School of Computing
FACULTY OF ENGINEERING

UNIVERSITY OF LEEDS

Comparison of Deep Learning with Multiple Machine Learning Methods for Prediction Using AqSolDB Dataset

Zhanyang Liao

Submitted in accordance with the requirements for the degree of MSC Advanced Computer Science (Data Analytics)

2019/2020

The candidate confirms that the following have been submitted*:*

Items	**Format**	**Recipient(s) and Date**
Deliverables 1	*Report*	*SSO (8/28/2020)*
Deliverable 2	*Software URL*	*Supervisor, assessor (8/28/2020)*

Type of Project: Exploratory Software

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) ___________________________

Summary

The water solubility of a compound is of great significance for the discovery and development of modern drugs.

For a single combination, it has a lot of attributes. After all, the most common and critical factors are octanol-water Partition Coefficient, Molecular Weight, etc. In addition, the data set used in this project has a total of 17 different characteristics.

There have been many attempts in the industry to use machine learning methods to predict the solubility of a compound, and many excellent solutions have been proposed.

However, there is a problem with all methods, that is, one method cannot solve all problems. This is because whether a method is suitable for a data set depends on the size of the data set as well as the number of compounds, complexity, and so on.

Therefore, in this project, seven different machine learning methods will be used to try to train aqueous solubility data, and after viewing and comparing the results produced by each method, it will be decided which method is more suitable for this data set.

Acknowledgements

First of all, I would like to thank Professor Brandon Bennett, my Supervisor, for his guidance and support for this project, as well as our valuable meetings and discussions. Under his guidance, I can finish this project the project well.

I was very grateful to Rob Thomas from the Ihasa Company for providing me with the data for the experiment, as well as giving me many useful Suggestions during the meeting and finally giving me valuable feedback on my project.

I would like to thank my parents, thank you for encouraging me to do my best.

List of Figures

List of Tables

Table of Contents

Chapter 1
Introduction

1.1 Overview

The solubility of water is an important property of chemical substances and occupies an important position in a lot of specialized fields such as geochemistry. Solubility represents the physical conditions under which a solute can dissolve in water, such as temperature and pressure.

High water solubility is of great benefit to drug design, because it can help to avoid some solute precipitation that leads to experimental errors and ultimately to the failure of clinical development (Sorkun et al., 2019).

1.2 Aims and Objectives

This project intends to use several different machine learning algorithms and deep learning network to train data to explore which algorithm the model can perform better under. At the same time, the characteristics of various methods in different situations would be explored.

During the experiment, data will be trained once under each method, and corresponding indicators will be provided to test the performance of the model, such as R-Squared, Mean Square Error, etc.

The objectives of this project are:

1. Find the most suitable method for training data set and put it into practical use under several methods.

2. Compare and summarize the characteristics of various algorithms.

1.3 Motivation

There are many companies in the chemical industry that are using computer technology to predict the properties of drugs, such as water solubility. People have already used basic machine learning algorithms to make predictions.

However, with the development of ensemble learning, people began to use GBDT, Xgboost for prediction, such machine learning methods can effectively control the overfitting and the accuracy is also higher.

Deep learning has also evolved over the years, and many people are using it for a variety of business problems.

Using these techniques can save the cost of chemical experiments and save time by calculating the solubility directly from its chemical properties. Therefore, combine computer technology with chemistry can effectively improve industrial development.

1.4 Project Methodology

CRISP-DM (Cross-industry Data Mining Standard process) reference model is implemented throughout the project, this method is the main idea of project analysis. Data mining is a creative process, and a standard framework can avoid being completely dependent on a specific person or team executing the project, and instead want the entire model to be in the hands of anyone. CRISP-DM has solved some of these problems by providing a framework for data execution(Wirth and Hipp, 2000).

This project will be based on crisp-DM model, and it has six stages: Business Understanding, Data Understanding, Data Preparation, Modelling and Evaluation, Deployment.

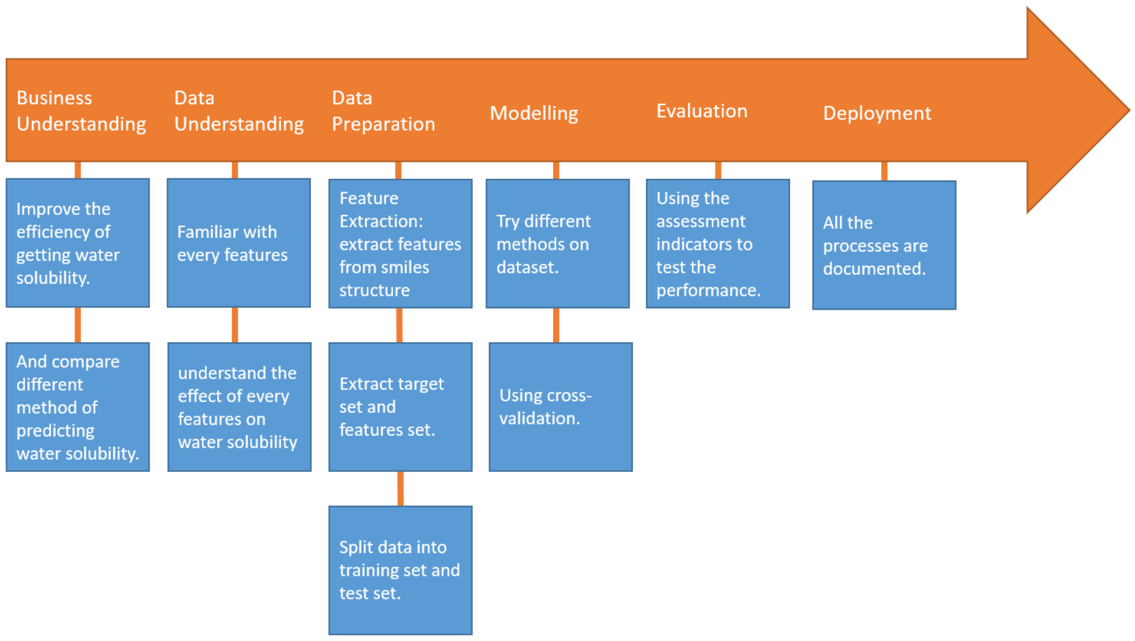

Business Understanding
Data Understanding
Data Preparation
Modelling
Evaluation
Deployment
Improve the efficiency of getting water solubility.
And compare different method of predicting water solubility.
Familiar with every features
understand the effect of every features on water solubility
Feature Extraction: extract features from smiles structure
Extract target set and features set.
Split data into training set and test set.
Try different methods on dataset.
Using cross-validation.
Using the assessment indicators to test the performance.
All the processes are documented.

Figure 1.1 Adapted CRISP- DM model for this project

Figure 1 shows the general steps of the six data mining phases.

Business Understanding: In this project, the business purpose is that a machine learning model will be created to predict the number of aqueous solubility. The purpose of such experiments is to save time and improve the work efficiency of those who work in the chemical industry. This is because using chemical methods to test solute solubility is a cumbersome process. Therefore, the additional artificial intelligence technology can greatly improve the time used to test the solubility of compound.

At the same time, another purpose is to find a data mining method suitable for this type of data to make its predictive ability as high as possible.

Data Understanding: The main task is to familiarize the data, understand the chemical implications of each feature and its effect on water solubility. Combine the data understanding with the business goals to form the initial insights from the data.

Data Preparation: Feature extraction is carried out in the data preparation stage. Some features of the data set need to be reextracted. Meanwhile, the target set and feature set in the data set are distinguished, and the data set is divided into the data set and the test set in equal proportion according to its reliability level.

Modelling: During the modelling phase, several different data mining methods will be used, including basic machine learning methods, integrated learning, and neural networks to train the data set. The cross-validation method will also be used to improve the training accuracy.

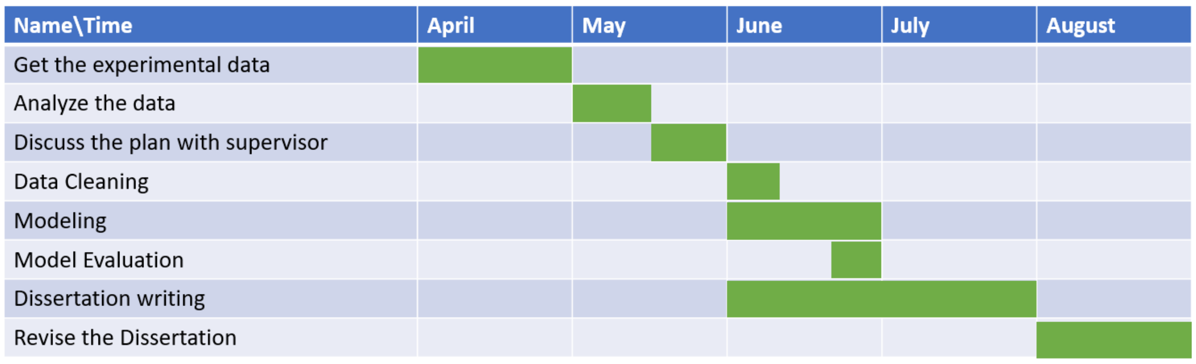
Evaluation: The training results were evaluated using the assessment indicators which commonly used in the industry in the prediction model, and the corresponding scores were then compared crosswise.

Deployment: The results and knowledge gained from each of the above phases must be documented so that others can use the currently created model later. This project report is considered a document and no further system deployments.

1.5 Project Management

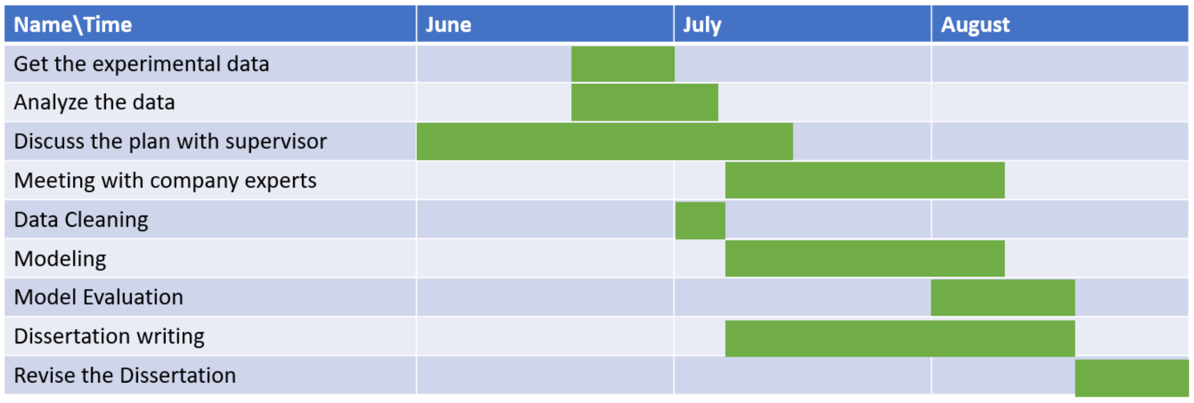
Since this project is to cooperate with the enterprise, the data needed for modelling needs to be provided by the partner company. However, due to the influence of Coronavirus and some problems within the enterprise, the project has been stalled until June. It wasn't until the end of June, when the data was actually available, that the project officially began. Moreover, the meeting with mentors and experts in enterprises can only be held online, which will affect the efficiency of discussion.

Therefore, the actual project runtime is shown in Figure 1.3.



Name\Time	April	May	June	July	August
Get the experimental data	🟩				
Analyze the data		🟩			
Discuss the plan with supervisor		🟩			
Data Cleaning			🟩		
Modeling			🟩		
Model Evaluation			🟩		
Dissertation writing			🟩	🟩	
Revise the Dissertation					🟩

Figure 1.2 Original Project Schedule



Name\Time	June	July	August
Get the experimental data	🟩		
Analyze the data	🟩	🟩	
Discuss the plan with supervisor	🟩	🟩	
Meeting with company experts		🟩	🟩
Data Cleaning		🟩	
Modeling		🟩	🟩
Model Evaluation			🟩
Dissertation writing		🟩	🟩
Revise the Dissertation			🟩

Figure 1.3 Revised Project Schedule

1.6 Deliverables

1. A Kaggle notebook that contains the source code of the system.
2. The MSc project report

1.7 Ethical Issues

All data used in this project are published by Kaggle and are free from any illegal activities.

The tools used in this project are all open source tools, which can be directly obtained for free on the Internet without any illegal use.

1.8 Project Structure

The report is designed as follows:

Chapter 1 focuses on the development objectives and objectives of the project, as well as the machine learning algorithms that can be used in the project. In addition, the management mode of the project will be introduced. Finally, the planned time of the project and the structure of the paper will be shown.

Chapter 2 provides details about Background research, including basic machine learning algorithms, such as Lasso regression, Support Vector Regression, etc., integrated learning methods, GBDT, Xgboost, etc., and finally, deep learning. This section mainly discusses the related concepts, operating modes, the model frameworks, and how to use them.

Chapter 3 is the project design. This section discusses the technologies and platforms needed to write software. The design process and evaluation method of each task of the project are described in detail.

Chapter 4 is the experiment and result of this project. According to the project design, the project experiment was implemented and the experimental results under each method were obtained, which were represented by some evaluation indicators, the most important of which was the MSE.

Chapter 5 is the project evaluation. The evaluation matrices are used to analyse the results produced by each method, and then the results are visualized. Some examples are extracted for testing, and the results are visualized.

Chapter 6 includes the conclusions of the project and areas for improvement in the future. It will discusses the conclusions of the project, the limitations of the project, and possible directions for improvement.

Chapter 2
Background Research

Many machine learning methods will be used during the project. After the model is built, some evaluation metrics will also be used to evaluate the performance of the models. Therefore, this chapter includes basic machine learning methods, ensemble learning, deep learning and evaluation matrices (Lovric et al., 2020).

2.1 Literature Review

In recent years, people in the aqueous solubility of forecasting problems are actively looking for the best solution. If a compound is low in water solubility, it not only creates problems for in vitro and in vivo testing in drug discovery, but also increases the economic cost of new drug development and puts a burden on researchers and companies. Therefore, drug solubility is very essential in drug discovery and development (Di et al., 2012).

Prior to the application of computer technology, the solubility equation derived from thermodynamics was commonly used for the solubility prediction of hydrophobic substances (Ruelle and Kesselring, 1997). In addition, aqueous solubility is also predicted by using semi-equilibrium, and equilibrium methods (Di et al., 2012).

After applying machine learning methods, many people have applied many different methods to predict solubility. This includes not only the methods used in this project, but also the methods not used in this project.

Lasso regression has been used by Lovric et al. (2020) to predict a collection of publicly intrinsic water solubility data for 829 pharmaceutical compounds, with the help of multilevel permutation feature selection. Finally, using Lasso Regression on those dataset got a good result, which exceeded some integrated learning methods.

Lind et al. (2003) proposed that SVR is utilized to estimate the solubility of organic compounds in water. The prediction ability of SVR will be affected by kernel function, Thus, Tanimoto similarity kernel was used as the kernel function, which is used to predict ARIZONA Database of Aqueous Solubility (Yalkowsky and Dannelfelser, 1990), finally R2 = 0.88.

Decision Tree is also used to predict the solubility of a compound to solve business problem. Xia et al. (2003) proposed that decision tree can improve the cost-effectiveness of selecting supplier compounds suitable for MRI screening, increasing the successful selection rate of compounds from 25% to 50%.

Palmer et al. (2007) in the use of the Random Forest to build the QSPR model to predict aqueous solubility after (based on 988 data of organic molecules), found that the ability to predict than Partial - further - Squares regression, Support Vector Machine and Artificial Neural Networks predictive ability was stronger.

Others use ANN to predict the aqueous solubility, according to the experiment of Huuskonen et al. (1998), find ANN for solubility prediction ability is very strong, but need more huge data and the kinds of compounds widely as a training set, otherwise the ANN predictive power can only be limited to certain compounds of similar structures.

In addition to the machine learning methods described above, there are many people who use methods not covered in this project. For example, Lovric et al. (2020) proposed that using LightGBM can effectively predict aqueous solubility. According to the experiment of Khurana et al. (2018), who can also use CNN to predict the solubility of the protein based on its sequence.

Overall, there are many people use various methods to build QSPR models for aqueous solubility prediction, it is impossible to deal with all problems in one way (Mitchell, 2014). The predictive power of each method was based on characteristics such as the volume of the data set they trained and the distribution of the categories of compounds.

Therefore, people do not usually apply only one method to predict the solubility, but apply multiple machine learning methods and compared using evaluation matrices, and the method with the highest R-squared value is selected as the best solution as the final prediction model, which is also consistent with the theme of the project.

2.2 Machine Learning

2.2.1 Linear Regression

Lasso Regression used in the experiment is one of the Linear regressions, because we used the ordinary Linear Regression method, which actually use the least squares Linear Regression method without Regularization (weight penalty)(Xu, 2019).

In Lasso Regression, L1 Regression is used to regularize the weight. Sparse weight matrix, namely the sparse model, can be generated. L1 can also prevent the occurrence of overfitting to a certain extent.

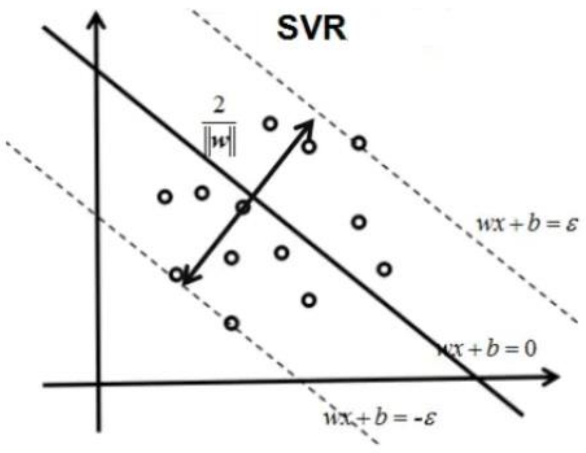
$$\frac{1}{2m}\sum_{i=1}^{m}(y-Xw)^2+alpha\sum_{j=1}^{p}\left|w_j\right|$$

Figure 2.1 Lasso Regression

Generally speaking, the reason for using Lasso Regression may be that not every feature in the data set is important and the data size is relatively large.

2.2.2 Support Vector Regression

Support vector machines are used to solve a classification problem. However, the basic characteristics of SVM also apply to regression, so SVM regression model also exists.(Demiriz et al., 2001) In SVM classification, it should involves fitting the maximum possible decision boundary between two categories while limiting margin violations. In SVM regression, it attempts to fit as many data instances as possible on the decision boundary and limit margin violation- that is, far away from the gap.



$$\begin{cases} \min \frac{1}{2}\|w\|^2 \\ s.t. \left|y_i - (wx_i + b)\right| \le \varepsilon, \quad \forall i \end{cases}$$

Figure 2.2 SVM regression

In general, the traditional regression method is considered to be correct when f(x) is exactly equal to y and the difference between the real value and the actual value, i.e., the loss value, needs to be calculated. Support vector regression, on the other hand, holds that as long as f(x) does not deviate too much from Y, the prediction can be regarded as correct and loss is not calculated. This is done by setting a threshold alpha, calculate the loss of a data point when $|f(x) - y| > \alpha$.

When we have some data that is not linearly distributed.

For nonlinear support vector regression machines, the usual approach is to scaling up the data space, find the linear separable hyperplane data space with high-dimension, and finally map the hyperplane in the high-dimensional space back to the low-dimensional space, so as to achieve SVM classification or SVR regression.

However, scaling up the data and doing computation in the high-dimensional data space is computational expensive, especially when the dimensions are very high, and it is also easy to overfit.

The kernel function is generated to deal with the problem. Replacing the linear term in the linear equation with kernel function can make the original linear algorithm nonlinear, that is, nonlinear regression can be done. At this point, the kernel function is introduced to achieve the purpose of dimension raising, and the overfitting can also be effectively controlled.

In layman's terms, a kernel is a computation performed on the data in the lower dimensions, and this computation can be viewed as a computation performed by mapping the data in the lower dimensions to the higher dimensions.

Among Sklearn, kernel types mainly include "linear", "poly", "RBF", "Sigmoid", and "RBF" by default.

2.2.3 Decision Tree

In decision tree modelling, the tree actually represents the recognition order of the features. This model generate a set of rules for prediction through repeated segmentation processes. The most common tree methods are Chi-square automatic interaction detection, classification regression tree, C4.5 and C5.0 (Tso and Yau, 2007).

One strength of decision trees over other machine learning algorithms is that they generate models that can represent interpretable rules or logical statements. Moreover, this model can be applied to both linear data and nonlinear data, and provides clear basis information for classification and prediction. Thus, this model is very simple and easy to use.

2.3 Ensemble Learning

Ensemble learning solves three important problems:

1. Statistical Problem: It allows to select multiple hypotheses that perform well in the training data as output, ensuring that the learning algorithm can perform equally well in the prediction of new data. Single hypothesis runs the risk of performing poorly on the new data.

2. Computational problems: Normal learning algorithms cannot guarantee finding the best solution in the current hypothesis space. Therefore, ensemble learning can improve accuracy by adopting multiple assumptions.

3. Representational Problem: Ensemble Learning makes it easier to find a more accurate approximation to the true function through voting (Dietterich and networks, 2002).

The ideas of Bagging and Boosting are very important in ensemble learning.

Bagging: Also known as Bootstrap Aggregating, is used to improve the accuracy and makes the model more generalize by reducing the variance i.e. by avoiding overfitting. This is generated by random sampling of the replacement of the original set (Mujtaba, 2020).

Boosting: Boosting urges to combine several simple decision tress, and each tree complements the previous one to avoid redundancy. Therefore, boosting needs to keep track of the errors of the previous trees.

2.3.1 Random Forest

Random forest is a bagging technique and not a boosting technique. There is no connection between these trees.

The principle of random forest is also very simple. Each decision tree classifier in a random forest produces a result, which is then aggregated into a set of random forests. All of these results are voted or averaged into a single decision model, and the final result trumps the output of any single decision tree (Chakure, 2019).

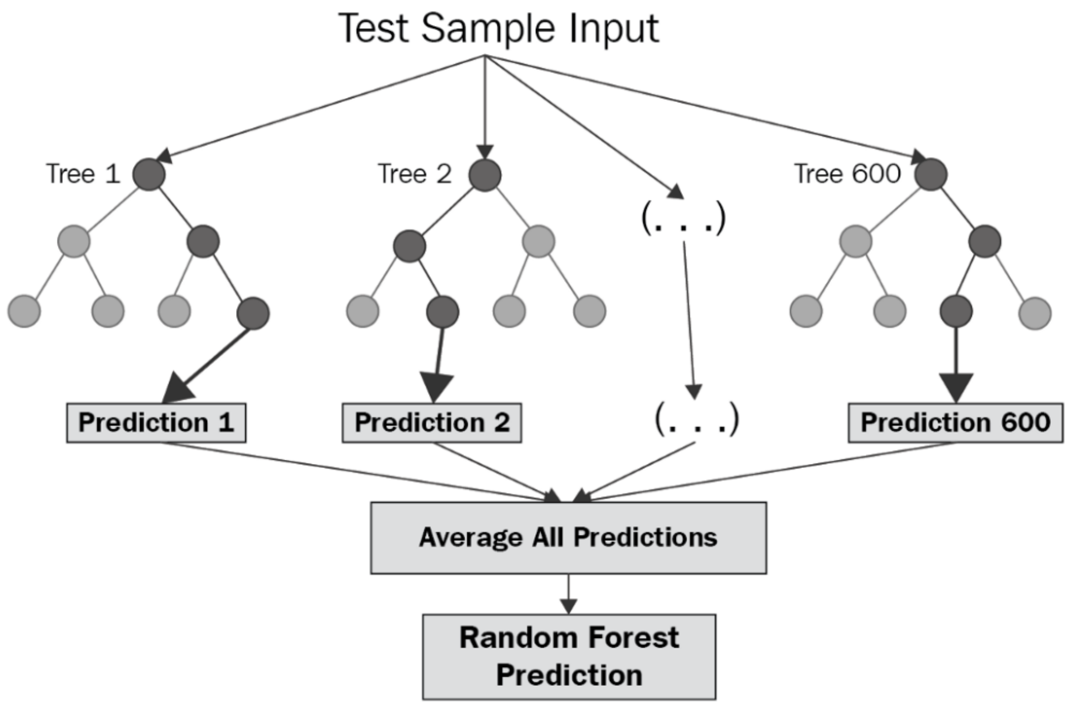


Figure 2.3 Random Forest Regression

Advantages of random forest:

1. It is applicable to many different types of data sets and performs well in terms of high accuracy.

2. It can run efficiently on large databases.

3. It can handle thousands of input variables without having to delete them.

4. It provides an estimate of the important variables in the classification.

5. With the progress of forest construction, it will generate the unbiased estimation of the internal generalization error.

6. It is an effective method of estimating lost data and maintaining accuracy when most of it is lost.

Disadvantages of random forest:

1. It has been observed that random forests are too suitable for some data sets with noisy classification/regression tasks.

2. When solving regression problem, the performance of random forest is not as good as when solving classification problem. When the new data has exceeded the scope of the training set data, the random forest may have no way to predict successfully or the prediction ability is very weak. This may lead to the fitting of some noises in the training of the random forest model, resulting in the poor generalization ability of the model.

2.3.2 Gradient Boost Decision Tree

GBDT algorithm which uses boosting method to combine individual decision trees. Each tree tries to minimize errors in the previous tree. Boosting tree is a tree with weak learning ability, but with many trees added in a row, each tree is focusing on the errors of the previous tree, so the model can be built efficiently and accurately. Boosting, which, unlike Bagging, involves Boostrap sampling, both agree with the modified version of the original data set when boosting the new tree each time (Yıldırım, 2020).

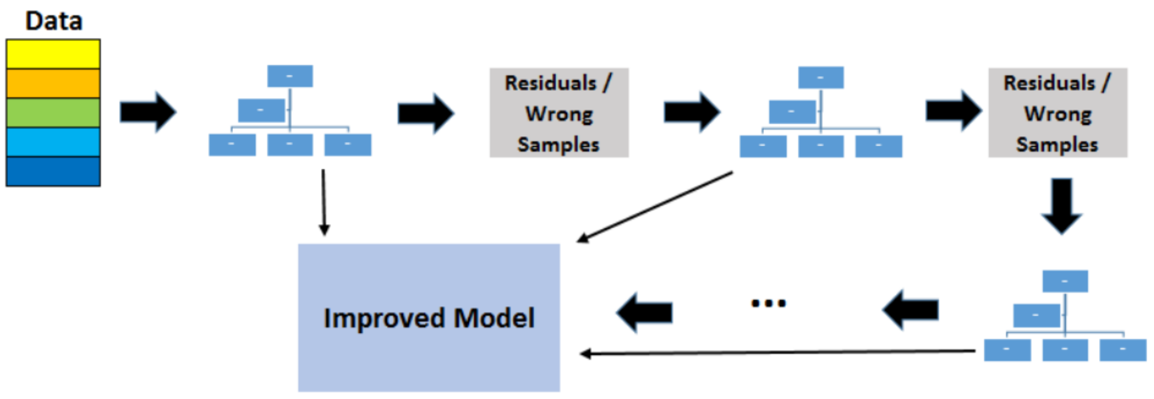


Figure 2.4 The Process of GBDT

Advantages of GBDT:

1. Has the strong prediction ability, the accuracy is high.

2. It can be used even if the dimension of the data is not high

3. Can handle nonlinear data

4. It can deal with discrete values and continuous values.

Disadvantages of GBDT:

1. Careful tuning is required, and the training may take a long time.

2. Like other tree-based models, it is generally not suitable for high-dimensional sparse data

2.3.3 Xgboost

Xgboost is an effective and extensible implementation of the gradient propulsion framework proposed by Friedman (2001). The package includes an efficient linear model solver and a tree learning algorithm. It supports regression, classification, sorting and other objective functions. The package is extensible so that users can easily define their own goals as well.

The difference between GBDT and Xgboost:

1. While traditional GBDT uses CART tree as a base learner, Xgboost also supports linear classifiers.

2. Shrinkage, which is learning rate. Xgboost multiplicate the weight of the leaf nodes by this factor at the end of an iteration, mainly to reduce the impact of each tree and allow more room for learning later.

3. Processing of missing values. Xgboost can also automatically learn the splitting direction of a sample that has a missing value for the feature.

2.4 Deep Neutral Network

In the field of drug discovery, the use of neural networks is getting more and more attention (Baskin et al., 2016). The deep learning approach is a powerful complement to classic machine learning tools and other analytical strategies. These methods have been used in many applications in computational biology, including regulatory genomics and image analysis(Angermueller et al., 2016).

When we extend the hidden layer from the simple single-layer neural network to multiple layers, we get the deep neural network.

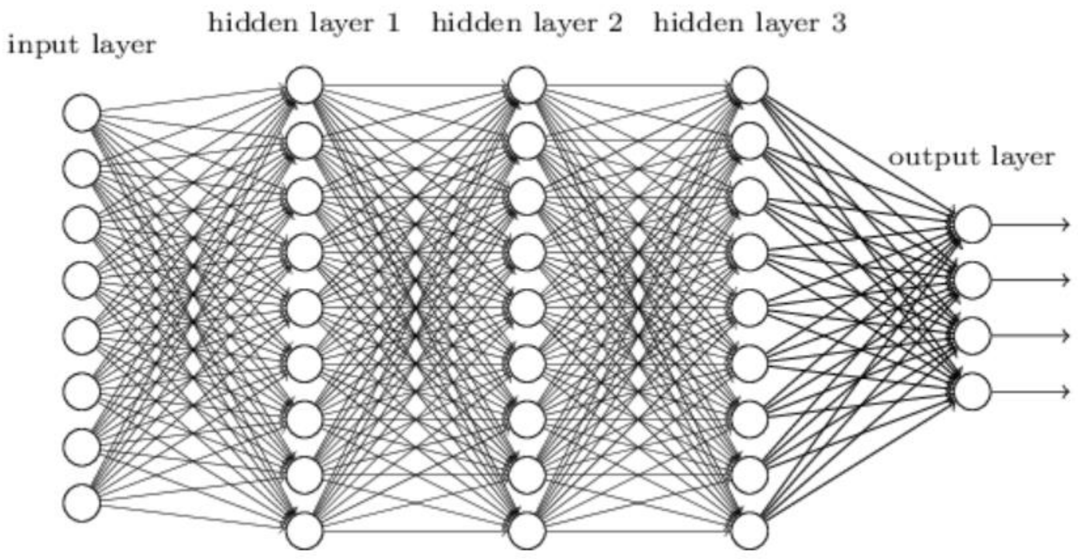


Figure 2.5 Deep Neutral Network

2.3.1 Forward Propagation

Assuming that the activation function we choose is σ(z), and the output value of the hidden layer and the output layer is ‘a’, then for the three layers DNN in the figure below, we can use the output of the previous layer to calculate the output of the next layer using the same train of thought as the perceptron, namely the so-called DNN forward propagation algorithm.

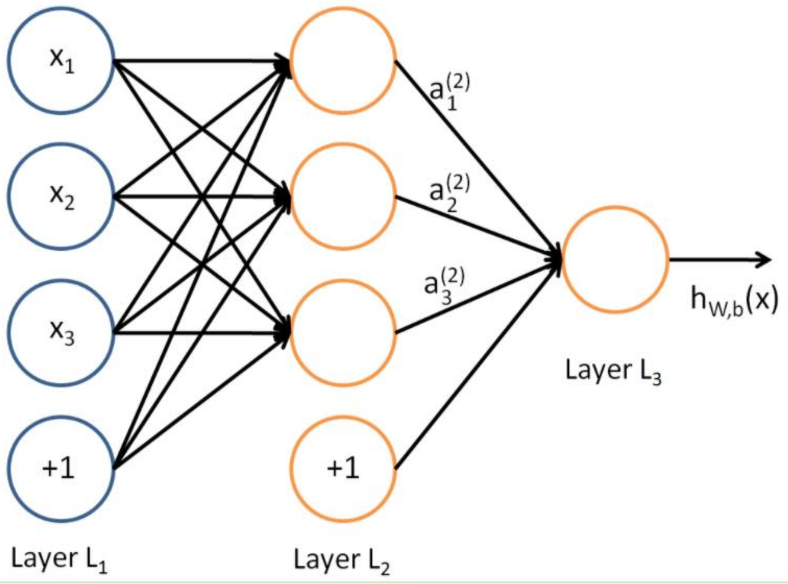


Figure 2.6 Forward Propagation

For the output of the second layer:

$$a_1^2 = \sigma(z_1^2) = \sigma(w_{11}^2 x_1 + w_{12}^2 x_2 + w_{13}^2 x_3 + b_1^2)$$

$$a_2^2 = \sigma(z_2^2) = \sigma(w_{21}^2 x_1 + w_{22}^2 x_2 + w_{23}^2 x_3 + b_2^2)$$

$$a_3^2 = \sigma(z_3^2) = \sigma(w_{31}^2 x_1 + w_{32}^2 x_2 + w_{33}^2 x_3 + b_3^2)$$

Figure 2.7 the output of the second layer

For the output of the third layer, the following contents can be obtained:

$$a_1^3 = \sigma(z_1^3) = \sigma(w_{11}^3 a_1^2 + w_{12}^3 a_2^2 + w_{13}^3 a_3^2 + b_1^3)$$

Figure 2.8 the output of the third layer

It can be seen from the above, it is assumed that layer of l - 1 m neurons, and the first layer of a total of n l neurons, the first layer of linear coefficient w a l n * m matrix Wl, the first layer of the bias b l formed a vector n x 1 bl, first l - 1 form a layer of the output of a vector al - 1 m x 1, the first l layer before activation of linear output vector z formed a n * 1 zl, of the first layer of the output of a l a n x 1 vector al.

Then, the output of layer L is expressed by matrix method:

$$a_j^l = \sigma(z_j^l) = \sigma(\sum_{k=1}^{m} w_{jk}^l a_k^{l-1} + b_j^l)$$

Figure 2.9 the output of the l th layer

2.4.2 Back Propagation

When we train a neural network, we need to find appropriate linear coefficient matrix W and bias vector B corresponding to all hidden layers and output layers, so that the calculated output of all training sample inputs is equal to or close to the sample output as far as possible. The loss of the training sample needs to be measured using the loss function, and then the value of the loss function needs to be minimized. The training effect is best when the loss function reaches the minimum value. The series for the corresponding linear coefficient matrix W and the deviation vector B is our final result.

In DNN, the most common loss function optimization extremum solution process is generally completed step by step through the gradient descent method. There are many loss functions available for DNN to choose from. To focus on the algorithm, we use the most common mean square error to measure the loss.

That is, for each example, we want to minimize the following formula :(see figure 2.10).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Figure 2.10 Mean Square Error

Then we need to adjust the values of the weights and bias to minimize the value of the error function(see Figure 2.11).

$$\mathrm{w} = \mathrm{w} - \nabla_w$$
$$\mathrm{b} = \mathrm{b} - \nabla_b$$

Figure 2.11 Gradient Decent

2.5 Evaluation Matrices

2.5.1 Mean Square Error

The range of MSE is [0,+∞), when the real value is exactly the same as the predicted value, MSE is equal to 0, that is, the model is perfect. The greater the error between the real value and the predicted value, the greater the MSE(see figure 2.10).

2.5.2 Root Mean Square Error

The mean of the sum of squares of errors between the predicted data and the original data.

The tree value is the square root of MSE, which is a better description of the data. For example: to do the housing price prediction, each square is ten thousand yuan for the unit, the prediction result is also ten thousand yuan for the unit. Then the square of the difference should be in the tens of millions. Therefore it's not easy for us to describe the effects of the models. Taking the square root of this so that the result of the error is at the same level as the data, which is easier to explain when describing the model.

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

Figure 2.12 Root Mean Square Error

2.5.3 Mean Absolute Error

The Mean Absolute Error is actually a more general form of the mean of the error. MAE and RMSE can be used together because RMSE is sensitive to outliers. When we work with large data sets, we cannot check each value to see if there is one or some outliers, or if all errors are systematically higher. Looking at the MAE/RMSE ratio can help us understand whether there are large but uncommon errors.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

Figure 2.13 Mean Absolute Error

2.5.4 R2 score(coefficient of determination)

The determination coefficient reflects how much of the variation in y can be described by the variation in X, that is, how much of the variation in y can be represented by the controlled independent variable X.

$$R^2 = 1 - \frac{\sum_i (\hat{y}_i - y_i)^2}{\sum_i (\hat{y}_i - y_i)^2}$$

Figure 2.14 R2 score

Chapter 3
Implementation Setup

3.1 Dataset Description

3.1.1 Data Collection

AqSolDB is a database on the detailed properties of compounds. The data in this database has high fidelity, that is, the accuracy of the data is very high and the reliability is very strong. This is because data validation is performed while filtering the data, using a generic algorithm that consolidates each data source (Sorkun et al., 2019). This makes every piece of data in AqSolDB applicable to experimental research, which can help improve the predictive ability of machine learning methods.

Therefore, in the data collection phase, AqSolDB collected nine publicly available data sets(Sorkun et al., 2019), and converted them into standard formats, and discarded some unusable data through data validation. The final data statistics are as follows:

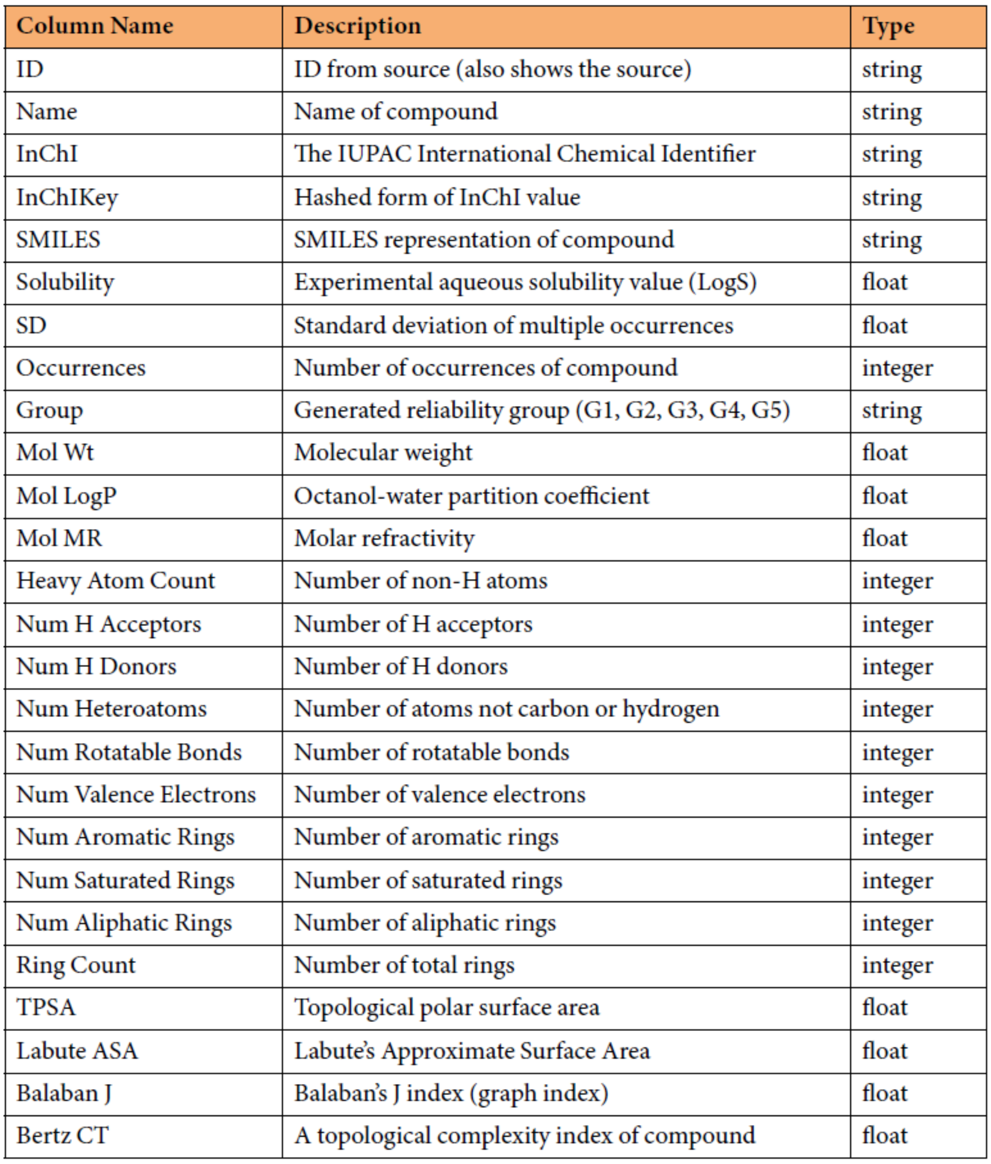
Table 3.1 AqSolDB Data Statistics

Dataset ID	Original Size	Filtered Size	Unique Size
A	14,180	6,110	3,656
B	5,764	4,651	3,542
C	2,603	2,603	1,325
D	2,267	2,115	163
E	1,291	1,291	249
F	1,210	1,210	798
G	1,144	1,144	139
H	578	578	64
I	105	94	47
SUM	29,142	19,796	9,983

As we can see from the table, each data set starts with a certain amount of data (29,142 rows in total), and after the data is verified, there is much less normal data left (19,796 rows). After the process of de-duplicating the data, there are only 9, 983 rows left, which is the data we use for the training.

3.1.2 Data integration

AqSolDB consolidates these nine data sets into one data set for training and experiments. Unify the features for each dataset, as well as the data types so that the nine datasets can be connected. The following are the features description of AqSolDB:



Column Name	Description	Type
ID	ID from source (also shows the source)	string
Name	Name of compound	string
InChI	The IUPAC International Chemical Identifier	string
InChIKey	Hashed form of InChI value	string
SMILES	SMILES representation of compound	string
Solubility	Experimental aqueous solubility value (LogS)	float
SD	Standard deviation of multiple occurrences	float
Occurrences	Number of occurrences of compound	integer
Group	Generated reliability group (G1, G2, G3, G4, G5)	string
Mol Wt	Molecular weight	float
Mol LogP	Octanol-water partition coefficient	float
Mol MR	Molar refractivity	float
Heavy Atom Count	Number of non-H atoms	integer
Num H Acceptors	Number of H acceptors	integer
Num H Donors	Number of H donors	integer
Num Heteroatoms	Number of atoms not carbon or hydrogen	integer
Num Rotatable Bonds	Number of rotatable bonds	integer
Num Valence Electrons	Number of valence electrons	integer
Num Aromatic Rings	Number of aromatic rings	integer
Num Saturated Rings	Number of saturated rings	integer
Num Aliphatic Rings	Number of aliphatic rings	integer
Ring Count	Number of total rings	integer
TPSA	Topological polar surface area	float
Labute ASA	Labute's Approximate Surface Area	float
Balaban J	Balaban's J index (graph index)	float
Bertz CT	A topological complexity index of compound	float

Figure 3.1 Features Description(Sorkun et al., 2019)**.**

We can see from the above table that the feature set of the database mainly consists of the compounds Occurrences (ID, Name, InChI,etc.), reliability (SD,Occurrences,Group) and properties (Solubility,Mol Wt,Mol LogP, etc.). What we mainly use in the training process is the properties of compounds. How to divide the data set will be discussed later.

In the process of integration, many compounds are repeated, so we need to take some methods to integrate the data of those repeating compounds. After synthesizing those data, the reliability of the compound data will be improved, while the compound data that appear only once will have lower reliability. As a result, AqSolDB also classifies the reliability of those compounds, with G1 being the least reliable and G5 the most reliable.

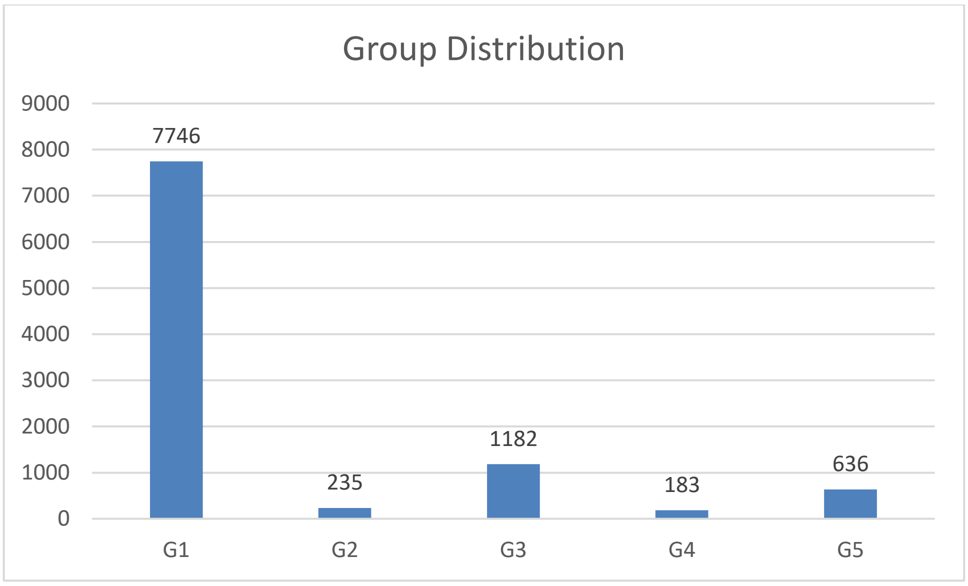


Figure 3.2 Group Distribution

As you can see from the table, only 636 of the more than 9,000 pieces of data are highly reliable (classified as G5), and most of the data is classified as G1, meaning that most of the compounds appear only once in the database.

3.1.3 Summary

In general, this is a very complete chemical database, which can provide comprehensive information for the experiment. Moreover, the data cleaning part has been completed, so it is no longer necessary to clean the data in the followed experiment.

3.2 Software Tool

3.2.1 Coding Environment

This experiment runs entirely on the Kaggle platform. Kaggle is a platform for data modelling competitions. Data scientists can use this platform to challenge tasks published by companies or researchers. On such a platform, competitors can create a notebook directly on Kaggle and run their own code using Kaggle's cloud resources (CPUs, GPUs), allowing people who might otherwise have poor hardware situation to learn data science.

3.2.2 Programming language

The computer language used for this project is Python. Python is dynamic object-oriented Programming language (Amos, 2020). Compared with R, MATLAB, SAS, Stata and other tools, Python is a more comprehensive tool, which provides users with very convenient tools and simple ways to use it not only in data analysis and interaction, but also in exploratory computing and data visualization. Here are the Python libraries used in this project:

Table 3.2 Python Libraries used in this project

Python Library	Description	Usage in the project
numpy	• Supports array and matrix operations • Provides a large library of functions and methods for digital operations	• Manipulate array • Convert array format, type
Pandas	• Solve data analysis tasks • Provide efficient data set operations • Provides a large number of functions and methods to process data	• Data read/ write (csv) • Data storage and manipulation in Pandas DataFrame
Scikit-learn	• Some common machine learning methods are encapsulated, and most machine learning tasks can be accomplished by simply calling the modules in Sklearn.	• Invoke the machine learning algorithm • Invoke the evaluation metrics module • Invoke cross validation • Invoke the data processing tool
Keras	• Keras is an open source artificial neural network library for the design, debugging, evaluation, application, and visualization of deep learning models	• Create deep neutral network
Matplotlib	• Data Visualization	• Data Visualization
Seaborn	• Visualization library based on Matplotlib	• Generate visualizations from the data

Chapter 4
Implementation

4.1 Implementation Overview

This part shows the realization process of the project in the way of flow chart. The flow chart is shown below:

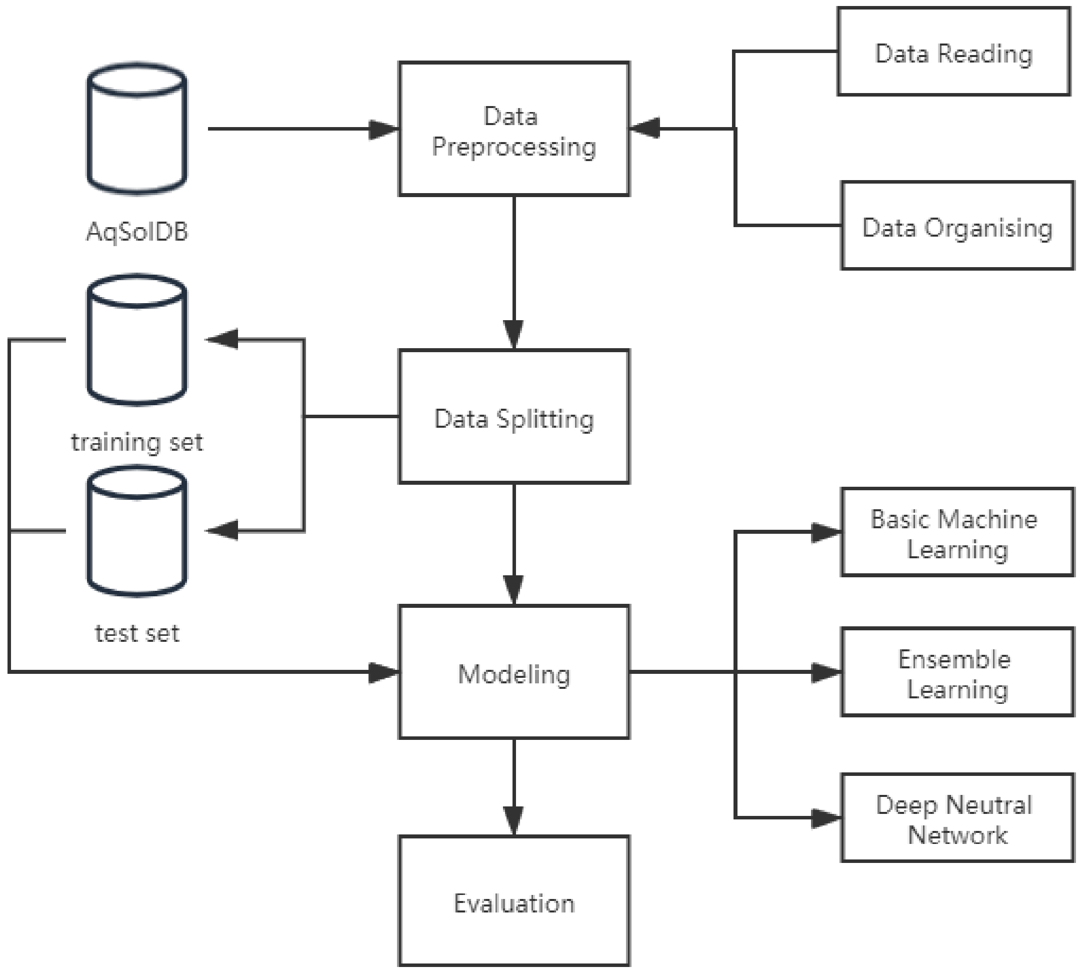


Figure 4.1 Overview of implementation steps

4.2 Data Pre-processing

This part is the preparation process of data, mainly reading data and deleting irrelevant data.

Data Reading: This part mainly uses Pandas to read CSV files. This data file is derived from The Kaggle platform, so the required data can be retrieved directly from the Kaggle folder using the “read_csv” function.

Data Organising: Delete the data from the data set that you are not going to train, such as' ID ', 'Name', 'Occurrences' and' Smiles'. Although many information of the compounds can be obtained from the “Smiles” structure, the numerical information provided by AqSolDB is sufficient for training, so the feature of Smiles was also deleted here.

The features “Solubility” and “Group” were extracted from the dataset for later process. The feature “Solubility” is the labels that we need to predict. And the feature “Group” is used for splitting the data in the next step.

4.3 Data Splitting

Since the function of the feature "Group" is to represent the reliability strength of the data of this compound, we should obtain the same proportion of each compound with different reliability to ensure the generalization ability of the training model.

Therefore, we first separate the data of each group, and then conduct the training set: the test set is divided in a ratio of 8:2, and finally integrate the training set and test set of each group into a complete training set and a complete test set.

And then I'm going to transform the tag set Y into a one-dimensional array. Standardize X and Y, namely linear transformation of data, which can improve the performance of data.

$$\frac{X_i - \mu}{\sigma}$$

Figure 4.2 Normalization

4.4 Modelling

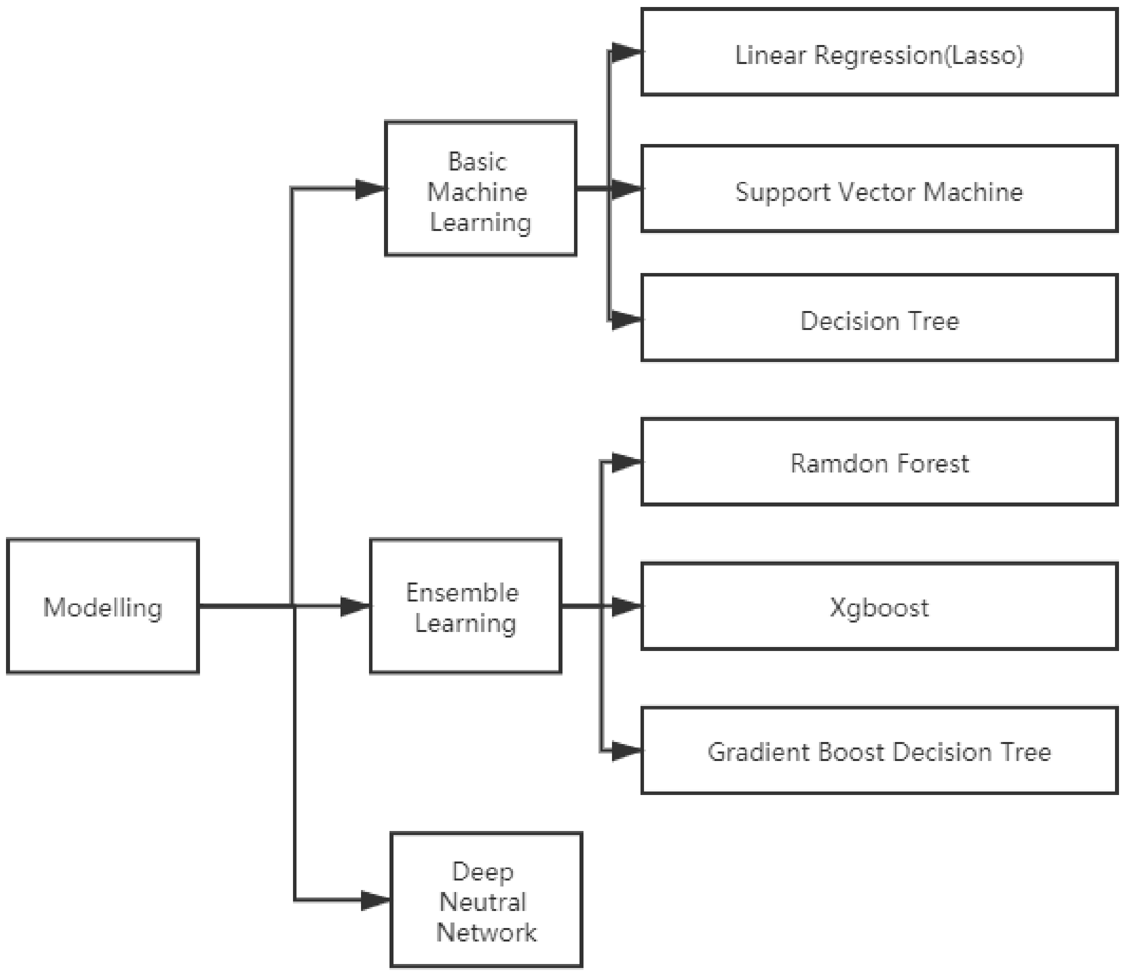


Figure 4.3 Categories of Modelling

In this section, we will use seven methods to train the data (see Figure 4.3). For each method, we used GridSearchCV to pick the parameters that best suited the method and dataset. GridSearchCV is a tool of Scikit - learn, it can search the candidate best parameters exhaustively from the grid of given parameters (DataTechNotes, 2019).

On each model, multiple parameters are tested under cross-validation, and the best-performing parameters are then extracted into the final prediction model. The training process of each method is as follows:

Linear Regression(Lasso):

Lasso means linear regression of L1 norm. L1 regularization can effectively extract more important features among numerous features and effectively reduce the overall Mean Square Error.

In linear regression, it is difficult to know what the learning rate should be, so the learning rate is put into the GridSearchCV to search, and the range is 0.001,0.0001 and 0.00001. The code USES numpy.logspace () to create such an array.

After having the best prediction model of Linear Regression, use the model to test the test set to see how well the prediction works. Then visualize the results.

Support Vector Machine:

In support vector machine regression, parameters to be considered include kernel function and penalty factor C, which are relatively unstable factors. Therefore, these two parameters are put into GridSearchCV for testing.

On the selection of kernel functions, we generally choose from 'linear', 'poly', 'RBF', 'Sigmoid'. They all have their characteristics, as follows:

Linear Kernel: Linear kernel is mainly used in the case of linear separability. Its purpose is to find the optimal linear classifier in the original space.

$$\kappa(x, x_i) = x \cdot x_i$$

Figure 4.4 Linear Kernel

Polynomial Kernel: Polynomial kernel functions can scaling up the input space and are suitable for orthogonal normalized data. Even remote data points can affect kernel values. In the formula below, the larger D is, the higher the dimension will be, and the calculation amount will also increase correspondingly.

$$\kappa(x, x_i) = ((x \cdot x_i) + 1)^d$$

Figure 4.5 Polynomial Kernel

Gaussian Kernel(RBF): RBF network can approximate arbitrary nonlinear functions and deal with the difficult analytic laws in the system. It has good generalization ability and fast convergence speed, and has been successfully applied to nonlinear functions. In the gaussian radial basis detection data has a good anti-interference ability to noise.

$$k(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$$

Figure 4.6 RBF

Sigmoid Kernel: When Sigmoid function is used as kernel function, support vector machine implements a multi-layer perceptron neural network.

For the understanding of the penalty factor C with the addition of the relaxation factor ξ (i), the objective function and constraints of support vector machine optimization were:

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi(i)$$
$$s.t. \, y^{(i)}(w^T\Phi(x^{(i)}) + b) \geq 1 - \xi(i), i = 1, 2...n$$

Figure 4.7 Support Vector Machine Formula

C:The value of penalty factor C balances the empirical risk and structural risk. The size of C is related to the degree of fitting.

With this knowledge in mind, put the two parameters into a GridSearchCV to find what the best model looks like. Then test on the test set, and visualize the results.

Decision Tree:

Decision tree algorithm is a very simple and easy to use, but sometimes we do not know how to set the parameters, especially everyone actually has differences in the estimation of sample size and feature size, there is a certain subjectivity. Thus, we need to use GridSearchCV to search for the best parameters. For the tuning parameters of the decision tree, the more volatile are 'max_depth', 'min_samples_leaf', and 'min_impur_decrease', so adjust for these three parameters.

Max_depth: In general, you can ignore this value when you have less data or fewer features.

When needed, limiting the maximum depth in the case of large sample size and multiple features of the model, and the specific valu e depends on the distribution of data.

This is because when there are many features, if the maximum depth is not limited, the decision tree will thoroughly classify the data, which will overfit the training set and lead to poor generalization ability of the model.

Therefore, limiting the maximum depth can prevent overfitting. In this project, we're going to pick the number from [3, 5, 10, 15] to make the most appropriate depth for this data set.

Min_samples_leaf: If the sample size of a node is less than ‘min_samples_split’, no further attempt is made to select the optimal feature for the partition.

If the sample size is of a very large order, you can increase this value. In this project, we choose a parameter between [2,3,5,10].

Min_impurity_decrease: This value limits the growth of the decision tree. If the impurity of a node is less than this threshold, the node will no longer generate child nodes. In this project, we selected one from [0.1,0.2,0.5] as the final parameter.

Finally, combine these three parameters into GridSearchCV to search for the best parameters, train the model, predict the test set, and visualize the results.

Random Forest:

Since the Random Forest is an integrated algorithm built by multiple decision trees by voting, the parameter regulation of each learning classifier in the Random Forest is the same as that of the decision tree. What we need to focus on is the parameters of the integrated learning framework, which can be adjusted on the Random Forest are ‘n_estimators’ and ‘OOb_score’.

n_estimators: This parameter is more important than other parameters for the Random Forest. If there are too many subtrees, the complexity of the model will increase; if there are too few, the ability of classification prediction will decrease. Therefore, in this project, we select parameters from [10,20,30,40].

OOb_score: That is, whether to use out-of-pocket samples to evaluate the model. The accuracy of the tree can be evaluated with the out_of_BAG sample. In addition, other subtrees are calculated according to this principle. Finally, the average value is the performance of the random forest algorithm. But we'll put both true and false possibilities into GridSearchCV and let it decide for itself.

Gradient Boost Decision Tree:

Since the parameters of each decision tree in GBDT are basically the same as the parameters of the decision tree method, they will not be repeated here. We look at the important parameters related to boosting framework.

n_estimators: Generally speaking, if n_estimators are too small then it prone to underfitting, while n_estimators are too large so that prone to overfitting, so a moderate value is generally chosen. The default is 100. In actual parameter tuning, we often consider n_estimators along with the learning_rate parameter described below. Thus, we will select the parameters from [100,150,200].

Learning rate: That is, each weak learner's weight reduction v , also known as the step length, our strong learner's iterative formula is $f_k(x)=f_{k-1}(x)+vh_k(x)$. The range of v is $0< v \leqslant 1$. For the same training set fitting, the smaller step size means that the weak learner needs more training to achieve the fitting effect. Thus, these two parameters, n_estimators and learning_rate, need to be adjusted together. We will set up a few candidate parameters [0.001, 0.005, 0.01, 0.05, 0.1].

Subsample: The subsampling here is different from the random forest, which USES put back sampling, not put back sampling here.

If the value is 1, all samples are used, i.e., no subsamples are used. If the value is less than 1, only part of the samples are fitted with GBDT decision tree. The recommendation is between [0.5, 0.8], and the default is 1.0, i.e., no subsampling is used. Therefore, the candidate parameters is [0.5,0.6,0.7,0.8].

Extreme Gradient Boosting(Xgboost):

Xgboost is called the enhanced GBDT. This is because XGB's final model is the same as GBDT's, and is based on an additive tree model. But XGB's loss function is slightly different from GBDT's, with the regularization term added. Therefore, the adjustment of regularization options will also be involved in the subsequent tuning of parameters.

Here are the parameters involved in tuning:

For the Xgboost framework parameters, the most important are the three parameters: booster, n_estimators, and objective.

booster: Booster determines the type of weak learner that Xgboost uses, either the default GBTree, which is the CART decision tree, or a linear weak learner, gblinear, and DART. In general, we use GBTree, and not need to adjust parameters.

n_estimators: The adjustment idea of n_estimators is the same as that of GBDT, and the parameter range is the same as that of GBDT, i.e., [100,150,200].

objective: Objective represents the problem that we're going to solve whether it's classification or regression, or whatever, and the corresponding loss function. There are many specific values that can be taken, and generally we only care about the parameters used in classification and regression.

In the regression problem we will use “reg:squarederror”, which is MSE mean square error.

In binary problem: “binary:logistic” is used for binary, while in multiple classified problem, “multi:softmax” will be used.

Thus, in this project, “reg:squarederror” will be selected as the parameter of objective.

Next, discuss the parameters of GBTree:
max_depth: This parameter corresponds to GBDT and is therefore set to the same value.

gamma: When a node splits, the node splits only if the loss function drops after the split. The larger the gamma, the more conservative the algorithm. Thus it's much less likely to split nodes when the tree is generated. Scope: [0, ∞].

subsample: The subsample parameters are the same as the GBDT subsample and will not be put back into the sample. If the subsample value is less than 1, then it has the effect of preventing overfitting, but at the same time, it will cause the deviation of data fitting to increase.

reg_alpha /reg_lambda: These are the regularization parameters for Xgboost. Reg_alpha is the L1 regularization coefficient, reg_lambda is the L2 regularization coefficient.

Deep Neutral Network:

In this project, Keras was used to build a neural network, and Sklearn GridSearchCV was used to gradually help the model find the final parameters, which included batch size, epochs, optimization algorithm, learning rate and other parameters.

The following parameters are presented and adjusted in the order of the actual project:

batch size and epochs: In iterative gradient descent, batch size is the number of samples that were input by the network before weight update. It determines how many samples are read into memory at a time.

The epoch is the number of times the training set was trained during the training.

Therefore we set the batch size and epochs in this range: ('epochs':[50,100,150,200], 'batch_size':[5, 10, 20,30])

Optimization algorithm: For the parameter optimizer, the parameters will be selected from ['SGD' , 'RMSprop', 'Adagrad', 'Adadelta', 'Adam'].

SGD: SGD, which calculates the gradient of mini-batch each iteration and then updates the parameters, is the most common optimization method

Disadvantages:

It is difficult to choose the appropriate learning rate

SGD tends to converge to local optima

$$g_t = \nabla_{\theta_{t-1}} f(\theta_{t-1})$$

$$\Delta\theta_t = -\eta * g_t$$

Figure 4.8 SGD formula

Therefore we can think about Momentum. SGD tends to get stuck in the case of ravines, where one side of the surface is steeper than the other, and SGD oscillates and is slow to approach the minimum. Momentum can accelerate SGD by adding γ v_t−1 and inhibit oscillation. The addition of this term can make the speed faster on the dimension with constant gradient direction, and the updating speed slower on the dimension with changed gradient direction, so as to accelerate the convergence and reduce the oscillation.

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta).$$

$$\theta = \theta - v_t.$$

Figure 4.9 Momentum

Adagrad: Adagrad solves the problem of differences in the rate of neuronal parameter renewal by automatically assigning different learning rates to neuronal parameters.

$$n_t = n_{t-1} + g_t^2$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{n_t + \epsilon}} * g_t$$

Figure 4.10 Adagrad formula

Adadelta: Adadelta is an extension of Adagrad, and the original scheme was still an adaptive constraint on learning rates, but with computational simplification. Whereas Adagrad accumulates all the previous gradients squared, Adadelta only accumulates fixed-size terms and does not store them directly, only approximating the average

$$n_t = \nu * n_{t-1} + (1-\nu) * g_t^2$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{n_t + \epsilon}} * g_t$$

Figure 4.11 Adadelta formula

RMSprop: Both RMSprop and Adadelta address the problem of a steep decline in The Adagrad learning rate by using an exponential weighted average designed to eliminate the wobbles in gradient descent, as is the effect of Momentum.

Adam: The algorithm is equivalent to the combination of RMSprop and momentum. Adam can store the average exponential decay of past gradients squared.

Learning Rate and Momentum: Usually we choose an optimization algorithm in advance to train our network and adjust its parameters. Usually SGD is selected for optimization. Set multiple candidate parameters for learning rate and momentum to find the right combination using GridSearchCV. Therefore, the learning rate is selected from here [0.001, 0.01, 0.1, 0.2, 0.3], momentum is selected from [0.0, 0.2, 0.4, 0.6, 0.8, 0.9].

Activation function: Activation functions control the nonlinearity of neurons during activation. In general, 'relu' activation functions are the most frequently used, but sigmoid and tanh functions may be more appropriate for different problems. Thus, we choose an appropriate activation function among ['softmax', 'relu', 'tanh', 'sigmoid', 'linear'].

Dropout: When propagating forward, there will be a probability P that stops some neurons from working, therefore the neurons stopped in each calculation will be different, which will make the neural network more generalized and not depend on one characteristic alone.

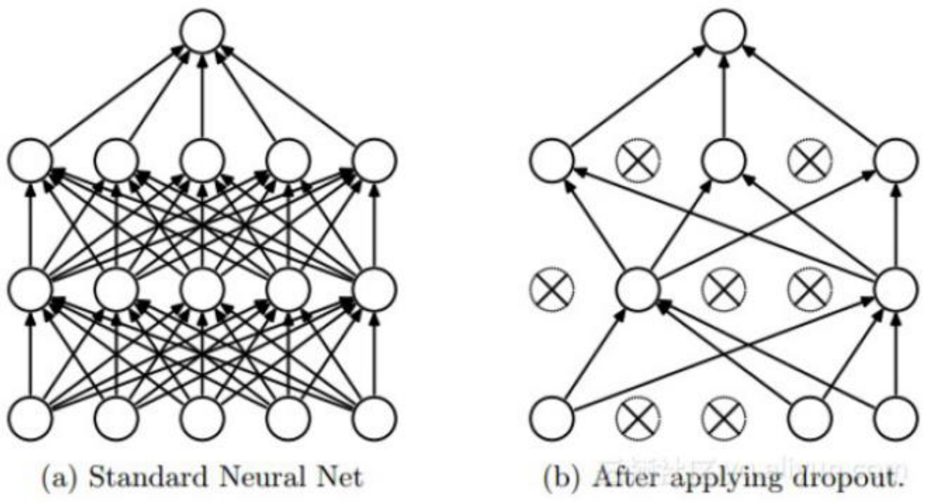


Figure 4.12 Dropout Effect

Number of neurons in the hidden layer: If there are too few nodes in the hidden layer, the training effect of neural network on data will be worse. However, if there are too many nodes, the speed of training will be slowed down and it is easy to fall into local fitting. There is only one hidden layer in this project, so the hidden layer will be obtained from [1, 4, 8, 16, 32, 64, 128].

A table is used to display the final optimal parameters of each model:

Table 4.1 Candidate Parameters

Model	Parameters	Value
Linear Regression(Lasso)	Learning rate	[0.01,0.001,0.0001]
Support Vector Machine	Kernel function	['linear', 'poly', 'RBF', 'Sigmoid']
	C	[1, 10]
Decision Tree	Max_depth	[3, 5, 10, 15]
	Min_samples_leaf	[2, 3, 5, 10]
	Min_impurity_decrease	[0.1, 0.2, 0.5]
Random Forest	n_estimators	[10,20,30,40]

	OOb_score	[True,False]
GBDT	Learning rate	[0.001, 0.005, 0.01, 0.05, 0.1]
	subsample	[0.5,0.6,0.7,0.8]
	n_estimators	[100,150,200]
Xgboost	n_estimators	[100,150,200]
	Objective	reg:squarederror
	subsample	[0.5,0.6,0.7,0.8]
	Learning rate	[0.001, 0.005, 0.01, 0.05, 0.1]
Deep Neutral network	Epochs	[50, 100, 150,200]
	Batch size	[5, 10, 20, 30]
	Optimization algorithm	['SGD' , 'RMSprop', 'Adagrad', 'Adadelta', 'Adam']
	Learning Rate	[0.001, 0.01, 0.1, 0.2, 0.3]
	Momentum	[0.0, 0.2, 0.4, 0.6, 0.8, 0.9]
	Activation function	['softmax', 'relu', 'tanh', 'sigmoid', 'linear']
	Dropout	[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
	Number of neurons in the hidden layer	[1, 4, 8, 16, 32, 64, 128]

4.5 Evaluation

This project will use four classical regression model evaluation matrices to evaluate the above models, which is mean square error, mean absolute error, root mean square error and R-squared. I will visually display the performance of each model by means of a radar map.

Chapter 5
Results

5.1 Results of GridSearchCV

These are the final results of each model after tuning parameters from GridSearchCV. This is not necessarily the best combination of parameters to train this data, but only the best performing parameters among the candidate parameters I selected.

Table 5.1 parameters setting of each model

Model	Parameters	Value
Linear Regression(Lasso)	Learning rate	0.001
Support Vector Machine	Kernel function	RBF
	C	10
Decision Tree	Max_depth	5
	Min_samples_leaf	10
	Min_impurity_decrease	0.1
Random Forest	n_estimators	40d
	OOb_score	False
GBDT	Learning rate	0.001
	subsample	0.8
	n_estimators	200
Xgboost	n_estimators	200
	Objective	reg:squarederror
	subsample	0.8
	Learning rate	0.001
Deep Neutral network	Epochs	100
	Batch size	100
	Optimization algorithm	SGD
	Learning Rate	0.01

	Momentum	0.9
	Activation function	tanh
	Dropout	0.0
	Number of neurons in the hidden layer	32

5.2 Evaluation of Each Model

Each model uses the best training parameters for the current data to train the data. Then we use the evaluation matrices to evaluate the model's performance on both the training set and the test set, and use the radar chart to show the performance, which can be used for comparison.

5.2.1 Linear Regression

As one of the simplest models, the Linear Regression model performs poorly on data. Its RMSE score is 0.7(see Figure 5.1), that is, the difference between it and the real value is 0.7. R2 had a score of 0.53, only slightly higher than the random model.

The Linear Regression model performs almost the same in both the training set and the test set, which indicates that the Linear Regression model has a strong generalization ability, but its prediction ability is also weak.

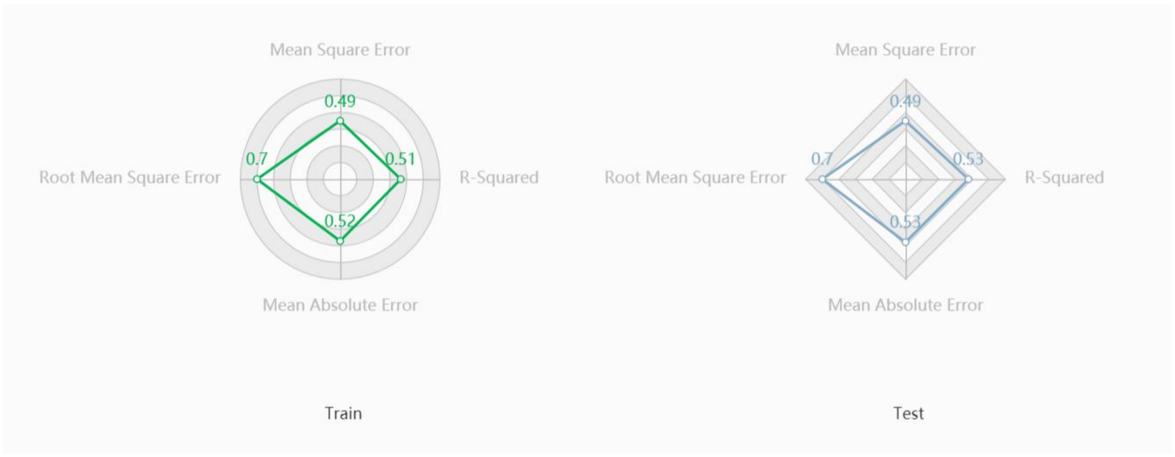


Figure 5.1 Linear Regression Radar Map

The scatter plot of the performance of the model on the test set is presented(see Figure 5.2), and it is found that each predicted value is roughly linear. However, the predicted points are not centralized, but relatively scattered, which indicates that the linear regression model does not perform a good regression on the data.

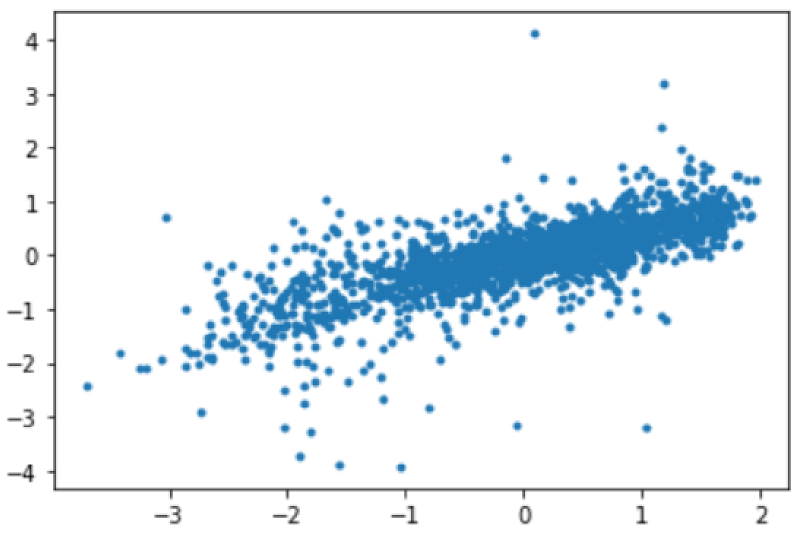


Figure 5.2 Scatter Plot of LR

After drawing a histogram for the regression model(see Figure 5.3), it was found that the residual basically met the normal distribution. Such regression model would be more sensitive to outliers, so there would be no extreme error in each prediction result.

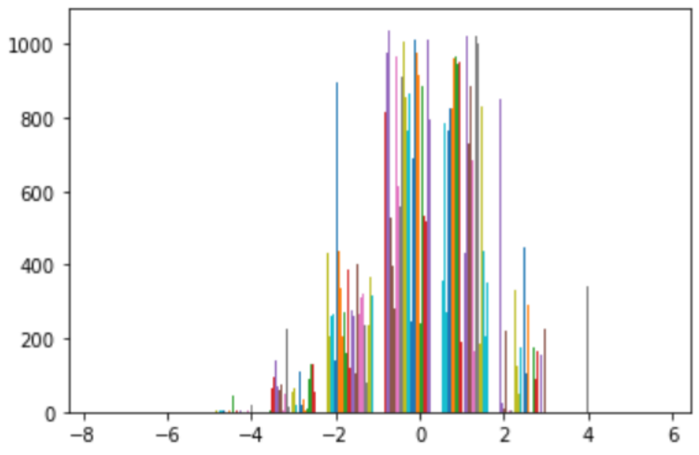


Figure 5.3 Histogram of LR

5.2.2 Support Vector Regression

As one of the classical machine learning algorithms, support vector machine has excellent performance. The MSE value is as low as 0.18 on the training set and as low as 0.24 on the test set. Moreover, the score of R2 on the test set reaches 0.78, which is a very effective regression model.

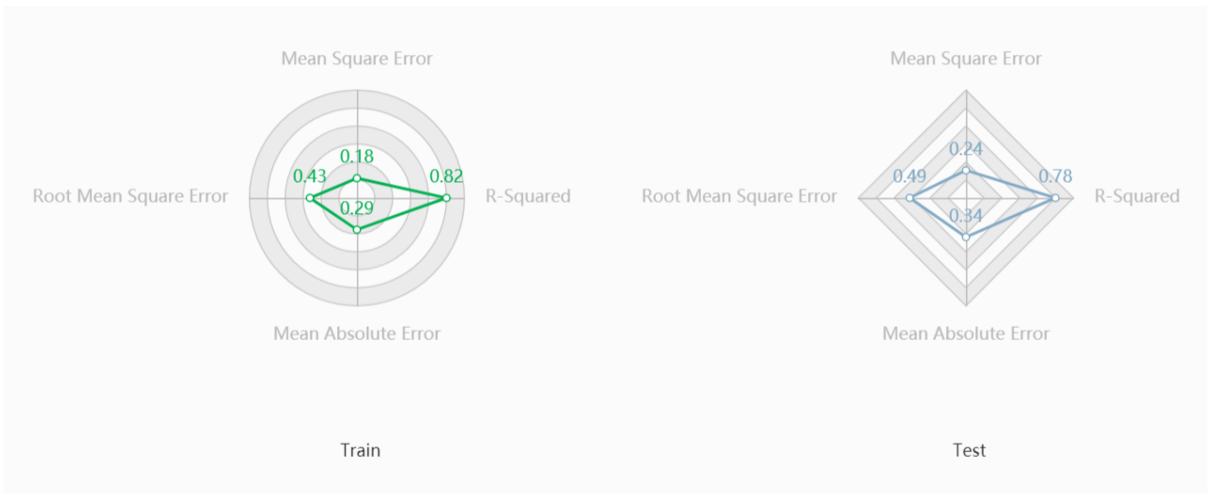


Figure 5.4 Radar Map of Support Vector Regression

As can be seen from the scatter plot, the Regression effect of SVR is much better than that of Linear Regression. Although there are outliers, the number of outliers is relatively small in general, and most of them are concentrated, indicating that the predicted value is close to the real value.

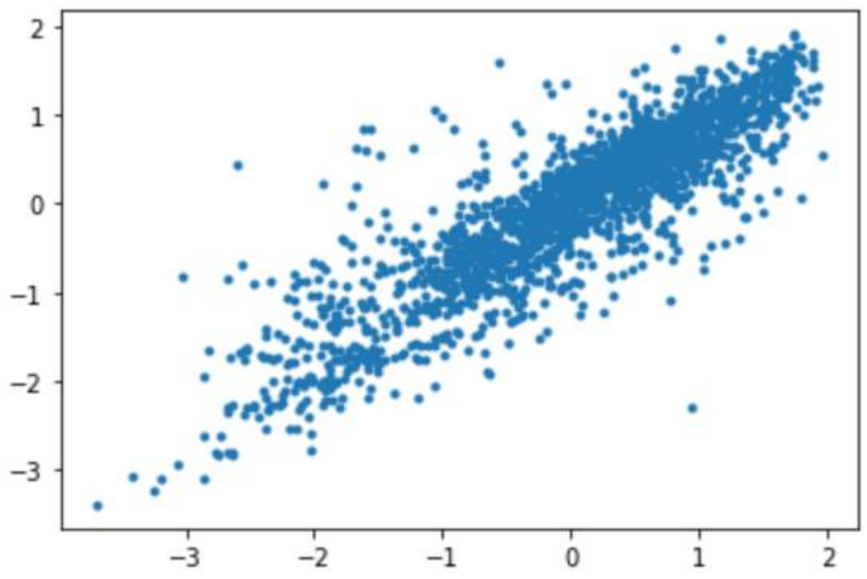


Figure 5.5 Scatter Plot of SVR

5.2.3 Decision Tree

As can be seen from the radar chart(see Figure 5.6), the training effect of decision tree is not very good. Its R2 value is around 0.43 in both the training set and the test set. While R2 needs to be at least 0.3 to be meaningful, the decision tree's R2 value is low compared to LR and SVR.

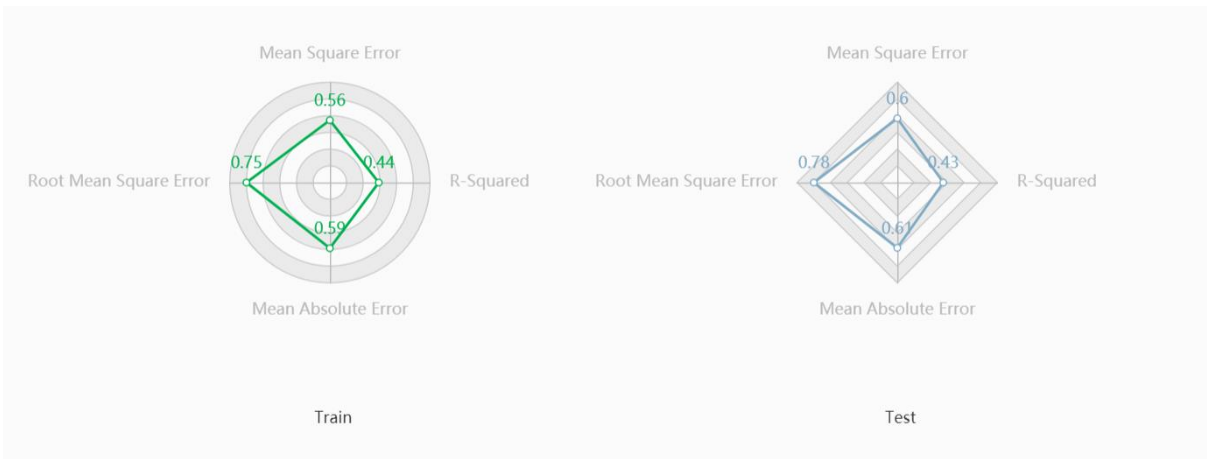


Figure 5.6 Decision Tree Radar Map

As can be seen from the figure below, the decision tree is only based on the judgment of 'MolLogP', while the other 16 features are ignored. The decision tree segmented by only one feature must be very simple, so its predictive ability will be very weak.

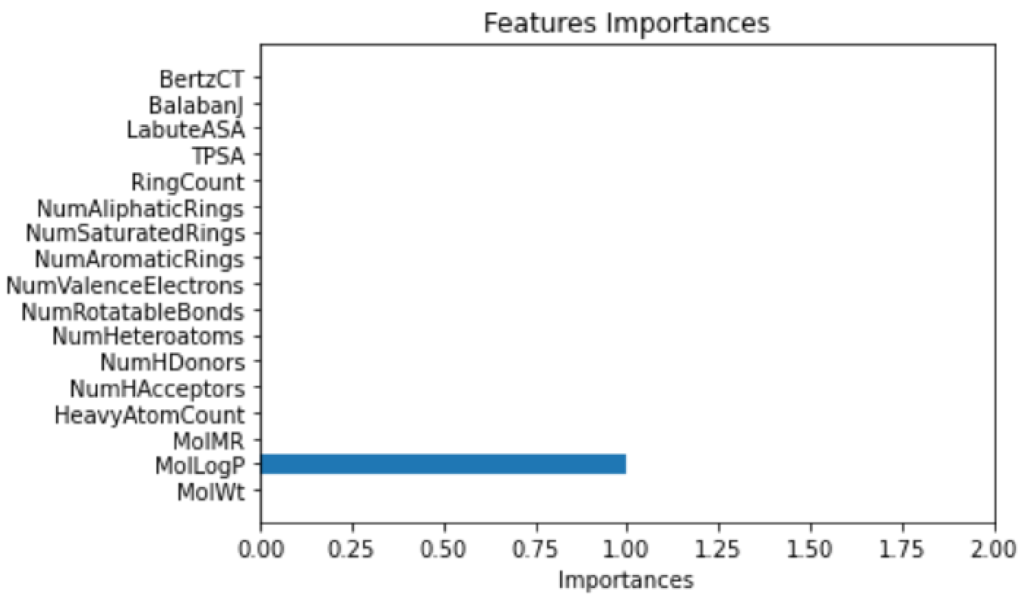


Figure 5.6 Features Importance of Decision Tree

5.2.4 Random Forest

As can be seen from the radar chart, the random forest is only a little better than the decision tree, with low data in all aspects, and its predictive ability is questionable.

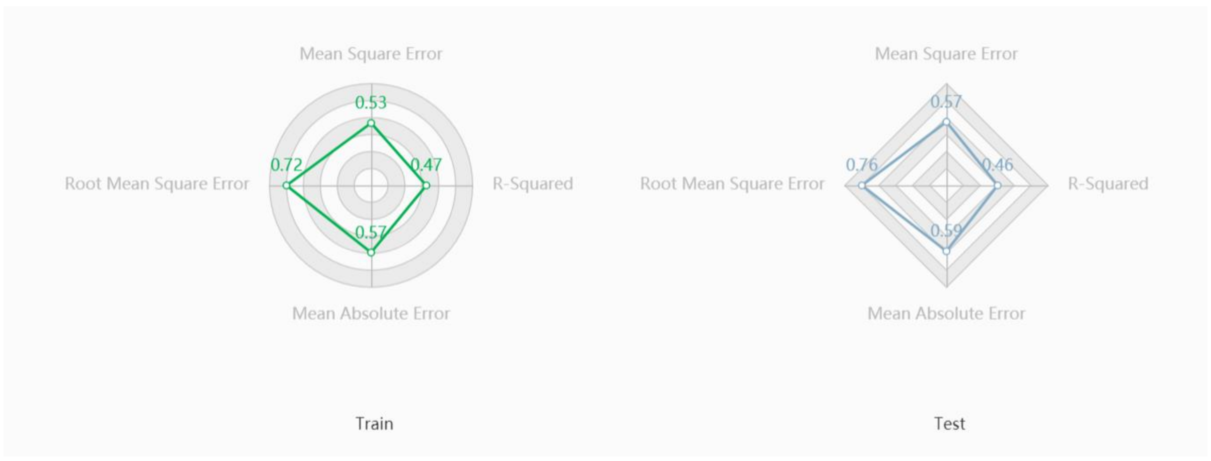


Figure 5.7 Random Forest Radar Map

The only difference between the random forest and the decision tree used in this project is that the random forest focuses more on the feature 'MolLogP'. Therefore, although the random forest can build many sub-decision trees for voting, because the decision tree built by single feature is very simple, the voting result will not be good.

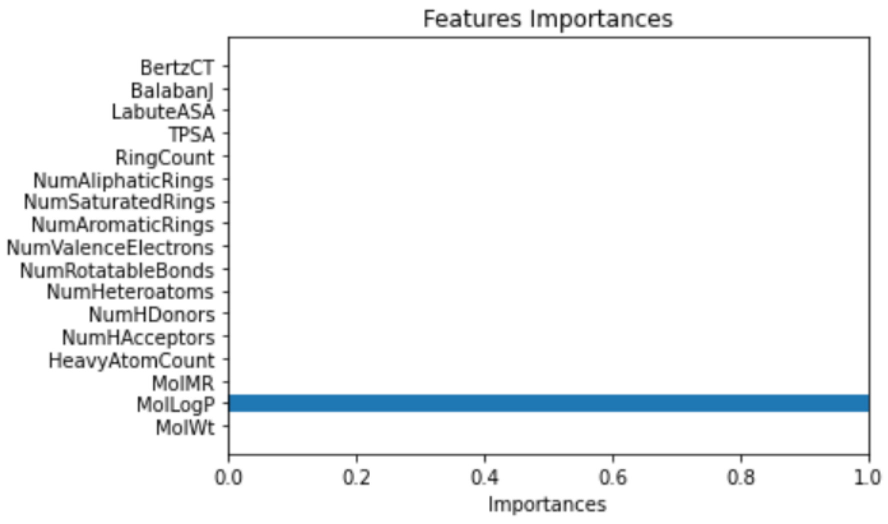


Figure 5.8 Features Importance of Random Forest

5.2.5 GBDT

As previously mentioned, the R2 value of a model needs to be more than 0.3 to be meaningful, while the performance of GBDT algorithm used in this project on the data set is less than 0.3 in both the training set and the test set (0.23 and 0.22 respectively)(see Figure 5.9).

Therefore, it can be judged that this data set may not be suitable for GBDT to be used to train a regression model.

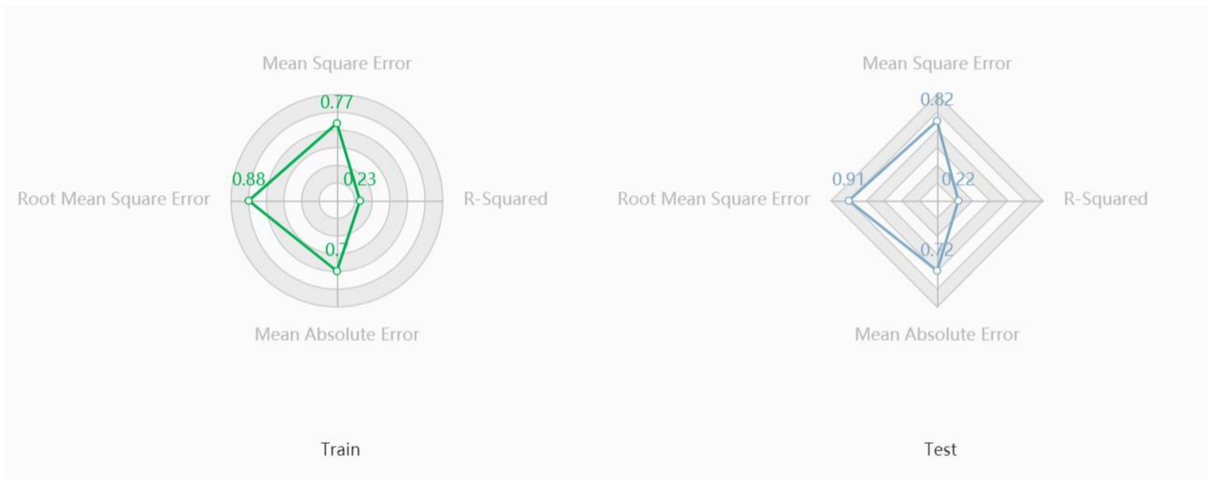


Figure 5.9 GBDT Radar Map

Although GBDT is also a Tree model, it pays more attention to features than Random Forest and Decision Tree. It can be seen in the figure that 'BertzCT', 'TPSA', 'MolMR', 'MolLogP' and 'MolWt' are all used to build subtrees, but most of the features are still not calculated.

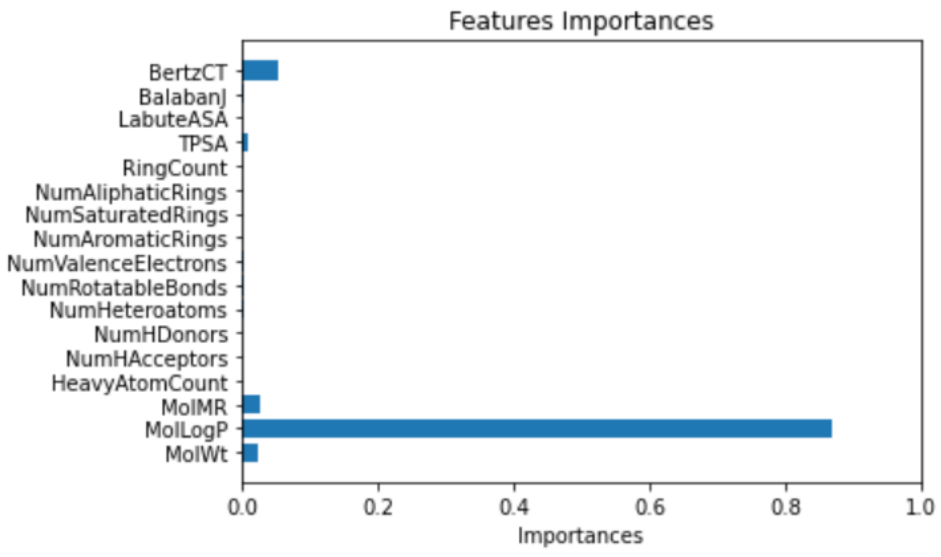


Figure 5.10 Feature Importance of GBDT

5.2.6 Xgboost

Xgboost performs best, and as you can see from the figure below(see Figure 5.11), Xgboost has a RMSE of only 0.5 on the test set and an R2 value of 0.76, which is already a very effective regression model.

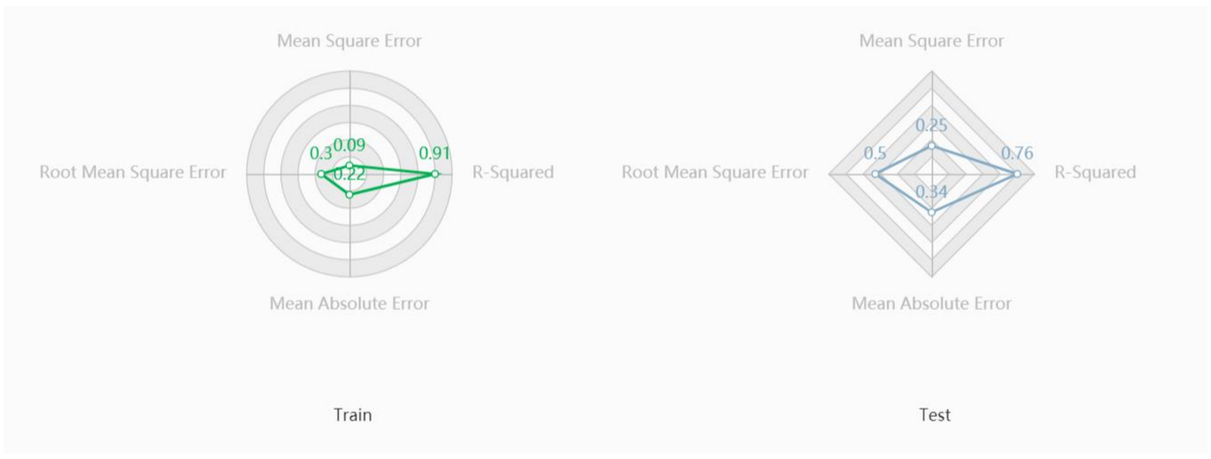


Figure 5.11 Xgboost Radar Map

The excellent performance of Xgboost is closely related to the construction of its sub-trees. As you can see from the figure below, all the features are taken into account. Although 'MolLogP' is still the most important feature, the rest of the features are relatively average. Overall, this is a very healthy regression model.

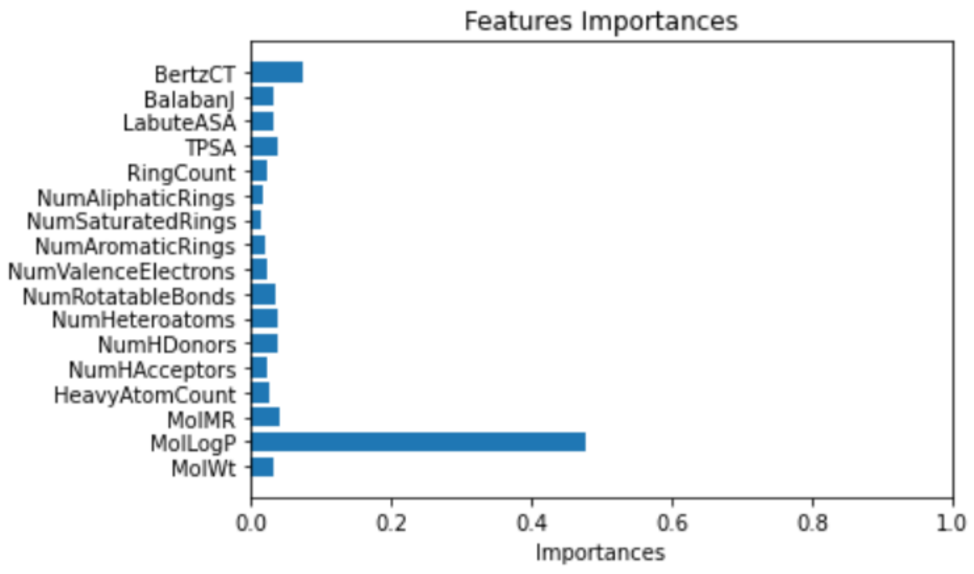


Figure 5.12 Features Importance of Xgboost

5.2.7 Deep Neutral Network

The DNN regression model performs better. As shown in the figure below, R2 scores 0.68 and RMSE 0.58 on the test set. All four matrices are close to what Xgboost does.

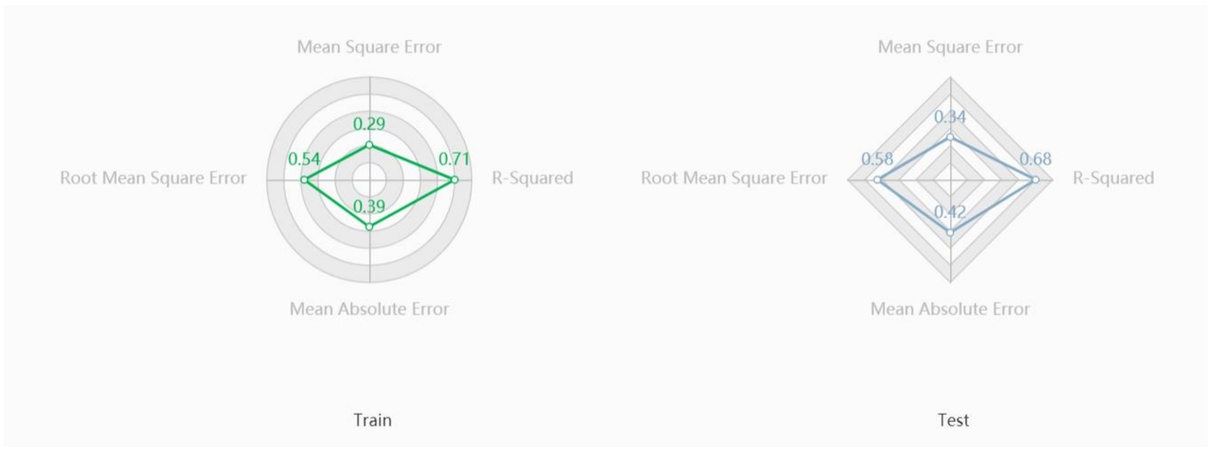


Figure 5.13 Deep Neutral Network Radar Map

5.3 Example Presentation

Ten samples randomly selected the compounds are listed in the table below after each machine learning method is used to predict the aqueous solubility of the results and actual results in two decimal places.

Table 5.2 Example Test Figure

	A	B	C	D	E	F	G	H	I	J
LR	-0.02	-0.13	0.82	0.09	-0.45	0.06	-0.56	0.46	-0.66	0.10
SVR	-0.27	0.28	1.67	-0.03	-0.68	0.51	-0.52	0.46	-1.04	0.24
DT	0.48	0.48	0.48	0.48	-0.92	0.48	-0.92	0.48	-0.92	0.48
RF	0.46	0.46	0.46	0.46	-0.96	0.46	-0.96	0.46	-0.95	0.46
GBDT	-0.05	0.02	0.21	0.02	-0.12	0.08	-0.25	0.08	-0.09	0.06
XGB	-0.40	0.39	1.71	-0.12	-0.23	0.54	-0.44	0.53	-0.95	0.39
NN	-0.34	0.27	1.00	-0.13	-0.48	0.61	-0.92	0.59	-0.95	0.15
TRUE	-0.19	0.57	1.50	-0.36	0.56	0.84	-0.82	0.26	-0.76	0.39

From the line graph below, the thickest line represents the True Value, which is a baseline. The more similar the polyline represented by a method to the True Value line, the better the method performs on the example data.

As you can see from the figure, except that none of the methods can predict the solubility of Data E, the line of Xgboost almost matches the line of True Value.

The second best approach is DNN, which, while more error than Xgboost, almost exactly matches the True Value line, except that Data C and Data E perform poorly.

The worst performers were the three other methods besides Xgboost, Decision Tree, Random Forest, and GBDT, whose predictions for all 10 compounds were very poor and could be judged to have little predictive power.

The remaining SVR and Linear Regression were flat. Although the general trend of the line was the same as that of the True Value line, the error was too large and the predictive ability was relatively weak.

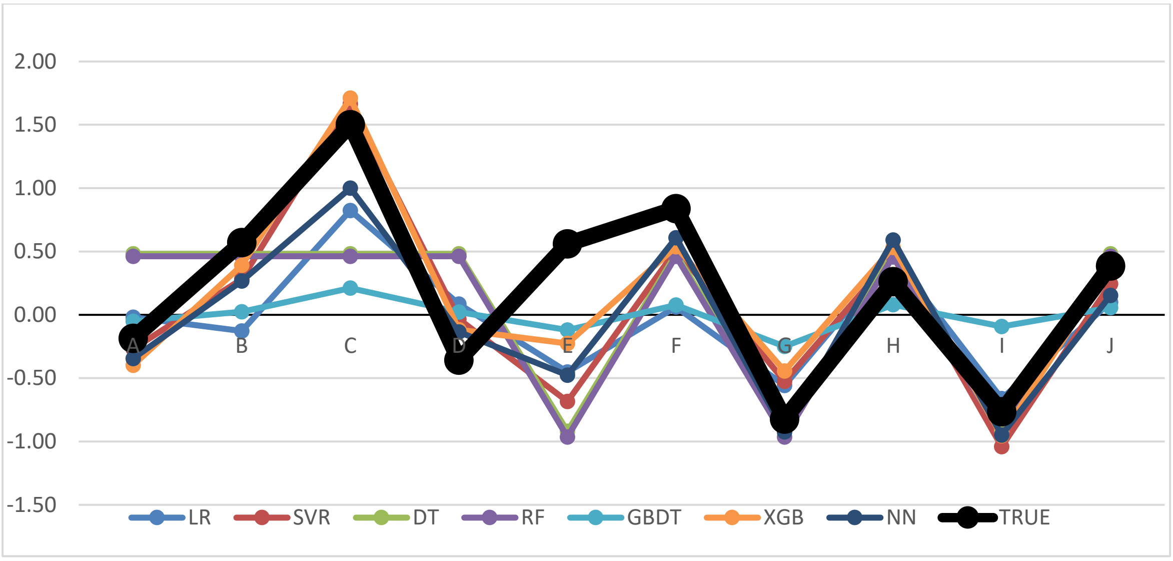


Figure 5.14 Line Chart Comparison

In order to more clearly compare the predictions of various methods for the example compound, the bar chart below can clearly see the difference between the predicted values of each method and the True Value.

As can be seen in the figure, the black and patterned column represents True Value, which facilitates the comparison between other predicted values and True values.

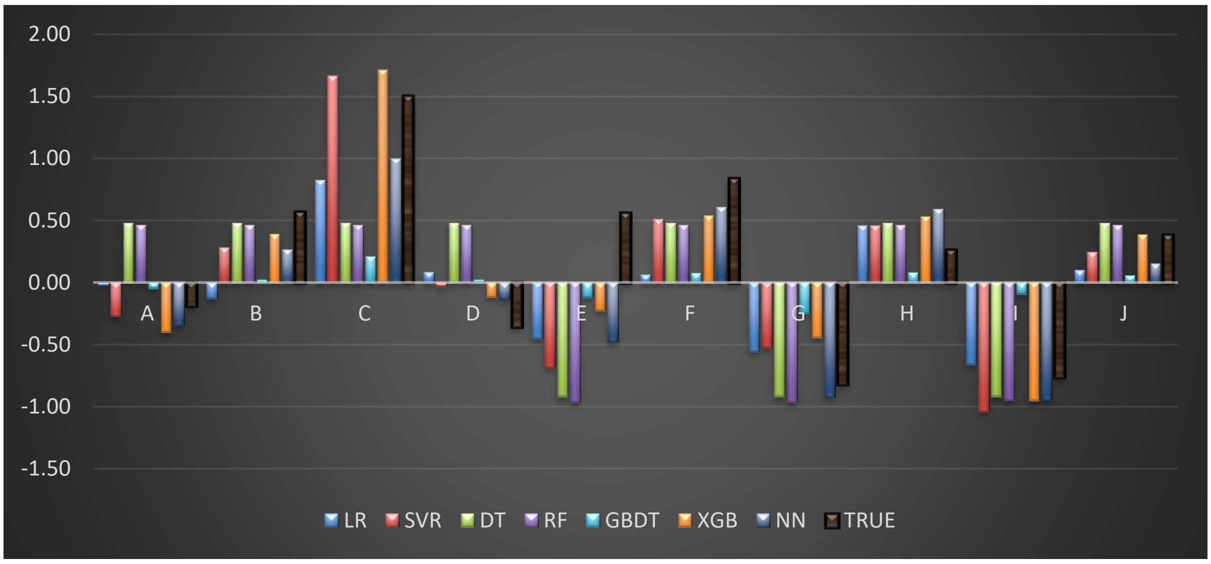


Figure 5.15 Bar Chart Comparison

Chapter 6
Conclusion

6.1 Project Conclusion

Table 6.1 R-squared Value of Each Method

Method	R-squared
Linear Regression	0.53
Support Vector Regression	0.78
Decision Tree	0.43
Random Forest	0.46
Gradient Boost Decision Tree	0.22
Xgboost	0.76
Deep Neutral Network	0.68

From the above data, the SVR has the highest R-squared value. Xgboost has a lower value than the SVR, but the RMSE of the SVR is higher than the Xgboost, so Xgboost is the best way to process this data and build prediction model, and SVR is the second best.

Tree model algorithm, decision tree, random forest, and GBDT performed worst, with poor prediction ability for new data.

The average performance is linear regression, and neural networks, with Numbers of 0.53 and 0.68, respectively.

Therefore, for the training of this compound data set, we prefer Xgboost, as well as SVR, and it is highly recommended to use neural networks for prediction if more data and compound types are available in the future.

This is because the methods used in this project are all tools from Sklearn and can be used by calling them directly. Only the neural network needs to design a complete neural network structure, so there is still a lot of space for optimization. The model trained on such a rudimentary neural network still had an R-squared score of 0.68, which suggests that if improved, its predictive power is likely to surpass that of Xgboost and SVR.

6.2 Project Limitation

First, the candidate parameter range of each method is not wide enough, so the currently selected best parameter is only selected among the candidate parameters, and does not necessarily represent the best parameter for all possible parameters.

Second, the current neural network has only three hidden layers, therefore it is not clear whether increasing or decreasing the number of hidden layers will have a more positive effect on predictive ability.

6.3 Future Work

First, for each method, as many parameters as possible should be tried to see if the prediction ability of each methods can get better results. At the same time, try to adjust the number of hidden layer of deep neutral network.

Second, there are many other approaches that can be tried to build a regression model. The current approach may not be the best one.

Third, the regression problem of tree model should be studied to solve the problem that the tree model cannot obtain a good prediction ability in this experiment.

List of References

Amos, D. 2020. *Object-Oriented Programming (OOP) in Python 3.* [Online]. [Accessed 14 August]. Available from: https://realpython.com/python3-object-oriented-programming/

Angermueller, C., Pärnamaa, T., Parts, L. and Stegle, O.J.M.s.b. 2016. Deep learning for computational biology. **12**(7), p878.

Baskin, I.I., Winkler, D. and Tetko, I.V.J.E.o.o.d.d. 2016. A renaissance of neural networks in drug discovery. **11**(8), pp.785-795.

Chakure, A. 2019. *Random Forest Regression.* [Online]. [Accessed 12 August]. Available from: https://towardsdatascience.com/random-forest-and-its-implementation-71824ced454f

DataTechNotes. 2019. *How to Use GridSearchCV in Python.* [Online]. [Accessed 14 August]. Available from: https://www.datatechnotes.com/2019/09/how-to-use-gridsearchcv-in-python.html

Demiriz, A., Bennett, K.P., Breneman, C.M. and Embrechts, M.J. 2001. Support vector machine regression in chemometrics. In: *In Computing Science and Statistics: Proceedings of the 33rd Symposium on the Interface*: Citeseer.

Di, L., Fish, P.V. and Mano, T.J.D.d.t. 2012. Bridging solubility between drug discovery and development. **17**(9-10), pp.486-495.

Dietterich, T.G.J.T.h.o.b.t. and networks, n. 2002. Ensemble learning. **2**, pp.110-125.
Friedman, J.H.J.A.o.s. 2001. Greedy function approximation: a gradient boosting machine. pp.1189-1232.

Huuskonen, J., Salo, M., Taskinen, J.J.J.o.c.i. and sciences, c. 1998. Aqueous solubility prediction of drugs based on molecular topology and neural network modeling. **38**(3), pp.450-456.

Khurana, S., Rawi, R., Kunji, K., Chuang, G.-Y., Bensmail, H. and Mall, R.J.B. 2018. DeepSol: a deep learning framework for sequence-based protein solubility prediction. **34**(15), pp.2605-2613.

Lind, P., Maltseva, T.J.J.o.c.i. and sciences, c. 2003. Support vector machines for the estimation of aqueous solubility. **43**(6), pp.1855-1859.

Lovric, M., Pavlović, K., Žuvela, P., Spataru, A., Lučić, B., Kern, R. and Wong, R.M.W. 2020. Machine learning in prediction of intrinsic aqueous solubility of drug-like compounds: generalization, complexity or predictive ability?

Mitchell, J.B.J.W.I.R.C.M.S. 2014. Machine learning methods in chemoinformatics. **4**(5), pp.468-481.

Mujtaba, H. 2020. *Understanding the Ensemble method Bagging and Boosting.* [Online]. [Accessed 12 August]. Available from: https://www.mygreatlearning.com/blog/bagging-boosting

Palmer, D.S., O'Boyle, N.M., Glen, R.C., Mitchell, J.B.J.J.o.c.i. and modeling. 2007. Random forest models to predict aqueous solubility. **47**(1), pp.150-158.

Ruelle, P. and Kesselring, U.W.J.C. 1997. Aqueous solubility prediction of environmentally important chemicals from the mobile order thermodynamics. **34**(2), pp.275-298.

Sorkun, M.C., Khetan, A. and Er, S.J.S.d. 2019. AqSolDB, a curated reference set of aqueous solubility and 2D descriptors for a diverse set of compounds. **6**(1), pp.1-8.

Tso, G.K. and Yau, K.K.J.E. 2007. Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks. **32**(9), pp.1761-1768.

Wirth, R. and Hipp, J. 2000. CRISP-DM: Towards a standard process model for data mining. In: *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*: Springer-Verlag London, UK, pp.29-39.

Xia, X., Maliski, E., Cheetham, J. and Poppe, L.J.P.r. 2003. Solubility prediction by recursive partitioning. **20**(10), pp.1634-1640.

Xu, W. 2019. *What's the difference between Linear Regression, Lasso, Ridge, and ElasticNet in sklearn?* [Online]. [Accessed 12 August]. Available from: https://towardsdatascience.com/whats-the-difference-between-linear-regression-lasso-ridge-and-elasticnet-8f997c60cf29

Yalkowsky, S. and Dannelfelser, R.J.T.i.n.c.r.f.t.r. 1990. The ARIZONA Database of Aqueous Solubility; College of Pharmacy, University of Arizona: Tucson, AZ, 1990.

Yıldırım, S. 2020. *A Beginner's Guide to Supervised Machine Learning Algorithms.* [Online]. [Accessed 13 August]. Available from: https://towardsdatascience.com/a-beginners-guide-to-supervised-machine-learning-algorithms-6e7cd9f177d5

Appendix A
Code

All of the code for this project is stored in the Notebook module on the Kaggle platform. The URL: https://www.kaggle.com/leoopenflying/comparison-of-deep-learning-with-machine-learning.

Appendix B
Ethical Issues Addressed

This exploratory software project is void of any ethical issue.