MDS6212 Fintech Theory and Practice
Assignment 01

Youcong Lu

Content

Week 1 Assignment Report

1.Data Overlook

There are total 5000 records with 47 columns. Based on past research on likelihood of loan delinquency and loan approval, we do some statistic summary on the key variables on the sample space, including age, gender, loan amount, interest rate, credit scores.

Based on the result, we can define these key variables into three types:

(1) Personal properties (such as age and gender):

We found that the average age of the sample is about 27.5 which means it is a rather "young sample". At the same time, after we transform gender into numeric value (0: Male, 1 : Female), the statistics indicates that the proportion of males in the sample is much higher. To be more specific, gentlemen are six times more than lady in the sample space. Based on the result it occurs to me that gender is likely to be a useless result in the sample.

(2) Properties of the loan (such as: instalments and rates):

As for these two variables, data in instalments is fairly clean, and we can easily get the statistics of instalments. However, there are three records (0.06%) missing in rates. Since the number of missing records is limited, we will simply fill the value with mean value of the sample.

(3) Credit score ("creditlevelsbuyer", "tencentscore","gaodescore")

There are three types of credit score, including "creditlevelsbuyer", "tencentscore" and "gaodescore". Among three types of credit score, the first type is most dirty, with about 20% missing data. Besides, it ranges from 0 to 1830, leading to a high variance of sample. Therefore, we need to fill the null value in this column.

(4) Digital footprint (such as "highcontact")

The column "highcontact" stands for if the borrower has frequent contact records. I create a dummy variable for this column. There are total 5000 observations, including similarly same number of two types of sample.

(5) Loan status ("default", "deal")

There are 5000 observations in deal column while only 2205 in column default. These two variables are originally labels so I create dummy variable to represent the original variables.

	age	gender	instalments_amount	nominalrates	creditlevelasbuyer	tencentscore	gaodescore	highcontact	default	deal
count	5000.000000	5000.000000	5000.00000	4997.000000	4031.000000	5000.000000	5000.000000	5000.000000	2205.000000	5000.000000
mean	27.675400	0.146600	406201.42000	0.276058	53.119077	58.608168	0.201975	0.492200	0.419501	0.441400
std	8.326146	0.353742	130623.36024	0.085912	108.629757	14.218112	0.076724	0.499989	0.493589	0.496604
min	18.000000	0.000000	50000.00000	0.130080	0.000000	9.000000	0.023518	0.000000	0.000000	0.000000
25%	21.000000	0.000000	320000.00000	0.204560	0.000000	53.888889	0.192094	0.000000	0.000000	0.000000
50%	25.000000	0.000000	398000.00000	0.204579	14.000000	60.200000	0.192094	0.000000	0.000000	0.000000
75%	32.000000	0.000000	498000.00000	0.359347	58.000000	65.258929	0.192094	1.000000	1.000000	1.000000
max	56.000000	1.000000	869000.00000	0.494185	1830.000000	98.000000	0.732120	1.000000	1.000000	1.000000

Table 1-1 Feature summary

2.Data Cleaning

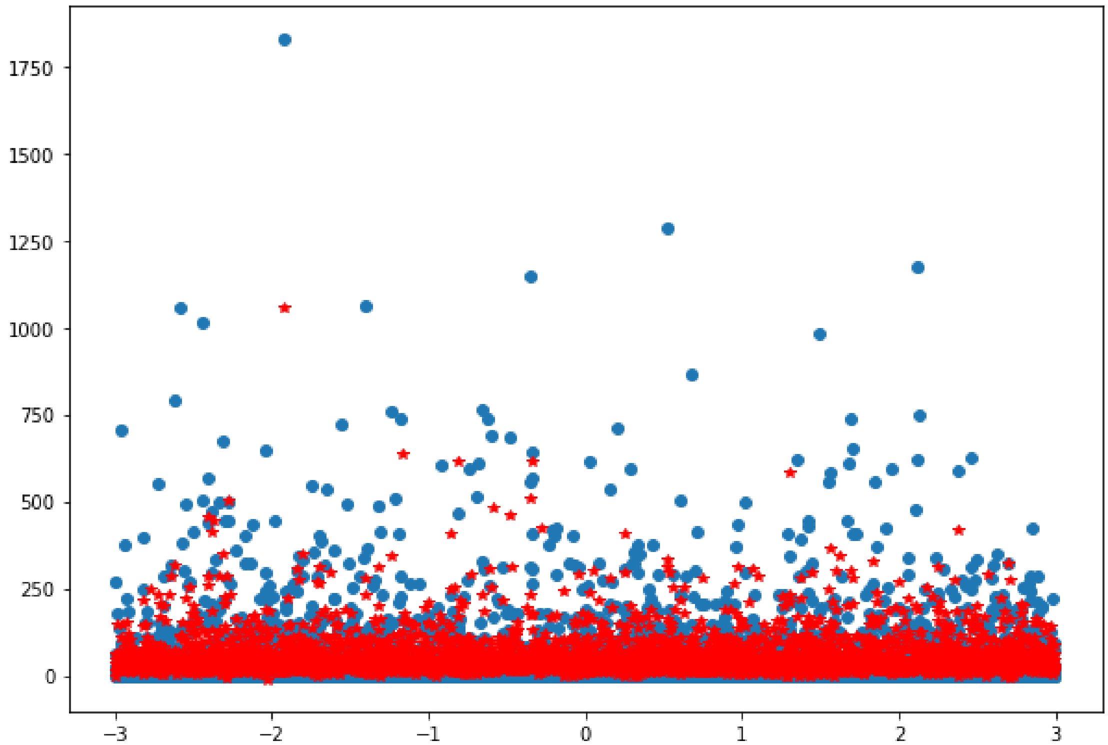
(1) Creditlevelasbuyer

I firstly conduct the correlation test on it and get the result in Figure (2), then I select highly related variables involving TENCENT score, GAODE score, HUABEI balance and HUABEI amount and conduct a linear regression on the data.

tencentscore	-0.187269
gaodescore	-0.152636
areaid	-0.081376
provincecode	-0.080796
uid	-0.066306
age	-0.052439
max_default_days	-0.043282
delaydate_max	-0.031309
instalments_num	-0.020504
nominalrates	0.006458
id	0.007998
highcontact	0.018981
highcontact20s	0.022421
apptimes	0.031294
yuebaobalance	0.031996
alipaybalance	0.034487
numbercontact20s	0.042521
taobaodealno	0.043939
numbercontact	0.049428
birthday	0.052443
apply_reject_sum	0.064167
apply_request_sum	0.077006
numbercontacttotal	0.092044
instalments_amount	0.101911
deal	0.129605
repay_fail_sum	0.134941
loan_offer_sum	0.160636
gender	0.218806
huabeibalance	0.287538
huabeiamount	0.503977
creditlevelasbuyer	1.000000

Table 2-1 Correlation test on credit score

Nevertheless, the regression result is fairly pleasant compared to using average value. Although due to its high variance the regression result can only cover part of special values, it is till has better performance than average number.



Graph 2-1 Regression result on Credit score

(2) Default

Default variable will work as dependent variable later, since its data type is Boolean, we create dummy variables for default.

3.Logit regression on single variables

(1) Default vs Credit score.

There are above half of the Default records are missing. Because it is the dependent variable, we just simply drop the missing records.

We run logit regression on three credit scores and default variable.Then we get the result:

	CREDITLEVELSBUYER	TENCENT SCORE	GAODE SCORE
COEF	-0.0004	0.0092	1.9983
P-VALUE	0.357	0.001	0.001

Table 3-1 Default likelihood regression

From the result we can come to a conclusion that, “creditlevelsbuyer” is useless since its p-value is up to 0.357. At the same time the “tencentscore” and “gaodescore” have a low p-value. And their coefficient is more than 0. It indicates that when someone has a higher score he is more likely to default.

(2) Deal vs Credit score.

We conduct logit regression on three types of credit score and deal variable. Different from the result on default, “creditlevelsbuyer” now has a low p-value, with a coefficient at 0.0030.

	CREDITLEVELSBUYER	TENCENT SCORE	GAODE SCORE
COEF	0.0030	-0.0317	-2.9408
P-VALUE	0.000	0.000	0.000

Table 3-2 Deal likelihood regression (credit score)

To illustrate the result better, I illustrate odds and with standardized credit score. From the line, we learn that when the individual’s credit score is higher, his application for loan is more likely to be approved. Further, compared its influence on the likelihood of default, it obviously influenced the likelihood of approval much stronger, ranging from odds at 0 to 100.

Graph 3-1 Odds -standard credit score

(3) Deal vs frequent contact.

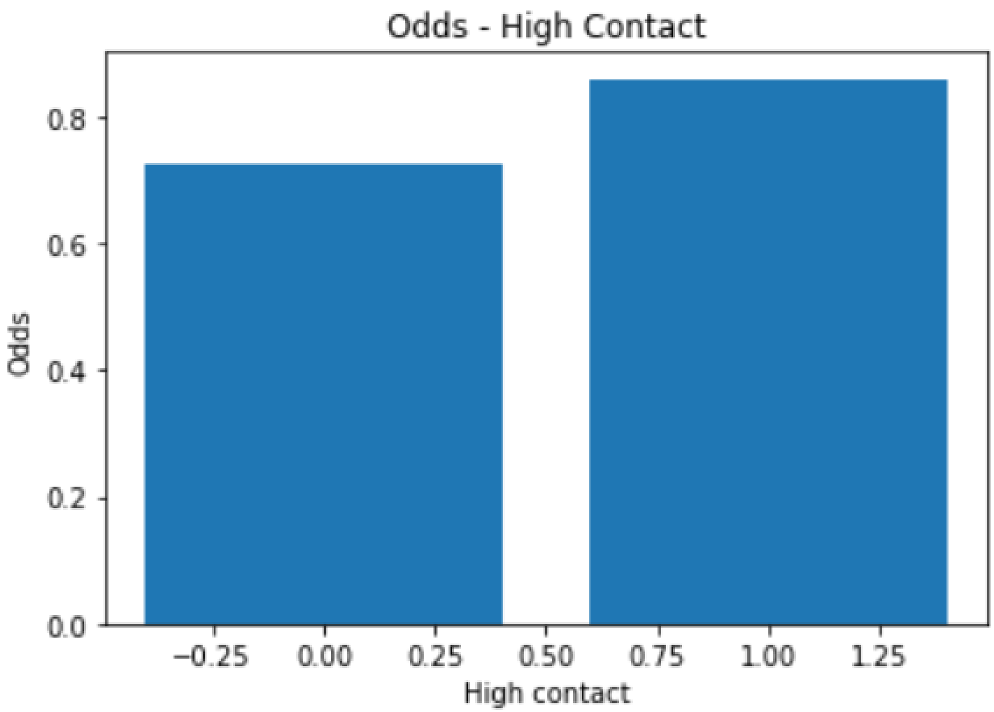
There are two types of values (True/False) in the contact variable and true value stands for the object has a frequent contact. Based on that, we only need to create one dummy variable for contact variable (0: False, 1: True).

After we run the logit regression on these two variables, we got the coefficient 0.1679, and a p-value at 0.003. Therefore, when an individual has a high frequency of contact, the likelihood of approval of his loan would be higher.

	HIGH-CONTACT
COEF	0.1679
P-VALUE	0.003

Table 3-3 Deal likelihood regression (High-contact)

According to the line, we can learn that when some has a frequent contact his loan is slightly more likely to be approved. Nevertheless, two value of odds is less than 1 which means their loans is more likely to be rejected. In other words, contact frequency only has small connection with the likelihood of loan approval.



Graph 3-2 Odds -High contact

4.Logit regression on multiple variables

The two logit regression models on single variable both end up at a pretty low score. Therefore, I then involve more variables to enhance the model.

MODEL	SCORE
DEAL~CREDIT SCORE	0.5794
DEAL~HIGH-CONTACT	0.5586

Table 4-1 Model score on single feature

Since there are many variables are available in the data sample, I simply run a correlation test on Deal in the sample. Then I pick variables which has high absolute value of correlation. To be more specific I pick variables rank high in absolute value of correlation and reaches at least 0.1, including:

“apptimes”,”huabeiamount”,”huabeibalance”,”nominalrates”,”credit_score”,”tencentscore”,”gaodescore”.

tencentscore	-0.213409
gaodescore	-0.105925
instalments_amount	-0.096088
id	-0.071709
apply_request_sum	-0.065933
apply_reject_sum	-0.062687
age	-0.056150
highcontact_False	-0.041664
loan_offer_sum	-0.032331
areaid	-0.031990
provincecode	-0.031802
repay_fail_sum	-0.019168
instalments_num	-0.004299
taobaodealno	0.002084
numbercontacttotal	0.005232
numbercontact20s	0.018830
numbercontact	0.022857
delaydate_max	0.025641
max_default_days	0.035355
highcontact20s	0.035964
yuebaobalance	0.040148
alipaybalance	0.040259
highcontact	0.041664
highcontact_True	0.041664
birthday	0.056075
gender	0.076812
credit_score	0.116620
creditlevelasbuyer	0.129605
nominalrates	0.131284
uid	0.215884
huabeibalance	0.231817
huabeiamount	0.231949
apptimes	0.237908
deal	1.000000

Name: deal, dtype: float64

Table 4-2 Correlation test on Deal

Next, I run the logit regression on the sample, the model has much better performance, rating 0.7016. The coefficients are shown in the table. According to the figure, the p-value of credit score is significantly big, so I replace it with another variable to improve the model.

	coef	std err	z	P>|z|	[0.025	0.975]
apptimes	1.4052	0.097	14.489	0.000	1.215	1.595
huabeiamount	9.185e-05	2.49e-05	3.688	0.000	4.3e-05	0.000
huabeibalance	0.0004	4.56e-05	9.587	0.000	0.000	0.001
nominalrates	3.3752	0.365	9.243	0.000	2.660	4.091
credit_reg	-6.987e-05	0.000	-0.169	0.866	-0.001	0.001
tencentscore	-0.0257	0.002	-10.831	0.000	-0.030	-0.021
gaodescore	-1.1375	0.437	-2.604	0.009	-1.994	-0.281
intercept	-1.3226	0.224	-5.910	0.000	-1.761	-0.884

Table 4-3 Deal likelihood regression on multiple features

To pick new variable in variable left, I run a correlation test on the most related variable “apptimes”. I then use the variable which is least relevant to “apptimes” to involve new data angle into the regression. Therefore, I replace “credit score” the useless variable with “gender” the least relevant variable to “apptimes”

id	-0.151790
highcontact_False	-0.127804
tencentscore	-0.068489
max_default_days	-0.059736
age	-0.040922
uid	-0.036182
instalments_amount	-0.023859
gaodescore	-0.012016
yuebaobalance	-0.001816
gender	0.001985
areaid	0.005894
provincecode	0.006104
alipaybalance	0.014274
instalments_num	0.014320
loan_offer_sum	0.016115
huabeibalance	0.025768
repay_fail_sum	0.026974
credit_score	0.028913
creditlevelasbuyer	0.031294
huabeiamount	0.033045
apply_reject_sum	0.039026
birthday	0.040825
apply_request_sum	0.043497
nominalrates	0.061899
numbercontacttotal	0.070455
delaydate_max	0.079056
highcontact20s	0.126409
highcontact	0.127804
highcontact_True	0.127804
deal	0.237908
taobaodealno	0.368402
numbercontact20s	0.404202
numbercontact	0.435978
apptimes	1.000000

Table 4-4 Correlation test on app-times

After I rerun the logit regression with new features. To control the number of features, I replace the useless feature (“credit score”) with new feature (“Gender”). And the model improves with score reaching 0.714.

MODEL	SCORE
MODEL ~ (CREDIT-SCORE```)	0.7016
MODEL ~(GENDER```)	0.714
DIFFERENCE	0.0124

Table 4-5 Model scores comparison

The new coefficient result:

	coef	std err	z	P>|z|	[0.025	0.975]
apptimes	1.4166	0.097	14.538	0.000	1.226	1.608
huabeiamount	8.602e-05	2.29e-05	3.748	0.000	4.1e-05	0.000
huabeibalance	0.0004	4.58e-05	9.719	0.000	0.000	0.001
nominalrates	3.3805	0.366	9.226	0.000	2.662	4.099
gender	0.4827	0.088	5.462	0.000	0.309	0.656
tencentscore	-0.0261	0.002	-11.002	0.000	-0.031	-0.021
gaodescore	-1.0601	0.438	-2.418	0.016	-1.919	-0.201
intercept	-1.4013	0.224	-6.264	0.000	-1.840	-0.963

Table 4-6 Deal likelihood regression on replaced features

Now we found all variables included in the model are grouped with a low p-value. And the model peaks the score at 0.714.

5.Appendix

Content

* [1. Data Summary](#1)

* [2. Logit Regression for one variable](#2)

  * [2.1 Default vs Credit score](#2.1)

  * [2.2 Deal vs Credit score](#2.2)

  * [2.3 Deal vs Contact](#2.3)

  * [2.4 logistic regression with multiple variables](#2.4)

```
import pandas as pd

import numpy as np

import math

import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression

from sklearn.linear_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

from sklearn.pipeline import Pipeline

import statsmodels.api as sm

import statsmodels.formula.api as smf
```


1.Data Summary

```
data = pd.read_csv(r"C:\Users\15161\Jupyter_git\Fin-Tech\Assignment1\Input\MDS6212 Week 1 Data.csv",encoding="gbk")

data.head()
```

```
data.shape
```

```
data.info()
```

```
## Convert gender to 0/1
data["gender"][data["gender"]==False] = 0
data["gender"][data["gender"]==True] = 1

data["gender"].value_counts()

data.describe()

data = pd.get_dummies(data,columns=["highcontact"])
## Convert default to 0/1
data["default"][data["default"]==False] = 0
data["default"][data["default"]==True] = 1
data['highcontact'] = data["highcontact_True"]
data["default"] = data["default"].astype("float64")

data[["age","gender","instalments_amount","nominalrates","creditlevelasbuyer","tencentscore","gaodescore","highcontact","default","deal"]].describe()

data.corr()['creditlevelasbuyer'].sort_values()

data[["gaodescore","tencentscore","huabeiamount","huabeibalance","gender"]].describe()

clf = LinearRegression()
Y = data["creditlevelasbuyer"][data["creditlevelasbuyer"].notnull()]
X = data[["gaodescore","tencentscore","huabeiamount","huabeibalance"]][data["creditlevelasbuyer"].notnull()]
clf.fit(X,Y)

clf.score(X,Y)
```

```
x = np.random.uniform(-3,3,size=len(Y))

y_pre = clf.predict(X)

plt.figure(figsize=[10, 7])

plt.scatter(x,Y)

plt.plot(x,y_pre,"r*")

plt.show()


data["credit_reg"] =data["creditlevelasbuyer"]


X_P =
data[["gaodescore","tencentscore","huabeiamount","huabeibalance"]][data["creditlevelasbuyer"].is
null()]

data["credit_reg"][data["creditlevelasbuyer"].isnull()] = clf.predict(X_P)


## Using poly regression to fill na value in credit score

# model = Pipeline(

# [

#     ('poly',PolynomialFeatures(degree=3)),

#     ('linear',LinearRegression(fit_intercept=False))

# ]

# )

# model =model.fit(X,Y)



# model.score(X,Y)


# x = np.random.uniform(-3,3,size=len(Y))

# y_pre = model.predict(X)

# plt.figure(figsize=[10, 7])
```

```
# plt.scatter(x,Y)

# plt.plot(x,y_pre,"r:*")

# plt.show()


<a id="2"></a>

## 2Logit Regression


<a id="2.1"> </a>

### 2.1 Default vs Credit score


## drop null value records in default since it is the target variables

data_default = data[data['default'].notnull()]

data_default.shape


## Fill the void with mean

data_default['creditlevelasbuyer'][data_default['creditlevelasbuyer'].isnull()] =
data_default['creditlevelasbuyer'].mean()

## Fill the void with other related variables

# data_default.corr()['creditlevelasbuyer']


## Defin X and Y

Y = data_default['default']

X = np.array(data_default['creditlevelasbuyer']).reshape(-1,1)


Y = Y.astype('int')


lg = LogisticRegression()

lg.fit(X,Y)


lg.score(X,Y)
```

```
lg.coef_

X_sort = data_default['creditlevelasbuyer'].sort_values()
X_sort = np.array(X_sort).reshape(-1,1)

P_1 = lg.predict_proba(X_sort)
P_1

odds =[]
for i in range(len(P_1)):
    odds.append(P_1[i][1]/P_1[i][0])
odds

plt.plot(X_sort,odds)
plt.ylabel("Odds")
plt.xlabel("CreditScore")
plt.title("Odds(Default) - CreditScore")
plt.show()

data_default['intercept'] = 1.0

# model = sm.GLM.from_formula("default ~ creditlevelasbuyer", family = sm.families.Binomial(),
data = data_default)
model = sm.Logit(data_default['default'].astype('int'),data_default[['creditlevelasbuyer','intercept']])
result = model.fit()
result.summary()

model = sm.Logit(data_default['default'].astype('int'),data_default[['gaodescore','intercept']])
result = model.fit()
```

```
result.summary()

model = sm.Logit(data_default['default'].astype('int'),data_default[['tencentscore','intercept']])
result = model.fit()
result.summary()

model = sm.Logit(data_default['default'].astype('int'),data_default[['gaodescore','intercept']])
result = model.fit()
result.summary()
```


2.2 deal vs Credit score

```
## Fill na
data['credit_score'] = data['creditlevelasbuyer']
data['credit_score'][data['credit_score'].isnull()] = data["credit_score"].mean()

Y2 = data["deal"]
X2 = np.array(data["credit_reg"]).reshape(-1,1)

data['intercept'] = 1.0

model = sm.Logit(data['deal'].astype('int'),data[['credit_reg','intercept']])
result = model.fit()
result.summary()

model = sm.Logit(data['deal'].astype('int'),data[['tencentscore','intercept']])
result = model.fit()
result.summary()
```

```
model = sm.Logit(data['deal'].astype('int'),data[['gaodescore','intercept']])
result = model.fit()
result.summary()

lg2 = LogisticRegression()
lg2.fit(X2,Y2)

lg2.score(X2,Y2)

lg2.coef_

X2_sort = data["credit_reg"].sort_values()
## standardlize
X2_plot = (X2_sort-X2_sort.min())/(X2_sort.max()-X2_sort.min())
X2_sort = np.array(X2_sort).reshape(-1,1)

P_2 = lg2.predict_proba(X2_sort)
P_2

odds =[]
for i in range(len(P_2)):
   odds.append(P_2[i][1]/P_2[i][0])
odds

plt.plot(X2_plot,odds,"b:*")
plt.ylabel("Odds")
plt.xlabel("Standard_CreditScore")
plt.title("Odds - CreditScore")
plt.show()
```


2.3 Deal vs Contact

See if there is null value in column contact

```
data["highcontact"].value_counts(dropna=False)
```

```
X3 = np.array(data["highcontact"]).reshape(-1,1)
Y3 = data["deal"]
```

```
model = sm.Logit(data['deal'].astype('int'),data[['highcontact','intercept']])
result = model.fit()
result.summary()
```

```
lg3 = LogisticRegression()
lg3.fit(X3,Y3)
```

```
lg3.score(X3,Y3)
```

```
lg3.coef_
```

```
P_3 = lg3.predict_proba(X3)
P_3
```

```
odds =[]
for i in range(len(P_3)):
    odds.append(P_3[i][1]/P_3[i][0])
odds
```

pd.Series(odds).describe()

X_contact = [0,1]

Y_Odds = [odds[0],odds[1]]

not useful,only value < 1 means with only high contact, no matter what its value is it will predict it as not a fail deal

plt.bar(X_contact, Y_Odds)

plt.ylabel("Odds")

plt.xlabel("High contact")

plt.title("Odds - High Contact")

plt.show()

2.4 Logist regression with multiple variables

data.corr()["deal"].sort_values()

take variables: tencent score,gaodescore,apptime,huabeiamount,huabeibalance,nominalrates,credit_score

data["nominalrates"][data["nominalrates"].isnull()] = data["nominalrates"].mean()

Y = data["deal"]

X = data[["apptimes","huabeiamount","huabeibalance","nominalrates","credit_reg","tencentscore","gaodescore"]]

```
model =
sm.Logit(data['deal'].astype('int'),data[["apptimes","huabeiamount","huabeibalance","nominalrates"
,"credit_reg","tencentscore","gaodescore",'intercept']])

result = model.fit()

result.summary()


lg4 = LogisticRegression()

lg4.fit(X,Y)


lg4.score(X,Y)


lg4.coef_[0]


index =
["apptimes","huabeiamount","huabeibalance","nominalrates","credit_score","tencentscore","gaode
score"]


model =
sm.Logit(data['deal'].astype('int'),data[["apptimes","huabeiamount","huabeibalance","nominalrates"
,"gender","tencentscore","gaodescore",'intercept']])

result = model.fit()

result.summary()


Y = data["deal"]

X =
data[["apptimes","huabeibalance","huabeibalance","nominalrates","tencentscore","gaodescore","g
ender"]]

lg = LogisticRegression()

lg.fit(X,Y)

lg.score(X,Y)
```