Report for Project 1

Author: 刘峻宏 from Intelligent Manufacturing Student ID: 201930133213

1. Introduction

(1) The project

League of Legends (LoL) is one of the most played eSports in the world at the moment. In this project, I will have access to about 3 Million match records of solo gamers as training set. Each record comprises of all publicly available game statistics of a match played by some gamer, including an important field called "winner". If "winner" is “1”, the team 1 won the match, or vice versa. My task is to create one or more classifiers that take as inputs from any fields from above (except “winner) such match record, and labels this record as a "1" or a "2". The test set comprises of ~2 Million of such records.

(2) The methodology

Firstly, import the training data file into the program and preprocess the data to generate the training data set.

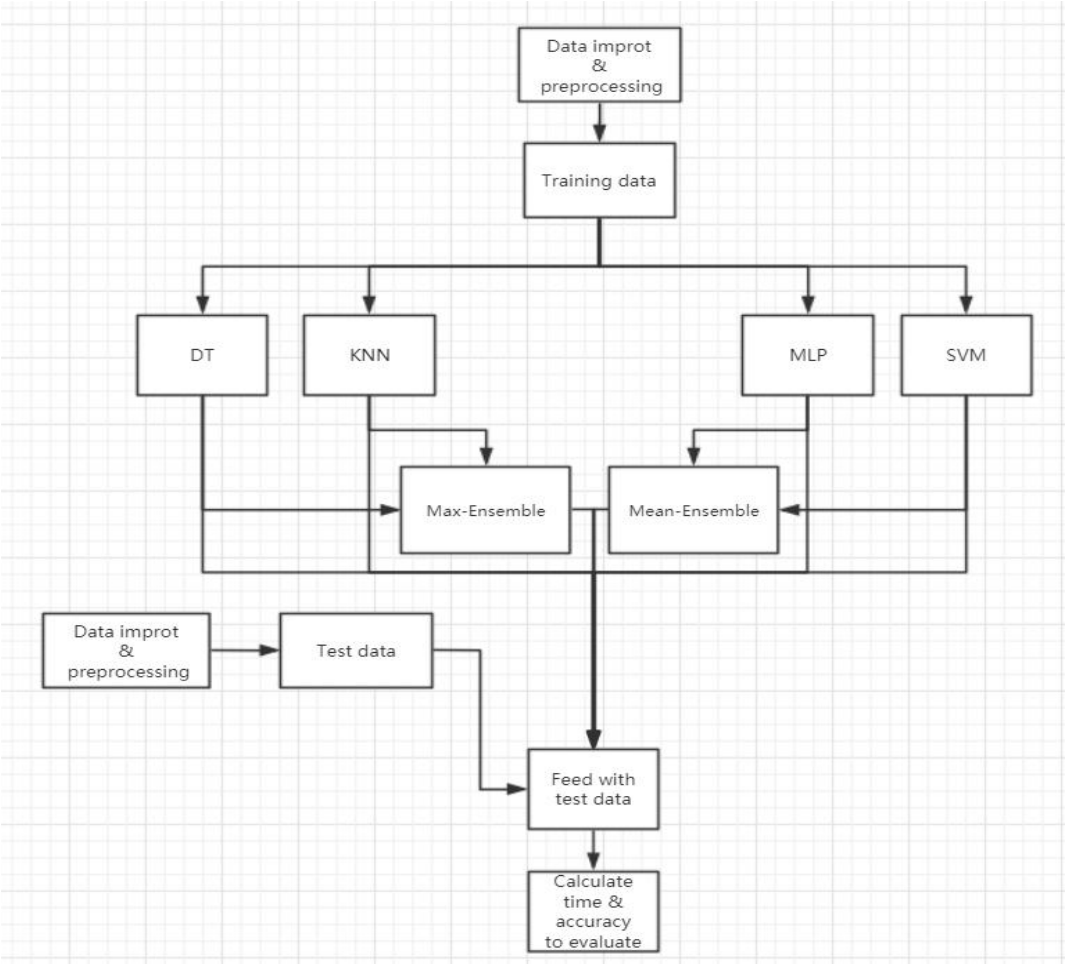
Secondly, build up several individual classifiers and ensemble classifiers, and feed them with training data set so that they learn from data automatically.

Thirdly, import the test data file and preprocess the data to generate the test data set.

Fourthly, use classifiers to predict results from test data set and compare the results with the actual values to judge the accuracy of each classifier.

Fifthly, time for each classifier and print the time and accuracy of each classifier on the screen.

The flow chart of methodology is shown below:



2. Algorithm

(1) Data preprocessing

Cut off obviously useless features and remove low-variance features.
In order to select the feature variables that contributed the most to the model output and from the 25 attributes, this step deletes the features whose variance is 0 (assume these features have no effect on the output label) or too low to save training time.

(2) Decision Tree(DT)

A decision tree divides the input space into different regions, each with independent parameters . Decision tree classification algorithm is an inductive learning method based on instances. It can extract tree models from given disordered training samples. Each non-leaf node in the tree records which feature was used to make the category determination, and each leaf node represents the final category determination. When testing a new sample, it only needs to start from the root node, test at each branch node, recursively enter the subtree along the corresponding branch, and the test until it reaches the leaf node. The category represented by the leaf node is the predicted category of the current test sample.

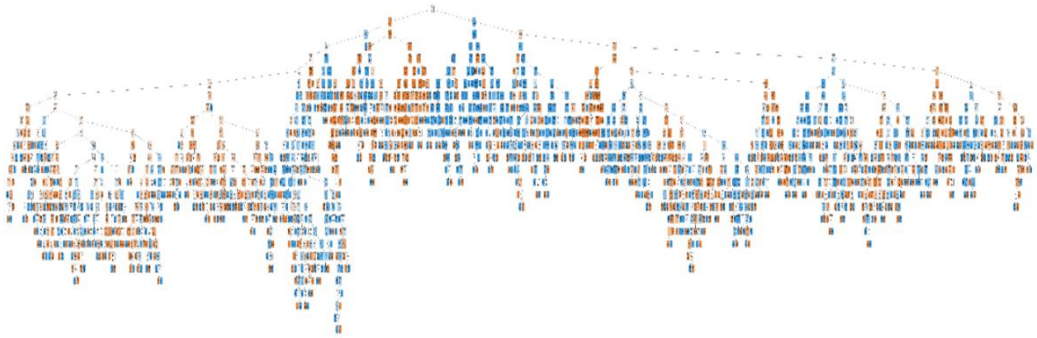
Function:

DecisionTreeClassifier(criterion, splitter, max_depth, min_samples_split)

Explanation for used parameters:

Name	Function
criterion	'gini' or 'entropy', 'gini' for CART algorithm
splitter	'best' or 'random', 'best' for small amount of samples, while 'random' for large scale data
max_depth	int or none, usually vary from 10 to 100
min_samples_split	int or float, as the minimum number of samples

The graph of decision tree of this project:



(3) Support Vector Machine(SVM)

To solve the separation hyperplane that can divide the training data set correctly and has the maximum geometric spacing. There are infinite such hyperplanes(perceptrons), but the separated hyperplanes with the greatest geometric spacing are unique.

Basic steps in linear SVM are following:

1) Input data set.

$$T = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$$

2) Select the punishing parameter C>0, construct and solve convex quadratic programming problems:

$$\min_{\alpha} \ \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\left(\boldsymbol{x_i}\cdot\boldsymbol{x_j}\right)-\sum_{i=1}^{N}\alpha_i$$

$$s.t. \quad \sum_{i=1}^{N}\alpha_i y_i = 0$$

$$0 \le \alpha_i \le C,\ i = 1, 2, \ldots, N$$

Then get the optimal solution:

$$\boldsymbol{\alpha}^* = \left(\alpha_1^*, \alpha_2^*, \ldots, \alpha_N^*\right)^T$$

3) Calculate the following formula

$$\boldsymbol{w}^* = \sum_{i=1}^{N}\alpha_i^* y_i \boldsymbol{x}_i$$

Select an appropriate solution between 0 and C, calculate:

$$b^* = y_j - \sum_{i=1}^{N}\alpha_i^* y_i\left(\boldsymbol{x_i}\cdot\boldsymbol{x_j}\right)$$

4) Find the separation hyperplane

$$\boldsymbol{w}^*\cdot\boldsymbol{x} + b^* = 0$$

Then we get the classification decision function f(x)

$$f\left(\boldsymbol{x}\right) = sign\left(\boldsymbol{w}^*\cdot\boldsymbol{x} + b^*\right)$$

For parameters in function SVC():

Name	Function
kernel	rbf, linear, poly, sigmoid or precomputed
gamma	default to be ‘auto’
probability	Probability estimation? Default to be false

(4) Muti-Layer Perceptron(MLP)

A muti-layer perceptron is a neural network composed of fully connected layers with at least one hidden layer, and the output of each hidden layer is transformed by activation function. The number of layers of the muti-layer perceptron and the number of hidden units in each hidden layer are all super parameters. Another possible function include rectified linear unit(ReLU) function:

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

Take single hidden layer as an example, the calculation and output are as following:

$$\boldsymbol{H} = \phi(\boldsymbol{X}\boldsymbol{W}_h + \boldsymbol{b}_h),$$
$$\boldsymbol{O} = \boldsymbol{H}\boldsymbol{W}_o + \boldsymbol{b}_o,$$

In which ϕ represents the activation function, w means the weight for each layer, b represents the bias, and O means the output.

For parameters in function MLPClassifier():

Name	Function
hidden_layer_sizes	The ith element represents the number of neurons in the ith hidden layer
alpha	Parameters of L2 penalizes(regularized item)
learning_rate	The speed of learning
max_iter	The maximum of iteration

(5) K-Nearest Neighbor(KNN)

In KNN, each sample can be represented by its nearest K neighbors. It is really simple to accomplish but difficult to calculate. KNN is more suitable than other methods for the sample sets with more overlapping or overlapping class domains to be divided.

For parameters in function KNeighborsClassifier():

Name	Function
n_neighbors	Detectable number of neighbors around

(6) Ensemble

For training data set, we train several individual learners, and through certain combination strategies, we can finally form a strong learner, so as to learn from the individuals. The fusion includes bagging, boosting and voting, and voting includes max-voting and mean-voting.

① Max-voting

The final result is determined by the subordination of the minority to the majority, which is also called hard-voting.

② Mean-voting

The average probability of all model prediction samples of a certain class is taken as the standard, and the corresponding type with the highest probability is the final prediction result, which is also called soft-voting.

For parameters in function VotingClassifier():

Name	Function
estimators	Include all individual classifiers with their parameters
voting	Soft or hard, determine the voting pattern

3. Requirements

(1) Scikit-Learn Package: import sklearn

(2) Timing: import time

(3) Import the data set: import pandas, from sklearn import datasets

(4) Different requirements for different classifiers

from sklearn import tree, from sklearn.svm import SVC

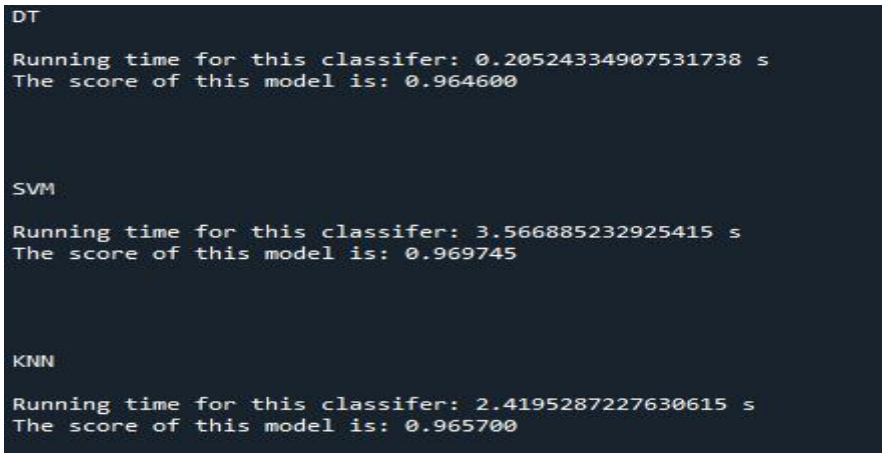
```
from sklearn.neighbors import KNeighborsClassifier,
from sklearn.neural_network import MLPClassifier,
from sklearn.ensemble import VotingClassifier
```

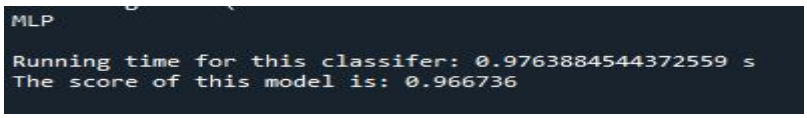
4. Results

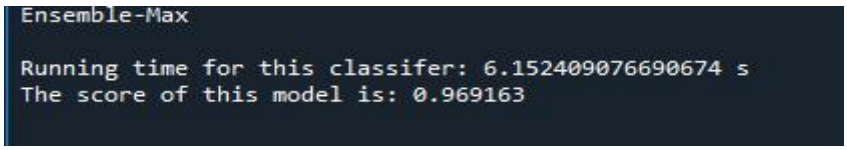
(1) Result Table

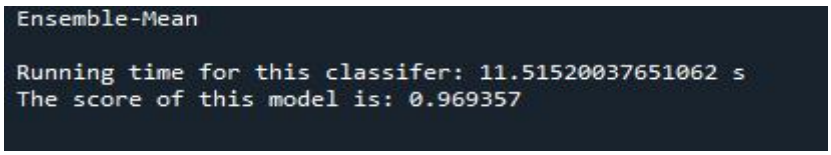
Classifier	Accuracy(Score)	Consuming Time
Decision Tree(DT)	0.964600	0.20524334907531738 s
Support Vector Machine(SVM)	0.969745	3.566885232925415 s
Muti_Layer Perceptron(MLP)	0.966736	0.9763884544372559 s
K-Nearest Neighbor(KNN)	0.965700	2.4195287227630615 s
Max-Ensemble	0.969163	6.152409076690674 s
Mean-Ensemble	0.969357	11.51520037651062 s

(2) Screenshot









5. Comparison and discussion

(1) What I learned:

1) How to preprocess your data

Actually, there are lots of useless features and data in raw data file. In this project, gameID, creation time and so on are not helpful to predict the final result, so we need to cut them off to generate the true data set. What's more, with data preprocessing functions in sklearn, we can evaluate the variance of each feature and eliminate those with low variance, which indicates they are not so important. Besides, we need to preprocess all the data, not only for the training set, but also for the test data set. Once I forgot to preprocess the test set, then the result is really terrible.

2) Learn different classifiers and how to program them

There are many classifiers in sklearn, and each classifier is suitable for different type of data, for different kind of problem. So what we need to do firstly is to understand how each classifier work and what are the advantages and disadvantages, which really helps a lot when choosing proper classifier. For example, KNN is more suitable than other methods for the sample sets with more overlapping or overlapping class domains to be divided.

Moreover, different types of ensemble provide a chance to generate all their advantages and offset the disadvantages. Bagging, boosting and voting are also suitable for different situations. After learning all of above, we start to program in practice. We have to figure out what are the necessary packages and which one is the correct function. At last, we must adjust all possible parameters once and once again to get higher accuracy in shorter time.

3) How to upload my working files including codes to Github

Once I transmitted all of my working files through WeChat or QQ, which are actually inconvenient on public computer. In this project, I get to know the Github. I love its pattern, convenience, compatibility. Moreover, Github even support collaboration in groups, which is definitely useful in future work.

4) How to visualize the output of my program

I was used to outputs with boring numbers and alphabets on the screen. However, now I learn how to plot graphs for output containing data distribution for each classifiers,

especially for decision tree. It is completely a new world for me, in which I can see my effort intuitively. In addition, I can promote my graph by adjusting some parameters.

(2) How to improve:

1) In data preprocessing , set a threshold for variance of features

In this project, we just cut off some useless features and eliminate features with “low” variance. However, here comes the question, how to define “low variance”? I think we need a exact value to be the threshold to judge. If the variance is higher than it, we will use it, if not, just ignore the corresponding feature. So how to get the threshold? I propose that we can design a for loop to iterate the threshold from 0 to 10, adding 0.1 each iteration, and evaluate the average accuracy and consuming time.

2) In model training, try more parameters with different values

Time limiting, we do not have enough time to try every possible parameter with every values or models. That means we may miss some better parameter combinations. So what we need to do is just to try every possible parameter combination for each classifier. For example, I am wondering what if I choose bagging or boosting in ensemble instead of voting.

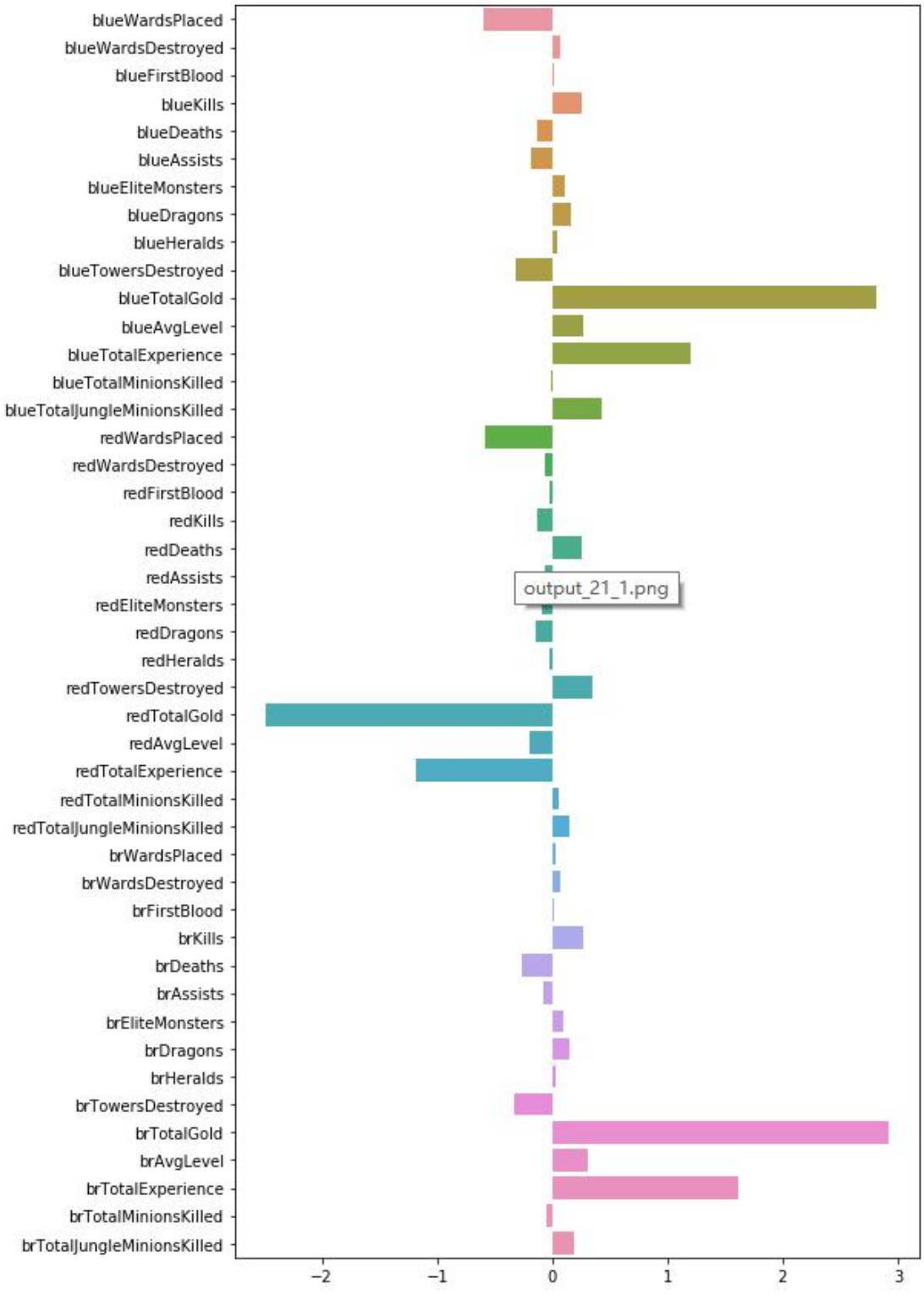
3) Using Cross-Validation

Sometimes, it is impossible to accurately estimate the effect of the obtained model on the invisible samples, and the single training test may be biased, so it is generally necessary to repeat the training test for several times to get the average value to achieve cross-validation. Cross-validation divides the whole data set into K parts, and takes one part as the test set each time, and the other K-1 part as the training set, and the average test result of K times as the accuracy of the final model. Sklearn also provides a very convenient interface cross_validate. After entering the defined model, all training data, and the number of fold K, we can get the results of each fold training test.

4) Visualize parameters in each model

In order to get better parameter combination, we hope to visualize the parameters. For example, in models with weighted features, the greater the absolute value of the weight of the corresponding feature, the greater the contribution to classification prediction.

So we can plot the graph as follows:


blueWardsPlaced
blueWardsDestroyed
blueFirstBlood
blueKills
blueDeaths
blueAssists
blueEliteMonsters
blueDragons
blueHeralds
blueTowersDestroyed
blueTotalGold
blueAvgLevel
blueTotalExperience
blueTotalMinionsKilled
blueTotalJungleMinionsKilled
redWardsPlaced
redWardsDestroyed
redFirstBlood
redKills
redDeaths
redAssists
redEliteMonsters
redDragons
redHeralds
redTowersDestroyed
redTotalGold
redAvgLevel
redTotalExperience
redTotalMinionsKilled
redTotalJungleMinionsKilled
brWardsPlaced
brWardsDestroyed
brFirstBlood
brKills
brDeaths
brAssists
brEliteMonsters
brDragons
brHeralds
brTowersDestroyed
brTotalGold
brAvgLevel
brTotalExperience
brTotalMinionsKilled
brTotalJungleMinionsKilled
output_21_1.png
−2
−1
0
1
2
3