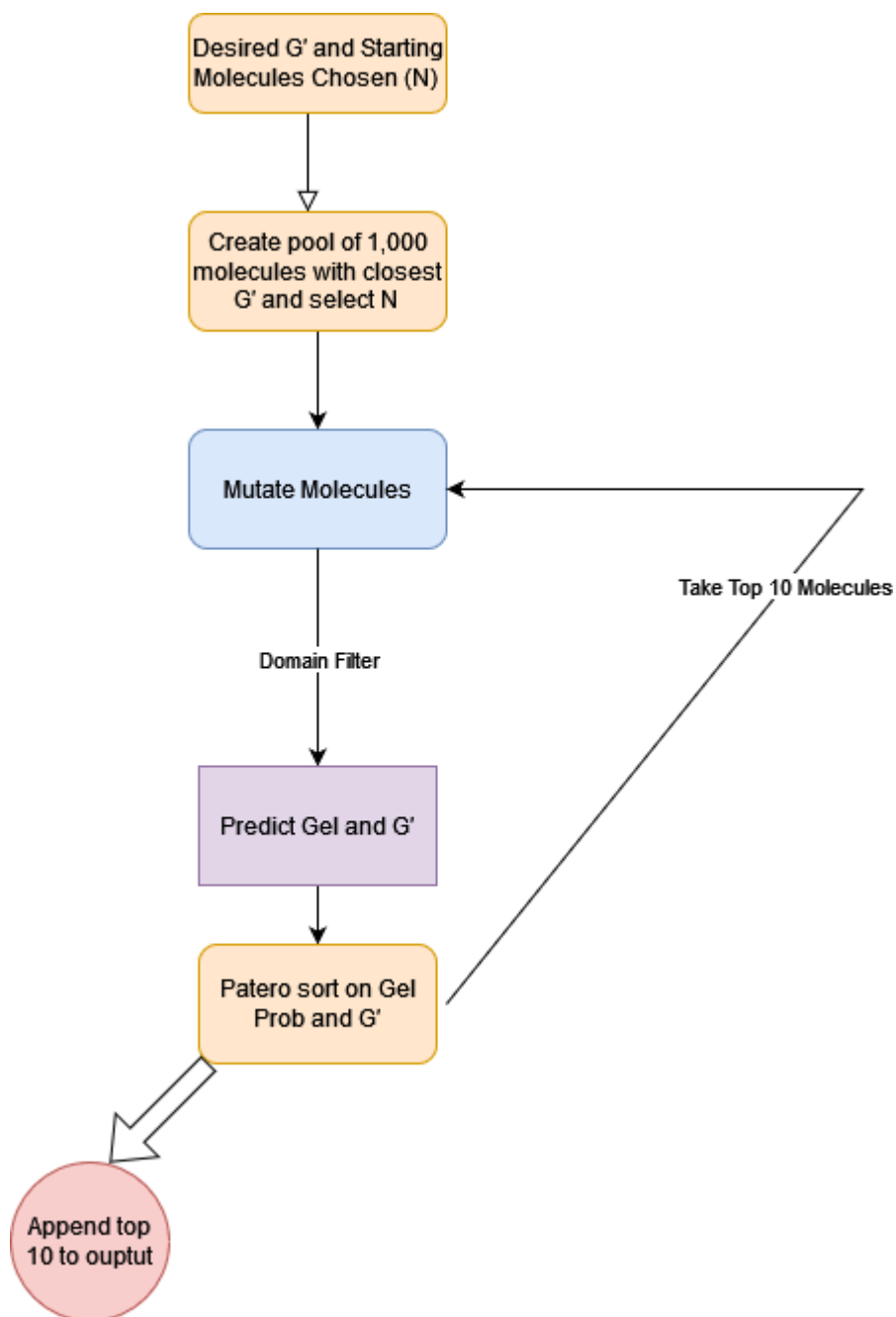


Figure 5.8: Overview of the CReM-GA Process

Next, we calculate the distance between the model's predicted value and the desired value and assess how close it is via a customised tanh function. The customised derivative of the tanh function (Figure 5.9) assigns scores closer to 1 as the distance between the calculated G' and the desired G' approaches 0.

We take the square root of the derivative to assign a more stringent scoring system, this is because our G' range encompassed 3 units and residuals from our model are generally less than 1 meaning that the majority of our calculated values will be less than 1. We therefore want to weight more highly scores very close to 0.

Tanh scoring function

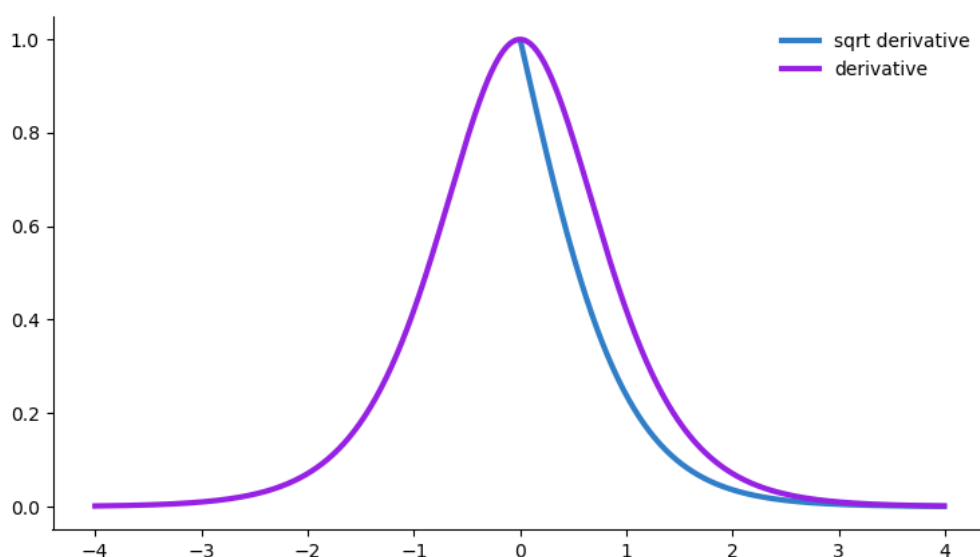


Figure 5.9: The derivative of the tanh function (purple) and the square root (blue) which forms our more stringent scoring system.

Once scored, each point is sorted using a Pareto sort⁴⁸ to maximise the probability of gel formation and maximise the assigned score. We then take the top 10 from our sorted mutated molecules as our initial pool for the next iteration. If the previous output failed to produce 10 molecules, molecules from the virtual library are appended before the next iteration. The algorithm can be run for a set output length or for a set number of iterations. We iterate until

we have 100 molecules in the output or 100 iterations have passed. The user is outputted a sorted list using the same Pareto sort carried out at each iteration.

5.2.1.3 Benchmarking

We aim to establish the performance of our GA approach at generating diverse molecules close to our desired values, across a range of G' values. Although there is possibility to modify the script to generate molecules with desired G'' values, the RNN below currently only has the capability to generate molecules based on G' and so we will focus the benchmarks on the GA's ability to successfully generate molecules with desirable G' . To do so, we define two hyperparameters for our search. 1) The number of initial molecules for the initial mutation and 2) The desired G' value.

For the number of initial molecules, we investigate 2, 5, 10, 20, 50 and 100 starting molecules and G' values of 5000, 20000, 50000, 75000, 100000 and 150000. The G' values encompass the range of values of our rheology models in chapters 3 and 4 whilst extending this range with the addition of 5000 and 150000.

We assess the performance of the model by, the time taken to run, the distance from the desired value and the similarity of the sampled molecules properties compared to the virtual library. All benchmarking runs were carried out on the Barkla, part of the High Performance Computing facilities at the University of Liverpool, UK.⁴⁹ Each hyperparameter pair (G' and starting molecules) was run on a single core of an Intel Xeon Gold Skylake processor with 9.6GB of memory.

5.2.2 Reinvent – Reinforcement Learning

5.2.2.1 Software used

For the RNN models, we utilise the published REINVENT workflow. REINVENT is a GRU based RNN model built within PyTorch (*version 1.10.2*). We train the RNN architecture on the enumerated virtual library allowing it to learn the syntax underpinning our peptide based gelators. Model training was carried out on a single GPU on Barkla on a Nvidia NVLink P100.⁴⁹ From the trained RNN, we define our custom scoring function as follows for agent sampling.

Generating new molecules is a simple sampling procedure from the probability distribution of all SMILES characters within the training set at each time step in the RNN. Sampling begins with a START token and the molecule is built sequentially character-wise until the model samples the END token.⁴⁰ Once a molecule's END token has been sampled, a scoring function is applied to the molecule to assist in the reinforcement learning for the generation of molecules with desirable G' values.

Like within the CReM model, for the sampled SMILES from the RNN, we first predict the gelation probability from the Python Classifier before predicting the G' value for the sampled molecule using our PyMC3 BART model. If the molecule is not predicted to form a gel, our scoring function assigns a score of 0. If it is predicted to form a gel, the tanh function is again used and the score assigned by the function is taken as the score for our reinforcement learning scoring function. The process is the same as that shown in Figure 5.7 and the model samples for a set number of steps (*i.e.* iterations). Scores for the sampled molecules for a step update the augmented likelihood for the next sampling step.

5.2.2.2 Benchmarking

To benchmark our RNN trained model, we again attempt to gauge its performance over a range of G' values. The two hyperparameters of the RNN-RL model are the G' value and the number of steps. The number of steps is the number of iterations to run the sampling for. We run the same G' values as the CReM benchmark, 5000, 20000, 50000, 75000, 100000 and 150000. We run the algorithm for 100, 500, 1000, 1500 and 2000 steps. We assess the performance of the model by, the time taken to run, the validity of the generated SMILES, the assigned scores from the tanh function and the similarity of the sampled molecules compared to the virtual library. Each hyperparameter pair (G' and number of steps) was run on a single core of an Intel Xeon Gold Skylake processor with 9.6GB of memory.

5.3 Results and Discussion

5.3.1 Exploration of virtual Library

As the virtual library forms the basis of both generative approaches, we wish to get a general profile of the virtual library as a comparison baseline to track how the sampled molecules differ from the virtual library. Given their importance in the Python Classifier models, we will investigate the distribution of Num_Rings, Molecular_Solubility, ALogP and also include the Molecular_PolarSASA due to its importance in our R models (*Chapter 2 section 2.3.2*). The distributions in Figure 5.10 will act as the comparator when we investigate the output from our de-novo methods.

Given the size of the virtual library, the properties will be summarised and comparisons with our mutated datasets can be made with the average and standard deviation of the properties in the virtual library. For the virtual library, the AlogP has an average value of 1.11 (\pm 2.30) and spans from -7.68 to 9.73. For the number of rings, the most common number of rings is 2 and spans from 2 to 10 rings. As for the Molecular_PolarSASA, the virtual library has an average value of 330.31 (\pm 68.28) with a minimum value of 159.13 and a maximum of 724.70. Finally the mean solubility is -7.50 (\pm 1.75) and spans the range -16.49 to -2.87.

Molecular properties of the virtual library

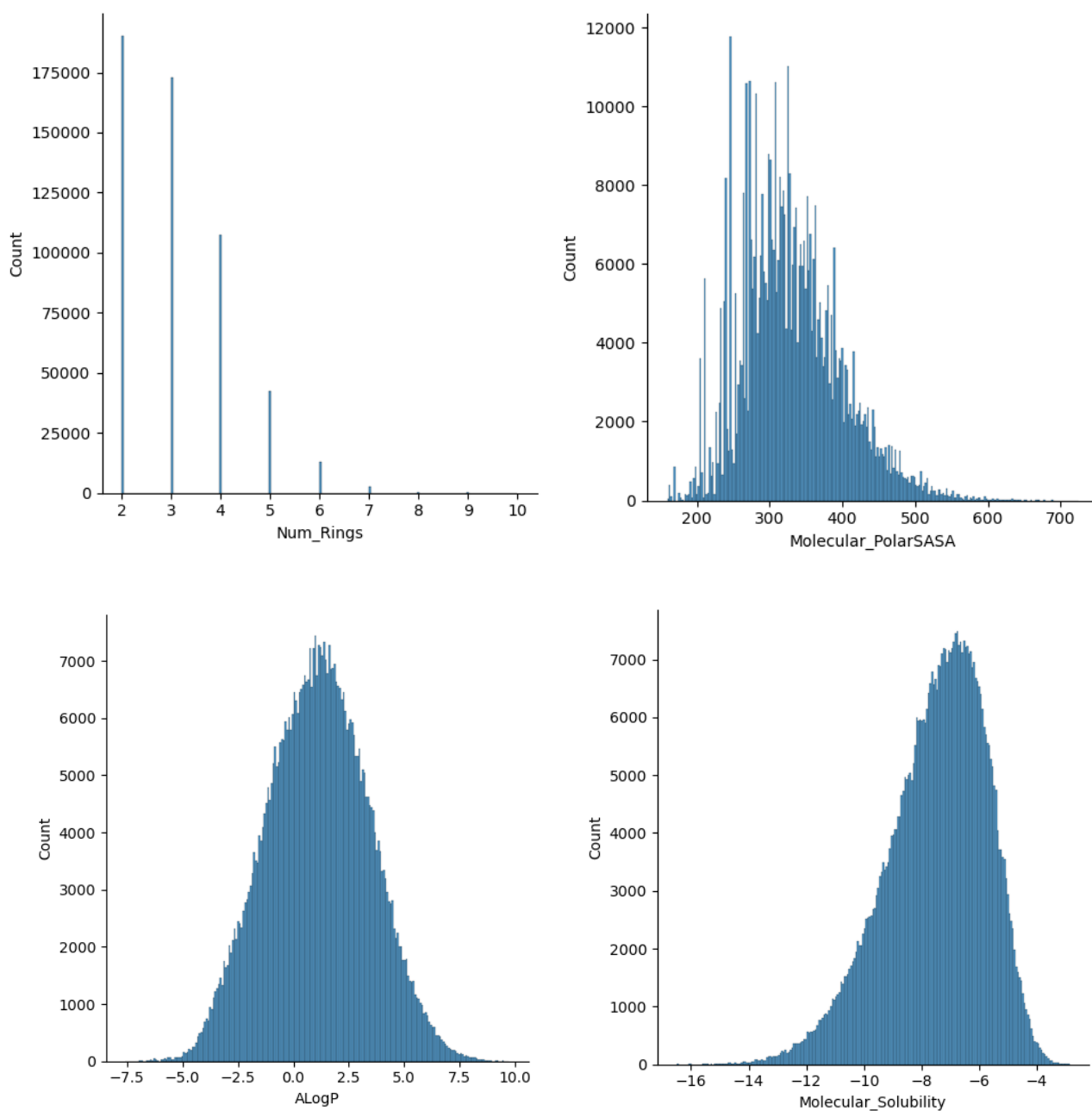


Figure 5.10: Values for Num_Rings, Molecula_PolarSASA, AlogP and Molecular_Solubility for the virtual library.

5.3.2 CReM

5.3.2.1 Running for set output length

5.3.2.1.1 Timings

To gauge the general time taken for the GA to produce 100 molecules for each starting molecule value and G' value, we ran the algorithm 10 times with each pair of benchmarking values. For each G' value, if the algorithm fails to produce 100 molecules after 100 iterations, it was assigned a time of 6000s. Each output contains 100 molecules that are the pareto optimal trade-off for that iteration between the distance to the desired value and the probability of gel formation (minimising the distance/maximising the gel probability). Pareto sorting is a form of multi-objective optimisation⁴⁸ whereby improving one dimension cannot occur without worsening a second dimension⁵⁰ *i.e.* we cannot further maximise the score without the gel probability decreasing. There exists a set of Pareto-optimal solutions that best maximise both the gel probability and the score from our model.

Figure 5.11 shows the distribution of time taken to run the CReM generative algorithm ten times across our G' and starting molecules hyperparameters. Across the range of G' values tested there are numerous distributions with very large standard deviations, which are caused by runs that failed to complete. Overall, out of the 360 runs in total 52 (14.4%) failed to produce 100 molecules within 100 iterations. The failures are more common for a G' of 5,000 and for 2 starting molecules. 21 of the 52 failures are when attempting to produce 100 molecules with a G' of 5,000. 8 of the 10 runs of 2 starting molecules for a G' of 5,000 failed to produce the 100 molecules.

Given that the algorithm attempts to produce 100 molecules, it is somewhat unsurprising therefore that the number of failures decreases as the number of starting molecules tends towards 100. 5 starting molecules results in 7 failures, 10 starting molecules has 11 failures, 20 has 4, 50 has 3 and 100 also has 3. It is not always clear why some runs fail but could be attributed at low starting molecules to a smaller set of mutated molecules resulting in a decreased chance of in-domain points for prediction. This would result in empty iterations and increase the chance of a failure.

CREM Timings

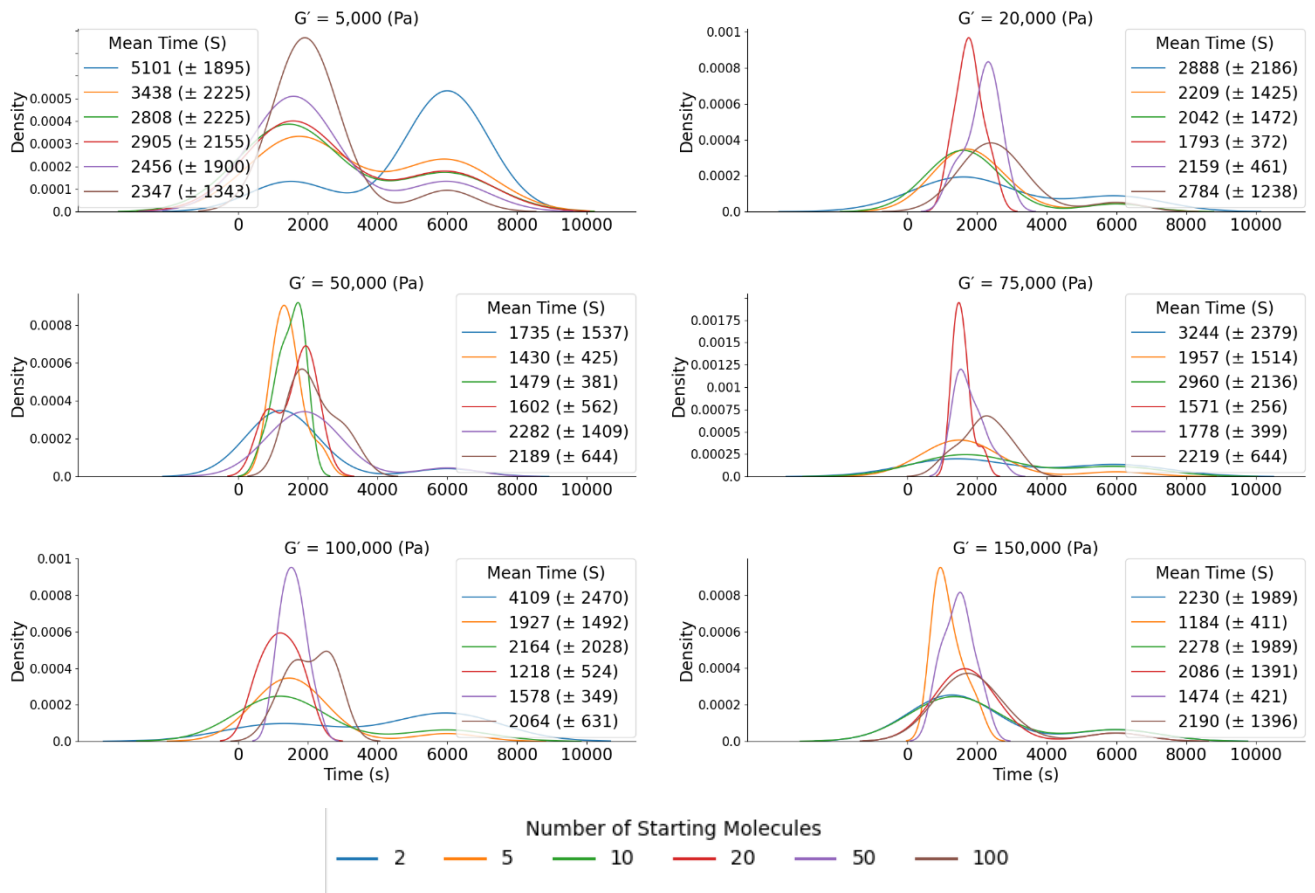


Figure 5.11: Distribution of the timings for 10 repeats for each hyperparameter (starting molecules and G'). Points that do not produce 100 molecules after 10 iterations are assigned a time of 6000s.

Since the failures skew the timings in Figure 5.11, replotting in Figure 5.12 with the failures removed gives a better indication of the time to complete. On the whole, when it does execute the script executes on a reasonable time scale (between 20 and 40 minutes). The smallest average time to execute is 1184 (± 411 s) for 5 starting molecules with a desired G' of 150,000 and the largest average time to execute is 2426s (± 536 s) starting with 100 molecules with a desired G' of 20,000.

The time taken for the script to execute is predominantly determined by the mutation operation of the molecules. However, this is only impactful during the very initial mutation as each subsequent mutation is only carried out on a maximum of 10 molecules, whereas the initial mutation can be carried out on up to 100. This accounts for the relative similarity in the timings in Figure 5.12 and the general increased time to complete for starting molecules of 100.

Therefore, we have a script that will execute on average within 40 minutes with an upper bound of approximately 50 minutes, resulting in a process that is not too computationally expensive. However, next the performance of the model for each hyperparameter pair is assessed to gauge how close the molecules generated are from the desired value – it is worth the extra computational time if it results in more desirable molecules (in terms of desired G').

CReM Timings – Failures removed.

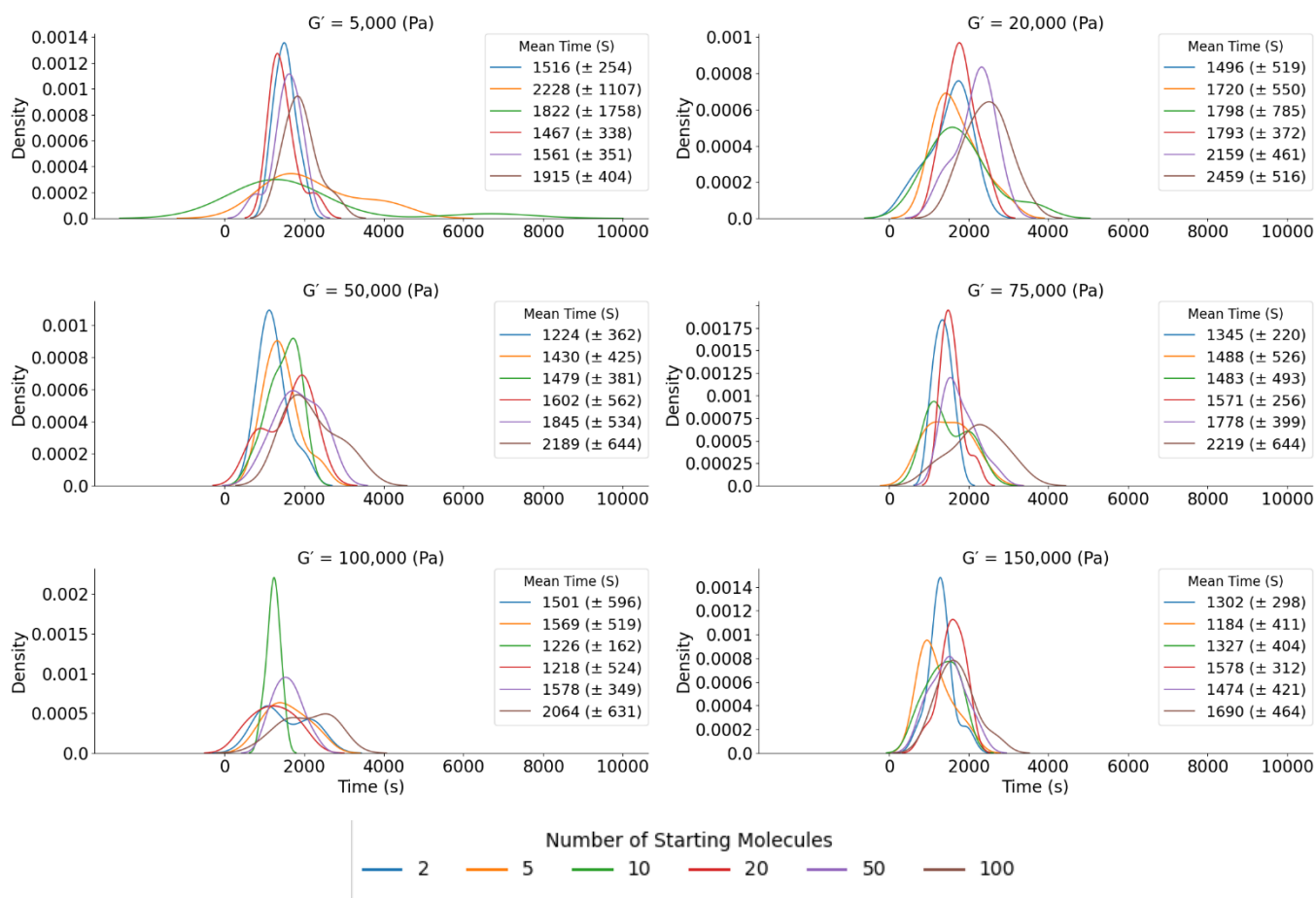


Figure 5.12: Timings for the fixed output CReM algorithm considering only successful completions.

5.3.2.1.2 Distribution of closeness to desired value

We assess the closeness to the desired value through the custom derivative of the tanh function (section 5.2.1.2) where values close to 1 indicate closeness to our desired value. Figure 5.13 shows the combined output from the 10 repeats of the script across all hyperparameter pairs in terms of their scores. The runs for a G' of 50,000 are the most consistent in terms of average score across all starting molecules considered. The range of scores these runs encompass is small, ranging from 0.89 (± 0.07) for 5 starting molecules to 0.93 (± 0.06) for 20 starting molecules. This is a very narrow range of scores, raising questions about the similarity of the molecules in the output and how chemically diverse the output for these runs are – this will be investigated later in the chapter.

For the remaining G' values, there is more variation to the quality (in terms of closeness of G' to the desired value) of the molecules generated by the CReM-GA. A desired G' of 100,000 does produce a good range of average scores across starting molecules (0.93 to 0.82) with relatively small standard deviations (0.07 to 0.16). Performance is slightly worse on average for a G' of 20,000 and 150,000 with average scores of 0.81 to 0.71 for 20,000 and 0.82 to 0.76 for 150,000. This gives confidence in the model to produce molecules for desired G' of 20,000, 50,000 100,000 or 150,000 – provided that the outputs are chemically diverse.

However, the performance of our model for G' values of 5,000 and 75,000 requires improvement in terms of score. For a G' of 75,000 there are some promising scores of 0.74, 0.7 and 0.72 for 2, 5 and 20 starting molecules but the remaining starting molecules give average scores of 0.68, 0.67 and 0.66 for 10, 50 and 100 starting molecules. Performance is poorest overall for a desired G' of 5,000 with scores ranging from 0.72 (± 0.21) for 5 starting molecules to 0.65 (± 0.13) for 50.

Overall, the ability of the approach appears to be heavily dependent on the G' value desired and it corresponds closely to the G' values of the enumerated library. Figure 5.14 shows the distribution of the G' values in the virtual library which shows a large density of molecules with predicted values around 50,000 and very small density around 5,000. However, there is also low density around 100,000 and 150,000 which performs relatively well. Closer inspection shows that there are only 44 molecules in the virtual library with G' values below 5,000 showing it to be at the very edge of our G' range. This results in the molecules in the pool of 1,000 being further away from the desired value for 5,000 than any other G' value. Compare this with the number of molecules with G' above 150,000 of which there are thousands. It is therefore unsurprising that the 5,000 runs perform poorest overall.

Distribution of CReM scores

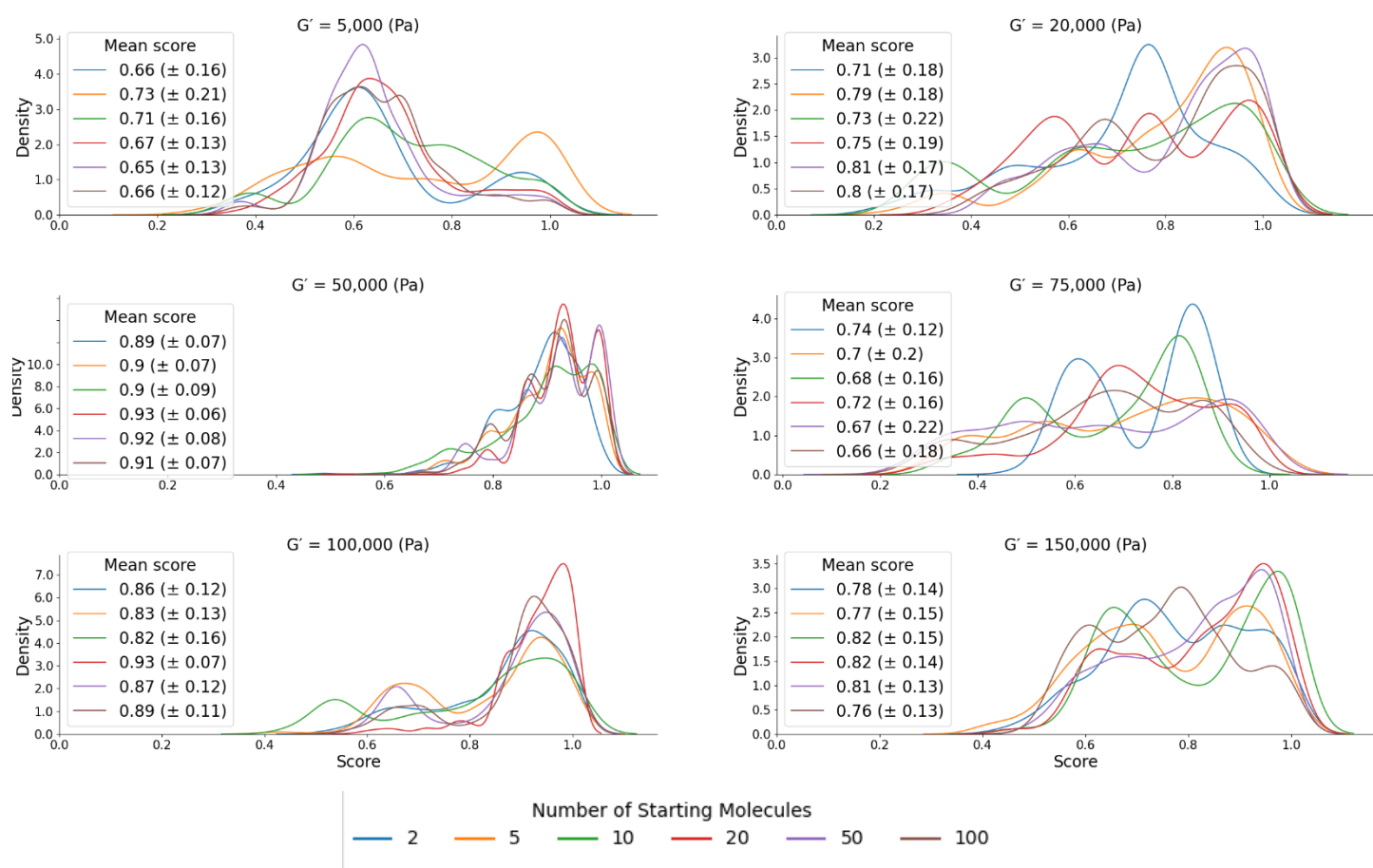


Figure 5.13: Distribution of score values for the 10 repeats across the G' and starting molecule hyperparameters.

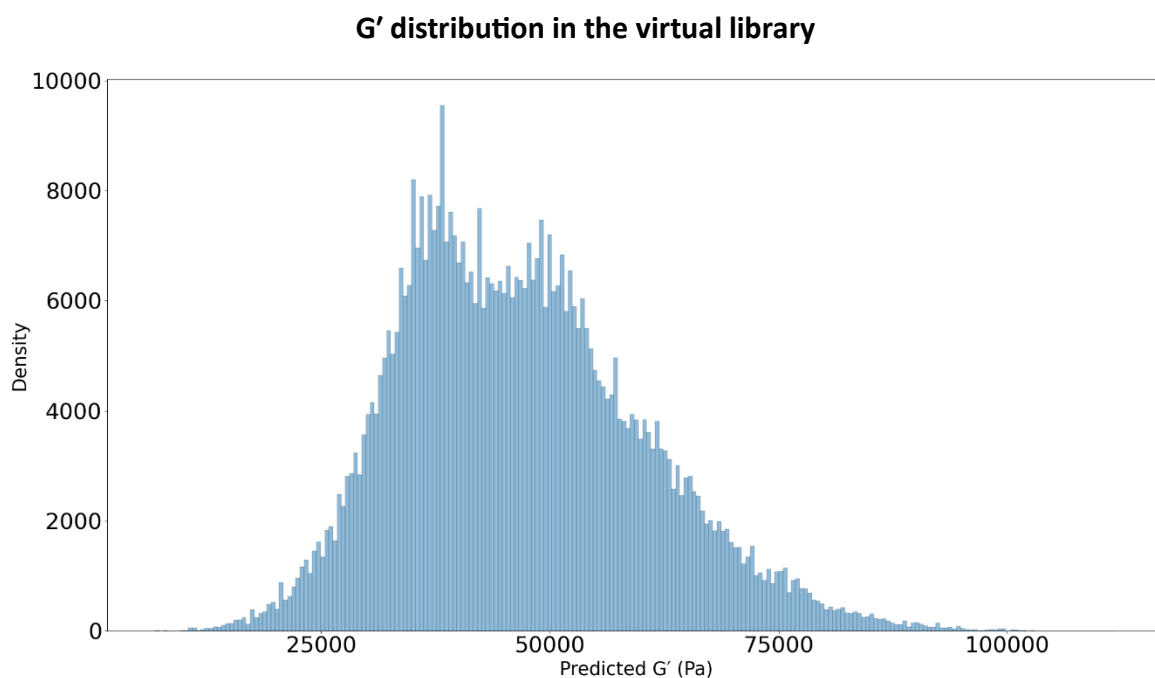


Figure 5.14: Distribution of the predicted G' values in the virtual library

5.3.2.1.3 Is it learning?

Since it appears that the score is varied depending on the G' value, whether or not the CReM-GA is learning and mutating molecules with desired G' properties is assessed in terms of the average score at each iteration.

Figure 5.15 shows how the score changes as a function of the iteration. Given that our starting molecules are chosen by how close they are to the desired value, they are likely to possess scores close to 1 meaning that even by iteration 0 the molecules are within a region of chemical space that possesses a high score. Therefore, the prospect of improving as a function of iteration is unlikely and a plateau around the initial high scores would correspond to a successful run. For a G' of 20,000 and 50,000 there is a distinct plateau of the scores regardless of the number of starting molecules suggesting that the algorithm can stay in the high score region for these values. However, the approach suffers slightly in the score for 75,000 and 150,000 with an initial drop but the scores remain above 0.7 at iteration 10.

Desiring a value of 100,000 also begins to drop and drops to a lower average score than the 75,000 and 150,000, dropping to 0.6 for 2 starting molecules. The remaining starting molecules do appear to plateau at 0.7 giving some comfort in its performance.

However, like in the score distributions of Figure 5.13, the 5,000 runs quickly begin to drop to poor scores, reinforcing the difficulty for the approach to produce molecules with this G' value which is unsurprising given the scarcity of these points in the virtual library suggesting it could be a value that few molecules possess.

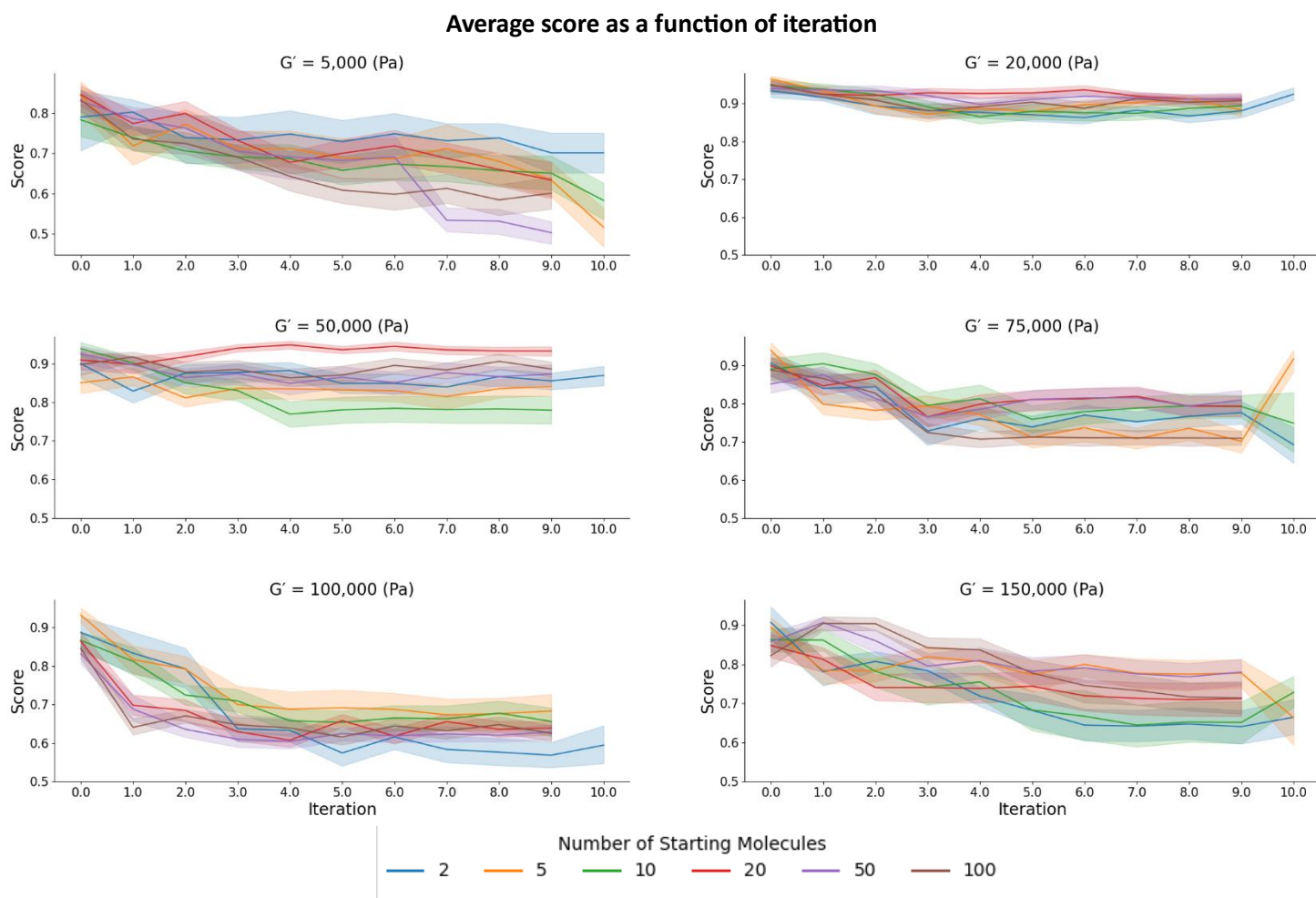


Figure 5.15: Average score for the 10 repeats across the G' and starting molecule parameters. The shading is the range across the 10 repeats.

5.3.2.1.4 Molecular variation within outputs

One potential drawback to running the CReM-GA for a set output length rather than for a set number of iterations is a lack of molecular diversity in the output if the GA runs for only a few iterations. To assess the molecular diversity of the output molecules, we calculate the Extended Connectivity Fingerprint (ECFP_4) and calculate the average pairwise Tanimoto similarity of the 2048-bit string within an output to gauge how similar, on average, the molecules within the set are to one another.

The Tanimoto similarity is a similarity metric used commonly in binary sets. It calculates the ratio of the size of the intersection over the size of the union.⁵¹ Where the intersection is the number of elements that exists in both sets being considered (*i.e.* the same bit set in two separate ECFP bit strings) and the union is all bits in both molecules. It gives a ratio of the number of common bits between molecules and is a measure of their similarity.

A popular threshold for the Tanimoto similarity is values about 0.85 classify two molecules as similar but this was defined within medicinal chemistry with the idea that molecules with similarity above 0.85 are more likely to possess similar activity⁵² and work has shown that this is not always true and is heavily related to the descriptor used to compare the molecules.⁵³ Other work into similarity thresholds comparing molecular similarity of drugs showed that a Tanimoto threshold of 0.5 using the ECFP_4 descriptors corroborated opinions from experts closely.⁵⁴ Therefore, with no definitive thresholds for what constitutes similar molecules with Tanimoto similarity scores of 0.5 and below will be considered dissimilar.

Interestingly, the results from Figure 5.16 suggest that there is very little difference in the similarity for each G' value, regardless of the number of starting molecules chosen. It would be expected that there would be less similarity when the number of starting molecules is high as there are more molecules being mutated in the initial iteration resulting in a larger pool for consideration as candidates to be mutated in the subsequent mutation, however, this does not appear to be the case.

Average Similarity within each output

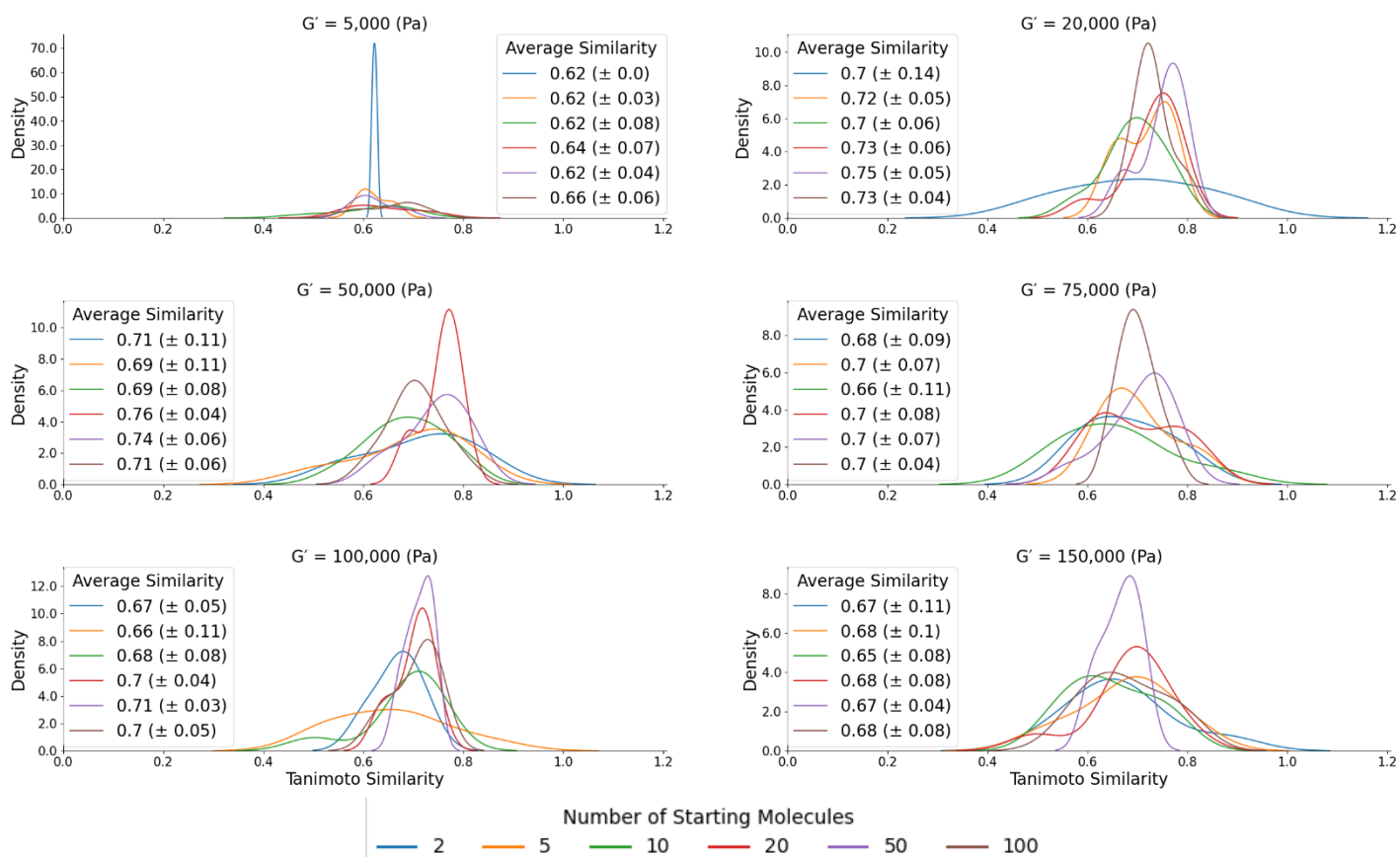


Figure 5.16: Average Tanimoto similarity within the outputs for each starting molecule and G' hyperparameter pair.

To visually inspect the molecules across all outputs, we randomly sample 50 molecules generated throughout the procedure and compare the sampled molecules. Upon visual inspection, it becomes clear that the molecules retain their naphthalene linker unit upon mutation with all 50 molecules containing some form of naphthalene moiety. Histidine and indole moieties are also very common in the molecules even though they were only present as R groups (rather than naphthalene which formed the molecular backbone) in the virtual library.

However, the molecules are rather similar in that they all remain some form of functionalised peptide (usually tri or tetrapeptides) with differing levels of R group functionality. Given that the molecules all seem to retain their peptide backbone, it is unsurprising that the similarity within each output remains high.

5.3.2.1.5 Molecular variation between iterations

Since Figure 5.16 demonstrated that the molecules are generally similar to each other in the output it would be hoped that the similarity to the molecules in iteration 0 should decrease as the number of iterations increase as this would demonstrate the “migration” of the molecules to different chemical spaces. The results from iteration 0 were chosen as the starting point as they demonstrate the variation caused by mutations to the molecule rather than any variation in the initial pool, which varies in size based on the number of starting molecules.

Figure 5.17 shows the average Tanimoto similarity of the molecules from each iteration with the molecules outputted from iteration 0. There is a modest decrease in similarity across all G' and starting molecule pairs and by iteration 9 there are outputs with similarities to iteration 0 molecules below 0.5 making them dissimilar outputs by the Tanimoto threshold set. 10 mols for a G' of 150,000 has an average Tanimoto similarity between iteration 0 to iteration 9 of 0.41 making these sets dissimilar. There are 4 other runs with similarities below 0.5 for iteration 9, all of these for a desired G' of 5000. There are 5 molecules (0.39), 20 (0.47), 50 (0.45) and 100 (0.49).

Overall, this approach completes quickly (within 40 minutes) and has the ability to produce dissimilar molecules to those outputted from iteration 0 within 10 iterations. There are concerns about the scores produced for the G' of 5,000 runs but overall the performance is promising.

As well as considering the approach as a whole, it is worthwhile investigating the results of a single output. 20 molecules for a G' of 50,000 is chosen as it is the best performing set of hyperparameters based on the average score. Of the 10 repeated runs, the one with the highest average score has been chosen as the single output to investigate.

Average similarity as a function of iteration

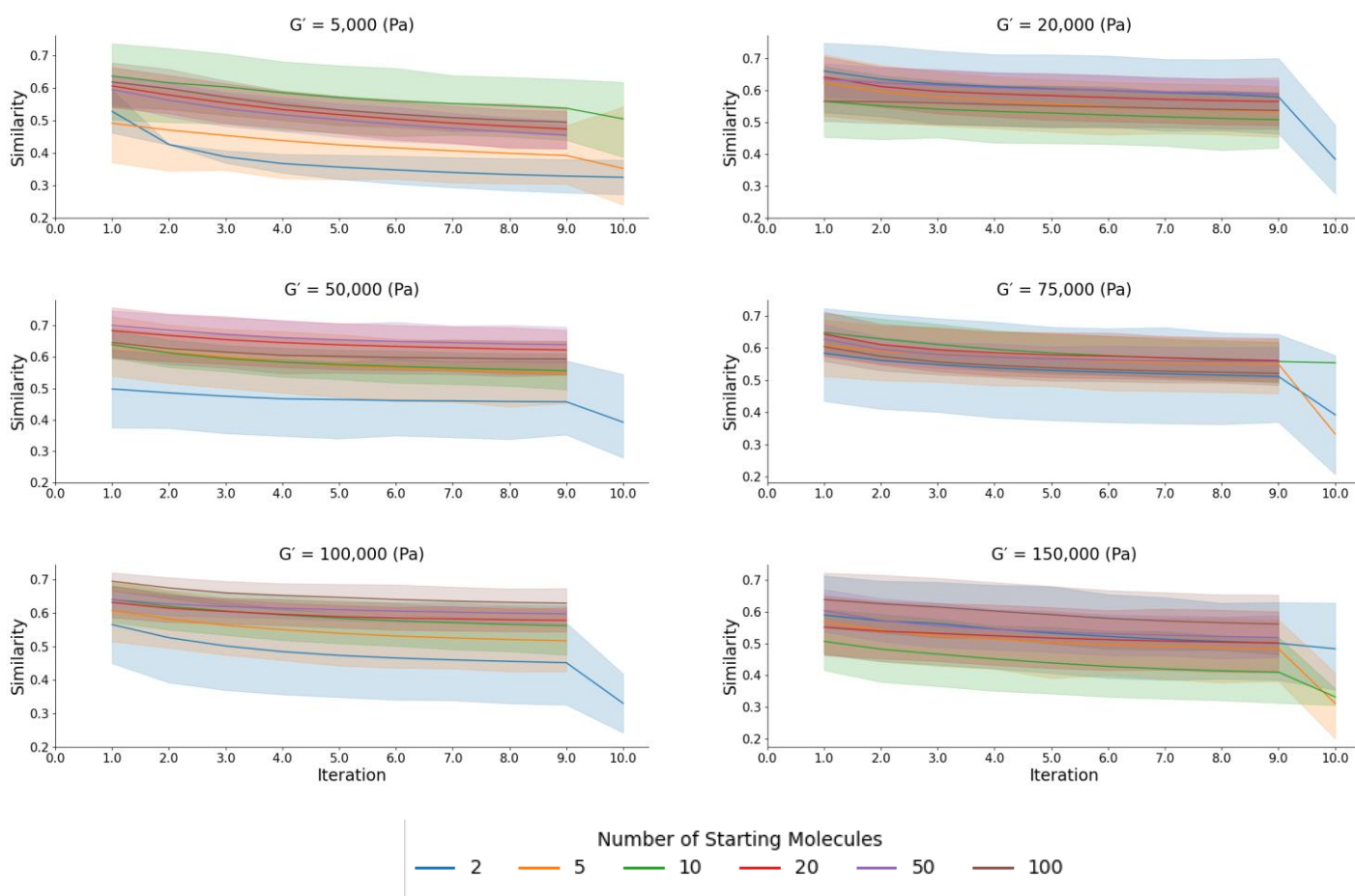


Figure 5.17: How the similarity of the molecules at each iteration to the results of iteration 0 varies as a function of iteration.

5.3.2.1.8 Example output

The run with the highest average score is repeat 2 and its output will be investigated. The molecules from this run are summarised based on their AlogP, Molecular Solubility, Number of Rings and Molecular Polar Solvent Accessible Surface area and compared to the virtual library. The run outputs a total of 100 molecules with the scores ranging from 0.63 to 0.99. Of these 100 molecules, none are also present in the virtual library showing that the CReM has expanded the chemical space somewhat. However, 57 are duplicates which suggests that there are duplicate molecules in each output and would explain the relatively high output similarity observed in *section 5.3.1.2.5 - Molecular variation within outputs* but still has the potential to mutate molecules back to their parent compound. In future, this would be fixed to ensure 100% novel molecules in the output.

Figure 5.18 shows the distribution of values for the 100 molecules outputted by this run. The average AlogP is $4.38 (\pm 0.69)$ which is within the upper range of the virtual library. Each molecule sampled had 5 rings whereas the virtual library more commonly has 2, 3 or 4 rings. The Molecular_PolarSASA was $292.87 (\pm 27.58)$ which is similar to the average of the entire library of 330. Finally, for the solubility the average solubility was $-10.82 (\pm 0.46)$ and is slightly lower than the average for the virtual library of -7.50 .

Example output molecular property distributions

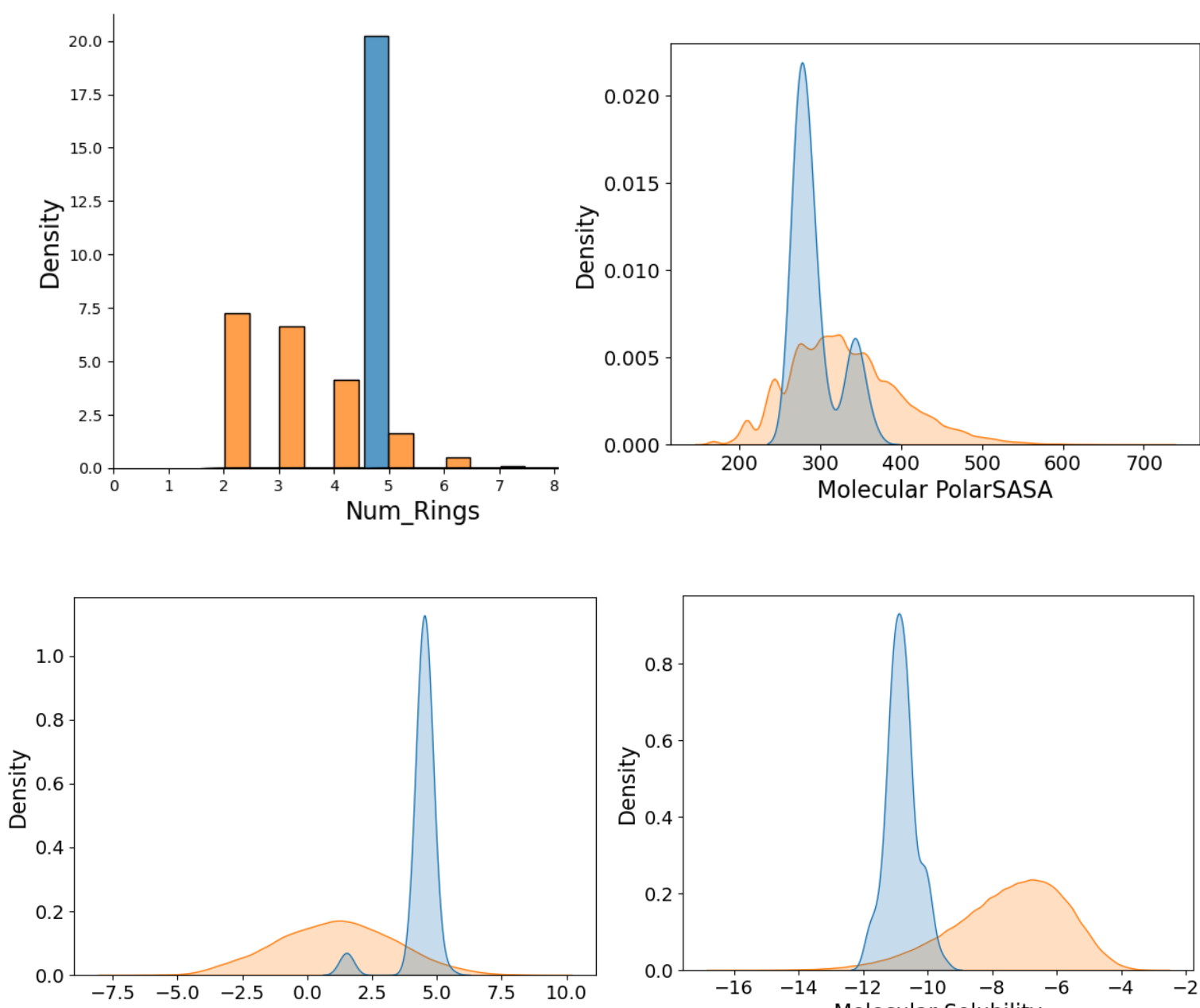


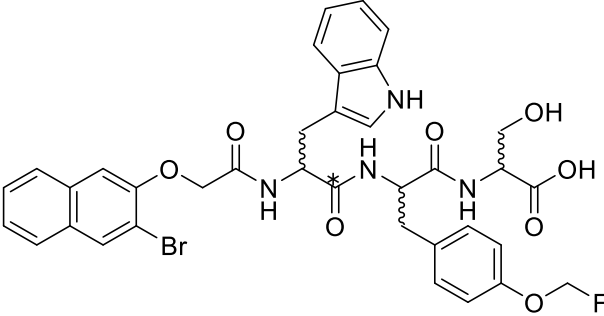
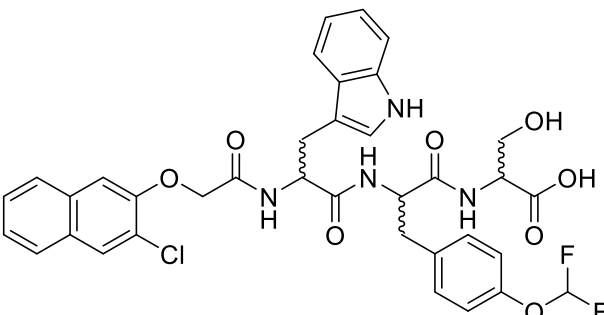
Figure 5.18: Distribution of the molecular properties for the example output with the virtual library distributions super imposed

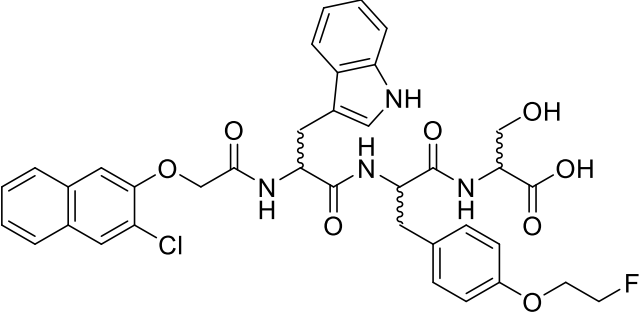
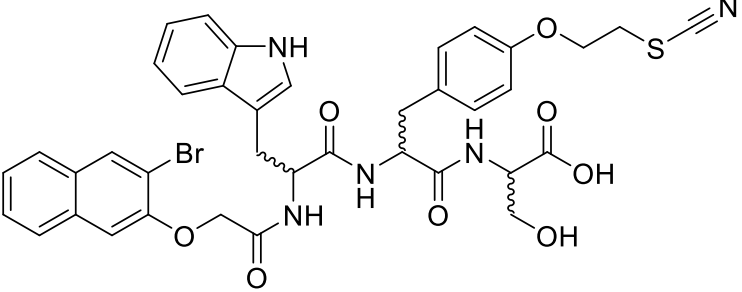
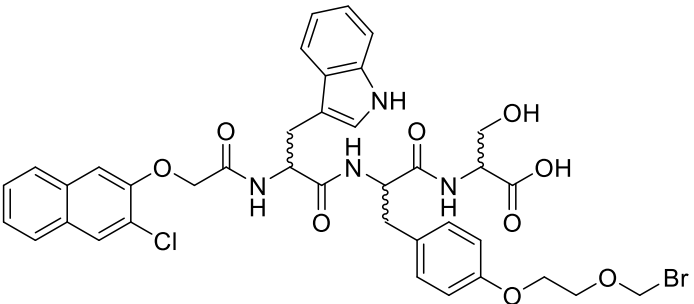
The molecules with the highest scores are shown in Table 5.1. The molecules in the table highlight the disadvantages of this CReM approach. The fragment-based nature of the replacements during mutations results in structurally very similar molecules. If the modification is small enough, the change may result in minor or no difference in the molecular descriptors used to predict the gel state or the G' of the molecule. This results in identical predictions for the mutated molecules and thus numerous analogues occupying the top spots in an output and is a contributory reason to the lack of dissimilarity between the outputs. The model uncertainty in this prediction also demonstrates some concern with the outputs. The standard deviation being larger than the predicted G' results in little faith into the models predictions for these “good” molecules.

To address this, in the future it would be useful to place either a Tanimoto similarity filter or include the similarity measure as an input to the pareto sort of all new molecules to be added to the output and only add the molecule if it is suitably dissimilar to the molecules already present in the set, say based on a 0.5 threshold. This would help produce more chemically diverse outputs.

With this caveat, the CReM-GA approach does produce some novel molecules with desirable properties and should be considered a useful starting point for the generation of new molecules albeit one which requires optimisation. The CReM-GA approach will now be compared to the RNN REINVENT approach. The approach will be compared based upon the time to run and the quality of the generated molecules.

Table 5.1: Top 5 molecules from the 20 starting molecules 50,000 run based on score.

Molecule	Score	Iteration	P(Gel)	Predicted G'
	0.999	2	0.91	49,900 (± 67,000)
	0.999	2	0.91	49,900 (± 67,000)

	0.999	2	0.91	49,900 (± 67,000)
	0.999	3	0.91	49,900 (± 67,000)
	0.999	3	0.91	49,900 (± 67,000)

5.3.3 Reinvent RL

5.3.3.1 Prior Validity + Uniqueness

Before we can use our Recurrent Neural Network to generate new models, the Prior model which forms the starting point for all sampling in the agent needs to be validated. The prior model is the RNN model that was trained on the virtual library to learn the syntax of the SMILES strings in the virtual library. Each time the reinforcement learning is carried out, the probabilities for each character at each time-step is initialised as the final values during training of the prior.

To validate the prior model, we generate 50,000 molecules by sampling from the prior and assess the molecule's validity and whether they are also present in the enumerated library. The hope is that our prior model can successfully replicate and sample from our virtual library, similar to the sampling of the gdb-13 space work by Arús-Pous *et al.*⁵⁵ As the agent learns, the hope would be that the number of molecules it samples that were also present in the virtual library would decrease, as it learns to sample new chemical space.

Of the 50,000 sampled molecules 49,899 (99.80%) were valid molecules and of the valid sampled molecules, 47,121 (94.43%) of them were unique molecules. When comparing the valid sampled molecules to the enumerated library, 47,016 (94.22%) of the molecules were also present in the virtual library demonstrating that our prior successfully samples from the virtual library. Figure 5.19 shows the distribution of the molecular property parameters of the sampled molecules compared with the enumerated library.

Given that a large proportion of the sampled molecules were also in the virtual library, the similarity of these molecular properties is unsurprising. Figure 5.20 shows the similar distributions of the Synthetic Accessibility of the sampled molecules along with the predicted G' again showing significant overlap. The synthetic Accessibility (SA) of a molecule is a numeric measure that attempts to capture the ease of synthesis for a molecule.⁵⁶ It is calculated using two terms, the fragmentScore and the complexityPenalty where the fragmentScore is calculated as the sum of all contributions by each fragment in a molecule. Each fragment is assigned a score based upon the prevalence of that fragment in the PubChem collection⁵⁷. The complexity penalty is a number that summarises the presence of complex fragments in the molecule in terms of the ring and macrocycle complexity, size of rings and stereochemistry.

Molecular property distributions of the prior samples

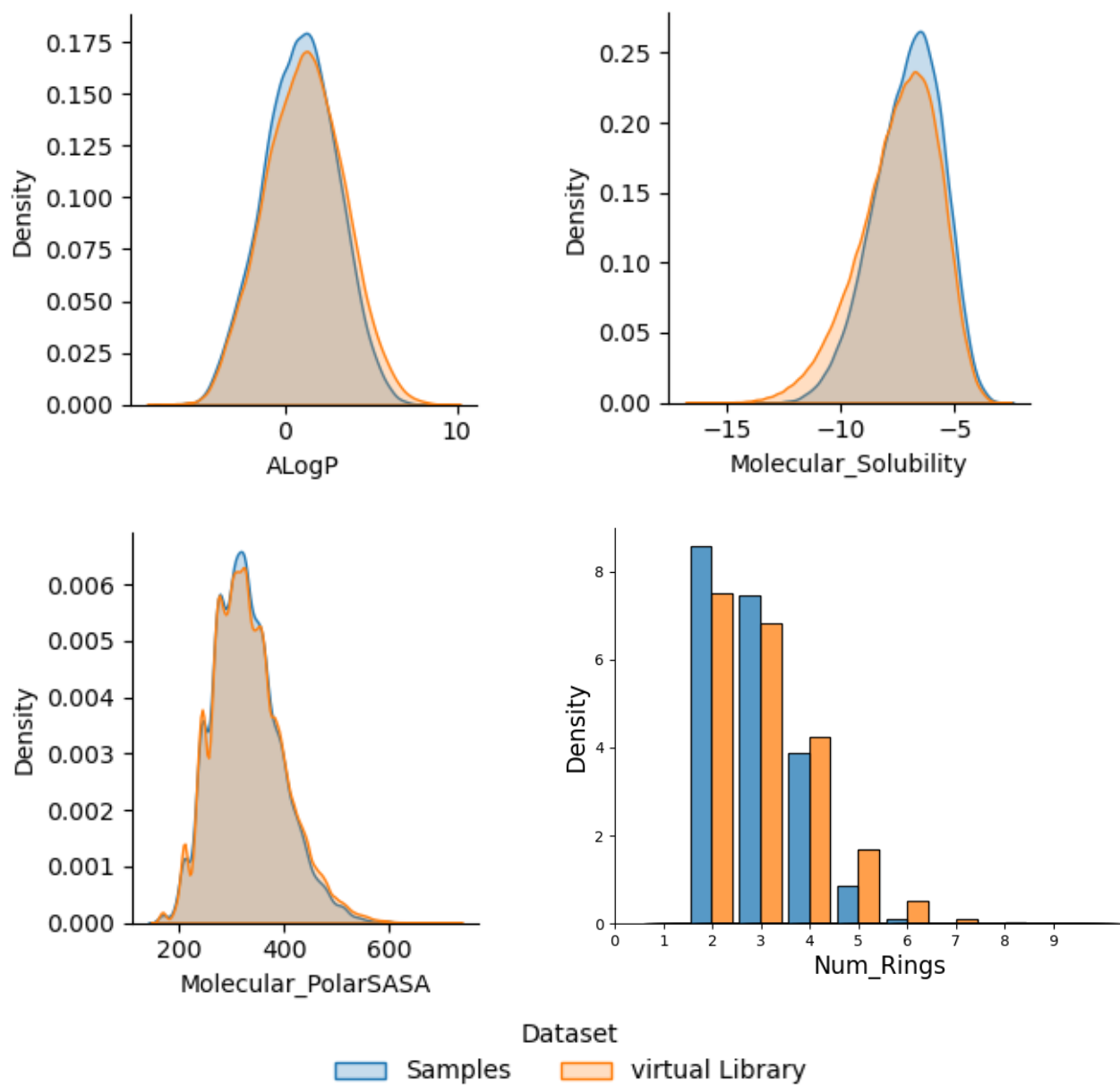


Figure 5.19: Distribution of the AlogP, Molecular_Solubility, Molecular_PolarSASA and Num_Rings for the virtual library (yellow) and the prior samples (blue).

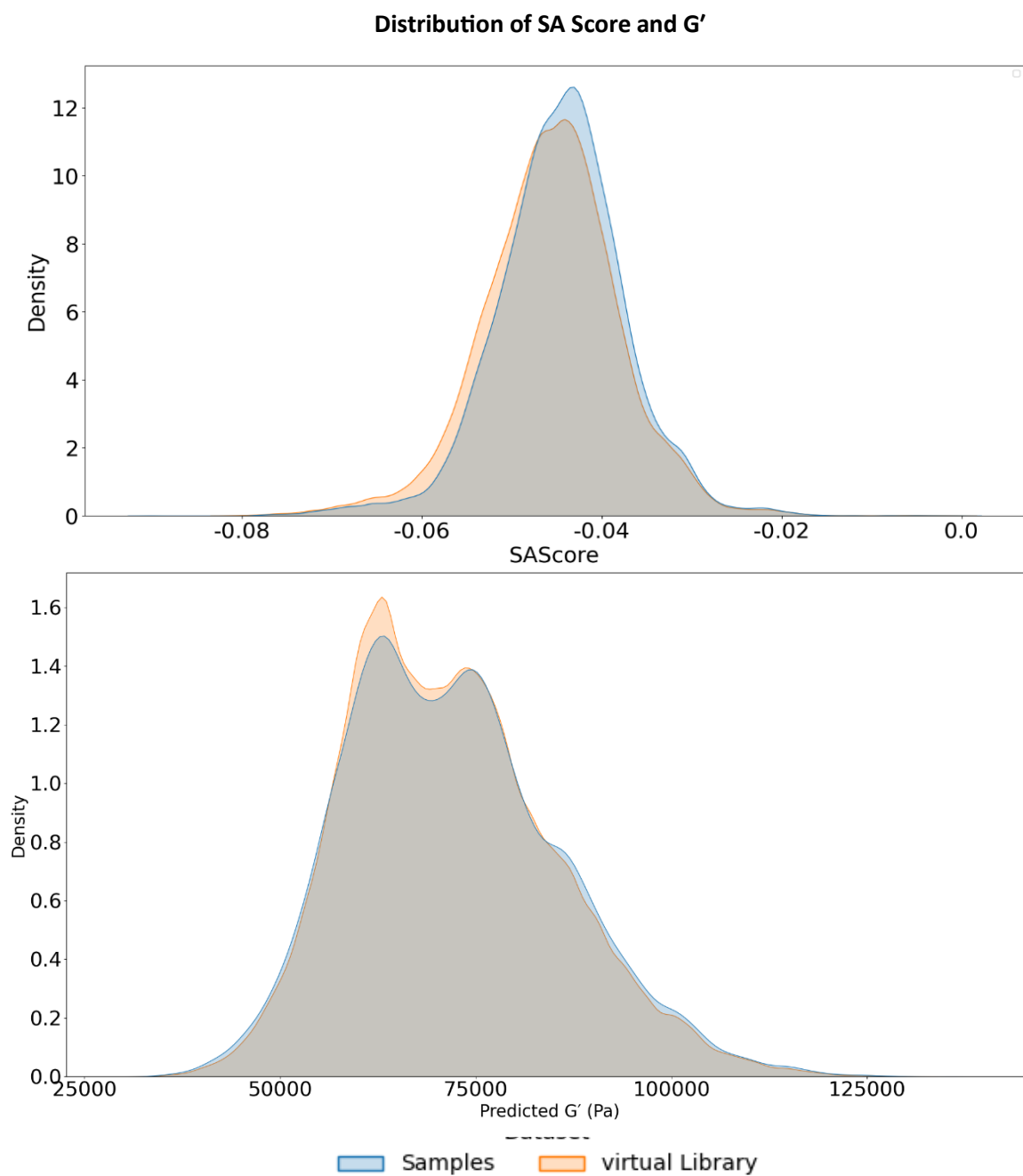


Figure 5.20: Synthetic Accessibility (SA) score and Predicted G' for the virtual library (yellow) and the prior samples (blue).

Figure 5.21 presents a random selection of molecules sampled by our prior RNN model. Each molecule sampled is a valid molecule and is also found to be present in the virtual library used to train the model.

Chemical depiction of prior samples

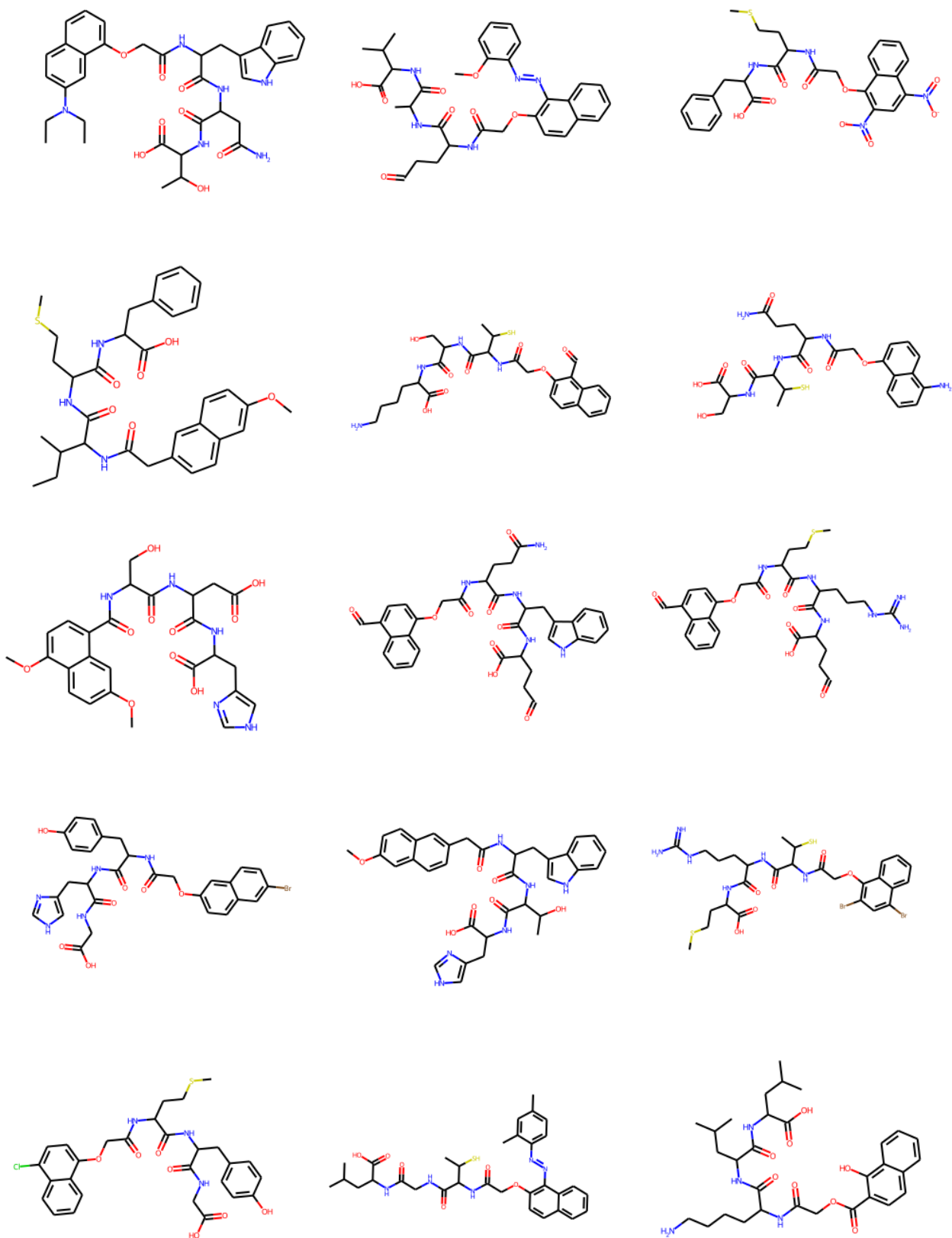


Figure 5.21: Example sampled molecules from the prior model

5.3.3.2 Agent Timings

Now that we have an underlying prior model that understands the syntax of our chemical space, we can use it, along with the agent, to generate molecules with the aim to minimise the difference between a desired G' value and our BART model's G' prediction (see Chapter 4 Section 4.3.1 – G' BART model). To assess the performance of our agent RNN we run the agent with the goals of generating molecules with G' values of 5,000, 20,000, 50,000, 75,000, 100,000 and 150,000 (the same as the values desired in the CReM GBGA work (section 5.3.2)) to investigate the ability of the model to generate molecules across a range of G' values.

Before investigating the quality of the molecules generated – we investigated the time taken for the RNN to run for a number of steps (100, 500, 1000, 1500 and 2000 steps) 10 times for each combination of steps and G' . The range of steps were chosen as they give a range of values either side of the default number of steps of 1000. The number of steps can be thought of as iterations of our agent. Figure 5.22 shows the distribution of the time taken to run the RNN across the range of G' values for each number of steps considered.

From Figure 5.22 it becomes clear initially that there is very little difference in the time taken for the RNN to execute for a given number of steps across all 6 G' values considered with the shortest time taken for the RNN to run for 100 steps at 2079s (± 126 s) and the longest at 2000 steps of 67024s (± 6140 s), giving a range of run-times between approximately 30 minutes and 18 hours. This is a very wide range of timings and much longer on average than the CReM-GA so it would be hoped that the scores would be high and the output diverse to justify running the RNN for that long.

RNN Timing Distributions

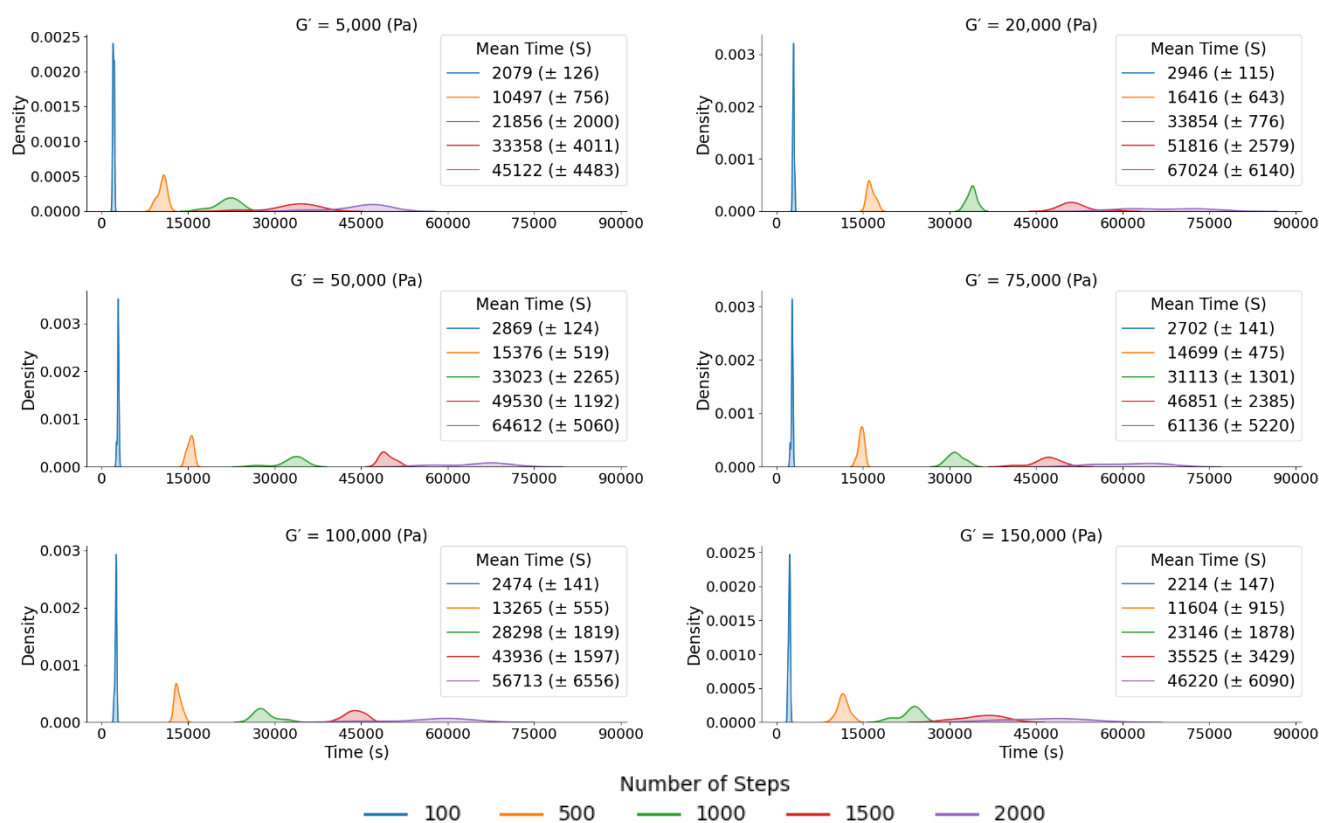


Figure 5.22: Timings for the RNN Agent learning for 100, 500, 1000, 1500 and 2000 steps.

5.3.3.3 Average closeness of generated molecules to desired value

Since our aim is to generate molecule with certain G' values, the average score of the output is investigated to gauge how well it does in generating high scoring (i.e. molecule with desired properties) molecules. Figure 5.23 shows that the model produces a significant number of non-gels (these are assigned a score of 0) resulting in a bimodal distribution showing that there are two distinct areas sampled by our model. Ideally, the number of molecules assigned a score of 0 should decrease as the model learns to produce molecules with preferable characteristics (see section 5.3.3.4 *Is it learning?*). The plots in figure 5.23 show that the model can struggle to sample gels, particularly at $G' = 5,000$, 100,000 and 150,000. From the distribution plot of our virtual library in figure 14, it becomes apparent why this is the case. The density of G' values is highest between 25,000 and 75,000 meaning that the chemical space the model has learnt is for molecules with G' values in this range.

Average Score for the RNN model

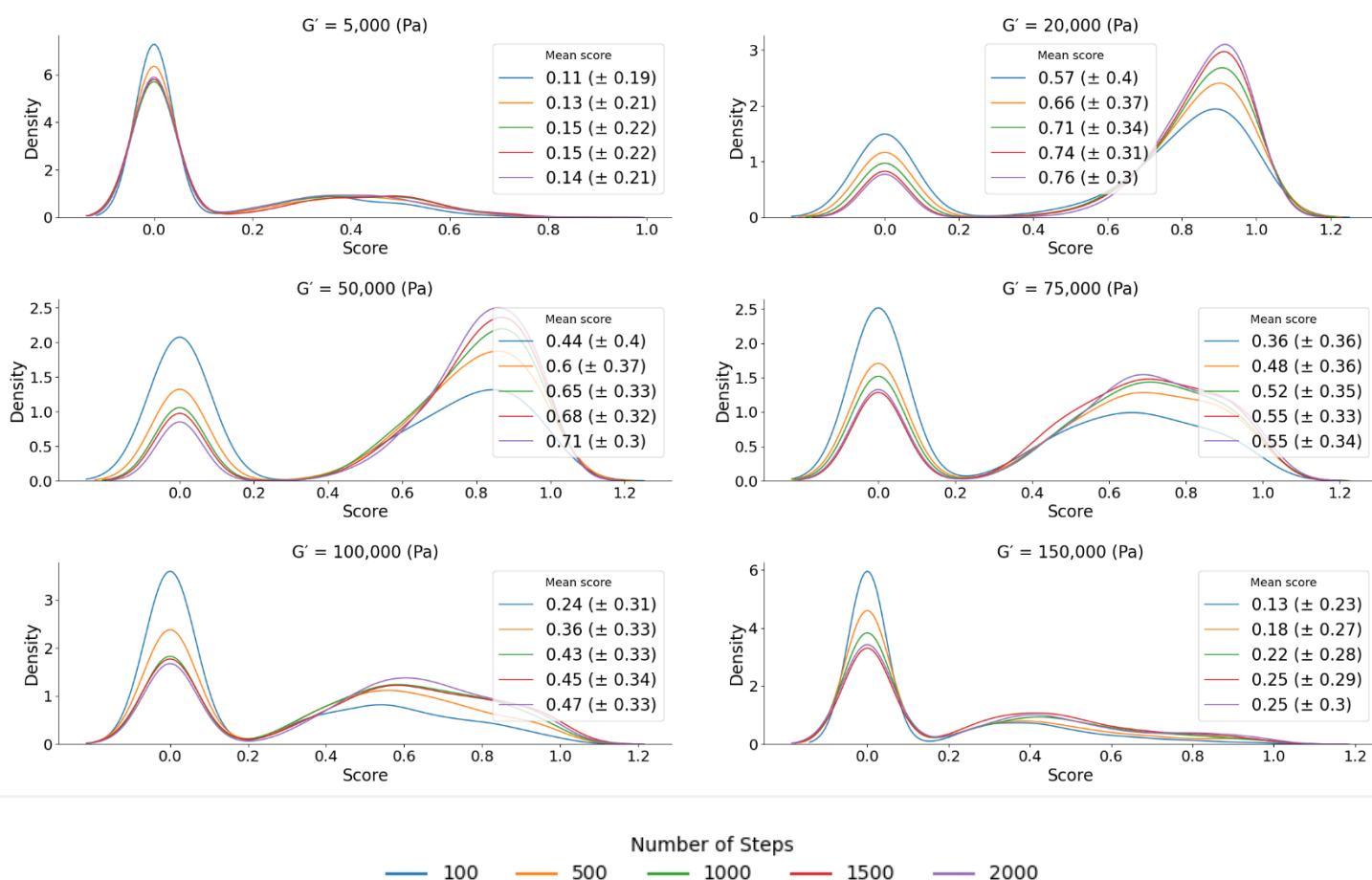


Figure 5.23: Average scores for the output from each run.

However, with the knowledge that a considerable amount of the sampled molecules are non-gels, Figure 5.24 shows the average score of the outputted molecules that are gels. Figure 24 here highlights further that the model was learnt on a dataset under sampled at a G' of 5,000, 100,000 and 150,000. Scores for a G' of 5000 are poor ranging from $0.38 (\pm 0.13)$ for 100 steps to $0.44 (\pm 0.13)$ for 1500 steps. Performance is still poor at a G' of 150,000 with the highest score $0.53 (\pm 0.21)$

However, in the $G' = 100,000$ plots there are some distributions at scores > 0.6 , giving hope that there are some molecules sampled with good G' values. Here, the best performing runs for 100,000 is 1500 steps with an average score of $0.66 (\pm 0.19)$.

For a G' of 20,000, 50,000 or 75,000 the results are much more promising. Average scores ranging from $0.86 (G' = 20,000 \text{ for } 1500/2000 \text{ steps})$ to $0.68 (G' = 75,000 \text{ for } 100 \text{ steps})$ shows that there are molecules sampled from these runs with values close to those desired.

Average Score for the RNN model – Non-gels removed

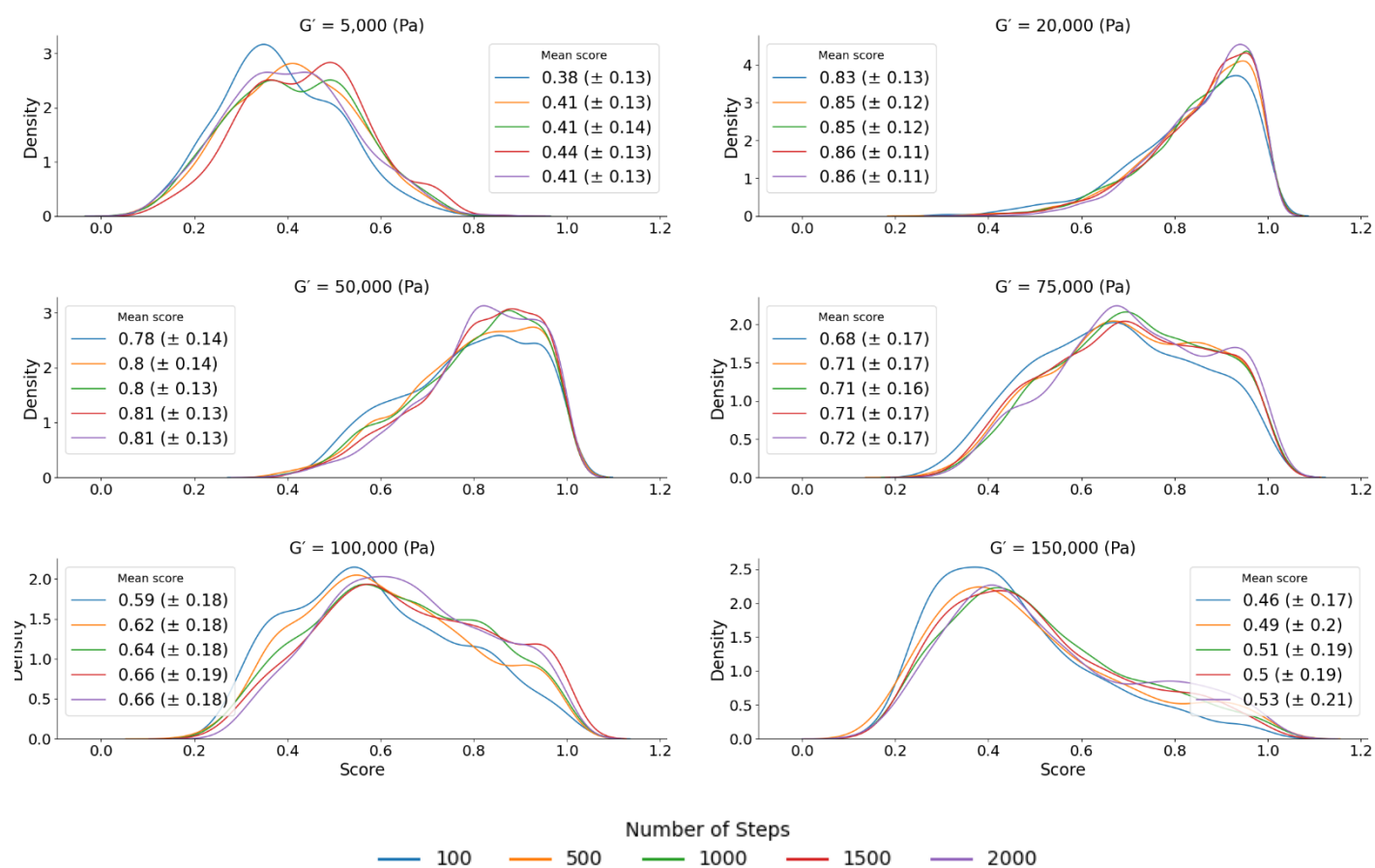


Figure 5.24: Average scores of the sampled molecules that are predicted to form gels.

5.3.3.4 Is it learning?

Since the RNN is being ran for a up to 2000 steps resulting in long time to complete (up to 18 hours on current architecture) it would be hoped that some of the poor average scores in Figure 5.24 are being hampered by poor initial performance and that scores in later steps would be high. To investigate this, the average score as a function of step is plotted in Figure 5.25 to gauge if the RNN is learning to sample not only molecules that are predicted to form gels but those that possess favourable G' values.

From Figure 5.25 there is a degree of learning that occurs quite quickly (in the case of G' values of 20,000, 50,000 and to a lesser extent 75,000) before a plateau for these values between 500 and 1000 steps. For the G' of 20,000 and 50,000 there is no obvious increase in average score from 1000 to 2000 steps so the results from 1000 steps onward don't warrant running the algorithm for twice as long.

Again, G' of 100,000 does improve but remains poor with an average score of approximately 0.50 after 200 steps. For 5,000 and 150,000 there doesn't appear to be any learning across the 2000 steps with the scores after 2000 steps being very similar to those at iteration 0. The results suggest that the virtual library G' values become the limiting factor for both the RNN and the CReM-GA. If there were more points in the virtual library at these values it is reasonable to expect comparable performance to 20,000, 50,000 and 75,000. For the remaining analysis, work will focus on the 20,000 and 50,000 runs for 500 and 1000 steps as these offer the best average score and the time to run for the 500 and 1000 steps is considerably quicker than 1500 and 2000 with marginal improvement in average score.

Average score as a function of step

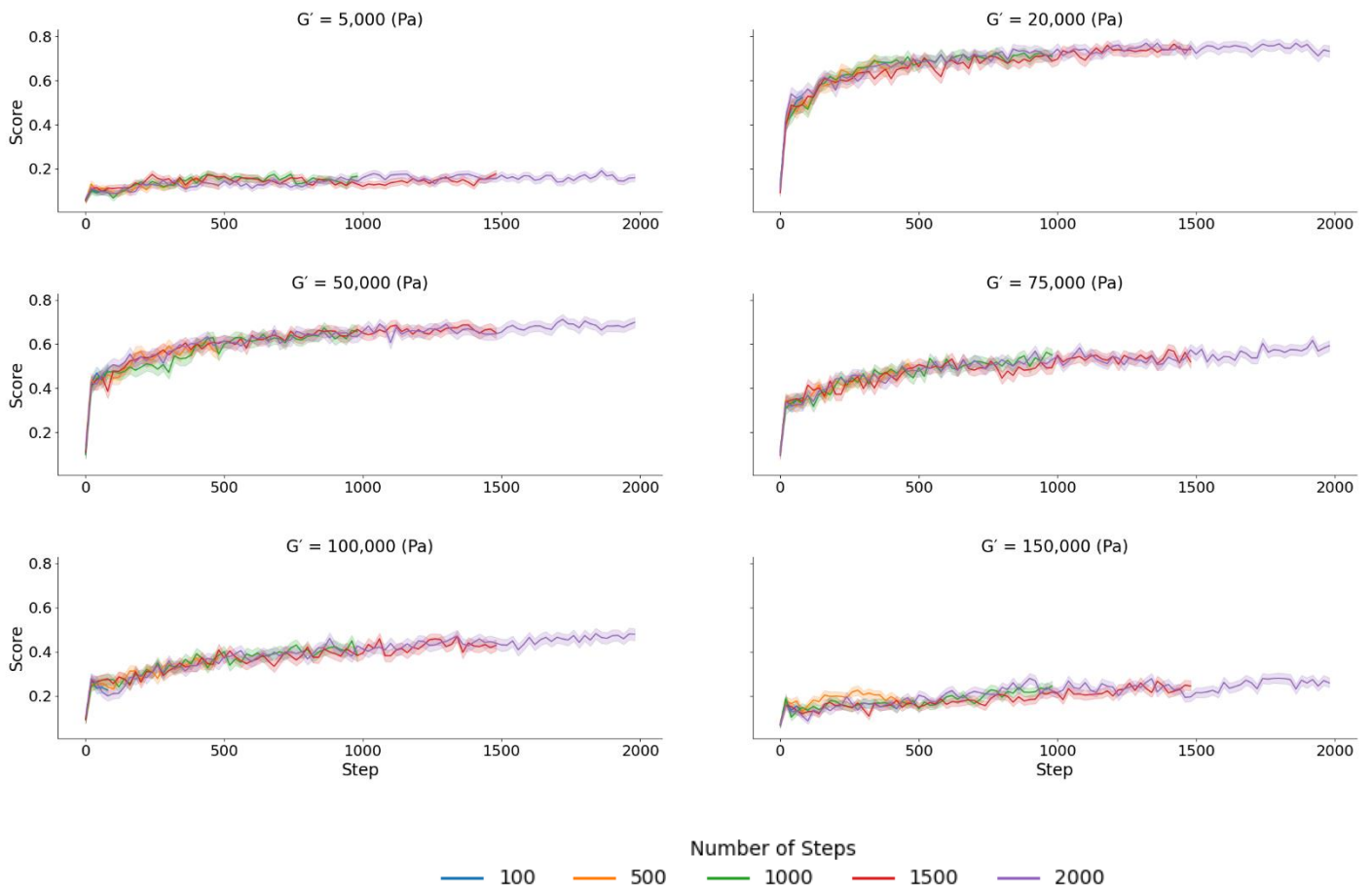


Figure 5.25: Average score at each iteration for each number of steps.

5.3.3.5 Assessing the output

Figures 5.26a-c display the molecules sampled at each step for the 20,000 and 50,000 runs of 500 and 1000 steps in terms of their validity, uniqueness with respect to each other and uniqueness in terms of their presence or absence in the virtual library. Figure 26a shows that the RNN has done a good job learning the syntax of SMILES from the virtual library as for every step, over 80% of the molecules are valid with the fraction eventually plateauing at around 0.95 resulting in an average validity of 95% for these later runs.

Figure 5.26b shows that the model very rarely samples the same molecule more than once which is unsurprising given that there is a check for this as part of the RNN. However, the plot in 5.26c is concerning for the results of the RNN approach and indicate a downfall in the approach. Although the percentage of molecules sampled from the prior model was very high (94%), the hope was that as the agent sampled, it would explore new chemical space. Although it does appear to be learning not to sample molecules in the virtual library – the time taken it takes to do this is concerning. As the number of steps approaches 1000, the model is still sampling over 80% of the molecules from the virtual library and given that 1000 steps takes approximately 8 hours to run – to only be sampling 20% of completely new and unique molecules after this length of time is concerning.

This lack of novelty in the sampling suggests a lack of diversity in the virtual library used to train the prior. This could be due to the relatively small number of molecules used to train the RNN (500,000) which is smaller than reported examples that use 1.4 million molecules⁵⁸, and 1 million molecules⁵⁵. Learning on a larger and more diverse set of molecules may allow more flexibility to be learnt by the prior which would pass on to the agent through the augmented likelihood. Moreover, how conservative the model is when sampling can be tuned through adding a parameter which applies a function to the probabilities at each time step to either increase or decrease the probability of each sequence to allow more flexibility in the sampling procedure used to generate new molecules.³³ Though this increased flexibility is a trade-off with validity as the model would begin to sample in regions of chemical space it hasn't learnt.

Overall, the RNN approach has the potential to produce novel molecules with desirable G' profiles but requires optimisation to improve the quality of the output. The RNN along with the CREM-GA both appear to be heavily impacted by the distribution of G' values in the virtual library and this causes both approaches to struggle at low and high G' values.

Validity, Uniqueness and Novelty as a function of steps

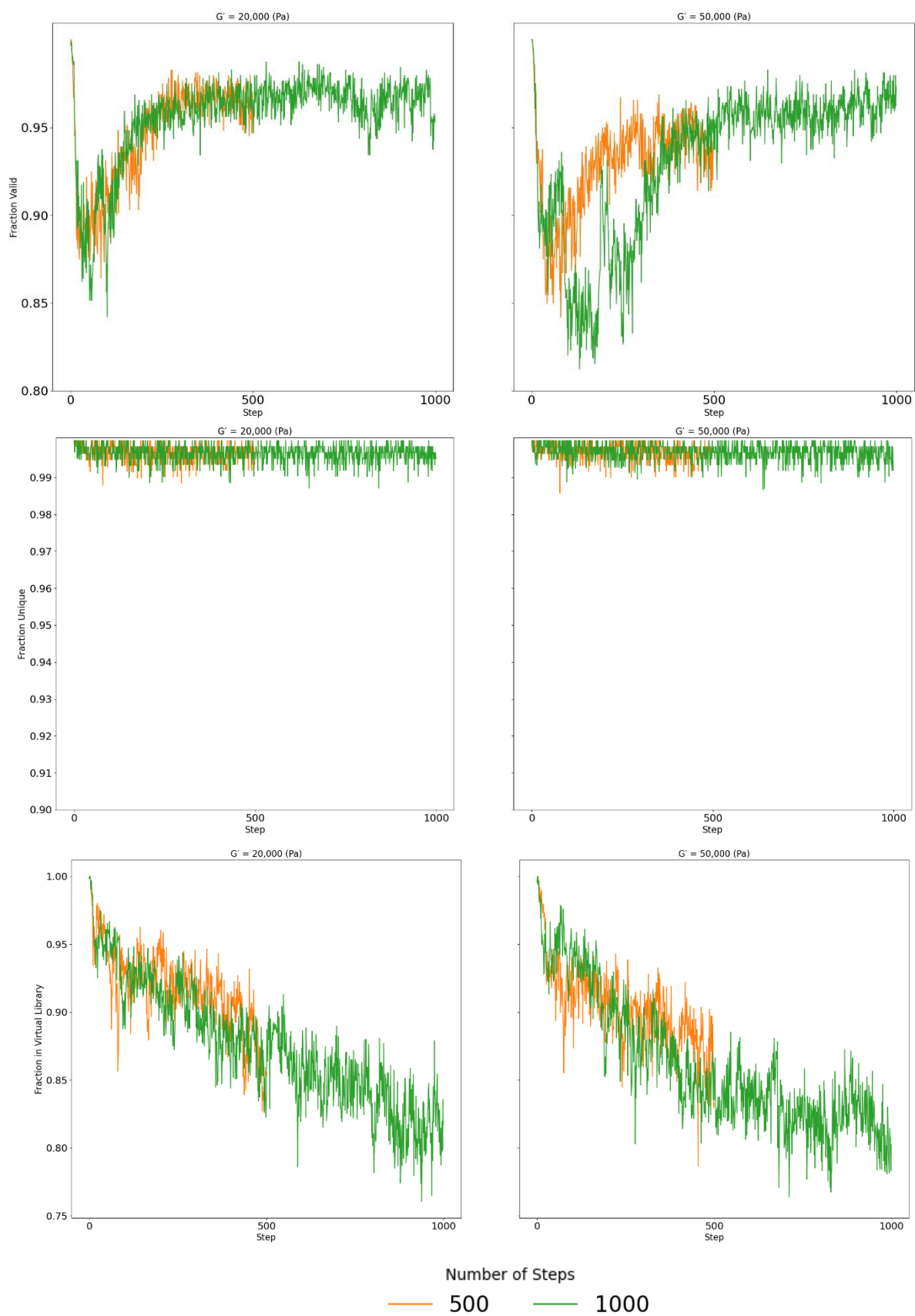


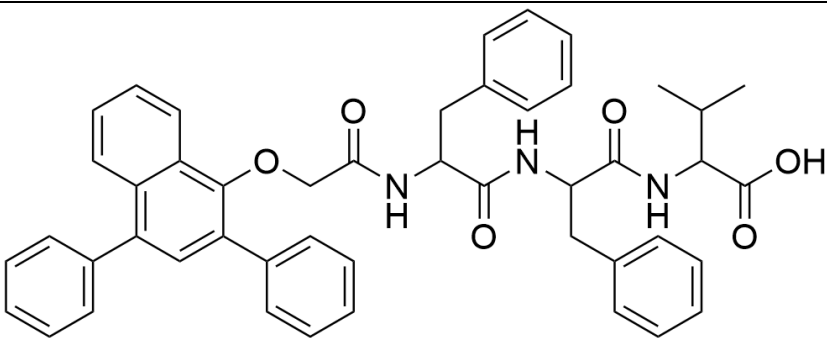
Figure 5.26: a) Fraction of Valid molecules at each step b) Fraction of the valid molecules that are not-duplicates and c) Fraction of valid molecules not also present in the virtual library.

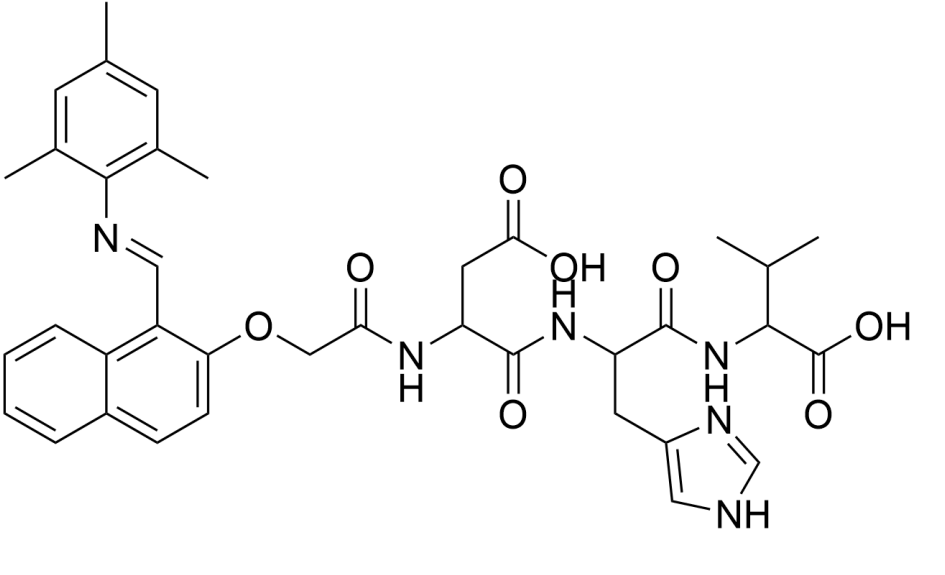
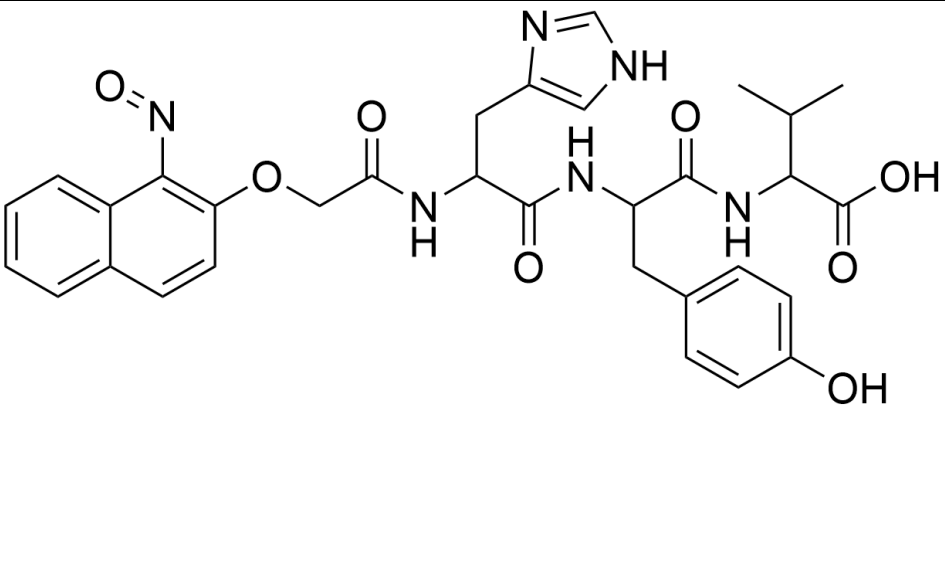
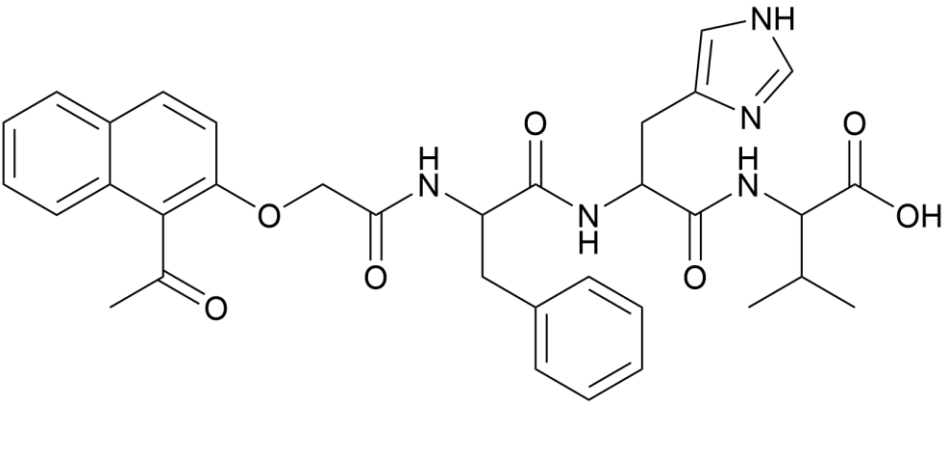
5.3.3.6 Visual Inspection of an output.

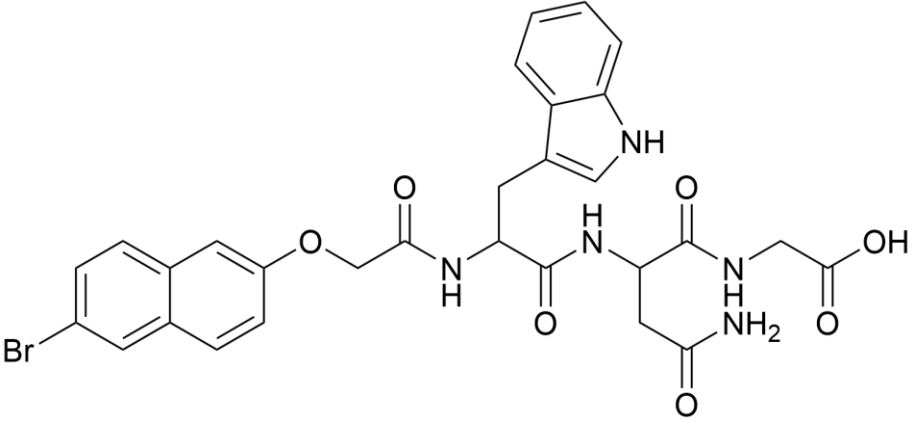
In the CReM runs (Section 5.3.2) we visualised the best performing molecules from a benchmarking run (Table 5.1). We replicate Table 5.1 to assess the output of the RNN with the highest average score ($G' = 75,000$, steps = 500) which is shown in Table 5.2. Visually there appears to be more variance in the molecules with the highest score in the RNN output than the CReM output (Table 5.1) with the RNN output not showing the same propensity to favour the same backbone with slight R group modification.

However, although these molecules are more diverse relative to those in the CReM output – the molecules here have a gel probability much closer to the decision boundary (~ 0.6 here vs 0.9 in Table 5.1). Moreover, the standard deviations in the predicted G' values become ridiculously large (maxing out at almost 250,000). Visual inspection of the distribution of the samples for these predictions shows that the standard deviation is majorly impacted by large outliers in the sampling procedure. This suggests that there needs to be some post-processing of the prediction distribution to remove values that are clearly outliers which should result in a more meaningful prediction standard deviation.

Table 5.2: Visual inspection of the best performing molecules from the RNN run with the highest average score.

Molecule	Score	Step	P(Gel)	Predicted G'
	0.999	15	0.66	75,000 ($\pm 92,000$)

	0.999	71	0.60	75,000 (± 142,000)
	0.999	122	0.63	75,000 (± 142,000)
	0.999	160	0.63	75,000 (± 142,000)

	0.999	340	0.61	75,000 (± 297,000)
--	-------	-----	------	-----------------------

5.4 Conclusions

In this chapter, two promising generative approaches are presented that can generate molecules with a desired storage modulus. However, both approaches require some improvement. With that said, both approaches lay the foundations for a potentially powerful approach to generate molecules that form gels with a desired storage modulus.

Looking forward, the lack of diversity in the CReM-GA approach could be addressed by adding a similarity check based on the Tanimoto similarity of each new candidate molecule to those already in the output. Moreover, fixing the duplicate molecule check will also allow for more diverse molecules.

Given the lack of diversity in the sampling of the RNN model, there are two potential approaches to address this. Firstly, training a model on a large dataset like the ChEMBL dataset, which isn't specifically in the dipeptide/tripeptide chemical space of the BART models would allow for greater variety in the prior model and learning on a larger more diverse set should reduce the level of sampled molecules also present in the training data. Moreover, the sampling approach can be modified to set how conservative the model is when sampling. Altering the probabilities for each SMILES character at each time step, which are defined by the prevalence of each character at that particular time step in the training set, to give greater probability for less likely characters would allow for more liberal sampling but comes with the added risk of more invalid molecules sampled.

5.5 Outlook & Future Work

Low molecular weight gels have the potential to impact society in a variety of innovative ways. Whether this will be felt most in medicine as drug-delivery mediums, in sensing as a form of pollution clean-up or in materials as a source of 3D-printing material remains to be seen. However, before their impact can be felt, there are many hurdles to overcome in the field.

Although these materials are exciting, the lack of design rules, the role of isomerism in gel formation and the variability in mechanical properties based on gelation method are just some of the hurdles that these materials face. The work presented in this thesis attempts to begin to address some of these by building upon already published work and presenting novel work that attempts to push the state of the art in the field.

Chapter 2 focused upon attempting to address the lack of design rules for gels through the use of interpretable machine learning. From this chapter, we have managed to derive conclusions around which molecular fragments are important for promoting and negating gel formation. Although these models are an important step towards a greater understanding around gelation, there is still room for improvement.

Given the nature of gel formation, the role of isomerism cannot be overlooked when making predictions as to whether a molecule is likely to gel. However, currently, the models presented in chapter 2 would give the same prediction for two stereoisomers. Therefore, there is scope to further extend these models by incorporating descriptors that consider 3D information about the molecule to allow the model to distinguish between stereoisomers. However, consideration into which type of 3D descriptor is chosen is needed to minimise the performance impact, from the extra computation needed for 3D over 1D descriptors, on the generative models in chapter 5.

Chapters 3 and 4 both attempt to address the hurdles around mechanical properties of gels. Again, the isomerism debate is pertinent here as any difference in the gel formation caused by isomerism will carry forward into the gels mechanical properties. The models in chapter 4 would benefit from further training with additional data points which would hopefully result from the generative models created in chapter 5. These new models would have lower uncertainty as the model has a greater understanding of the underlying relationship between molecular structure and rheology.

Chapter 5 is the culmination of the previous chapters and lays the foundation of an approach that have the potential to accelerate and diversify the library of known gelators. The approach could act as a

feedback loop, with the results of the CReM and RNN models being experimentally validated and being used to update the classification and regression models.

For this to occur, improvements must focus on the generation of novel molecules and address the concerns of chapter 5. Namely, the diversity of the generated molecules needs to increase by either using a more diverse library as the training set for the RNN model or by introducing a temperature parameter that governs how conservative the model is during sampling.

Linking to the limitations of the models in chapter 4, any gels successfully synthesised and whose rheology is experimentally validated should be introduced as training set points to update both classification and regression models. If the molecules are chemically dissimilar to the molecules in the original training sets, this has the benefit of making the chemical space in which the models are applicable larger and allows for a further expansion of the gel chemical space.

Overall, this thesis has created a workflow that could be harnessed to accelerate the discovery of new and exciting low molecular weight gels to allow these materials to be rapidly discovered. Any minimisation of wasted time in the traditional trial and error discovery of novel low molecular weight gels will accelerate the realisation of the potential of these materials.

5.6 References

- 1 J. K. Gupta, D. J. Adams and N. G. Berry, *Chem. Sci.*, 2016, **7**, 4713–4719.
- 2 E. R. Draper and D. J. Adams, *Chem*, 2017, **3**, 390–410.
- 3 J. Meyers, B. Fabian and N. Brown, *Drug Discov. Today*, 2021, **26**, 2707–2715.
- 4 S. Awhida, E. R. Draper, T. O. McDonald and D. J. Adams, *J. Colloid Interface Sci.*, 2015, **455**, 24–31.
- 5 D. J. Adams, M. F. Butler, W. J. Frith, M. Kirkland, L. Mullen and P. Sanderson, *Soft Matter*, 2009, **5**, 1856–1862.
- 6 D. Whitley, *Stat. Comput.*, 1994, **4**, 65–85.
- 7 K. Jong, *Mach. Learn.*, 1988, **3**, 121–138.
- 8 O. Kramer, *Genetic Algorithm Essentials*, 2017, vol. 679.
- 9 A. Hassanat, K. Almohammadi, E. Abunawas, A. Hammouri and V. B. Surya Prasath, 2019, **10**, 390.
- 10 L. David, A. Thakkar, R. Mercado and O. Engkvist, *J. Cheminform.*, 2020, **12**.
- 11 A. I. Globus, J. Lawton and T. Wipke, *Nanotechnology*, 1999, **10**, 290–299.
- 12 D. H. Smith, R. E. Carhart and R. Venkataraghavan, *J. Chem. Inf. Comput. Sci.*, 1985, **25**, 64–73.
- 13 J. H. Jensen, *Chem. Sci.*, 2019, **10**, 3567–3572.
- 14 J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad and R. G. Coleman, *J. Chem. Inf. Model.*, 2012, **52**, 1757–1768.
- 15 P. Polishchuk, *J. Cheminform.*, 2020, **12**, 1–18.
- 16 A. Gaulton, A. Hersey, M. -I Nowotka, A. Patrícia Bento, J. Chambers, D. Mendez, P. Mutowo, F. Atkinson, L. J. Bellis, E. Cibri an-Uhalte, M. Davies, N. Dedman, A. Karlsson, M. Paula Magariños Magari, J. P. Overington, G. Papadatos, I. Smit and A. R. Leach, *Nucleic Acids Res.*, 2016, **45**, 945–954.
- 17 Z. R. Yang and Z. Yang, in *Comprehensive Biomedical Physics*, 2014, vol. 6, pp. 1–17.
- 18 M. Schuster and K. K. Paliwal, *IEEE Trans. Signal Process.*, 1997, **45**, 2673–2681.

- 19 H. Salehinejad, S. Sankar, J. Barfett, E. Colak and S. Valaee, 2017, 1–21.
- 20 K. G. Kanagachidambaresann, *Programming with TensorFlow*, Springer, 2021.
- 21 K. M. Tarwani and S. Edem, *Int. J. Eng. Trends Technol.*, 2017, **48**, 301–304.
- 22 A. Hermanto, T. B. Adji and N. A. Setiawan, *Proc. - 2015 Int. Conf. Sci. Inf. Technol. Big Data Spectr. Futur. Inf. Econ. ICSITech 2015*, 2016, 132–136.
- 23 K. Goel, R. Vohra and J. K. Sahoo, *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 2014, **8681 LNCS**, 217–224.
- 24 S. Hochreiter, *Int. J. Uncertainty, Fuzziness Knowledge-Based Syst.*, 1998, **6**, 107–116.
- 25 S. Hochreiter and J. Schmidhuber, *Neural Comput.*, 1997, **9**, 1735–1780.
- 26 K. Cho, B. van Merriënboer, D. Bahdanau and Y. Bengio, in *Proceedings of SSST 2014 - 8th Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014, pp. 103–111.
- 27 J. Zhao, Y. Nie, S. Ni and X. Sun, *IEEE Access*, 2020, **8**, 46713–46722.
- 28 Y. Duan, Y. Lv and F. Y. Wang, *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, 2016, 1053–1058.
- 29 A. Pulver and S. Lyu, *Proc. Int. Jt. Conf. Neural Networks*, 2017, **2017-May**, 845–851.
- 30 G. Shen, Q. Tan, H. Zhang, P. Zeng and J. Xu, *Procedia Comput. Sci.*, 2018, **131**, 895–903.
- 31 D. Weininger, *J. Chem. Inf. Comput. Sci.*, 1988, **28**, 31–36.
- 32 M. H. S. S. Segler, T. Kogej, C. Tyrchan and M. P. Waller, *ACS Cent. Sci.*, 2017, **4**, 120–131.
- 33 E. J. Bjerrum and R. Threlfall, *arXiv*, 2017, *arXiv:1705.04612*.
<https://arxiv.org/abs/1705.04612>
- 34 R. S. Sutton and A. G. Barto, *Learning*, 2012, **3**, 322.
- 35 M. Wiering and M. van Otterlo, *Reinforcement Learning*, 2012, vol. 12.
- 36 Deep Mind, AlphaGo | DeepMind, <https://deepmind.com/research/case-studies/alphago-the-story-so-far>, (accessed 7 March 2022).
- 37 M. Popova, O. Isayev and A. Tropsha, *Sci. Adv.*, 2017, **4**, 1–14.
- 38 Z. Zhou, S. Kearnes, L. Li, R. Zare and P. F. Riley, *Sci. Rep.*, 2018, **9**, 1–10.
- 39 A. Zhavoronkov, Y. Ivanenkov, A. Aliper, M. Veselov, V. Aladinskiy, A. V Aladinskaya, V.

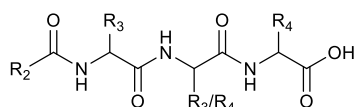
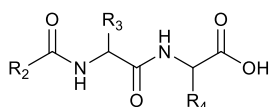
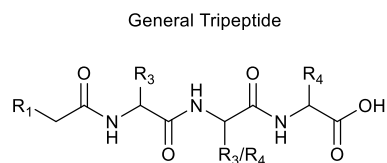
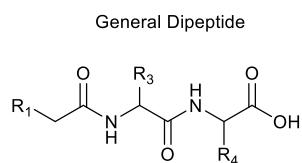
- Terentiev, D. Polykovskiy, M. Kuznetsov, A. Asadulaev, Y. Volkov, A. Zholus, S. Rim, A. Zhebrak, L. I. Minaeva, B. Zagribelnyy, L. H. Lee, R. Soll, D. Madge, L. Xing, T. Guo and A. Aspuru-Guzik, *Nat. Biotechnol.*, 2019, **37**, 1038–1040.
- 40 M. Olivecrona, T. Blaschke, O. Engkvist and H. Chen, *J. Cheminform.*, 2017, **9**, 1–14.
- 41 I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, .
- 42 A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta and A. A. Bharath, *IEEE Signal Process. Mag.*, 2018, **35**, 53–65.
- 43 Ł. Maziarka, A. Pocha, J. Kaczmarczyk, K. Rataj, T. Danel and M. Warchoń, , DOI:10.1186/s13321-019-0404-1.
- 44 J. Ho, A. Jain and P. Abbeel, .
- 45 A. Schneuing, Y. Du, C. Harris, A. Jamasb, I. Igashov, W. Du, T. Blundell, P. Lió, C. Gomes, M. Welling, M. Bronstein and B. Correia, .
- 46 G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*, CreateSpace, Scotts Valley, CA, 2009.
- 47 RDKit: Open-source cheminformatics. <https://www.rdkit.org>;
- 48 P. Ngatchou, A. Zarei and M. A. El-Sharkawi, *Proc. 13th Int. Conf. Intell. Syst. Appl. to Power Syst. ISAP'05*, 2005, **2005**, 84–91.
- 49 High Performance Computing - University of Liverpool, <https://www.liverpool.ac.uk/it/advanced-research-computing/facilities/high-performance-computing/>, (accessed 30 May 2022).
- 50 H. Y. Alhammadi and J. A. Romagnoli, *Comput. Aided Chem. Eng.*, 2004, **17**, 264–305.
- 51 S. Fletcher and M. Z. Isla, *Australas. J. Inf. Syst.*, 2018, **22**, 1–17.
- 52 D. E. Patterson, R. D. Cramer, A. M. Ferguson, R. D. Clark and L. E. Weinberger, *J. Med. Chem.*, 1996, **39**, 3049–3059.
- 53 G. Maggiora, M. Vogt, D. Stumpfe and J. Bajorath, *J. Med. Chem.*, 2014, **57**, 3186–3204.
- 54 P. Franco, N. Porta, J. D. Holliday and P. Willett, *J. Cheminform.*, , DOI:10.1186/1758-2946-6-5.
- 55 J. Arús-Pous, T. Blaschke, S. Ulander, J. Reymond, H. Chen and O. Engkvist, *J. Cheminform.*, ,

DOI:10.1186/s13321-019-0341-z.

- 56 P. Ertl and A. Schuffenhauer, *J. Cheminform.*, 2009, **1**, 1–11.
- 57 Y. Wang, J. Xiao, T. O. Suzek, J. Zhang, J. Wang and S. H. Bryant, *Nucleic Acids Res.*, 2009, **37**, 623–633.
- 58 M. H. S. Segler, T. Kogej, C. Tyrchan and M. Waller, *ACS Cent. Sci.*, 2017, **4**, 120–131.

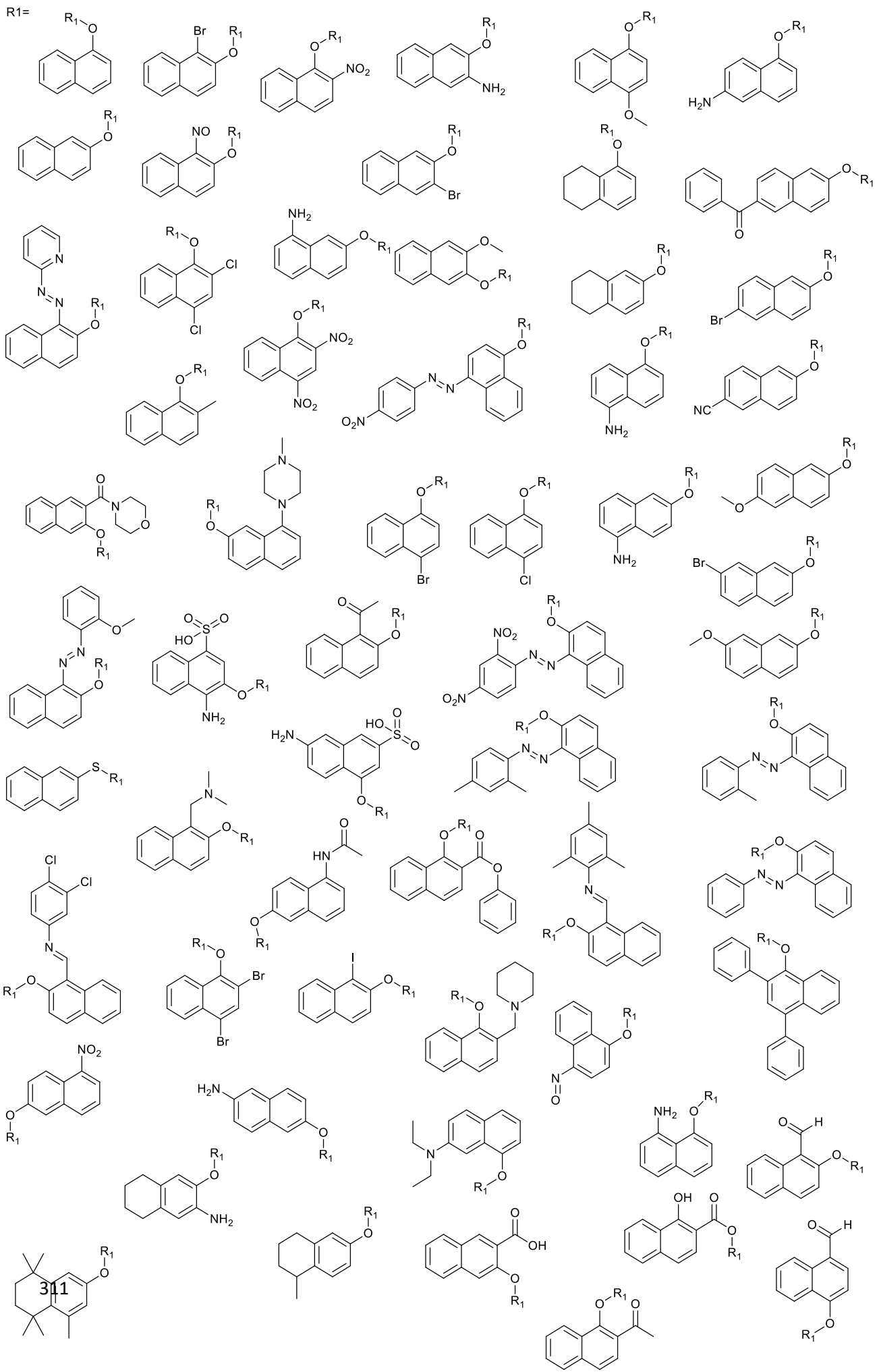
5.7 Appendix

5.7.1 Virtual Library Structures

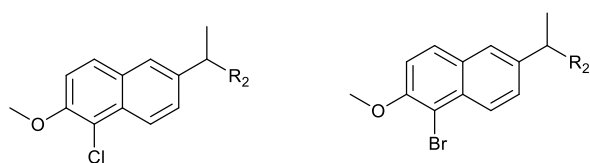
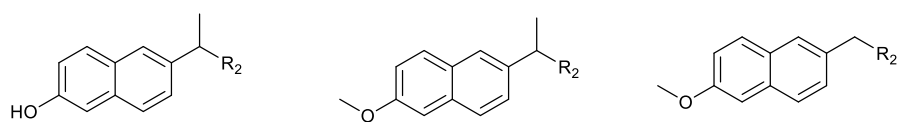
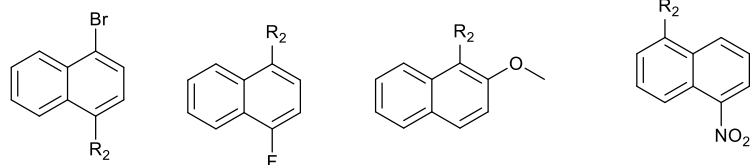
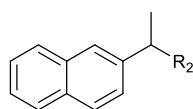
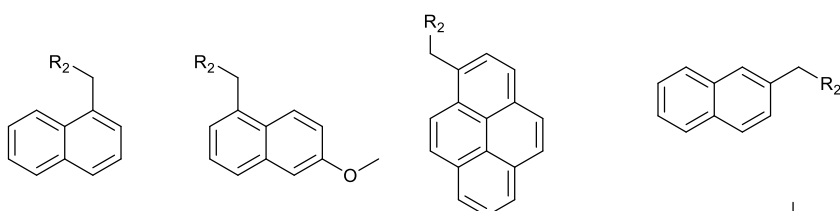
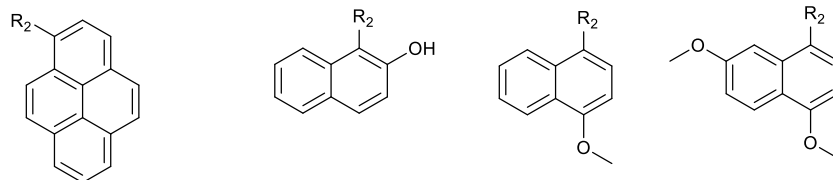
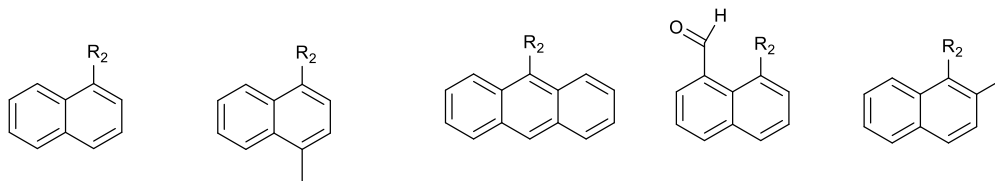


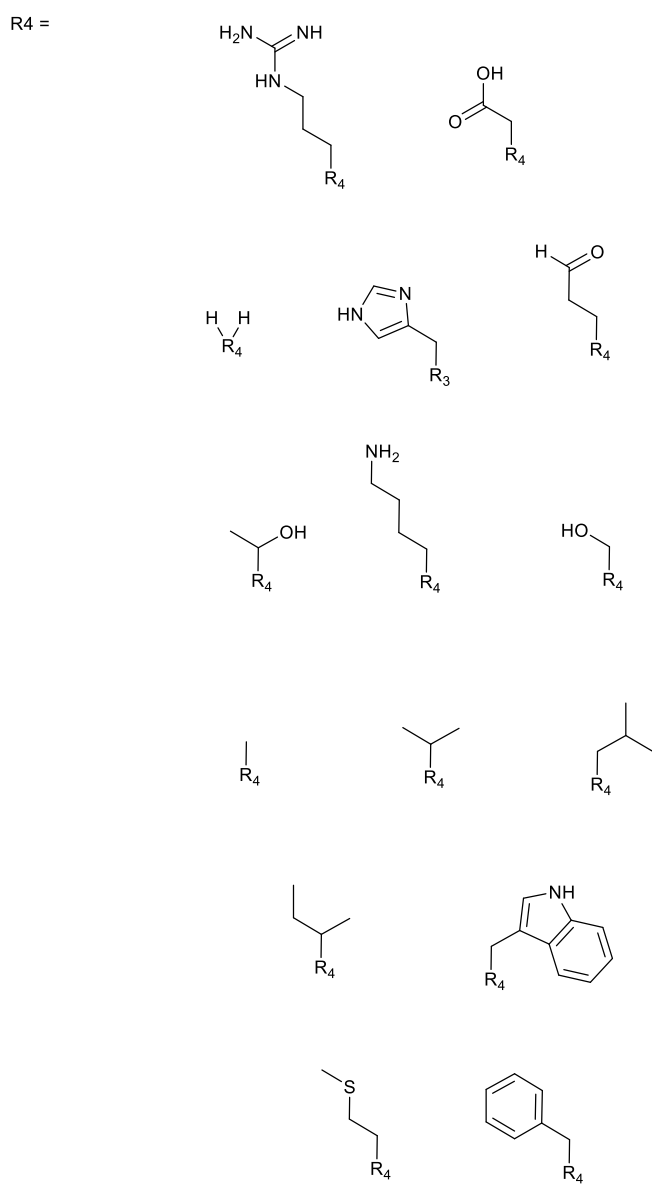
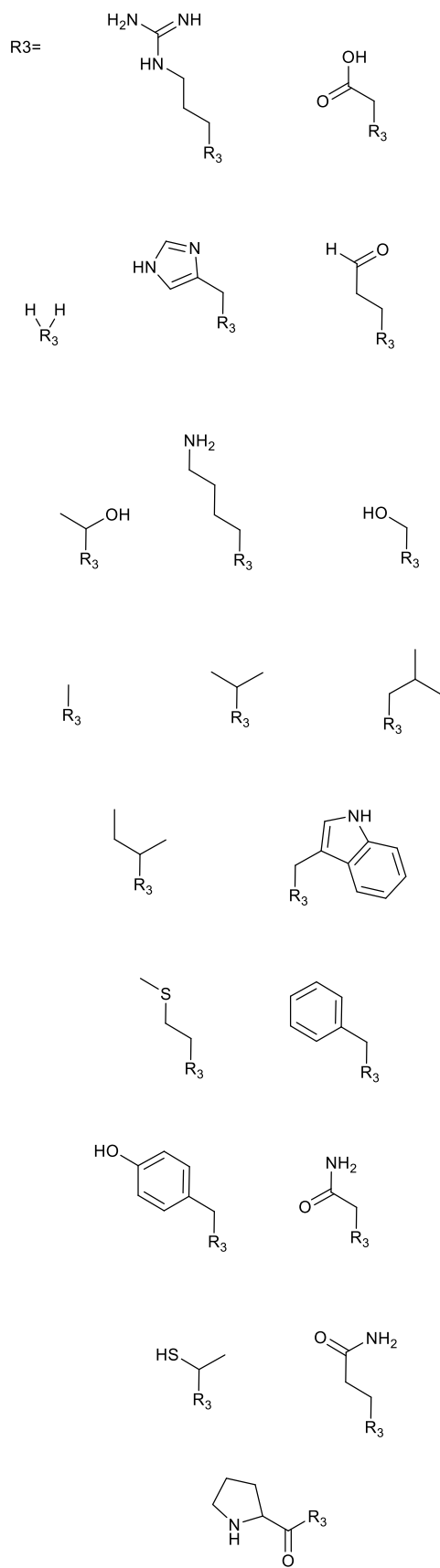
R3 does not have to equal R4, the tripeptide can have three different amino acids at once

R1=



R₂ =





List of Supporting Information

All files and data presented in this thesis can be found at:

<https://datacat.liverpool.ac.uk/id/eprint/1767>

It contains a zip file with the following structure:

Chapter 2 - Classification Models

For each descriptor set explored in the chapter the data in this folder is as follows:

- Contains Pipeline Protocol protocols for descriptor calculation
- Contains R scripts for data cleaning and model building (True and randomized models)
- Contains R objects containing the trained classifier models for each algorithm explored in Chapter 2. (True and randomized models)
- Contains R and Python scripts that carry out the SHAP model interpretation (all except fp_as_bits).

Chapter 3 - Non-Bayesian Regression

- Contains Pipeline Protocol protocols for descriptor calculation
- Contains R scripts for data cleaning and model building (True and randomized models)
- Contains R objects containing the trained regression models for each algorithm explored in Chapter 3 and each data set generated. (True and randomized models)
- Contains R and Python scripts that carry out the SHAP model interpretation.

Chapter 4 - Bayesian Regression

- Contains Python scripts for:
 - Descriptor calculation
 - Model Building (True and randomized models)
 - SHAP analysis.

Chapter 5 - DeNovo Generation

- Contains python scripts for the genetic algorithm
- Contains the necessary scripts to train a Recurrent Neural Network using the approach undertaken in Chapter 5 and also contains a range of trained models.
- Contains benchmarking work including:
 - Scripts used to run the benchmarking
 - Outputs for each run
 - Scripts used to analyze the benchmarks.

