ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

Rapport de project 1A :

data analysis

du 17/03/19 au 12/05/19

Table des matières

2. Presentation of tools and work environment

> - MySQL: Dateset is saved on the MySQL database management system (DBMS) from 1995. It is one of the most used software in the world for database management. In addition, it works on a large number of operating systems including the most popular macOS, Windows and Linux.

> - Navicat for MySQL is the ideal solution for MySQL/MariaDB administration and development. It is a single application that allows you to connect to MySQL and MariaDB databases simultaneously. Compatible with cloud databases like Amazon RDS, Amazon Aurora, Oracle Cloud, Google Cloud and Microsoft Azure. This all-inclusive frontend provides an intuitive and powerful graphical interface for database management, development, and maintenance.

> - scikit-learn:Simple and efficient tools for data mining and data analysis
>   Accessible to everybody, and reusable in various contexts
>   Built on NumPy, SciPy, and matplotlib
>   Open source, commercially usable - BSD license
> - Jupyter: JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. JupyterLab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible and modular: write plugins that add new components and integrate with existing ones.

3.Case1:Refined user management based on RFM

3.1 Case background:

User value segmentation is an important way to understand user value, while sales companies are particularly concerned about order transactions.

Therefore, the value model based on order transactions will be more suitable for operational needs. The common model for transaction data analysis is RFM model. Not only simple, easy to understand, but also capable of landing.
In the RFM results, the business unit hopes not only to group users, but also to summarize and summarize the user characteristics of each group, so as to facilitate the subsequent refinement of different customer groups, and customize according to different groups or Differential marketing and care.

3.2 Introduction

Python module:

> - time(record time)
> - numpy,pandas(data processing)
> - pymysql(read and write MySQL)
> - sklearn(Sklearn-RandomForest)
> - pyecharts(plot)

3.3 Case Data

```
Overview:
      member_ID   num_order update_date  amount_order
0  15278002468  3000304681  2015-01-01         499.0
1  39236378972  3000305791  2015-01-01        2588.0
2  38722039578  3000641787  2015-01-01         498.0
3  11049640063  3000798913  2015-01-01        1572.0
```

> - member_ID means unique ID for every member_ID
> - num_order means order_ID unique
> - update_date means order transaction date
> - amount_order means order amount

3.4 Main application conception

Member value is used to evaluate the value of users. It is an important model and reference basis for distinguishing member values. It is also one of the key indicators to measure different marketing effects. Value model is generally based on trading behavior, which measures the value of entity conversion. behavior,

The RFM model is based on the member's last purchase time R (Recency), purchase frequency F (Frequency), purchase amount M (monetary), calculate the RFM score, and evaluate the customer's order active value through these three dimensions, commonly used to do Customer grouping or value differentiation.

3.5 Member value model

Member value is used to evaluate the value of users. It is an important model and reference basis for distinguishing member values. It is also one of the key indicators to measure different marketing effects. Value model is generally based on trading behavior, which measures the value of entity conversion. Behavior, a commonly used value model is RFM.

The RFM model calculates the RFM score based on the member's last purchase time R (Recency), purchase frequency F (Frequency), and purchase amount M (Monetary). The three dimensions are used to evaluate the customer's order active value, which is often used as a customer. Skirt or value analysis.

The RFM model is based on a fixed point in time for model analysis because each customer gets different data at different time points.

The following is the basic implementation of the RFM model:

1) Set the cutoff time node to be calculated. Used to make data selection and calculation based on this time

2) In the member database, extract the original data set containing the member ID, order time, and order amount of each member. One member may generate multiple order records.

3) Data and calculations. From the order time, find the purchase time of each member from the time node, count the order quantity of each customer as the purchase frequency by using the member ID as the dimension, and sum the order amount of the multiple orders of the user to obtain the total order amount, thereby obtaining R, F, M three raw data quantities.

4) R, F, M partition, for the F, M variable, the larger the value, the higher the purchase frequency, the higher the order amount, but for R, the smaller the value, the closer the cut-off time node, so the value The better, the R, F, M distribution uses the quintile method for data partitioning.

5) Combine or add the three values to get the total RFM score. There are two ways to calculate the total score of the RFM. One is to directly splicing the three values together, for example, the RFM score is 312, 333, 132. Kind is a new summary value that directly adds the three values, such as RFM scores 6, 9, 6

6) After getting the RFM of different members, two application ideas are generated:

1.Based on the three dimension values, the user group is divided and interpreted, and the user's value is analyzed. For example, members with a score of 212 tend to purchase less frequently, and customers with lower purchase frequency should regularly send promotional emails with a score of 321 Members, although the frequency of purchase is high, but the order amount is low, these customers tend to have a high purchase stickiness, and can increase the order amount by considering association or matching sales;

2.Evaluate the value of all members based on the RFM summary score, and can be used for value ranking. The score can also be used as an input dimension along with other dimensions as input variables for other data analysis and mining models, providing a basis for analytical modeling.

3.6 Read data and data Overview

```
sheet_names = ['2015','2016','2017','2018','member_level']
sheet_datas = [pd.read_excel('sales.xlsx',sheet_name=i) for i in
sheet_names]#"sales.xlsx"record members data in the last 4 years

for each_name,each_data in zip(sheet_names,sheet_datas):
#The zip() function take iterables (can be zero or more), makes iterator that
aggregates elements based on the iterables passed, and returns an iterator of
tuples.
    print('[data summary for {0:=^50}]'.format(each_name))
    print('Overview:','\n',each_data.head(4))
    print('DESC:','\n',each_data.describe())
    print('NA records',each_data.isnull().any(axis=1).sum()) # record missing
values
    print('Dtypes',each_data.dtypes)
```

Just show some details,more information,we can check "caseRFM.ipynb"

```
          member_ID     num_order   amount_order
count  3.077400e+04  3.077400e+04   30774.000000
mean   2.918779e+10  4.020414e+09     960.991161
std    1.385333e+10  2.630510e+08    2068.107231
min    2.670000e+02  3.000305e+09       0.500000
25%    1.944122e+10  3.885510e+09      59.000000
50%    3.746545e+10  4.117491e+09     139.000000
75%    3.923593e+10  4.234882e+09     899.000000
max    3.954613e+10  4.282025e+09  111750.000000
```

Observe the data and get the conclusion that the distribution of the order amount is not uniform, and there are obvious maximum values. (more data details ,please check caseRFM.ipynb)

For example, in 2016 data, the maximum value is 174900. The minimum value is only 0.1, so the distribution state, the data will be extremely the impact of the value.

The minimum value of the order amount actually includes the amount of 0, 0.1, which is obviously not a normal order. After confirming with the business party, the maximum amount of the order is valid.

Usually, the customer purchases multiple home electrical appliances at one time, and the order amount is 0.1 RMB. Such orders paid using coupons have no practical meaning and will therefore be removed in subsequent processing.

3.7 data cleaning et merge

```
# Member ID summary
rfm_gb =
data_merge.groupby(['year','member_ID'],as_index=False).agg({'date_interval':
'min',  # Calculate the most recent order time
                                                   'update_date': 'count', #
Calculate order frequency
                                                   'amount_order': 'sum'})  #
Calculate all amount of orders
# rename
rfm_gb.columns =  ['year','member_ID','r','f','m']
rfm_gb.head()
```

function ".agg()" : Aggregate using callable, string, dict, or list of string/callables
after that ,we get a new dataset ,show somethings

```
year    member_ID   r   f   m
0   2015    267 197 2   105.0
1   2015    282 251 1   29.7
2   2015    283 340 1   5398.0
3   2015    343 300 1   118.0
4   2015    525 37  3   213.0
```

Implement aggregate calculation of RFM raw values based on year and member ID.
When doing RFM partitioning, the basic logic is to discretize R, F, and M separately, and then get the score after the score is discretized.
For discretization, saving computing resources, improving computational efficiency, the need for algorithm models, enhancing the stability and accuracy of the model, and discretization of specific data processing and analysis Discrete itself has a variety of options.

view the data distribution:

```
 count         mean          std  min   25%    50%     75%       max
r  148591.0   165.524043   101.988472  0.0  79.0  156.0   255.0     365.0
f  148591.0     1.365002     2.626953  1.0   1.0    1.0     1.0     130.0
m  148591.0  1323.741329  3753.906883  1.5  69.0  189.0  1199.0  206251.8
```

From the basic outline, there are a total of 140,000 pieces of data after aggregation. The data resolution of r and m is relatively discrete, and the data in min, 25%, 50%, 75% and max are not particularly concentrated; The frequency of purchase) can be seen that the distribution of most users is close to 1, the performance is from 1 to 55% of the segment value is 1 and the mean is 1.365.
Here, 25% and 75% are selected as the boundary values of the interval division, but the problem is that r and m themselves can better distinguish the user characteristics, and f can not be distinguished (a large number of users only have one order).

For this problem, we Communicate with the business department, the conclusion is that due to the industry specificity (the mains electricity),
the user's repurchase is really rare. It is more common to buy once in one year, so the division can use 2 and 5 as the boundary, choose 2 is because the general business department thinks that the purchase is 2 times or more in the current year and is defined as the repurchase user. The 5 times is that the business department thinks that the ordinary user purchases 5 this

is already a very high number. If the number of times exceeds the number, it belongs to the very high value user group. .

3.8 define interval boundary

```
# Define interval boundaries
r_bins = [-1,79,255,365] # Note that the starting boundary is less than the
minimum
f_bins = [0,2,5,130]
m_bins = [0,69,1199,206252]
```

Based on the above analysis, two intermediate boundaries:
r and m are obtained by 25% and 75%, respectively, f is defined by the business and data department,
the minimum boundary is smaller than the minimum of each dimension,
the maximum boundary is bigger than each dimension Maximum value.

3.9 Calculate RFM factor weights

When calculating the RFM combination score, we can directly combine the results into a new group, such as 322/132.

However, when weighted sum is used to obtain a new RFM score indicator, a weight value must be determined. When it comes to member data, there is usually a membership system, and another dimension in the membership system is the membership level that measures the value of the member.

When assigning a membership level, company has comprehensively considered a variety of factors related to the company's overall interests. Setting various membership benefits is related to the membership level, such as free shipping thresholds, coupons, membership activities, etc., so we can Based on the membership level to determine the weight of the three RFMs, the basic idea is to establish a classification model of RFM three dimensions and membership levels, and then output the weight of the dimension through the model.

Match member level and rfm score.

```
# Match member level and rfm score
rfm_merge = pd.merge(rfm_gb,sheet_datas[-1],on='member_ID',how='inner')
```

Match the member's order data with the rating data, and merge the two data frames using the merge method. The associated primary key is the member id, and the matching method is internal matching.

3.10 Obtain RFM factor score by rf

```
# rf get rfm factor score
clf = RandomForestClassifier(n_estimators=100)
clf = clf.fit(rfm_merge[['r','f','m']],rfm_merge['member_level'])
weights = clf.feature_importances_
print('feature importance:',weights)
```

In the above process, the rf model object is first established, then the rfm three columns are used as features, the membership level is used as the target input model for training, and finally the weight message is obtained by the model's feature_importances_, and the result is as follows:

```
feature importance: [0.40466904 0.00607049 0.58926047]
```

From the above results, it can be seen that among the three dimensions, the user's level first focuses on the member's value contribution (the actual order contribution), followed by the recent degree, and finally the frequency.
This logic is consistent with the company's overall membership level. For example, the e-commerce membership level is upgraded based on the historical cumulative consumption amount. If there is no purchase in the last year, the membership level will be lowered.

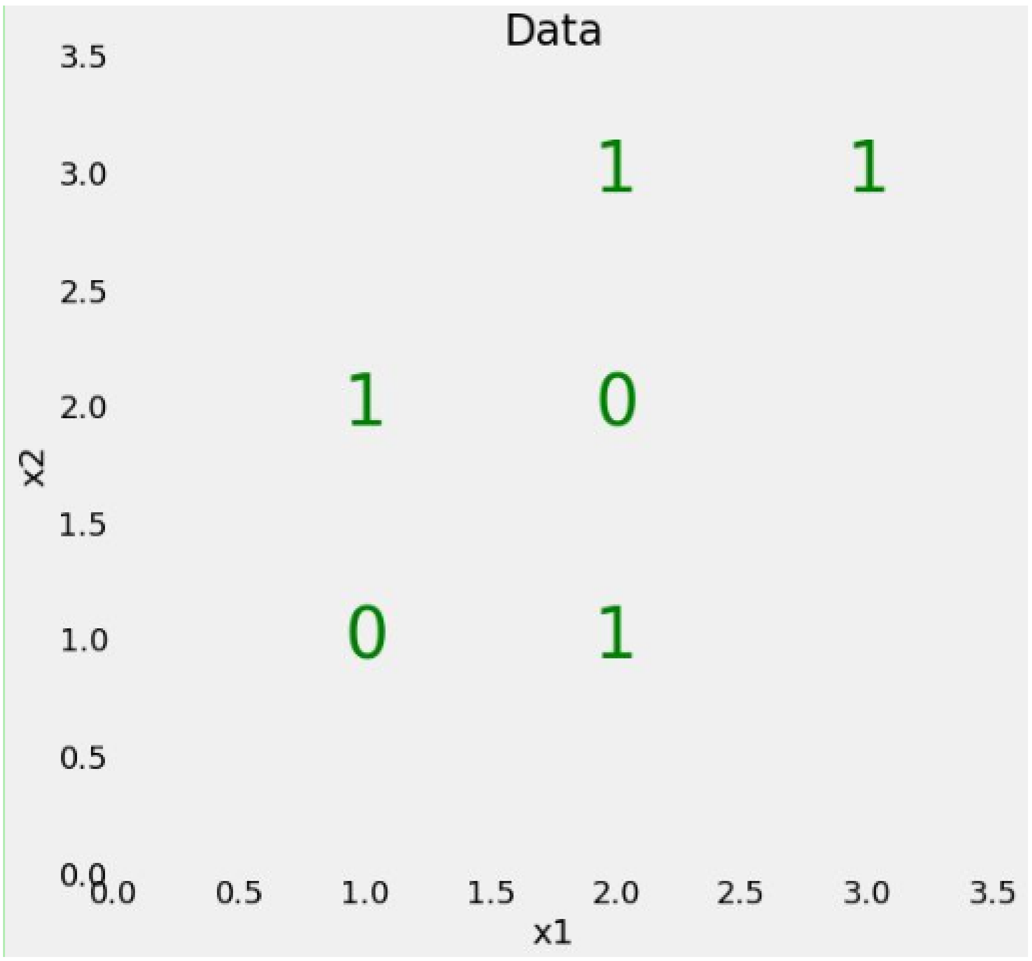
3.11 Random Forest

we'll try to get an understanding of how this model works. Because a random forest in made of many decision trees, we'll start by understanding how a single decision tree makes classifications on a simple problem.

3.12 Understanding a decision tree

A decision tree is the building block of a random forest and is an intuitive model. We can think of a decision tree as a series of yes/no questions asked about our data eventually leading to a predicted class (or continuous value in the case of regression). This is an interpretable model because it makes classifications much like we do: we ask a sequence of queries about the available data we have until we arrive at a decision (in an ideal world).

Usually an example is more to be proved
decision tree in a easy problem:



Our data has only two features (predictors), x1 and x2, with a total of 6 data points (samples), divided into 2 different labels.

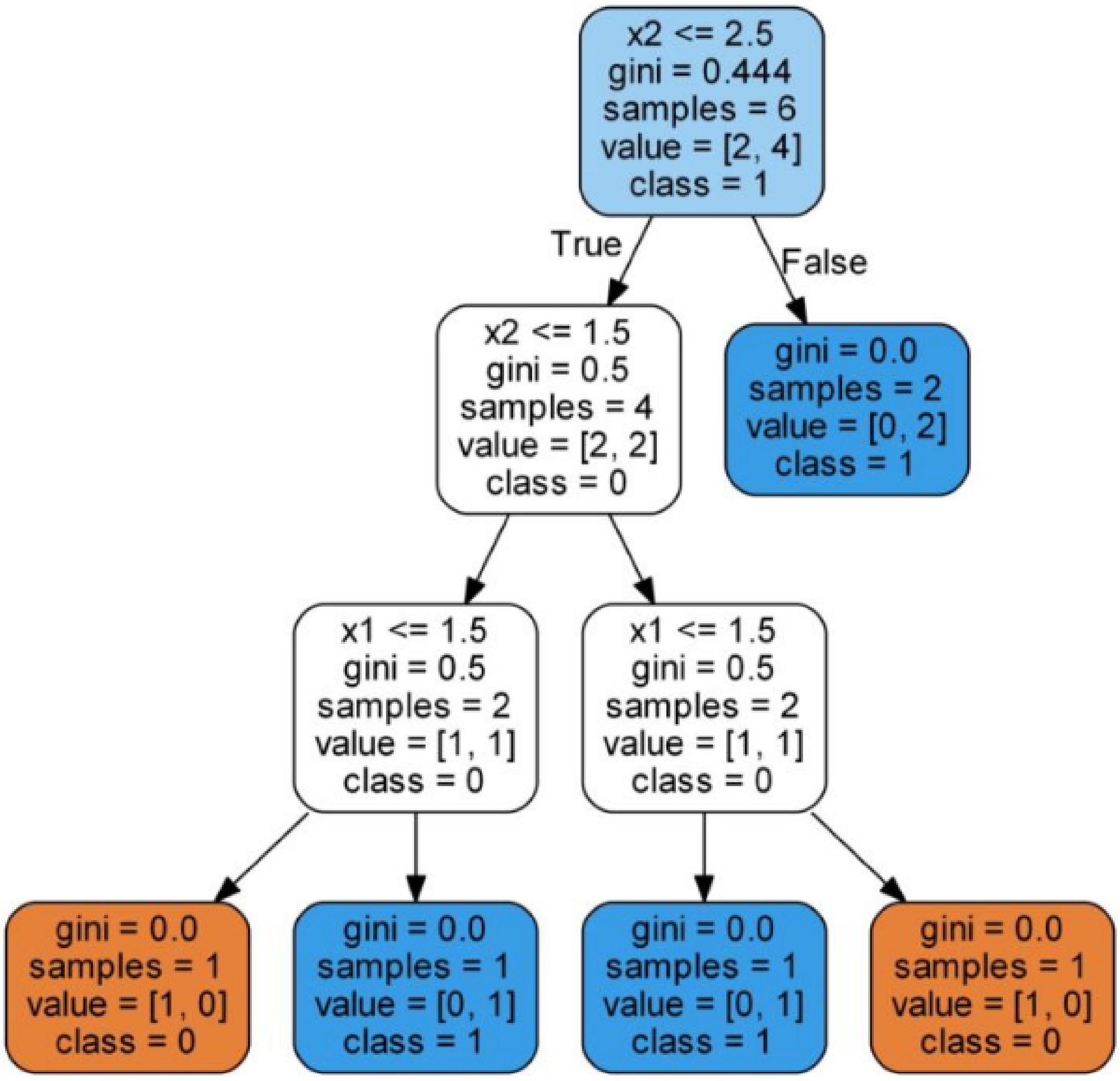
Here we quickly build and train a single decision tree on the data using Scikit-Learn. The tree will learn how to separate the points, building a flowchart of questions based on the feature values and the labels. At each stage, the decision tree makes splits by maximizing the reduction in Gini impurity.

We'll use the default hyperparameters for the decision tree which means it can grow as deep as necessary in order to completely separate the classes. This will lead to overfitting because the model memorizes the training data, and in practice, we usually want to limit the depth of the tree so it can generalize to testing data.

```
from sklearn.tree import DecisionTreeClassifier

# Make a decision tree and train
tree = DecisionTreeClassifier(random_state=RSEED)
tree.fit(X, y)
```

Visualize Decision Tree



A decision tree is an intuitive model: it makes decisions much as we might when faced with a problem by constructing a flowchart of questions. For each of the nodes (except the leaf nodes), the five rows represent:

> - Question asked about the data based on a feature: this determines the way we traverse down the tree for a new datapoint.

> - gini: the Gini Impurity of the node. The average (weighted by samples) gini impurity decreases with each level of the tree.
> - samples: number of training observations in the node
> - value: [number of samples in the first class, number of samples in the second class]
> - class: the class predicted for all the points in the node if the tree ended at this depth (defaults to 0 for a tie).

The leaf nodes (the terminal nodes at each branch) do not have a question because they are where the tree makes a prediction.

3.13 GINI Impurity

The Gini Impurity of a node is the probability that a randomly chosen sample in a node would be incorrectly labeled if it was labeled by the distribution of samples in the node.
Gini impurity of a node n:

$$I_G(n) = 1 - \sum_{i=1}^{J}(p_i)^2$$

The Gini Impurity of a node n is 1 minus the sum over all the classes J (for a binary classification task this is 2) of the fraction of examples in each class p_i squared. That might be a little confusing in words, so let’s work out the Gini impurity of the root node.

$$I_{root} = 1 - ((\frac{2}{6})^2 + (\frac{4}{6})^2) = 1 - \frac{5}{9} = 0.444$$

At each node, the decision tree searches through the features for the value to split on that results in the greatest reduction in Gini Impurity.
It then repeats this splitting process in a greedy, recursive procedure until it reaches a maximum depth.

There is a question Overfitting.
Overfitting occurs when we have a very flexible model (the model has a high capacity) which essentially memorizes the training data by fitting it closely. The problem is that the model learns not only the actual relationships in the training data, but also any noise that is present. A flexible model is said to have high variance because the learned parameters (such as the structure of the decision tree) will vary considerably with the training data.

The reason the decision tree is prone to overfitting when we don’t limit the maximum depth is because it has unlimited flexibility, meaning that it can keep growing until it has exactly one leaf node for every single observation, perfectly classifying all of them. If you go back to the image of the decision tree and limit the maximum depth to 2 (making only a single split), the classifications are no longer 100% correct. We have reduced the variance of the decision tree but at the cost of increasing the bias.
As an alternative to limiting the depth of the tree, which reduces variance (good) and increases bias (bad), we can combine many decision trees into a single ensemble model known as the random forest.

The random forest is a model made up of many decision trees. Rather than just simply averaging the prediction of trees (which we could call a “forest”), this model uses two key concepts that gives it the name random:

> 1.Random sampling of training data points when building trees

> 2.Random subsets of features considered when splitting nodes

We can use these to try and figure out what predictor variables the random forest considers most important.
Feature importances can be used for feature engineering by building additional features from the most important. We can also use feature importances for feature selection by removing low importance features.

3.14 Feature importance

In this case, we use conception "Feature importance".
The feature importances in a random forest indicate the sum of the reduction in Gini Impurity over all the nodes that are split on that feature. We can use these to try and figure out what predictor variables the random forest considers most important

Here we are going to introduce the use of random forests to feature screening.
The idea of using a random forest to assess the importance of a feature is actually very simple. It is to see how much each feature contributes to each tree in the random forest, then take an average and finally contribute to the contribution between the features. size.
Usually you can use the Gini index.

We use variable VIM to represent variable importance measures, and GI represent for Gini index . Suppose there are m features X1, X2, X3, ... Now we need to calculate Gini index score of each feature Xj.

$$VIM_j^{Gini}$$

That is, the j-th feature is the average amount of change in node splitting in the RF decision tree.

The formula for calculating the Gini index is:

$$I_G(n) = 1 - \sum_{i=1}^{J} (p_i)^2$$

The importance of the feature Xj at the node n, that is, the Gini index change before and after the node n branch is:

$$VIM_{jn}^{Gini} = I_G(n) - I_G(l) - I_G(r)$$

Among them, $$ I_G(l) and I_G(r) $$ respectively represent the Gini index of the two new nodes after branching.

If the node in which feature Xj appears in decision tree i is in set M, then the importance of Xj in the i-th tree is:

$$VIM_{ij}^{Gini} = \sum VIM_{jm}^{Gini}$$

Suppose RF has n trees ,then:

$$VIM_j^{Gini} = \sum_{i=1}^{n} VIM_{ij}^{Gini}$$

Fortunately, sklearn has packaged everything for us, we just need to call the function.

```
# match member level and 'rfm' scores
rfm_merge = pd.merge(rfm_gb,sheet_datas[-1],on='member_ID',how='inner')


clf = RandomForestClassifier()
clf = clf.fit(rfm_merge[['r','f','m']],rfm_merge['member_level'])

weights = clf.feature_importances_
print('feature importance:',weights)
```

First establish the rf model object, then use the rfm three columns as the feature, train the membership level as the target input model, and finally obtain the weight information through the feature_importances_ of the model.

output:

```
feature importance: [0.39042291 0.00670539 0.6028717 ]
```

3.15 Calculate RFM score

There are two ways to calculate.

Methode 1:Calculate by weights

```

# Calculate RFM total score
# Method 1: Weighted score
rfm_gb = rfm_gb.apply(np.int32) # cate转数值
rfm_gb['rfm_score'] = rfm_gb['r_score'] * weights[0] + rfm_gb['f_score'] *
weights[1] + rfm_gb[
'm_score'] * weights[2]
rfm_gb.head()
```

```
out:
    year    member_ID    r   f   m     r_score  f_score  m_score   rfm_score
0   2015    267          197 2   105   2          1        2        1.993930
1   2015    282          251 1   29    2          1        1        1.404669
2   2015    283          340 1   5398  1          1        3        2.178521
3   2015    343          300 1   118   1          1        2        1.589260
4   2015    525          37  3   213   3          2        2        2.404669
```

Method 2:
Measure all summaries from one dimension, especially for value sorting.

RFM combination

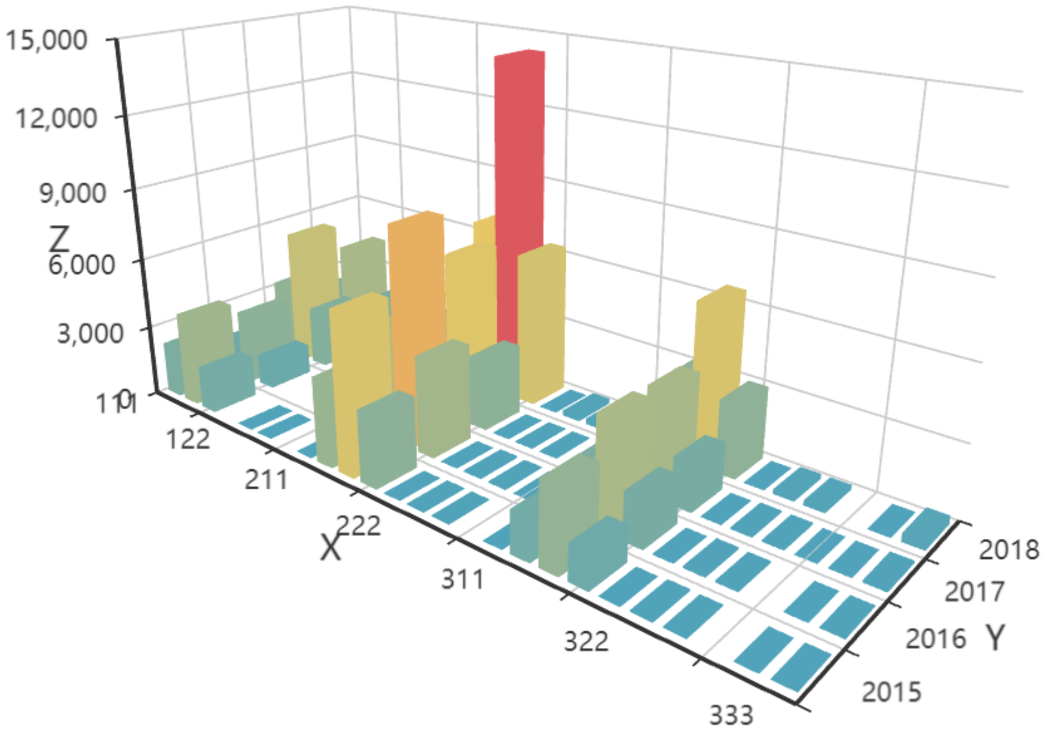
```
# Method 2: RFM combination
rfm_gb['r_score'] = rfm_gb['r_score'].astype(np.str)
rfm_gb['f_score'] = rfm_gb['f_score'].astype(np.str)
rfm_gb['m_score'] = rfm_gb['m_score'].astype(np.str)
rfm_gb['rfm_group'] = rfm_gb['r_score'].str.cat(rfm_gb['f_score']).str.cat(
rfm_gb['m_score'])
rfm_gb.head()
```

```
year member_ID  r    f    m    r_score f_score m_score rfm_score     rfm_group
0    2015        267  197 2    105     2       1       2    1.993930       212
1    2015        282  251 1    29      2       1       1    1.404669       211
2    2015        283  340 1    5398    1       1       3    2.178521       113
3    2015        343  300 1    118     1       1       2    1.589260       112
4    2015        525  37   3    213     3       2       2    2.404669       322
```

This method is a traditional way of grouping members. The goal is to combine 3 columns as a string into a new group.

Graphic display

in the document "render.html"



As can be seen from the figure, the X-axis is the RFM grouping, the Y-axis is the year, and the Z-axis is the number of users.

Analysis data

1. Graphic-based interactive analysis
   Key population distribution: Through 3D column chart for simple analysis, the users of the 212 group are relatively concentrated and largest in the whole group. It can be found from the figure that the number of user groups varies little from 2016 to 2017, but It has nearly doubled in 2018, so this group of people will be the focus of analysis.
   The key groups are: In addition to the 212 population, the figure also shows that 312, 213, 211, and 112 people occupy a large number in each year, although their respective scale is small.

2. Analysis based on RFM grouping results

RFM grouping	user share
212	24.79%
211	12.80%
312	12.55%
112	11.34%
213	11.02%
311	6.23%
111	6.11%
313	5.61%
113	5.07%

Therefore, we focus our analysis on these 9 groups of people.

3.26 RFM user characteristics analysis

After the above analysis, the key customer groups to be analyzed were obtained. It can be divided into two categories according to the user's magnitude: the first category is a group with a user group accounting for more than 10%; the second category is a group with a single digit. These two types are different in magnitude, so they need to be distributed.

In addition to the targeted strategy scenario, in addition to the third category of people, although the magnitude of the user is small, the value of the individual is very high.

The first group of people: the group with more than 10% of the population. These people have a large base. They must adopt a batch operation and operation method to implement the operation strategy. Generally, they need to be implemented through systems or products, and cannot rely mainly on labor.

The second group of people, accounting for 1%-10% of the group, this part of the population is in the middle, can be accessed in the landing product or manual

The third group, which has a very small but very important group, such as 333, 323, 233,etc, we can provide special VIP services.

4. Case2: Marketing Response Prediction Based on Nested Pipeline and FeatureUnion Composite Data Workflows

Case background:

The case introduces the practical application of the member marketing forecast. When the member department is doing affiliate marketing, it hopes to use the data forecast to respond to the specific list and response probability of the active members in the next marketing campaign, so as to specify the targeted marketing strategy.

Main application technology:

> - Data preprocessing
> - Pipeline technology: Nesting Pipeline and FeatureUnion to form a multi-processing link based on resurrection pipeline
> - Data Modeling: Cross-checking cross_val_score with StratifiedKFold, integrated classification algorithm RandomForestClassifier application
> - The main libraries used include time, numpy, pandas, sklearn

The main technical application of this case is to combine multiple feature processing projects through pipeline method, and then form a feature engineering pipeline. Combine the feature engineering pipeline with RandomForestClassifier to form a composite Pipeline. The advantage is that it is easy to do the various processes. Unified management, when deploying and applying, reduce single management objects and be more concise and convenient.

4.1 Read data

The case data is located in "order.xlsx ", including two sheets: sheet1 is the training set, and sheet2 is the prediction set.

> - Number of characteristic variables: 113
> - Is there a NA value: Yes
> - Is there an outlier: No
> - Total_pageviews: total page views
> - Edu: education level
> - Edu_ages
> - User_level: user level, subtype variable, value range [1,10]
> - Value_·level, act_level: user value, user activity
> - Blue_money, red_money: coupon order amount
> - Label_1 to label_100: the value of the user on different labels, each label represents a label of activity, the value field [0,1] target variable response: 1 means the user has a response, 0 means not responded

4.2 Building Pipeline Applications with Pipeline and FeatureUnion

```
# Create a model object used in the pipeline

model_etc = ExtraTreesClassifier()  # ExtraTree，用于EFE的模型对象
model_rfe = RFE(model_etc)  # 使用RFE方法提取重要特征
model_lda = LinearDiscriminantAnalysis()  # LDA模型对象 降维
model_rf = RandomForestClassifier()  # 分类对象


# Build a pipeline with nesting
```

```
pipelines = Pipeline([
    ('feature_union', FeatureUnion(  # Combined feature pipeline
        transformer_list=[
            ('model_rfe', model_rfe),  # Extracting features through RFE
            ('model_lda', model_lda),  # Extracting features through LDA
linearDiscriminantAnalysis()
        ],
        transformer_weights={  # Establish weights for different feature models
            'model_rfe': 1,  # RFE weight
            'model_lda': 0.8,  # LDA weight
        },
    )),
    ('model_rf', model_rf),
])
```

In the outer Pipeline, the work step is a combination of multiple tuples,

the first tuple is ('featrue_union'.FeatureUnion), the second tuple is ('model_adaboost', model_adaboost), English "Adaptive Boosting" The abbreviation for (Adaptive Enhancement) is a machine learning method proposed by Yoav Freund and Robert Schapire.
The adaptation of the AdaBoost method is that the sample of the previous classifier error is used to train the next classifier. The former in the tuple is the instance name and the latter is the instance object.

In the inner FeatureUnion, the weight of the two instance objects is set for the transformer_weights, and the order is the same as the order of the tuple.

4.3 The role of Pipeline

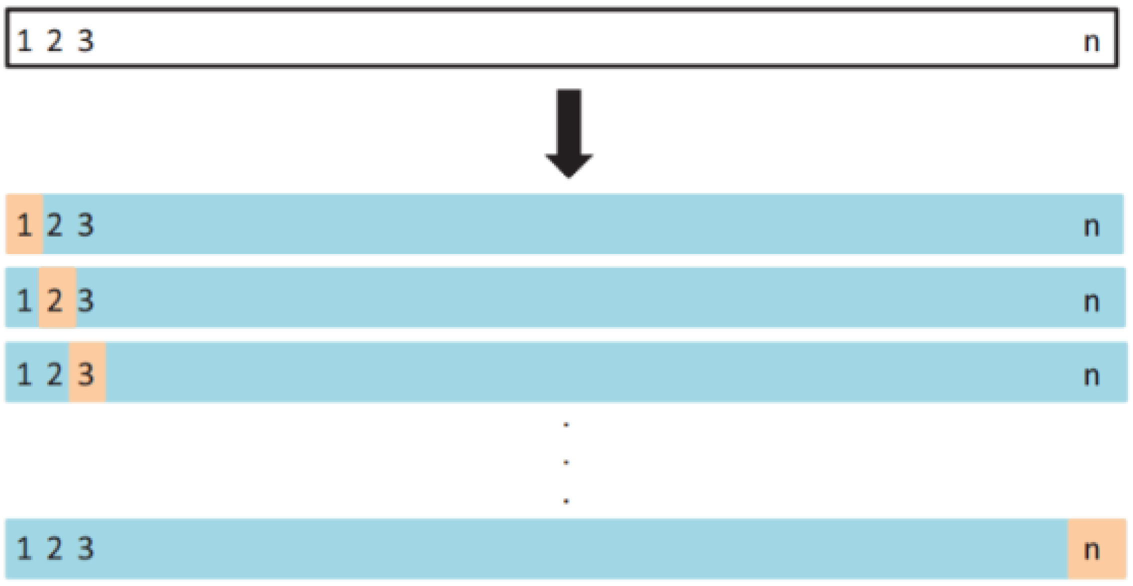
Pipeline can connect many algorithm models in series, which can be used to cascade multiple estimators into an estimator, such as feature extraction, normalization, and classification to form a typical machine learning problem workflow. All estimators outside the last Pipeline must be transformers. The last estimator can be any type (transformer, classifier, regresser). If the last estimator is a classifier, the entire pipeline can be used as a classifier. If the last estimator is a cluster, the entire pipeline can be used as a cluster.

In fact, calling the pipeline's fit method uses the first n-1 transformers to process the features and then passes them to the final estimator for training. The pipeline inherits all the methods of the last estimator.
This brings two direct benefits:

1. Directly call the fit and predict methods to train and predict all algorithm models in the pipeline.
2. The parameters can be selected in conjunction with grid search.

4.4 Cross-Validation

The LOOCV (Leave-one-out cross-validation)method also includes the step of dividing the data set into a training set and a test set. But the difference is that we only use one data as the test set, the other data is used as the training set, and this step is repeated N times (N is the number of data in the data set)



As shown in the figure above, assuming that we now have a data set of n data, the LOOCV method is to take one data each time as the only element of the test set, while the other n-1 data are used as training sets for training the model and Tuning. The result is that we finally train n models and get an MSE each time. Calculating the final test MSE is to average the n MSEs.

The **Mean Squared Error** (MSE) or Mean Squared Deviation (MSD) of an estimator measures the average of error squares i.e. the average squared difference between the estimated values and true value:

$$CV_n = \frac{1}{n}\sum_{i=1}^{n} MSE_i$$

However, the shortcomings of LOOCV are also obvious, that is, the amount of calculation is too large.

4.5 K-fold Cross Validation

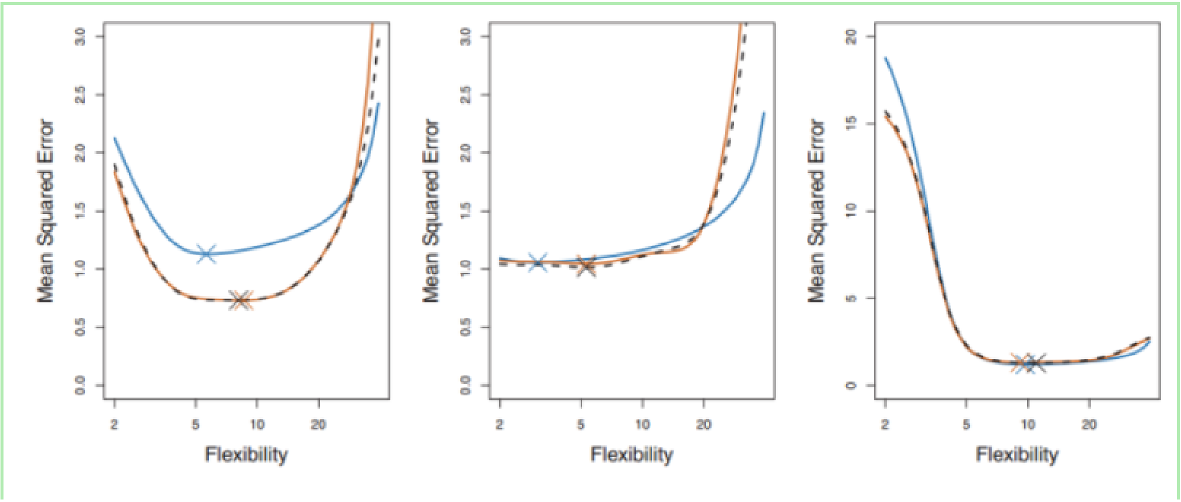
The difference with LOOCV is that each test set will no longer contain only one data, but multiple, and the specific number will be determined according to the selection of K. For example, if K=5, then the steps we take with 5-fold cross-validation are:

1. Divide all data sets into 5 parts
2. Do not repeat one of each time to do the test set, use the other four to do the training set training model, and then calculate the MSE_i of the model on the test set.
3. Average the 5 times MSE_i to get the final MSE

$$CV_k = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

It is easy to understand that LOOCV is a special K-fold Cross Validation (K=N).

let's take a look this picture:



Each picture is represented by the true test MSE in blue, while the black dotted line and the orange line are decibels representing the LOOCV method and the test MSE obtained by the 10-fold CV method. We can see that the estimates of test MSE for LOOCV and 10-fold CV are very similar, but the computational cost of 10-fold CV is much smaller and less time-consuming than LOOCV.

So,let's set the pipeline cross-validation:

```
# pipeline Cross-Validation
cv = StratifiedKFold(3)  # set Cross-Validation
score_list = list()  # Create an empty list for storing cross-validation scores
time_list = list()  # Create an empty list for storing time
n_estimators = [10, 50, 100]  # Set the n_estimators value field of rf in the
pipeline
for parameter in n_estimators:
    t1 = time.time()  # Record the time at which the cross-check begins
    print('set parameters: %s' % parameter)  # Print the parameters used by the
current model
    pipelines.set_params(model_rf__n_estimators=parameter)  # Set classification
model parameters through pipes
    score_tmp = cross_val_score(pipelines, X_train, y_train,
scoring='accuracy',cv=cv,n_jobs=-1)  # Calculate score using cross-check
    time_list.append(time.time() - t1)  #  add time to the list
    score_list.append(score_tmp)  #  add score to  the lisit
```

> - Cv sets the sample split method for cross-checking
> - Score_list and time_list are used to store cross-check scores and spend time
> - N_estimators_range defines the range of the cross-check loop, here is the value of the random forest's n_estimators.
>   In the following, each n_estimator value is read by parameter, and then the random forest is set to the following parameters based on the number of base models at different scales:
>   Record current time by t1
> - Project_pipeline.set_params sets the specific value of each paramter to verify the model effect
> - Cross_val_score calls the method defined above for cross-testing
> - Computation completed calculation time difference record

output:

```
   n_estimators    time    score1    score2    score3  score_mean  score_std
0            10  70.203  0.879794  0.875603  0.878804    0.877699   0.002964
1            50  68.001  0.888579  0.887496  0.887913    0.888038   0.000766
2           100  69.018  0.887508  0.891889  0.887484    0.889699   0.003098
```

The higher score of score_mean, the more accurate of model prediction is, the smaller the score_std means indicating that the more stable the results , the smaller of time indicates that the time is shorter.
Comprehensive consideration, the score_mean difference is not large, choose score_std is small, choose n_estimators=50, less time, and the effect is slightly better.

4.6 Model effect test

```
# Model effect test
pre_test = pipelines.predict(X_test)
scores = [i(y_test,pre_test) for i in [f1_score,accuracy_score,precision_score]]
print('socres result: f1:{0},accuracy:{1},precision:
{2}'.format(scores[0],scores[1],scores[2]))
```

```
socres result: f1:0.7611671469740635,accuracy:0.8895,precision:0.786378861183476
```

From the above results, the result of the test set's accuracy is 0.8895, and the mean of the three cross-checks in the model is 0.888.38, which is basically consistent with the results of the cross-test model.

Forecast new data set:

```
# predict probability
pre_labels = pd.DataFrame(pipelines.predict(new_data_fillna), columns=
['labels'])  #forecast label
pre_pro = pd.DataFrame(pipelines.predict_proba(new_data_fillna), columns=
['pro1', 'pro2'])  #  forecast probability
predict_pd = pd.concat((pre_labels, pre_pro), axis=1)  # merge data
print(predict_pd.head())
```

output:

```
    labels  pro1  pro2
0        0  0.86  0.14
1        0  0.96  0.04
2        0  0.98  0.02
3        0  1.00  0.00
4        0  0.92  0.08
```

The results of this case are given to the business department, and the response marketing activities are taken. For this marketing activity, a KPI with a marketing response rate of at least 80% has been developed.

conclusion

The number of feature selections is one of the key factors affecting the accuracy of the classification results. The more the number of features, the higher the classification accuracy, but this is not absolute.

In addition to the number of features, it depends on the feature quality, sample size, and classification. The reliability of the algorithm and other factors, so the two are not completely linear 2.

The purpose of the Pipeline application is not to improve the performance of the code itself, but to the unified management of complex objects. For example, in this case, if there is no pipeline management, you need to apply the same application for each model, such as fit, transform, and prediction. This kind of application itself can be managed uniformly through the pipeline.

We first looked at an individual decision tree, the building block of a random forest, and then saw how we can overcome the high variance of a single decision tree by combining hundreds of them in an ensemble model known as a random forest.

The random forest uses the concepts of random sampling of observations, random sampling of features, and averaging predictions.
The key concepts to understand from this are:

> - Decision tree: an intuitive model that makes decisions based on a sequence of questions asked about feature values. Has low bias and high variance leading to overfitting the training data.
> - Gini Impurity: a measure that the decision tree tries to minimize when splitting each node. Represents the probability that a randomly selected sample from a node will be incorrectly classified according to the distribution of samples in the node.
> - Bootstrapping: sampling random sets of observations with replacement.
>   Random subsets of features: selecting a random set of the features when considering splits for each node in a decision tree.
> - Random Forest: ensemble model made of many decision trees using bootstrapping, random subsets of features, and average voting to make predictions. This is an example of a bagging ensemble.
> - Bias-variance tradeoff: a core issue in machine learning describing the balance between a model with high flexibility (high variance) that learns the training data very well at the cost of not being able to generalize to new data , and an inflexible model (high bias) that cannot learn the training data. A random forest reduces the variance of a single decision tree leading to better predictions on new data.

Bibliographe

- [1] Raschka S. Python Machine Learning[M]. Packt Publishing, 2015.
- http://faculty.marshall.usc.edu/gareth-james/
- https://zhuanlan.zhihu.com/p/24825503
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76