Report of Project 1

Introduction:

The goals and requirements of the project are very clear-use a certain method or multiple methods to adjust the classifier in a given set of training data, so as to get a stronger classifier for the test data. The prediction is compared with the actual results of the test data to get the accuracy of the enhanced classifier. At the same time, output the obtained decision tree, even if the obtained decision tree is visualized. Due to the large number of training samples, machine learning is initially considered.

Algorithms:

Because I am not familiar with using functions in python to read excel, I merged the two excel files together and used the functions in the pandas library to read the data. According to the ratio of data before merging, the number of samples used as training in an excel accounted for 0.6 of the entire sample, so I set the parameters of the training samples to 0.6. Since I am not familiar with the code of machine learning, I also fixed the learning rate to 1. To simplify calculations. In the selection of sample features, except that the winner, duration, and game id are not selected, the rest are selected as features. Based on the large amount of data, I choose supervised learning as a method of strengthening the classifier (that is, importing relevant libraries in sklearn learning). At the same time, the construction

of the decision tree is used to further strengthen the classifier. At this time, in order to better fit the training samples, the depth parameter of the decision tree is modified to 10. Use the fix function to build and train the classifier, and the accuracy function to get the classifier The accuracy and predict function are used to estimate the detection samples, and the accuracy function is used to judge the effect of the classifier on the detection samples. After that, use graphviz, IPython. display, plot and other function libraries to output the constructed decision tree.

Requirement:

Pandas：Used to read data from excel

Numpy：Can be used to store and process large matrices

Sklearn.metrics: Obtain the accuracy between the predicted result and the actual result (ie, evaluate the effect of the classifier)

Sklearn.tree: Build a decision tree

Sklearn.model_selection: Learning model selection

IPython.display: Create this decision tree in your computer

Pydotplus: Draw image

Os: Display all files in the current directory/delete a file/get file size...

Matplotlib：It is used to generate plots, histograms, power spectra, bar graphs, error graphs, scatter plots, etc.

Matplotlib.image: Instantiate image

Comparison and discussion:

Regarding the prediction of the final result of the League of Legends game, I used

a supervised learning method. After the model was established, the classifier was continuously trained to obtain a powerful classifier. In this process, the library I used the most was sklearn. At the same time, I have a clearer understanding of the functions and parameter settings in the library, and also understand the method of adjusting the weight of each feature value and the impact on the final result. Out of my incomplete understanding of supervised learning, in the whole project, I only used one method to strengthen the classifier, and the training samples are very large. Although there are many feature values in the samples, this will still This leads to overfitting, especially since I only used one method to process the training samples. This leads to the final classifier that is suitable for processing training samples, but is extremely sensitive to certain "impurities" in the test samples, leading to erroneous results when processing these data. If time permits, I will divide the training sample into three parts (Part A and Part B and Part C). Part A uses a supervised learning method to strengthen a classifier, and Part B uses a random forest method to strengthen a classifier. Part C uses bagging to strengthen a classifier. The three classifiers obtained are used to process the test samples to obtain three results, and the result with the most occurrences is selected as the final result to reduce over-fitting or under-fitting.