## Loan Granting

### Background

Analyze loan data from a bank to decide whether grant a loan to the customer or not.

The bank has a model. The bank wants to improve its loan strategies, so it asks us to build a model better than the existing one.

The target variable is ‘if\_repay’, whether the customer will repay the loan successfully.

Data only includes the granted loans

bank model，只考虑发放贷款的情况。

Ratio of ‘able to repay’ to ‘unable to repay’ is about 2:1

#### 1.1Data statistics

Two tables: table ‘customer’ and table ‘loan’ merge之后，

Row: about 50,000

Column: 16

Memory usage: 6MB

### Question

#### 2.1 Are there any other variables that you'd like to include in the model?

Marriage status: his spouse can help to repay偿还.

housing status, own a house or not? Or rent an apartment? monthly rent amount?

#### 2.2 The important features on the prediction.

According to the figure, we can observe that the most important feature is ‘saving amount’. This makes sense. If he has more money in the saving account, he is more likely to repay.

For the feature ‘is\_employed’, since it is highly correlated with the feature ‘salary’. We just consider ‘salary’. We can observe that salary is very important for our model, so ‘is\_employed’ is also important. This makes sense. If he has a high salary, he is more likely to repay.

### Problem Definition

Compare the total profits of the two models (the new model uses an optimal threshold), we observe that the new model outperforms the original model.

LR, RF and LightGBM. LightGBM outperforms the others (auc value). Profit increased from 3400 to 6500

### Data Clean

Connect to MYSQL and review data using SQL statements. Sometimes, review the data using python.

#### 4.1 Inspect and deal with missing values

'repaying'（Is he currently repaying any other loans?）

'limit\_used'(The average ratio of monthly credit card payment to credit card limit last year.)

* For 'repaying', fill missing values with -1 (minus one).

Missing values occur because this is the customer’s first loan.

* for 'limit\_used', fill missing values with the median (to prevent decimals)

#### 4.2 Inspect and deal with outliers

There are negative values in some features. For instance, the negative saving amount that doesn’t make much sense. Let’s clean the data.

#### 4.4 format transform: date

#### 4.5 Rename

### Feature engineering

There are about 20 features, delete 8

#### 5.1 Transform categorical features into numerical features

Convert text or categorical values into numerical values

* categorical feature: doesn't have mathematical meaning, cannot calculate directly
* numerical feature may be either discrete or continuous.

The feature value is text, using OneHotEncoding(001,010, 100) or LabelEncoding (0, 1, 2)。OneHotEncoder will increase the number of features (the number of categorical feature’s values).

#### 5.2 Delete redundant features

Cause dimension curse, prone to overfitting, increase computational cost

Find redundant features intuitively+ plot a heatmap to show the linear correlation for each pair of features. (after filling missing values, categorical features--numerical features, then calculate the correlation matrix)

Each value in Correlation matrix is a pearson correlation coefficient, both 1 and -1 indicate a perfect relationship，0 indicates no relationship exists.

Delete feature ‘first\_loan’, because it is highly correlated with 'repaid' and 'repaying'.

Delete feature ‘is-employed’, because it is highly correlated with 'salary'.

#### 5.3 Remove features which are not related to the target variable

Delete id, date, granted, year, month, day

Find each feature’s relationship with the target variable by

* Pearson Correlation Coefficient (the linear relationship & regression)
* Chi-squared (the linear relationship & classification & sparse matrix)
* Tree-based feature selection: RF, LightGBM (nonlinear relationship)

For high dimension data, select the best k features based on their relationship with the target variable.

SelectKBest(chi2,k=3).fittransform(features,target) #output: chi2, p-value of each score

SelectKBest(f\_regression,k=3).fittransform(features,target) #output: F-scores, p-values of F-scores

sfm=SelectFromModel(RF, max\_features=3).fit(trainfeatures, traintarget)

#### 5.4 Create new features that may be related to the target variable

Extract features from dates, such as ‘year’, ‘month’, ‘day of the week’

Sum, difference, product, division

The most important thing may be to create new features that are related with the target variable. e.g. the illegal activity detection, create a new feature: the difference of the transaction time and the signup time.

#### 5.5 Overview the Feature distribution

numerical feature distribution: hist

categorical feature distribution: bar or pie (tableau)

### Modelling

#### 6.1 Check imbalanced classes

Inspect the distribution of the target variable, ‘if\_repay’.

For imbalanced classes, choose AUC as the evaluation metric for model selection (for different LightGBM models, choose the best model with the greatest AUC, different LightGBM models，只是参数不同)+find a better threshold

For balanced classes, choose ‘accuracy’+use the default threshold 0.5.

saving\_amount | card\_limit | salary | card\_limited | dependent\_number(亲属数量)| loan\_purpose | repaid | repaying |

#### 6.2 Choose the algorithm, LR, RF and LightGBM

Why lightGBM?

Robust against imbalanced classes

an improved GDBT (an ensemble method, sequence decision tree,) algorithm. **can handle categorical feature**, run fast, achieve high prediction accuracy.

#### 6.3 Split the data into training, validation and test datasets

We just split data into training75% and test datasets. This is because, LightGBM will split the training dataset into training and validation subsets automatically. Perform cross validation to choose the best model.

train\_features, test\_features, train\_target, test\_target = train\_test\_split(

features, target, test\_size=0.25, random\_state=42, stratify=target #保证分割后target variable的比例分布与原数据一致 )

#### 6.4 Train and get optimal parameters

lgb.cv(parameters, train\_set= train\_data, num\_boost\_round=1000, nfold=5,

early\_stopping\_rounds=20, seed=42, verbose\_eval=False)

Parameters selection: nfold=5 represent 4 training datasets and 1 test dataset

Evaluation metric, say AUC

If the performance isn’t improved (the auc value for the validation dataset) until ‘early\_stopping\_rounds’=20, the training process stops.

Get optimal parameters

lgb.cv会得到一个性能好（如采用AUC作为评价）的模型及其参数如best\_iteration

The index of iteration that has the best performance will be saved in the best\_iteration field if early stopping logic is enabled by setting early\_stopping\_rounds. Note that train() will return a model from **best number of trees**=767.

#### 6.5 Retrain the model using optimal parameters

clf = lgb.train(parameters, train\_set= train\_data, num\_boost\_round=767)

pred = clf.predict(test\_feature)

#### 6.6 Choose the best threshold for classification problem, for imbalanced classes,

For classification problem, two functions give the prediction

.predict() gives the label 0 or label 1 using default threshold 0.5

.predict\_proba() gives the probability of label 1.

How to find

* **Imbalanced classes**

find the best threshold instead of the default 0.5 to achieve a better prediction precision.

Obtain the recall and TPR values for validation set.

* Plot the ROC curve and find the point closest to the top left corner of the ROC curve.
* The point can also be determined by the intersection of one line with the ROC curve. The line is y=-x+1. The corresponding threshold is the best threshold.
* Draw a curve: thresholds vs prediction accuracy，find the corresponding threshold of the highest accuracy.
* **consider the cost of FP and FN**

For the target variable ‘if\_repay’, the cost of FP is higher than that of FN, so **choose a larger threshold** (less observations will be classified as positive), say 0.5. This is because if ‘not repaid’ is identified as ‘repaid’, the company will lose money. On the contrary, we can ask colleagues or domain experts for further investigation.

Choose the points around the best point. From them, choose a larger threshold.

The selected threshold is 0.44

#### 6.7 Get predicted labels using the best threshold

for pred in preds:

result = 1 if pred > best\_threshold else 0

AUC = [0.85, 0.95], 效果很好

AUC = [0.7, 0.85], 效果一般

#### 6.8 Visualize the feature importance

Plotting and show which features contribute much to the predictions.

lgb.plot\_importance(clf, ax=ax, height=0.5)

### Model performance analysis and choose the best model

Calculate the profits of the two models

* For original model: How to calculate ‘profit’? 3400

1. ‘if\_repay’ ==1 Profit+1
2. ‘if\_repay’ ==0 Profit-1

* For our model: how to calculate ‘profit’? sum 6500

1. ‘predicted\_repaid’ ==1 and ‘if\_repay’==1. Profit+1
2. ‘predicted\_repaid’ ==1 and ‘if\_repay’==0 Profit-1

Compare different models, generally we use AUC

AUC = [0.85, 0.95], 效果很好

AUC = [0.7, 0.85], 效果一般

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Accuracy | precision | Auc | Profit | threshold |
| LR | 0.83 | 0.846 | 0.92 | 5663 | default0.5 |
| LR+standardscaler()  和LRcv+standardscaler差不多 | 0.9 | 0.927 | 0.964 | 6495 | default0.5 |
| LR+standardscaler()+  Tuning threshold | 0.9 | 0.938 | 0.964 | 6518 | 0.55 |
| LightGBM+  Tuning paratemers | 0.92 | 0.972 | 0.978 | 6713 | 0.5 |
| LightGBM+  Tuning paratemers+  Tuning threshold | 0.925 | 0.942 | 0.978 | 6789 | profit0.3  roc 0.38 |
| LightGBM+  Tuning paratemers+  Tuning threshold | 0.91 | 0.979 | 0.978 | 6613 | 0.6  选择，这样更安全 |
| LightGBM+  Tuning paratemers+  Tuning threshold | 0.88 | 0.99 | 0.978 | 6248 | 0.8 |
| RF | 0.921 | 0.949 | 0.975 | 6732 | 0.5 |
| RF+  Randomizedsearch  Tuning paratemers | 0.922 | 0.955 | 0.976 |  |  |
| RF+  Tuning paratemers+  Tuning threshold |  |  |  | 6755 | profit最佳0.52  roc最佳0.5 |
| MLP | 0.92 |  | 0.975 |  |  |

RF+ Randomizedsearch+tuning parameters

6个参数，100个组合，3foldscv, 70min+

{'n\_estimators': 700, 'min\_samples\_split': 12, 'min\_samples\_leaf': 2, 'max\_features': 'sqrt', 'max\_depth': 13, 'bootstrap': False}

### MLP

和RF性能相当

batch\_size = 256, epochs=10, 一般我的数据集都比较小，不需要batch\_size

11 input features,

The input layer

3 hidden layers (16 nodes, 32 nodes, 16 nodes), activation = 'relu'

The output layer (1 node), activation = 'sigmoid'

The first hidden layer+dropout layer+the second hidden layer+dropout layer+ the third hidden layer+output layer

Dropuout layer to reduce overfitting, Dropout(0.2)

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

optimizer：loss优化函数，用 adam等

loss：常用的loss损失函数，二分类用binary\_crossentropy

**Word embedding**

method (word---vector)

traditional ML: TF-IDF

one-hot, k-hot

word embedding: cosin ---similarity between two words,比如用50长度表示一个单词

Glove比Word2Vec更高效

给每个单词初始化一个vector value，之后再给定的文本中比如“所有text message的数据”学习到vector value。

网络模型构建的时候，第一层是embedding层，这样neural network会自己从“短信数据”中学习并将每个单词转化为vector

model.add(layers.Embedding(10000,16,input\_length=200)) #“短信数据”中的10000个word map 到 长度为16的vector,每条短信的长度是200

每条语句填充成相同长度即200个单词

模型构建的时候word embedding

model.add(layers.Embedding(10000,16,input\_length=200))

对于MLP

比如2500条短信，填充每条短信200个单词，每个单词16个单位的vector,则embedding后，输出变为[2500,200,16]，三维不能作为传统NN模型的输入

model.add(layers.Flatten()) 将3维降为2维

model.add(layers.)

对于LSTM

LSTM是接收3维数据的，MLP是接收2维数据的

model.add(layers.LSTM(128)) #64也行

model.add(layers.Dense(1, activation='sigmoid')) #输出

## Suspicious transactions unsupervised (bank)

### Background

Analyze a dataset of credit card transactions and identify suspicious transactions.

We use the unsupervised algorithm k-means to solve this problem. Because many fraudulent transactions have been committed犯 before we can identify them.

#### 1.1Data statistics

Row: about 300,000

Column: 9 features

Memory usage: 11MB

12 features----4features

### Questions

#### 2.1 Once an account exceeds its credit limit, the bank notify him.

Each day, we return a list of account IDs which exceed their monthly credit limits on that day(刚好在那天). We send text messages to notify them. First the data is grouped by ID, then grouped by month for each account, calculate the amount for each account each month.

#### 2.2 Find all suspicious transactions

Choose k-means algorithm to find a cluster of suspicious transactions. For unsupervised algorithms, the key point is to find and create features which can perfectly separate the target data from the others. After data review, I notice that transaction location and transaction amount are such important features. Create such new features through feature engineering.

* **With regard to the transaction amount**

Any suddenly high transaction is more likely to be suspicious.

* **With regard to the transaction amount**

Any suddenly long distance transaction is more likely to be suspicious.

Then, visualize the data to choose an optimal k. in order to display the data in a 2D space, we need to reduce the dimensionality using PCA. Before PCA, we need to normalize the features.

features = StandardScaler().fit\_transform(features)

### Feature engineering

#### Create new features that may be related to the target variable

Sum, difference, product, division

* **With regard to the transaction location**

Distance: the distance between the transaction location and the transaction center of the customer 距离 (当前交易地点, 过去交易地点的均值中心the average longitude and the average latitude)

* **With regard to the transaction amount**

ratio\_amount\_ave: The ratio of current amount to the 50th percentile of the daily amount 当天交易金额/每天交易金额的中位数median

ratio\_amount\_75: The ratio of current amount to the 75th percentile of the daily amount 当天交易金额/每天交易金额的75百分位数

ratio\_amount\_creditlimit: The ratio of current amount to credit limit 当天交易金额/信用卡额度

Original features: 'amount', credit \_limit, 'Long', 'Lat', 'city', 'state', 'zipcode', date……

Keep useful features: distance | ratio\_amount\_50 | ratio\_amount\_75 | ratio\_amount\_creditlimit

### Modelling

#### 4.1 Choose the algorithm, k-means

Why k-means?

K-Means Clustering is a simple but powerful unsupervised algorithm.

#### 4.2 Find an optimal number of clusters, k

* **Visualize the data and choose k**

1. Normalize the features

Before implementing PCA, normalize all features.

because some feature such as ‘credit\_limit’ has a high variance.

scaler = StandardScaler()

features= scaler.fit\_transform(features)

1. Reduce the dimensionality to 2D using PCA.

# pca=PCA(n\_components=2)

Pca.fit\_transform(features)

1. Find the optimal k through visualizing the data in a 2 dimensional space.

Clearly, the figure shows there are about 5 or 6 clusters. We assign 6 to k.

* **Evaluate intra-cluster similarity and inter-cluster similarity by silhouette score.**

Normalize all features. ,因为数据降维后能明显看出k，否则会选用silhouette

Find the k whose silhouette score close to +1. 越接近1，聚类效果越好

for k in range(2,20):

labels = KMeans(n\_clusters=k).fit(features).labels\_

score = metrics.silhouette\_score(features, labels)

scores.append(score)

#### normalize all features, then Implement k-means clustering

features = StandardScaler().fit\_transform(features)

The target cluster has the smallest number of data.

110,000

70,000

…

5,000----obviously, the cluster has fewer data compared with others. This makes sense.

Look at the transactions in the target cluster. The transaction ‘amount’ is significantly higher than the customer’s 75th percentile of the daily payment amount. These suspicious transactions need further investigation.

Further investigation表明，我们的检测包括了100%的可疑交易，从300,000降低到5000, the data size has reduced by 84%

## Sentiment classification

评语，每条评语是一个句子，输出该句子是P还是N，二分类问题。

RNN: 对有时间先后顺序的数据进行处理（text, audio），参数少，不容易overfitting，现实生活中图片识别效果不好。

一次iteration=[batch size]个训练数据forward+backward后更新参数过程。

一次epoch=所有训练数据forward+backward后更新参数的过程。

因为一次跑完所有训练数据来更新参数可能太慢，如果所有数据量大，将数据分为多个部分，每个部分是batch size大小，每次iteration完成一次参数更新

* **Word to vector (string to number vector) 方法：word embedding**

**Word embedding 方法有**Word2Vec 和GloVe

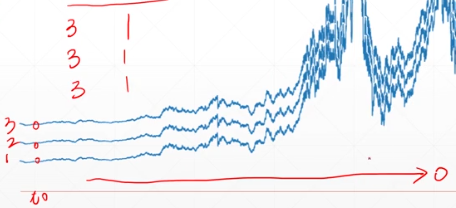
将每个word转化为3维空间（x, y, z）上的一个点，两个点近，则semantic similarity高

Text----[word number, batch, word vec] 即 [单词数量，句子数量，表示每个单词的一维向量的长度]

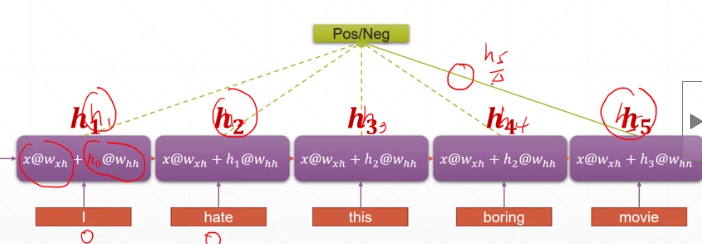
比如每个单词用3个单位的一维向量表示即[1,0,0]

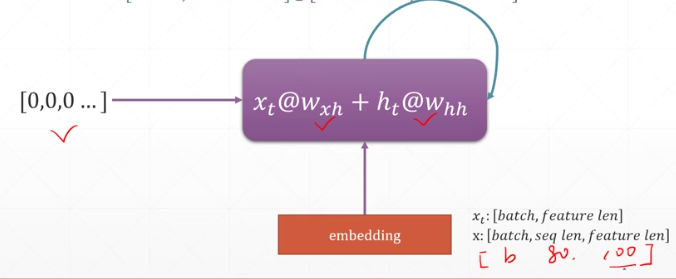
对于多条curve，可以这么理解，word number是横轴即n个时刻（n个word），b是曲线数量，word vec是曲线上的一个点.

比如在时刻t0，三条曲线，采样3个点



下面考虑单词输入顺序，最后得到的状态是h5，用h5来预测，每个part的权重相同，每个part是一个layer





## Ads optimization

The project is to analyze ads data for 40 products to measure ads performance, predict ads' future performance.

Each record includes the ad\_ID, the number of impressions, the number of clicks | the number of conversions | the avg\_cost\_per\_click | the total revenue | date.

ad\_ID | impressions | clicks | conversions | avg\_cost\_per\_click | total\_revenue | date

#### Identify the 5 best ads?

Choose metrics to evaluate the ‘best’. Select metrics depending on the main business focus.

We want to select the ads with high customer growth potential and high profit.

Thus, we onsider both the customer growth and the profit.

In view of the customer growth, I chose click-through-rate CTR=no. of clicks/ no. of impressions (how many times show)

In view of the profit, I created a new feature, profit per click=revenue- ad costs= revenue-no. of clicks\*avg\_cost\_per\_click.

I Plotted the scatter chart CTR (y-axis) vs profit\_per\_click for 40 ads and show it to the advertising manager.

He then gave a threshold for selection. Then we chose the best ads. (a scatter chart, one point for one ad, the high CTR and the high profit\_per\_click)

* **Click-through-rate (CTR)**=the ratio of no. of clicks to the no. of impressions(广告show的次数).

This metric focuses on the customer growth.

Pros: test and improve ad characteristics

Cons: It has no information about conversion. High CTR---low conversion rate.

* Conversion rate (CR)= ratio of the no. of conversions to the no. of showns

This metric focuses on both growth and product.

Cons: It has no information about costs.

* **Profits**= revenue from conversions- ad costs

impressions | clicks | conversions | avg\_cost\_per\_click | total\_revenue | ads\_ID

no. of impressions(广告显示的次数) [0, 200,000] mean 70,000

no. of clicked [0, 20,000] mean 3,000

no. of converted [0, 1,600] mean 130

avg\_cost\_click [0, 4] mean 1.4

total\_revenue [0, 40,000] mean 2,000

50 days 1001---1122

### Data cleaning

Errors & outliers

delete the data with negative revenue which doesn’t make sense.

delete the data, for example, no. of clicks> no. of impressions

### Data engineering

Convert categorical or text values into numerical values.

product name----number through label coding

### Data visualization and analysis

Plot and show feature distribution, feature’s relationship with the target variable.

no. of impressions vs date for 40 ads (one curve for one ad, 40 curves for 40 ads)

no. of clicks vs date for 40 ads (one curve for one ad, 40 curves for 40 ads)

no. of conversions vs date for 40 ads (one curve for one ad, 40 curves for 40 ads)

#### 2.2 For each ad, predict the total number of clicks after 25 days (assume each ad keeps following its trend).

We have 50 days data from Oct to Nov (20161001 to 20161120), predict the no. of clicks the next 25 days. Group the data based on ad ID.

* For each ad, predict its clicks for the next 25 days using **the prophet forecasting model**.
* Then, sum the clicks of 40 ads.
* Plot one curve: no. of clicks vs Date.

We can see that the no. of clicks is generally flat in October and starts going up markedly in Nov. Predictions for the next 25 days show the up trend. This is because the model assumes the no. of clicks keeps following its recent trend. In order to make a more accurate prediction, I give two suggestions:

Firstly, require more data to capture yearly pattern.

Secondly, discuss with colleagues from the product department about the potential reason of the trend and yearly pattern.

#### Cluster 40 ads into 3 groups:

* the avg\_cost\_per\_click is going up
* the avg\_cost\_per\_click is flat
* the avg\_cost\_per\_click is going down

Build a linear regression for each ad (40 linear regressions, 40 ads). The target variable is the avg\_cost\_per\_click. The feature is the number of days since the first day (1,2,3,etc). Then check

the coefficient and p\_value:

Coefficient is positive and significant(p\_value/40≤5%) -> going up curve波动较大

Coefficient is negative and significant coefficient (p\_value/40≤5%)-> going down curve波动较大

Non-significant coefficient (p\_value/40≥5%) -> flat curve波动不大

## Targeted display Ads

### Background

Analyze the ads data from an online shopping website, build a model to predict whether a customer will click the ad of a product or not? Group ads and find the target customers, display different type of ads to its target customers. The e-commerce website wants to improve its ads display strategies.

#### 1.1Data statistics

Row: about 100,000

Column:

Memory usage: 5MB

The ratio of clicked to the total: 2% of users clicked the ads

### Question

#### 2.1 Build a model to optimize ads display?

Analyze the ads data from an online shopping website, build a model to predict whether a customer will click the ad of the product or not?

In order to improve the ads performance,

Firstly, group ads according to their product type, their price and so on.

Then, find the target customers of each group using our model.

Last, send ad to its target customers.

After model building, obtain the

The no. of customers in the test dataset: 33,333

* The predicted number of target customers on the test dataset: 9900, reduced by 70%
* The recall on the test dataset: 60% (65 % original method, shown to all customers)
* The precision on the test dataset: 4.35% (2% original method, shown to all customers)

The no. of target customers has been reduced by 70%.

The ads can cover 60% of valued users which will click the link.

Our model double the click rate. The click rate has been increased from 2% to 4.35%.

#### 2.2 By how much do you think your model would improve click through rate. How would you test that?

perform an A/B test to verify the performance of our model

split users into two groups, Control group and test group.

send ads to all randomly to control group.

whereas send different type of ads to its target customers in test group

#### 

### Data Clean

#### 4.1 Only consider the granted loans

no\_purchases | sex | age | price\_product | date| clicked…

no\_purchases | sex | age\_range | price\_range | is\_weekend | clicked…

#### 4.3 deal with missing values, errors

Delete records with missing values in ‘sex’ and ‘age’. Only a few, so just delete them.

Delete several records with error price, say the price is significantly high, about 10^8. 10 to the power of 8. Almost all price is within 1000.

### Feature engineering

#### 5.1 Transform feature data type

* Convert text values into numerical values

‘sex’ into int. (F/M🡪int)

#### 5.3 Find features’ relationship with the target variable: click

Feature vs target variable

* Chi-squared (the linear relationship & classification & sparse matrix)
* Tree based model
* Plotting charts: no. of different feature values vs feature values for different target variable 0/1

#### 5.4 Create new features that may be related to the target variable

Create a new feature: ‘is\_weekend’, whether it is clicked on weekend. extract the feature from dates, delete date

Create a new feature: ‘price\_range’, since the price is less than 1000, we split the price into ten equal ranges. Each number from 0 to 9 represents one range respectively. Delete price of product

Create a new feature: ‘age\_range’, split the age value into 6 equal parts, each number from 0 to 6 represents one range respectively. For example, 0 represents less than 10, 1 represents between 10 to 20… 6, more than 60, delete age

no\_purchases | sex | age | price\_product | date| …

no\_purchases | sex | age\_range | price\_range | is\_weekend | …

### For each feature, plot two figures. (1) the distribution of different feature values and (2) the feature’s relationship with the target variable, feature values vs target variable.

#### 6.1 ‘price range’

* For all price ranges, the click rate is about 5%. We observe that the price range 0 which represents the lowest price range, however has the highest click range 5.92%.

We just obtain the data about 8 days. Due to lack of data 去年同期，it is hard to tell whether 5% is normal, need further investigation.

* We further look at the data of price range 0. Plot the distribution of the no. of purchases. We notice that the customers who bought a lot also clicked a lot.

Conclusion talk with product manager, ads of price range 0 既可以对浅层用户进行流量变现又能让广告让更多具有消费意愿的中、深层用户看见。

#### 6.1 ‘sex’

The ratio of males to females is 1 : 1.6.

The ratio of male clicks to female clicks is 1: 1.7.

Apparently, female likes online shopping more than male. The click rate of female is a little higher than that of male.

We just obtain the data about 8 days. Due to lack of data 去年同期，it is hard to tell whether 5% is normal, need further invest

### Modelling

#### 6.1 Check imbalanced classes

Inspect the distribution of the target variable, ‘clicked’.

Only 2% of users clicked the ads----imbalanced seriously

For imbalanced classes, choose AUC as the evaluation metric for model selection (for different RF models, choose the best model with the greatest AUC, different RF models，只是参数不同)+find a better threshold

#### 6.2 Choose RF

LR和RF不能直接处理categorical feature，convert to numerical values at first.

LightGBM can handle categorical feature automatically.

RF outperforms LightGBM

#### 6.4 Split the data into training test datasets

We just split data into training 75% and test datasets.

#### 6.5 Train and get optimal parameters

lgb.cv(parameters, train\_set= train\_data, num\_boost\_round=1000, nfold=5,

early\_stopping\_rounds=20, seed=42, verbose\_eval=False)

Get optimal parameters

lgb.cv会得到一个性能好（如采用AUC作为评价）的模型及其参数**best number of trees**=767.树的数量是767

#### 6.6 overfitting

Check if overfitting happened? AUC on the training dataset 0.86 vs AUC on the test dataset 0.71

RF+ grid search ----find optimal parameters----address overfitting

try more hyper-parameters to find better ones.

#### 6.7 split the original training data into training and validation datasets

#### 6.8 Retrain the model on training dataset using optimal parameters

#### 6.9 Make prediction and plot ROC curve on test dataset using optimal parameters

#### 6.10 Choose the best threshold for classification problem

since the data is highly imbalanced (no. of clicked is only 2%), if using default probability threshold (0.5), the model just classify every example as negative, so we need to plot the ROC curve on validation dataset and choose a better threshold.

Plot the ROC curve and find the point closest to the top left corner of the ROC curve.

For the target variable ‘clicked’, we focus on the cost of FN, so **choose a smaller threshold** (less observations will be classified as positive). This is because if ‘clicked’ is identified as ‘not clicked’, the company will lose money.

Choose the points around the best point. From them, choose a smaller threshold.

The selected threshold is **0.028.**

#### 6.11 Get predicted labels using the best threshold on test data

predicted\_repaid = clf.predict (TestFeatures)

for pred in preds:

result = 1 if pred > best\_threshold else 0

highly imbalanced, since we use an optimal threshold, we can find that the recall increases.

#### 6.12 Visualize the feature importance

Plotting and show which features contribute much to the predictions.

lgb.plot\_importance(clf, ax=ax, height=0.5)

## Identifying fraudulent transactions similar to loan (bank) (不看)

### Background

Analyze the data about users' first transactions on an e-commerce website to predict whether a transaction is fraudulent欺诈 or not. e.g. using stolen credit cards to buy clothes.

The target variable is ‘if\_fraud’, whether the transaction is fraud or not?

#### 1.1Data statistics

Row: about 150,000

Column: 10 features+1target

Memory usage: 6MB

### Questions

#### 2.1 Explain how the cost of FPs vs false FNs would impact the model.

For fraudulent transactions detection, the cost of FN is higher than that of FP, so minimizing FN is the focus, so **choose a smaller threshold**, that means more observations will be classified as positive. This is because if ‘fraud’ is identified as ‘not fraud’, the company will lose money. On the contrary, we can ask colleagues or domain experts for further investigation.

Choose the points around the best point. From them, choose a smaller threshold.

Cancer/fraudulent transactions detection---minimize FN----smaller threshold

if\_repay (loan)/ if\_customer (Recommendation) ----minimize FP----larger threshold

一些时候也可以画图：thresholds vs prediction accuracy（或其他评价指标如profit），直接看图找到highest accuracy 对应的thresholds

#### 2.2 How would you explain to her how the model is making the predictions?

To better understand, I’d like to plot the feature importance from lightGBM, visualize the feature importance of each feature in descending order. We can see the most important feature which usually also make sense in real life.

Furthermore, I’d like to build a shallow decision tree and plot it to display the prediction process.

dt = DecisionTreeClassifier(max\_depth=3) #3----small, shallow

dt.fit(features, target)

export\_graphviz(dt, feature\_names = features.columns, class\_names=['NotFraud', 'Fraud'], filled=True)

#### 2.3 What kinds of users are more likely to be classified as at risk? What are their characteristics?

From the simple decision tree, we can observe that if the ’interval’ is <69seconds, which means the customer purchases immediately after signup, then this transaction is more likely to be fraudulent.

Look at the parameter ‘value’ in each leaf node. For binary classification problem, there are two numbers for ‘value’. Normally, their **difference** is approach to 1.这两个值之和为1，他们的差接近1，% are labeled as 0，% are labeled as 1，e.g. (0.77, 0.23)说明错分了很多,。

#### 2.4 This model can be used live to predict if a transaction is fraudulent or not. What kind of different user experiences would you build based on the model output?

I will use two thresholds.

For current transaction, my model will give the probability *p* that the transaction is fraud or not.

If *p*<threshold1, then I assume the transaction is normal.

If threshold1≤*p*<threshold2, (so the customer is at risk, but not too much) I will ask the customer for additional authorization. For example, send an SMS to the customer for verification

If *p*>threshold2 (the customer is likely to commit a fraud), I will tell the customer that the transaction is put on hold. And asks the domain experts for further investigation.

### Feature engineering

最终得到的features:

interval | age | no\_shared\_dev | no\_shared\_ip | transaction\_amount…

interval----It is very suspicious for a user signup and purchase immediately.

no\_shared\_dev ----how many times a device is shared. Many people share the same ip would indicate the transaction is fault.

no\_shared\_ip----how many times an ip is shared. the more an ip is shared, the more suspicious

#### 3.2 Create new features which may be related to the target variable

Transform the string type into int type

Sex ‘F’/’M’-----0/1

interval=purchase\_time-signup\_time

no\_shared\_dev= is created from ‘device’, delete ‘device’

no\_shared\_ip= is created from ‘ip’, delete ‘ip‘

#### plotting the feature vs target variable

Cannot understand

For regression problems: plotting figures: numerical feature vs target variable: scattered points

For classification problems: plotting figures: numerical feature vs target variable: hist

### Data merge

### Feature engineer

#### transform the categorical feature to numerical feature

categorical feature: doesn't have mathematical meaning, cannot calculate directly

numerical feature may be either discrete or continuous.

#### Feature selection for labeled data

1. delete duplicate features (Is feature is-employ highly related with feature salary?) more features, more complicated and prone to overfitting

deal with missing values, categorical feature---numerical feature

then draw a heatmap to show the linear correlation between pairs of features. 1和-1正负相关性都高，0表示完全不相关,本质是correlation matrix，每个值是pearson correlation

fig, ax = plt.subplots(figsize=(12, 10))

sns.heatmap(features.corr())

2. delete irrelevant features based on their relationship with the target variable (Is the feature ‘loan purpose’ related with the target variable?)

3. select the best k features based on their relationship with the target variable.

calculate the following.

1. **If the relationship is linear**:

* **for classification:** Chi-Squared

SelectKBest(chi2,k=3).fittransform(features,target)

output: chi2, p-values of each score, if one feature has high chi2, however, its correspoinding p-value is high, then the feature is irrelevant.

* **for regression**: Pearson’s Correlation Coefficient

SelectKBest(f\_regression,k=3).fittransform(features,target)

can be used for sparse matrix

output: F-scores, p-values of F-scores, if one feature has high F-score, however, its correspoinding p-value is high, then the feature is not so relervant.

1. **If the relationship is nonlinear**: Tree-based feature selection

SelectFromModel(randomforest, threshold=None, prefit=False, norm\_order=1, threshold=None, max\_features)

e.g. sfm=SelectFromModel(RF,max\_features=3).fit(trainfeatures, traintarget)

prefit----RF还没有fit过，prefit=False,可以边fit边select features；RF fit过，prefit=True

max\_features----the number of the best features

threshold---->thershold, then select the feature

show which features are selected: features.columns[sfm.get\_support()]

show the parameters: sfm.get\_params

adjust the parameters:sfm.set\_params(threshold='mean')

## Conversion Rate

不同国家，不同年龄，不同新旧用户，是否转化（0,1）

**Index**

**1. Load and inspect data**

* 1. Data observation and data cleaning
  2. Inspect each feature’s distribution and its relationship with the target variable

### Load and inspect data

|  |
| --- |
|  |

* 1. **Data observation and data cleaning**

duplicates, missing data, format issue（field types, date）, outliers...

|  |
| --- |
| df.read\_csv(), df.describe(), df.head() |

From above, we observe that the max age is 123 which must be an outlier.

From above, we observe that we have 316200 samples, but just two outliers, so we can remove them.

* 1. **Inspect each feature’s distribution and its relationship with the target(conversion)**

We will investigate features and how their distributions differ for the two classes.

|  |
| --- |
| 用tableau画图，看附录 |

From above figures, we observe that

1. There are lots of users from China, but the conversion rate in China is the lowest.

2. There aren’t a lot of users from Germany, but the conversion rate in Germany is very high.

3. The conversation rate of the people more than 60 is nearly 0.

4. The conversion rate is about 3% which is industry standard.

5. The longer the user stays on the website, the higher probability of conversion will be.

### 2. Feature engineering

Convert categorical features such as ‘country’ and ‘source’ into numerical values

### 3. Modelling

3.1 Observe the data for parameter selection

|  |
| --- |
|  |

From y’s mean, we observe that the target value is seriously imbalanced. So we will choose the model performance metrics such as Recall, ROC/AUC…

* 1. Split the data into training, test and validation dataset. (RF不需要validation dataset)

|  |
| --- |
|  |

We can see that both ytrain.mean() and ytest.mean() are similar to y.mean(), so the splitting process is not biased.

* 1. model selection

I choose RF due to the following reasons:

1) Generally, RF has very high prediction accuracy.

2) It requires very little time to optimize it (its default parameters are usually close to the best ones).

3) It is robust to imbalanced data, outliers, missing data.

3) robust against overfitting since it is an ensemble method

7) It can estimate the importance of each feature.

8) It provides OOB error to generate an internal unbiased estimate of the generalization error as the forest building progresses.

|  |
| --- |
|  |

Inspect the importance of each feature

|  |
| --- |
|  |

From the figure, we can observe that the total pages visited is the most important feature. Unfortunately, it is probably the least “actionable”. So, we find the next important feature without ‘the total pages’. Remove the feature, rebuild the RF model and inspect the most importance feature again.

### 4. Performance evaluation

|  |
| --- |
| 计算model 在训练集的OOB error和在测试集上的error |

From above, we see that our model is not overfitting since the OOB error on training data and the error on testing data are similar:1.5% and 1.4%.

绘制confusion matrix

|  |
| --- |
| confusion\_matrix=confusion\_matrix(y\_test,y\_pred) |

对于网站转化率而言，FP的cost比较大（一个真的转化用户预测成假的），Recall就是minimize FP的。

### Conclusions, (possible reasons) and suggestions

* The conversion rate of the young users is higher than that of the older users.
* Try to attract more young people by advertising.
* From the age of 30，conversion rate starts to dropping.
* UI design?
* The Germany has a high conversion rate
* Relatively less users from German with a high conversion rate. Possible reasons may be the website 1) is poorly translated. 2) doesn’t fit the local culture. 3) has payment issue…
* Try to attract more users from Germany. They are easily to be converted. Big opportunity!
* People with old accounts has a high conversion rate.
  + Try to attract them back by sending emails or discount promotion.
* The conversion rate of Chinese is the lowest while the number of Chinese is large.
  + Possible reasons may be the website 1) is poorly translated. 2) doesn’t fit the local culture. 3) has payment issue…
  + Pay attention to the large number of users from Chinese. Big marketing from Chinese.
* People who has visited many pages (more than ) has a high conversion rate.
  + Possible reasons may be the website 1) is poorly translated. 2) doesn’t fit the local culture. 3) has payment issue…
  + Pay attention to those who visited many pages and stay on the website for a long time. They are more likely to be converted.

**附页**

|  |  |
| --- | --- |
| Feature为categorical如country | Feature为numerical如age，total pages visited |
| **Distribution map**  3分类就是triple bar,2分类就是双柱状图  x-axis: different values of the feature中国，美国…  y-axis: 不同国家在每个分类下有多少个  bar  一定要点击analysis---aggregate measures  创建2个字段converted0和converted1分别统计未转化和转化了用户数量  右击----create---calculated…---- count([converted])-sum([converted])sum([converted])  Drag country到columns  右击----create---calculated…---- sum([converted])  Drag ‘measure values’到rows,右击 measure values----edit filter----勾选converted0和converted1  Drag ‘measure names’到columns  Drag’measure names’到marks下的color  filter---age---range----0,79  edit title  纵坐标：count(country)  在dashboard里能够float legends | **Distribution map**  Bar/area  一定要点击analysis---aggregate measures  创建2个字段converted0和converted1分别统计未转化和转化了用户数量  右击----create---calculated…---- count([converted])-sum([converted])sum([converted])  右击----create---calculated…---- sum([converted])  Drag ‘measure values’到rows,右击 measure values----edit filter----勾选converted0和converted1  Drag’measure names’到marks下的color  filter---age---range----0,79  edit title  纵坐标：count(country)  在dashboard里能够float legends |
| **relationship with the target**  bar  x-axis: different values of the feature, 中国，美国…  y-axis: 中国转化率的均值，美国转化率的均值…  有个问题：当data imbalanced，转化的数据量很少比如橙色很少，无法观察。私底下最好还是在两张图显示。就是单柱状图  一定要点击analysis---aggregate measures  columns: country 注意country要在dimension里，否则要右击---dimension  rows: agg(conrate)  conrate是转化率字段新建的，SUM([converted])/count([converted])  filter---age---range----0,79  纵坐标：Mean Convertion Rate per Contry  在dashboard里能够float legends | **relationship with the target**  line  x-axis: different values of the feature, 不同年龄  y-axis: 不同age转化率的均值，美国转化率的均值…  有个问题：当data imba  一定要点击analysis---aggregate measures  columns: age，然后右击---dimension  rows: agg(conrate)  conrate是转化率字段新建的，SUM([converted])/count([converted])  filter---age---range----0,79 |

|  |
| --- |
| OOB error  对于某个训练样本(xi,yi)，use some decision trees to predict yi。这些trees在构建时没有用到(xi,yi).取所有预测的误差均值。 OOB error近似于需要大量计算的k-fold cross validation. RF没有必要交叉验证  下面计算OOB error和AUC的值  rf0 = RandomForestClassifier(oob\_score=True, random\_state=10)  rf0.fit(X,y)  print rf0.oob\_score\_  y\_predprob = rf0.predict\_proba(X)[:,1]  print "AUC Score (Train): %f" % metrics.roc\_auc\_score(y, y\_predprob)      关键问题是如何选择最优的m，基于OOB error。随机地从M个特征中选取m个特征。  rf = RandomForestClassifier(n\_estimators= 60, max\_depth=13, min\_samples\_split=120, min\_samples\_leaf=20,max\_features="auto" ,oob\_score=True, random\_state=10)  rf.fit(X,y)  print rf.oob\_score\_  oob\_error = 1 - rf.oob\_score\_  n\_estimators迭代次数  max\_depth决策树最大深度  min\_samples\_split内部节点再划分所需最小样本数  max\_features最大特征数  oob\_score=True 使用OOB验证  为特征min\_samples\_leaf不断尝试选择最优的值。  sample\_leaf\_options = [1,5,10,50,100,200,500]    for leaf\_size in sample\_leaf\_options :  model = RandomForestRegressor(n\_estimator = 200, oob\_score = TRUE, n\_jobs = -1,random\_state =50,  max\_features = "auto", min\_samples\_leaf = leaf\_size)    model.fit(x,y)    print "AUC - ROC : ", roc\_auc\_score(y,model.oob\_prediction)  #获得特征重要性信息  importances = list(rf.feature\_importances\_)  feature\_importances = [(feature, round(importance, 2))  for feature, importance in zip(feature\_list, importances)]  #重要性从高到低排序  feature\_importances = sorted(feature\_importances, key = lambda x: x[1], reverse = True)  # Print out the feature and importances  [print('Variable: {:20} Importance: {}'.format(\*pair)) for pair in feature\_importances]  特征重要性绘图  import matplotlib.pyplot as plt  %matplotlib inline  #设置画布风格  plt.style.use('fivethirtyeight')  # list of x locations for plotting  x\_values = list(range(len(importances)))  # Make a bar chart  plt.bar(x\_values, importances, orientation = 'vertical')  # Tick labels for x axis  plt.xticks(x\_values, feature\_list, rotation='vertical')  # Axis labels and title  plt.ylabel('Importance');  plt.xlabel('Variable');  plt.title('Variable Importances');  importance\_frame = pd.DataFrame({'Importance': list(clf.feature\_importances\_), 'Feature': list(names)})  importance\_frame.sort\_values(by='Importance', inplace=True)  importance\_frame.plot(kind='barh', x='Feature', figsize=(8, 8), color='orange') |

## Online courses (A/B test)

To test the effect of different languages of online courses. Dalian Maritime University changed a part of its online courses’ language from Chinese to English. Before expand the new version to all students, we want to see the effect.

The target metric is the number of clicks.

The old version performs better than the new version

table： user\_id | date | number of clicks | university grade | test

the control and test distribution double bar

Plotting double bars (test=0, test=1): x-axis: university grade y-axis: count (university grade)

Plotting double bars (test=0, test=1): x-axis: date y-axis: count (date)

2个bar的高度差不多,否则 biased data collection

More students learn online courses in Guangzhou University on weekdays than weekend.

2)

double bar, Date/ university grade vs avg(number of clicks)

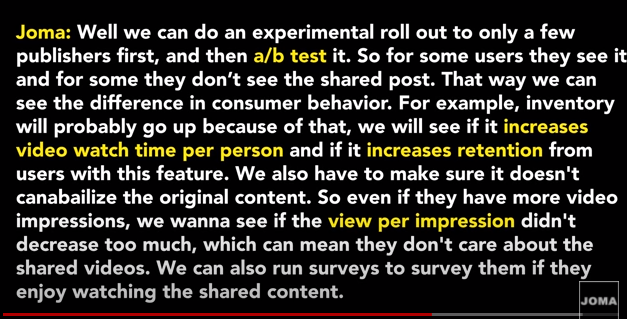
查看test0和test1的均值即可看出, the old version performs better。Confirm that the average number of clicks for users in the control group is higher than that of test group.

T-test is used to check whether the means of the two populations are significantly different. If p-value<0.05, significantly different.

Conclusions, (possible reasons) and suggestions

1. The test result shows that it is hard for most students to learn professional courses in English. So, add Chinese subtitle or explain some obscure knowledge points in Chinese would increase students’ engagement.

## WEIBO(A/B test): Potential friends recommendation



Design an A/B test to test whether the new function, that is friends recommendation, increases users’ engagement.

The target variable is ‘the number of visited pages’ during the user’s first session since the test started.

user\_id | date | signup\_date | device | if\_testgroup | pages\_visited

Does the test perform similarly for all user segments? for all features?

Row: about 100,000

Column: 16

Memory usage: 5MB

test group: pages\_visited.mean()=4.59

control group: pages\_visited.mean()=4.61

p\_value=57%>5%

Unpaired t-test compares the means of two independent groups.

Paired t-test: compare the means of the same group.(e.g. the people drink beer before or after is the same group.)

T Score：The larger the T score, the more difference there is between groups.

We observe that the *p* value is 57% greater than 5%, so there is no significant difference between the control and test group. So, the new function shouldn't expand to all users.

However, we further investigate the target value distribution on each feature for control and test group. We notice that the average number of visited pages in Android devices is very low, approach to 0. There may be some bugs in showing the new function in android devices. Talk with the soft engineer in charge of android version development.  
Remove the data from android devices, the *p* value is less than 5%. That means the new function is significantly useful. So, I would suggest expanding this function to other users.

### Determine the target variable

The number of visited pages, the average click-through rate for ads, the conversion rate,

### Find the possible components behind the new function which may affect the target variable

This is because we need to test each component separately.

* The algorithm behind the new function, The UI behind the new feature
* The target variable change may be driven by the algorithm behind the new function or the UI changes needed to accommodate the new function.

We need to test each component separately.

A way to isolate the two components exactly is to run 3 versions of the site

* Novelty effect
* The performance is improved just because the added function is new and the users are curious好奇.

How to isolate the effect of the new funciton vs the effect of novelty

One obvious solution for novelty effect would be to split users in new and old. 上面那个例子就是，this is a really strong sign of novelty effect，因为在新用户中效果不好

Another obvious solution for novelty effect would be to run the test longer. However, that is inefficient, and the cost is high. 测试长，获取数据时间长

### Determine the sample size

In order to estimate the sample size, we need the current performance and the expected increase rate. For example, e.g. The current conversion rate is 10%. The product manager asks for more than 1% increase rate. We then call a function in Python to obtain the sample size.

> import statsmodels.stats.api as sms

> es = sms.proportion\_effectsize(0.1, 0.01)

> sms.NormalIndPower().solve\_power(es, power=0.8, alpha=0.05) # power of the test, significance level,基本用默认值

也可以和产品经理商量，他给出a range of increase rate，then we can draw a curve to show the sample size vs increase rate。看一系列的sample size。Sample size是control group，是对于一个group的

### Data collection

If the data provided by the website is huge each day, in order to capture the weekly pattern, we just run tests on 1% or less of its users. Otherwise, we generally run tests more than 1 week to meet the sample size requirement.

Randomly split users into two groups, one Control and one Test. The control group has no recommendation strategy.

### Select features and generate new features which may be related to the target variable

* Select features, e.g. the type of devices (iOS, Android, web)
* Generate a new feature, a new user or an old user, the testing date- the registration date

Get the day of the week

### Preform an unpaired t-test to test whether the new version is better than the old one?

#### 6.1For all data, perform a t-test

Find control group: 找到target variable即pages\_visited 那一列的值： test=0

Find test group:找到target variable即pages\_visited那一列的值： test=1

|  |
| --- |
| ttest\_ind(target values in test group, target values in test1, equal\_var=false) //返回t-value 和 p-value |

Unpaired t-test compares the means of two independent groups.

We observe that the *p* value is greater than 5%, so there is no significant difference between the control and test.

So, the new feature shouldn't expand to all users.

#### 6.2 For each feature, check randomness, perform t-tests.

#### For each feature, check randomness.

Detect biased data collection

一般来说都会control 和test half to half，但是对于某个特征就可能biased

For each feature, check the feature’s distribution on control and test are comparable. Otherwise, the testing result may be invalid.

Double bar，对于每个特征，不同特征值，分别统计control中和test中的总数, 2个bar的高度差不多,否则 biased data collection

double bar(test=0, test=1) x-axis: devices y-axis: count(device)

#### 5.2.3 Check whether the new version is significantly different from the old one.

The basic idea is: for each feature, inspect the average target values distribution on control and test. Two methods

* **For each feature, plotting the average target value on control and test**

target value.mean()，看是不是the average target value on test is constantly higher than on control?

2) double bar(test=0, test=1) x-axis: device y-axis: visited\_pages.mean()

* **For each value of the feature, perform a t-test and obtain a p\_value**

For each feature, list the following table. (Each row in the table stands for a feature value.)

Conclusion | number\_test | number\_ctrl | test\_mean | ctrl\_mean | mean\_diff | p\_value

结论是significant or not significant, test group的数量，test group的target均值,test group和control group的均值差，p\_value

Feature value1 (ios)

Feature value2 (android)

可以看出每个feature在control/test group的distribution， 可以看出每个feature的target variable 在control/test group的mean target value是否显著不同（显著improve/decrease，或不变）

### Draw conclusions, possible reasons and give suggestions

**test效果比control好，有些效果不好，possible reasons：**

From the above table…we can see that, by applying this new feature

Pages visited in IOS devices significantly increases.

Pages visited in web doesn’t significantly increase. The possible reason may be that the recommendation function is not shown in a noticeable position.

Pages visited in Android devices has reduced to zero, there may be some bugs in implementing the recommendation function in Android.

Removing the records with Android, the new feature becomes significantly useful. So, I would suggest expanding this function to other users.

Discuss with the soft engineer in charge of the android version development.

对于第二个feature如new/old user

For old users, the new feature improves performance in terms of user engagement but the improvement is not significant.

For new users, the new feature significantly decrease the performance.

**test效果比control差，possible reasons：**

* We didn’t collect enough data，we just have data for 5 days less than a week. Cannot obtain weekly patterns.
* Biased data collection

**Bias data selection**

Discuss with the software engineer in charge of data splitting.

Remove the biased data and rerun the test to see the result.

Adjust the weights reduce the bias for those two segments

### 一个新的function，到底是其后的算法还是其后的UI贡献了性能提升？

A way to exactly isolate the two components is to run 3 versions of the site at the same time:

Version 1 is the old version

Version 2 is the site with the new Feature, where suggestions are based on the machine learning model.

Version 3 is the site with the new Feature, but suggestions are based on the baseline or from historical data.

For instance, you can use a history-based model (suggest users whose profiles were visited in the past by that user) or simply suggest users with the highest number of shared connections.

If the data amount is huge, then select a small proportion of the data, say 1% to control and test.

## Coding

### dataframe, series, list, array

* **dataframe**

row index+ column name + value

多个series组成，接受dict类型数据，如果不指定，dict中的key默认为column name。

* **ndarray**

多维数组,数据类型相同，和list差不多，list内的数据类型可不同

* **series**

是一个字典，row index+value

#list to series,或直接接受dict

myseries = Series(mylist, index = ['one', 'two'])

# list to dataframe或直接接受dict

mydataframe = DataFrame(mylist, index = ['one', 'two'], columns = ['year', 'state', 'pop'])

# dataframe to array

ndarray = mydataframe.values #移除row index and column names

# list to array

ndarray = np.array(mylist)

# dataframe to list, 或直接接受dict

# dataframe to array

arr = mydataframe.values

### Import

import numpy as np

import pandas as pd

import itertools #迭代循环

import cPickle #使用pickle模块你可以把Python对象直接保存到文件，而不需要把他们转化为字符串

|  |
| --- |
| **For plotting** |
| import seaborn as sns # need install, pip install seaborn  import matplotlib.pyplot as plt  plt.style.use("ggplot") #无需自定义图中网格等，自己会美化 |

|  |
| --- |
| **For AB test** |
| from scipy.stats import ttest\_ind |

|  |
| --- |
| **For algorithm** |
| import lightgbm # pip install lightgbm  from sklearn import tree  from sklearn.tree import DecisionTreeClassifier  import graphviz # decision tree visualization  from sklearn.cluster import KMeans  from sklearn.metrics import silhouette\_score #评价kmeans性能，越接近1，聚类越好  from sklearn.decomposition import PCA |

|  |
| --- |
| **For data splitting** |
| from **sklearn**.model\_selection import train\_test\_split |

|  |
| --- |
| **For evaluation metrics** |
| from **sklearn**.metrics import auc, roc\_curve, classification\_report, precision\_score, recall\_score,  accuracy\_score, roc\_auc\_score |

|  |
| --- |
| **For feature engineering** |
| from sklearn.preprocessing import LabelEncoder #feature encoding  from sklearn.preprocessing import normalize # feature normalization  from sklearn.preprocessing import StandardScaler # feature normalization  from sklearn.feature\_selection import chi2,f\_classif #计算特征间的线性相似度 |

|  |
| --- |
| **For tuning parameters** |
| from sklearn.model\_selection import GridSearchCV  from sklearn.model\_selection import RandomizedSearchCV |

### Data loading

#### 3.1从csv读入数据

将c1列和c2列解析为时间数据，如将20200102解析为2020-01-02，返回所有原始数据和解析后的列替换原来的列

data = pd.read\_csv(r'C:\Users\juanchen\Desktop\120 jifu zhao\data\loan\_table.csv', parse\_dates=['c1', 'c2'])

data=

#### 3.2从a.txt读入数据，数据间，逗号

如果输入文档名称为a.txt,文档内容如下

1, 2

3, 6

data=pd.read\_csv('a.txt',sep=',')

data.columns=['feature1', 'target']

X=data[['feature1']] # data[['feature1']].values ?

y=data[['target']]

### Data overview

#### 3.1 list top 10 rows of a data frame

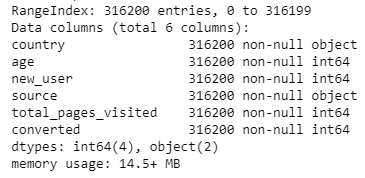
data.head(10)

#### 3.1 list last 10 rows of a data frame

data.tail(10)

#### 3.2 Views the number of rows and columns, the memory usage, the data type of each column.

data.info()



#### 3.3 View the count, mean, max, min, 25% percentile, 50% percentile, std etc. of each column

def f1(mylist): #mylist=[1,2,5,11,232]

mylist \_25 = np.percentile(mylist, 25)

mylist \_50 = np.percentile(mylist, 50)

mylist \_75 = np.percentile(mylist, 75)

mylist \_max = np.max(mylist)

mylist\_mean = np.mean(mylist)

mylist \_min = np.min(mylist)

return pd.Series([mylist \_mean, mylist\_min, mylist\_25, mylist\_50, mylist\_75, mylist\_max],

index=['mean', 'min', '25%', '50%', '75%', 'max']) #rename

stats = data.groupby('c1').apply(f1).reset\_index()

stats.head()

#### 3.4 Merge two tables

按照列c1进行连接，左连接，连接两张表：data和data1返回merge后的表,如果data和data1合并时，有两列重名，data该列名字加后缀\_l, data1该列名字加后缀\_r。

pd.merge(left=data, right=data1, how='left', on='c1', suffixes=['\_l', '\_r'])

### Data cleaning

* View part of data ‘age’<100

data[data['age'] <100]

#### 4.1 missing data

* **Find missing data**
* 统计每列null 值的数量

返回两列，第一列为索引列，第二列为统计的有空值的特征的空值的数量。第一列的值为有空值的列名。

data.isnull().sum()

增加索引列0,1,2…第一列列名为index,不再是索引，给第二列重命名为c1

pd.DataFrame(data.isnull().sum()).rename(columns={0: 'c1'} .reset\_index()

* 选出c1列，条件为c1列值为空

data['c1'].isnull()

* 选出行，条件为c1列值为空

data[data['c1'].isnull()]

* 选出c1列，条件为c1列值非空

~data['c1'].isnull()

* 选出行，条件为c1列值非空

data[~data['c1'].isnull()]

* 计算非空c1列的值占总数的比率

1 – data['c1'].isnull().sum() / len(data)

* **Fill missing values**
* fill missing values with -1

data = data.fillna({'c1': -1, 'c2': -1})

* fill missing values with median

data = data.fillna({'c2': data['c2'].median()})

#### 4.1 Delete

* select rows based on some conditions

data=data[data['c1'] <100] # select rows, c1列的值<100的

data[~data['c1'].isnull()] #select rows，c1列的值not null的

* select columns

data['c1'] #选出c1列

* select columns

data[data['c2'] <100]['c1'] #选出行，c2列中c1列，其中c1列中的c2

#### duplication， 唯一值，unique

查看c1列的值是否有重复，没有重复，返回True

len(data['c1'].unique()) == len(data)

返回c1列的值，只返回唯一的值

data['c1'].unique()

c1列唯一值的数量

len(data['c1'].unique())

#### Sort the data in ascending order

返回根据列c1值降序排序的

data.sort\_values(by='c1', ascending=False)

选出c1列，并对c1升序排序

sorted(data['c1'])

#### select rows and columns选择行，列

选择列c1,c2

data1 = data[['c1', 'c2']] #两个]]

选择c1列的值，选择dataframe里，c1列第1行的值，没有.values[0]就只是返回一个索引

tmp['c1'].values[0]

选择行

data['c1'] == 'Mobile'

#### 选择某些行select rows，多张表

返回选择的行,注意这里用到了2张表

data1[(data['c1'] <= 100) & (data['c3'] >= 20)]

#### Create data, 用已有的列的值，构造新的data

用已经有列数据c11,c22插入data中，data中包括2列，c1,c2

data = pd.DataFrame({'c1': c11, 'c2': c22}, columns=['c1', 'c2'])

c11=data[['home', 'search', 'payment']]

c22=data[[0.5, 0.1, 0.07]]

#### Add new rows/columns新增

dada中新增一列c2，用已有的列值c3赋值

data['c2'] = c3

c3 = data.groupby('c1')['c2'].count().reset\_index()

c3 = c3.rename(columns={'c2': 'c2\_count'})

#### date to week, month, hour, seconds

返回新的一列，c1列的每个日期对应的月

data[‘month’]=data['c1'].apply(lambda x: x.month)

返回新的一列，c1列的每个时间对应的小时

data['hour'] = data['c1'].apply(lambda x: x.hour)

返回新的一列，c1列的每个日期对应‘一年中的第几周（1,2....）

data['c1'].apply(lambda x: x.week)

返回新的一列两个日期之间（c3和c2）间隔多少天

data[cn]=(data['c3'] - data['c2']).apply(lambda x: x.days)

data[cn]=(data['c3'] - data['c2']).apply(lambda x: x.dt.days) 返回int值

返回新的一列两个日期之间（c3和c2）间隔多少秒

np.array(list(map(lambda x: x.seconds, data['c3'] - data['c2'])))

生成dataframe,每个元素是一个日期，每两个元素间隔1天，元素介于'2011-01-24'和'2015-12-13'之间，包括'2011-01-24', 和'2015-12-13'

pd.date\_range(start='2011-01-24', end='2015-12-13', freq='D')

#返回新的一列，c1列的每个日期对应的day of the week，0~6

data[‘dayofweek’]=data['c1'].apply(lambda x: x.dayofweek)

解析时间 parse time

data['c1'] = pd.to\_datetime(data['c1'])

将c1列和c2列解析为时间数据，如将20200102解析为2020-01-02，返回所有原始数据和解析后的列替换原来的列

pd.read\_csv('./juanchen/raw\_data.csv', parse\_dates=['c1', 'c2'])

取出c1中最早的一天

data['c1'].min( )

#### 新增列， add a new column

* **每行的和 sum**

仅保留原索引列，并增加一列，重命名为count,该列是原来每一行数据之和（数据之和不包括索引列的值），axis=1表示按行加，新增索引列0,1,2,原索引列为普通数据列

data1 = data.sum(axis=1).reset\_index().rename(columns={0: 'count'})

data1['is\_exceed'] = (data['c1'] > data ['c2']).astype(int)

#### 新增列，表示另一张表在一张表中是否存在

连接两张表data 和data1，在data中新增列c2,值为1表示该row在两张表都存在。

data1['c2'] = 1 #data1中新增列c2,值都为1

data = pd.merge(left=data, right=data1, how='left', on='c1')

data = data.fillna(value=0) #给所有NA填充0，因为data中有，data1中没有的值在c2列都是NA

data['c2'] = data['c2'].astype(int)

#### axis=1, axis=0

重点在于方向，而不是行和列

当axis=0时，方向从上到下，从上到下求平均，上下纵向拼接，drop表示纵向发生变化，即行的减少

df.dropna(axis=0,how='any', inplace=True)本列是否有NaN，若有，删去该NaN的行,inplace表示语句立即执行生效

当axis=1时，方向从左到右，从左到右横向求平均，左右横向拼接，drop表示横向发生变化，即删除列，这个太难理解了

#### 删除c1列

data = data.drop(labels='c1', axis=1)

#### 4.2Groupby

Groupyby之后是没有index的，例如根据c1的值分组（相同c1为一组），计算每组的c2的均值。返回2列，第一列为c1也是索引列，第二列为该组c2的均值，返回的是series, reset\_index()给返回的数据增加索引：0,1,2…使得返回的是dataframe。c1变为非索引的普通列

series类似于1维数组，由索引+数值组成，dataframe的某一列，返回的就是series

data1=data.groupby('c1')['c2'].mean().reset\_index()

data1 = data1.rename(columns={'c2': 'c2\_mean'}) #一般groupby之后都要重命名

data中新增列c2\_mean，值为每组的数量/均值

data = pd.merge(left=data, right=data1, on='c1')

根据c1的值分组（相同c1为一组），计算每组的c2的唯一数量。 unique\_count是一个函数，输入为该组的c2的值，求不同值的数量

data.groupby('c1')['c2'].apply(unique\_count).reset\_index()

def unique\_count(x):

return len(np.unique(x))

#### 分组后返回两列或多列，经常自定义函数并apply于组内groupby

根据c1分组，分组后返回多列c1,c2,c3，找到组内c2值最小的那一行对应的c2,c3值

def f1(df):

index = df['c2'].argmin() #该组内c2值最小的行的索引

return df.loc[index, ['c2', 'c3']] #该索引对应行的，c2和c3的值

data1=data.groupby('c1').apply(f1).reset\_index()

data1.rename(columns={'c2': 'c2\_min', 'c3': 'c3\_min'})

another example

当我们需要分组，并且得到多列，每列或者为统计，或者为其他值…

groupy+function

Take the feature ‘device’ for example

def run\_ttest(df):

c2 = df['c5'].values

c3 = c2.mean()

return pd.Series({'c2': len(test\_data), 'c3': test\_mean,})

tests.groupby('c1').apply(data). .reset\_index()

another example

def f1(df):

a = df['c3'].sum()

b = df['c4'].values[0] #组内，c3列的值是相同的，随便取一个值

return pd.Series([a, b], index=['c33', 'c44']) 返回两列，列名为c22和c33，值为a,b

data1 = data.groupby(['c1', 'c2']).apply(f1).reset\_index()

#### 最小值,最大值的行的索引index

c1最小值的那一行的索引

index = data['c1'].argmin()

该索引对应的行的c2和c3的值

data.loc[index, ['c2', 'c3']]

#### 4.2 rename 重命名

给第二列重命名为c1，注意是第0列，第一列为index即每个列的列名

conv\_ratio = 1 - pd.DataFrame(data.isnull().sum()).rename(columns={0: 'c1'}) / len(data)

给c1列重命名为c11，返回所有列，其中c1列更名为c11。

data.rename(columns={'c1': 'c11'})

#### 4.2 行、列取值

data.loc[‘r1’, ’c1’] #行列名，第r1行，第c1列

data.iloc[0, 3] #索引第1行，第4列

#### Read the row index, column index, column name 列名

DataFrame：列索引+行索引+行列数据

row index一般就是0,1,2...

>>>data.index

column index就是column name

>>>data.column

#### 计算某个矩阵特征的相似性similarity

根据c1和c2对data进行分组，计算组内c3的数量，索引为c1和c2，值为c3\_count。应用unstack后，c1变为行索引，c2变为列索引

data1 = data.groupby(['c1', 'c2'])['c3'].count().unstack(fill\_value=0) #缺失值用0填充

计算相似性similarity

data2 = normalize(data1, axis=1) # normalize the matrix

data3 = np.dot(data2, data2.T) # calculate the similarity matrix 点积即A\*A^T

similarity\_df = pd.DataFrame(data3, index=data1.index, columns=data1.index) 重新确定索引和列名称

def find\_topk('A', similarity\_df, k=10): #find the top 10

df = similarity\_df.loc['A'].sort\_values(ascending=False)[1:k + 1].reset\_index() #similarity\_df中，列‘A’中最大的10个值取出，保留原索引，并增加新索引0,1,2...

df = df.rename(columns={'c1': 'c11', 'A': 'c22'}) #c1是原来的索引列

return df

#### 求每一天比前一天的变化率，如增长率等，一段时间的trend

#df是一个dataframe，按照date升序排序后，取出c1列的值即为一个array

#计算c1\_array中每一天相对于昨天的变化率即 今天的值/昨天的值，c1\_array[1:]表示取出除了第一个值

的所有值，c1\_array[:-1]表示取出除了最后一个值外的所有值

c1\_array = df.sort\_values(by='date')['c1'].values

ratio = c1\_array[1:] / c1\_array[:-1]

### Feature engineering

There are about 20 features, delete 8

#### 5.1 Convert text or categorical values into numerical values. Transform categorical features into numerical features

* **Through label encoding**

features = subset[['c2', 'c3']]

# c1不变，新增c2列为c1的numerical values

le = LabelEncoder()

data['c2'] = le.fit\_transform(data['c1'])

* **Through one hot encoding**

（1）对data中所有categorical进行 one hot encoding

data = pandas.get\_dummies(data, drop\_first=True) #data中的所有categorical---numerical,其余特征不变

（2）保留c1列，新增encoding的若干列，并对若干列重命名，对c1列进行encoding后存入data1

data1 = pd.get\_dummies(data, columns=["c1"], prefix=["Type\_is"] ) #pd是pandas的缩写

data = data.join(data1)

* **Transform from categorical type into int type**

encoder = LabelEncoder()

data[target] = encoder.fit\_transform(data[target])

#### 5.1 Transform text values into numerical values

c1列有两个值’short’和’long’，分别转化为2和4变为c2列，删除c1，新增c2

data['c2'] = np.where(data.c1 == 'short',2,4)

del data['c1']

c1列有两个值’true’和’false’，分别转化为2和4变为c2列，删除c1，新增c2

data['c2'] = (data.c1 =='true').astype(np.int)

del data['c1']

weekday2index = {"Monday":1,"Tuesday":2,"Wednesday":3,"Thursday":4,"Friday":5,"Saturday":6,"Sunday":7}

emails["weekday"] = emails.weekday.map(weekday2index)

# rename long column names to shorter names, make it easier to read

emails.rename(columns={'user\_past\_purchases':'purchases','user\_country':'country'},inplace=True)

#### 5.2 Delete redundant features

* **plot the feature-feature pair correlation**

#df.dropna(axis=0,how='any')本列是否有NaN，若有，删去该NaN的行

fig, ax = plt.subplots(figsize=(12, 10))

sns.heatmap(data.dropna(axis=0, how='any').corr(), ax=ax)

plt.show()

#### 5.3 Delete features which are not related to the target variable

#将c1列和target列label encoding后，再计算每个feature和target间的相似性

#Print: feature name + chi-squared score+Fscore

resp\_lb\_encoder = LabelEncoder()

cnty\_lb\_encoder = LabelEncoder()

target = resp\_lb\_encoder.fit\_transform(target)

features['c1'] = cnty\_lb\_encoder.fit\_transform(features.c1)

chi2scores,\_ = chi2(features, target) #计算每个特征和target的chi-squared score

fscores, pvalues = f\_classif(features, target) #计算每个特征和target的Fscore

feat\_scores = pd.DataFrame({'chi2scores': chi2scores, 'fscores': fscores, 'chi2\_pvalue':pvalues },index= names) # names是列名，可以通过获取data.column，然后names=['c1', 'c2', 'c3']

feat\_scores.sort\_values(by='chi2scores',ascending=False)

#### 5.4 Create new features that may be related to the target variable

Sum, difference, product, division

### Modelling

#### Check imbalanced classes

print(data[data[‘target’] ==1][ ‘target’] .count())

print(data[data[‘target’] ==0][ ‘target’] .count())

#### 6.1 Logistic Regression

|  |
| --- |
| **Logistic Regression** |
| Logistic Regression LR  import numpy as np  import pandas as pd  import seaborn as sns  import matplotlib.pyplot as plt  from sklearn.preprocessing import LabelEncoder  from sklearn.preprocessing import StandardScaler  from sklearn.model\_selection import train\_test\_split  from sklearn.linear\_model import LogisticRegression  from sklearn.linear\_model import LogisticRegressionCV #crossvalidation+ LR,之后其他都一样  #不适用imbalanced classes严重的，features是int,float，不能是object,categorical  ss = StandardScaler() #对训练和测试数据的特征归一化  X\_train = ss.fit\_transform(X\_train)  X\_test = ss.transform(X\_test)  #LR = LogisticRegressionCV(cv=5, random\_state=0)  LR = LogisticRegression()  lrmodel=LR.fit(X\_train,y\_train)  lrpred=lrmodel.predict(X\_test) #predicted laber on test dataset  lrpred\_prob=lrmodel.predict\_proba(X\_test)[:,1] #predicted probability on test dataset，因为会产生对label0和label1的预测概率，只取label1的概率即可  LR主要看每个特征的p\_value and coefficients, p\_values越小，该特征对于target越重要，其次看系数，系数绝对值越大越好。系数为正表明和特征正相关。负值绝对值大也好  print(lrmodel.coef\_) #coefficients  feature\_selection库的SelectKBest类查看每个特征的pvalue |

#### 6.2 Linear Regression

|  |
| --- |
| **Linear Regression** |
| 用statsmodels来做linear regression  每一个linear regression will output cooefficient+p\_value  import statsmodels.api as sm  data['intercept'] = 1    #build the linear regression  lr = sm.OLS(data['target'],data(‘feature’)).fit()  coefficient=lr.params[0]  p\_value=lr.pvalues[0]  下面是用sklearn来做linear regression  import numpy as np  import pandas as pd  import sklearn  from sklearn.preprocessing import StandardScaler  from sklearn.linear\_model import LinearRegression  from sklearn.model\_selection import train\_test\_split  import matplotlib as mpl  import matplotlib.pyplot as plt  plt.style.use('ggplot')  import seaborn as sns  import warnings  warnings.filterwarnings('ignore')  #data statistics  data.head()  data.info()  data.describe()  data cleaning  #make a bar plot 看the total number of missing data for each input variable  import missingno as mis  mis.bar(data,labels=True)  data visualization  1. plot and show the relationship between the input variable and the target variable.  可以看出，第3幅图的特征和target variable relationship不大  sns.pairplot(data,x\_vars=['TV','Radio','Newspaper'],y\_vars='Sales',size=7,aspect=0.8)    Data splitting, feature scaling  x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.3, random\_state=2)  用statsmodel来build the model  import statsmodels.api as sm  X\_train=sm.add\_constant(X\_train)  model=sm.OLS(y\_train,X\_train).fit()  #调用summary，查看模型的各项指标值  print(model.summary())  summary可以看到如下信息  1.R-squared, (0, 1)之间，越接近1，model越好  2.F-statistic和F\_theory， F-statistic>>F\_theory,好  3.coef: 回归系数  4.对每一个input variable, p\_value越小越好，如果>=0.05，则说明该input variable和target variable关系不大  5.Durbin-Watson，用于No autocorrelation of residuals检验，如果Durbin-Watson约为2，则residuals之间不太相关，满足。  LR存在一些假设，构建模型后测试这些假设是否符合  1.The residuals should be normally distributed.  画图看residuals distribution，是否是正态分布。  model=sm.OLS(y\_train,X\_train.iloc[:,0:-1]).fit()  import scipy.stats as stats  sns.distplot(model.resid,bins=10,fit=stats.norm,norm\_hist=True,  hist\_kws={'color':'steelblue','edgecolor':'black'},  kde\_kws={'color':'black','linestyle':'--'},  fit\_kws={'color':'red','linestyle':'--'})  plt.xlabel('residual',fontsize=14)  核密度曲线与正态密度曲线的趋势比较吻合，故直观上可认为误差项服从正态分布。    也可以用Q-Q Plot里，散点会近似的落在一条直线上，如下图    2.No perfect multicollinearity input variables之间不存在线性关系。  VIF<3即符合  from statsmodels.stats.outliers\_influence import variance\_inflation\_factor  X=sm.add\_constant(data.loc[:,['特征1','特征2']])  vif=pd.DataFrame()  vif['features']=X.columns  vif['VIF Factor']=[variance\_inflation\_factor(X.values,i) for i in range(X.shape[1])]  vif  3. Residuals are independent of each other. 彼此不相关 This is applicable especially for time series data. When the residuals are autocorrelated, it means that the current value is dependent of the previous (historic) values。  如果Durbin-Watson约为2，如2.03等，则residuals之间不太相关，满足。  4.方差齐性是指要求模型残差项的方差  plot residual vs 特征1, residual vs 特征2  plt.figure(figsize=(30,6))  plt.subplot(121)  plt.scatter(X\_train.loc[:,'特征1'],(model.resid-model.resid.mean())/model.resid.std())  plt.hlines(y=0,xmin=X\_train.loc[:,'特征1'].min(),xmax=X\_train.loc[:,'特征1'].max())  plt.xlabel('特征1',fontsize=24)  plt.ylabel('std\_residual',fontsize=24)  plt.subplot(122)  plt.scatter(X\_train.loc[:,'特征2'],(model.resid-model.resid.mean())/model.resid.std())  plt.hlines(y=0,xmin=X\_train.loc[:,'特征2'].min(),xmax=X\_train.loc[:,'特征2'].max())  plt.xlabel(特征2',fontsize=24)  plt.ylabel('std\_residual',fontsize=24)  由图可知，残差几乎均匀地分布在参考线y=0的附近，满足假设。    满足所有假设条件后，可以用trained model , make predictions  X\_test=sm.add\_constant(X\_test)  y\_pred=model.predict(X\_test.iloc[:,0:-1])  plt.figure(figsize=(10,6))  plt.scatter(y\_test,y\_pred)  plt.plot([y\_test.min(),y\_test.max()],[y\_pred.min(),y\_pred.max()],color='blue',linestyle='--')  plt.xlabel('y\_true',fontsize=14)  plt.ylabel('y\_predict',fontsize=14)  plt.show()  Feature scaling, 用scikit learn 来build the model.  ss = StandardScaler()  x\_train\_s = ss.fit\_transform(x\_train)  x\_test\_s = ss.transform(x\_test)    lrmodel = LinearRegression().fit(x\_train\_s,y\_train)  y\_predict=lrmodel.predict(x\_test\_s)  print( lrmodel.score(x\_test\_s,y\_test) ) # 拟合程度，1.0最高  print( lrmodel.coef\_ ) #系数  print( lrmodel.intercept\_ ) #截距  #plotting two curve, ‘actual target value’ vs ‘predicted target value’  t=np.arange(len(x\_test\_s))  plt.figure(facecolor='w')  plt.plot(t, y\_test, 'r-', linewidth=2, label='actual target value')  plt.plot(t, y\_predict, 'g-', linewidth=1, label='predicted target value')  plt.legend(loc = 'upper left')  plt.title("actual target value vs predicted target value", fontsize=20)  plt.grid(b=True)  plt.show() |

#### 6.2 Linear Regression （加强版，包括）

#### 6.3 LightGBM

|  |
| --- |
| **LightGBM** |
| **Windows里jupyter安装LightGBM模块，进入anaconda prompt pip install lightgbm**  Data splitting  features和target是从data中取出的特征和label  features\_train, features\_test, target\_train, target\_test = train\_test\_split(featuers, target, test\_size=0.25, stratify=y, random\_state=42)  # create LightGBM dataset,注意categorical\_feature是data中的categorical 类型的所有features，target不能是categorical类型的，如果是，需要通过labelencoding转化  tree\_train=lgb.Dataset(data=features\_train, label=target\_train, categorical\_feature=categorical\_feature, free\_raw\_data=False)  Cross validation and find optimal parameters  parameters= {'learning\_rate': 0.01,  'boosting\_type': 'gbdt',  'objective': 'binary',  'metric': ['binary\_logloss', 'auc'],  'sub\_feature':0.5,  'num\_leaves': 31,  'min\_data': 50,  'max\_depth': 30,  'is\_unbalance': True}  history = lgb.cv(parameters, train\_set=tree\_train, num\_boost\_round=1000, nfold=5,  early\_stopping\_rounds=20, seed=42, verbose\_eval=False)  print('Best rounds:\t', len(history['auc-mean']))  Best rounds: 767 #an optimal parameter: run 767 rounds    Re-train the model and make predictions on test dataset  clf = lgb.train(parameters, train\_set=d\_train, num\_boost\_round=767)  pred = clf.predict(features\_test)  Plotting feature importance  features = clf.feature\_name()  importance = clf.feature\_importance()  fig, ax = plt.subplots(figsize=(10, 8))  lgb.plot\_importance(clf, ax=ax, height=0.5)  plt.show()  画图threshold vs myaccuracy,找到an optimal threshold  #given 100 thresholds between (0, 1)  thresholds = list(np.linspace(0, 1, 100)) ☺  Calculate predicted labels, given a threshold  pred\_label =(pred > threshold).astype(int) #pre是根据算法预测出来的label的概率  accuracy=根据pred\_label计算预测的精确度（如果不是已有的accuracy,auc等）  对于每一个threshold都预测一个精确度，所有精确度存入myaccuracy  myaccuracy.append(accuracy)  fig, ax = plt.subplots(figsize=(8, 6))  ax.plot(thresholds, pred\_label, label='New Model')  ax.set\_xlabel('Threshold', fontsize=12)  ax.set\_ylabel('Profit', fontsize=12)  ax.legend(fontsize=12)  plt.tight\_layout()  plt.show() |

#### 6.4 Random Forest

|  |
| --- |
| **Random Forest** |
| **Windows里jupyter安装seaborn模块绘图，进入anaconda prompt**  **pip install seaborn**  Compare 3 models   * Base line RF * Baseline RF with PCA (降维) * RF with PCA (降维)+ parameter selection (RandomizedSearch+gridsearch)   #data splitting  #stratify = y 以确保训练集和测试集与原始数据集的 0 和 1 的比例一致。  #设定random\_state后，每次拆分的训练集和测试集也是相同的。  #data 包括features 和target  train\_features, test\_features, train\_target, test\_target = train\_test\_split(features, target, test\_size=0.3, random\_state = 2020, stratify=y)  #data normalization  from sklearn.preprocessing import StandardScaler  ss = StandardScaler()  train\_features\_scaled = ss.fit\_transform(train\_features)  test\_features\_scaled = ss.transform(test\_features)  train\_target = train\_target.values  #modelling and display the recall\_score of the model  from sklearn.ensemble import RandomForestClassifier  from sklearn.metrics import recall\_score    rf = RandomForestClassifier()  rfmodel =rf.fit(train\_features\_scaled, train\_target)      rfpred=rfmodel.predict(test\_features\_scaled) #predicted laber on test dataset  rfpred\_prob=rfmodel.predict\_proba(test\_features\_scaled)[:,1] #predicted probability on test dataset    print(accuracy\_score(test\_target,rfpred)) #default threshold 0.5  print(precision\_score(test\_target,rfpred))  print(roc\_auc\_score(test\_target,rfpred\_prob))  #feature importance using chart and table 服务于降维，之后再用PCA double check  feats = {}  for feature, importance in zip(data.columns, rf.feature\_importances\_):  feats[feature] = importance  #第一列为索引，第二列重命名为gini-importance  importances = pd.DataFrame.from\_dict(feats, orient='index').rename(columns={0: 'Gini-Importance'})  #按照gini-importance降序排序  importances = importances.sort\_values(by='Gini-Importance', ascending=False)  #新增索引列，原来的索引列为第一列  importances = importances.reset\_index()  #原来的索引列重命名为features  importances = importances.rename(columns={'index': 'Features'})  #画,3列，第一列索引，第二列特征名称，第三列特征重要性值  sns.set(font\_scale = 5)  sns.set(style="whitegrid", color\_codes=True, font\_scale = 1.7)  fig, ax = plt.subplots()  fig.set\_size\_inches(30,15)  sns.barplot(x=importances['Gini-Importance'], y=importances['Features'], data=importances, color='skyblue')  plt.xlabel('Importance', fontsize=25, weight = 'bold')  plt.ylabel('Features', fontsize=25, weight = 'bold')  plt.title('Feature Importance', fontsize=25, weight = 'bold')  display(plt.show())  print(importances)  #find the number of components for PCA. find the most important compents 降到多少维即the number of components，性能不再显著提升即cvr不再明显增长  import matplotlib.pyplot as plt  import seaborn as sns  from sklearn.decomposition import PCA  pca\_test = PCA(n\_components=30)  pca\_test.fit(train\_features\_scaled)  #plot a curve, no. of features vs cvr  sns.set(style='whitegrid')  plt.plot(np.cumsum(pca\_test.explained\_variance\_ratio\_))  plt.xlabel('number of features')  plt.ylabel('cumulative explained variance')  plt.axvline(linewidth=4, color='r', linestyle = '--', x=10, ymin=0, ymax=1)  display(plt.show())  #visualize the no. of features and the cvr, 大概多少个features之后，cvr不再明显增长，则不再多选feature  cvr = np.cumsum(pca\_test.explained\_variance\_ratio\_)  pca\_df = pd.DataFrame()  pca\_df['Cumulative Variance Ratio'] = cvr  display(pca\_df.head(10))  #apply PCA on训练数据和测试数据，降到10维，retrain using RF  pca = PCA(n\_components=10)  pca.fit(train\_features\_scaled)  train\_features\_scaled\_pca = pca.transform(train\_features\_scaled)  test\_features\_scaled\_pca = pca.transform(test\_features\_scaled)  rf2= RandomForestClassifier()  rf2.fit(train\_features\_scaled\_pca, train\_target)  print(rf2.score(train\_features\_scaled\_pca, train\_target)) # recall\_score  the third model  parameter selection using RandomSearchCV  from sklearn.model\_selection import RandomizedSearchCV  n\_estimators = [int(x) for x in np.linspace(start = 100, stop = 1000, num = 10)]  max\_features = ['log2', 'sqrt']  max\_depth = [int(x) for x in np.linspace(start = 1, stop = 15, num = 15)]  min\_samples\_split = [int(x) for x in np.linspace(start = 2, stop = 50, num = 10)]  min\_samples\_leaf = [int(x) for x in np.linspace(start = 2, stop = 50, num = 10)]  bootstrap = [True, False]  param\_dist = {'n\_estimators': n\_estimators,  'max\_features': max\_features,  'max\_depth': max\_depth,  'min\_samples\_split': min\_samples\_split,  'min\_samples\_leaf': min\_samples\_leaf,  'bootstrap': bootstrap}  rf3= RandomForestClassifier()  #迭代100次，测试100组参数组合，3folds crossvalidation,构造3\*100=300个森林，n\_jobs=-1使用所有CPU  rs = RandomizedSearchCV(rf3, param\_dist, n\_iter = 100, cv = 3, verbose = 1,  n\_jobs=-1, random\_state=0)  rs.fit(train\_features\_scaled\_pca, train\_target) #用最好的参数在训练集上重新训练  rs.best\_params\_ #print best parameters      ————————————————————————————————————————————  # {'n\_estimators': 700,  # 'min\_samples\_split': 2,  # 'min\_samples\_leaf': 2,  # 'max\_features': 'log2',  # 'max\_depth': 11,  找到每个参数取值范围，使得score值最大。对每个参数，画一幅图。mean\_test\_score vs 参数。  fig, axs = plt.subplots(ncols=3, nrows=2)  sns.set(style="whitegrid", color\_codes=True, font\_scale = 2)  fig.set\_size\_inches(30,25)sns.barplot(x='param\_n\_estimators', y='mean\_test\_score', data=rs\_df, ax=axs[0,0], color='lightgrey')  axs[0,0].set\_ylim([.83,.93])axs[0,0].set\_title(label = 'n\_estimators', size=30, weight='bold')sns.barplot(x='param\_min\_samples\_split', y='mean\_test\_score', data=rs\_df, ax=axs[0,1], color='coral')  axs[0,1].set\_ylim([.85,.93])axs[0,1].set\_title(label = 'min\_samples\_split', size=30, weight='bold')sns.barplot(x='param\_min\_samples\_leaf', y='mean\_test\_score', data=rs\_df, ax=axs[0,2], color='lightgreen')  axs[0,2].set\_ylim([.80,.93])axs[0,2].set\_title(label = 'min\_samples\_leaf', size=30, weight='bold')sns.barplot(x='param\_max\_features', y='mean\_test\_score', data=rs\_df, ax=axs[1,0], color='wheat')  axs[1,0].set\_ylim([.88,.92])axs[1,0].set\_title(label = 'max\_features', size=30, weight='bold')sns.barplot(x='param\_max\_depth', y='mean\_test\_score', data=rs\_df, ax=axs[1,1], color='lightpink')  axs[1,1].set\_ylim([.80,.93])axs[1,1].set\_title(label = 'max\_depth', size=30, weight='bold')sns.barplot(x='param\_bootstrap',y='mean\_test\_score', data=rs\_df, ax=axs[1,2], color='skyblue')  axs[1,2].set\_ylim([.88,.92])  通过上面的图，了解每个参数取值对score的影响。  n\_estimators：300、500、700 的平均分数几乎最高；  min\_samples\_split：较小的值（如 2 和 7）得分较高。23 处得分也很高。我们可以尝试一些大于 2 的值，以及 23 附近的值；   min\_samples\_leaf：较小的值可能得到更高的分，我们可以尝试使用 2–7 之间的值；   max\_features：「sqrt」具有最高平均分；   max\_depth：没有明确的结果，但是 2、3、7、11、15 的效果很好；  bootstrap：「False」具有最高平均分。  第 2 轮超参数调整：GridSearchCV  使用 RandomSearchCV 之后，对于相同的参数，缩小参数值的搜索范围，使用 GridSearchCV 进行更精确的搜索。  6 个参数搜索 10 个不同的参数值， 3 折交叉验证，拟合模型 3,000,000 次！  parameter selection using GridSearchCV  from sklearn.model\_selection import GridSearchCV  n\_estimators = [300,500,700]  max\_features = ['sqrt']  max\_depth = [2,3,7,11,15]  min\_samples\_split = [2,3,4,22,23,24]  min\_samples\_leaf = [2,3,4,5,6,7]  bootstrap = [False]  param\_grid = {'n\_estimators': n\_estimators,  'max\_features': max\_features,  'max\_depth': max\_depth,  'min\_samples\_split': min\_samples\_split,  'min\_samples\_leaf': min\_samples\_leaf,  'bootstrap': bootstrap}  rf4= RandomForestClassifier()  #这里作者没有使用scoring参数，在一些情况下，sklearn中没有现成的scoring函数，需要自己定义  gs = GridSearchCV(rf4, param\_grid, cv = 3, verbose = 1, n\_jobs=-1)  gs.fit(train\_features\_scaled\_pca, train\_target)  rf4 = gs.best\_estimator\_ #最好的model  gs.best\_params\_ #最好的参数组合，返回最优的精度gs.best\_score\_, gs.v\_results\_ 返回结果如参数等  #打印3个model的recall score,分别对positive, negative的number of correctly predicted observations  pred = rf.predict(test\_features\_scaled)  pred2= rf2.predict(test\_features\_scaled \_pca)  pred3 = gs.best\_estimator\_.predict(test\_features\_scaled\_pca)  from sklearn.metrics import confusion\_matrix  conf\_matrix\_baseline = pd.DataFrame(confusion\_matrix(y\_test, y\_pred), index = ['actual 0', 'actual 1'], columns = ['predicted 0', 'predicted 1'])  conf\_matrix\_baseline\_pca = pd.DataFrame(confusion\_matrix(y\_test, y\_pred\_pca), index = ['actual 0', 'actual 1'], columns = ['predicted 0', 'predicted 1'])  conf\_matrix\_tuned\_pca = pd.DataFrame(confusion\_matrix(y\_test, y\_pred\_gs), index = ['actual 0', 'actual 1'], columns = ['predicted 0', 'predicted 1'])  display(conf\_matrix\_baseline)  display('Baseline Random Forest recall score', recall\_score(test\_target, pred))  display(conf\_matrix\_baseline\_pca)  display('Baseline Random Forest With PCA recall score', recall\_score(test\_target, pred2))  display(conf\_matrix\_tuned\_pca)  display('Hyperparameter Tuned Random Forest With PCA Reduced Dimensionality recall score', recall\_score(test\_target, pred3)) |

#### 6.5 Decision Tree

|  |
| --- |
| **Decision Tree** |
| clf = DecisionTreeClassifier(max\_depth=3, min\_samples\_leaf=30, random\_state=42)  clf = clf.fit(X=train\_features, y=train\_target)    # Visualization 画图看生成的DT整个过程  features = list(train\_features.columns)  #target 的两个值0/1，如果想在树叶子节点显示的两个值为'Not quit'和 'Quit'，则 targets = ['Not quit', 'Quit']  targets = decision\_tree\_classifier.classes\_  dot\_data = tree.export\_graphviz(clf, out\_file=None, feature\_names=features, class\_names=targets,  filled=True, rounded=True, special\_characters=True, )  graph = graphviz.Source(dot\_data)  graph  #View feature importance in a descending order default  importance = sorted(zip(features, clf.feature\_importances\_), key=lambda x:x[1], reverse=True)  for feature, importance\_val in importance:  print('{0:10s} | {1:.5f}'.format(feature, importance\_val)) |

#### 6.6 PCA

|  |
| --- |
| **PCA** |
| Reduce dimensionality  # normalize the data  scaler = StandardScaler()  norm\_features = scaler.fit\_transform(features)  # apply PCA降维2维，feature为没有label的数据  pca = PCA(n\_components=2, random\_state=42)  new\_features= pca.fit\_transform(norm\_features)  visualize the new data after implementing PCA  fig, ax = plt.subplots(figsize=(10, 8))  ax.plot(new\_features[:, 0], new\_features[:, 1], '.', markersize=1)  ax.set\_xlabel('PCA Component 1', fontsize=12)  ax.set\_ylabel('PCA Component 2', fontsize=12)  plt.show() |

#### 6.7 K-Means

|  |
| --- |
| **K-Means** |
| import warnings  warnings.simplefilter('ignore')  import numpy as np  import pandas as pd  import seaborn as sns  import matplotlib.pyplot as plt  from sklearn.preprocessing import StandardScaler  from sklearn.decomposition import PCA  from sklearn.cluster import KMeans  from sklearn.metrics import silhouette\_score  找最佳k值  画图看no. of clusters vs silhouette\_score，找到silhouette\_score越接近1的k值  #feature为没有label的数据，最好data normalize一下  silhouettes = []  for k in range(2, 30): #尝试k=2~30  kmodel = KMeans(n\_clusters=k, init='k-means++', random\_state=42, n\_jobs=-1) .fit(feature)  label = kmodel.predict(feature) //计算silhouette非常非常慢，跑一晚上没跑出来    silhouettes.append(silhouette\_score(feature, label))  #开始画图  fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(18, 6))  ax[1].plot(range(2, 30), silhouettes, 'o-', label='Silhouette Coefficient')  ax[1].grid(True)  plt.legend(fontsize=12)  plt.tight\_layout()  plt.show()  用最佳的k值，聚类并预测  kmeans = KMeans(n\_clusters=10, init='k-means++', random\_state=42, n\_jobs=-1) #分为10个类  kmeans = kmeans.fit(feature)  label = kmeans.predict(feature)  画图，散点图，不同的类不同颜色  #5个类，label值为0,1,2…4。每个类的点颜色分别为r,b,y,g,c,k(black)  color = ('r', 'b','y','g','c','k')  label\_color = np.array(color)[label5]  plt.scatter(pca\_x[:, 0], pca\_x[:, 1], c=label\_color)  plt.show()  print(len(pca\_x[label5==0])) #每一类点的数量  print(len(pca\_x[label5==1] ))  print(len(pca\_x[label5==2] ))  print(len(pca\_x[label5==3] ))  print(len(pca\_x[label5==4] )) |

#### 6.7 IsolationForest

|  |
| --- |
| **IsolationForest** |
| import warnings  warnings.simplefilter('ignore')  import numpy as np  import pandas as pd  import seaborn as sns  import matplotlib.pyplot as plt  from sklearn.preprocessing import StandardScaler  from sklearn.decomposition import PCA  from sklearn.ensemble import IsolationForest  IsolationForest is an ensemble method. It is linear 具有线性时间复杂度。是ensemble的方法，所以可以用在含有海量数据的数据集上面。  iForest不适用于特别高维的数据。所以可以用PCA来降维后使用。‘fraudulant transaction identification’项目中，十几分钟就跑完了。但是DBscan和kmeans时间都很长。  #norm\_feature是归一化后的特征  forest = IsolationForest(n\_estimators=100, max\_samples='auto', contamination=0.1,  max\_features=1.0, bootstrap=False, n\_jobs=-1, random\_state=42).fit(norm\_feature)    score = forest.decision\_function(norm\_feature)  label = forest.predict(norm\_feature) # label：+1 表示正常样本， -1表示异常样本。就是一个一维数据，然而kmeans是0,1,2…没有负值  #画图用不同颜色显示two classes。  #需要用到PCA降到2维的数据pca\_x（归一化后再降维），第一维和第二维分别为pca\_x[:, 0]和pca\_x[:, 1]，label会对应到pac\_x中  plt.scatter(pca\_x[:, 0], pca\_x[:, 1], c=label)  plt.show()  print(len(pca\_x[label5==-1])) #每一类点的数量  print(len(pca\_x[label5==1] ))  降维  pca = PCA(n\_components=2, random\_state=42)  pca\_x = pca.fit\_transform(norm\_feature) |

#### 6.8 Time series prediction

|  |
| --- |
| **Time series prediction Prophet ‘profit’** |
| Windows系统下安装Python版本的Prophet  import matplotlib.dates as mdates  from matplotlib import rcParams    #预测data1中2016-12-15的c1的值，目前c1的值肯定在2016-12-15前  data1= data[['date', 'clicks']]  interval= (pandas.to\_datetime("2016-12-15")-data1['date'].max()).days #间隔多少天  ts = fbprophet.Prophet() #modelling  ts.fit(data1)  future\_data = ts.make\_future\_dataframe(periods = interval)  pred = ts.predict(future\_data) #the predicted the no. of clicks for the interval    #get the predicted value for the last day 2016-12-15  pred\_lastday=pred['yhat'].iat[-1]  #plot date vs 'clicks' for previous period+interval  ts.plot(pred)  plt.show() |

#### 6.9 XGB

|  |
| --- |
| **XGB** |
| # split for training and testing  train\_features,test\_features,train\_target,test\_target = train\_test\_split(features,taraget,test\_size=0.33333,random\_state = seed)  train\_matrix = xgb.DMatrix(train\_features,train\_target)  test\_matrix = xgb.DMatrix(test\_features)  params = {}  params['objective'] = 'binary:logistic' # output probabilities  params['eval\_metric'] = 'auc'  params["num\_rounds"] = 300  params["early\_stopping\_rounds"] = 30  # params['min\_child\_weight'] = 2  params['max\_depth'] = 6  params['eta'] = 0.1  params["subsample"] = 0.8  params["colsample\_bytree"] = 0.8  cv\_results = xgb.cv(params,train\_matrix,  num\_boost\_round = params["num\_rounds"],  nfold = params.get('nfold',5),  metrics = params['eval\_metric'],  early\_stopping\_rounds = params["early\_stopping\_rounds"],  verbose\_eval = True,  seed = seed)  n\_best\_trees = cv\_results.shape[0]  print "best number of trees: {}".format(n\_best\_trees) # output 53  watchlist = [(train\_matrix, 'train')]  gbt = xgb.train(params, train\_matrix, n\_best\_trees,watchlist)  # plot feature importances  xgb.plot\_importance(gbt)  Plotting ROC curve  def validation\_roc():  train\_features1,valid\_features,train\_target1,valid\_target = train\_test\_split(train\_features,train\_target,test\_size=0.2,random\_state=seed)    train\_data = xgb.DMatrix(train\_features1,train\_target1)  valid\_data = xgb.DMatrix(valid\_features)    # retrain on training set  xgb\_train = xgb.train(params, train\_data, n\_best\_trees)    # predict on validation set  valid\_probas = xgb\_train.predict(valid\_data, ntree\_limit=n\_best\_trees)    d = {}  d['FPR'],d['TPR'],d['Threshold'] = roc\_curve(valid\_target,valid\_probas)  return pd.DataFrame(d)  roc\_results = validation\_roc()  \_ = plt.figure()  plt.plot(roc\_results.FPR,roc\_results.TPR)  plt.xlabel("FPR")  plt.ylabel('TPR')  roc\_results.loc[(roc\_results.TPR > 0.6) & (roc\_results.TPR < 0.65),:] |

#### 6.10 evaluation metrics

from **sklearn**.metrics import auc, roc\_curve, classification\_report, precision\_score, recall\_score,

accuracy\_score, roc\_auc\_score

**accuracy：**

accuracy = accuracy\_score(actual\_target,predict\_target) #真实的label，预测的label

**calculate recall**： minimize FN

recall\_score(actual\_target,predict\_target) #真实的label，预测的label

**calculate precision:** minimize FP

precision\_score(actual\_target,predict\_target) #真实的label，预测的label

**calculate auc：**

auc=roc\_auc\_score(actual\_target,predict\_target\_prob) #真实的label，预测的label概率

#### 6.11 plot roc curve and find the best threshold

#the intersection between the line y=-x+1 and the roc curve

from sklearn.metrics import roc\_auc\_score,roc\_curve

from matplotlib import pyplot

#label\_proba是LR，RF预测的label的概率，如lrpred\_prob=lrmodel.predict\_proba(X\_test)[:,1]

#test\_target为test dataset的target

fpr, tpr, thresholds = roc\_curve(test\_target, label\_proba)

pyplot.plot(fpr, tpr, marker='.', label='Logistic')

pyplot.xlabel('False Positive Rate')

pyplot.ylabel('True Positive Rate')

pyplot.legend()

pyplot.show()

#找到最佳cut-off 即thresholds[i]

for i in range(len(fpr)):

if fpr[i] + tpr[i] >= 1: #最靠近left top的那个threshold

i = i -1

break

#the best threshold is (fpr[i], tpr[i])的thresholds[i]

print(thresholds[i])

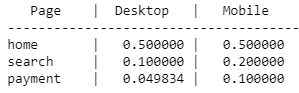
### Print

print('{0:^10s} | {1:^10s} | {2:^10s}'.format('Page', 'Desktop', 'Mobile'))#打印page | desktop

print('-' \* 40) #打印--------

for name, desk\_rate, mobile\_rate in zip(names, desk\_rates, mobile\_rates): #逐行打印

print('{0:10s} | {1:10.6f} | {2:10.6f}'.format(name, desk\_rate, mobile\_rate))



### Visualization

#### 8.1Feature distribution + feature vs target, feature是device

* **一行两幅图，feature是categorical，类似age这种也能用，就是bar比较密集**

1. (feature distribution)

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(16, 6))

sns.countplot(x='device', data=data, ax=ax[0])

ax[0].set\_xlabel('device', fontsize=12)

ax[0].set\_ylabel('Count', fontsize=12)

ax[0].set\_title('different devices distribution', fontsize=16)

1. (feature vs target)

sns.barplot(x='device', y='target', data=data, ax=ax[1])

ax[1].set\_xlabel('device', fontsize=12)

ax[1].set\_ylabel('target', fontsize=12)

ax[1].set\_title('device vs. target', fontsize=16)

plt.tight\_layout()

plt.show()

* **一行两幅图，feature是continuous**

1. (feature distribution) bar+curve

#sns.distplot绘制a histogram with a line on it

hist\_kws={'histtype': 'bar', 'edgecolor':'black', 'alpha': 0.2}

warnings.filterwarnings('ignore')

#feature c1 distribution feature c1在target取0和取1时的两个分布，每个分布都带有一条拟合曲线

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(16, 6))

sns.distplot(data[data['target'] == 0]['c1'], label='taraget = 0', ax=ax[0], hist\_kws=hist\_kws)

sns.distplot(data[data['target'] == 1]['c1'], label='target = 1', ax=ax[0], hist\_kws=hist\_kws)

ax[0].set\_title('Histogram of c1', fontsize=16)

ax[0].legend()

* (feature distribution) **三行，每行一幅图，同横坐标，便于对比**

#每幅图40个bar，横坐标是1~40，纵坐标是c1的值，c1有40个值。本来是c1值的数量，但是因为指明40个bar，显然每个横坐标对应1个值，这样统计总数量没有意义

hist\_kws={'histtype': 'bar', 'edgecolor':'black', 'alpha': 0.2}

fig, ax = plt.subplots(nrows=3, ncols=1, figsize=(12, 10), sharex=True)

sns.distplot(data['c1'], bins=40, ax=ax[0], label='25%', hist\_kws=hist\_kws)

ax[0].legend(fontsize=12)

sns.distplot(data['c2'], bins=40, ax=ax[1], label='50%', hist\_kws=hist\_kws)

ax[1].legend(fontsize=12)

sns.distplot(data['c3'], bins=40, ax=ax[2], label='75%', hist\_kws=hist\_kws)

ax[2].legend(fontsize=12)

plt.tight\_layout()

plt.show()

(2) (feature vs target) curve .-

#取出两列c1和target，根据c1分组，求target的均值，如果target是0/1，则求的是target为1的百分率。返回两列c1和target的百分率（但列名依然为'target'），新增索引0,1,2...

data1 = data[['c1', 'target']].groupby('c1').mean().reset\_index()

ax[1].plot(data1['c1'], data1['target'], '.-') #绘制一条线

ax[1].set\_title('c1 vs. Mean target', fontsize=16)

ax[1].set\_xlabel('c1')

ax[1].set\_ylabel('Mean target')

ax[1].grid(True)

plt.show()

#### target vs feature ----4 features 4 charts 2\*2

4幅图，每行2幅，2行。一幅图：特征vs target

plt.style.use(‘fivethirtyeight‘)

fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(ncols=2, nrows=2, figsize=(12, 12))

#对于日期自动旋转60度，比如x-axis是日期类型，则如果横轴密密麻麻，日期重叠，则旋转，比如c1是日期

fig.autofmt\_xdate(rotation=60)

ax1.plot(data[‘c1’], data[‘target ‘], linewidth=4)

ax1.set\_xlabel(‘‘), ax1.set\_ylabel(‘\*\*\* ‘), ax1.set\_title(‘\*\*\*‘)

ax2.plot(data[‘c2’], data[‘target ‘], linewidth=4)

ax2.set\_xlabel(‘‘), ax1.set\_ylabel(‘\*\*\* ‘), ax1.set\_title(‘\*\*\* ‘)

ax3.plot(data[‘c3’], data[‘target ‘], linewidth=4)

ax3.set\_xlabel(‘‘), ax1.set\_ylabel(‘\*\*\* ‘), ax1.set\_title(‘\*\*\* ‘)

ax4.plot(data[‘c4’], data[‘target ‘], linewidth=4)

ax4.set\_xlabel(‘‘), ax1.set\_ylabel(‘\*\*\*‘), ax1.set\_title(‘\*\*\* ‘)

plt.show()

## Coding A/B test

### Import

import numpy as np

import pandas as pd

import seaborn as sns #for plotting

import matplotlib.pyplot as plt #for plotting

from scipy.stats import ttest\_ind #for t-test

### Generate new features

Generate a new feature 'is\_new' representing a new user or an old user.

data['interval'] = (data['c2'] - data['c1']).apply(lambda x: x.days)

data['is\_new'] = (data['interval'] == 0).astype(int)

### Target values distribution on control and test

#Print the target values distribution on control and test.

data.groupby('if\_testgroup')['target'].mean()

#Plot the target values distribution on control and test.

fig, ax = plt.subplots(figsize=(8, 5))

sns.barplot(x='if\_testgroup', y='target', data=data, ax=ax)

ax.set\_xlabel('device', fontsize=12) #横坐标label

ax.set\_ylabel('Number of devices', fontsize=12)

plt.tight\_layout()

plt.show()

### For each feature, feature distribution on control and test + target distribution on control and test

Take the feature device for example, 一行两幅图

#For each feature, feature distribution on control and test

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(18, 6))

#横轴是data中的feature’device’，纵轴是count(device),hue表示double bar每个bar对应'if\_testgroup'一个值

sns.countplot(x='device', hue='if\_testgroup', data=data, ax=ax[0])

ax[0].set\_title('Device distribution on test and target', fontsize=16)

#target distribution on control and test for each feature

sns.barplot(x='device', y='target', hue='if\_testgroup', data=data, ax=ax[1])

ax[1].set\_title('Pages Visited vs. device', fontsize=16)

plt.tight\_layout()

plt.show()

两行两幅图

fig, ax = plt.subplots(nrows=2, ncols=1, figsize=(18, 12))

sns.countplot(x='device', data=data, ax=ax[0])

ax[0].set\_ylabel('Device distribution on test and target', fontsize=12)

sns.barplot(x='device', y='target', data=data, ax=ax[1])

ax[1].set\_ylabel('Pages Visited vs. device', fontsize=12)

plt.tight\_layout()

plt.show()

### T-test for all features

#test这一列表示是否属于control group or test group,target variable is target

control\_target = data[data['if\_testgroup'] == 0]['target'].values

test\_target = data[data[' if\_testgroup'] == 1]['target'].values

print('T-test:\t', ttest\_ind(a=control\_target, b=test\_target, equal\_var=False)) #output t\_value, p\_value

### T-test for each feature

Take the feature ‘device’ for example

def run\_ttest(df):

test\_data = df[df['if\_testgroup'] == 1]['target'].values #或者df.loc[df. if\_testgroup == 0, 'target']

test\_mean = test\_data.mean()

ctrl\_data = df[df['if\_testgroup'] == 0]['target'].values

ctrl\_mean = test\_data.mean()

result = ss.ttest\_ind(ctrl\_data, test\_data, equal\_var=False)

conclusion = 'Significant' if result.pvalue < 0.05 else 'Not Significant'

return pd.Series({'number\_test': len(test\_data),

'number\_ctrl': len(ctrl\_data),

'mean\_test': test\_mean,

'mean\_ctrl': ctrl\_mean,

'mean\_diff': test\_mean - ctrl\_mean,

'pvalue':result.pvalue,

'conclusion':conclusion})

tests.groupby('device').apply(data). .reset\_index()

## Parses a JSON string and converts it to a Pandas DataFrame

Common examples of unstructured data are text, picture, vedio

Common examples of semi-structured data are XML、JSON.

Common examples of structured data are Excel files or SQL databases. They have structured rows and columns that can be sorted.

格式[{ }, { }, ….{ }]

每个{ }里的内容如下：

Use two method pd.read\_json(). The josn file string format有特定的格式要求

json.load 需要自己解析

{"session\_id":["D258NVMV202LS"],

"unix\_timestamp":[1442640552],

"cities":["San Jose CA, Montreal QC"],

"user":[[{"user\_id":5749,"joining\_date":"2015-04-02","country":"FR"}]]}

session\_id timestamp cities user\_id joining\_date country

**import json**

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

#将从.josn文档读入的data，parse data read from a josn file into dataframe

def parse\_json(data): # function to parse json data

session\_id = [] #解析后有几个feature，就初始化几个如session\_id, timestamp, cities…

timestamp = []

cities = []

user\_id = []

joining\_date = []

country = []

for item in data:

session\_id.append(item['session\_id'][0])

timestamp.append(item['unix\_timestamp'][0])

cities.append(item['cities'][0])

user\_id.append(item['user'][0][0]['user\_id'])

joining\_date.append(item['user'][0][0]['joining\_date'])

country.append(item['user'][0][0]['country'])

# create DataFrame

maps = {'session\_id': session\_id, 'timestamp': timestamp, 'cities': cities,

'user\_id': user\_id, 'joining\_date': joining\_date, 'country': country}

columns=['session\_id', 'timestamp', 'cities', 'user\_id', 'joining\_date', 'country']

return pd.DataFrame(maps, columns=columns)

with open('./data/city\_search.json', 'r') as f:

data = json.load(f)

data = parse\_json(data)

data['joining\_date'] = pd.to\_datetime(data['joining\_date'])

### hackerrank

#### 输入

#输入多个整数n, m

n, m = map(int, input().split())

#输入1个整数n

n=int(input())

#输入1个浮点数f

f=float(input())

#输入n行float数据，存入array(如果不让用numpy就不能用这个）

train = np.array([input().split() for \_ in range(n)], float) #训练数据，每一行最后一个数是y，前面是attributes

#输入n行float数据，存入list

mylist = []

for \_ in range(n):

mylist.append( map(float, input().split()) )

## Git

创建（**Repository）**增加内容，提交修改 ，并把修改同步到远程库，常用的命令是git clone、git checkout、git commit、git push、git pull等。

## Machine Learning

### 安装TF，Keras

注意anaconda里python的 版本、 TF版本, keras版本的兼容性

1.构建TF虚拟环境

prompt中输入python，查看python的版本，比如是3.7.6

prompt---conda create -n tfjane python=3.7.6

等待一段时间，可能还要输入y，安装完成后输入activate tfjane或者conda activate tfjane，回车，如果下一行开头出现(tfjane)，则表示环境设置成功，即进入TF环境

2.在TF中安装TF和Keras

TF环境中---conda install tensorflow

完成后，输入python---import tensorflow as tf如果不报错，则继续安装

conda install ipython

conda install jupyter notebook

conda install nb\_conda

继续安装keras

TF环境中--- pip install keras==2.3.1 ---注意这里指定了keras的版本号，一开始可以不指定，可能会报错，因为keras的版本和tensorflow不兼容

完成后，输入python---import keras如果出现‘using tensorflow backend’则成功

### 启动tensorflow环境下的jupyter

anaconda >anaconda prompt---activate tfjane—打开jupyter(tfjane)---new---选择tfjane

不用了，关闭jupyter notebook

anaconda >anaconda prompt---deactivate

### 如何运行TF，Keras

anaconda里Jupyter里运行tensorflow，

1.打开 Anaconda Prompt 终端

2.输入：conda activate tensorflow

3.等待，输入：jupyter notebook

4.不想要写程序的时候，Anaconda Prompt 终端输入deactivate

### MLP (Multiple Layer Perception)

* **MLP with 3 hidden layers for binary classification**

from keras.layers import Dense,LSTM,Dropout

from keras.models import Sequential

from keras import optimizers

#用归一化后的training data训练，接收的是2维数据，LSTM接收3维数据

#2维数据和LightGBM的training data是一样的

#one input layer, three hidden layers, one output layer，2 dropout layers也可以不要。

#如果修改了model.add(layers.Dense...),重新执行model=keras.Sequential()

#网络构建

model = Sequential()

model.add(Dense(16, input\_dim = 11, activation = 'relu'))

model.add(Dropout(0.2)) #deal with overfitting，本层输入32个节点,丢弃20%

model.add(Dense(32, activation = 'relu'))

model.add(Dropout(0.2)) #dropout layer

model.add(Dense(16, activation = 'relu')) #hidden layer 16 nodes

model.add(Dense(1, activation = 'sigmoid')) #output layer，1 node二分类

model.summary()#查看构建的网络，此时并未训练网络

#网络compile

model.compile(optimizer ='adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

#training

#history 记录model训练过程，可以从下面两种中选择

#Training dataset中取20%作为validation dataset

#Test dataset作为validation dataset

选history = model.fit(train\_x, train\_y, batch\_size =256, epochs=10, validation\_split=0.2)

选history = model.fit(x\_train, y\_train, batch\_size =256, epochs=10, validation\_data=(x\_test, y\_test) )

* **MLP 多分类**

与MLP二分类，两个方面不同

1.网络构建中，output layer的参数，

model.add(Dense(3, activation = 'softmax')) #三分类问题

2.compile中的loss参数

#target variable 做one hot encoding，100,010,001, loss='categorical\_crossentropy'

#target variable 做顺序编码，比如用0,1,2表示3个类别,loss='sparse\_categorical\_crossentropy'

model.compile(optimizer ='adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])

* **MLP Regression**

model.compile(optimizer='adam', loss='mse', metrics=['mae'])

* **L2 deal with overfitting**

#自己设置的L2参数0.005，对weights惩罚的参数，这个对性能影响大。

from keras import regularizers

model.add(Dense(32, input\_dim = 11, kernel\_regularizer=regularizers.l2(0.005), activation = 'relu'))

### Prediction

#使用训练好的模型在test data上进行预测

y\_pred\_keras = model.predict(X\_test)

### Evaluation

#使用训练好的模型在training data上的性能评价

loss, accuracy = model.evaluate(X\_train, y\_train, verbose=0)

### LSTM (Long Short Term Memory)

是RNN的一种

* **单层LSTM 网络, binary classification**

#LSTM输入是3维数据

#32个节点(128, 64个节点也行)，input\_shape是数据的后两维，数据的shape是(10000,120,11)

model = keras.Sequential()

model.add(layers.LSTM(32, input\_shape=(train\_x.shape[1:]), activation='tanh'))

model.add(layers.Dense(1)) #output layer

* **3层LSTM网络,binary classification**

3 hidden layers，为什么return\_sequences因为LSTM输入是3维，只有output layer前一层可以不用return\_sequences

model = keras.Sequential()

model.add(layers.LSTM(32, input\_shape=(train\_x.shape[1:]), return\_sequences=True))

model.add(layers.LSTM(32, return\_sequences=True))

model.add(layers.LSTM(32))

model.add(layers.Dense(1)) #output layer

* **3层LSTM网络,,binary classification**

model.compile(。。。。。) #回归问题

#训练中不断降低Learning Rate，连续3个val\_loss没有降低，则学习速率\*factor,当LR降到min\_lr时，不能再降了

LR\_reduction = keras.callbacks.ReduceLROnPlateau(monitor='val\_loss', patience=3, factor=0.5, min\_lr=0.00001)

history = model.fit(train\_x, train\_y, batch\_size = 128, epochs=200, validation\_data=(test\_x, test\_y), callbacks=[LR\_reduction])

### Bidirectional LSTM+recurrent\_dropout文本处理

主要用于text, audio处理，从两个方向读取信息

from keras.layers import Bidirectional

model=keras.Sequential()

model.add(layers.Embedding(10000, 16, input\_length=200))

model.add(layers.Bidirectional(layers.LSTM(64, dropout=0.2, recurrent\_dropout=0.5))) #the first hidden layer

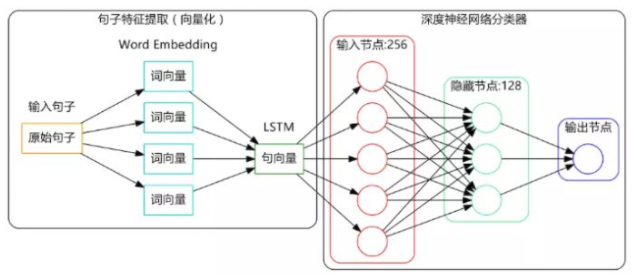
model.add(layers.Dropout(0.5))

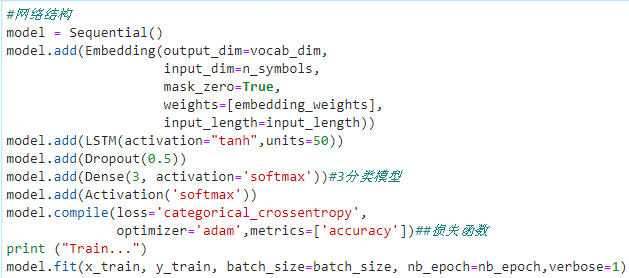
model.add(layers.Dense(1,activation='sigmoid')) #the output layer

recurrent\_dropout，每次删除相同位置的node，我也不太懂

### GRU文本处理

GRU相比LSTM结构更简单，参数更少，因此应用也比较广泛





注意

 inspect imbalanced classes----过采样或者欠采样



## Attention+LSTM文本处理

### 1.安装NTLK, gensim, word2vec, spacy

在TF虚拟环境下装，才能在tfjane中使用即在tensorflow中使用

anaconda >anaconda prompt---tfjane---输入pip install --upgrade gensim

anaconda >anaconda prompt--- tfjane---输入conda install word2vec 或pip…

anaconda >anaconda prompt--- tfjane---输入pip install nltk

anaconda >anaconda prompt--- tfjane---输入**pip install spacy**

conda install -c conda-forge spacy-model-en\_core\_web\_sm #Spacy的英文包

pip install neuralcoref

pip install textacy

注意，anaconda版本新，python版本过高，可能安装不上

测试安装成功 否？

Anaconda prompt---python---import gensim

### 2.启动tensorflow环境下的jupyter

anaconda >anaconda prompt---activate tfjane—打开jupyter---new---选择tfjane

不用了，关闭jupyter notebook

anaconda >anaconda prompt---deactivate

n-gram模型，n=1表示计算一个句子的概率的时候，只考虑每个word与前面的1个word相关，n=2表示。。。通常选择n=2或3或4



### 3.短信数据预处理

delete: 标点符号、括号、问号，只留下字母、数字和字符

将大写字母转化为小写

### 4.每个word转化为vector

从所有短信数据中学习，为每个word产生一个vector

Word2Vec

Glove更优

idea case: Google已经为300万word训练构建了vector,每个vector维度为300, 300-dimensional vectors for 3 million words

这个单词向量矩阵太大了（3.6G）

GloVe 进行训练得到包含 400,000 个word向量，00-dimensional vectors for 400,000 words

### 5. RoBERTa

6.XLNet

7.BiLSTM+Attention

### 8.评价方法

accuracy+Recall+F1-score



### 

### Spacy

### Data preprocessing

初始同一类别的数据在一起，数据预处理的时候，打乱，有利于model training

index=np.random.permutation(len(data)) #乱序化(0，len(data))的数字

data=data.iloc[index] #乱序后的index给data

features=data[data.columns[1:-3]] #取第一列到倒数第4列为features

target=data.iloc[:, 3:]取最后3列为target，target有3个值，多分类