Tracfone-Trac·Hack Machine Learning Competiton

Introduction

This article aims to predict and analyze the user replacement needs of TracFone, an American telecom operator, based on the user's past usage data and mobile phone information.

A brief introduction to TracFone - TracFone was established in 1996, and at the time, was a pioneer in the US market by offering mobile phones and plans without extensive credit requirements. This allowed smartphones to become much more accessible to the wider community. Since then, their focus has always been on providing the highest quality coverage and access for all.

The following are the raw data that can be used:

1. The line id and observation date, as well as whether the user changed their phone on the observation date.
2. The user's personal information, including but not limited to line id, operator, first activation date, plan name, sub-plan name and redemption_date
3. User phone information, including but not limited to the type of the user's phone (feature phone, smart phone, etc.), phone model, phone release year, number of CPU cores, whether there is WiFi, whether there is Bluetooth, whether it supports touch, etc.
4. User usage: plan redemption channel, deactivation date, reactivation_date, suspension_start_date, suspension_end_date, whether the customer has participated in the customer loyalty program, customer loyalty program points.

It is worth noting that it is not advisable to directly substitute the raw data obtained as parameters into the model. This is because the data needs to be processed, with different methods for different data types.

In conclusion, our team chose about 20 features to be substituted into the six major models, and chose the random forest classifier as the final model as it produced the highest accuracy and F1 score.

Exploratory data analysis and pre-process

Part 0:Import the required python module and data

After importing both the training and prediction sets, to ensure the accuracy of the model, we had to confirm that they have similar distributions.
We import the following files from 's3://tf-trachack-data/212/dev':
upgrades,customer_info,phone_info,lrp_points,network_usage_domestic,redemptions,lrp_enrollment. And the same copy from 's3://tf-trachack-data/212/eval'

Part 1:Data usage

From the user's mobile phone usage habits and business perspective, we can associate a user's usage data with their willingness to upgrade their mobile phone.

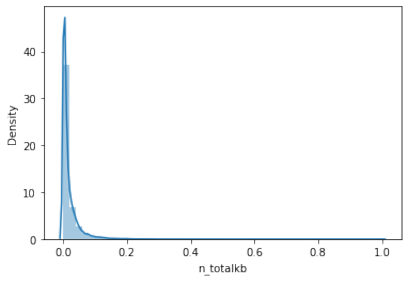
The table network_usage_domestic provides us with the user's daily usage. In order to observe the user's traffic usage from a macro perspective, we need to know the total data usage of each user, by summing the customer's daily data usage.

```
sum_network_in_function=df.groupby(['line_id'])['total_kb'].sum()
sum_network_in_function=sum_network_in_function.to_frame().reset_index()
print(sum_network_in_function)

max_kb=sum_network_in_function['total_kb'].max()
min_kb=sum_network_in_function['total_kb'].min()
sum_network_in_function['n_totalkb']=(sum_network_in_function['total_kb']-
min_kb)/(max_kb-min_kb)

sns.distplot(sum_network_in_function['n_totalkb'])
plt.show()
```

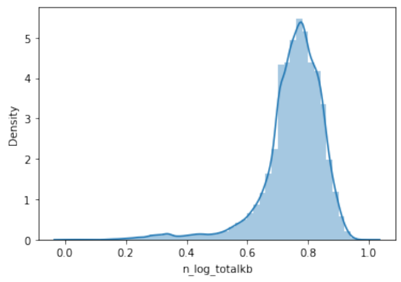
After adding and normalizing the data, we plot using the seaborn module to get:



From the plot we can see that most of the data is concentrated around 0, and thus, the data cannot be used meaningfully without further processing. In this case, we take a logarithm of the entire non-zero data.

```
sum_network_in_function=sum_network_in_function[sum_network_in_function['total_
kb']>0]
max_log_kb = np.log(sum_network_in_function [ 'total_kb' ].max ())
min_log_kb = np.log(sum_network_in_function [ 'total_kb' ].min ())
sum_network_in_function [ 'n_log_totalkb' ] = ((np.log(sum_network_in_function
[ 'total_kb' ]) - min_log_kb) / (max_log_kb - min_log_kb))

sns.distplot(sum_network_in_function [ 'n_log_totalkb' ])
plt.show()
return sum_network_in_function
```



The distribution of the data presented in the figure above is roughly a left-skewed normal distribution.

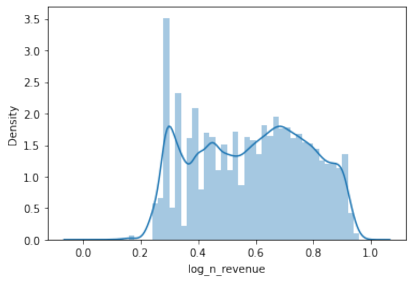
At the same time the distribution of the predicting set data looks almost the same.

Part 2:Total Revenue

Consistent with part 1, we believe that the more users spend on mobile phones, the stronger the user's reliance on mobile phones and thus the greater the probability of upgrading their mobile phones.

Using the same method, we can get the following results.

The distribution of the training set data is as follows:

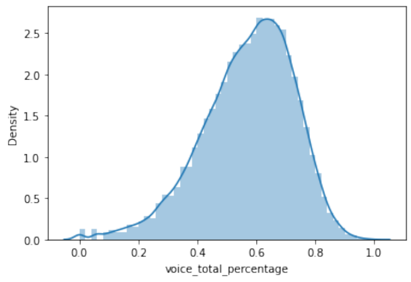


At the same time the distribution of the predicting set data looks almost the same.

Part 3:The total time of vocie time

The overall processing method is the same as the first and second part. Here we consider the total of the incoming and outgoing call times as the phone usage time.

The distribution of the training set data is as follows:

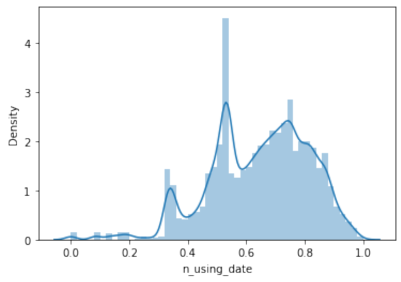


At the same time the distribution of the predicting set data looks almost the same.

Part 4:Using date

Generally, for ease of calculation, we believe that the length of time a user has joined TracFone is the difference between the observtion day and the first activation day in the upgrades table (we are not considering the impact of deactivation and subsequent reactivation etc.).

The distribution of the training set data is as follows:



At the same time the distribution of the predicting set data looks almost the same.

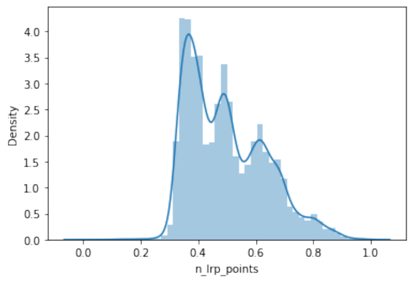
Part 5:loyalty reward program

5.1 Enrollment or not

We also need to consider another situation, whereby the user has changed telecommunications operators, which means that the customer no longer uses TracFone's phone and services.

To counter this, TracFone has launched a loyalty rewards program in order to improve customer loyalty. Thus, we believe that whether a customer upgrades their mobile phone with TracFone can be associated with whether they are a part of the loyalty rewards program.

The distribution of the training set data is as follows:

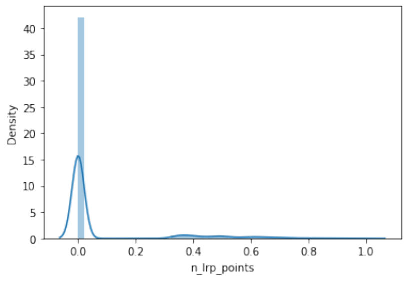


At the same time the distribution of the predicting set data looks almost the same.

5.2 Enrollment point

We also noticed that in the training set, only about 8,000 customers had joined the loyalty rewards program, which means that about 40% of users have not yet joined. In constructing the model, we must take all users into consideration, so members who have not joined the loyalty rewards program are assigned 0 points.

The distribution of the training set data is as follows:



At the same time the distribution of the predicting set data looks almost the same.

Part 6:The release year of the phone

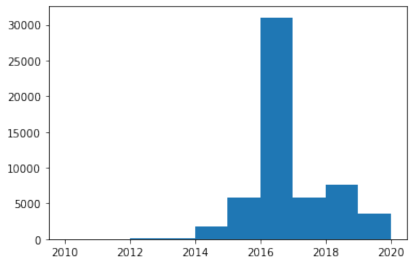
With the development and rise of mobile Internet, mobile phones (especially smart phones) are developing faster and faster. In many cases, smartphones released 2 to 3 years ago can't even run today's software smoothly.
Therefore, we believe that the number of years that a user has used thire mobile phone will also affect their decision on whether to upgrade their mobile phone or not.
Since there is no activation date of the user’s mobile phone in the data, we think that the release year of the mobile phone is roughly the year when the user purchased the mobile phone.
One thing to note here is that the years cannot be directly compared (for example, we can’t say that 2020 is bigger than 2010), but we believe the probability of a user upgrading is lower if they use phones released more recently.

So after processing the data, this comparison still makes sense.

While processing the data, we found that there were many blank data points in the mobile phone release year (not only the release year, but also other mobile phone information).
When faced with such data, there are several popular methods, such as deleting blank rows or filling in missing data.
By deleting blank rows, about 40% of the mobile phone information data would be missing. As a result, by using this amended dataset, the accuracy of the model will be drastically reduced. In addition, when predicting the results, the model will also return an error due to missing values in the predicting set.
Here we use the multiple-value filling method - among the existing data, the probability of occurrence in 2016 is the highest, so we treat all blank data as 2016. Although this would increase bias and thus distort the accuracy of the model, we think these deviations would be acceptable.

We used 2010 as the benchmark. The calculation method for the user’s mobile phone score is the year the mobile phone was released minus 2010. For example, the score of the mobile phone released in 2016 is 2016-2010 = 6 points.

The distribution of the training set data is as follows:



At the same time the distribution of the predicting set data looks almost the same.

Part 7: The carrier and plan_name

There are carrier and plan_name data in the customer_info table, which can also be used as two major features.
It should be noted that there are three carriers - carrier1, carrier2, and carrier3, but these three numbers cannot be compared. For example, if we mark males as 1 and females as 0, the model will mistakenly think that "males are bigger than females", but this is not logically true. Therefore, we need to mark male as {1,0} and female as {0,1}, so that the machine can correctly recognize its meaning.
In the same way, carrier1, carrier2, and carrier3 need to be marked as {1,0,0}, {0,1,0}, {0,0,1} respectively. plan_name needs to be marked as {1,0,0,0},{0,1,0,0},{0,0,1,0},{0,0,0,1}.

In order to achieve the above functions, get_dummies in pandas will be used.

```
def customer_plan_in_function(customer_info):
    customer_plan=customer_info.loc[:,
['line_id','carrier','plan_name']].drop_duplicates()
    customer_plan['carrier'].fillna(customer_info['carrier'].mode()[0],
inplace=True)
    customer_plan['plan_name'].fillna(customer_info['plan_name'].mode()[0],
inplace=True)
    customer_plan=pd.get_dummies(customer_info,columns=
['carrier','plan_name'],drop_first=False)

    print(customer_plan)
    return customer_plan

customer_plan=customer_plan_in_function(customer_info)
customer_plan_eval=customer_plan_in_function(customer_info_eval)
```

Part 8:Merge the table all togther

We need to merge all the tables generated above into a complete table for the following modeling and input feature values.

Part 9： Advanced process on the merged table

In the process of merging the tables, we found two interesting features: n_using_date and n_totalkb, which represent the length of the user’s mobile phone use and the total data usage, respectively. It is not difficult to find that the two are positively correlated. Therefore, by dividing the original data of the data flow and the number of days of use, the user's daily data usage can be obtained, which can more accurately reflect the user's demand.

Establish the model and choose the future

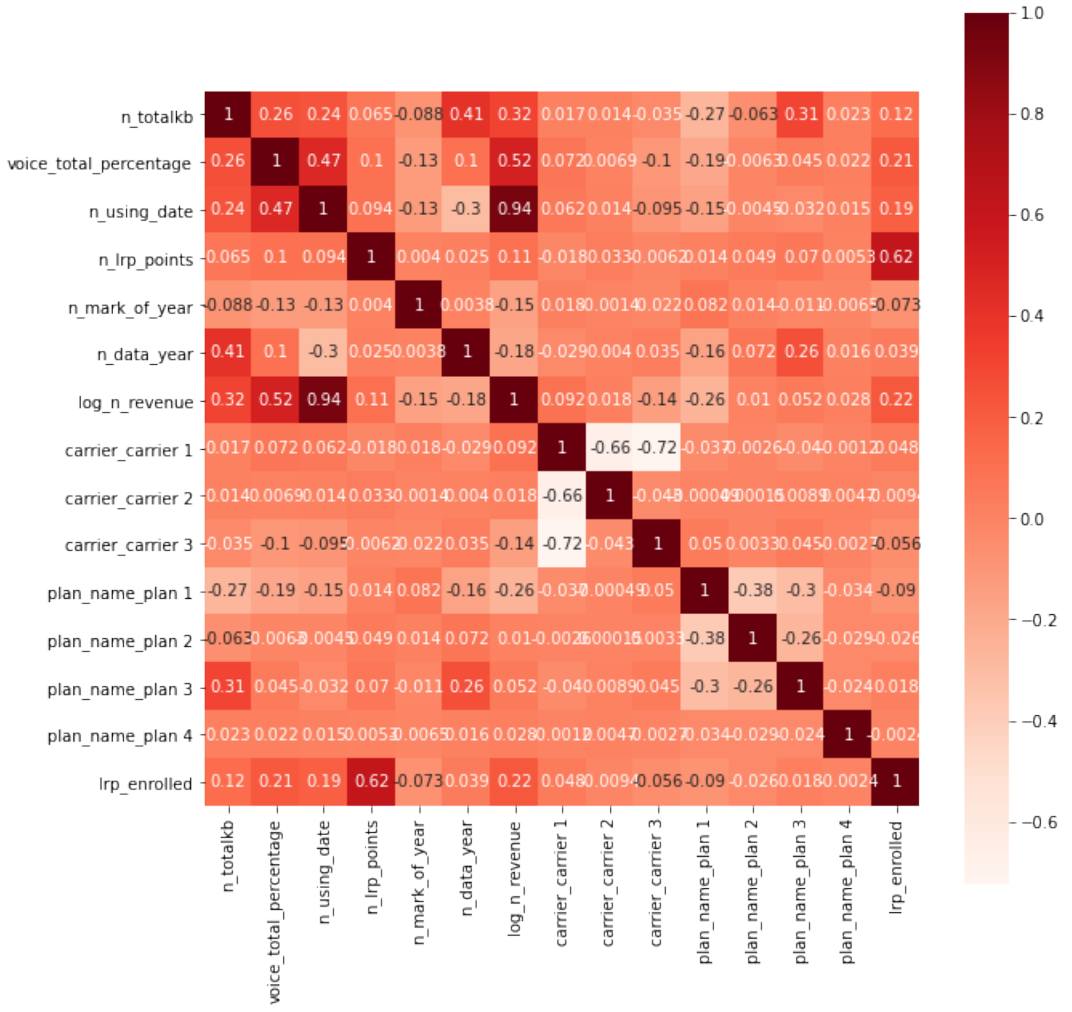
Intuitively, we will use all the features extracted above.

```
x=dev_merge1.loc[:,
['n_totalkb','voice_total_percentage','n_using_date','n_lrp_points','n_mark_of_
year','n_data_year','log_n_revenue','carrier_carrier 1','carrier_carrier
2','carrier_carrier 3','plan_name_plan 1','plan_name_plan 2','plan_name_plan
3','plan_name_plan 4','lrp_enrolled']]
y=dev_merge1.loc[:,['yesorno']]
```

However, when the correlation between features is too high, we believe that the accuracy of the model will be affected to a certain extent. So we need to know the correlation of all features in the model, and then decide whether to exclude a feature according to the level of correlation to make the model fit better.

```
fig, ax = plt.subplots ( figsize=(10, 10) )
sns.heatmap(x.corr(),annot=True, vmax=1, square=True, cmap="Reds")

plt.show()
```



In the above figure, there is a data that has caused our ism: log_n_revenue, which has a correlation greater than 0.5 with two sets of data, namely the call time and the date of use, which is also understandable from the perspective of common sense. However, in order to better fit the model, we think it is removed from the feature.

```
new_x=dev_merge1.loc[:,
['n_totalkb','voice_total_percentage','n_using_date','n_lrp_points','n_mark_of_
year','n_data_year','carrier_carrier 1','carrier_carrier 2','carrier_carrier
3','plan_name_plan 1','plan_name_plan 2','plan_name_plan 3','plan_name_plan
4','lrp_enrolled']]
y=dev_merge1.loc[:,['yesorno']]
```

Choose a appropriate model

We traverse all models and select the model with the highest accuracy and f1 score, including SVC, LogisticRegression, AdaBoostClassifier, RandomForestClassifier, DecisionTreeClassifier, MLPClassifier.

```
from sklearn.metrics import f1_score
list_operation=
[SVC(),LogisticRegression(),AdaBoostClassifier(),RandomForestClassifier(),Decis
ionTreeClassifier(),MLPClassifier()]
name_operation=
["SVC","LogisticRegression","AdaBoostClassifier","RandomForestClassifier","Deci
sionTreeClassifier","MLPClassifier"]
for i in range(6):

 Xtrain,Xtest,Ytrain,Ytest=train_test_split(new_x,dev_merge1['upgrade'],test_si
ze=0.5)
    operation=list_operation[i]
    operation.fit(Xtrain,Ytrain)
    accuracy=operation.score(Xtest,Ytest)
    f1=f1_score(Ytest, operation.predict(Xtest), labels=['yes','no'],
pos_label='yes')
    print("%s accuracy is %.3f and f1_score is %.3f "%
(name_operation[i],accuracy,f1))
```

In the case that all models choose the default hyperparameters, we have the following results:

SVC accuracy is 0.839 and f1_score is 0.667
LogisticRegression accuracy is 0.837 and f1_score is 0.656
AdaBoostClassifier accuracy is 0.847 and f1_score is 0.686
RandomForestClassifier accuracy is 0.874 and f1_score is 0.751
DecisionTreeClassifier accuracy is 0.833 and f1_score is 0.685
MLPClassifier accuracy is 0.852 and f1_score is 0.702

Obviously, the accuracy and f1 score of using RandomForestClassifier are higher than other models, so we choose it as our classifier.

```
test_model=RandomForestClassifier()
operation.fit(Xtrain,Ytrain)
scores=cross_val_score(estimator=operation,X=Xtrain,y=Ytrain,cv=10,n_jobs=1)
print("CV Accuracy Scores: %s\n" % scores)
```

Of course, we need to ensure that the model is not over-fitting. Cross-validation is used here to check whether the model is over-fitting. The effects of the model are as follows:

```
CV Accuracy Scores: [0.83728566 0.83905109 0.84525547 0.84379562 0.82481752
0.83065693
 0.84781022 0.8459854 0.84160584 0.83613139]
```

The accuracy does not change significantly with the sample, and we believe that the model is not overfitting.

Determine the hyper-parameter

For RandomForestClassifier, even though there are many parameters in the model, there are two most important parameters: n_estimators and max_features. We will use the GridSearchCV module to find out how to take the values of these two important hyperparameters.

The value of n_estimators

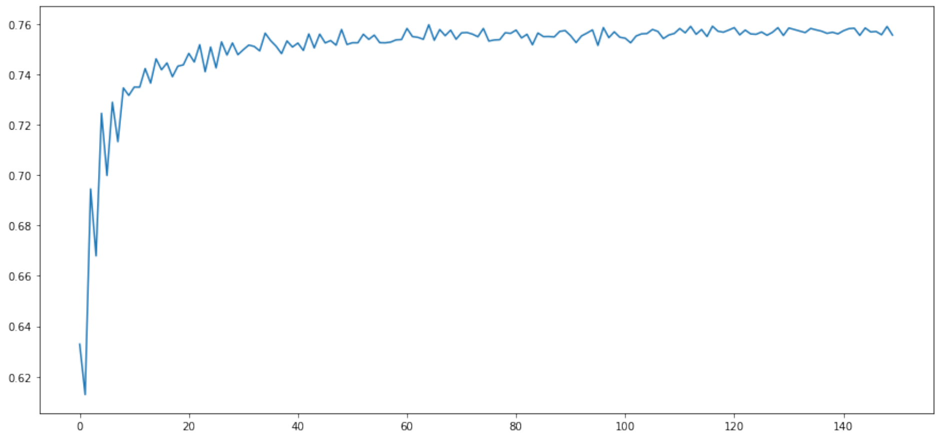
```
from sklearn.model_selection import GridSearchCV
param_test1 = {"n_estimators":range(1,201,10)}
gsearch1 =
GridSearchCV(estimator=RandomForestClassifier(),param_grid=param_test1,
                        scoring='f1',cv=10)
gsearch1.fit(Xtrain,Ytrain)

print(gsearch1.best_params_)
print("best accuracy:%f" % gsearch1.best_score_)
```

```
{'n_estimators': 200}
best accuracy:0.751528
```

We found that when n_estimators reached the maximum value of 200 in our specified range, the model got the highest f1_score. Is this because our value range is not large enough, or is it due to other reasons? Let's take a look at the trend graph of f1_score with n_estimators.

```
Xtrain,Xtest,Ytrain,Ytest=train_test_split(x,dev_merge1['upgrade'],test_size=0.
5)
score= []
for i in range(150):
    rfc = RandomForestClassifier(n_estimators= i+1)
    rfc.fit(Xtrain,Ytrain)
    f1=f1_score(Ytest, rfc.predict(Xtest), labels=['yes','no'],
pos_label='yes')
    score.append(f1)
plt.figure(figsize=(15,7))
plt.plot(score)
```



We found that when the value of n_estimators is greater than 100, f1_score basically does not change with the increase of n_estimators. So n_estimators takes the default value (if no parameters are passed in, the default is 100) or 200 has little effect on the model.We consider n_estimators to be 200 here.

The value of max_features

Using the same method as choosing the n_estimators,we can get the following code and results:

```
Xtrain,Xtest,Ytrain,Ytest=train_test_split(x,y,test_size=0.5)
param_test2 = {"max_features":range(1,15,1)}
gsearch2 =
GridSearchCV(estimator=RandomForestClassifier(n_estimators=200),param_grid =
param_test2,scoring='f1',cv=10)
gsearch2.fit(Xtrain,Ytrain)

print(gsearch2.best_params_)
print("best accuracy:%f" % gsearch2.best_score_)
```

```
{'max_features': 4}
best accuracy:0.746594
```

Final model, accuracy and its f1_score

Since we have decided the value of n_estimators and max_features,we can finally determine the used model and hyper-parameter as showed following:

```
test_model2=RandomForestClassifier(n_estimators=200,max_features=4)
test_model2.fit(Xtrain,Ytrain)
accuracy=test_model2.score(Xtest,Ytest)
print(accuracy)
f1=f1_score(Ytest, test_model2.predict(Xtest), labels=['yes','no'],
pos_label='yes')
print("accuracy is %.3f and f1_score is %.3f "%(accuracy,f1))
```

```
Accuracy is 0.880 and f1_score is 0.760
```

Prediction and output

```
test_model=RandomForestClassifier(n_estimators=200,max_features=4)
test_model.fit(Xtrain,Ytrain)

x_predict=eval_merge1.loc[:,
['n_totalkb','voice_total_percentage','n_using_date','n_lrp_points','n_mark_of_
year','n_data_year','carrier_carrier 1','carrier_carrier 2','carrier_carrier
3','plan_name_plan 1','plan_name_plan 2','plan_name_plan 3','plan_name_plan
4','lrp_enrolled']]1

eval_merge1['upgrades']=test_model.predict(x_predict)
eval_merge1['upgrade']=eval_merge1['upgrades'].replace({'yes':1,'no':0})
submission=eval_merge1.loc[:,['line_id','upgrade']]

root_folder='2021-04-25.csv'
submission.to_csv(root_folder,index=False)
```

Discussion and Conclusion

Discussion on different model

When determining the model, we found that the performance of different models varies greatly. Obviously, we found that RandomForestClassifier has a good classification effect, but at the same time the amount of calculation is significantly enlarged and the calculation time is long. When determining n_estimators and max_features later, the computer even needs to calculate from several minutes to tens of minutes.
In the case of sufficient time, we can choose to optimize the parameters one by one and wait for the computer to output the training results, but we can not do this in all cases.
In the case of sufficient time, we can choose to optimize the parameters one by one and wait for the computer to output the training results, but we can not do this in all cases.

For other classifiers, the characteristics are different:

1. Decision trees are easier to understand and deduct, but it is easy to overfit and ignore the correlation between data.
2. The neural network has high classification accuracy, strong parallel distributed processing ability, strong distributed storage and learning ability, but it is difficult to observe the process between learning and further deduct it.
3. Support vector machines can solve high-dimensional and nonlinear problems, but they lack sensitivity to data.
4. The naive Bayes classifier has a solid mathematical foundation, but it needs to know the prior probability and ensure that the features are independent of each other.
5. Adaboost is a classifier with high classification accuracy and simple algorithm, but the number of weak classifiers is difficult to determine.
6. And so on...

When faced with a wide variety of data sets with complex relationships, we should first perform exploratory data analysis, determine the relationship between the data and preprocess the data, and then select the appropriate model to classify to achieve the best classification effect .

Conclusion

After selecting the appropriate model and hyperparameters, the accuracy of the model reached 88%, and the f1_score reached 76%,which is basically satisfactory.
编不下去了。。。帮我编一点。。。

If we have more time, we will conduct exploratory data analysis on the unprocessed tables deactivation, reactivation, suspension, and redemption to find out the days and times of deactivation, reactivation, and usage habits of each user.. .. More features usually means higher accuracy.