MSc Project - Binding Affinity Prediction of Protein-Ligand Complex

Abdus Salam Khazi
abdus.khazi@students.uni-freiburg.de

Github Repository [6]

Supervisors: Simon Bray & Alireza Khanteymoori

August 4, 2021

Contents

1 Introduction

1.1 Biological Background

Proteins are the workhorses of our body. They are necessary for many important functions in the body. Ligands are molecules that bind to proteins to form protein-ligand complexes. They can be molecules that the protein transports (e.g., a Haemoglobin transporter) or act as stimulating agents. In addition to this, they can also start/stop the protein from doing its function. The correct functioning of these protein-ligand complexes is essential for any living organism.

The study of protein-ligand complexes is an intrinsic part of the drug discovery field. It is because drugs are small molecules that act as ligands. As the drug molecules (ligands) bind to the target proteins, they can artificially influence the protein behavior. This causes a therapeutic effect.

may cause

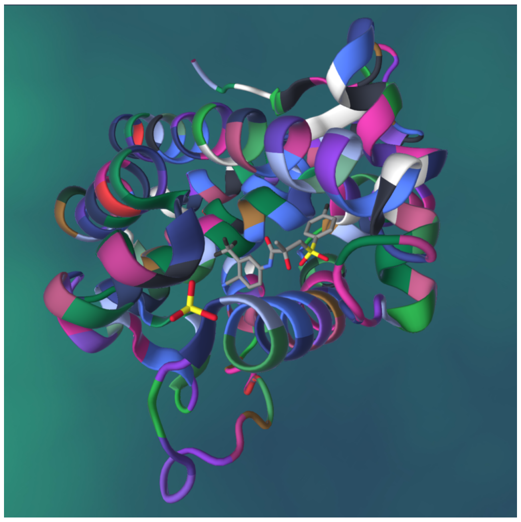


Figure 1: Haemoglobin transporter protein. [15]

When we find a target drug candidate, we have to answer questions like - How easily does the drug bind to the target protein? Does it bind to any other protein - If so, is it desirable? Does it have any unforeseen effect on the protein function? etc... To answer these questions biologists and pharmacists conduct wet-lab experiments that are expensive.

One way to reduce the cost of these experiments is to make a data-driven selection of the drugs. Using experimental data collected over many years, one can build models to predict the behavior of the proposed drug computationally. These 'In-Silico' computational methods can aid in the elimination of undesirable drugs as well as guide the drug selection process.

Our project aims to answer one of the above questions - How well does a given drug bind to the target protein? We determine this computationally by building a machine learning model that trains on the previous data. We hope that this model will help reduce the costs of drug discovery.

1.2 Understanding Binding Affinity

The binding affinity between a protein and a ligand is quantified by the K_d, K_i and IC_{50} measures in the PDBBind Data bank. Here K_d refers to disassociation constant, K_i refers to the inhibition constant, and IC_{50} refers to inhibitory concentration 50%. The reason for having different measurements is because it is not possible to use the same measurement techniques for all biological complexes/processes.

To understand K_d, consider a protein and a ligand binding and unbinding continuously in a kinetic system. In this system, let $[P]$, $[L]$, and $[PL]$ represent the concentrations of the Protein, the Ligand, and the Protein-Ligand complex respectively. This is represented by the following equation:

$$[P] + [L] \rightleftharpoons [PL]$$

We can quantify the binding affinity K_d by using the concentrations in the above system at equilibrium.

$$K_d = \frac{[P][L]}{[PL]} = \frac{k_{-1}}{k_1}$$

where k_{-1} is the disassociation rate constant and k_1 is the association rate constant. Similarly, K_i and IC_{50} are defined using concentration albeit non-trivially. [11]

1.3 PDBBind Dataset

Over the last few decades, researchers have been successful in building a single data archive for proteins. This archive, called **Protein Data bank** [8] , holds 3-D structural data of the proteins determined by experiments like X-ray crystallographic, Nuclear magnetic resonance (NMR), and cryoelectron microscopy (cryoEM). A subset of this data also contains information about how well a given protein and ligand bind together. It is called binding affinity between a protein and ligands. (It also contains data about protein-protein complexes that our project does not deal with) [7]

The PDBBind database

As we study the protein-ligand binding affinity here, we would like to filter out this data from the protein data bank. It is what is done by the maintainers of the **PDBBind Data bank**. [13] Using the curated protein-ligand affinity data present in the PDBBind Data bank, we build a machine learning model that learns to predict the affinity.

2 Problem Formulation

2.1 Problem Overview

The problem that we are solving is - *Given* $K_d/K_i/IC_{50}$ *for various complexes in the PDBBind Data bank, can we predict this affinity measure for new protein-ligand complexes?* Figure 2 shows how the Protein-Ligand problem can be classified.

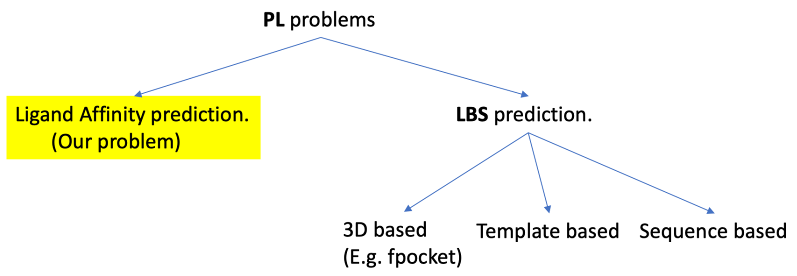


Figure 2: Protein-Ligand problem classifcation.

The binding of proteins and ligands is heavily influenced by their respective 3D structures. Figure 3 illustrates a hypothesis called *Lock and Key*. As you can see it is very crucial that the shape of the protein's binding location and the shape of the ligand be complementary for the binding.

it can be seen that

The other parts of the protein are not involved in the binding process directly. Hence, we only get the features of the binding location. Any potential binding location in the 3D structure of a protein is called a pocket. We use a 3D based LBS prediction package called *fpocket* to find the potential pockets. A submodule in the package called *dpocket* is used to extract the binding pocket's features.

To give a plug-and-play input to our model, we keep the features of proteins and ligands distinct till the training phase. That is, we do not use the combined features in any pre-processing of the ML pipeline e.g. feature reduction. This helps our model to provide the binding affinity between any protein and any ligand.

can you cite the fpocket paper? Le Guilloux et al, 2009

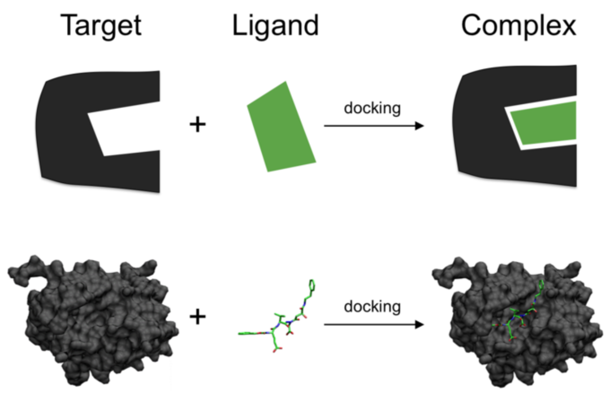


Figure 3: Lock and Key hypothesis in molecular docking. [12]

2.2 Overview of file formats

The **PDBBind Data bank** extracts information about the PL complexes from the **Protein Data bank** and creates the following files for every complex

- **PDB Format** - For the Protein.
- **Mol2** - For the ligand.
- **SDF** - For the ligand.

All of the above formats contain the 3D information that is essential in the prediction of the binding affinity. All the above mentioned formats use the **XYZ format** internally to represent the 3D structure of their molecules.

2.2.1 XYZ format

XYZ format is a chemical file format that represents the geometry of a molecule. It specifies the number of atoms and their Cartesian X, Y, Z coordinates hence the name XYZ format. The coordinates are relative to each other. Hence, translation and rotation do not change the molecule's representation. The following text illustrates the XYZ format. Section 8.1 gives an example. [18]

I don't think this is true

```
<number of atoms>
comment line
<element> <X> <Y> <Z>
...
```

The unit of distance used is Angstrom (Å). 1 Å = 10^{-10} m. [18]

2.2.2 PDB format

PDB format is a human-readable file format used to represent the protein molecules (macromolecules). Within the PDB format, the coordinates of atoms are represented like the XYZ format [see 2.2.1]. Because of the 3D information in this format, molecular visualization of proteins is possible with specialized software. It also contains information about atomic connectivity and the protein's primary, secondary, tertiary, and quaternary structures. [14] [4] Please see [1] for an example pdb file.

2.2.3 Structure Data File (SDF) Format

SDF format file is a Chemical Table file (CT File) that contains structure of the molecule in the X,Y,Z format. It contains information like atomic bonds, connectivity information, molecular weight, and molecular formula. [17] Section 8.2 illustrates the SDF file format.

2.2.4 Mol2 format

Similar to SDF format, Mol2 also represents the 3D structure of a molecule in the X,Y,Z format. It contains the atomic bond and connectivity information but does not contain the other data like molecular weight and formula. We use the ligands given in this format because more ligands in mol2 format could be processed with the RDKit feature extractor. Section 8.3 illustrates the Mol2 file format.

2.2.5 SMILES format

SMILES is an acronym for Simplified Molecular-Input Line-Entry System. It represents a molecule using an ASCII string. Using the 3D data in SDF and Mol2 formats, we can create an atomic graph representation. Using this graph the SMILES string for the molecule can be generated. The smiles format itself is not very helpful for us as we lose the 3D structural information after converting to it. [16]

2.3 Extraction of features

As the file formats are different for proteins and ligands, we use different tools to extract their features.

2.3.1 Ligand Features using RDKit

RDKit is an open-source cheminformatics software [9]. The core data structures and algorithms of RDKit are written in C++ and the python wrappers are generated using Boost.Python. Using the module, *RDKit.Chem.Descriptors* we extract 402 features for each ligand [2]. Each descriptor value is taken as a real number, hence the input space of ligand features is $\mathbf{R}^{402}$ before any feature elimination.

2.3.2 Protein Features using fpocket/dpocket descriptors

Fpocket stands for "Find pocket" whereas *Dpocket* stands for "Describe pocket". *Fpocket* uses 3D Voronoi tessalation and the concepts of "Alpha Spheres" to find out pockets in the protein structure [5] [19]. Given a protein PDB file, we can extract the descriptors of all pockets in the protein. For every pocket, we get 55 descriptors in total which are take as $\mathbf{R}$ values. Hence the input space for protein features is $\mathbf{R}^{55}$.

To get the descriptors of the ligand-binding pockets, we use *Dpocket*. *Dpocket* is provided with the protein PDB file and the ligand ID of the PL complex as input. It generates 3 files as output files, namely, dpout_fpocketp.txt, dpout_fpocketnp.txt, and dpout_explicitp.txt.

- **dpout_fpocketp.txt**. This file contains the descriptors (a.k.a features) of all pockets that are considered to be binding pockets based on a binding criterion. Multiple pockets can bind with the same ligand. Hence, there may be features of more than 1 pocket in this file.

"A protein may posses

- **dpout_fpocketnp.txt**. This file contains the descriptors of all pockets that are non-binding according to the criteria.

- **dpout_explicitp.txt**. An explicit pocket is defined as a pocket consisting of all vertices/atoms situated at a specific distance from the ligand in the PL complex. This distance is 4 Å by default. This file contains the descriptors of all explicit pockets in the PL complex.

In our project, we do not use dpout_fpocketnp.txt as they contain non-binding pockets. In the other 2 files, we prefer using pocket descriptors given by dpout_fpocketp.txt as explicitly defined pockets are heavily biased towards the ligand.

3 Feature selection

3.1 Requirement of feature selection

Both the protein and the ligand are equally responsible for the affinity of the PL complex. Hence we concatenate their descriptors to get a high dementional $\mathbf{R}^{457}$ input for our model. There are a couple of issues with using all of the descriptors.

- The amount of data is not very large. For example, we could only get ≈ 35000 data points after concatenating the ligand features with the *fpocketp* pocket descriptors.
- The number of ligand descriptors >> protein descriptors. This creates a data imbalance and may lead the model to select only ligand features for their prediction.

Hence we need some methods to reduce the input dimensions.

3.2 Feature Selection Strategies

We try to select features of the protein and the ligand separately. This helps us make our model plug-and-play as discussed in section 2.1. The best combination, i.e Global Optima, cannot be obtained practically by brute force algorithms. This is because we would have to try $\binom{402}{k_1} * \binom{55}{k_1}$ $(k_1, k_2 \in \mathbf{I}^+)$ possibilities which is impractical. The following feature selection heuristics were used instead -

3.2.1 Selection by output correlation

The *pearson* and *spearmann* correlations of each feature were calculated against the output variable. We assumed that the features were either linearly related to the output (in the case of Pearson correlation) or had a monotonic relationship (in the case of Spearmann correlation). The features with the highest correlation were selected as inputs to the model.

3.2.2 Using genetic algorithms [3]

Since the feature space is a non-continuous problem of combinatorial complexity, we also studied genetic feature selection algorithms. We represent each feature by a binary number, 1 for the inclusion of the feature and 0 for the exclusion. Each feature selection $p \in \mathbb{B}^{456}$ (401 for ligands + 55

B not defined, did you mean R? is it 456 for proteins) is called a chromosome. A ”population” of n chromosomes is maintained. For each generation, the best pairs of chromosomes are selected as parents. The next generation is created by crossover and mutation of the chromosomes. Algorithm 1 below gives complete pseudocode for this.

The scoring function used to select the best chromosome can vary according to the type of the model being fit. [See Section 5]

3.2.3 Manual Feature selection

Features selected by expert. Given by Simon Bray. (Yet to do)

Algorithm 1 Selection of features in our model using genetic algorithm [3]

```
procedure GENETIC_ALGORITHM_BASED_SELECTOR
    scoringfunction ← Get model specific scoring function
    $population = \{C_1, C_2, C_3 ... C_n\} \in \mathbb{B}^{456}$ (initial chromosomes).
    best ← $C_1$ // Arbitrarily initialized
    $i \leftarrow 0$
    $gen \leftarrow$ number of generations to run.
    for $i < gen$ with step 1 do
        $\{S_1, S_2, S_3 ... S_n\} \leftarrow$ scoringfunction($population$) $\forall S_i \in \mathbf{R}$.
        // Do a tournament selection for the best chromosomes
        $genetically\_better\_population \leftarrow$ empty list
        for $j < len(population)$ do
            Set ← random_k_selections($population$)
            $c \leftarrow best$(Set) // Based on scoringfunction.
            $genetically\_better\_population.add(c)$
        end for
        $children \leftarrow$ empty list
        for $j < len(population)$ with step 2 do
            $P_1, P_2 \leftarrow population[j], population[j+1]$
            $c_1, c_2 \leftarrow crossover(P_1, P_2)$
            $c_1 \leftarrow mutation(c_1)$
            $c_2 \leftarrow mutation(c_2)$
            $children.add(c_1, c_2)$
        end for
        $population \leftarrow children$
    end for
    return best($population$)
end procedure
```

4 Testing

4.1 Reproducibility

Reproducible ML models are very crucial for verifying any project or research results. Due to the stochastic nature of many ML training processes, reproducing the exact model (and consequently the exact output) is a challenge. Two methods can be employed to produce verifiable results:

- Training many models and reporting the average results.
- Controlling the randomness of the trained models. It is done by setting the seed of the pseudo-random algorithms.

We use the second approach in our project. Every script, when executed, reports an execution ID. This is the random seed used during the execution. If we want to reproduce the exact results, we give this execution ID to the script as the first argument.

4.2 Model Quality Analysis

We are trying to predict the following function

$$\text{Binding affinity prediction} : \mathbf{R}^n \mapsto \mathbf{R} \;\text{ where }\; n \in \mathbf{I}^+$$

Since the input space is multi-dimensional, we cannot fully visualize our model as a function of the input space. To get around this using the following methods

- We report *Coefficient of determination*, $R^2 \in (-\infty, 1.0]$ where 1.0 is the best score. [10]

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \text{ where } \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \text{ and } \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}\epsilon_i^2$$

- For visualizing the results, we plot a 2D plot of expected values against the model's output.

A perfect model would have all the points on the $y = x$ line. This corresponds to R^2 score of 1.0. Figure 4, shows the validation accuracy of a sample *Random Forest Regressor* model. *y_validate* (x axis) represents the actual validation data. *y_validate_pred* (y axis) represents the predicted output from the model.

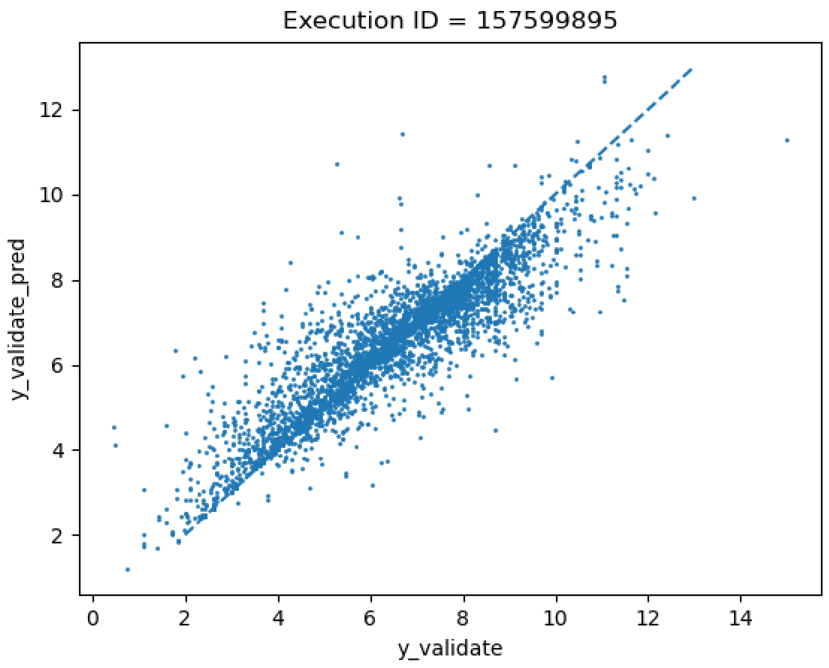


Figure 4: (Sample) Visualizing accuracy. $R^2 \approx 0.805$.

5 Machine Learning Models

Various machine learning models were trained using the extracted data. We tried out Linear Regression, Support Vector Regression, a small Neural Network, as well as a Random Forest Regression. But by far the most impressive performance was given by Random Forest Regression.

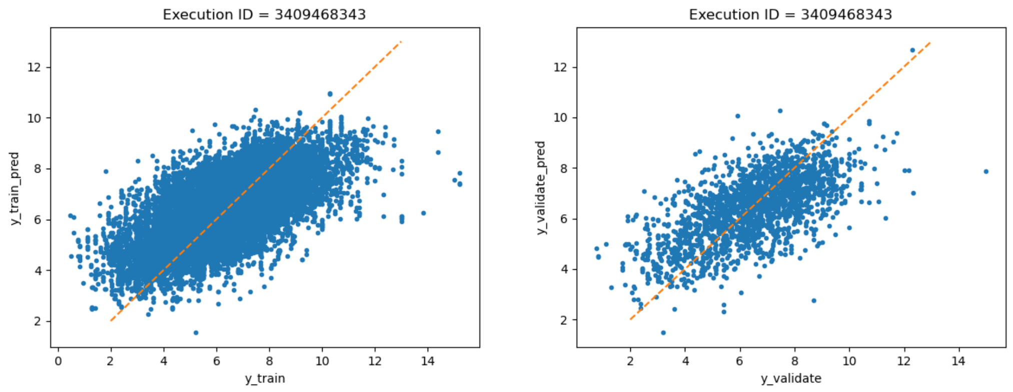
5.1 Linear regression

Linear regression fits a linear model to a given data. It tries to minimize the square of errors between the predicted output value and the actual output value. This is the cheapest model (computationally) that we used to fit our data. Table 1 shows the overview of our analysis with this model. Figure 5 shows the best results obtained with linear regression.

The reason for the low R^2 score is that linear models assume a strong linearity between the input variables and the output variable. This is not always true.

Features Selected	Training R^2 Score	Validation R^2 Score
457 (all)	0.458	0.414
50 (Genetic - Random init)[1]	≈0.379	≈0.359
49 (Genetic - Specific init)[1]	≈0.378	≈0.368
- (Output Correlation)	-	-
- (manual)	-	-

Table 1: Simple linear model overview



(a) Training Accuracy ($R^2 \approx 0.458$)　　(b) Validation Accuracy ($R^2 \approx 0.414$)

Figure 5: Best Linear Model. All 457 features selected.

5.1.1 Score function of genetic feature selection

Since the fitting of the linear regression model was very cheap, we could afford to refit the model every time in our genetic algorithm. We had 2 objectives at hand

- Get the best performing model.
- Reduce the number of input features to make our model simple and explainable.

conflict with

The above objectives may not completely be against each other. This is because removing a feature that has no correlation (or random correlation) with the output may improve the model whereas removing a feature that the output is highly correlated on degrades the model.

[1]Approximately reproducible as the reproducibility module was introduced later.

Hence, taking inspiration from the hypervolume-based multi-objective optimization, we designed the following score function.

$$\text{score} = \mathbf{R}^2 * \text{Features Eliminated}$$

Figure 6 gives us an intuition about this function. Here the score function represents the area of the square formed between a given point and the origin. Trying to improve the score means, it will try to eliminate more features as well as try to get a higher R^2 score. In the example below, the point a is preferred over b as it gives almost the same R^2 score and eliminates a larger number of features. Another advantage of this score function is that the scale of both axes is irrelevant.

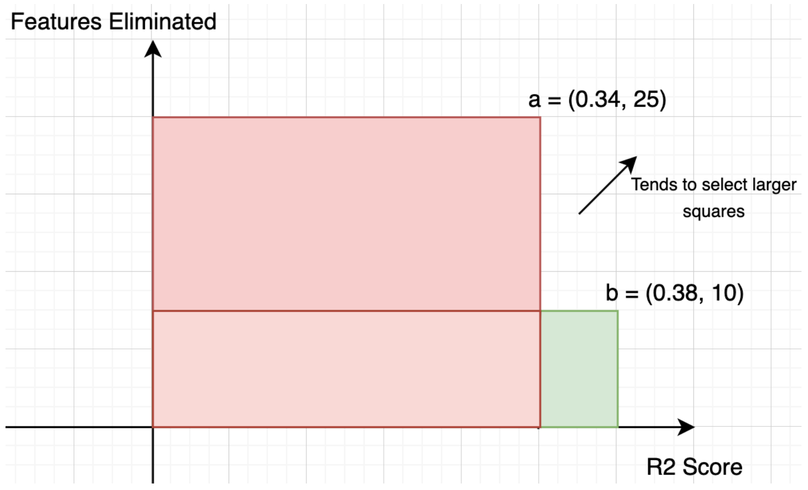


Figure 6: Genetic Algorithm score function representation.

5.1.2 Initialization strategies of population

Two main strategies were used to initialize the population for the genetic algorithm

- **Random Initialization**. A population of 2400 chromosomes was randomly initialized and run for 500 generations.
- **Specific Initialization**. Here, the initial population was kept at 457. The algorithm was run for 2000 generations. Each chromosome in the population had 1 distinct feature excluded. The following represents the initialization of the chromosomes.

$c_1 = [01111......11]$
$c_2 = [10111......11]$
$c_3 = [11011......11]$
...
$c_{457} = [11111......10]$

The rationale was that using the above score function we could go to a local minimum by eliminating the worst performing features rather than trying to select the best performing features as in the case of random initialization.

Both the initialization strategies gave similar results as shown in Table 1

5.2 Random Forest Regression

Random forest regression is an ensemble ML model of regression trees. When training each tree, the algorithm finds out which feature value can divide the data into 2 groups to yield the lowest sum of squared values on both sides (The criterion can be different as well e.g. lowest absolute error). Then each of the sub-data is split recursively till a stopping criterion (e.g a set number of data points in the leaf).

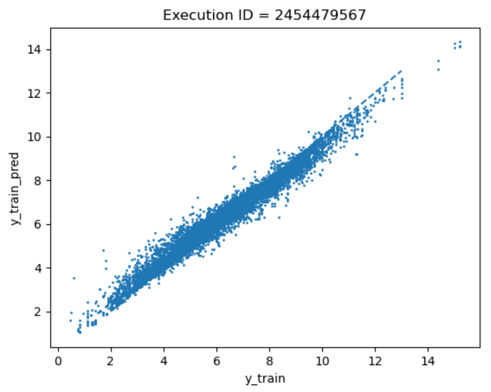
For each tree, a subset of the data is used for training. The subset of data is sampled with replacement a.k.a bagging. After training, the bunch of "Experts" that are good at different data subsets predict the output of new given inputs. The average of the predicted outputs is the result of the whole random forest regressor.

This model is non-linear. One advantage is that there is no need for any assumption about the data. Moreover, random forests can handle categorical features together with the real-valued features very easily. On the negative side, the function represented by the ensemble cannot be easily represented and has to be interpreted as a black-box function.

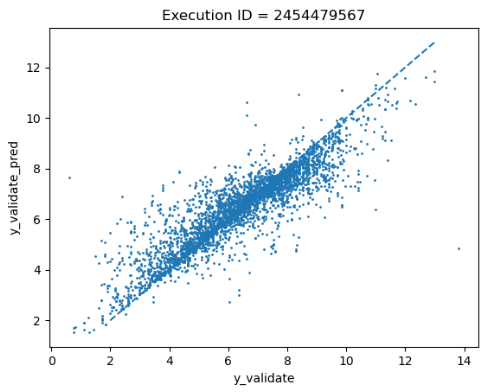
Figure 7 shows the validation and accuracy results obtained by a random forest of 100 trees when we use all the 457 features as input. As you can see, this model far outperforms the linear one.

5.2.1 Feature Importance calculation

The Random Forest regression model fitting is very expensive as compared to the simple linear regression. Hence the same strategy cannot be used for feature selection as used in the linear models. We used the following strategies to determine the importance of features in the model.



(a) Training Accuracy ($R^2 \approx 0.971$)



(b) Validation Accuracy ($R^2 \approx 0.797$)

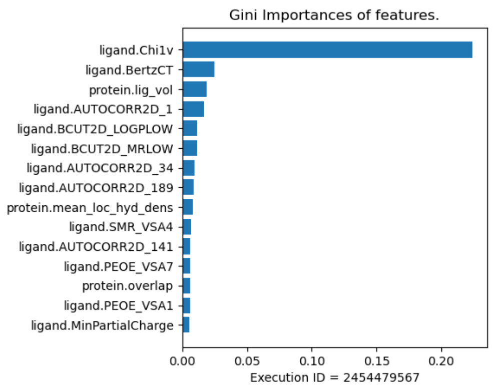
Figure 7: Random Forest Regressor with 457 features and 100 trees.

- **Gini Importance (or) Decrease in impurity**. This is calculated by the model itself. For every feature, it calculates how much decrease in the split criterion the feature contributes to the entire ensemble (i.e in every node its used in all the trees). The disadvantage is that they cannot be reliable when the features have a high cardinality. Figure 8a illustrates this importance.

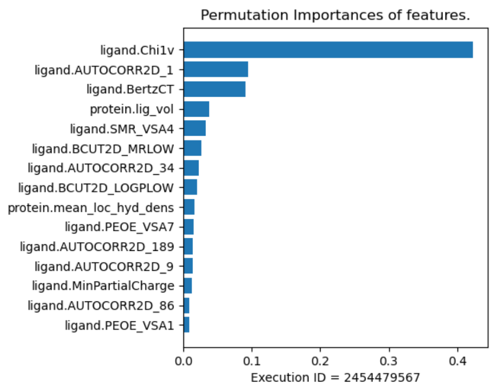
- **Permutation Importance** - This is a model agnostic method. It tries to calculate the reduction in the accuracy we shuffle each feature. The more the reduction the more important the feature is. The disadvantage is that since each feature is checked independently, correlated input features make the results unreliable. If feature A and feature B are correlated, the shuffling of A may not impact the model as the model can rely on B for its prediction. Figure 8b illustrates this importance.

5.3 Permutation Importance and genetic algorithm

To overcome the issue with the above feature importance selection, we plan to use a combination of the concepts of permutation importance and genetic algorithm to find out the importance of the selected features. This is because it is very expensive to refit the model and cheap enough to check the results of shuffling the model.



(a) Gini importance



(b) Permutation importance

Figure 8: Feature Importance calculation of Random Forest Regressor

6 Discussion

7 Conclusion

References

[1] Protein Data Bank. Example Protein - 2Y07. Link. [Online; accessed 1-Aug-2021].

[2] Uni-Freiburg) Björn Grüning (Department of Bioinformatics. Galaxy Tool wrappers. Link. [Online; accessed 1-Aug-2021].

[3] Jason Brownlee. Simple Genetic Algorithm From scratch. Link. [Online; accessed 28-June-2021].

[4] TMP Chem. Computational Chemistry 1.2 - PDB File Format. Link. [Online; accessed 22-July-2021].

[5] Vincent Le Guilloux and Peter Schmidtke. fpocket User Manual. Link. [Online; accessed 2-Aug-2021].

[6] Abdus Salam Khazi. Code for the whole project. Link. [Online; accessed 22-July-2021].

[7] PDBank. PDBank History. Link. [Online; accessed 22-July-2021].

[8] PDBank. PDBank Homepage. Link. [Online; accessed 22-July-2021].

[9] RDKit. RDKit: Open-Source Cheminformatics Software. Link. [Online; accessed 1-Aug-2021].

[10] scikit learn. R2 score, the coefficient of determination. Link. [Online; accessed 22-July-2021].

[11] The Science Snail. Difference between Ki, Kd, IC50 and EC50 values. Link. [Online; accessed 22-July-2021].

[12] Wikipedia. Docking (molecular). Link. [Online; accessed 1-Aug-2021].

[13] Wikipedia. PDBbind database. Link. [Online; accessed 22-July-2021].

[14] Wikipedia. Protein Data Bank (file format). Link. [Online; accessed 22-July-2021].

[15] Wikipedia. Protein–ligand complex. Link. [Online; accessed 24-June-2021].

[16] Wikipedia. Simplified molecular-input line-entry system. Link. [Online; accessed 1-Aug-2021].

[17] Wikipedia. Structure Data File format. Link. [Online; accessed 1-Aug-2021].

[18] Wikipedia. XYZ Format. Link. [Online; accessed 22-July-2021].

[19] Gaming World. Procedural Terrain Generation with Unity : What is Voronoi Tessellation. Link. [Online; accessed 2-Aug-2021].

8 Appendix

8.1 XYZ File format

The following represents the pyridine molecule in the XYZ format.

```
11

C       -0.180226841      0.360945118     -1.120304970
C       -0.180226841      1.559292118     -0.407860970
C       -0.180226841      1.503191118      0.986935030
N       -0.180226841      0.360945118      1.29018350
```

```
C       -0.180226841     -0.781300882      0.986935030
C       -0.180226841     -0.837401882     -0.407860970
H       -0.180226841      0.360945118     -2.206546970
H       -0.180226841      2.517950118     -0.917077970
H       -0.180226841      2.421289118      1.572099030
H       -0.180226841     -1.699398882      1.572099030
H       -0.180226841     -1.796059882     -0.917077970
```

8.2 SDF File format

```
2uzn_ligand

Created by X-TOOL on Fri Nov 18 14:55:27 2016
 37 39  0  0  0  0  0  0  0  0999 V2000
    7.1480   60.4530    6.6830  O 0  0  0  1  0  1
    6.0470   60.1670    7.5640  S 0  0  0  1  0  4
    .......
    .......
   -2.6338   67.4589    8.1225  H 0  0  0  1  0  1
  1  2  2  0  0  2
  2  3  2  0  0  2
  .......
  .......
 23 37  1  0  0  2
M  END
> <MOLECULAR_FORMULA>
C15H13N3O4S2

> <MOLECULAR_WEIGHT>
363.3

> <NUM_HB_ATOMS>
7

> <NUM_ROTOR>
1

> <XLOGP2>
1.31
```

8.3 MOL2 File Format

```
###
### Created by X-TOOL on Fri Sep 26 17:34:18 2014
###

@<TRIPOS>MOLECULE
1fo2_ligand
   25    25     1     0     0
SMALL
GAST_HUCK


@<TRIPOS>ATOM
      1 C4         39.0090   40.2680   25.5130 C.3       1 DMJ          0.1280
      2 O4         39.2170   40.5810   26.8980 O.3       1 DMJ         -0.3835
     .......
     25 H14        38.0787   41.8134   21.3802 H         1 DMJ          0.2097
@<TRIPOS>BOND
     1    1    9 1
     2    1    3 1
    .......
    25   11   25 1
@<TRIPOS>SUBSTRUCTURE
     1 DMJ         1
```