

Fashion MNIST Dataset training and analysis based on KNN and Decision Tree classifier

Name: Bowen Chen Student ID: 2200373

Course: Artificial Intelligence

Abstract: In this paper, Fashion MNIST dataset are used to measure the performance of KNN and decision tree classifier. Different k value of KNN algorithm are tried to test the influence of k. Randomized Search CV method are used to get the better performance of the decision tree classifier. With all this work being done, the different parameters of each classifier algorithm is tested , and the different performance between these two classification algorithm is being analyzed.

Key words: KNN; Decision tree; classification; Fashion MNIST dataset;

Introduction

MNIST handwriting data set is a former widely used dataset to test the ML algorithm's performance. As the time goes by, there is another dataset which can substitute this one, called Fashion-Mnist dataset. It contains 10 categories, namely: T-Shirt (T-shirt), Trouser (jeans), Pullover, Dress (skirt), coat (coat) , Sandal (sandals), shirt (shirt), Sneaker. Fashion MNIST dataset is an image classification entry -level data set provided by Kaggle. It contains 70,000 gray images of 10 categories. This dataset consists of of 70,000 fashion products from 10 categories which is shown as 28x28 grayscale images. There is 7000 images in each category. The training set has 60,000 images and the test set has 10,000 images. It is used to test different machine learning algorithms [1]. For MNIST dataset, all the vector are color, which are of the same scale, meaning that we do not need a normalization process before train the dataset. Decision tree classifier and KNN classifier are the most common and widely used machine learning algorithm's efficacy and performance. In this paper I use this dataset to analyze the performance and difference between decision tree classifier and KNN classifier.

1.KNN (K-Nearest Neighbor) Classifier

KNN (K-Nearest Neighbor) classification algorithm is one of the widely-used and simplest algorithms in data analysis field. The principle of KNN classification algorithm: In order to judge the category of the object we need to judge, all the known samples' category are used as a reference. All the elements in each sample need to be quantified and can be calculated through math functions. Calculate the distance between the unknown sample and the known samples nearby. Draw a circle to include "k" samples. At last, the unknown sample is belong to the category whose amount is the most in the circle. In a word, the KNN classifier don't need a training process, but we need to know Some of the sample's categories.

The above is the basic principle of the KNN algorithm in the classification task. In fact, the meaning of the letter K is the number of neighboring sample instances to be selected. The default value is 5 in scikit-learn's knn classifier. We can modify the parameter "n_neighbors" to test the impact of different k value.

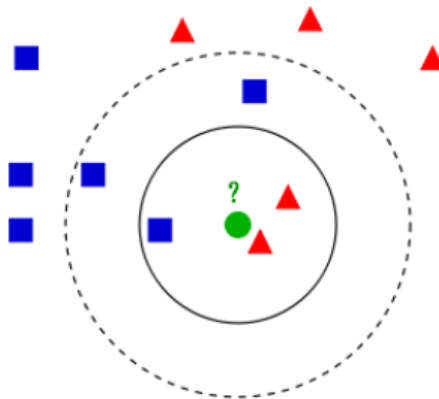


Figure1. KNN classifier's image

1.1 The key to the KNN algorithm:

- (1) All the elements of the sample can be quantified.

If there is a non -numeric element in the sample, we must take measures to quantify it into a numerical value. For example, the sample feature contains the color, which can be quantified by converting the color to the grayscale. In Fashion MNIST dataset, we divide each sample into 28*28 pixel. Each pixel can be measured from 0-9. In this case, we use number to represent color, which is the first step we need to do before using classifier.

- (2) The characteristics of samples should have normalization process.

There are multiple elements in each sample describing different characteristics, and each element has its own value defined rules. Their impact on the distance calculation may have a huge difference. Therefore, the samples' different elements must be measured by a certain scale. The easiest way is to take a normalized process of all features.

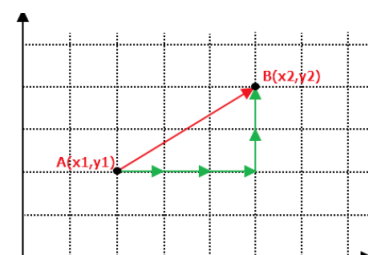
- (3) The distance or differences can be measured by math functions :

Euclidean distance, Cosine distance, Hamming distance, Manhattan distance, etc^[4]. When comes to the continuous variables, we usually pick the Euclidean distance function to calculate. When it comes to the discrete variables such as the classification problem, the hamming distance could be a better choice for calculating the distance between two objects in state space. In normal circumstances, if some special algorithms are used to calculate the measurement, the prediction accuracy of the KNN algorithm can be significantly improved.

Figure2. Hamming Distance

$$EuclideanDistance(d) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$ManhattanDistance(d) = |x_2 - x_1| + |y_2 - y_1|$$



(4) Determine the value of K

If the value of k is too large, it will cause underfitting. Too small, it is easy to overfit. The choice of k will definitely influence the classifier's efficacy and accuracy. Tests need to be done before deciding the value of k.

1.2 Step of KNN algorithm

- (1) Calculate the distance between the point of the known sample and the current point;
- (2) Sorted from small to large according to the results
- (3) circle and include k samples according to the sort.
- (4) The current point belongs to the category which has the most occurrence in the range.
- (5) The category which has most frequency in the range is returned as the prediction category of the current point.

1.3 Advantages of KNN algorithm:

- (1) Simple, easy to understand, easy to implement, no estimation parameters, no training, the data's classes must be pre-labelled;
- (2) Suitable to use for classify objects;
- (3) Suitable for multi-classification (Multi-Modal, objects have multiple categories tags), KNN's performance is better than SVM.

1.4 Disadvantages of KNN algorithm:

- (1). Complicated calculation, especially when there are many features.
- (2). When the sample is unbalanced, the accuracy of the rare category is low
- (3). It is a lazy learning method^[8], which basically does not learn, resulting in the speed of prediction is slower than logical regression algorithm.
- (4). Compared with the decision tree model, the explanatory explanation of the KNN model is not strong.

1.5 train, test, and results

60000 samples as train set, and additional 10000 samples as test set. Use KNeighborsClassifier function in sklearn to train the dataset. All the parameters are chosen as default except k. I try different k value to test the impact of k. Finally, draw a graph to show the test prediction error rate among different k values

```
error_rate = []  
  
for i in range(1,40):  
  
    knn = KNeighborsClassifier(n_neighbors=i)  
    knn.fit(X_train, y_train)  
    pred_i = knn.predict(X_test)  
    error_rate.append(np.mean(pred_i != y_test))
```

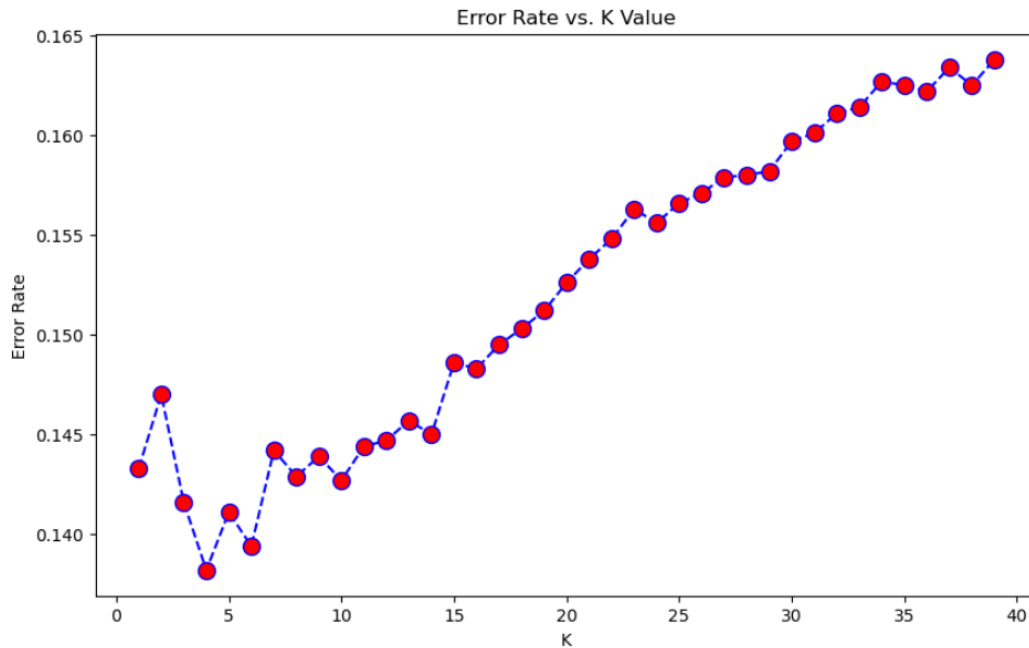


Figure 3. prediction error rate with different k value

1.6 Analysis

I use the test dataset to test the performance of the KNN algorithms . As is shown in the graph above, the best performance happens when we choose k equal to 4. And the confusion matrix is employed to show the predication performance when k=4;

	precision	recall	f1-score	support
0	0.75	0.89	0.81	1000
1	0.99	0.97	0.98	1000
2	0.73	0.83	0.78	1000
3	0.91	0.89	0.90	1000
4	0.80	0.77	0.79	1000
5	0.99	0.85	0.92	1000
6	0.71	0.55	0.62	1000
7	0.88	0.96	0.92	1000
8	0.98	0.95	0.96	1000
9	0.91	0.95	0.93	1000
accuracy			0.86	10000
macro avg	0.86	0.86	0.86	10000
weighted avg	0.86	0.86	0.86	10000

Figure 4. confusion matrix when k=4

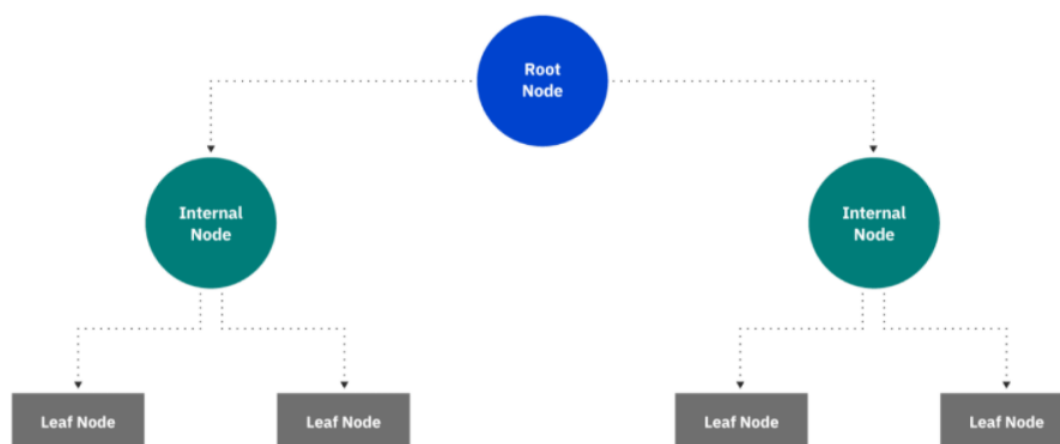
Cross-verification needs to be done before determining the choice of k. After splitting the dataset into twos parts, we get the train group and the test group. The value of k is starting from 1 to 30 or more, the curve runs in tow ways. First, the error rate goes down as the increasing of k. The error rate reaches the minimum value when k is 4. However, the error rate goes up as k is increasing in range (5,30). In conclusion, a small k is unreliable because the known samples for references are too little to judge by. However, k can't be too big, because a big k will also cause errors. For instance, if there are only 40 samples for training, and k is chosen as 30 or even 40, it is certainly unfit to use

the algorithm to classify these samples' category. At this time, training instances far away from the input instance also effectively affect the classifier, making the error rate of the prediction much higher.

2. Decision Tree Classifier

Decision tree algorithm is a supervised learning approach used in machine learning, data mining and statistics. In this formalism, a classification or regression decision tree is used as a predictive model to draw conclusions about a set of observations .Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.^[2]

The method of decision tree is to use the information gain in the information theory to find the attribute field with the maximum amount of information in the database. Establish a node of the decision tree, and then establish the branch of the tree according to the different values of the attribute field, and then concentrate on each branch. Repeat the process of the lower layer and branch of the tree. The specific process of constructing the decision -making tree is: first find the initial division, the entire training set is used as a collection of decision trees. Each record of the training set must be divided into categories, and determine which attribute domain is the best classification indicator at present. The general approach is to exhaust all the attribute domains, quantify the quality of the division of each attribute domain, and calculate for the best division. Second, repeat the first step, until the records in each leaf node belong to the same category.



2.1 The way to split the training data into subsets

(1) ID3 uses information entropy as reference to split the training data.

$$H(D) = - \sum_{i=1}^n p(x_i) \log_2 p(x_i)$$

When there is only one category of data, Then $H(D)=0$, which represents the purest data.

(2) CART

Purity refers to a node in the tree. A node is pure if all training instances it applies to belong to the same class. A Gini purity score of 0.0 implies the node is pure. It is calculated as follows:-

$$G_i = 1 - \sum_{k=1}^n p_{i,k}^2$$

The k is an iterator for the summation operator – iterating through each class, n is the number of classes and $p_{i,k}$ is the ratio of class k instances to the number of samples for the given node, i.

2.2 decision tree pruning operation:

When using the decision tree algorithm for classification problem, it is common that overfitting happens. Therefore, there are a lot of ways to prune trees. There are two main methods. The first way is to pre-pruning the tree to stop the growth earlier. In *post-pruning*, we focus on the status of the tree consistently while the tree is growing, and never forget to check the results which illustrate the overfitting status. Once is overfitting, prune the tree immediately. A lot of researches use the method of cross-validation to test the need for pruning.^[9]

(1) Pre-cutting

That is to say, the growth of the tree is in advance in the process of generating decision trees. The most important idea is that before the node in the tree is expanded, the current division can be calculated whether the current division can bring to the improvement of the model's generalization ability. At this time, there may be different categories of samples in the nodes at the same time, judging the category of the node according to the principle of most voting. When the tree is growing with insufficient data, the tree's growth is halted. Once the decision tree is generated, we cut off the subtrees which don't have enough data. This process is called as post-pruning. There are several methods for the growth of pre-cut pruning to the growth of decision trees[6]:

- ① When the tree is expanding to reach to a certain depth, stop the growth of the tree.
- ② When the number of samples in the current reaching node is less than a threshold, stop the tree from expanding.
- ③ Calculate the improvement of the prediction accuracy during the division of each node. When the quantified number is less than a threshold, the tree is no longer expanding.

Pre -branches have the characteristics of direct ideology, simple algorithms, high efficiency, and are suitable for solving large -scale problems. But how to accurately estimate when the growth of the tree (that is, the depth or threshold in the above method), there will be a big difference in different issues, and some experience judgments are required. And there is a certain limitations of pre -cut pruning, and the risk of high arrears. Although the current division will lead to a decrease in the accuracy of the test set, in the subsequent division, the accuracy may increase significantly.

(3) Post pruning

Doing cutting process on the generated decision tree which is overfitting to obtain a simple version of the pruning decision tree. The core idea is to let the algorithm runs unrestrictedly to generate a completely grown decision tree, and then calculate whether the pruning is being done from the bottom layer to the upper layer. The pruning process deletes the subtree and replaces it with a leaf node. The category of the node also is judged according to the principle of most voting . Similarly, the rear -cutting can also be judged by the accuracy of the test set. If the accuracy rate is improved after the pruning, the pruning is performed. Compared with pre -cutting, the rear pruning method can usually obtain a stronger decision tree with a stronger generalization capacity, but the time for training and predicting will be greater.

Common rear-cutting methods include reducing error pruning (REP), Pessimistic error pruning (PEP), Cost Complexity Pruning (CCP), and Minimum Error Pruning, MEP), CVP (Critical Value Pruning), OPP (Optimal Pruning) and other methods^[7]. These pruning methods have their own advantages and disadvantages, focusing on different optimization angles.

2.3 Train, test and analysis.

2.3.1. Use RandomizedSearchCV function to try to find the best parameter which results in the best prediction performance.^[3].

```

trees = DecisionTreeClassifier()
param_grid1 = {'max_depth': np.arange(5, 30),
               'min_samples_leaf': np.arange(1, 300)
               }
tree_model = RandomizedSearchCV(trees, param_grid1, n_iter=30,
                                scoring='accuracy',
                                n_jobs=-1,
                                cv=5,
                                verbose=1)

```

- (1) The search scope is the super parameter of the distility, random sampling according to the given distribution;
- (2) For the super parameter of the search range, sample in the given list with average probabilities.
- (3) The sampling results of the n_iter group obtained in A and B are traversed.

```
print("tuned hpyerparameters :(best parameters) ", tree_model.best_params_)
```

```
tuned hpyerparameters :(best parameters) {'min_samples_leaf': 37, 'max_depth': 14}
```

Graph: the parameters that RandomizedSearchCV finds

As is shown above, The best performance occurs when we choose min_samples_leaf as 37, max_depth as 14.

```
print(classification_report(y_test, test_predictions_label))
```

	precision	recall	f1-score	support
0	0.73	0.81	0.77	1000
1	0.96	0.96	0.96	1000
2	0.73	0.72	0.72	1000
3	0.84	0.83	0.84	1000
4	0.69	0.76	0.72	1000
5	0.91	0.89	0.90	1000
6	0.63	0.53	0.58	1000
7	0.87	0.90	0.88	1000
8	0.91	0.92	0.92	1000
9	0.91	0.88	0.90	1000
accuracy			0.82	10000
macro avg	0.82	0.82	0.82	10000
weighted avg	0.82	0.82	0.82	10000

2.3.2 Use the DecisionTreeClassifier from sklearn to train the data

All the parameters not mentioned are chosen as default

(1) the parameter “min_samples_leaf” is selected as 37, and try different value of “max_depth”

```
error_rate = []  
  
for i in range(2, 30):  
  
    tree = DecisionTreeClassifier(max_depth=i, min_samples_leaf=37)  
    tree.fit(X_train, y_train)  
    pred_i = tree.predict(X_test)  
    error_rate.append(np.mean(pred_i != y_test))
```

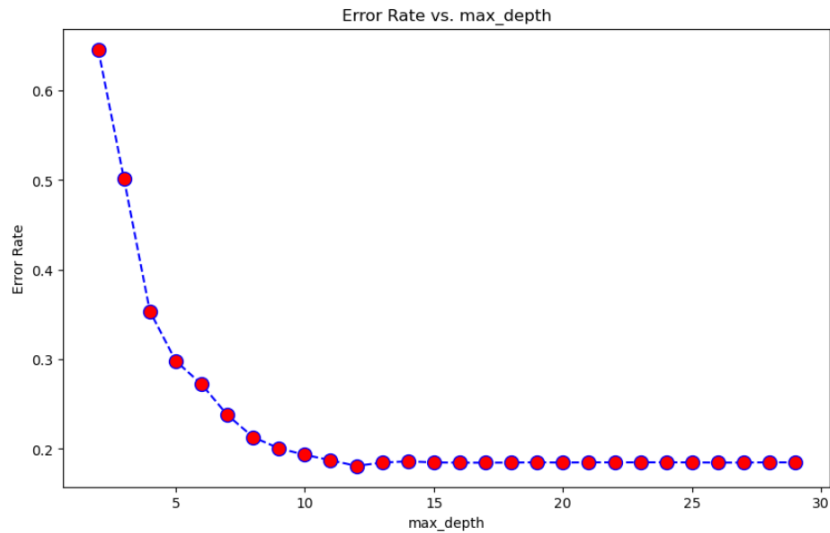


Figure 5: when min_samples_leaf=37, error rate vs max_depth

Note: Error rate = 1 - accuracy; (Using the test dataset to test)

As is shown above, when the “max_depth” is 2, the prediction error rate is almost over 60%, that is unreliable prediction. The error rate falls sharply as the max_depth increases from 2 to 5, which shows that in decision tree classifier, the value of “max_depth” parameter should not be so low. The prediction performance get better as the “max_depth” increases over 10. The best performance occurs when the value is 12. And as the value of “max_depth” goes up in range(12,30), the error rate remains almost the same.

In conclusion, using decision tree classifier training Fashion MNIST dataset, I find that we’d better choose the “max_depth” parameters in range (10,15), maybe 12, maybe 14, the prediction performance may be influenced by other parameters.

(2) the parameter “max_depth” is selected as 12, and try different value of “min_samples_leaf”

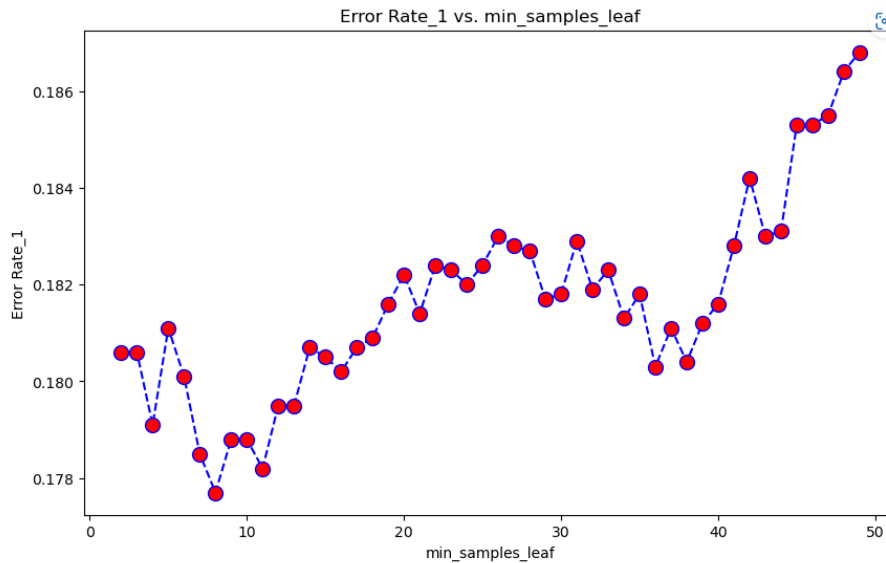


Figure 6: when the value of max_depth=12 , error rate vs “min_samples_leaf”

Note: Error rate = 1 - accuracy; (Using the test dataset to test)

As is shown in the figure above, the prediction error_rate is in range(0.176,0.188), that is acceptable result, showing that the “min_samples_leaf” parameter is less important than “max_depth”. It is really important to choose a suitable value of “max_depth”.

In the figure, the curve shows the best prediction performance occurs when “min_samples_leaf” is equal to 8. It is different from the former conclusion made, mostly because of other parameters’ difference.

2.4 In all of this section

Decision tree algorithm is simple and easy to understand, can deal with multi -route output issues. However, the generation of decision trees is unstable, The small changes in data or parameters may lead to the different decision trees generated. And more, it is easy to overfit. When the generated decision tree is very complicated, it always lead to overfit problems, which results in the algorithm not suitable to predict with the new test set. This outcome can be improved and avoided by means of pre-pruning or post-pruning. When the tree is growing with insufficient data, the tree’s growth is halted. Once the decision tree is generated, we cut off the subtrees which don’t have enough data. This process is called as post-pruning.

Traditional decision-tree algorithms have problems such as weak classification properties and complicated scale of the training data which consists of a lot of categories. Traditional decision tree algorithms have some problems such as weak classification properties and complicated scale of decision - trees in multiple categories.

3. Conclusion

The best prediction accuracy of my KNN classifier is 0.86, and the best prediction accuracy of my decision tree classifier is 0.82. The accuracy result may have potential to be more as I didn't try as many parameters as possible. Decision tree algorithm is simple and easy to understand, can deal with multi-route output issues. However, the generation of decision trees is unstable, The small changes in data or parameters may lead to the different decision trees generated. And more, it is easy to overfit. KNN algorithm is easy to understand, with no training, no estimation parameters. It has high accuracy and suitable for multiple labels classification.

References

- [1] Xiao H , Rasul K , Vollgraf R . Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms[J]. 2017.
- [2] https://en.wikipedia.org/wiki/Decision_tree_learning
- [3] Bergstra J , Bengio Y . Random Search for Hyper-Parameter Optimization[J]. Journal of Machine Learning Research, 2012.
- [4] Zhang M L , Zhou Z H . ML-KNN: A lazy learning approach to multi-label learning[J]. Pattern Recognition, 2007, 40(7):2038-2048.
- [5] Ijartet R , Harikumar R , Manjusha M . Performance analysis of KNN classifier and K-means clustering for robust classification of epilepsy from EEG signals[C]// IEEE International Conference on Wireless Communications, Signal Processing and Networking. 0.
- [6] Alhariri A . Detection of Covid-19 Fake News text data using Random Forest and Decision tree Classifiers. 2021.
- [7] Furui K , Ohue M . Compound virtual screening by learning-to-rank with gradient boosting decision tree and enrichment-based cumulative gain[C]// 2022.
- [8] https://blog.csdn.net/weixin_44715077/article/details/124577011
- [9] <https://plainenglish.io/blog/decision-tree-parameters-explanations-tuning-a2b0749976e5>

