Python-based LOL data classification model

Author: Yu Yan

1.Introduction

With the advent of the big data era, the technology of data mining continues to evolve. This project is to make classification of unlabeled LOL match records with “Win/Loss”, whose objective is mastering the methods of processing, classifying, predicting the data set and using them in the code. There are three main requirements for this project. Firstly, python program must be used. Secondly, at least one classification method is implemented. Finally, the evaluation result should be higher than 50%.

2. Algorithms

First classifier is decision tree. Decision tree uses a top-down recursive method, whose basic idea is to construct a tree with the fastest decrease in entropy as a measure of information entropy. The classifier of the decision tree has been built for us in the sklearn library. There are more than ten parameters in the function of “DecisionTreeClassifier()”. However, the most commonly used parameters are “criterion” and “max_depth”. “gini” and “entropy” can be chosen for “criterion”. “max_depth” is to set the max depth for the decision tree. In “test.py” file, using the for loop combined with the plot function, draw the figure to display the trend of accuracy with the maximum depth. Next, narrow the scope and continue to use the for loop to find the best depth. I visualized the decision tree model and stored it as “out.png” shown in Figure 1. Since the decision tree is too large, part of the figure is shown in figure 2.

Second classifier is artificial neural networks (ANN). Figure 3 is about the model structure diagram of the ANN algorithm. In this project, the artificial neural network with three layers is built. In the first layer, there are twenty input nodes which stand for twenty features. The learning process consists of forward propagation and back propagation. Forward propagation is used to calculate the forward network, that is, to obtain the output result of a certain input information after the network calculation. During forward propagation, the activation function plays a key role. “relu”, “sigmoid”, “tanh”, “softplus” and “softmax” are common activation functions. They all have their own advantages and disadvantages. Users need to choose these functions according to their specific circumstances. Due to more than twenty thousand data need to be trained, “tanh” function is considered which is benefit to gradient solution compared with “sigmoid” function. In addition, “softmax” is also chosen because of its convenience for subsequent gradient derivation. Backpropagation is used to transfer errors layer by layer and modify the connection weights between neurons so that the output obtained after the network is calculated on the input information can meet the expected error requirements. During backpropagation, cross entropy loss was used to compute loss and optimizer was used to optimize parameters. In “test.py” file, through a simple for loop, the respectively best learning rate is found, which equals to 0.01.

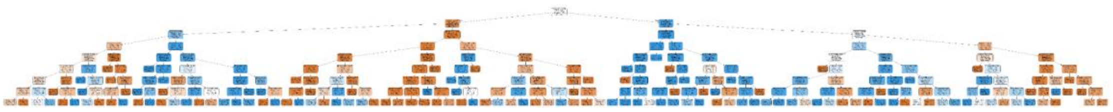


Figure 1

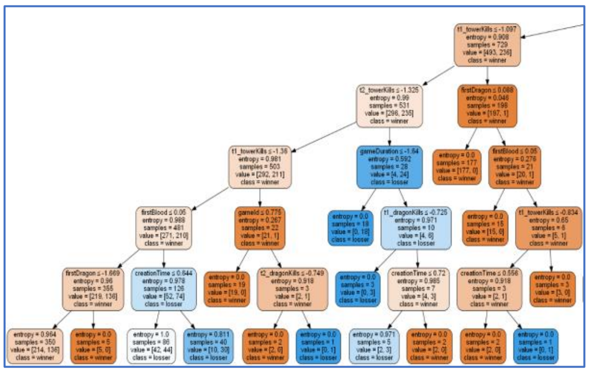


Figure 2

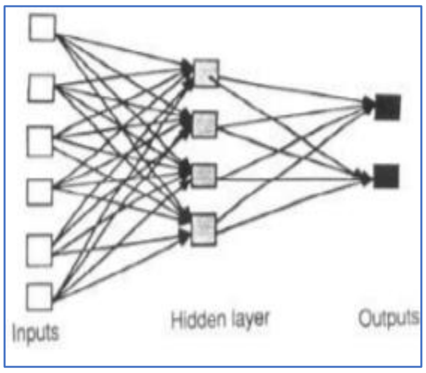


Figure 3

The third classifier is K-NearestNeighbor(KNN). The idea of KNN is that data can be classified by measuring the distance between different feature values (Figure 4). The algorithm of KNN can be described as five steps. (1) Calculate the distance between the test data and training data. (2) Sort them according to the increasing relationship of distance. (3) Select K points with the smallest distance. (4) Determine the frequency of occurance of the category of the first K points. (5) Return the category with the highest frequency among the first K points as the predicted category of the test data. The function of KNeighborsClassifier() also consists of many parameters. “n_neighbors” is the number of neighbors to use by default for kneighbors queries. “weights” can be chosen from ‘uniform’ which represents that all points in each neighborhood are weighted equally, “distance” which denotes weight points by the inverse of their distance and ‘callable’ which can be customize. “algorithm” can be chosen from ‘auto’, ‘ball_tree’, ‘kd_tree’ and ‘brute’. “leaf_size” means the size of leaf passed to BallTree or KD. This can affect the speed of the construction and query, as well as the memory required to store the tree. The optimal value depends on the nature of the problem. In “test.py”, according to the grid search, the best parameters within a certain range are found. GridSearchCV() is a function in sklearn, used to automatically debug parameters. There are lots of parameters in this function, ‘estimator’ which indicates what classifier is selected and pass in other parameters besides the parameters that need to be determined, ‘param_grid’ for the value of the parameter that needs to be optimized, ‘cv’ which is the cross-validated parameters.

The last classifier is bagging which combines several models to reduce generalization errors. The following picture shows an example of an ensemble (Figure 5). In the picture, an input array X is fed through two preprocessing pipelines and then to a set of base learners f_i. The ensemble combines all base learner predictions into a final prediction array P. There are several parameters in the function of “BaggingClassifier()”. Among them, the most basic several parameters are “base_estimator”,” bootstrap”, “n_estimators” and “max_samples”. “base_estimator” is the base classifier chosen. “max_samples” is the sample number drawn from the feature set to train each base classifier. “n_estimators” is the number of base classifiers.

Most of parameters in these models play a crucial function in preventing the model from overfitting or underfitting and reducing the time complexity of training the model. More details on the ways to find the relatively best parameters will be discussed in the last part.

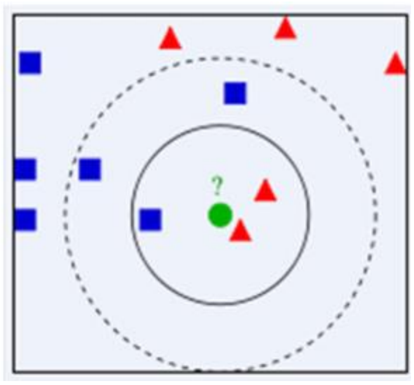


Figure 4

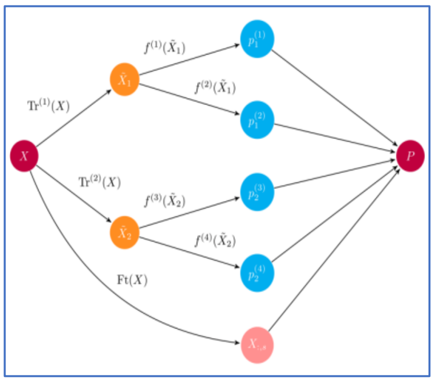
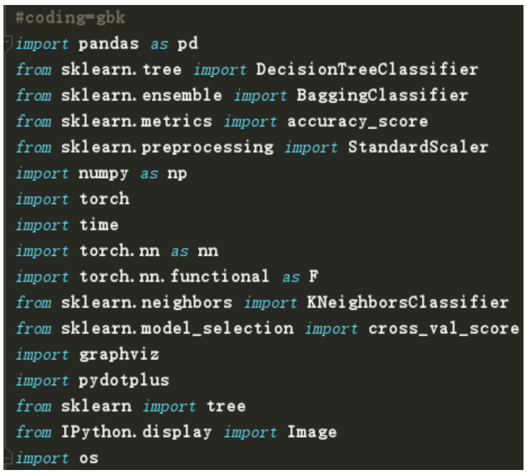


Figure 5

3.Requirements

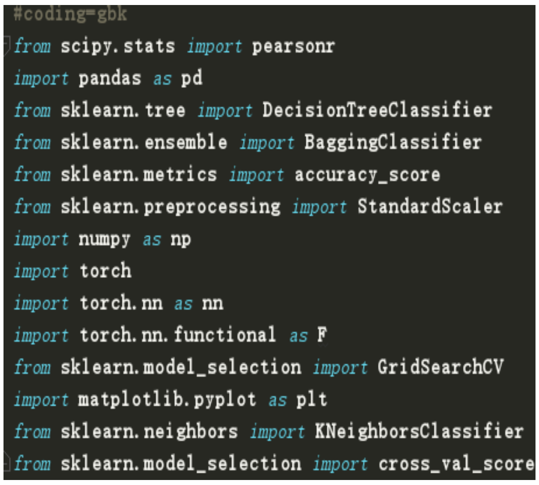
In this project, all of packages used are pandas, sklearn, numpy, time, matplotlib, scipy, torch, graphviz and pydotplus. Furthermore, there are some important modules from them. The whole modules in “bigdata_proj1.py” are shown in Figure 6. The package of “sklearn” integrates some commonly used machine learning methods. In this project, I mainly used the five modules and six functions from them. They are “DecisionTreeClassifier()” function in tree module, “BaggingClassifier” function from ensemble module, “KNeighborsClassifier” from neighbors module ,” accuracy_score” function from metrics module and “StandardScaler” function from preprocessing module. These functions have built the classifier models and provided formula for us so that we only need to input the data needed to processed and parameters. “Panda” is a useful package for data analysis, which provides a quantity of tools for processing data quickly and easily. I mainly use it to read csv file and obtain data. “numpy” packet is a particularly powerful scientific computing toolkit. I use it for array and list operations. The package of time has a close relationship with the acquisition time. Using this function, the running time of classifiers can be obtained. Package torch contains the data structure of multi-dimensional tensor and various mathematical operations based on it. In addition, it also provides a variety of tools, some of which can serialize tensors and arbitrary types more efficiently. The modules of graphviz and pydotplus are used for the visualization of decision tree.

The whole modules in “test.py” are shown in Figure 7. In “test.py”, the additional libraries used are scipy, matplotlib and model_selection. Matplotlib is a drawing library of python. I use it to visualize the best parameters of the decision tree. Scipy is a commonly used software package used in the fields of mathematics, science and engineering. It can handle interpolation, integration, optimization, image processing and other issues. I use ‘pearson’ function for data analysis. The person correlation coefficient is used to measure whether two data sets are on a line. Meanwhile, it can also measure the linear relationship between the distance variables. The GridSearchCV() function in model_selection module adjust parameters automatically which is useful for small data sets.



```
#coding=gbk
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import numpy as np
import torch
import time
import torch.nn as nn
import torch.nn.functional as F
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
import graphviz
import pydotplus
from sklearn import tree
from IPython.display import Image
import os
```

Figure 6

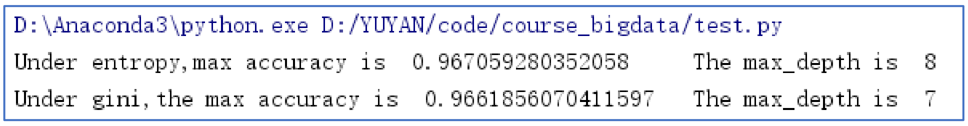


```
#coding=gbk
from scipy.stats import pearsonr
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import numpy as np
import torch
import torch.nn as nn
import torch.nn.functional as F
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
```

Figure 7

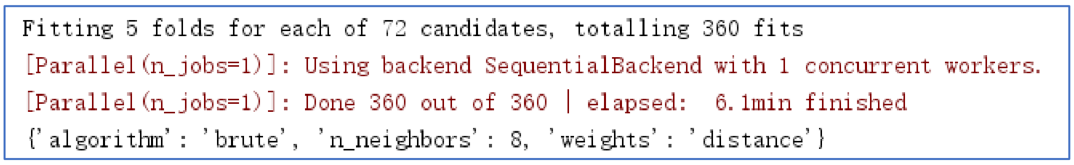
4.Results

The function of “test.py” is trying to find the best parameters of all of classifiers’ models which are built in the “bigdata_proj1.py”. Figure 8 shows the relative optimal parameters of decision tree and figure 9 show the relative optimal parameters of KNN. After that, these parameters are applied to obtain the accuracy and training time (Table 1). The result of a program run in the “bigdata_proj1.py” is also clear shown in Fig 10.



```
D:\Anaconda3\python.exe D:/YUYAN/code/course_bigdata/test.py
Under entropy,max accuracy is  0.967059280352058     The max_depth is  8
Under gini,the max accuracy is  0.9661856070411597   The max_depth is  7
```

Figure 8

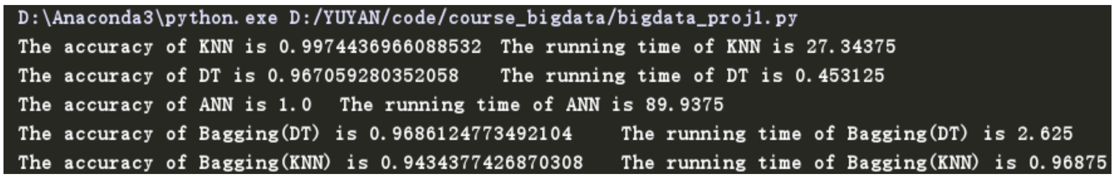


```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 360 out of 360 | elapsed:  6.1min finished
{'algorithm': 'brute', 'n_neighbors': 8, 'weights': 'distance'}
```

Figure 9

Classifier	Accuracy	Trainning time(s)
Decision Tree	0.967059280352058	1.984375
KNN	0.9974436966088532	26.3125
ANN	0.999967641729226	85.015625
Bagging (Decision Tree)	0.9686771938907585	0.859375
Bagging (KNN)	0.9450232979549573	1.078125

Table 1



```
D:\Anaconda3\python.exe D:/YUYAN/code/course_bigdata/bigdata_proj1.py
The accuracy of KNN is 0.9974436966088532  The running time of KNN is 27.34375
The accuracy of DT is 0.967059280352058    The running time of DT is 0.453125
The accuracy of ANN is 1.0  The running time of ANN is 89.9375
The accuracy of Bagging(DT) is 0.9686124773492104    The running time of Bagging(DT) is 2.625
The accuracy of Bagging(KNN) is 0.9434377426870308   The running time of Bagging(KNN) is 0.96875
```

Figure 10

5.Comparison and discussion

In this project, I totally use two python files. "bigdata_proj1.py" is used for model building. The other file called "test.py" is used to finding the relative optimum parameter of each classifier.

i. The experience gained

After this project, I am sure that I have gained a lot of experience in many aspects. The first thing I want to discuss is data processing. Obviously, the numerical difference between the feature and the feature is too large. It cannot be avoided for us to be faced with a large amount of data that differs greatly. Hence, preprocessing the data can make the model converge as soon as possible.

We can find that the feature of gameId, creationTime, gameDuring and seasonId are irrelevant features which do not make a difference on the result of a game (win or lose). When I first train data with ANN without deleting these irrelevant features, the accuracy of ANN model is always be 0.509804556044525. After looking up information, I found that it might be helpful to standardize the data. StandardScaler() is a useful function in sklearn library to standardize data. The data after standardizing has a mean value for 0 and a variance for 1. The formula is shown as Figure 11. After using StandardScaler() to standardize the value, the accuracy increased to 0.999967641729226.

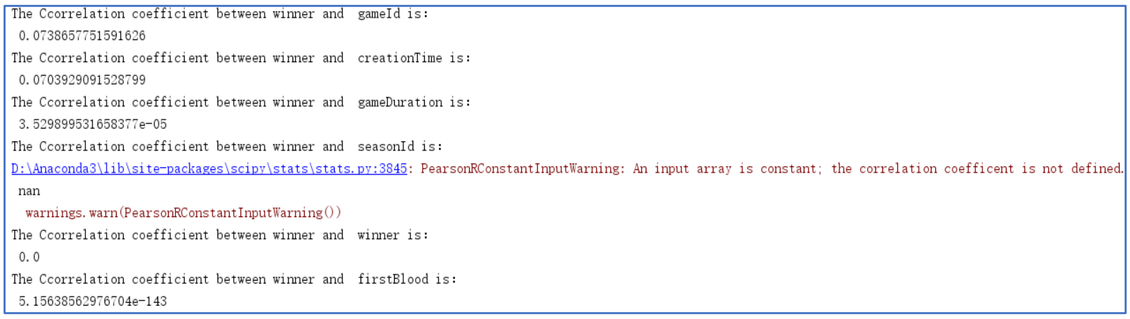
$$x^* = \frac{x - \mu}{\sigma}$$

Figure 11

$$\rho_{X,Y} = \frac{N\sum XY - \sum X \sum Y}{\sqrt{N\sum X^2 - (\sum X)^2}\sqrt{N\sum Y^2 - (\sum Y)^2}}$$

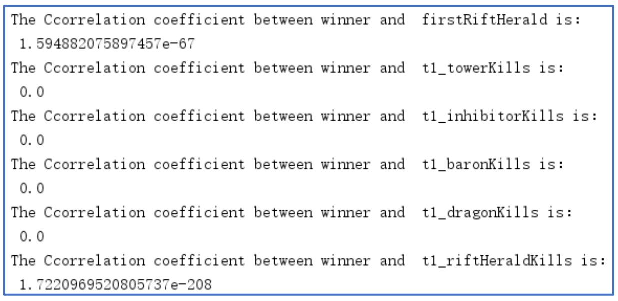
Figure 12

Next, I determine to delete irrelevant features to achieve the purpose of feature dimensionality reduction. The method is computing the correlation among variables according to Pearson coefficient (Figure 12). From the output of the pearsonr function, the redundant features can be found (Figure 13 and Figure 14). Except "seasonId" variable, we can artificially judge that there are at least the first four variables belong to irrelevant variables. However, something strange happened. I tried to manually delete those irrelevant variables and found that the accuracy was unexpectedly lower than before, only 0.9631989386487186. So far, I wonder if some features data deleted actually make a difference on the game result. But one thing can be convinced that a suitable method of processing data will greatly increase computing efficiency and improve computing results.



```
The Ccorrelation coefficient between winner and  gameId is:
 0.0738657751591626
The Ccorrelation coefficient between winner and  creationTime is:
 0.0703929091528799
The Ccorrelation coefficient between winner and  gameDuration is:
 3.529899531658377e-05
The Ccorrelation coefficient between winner and  seasonId is:
D:\Anaconda3\lib\site-packages\scipy\stats\stats.py:3845: PearsonRConstantInputWarning: An input array is constant; the correlation coefficent is not defined.
 nan
  warnings.warn(PearsonRConstantInputWarning())
The Ccorrelation coefficient between winner and  winner is:
 0.0
The Ccorrelation coefficient between winner and  firstBlood is:
 5.15638562976704e-143
```

Figure 13



```
The Ccorrelation coefficient between winner and  firstRiftHerald is:
 1.594882075897457e-67
The Ccorrelation coefficient between winner and  t1_towerKills is:
 0.0
The Ccorrelation coefficient between winner and  t1_inhibitorKills is:
 0.0
The Ccorrelation coefficient between winner and  t1_baronKills is:
 0.0
The Ccorrelation coefficient between winner and  t1_dragonKills is:
 0.0
The Ccorrelation coefficient between winner and  t1_riftHeraldKills is:
 1.7220969520805737e-208
```

Figure 14

In addition, my ability to find the best parameters of the model has been improved. As I mentioned earlier, basically every classifier has many parameters. These parameters more or less affect the final fitting results and time. As the number of function parameters and the amount of data sets increases, simply using a for loop or grid search will kill a lot of time. Hence, I first use a for loop and draw the figure to find out the influence of each parameter on the accuracy of the model, and them judged the interval of the best parameter according to the trend (Figure 15). Then use the grid method to find the best parameters in this interval.

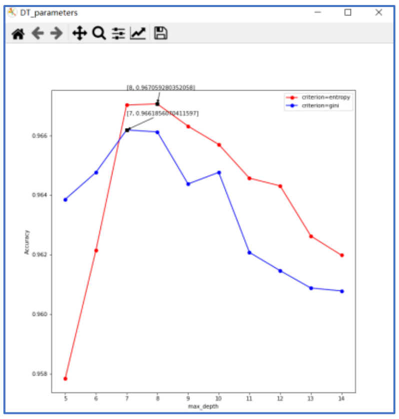


Figure 15

ii. Something to improve

In this project, it is pity for me that I cannot find the best parameters for bagging model. I encountered two serious problem. For one thing, using the grid search for bagging which is based on decision tree need lots of time. I wait for a long time, but the program did not run out of results. For another thing, there is a code error in the usage of the grid search for bagging which is based on KNN. It seems that I make some mistakes on the input of the KNN parameter. I will try to solve these two problems for improvement. Moreover, I will further explore the processing of redundant features with taking more images to visualize the data sets. If more time given, I also want to build more hidden layers in the ANN model in order to improve the model’s learning ability.

iii. Comparison

From Figure 10, the accuracy of artificial neural network is highest. At the same time, it took the longest time. After standardize the data, the influence of noise data on the ANN model is rapidly reduced. Besides, the time spent on finding parameters was large with the increase of hidden layer nodes.

In this project, KNN model has an excellent effect. Meanwhile its running time is much lower than ANN’s. What’s more, the algorithm of KNN is easy to understand which is benefit to us to master it. Compared with the other three models, this algorithm is more suitable for automatic classification of classes with larger sample sizes. Nevertheless, the biggest disadvantage is that it is more difficult to find the best parameters compared with decision tree and ANN. It is the model that takes the longest time to find parameters among the three models.

One of the obvious advantages of decision tree is that the amount of calculation is relatively small and the training speed is fast. It also does not require any domain knowledge and parameter assumptions, which is friendly to beginner learners. Moreover, decision tree is easy to understand and explain. The tools of graphviz and pydotplus facilitate our visual analysis of decision tree.

The bagging algorithm has a good effect in dealing with imbalances. Even though the running time of Bagging algorithm is short, the accuracy of Bagging classifier is not significantly improved compared with that of a single base classifier. It is also more difficult for me to adjust the parameters of the Bagging classifier and it takes longer.