Electronics and Computer Science

Faculty of Engineering and Physical Sciences

University of Southampton

**Oluwatosin Ogunribido**

2nd May 2023

# Using Machine Learning to Predict the Solubility of Drugs.

Project Supervisor: Dr Jo Grundy

Second Examiner: Dr Jize Yan

A project report submitted for the award of: **MEng Computer Science with Artificial Intelligence**

# Abstract

As diseases and infections become more complex, evolving into new variants, the drugs that are needed to fight them also become more complex and novel. As a result, faster and more efficient methods are required to improve the development of new drugs.

One of the more novel and innovative ways of developing these drugs is through the use of computational methods to quantitatively analyse the properties of the drugs. Previous research has proposed different models including classical machine learning models, decision trees, artificial neural networks, graph neural networks and many others. However, most of the optimal models require a semi-experimental approach to get the features for models except the graph neural network which uses only features of the atoms and bonds.

In this project, I developed a fully computational approach where the features are generated from an unsupervised clustering of the compounds, and the descriptors generated from SMILES to train the models. The models compared in this approach include Ridge Linear Regression, Support Vector Machine, Random Forest, Light Gradient Boosting and Attentive Feature Pyramids (AttentiveFP).

As a result of this analysis, it was found that the decision tree models performed better with larger datasets, i.e. the Light Gradient Boosting Method (LGBM) and Random Forest Method (RF) with an RMSE and MAE value of 0.915 and 0.658 for LGBM, and RMSE and MAE value of 1.158 and 0.764 for RF. These results show that the models were performing better when compared to other computational models, both within and outside the drug discovery industry. Although it does not predict the result to the utmost precision, these models provide a good approximation for early-stage decision-making, also, more information can be extracted.

# Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

**I have acknowledged all sources, and identified any content taken from elsewhere.**

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

**I have not used any resources produced by anyone else.**

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

**I did all the work myself, or with my allocated group, and have not helped anyone else.**

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced. **The material in the report is genuine, and I have included all my data/code/designs.**

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

**I have not submitted any part of this work for another assessment.**

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

# Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr Jo Grundy, for their guidance and support throughout the completion of this thesis. I am grateful for their patience, encouragement, and willingness to answer my questions. I would also like to thank my second examiner, Dr Jize Yan, for their helpful comments and suggestions.

I would also like to thank my family and friends for their support and encouragement. I am grateful for their patience and understanding during the long hours I spent working on my Individual Project.

Finally, I would like to thank the University of Southampton for providing me with the opportunity to study and conduct research. I am grateful for the resources and support that I received during my time at the university.

# Contents

# Chapter 1

# Introduction

There has been an increased demand to produce effective drugs in a shorter period to reduce the cost of research and improve accessibility. This was famously seen during the pandemic when there was a race to develop the COVID-19 vaccine[1].

One factor that helped to reduce the time of research was the increased use of technology in the pharmaceutical industry to predict and model the chemical and physical properties of drugs. This helped particularly in eliminating the most unfeasible compounds, leaving a smaller subset of substances. These substances can then be used for further experimental research and trials, which eventually lead to mass production.

Many studies have found that there are specific physical and chemical properties that influence the viability of a drug substance. These properties include toxicity, solubility, acidity/alkalinity and many others[2]. Solubility specifically is crucial as it affects the rate of absorption of a drug. Additionally, it could also help to determine how it is administered to people, e.g. through a tablet or an injection[3]. Predicting the solubility of compounds when discovering new substances has now become essential in discovery due to these reasons.

Many methods have been developed to improve the accuracy of these predictions, mostly through the use of mathematical models. However, recently there has been further research into the use of machine learning and deep learning models to improve these mathematical models such that they are more specific, accurate and efficient.

The core aim of this project is to build a computational model which utilizes machine learning to predict the solubility of chemical compounds, including drugs. This model of this project will be developed by comparing the current models implemented to predict the solubility of chemical compounds and finding their limitations. By modifying the models and using additional features generated from unsupervised clustering, an improved model can be devised such that it improves the performance based on the metrics provided.

These models will include machine learning models such as regression and ensemble learning, and deep learning models such as graphical convolutional networks. Additionally, the data will also be classified to observe if data clustering affects the type of model used and improves the score.

# Chapter 2

# Background Research

## 2.1 Solubility

The solubility of a substance (measured in mol/L or g/L) can be defined as the degree to which the substance dissolves in a solvent to form a homogeneous solution in which its atoms or ions are uniformly dispersed[4].

Solubility is measured relative to the solvent and can differ in different solvents. As a result, most substances are commonly measured using one solvent, which enables the comparison of different substances with respect to their solubility[5]. In most cases, the solvent used is water. However, this changes depending on the goal of the project and the environment in which the substances are commonly used.

The solubility of drugs can be affected by different factors. Many of these factors are not specific to compounds used in drug discovery. These factors include temperature, concentration of the substance, pressure, and polarity. Furthermore, there are properties which are specific due to the environment in which the substance is being dissolved[6] e.g. addition of electrolytes or precipitation which can increase or reduce solubility. These properties could also lead to complex ion formation, which increases the solubility of the compound.[6].

In general, there 7 are classes of substances based on solubility which are classified regardless of the solvent it is dissolved in. These classifications are based on the ratio of solute to solvent and range from being very soluble to practically insoluble. This table was developed by The United States Pharmacopoeia and the British Pharmacopoeia.[3] However, the Biopharmaceutics Classification System (BCS) is used by the United States Food and Drug Administration to classify the intestinal solubility of drugs where the most soluble is dissolved using at most 250ml of liquid media with a pH range of 1-7.5. In total, there are 4 classes based on permeability and solubility.[3]

## 2.2 Experimental and Mathematical Approach

Experimentation is the most accurate way of measuring solubility. This is typically done by gradually dissolving the solute into the solvent until it does not entirely dissolve. This can be affected by many factors, including temperature and pH of the

Table 1

USP and BP solubility criteria.

| Descriptive term | Part of solvent required per part of solute |
|---|---|
| Very soluble | Less than 1 |
| Freely soluble | From 1 to 10 |
| Soluble | From 10 to 30 |
| Sparingly soluble | From 30 to 100 |
| Slightly soluble | From 100 to 1000 |
| Very slightly soluble | From 1000 to 10,000 |
| Practically insoluble | 10,000 and over |

Figure 2.1: Classification of Substances[3]

**Table II.** *In Vitro–in Vivo* (IVIV) Correlation Expectations for Immediate Release Products Based on Biopharmaceutics Class

| Class | Solubility | Permeability | IVIV Correlation Expectation* |
|---|---|---|---|
| I | High | High | IVIV correlation if dissolution rate is slower that gastric emptying rate, otherwise limited or no correlation. |
| II | Low | High | IVIV correlation expected if *in vitro* dissolution rate is similar to *in vivo* dissolution rate, unless dose is very high (see discussion). |
| III | High | Low | Absorption (permeability) is rate determining and limited or no IVIV correlation with dissolution rate. |
| IV | Low | Low | Limited or no IVIV correlation expected |

* A limited correlation means that the dissolution rate while not rate controlling may be similar to the absorption rate and the extent of correlation will depend on the relative rates.

Figure 2.2: Classification of Drugs according to permeability and solubility [3]

solution[7]. Knowing these factors is useful as they can be changed to measure their impact on solubility, but the method is not cost-effective due to the required time and resources.

A slightly faster but less accurate approach involves a mathematical approach. This calculation can be done using thermodynamics by modelling the thermodynamic features of a compound, including free energy and lattice energy [8]. This method although more efficient still relies on some experimental data, as a result, it is still quite expensive.

Due to the cost, computational models have also been developed to improve the process in terms of performance and efficiency.

## 2.3 Computational Approach

Multiple computational methods are used in research and industries to predict the solubility of both old and new compounds. These methods vary in different stages of the process, which can be generally categorised into three stages.

### 2.3.1 Choosing the Dataset

The first stage involves the selection of data. Generally, most studies curate their training dataset based on analysed chemical datasets from public databases, which allows them to measure the performance of their model. Also, there are studies where novel chemical compounds were added to the datasets to improve the diversity of data seen by the model. This helped to optimise the importance of the features to the solubility measures[9].

It is important, when curating the data, to consider the categories of compounds and how these categories performed with different solvents. As a result, some studies were based on a singular solvent, while other studies compared model predictions on different solvents, for example, ethanol which is more likely to dissolve organic compounds[10].

Another distinction, made with regard to the data, was the choice of data for the test dataset. Several studies chose to use different datasets for testing and validation. This ensured the model performance was being tested based on its ability to see patterns in data, rather than its ability to recall data it had seen before.

### 2.3.2 Preprocessing

The second step involved processing the data and selecting or generating features which will be weighted to predict the solubility.

Due to the merging of multiple datasets to create the training and test data, the features of the data contained a lot of null values as these datasets do not measure the same features. To standardise this, packages have been developed that generate descriptors to show Quantitative Structure-Property Relationships. These relationships link their physical properties to chemical properties numerically, allowing them to be used in calculations.[11]. As a result, it has been used frequently in the drug industry for analysis. Examples of packages used to do this include rdKit [12] and modred[13].

These descriptors can be calculated from the one-dimensional structure such as the Simplified Molecular-Input Line-Entry System (SMILES) of the compound, the two-dimensional structure which is normally represented using a graph of atoms and bonds and the three-dimensional space of the compound where each compound is

represented using x, y, z coordinates. However, depending on the complexity of the model, it was more likely that only two-dimensional features were used. In total, most packages produce approximately 150-200 two-dimensional features and approximately 10-20 three-dimensional features[14].

Another set of features that can be generated is fingerprints. These are values represented as bit vectors that uniquely identify a compound based on its structural composition[9]. These are helpful for providing information to the model about chemical space mapping.

These features are generated from the chemical compound Simplified Molecular-Input Line-Entry System (SMILES). This describes the chemical composition of compounds using ASCII strings. [15]. These SMILES have enabled different algorithms to be developed to extract features based on the atom and bond values, which are represented in the SMILES using the alphabetic symbols of the elements and brackets for the bonds.

SMILES can also be standardised during preprocessing to ensure that the dataset does not contain the same compound with different features. [16] This can be done using software packages including ChemAxon. [17]

Once the features had been generated, they were normalised using methods including min-max scaling[16]. This ensures that features are on the same scale, reducing the importance of a singular feature size compared to other features.

## 2.4 Modelling using Classical Machine Learning and Deep Learning

Computational Models which can be trained to predict solubility can be grouped into multiple categories, which include classical regression models, decision trees and deep learning models. Before most models are trained, feature selection is done to improve efficiency and performance.

These features were selected manually or selected using methods such as recursive feature elimination and correlation. However, some models have embedded feature selection due to dropouts or low weights assigned to the features.

From this process, it was found that the feature with the most weight when calculating solubility for all models was logP also known as the Partition Coefficient. This measures the preference of solute between two compounds, usually an organic compound and water. Additionally, other features which affected different models include the molecular weight MWT and Distance Ring Index at 6. This result showed that many of these were related to the chemical properties of the compound, as the distance

ring index at 6 is higher in compounds with Benzene Rings which do not dissolve in water due to the strength of the carbon ring.

To measure performance, the $R^2$, The Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) were used.

### 2.4.1 Classical Regression Models

#### Linear Regression

The simplest model which can be used for predicting solubility is the Ordinary Least Squares Regression. These models performed relatively poorly as they were prone to both underfitting and overfitting, where the former is usually reduced by feature selection.

In many studies, overfitting could be traced back to features which have a very low or no correlation to the target variable LogS. As a result, these features were discarded or given low weights using LASSO regularisation. [11]

This helped to shrink the weights of irrelevant features to either 0 or a very small value, causing feature selection and improving the performance.

Another way of improving regression was to use a regression support vector machine that predicts the value using a plane rather than a line.[11] This helps to deal with incorrect outlier predictions as the data is projected to a higher dimensional space using the kernel trick where the outliers can be easily extracted.

The Lasso regression combined with a support vector machine also performed comparatively better than the ordinary least squared regression with an average $R^2$ value of 0.949 compared to $R^2$ value of 0.846 from the ordinary Least squared regression.[11]

### 2.4.2 Decision Trees

An alternative model which has been greatly improved over time to predict solubility is the Decision Tree Model. Although mentioned in many papers as a type of model to compare, there are relatively few papers which aim to find the best optimal Decision Trees model using ensemble methods. One type of decision trees that has shown better performance is the Random Forest, which was analysed and tuned using multiple variables that reduce overfitting[18],[19]. These variables were found using the grid search method which observed multiple combinations of parameters such as tree depth, number of features and minimum number of samples before split. From this method, it was found that the best parameters were 486 for the maximum number of features, 203 for the number of trees, 45 for the maximum depth, 4 for the minimum number of samples before the split and 2 for the minimum number of samples in a child leaf[18].

Table 1. Comparison of the regression methods.

| Method | Data | $R^2$ | RMSE | MAE |
|---|---|---|---|---|
| **OLS** | Training | 0.933 | 0.531 | 0.399 |
| | Test | 0.898 | 0.664 | 0.474 |
| | Validation | 0.856 | 0.780 | 0.617 |
| **Lasso** | Training | 0.868 | 0.744 | 0.582 |
| | Test | 0.869 | 0.752 | 0.572 |
| | Validation | 0.757 | 1.012 | 0.745 |
| **SVR** | Training | 0.958 | 0.418 | 0.265 |
| | Test | 0.898 | 0.663 | 0.417 |
| | Validation | 0.862 | 0.763 | 0.502 |
| **Lasso+SVR** | Training | 0.944 | 0.483 | 0.359 |
| | Test | 0.921 | 0.583 | 0.429 |
| | Validation | 0.949 | 0.463 | 0.364 |

Figure 2.3: Performance of linear regression and regularised model

From: Prediction of small-molecule compound solubility in organic solvents by machine learning algorithms

| Machine learning algorithms | Unseen solutes | | | Seen solutes, but not seen the same combination of solutes and solvents | | | Seen the same combination of solute and solvent, but at different temperatures | | |
|---|---|---|---|---|---|---|---|---|---|
| | MAE | MSE | $R^2$ | MAE | MSE | $R^2$ | MAE | MSE | $R^2$ |
| PLS | 0.8467 | 1.2267 | 0.1707 | 0.6155 | 0.7112 | 0.6538 | 0.4408 | 0.3758 | 0.7781 |
| Ridge regression | 0.5909 | 0.8410 | 0.4314 | 0.6128 | 0.7123 | 0.6532 | 0.4403 | 0.3762 | 0.7779 |
| kNN | 1.2722 | 2.6338 | -0.7806 | 0.8734 | 1.4487 | 0.2947 | 0.1919 | 0.1360 | 0.9197 |
| DT | 0.9560 | 1.5480 | -0.0466 | 0.6946 | 0.9234 | 0.5505 | 0.2360 | 0.1446 | 0.9146 |
| ET | 0.7860 | 1.0640 | 0.2807 | 0.5514 | 0.6021 | 0.7069 | 0.1982 | 0.1124 | 0.9337 |
| RF | 0.7655 | 1.0518 | 0.2889 | 0.5436 | 0.5724 | 0.7213 | 0.1624 | 0.0939 | 0.9446 |
| SVM | 0.7959 | 1.3027 | 0.1193 | 0.5677 | 0.6375 | 0.6896 | 0.1216 | 0.0827 | 0.9512 |
| DNN | 0.8494 | 1.1523 | 0.2210 | 0.4897 | 0.5073 | 0.7530 | 0.1300 | 0.0762 | 0.9550 |
| lightGBM | 0.5968 | 0.7511 | 0.4922 | 0.4729 | 0.4243 | 0.7934 | 0.1307 | 0.0788 | 0.9535 |

Figure 2.4: Performance of ensemble model compared to nearest neighbour (KNN) and Deep Learning (DNN). From [18]

### Light Gradient Boosting

Although not frequently the highlight of many studies, the Light Gradient Boosting Ensemble method has shown to work very well in predicting the solubility of chemical compounds when compared to other machine learning methods[10]. This is mostly due to its use of gradient descent to extend the decision tree leaf-wise instead of nodes.

In many studies, this model was optimised similarly to other ensemble methods using features such as the learning rate of the gradient descent, number of leaves, maximum depth of the tree and many others. These parameters help to reduce overfitting by controlling how much the decision tree tries to fit a sample based on its features. This method found that the best feature combination was having a maximum depth of 20, a maximum number of leaves of 43, subsampling rate of 0.7

and the col_sampling_tree/maximum number of features of 0.7. This resulted in an $R^2$ value of 0.8575 and an RMSE value of 0.7785.

### 2.4.3 Deep Learning

The main types of neural networks used in studies to predict include Graph Neural Networks, Artificial Neural Networks and Deep Neural Networks [9], [20]–[22]. These models are usually formed by combining multiple machine-learning models in layers. Additionally, due to the size of the models, Deep Learning Models require a large amount of data when compared to other machine learning models such that they are usually trained with millions of features.

There are several ways to implement deep neural networks; however, one of the most common approaches involves the use of SMILES strings. This involves encoding the SMILES string to minimise reconstruction error. This encoded string is passed through a latent space, where the task is performed using multiple convolutional layers before being decoded to get the output. Encoding is done usually using one hot encoder. [23].

Although this method works well in predicting solubility, it also ignores the spatial information of the chemical compound which provides information about the interactions in the chemical space. Spatial information can be very helpful in distinguishing the solubility of the substance based on the bonds formed within atoms, i.e. the more bonds exist between the atoms, the higher the solubility in water.

One way of preserving this information is to use Graph Neural Networks. These networks are an extension of artificial neural networks, which aims to represent the data and its features as a graph showing their relationships. It has been commonly used as transformers for Natural Language Processing and the form of Convolutional Neural Networks for Computer Vision. Additionally, this model has worked relatively well in molecular biology and Chemistry[24].

In molecular property prediction, Graph Neural Networks have provided a new way of representing compounds such that the spatial data is preserved. This provides information about the relationships between bonds and atoms, which is otherwise lost using SMILES. To implement this model, each chemical compound is transformed into a graph embedding known as the molecular graph, where each node stores the atom's features and the edge stores the bond features. These nodes and edges are connected using index pairs, resulting in an undirected graph embedding which can be passed to the neural network layers.

A broad spectrum of permutation equivalent layers can be used to update the features and weights that are used to predict molecular properties. However, the models with the best performance on a range of tasks use Self Attention Mechanism or Message

Passing Neural Network where in the case of solubility, the former had an RMSE value of 0.503 and the latter had an RMSE value of 0.587 when optimised on the ESOL dataset.

In both methods, the molecular graphs were derived from the SMILES, by encoding the atom and bond features using one hot encoding where each atom feature was matched to a list of possible values for each feature. However, the method of training the model differs.

**Table 1. Initial Atomic and Bond Features**

| atom feature | size | description |
|---|---|---|
| atom symbol | 16 | [B, C, N, O, F, Si, P, S, Cl, As, Se, Br, Te, I, At, metal] (one-hot) |
| degree | 6 | number of covalent bonds [0,1,2,3,4,5] (one-hot) |
| formal charge | 1 | electrical charge (integer) |
| radical electrons | 1 | number of radical electrons (integer) |
| hybridization | 6 | [sp, $sp^2$, $sp^3$, $sp^3d$, $sp^3d^2$, other] (one-hot) |
| aromaticity | 1 | whether the atom is part of an aromatic system [0/1] (one-hot) |
| hydrogens | 5 | number of connected hydrogens [0,1,2,3,4] (one-hot) |
| chirality | 1 | whether the atom is chiral center [0/1] (one-hot) |
| chirality type | 2 | [R, S] (one-hot) |
| **bond feature** | **size** | **description** |
| bond type | 4 | [single, double, triple, aromatic] (one-hot) |
| conjugation | 1 | whether the bond is conjugated [0/1] (one-hot) |
| ring | 1 | whether the bond is in ring [0/1] (one-hot) |
| stereo | 4 | [StereoNone, StereoAny, StereoZ, StereoE] (one-hot) |

Figure 2.5: Feature Encoded to Form Molecular Graph in AttentiveFP [25]

In the Message Passing Neural Network, a recurrent neural network is used to learn about the data. This uses a recursive message passing which can be associated with nodes (atoms). Alternatively, directed edges (bonds) are used to avoid noise[26]. These messages are aggregated and used to update the feature weights, used to re-train the weights again or predict the value.

The best model for molecular property prediction is known as AttentiveFP. This uses the graph attention mechanism. This works by iteratively assigning weights to different features of the atoms and bonds in the graph until all or most of the graphs have been updated. This is done through convolutional layers, which results in different contributions of the features. Due to this update based on the feature weights, the importance of each bond and atom changes and contributes differently to the contribution.

In both studies, the models were implemented using PyTorch where the models

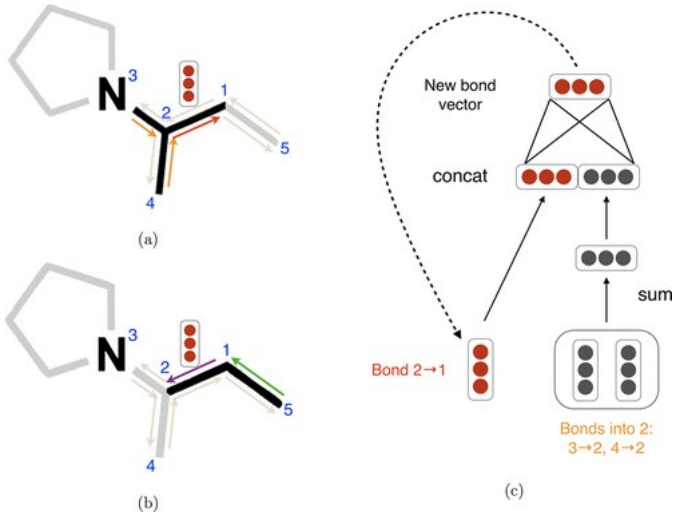



Figure 2.6: Image Illustrating the Passing of Messages Between Bonds [26]

use multiple parameters i.e. fingerprint dimension, L2 weight decay learning rate, number of message-passing steps and dropout rate to optimise performance. As a result of this, Bayesian optimisation was used to find the most optimal set of hyperparameters due to efficiency and reduced time consumption. Additionally, the Adam optimizer was used when calculating the gradient for the best performance.

To summarise, it can be observed that there are research gaps which need to be addressed to make current solutions more efficient and useful in the drug industry. One of the gaps observed during the research is the performance of different models with different groups of chemical compounds in reference to their structure. This is commonly seen when observing the results from algorithms and models trained using different solvents [22] or compounds with different functional groups[10].

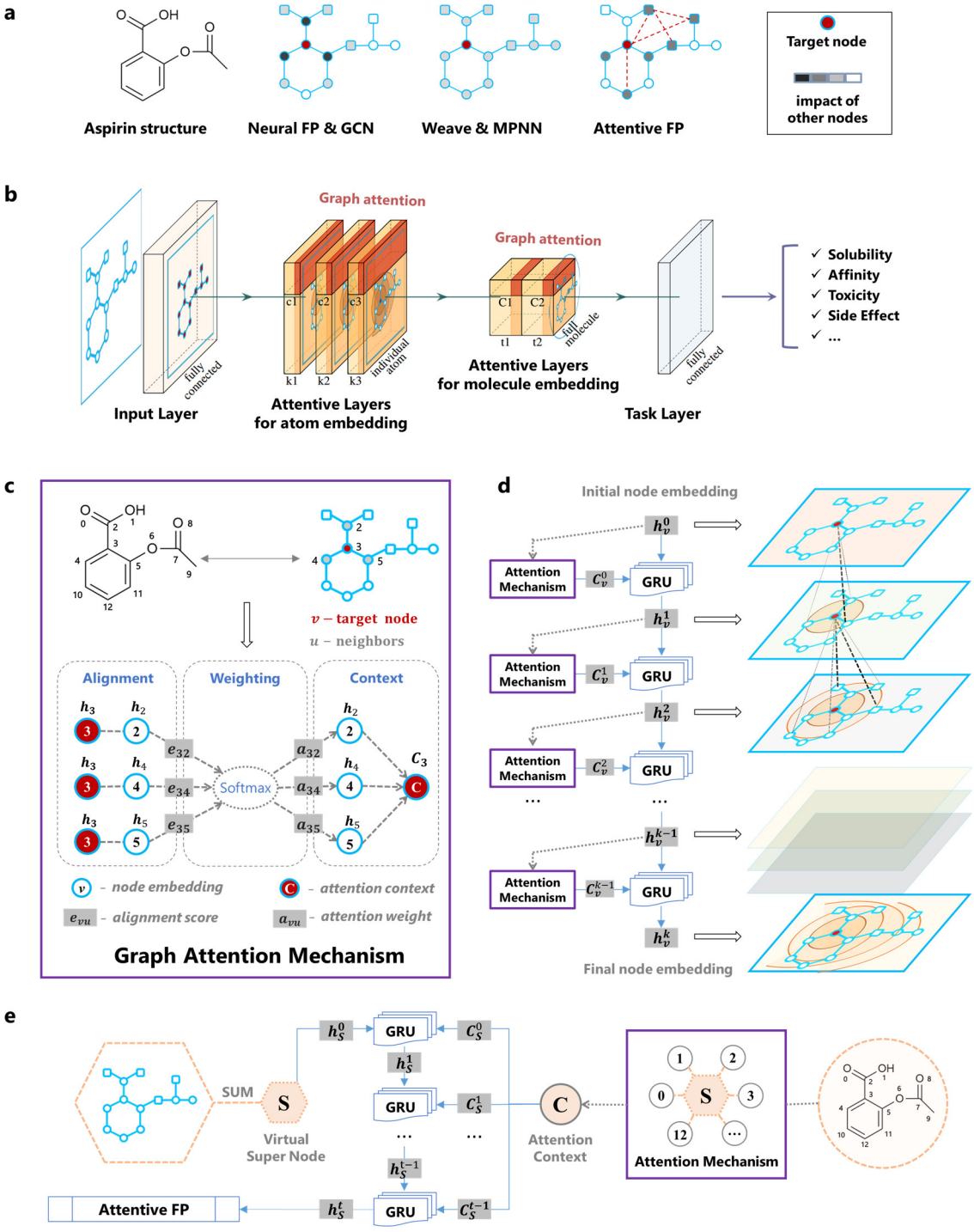



Figure 2.7: Structure of the network in AttentiveFP [25]

# Chapter 3

# Proposed Design

Due to the gaps in the research mentioned in the background research, questions were developed to try to fill the gaps in research about practicality and optimisation. These questions formed the basis of the project, where the aim became to find a possible model that performed well and was efficient enough to be used as a possible industrial solution. These questions include:

- Will classifying the data based on chemistry research improve the model's performance?
- Are the current models reproducible if all other environmental characteristics of the experiment-related solubility are ignored?
- How can the current models be used to aid research without compromising safety?
- Can the current model be used in a way that is efficient to reduce cost?

## 3.1 Requirements

The questions mentioned above were used to form the requirements of the proposed system, such that the results of the system could be interpreted to answer them. These requirements are defined in Table 3.1

## 3.2 System Design

The requirement mentioned were used to formulate a machine learning system which compared two classical regression models, two decision tree models and a deep learning model. Additionally, it also compared the results of using classification-informed data and not using classification-informed data.

The system involved multiple steps, which is shown in the diagram in figure 3.1 . These steps are similar to other machine learning pipelines where data is collected, their features are extracted and used to predict the target value which is compared to the actual target.

| Question | System Requirement |
|---|---|
| Are the current models reproducible if all other environmental characteristics of the experiment to discard solubility are ignored? | The system must be able to accurately predict the solubility of a compound regardless of the temperature, pressure, or state of a chemical compound |
| Will classifying the data based on chemistry research improve the performance of the model, | The system should be able to classify chemical compounds such that the result can be used as information to predict the solubility of the compound. |
| How can the current models be used to aid research without compromising safety, | The system must be able to compare the current result of the compound with the actual solubility of the compound i.e. Supervised learning. |
| Can the current model be used in a way that is efficient to reduce cost? | The system must have a low computational and storage cost where improving the performance of the model does not significantly increase the computational cost of running it. Additionally, the system needs to show that the cost of running the model is less than the cost of running an experiment. |

Table 3.1: Table of Requirements

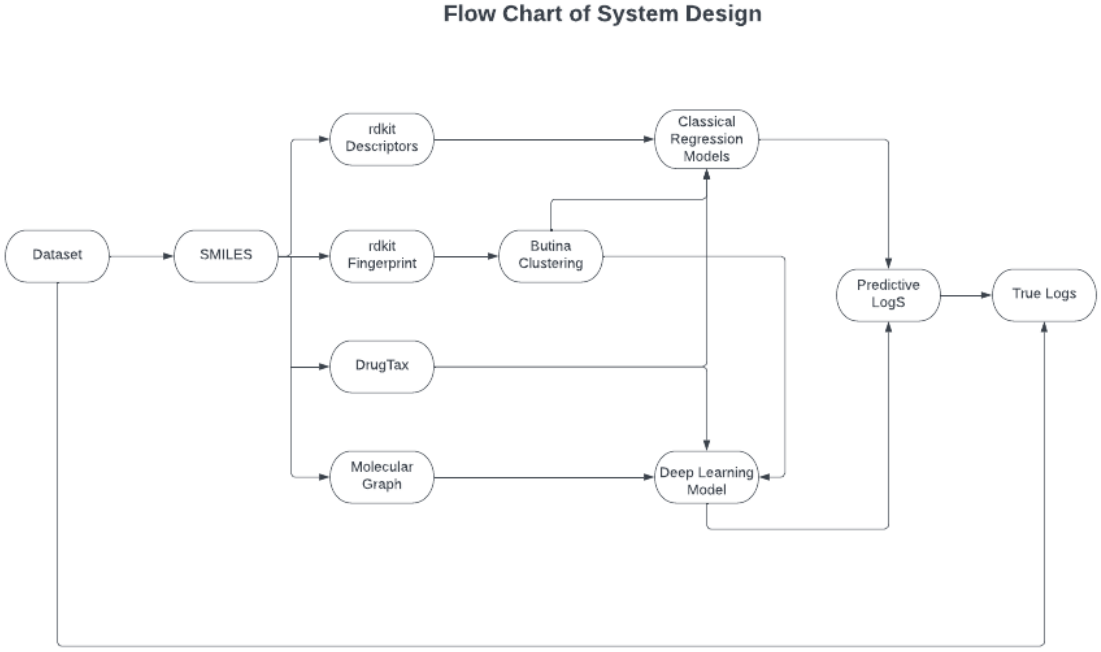



Figure 3.1: Proposed Design

## 3.3 Required Hardware and Software

To run the system, a personal computer was used which had a 16 GB RAM, intel i7 12th Generation CPU and 512 GB. However, this was with the exception of the deep learning model, which needed to be run on the GPU system due to its high computational intensity.

On both systems, a Conda environment was set up which used Python version 3.9 as the programming language. This enabled packages required for extracting features and predicting the data to be downloaded. For the packages and their version, see Appendix in Section 10.1

## 3.4 Evaluation Metrics

Like the studies researched, to measure the performance of the model. Two metrics were used:

- Root Mean Squared Error (RMSE): this shows the difference between the predicted value and true value, based on the standard deviation of the model.

  The RMSE value can be defined as:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(\frac{d_i - f_i}{\sigma_i}\right)^2} \tag{3.1}$$

- Mean Absolute Error(MAE): this finds the absolute difference between pairs of values, which in this case is the target logS and the predicted logS. The MAE value can be defined as :

$$MAE = (\frac{1}{n})\sum_{i=1}^{n}|y_i - x_i| \tag{3.2}$$

These metrics were chosen due to the variety of models analysed in this project. Additionally, the $R^2$ metric was also observed however it was not very helpful when looking at models such as the decision trees. As a result, it was discarded in this project as it could not be used to compare all the models.

# Chapter 4

# Data Extraction and Processing

The project's first stage involved extracting the data and deciding what features were required to extract other features. This stage also involved deciding on what could be used as the label in supervised learning for regression.

## 4.1 Data Extraction

Most studies use either one curated dataset or data from Chemistry specific dataset to ensure a variety of chemical compounds. However, in this project, the dataset has been curated by combining smaller datasets mostly made of organic compounds.

The datasets used in the curation comprised of the AQSolDB[27], ESOL[28], Osada et al[11], Deng et al[22] and Cui et al[29]. All these datasets were compiled with the sole purpose of predicting the solubility of compounds with Cui et al [29] focusing on novel compounds with experimental solubility.

Only logarithms of the experimental solubility in water and the SMILES were extracted to ensure uniform features and reduce underflow error. The experimental solubility allows for the modelling of solubility using a supervised learning approach where the dataset is divided into an 80:20 ratio for the training and test dataset.

The SMILES and solubility were added into a Pandas data frame. These SMILES were converted into canonical SMILES, allowing the removal of duplicate smiles and their associated solubility. After this, any SMILES not recognised by rdkit are also removed, as all the feature generation methods require recognised SMILES by the rdkit package. After cleaning, the dataset had a size of 14,888. Once the dataset was cleaned, the SMILES were used to generate other features, which differed depending on the type of model to be implemented. However, for all models, the results of the classification were added to the dataset.

The curated dataset logarithms of solubility can be approximated to a Gaussian Distribution which has a mean of -3.895 and a standard deviation of 1.465. This is illustrated by the graph shown figure 4.4.1

It is also worth noting that due to the use of multiple datasets and lack of a standardised process of experimentation, many features related to solubility such as temperature and pressure were discarded from the original datasets. It was also

| Name/Citation | Size |
|---|---|
| ESOL | 1144 |
| AqSOLDB | 9982 |
| Cui et al | 9943 |
| Deng et al | 1334 |
| **Total after cleaning** | **14888** |

Table 4.1: Datasets and their sizes

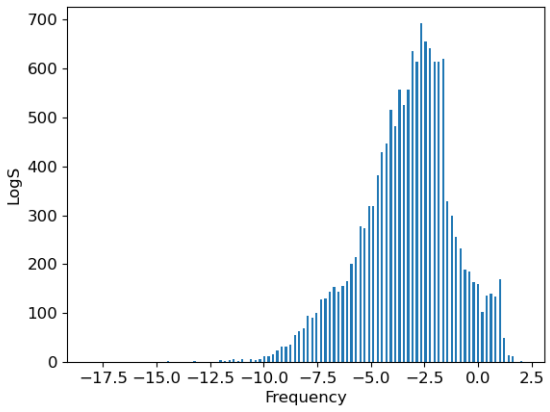



Figure 4.1: Distribution of the Logarithms of Solubility in the Curated Dataset

observed that many of these compounds occur in different states. One of these states at room temperature is gas, which cannot be used as a drug. However, it has been included in the dataset to act as noise to ensure that the models are not fitting the features of the training data but the patterns that it finds in the training data.

## 4.2 Feature Generation and Classification

More features need to be extracted using a software package to enable regression modelling. The compounds also need to be classified into clusters using unsupervised learning methods, whose results can be encoded into a vector. This vector is embedded into the feature matrix, which is used to train and evaluate the models.

## 4.3 Feature Extraction

To ensure homogeneity in our dataset when using the classical regression models and decision trees, physical features of the chemical compounds were used to develop input vectors for the models. This method of modelling is known as Quantitative Structure Analysis Modelling.

This involves using 2d and 3d descriptors of chemical compounds, represented numerically, to predict other chemical properties. In this project, the rdkit package was used to

generate the descriptors from SMILES, which resulted in 208 descriptors of each compound. For the full list of descriptors, please see Appendix in Section 10.1. Figure 4.2 shows how the different features are distributed from rdkit where many of the features are sparse and some have large ranges.

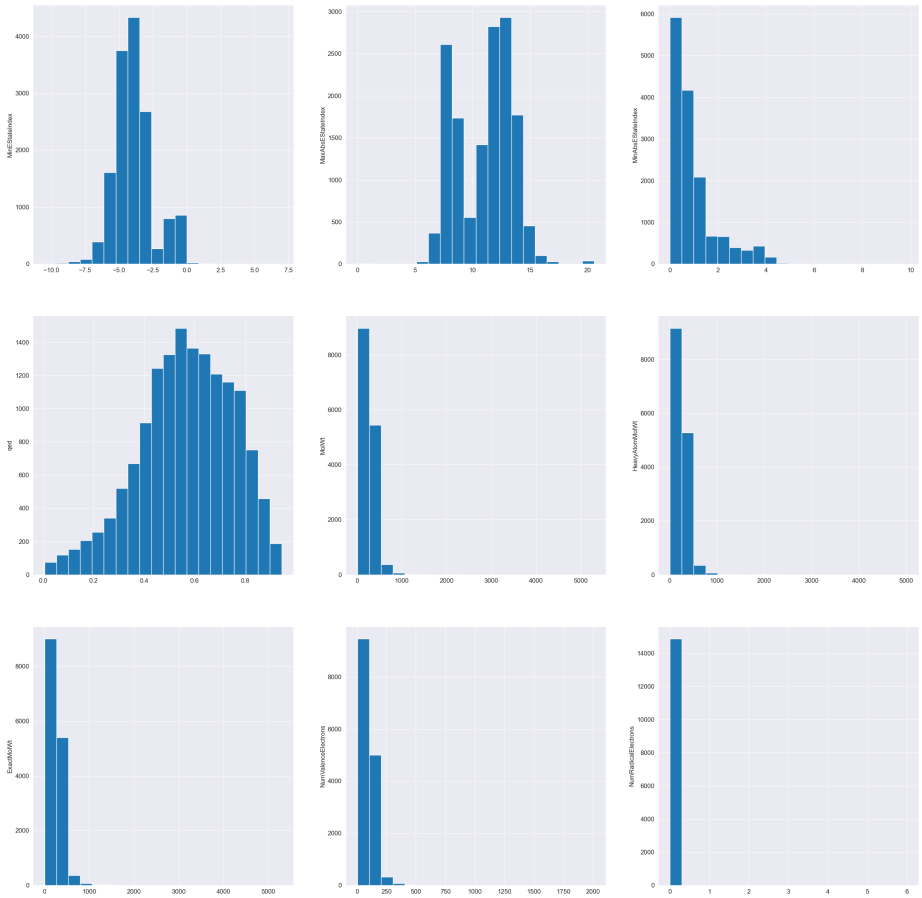


Figure 4.2: Distribution of Some Features Generated by rdkit

## 4.4 Classification Models

In Chemistry research, many of the physical and chemical properties have been derived by looking at the atomic and bond structure of compounds. Due to experiments and research, it has also been observed that it was possible to predict the structure and properties of a compound by looking at the neighbouring cluster of compounds with already known properties. As a result, it is reasonable to imply that given a set

of chemical compounds, by predicting the group or cluster that they belong to, more information about their solubility can be extracted. This information could also help the model understand the compound better, therefore improving its performance on prediction.

In this project, two classification models were implemented. These models include DrugTax[30] and Butina Clustering [31]. Both of these methods were chosen due to their stability, to ensure they gave similar results on different iterations.

### 4.4.1 Butina Clustering using Tanimoto Distance.

This method of classification developed by Darko Butina involves the use of daylight fingerprints to determine the cluster to which the group belongs. It is a flat-based clustering system which uses an algorithm similar to the K-means unsupervised learning algorithm[31]. This process involves three stages of implementation.

#### Fingerprint Generation

The first stage of clustering was the fingerprints of the chemical compounds. As defined by Wiktionary, Fingerprints are '*the unique combination of analytical results from a number of techniques that identifies a particular compound in a sample*" [32]. These fingerprints are developed by comparing the similarities and differences between compounds. As a result, they are often used when clustering chemical compounds into groups or identifying physical features.

Many fingerprints have been designed to help identify the patterns in chemical compounds. These fingerprints include the Atom-pair fingerprints, Morgan fingerprints, ECFP6 Fingerprints. However, for this project, only the rdKit fingerprint is used. This is a modified implementation of the Daylight Fingerprint[33]. It is a substructure fingerprint which is represented by a vector encoding of the atom types and bond types.

The rdkit fingerprints were generated from the rdkit package using their SMILES converted to rdkit Molecules. In most cases, these fingerprints can be inefficient to use, as each compound is represented by a bit vector of size 1024. As a result, the fingerprint is usually given a unique bit hash to improve efficiency. Nevertheless, for this project, the vectors were stored in the original state.

The next stage was identifying the possible centroids of clusters to which the chemical compound would belong to. This was done by finding the number of neighbours each molecule had in the set. The Tanimoto similarity distance between each compound pair needs to be calculated to do this. This is the ratio of common features of the fingerprints to all the features in the fingerprints. This distance is given by the

equation [31]:

$$Tanimoto = 1 - Similarity = \frac{c}{a + b - c} \tag{4.1}$$

Where:

**c** ⇒ intersection of the two fingerprints, i.e. the number of common features in compound A and compound B

**a** ⇒ number of features in compound A

**b** ⇒ number of features in compound B

This coefficient is compared to a threshold to find the possible centroids where a molecule is said to be neighbouring another molecule when it is below the threshold. Once calculated, this centroid is found by sorting the molecules in descending order based on their Tanimoto coefficient.

After annotating possible centroids, clusters are extracted based on exclusion spheres using the Tanimoto similarity index. Molecules meeting the clustering value are flagged, placed in a cluster, and removed from the list. A new centroid is selected from nearby molecules, repeatedly until all similar molecules are marked. The remaining compounds are placed in individual clusters.

This process usually results in multiple clusters, depending on the size of the data. However, the threshold needs to be optimised such that it is high enough where there are not many singleton clusters but low enough that only similar compounds exist in each cluster. An example of the resulting cluster is shown in Figure 4.3.

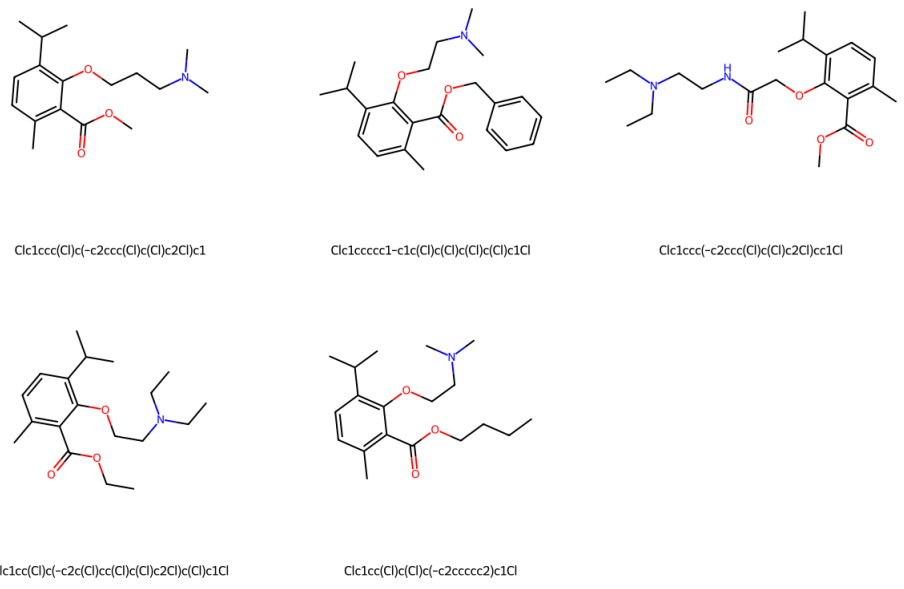



Figure 4.3: Five Compounds from the Same Cluster using Butina Clustering

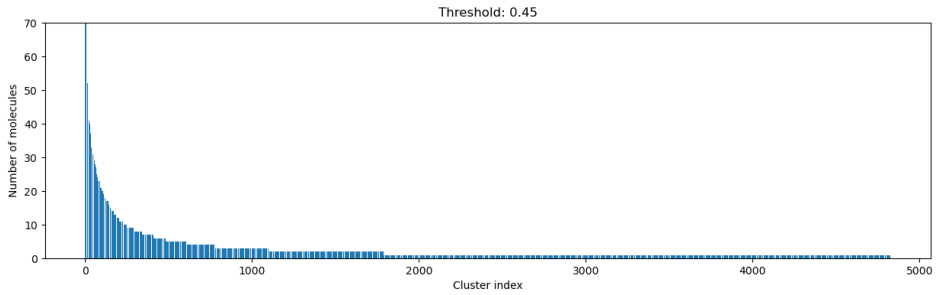



Figure 4.4: Distribution of cluster with a threshold of 0.45

**Implementation**

To find the threshold and clusters, the Tanimoto coefficient was stored in the matrix of the size of NxN where N is the number of compounds. This matrix had a computational cost of $O(n^2)$. Once calculated, the clusters were found using the rdkit implementation of Butina clustering, which required a cut-off value. To optimise this value, a grid search was performed for a value between 0 and 1 with a step size of 0.05. The best value was found by looking at the distribution histogram of cluster size. This is shown in the graph below where the optimal value had enough clusters where the compound had similar properties but small enough where there were fewer single compounds. The cluster indexes were extracted and then used as a feature of the regression model. In total, it had approximately, 4825 clusters with a threshold of 0.45. The distribution of this graph is shown in Graph .

### 4.4.2 DrugTax

Unlike the Butina clustering, this method uses a classification system similar to like ClassyFire [35] which aims to cluster compounds based on hierarchy. It is a 2-level hierarchy system where the first level divided the compounds into 2 classes: organic and inorganic compounds and the second divides the compounds into 31 superclasses which are based on their current kingdom where organic had 26 superclasses and inorganic had 5 superclasses. For more information about the superclasses and kingdoms, please see Appendix in Section 10.3.

This model is very helpful as it provides information based on the actual chemistry taxonomy of a compound. This allows regression modelling based on the relationship between the descriptors, similar to the relationships that are seen in experimental methods. This model is non-parametric, which means it can be used for both smaller and larger datasets with the same performance.

The kingdoms are detected based on the presence of carbon atoms, where all compounds with carbon atoms belong to the organic kingdom except carbon sulphides, oxides, and cyanides. Once classified, the superclasses are deduced based on the presence of

functional groups, for example, a compound is said to be an organosulphur if it has one carbon-sulphur bond. Due to this, a compound can have multiple superclasses which belong to the same kingdom.

### Implementation

To extract the superclasses and kingdom for the compound in the dataset, their smiles were passed using the drugTax package function, whose output gives the kingdom and list of superclasses. To ensure the superclasses and kingdom are independent and make the feature matrix more independent, it was one hot encoded. This resulted in a binary vector of the superclasses and kingdom, which was concatenated with the Butina Cluster Index.

# Chapter 5

# Linear Regression and Decision Trees

To fulfil the design requirements, five models including Linear Regression, Support Vector Machines, Random Forest Trees, Light Gradient Boosting and the Deep Learning Model called AttentiveFP were implemented with different combinations of the classification-informed data.

## 5.1 Linear Regression

In machine learning, one of the simplest ways to predict a value is using Ordinary Linear Regression. This model attempts to predict the value by fitting the data to a line which is defined by

$$\xi_i(\beta_0, \beta_1, \ldots, \beta_k) = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_k x_{ki} \tag{5.1}$$

In equation 5.1, $\beta$ represents the weight assigned to the features X with a bias of $\beta_0$ where $\xi_i(\beta_0, \beta_1, \ldots, \beta_k)$ is the predicted value of each output. This equation defines the output of the linear regression function when given input features.

Alternatively, it can also be defined by the normal equation which finds the feature matrix $\theta$ given the input matrix X and output matrix y with the aim of minimising the loss This is defined by:

$$\hat{\theta} = (X^{\mathrm{T}} X)^{-1} X^{\mathrm{T}} y \tag{5.2}$$

This is usually optimised using gradient descent where the gradient is calculated, and the weights are increased or reduced based on the value of the object function i.e.

$$\theta_j := \theta_j - \alpha \frac{\partial J}{\partial \theta_j} \tag{5.3}$$

where
$\Theta$ is the weight at index j of the model
$\alpha$ is the learning rate and
J is the cost function

### 5.1.1 Regularisation

Noise can cause models to overfit by falsely representing features. This is due to a lack of generalisation, where the model assigns higher weights to data to reduce loss. To deal with this in linear regression modelling, regularisation is used. This technique involves the use of penalties to minimise the error of the model. There are two major regularisation techniques which can be used; The Least Absolute Shrinkage and Selection Operator (LASSO)/L1 regularisation and Ridge/L2 regularisation.

Lasso regression penalises the features using a linear function which aims to penalise only high coefficients. This causes weight features to shrink such that they tend to towards zero, causing feature selection. On the other hand, the Ridge Regression uses a quadratic function with the aim of penalising the multi-collinearity within the features. Both of these methods aim to reduce the variance of features by simplifying the feature space, as illustrated in Figure 5.1

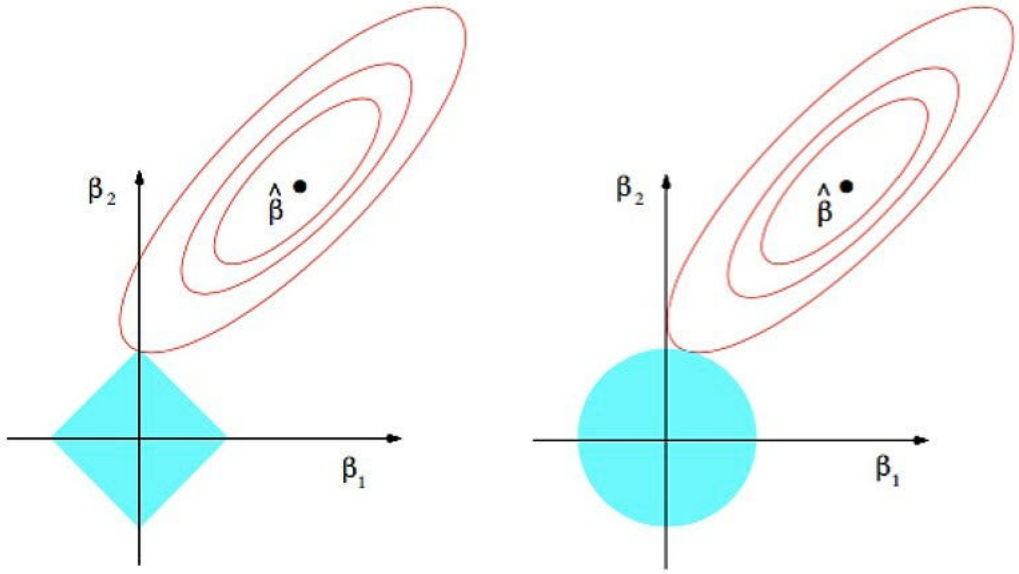



Figure 5.1: Bias and Variance of Regularisationm from [36]

These techniques are defined by the equation:

$$Loss = Error(Y - \widehat{Y}) + \lambda \sum_{1}^{n} |w_i|^N \tag{5.4}$$

where N is 1 for L1 regularisation and 2 for L2 regularisation. In this equation, the Error is calculated using the gradient descent and the combination of these values finds the minimum loss function, which is used to update the weights and recalculate the loss. Alternatively, in models where the size of the feature space is small, it is calculated using the Normal Equation defined by:

$$J(\theta) = \frac{1}{2n}\sum_{t=1}^{n}[y^t - \theta(x)^t]^2 + \lambda||\beta||^N \tag{5.5}$$

### 5.1.2 Support Vector Machines

The support vector machine for regression works similarly to linear regression using a hyperplane instead. The width of the hyperplane is calculated by minimising the errors of the margin i.e. the distance of points which fall out of the margin known as the $\varepsilon$ -sensitive tube This is calculated using an equation

$$\frac{1}{2}||\omega^2|| + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \rightarrow min \tag{5.6}$$

where **w** represents the equation of the linear regression
**C** represents the box constraints and
$\xi_i$ and $\xi_i^*$ represent the slack variables which determine the soft margin width.

The dimensionality of this margin can also be increased or reduced using mappings like the radial basis function, which allows the feature to be projected in a different dimensional space where more information can be retrieved about the relationships between the features.

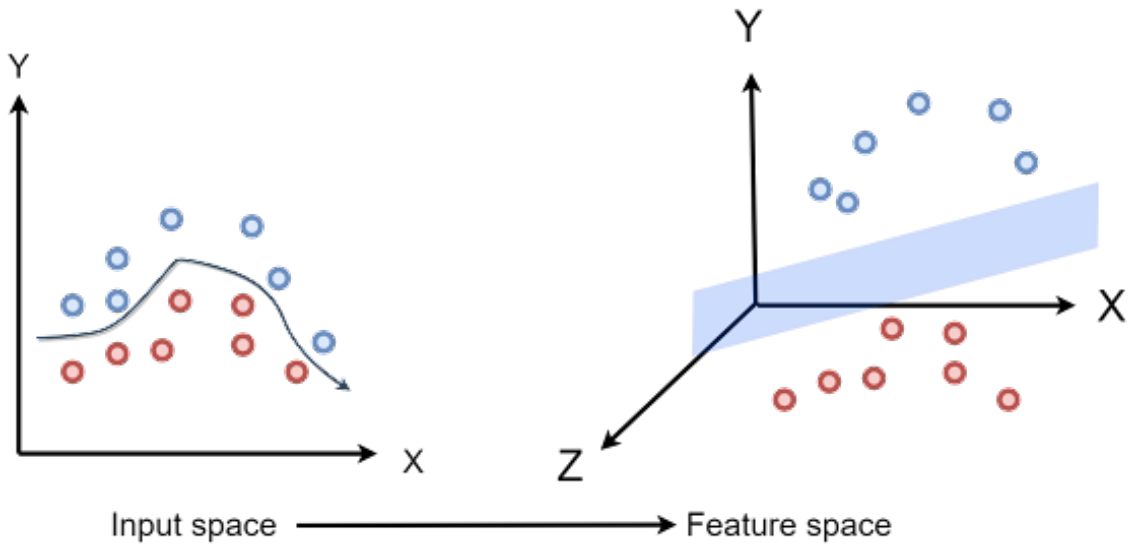



Figure 5.2: Mapping of feature space to a higher dimension [37]

## 5.2 Decision Trees

### 5.2.1 Overview

Decision Trees are non-parametric supervised training models which aim to learn a classification and regression task based on a hierarchical tree of binary questions[38].

There are two methods of finding the feature for the next internal nodes. These methods include Gini impurity, which measures the probability of a wrong prediction of a random instance in the dataset and Entropy, which measures the state of disorder or impurity in the dataset[38]. When Gini impurity is used, the next estimator is determined by finding the feature with the minimum impurity in the training data. This is calculated for regression using a threshold value, which is used to split the data into two. For each group, the variance is calculated, and the weighted sum is calculated using the proportion of samples i.e. the sum of squared error $p^2$. This is then subtracted from 1 to give the Gini Index. This is defined by:

$$Gini = 1 - \sum_i p_i^2$$

### 5.2.2 Random Forest

Decision Trees are known for overfitting the data. This is caused mostly by the infinite combination of functions, i.e. the depth of the tree. One of the most effective ways of improving this is Ensemble learning.

Random Forest is a type of ensemble model which aims to find the optimal combination of simple decision trees trained in parallel on randomly selected replaceable samples of the dataset i.e. Bootstrap Aggregation, where the performance of the model is defined by how much the variance is reduced. Additionally, as a result of the split, Random Forest algorithms have embedded feature selection where only relevant features are selected for each tree such that they are given higher preference. This is illustrated in Figure 5.3.

### 5.2.3 Light Gradient Boosting

Light Gradient Boosting Method builds the trees one after the other using a gradient boosting ensemble method. Gradient Boosting Methods work by combining weak learners iteratively where it is assumed by adding some function $h(x)$ to a current function $F(x)$ it will result in the true prediction i.e.

$$F_{m+1}(x_i) = F_m(x_i) + h_m(x_i) = y_i \tag{5.7}$$

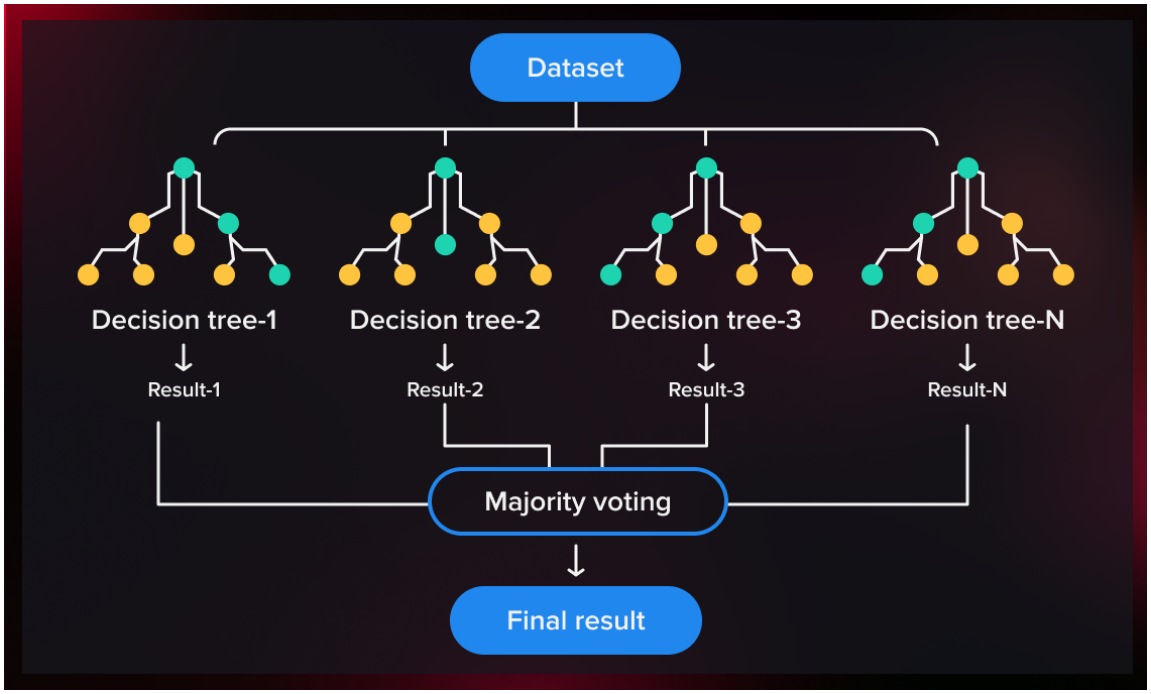



Figure 5.3: Illustration of Random Forest from [39]

The goal of this method is to find some approximation of $\hat{F}$ which minimises the loss function of y. This is defined as:

$$\hat{F}(x) = \sum_{m-1} M\gamma_m h_m(x) + constant \tag{5.8}$$

where $\gamma_m$ represents the weights attributed to the weak learners calculated from the gradients. $F_i(x)$ is calculated iteratively to find the function which has the minimum loss[40].

$$\hat{\gamma}_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

In Light Gradient Boosting, the iterative search is done using a bottom-up approach starting from the target values where it uses the gradient based-algorithm to split the nodes, the largest gradient first. This method employs the Gradient-Based One Side Sampling Technique to make it more efficient.

Gradient-Based One Side Only Sampling Technique works by attributing different weights, based on the information gained using the gradient. This method keeps instances with gradients larger than the specified threshold, i.e. gradients that are under-trained, and randomly drops smaller gradients with the aim of retaining or improving the information [41].

Additionally, the Exclusive Feature Bundling Technique is used to improve the algorithm when sparse data is used. This works by reducing the number of features

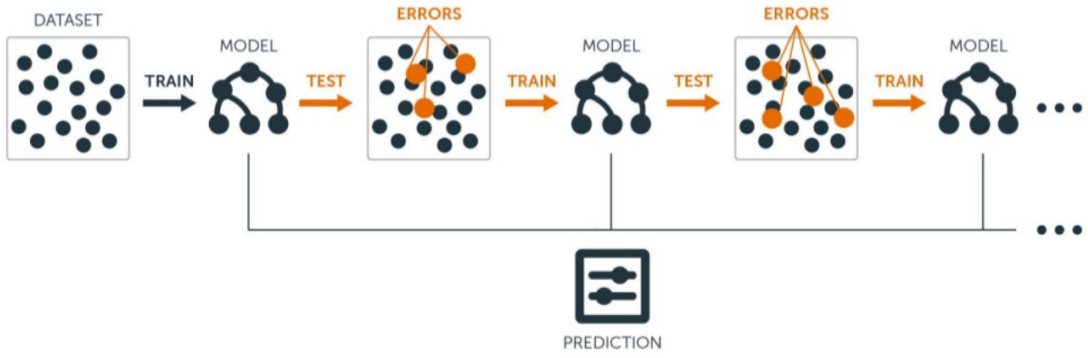



Figure 5.4: Illustration of Light Gradient Boosting from [42]

used in decision trees. Features which are mutually exclusive to one another are put into histogram bins. This helps the algorithm ignore unnecessary nodes, which may not improve the performance of the model given the task.

## 5.3 Regression Implementation

The classical regression and Random Forest were implemented using the package Scikit-learn, where the input parameters were the train-test-split of the feature matrix and target labels i.e. LogS. However, due to the ranges of value in the feature matrix and its sparsity, the models were more prone to underflow error. To mitigate this, the feature matrix was normalised using the MinMax Scaler, where the values were normalised to fit a range between 0 and 1. Similarly, the Light Gradient Boosting Algorithm was implemented using the Lightgbm model using the same inputs and target variables as the models mentioned previously.

In the first iteration, all these methods were implemented using their default parameter to measure the average performance as well as to check whether the models were underfitting i.e. the performance of the training and validation data were poor due to over-generalisation or overfitting i.e. the performance of the train data was better than that of the validation data. Hyperparameter tuning was done in the next iteration to try to improve the performance.

### 5.3.1 Hyperparameter Tuning

Hyperparameter tuning was done using a randomised search for each model with the values of parameters shown in the table below:

To improve the models, k-fold cross-validation was also implemented where the number of splits was based on the time of computation with the minimum splits

| Model | Hyperparameternames | Range |
|---|---|---|
| Ridge Regression | Alpha | 0.0001, 0.001, 0.01, 0.1, 1, 10, 100 |
| Ridge Regression | Fit_Intercept | True,False |
| Ridge Regression | Solver | auto, svd, cholesky, lsqr, sparse_cg, sag, saga |
| SVM | Kernel | linear, poly, rbf |
| SVM | Gamma | scale, auto, 0.1, 0.01, 10, 0.001 |
| SVM | Degree | 1,2,3 |
| SVM | C | 0.0001, 0.001, 0.01, 0.1, 1, 10, 100 |
| Random Forest | n_estimators | 5, 10, 20, 30, 40, 50, 100 |
| Random Forest | min_samples_split | 2, 4, 6, 10, 20, 40, 60 |
| Random Forest | min_samples_leaf | 1, 3, 5 |
| Random Forest | bootstrap | True,False |
| Random Forest | max_depth | Numpy linespace with a minimum of 10 and maximum of 120 of length 12 |
| Light Gradient Boosting Method | learning rate | Numpy linespace with a minimum of 0.001 and a maximum of 0.5 of length 10 |
| Light Gradient Boosting Method | num_leaves | Numpy linespace with a minimum of 20 and a maximum of 300 of length 5 |
| Light Gradient Boosting Method | max_depth | Numpy linespace with a minimum of 5 and a maximum of 300 of length 5 |
| Light Gradient Boosting Method | boosting_type | gbdt, dart, goss |

Table 5.1: Hyperparameter values used for tuning

being three. This resulted in 98 iterations for Ridge Linear Regression, 100 for Random Forest Trees, 10 for Support Vector Regressors and 300 for Light Gradient Boosting Trees.

Consequently, these models were also used to compare the performance difference between the classical regression models, decision trees and deep learning models.

## 5.4 Implementation of Classification-Informed Models

Once the models were implemented with the curated dataset, it was found that the RMSE value was slightly higher than that of most studies, with a difference between 0.1-0.8 for all models. As a result of this, the models were compared to results from the reproducible models of '*Machine learning with physicochemical relationships: solubility prediction in organic solvents and water*'[10] and models generated with AQSolDB.

The next iteration compared the non-classification-informed models to classification-informed models of the classical machine learning models and decision trees. To see how the classification affects the models, multiple combinations of the two classification strategies were used. These combinations include:

1. QSPR

2. QSPR and Butina Clustering
3. QSPR, Butina Cluster and DrugTax Superclasses
4. QSPR, Butina Cluster and DrugTax Kingdom
5. QSPR and DrugTax Superclasses
6. QSPR and DrugTax Kingdom
7. QSPR and DrugTax complete taxonomy
8. QSPR, Butina Clustering and DrugTax complete taxonomy

# Chapter 6

# Graph Neural Network - AttentiveFP

## 6.1 Graph Neural Networks

Graph neural networks work by representing data as graphs of nodes and vertices, where a graph may represent the whole dataset or a data point in the data. These graphs are used as inputs to a special type of neural network with 3 fundamental layers. These layers include the permutation equivalent layer which updates the graph representations, local pooling which down-samples the graph and the global pooling layer which provides the fixed-size representation of the graph such that the size is not affected by the permutation layer[24]. The output of this neural network provides the result of the task.

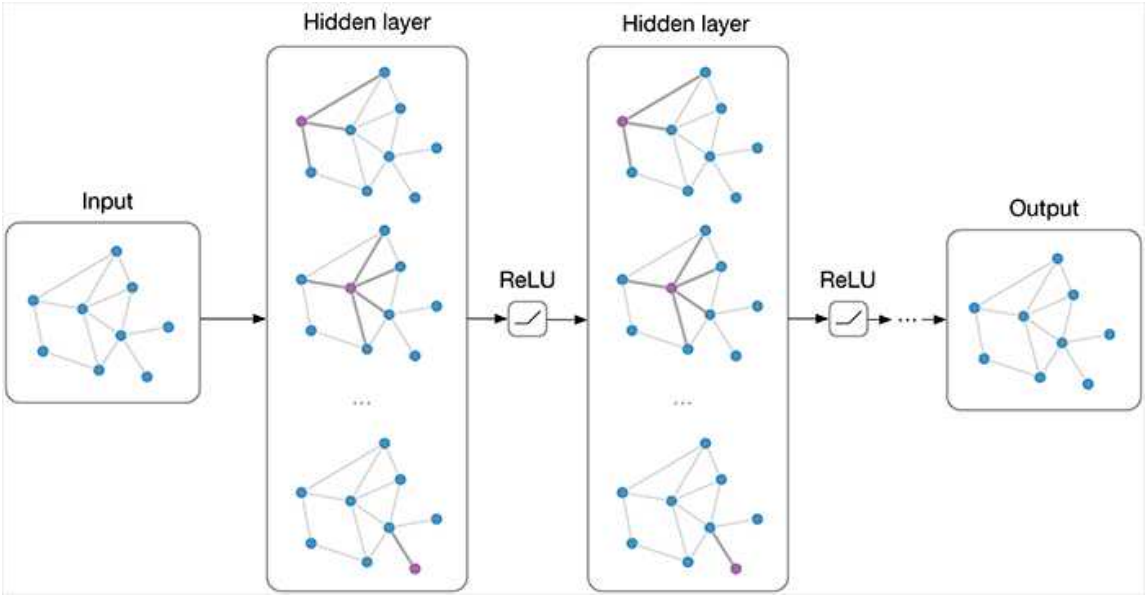



Figure 6.1: Illustration of Graph Neural Network from [24]

In most graphical neural networks, the permutation-equivalent (also known as message passing) layers are usually implemented using pairwise message passing, where nodes are updated based on the aggregation function of messages from their closest neighbours. This function changes based on the type of neural network implemented. These types include Graphical Convolution Networks based on filters and Graphical Attention Networks or Gated Sequence Neural Networks based on the attention mechanism, where the latter is also based on Recurrent Neural Networks[24].

### 6.1.1 Graph Attention Mechanism

Graph neural networks have been developed to use the self-attention mechanism to learn the features of a graph's nodes and edges. This process usually involves using three operations to get the context of the data. This includes the Alignment/Self-Attention operation, which transforms the input features to a higher level transformation using a weight matrix which indicates the importance of the node. This is transformed using the Weighting function, which enables the nodes to be easily comparable by dropping all structural information in most models. This is known as the Weighting operation. Once calculated, the coefficient is used to find the linear combination of features which become the output features of the node. This operation is known as the Context operation, which is commonly the ELU function[25]. These operations are defined by:

**Alignment**

$$e_{vu} = \phi(W.[h_v h_u]) \tag{6.1}$$

where $\phi$ can be any activation function.

**Weighting[25]**

$$a_{vu} = \psi(e_{vu}) \tag{6.2}$$

where $\psi$ can be any activation function.

**Context[25]**

$$C_V = \varphi[\sum_{w \in N(v)} a_{vu}.W.h_u] \tag{6.3}$$

where $\varphi$ can be any activation function. In graphical neural networks, this introduces two phases which include the Messaging phase and the Readout phase where the readout phase updates the previous node, by aggregating the messages from neighbouring nodes calculated using the attention mechanism. This result updates the value stored in the Gated Recurrent Unit. This process is repeated multiple times using different layers similar to convolutional layers, and it is defined by:

**Messaging**

$$C_v^{k-1} = \sum_{w \in N(v)} M^{k-1}(h_v^{k-1}, h_v^{k-1}) \tag{6.4}$$

**Readout**

$$h_v^k = GRU^{k-1}(C_v^{k-1}, h_v^{k-1}) \tag{6.5}$$

In this project, the type of Self Attention Neural Network used is the AttentiveFP. The operations for the self-attention mechanism in AttentiveFP are defined by:
**Alignment[25]**

$$e_{vu} = leaky\_relu(W.[h_v h_u]) \tag{6.6}$$

**Weighting[25]**

$$a_{vu} = softmax(e_{vu}) = \frac{exp(e_{vu})}{\sum_{w \in N(v)} exp(e_{vu})} \tag{6.7}$$

**Context[25]**

$$C_V = elu[\sum_{w \in N(v)} a_{vu}.W.h_u] \tag{6.8}$$

## 6.2 AttentiveFP Implementation

The AttentiveFP model implemented in this project was similar to the one implemented in the original paper[25]. This project uses the InMemory Dataset from PyTorch geometric to derive the graph and store them in the form of data objects which are used when training the model.

This process involved creating a pytorch_geometric data object which stores global features, node and edge attributes as well as the edge index which shows the connections.

Variations of the nodes include the classification output to support the evaluation of the classification-informed models. Additionally, two different global features embedding were added to represent the different combinations of classification and descriptors.

Once implemented and stored, the dataset was divided into a training, validation and test set with an 80:10:10 split which is trained using batch training. This data was passed to the AttentiveFP model, which is already implemented in the pytorch_geometric package.

Only the node embeddings without classification output and QSAR descriptors were used in the first iteration, and it was trained for 200 epochs.

For the iteration of the non-classification-informed AttentiveFP model and the best classification-informed AttentiveFP model, hyperparameter tuning was also done to find the best hyperparameters. This search was done using the Optuna package, which is based on Bayesian Optimisation. In this optimisation technique, the model is treated as a black box, where the objective function of the model is unknown to the optimiser. This process improves the likelihood of the model converging at the

global minima, i.e. the lowest RMSE values of the model.

The result of the AttentiveFP model was compared to the original paper, where it showed that the RMSE value was significantly higher than the paper. To understand the reason behind this, the test values were compared to the model using the ESOL dataset, the dataset from *A comparison study of descriptor-based and graph-based models*[43], and the model train on ESOL tested with the curated dataset from this project.

## 6.3 Implementation of Classification-Informed AttentiveFP

The previous classification models were compared to AttentiveFP. Due to its cost, only four versions of AttentiveFP were compared for differences. These versions were created for various graph embedding types. The model was initially trained using node-embedded classification data. These results were poor with RMSE at 60.

Graph embeddings were subsequently developed for distinct classifications and descriptors. A new model was implemented by combining a linear neural network and the AttentiveFP model. The output was obtained by passing the result to a feed-forward network.

The linear layer worked by passing the embedded matrix through two hidden layers with 200 neurons to get the final result. Weights were updated with a linear function, and outputs were transformed by ReLU to prevent gradient explosion. 200 epochs were used in this model training.

# Chapter 7

# Results and Analysis

## 7.1 Evaluation of Non-Classification Informed Models.

Performance metrics were used to compare models before embedding classification data into the feature matrix. Linear Regression with Ridge Regularisation was initially underfitted, but hyperparameter tuning helped mitigate issues, and it was more prone to overfitting. Results improved using Support Vector Machines, with 1.0 and 0.2 improvement in metrics for training and 0.6 and 0.3 for testing. These results are shown in Tables 7.1, 7.2, 7.3 for Root Mean Squared Error and Tables 7.4, 7.5 where the performance of the training in data is annotated using **Train_** before the model name. Furthermore, the minimum loss is highlighted.

These results were further improved when using Decision Trees, where the Random Forest and Light Gradient Boosting improved the performance by 0.43 and 0.2 for the RMSE of the training data and 0.02 and 0.3 for the RMSE of the test data. Despite improved results and hyperparameter tuning, all models still overfit data. Figure 7.1 compares models with and without tuning; Random Forest performs better for train data and Light Gradient Boosting Method performs better for test data.

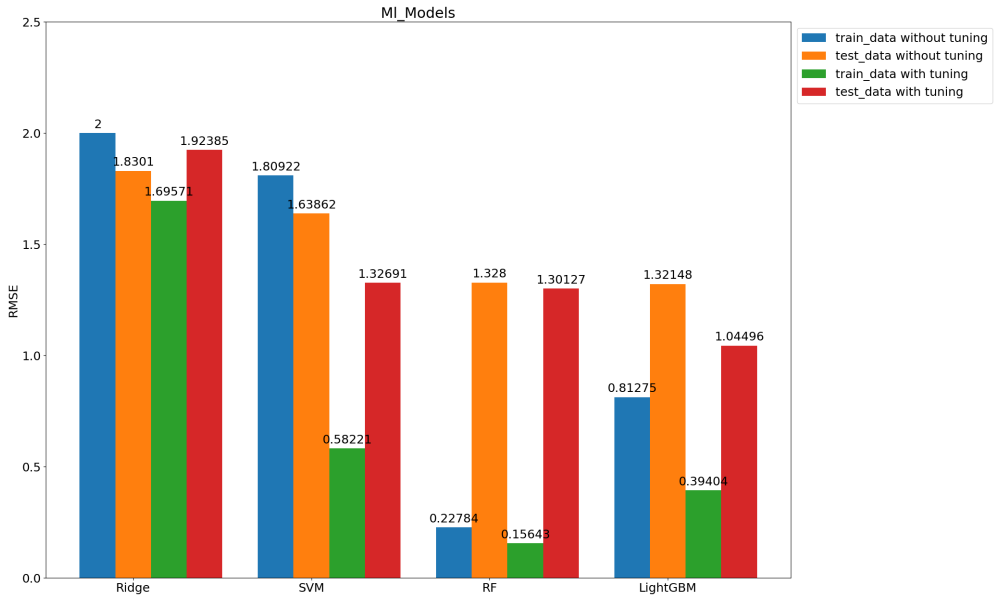



Figure 7.1: Linear Regression and Decision Tree Models Without Classification

These errors may be due to varying feature weightings, with some features having

similar weights and many having different weights. Yet, these features have a significant impact on compound solubility. Features like SLogP and MolLogP are connected to LogP, which measures a compound's dissociation between water and an organic solvent. The solubility is affected by these values, as higher values in the organic solvent make it less likely to dissolve in water and vice versa. These feature importance are illustrated in Figure 7.2 which shows the twenty best features of each of the models.

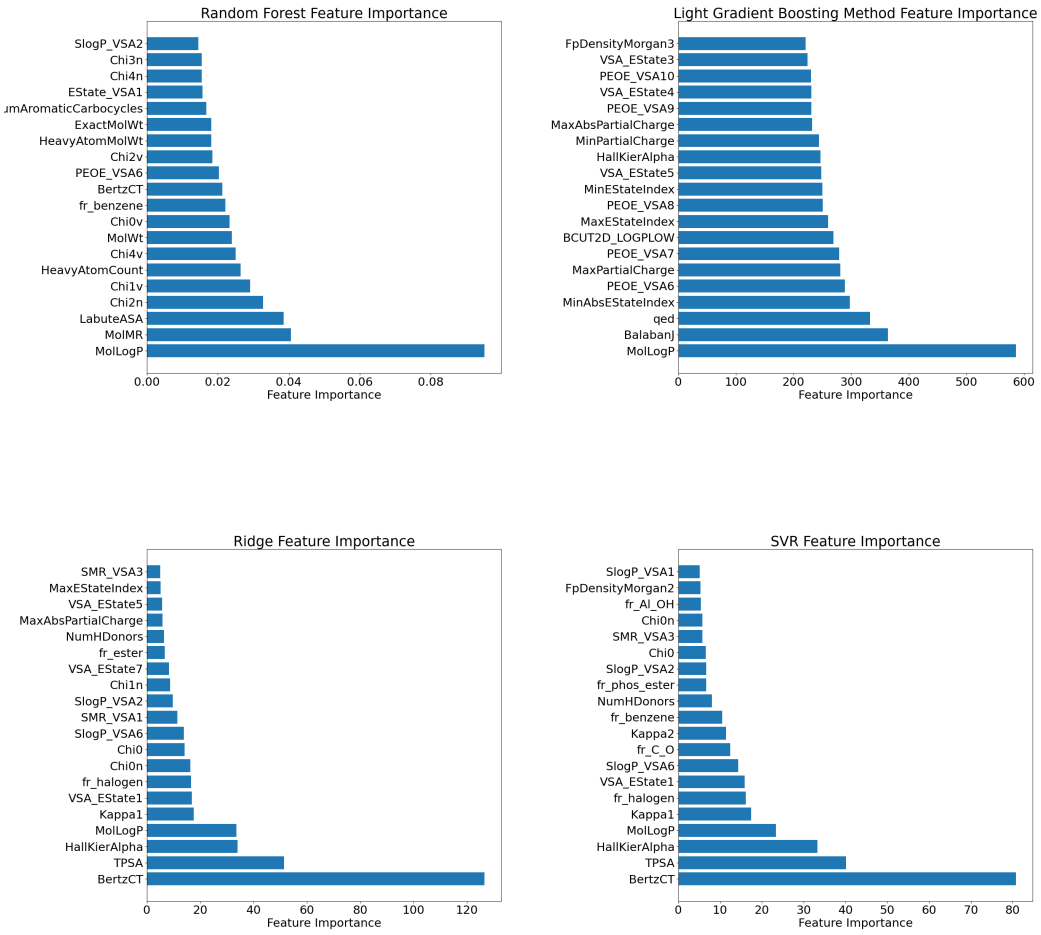



Figure 7.2: Diagram of Feature Importance

To compare these results to previous studies, the dataset, and model were changed to model a specific study or part of the study before feature generation. It was observed during this process that the change in training data did not change the performance of the model, additionally, the result when using the ESOL dataset was identical to that of the ESOL dataset performance from the study. Furthermore,

the performance of the model when handling the dataset from *A comparison study of descriptor-based and graph-based models*[43], was identical to the paper, therefore it is proposed that the change in performance was due to the change in the size of the dataset. Consequently, the performance of the model using the curated dataset was used as a benchmark for further comparison and iteration. This comparison is shown in Graph 7.3

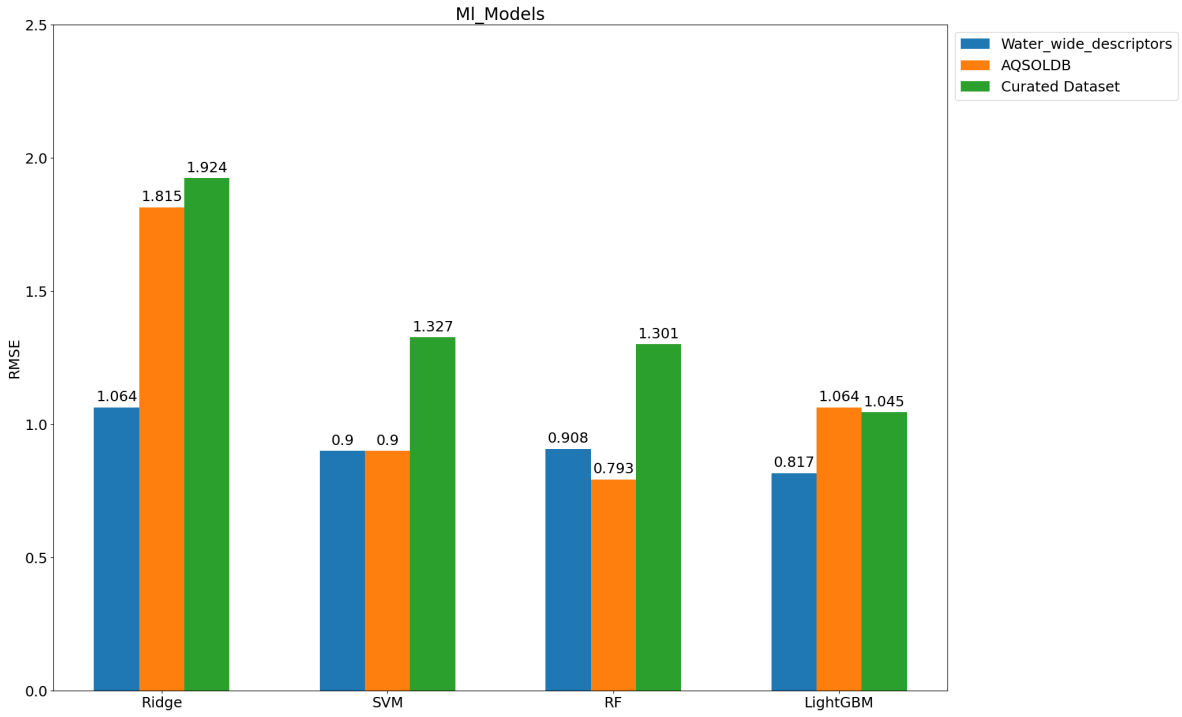



Figure 7.3: Curated Dataset Compared to AQSOLDB and Water Wide Descriptors[10]

Table 7.1: RMSE Results from Decision Trees and Classical Machine Learning Model

| Feature_type | QSPR | QSPR+butina | QSPR+butina+kingdom | QSPR+butina+superclass |
|---|---|---|---|---|
| **Train_Ridge** | 1.696 | 1.786 | 1.800 | 1.744 |
| **Train_SVM** | 0.582 | 1.147 | 1.116 | **0.666** |
| **Train_RF** | 0.156 | 0.168 | 0.034 | 0.020 |
| **Train_LightGBM** | 0.394 | 0.428 | 0.422 | 0.248 |
| **Ridge** | 1.924 | 2.294 | 2.206 | 1.898 |
| **SVM** | 1.327 | 1.482 | 1.498 | 1.793 |
| **RF** | 1.301 | 1.365 | 1.392 | **1.158** |
| **LightGBM** | 1.045 | 1.131 | 1.090 | **0.915** |

Table 7.2: RMSE Results from Decision Trees and Classical Machine Learning Models-2

| Feature_type | QSPR+kingdom | QSPR+superclass | QSPR+full_taxonomy | QSPR+full_classification |
|---|---|---|---|---|
| **Train_Ridge** | 1.827 | 1.725 | **1.705** | 1.710 |
| **Train_SVM** | 1.953 | 1.501 | 1.200 | 0.703 |
| **Train_RF** | 0.184 | 0.024 | 0.024 | **0.019** |
| **Train_LightGBM** | 0.359 | 0.474 | **0.119** | 0.205 |
| **Ridge** | 1.943 | 2.041 | 1.866 | **1.815** |
| **SVM** | 2.135 | 1.518 | 1.547 | **1.367** |
| **RF** | 1.360 | 1.272 | 1.368 | 1.305 |
| **LightGBM** | 1.336 | 1.191 | 1.204 | 1.064 |

Table 7.3: RMSE Results from Decision Trees and Classical Machine Learning Models-3

| Feature_type | Non_classification with Hyperparmater Tuning | AqSolDB | ESOL |
|---|---|---|---|
| **Train_Ridge** | 2.000 | 1.710 | 1.088 |
| **Train_SVM** | 1.809 | 0.581 | 0.581 |
| **Train_RF** | 0.228 | 0.009 | 0.005 |
| **Train_LightGBM** | 0.813 | 0.205 | 0.036 |
| **Ridge** | 1.830 | 1.815 | 1.064 |
| **SVM** | 1.639 | 0.900 | 0.900 |
| **RF** | 1.328 | 0.793 | 0.908 |
| **LightGBM** | 1.321 | 1.064 | 0.817 |

To see if these results could be further improved using more features at a trade-off of computational cost, it was compared to the performance of the AttentiveFP without classification. In this scenario, the Light Gradient Boosting Model still performed better than the Graph Neural Network, with an RMSE difference of 0.3 for the test data as shown in the Graph 7.4.

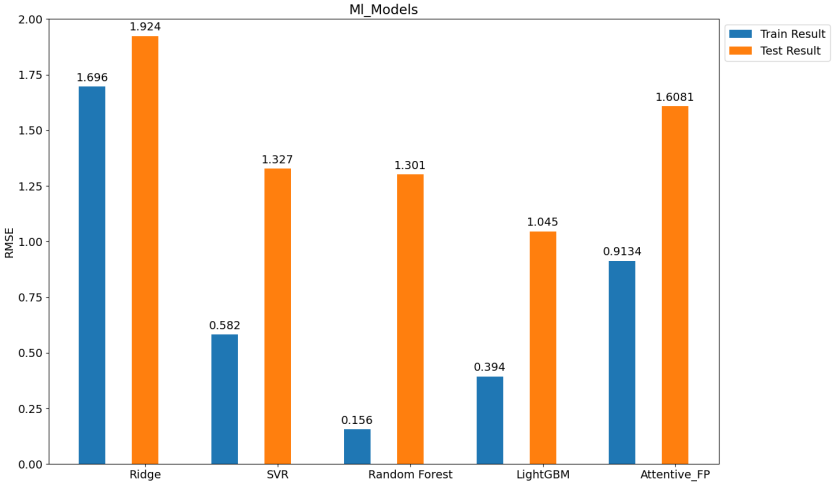



Figure 7.4: Final Results of the Non-Classification Informed Models - RMSE

## 7.2 Evaluation of Classification Informed Models

Similar to the non-classification-informed models, Light Gradient Boosting Method and Random Forest Method also performed better than the Linear Regression Model and the Support Vector Machines. The performance difference between the Ridge Regression and Support Vector Machine was 0.75 for the RMSE and 0.01 for the MAE of the test data. The Light Gradient Boosting performed the best compared with the Support Vector Machine, where the performance difference was 0.44 for the RMSE value and 0.29 for the MAE value.

Table 7.4: MAE Results from Decision Trees and Classical Machine Learning Models -1

| Feature_type | QSPR | QSPR+butina | QSPR+butina+kingdom | QSPR+butina+subclass |
|---|---|---|---|---|
| **Train_Ridge** | 0.975 | 0.976 | 0.978 | 0.944 |
| **Train_SVM** | 0.706 | 0.691 | 0.676 | 0.941 |
| **Train_RF** | 0.247 | 0.236 | 0.080 | 0.228 |
| **Train_LightGBM** | 0.547 | 0.468 | 0.460 | 0.340 |
| **Ridge** | 1.015 | 1.006 | 0.993 | 0.966 |
| **SVM** | 0.840 | 0.799 | 0.831 | 0.970 |
| **RF** | 0.815 | 0.773 | 0.780 | 0.764 |
| **LightGBM** | 0.824 | 0.683 | 0.693 | 0.658 |

Table 7.5: MAE Results from Decision Trees and Classical Machine Learning Models - 2

| Feature_type | QSPR+kingdom | QSPR+subclass | QSPR+full_taxonomy | QSPR+full_classification |
|---|---|---|---|---|
| **Train_Ridge** | 0.975 | 0.935 | 0.935 | 0.948 |
| **Train_SVM** | 0.963 | 0.935 | 0.935 | 0.949 |
| **Train_RF** | 0.246 | 0.074 | 0.075 | 0.231 |
| **Train_LightGBM** | 0.546 | 0.493 | 0.240 | 0.444 |
| **Ridge** | 0.985 | 1.003 | 1.003 | 0.953 |
| **SVM** | 0.986 | 1.004 | 1.004 | 0.943 |
| **RF** | 0.789 | 0.784 | 0.783 | 0.740 |
| **LightGBM** | 0.789 | 0.751 | 0.722 | 0.662 |

Linear Regression and Decision Trees had different optimal combinations. Linear

Regression favoured full taxonomy classification and the Butina cluster index, while Decision Tree preferred superclasses of taxonomy and the Butina cluster index. This is due to the majority of organic compounds in the dataset, resulting in lower weight assigned to kingdoms. When decision trees are tuned, the low weight of the kingdom features likely results in their pruning.

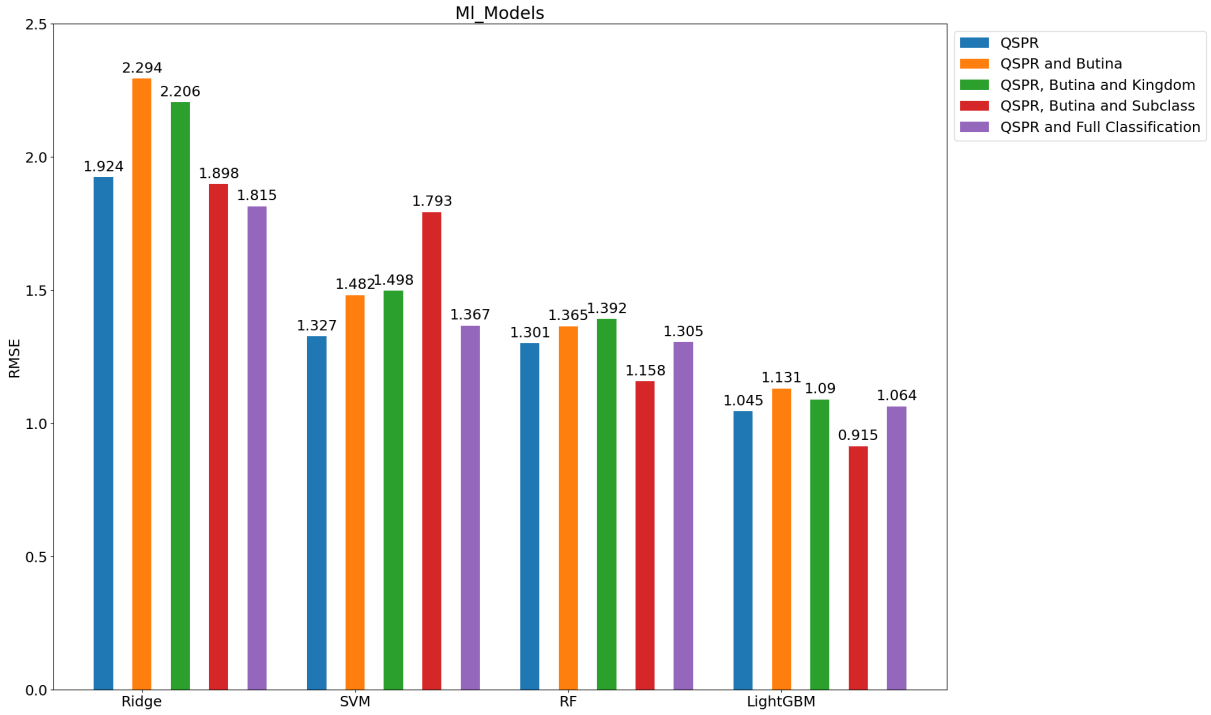



Figure 7.5: Best Results after HyperParameter Tuning with Classification

The results from modelling the classification-informed data also showed that all the graph embedding with extra information had the same performance as the AttentiveFP model. However, it is also important to note that the result of the best model still performed worse than the reproducible model and dataset in the paper. However, this was due to the smaller size of data used as when compared to *A comparison study of descriptor-based and graph-based mode*l [43], the error showed to increase with size. This result is shown in Graph 7.6 where the letter C stands for the model with classification and CD stands for the model with classification and descriptors.

Additionally, it is also worth noting that similar to the decision tree models, the AttentiveFP model also overfitted the data even with hyperparameter tuning such that the best training RMSE loss was 0.547 while the validation loss and test loss were 1.8 and 1.747 respectively. This is shown in Table 7.6. For the different combinations, the AttentiveFP models also had different mean and standard deviation for 200 epochs where the curated dataset with no global descriptors still had the minimum mean and standard deviation for all classification combinations as shown in Table

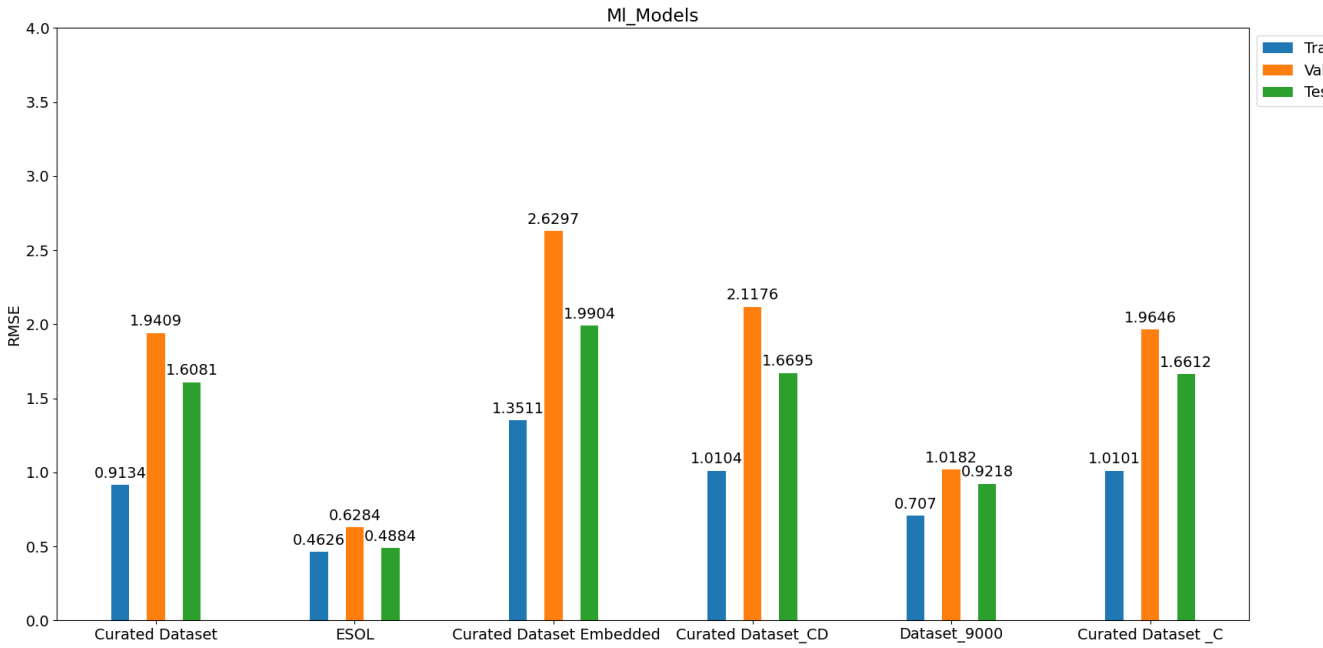



Figure 7.6: Results from the Four Combinations of Features used to train the AttentiveFP model

7.7 and 7.2.

| Model | Minimum_Train | Minimum_Validation | Minimum_Test | Epoch with Minimum Value |
|---|---|---|---|---|
| **Curated Dataset** | 0.9134 | 1.9409 | 1.6081 | 16 |
| **Curated Dataset Embedded** | 1.3511 | 2.6297 | 1.9904 | 172 |
| **Curated Dataset_CD** | 1.0104 | 2.1176 | 1.6695 | 193 |
| **Curated Dataset _C** | 1.0101 | 1.9646 | 1.6612 | 200 |
| **Dataset_9000** | 0.707 | 1.0182 | 0.9218 | 44 |
| **ESOL** | 0.4626 | 0.6284 | 0.4884 | 146 |

Table 7.6: RMSE Results from AttentiveFP

This established that the best AttentiveFP model to compare with the Light Gradient Boosting Method was the original model proposed in the paper reproduced using the curated dataset, as the performance of the classification model was not high enough when compared to the computational cost of running.

Subsequently, given the results observed during this project, the model which is proposed for the more practical use case when comparing AttentiveFP and Light Gradient Boosting Method is the Light Gradient Boosting Method.

This decision tree model is less computationally expensive and has better performance where the MAE value and RMSE value were lower, i.e. 0.3403 for the MAE of the Training Dataset, 0.65837 for the MAE of the Test Dataset, 0.248 for the RMSE of

| Model | AVG_Train | AVG_Validation | AVG_Test |
|---|---|---|---|
| **Curated Dataset** | **0.6871** | **2.1403** | **1.8535** |
| **Curated Dataset Embedded** | 2.0793 | 2.8183 | 2.5484 |
| **Curated Dataset_CD** | 1.0792 | 2.3444 | 1.8646 |
| **Curated Dataset _C** | 1.0841 | 2.2843 | 1.87 |
| **Dataset_9000** | 0.8547 | 1.0921 | 1.0396 |
| **ESOL** | 0.4186 | 0.7062 | 0.6437 |

Table 7.7: Mean of RMSE Results from AttentiveFP

| Model | STD_Train | STD_Validate | STD_Test |
|---|---|---|---|
| **Curated Dataset** | 0.1685 | **0.1551** | **0.0849** |
| **Curated Dataset Embedded** | 6.8001 | 2.7351 | 2.6238 |
| **Curated Dataset_CD** | 0.1301 | 0.2964 | 0.2196 |
| **Curated Dataset _C** | **0.1191** | 0.2632 | 0.203 |
| **Dataset_9000** | 0.1797 | 0.0789 | 0.11 |
| **ESOL** | 0.2476 | 0.1342 | 0.2515 |

Table 7.8: Standard Deviation of RMSE Results from AttentiveFP

the Training Set and 0.915 for the RMSE value of the Test Set when compared to the RMSE value of AttentiveFP where the Training Set was 0.9134 and the Test set was 1.6081. Additionally, the variation in loss given different hyperparameters and features was less than that of any other model implemented during the project. This is shown in Figure 7.7

Based on the result, the Light Gradient Boosting Method was analysed to determine what features had a larger significance, where Figure 7.8 shows the 20 best features of the Light Gradient Boosting Method. It was found that the butina_cluster which was a novel feature added to the data and MolLogP which is the partition coefficient that evaluates whether it is more soluble in water or Octanol had the highest feature importance. Additionally, multiple Partial Charges based descriptors and absolute E-state index also had higher feature importance compared to others. However,

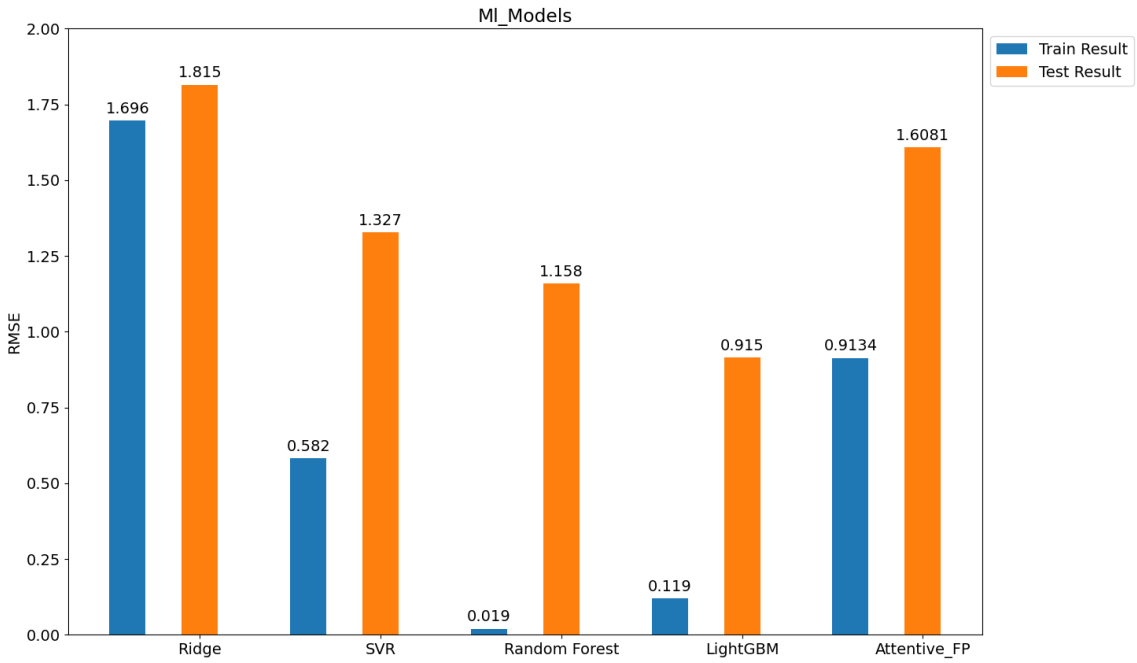



Figure 7.7: Final Results

some features including multiple superclasses and indexes related to the number of functional groups had no importance to the model.

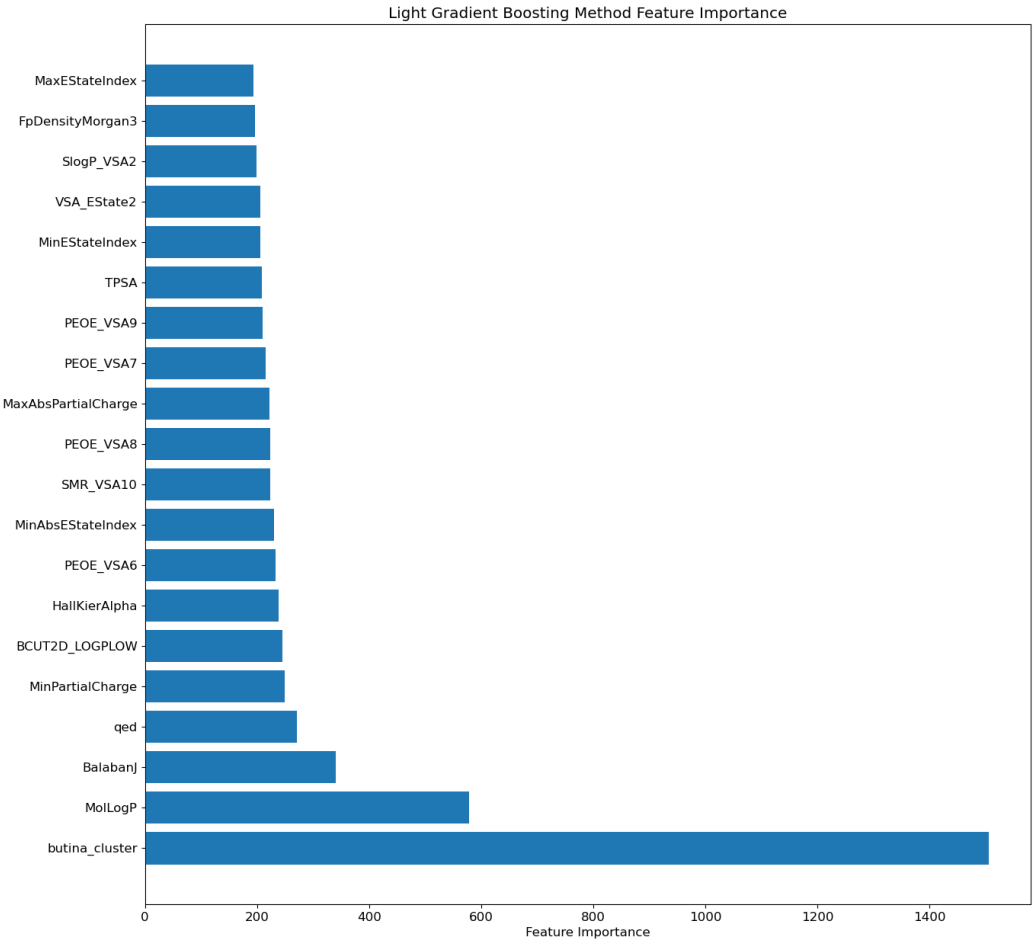



Figure 7.8: Feature Importance of the 20 most important features using Light Gradient Boosting

# Chapter 8

# Conclusion

To a large extent, this project has been successful in predicting the solubility of compounds. Additionally, it has also provided explanations and answers to the questions set out in the project, i.e. This project further demonstrated that the

| **Questions** | **Answers** |
|---|---|
| Will classifying the data based on chemistry research improve the performance of the model? | Yes it does. However, it works better when using unsupervised learning techniques e.g. Butina clustering which is closely related to K-means clustering |
| Are the current models reproducible if all other environmental characteristics of the experiment to discard solubility are ignored? | Yes they are. However, the performance of the model is dependent on the type of features generated computationally and the accuracy of that data. |
| How can the current models be used to aid research without compromising safety? | They can be used to approximate the values of novel compounds where there is no official standard of solubility or no physical compound to test, i.e. discovery stage. |
| Can the current model be used in a way that is efficient to reduce cost, | Possibly? If combined with other predictions, they can provide more insight into the physical stability of the compound used to produce the drug. |

Table 8.1: Answered Questions from the Requirements

effectiveness of the model proposed in the original publication of AttentiveFP can primarily be attributed to the limited size of the dataset rather than its complexity. Consequently, the model's susceptibility to degradation becomes increasingly apparent as the size of the dataset expands.

Furthermore, the modelling and analysis demonstrated that whilst machine learning models may not predict the solubility of compounds to absolute precision, they can yield a reliable approximation that can serve as a foundation for further data analysis within the field of Chemistry. This approach can be utilized for the estimation of

novel compounds since all the important features can be determined without carrying out experimental procedures.

In conclusion, it can be proposed that by extracting data from unsupervised learning techniques and combining them with descriptors, it is possible to train a model with a larger dataset that performs better than previous models shown in research for novel compounds.

## 8.1 Limitations

Although this project has been completed, there were also limitations based on the nature of the models currently available and the features used that might have reduced the performance. One of such limitations is the lack of experimental features that were common for all 14,000 compounds used in the project. This was caused mostly by the nature of the dataset used, where datasets from papers were used instead of databases such as PubChem or ChemBL which required payments or the use of APIs to access the data.

In addition to this, when modelling features using different types of graphical neural networks including the one implemented in this paper, it is very difficult to model and contextualise the global context of the entire graph in the model [44]. As a consequence of this, an additional model needs to be trained where information can be extracted from the whole graph. This usually leads to the same or a decrease in performance as seen in this project.

## 8.2 Future Work

Future research on this study might extend training the Light Gradient Boosting Model on databases consisting of novel compounds and increasing the dataset size from 14,000 to millions of compounds. Given that the size of the dataset is increasing, a Deep Learning Model using transformers might be implemented to improve the performance of AttentiveFP.

# Chapter 9

# Project Management and Risk Analysis

### 9.0.1 Risk Analysis

There are three major risks involved in my project. These risks were characterised using a risk table shown in Figure 9.1 where they were evaluated based on three levels of High(H) - most likely to cause damage or happen during the project, Medium (M), Low (L) - Least likely.

| **Event** | **Risk Probability** | **Risk Impact** | **Risk Mitigation** |
|---|---|---|---|
| Falling ill | M | M | I left days free and overestimated the time taken. I also left at least a week before the deadline to allow for further improvements. |
| Loss of data | L | H | All the data was stored to my OneDrive and frequently updated. Additionally, files were stored on GitLab and regularly updated to ensure the retrieval of previous versions. |
| Lack of hardware | H | M | Using the university laboratory systems and GPU programs helped mitigate this issue when it occurred. |

Table 9.1: Risk Assessment

### 9.0.2 Project Management

Technical tasks have been allocated on a bi-weekly basis. However, I have left the weekends and at least 2 days in between dependent tasks for contingency situations where risks occur, delaying my work. The task of writing the report has also been put into monthly periods, which start once there is enough information from the model to start the write-up. This allows me to make changes to the implementation that I find during the write-up. This has been managed using a Gantt chart.

Figure 9.1: Gantt Chart for Project Management

# Bibliography


[1] A. Sharma, T. Virmani, V. Pathak **andothers**, "Artificial intelligence-based data-driven strategy to accelerate research, development, and clinical trials of COVID vaccine," *BioMed Research International*, **jourvol** 2022, **page** 7 205 241, 6 **july** 2022, ISSN: 2314-6133. DOI: 10.1155/2022/7205241. **url**: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9279074/ 20/11/2022.

[2] "Solubility | chemistry | britannica." (), **url**: https://www.britannica.com/science/solubility-chemistry 20/11/2022.

[3] K. T. Savjani, A. K. Gajjar **and** J. K. Savjani, "Drug solubility: Importance and enhancement techniques," *ISRN Pharmaceutics*, **jourvol** 2012, **page** 195 727, 5 **july** 2012, ISSN: 2090-6145. DOI: 10.5402/2012/195727. **url**: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3399483/ 20/11/2022.

[4] A.-R. Coltescu, M. Butnariu **and** I. Sarac, "The importance of solubility for new drug molecules," *Biomedical and Pharmacology Journal*, **jourvol** 13, **number** 2, **pages** 577–583, 25 **june** 2020. **url**: https://biomedpharmajournal.org/vol13no2/the-importance-of-solubility-for-new-drug-molecules/ 20/11/2022.

[5] "Factors influencing the solubility of drugs | pharmlabs." (), **url**: https://pharmlabs.unc.edu/labexercises/pharmaceutics/drugsolubilityfactors/ 20/11/2022.

[6] S. B. Bunally, C. N. Luscombe **and** R. J. Young, "Using physicochemical measurements to influence better compound design," *SLAS DISCOVERY: Advancing the Science of Drug Discovery*, **jourvol** 24, **number** 8, **pages** 791–801, 1 **september** 2019, Publisher: SAGE Publications Inc STM, ISSN: 2472-5552. DOI: 10.1177/2472555219859845. **url**: https://doi.org/10.1177/2472555219859845 20/11/2022.

[7] S. Black, L. Dang, C. Liu **and** H. Wei, "On the measurement of solubility," *Organic Process Research & Development*, **jourvol** 17, **number** 3, **pages** 486–492, 15 **march** 2013, Publisher: American Chemical Society, ISSN: 1083-6160. DOI: 10.1021/op300336n. **url**: https://doi.org/10.1021/op300336n 30/11/2022.

[8] C. A. S. Bergström **and** P. Larsson, "Computational prediction of drug solubility in water-based systems: Qualitative and quantitative approaches used in the current drug discovery and development setting," *International Journal of Pharmaceutics*, **jourvol** 540, **number** 1, **pages** 185–193, 5 **april** 2018, ISSN: 0378-5173. DOI: 10.1016/j.ijpharm.2018.01.044. **url**: https://www.sciencedirect.com/science/article/pii/S0378517318300632 16/10/2022.

[9] S. Lee, M. Lee, K.-W. Gyak, S. D. Kim, M.-J. Kim **and** K. Min, "Novel solubility prediction models: Molecular fingerprints and physicochemical features vs graph convolutional neural networks," *ACS Omega*, **jourvol** 7, **number** 14, **pages** 12 268–12 277, 4 **april** 2022, ISSN: 2470-1343. DOI: 10.1021/acsomega.2c00697. **url**: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9016862/ 06/11/2022.

[10] S. Boobier, D. R. J. Hose, A. J. Blacker **and** B. N. Nguyen, "Machine learning with physicochemical relationships: Solubility prediction in organic solvents and water," *Nature Communications*, **jourvol** 11, **number** 1, **page** 5753, **december** 2020, ISSN: 2041-1723. DOI: 10.1038/s41467-020-19594-z. **url**: http://www.nature.com/articles/s41467-020-19594-z 16/10/2022.

[11] M. Osada, K. Tamura **and** I. Shimada, "Prediction of the solubility of organic compounds in high-temperature water using machine learning," *The Journal of Supercritical Fluids*, **jourvol** 190, **page** 105 733, 1 **november** 2022, ISSN: 0896-8446. DOI: 10.1016/j.supflu.2022.105733. **url**: https://www.sciencedirect.com/science/article/pii/S0896844622002169 16/10/2022.

[12] "RDKit." (), **url**: https://www.rdkit.org/ 30/11/2022.

[13] "Mordred: A molecular descriptor calculator | journal of cheminformatics | full text." (), **url**: https://jcheminf.biomedcentral.com/articles/10.1186/s13321-018-0258-y 21/04/2023.

[14] "My RDKit cheatsheet," Towards Cheminformatics. (6 **january** 2021), **url**: https://xinhaoli74.github.io/blog/rdkit/2021/01/06/rdkit.html 29/11/2022.

[15] *Simplified molecular-input line-entry system*, **in** *Wikipedia* Page Version ID: 1120353002, 6 **november** 2022. **url**: https://en.wikipedia.org/w/index.php?title=Simplified_molecular-input_line-entry_system&oldid=1120353002 29/11/2022.

[16] Y. Jiang, G. Zhang, J. Wang **and** B. Vaferi, "Hydrogen solubility in aromatic/cyclic compounds: Prediction by different machine learning techniques," *International Journal of Hydrogen Energy*, **jourvol** 46, **number** 46, **pages** 23 591–23 602, 6 **july** 2021, ISSN: 0360-3199. DOI: 10.1016/j.ijhydene.2021.04.148. **url**: https://www.sciencedirect.com/science/article/pii/S0360319921015688 16/10/2022.

[17] "Standardization in step-by-step | chemaxon docs." (), **url**: https://docs.chemaxon.com/display/docs/standardization-in-step-by-step.md 30/11/2022.

[18] Z. Ye **and** D. Ouyang, "Prediction of small-molecule compound solubility in organic solvents by machine learning algorithms," *Journal of Cheminformatics*, **jourvol** 13, **number** 1, **page** 98, 11 **december** 2021, ISSN: 1758-2946. DOI: 10.1186/s13321-021-00575-3. **url**: https://doi.org/10.1186/s13321-021-00575-3 16/10/2022.

[19] D. S. Palmer, N. M. O'Boyle, R. C. Glen **and** J. B. O. Mitchell, "Random forest models to predict aqueous solubility," *Journal of Chemical Information and Modeling*, **jourvol** 47, **number** 1, **pages** 150–158, 1 **january** 2007, Publisher: American Chemical Society, ISSN: 1549-9596. DOI: 10.1021/ci060164k. **url**: https://doi.org/10.1021/ci060164k 16/10/2022.

[20] Y. Hou, S. Wang, B. Bai, H. C. S. Chan **and** S. Yuan, "Accurate physical property predictions via deep learning," *Molecules*, **jourvol** 27, **number** 5, **page** 1668, **january** 2022, Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1420-3049. DOI: 10.3390/molecules27051668. **url**: https://www.mdpi.com/1420-3049/27/5/1668 16/10/2022.

[21] P. G. Francoeur **and** D. R. Koes, "SolTranNet-a machine learning tool for fast aqueous solubility prediction," *Journal of Chemical Information and Modeling*, **jourvol** 61, **number** 6, **pages** 2530–2536, 28 **june** 2021, ISSN: 1549-960X. DOI: 10.1021/acs.jcim.1c00331.

[22] C. Deng, L. Liang, G. Xing **andothers**, "Multi-channel GCN ensembled machine learning model for molecular aqueous solubility prediction on a clean dataset," *Molecular Diversity*, 23 **june** 2022, ISSN: 1573-501X. DOI: 10.1007/s11030-022-10465-x.

[23] M. Galushka, C. Swain, F. Browne, M. Mulvenna, R. Bond **and** D. Gray, "Prediction of chemical compounds properties using a deep learning model," *Neural Computing and Applications*, **jourvol** 33, 1 **october** 2021. DOI: 10.1007/s00521-021-05961-4.

[24] *Graph neural network*, **in** *Wikipedia* Page Version ID: 1142323753, 1 **march** 2023. **url**: https://en.wikipedia.org/w/index.php?title=Graph_neural_network&oldid=1142323753 07/04/2023.

[25] Z. Xiong, D. Wang, X. Liu **andothers**, "Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism," *Journal of Medicinal Chemistry*, **jourvol** 63, **number** 16, **pages** 8749–8760, 27 **august** 2020, Publisher: American Chemical Society, ISSN: 0022-2623. DOI: 10.1021/acs.jmedchem.9b00959. **url**: https://doi.org/10.1021/acs.jmedchem.9b00959 30/11/2022.

[26] K. Yang, K. Swanson, W. Jin **andothers**, "Analyzing learned molecular representations for property prediction," *Journal of Chemical Information and Modeling*, **jourvol** 59, **number** 8, **pages** 3370–3388, 26 **august** 2019, Publisher: American Chemical Society, ISSN: 1549-9596. DOI: 10.1021/acs.jcim.9b00237. **url**: https://doi.org/10.1021/acs.jcim.9b00237 07/04/2023.

[27] M. C. Sorkun, A. Khetan **and** S. Er, "AqSolDB, a curated reference set of aqueous solubility and 2d descriptors for a diverse set of compounds," *Scientific Data*, **jourvol** 6, **number** 1, **page** 143, 8 **august** 2019, Number: 1 Publisher: Nature Publishing Group, ISSN: 2052-4463. DOI: 10.1038/s41597-019-0151-1. **url**: https://www.nature.com/articles/s41597-019-0151-1 16/10/2022.

[28] J. S. Delaney, "ESOL: estimating aqueous solubility directly from molecular structure," *Journal of Chemical Information and Computer Sciences*, **jourvol** 44, **number** 3, **pages** 1000–1005, 1 **may** 2004, Publisher: American Chemical Society, ISSN: 0095-2338. DOI: 10.1021/ci034243x. **url**: https://doi.org/10.1021/ci034243x 08/11/2022.

[29] Q. Cui, S. Lu, B. Ni **andothers**, "Improved prediction of aqueous solubility of novel compounds by going deeper with deep learning," *Frontiers in Oncology*, **jourvol** 10, 2020, ISSN: 2234-943X. **url**: https://www.frontiersin.org/articles/10.3389/fonc.2020.00121 17/10/2022.

[30] A. J. Preto, P. C. Correia **and** I. S. Moreira, "DrugTax: Package for drug taxonomy identification and explainable feature extraction," *Journal of Cheminformatics*, **jourvol** 14, **number** 1, **page** 73, 27 **october** 2022, ISSN: 1758-2946. DOI: 10.1186/s13321-022-00649-w. **url**: https://doi.org/10.1186/s13321-022-00649-w 10/04/2023.

[31] D. Butina, "Unsupervised data base clustering based on daylight's fingerprint and tanimoto similarity: A fast and automated way to cluster small and large data sets," *Journal of Chemical Information and Computer Sciences*, **jourvol** 39, **number** 4, **pages** 747–750, 26 **july** 1999, ISSN: 0095-2338. DOI: 10.1021/ci9803381. **url**: https://pubs.acs.org/doi/10.1021/ci9803381 07/12/2022.

[32] *Chemical fingerprint*, **in** *Wiktionary* Page Version ID: 41580495, 15 **november** 2016. **url**: https://en.wiktionary.org/w/index.php?title=chemical_fingerprint&oldid=41580495 14/03/2023.

[33] "Daylight theory: Fingerprints." (), **url**: https://www.daylight.com/dayhtml/doc/theory/theory.finger.html 07/12/2022.

[34] "T005 · compound clustering." (), **url**: https://projects.volkamerlab.org/teachopencadd/talktorials/T005_compound_clustering.html 21/11/2022.

[35] "ClassyFire: Automated chemical classification with a comprehensive, computable taxonomy | journal of cheminformatics | full text." (), **url**: https://jcheminf.biomedcentral.com/articles/10.1186/s13321-016-0174-y 29/04/2023.

[36] F. Trotta. "Understanding l1 and l2 regularization," Medium. (19 **february** 2023), **url**: https://towardsdatascience.com/understanding-l1-and-l2-regularization-93918a5ac8d0 24/04/2023.

[37] "The RBF kernel in SVM: A complete guide," PyCodeMates. (12 **december** 2022), **url**: https://www.pycodemates.com/2022/10/the-rbf-kernel-in-svm-complete-guide.html 08/03/2023.

[38] "What is a decision tree | IBM." (), **url**: https://www.ibm.com/uk-en/topics/decision-trees 07/04/2023.

[39] "Guide to random forest classification and regression algorithms," Serokell Software Development Company. (), **url**: https://serokell.io/blog/random-forest-classification 24/04/2023.

[40] *Gradient boosting*, **in***Wikipedia* Page Version ID: 1146274757, 23 **march** 2023. **url**: https://en.wikipedia.org/w/index.php?title=Gradient_boosting&oldid=1146274757 24/03/2023.

[41] G. Ke, Q. Meng, T. Finley **andothers**, "LightGBM: A highly efficient gradient boosting decision tree," **in***Advances in Neural Information Processing Systems* **volume** 30, Curran Associates, Inc., 2017. **url**: https://proceedings.neurips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html 24/03/2023.

[42] "LightGBM-light gradient boosting machine | by ashok kumar | medium." (), **url**: https://ashokkumar2216.medium.com/lightgbm-light-gradient-boosting-machine-eb83e1a2ca0a 24/04/2023.

[43] D. Jiang, Z. Wu, C.-Y. Hsieh **andothers**, "Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models," *Journal of Cheminformatics*, **jourvol** 13, **number** 1, **page** 12, 17 **february** 2021, ISSN: 1758-2946. DOI: 10.1186/s13321-020-00479-8. **url**: https://doi.org/10.1186/s13321-020-00479-8 07/04/2023.

[44] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre **andothers**, "Convolutional networks on graphs for learning molecular fingerprints," **in***Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2* **jourser** NIPS'15, Cambridge, MA, USA: MIT Press, 7 **december** 2015, **pages** 2224–2232. 10/04/2023.

# Chapter 10

# Appendix

## 10.1 Appendix 1: List of Packages and their Versions

| Package Name | Version |
|---|---|
| Numpy | 1.23.5 |
| Pandas | 1.5.2 |
| Tensorflow | 2.11.0 |
| Pytorch | 1.13.1 |
| Pytorch_geometric | 2.2.0 |
| RDkit | 2022.9.4 |
| DrugTax | 1.0.14 |
| Torch_Lightening | 2.0.1 |
| Lightgbm | 3.2.1 |
| Scikit-Learn | 1.2.1 |

Table 10.1: Essential Packages used in the project

## 10.2 Appendix 2: 2D Descriptors from RDKIT

| **Descriptor Name** | **Full Name** |
|---|---|
| MaxEStateIndex | Maximum E-state index |
| MinEStateIndex | Minimum E-state index |
| MaxAbsEStateIndex | Maximum absolute E-state index |
| MinAbsEStateIndex | Minimum absolute E-state index |
| qed | Quantitative Estimate of Drug-likeness |
| MolWt | Molecular weight |
| HeavyAtomMolWt | Heavy atom molecular weight |
| ExactMolWt | Exact molecular weight |
| NumValenceElectrons | Number of valence electrons |
| NumRadicalElectrons | Number of radical electrons |
| MaxPartialCharge | Maximum partial charge |
| MinPartialCharge | Minimum partial charge |
| MaxAbsPartialCharge | Maximum absolute partial charge |
| MinAbsPartialCharge | Minimum absolute partial charge |
| FpDensityMorgan1 | Morgan fingerprint density 1 |
| FpDensityMorgan2 | Morgan fingerprint density 2 |
| FpDensityMorgan3 | Morgan fingerprint density 3 |
| BalabanJ | Balaban's J index |
| BertzCT | Bertz's topological index |
| Chi0 | Chi0 index |
| Chi0n | Chi0n index |
| Chi0v | Chi0v index |
| Chi1 | Chi1 index |
| Chi1n | Chi1n index |
| Chi1v | Chi1v index |
| Chi2n | Chi2n index |
| Chi2v | Chi2v index |
| Chi3n | Chi3n index |
| Chi3v | Chi3v index |
| Chi4n | Chi4n index |
| Chi4v | Chi4v index |
| HallKierAlpha | Hall-Kier alpha |
| Ipc | Information content |
| Kappa1 | Kappa1 shape index |
| Kappa2 | Kappa2 shape index |
| Kappa3 | Kappa3 shape index |

| | |
|---|---|
| LabuteASA | Labute's accessible surface area |
| PEOE_VSA1 | Partial charges-based PEOE VSA descriptor 1 |
| PEOE_VSA10 | Partial charges-based PEOE VSA descriptor 10 |
| PEOE_VSA11 | Partial charges-based PEOE VSA descriptor 11 |
| PEOE_VSA12 | Partial charges-based PEOE VSA descriptor 12 |
| PEOE_VSA13 | Partial charges-based PEOE VSA descriptor 13 |
| PEOE_VSA14 | Partial charges-based PEOE VSA descriptor 14 |
| PEOE_VSA2 | Partial charges-based PEOE VSA descriptor 2 |
| PEOE_VSA3 | Partial charges-based PEOE VSA descriptor 3 |
| PEOE_VSA4 | Partial charges-based PEOE VSA descriptor 4 |
| PEOE_VSA5 | Partial charges-based PEOE VSA descriptor 5 |
| PEOE_VSA6 | Partial charges-based PEOE VSA descriptor 6 |
| PEOE_VSA7 | Partial charges-based PEOE VSA descriptor 7 |
| PEOE_VSA8 | Partial charges-based PEOE VSA descriptor 8 |
| PEOE_VSA9 | Partial charges-based PEOE VSA descriptor 9 |
| SMR_VSA1 | Surface-based SMR VSA descriptor 1 |
| SMR_VSA10 | Surface-based SMR VSA descriptor 10 |
| SMR_VSA2 | Surface-based SMR VSA descriptor 2 |
| SMR_VSA3 | Surface-based SMR VSA descriptor 3 |
| SMR_VSA4 | Surface-based SMR VSA descriptor 4 |
| SMR_VSA5 | Surface-based SMR VSA descriptor 5 |
| SMR_VSA6 | Surface-based SMR VSA descriptor 6 |
| SMR_VSA7 | Surface-based SMR VSA descriptor 7 |
| SMR_VSA8 | Surface-based SMR VSA descriptor 8 |
| SMR_VSA9 | Surface-based SMR VSA descriptor 9 |
| SlogP_VSA1 | SollogP-based VSA descriptor 1 |
| SlogP_VSA10 | SollogP-based VSA descriptor 10 |
| SlogP_VSA11 | SollogP-based VSA descriptor 11 |
| SlogP_VSA12 | SollogP-based VSA descriptor 12 |
| SlogP_VSA2 | SollogP-based VSA descriptor 2 |
| SlogP_VSA3 | SollogP-based VSA descriptor 3 |
| SlogP_VSA4 | SollogP-based VSA descriptor 4 |
| SlogP_VSA5 | SollogP-based VSA descriptor 5 |
| SlogP_VSA6 | SollogP-based VSA descriptor 6 |
| SlogP_VSA7 | SollogP-based VSA descriptor 7 |
| SlogP_VSA8 | SollogP-based VSA descriptor 8 |
| SlogP_VSA9 | SollogP-based VSA descriptor 9 |
| TPSA | Topological polar surface area |
| EState_VSA1 | E-state-based VSA descriptor 1 |

| | |
|---|---|
| EState_VSA10 | E-state-based VSA descriptor 10 |
| EState_VSA11 | E-state-based VSA descriptor 11 |
| EState_VSA2 | E-state-based VSA descriptor 2 |
| EState_VSA3 | E-state-based VSA descriptor 3 |
| EState_VSA4 | E-state-based VSA descriptor 4 |
| EState_VSA5 | E-state-based VSA descriptor 5 |
| EState_VSA6 | E-state-based VSA descriptor 6 |
| EState_VSA7 | E-state-based VSA descriptor 7 |
| EState_VSA8 | E-state-based VSA descriptor 8 |
| EState_VSA9 | E-state-based VSA descriptor 9 |
| VSA_EState1 | E-state-based VSA descriptor 1 |
| VSA_EState10 | E-state-based VSA descriptor 10 |
| VSA_EState2 | E-state-based VSA descriptor 2 |
| VSA_EState3 | E-state-based VSA descriptor 3 |
| VSA_EState4 | E-state-based VSA descriptor 4 |
| VSA_EState5 | E-state-based VSA descriptor 5 |
| VSA_EState6 | E-state-based VSA descriptor 6 |
| VSA_EState7 | E-state-based VSA descriptor 7 |
| VSA_EState8 | E-state-based VSA descriptor 8 |
| VSA_EState9 | E-state-based VSA descriptor 9 |
| FractionCSP3 | Fraction of sp3-hybridized carbons |
| HeavyAtomCount | Number of heavy atoms |
| NHOHCount | Number of hydroxyl groups |
| NOCount | Number of nitro groups |
| NumHeteroatoms | Number of heteroatoms |
| NumRotatableBonds | Number of rotatable bonds |
| NumValenceElectrons | Number of valence electrons |
| RingCount | Number of rings |
| MolLogP | MolLogP |
| MolMR | Molecular refractivity |
| fr_Al | Number of aldehydes |
| fr_Al_COOH | Number of carboxylic acids |
| fr_Al_OH | Number of alcohols |
| fr_Amine | Number of amines |
| fr_Aromatic | Number of aromatic rings |
| fr_Barbitur | Number of barbiturate groups |
| fr_Benzene | Number of benzene rings |
| fr_Benzodiazepine | Number of benzodiazepine rings |
| fr_Bicyclic | Number of bicyclic rings |

| | |
|---|---|
| fr_DIAZO | Number of diazo groups |
| fr_Dihydropyridine | Number of dihydropyridine rings |
| fr_Epoxide | Number of epoxide groups |
| fr_Ester | Number of ester groups |
| fr_Ether | Number of ether groups |
| fr_Furan | Number of furan rings |
| fr_Guaiacol | Number of guaiacol moieties |
| fr_Halogen | Number of halogen atoms |
| fr_Hdrzine | Number of hydrazine groups |
| fr_Imines | Number of imines |
| fr_Imidazole | Number of imidazole rings |
| fr_Imide | Number of imide groups |
| fr_Ketone | Number of ketone groups |
| fr_Lactam | Number of lactam groups |
| fr_Lactone | Number of lactone groups |
| fr_Methoxy | Number of methoxy groups |
| fr_Morpholine | Number of morpholine rings |
| fr_Nitrile | Number of nitrile groups |
| fr_Nitro | Number of nitro groups |
| fr_Phenol | Number of phenol groups |
| fr_Phenol_noOrthoHbond | Number of phenol groups with no ortho-hydrogen bond |
| fr_PhosAcid | Number of phosphonic acid groups |
| fr_PhosEster | Number of phosphoester groups |
| fr_Piperdine | Number of piperidine rings |
| fr_Piperzine | Number of piperazine rings |
| fr_PriAmine | Number of primary amines |
| fr_Pyridine | Number of pyridine rings |
| fr_QuatN | Number of quaternary nitrogen atoms |
| fr_TertAmine | Number of tertiary amines |
| fr_Tetrazole | Number of tetrazole rings |
| fr_Thiazole | Number of thiazole rings |
| fr_Thiophene | Number of thiophene rings |
| fr_I_1 | Number of Iodine atoms |
| fr_I_2 | Number of two Iodine atoms |
| fr_I_3 | Number of three Iodine atoms |
| fr_N_H | Number of Nitrogen atoms with Hydrogen atoms attached |
| fr_N_heterocycles | Number of Nitrogen atoms in heterocyclic rings |

| | |
|---|---|
| fr_N_oxide | Number of Nitrogen oxide groups |
| fr_NH1 | Number of primary Nitrogen atoms |
| fr_NH2 | Number of secondary Nitrogen atoms |
| fr_NH_noadjacent | Number of Nitrogen atoms with Hydrogen atoms attached that are not adjacent |
| fr_N_O | Number of Nitrogen atoms bonded to Oxygen atoms |
| fr_N_O_noadjacent | Number of Nitrogen atoms bonded to Oxygen atoms that are not adjacent |
| fr_Ndealkylation1 | Number of dealkylation sites for Nitrogen atoms with one carbon attached |
| fr_Ndealkylation2 | Number of dealkylation sites for Nitrogen atoms with two carbons attached |
| fr_Ndealkylation3 | Number of dealkylation sites for Nitrogen atoms with three carbons attached |
| fr_Nitroso | Number of nitroso groups |
| fr_NonCycle | Number of non-cyclic structures |
| fr_Not_H | Number of atoms that are not Hydrogen |
| fr_N_Ring | Number of Nitrogen atoms in rings |
| fr_N_S | Number of Nitrogen atoms bonded to Sulfur atoms |
| fr_Oxazole | Number of oxazole rings |
| fr_Oxime | Number of oxime groups |
| fr_Para_Hydroxylation | Number of para-hydroxylation sites |
| fr_Phenol | Number of phenol groups |
| fr_Primary_Amine | Number of primary amine groups |
| fr_Pyrazole | Number of pyrazole rings |
| fr_Pyridine | Number of pyridine rings |
| fr_Pyridium | Number of pyridium groups |
| fr_Pyrimidine | Number of pyrimidine rings |
| fr_QuatN | Number of quaternary Nitrogen atoms |
| fr_Secondary_Amine | Number of secondary amine groups |
| fr_Thiophene | Number of thiophene rings |
| fr_Urea | Number of urea groups |
| fr_Vinyl_Chloride | Number of vinyl chloride groups |
| fr_Zn | Number of Zinc atoms |
| fr_Al | Number of Aluminum atoms |
| fr_Bromine | Number of Bromine atoms |
| fr_Carbon_3 | Number of Carbon atoms with 3 bonds |
| fr_Carbon_4 | Number of Carbon atoms with 4 bonds |

| | |
|---|---|
| fr_Carbon_Any_NonTerminal | Number of Carbon atoms with any non-terminal bond |
| fr_Carbon_Terminal_Any _NonTerminal | Number of Carbon atoms with a terminal bond and any non-terminal bond |
| fr_Carbon_Terminal | Number of Carbon atoms with a terminal bond |
| fr_Chlorine | Number of Chlorine atoms |
| fr_Fluorine | Number of Fluorine atoms |
| fr_Iodine | Number of Iodine atoms |
| fr_N_H | Number of Nitrogen atoms with Hydrogen atoms attached |
| fr_N_H_noadjacent | Number of Nitrogen atoms with Hydrogen atoms attached that are not adjacent |
| fr_N_Ring | Number of Nitrogen atoms in rings |

## 10.3 Appendix 3: DrugTax Kingdoms and their Superclasses

The table and image below has been copied from the original drugTax paper to give the reader an idea of the different superclasses that can occur in the compound. To get more explanation on the different type of superclasses please refer to the original paper i.e [30]

| Kindgom | Compound Superclasses | Compound Name | Label in Figure |
|---|---|---|---|
| **Organic** | Organoheterocyclic | Imidazole (i) | |
| | Organosulphur | Glutathione | (ii) |
| | Lipid Molecule | Behenic Acid (Fatty Acid) | (iii) |
| | Allene | Fucoxanthin | (iv) |
| | Benzenoid | Benzene Hexacarboxylic Acid | (v) |
| | Phenylpropanoid | Phenylalanine | (vi) |
| | Organic Acid | Butyric Acid | (vii) |
| | Alkaloid | Morphine | (viii) |
| | Organic Salt | Acetate | (ix) |
| | Organohalogen | Acetyl Chloride | (x) |
| | Organometallic | Ferrocene | (xi) |
| | Organic Nitrogen | Pyrrole-2-Carboxylate | (xii) |
| | Nucleotide | Guanine | (xiii) |
| | Organic Oxygen | Ethanol | (xiv) |
| | Organophosphorus | Diethyl Phosphonate | (xv) |
| | Lignans and Neolignans | Matairesinol | (xvi) |
| | Organic Polymer | Starch | (xvii) |
| | Hydrocarbon | Octane | (xviii) |
| | Hydrocarbon Derivative | Ethanol | (xix) |
| | Organic Anion | Phosphate | (xx) |
| | Organic Cation | Choline | (xxi) |
| | Organic Zwitterion | Ammonium Propionate | (xxii) |
| | Carbene | Dichlorocarbene | (xxiii) |
| | Organic 1,3-Dipolar | Nitrone Molecule | (xxiv) |
| | Organopnictogen-N-(4-phenylamino-quinazolin-6-yl) | Acrylamide | (xxv) |
| | Acetylide | Lithium Acetylide | (xxvi) |
| **Inorganic** | Homogenous Metal | Cerium with Mixed Metals | (xxvii) |
| | Homogenous Non-Metal | Noble Gas Helium | (xxviii) |
| | Mixed Metal/Non-Metal | Potassium Nitrate | (xxix) |
| | Inorganic Salt | Sodium Chloride | (xxx) |
| | Miscellaneous Inorganic | Cyanide | (xxxi) |

Table 10.3: Kingdoms and Superclasses with Example Compounds - Table

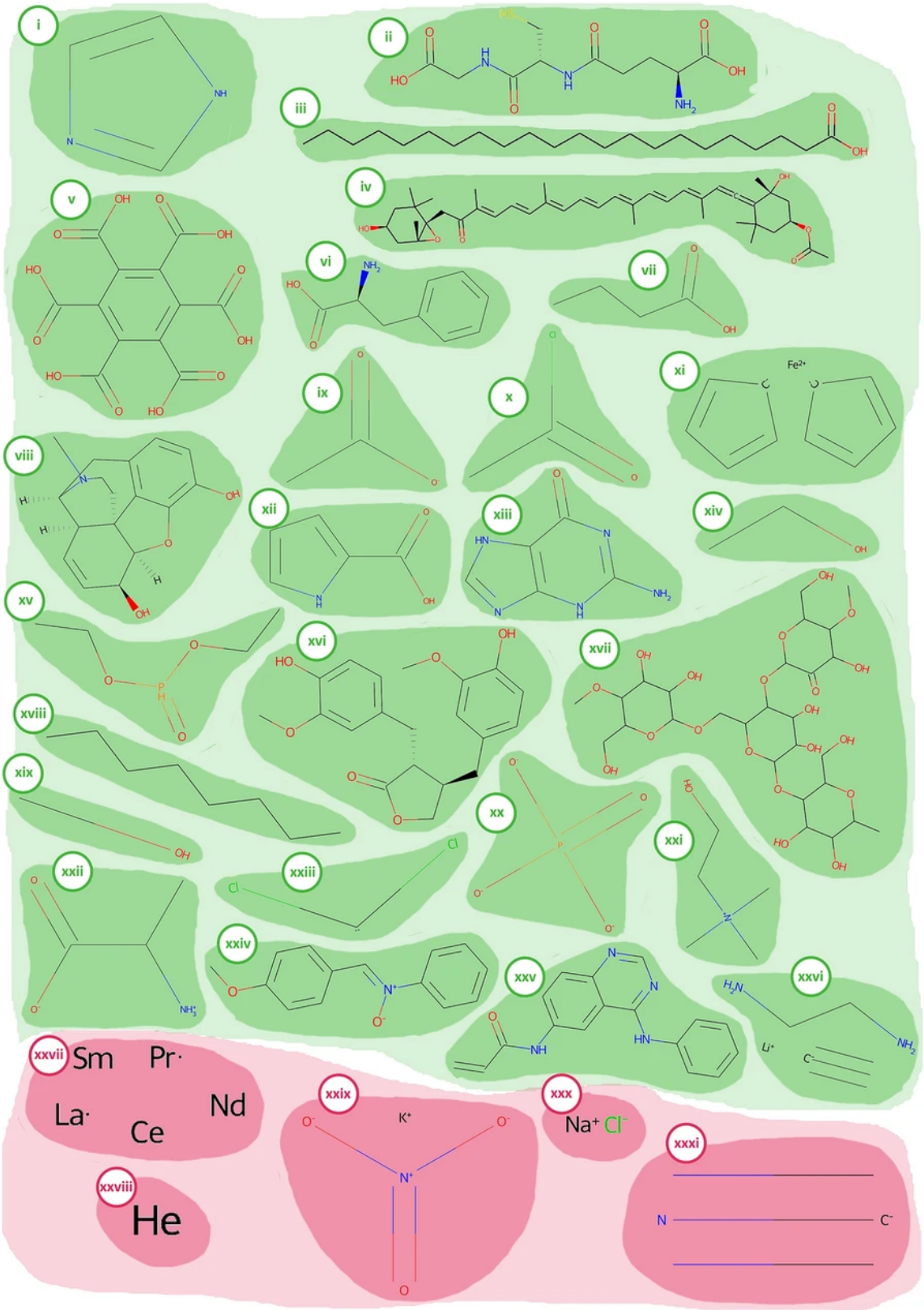



Figure 10.1: Kingdoms and Superclasses with Example Compounds - Figure

## 10.4 Appendix 4: Brief

# Problem

The solubility of a drug can affect how effective it is in the body. Consequently, if a drug is shown to be insoluble in the experimental phase, it becomes useless and the whole research process needs to start all over in order to find modifications to solve the issue or find a whole new compound to replace it. However, if the research process continues and the drug is absorbed poorly by the body, it could lead to serious side effect which could lead to severe digestive problems.
As a result, being able to predict the solubility of a compound to a certain level of accuracy improves the cost acquired and time taken to find the best possible compound for a drug. Additionally, by predicting the solubility of drugs enables us to figure out the best possible state in which a drug should be administered e.g as a capsule with liquid or a solid tablet

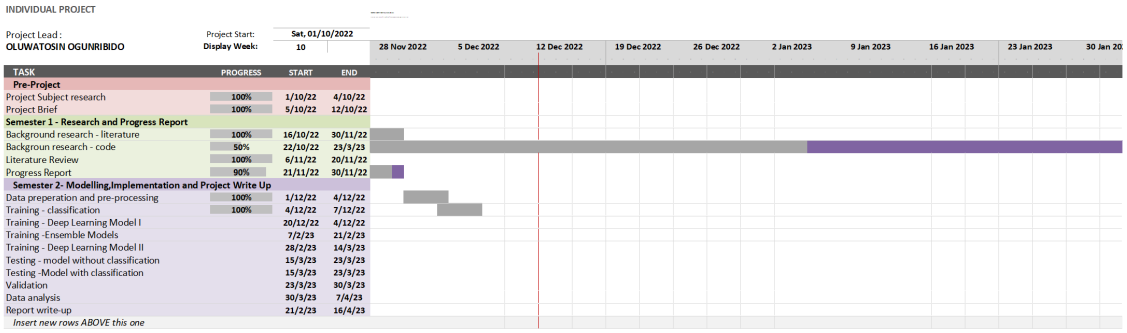
# Scope

The scope of this project will be mostly based on the primary dataset used and the machine learning algorithms implemented to design the model. The current dataset being used is the Huuskonen dataset which has the calculated values for solubility. This allows us to use it as a training dataset as they can be used as annotations during training. Additionally, it is also contains approximately a thousand compound which means it will be large enough to train and test the model.

# Goals

The core aim of this project is to build a model which can predict the solubility of compounds including drugs using machine learning. Initially, the model will use only compounds from the chosen dataset as training but during the testing phase other compounds will be tested such that the model does not know the solubility of them. The model of the project will be calculated by comparing current models used for predicting the solubility of chemical compounds and finding their limitations. By modifying the model to reduce the limitations, an improved model can be devised that aims to improve the measure of validation. Currently, the most common measure of validation used is the correlation between the experimental solubility and predicted solubility.

## 10.5 Appendix 4b: Previous Gant Chart



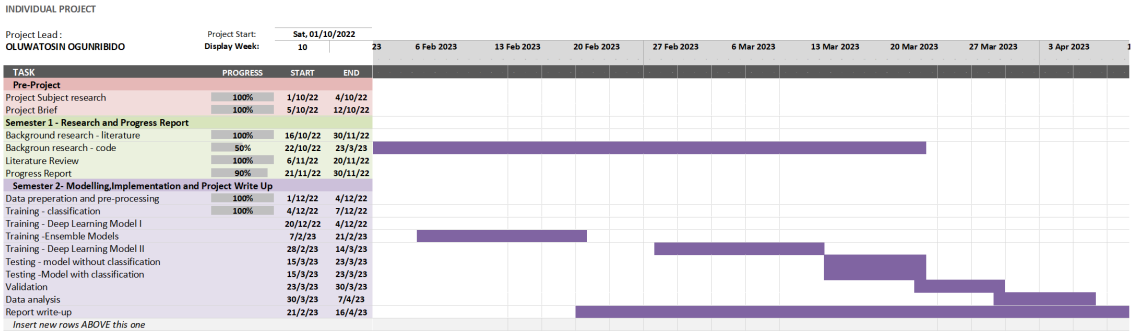
INDIVIDUAL PROJECT

Project Lead : OLUWATOSIN OGUNRIBIDO | Project Start: Sat, 01/10/2022 | Display Week: 10

| TASK | PROGRESS | START | END |
|---|---|---|---|
| **Pre-Project** | | | |
| Project Subject research | 100% | 1/10/22 | 4/10/22 |
| Project Brief | 100% | 5/10/22 | 12/10/22 |
| **Semester 1 - Research and Progress Report** | | | |
| Background research - literature | 100% | 16/10/22 | 30/11/22 |
| Backgroun research - code | 50% | 22/10/22 | 23/3/23 |
| Literature Review | 100% | 6/11/22 | 20/11/22 |
| Progress Report | 90% | 21/11/22 | 30/11/22 |
| **Semester 2- Modelling,Implementation and Project Write Up** | | | |
| Data preperation and pre-processing | 100% | 1/12/22 | 4/12/22 |
| Training - classification | 100% | 4/12/22 | 7/12/22 |
| Training - Deep Learning Model I | | 20/12/22 | 4/12/22 |
| Training -Ensemble Models | | 7/2/23 | 21/2/23 |
| Training - Deep Learning Model II | | 28/2/23 | 14/3/23 |
| Testing - model without classification | | 15/3/23 | 23/3/23 |
| Testing -Model with classification | | 15/3/23 | 23/3/23 |
| Validation | | 23/3/23 | 30/3/23 |
| Data analysis | | 30/3/23 | 7/4/23 |
| Report write-up | | 21/2/23 | 16/4/23 |
| *Insert new rows ABOVE this one* | | | |



Figure 10.2: Gantt Chart - December – February



INDIVIDUAL PROJECT

Project Lead : OLUWATOSIN OGUNRIBIDO | Project Start: Sat, 01/10/2022 | Display Week: 10

| TASK | PROGRESS | START | END |
|---|---|---|---|
| **Pre-Project** | | | |
| Project Subject research | 100% | 1/10/22 | 4/10/22 |
| Project Brief | 100% | 5/10/22 | 12/10/22 |
| **Semester 1 - Research and Progress Report** | | | |
| Background research - literature | 100% | 16/10/22 | 30/11/22 |
| Backgroun research - code | 50% | 22/10/22 | 23/3/23 |
| Literature Review | 100% | 6/11/22 | 20/11/22 |
| Progress Report | 90% | 21/11/22 | 30/11/22 |
| **Semester 2- Modelling,Implementation and Project Write Up** | | | |
| Data preperation and pre-processing | 100% | 1/12/22 | 4/12/22 |
| Training - classification | 100% | 4/12/22 | 7/12/22 |
| Training - Deep Learning Model I | | 20/12/22 | 4/12/22 |
| Training -Ensemble Models | | 7/2/23 | 21/2/23 |
| Training - Deep Learning Model II | | 28/2/23 | 14/3/23 |
| Testing - model without classification | | 15/3/23 | 23/3/23 |
| Testing -Model with classification | | 15/3/23 | 23/3/23 |
| Validation | | 23/3/23 | 30/3/23 |
| Data analysis | | 30/3/23 | 7/4/23 |
| Report write-up | | 21/2/23 | 16/4/23 |
| *Insert new rows ABOVE this one* | | | |



Figure 10.3: Gantt Chart - March – May

# List of Figures

# List of Tables

## 10.6 Archive Table of Contents

- IP
  - Datasets .. This folder contains all the original datasets extracted from the paper merged together in this project.
  - Features
    - featureExtractor.py ...This file adds graphs and classification given a dataset used for modelling.
    - featuriser.py ... This file merges multiple datasets and cleans values.
  - GCN-Files
    - featuriser grap.py This file encodes the SMILES into feature graphs and stores it into a dataset.
    - geometric-attentive-fp .py.... This file trains the attentivefp model based on the featuriser-graph data.
    - Attentive-FP-files
      - pytorch-dataset.py This file creates the InMemory Dataset for the models in this folder.
      - attentive-fp.py ..... This file trains the model without global descriptors.
      - combine-gcn.py .. This file implements the Linear Network used for the global descriptors.
      - classification-graph.py ..... This file trains the AttentiveFP model using global descriptors including the QSAR descriptors and classification.
      - gcn-no-descriptor.py This file trains the AttentiveFP model using just the classification descriptors.
  - ML-Models
  - images . This folder contains all the images extracted as output to visualise the performance of the different models.
  - Literature-Review Code .. This folder contains all the Jupyter notebooks and datasets used during the literature review.
  - processed This folder stores the data of the PyTorch-geometric InMemory Dataset used in training the model.
  - raw This folder stores the data required for training the Pytorch-geometric InMemory Dataset and Scikit-learn models.
  - Results ..... This folder stores the results of training and tuning the Scikit-Learn and LightGBM models.