Report

Introduction

Aiming to the establishment of well-performed classifiers which can predict the winner with given records, I used anaconda and python 3.8 to write my codes. I created three base classifiers, namely, decision tree, k-nearest neighbor and multi-layer perceptron. Then, I applied for fusion method for ensemble to use mean and max these two approaches to classify. Afterwards, I compared these five with training accuracy and training time to determine which one is the best and used the best to estimate the test dataset.

Algorithms

1. Decision tree

   Decision tree is essentially a set of nested if-else structure decision rules. I used gini index as classification criterion, it is commonly used and always performed well. Besides, I set its maximum depth at 18 and minimum samples split at 15, then the noisy data can be filtered out and improved the accuracy simultaneously.

2. K-nearest neighbor

   Template matching, the samples are divided into the classes of the most similar samples. The only parameter is n_neighbors, I set it at 15, corresponding to the minimum samples split for DT, this number comes from a lot of adjusting.

3. Multi-layer perceptron

   In addition to the input and output layers, there can be multiple hidden layers in the middle of the multi-layer perceptron. All parameters of MLP are the connection weight and bias between layers. The size of hidden layer is 18, equals to the number of maximum depth of DT, its maximum iteration is set at 1000, and random state is None. These parameters yield higher accuracy.

4. Mean value of the fusion method

   Predict the possibility of the three above base classifiers for each attribute of each sample corresponding to the two labels, and compute the mean possibility for making decisions.

5. Max value of the fusion method

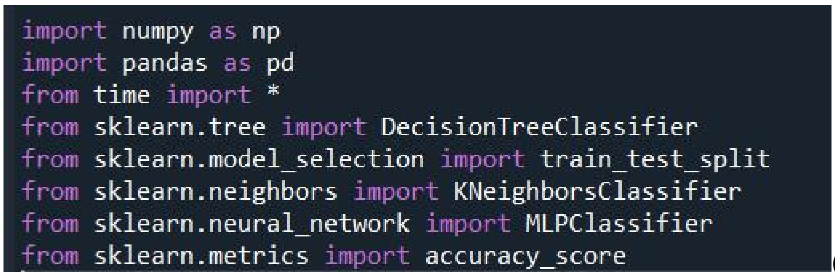
   Predict the possibility of the three above base classifiers for each attribute of each sample corresponding to the two labels, and compute the maximum possibility for making decisions.

6. Others

   I filtered the gameId, creationTime and seasonId in advance. This is because these are useless information. I printed the running time of each classifier for training and testing. For the fusion method, it is difficult to realize this function for the two different methods, then I calculated the total time and divided it into two equal pieces and distributed.

Requirements

The prerequisite packages are numpy, pandas, time and scikit-learn.

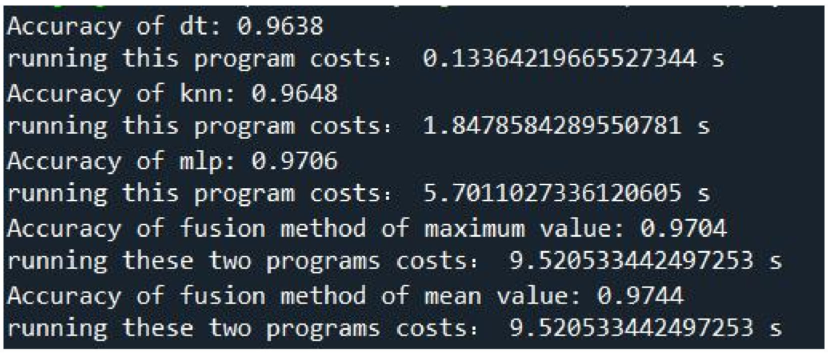


```
import numpy as np
import pandas as pd
from time import *
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
```

(figure 1)

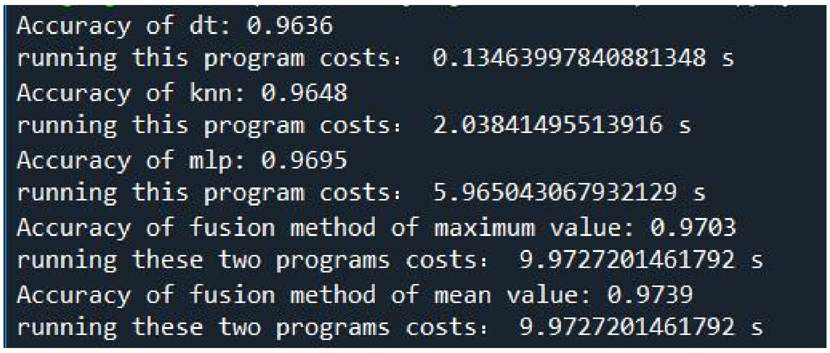
Results

The results of accuracy and time for each method:



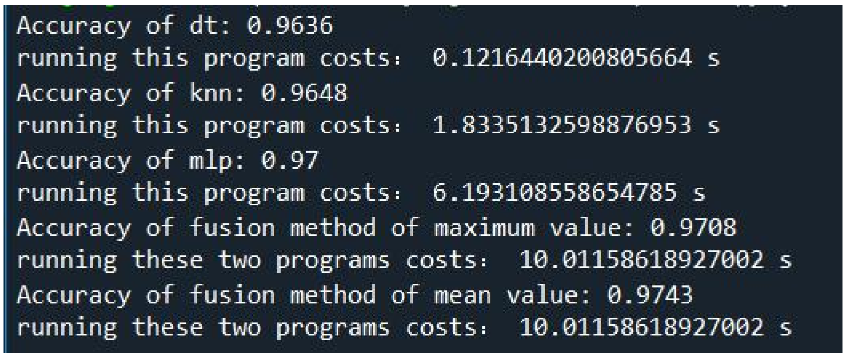
```
Accuracy of dt: 0.9638
running this program costs:  0.13364219665527344 s
Accuracy of knn: 0.9648
running this program costs:  1.8478584289550781 s
Accuracy of mlp: 0.9706
running this program costs:  5.7011027336120605 s
Accuracy of fusion method of maximum value: 0.9704
running these two programs costs:  9.520533442497253 s
Accuracy of fusion method of mean value: 0.9744
running these two programs costs:  9.520533442497253 s
```

(figure 2)



```
Accuracy of dt: 0.9636
running this program costs:  0.13463997840881348 s
Accuracy of knn: 0.9648
running this program costs:  2.03841495513916 s
Accuracy of mlp: 0.9695
running this program costs:  5.965043067932129 s
Accuracy of fusion method of maximum value: 0.9703
running these two programs costs:  9.9727201461792 s
Accuracy of fusion method of mean value: 0.9739
running these two programs costs:  9.9727201461792 s
```

(figure 3)

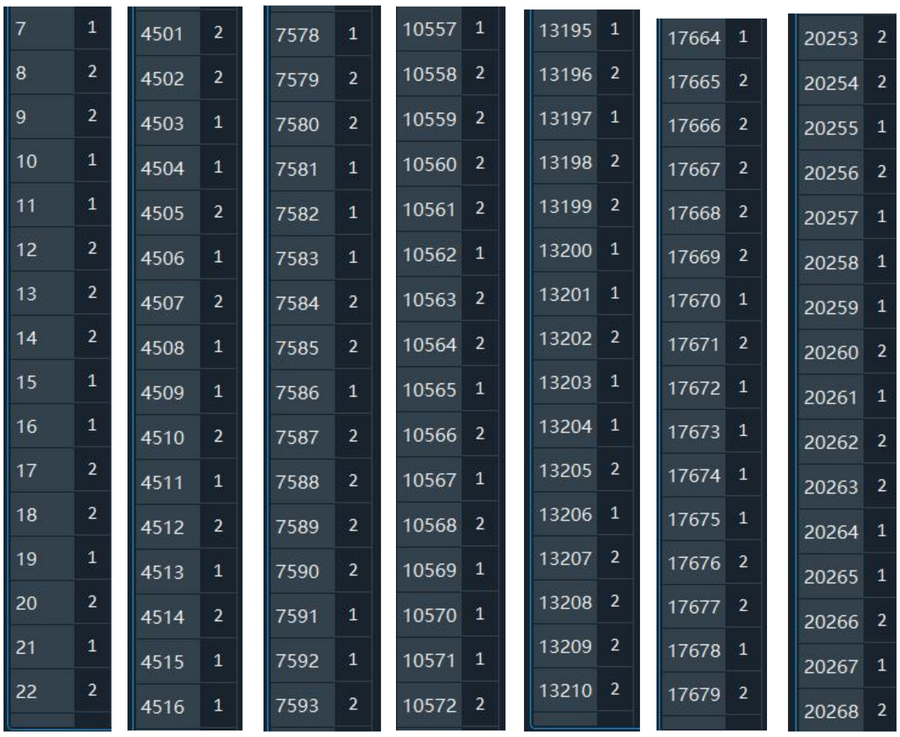


```
Accuracy of dt: 0.9636
running this program costs:  0.1216440200805664 s
Accuracy of knn: 0.9648
running this program costs:  1.8335132598876953 s
Accuracy of mlp: 0.97
running this program costs:  6.193108558654785 s
Accuracy of fusion method of maximum value: 0.9708
running these two programs costs:  10.01158618927002 s
Accuracy of fusion method of mean value: 0.9743
running these two programs costs:  10.01158618927002 s
```

(figure 4)

We can conclude that the five methods are all well-performed and the accuracy of fusion method of mean value is the highest. Furthermore, the time of decision tree is the least, almost finished in a flash. Maybe decision tree is the best choice when time is strictly limited and mean value is the best when enough time is given.

Then, I used mean value to predict the winner of the test dataset. Some of the outcomes are below:



7	1	4501	2	7578	1	10557	1	13195	1	17664	1	20253	2
8	2	4502	2	7579	2	10558	2	13196	2	17665	2	20254	2
9	2	4503	1	7580	2	10559	2	13197	1	17666	2	20255	1
10	1	4504	1	7581	1	10560	2	13198	2	17667	2	20256	2
11	1	4505	2	7582	1	10561	2	13199	2	17668	2	20257	1
12	2	4506	1	7583	1	10562	1	13200	1	17669	2	20258	1
13	2	4507	2	7584	2	10563	2	13201	1	17670	1	20259	1
14	2	4508	1	7585	2	10564	2	13202	2	17671	2	20260	2
15	1	4509	1	7586	1	10565	1	13203	1	17672	1	20261	1
16	1	4510	2	7587	2	10566	2	13204	1	17673	1	20262	2
17	2	4511	1	7588	2	10567	1	13205	2	17674	1	20263	2
18	2	4512	2	7589	2	10568	2	13206	1	17675	1	20264	1
19	1	4513	1	7590	2	10569	1	13207	2	17676	2	20265	1
20	2	4514	2	7591	1	10570	1	13208	2	17677	2	20266	2
21	1	4515	1	7592	1	10571	1	13209	2	17678	1	20267	1
22	2	4516	1	7593	2	10572	2	13210	2	17679	2	20268	2

Comparison and discussion

Via this project, I learnt how to apply for python to solve the big data classification problems. I now can skillfully use these classifiers and can quickly select the proper parameters. Besides, I finished this project by searching for plenty of blogs and websites, my abilities and horizons are improved and broadened. If enough time is allowed, I think I can created more classifiers and assign weight to different classifiers to fuse, and I believe better outcomes will be realized.