movie score prediction classification model

Team: salted fish

Data set:

The data set uses a movie set containing more than 5,000 movie details.

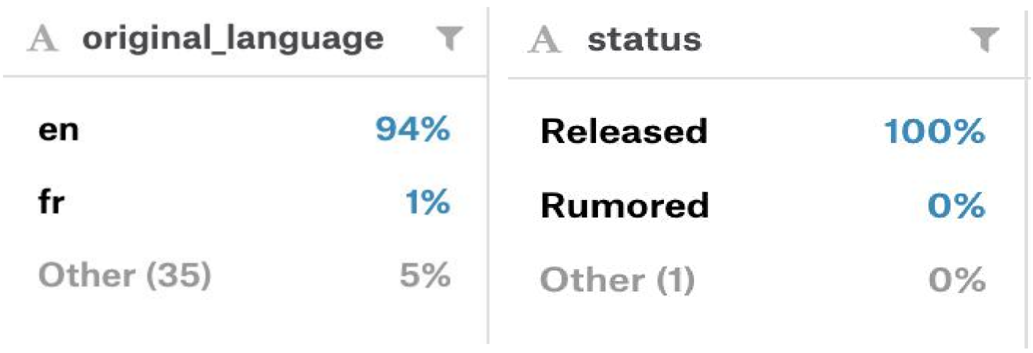
Contains the following information for each movie:

budget	id	homepage	Run_time	keywords	original_language	production_companies
status	revenue	overview	release_date	genres	original_title	production_countries
title	Tag-line	vote_count	vote_average	popularity	spoken_languages	

Data processing:

1. First, we remove the URL, id, tag-line, and movie name on other sites, because it's hard to provide a larger reference to the model.

2.Second, we remove features with low variance, For example: Status And original_language.



A original_language		A status	
en	94%	Released	100%
fr	1%	Rumored	0%
Other (35)	5%	Other (1)	0%

There are almost no variances, almost all values are the same, so these features will not have a large positive effect on the results.

3.In the rest, we hope to achieve a correct rate of 70 or more, and hope that the user's input is as simple as possible, so we chose the budget, genres, keywords, popularity, production_companies, release_date.

4.The release date is difficult to locate specific months and days, so we use the year as a feature and calculate the number of days between the day of the movie release and January 1 of the release year as another feature.

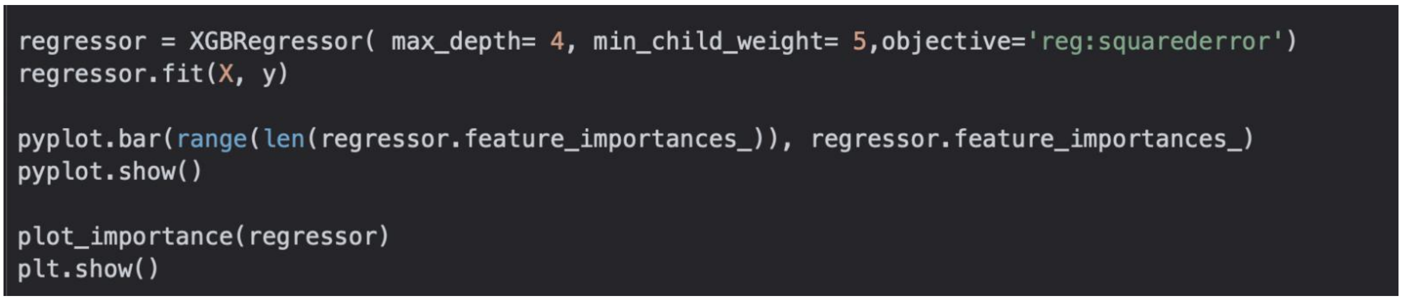
5.Invalid data with a score 0 was cleaned.

6.We took out all the movie types, keywords, company names that appeared in the data set, and set each occurrence value to a feature.

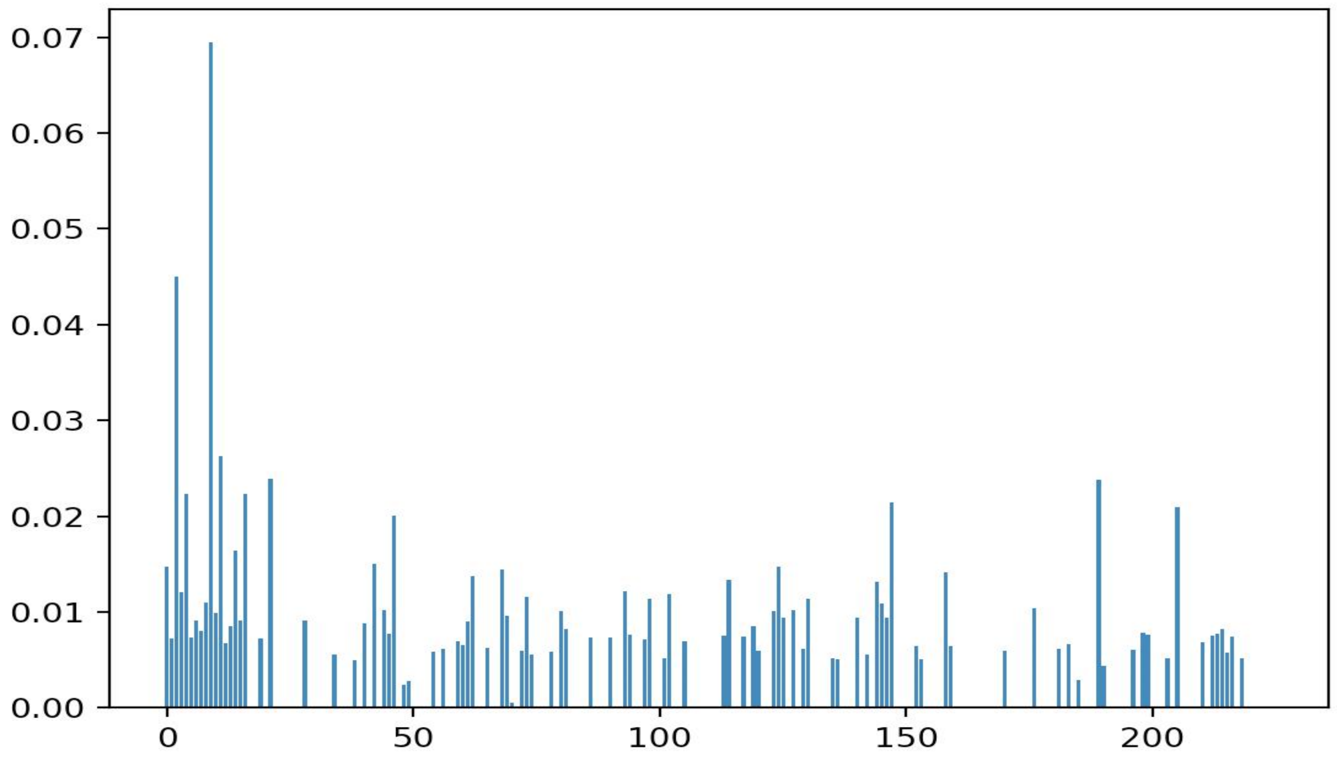
The first 4 columns are [Budget,popularity,run time,release time]

The rest are different type(all value were taken here), keywords(only 100 were taken here), companies(only 100 were taken here.).

7.After that, we use regressions to determine the importance of their scores which we need to predict.Then we got those graph:

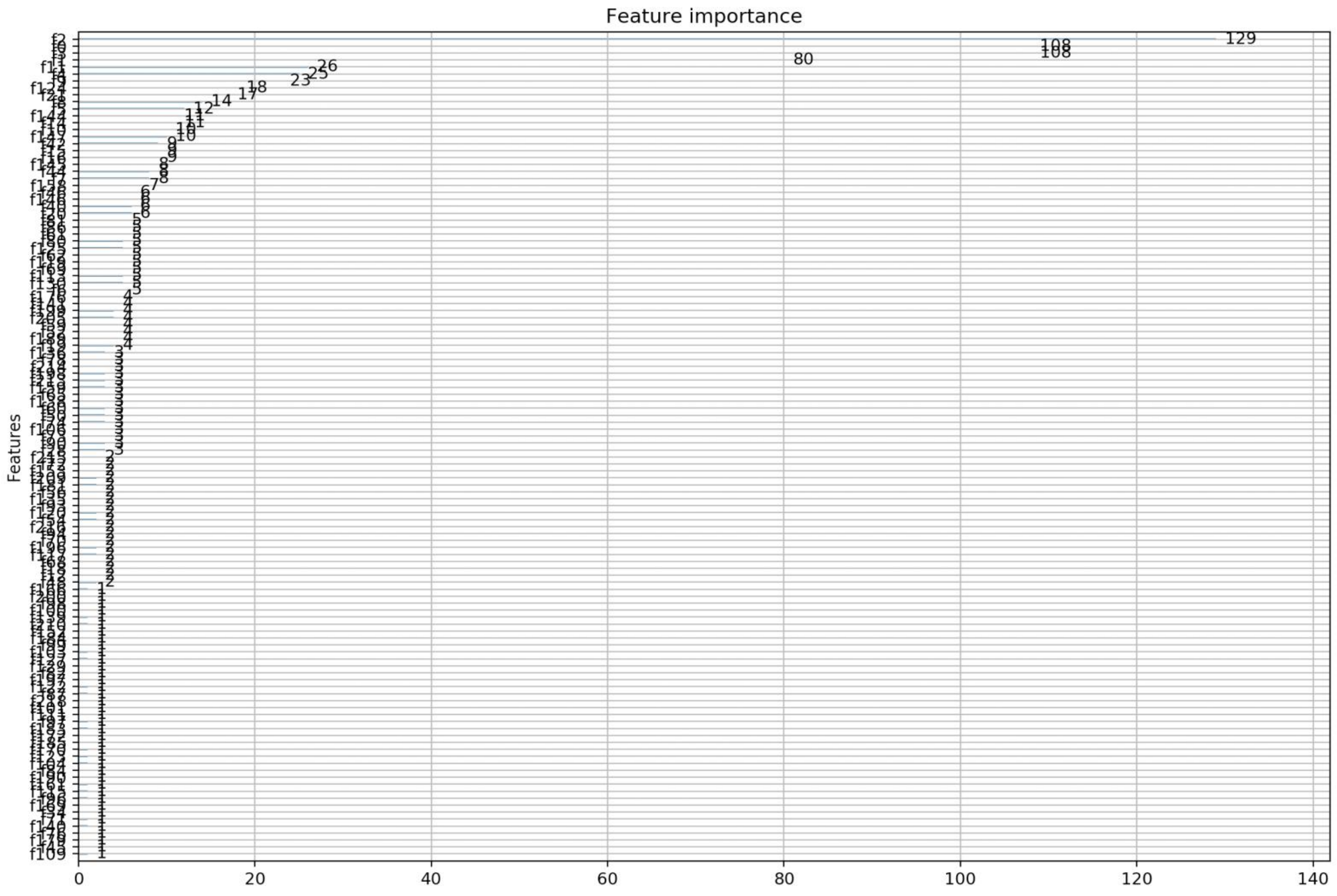


```
regressor = XGBRegressor( max_depth= 4, min_child_weight= 5,objective='reg:squarederror')
regressor.fit(X, y)

pyplot.bar(range(len(regressor.feature_importances_)), regressor.feature_importances_)
pyplot.show()

plot_importance(regressor)
plt.show()
```

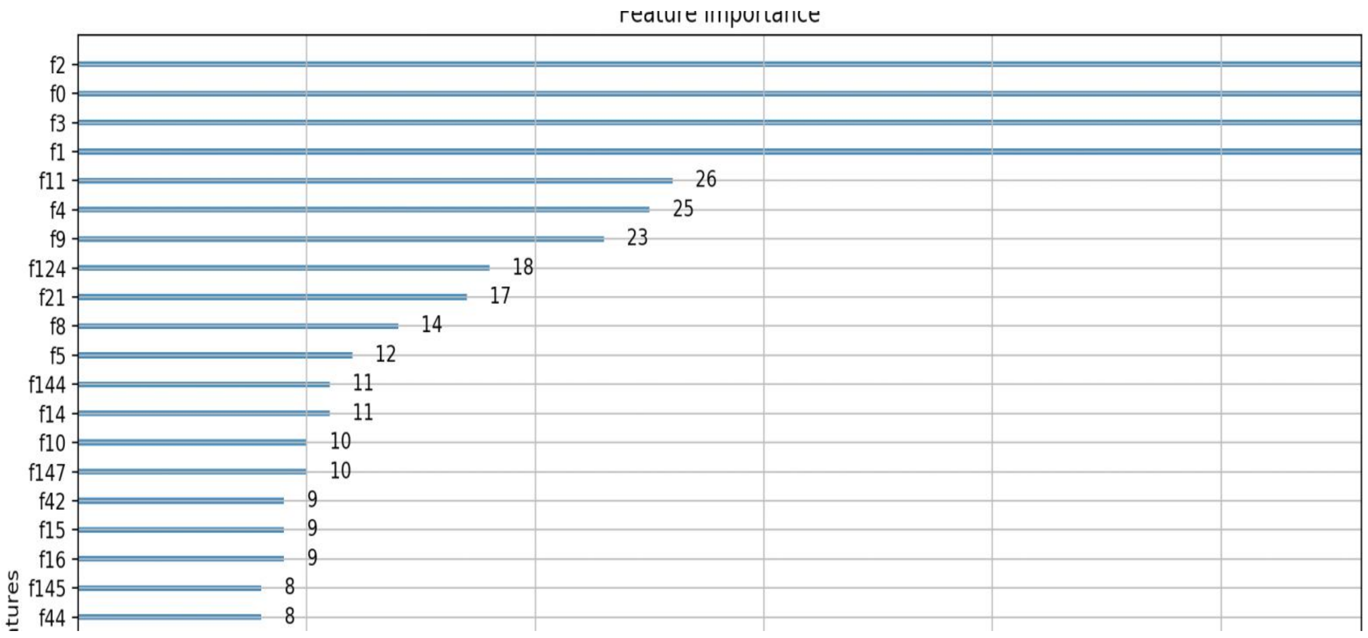


Graph 1

From this figure, it is clear that the important features are concentrated in the higher position.



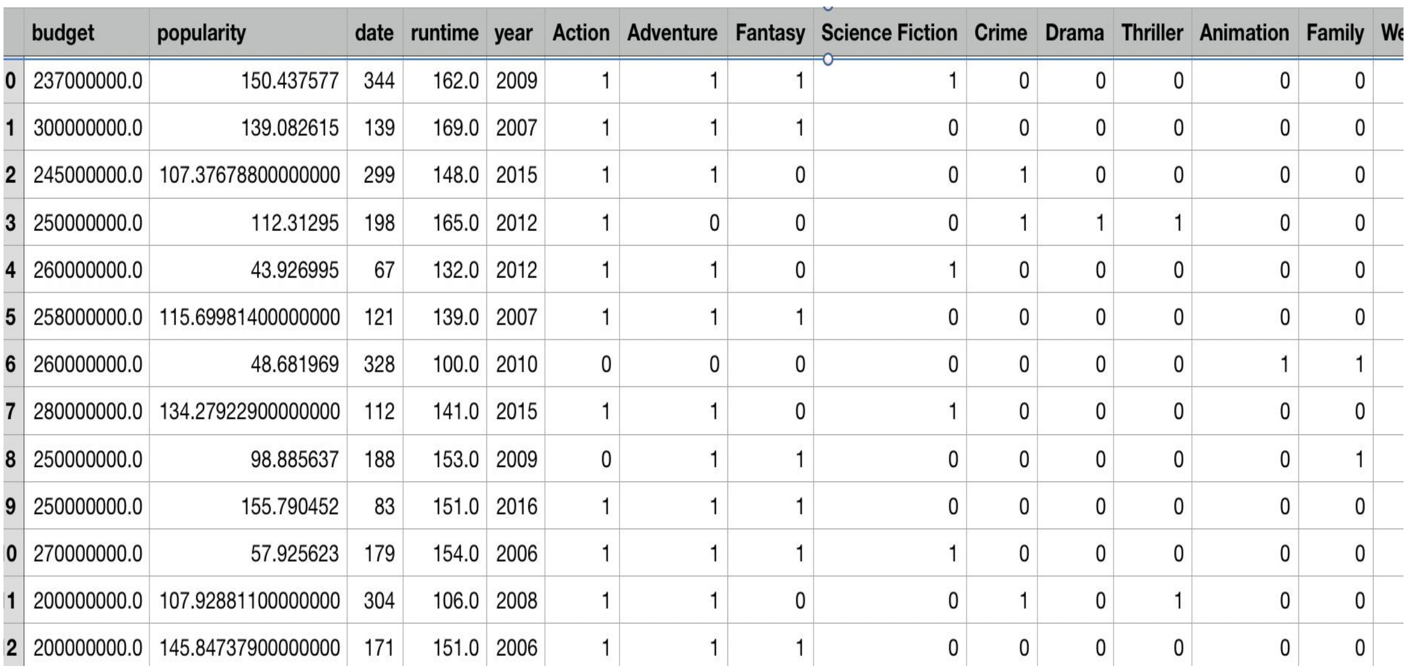
Graph 2



Graph 3

It can be clearly seen from this figure that 9 of the top 10 importance levels are between f0-f26, and in the top 20 important feature, 14 are between f0-f26 (Budget, popularity, run time, release time and different type total 26 columns)

8.In summary, we chose Budget, popularity, run time, release time and different type:



	budget	popularity	date	runtime	year	Action	Adventure	Fantasy	Science Fiction	Crime	Drama	Thriller	Animation	Family	We
0	237000000.0	150.437577	344	162.0	2009	1	1	1	1	0	0	0	0	0	
1	300000000.0	139.082615	139	169.0	2007	1	1	1	0	0	0	0	0	0	
2	245000000.0	107.37678800000000	299	148.0	2015	1	1	0	0	1	0	0	0	0	
3	250000000.0	112.31295	198	165.0	2012	1	0	0	0	1	1	1	0	0	
4	260000000.0	43.926995	67	132.0	2012	1	1	0	1	0	0	0	0	0	
5	258000000.0	115.69981400000000	121	139.0	2007	1	1	1	0	0	0	0	0	0	
6	260000000.0	48.681969	328	100.0	2010	0	0	0	0	0	0	0	1	1	
7	280000000.0	134.27922900000000	112	141.0	2015	1	1	0	1	0	0	0	0	0	
8	250000000.0	98.885637	188	153.0	2009	0	1	1	0	0	0	0	0	1	
9	250000000.0	155.790452	83	151.0	2016	1	1	1	0	0	0	0	0	0	
0	270000000.0	57.925623	179	154.0	2006	1	1	1	1	0	0	0	0	0	
1	200000000.0	107.92881100000000	304	106.0	2008	1	1	0	0	1	0	1	0	0	
2	200000000.0	145.84737900000000	171	151.0	2006	1	1	1	0	0	0	0	0	0	

Graph 4

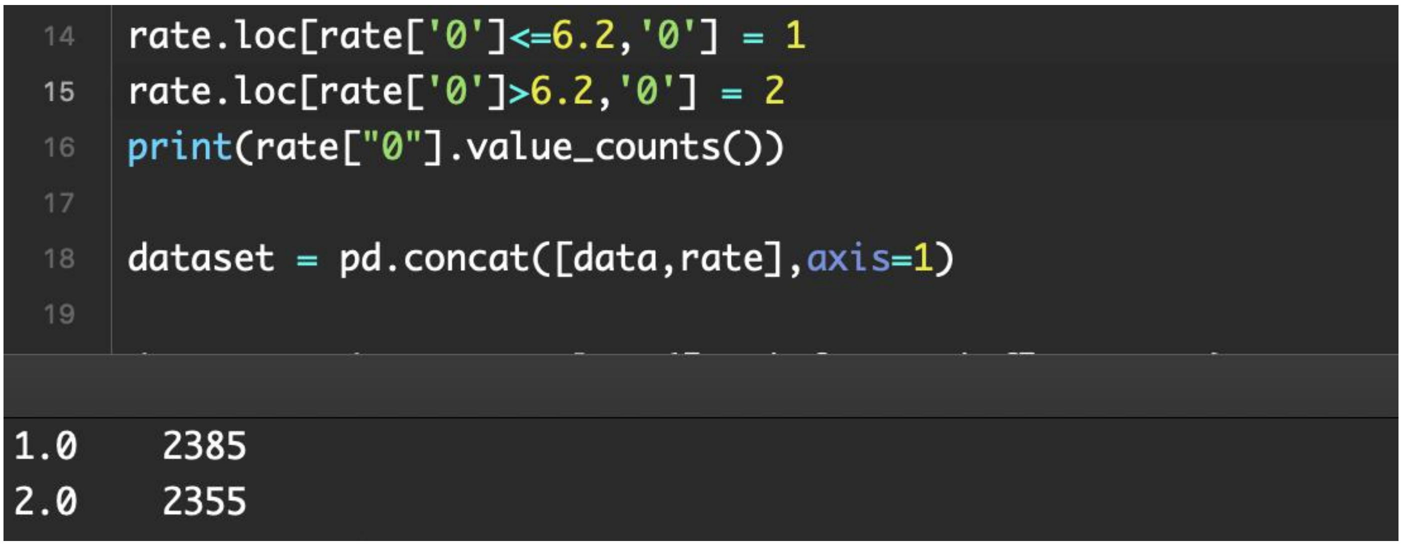
Model:

1. We conducted a two-category training on the score. We use XGBClassifier in XGBoost with Scikit-learn.

2. Arguments: (min_child_weight =1, max_depth=4, learning_rate=0.01, n_estimators=800, objective = 'binary:logistic')

3. Select 20% of the test set as a test set.

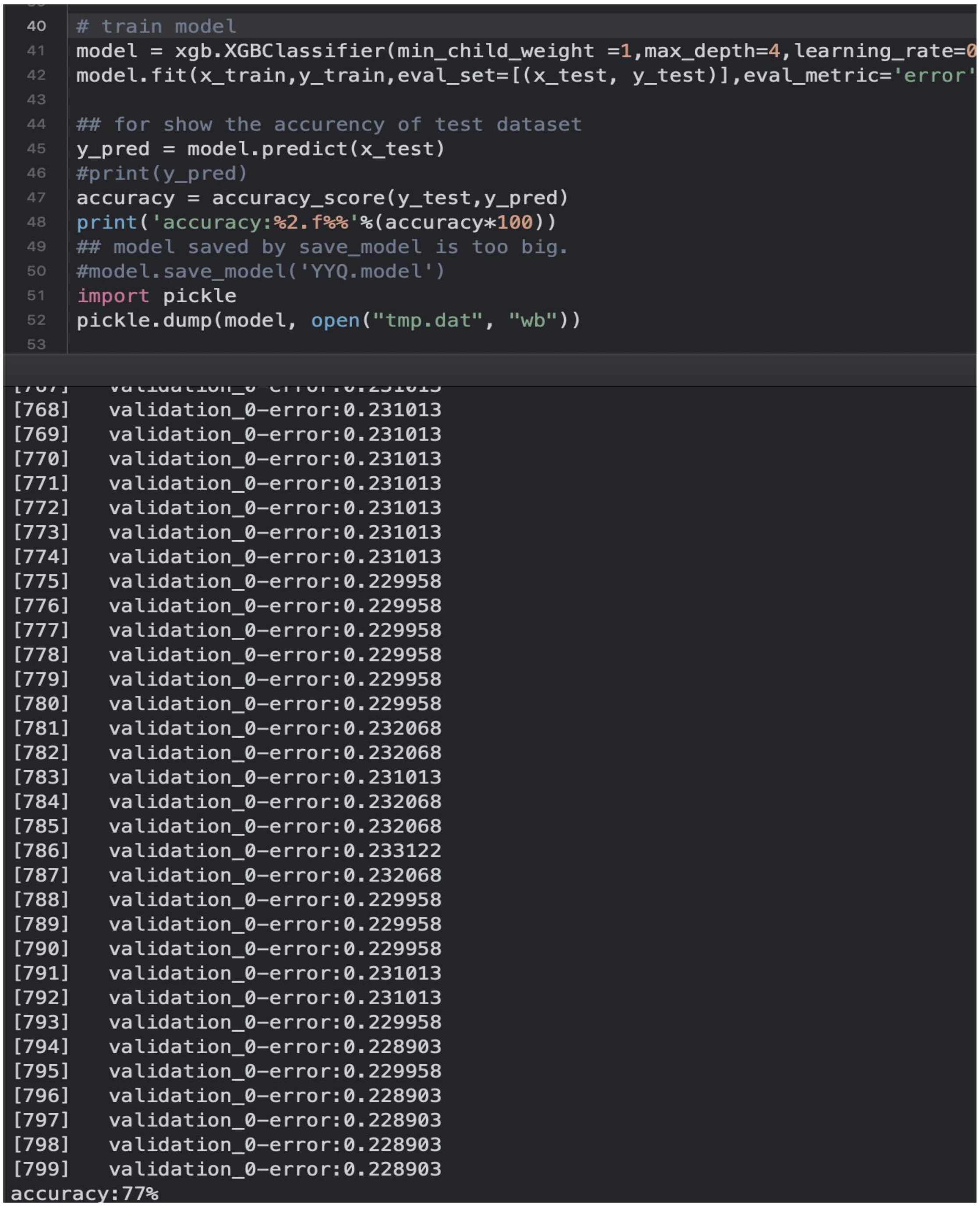
4. In order to obtain a nearly balanced binary data set, we divided the score by more than 6.2. points and less than 6.2 points.



```python
rate.loc[rate['0']<=6.2,'0'] = 1
rate.loc[rate['0']>6.2,'0'] = 2
print(rate["0"].value_counts())

dataset = pd.concat([data,rate],axis=1)

```

```
1.0    2385
2.0    2355
```

6. Get about 77% correct rate in the test set:



```python
# train model
model = xgb.XGBClassifier(min_child_weight =1,max_depth=4,learning_rate=0
model.fit(x_train,y_train,eval_set=[(x_test, y_test)],eval_metric='error'

## for show the accurency of test dataset
y_pred = model.predict(x_test)
#print(y_pred)
accuracy = accuracy_score(y_test,y_pred)
print('accuracy:%2.f%%'%(accuracy*100))
## model saved by save_model is too big.
#model.save_model('YYQ.model')
import pickle
pickle.dump(model, open("tmp.dat", "wb"))

```

```
[768]	validation_0-error:0.231013
[769]	validation_0-error:0.231013
[770]	validation_0-error:0.231013
[771]	validation_0-error:0.231013
[772]	validation_0-error:0.231013
[773]	validation_0-error:0.231013
[774]	validation_0-error:0.231013
[775]	validation_0-error:0.229958
[776]	validation_0-error:0.229958
[777]	validation_0-error:0.229958
[778]	validation_0-error:0.229958
[779]	validation_0-error:0.229958
[780]	validation_0-error:0.229958
[781]	validation_0-error:0.232068
[782]	validation_0-error:0.232068
[783]	validation_0-error:0.231013
[784]	validation_0-error:0.232068
[785]	validation_0-error:0.232068
[786]	validation_0-error:0.233122
[787]	validation_0-error:0.232068
[788]	validation_0-error:0.229958
[789]	validation_0-error:0.229958
[790]	validation_0-error:0.229958
[791]	validation_0-error:0.231013
[792]	validation_0-error:0.231013
[793]	validation_0-error:0.229958
[794]	validation_0-error:0.228903
[795]	validation_0-error:0.229958
[796]	validation_0-error:0.228903
[797]	validation_0-error:0.228903
[798]	validation_0-error:0.228903
[799]	validation_0-error:0.228903
accuracy:77%
```