Project1 Report

Introduction:

After learning the basic knowledge of big data and classification, we have an strong demand to apply these theorems into practice. In order to inspire our excitement, our teacher assign a project to us, which can make full use of the classification methods we have learnt in this course.

The purpose of this project is to train some classifiers by the given dataset, and then use them to successfully evaluate the given test dataset. The dataset is formed by the game records of League of Legends(LOL), which contains some basic information of the game, public game statistics and the winner of the game. What we need to do is to select some classification methods and extract some features from the dataset, then build the classify model. The model will predict the result of a specific game, and also compare the effect of different classifiers. This model also has some practical functionalities, such as evaluate the capability of different players and match the players of similar level into a game. The better matching strategy can let the players have more sense of participation and increase the enjoyment of the game.

My program is based on Python. I totally apply 3 different classifiers(DT, MLP, KNN) with some suitable parameters. After the labels are output by these classifiers, I use the ensemble learning method to combine the output of them and apply an Artificial Neutron Network(ANN) to perform the weighted voting process. The output of the last node of ANN will be the ultimate predicted label of my program.

Below are more details about my project.

Algorithms:

1. Load the training dataset and preprocess the data:

To begin the program, load the training dataset form new_data.csv. We can use the function in pandas to read the csv format. Then we will attain a table, in which every row stands for a sample and every column stands for an attribute. First extract the column of winner as the standard labels for training. Then extract some effective("effective" means these attributes will contribute to the result of the game) attributes from the remain columns as the features for training.

2. Build the Decision Tree model(DT):

The first selected model is Decision Tree. This model can be easily fit with the dataset, it has some nodes within a specific depth and each node can split the samples according to the change of entropy. The training will stop when most of the leaves contain only a kind of sample.

When I build DT model, the parameters are shown below:

criterion='gini': use Gini index as the split standard

splitter='best': choose the feature that has the highest gain

max_depth=5: the maximum depth of the tree is 5

min_samples_split=2: each node at least splits to 2 son nodes

The max depth is set to be 5, rather than None, because this can avoid overfitting and generate the more reasonable decision tree. We can use the trained model to calculate the accuracy of the model by input the training dataset again.

After the DT model are built, then visualize the tree. Display the tree figure on the page and save it in PNG format.

3. Find the optimal depth for the DT model:

Split training dataset into training set and validation set, training set for training and validation set for testing the accuracy of the model.

After we change the depth of the tree and training the model for some times, we can find that when the depth is increasing, the accuracy for training set will increase, but the accuracy for validation set may not always increase.

In order to find the optimal depth, the program will iteratively train the model by applying different depth, then calculate the accuracy by testing the validation set. Draw the figure of the accuracy with different depth, and find the corresponding depth for the highest accuracy in the figure.

At last, build the optimal DT with optimal depth and evaluate its accuracy. Also record the time cost for model training.

4. Build the Multi-layer Perceptron model(MLP):

This model is based on ANN. It has a deep architecture which can represent more complicated function than a shallow one. It is designed to performed a classification like human's brain. The disadvantages of it are too complicate and hard to train. The parameters are shown below:

hidden_layer_sizes=(100,): the number of neurons in the hidden layer

alpha=0.0001: L2 penalty (regularization term) parameter

learning_rate='constant': the learning rate will remain constant

learning_rate_init=0.01: initial learning rate for training

max_iter=200: maximum number of training iterations

5. Build the K-Nearest Neighbor model(KNN):

In this model, each sample will transform to a point in the space. The output label for a sample is classified by a majority vote of its k nearest neighbors(k training samples with shortest distance). The parameters are shown below:

n_neighbors=5: the number of nearest neighbors

weights='uniform': the vote weights of each neighbor are equivalent

algorithm='auto': the classify process will adjust automatically

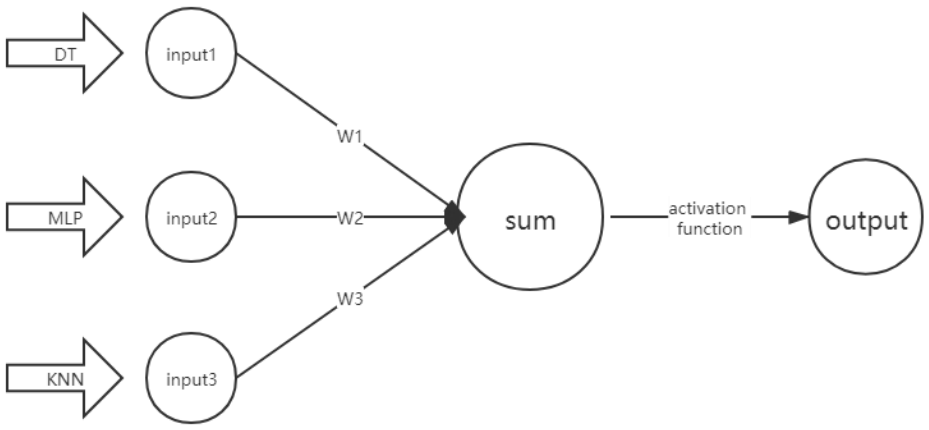
6. Ensemble the models and train the optimal voting weight:

After 3 models are built, the last step is to combine them and attain an ensemble model. I choose the voting-based ensemble method.

In the ensemble, the output(X1, X2, X3) of each classifier will have a weight when voting. The default weights(W1, W2, W3) are all 1, and the weights will be updated iteratively. The ensemble process will be performed by an artificial neural network(ANN), which has 3 layers. The input layer is the label from each classifier. To simplify the model, I change all labels 2 to -1, which can help to eliminate the bias in ANN. The second layer is the sum of the product of each input and its weight. the third layer is the output label, which apply an activation function:

$$\mathrm{f(sum)} = \begin{cases} 1 & (net > 0) \\ 2 & (net \leq 0) \end{cases}$$

The structure of the ensemble ANN is shown as the figure below:



To train the optimal weights for the ANN, update the weights by the formula below(α is the learning rate):

$$W_i(k+1) = W_i(k) + \alpha[Y_i - f\left(\sum W_i \cdot X_i\right)]X_i$$

At last, use optimal weights to build the ensemble model and predict the labels of the test dataset(test_set.csv).

Requirements:

I have loaded and used several packages in my program. Below is a brief view of these python packages:

pandas: to read the csv format file

numpy: to perform the operation of matrix

time: to record the current time

sklearn: all the prototypes of the classifiers model are from this package, it is very useful in Machine Learning.

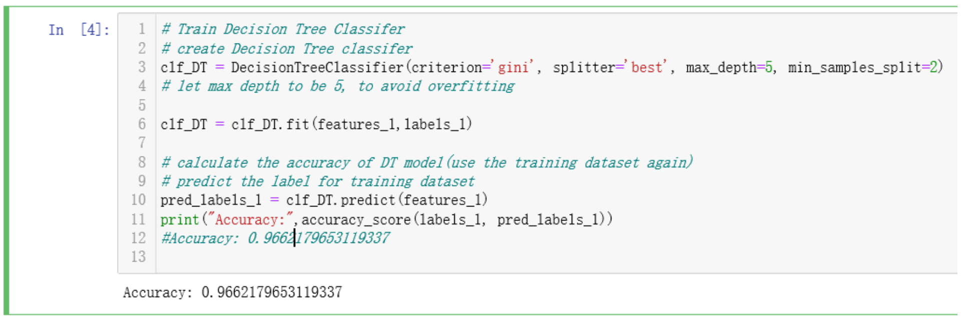
export_graphviz: to visualize the DT model

matplotlib: to plot the relation figure, visualize the statistic

Experimental Results:

1. The implement and visualization of the first Decision Tree:

Run the program about the first DT model, the test accuracy of the model and the figure of the tree will be printed on the screen:

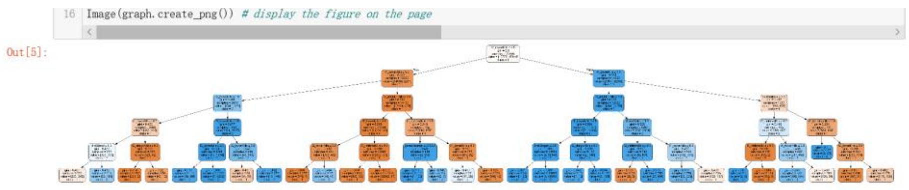


```
In  [4]:
# Train Decision Tree Classifer
# create Decision Tree classifer
clf_DT = DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=5, min_samples_split=2)
# let max depth to be 5, to avoid overfitting

clf_DT = clf_DT.fit(features_1,labels_1)

# calculate the accuracy of DT model(use the training dataset again)
# predict the label for training dataset
pred_labels_1 = clf_DT.predict(features_1)
print("Accuracy:",accuracy_score(labels_1, pred_labels_1))
#Accuracy: 0.9662179653119337

Accuracy: 0.9662179653119337
```

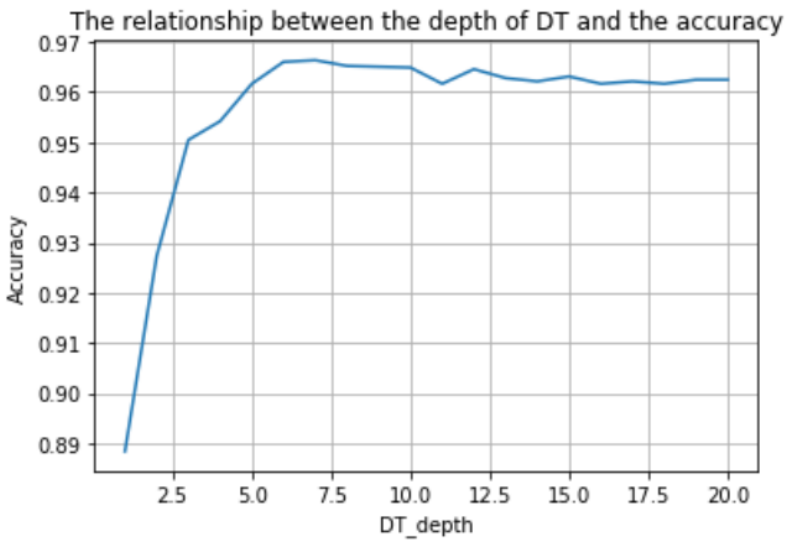


```
Image(graph.create_png()) # display the figure on the page
Out[5]:
```

The DT figure is not very clear on the screen because the tree has many nodes. In order to have a more detailed view of the tree, I will save the figure as PNG format. This picture will be attached with my code.

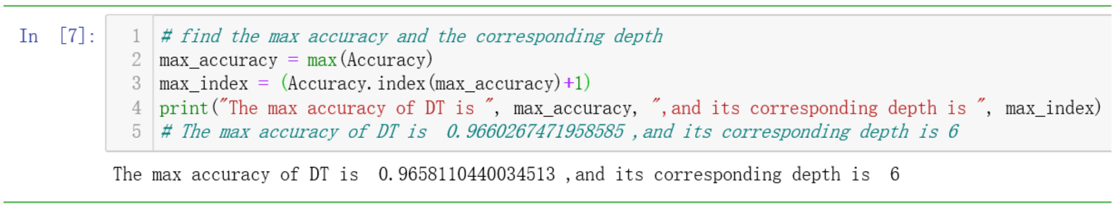
2. Optimize the depth of the DT model:

After iteratively train the model by applying the depth from 1 to 20, we can calculate each accuracy by testing the validation set. Draw the figure of these accuracy, and find the corresponding depth for the highest accuracy in the figure. The figure will be printed on the screen:



As we can see, with the increment of DT depth, the accuracy of DT model will first increase and then decrease sightly. This phenomenon is called as overfitting, which will increase complexity the generalization error of the model. In order to avoid overfitting, we should stop the training algorithm before it becomes a fully-grown tree.

Next, find the highest test accuracy and the corresponding depth:

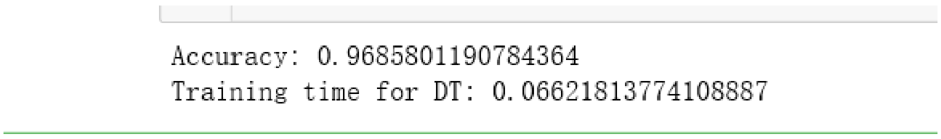


```
In [7]:
# find the max accuracy and the corresponding depth
max_accuracy = max(Accuracy)
max_index = (Accuracy.index(max_accuracy)+1)
print("The max accuracy of DT is ", max_accuracy, ",and its corresponding depth is ", max_index)
# The max accuracy of DT is  0.9660267471958585 ,and its corresponding depth is 6
```

```
The max accuracy of DT is  0.9658110440034513 ,and its corresponding depth is  6
```

Then we can build the corresponding DT tree which is supposed to have the highest accuracy when classify the test dataset. This will be the first classifier in the ensemble process.

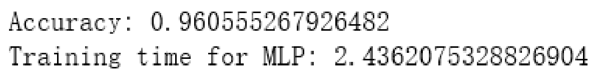
Estimate its accuracy and record training time:



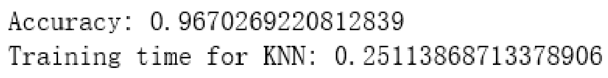
```
Accuracy:  0.9685801190784364
Training time for DT: 0.06621813774108887
```

3. The implement process of MLP and K-NN model:

For MLP and K-NN model, also train the classifiers with the training set and test them with validation set. The result is shown below:



```
Accuracy:  0.960555267926482
Training time for MLP: 2.4362075328826904
```



```
Accuracy:  0.9670269220812839
Training time for KNN: 0.25113868713378906
```

The accuracy and the training of each model are recorded in the table below, we can compare the effect of them.

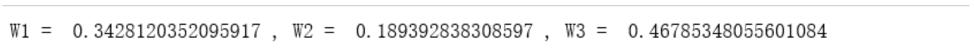
Model	DT	MLP	KNN
Accuracy	0.96858011907843	0.95013590473725	0.96702692208128
Training time	0.06252050399780	9.73375201225280	0.216989278793334

We can conclude that Decision Tree is the most accurate and efficient classifier within the three models.

4. The implement of ensemble and the weights optimization:

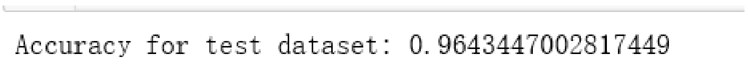
Now three models(DT,MLP,KNN) for classification are built, they all have an accuracy higher than 0.9. Each classifier has its own advantages and disadvantages. The last step is to combine them and attain an ensemble model which can minimize the error.

For weighted voting, I build an ANN and set the original weights as 1. Then iterate for 10000 times to update the weights by the first 10000 samples in training dataset. The updated weights are shown below:



```
W1 =  0.3428120352095917 ,  W2 =  0.189392838308597 ,  W3 =  0.46785348055601084
```

Then use the updated weights to build the final ANN and perform the classification for the test dataset(test_set.csv). Calculate the ultimate accuracy for test dataset.



```
Accuracy for test dataset: 0.9643447002817449
```

Comparison and Discussion:

From this experience, I attained many techniques and benefitted a lot. I learnt how to put the basic theory into practice. I learnt how to select the suitable classifiers for a specific task. I learnt how to ensemble the output of different classifiers to minimize the error. I also acknowledged the principle that study is a life-long and indefinite process, which means the world is so large that only by keep learning can we follow the rapid development of the time.

There are some innovative points in my classification program:

1. I apply a mathematic method to find the optimal depth of my DT model. The figure is able to illuminate the relation between the accuracy and the depth distinctly. It not only proves the process of overfitting, but also tells the optimal depth in a clear way.
2. When implement the weighted voting process, I apply the ANN method. After iteratively update the weights of each classifier, the ANN will output a relatively reasonable label for the test sample.

It is impossible to be perfect. There are also some shortcomings in my program that remain to be improve if the time is allowed:

1. In the adjustment of the parameters of DT model, I only optimize the depth of the tree to minimize the error. Although this parameter is very important, but is not enough. There are some other parameters remain to be optimized, such as the split criterion and split number. They can also contribute to the accuracy of the DT model.
2. When build the MLP and K-NN model, I only apply the default parameters provided

by sklearn classifiers, without the details of how to choose these parameters. If add more process about the parameters optimalization into my program, the accuracy of these two classifiers will increase, which leads to the increase of the accuracy of the ensemble classifier.

3. I only display the training time of each model in the table, but not use the training time to evaluate the efficiency of a classifier.

4. The structure of the ANN in ensemble is too simple, it has no bias and the hide layer is formed by only one layer. A little more complex ANN will be more reasonable for the ensemble process.

Summary:

In this project, I successfully trained an ensemble classifier which consists of 3 different classifier models, DT, MLP and KNN. The classifier can predict the result(winner) of a LOL game by inputting the basic attributes of the game. This model can be very useful for the game operators.

I practice my personal capability and enrich my experience in this process. And I have a deeper understanding about the practical application of big data and machine learning. This capability will benefit me a lot in the future study and work.