Classification of unlabeled LoL match records with "Win/Loss" using team feature.

Abstract:

The objective of this project is to predict which team will win in the LoL match given parameters likes game duration, the number of towers, inhibitors and so on. The algorithm requires Python and panda, numpy and sklearn libraries in Python. In this project, decision tree, svm, multi-layer perception is used to build different models, then ensemble method is used to combine then together. In decision tree, bagging method is used to improve its accuracy.

Introduction:

Nowadays, more and more people are fall in love with playing electronic games. If an algorithm can be designed to predict the result of the games using features of the teams, it will be attractive and interesting. Our model is aiming at predicting the final result according to the features like first blood in the data.

Algorithm:

The algorithm contains three basic classifiers: decision tree, support vector machine and multi-layer perception. In decision tree, it has a loop. Inside the loop, it will first select the best feature (A), then for each value of A,

create new descendant of node. After that, it will sort training sample to leaf nodes. The loop carries on until training sample already classified. The parameters are set as “criterion equals to gini, max_depth equals to 2”.

In support vector machine, the algorithm aims at finding out miximum margin classifier, which depends on support vector given by few samples. Generally, the algorithm changes the problem into a quadratic optimization problem. If the data is non-linearly separable, a kernel must be added to transform it to linearly separable.

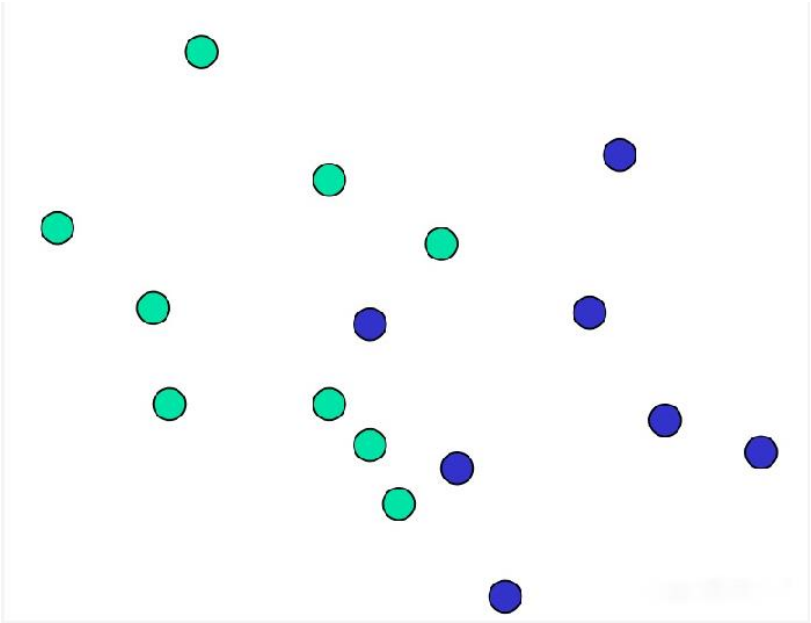

Figure 1 non-linearly separable data

In multi-layer perception, it scratch features in sample, multiplies with different weights, judge the class through activation function, then prints the output. Inside this algorithm, it has input layer, hidden layers and output layer. Its parameter “max_iter” is set as 10000. If the weight doesn’t converse after 10000 times, the classifier will use the weight updated lastly.

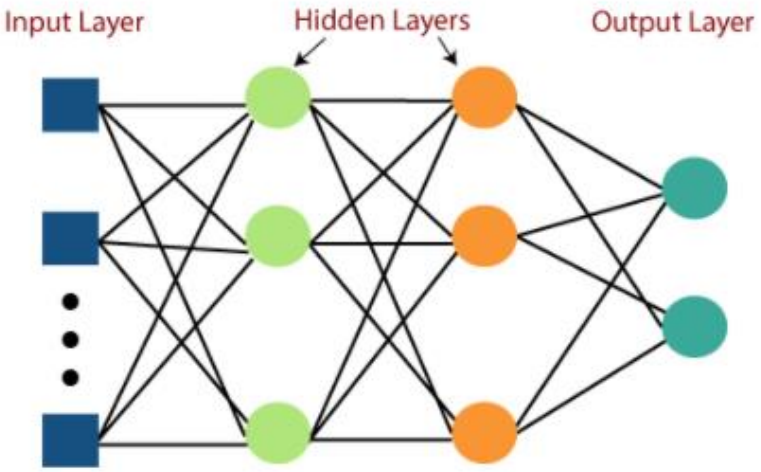

Figure 2 the structure of multi-layer perception

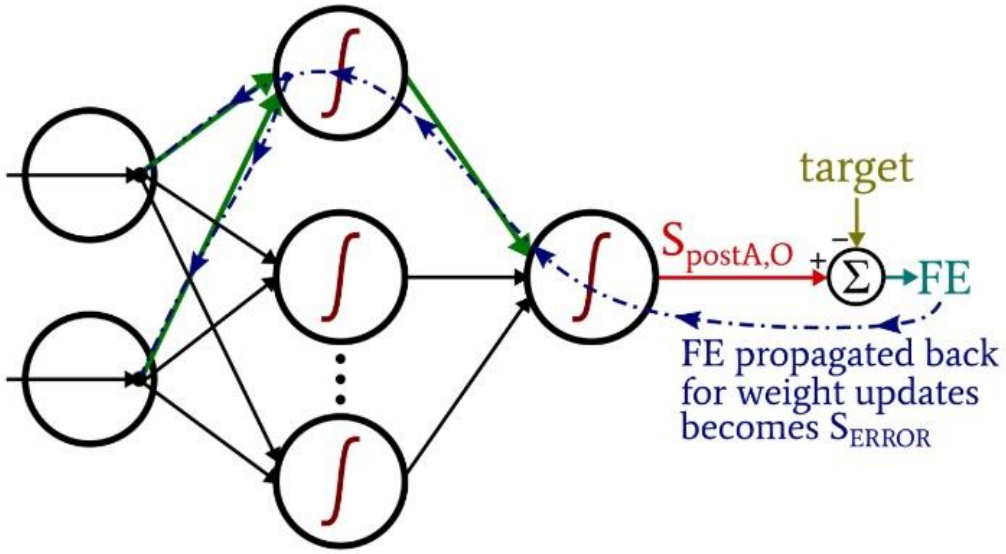

Figure 3 the iteration of multi-layer perception. The weight of each node will update backwards if there is error between real value and expected value after iteration.

In bagged decision tree, the dataset is split into 10 independent subsets in order to carry on cross validation, making full use of the data. The number of base estimators in the ensemble is set as 100 and the random state is set as 7.

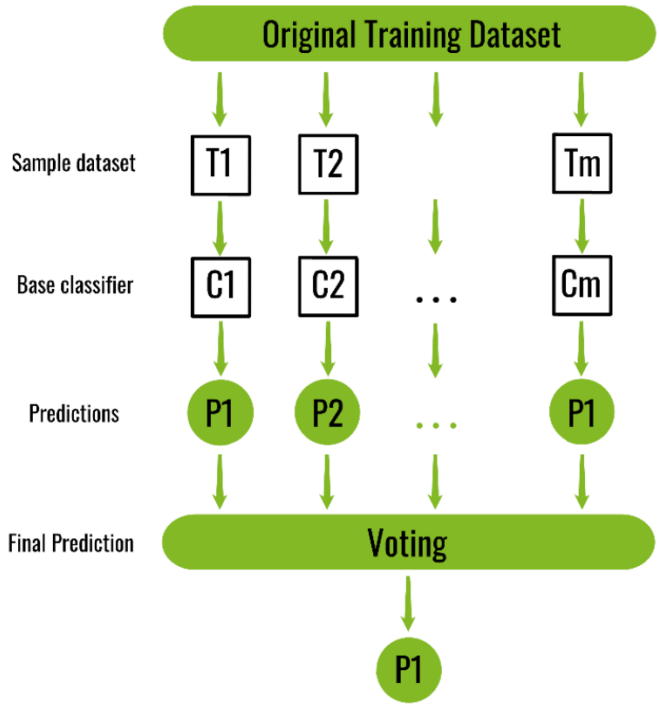

Figure 4 The bagging method. The dataset is split into different sample dataset, each of the dataset is used to train the classifier, then voting for the final result.

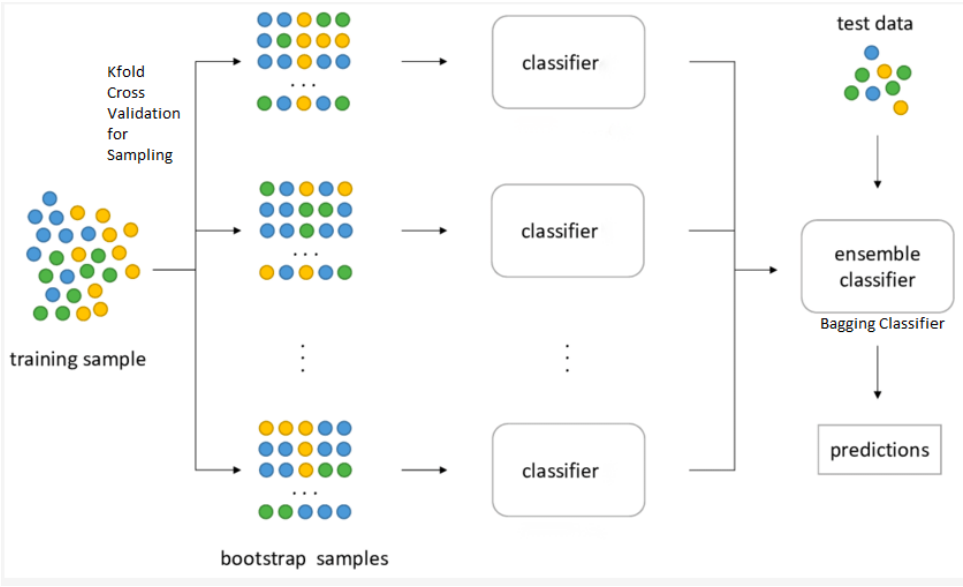

Figure 5 Kfold is used to separate the training sample into bootstrap samples.

In the voting classifier, it combines decision tree, support vector machine and multi-layer perception, using hard voting to classify the data.

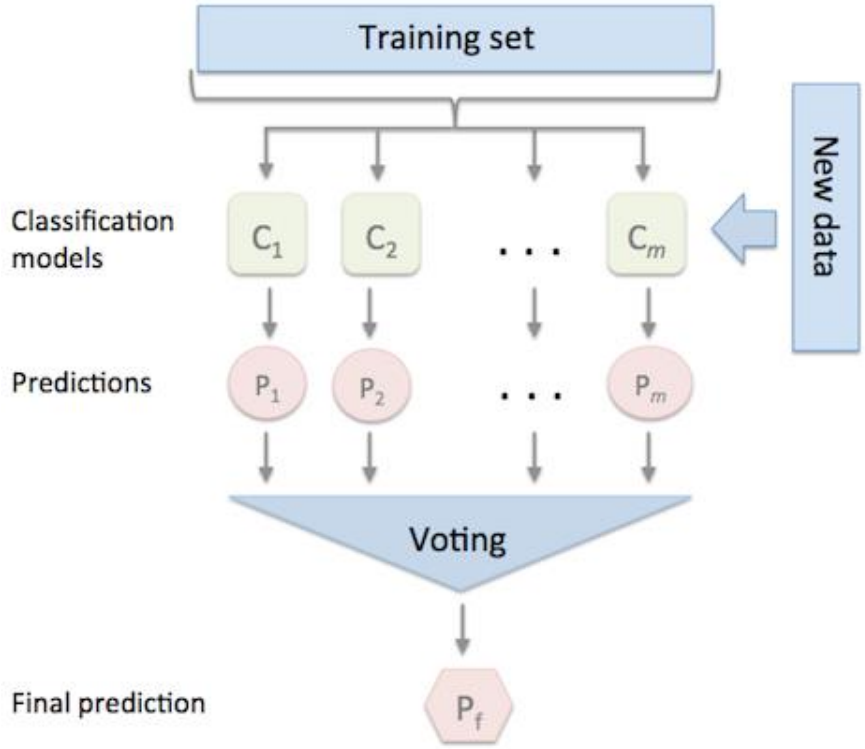

Figure 6 The model of voting classifier. The same data is predicted by different classifier.

The training set is split into training data and testing data in order to observe whether the model is overfit.

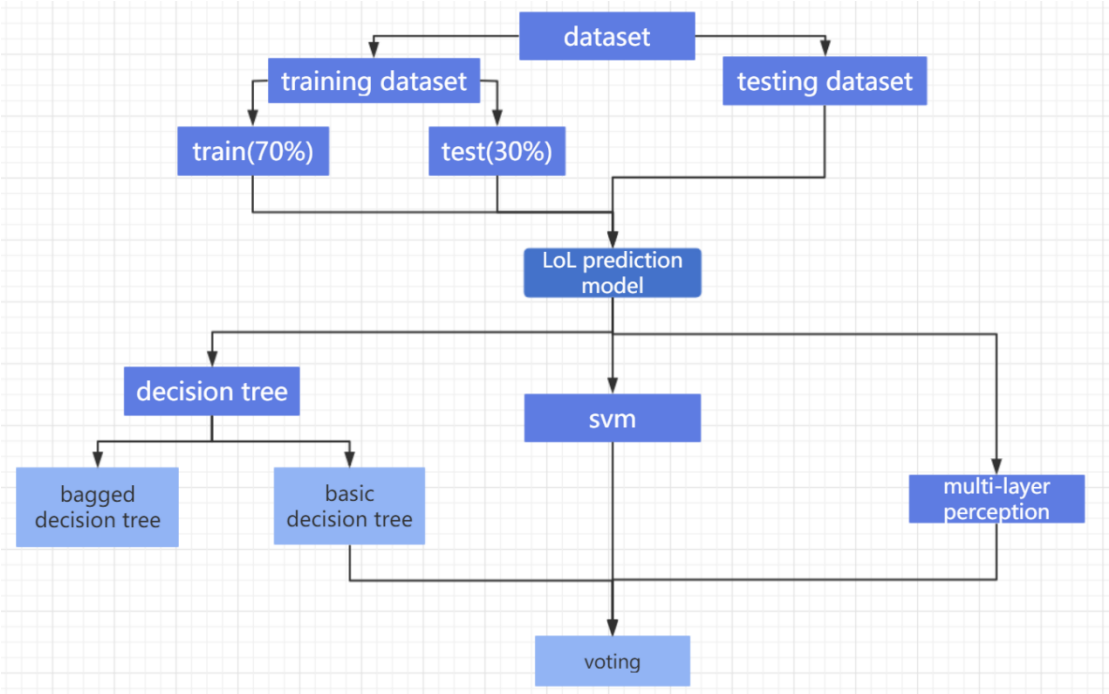

Figure 7 general overview

Requirement:

The algorithm requires Python and panda, numpy and sklearn libraries in Python. In order to use the algorithm, you should get python installed first. Then check whether panda, numpy, sklearn are in your library. If not, get them installed.

Result:

After training, it is cheerful that all classifiers have accuracy up to 96%. In decision tree and support vector machine, their running time is very small, in multi-layer perception and voting classifier, their run time is a little big large.

model	accuracy	training time(s)
decision tree	96.16%	0.061
svm	96.80%	1.813
multi-layer perception	96.98%	26.868
bagged decision tree	96.87%	2.998
voting classifier	97.01%	21.089

Figure 8 the table of training time and accuracy

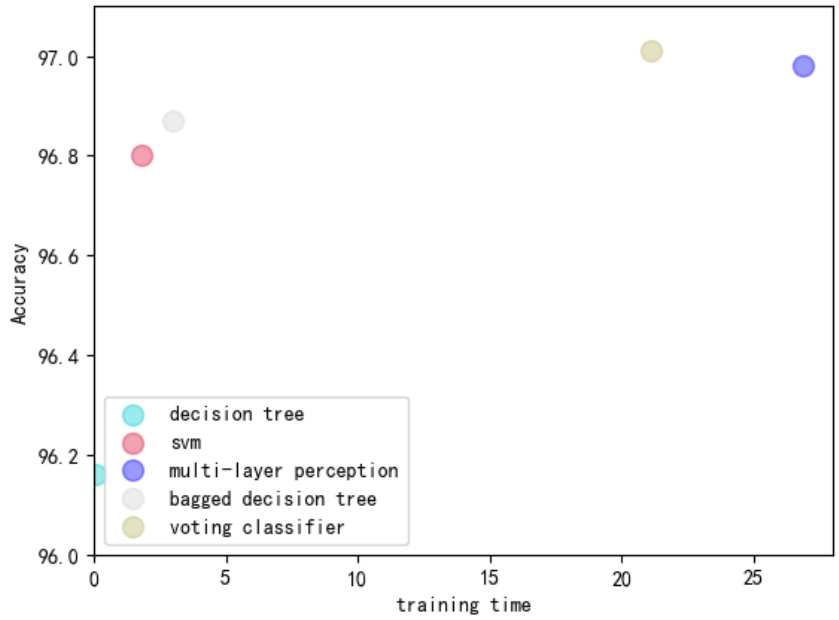

Figure 9 accuracy and training time of different classifiers, it is apparent that voting classifier and bagged decision tree performs well with high accuracy, while decision tree, support vector machine, multi-layer perception has small training time. As a result, we can see that after bagging and voting technology, the model performs better than before.

```
Trainning time of decision tree: 0.061416100012138486 Seconds
Accuracy of decision tree in training set: 0.9616048317515099
The Accuracy of decision tree: 0.961624404935393
The f1_score of decision tree: 0.9616237659705118
Trainning time of svm: 1.8128379000118002 Seconds
Accuracy of svm in training set: 0.9668895599654875
The Accuracy of svm: 0.9680365296803652
The f1_score of svm: 0.9680340311380377
Trainning time of multi-layer perception: 26.867714399995748 Seconds
Accuracy of multi-layer perception in training set: 0.9694779982743744
The Accuracy of multi-layer perception: 0.9697852909744487
The f1_score of multi-layer perception: 0.9697835557781349
Trainning time of BaggingClassifier: 2.9978714999742806 Seconds
The accuracy of bagged decision tree: 0.9687173002830495
Trainning time of voting classifier: 21.08869489998324 Seconds
The accuracy of voting classifier: 0.970126175773855
```

```
The classification_report of decision tree:
              precision    recall  f1-score   support

           1     0.9614    0.9640    0.9627     10322
           2     0.9637    0.9610    0.9623     10264

    accuracy                         0.9625     20586
   macro avg     0.9625    0.9625    0.9625     20586
weighted avg     0.9625    0.9625    0.9625     20586

The classification_report of svm:
              precision    recall  f1-score   support

           1     0.9626    0.9741    0.9683     10322
           2     0.9737    0.9619    0.9678     10264

    accuracy                         0.9680     20586
   macro avg     0.9681    0.9680    0.9680     20586
weighted avg     0.9681    0.9680    0.9680     20586
```

```
The classification_report of multi-layer perception:
              precision    recall  f1-score   support

           1     0.9798    0.9587    0.9692     10322
           2     0.9594    0.9801    0.9696     10264

    accuracy                         0.9694     20586
   macro avg     0.9696    0.9694    0.9694     20586
weighted avg     0.9696    0.9694    0.9694     20586

The classification_report of BaggingClassifier
              precision    recall  f1-score   support

           1     0.9684    0.9685    0.9685     10322
           2     0.9683    0.9682    0.9683     10264

    accuracy                         0.9684     20586
   macro avg     0.9684    0.9684    0.9684     20586
weighted avg     0.9684    0.9684    0.9684     20586
```

```
The classification_report of voting classifier:
              precision    recall  f1-score   support

           1     0.9721    0.9683    0.9702     10322
           2     0.9683    0.9720    0.9701     10264

    accuracy                         0.9702     20586
   macro avg     0.9702    0.9702    0.9702     20586
weighted avg     0.9702    0.9702    0.9702     20586
```

Figure 10 result after running the code

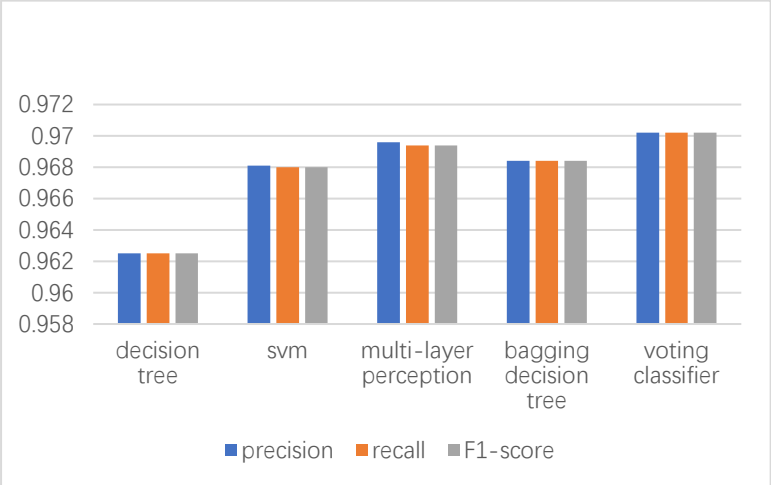

Figure 11 Precision, recall, F1-score of different classifiers

Comparison and discussion:

Through doing this program, I can gain that when using computer to classify which class a data belongs to, there are diverse classifiers to choose. Different classifiers will have different training time and accuracy. In this program, decision tree, support vector machine, multi-layer perception, bagged decision tree and voting classifier.

In terms of decision tree, it has advantages that it requires less effort for data preparation during pre-processing and it is very intuitive and easy to understand. However, when the data has small change, it will cause a big change in the decision tree, which cause the instability of decision tree. As figure 5 shows, decision tree sometimes can be very complex. What's more,

decision tree is not suitable for regression problem, only suitable for classification problem.

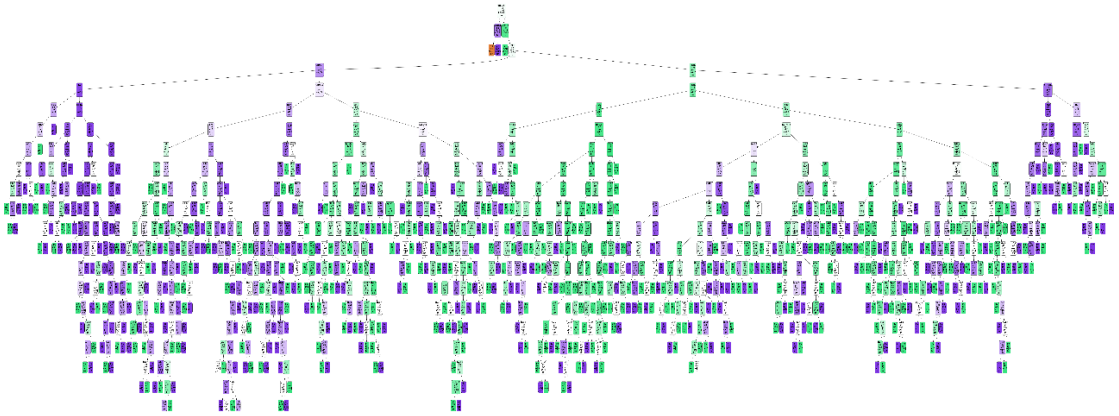

Figure 12 the complexity of decision tree

As for support vector machine, it can be train easily because no local optimal and scale well to high dimensional data. The tradeoff between classifier complexity and error can be controlled explicitly. It also has drawbacks like it will turn slow when the sample is large and a good "kernel function" is needed.

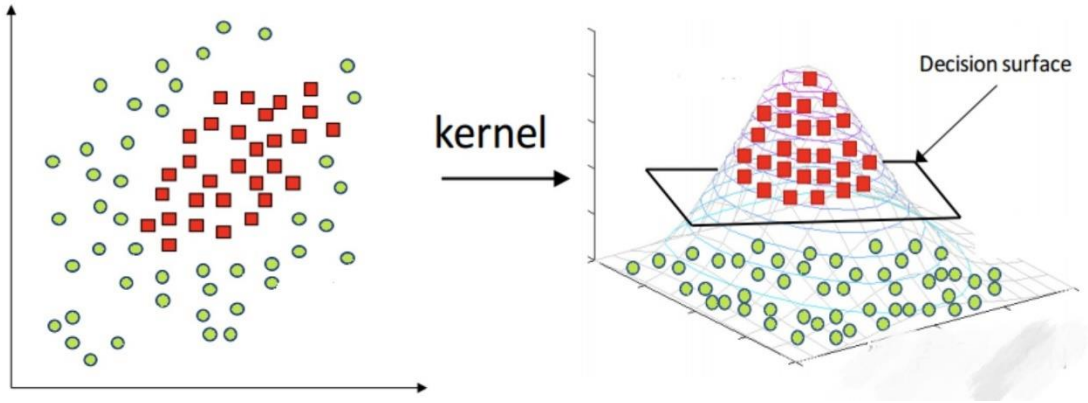

Figure 13 the classification of support vector machine with a good "kernel function"

With regard to multi-layer perception, it is a very good algorithm to use for the regression and mapping. However, it cannot guarantee that the minima are the global minima. Usually, the training time of multi-layer perception is very large.

Reference:

1. [J] EnsembleVoteClassifier (2019.5.20) http://rasbt.github.io/mlxtend/user_guide/classifier/EnsembleVoteClassifier/
2. [J] ML | Bagging classifier https://www.geeksforgeeks.org/ml-bagging-classifier/
3. [A] How to Create a Multilayer Perceptron Neural Network in Python (2020.1.19) https://www.allaboutcircuits.com/technical-articles/how-to-create-a-multilayer-perceptron-neural-network-in-python/