Classifier for predicting the winner of LOL

一、Introduction

League of legend(LOL) is popular at home and abroad in these years rely on its cool special effects, fierce confrontation and various character settings. In all this points, fierce confrontation is always the core of the fascinating games which is also the point chasing by the young men to release them from packages of pressure and work. Fierce confrontation always build on a match of a Roland for an Oliver, which has a highly recommend on the algorithm about the match of teammates and choice of the opponents. In a single game, There are many factors that affect the final result of this game. The team which take priority status in the aspect of pushing towers, fighting creeps, killing the dragon will have a higher probability of winning. In this project, we have many data about the key point in a game that have impact on the final result such as first Baron, dragon, tower, blood, inhibitor and the final winner. In all of this data, 1 represent the first team, 2 represent the second team, 0 is used to represent nether the first team nor the second team win in this feature. By using this features and label, we can train some classifiers to predict the winners of other games. I use the method of artificial neutral network, decision tree support vector machines, Multilayer Perceptron and K-Nearest Neighbor to accomplish the task.

二、Algorithms.

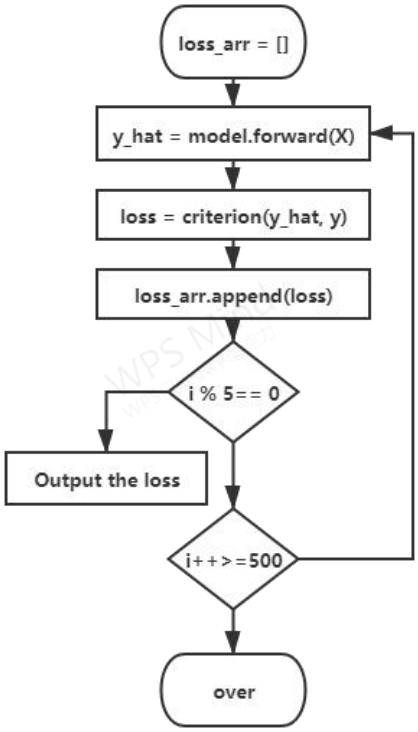
1. Data processing

I first build the model of artificial neutral network to using the data in the file new_data.csv. The package pandas is used to read the data from the file at the beginning, then output the first five rows using the function head to see if the data is read correctly. After analyzing the data in the file, I found that some of the features like 'gameId' , 'creationTime' and 'seasonId' are useless to our fit of classifiers. I remove these columns to reduce the computing load and improve the computing velocity of the computer. To training the classifier, we need to preprocess the data being read from the file. The data need to be separate into two parts features and label.

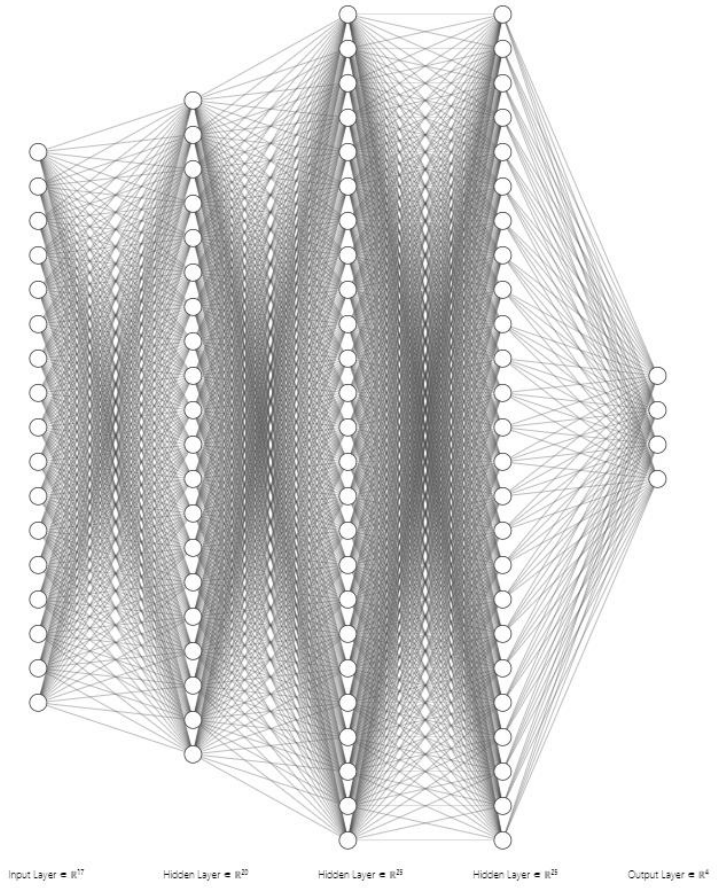
Features include all the feature columns such as game duration, first baron, dragon and tower except the final winner. The label include the winner column which represent by number 0, 1 or 2. In order to highlight the characteristics of the data and facilitate classification, I use the method of MinMaxScaler to preprocess the data. During the pretreatment, the program will map data between 0 and 1 to scale the interval of the data. By using this method, some of the features of this data will be expend to improve the accuracy of this model. Change the type of the data represent features into float type tensor and change the type of the data represent label into long type tensor to facilitate data processing and transmission.

2. ANN model

After preprocessing the data, we now need to create the artificial neutral network model and train this model. The model is built by a class named ANN in python. In the ANN class, I define two functions called _init_ and forward. In the function _init_ we define a four layers' artificial neutral network which include the first second third convolutional layer and the output layer. Input seventeen features into the first convolutional layer and then the program will output twenty features which will be input into the second convolutional layer. Then twenty five features will be inputted into the third convolutional layer. The final output of the four layers' artificial neutral network is four features. In the forward function, I designed the data calculation method. After the data being input into the convolutional layer, use the sigmoid function in all the three convolutional layers to calculate the data and then input them into output layer. In the output layer, use the softmax function to calculate. In the end of this function, return the value outputted from the output layer. The model of four layers' artificial neutral network is too weak to well fit the training model. We need to optimize the model. I use the Cross entropy loss function CrossEntropyLoss to calculate the loss in the train process. What's more, I use the optimization function optim in the package torch to optimize the algorithms. I designed a loop with length 500 which will output the loss message every two cycles. After all these process is down, we will get a well trained artificial neutral network mode.

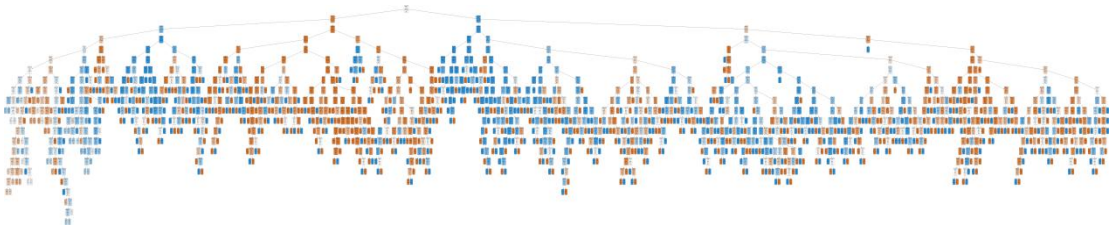


We need to prerpocess the testing data in the same method as we mentioned before. Inputting the test data into the trained model, we can get the predicted winners. Compare the predicted winners with the real winners using the function accuracy, we will get the final score of this model. After some times' test, I fount that when the length of the training data is 500 the final accuracy will roughly locate between 96 and 97 percent. Before training the model, I use the 'now' function in datetime library to record the current time. After the accuracy of this model has been calculated, do the same things to got the current time. The difference between the two times is the time spent training the model.



3. DT model

The second model I built is the decision tree model. For we have preprocessed the data in the process of training the artificial neutral network model, we now can build the decision tree model easily by calling the trainer and input the data that has been sorted to train the model. The final step is still using the accuracy function to get the final score. After some time's test, I found that the final score of the decision tree model will locate between 99 and 100 percent. Then I use the pydotplus library to built the decision tree figure. After outputting the photograph, I got the following picture. The decision tree built due to the huge amount of data is extremely complex.



When debugging parameters, I set the criterion parameter be gini which will use the

gini coefficient to judge the pros and cons of data division. In addition, comparing with the method of entropy, gini coefficient can greatly improve computing efficiency. The splitter parameter was set 'best' here to find the best cut point among all the features. When it comes to min_samples_split, I set the integer 4 to reduce the computational complexity of the model. I set the min_samples_leaf to be 4 to meet the same effect as min_samples_split.

4. SVM model

The third model is support vector machines. Support vector machine is a two-classification model. Its basic model is a linear classifier with the largest interval defined in the feature space. The largest interval makes it different from the perceptron; SVM also includes kernel techniques, which makes it essential The nonlinear classifier. The learning algorithm of SVM is the optimal algorithm for solving convex quadratic programming. We call the svm algorithm in the sklearn library to build a support vector machine model and then train the model using the data which has been preprocessed before. As for the parameter setting, I set the kernel parameter as linear which means linear kernel. The parameter gamma is one of the parameter of kernel function. When thing comes to the probability parameter, I choose 'True' in the two values.

5. KNN model

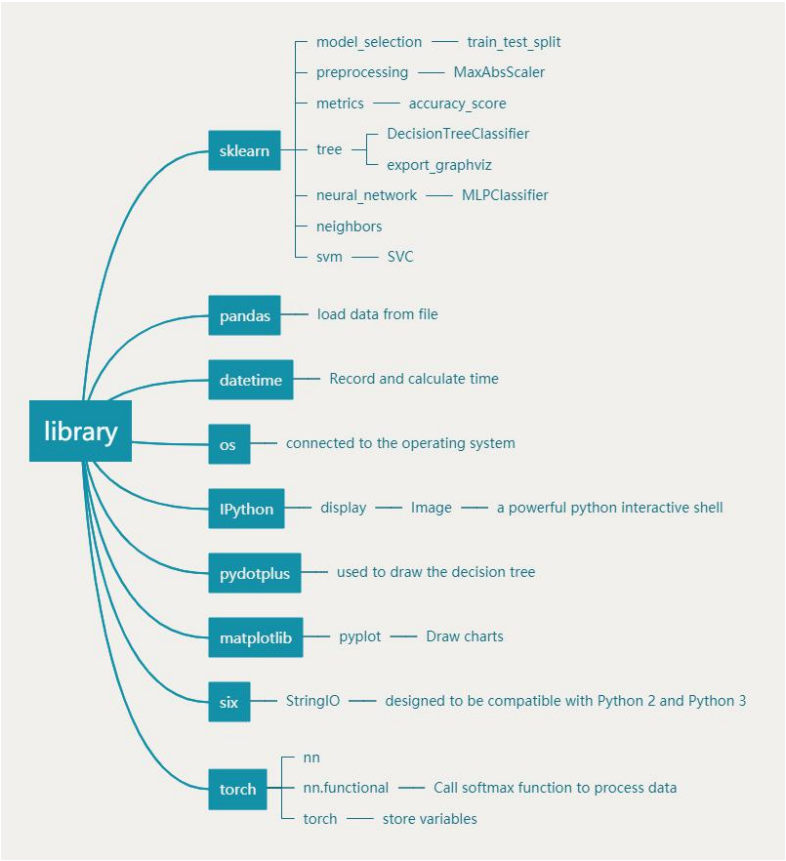
The fourth model is K-Nearest Neighbor. K-Nearest Neighbor is one of the simplest machine learning algorithms, which can be used for classification and regression, and is a supervised learning algorithm. The idea is that if most of the K most similar (ie the nearest neighbors in the feature space) samples of a sample in the feature space belong to a certain category, the sample also belongs to this category. That is to say, this method only determines the category of the sample to be classified based on the category of the nearest one or several samples in the classification decision. As we all know the most important step to fit a K-Nearest Neighbor model is to choose the most suitable value of K. After sever times' attempts from 1 to 20, I got the most suitable value 4 of K. Another parameter we need to concern about is the weights of neighbors. I set it to be uniform which means all these K neighbors would have the same weight.

I set the algorithm to be ‘brute’ which will use a brute-force search to accomplish the task of search. Manhattan distance and euclidean distance is two of the method of expressing distance. I set the value of P be 1 to use the method of Manhattan distance to calculate the distance between two points.

6. MLP model

The last model is Multilayer Perceptron. Multilayer Perceptron is an artificial neural network with a forward structure that maps a set of input vectors to a set of output vectors. MLP can be regarded as a directed graph, composed of multiple node layers, and each layer is fully connected to the next layer. Multilayer perceptrons follow the principles of the human nervous system to learn and make data predictions. The parameter hidden_layer_sizes is used to determine the size of the hidden layer. Too many hidden layers will cause a serious reduction in computational efficiency. I set it to be (5 , 5) here. The optimization of weights will improve the accuracy of the prediction of the MLP algorithm. The parameter slover was set to be lbfgs here to optimizer optimization algorithm using quasi-Newton method family. In order to avoid a large number of weak signals from interfering with the model, we set the activation parameter to relu and use the relu function as the model's activation function.

三、Requirements



I have to thank various powerful feature packs for helping me greatly simplify the code needed to complete the project. In the process of creating the artificial neutral network model, we use torch type array to store variables. The function used to linearly transform the data in artificial neural network also comes from torch package. I then use the optimization plan' optim. Adam' to optimize the ANN model. These are all the credit of the torch package. We use the package pandas to load the data from the 'csv' file and do some initial data processing. The sklearn package is the most powerful package used here. The Sklearn library contains a large number of machine learning algorithm implementations. It provides a complete machine learning toolbox that supports powerful machine learning libraries such as preprocessing, regression, classification, clustering, dimensionality reduction, prediction and model analysis, and nearly half of machine learning and data science projects use this package. We use this package here to preprocess the data and training the classifiers' model such as artificial neutral network and decision tree. The accuracy function used to evaluate the

performance of the trainer is also one of the function of sklearn. The six library is a library specifically designed to be compatible with Python 2 and Python 3. The IPython library is a powerful python interactive shell and it is also the best platform for scientific computing and interactive visualization using Python. The pydotplus library is a powerful drawing tools which is used to draw the decision tree. The os library is a window connected to the operating system. The code here calls the os library to add and modify the path. Data processing and fitting time is one of the important indicators for judging the pros and cons of a classifier. Some classifiers with higher scores may be discarded because they spend too much time. To calculate the time spent by building a classifiers, we import the library datetime which is used to show and calculate the time.

四、Results

	时间/s	得分
ANN	33.158	0.967
DT	0.085	0.962
SVM	14.577	0.960
KNN	11.308	0.961
MLP	2.461	0.966

In all this five classifiers, the decision tree model use the shortest time but do not get the highest accuracy. This classifier must the best choice for the one pay more

attention to the speed of time, and do not have high requirements on the accuracy of the model. Apart from the DT model, the KNN model get the second lowest accuracy but use a little longer time than other classifiers. The other three classifiers have a very close final accuracy but the training time have a big difference. The average training time of the ANN model is 33 seconds, which is much higher than the DT model's less than one second. I ranked these models through my own evaluation method.

Rank	Classifier	时间/s	得分
1	MLP	2.461	0.966
2	DT	0.085	0.962
3	ANN	33.158	0.967
4	KNN	11.308	0.961
5	SVM	14.577	0.960

五、Comparison and discussion.

1.Problem analysis and discussion

When I first build the ANN model, I neither preprocessed the data nor modified the default parameters of the function. The final result was not satisfactory, which hovered between 49 and 50 percent. Through some thinking and analysis of the training data, I think the reason may be that the difference in data characteristics is not obvious enough. Then I tried some methods of preprocessing the data such as regularization, standardization, MinMaxScaler, Normalization and Binarization. I finally got the best score located between 96 percent and 97 percent by using the method of MinMaxScaler. But when the same things happened to the decision tree model, I was surprised to find that the method of data preprocessing did not effectively improve the final score of the model, but slightly decreased, from 96% before to 95% now. This phenomenon may be due to different data processing modes suitable for different types of data.

When designing the ANN model, I choose to set a two layer classifier at the first time. The training results of the model are extremely satisfactory, and the score is always

stable above 96 points. Then I try to increase the number of layers of the artificial neural network to try to get better results. Then I found that when improving the layers of the model, the final score did not improve much. The contribution value to the data improvement is all less than one percent. But at the same time, the model training time has increased. From a dozen seconds for a two-layer network to twenty seconds for a three-layer network to thirty seconds for a four-layer network today. However, even the most complex four-layer network does not take a long time. If there is no high requirement for time, but you want higher accuracy, then a four-layer or higher network may be more suitable. I had another attempt when training the ANN model. I try to use the softmax to replace the sigmoid function to process data. Surprisingly, the final accuracy decrease significantly until fifty percent. This change emphasizes the role of choosing a suitable data processor, which will greatly improve the accuracy of the model.

2. Future vision

If more time is given, I will try to ensemble this classifiers using the method of bagging and boosting. There are also many methods of ensemble I want to have a try to further improve the model performance such as majority vote, simple majority, weighted majority vote and simple mean which will get the final answer according to the results predicted by all classifiers. I also want to try the method of logistic regression algorithm which is very suitable for handling binary classification problems.

六、Summary

After seeing the task of this project, I choose some classifiers which I am good at using to try to accomplish this task. By learning the teacher’s sample code, I learned about the usage methods and techniques of some libraries, and then built a model based on the data set to be trained and my own ideas. Grasp the characteristics of the data through data visualization and then preprocess the data. Input the processed data into the classifier for training to get the final classifier. Through this project, I learned

how to establish a suitable classifier to classify data and the related skills of preprocessing. Learned how to use JupyterLab for data visualization. This project not only helped me improve my programming ability, but also improved my ability to write reports.