Building Machine Learning Models to Explore Protein-Ligand Interactions for Drug Discovery

Daniel Catania

Supervisor: Dr Jean-Paul Ebejer

Faculty of ICT

University of Malta

May 2017

Submitted in partial fulfilment of the requirements for the degree of B.Sc. Computing Science (Hons.)

UNIVERSITY OF MALTA
FACULTY~~/INSTITUTE/CENTRE/SCHOOL~~ of ICT

DECLARATION OF AUTHENTICITY FOR UNDERGRADUATE STUDENTS

Student's I.D. /Code 368396M

Student's Name & Surname Daniel Catania

Course Bachelor of Science (Honours) (Computing Science)

Title of Long Essay/Dissertation

Building Machine Learning Models to Explore Protein-Ligand Interactions for Drug Discovery

Word Count 9710

I hereby declare that I am the legitimate author of this ~~Long Essay/~~Dissertation and that it is my original work.

No portion of this work has been submitted in support of an application for another degree or qualification of this or any other university or institution of higher education.

I hold the University of Malta harmless against any third party claims with regard to copyright violation, breach of confidentiality, defamation and any other third party right infringement.

Signature of Student

25/05/2017
Date

6.12.2016

Acknowledgements

This project would not have been possible without the help and support of many people around me. I would sincerely like to thank my supervisor Dr Jean-Paul Ebejer for his continuous support, patience and motivation throughout this project. I would like to thank my family and friends for believing in me and for the ongoing support during my studies.

Abstract:

Computational methods have become increasingly popular in drug discovery. In recent years, such methods have been used to aid physical experiments during the early stages of the drug discovery process, in order to reduce expense and time. Despite the usage of these methods, only a small increase in new molecular entities, over recent years, has been witnessed.

The aim of this final year project was to gain a better understanding of protein-ligand interactions, and to build machine learning models based on this understanding. These computational models help us identify small-molecules (also known as ligands) which interact with proteins, a necessary requirement for most medicinal drugs. These small-molecules interact, or bind, with proteins at some strength (binding affinity). This binding affinity is measured experimentally using three alternate measures (IC_{50}, K_i and K_d).

Presently, the only existing database of protein-ligand interactions is not publicly available, so we set out to build a database by mining and filtering proteins, bound with ligands, from the Protein Data Bank, in order to extract their interacting features. We augmented this database with experimental binding affinity data from other sources. We then built machine learning models to predict the binding affinities based on interactions between a protein and a ligand. We explored several machine learning approaches, including Nearest Neighbours, Support Vector Machines, Random Forest and Artificial Neural Networks to find the best model which describes this binding relationship.

While acknowledging that predicting binding affinity based on only feature interactions is hard, we also concluded that the model selection depended on the type of binding affinity measure. The protein-ligand interactions together with their respective binding affinity data have been made publicly available on Github and may allow for further research in this area.

Contents

List of Figures

List of Tables

Building Machine Learning Models to Explore Protein-Ligand Interactions for Drug Discovery

Daniel Catania*

Supervised by: Dr Jean-Paul Ebejer

May 2017

1 Introduction

During the past years, a small observable increase in the pharmaceutical productivity for discovering new drugs was witnessed. According to the FDA (U.S. Food and Drug Administration), the number of new molecular entities (NMEs) approved annually is of approximately 29 NMEs [Bre17, FDA17], as shown in Figure 1. Most small-molecule (or ligand) drugs work by binding to a larger molecule, known as a protein. We aim to build computational systems to further increase pharmaceutical productivity.

1.1 How do drugs work?

Many common drugs are small-molecules (< 100 atoms), also known as ligands. These small-molecules bind to proteins to exhibit an effect on other molecules [GILS10]. Proteins are macro-molecules (of a larger scale than ligands) which carry out a specific function and are made up of thousands of atoms.

An example of a protein is the *Neuraminidase Protein* [MBTH+03], which forms part of the Influenza Virus. Active small-molecules function by binding to the viruses' proteins (in this case *Neuraminidase*) in order to stop the virus from replicating. An example of an active small-molecule would be *Oseltamivir*, which is mainly known by its trade name *Tamiflu*. Another more popular example is aspirin binding to a human protein (shown in Figure 2).

*Submitted in partial fulfilment of the requirements for the degree of B.Sc. Computing Science (Hons.).

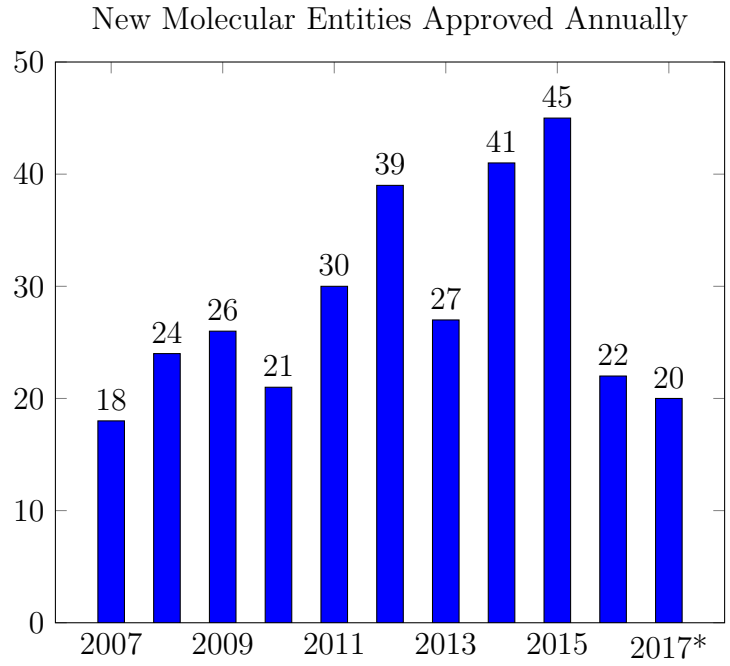


Figure 1: Statistics according to the FDA (U.S. Food and Drug Administration) showing the NMEs Approved Annually [Bre17]. (2017 shows the number of drugs approved until 6th May 2017.)

1.2 Motivation

The motivation of this study was to predict the binding affinity of protein-ligand interactions as these are crucial for drug discovery. The predictions were built by mining existing interaction data and applying machine learning models. These predictions aid in identifying whether a ligand is suitable to bind with a protein.

1.3 Why is Drug Discovery Challenging?

Drug discovery is considered challenging because biology is complex in its nature [Bun11]. It is a classic multiple variable optimisation problem having the variables being re-optimised belonging to a complicated, poorly known system.

The drug discovery pipeline is constantly improving with the use of innovative technologies, in order to narrow down possibilities to the most favourable compounds for clinical testing. *In silico* studies aid in expediting this drug discovery process, mostly by reducing the number of *in vitro* (in test-tube) and *in vivo* (animal) testing [GCB17].

In silico studies are also referred to as Computer-Aided Drug Design (CADD) methods.

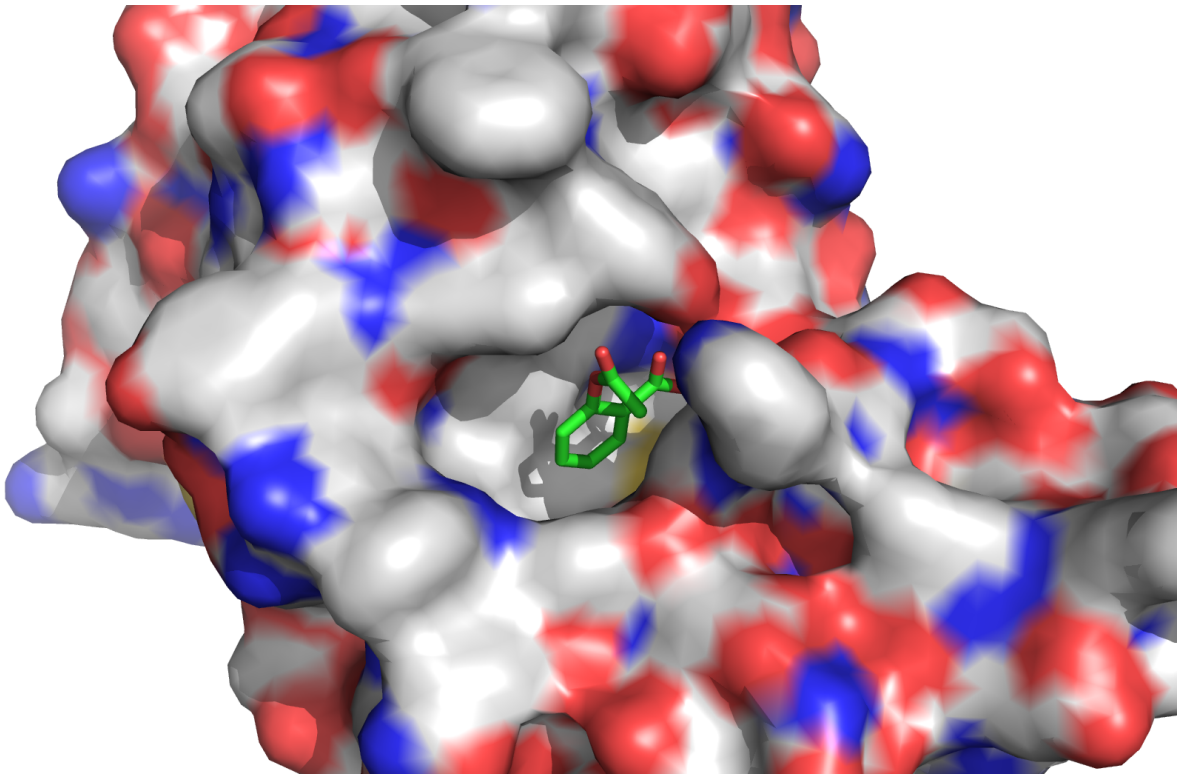


Figure 2: 1OXR - Aspirin binding to the Protein *Phospholipase A2* showing the Aspirin in Green and the Protein's surface in White/Red/Blue.

CADD methods are computer modelling techniques which are used to predict whether a small-molecule will bind to a target molecule and if this happens it calculates the binding strength between the two entities. CADD methods are possible because of the huge volume of data available regarding protein structures and small molecules, which improve the accuracy of these methods [Ebe14].

Computational models offer a cheap and fast approach compared to the traditional methods of drug identification and discovery. One CADD method is Virtual Screening. Virtual screening is the use of computers to discover small-molecule inhibitors which have a therapeutic effect against a certain condition/disease.

1.4 Related Work

CREDO is a database of protein-ligand interactions. It comprises of inter-atomic contacts to be expressed as structural interaction fingerprints (SIFts), and presents key features, vital to drug discovery [SB09]. A SIFt is a method for describing and evaluating 3D

target molecule-ligand intermolecular interactions [SCD04]. The CREDO database is not publicly available, so we mined existing repositories to extract protein-ligand interactions.

A database of binding data, consisting of matching protein-ligand interactions together with their binding affinity, does not exist, therefore such a database was created. This was achieved by merging the database containing protein-ligand interaction features together with the PDBbind database which contains the binding affinity of protein-ligand interactions.

1.5 Aims & Objectives

The aim of this thesis was to gain a better understanding of protein-ligand interactions and build models based on this understanding to predict protein-ligand binding affinities. The objectives of this thesis were to:

- Mine the Protein Data Bank for proteins containing ligands to extract protein-ligand interactions;
- Build a database of protein-ligand interactions containing binding data; and
- Build and train machine learning models to predict the binding affinity of protein-ligand interaction features.

1.6 Our Approach

The protein data bank (PDB) is a database of 3D protein structures [BWF$^+$00]. The archived data was downloaded, extracted and cleaned to keep only the 3D structures which contained ligands. Following was the stage of standardisation, which standardises and removes inconsistent data from the 3D structures. The next stage was for the data to be filtered down, according to a ligand's drug-likeness. The data was then refined down to structures containing interaction features in a molecule. A binding data database was then created by intersecting the structures containing interaction features in a molecule, with structures having a defined binding affinity found in the PDB bind Database (this is a database of experimentally measured binding affinities) [LLH$^+$14, WFL$^+$05, WFLW04]. This database was then used to build and train machine learning models to predict the binding affinity for new protein-ligand complexes. The machine learning models considered were the Nearest Neighbours, Support Vector Machines, Random Forest and Artificial Neural Networks.

1.7 Document Structure

Chapter 2 (Background) gives an overview of the concepts used for feature extraction, and also describes the machine learning methods used to predict the binding affinity of protein-ligand interactions. Chapter 3 (Design) presents an in depth description of the features being observed and goes through the process of data acquisition. Chapter 4 (Implementation) exhibits how big data was dealt with, and demonstrates the in depth process of data gathering and cleaning to extract the features. The parameters for the machine learning models implemented were further described in this chapter. In Chapter 5 (Evaluation) the results from the models built are evaluated against the proper results. Chapter 6 (Conclusion) outlines some of the possible future works to improve the prediction of binding affinities. Concluding statements with regards to this study were also presented in this chapter.

2 Background

This section of this study will present an overview of the concepts used in order to explore and extract protein-ligand interactions. Also, it will provide sufficient information to understand the machine learning models which were used to predict the binding affinity of protein-ligand interactions.

2.1 Protein-Ligand Interactions

Ligands bind with proteins in a *lock and key fashion* in order to stop or alter their functionality, as shown in Figure 3. These small-molecules work in two definitive methods of action, which are classified as agonist and antagonist small-molecules. An agonist small-molecule elicits a response, whilst an antagonist small-molecule blocks a response [AS59].

An example of a protein-ligand interaction is the attraction between a positive charge on the protein and a negative charge on the ligand side, when in close proximity to each other.

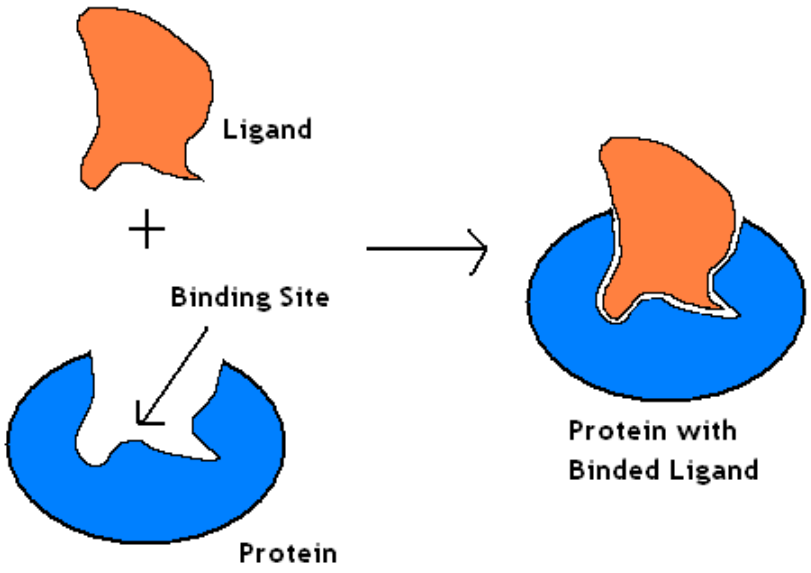


Figure 3: A schematic diagram showing how a Ligand binds to a Protein at the Binding Site [Wik08].
A ligand binds to a protein with a particular strength (called binding affinity).

2.2 Lipinski's Rule of Five

Lipinski, a medicinal chemist, studied the properties of over 2,000 drugs and possible drugs in clinical trials [LLDF97, Lip04]. From this study, he concluded that a compound is likely to diffuse and be absorbed by a living organism if it matches the criteria of the rule of five.
Such a typical compound would have:

- Molecular weight less than 500;
- Lipophilicity (the logarithm of the partition coefficient between water and 1-octanol) smaller than 5;
- A maximum of 5 groups that can donate hydrogen atoms to form hydrogen bonds; and
- A maximum of 10 groups that can accept hydrogen atoms to form hydrogen bonds.

The overall drug-likeness of a small-molecule was predicted through the use of this method, so as to keep only those structures containing small-molecules which have properties of drugs. Hence, the database being built would only contain binding data of protein-ligand interactions of ligands which qualify to become drugs.

2.3 Computational Methods in Drug Discovery - Virtual Screening

Virtual screening is a field used to find new active molecules either through using ligand-based virtual screening (LBVS) or structure-based virtual screening (SBVS). LBVS uses information from known binders to find different inhibitors, without considering the target protein structure [Sho04]. On the other hand, SBVS uses a protein structure, a library of small molecules we want to test and a docking protocol to find new active-molecules. Virtual screening could be applied at an early stage in the drug discovery process and has the advantage of being cheap and fast [AS05]. This study focuses on improving SBVS systems by using interaction data. The 3D structures of proteins and small-molecules were analysed for common features between them. These features were extracted into a database.

2.4 Machine Learning Models

A machine learning problem could be classified as one of the following:

- Inductive learning - Assuming that if a large number of items have a particular property, then all items have this property.
- Deductive learning - Problems that could also be referred to as provable conclusions, and these could take the shape of Hypothesis/Conclusion, If/Then or Condition/Action;
- Supervised learning - problems train using training sets. These training sets contain pairs of a training instance (x) together with a function used to classify the same training instance ($c(x)$). These supervised learning problems could be posed as:
  - Classification - output variable takes class labels; and
  - Regression - output variable takes continuous values.
- Unsupervised learning - A structure is found by training given a set of unlabelled instances.

2.4.1 Nearest Neighbours Classification

The nearest neighbours is a classification learning problem [PVG+11]. It is the simplest of all learning algorithms to predict the output of a test example [Elk11]. The nearest neighbours algorithm simply stores all the instances of the training set, with both the attributes and output result, other than attempt to construct a general model using the same data. A test example is predicted by computing, using a distance metric (ex. Euclidean distance), and keeping the closest k training examples, where k is a positive integer. The output of the training example with the most common attributes is the prediction for this test example. This type of classification is not recommended with use of more than 20 dimensions, but in the case of this thesis, this would suffice since the number of features is significantly smaller than 20. An example of the nearest neighbours classification is shown in Figure 4.

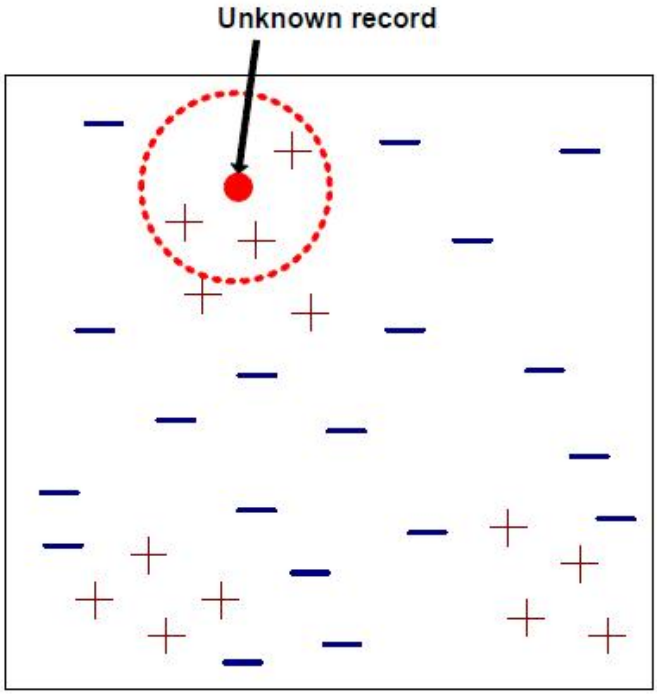
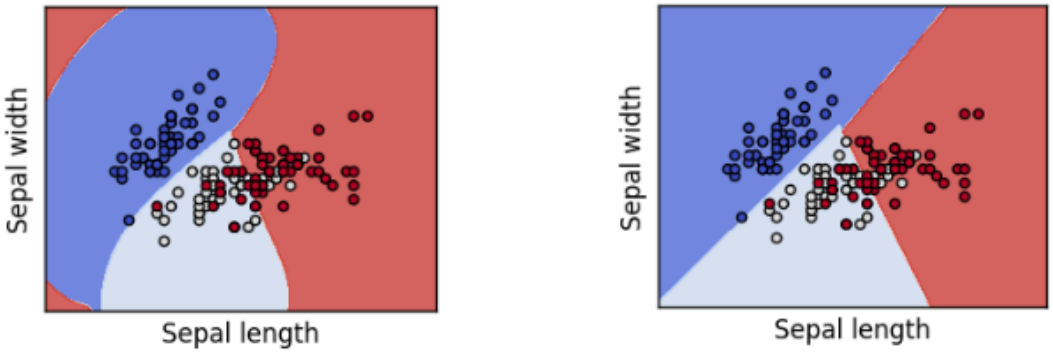


Figure 4: Nearest Neighbours - Schematic illustration showing the nearest 3 neighbours to a new data point. The new data point would be classified as positive since the nearest 3 neighbours are positive [PVG+11].

2.4.2 Support Vector Machine Classifier

A popular classification technique is the support vector machine classifier (SVC). The aim of a SVC is to train a model using the training data's attributes, also known as training vectors, and target values. This model is then used to predict the outcome values of test data using just the test data attributes [HCL+03]. A higher dimensional space is then formulated with the use of the training vectors. A linear separating hyperplane is then found, using the SVC, with the maximal margin in the previously mentioned higher dimensional space. SVC works using different kernels, two of which are the radial basis function (RBF) and linear kernels. A radial basis function kernel non-linearly maps samples, making it able to handle the instance when a non-linear relation exists between attributes and class labels. A special case of the RBF kernel is the linear kernel, since a linear kernel with a penalty parameter $\tilde{C}$ has a similar performance to the RBF kernel with some parameters (C, γ) [KL03]. An example of an SVC with a RBF kernel is shown in Figure 5a, whilst an example of an SVM with a linear kernel is shown in Figure 5b.



(a) RBF Kernel

(b) Linear Kernel

Figure 5: Support Vector Machine Classifiers - Schematic illustrations showing the SVC with both a RBF and a Linear Kernel [PVG+11].

2.4.3 Random Forest Classifier

Random Forest is an ensemble method that enhances the efficiency of Decision Tree-based algorithms [SLT+03]. Over other ensemble methods, random forest provides some exceptional features that include similarity measure between molecules, prediction accuracy estimation, and measures of descriptor importance.

A collection of B decision trees $\{T_1(X), ..., T_B(X)\}$ make up a Random Forest, where $X = \{x_1, ..., x_p\}$ is a p-dimensional vector of molecular descriptors or characteristics associated with a molecule. The collection generates B outputs which are combined to construct one final prediction. The class predicted by the majority of trees is the final prediction for classification problems. An example of a random forest classification is shown in Figure 6.

The algorithm was proved to be very efficient in most cases. In contrast with the usual tree growing algorithm, random forest usually tests the square root of the descriptors for classification for their splitting performance at each node, while the usual tree growing algorithm tests all the descriptors. Another factor that effects performance is that random forest does not do any pruning at all. It was also proved that, in cases where there are an abundant number of features, random forest could be trained in less time than single decision trees.

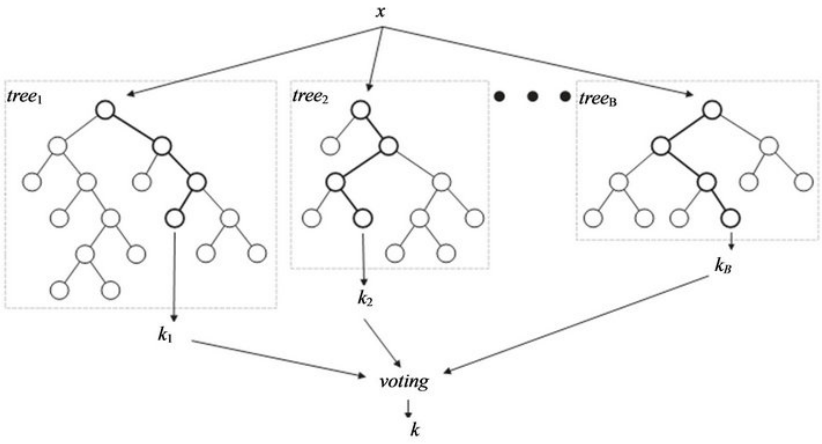


Figure 6: Random Forest - Schematic illustration showing that new data is classified by B trees, and then the class with the majority in the set of B outputs is chosen to be the final prediction [NWN13].

2.4.4 Artificial Neural Networks

Another machine learning algorithm developed from the idea of simulating the human brain is the Artificial Neural Networks (ANN) [ZHS09]. The ANN is an interconnection of nodes, similar to neurons. The three critical elements of a neural network are:

- Node character - establishes the way that signals are processed by the node,
- Network topology - establishes how the nodes are organised and connected,
- Learning rules - establish the way that the weights are initialised and adjusted.

A simple model for a node in the ANN is shown in Figure 7. Every node in the ANN receives multiple inputs from other nodes via connections. These connections have weights associated with them. When the threshold value of the node is exceeded by the weighted sum of inputs, the node activates and passes the signal through a transfer function. This function then sends the signal to neighbouring nodes.

The ANN we used in this study uses the multi-layer perceptron (MLP) algorithm with backpropagation training (shown in Figure 8). The algorithm works by first propagating the input through the network to get a calculated output. The cost function, which is the error between the correct and calculated output, is then propagated backwards, from the output to the input, so as to modify the weights to correct the error. A gradient descent method is then used to minimise the cost function, in order to be able to apply MLP to networks with differentiable transfer functions.

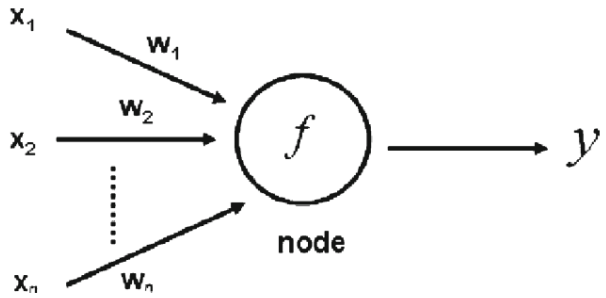


Figure 7: Artificial Neural Network: A basic model of a single node where x_i = input, w_i = weight, f = transfer function, and y = output [ZHS09].

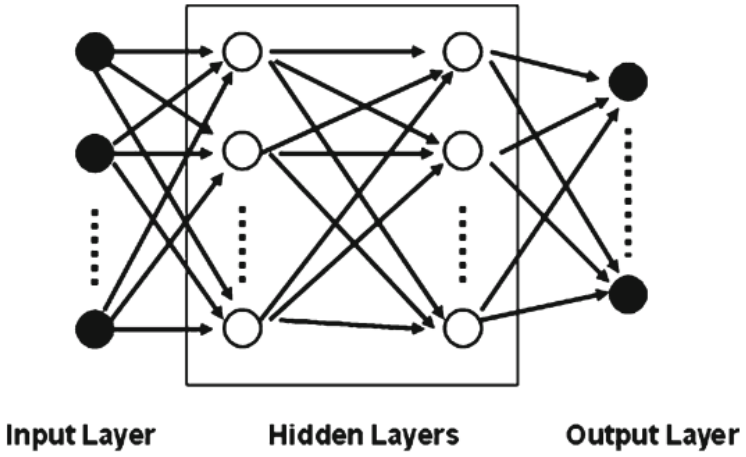


Figure 8: Multiple-Layer Perceptron [ZHS09].

2.5 Evaluating Machine Learning Models

2.5.1 Balanced and Unbalanced Datasets

Data is often skewed because of unbalanced datasets [Ols05]. Unbalanced datasets have many entries matched to a class or two, with the remainder belonging to other classes. Problems arise when predicting the class for new data, as there is a high probability that it is classified to a big class. A solution to this would be the balancing of datasets. A dataset can be balanced by determining the number of entries for every class, and randomly keeping this number of records for these classes. Therefore, all the classes would have an equal number of entries resulting in a higher probability of predicting the correct class of new data. Unfortunately, balancing could leave out important records for a class.

2.5.2 Performance Measures

There are various models that are model validation techniques [RB99]. Hold-out and cross-validation methods are two such methods that were used to evaluate the models built.

2.5.2.1 Hold-out

A common method that estimates the performance of machine learning models is the hold-out method. A dataset is randomly split into three sets: a training set, a validation set and a testing set. The machine learning model is first trained on the training set. The validation set is then used to cross-validate the trained model so as to be able to tweak the performance of these models. The testing set is then used to test the trained model by predicting the labels for the testing set's data, and then matching these labels to the actual labels in the same set. The results obtained vary on every run as these depend on the random subdivision of the original dataset into training, validation and testing sets. The results from this method could also be overestimated since not all of the original dataset was used in the training of the model.
In this study, the datasets were split into three sets of 60%, 20% and 20%, being the training, validation and testing sets respectively. The measurements observed are described in Table 1.

2.5.2.2 Cross-validation

A better estimation method is the cross-validation. Cross-validation could be distinguished by two types, exhaustive and non-exhaustive cross-validation [K$^+$95]. Exhaustive cross-validation methods compute all possible ways of splitting the validation set. On the other hand, non-exhaustive cross-validation methods do not learn and test on all possible variants of dividing the validation set. This study focuses on the use of a non-exhaustive cross-validation method.

2.5.2.3 k-fold cross-validation

A non-exhaustive cross-validation method is the k-fold cross-validation. This method splits a validation set into k subsets of roughly equal size. A model is then trained for k times, every time omitting one of the subsets to be used as the testing set, with the rest being used as the training set. The average accuracy of the k runs is the accuracy estimation of that model.

Measure	Equation	Definition
Error	$\frac{FP+FN}{FP+FN+TP+TN}$	Error is a calculation of a dissatisfactory model. This gives an indication of all predictions that are incorrect.
Accuracy	$\frac{TP+FN}{TP+FN+TP+TN}$	Accuracy is a computation of how reliable a model is. This gives the proportion of all predictions that are correct.
Precision	$\frac{TP}{TP+FP}$	Precision is an outline of how many positive predictions were actual positive observations. This gives the proportion of all positive predictions that are correct.
Recall	$\frac{TP}{TP+FN}$	Recall is an evaluation of how many actual positive observations were predicted correctly. This gives the amount of all real positive predictions that are true.
F1 Score	$2\frac{Precision*Recall}{Precision+Recall}$	F1 Score is an assessment of a test's preciseness based on both the Precision and Recall methods. The score is the harmonic mean of precision and recall.

Table 1: Performance Measures where TP - True Positive;
FN - False Negative; TN - True Negative; FP - False Positive [MCM13].

2.6 Conclusion

In this section, an in depth background required to understand protein-ligand interactions and machine learning models was presented. The rules used to virtually tag small-molecules as drug-like and the field used to screen such small-molecules were also presented. The machine learning models built in this study were also described to present what they offer. These concepts are crucial to the design of the new database and the models used for prediction.

3 Design

The aim of this thesis is to build machine learning models to predict protein-ligand interactions in drug discovery. The models will be based on the database containing binding data that once built would be split into 60%, 20% and 20% splits, being the training, validation and testing datasets respectively. This study was conducted to computationally predict the binding affinity that a small-molecule will bind to a particular protein given the number of interactions between them. The effectiveness of this study in predicting the correct binding affinity will in turn aid the present day drug discovery process. A successful result would offer a cheap and fast alternative approach when compared to the traditional methods of drug identification and discovery. One of the objectives of this study was to create a publicly-available database, containing binding data together with protein-ligand interactions.

3.1 Features for Model Building

The outcome of this study is to build models to predict the binding affinity between a protein and a ligand depending on their interacting features. These interacting features are the result of a protein's interaction points, also known as pharmacophores, being in close proximity to a ligand's interaction points. A pharmacophore could be described as the important geometric arrangement of atoms to produce a given biological response [MNW98]. A brief description of some pharmacophores are as follows:

- Hydrophobe - a pharmacophore that has a repulsive force for water molecules;
- Hydrogen Bond Donor - a usually strong electronegative atom which bonds to a hydrogen atom when participating in a hydrogen bond;
- Hydrogen Bond Acceptor - a neighbouring electronegative molecule holding a lone electron pair so as to bond with a hydrogen bond donor;
- Cation - a positive charged atom;
- Anion - a negative charged atom;
- Aromatic - an organic pharmacophore that contains one or more benzene or equivalent heterocyclic rings.

The interacting features being considered in this study are shown in Table 2.

Receptor Pharmacophore	Ligand Pharmacophore	$\leq$ Distance $\mathring{A}$
Hydrophobe	Hydrophobe	4.5
Acceptor	Donor	3.9
Donor	Acceptor	3.9
Cation	Anion	4.0
Anion	Cation	4.0
Aromatic	Aromatic	4.5
Aromatic	Cation	4.0
Cation	Aromatic	4.0

Table 2: Interaction definition between Protein and Ligand Pharmacophores [Ebe14].

3.1.1 Data Acquisition

First, the protein 3D structures were downloaded from the Protein Data Bank (PDB) and extracted. These structures were then filtered to keep only structures which contained small-molecules. Also, since the interactions between the small-molecules were being explored, only residues (an atom or group of atoms forming part of a molecule), having at least one atom which is less than or equal to 12 angstroms away from a point of the small-molecule ($d \leq 12$), were extracted. An Angstrom is a unit of length which is equal to 1×10^{-10} of a metre. The stage of standardisation followed. This standardises and removes inconsistent data from the 3D structures. Furthermore, the structures were filtered to those which contained small-molecules obeying Lipinski's rules. In addition, the structures containing Pharmacophores on both the protein and small-molecule side of a particular structure were kept. These pharmacophore interactions were extracted with the use of the *RDKit* library. When given a molecule, the *RDKit* returns back the features together with their 3D coordinates. The pharmacophore interactions mined are defined in Table 2. Moreover, the structures were then intersected with the structures in the PDBbind database, and only the matched structures were used to build the database containing binding data.

The primary data contained 127,562 structures. When the data was not standardised halfway through the process, the new database built contained 4,739 structures, whereas when it was standardised, the new database contained 5,416 structures. A breakdown of the process of getting features is shown in Figure 9.

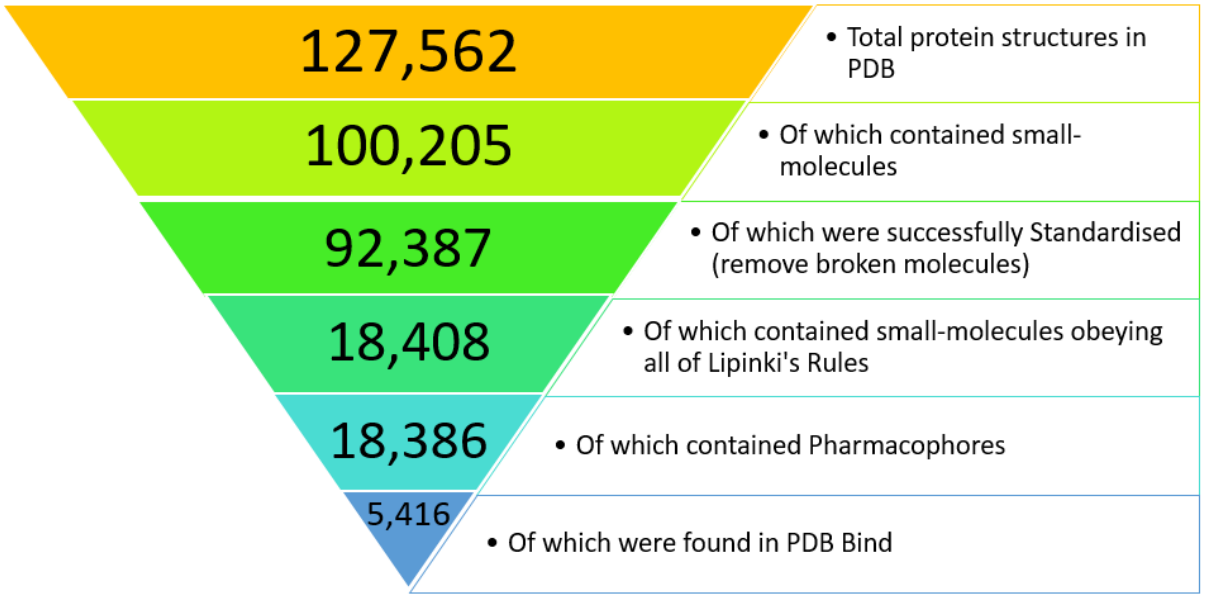


Figure 9: Process of Getting Features with Standardisation.

3.2 Validation

The processing of huge amounts of data required data to be validated so as to verify and keep track of information. The successful loading of files by the libraries used to mine the information needed also acted as a confirmation that these files were in a genuine format. The machine learning models built were also validated by the use of scores and cross-validation, as described in Section 2.4.5.

3.3 Conclusion

This section presented the features being observed. The process of data acquisition was also explained together with how this was validated. The presented approaches are crucial to the implementation of this study.

4 Implementation

The Python programming language (version 3.5), on a Linux Machine running *Ubuntu 16.04*, was used to carry out this study. Python is known to be a very expressive language whilst also being very easy to understand and code. The advantage of using this language is that it is a cross-platform language [Sum10]. This means that a program should run correctly on any system, given that the used libraries in the program are available for the system and installed. This study involved the processing of large volumes of data, so it was decided to use Python due to its powerful language capabilities.

4.1 Dealing with Big Data

The data cleaning aspect of this study involved dealing with 127,562 structures, amounting to more than 216GB of processed data. The Python language offers the threading and multiprocessing modules to aid in dealing with amounts of data. Multi-threading allows for memory to be shared, is lightweight as it has a low memory footprint, and runs all threads on the same core at the apparent same time (concurrency) [AR06]. A disadvantage of threading is that it is subject to the Global Interpreter Lock (GIL), which locks resources in order for multiple threads to write to the same memory space at the same time. On the other hand, multiprocessing allows for a process to be opened for every execution, which results in extra overhead and a larger memory footprint. It also takes advantage of multiple cores, implements parallelism (multiple tasks running at the same time), and avoids the GIL interpretation.

Multiprocessing was used in this study together with the concepts of a queue and a process manager. A queue was executed as a process on its own, so that a process could message back the structure name of a successful structure that was cleaned. Since a process was created for every structure being executed, a process manager was deemed necessary. This process manager would create all the processes as jobs to do, and then allow a specific number of processes to be run simultaneously. The number of simultaneous processes depends on how many cores the machine being run on offers. The manager was made to utilise all cores with the exception of one thread for the host operating system to run on. Through multiprocessing, the time taken to process the structures amounted to around three days.

4.2 Data Acquisition

The raw data used in this study was acquired from the Protein Data Bank (PDB). It is a worldwide repository of structural data of biological macromolecules. The repository started off containing just seven structures in 1971. After this date, a dramatic increase in structures added to the archive was noticed due to the improved technology for the discovery of structures. As at 1^{st} March 2017, this repository contained 127,562 structures.

The PDB File Transfer Protocol (FTP) Archive was used in order to clone the current contents of the data directory containing the PDB format 3D coordinates. This data served as the primary basis of this project. The PDB format is a standard for files having atomic coordinates [Lab14]. It consists of lines of information, also known as records, in a text file. A PDB file normally consists of records, organised in a particular order to describe a structure. The record types that pertain to this study are as follows:

- ATOM records - atomic coordinate records containing the 3D coordinates for atoms that form part of the protein, also known as standard residues;
- HETATM records - atomic coordinate records containing the 3D coordinates for atoms in small-molecules, also known as non-standard residues;
- TER records - indicate the end of a chain of residues.

A sample of the contents of a PDB File is shown in Listing 1, showing examples of ATOM, HETATM and TER records. The general format of the ATOM and HETATM records are shown in Figure 15 (Appendix A).

4.3 Mining and Data Cleaning

Part of this study involved the cleaning of the structural data in order to extract the useful information pertaining to this study. The following are the tools/packages used to achieve this:

- **Biopython** - The Biopython libraries are a collection of tools for computational biology [CC00]. The philosophy behind the existence of these libraries is to reduce the time wasted on developing the same tools over and over again, whilst making the code more robust due to many diverse developers using these same libraries. The Biopython module used in this study is the *Bio.PDB* module. This module supplies a PDB file parser and macromolecular structure functions [CAC+09]. This module will aid in extracting the required data to gather the protein-ligand interactions.

Listing 1: 1OXR PDB File Contents Sample.

```
HEADER    HYDROLASE                                   03-APR-03   1OXR
TITLE     ASPIRIN INDUCES ITS ANTI-INFLAMMATORY EFFECTS THROUGH ITS
TITLE    2 SPECIFIC BINDING TO PHOSPHOLIPASE A2: CRYSTAL STRUCTURE OF
TITLE    3 THE COMPLEX FORMED BETWEEN PHOSPHOLIPASE A2 AND ASPIRIN AT
TITLE    4 1.9A RESOLUTION
...
ATOM      1  N   ASN A   1       7.648  22.502   9.899  1.00 15.18           N
ATOM      2  CA  ASN A   1       8.189  21.131  10.221  1.00 16.24           C
ATOM      3  C   ASN A   1       7.733  20.099   9.194  1.00 16.63           C
ATOM      4  O   ASN A   1       6.870  20.392   8.351  1.00 16.13           O
ATOM      5  CB  ASN A   1       7.863  20.711  11.663  1.00 16.53           C
...
TER
HETATM  915  O1  AIN A 141      13.907  16.130   0.624  0.50 28.52           O
HETATM  916  C7  AIN A 141      13.254  15.778   1.723  0.50 32.12           C
HETATM  917  O2  AIN A 141      13.911  15.759   2.749  0.50 32.90           O
HETATM  918  C3  AIN A 141      11.830  15.316   1.664  0.50 32.34           C
HETATM  919  C4  AIN A 141      11.114  15.381   0.456  0.50 32.08           C
...
TER
END
```

- **RDKit** - RDKit is a software suite used for cheminformatics, computational chemistry and predictive modelling [Lan13]. Advantages of this software suite include the support of reading both standardised and non-standardised PDB files, the yielding of data required to check against Lipinski's rules, and the assistance of extracting the features from every atom. This software will thus assist in being able to accomplish the aim of this study in exploring protein-ligand interactions.
- **Open Babel** - Open Babel is an open-source chemical toolbox which provides the ability to convert files from one chemical file format to another [OBJ+11]. In this study, Open Babel aided in the interconversion of files, containing structural data, before these were standardised.

4.3.1 Data Filtering

The data filtering process featured an important role in this study, as it leads to the creation of databases. The databases created were made publicly available on Github[1]. The steps taken to complete this process were as follows:

1. **Extracting PDB Files and Checking for Ligands** - The first step was to acquire the raw data. The total protein structures in the PDB amounted to 127,562 structures. Whilst extracting the data, the data was filtered to only keep that which

[1]Github Repository: https://github.com/dcatania318/ProtLigInteractionDB

contained binding partners, which amounted to 100,205 structures (as shown in Figure 9).

2. **Splitting PDB Files into Proteins and Ligands** - Every PDB file containing small-molecules was then processed to keep only the relevant data. Every ligand in every chain for every structure was extracted into a separate file. Also, the residues, which had at least one atom which is less than 12 angstroms away from an atom of the small-molecule, were also extracted into another file. The distance between atoms on the protein and atoms on the small-molecules was calculated using the *3D Euclidean Distance* (stated in Equation 1) given the 3D coordinates of the protein (P) and of the small-molecule (L). Both files were similarly named so as to easily match each other when extracting features of their interactions.

$$d = \sqrt{(x_P - x_L)^2 + (y_P - y_L)^2 + (z_P - z_L)^2} \tag{1}$$

3. **Standardising Atoms and Hetatoms** - In standardisation, every structure is checked for inconsistencies and where possible is modified in order for it to be normalised. Before the actual standardisation, the files to be processed were first converted to a better file format for standardisation using Open Babel, which increased the success rate of a file to be successfully standardised. 92,387 structures were successfully standardised.

4. **Applying Lipinski's rule of five** - As previously described, Lipinski's rule of five predicts the overall drug-likeness of small-molecules. A script to check the drug-likeness of small-molecules was written with the aid of a cheminformatic library, RDKit. 18,408 small-molecules matched all of Lipisnki's rules.

5. **Checking and Extraction of Pharmacophores** - After applying Lipinski's rules on the small-molecules, the next stage was to check if both the small-molecules and their nearby atoms contained pharmacophores. The pharmacophores, and their coordinates, for the small-molecules and their nearby atoms were extracted into another simple text file so as to improve performance, as in Listing 2. The number of structures containing pharmacophores totalled to 18,386.

6. **Extract Pharmacophore Counts for Protein-Ligand Interactions** - The extracted pharmacophores were then used to build the database of protein-ligand interactions. The receptor-ligand pharmacophore relationships were matched and counted by analysing a structure's, matching protein and small-molecule, features and search-

Listing 2: Pharmacophore Features for a Small-Molecule in PDB Structure 3EML.

```
1,Acceptor,-2.118,-20.474,26.029
2,Acceptor,-0.204,-19.579,25.428
3,NegIonizable,-1.123,-19.986,25.909
5,Hydrophobe,-0.729,-19.55,27.709
6,Hydrophobe,-1.574,-20.321,28.715
7,Hydrophobe,-1.071,-20.087,30.137
8,Hydrophobe,-2.21,-20.089,31.126
9,Hydrophobe,-1.759,-20.748,32.401
10,Hydrophobe,-2.794,-20.596,33.492
11,Hydrophobe,-2.128,-20.715,34.853
12,Hydrophobe,-3.14,-21.015,35.946
13,Hydrophobe,-2.471,-21.555,37.199
14,Hydrophobe,-3.435,-21.496,38.371
15,Hydrophobe,-3.3,-22.696,39.292
16,Hydrophobe,-2.601,-22.325,40.583
17,Hydrophobe,-2.992,-23.264,41.711
18,Hydrophobe,-2.591,-24.716,41.436
19,Hydrophobe,-2.905,-25.63,42.614
20,Hydrophobe,-1.785,-26.625,42.842
```

ing for relationships having a distance less than or equal to that specified between them, as shown in Table 2. A structure having no protein-ligand relationships would have been discarded.

7. **Intersection with PDBbind** - The last stage was to intersect the created database with the PDBbind Database, as shown in Figure 10. The PDBbind Database is a collection of experimentally calculated binding data for the biomolecular complexes in the PDB. Its latest release provided the binding data from 13,308 protein-ligand biomolecular complexes. The binding affinity data was evaluated in three different experimental measures being K_d, K_i, and IC_{50}, and so the structures' data, containing the pharmacophore relationship counts, together with the binding affinity were collected in three different databases according to their binding affinity measure. Therefore, upon intersecting, three other databases were created, one for every experimental measure, which altogether featured 5,416 structures. The binding affinity in the PDBbind Database ranged from molar (M / 10^0M) all the way to femtomolar (fM / 10^{-15}M), and so they were all normalised to be nanomolar (nM / 10^{-9}M). Also, the log of the binding affinity was also taken to reduce the skewness of the datasets, so as to build more accurate machine learning models to predict the binding affinity for protein-ligand interactions.

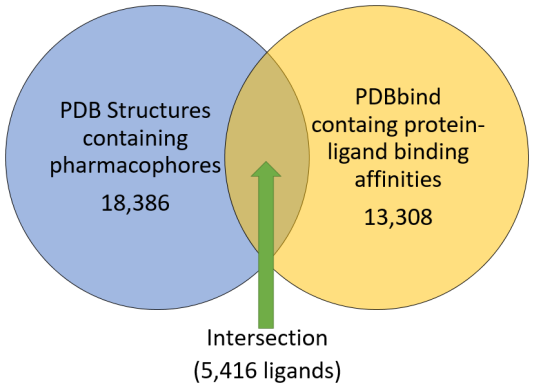


Figure 10: Intersection of Protein-Ligand Database with PDBbind Database.

4.4 Model Building

The methods used for model building form part of the Scikit-learn module. This module provides implementations for various machine learning algorithms. It stands out from other modules implementing these algorithms as it is more efficient and only depends on two other widely used modules, *numpy* and *scipy*.

The binding affinity data used is furthermore discussed in this section. The machine learning model methods used are also presented together with the parameters used to execute such models for training.

4.4.1 Binning the Binding Affinity

The binding affinity was binned in order to group continuous values into categories, also known as bins. This resulted in thousands of bins being created since the binding affinity ranged from molar to femtomolar. In order to reduce the number of bins, the logarithm, to base 10, of the binding affinity was then binned. This resulted in around 20 bins being created. The binning of data allowed a smoother process of training, validating and testing a classifier in model building. The binning size was separately calculated for the three different databases containing different binding affinity measures. This was done by using *Freedman-Diaconis rule* which uses the interquartile range *(IQR(x))* of the data and n observations in the sample [CL14], as stated in Equation 2.

$$binSize = 2\frac{IQR(x)}{\sqrt[3]{n}} \tag{2}$$

4.4.2 Performance Measures - Multiclass Classification

The performance measures, discussed in Section 2.5.2, were developed to produce scores for binary classes. The classification task for this study was multiclass, since the binding affinity classes being predicted were more than two. The method used to average the measures of the scores was the micro-average method. This method sums up the true positives, false positives and false negatives for different sets of classes, and applies them to get the statistics [TV07]. Equal precision, recall and F1 scores are produced when using micro-averaging.

4.4.3 Methods Used

The problem set in this study was to identify to which set of categories a structure belongs to. To address this problem, machine learning model classifiers were used.
The classifiers used were:

- *KNeighborsClassifier* - a classifier to build the k-nearest neighbours model with the following parameters:
  - querying the nearest 10 neighbours;
  - equally weighing all points in every neighbourhood;
  - computing the nearest neighbours brutally;
  - setting the distance metric to be equivalent to the standard Euclidean metric; and
  - running a single neighbours search job.
- *SVC* - a classifier to build the support vector machine model with the following parameters:
  - having a penalty parameter of 1;
  - using both a *linear* and a *RBF* kernel;
  - disabling probability estimates;
  - using the shrinking heuristic; and
  - setting no limit on iterations.

- *RandomForestClassifier* - a classifier to build the random forest model with the following parameters:
  - having 20 trees in the forest;
  - having a split quality for information gain;
  - considering all the features when looking for the best split;
  - having no maximum depth of the tree;
  - having the minimum number of samples required to split an internal node to be two;
  - having the minimum number of samples required to be at a leaf to be one;
  - having unlimited number of leaf nodes; and
  - using bootstrapping, the practice of estimating properties of an estimator, when building the trees.
- *MLPClassifier* - a classifier implementing the multi-layer perceptron (MLP) model, which is a feedforward artificial neural network, with the following parameters:
  - having 100 hidden layers;
  - using the logistic sigmoid activation function which returns $f(x) = \frac{1}{1+e^{-x}}$;
  - using a stochastic gradient-based optimizer (adam);
  - having a maximum of 200 iterations until convergence; and
  - shuffling samples on each iteration.

4.5 Conclusion

This section exhibited how big data was dealt with and presented the data acquisition and cleaning processes. It also goes through the models that were built to predict the binding affinity class of new protein-ligand interactions. These models are crucial to the evaluation of this study.

5 Results & Evaluation

In this section, the results obtained from the machine learning models will be presented. These models were built using the eight feature counts and the binding affinity of protein-ligand interactions. The eight feature counts are the protein-ligand interactions mined from structures as in Table 2. The binding affinity is measured in three different experimental measures, which are not easily convertible from one to another (without knowing the experiment's parameters). These measures are:

- IC_{50} - The *half maximal inhibitory concentration* is a measure of influence in inhibiting a specific biological or biochemical activity;
- K_i - The *equilibrium constant* is a measure of inhibition of a process;
- K_d - The *dissociation constant* is a measure of substrate binding.

The machine learning models built for each measure of binding affinity were:

- Nearest Neighbours Classifier (NNC);
- Support Vector Machine - Linear Kernel (SVM-Linear);
- Support Vector Machine - RBF Kernel (SVM-RBF);
- Random Forest Classifier (RFC);
- Artificial Neural Network (ANN).

The dataset used to build a model was split into a training set (60%), a validation set (20%) and a testing set (20%). The training set was used to train a model. The validation set was used to cross-validate a model. The cross-validation (CV) accuracy was calculated using ten folds and is presented as $x(\pm y)$, where x is the mean accuracy percentage and y is the standard derivation percentage of the ten folds. The *Accuracy*, *Precision*, *Recall* and *F1* Scores were also calculated by using the testing set. The results of these four scores were equal, due to the use of micro-averaging since we dealt with multiclass problems. In the following evaluation of models, these scores will be treated as H-score (Hold-out score). Every model in each measure was built using the three following sub-datasets:

- An unbalanced sub-dataset using the eight features and the binned binding affinity (*unbalancedNonLog*);
- An unbalanced sub-dataset using the eight features and the binned logarithm binding affinity (*unbalancedLog*); and
- A balanced sub-dataset using the eight features and the binned logarithm binding affinity (*balancedLog*).

The prediction class for an *unbalancedNonLog* sub-dataset was the binned binding affinity, whilst that for both an *unbalancedLog* and a *balancedLog* sub-dataset was the logarithm binned binding affinity.
The data in the *unbalancedNonLog* sub-datasets was skewed because of the bins taken for the binding affinity. These bins could contain lots of data or no data at all, as the data ranged from molar to femtomolar. An example of this skewness is shown for the IC_{50} dataset in Figure 11, where the majority of bins are seen to be at the lower end of the binding affinity measures.

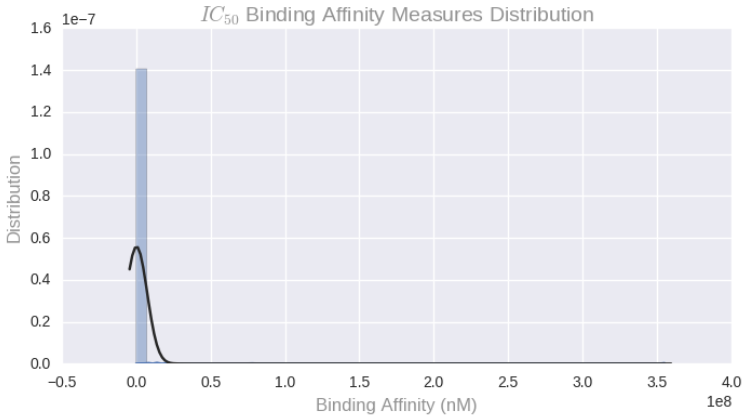


Figure 11: Histogram Plot of IC_{50} Binding Affinity Measures.

5.1 Evaluating the IC_{50} dataset

The IC_{50} dataset contained data belonging to 2,600 structures. The bin ranges of the binned logarithm binding affinity for IC_{50} datasets are shown in Table 6 (Appendix B). The sub-dataset using the eight features and the binned logarithm binding affinity was balanced by keeping the latter's groups which had *150* occurrences. The balanced sub-dataset was built by keeping a random sample of the same amount.
The H-score results of the three sub-datasets are shown in Figure 12, whilst the cross-validation results are shown in Table 3. The dataset *unbalancedNonLog* showed a higher cross-validation accuracy and a higher H-score due to skewness caused by the majority of bins residing at the lower end of the binding affinity measures (shown in Figure 11). Ignoring the skewed dataset *unbalancedNonLog*, the best accuracy rate, for both the cross-validation and H-score, of the IC_{50} dataset was achieved using the Random Forest Classifier.

ML Models	unbalancedNonLog	unbalancedLog (21-class)	balancedLog (21-class)
NNC	65%(±18%)	11%(±4%)	9%(±6%)
SVM-Linear	65%(±17%)	12%(±4%)	10%(±6%)
SVM-RBF	65%(±17%)	14%(±5%)	10%(±7%)
RFC	53%(±16%)	9%(±4%)	15%(±5%)
ANN	65%(±17%)	13%(±3%)	11%(±4%)

Table 3: The cross-validation accuracy of the three sub-datasets for IC_{50}.

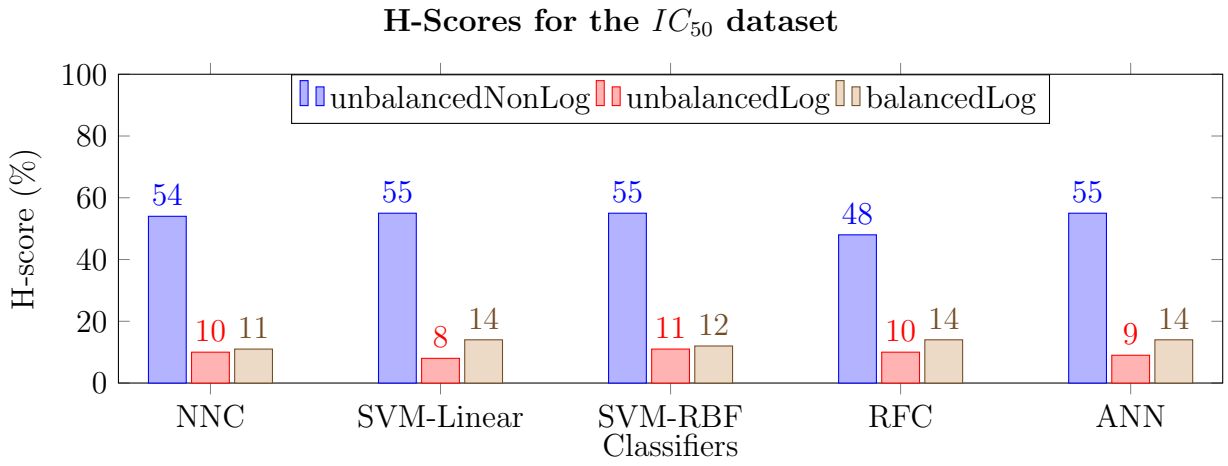


Figure 12: The H-scores of the three sub-datasets for IC_{50}.

5.2 Evaluating the K_i dataset

The K_i dataset contained data belonging to 1,312 structures. The bin ranges of the binned logarithm binding affinity for K_i datasets are shown in Table 7 (Appendix B). The sub-dataset using the eight features and the binned logarithm binding affinity was balanced by keeping the latter's groups which had *130* occurrences. The balanced sub-dataset was built by keeping a random sample of the same amount.

The H-score results of the three sub-datasets are shown in Figure 13, whilst the cross-validation results are shown in Table 4. Similar to the IC_{50} dataset, the dataset *unbalancedNonLog* showed a higher cross-validation accuracy and a higher H-score. Ignoring the skewed dataset *unbalancedNonLog*, the best cross-validation accuracy for the K_i dataset was achieved using the Support Vector Machine using a RBF Kernel, whilst the best H-score was achieved using the Support Vector Machine using a Linear Kernel.

ML Models	unbalancedNonLog	unbalancedLog (20-class)	balancedLog (20-class)
NNC	68%(±20%)	19%(±6%)	68%(±21%)
SVM-Linear	68%(±20%)	25%(±5%)	63%(±18%)
SVM-RBF	68%(±20%)	22%(±5%)	70%(±16%)
RFC	57%(±19%)	15%(±8%)	61%(±12%)
ANN	69%(±20%)	24%(±5%)	68%(±17%)

Table 4: The cross-validation accuracy of the three sub-datasets for K_i.

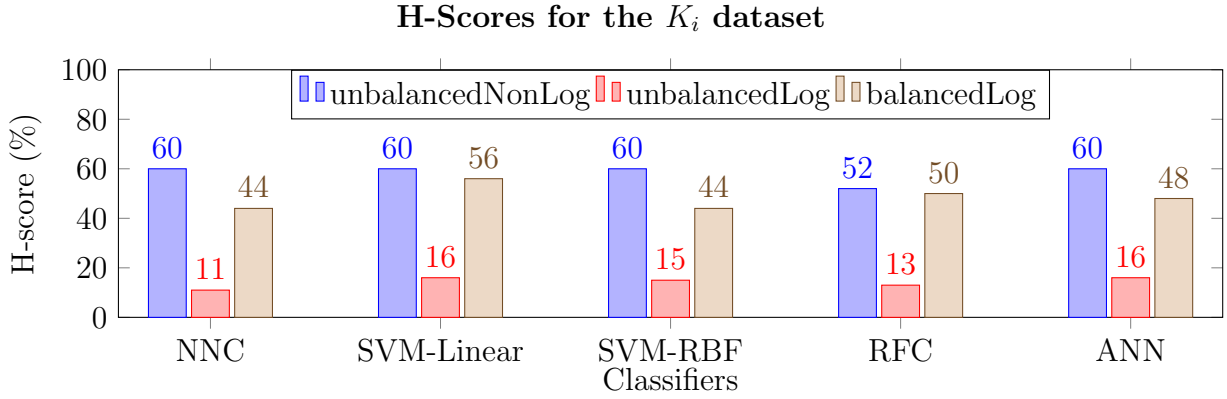


Figure 13: The H-scores of the three sub-datasets for K_i.

5.3 Evaluating the K_d dataset

The K_d dataset contained data belonging to 1,117 structures. The bin ranges of the binned logarithm binding affinity for K_d datasets are shown in Table 8 (Appendix B). The sub-dataset using the eight features and the binned logarithm binding affinity was balanced by keeping the latter's groups which had *70* occurrences. The balanced sub-dataset was built by keeping a random sample of the same amount.

The H-score results of the three sub-datasets are shown in Figure 14, whilst the cross-validation results are shown in Table 5. Similar to the IC_{50} dataset, the dataset *unbalancedNonLog* showed a higher cross-validation accuracy and a higher H-score. Ignoring the skewed dataset *unbalancedNonLog*, the best cross-validation accuracy for the K_d dataset was achieved using the Artificial Neural Network, whilst the best H-score was achieved using the Support Vector Machine using a Linear Kernel.

ML Models	unbalancedNonLog	unbalancedLog (19-class)	balancedLog (19-class)
NNC	71%(±22%)	10%(±6%)	11%(±10%)
SVM-Linear	73%(±22%)	9%(±5%)	16%(±12%)
SVM-RBF	73%(±22%)	10%(±7%)	14%(±9%)
RFC	58%(±20%)	6%(±7%)	9%(±8%)
ANN	73%(±23%)	12%(±3%)	17%(±12%)

Table 5: The cross-validation accuracy of the three sub-datasets for K_d.

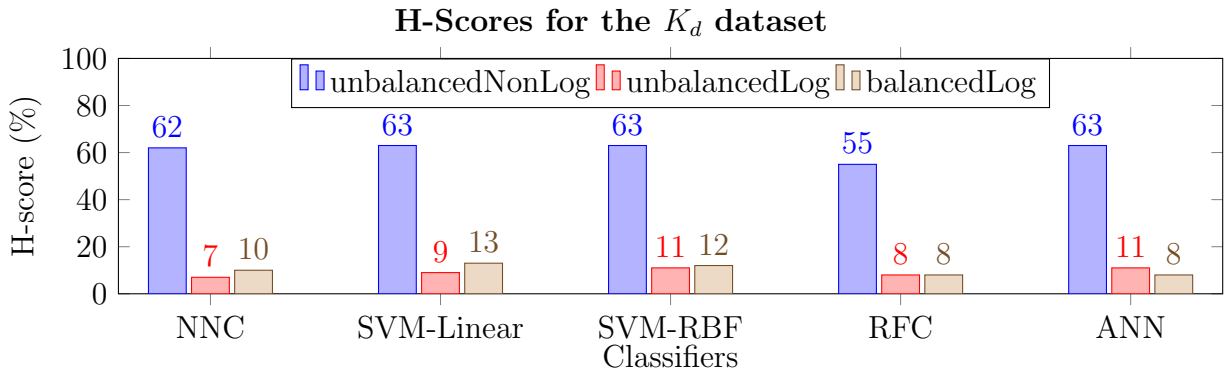


Figure 14: The H-scores of the three sub-datasets for K_d.

5.4 Conclusion

From the three different experimental measures, the K_i measure produced the best cross-validation accuracy of 70%(±16%), achieved by an Artificial Neural Network model, and the best H-score accuracy of 56%, achieved with the Support Vector Machine using a Linear Kernel. The random model accuracy for the K_i dataset is that of 5%, since this dataset deals with a 20-class problem. The results produced by these models are better than those obtained randomly and so could be concluded that the K_i measure is the best experimental measure to be predicted. Although the 56% accuracy is satisfiable, there is always room for improvement to improve this accuracy rate.

The outcome of this study is that it is hard to predict the binding affinity based on the feature counts of protein-ligand interactions. This outcome could be the result of:

- The binding affinity measurement - as this could have been measured in different laboratories with a different setup causing a systematic variation; and
- The repeated feature counts in the datasets - repeated data with different binding affinity classes might have made the building of the model harder since the data was not that distant.

6 Conclusion

The aim of this thesis was to predict the binding affinity of protein-ligand interactions as these are critical for drug discovery. This was achieved by building databases of protein-ligand interactions and using them to build machine learning models.

Initially, the structural data was acquired from the Protein Data Bank, which is a 3D repository of protein structures. The acquired structural data was then filtered to keep those which contained small-molecules. The small-molecules and their nearby atoms on the protein were then split and stored in separate files. These files were then checked for inconsistencies by standardising them. The small-molecules were then checked for their drug-likeness by applying Lipinski's rule of five, and only keeping those small-molecules which matched all of the five rules. Thereafter, the pharmacophores (features) of these small-molecules and their nearby atoms were extracted, for those which contained such features. The feature counts were then calculated for every structure and stored in a database. This newly created database was then intersected with the PDBbind database, so as to match the protein-ligand interaction features with their respective binding affinities. Due to this intersection, three databases were created, one for every different experimental measure of the binding affinity. Subsequently, whilst matching the features with the binding affinity measure, the same binding affinities were converted to nanomolar (nM / 10^{-9}M), as these ranged from molar (M / 10^{0}M) to femtomolar (fM / 10^{-15}M), logged and binned. The databases created were uploaded to a Github repository[2] and are available publicly. We are sure that these are a useful resource to other researchers in the field.

These databases were then used to build machine learning models to predict the binding affinity. The machine learning models considered were:

- Nearest Neighbours Classifier;
- Support Vector Machine with a Linear Kernel;
- Support Vector Machine with a RBF Kernel;
- Random Forest Classifier;
- Artificial Neural Network.

Furthermore, these machine learning models were then trained and tested using the features and binned binding affinity, so as to predict the binding affinity range. The results from these models were then evaluated. The accuracy rate of these models varied widely.

[2]Github Repository: https://github.com/dcatania318/ProtLigInteractionDB

The K_i experimental measure was considered to be the best measure from the three experimental measures of the binding affinity, as a Support Vector Machine with a Linear Kernel produced an accuracy of 56%. Although this outcome was satisfiable, predicting the binding affinity based on the feature counts of protein-ligand interactions is still considered to be hard. In conclusion, the main objectives, stated in Section 1.5, set for this thesis were achieved.

6.1 Future Work

The principal focus of this study was to build machine learning models to explore protein-ligand interactions for drug discovery. The development of databases delayed the building of these models. The future works of this study could involve:

- Using ensemble models to predict the binding affinity - such models could consist of deep learning models as these allow the representation of data with multiple levels of abstraction [LBH15];
- Looking at specific classes of proteins - perhaps binding affinities for some protein classes are easier to predict than others; and
- Adding more features to the structure entries in the databases so as to make every structure entry more distant - features could include the number of hydrogen bonds and the weight of a small-molecule.

6.2 Final Remarks

Through meticulous research a new protein-ligand interaction database together with binding affinity data has been constructed. This database has been made publicly accessible with the intention to aid researchers as they further explore and refine this area of study. This database was also used to build successful machine learning models which produced a satisfiable outcome in predicting the binding affinity of protein-ligand interactions. This result would in turn assist in the drug discovery process.

References

[AR06] Shameen Akhter and Jason Roberts. *Multi-core programming*, volume 33. Intel press Hillsboro, 2006.

[AS59] O Arunlakshana and HO Schild. Some quantitative uses of drug antagonists. *British journal of pharmacology*, 14(1):48–58, 1959.

[AS05] Juan Alvarez and Brian Shoichet. *Virtual screening in drug discovery*. CRC press, 2005.

[Bre17] Zachary Brennan. New fda drug approvals: Breaking down the numbers. `http://raps.org/Regulatory-Focus/News/2017/01/04/26501/New-FDA-Drug-Approvals-Breaking-Down-the-Numbers`, January 2017. Last Accessed: 2017-05-24.

[Bun11] Mark E Bunnage. Getting pharmaceutical r&d back on target. *Nature Chemical Biology*, 7(6):335, 2011.

[BWF+00] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.

[CAC+09] Peter JA Cock, Tiago Antao, Jeffrey T Chang, Brad A Chapman, Cymon J Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 2009.

[CC00] Brad Chapman and Jeffrey Chang. Biopython: Python tools for computational biology. *ACM Sigbio Newsletter*, 20(2):15–19, 2000.

[CL14] Claudia Canali and Riccardo Lancellotti. Exploiting ensemble techniques for automatic virtual machine clustering in cloud systems. *Automated Software Engineering*, 21(3):319–344, 2014.

[Ebe14] Jean-Paul Ebejer. *Data driven approaches to improve the drug discovery process: a virtual screening quest in drug discovery*. PhD thesis, University of Oxford, 2014.

[Elk11] Charles Elkan. Nearest neighbor classification. *elkan@ cs. ucsd. edu, January*, 11:3, 2011.

[FDA17] FDA. New molecular entity and new therapeutic biological product approvals. `http://www.fda.gov/Drugs/DevelopmentApprovalProcess/DrugInnovation/default.htm`, January 2017. Last Accessed: 2017-05-24.

[GCB17] Aravindhan Ganesan, Michelle L Coote, and Khaled Barakat. Molecular dynamics-driven drug discovery: leaping forward with confidence. *Drug Discovery Today*, 22(2):249–269, 2017.

[GILS10] Warren RJD Galloway, Albert Isidro-Llobet, and David R Spring. Diversity-oriented synthesis as a tool for the discovery of novel biologically active small molecules. *Nature communications*, 1:80, 2010.

[HCL+03] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[K+95] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Stanford, CA, 1995.

[KL03] S Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural computation*, 15(7):1667–1689, 2003.

[Lab14] UCSF Computer Graphics Laboratory. Introduction to protein data bank format. `https://www.cgl.ucsf.edu/chimera/docs/UsersGuide/tutorials/pdbintro.html`, 2014.

[Lan13] Greg Landrum. Rdkit: A software suite for cheminformatics, computational chemistry, and predictive modeling, 2013.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[Lip04] Christopher A Lipinski. Lead-and drug-like compounds: the rule-of-five revolution. *Drug Discovery Today: Technologies*, 1(4):337–341, 2004.

[LLDF97] Christopher A Lipinski, Franco Lombardo, Beryl W Dominy, and Paul J Feeney. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced drug delivery reviews*, 23(1-3):3–25, 1997.

[LLH+14] Zhihai Liu, Yan Li, Li Han, Jie Li, Jie Liu, Zhixiong Zhao, Wei Nie, Yuchen Liu, and Renxiao Wang. Pdb-wide collection of binding data: current status of the pdbbind database. *Bioinformatics*, page btu626, 2014.

[MBTH+03] J McKimm-Breschkin, T Trivedi, A Hampson, A Hay, A Klimov, M Tashiro, F Hayden, and M Zambon. Neuraminidase sequence analysis and susceptibilities of influenza virus clinical isolates to zanamivir and oseltamivir. *Antimicrobial agents and chemotherapy*, 47(7):2264–2272, 2003.

[MCM13] Ryszard S Michalski, Jaime G Carbonell, and Tom M Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[MNW98] GWA Milne, MC Nicklaus, and S Wang. Pharmacophores in drug design and discovery. *SAR and QSAR in Environmental Research*, 9(1-2):23–38, 1998.

[NWN13] Cuong Nguyen, Yong Wang, and Ha Nam Nguyen. Random forest classifier combined with feature selection for breast cancer diagnosis and prognostic. 2013.

[OBJ+11] Noel M O'Boyle, Michael Banck, Craig A James, Chris Morley, Tim Vandermeersch, and Geoffrey R Hutchison. Open babel: An open chemical toolbox. *Journal of cheminformatics*, 3(1):33, 2011.

[Ols05] David L Olson. Data set balancing. In *Data Mining and Knowledge Management: Chinese Academy of Sciences Symposium CASDMKD 2004, Beijing, China, July 12-14, 2004, Revised Paper*, volume 3327, page 71. Springer, 2005.

[PVG+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[RB99] Yoram Reich and SV Barai. Evaluating machine learning models for engineering problems. *Artificial Intelligence in Engineering*, 13(3):257–272, 1999.

[SB09] Adrian Schreyer and Tom Blundell. Credo: a protein–ligand interaction database for drug discovery. *Chemical biology & drug design*, 73(2):157–167, 2009.

[SCD04] Juswinder Singh, Claudio Chuaqui, and Zhan Deng. Structural interaction fingerprint, July 1 2004. US Patent App. 10/562,974.

[Sho04] Brian K Shoichet. Virtual screening of chemical libraries. *Nature*, 432(7019):862–865, 2004.

[SLT+03] Vladimir Svetnik, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan, and Bradley P Feuston. Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, 43(6):1947–1958, 2003.

[Sum10] Mark Summerfield. *Programming in Python 3: a complete introduction to the Python language*. Addison-Wesley Professional, 2010.

[TV07] Grigorios Tsoumakas and Ioannis Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *European Conference on Machine Learning*, pages 406–417. Springer, 2007.

[WFL+05] Renxiao Wang, Xueliang Fang, Yipin Lu, Chao-Yie Yang, and Shaomeng Wang. The pdbbind database: methodologies and updates. *Journal of medicinal chemistry*, 48(12):4111–4119, 2005.

[WFLW04] Renxiao Wang, Xueliang Fang, Yipin Lu, and Shaomeng Wang. The pdbbind database: Collection of binding affinities for protein- ligand complexes with known three-dimensional structures. *Journal of medicinal chemistry*, 47(12):2977–2980, 2004.

[Wik08] Wikibooks. Protein ligand binding - image. `https://en.wikibooks.org/wiki/File:Proten_ligand_binding.PNG`, December 2008. Last Accessed: 2017-05-14.

[ZHS09] Jinming Zou, Yi Han, and Sung-Sau So. Overview of artificial neural networks. *Artificial Neural Networks: Methods and Applications*, pages 14–22, 2009.

Appendix A - Protein Data Bank File Format

Protein Data Bank Format: Coordinate Section				
Record Type	**Columns**	**Data**	**Justification**	**Data Type**
ATOM	1-4	"ATOM"		character
	7-11[#]	Atom serial number	right	integer
	13-16	Atom name	left[*]	character
	17	Alternate location indicator		character
	18-20[§]	Residue name	right	character
	22	Chain identifier		character
	23-26	Residue sequence number	right	integer
	27	Code for insertions of residues		character
	31-38	X orthogonal Å coordinate	right	real (8.3)
	39-46	Y orthogonal Å coordinate	right	real (8.3)
	47-54	Z orthogonal Å coordinate	right	real (8.3)
	55-60	Occupancy	right	real (6.2)
	61-66	Temperature factor	right	real (6.2)
	73-76	Segment identifier[¶]	left	character
	77-78	Element symbol	right	character
79-80	Charge		character	
HETATM	1-6	"HETATM"		character
	7-80	same as ATOM records		
TER	1-3	"TER"		character
	7-11[#]	Serial number	right	integer
	18-20[§]	Residue name	right	character
	22	Chain identifier		character
	23-26	Residue sequence number	right	integer
	27	Code for insertions of residues		character

Figure 15: Protein Data Bank File Format: Coordinate Section [Lab14].
Columns - refers to the position in the line where data resides;
Data - refers to what the data means;
Justification - refers to where the data is aligned for trimming purposes; and
Data Type referring to the data's data type.

Appendix B - Bin Ranges of the Binned Logarithm Binding Affinity

Bin ID	Bin Range
A	$0 <= x < 1.0$
B	$1.0 <= x < 2.0$
C	$2.0 <= x < 3.0$
D	$3.0 <= x < 4.0$
E	$4.0 <= x < 5.0$
F	$5.0 <= x < 6.0$
G	$6.0 <= x < 7.0$
H	$7.0 <= x < 8.0$
I	$8.0 <= x < 9.0$
J	$9.0 <= x < 10.0$
K	$10.0 <= x < 11.0$
L	$11.0 <= x < 12.0$
M	$12.0 <= x < 13.0$
N	$13.0 <= x < 14.0$
O	$14.0 <= x < 15.0$
P	$15.0 <= x < 16.0$
Q	$16.0 <= x < 17.0$
R	$17.0 <= x < 18.0$
S	$18.0 <= x < 19.0$
T	$19.0 <= x < 20.0$
U	$20.0 >= x$

Table 6: Bin Ranges of the Binned Logarithm Binding Affinity for the IC_{50} Dataset.

Bin ID	**Bin Range**
A	$0 <= x < 1.0$
B	$1.0 <= x < 2.0$
C	$2.0 <= x < 3.0$
D	$3.0 <= x < 4.0$
E	$4.0 <= x < 5.0$
F	$5.0 <= x < 6.0$
G	$6.0 <= x < 7.0$
H	$7.0 <= x < 8.0$
I	$8.0 <= x < 9.0$
J	$9.0 <= x < 10.0$
K	$10.0 <= x < 11.0$
L	$11.0 <= x < 12.0$
M	$12.0 <= x < 13.0$
N	$13.0 <= x < 14.0$
O	$14.0 <= x < 15.0$
P	$15.0 <= x < 16.0$
Q	$16.0 <= x < 17.0$
R	$17.0 <= x < 18.0$
S	$18.0 <= x < 19.0$
T	$19.0 >= x$

Table 7: Bin Ranges of the Binned Logarithm Binding Affinity for the K_i Dataset.

Bin ID	**Bin Range**
A	$0 <= x < 1.0$
B	$1.0 <= x < 2.0$
C	$2.0 <= x < 3.0$
D	$3.0 <= x < 4.0$
E	$4.0 <= x < 5.0$
F	$5.0 <= x < 6.0$
G	$6.0 <= x < 7.0$
H	$7.0 <= x < 8.0$
I	$8.0 <= x < 9.0$
J	$9.0 <= x < 10.0$
K	$10.0 <= x < 11.0$
L	$11.0 <= x < 12.0$
M	$12.0 <= x < 13.0$
N	$13.0 <= x < 14.0$
O	$14.0 <= x < 15.0$
P	$15.0 <= x < 16.0$
Q	$16.0 <= x < 17.0$
R	$17.0 <= x < 18.0$
S	$18.0 >= x$

Table 8: Bin Ranges of the Binned Logarithm Binding Affinity for the K_d Dataset.

Appendix C - CD Contents

The attached CD contains the following content:

1. A soft-copy of this report.
2. A *code* folder containing the code used for this study.
3. A *data* folder which is to contain the data related to this study:
    - The *PDB*, *PDB2*, *PDB3* and *PDB4* folders containing the data processed are to be requested (as they amount to 216GB) and extracted here.
    - The *PDBbind* folder containing the PDBbind databases used in this study.
4. A *datasets* folder containing the datasets produced in this study. together with a *bin_ranges* folder containing the binned ranges for the datasets.
5. A readme file explaining the hierarchy of the CD.