The Application of Supervised Learning on Classifiers Problem: Apply Scikit-Learn to Label LOL Match Records with "Win/Loss"

Du Jiahe

Content

I. Introduction

League of legends (LOL), a prevalent e-sport in the world, excites its followers because of the uncertainty of its outcomes. In this game, all kinds of information, such as the creation time and game duration, are related to the outcome of the game. In this project, approximate three million records of solo gamers are provided as the training set, and each record contains all publicly available statistical data of a game. In these data, label “1” in the field “winner” represents the team one won the game, and vice versa.

The main objective in this project is to create classifiers to process the data in training set and then label the test set, which comprises of about two million records, as “1” or “2”. After successfully completing previous tasks, the performance of the models should be evaluated using scoring parameters and the result must be higher than 50%. To pachase further improvement, comparison of these classifiers will be discussed and the possible extension of the program to bettering this project will be proposed.

The implementation is realized by applying the library scikit-learn in Python. Classifiers include support vector machine, decision tree, and ensemble methods of bagging, boosting and random forest are applied in the project. Aim of evaluating them in various aspects, scoring parameters accuracy, average_ precision, f1_score and AUC_score are used.

II. Algorithms

- **Classification method**

  The following table (table2.1) demonstrates the classifers build in this project. The second column indicates the parameters needed modified and their meaning. The field *Function* provides a brief introduction of each classifer and simply describe their advantages and disadvantages, which will be detailedly illustrated in the *Comparison and Discussion* section.

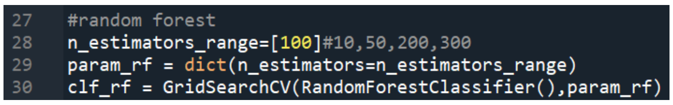
Table 2.1

Classifer	**Parameter**	**Function**
Support Vector Machine	**Kernel = linear** Choose from [linear, rbf, poly], which corresponding to not use kernel function to ascend dimension in fitting, using kernel to ascend dimension, and using radial basis function (Gaussian surface) to enhance dimension.	This classifer is only applicable to binary classification problems. It is efficient for high-dimension data and large number of samples, but when applies to large number of samples, you should avoid overfitting when choosing the kernel.
Decision tree	**Max_depth = 8** maximum depth of the tree, no limitation by default.	It supports multi-label classification. This model is easlier to understand than the others, and it can be trained with small samples. However, it is more likely to generate a model that is too complicated. Usually, decision tree is not so stable and may overfit.
Ensemble		
Random forest	**n_estimator = 100** The number of estimators in the ensemble.	This ensemble method combines the results of a certain number of base estimators, decision trees, whose aim is to reduced variance then obtain better robustness. Moreover, this model can reduce the overfitting of decision tree. For each estimator, set number of

		samples and features will be used, and the samples are chosen randomly with replacemnet. In addition, when the nodes are segmented in the process of building the tree, the selected segmentation points are the best segmentation points for all features
Adaboost	**n_estimator = 350** The number of estimators in the ensemble.	The key of Adaboost is to adjust the weight of the sample in each boosting. The sample that is mistakenly classified to a wrong class will get a greater weight, so the classifier will be forced to pay more attention to the sample with wrong label.

- **Parameter selection**

In the process of building these classifers, gird-search (figure 2.1) is used to find the best parameter. This method summarized as exhaustive search, that is, in the selection of all candidate parameters, every possibility is tried by loop traversal, and the parameter with the best result will be chosen.



```
#random forest
n_estimators_range=[100]#10,50,200,300
param_rf = dict(n_estimators=n_estimators_range)
clf_rf = GridSearchCV(RandomForestClassifier(),param_rf)
```

Figure 2.1

III. Requirements

The following table demonstrates the prerequisite packages used in this project.

Table 3.1

Prerequisite packages	**Function**
pandas	Read data from csv file
sklearn.preprocessing	Preprocessing the data
sklearn.model_selection	Use gird-search to select the best paramete
sklearn	Decision tree model
sklearn.svm	Ssupport vector machine model
sklearn. ensemble	Adaboost and Random forest
Sklearn.metrics	Use scoring parameter to evaluate the models
time	Calculate the running time

IV. Results

- **Scoring parameter**

  To illustrate the parameter clearly, the *confusion matrix* (table4.1) will be demonstrated first.

  The total number of samples is $N = TP + FN + FP + TN$

Table 4.1

	Predicted class		
Actual Class		Yes	No
	Yes	TP	FN
	No	FP	TN

In table 4.2 are the parameters used to evalutate the classifiers in this project.

Table 4.2

Parameter	Introduction
Accuracy	$\text{accuary} = \frac{\text{number of samples correctly predicted}}{\text{total number of samples}} = \frac{\text{TP} + \text{TN}}{\text{N}}$ It reflects the ability of the classifier to judge the whole samples and the probability of correctly classified samples.
Precison	$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$ It only reflects the ability of the classifier to correctly classify class 1.
Recall	$\text{recall} = \frac{\text{samples correctly classified as class 1}}{\text{tatal samples of calss1}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$ it reflects the proportion of samples of class1 correctly classified as class1.
F1_score	$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$ This parameter combines the performance of precision and recall and shows the balanced point of them.
Roc_auc_score	AUC is the area under curve of ROC, and ROC is based on FP and TP to evaluate the prediction ability of the model. Usually, this value is between 0.5 and 1, and bigger th better.

- **Evaluating each classifier**

Table 4.3

Parameter	SVM	Decision tree	Adaboost	Random forest
Accuracy	0.9563	0.9632	0.9638	0.9646
Precison	0.9366	0.9397	0.9410	0.9421
Recall	0.9697	0.9781	0.9776	0.9783
F1_score	0.9599	0.9636	0.9642	0.9649
Roc_auc_score	0.9596	0.9632	0.9638	0.9646

Running time/s	59.6854	0.2309	45.1608	6.7080

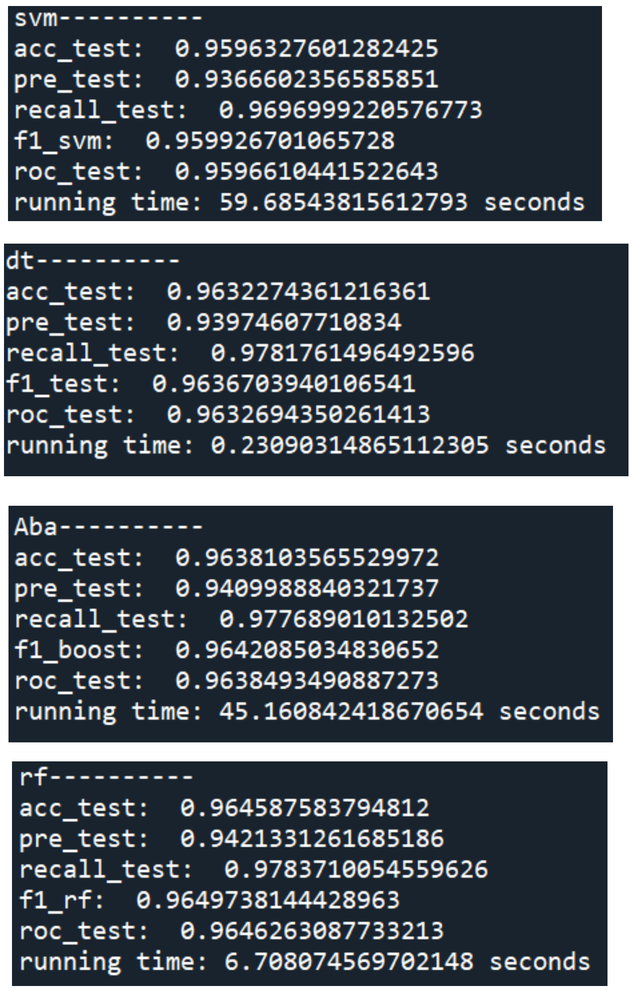
From table 4.3 it can be concluded that *Random Forest* performs best.

Its scoring parameters score the highest, and the running time is relatively short.

Although *Decision Tree* only takes a very short running time, its performance is slightly worse than that of *Random forests*. *Adaboost* is also doing well, but its running time is too long, which is unsatisfactory.

- **Screencut of the result**

The following figures are the output on console of the four classifers. They are given the same order as table 4.3.



```
svm----------
acc_test:  0.9596327601282425
pre_test:  0.9366602356585851
recall_test:  0.9696999220576773
f1_svm:  0.959926701065728
roc_test:  0.9596610441522643
running time: 59.68543815612793 seconds
```

```
dt----------
acc_test:  0.9632274361216361
pre_test:  0.93974607710834
recall_test:  0.9781761496492596
f1_test:  0.9636703940106541
roc_test:  0.9632694350261413
running time: 0.23090314865112305 seconds
```

```
Aba----------
acc_test:  0.9638103565529972
pre_test:  0.9409988840321737
recall_test:  0.977689010132502
f1_boost:  0.9642085034830652
roc_test:  0.9638493490887273
running time: 45.160842418670654 seconds
```

```
rf----------
acc_test:  0.964587583794812
pre_test:  0.9421331261685186
recall_test:  0.9783710054559626
f1_rf:  0.9649738144428963
roc_test:  0.9646263087733213
running time: 6.708074569702148 seconds
```

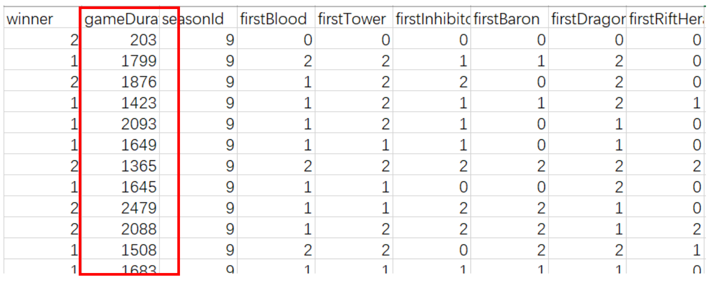
Figure 4.1

V. Comparison and Discussion

- **Data proprocessing**

When dealing with big data project, it may happen that the difference between the orders of magnitude of the feature is so large that leads to the dissaatified performance of the classifer, such as low accuracy and long running time.

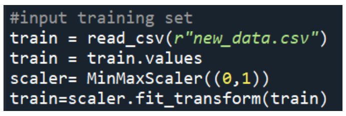
in this project, the fields of the records contain the *game ID*, winner, and other 18 features. Since the orders of magnitude of the data in certain field is much larger than the others (figure 5.1), which means the value of this feature does not obey the standard normal distribution, the performance of the classifers will become very poor.



winner	gameDura	seasonId	firstBlood	firstTower	firstInhibitc	firstBaron	firstDragor	firstRiftHer
2	203	9	0	0	0	0	0	0
1	1799	9	2	2	1	1	2	0
2	1876	9	1	2	2	0	2	0
1	1423	9	1	2	1	1	2	1
1	2093	9	1	2	1	0	1	0
1	1649	9	1	1	1	0	1	0
2	1365	9	2	2	2	2	2	2
1	1645	9	1	1	0	0	2	0
2	2479	9	1	1	2	2	1	0
2	2088	9	1	2	2	2	1	2
1	1508	9	2	2	0	2	2	1
1	1683	9	1	1	1	1	1	0

Figure 5.1

Therefore, standardlization (figure 5.2) is used to preprocess the data to ensure the classifers work well. With this algorithm, the value range of each feature can be set between 0 and 1, and thus increasing the robustness to the feature.



```
#input training set
train = read_csv(r"new_data.csv")
train = train.values
scaler= MinMaxScaler((0,1))
train=scaler.fit_transform(train)
```

Figure 5.2

- **Feature selection**

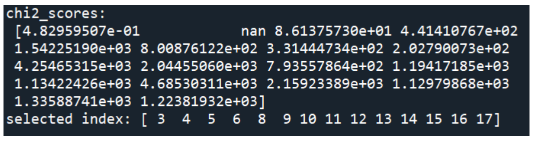
After data preprocessing, it is necessary to select meaningful features then input them into the models of machine learning for training. This is related to the divergence of the feature and relanvance of the features and the target. If a feature is not divergent and its variance is closer to 0, it not so helpful for the model to do the classification. Also, the feature that is not relavant to the target is less important for the model. Therefor, feature selection can descent the dimension of the features, thus enhances the accuracy and reduces the running time of the program.

In this project, selection methods from scikit learn are used. The following table show the evaluation function applied in the project.

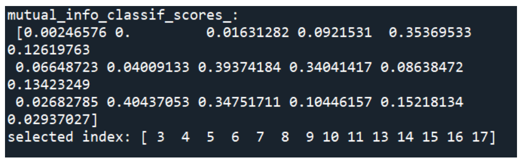
Table 5.1

Evaluation function	**Introduction**
chi2	Chi-square test is a widely used hypothesis testing method. It counts the deviation degree between actual observation value and theoretical inference value, which determines the chi-square value. The smaller the chi-square value, the greater the consistency of the feature and target.
f_classif	It is mainly used to judge whether the variance between two samples is obviously different, and it is mainly used in the task of classifying feature into category.
mutual_info_classif	This value is the mutual information between variables and target values, and it usually used in classification tasks. Mutual information is a significant measurement of information, which can be regarded as the amount of information about one random variable contained in another random variable, or the uncertainty of a random variable reduced by knowing another random variable.

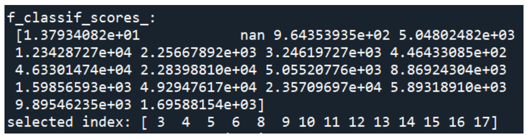
The following are the result of the feature selection of the 18 features (figure 5.3), which are the columns 0 to 17. The feature chosen by all three algorithms are set to be the features used in the models.



```
chi2_scores:
 [4.82959507e-01            nan 8.61375730e+01 4.41410767e+02
 1.54225190e+03 8.00876122e+02 3.31444734e+02 2.02790073e+02
 4.25465315e+03 2.04455060e+03 7.93557864e+02 1.19417185e+03
 1.13422426e+03 4.68530311e+03 2.15923389e+03 1.12979868e+03
 1.33588741e+03 1.22381932e+03]
selected index: [ 3  4  5  6  8  9 10 11 12 13 14 15 16 17]
```



```
mutual_info_classif_scores_:
 [0.00246576 0.         0.01631282 0.0921531  0.35369533
0.12619763
 0.06648723 0.04009133 0.39374184 0.34041417 0.08638472
0.13423249
 0.02682785 0.40437053 0.34751711 0.10446157 0.15218134
0.02937027]
selected index: [ 3  4  5  6  7  8  9 10 11 13 14 15 16 17]
```



```
f_classif_scores_:
 [1.37934082e+01            nan 9.64353935e+02 5.04802482e+03
 1.23428727e+04 2.25667892e+03 3.24619727e+03 4.46433085e+02
 4.63301474e+04 2.28398810e+04 5.05520776e+03 8.86924304e+03
 1.59856593e+03 4.92947617e+04 2.35709697e+04 5.89318910e+03
 9.89546235e+03 1.69588154e+03]
selected index: [ 3  4  5  6  8  9 10 11 12 13 14 15 16 17]
```

Figure 5.3

- **Overall performance of the four classifers**

  From the outcomes it can be summarized that the performance of ensemble methods is better than a single classifier. Since the goal of the ensemble methods is to combine the predicted results of multiple base estimators, built by using a given learning algorithm, to achieve better generalization capability and robustness than a single estimator. For *Adaboost* and *Random Forest*, they combine multiple weak models, *Decision Tree*, and strengthen the integrated model. However, when dealing with different data sets, their algorithms will contribute different performances, and the following table (table 5.2) shows some of their disctintions.

Table 5.2

Difference	**Adaboost**	**Random Forest**
Sample selection	All samples	Ramdonly choose the subset of all samples with replacement
Weight	Misclassified samples have greater weight	All have same weight
Estimator	Iteratively generate in order	Parallelly generate
Characteristic	All tree weighted voting to	

	determine the predicted value of the dependent variable	Split nodes during tree construction
Advantage	The algorithm is easier to understand and it can be used without feature selection	Reduce variance of the feature and avoid overfitting.

Although their performance for this dataset are similar, but the running time of *Random Forest* is much shorter than *Adaboost.* For *Support Vector Machine*, when the dataset is large, the model is hard to implement because of its design of algorithm. Meanwhile, on a great extend, its performance depends on the choice of the kernel and other parameters, which are selected according to experience and with a certain degree of randomness. As the outcome above, this classifier takes a long time and not performs so well.

- **Experience gained in this project**

  This project brought me into the field of big data processing. In this project, I mainly applied the algorithm in scikit learn to achieve the objectives. For the dataset, I obtain a preliminary understanding of data preprocessing and feature selection, and put it into practice. As for classifiers, I learnt the basic knowledge of their principles, parameters, advantages and disadvanges, then compared them to master their relationship. In order to get their performance, I also attempt to figure out the meaning and application condition of the scoring parameters. In general, I gained tremendous experience of how to deal with a big data project.

- **Possible improvement if time permitting**

  I complete the project with a relatively short time, so if more time is provided, I will definitely make improvements to optimize my project. First of all, I want to explore more about parameters and parameters selection and find better parameters to fit the samples so as to improve the accuarcy. I also intend to acquire more knowledge about classifiers and their characteristic. Secondly, I am interested in feature engineering, and I am willing to learn more about it even if it doesn't involve much of this task. Additionally, to make my result more convincing and reducing randomicity, I will run the program for a few more time to obtain the average scores. In conclusion, I want to go deeper into the field of big data processing and complete my work better.