Use Multiple Classifiers to Analyze Lol Data and Predict the Winning Teams

1. Abstract：

This project presents a method to effectively predict the winning team of the LOL electronic competition through a variety of classifiers training the existing game data. First, the existing data is divided into the features and the result, after the data preprocessing, the redundant feature will be discarded. Then train the obtained features with multiple classifiers, and use the Fusion method to integrate basic classification for new classification result. Finally, best performing classifier is selected to predict the winning team by evaluating the above classifications.

2. Introduction:

As an emerging e-sports activity, Lol game is becoming more and more popular in the younger generation. The attribution of the winner of each match always arouse extensive discussion. However, there will be a large number of matches recorded during every match. By processing and training the game data. There is a 97% accuracy in predicting the success of the winning team in this project. First of all, the training data is preprocessed by the Max-min scaling and standardization of the features, which is reducing the data dimension. Secondly, in the processing of training data, this project uses decision trees (DT), K-nearest neighbor algorithm (KNN), multi-layer perceptron (MLP) and artificial neural network (pytorch) (ANN) as base classifiers which are trained separately. Meanwhile, the fusion method is also used to ensemble the four classifiers to obtain new classification result. The specific process is shown in **Figure 1** :

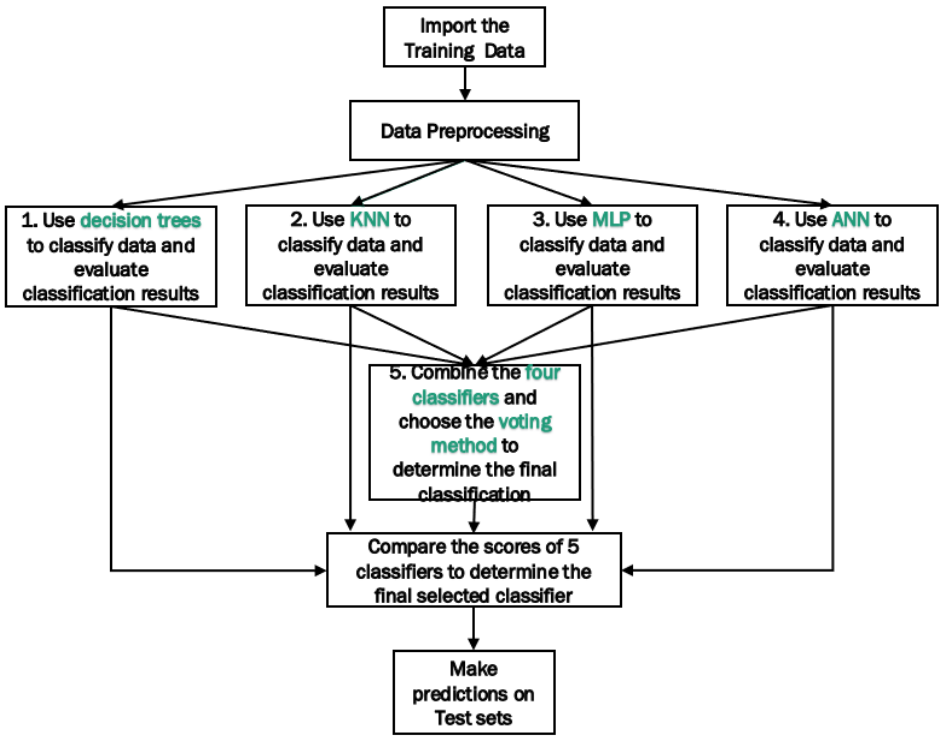

Import the Training Data
Data Preprocessing
1. Use decision trees to classify data and evaluate classification results
2. Use KNN to classify data and evaluate classification results
3. Use MLP to classify data and evaluate classification results
4. Use ANN to classify data and evaluate classification results
5. Combine the four classifiers and choose the voting method to determine the final classification
Compare the scores of 5 classifiers to determine the final selected classifier
Make predictions on Test sets

Figure 1 Data training process

Finally, the prediction results are compared with the true classification results to obtain classification algorithm having the highest score, and use this type of the algorithm to predict the results of the new matches.

3. Algorithms:

3.1 Data observation and processing:

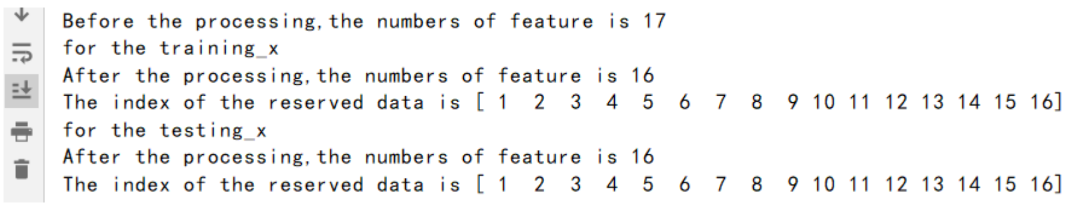
Divide the competition data into two parts, use 80% of the data to train the classifier, and use the remaining data to test the accuracy of the classifier. For the data to be trained, we divide it into feature attributes and class labels. Since the number of feature attributes is 21 and the value ranges of different features are quite different, we normalize and standardize the feature data. The processing methods are as follows:

Min-Max scaling:

Since these 22 feature values have different dimensions (for example, the game ID does not even have dimensions!), and ranges (such as game duration and the number of defensive towers destroyed) due to various standards , in order to eliminate the influence, we use Min- Max scaling maps the data to the interval [0~1], making the standard in the same order of magnitude.

Remove low-variance features：

After the first step of the data, in order to select the feature variables that contributed the most to the model output and from the 22 attributes, this step deletes the features whose variance is 0 (assume these features have no effect on the output label) to save training time.



```
Before the processing,the numbers of feature is 17
for the training_x
After the processing,the numbers of feature is 16
The index of the reserved data is [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16]
for the testing_x
After the processing,the numbers of feature is 16
The index of the reserved data is [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16]
```

After data preprocessing, what can be found is that some features are considered useless and discarded. We use the processed features and class labels for training

3.2 Base classifiers:

3.2.1. Decision tree：

The decision tree algorithm produces a weightless tree with the criterion-Gini coefficient (CART algorithm), and at the same time there is no restriction on the depth of the tree. When splitting features, it finds the local optimal division point randomly among the local division points, it also limits the minimum number of samples required for subdividing internal nodes and samples required by leaf nodes to prune the tree. When the former is less than 10 or the latter is less than 5, the decision tree stops splitting the data. On the contrary, the classifier does not limit the maximum number of samples that the two can accommodate. Finally, the decision tree classifier is shown in the **Figure 2** :

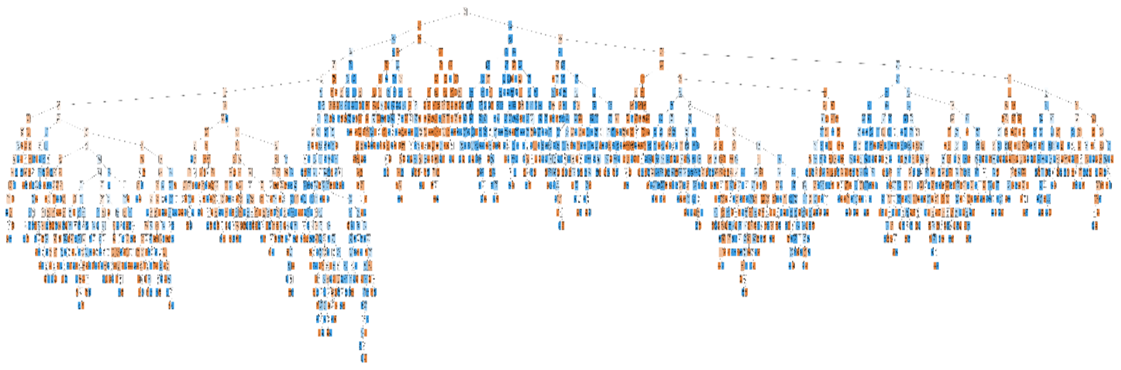


Figure 2 Decision Tree

3.2.2. K-nearest neighbor algorithm：

The principle of the KNN algorithm is to automatically select the K points closest to the prediction point and observe the classes of these points, and then divide the labels of the predicted points into the categories that appear the most times. In order to find the appropriate K, the KNN classifier of the project is trained with K= 1~36. It is found that the classifier performs best when K=9. At the same time, brute algorithm is used to build the KNN model. For the definition of distance, the common Euclidean geometric distance formula is used to judge the distance between the predicted point and the surrounding points.

3.2.3. multi-layer perceptron：

Multilayer perceptron is a common artificial neural network structure. It achieves the fitting effect by processing the incoming data of the perceptual input layer and adjusting the weights on the network. The activation function of this project is rectified linear unit (Relu), function The formula is as follows：

$$Relu(x) = \begin{cases} x \ \ if\ x > 0 \\ 0 \ \ if\ x < 0 \end{cases}$$

Since the forward calculation and the partial derivative of the Relu(x) function are simple with no exponential operation required, and the right side is closed at the same time, which will make many hidden layer outputs 0, that is, the network becomes sparse, which can alleviate the problem of overfitting to a certain extent.

At the same time, Relu is not prone to the problem of gradient divergence, so we choose the gradient descent method to optimize the weight of the neural network, which defines the learning rate of the gradient descent method as 1, and chooses to "reshuffle the card" after each iteration of the sample to ensure the independence. Besides, this classifier will stop training when the accuracy of classification is not significantly improved after multiple iterations.

3.2.4. artificial neural network (Pytorch)

This is a neural network defined by the user according to the library Pytorch. There are only two hidden layers in the network. Meanwhile, the neural network is trained 400 times using the activation function Sigmod with the learning rate-0.01. After each iteration, the Adam algorithm (a stochastic gradient-based optimizer) provided by torch to optimize and update the weights. The intermediate nodes and structure of the neural network are in **Figure 3**:

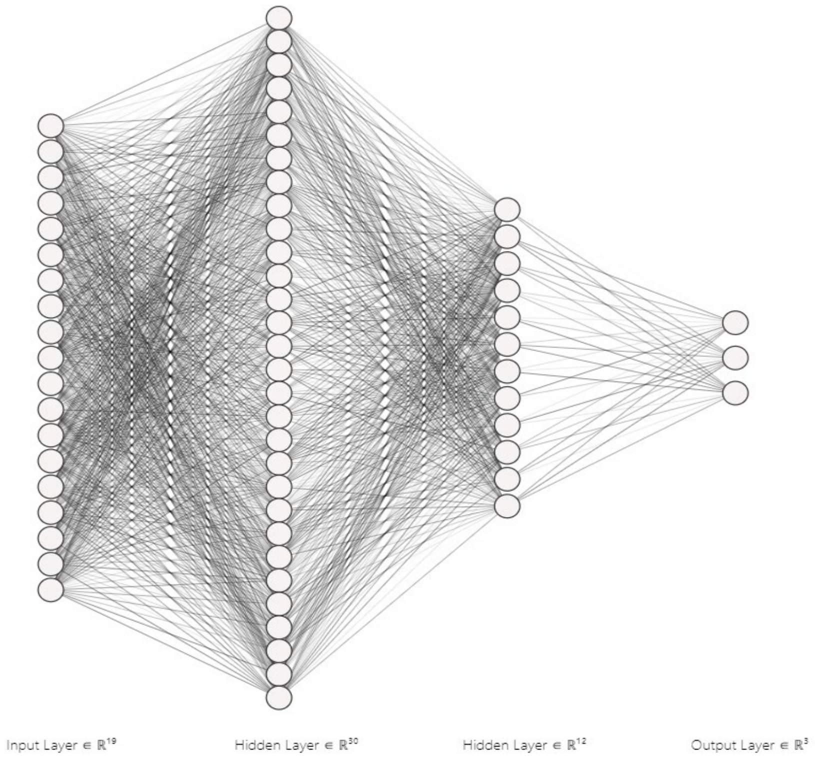
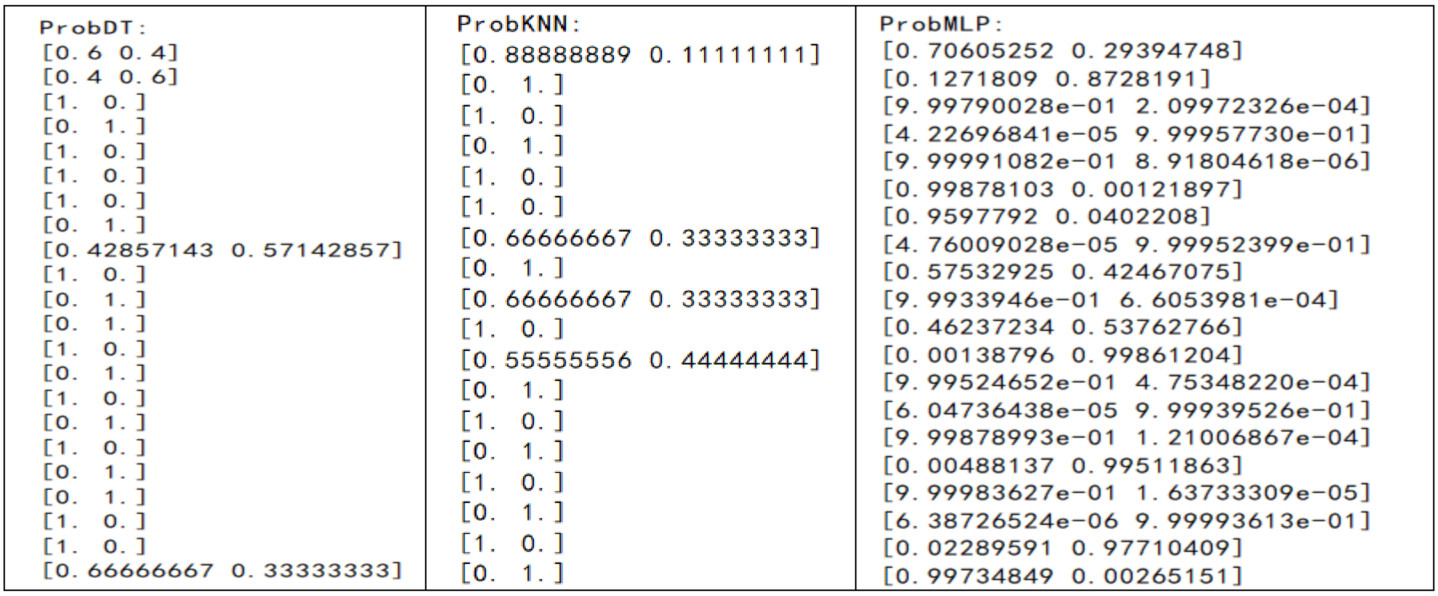

Input Layer ∈ $\mathbb{R}^{19}$
Hidden Layer ∈ $\mathbb{R}^{30}$
Hidden Layer ∈ $\mathbb{R}^{12}$
Output Layer ∈ $\mathbb{R}^{3}$

Figure 3 ANN structure

3.3 Fusion Method:

In order to reduce the running time of the fusion method, at the same time, both MLP and ANN have similarities in some ways(for example, they use the neural networks to fit the data,). Therefore, this method integrates KNN, MLP and DT three classifiers to produce new classification results, because for each feature value group (each row), the classifier will evaluate the possibility that the class label obtained by the set of feature values belongs to two classes respectively. The evaluation results of the above three classifiers for the possibility are as follows:



```
ProbDT:
[0. 6 0. 4]
[0. 4 0. 6]
[1.  0. ]
[0.  1. ]
[1.  0. ]
[1.  0. ]
[1.  0. ]
[0.  1. ]
[0. 42857143 0. 57142857]
[1.  0. ]
[0.  1. ]
[0.  1. ]
[1.  0. ]
[0.  1. ]
[1.  0. ]
[0.  1. ]
[1.  0. ]
[0.  1. ]
[0.  1. ]
[1.  0. ]
[1.  0. ]
[0. 66666667 0. 33333333]
```

```
ProbKNN:
[0. 88888889 0. 11111111]
[0.  1. ]
[1.  0. ]
[0.  1. ]
[1.  0. ]
[1.  0. ]
[0. 66666667 0. 33333333]
[0.  1. ]
[0. 66666667 0. 33333333]
[1.  0. ]
[0. 55555556 0. 44444444]
[0.  1. ]
[1.  0. ]
[0.  1. ]
[1.  0. ]
[0.  1. ]
[1.  0. ]
[0.  1. ]
```

```
ProbMLP:
[0. 70605252 0. 29394748]
[0. 1271809 0. 8728191]
[9. 99790028e-01 2. 09972326e-04]
[4. 22696841e-05 9. 99957730e-01]
[9. 99991082e-01 8. 91804618e-06]
[0. 99878103 0. 00121897]
[0. 9597792 0. 0402208]
[4. 76009028e-05 9. 99952399e-01]
[0. 57532925 0. 42467075]
[9. 9933946e-01 6. 6053981e-04]
[0. 46237234 0. 53762766]
[0. 00138796 0. 99861204]
[9. 99524652e-01 4. 75348220e-04]
[6. 04736438e-05 9. 99939526e-01]
[9. 99878993e-01 1. 21006867e-04]
[0. 00488137 0. 99511863]
[9. 99983627e-01 1. 63733309e-05]
[6. 38726524e-06 9. 99993613e-01]
[0. 02289591 0. 97710409]
[0. 99734849 0. 00265151]
```

Figure 4 some evaluation of the classifier

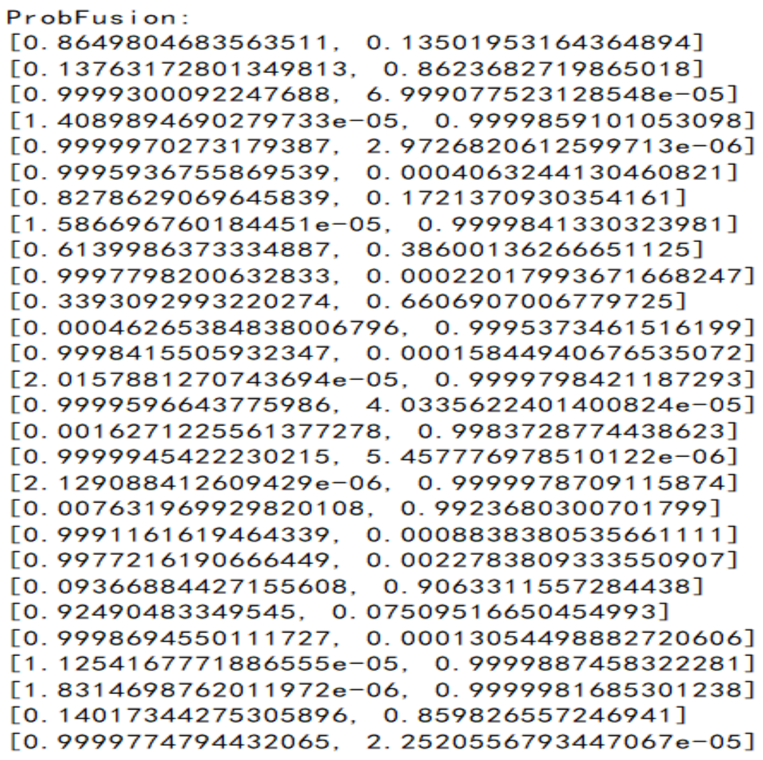
For each set of feature values, the three classifiers will give their own evaluations, among them, "ProbDT", "ProbKNN", "ProbMLP" represent the evaluation of DT classifier, KNN classifier and MLP classifier respectively.

The fusion method is as follows:

1. Multiply the evaluations of the three classifiers on the possibility of the same set of features (each row) belonging to different classes by the weights of the three classifiers. The weight calculation formula is

$$w_i = \frac{Accuracy\ score\ of\ classifier\ i}{100}$$

2. Calculate the average value generated by the results of the first step, and then classify according to the results and then determine the final output class label based on the average value. Finally, the evaluation of Fusion method is shown in the figure:



```
ProbFusion:
[0.8649804683563511, 0.13501953164364894]
[0.13763172801349813, 0.8623682719865018]
[0.9999300092247688, 6.999077523128548e-05]
[1.4089894690279733e-05, 0.9999859101053098]
[0.9999970273179387, 2.9726820612599713e-06]
[0.9995936755869539, 0.00040632441304608 21]
[0.827862906964 5839, 0.172137093035416 1]
[1.586696760184451e-05, 0.9999841330323981]
[0.6139986373334887, 0.38600136266651125]
[0.9997798200632833, 0.00022017993671668247]
[0.339309299322027 4, 0.6606907006779725]
[0.00046265384838006796, 0.9995373461516199]
[0.9998415505932347, 0.00015844940676535072]
[2.015788127074369 4e-05, 0.9999798421187293]
[0.9999596643775986, 4.0335622401400824e-05]
[0.0016271225561377278, 0.9983728774438623]
[0.9999945422230215, 5.457776978510122e-06]
[2.129088412609429e-06, 0.9999978709115874]
[0.0076319699298201 08, 0.9923680300701799]
[0.999116161946433 9, 0.0008838380535661111]
[0.997721619066644 9, 0.0022783809333550907]
[0.0936688442715560 8, 0.906331155728443 8]
[0.92490483349545, 0.0750951665045499 3]
[0.9998694550111727, 0.00013054498882720606]
[1.1254167771886555e-05, 0.9999887458322281]
[1.8314698762011972e-06, 0.9999981685301238]
[0.14017344275305896, 0.859826557246941]
[0.9999774794432065, 2.252055679344706 7e-05]
```

Figure 5 Some evaluation of Fusion method

4. Requirements:

In order to successfully run this project, the required libraries are as follows:

4.1 Record running time:

Datatime

4.2 Data preprocessing:

numpy, panda,

StandardScaler in sklearn.preprocessing,

VarianceThreshold in sklearn.feature_selection

4.3 Run the KNN classifier:

KNeighborsClassifier in sklearn.neighbors

train_test_split in sklearn.model_selection

accuracy_score in sklearn.metrics

4.4 Run ANN (Pytorch)

torch

train_test_split in sklearn.model_selection

accuracy_score in sklearn.metrics

4.5 Run MLP:

MLPClassifier in sklearn.neural_network

train_test_split in sklearn.model_selection

accuracy_score in sklearn.metrics

4.6 Run DT:

Image in IPython.display

StringIO in six

tree in sklearn

train_test_split in sklearn.model_selection

accuracy_score in sklearn.metrics

os

export_graphviz in sklearn.tree

4.7 Run fusion method:

numpy, matplotlib.pyplot

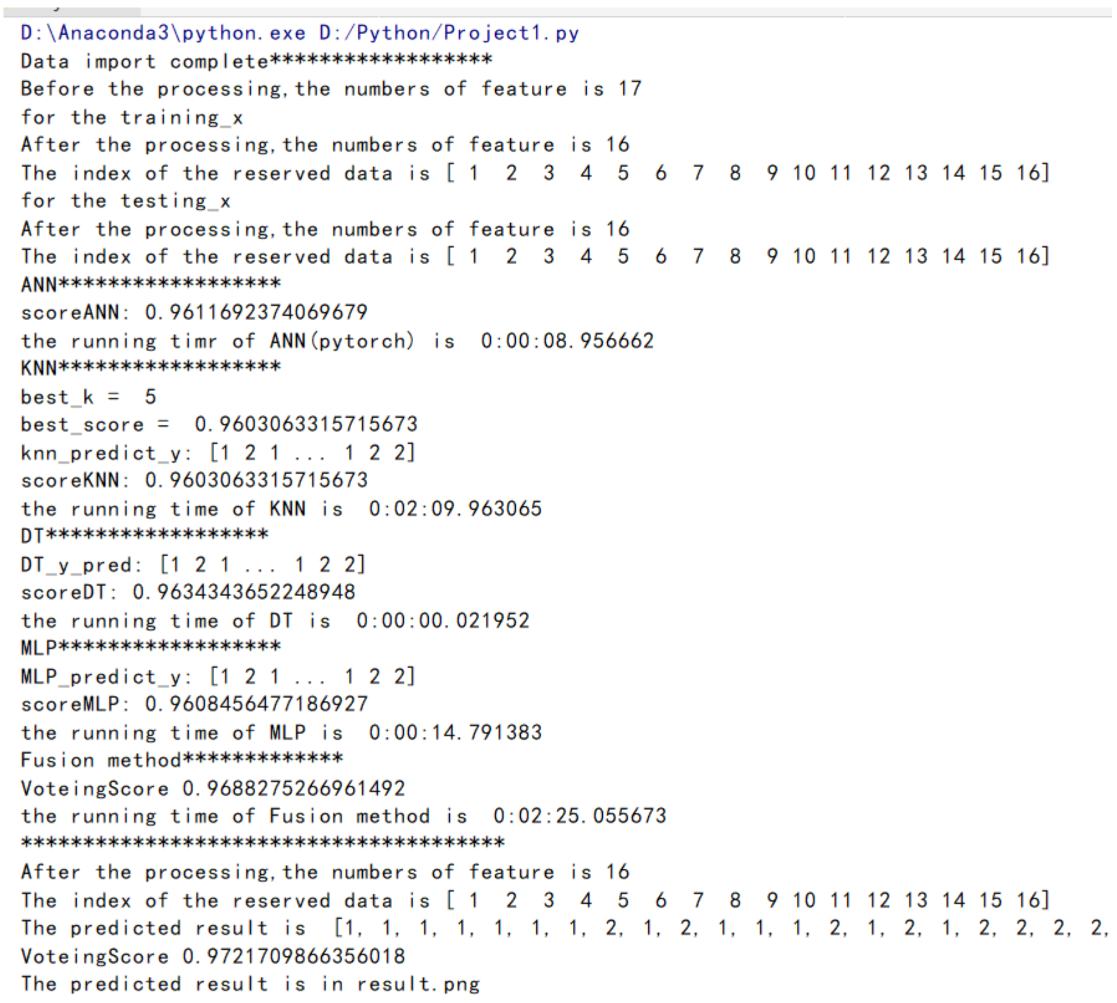
other libraries needed to run base classifiers

5. Results:

After many tests, the accuracy and running time of the classifier (both are averaged) are shown in the **Table 1** and **Picture 1**:

Classifier name	Accuracy Score	Running time
Decision Tree (DT)	0.9611	0:00:00.029
K-nearest neighbor algorithm(KNN)	0.9603	0:02:13.886
multi-layer perceptron(MLP)	0.9608	0:00:16.178
artificial neural network (pytorch)　(ANN)	0.9612	0:00:08.956
Fusion method	0.9677	0:02:30.444

Table 1 Result of classifiers



```
D:\Anaconda3\python.exe D:/Python/Project1.py
Data import complete*****************
Before the processing,the numbers of feature is 17
for the training_x
After the processing,the numbers of feature is 16
The index of the reserved data is [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16]
for the testing_x
After the processing,the numbers of feature is 16
The index of the reserved data is [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16]
ANN*****************
scoreANN: 0.9611692374069679
the running timr of ANN(pytorch) is  0:00:08.956662
KNN*****************
best_k =  5
best_score =  0.9603063315715673
knn_predict_y: [1 2 1 ... 1 2 2]
scoreKNN: 0.9603063315715673
the running time of KNN is  0:02:09.963065
DT*****************
DT_y_pred: [1 2 1 ... 1 2 2]
scoreDT: 0.9634343652248948
the running time of DT is  0:00:00.021952
MLP*****************
MLP_predict_y: [1 2 1 ... 1 2 2]
scoreMLP: 0.9608456477186927
the running time of MLP is  0:00:14.791383
Fusion method*************
VoteingScore 0.9688275266961492
the running time of Fusion method is  0:02:25.055673
***************************************
After the processing,the numbers of feature is 16
The index of the reserved data is [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16]
The predicted result is  [1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 2, 1, 2, 1, 2, 2, 2, 2,
VoteingScore 0.9721709866356018
The predicted result is in result.png
```

Picture 1 Running process

After analyzing the table and figure, we observed that the Fusion Method classifier performs best among these classifiers, so we used this trained classifier to predict the winning team of the new game. The prediction results are as follows in **Figure 6**:

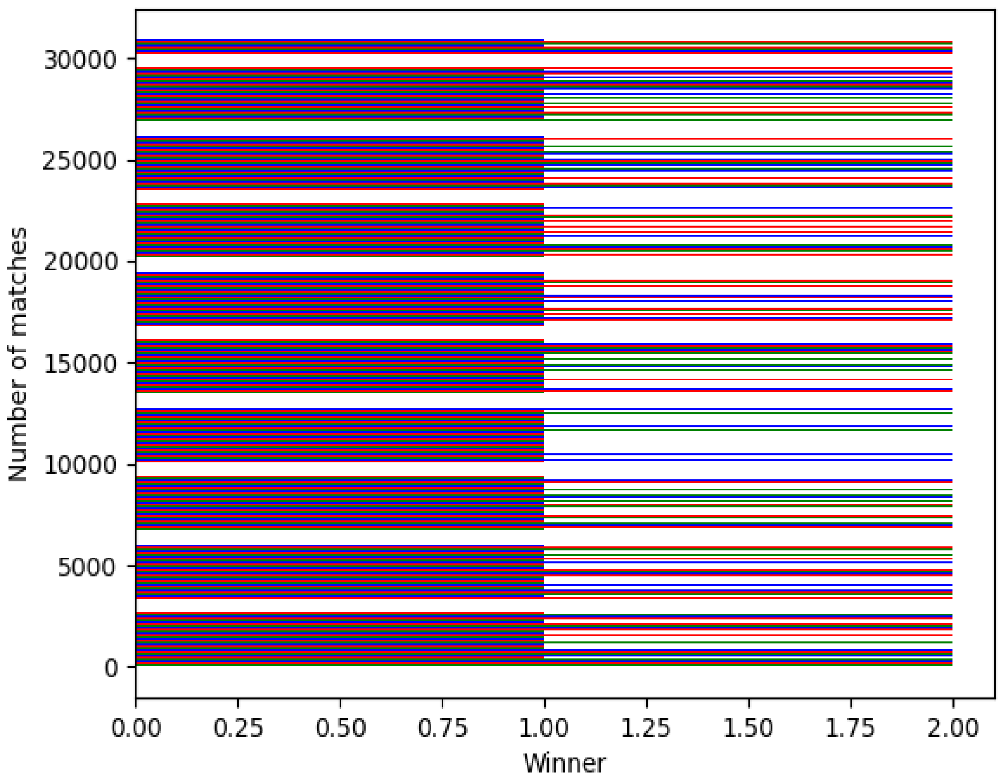


Figure 6 some of Predicted data results

6. Comparison and discussion:

Through this project, I have gained numerous experiences:

1. Understand a lot of methods about data preprocessing. When faced with lots of complicated data, observing the data should be the first action, discard useless data (such as the feature --"season" in the training data), reduce the data dimension, and merge similar attributes to improve the efficiency of the classifier.

2. Understand the principles, implementation methods of different classifiers, and how to compare the performance among different classifiers. Firstly, although the KNN algorithm is the simplest, the accuracy score is different due to the uncertainty of K, and the multiple calculations of the distance make the amount of calculation large. Secondly，MLP has strong predictive ability for training data, but the running time is long, and there are a large number of important parameters that need to be manually specified, at the same time, the hidden layer in the network is also difficult to get. Thirdly, DT uses the shortest time for data training, in addition, it provides a detailed

visual analysis, which helps user clearly understand every decision.
3. Try to write an ensemble method to combine the evaluation of the possibility of different class labels for the same feature group (each row) by different classifiers to obtain new classification results.

The difficulties encountered in this project and the improvement planned in the future are as follows:

1. Compared with other classifiers, the classification accuracy of the ANN (Pytorch) designed by the user is not very good, only about 50.3% at first. After analyze the process, I think it is because learning rate, number of iterations, and the redundant features are not discarded or merged. When improving the classification accuracy of ANN, I try to increase the number of hidden layers, appropriately change the number of iterations and improve the learning data, at last the accuracy is up to 96%.

2. When integrating base classifiers, the fusion method I use is just to multiply the evaluations of the three classifiers on the possibility of the same set of features (each row) belonging to different categories by the weights of the three classifiers. This classification method is not detailed and intelligent enough. If I have time, I will use other more efficient classification methods, such as Adaboost algorithm, etc.