Classification of unlabeled LoL match records with “Win/Loss”

Introdution:

League of Legends, which is well as known as its abbreviayion, LoL, is one of the most famous MOBA(Multiplayer Online Battle Arena) games in the world. Electronic Sports may be a part of sports in future and LoL is the game that is often used in Electronic Sports. Despite of its controversy, electronic sports are now being more and more popular all over the world, especially in China. S10 games (Games for LoL teams), which are held recently, attracted plenty of crazy fans who love this game to watch.

There is large amount of statistics in every sports, soccer, basketball, marathon and so on. If we analyze these statistics of all competitors, we can get much important information of each game and managers or players can improve their tactics to make their competitions better. That’s why statistic analyzers are very important for all sport teams. With a good group of statistic analyzers, the sport teams can go to a higher competitive level. Statistics are more important in electronic sports than any other sports because electronic sports are completely based on statistics. The players of electronic sports teams can adjust themselves by analyzing their statistics after playing.

10 players will take part in one LoL match, and they will be divided into two teams. To make things simply, we can call them team 1 and team 2. In each match, only one team could win. There are millions of solo players’ statistics provided for this project. From these solo players’ statistics, we can find which team win at last. Players’ statistics are important because the solo player’s performance determine the final result of the match. If one player perform disastrously, his team would probably lose.

Our objectives of this project is to create classifiers that take as inputs from any fields from the match record, and labels this record as a "1" or a "2"(which stands for the team’s name). The requirement of this project is that we must label the record with the accuracy higher than 50%(if we guess the labels randomly, the accuracy will be 50%

when the size of the data is huge enough). Of course, we should always pursuit higher accuracy.

Algorithm:

The classifier that would be used in this program is the decision tree and k-NN.

Now I would state how the decision tree algorithm works.

The first thing we must do is that we must import all packages that would be used(mentioned below). Then, we have to load the data into the program. The data of the solo players is in csv file format. The statistic of each player could be seen as a list in Python. Numpy format can be input to the classifier so we need to convert the values in csv format to Numpy by using pandas. The third thing we should do is to create the classifier. After creating classifiers, we will train the data and then use the classifier to predict the labels with our created classifier.

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=20, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

These are all the parameters that are used in the decision tree. From the picture above we can find it that there are too many parameters for a decision tree. In order to simplify the work, we do not need to set all of them by ourselves to classify these data. Python set them automatically so it can reduce our quantity of work. The most important parameter in the decision tree is the depth of the tree, which majorly affect the accuracy of the decision tree classifier.

Generally speaking, as the depth of the decision tree increases, the accuracy will also increase as well. However, the accuracy could only increase very slowly when the decision tree is very deep. When the depth of the decision tree is 20, the accuracy is almost satrurated. Therefore, I set its depth as 20.

K-NN is another simple classifier. In chapter 1 of our course, we learnt the concept of data's similarity and dissimilarity. In data's similarity, distance measure is an important way to determine the whether the datas are similar. K-NN algorithm use the distance as judgment. The main idea of k-NN algorithm is that if most of the k nearest

samples of a sample in the feature space belong to a certain category, then the sample also belongs to this category and has the characteristics of the samples in this category.

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

The picture above show all parameters of k-NN classifier. The parameter n_neighbors is the most important and we should set if by ourselves. I set the number of neighbors as 5, which is not too small or large. The other important parameter is metric. In our course, we learnt the concept of Minkowski distance as well. This parameter is set automatically and we do not need to adjust it.

Ensemble that we have learnt include Bagging and Adaboosting. I will also use them in this project because they can help me to increase the accuracy. I would create ensembles Adaboosting and Bagging with decision tree to compare their accuracy with use decision tree only.

```
BaggingClassifier(base_estimator=DecisionTreeClassifier(ccp_alpha=0.0,
                                                        class_weight=None,
                                                        criterion='gini',
                                                        max_depth=None,
                                                        max_features=None,
                                                        max_leaf_nodes=None,
                                                        min_impurity_decrease=0.0,
                                                        min_impurity_split=None,
                                                        min_samples_leaf=1,
                                                        min_samples_split=2,
                                                        min_weight_fraction_leaf=0.0,
                                                        presort='deprecated',
                                                        random_state=None,
                                                        splitter='best'),
                  bootstrap=True, bootstrap_features=False, max_features=1.0,
                  max_samples=1.0, n_estimators=10, n_jobs=None,
                  oob_score=False, random_state=None, verbose=0,
                  warm_start=False)
```

These are the parameters of Bagging classifier.

```
AdaBoostClassifier(algorithm='SAMME.R',
                   base_estimator=DecisionTreeClassifier(ccp_alpha=0.0,
                                                         class_weight=None,
                                                         criterion='gini',
                                                         max_depth=None,
                                                         max_features=None,
                                                         max_leaf_nodes=None,
                                                         min_impurity_decrease=0.0,
                                                         min_impurity_split=None,
                                                         min_samples_leaf=1,
                                                         min_samples_split=2,
                                                         min_weight_fraction_leaf=0.0,
                                                         presort='deprecated',
                                                         random_state=None,
                                                         splitter='best'),
                   learning_rate=1.0, n_estimators=50, random_state=None)
```

These are the parameters of Adaboosting classifier.

Requirement:

To finish this project, we should use Python with these pakages and import them so that the program can work:

import numpy as np

import pandas

from sklearn.model_selection import train_test_split

import time

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score

from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.ensemble import BaggingClassifier

Results:

```
The predicted accuracy of the decision tree only: 0.9607817758219
Running time of decision tree only: 0.047096199999941746


The predicted accuracy of the k-NN only: 0.9630144965053068
Running time of decision tree with bagging: 5.011762899999667

The predicted accuracy of the decision tree with bagging 0.9616878074035724
Running time of decision tree with bagging: 0.4907569000006333

The predicted accuracy of the decision tree with Adaboosting 0.9664768314781258
Running time of decision tree with Adaboosting: 2.113553699999102
```

In order to compare the results above directly, let's make a table for these results:

Classifier	DT	K-NN	Bagging	Boosting
Accuracy	0.96078	0.96301	0.96169	0.96648
Running time	0.047	4.921	0.491	2.114

Comparison and discussion:

This project is successful because the accuracy is much higher than 50%. The accuracy is high enough because it tends to 1, that is, these classifiers almost label the solo player's statistics correctly.

From the results above, we can find it that the accuracy of decision tree is the least and its running time is also the least. Bagging classifier with decision tree improve the accuracy than just using decision tree, but its running time also increase a little. The most accurate classifier is the Boosting classifier and k-NN classifier wastes most time among these four classifiers.

If we need the quickest classification, we should use the decision tree classifier only. Boosting classifier seems to be quicklier than Bagging classifier but it waste more time. K-NN classifier wastes time and its accuracy is also higher than decision tree.

However, there is still some space for me to improve. I list many parameters but I just set little of them by myself. The major job of setting parameters was relied on the Python itself. If there is more time, I would change more parameters to see their differnences of accuracy and running time. I will also learn more knowledge of these classifiers and learn all the meanings of these parameters.

What's more, in this project I used only 4 classifiers. In fact, there are many kinds of classifiers which are not be used in this project yet. For example, the Artificial Neural Network classifier was not used, which is also one important kind of classifiers. In future, I would add more kinds of classifiers into this project.

From this project, I got better command of classifiers. The most valuable experience I got in this project is that I had a chance to apply it in something that is very closed to our real lives, though I seldom play LoL and I am not very familiar with it.