Questions:

1. We have limited data on these merchants and their transactions, but we are still interested in understanding their payments activity to try to infer the types of merchants using our application. Using only the given data, how would you identify different kinds of businesses in the sample?

2. Sometimes a merchant may stop processing with our platform, which we call churn. We are interested in identifying and predicting churn. Please a) come up with a concrete definition for churn b) identify merchants that have already churned in the dataset, and c) build a model to predict which active merchants are most likely to churn in the near future.

Preparation

Before starting the analysis, we should look at the dataset, and try to find some interesting patterns in different features so that we can have a better understanding of the entire dataset.

Let's look at the dataset.

```
df.shape
```

```
(1513719, 4)
```

```
df.head()
```

	Unnamed: 0	merchant	time	amount_usd_in_cents
0	1	faa029c6b0	2034-06-17 23:34:14	6349
1	2	ed7a7d91aa	2034-12-27 00:40:38	3854
2	3	5608f200cf	2034-04-30 01:29:42	789
3	4	15b1a0d61e	2034-09-16 01:06:23	4452
4	5	4770051790	2034-07-22 16:21:42	20203

```
df.isnull().sum()
```

```
Unnamed: 0            0
merchant              0
time                  0
amount_usd_in_cents   0
dtype: int64
```

From above, we know that the values in our dataset are cleaned, so we do not need to handle Nan or error values.

Since the time column includes year, month, day, hour, minute and second, we could try to split them out and see whether we can find some patterns.

We can discover that there are some cycles in the daily distribution, day of week distribution and hourly distribution.

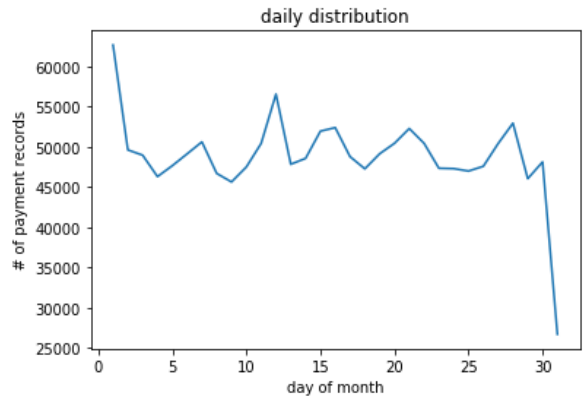

Figure 1

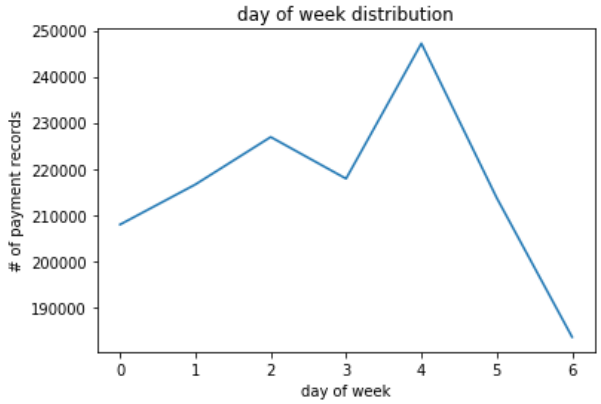

Figure 2

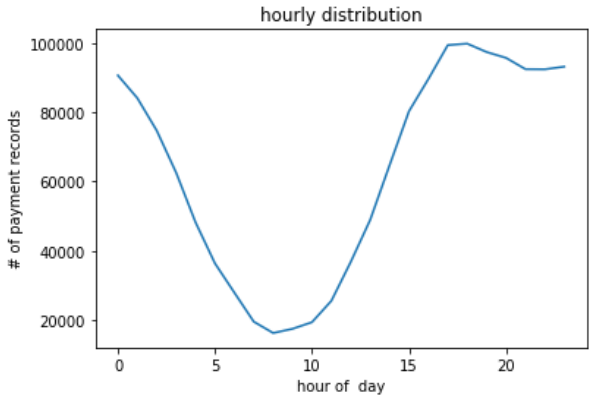

Figure 3

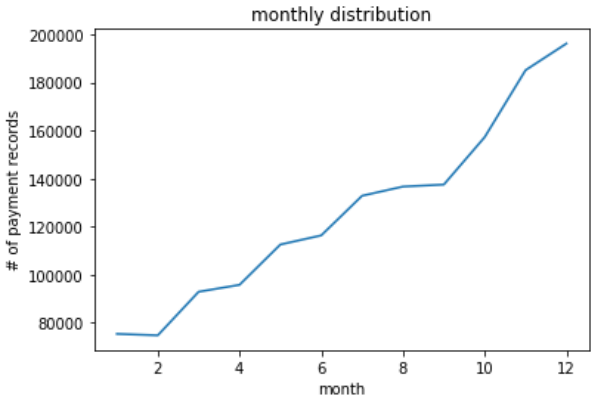

Figure 4

From figure 1 and figure 2, we can see a weekly pattern in the payment records, which is likely to be a characteristic of merchants (i.e. some merchants love to make a payment on Mondays).

From figure3, we can see an hourly pattern in the payment records, which is also likely to be another characteristic of merchants (i.e. some merchants mostly pay at night)

Figure 4 give us a linear trend and some steps as well, which is also could be a characteristic of merchants (i.e. some merchants make the payment at the end of the year)

We can also plot the total amounts verse different time ranges (weeks, days, hours), but the results are like the figures above.

We will try on these features in the following questions.

Q1. Merchant Classification

We assume that every merchant's activity is independent from each other.

We want to use some clustering method (i.e., K-means) to classify the merchants, because we do not know the number of different kinds of merchants, and we do not have labels for merchants to train as well. To perform a cluster learning, we need to create a new dataset.

Dataset transformation and data preprocessing

We should first aggregate the records to generate another data frame in terms of individual merchant.

Considering the patterns we have discovered before, the features we will use are:

count (the number of records for a merchant),

total amounts (the total payment amount for a merchant),

average amount (the average amount for a merchant)

7 days of week (Monday to Sunday),

4 seasons (spring for month 1-3, summer for month 4-6, fall for month 7-9, winter for month 10-12),

3 time-integrals among a day (0-4, 5-13, 14-23)

```
df_group_by_merchant.describe()
```

	count	total_amount	avg_amount	Friday	Monday
count	14351.000000	1.435100e+04	1.435100e+04	14351.000000	14351.000000
mean	105.478294	1.633296e+06	3.810458e+04	17.226047	14.487562
std	527.717966	6.431726e+06	1.415186e+05	87.676754	79.291155
min	1.000000	2.010000e+02	2.010000e+02	0.000000	0.000000
25%	3.000000	3.626300e+04	5.738182e+03	0.000000	0.000000
50%	11.000000	1.602620e+05	1.096500e+04	1.000000	1.000000
75%	45.000000	8.230995e+05	2.732122e+04	7.000000	5.000000
max	25512.000000	2.369072e+08	8.887465e+06	3750.000000	3569.000000

Figure 5

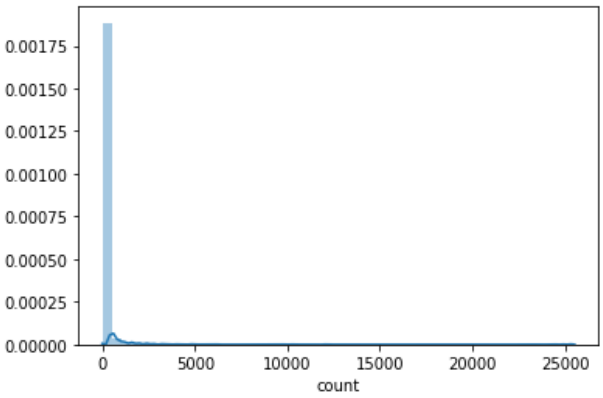

Figure 6

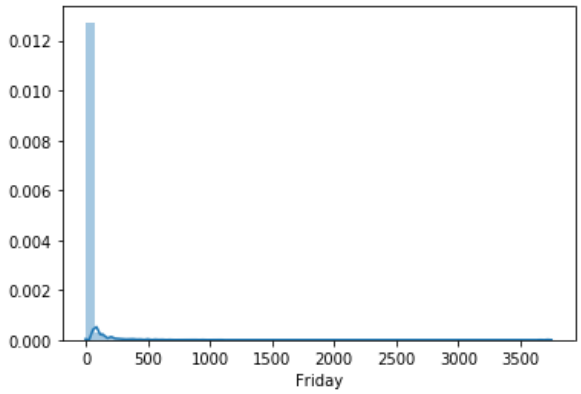

Figure 7

From figure 5, 6, 7, we find that the features are very skewed. We try to take logs for all the features in order to remove some skewness, but the results are not desirable.

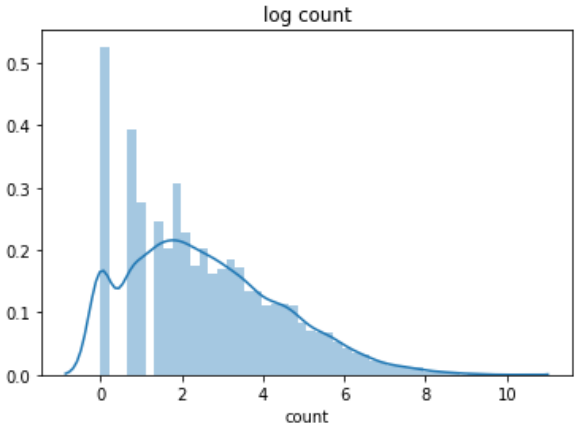

Figure 8

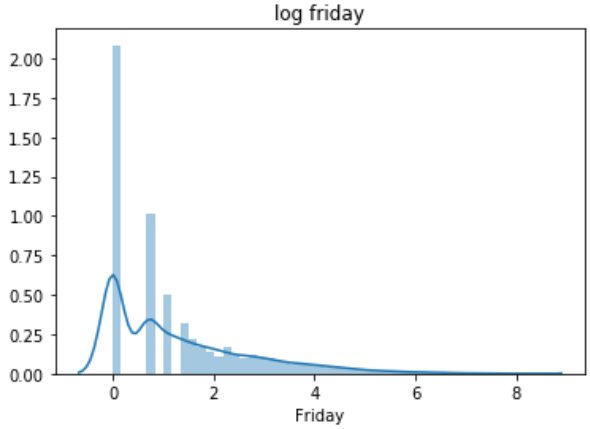

Figure 9

The distributions are still skewed, so we cannot perform K-means directly on the dataset, we need to make some changes.

First, we have noticed that the median value of count attribute is 11, which is a lot lower than the mean (105.48). The same problems also occur in total_amount and avg_amount. Thus, we should set a threshold to extract (new) merchants from the df_group_by_merchant dataset.

For merchants who have count value less or equal to 11, particularly, I define this group of merchants as **the fresh merchants**.

For merchants who have a count value more than 11, I define them as **the regular merchants**.

The reason I use the count feature to perform the split is that the number of records for individual merchants is so important for the analysis. We could easy get some biased result if we have not already collected enough data to describe customer behaviors. Thus, unless I get enough data, I do not tend to use it for further analysis.

Fresh merchant analysis

The fresh merchants are relatively new to the app. So, we just need to perform some basic analysis. It does not mean that they are not important, since they are more than half of the dataset, and they are of great profit potentiality in the future.

After several feature combinations, we choose to use the avg_amount as the only feature for classification.

To remove skewness, we take logarithm for avg_amount values.

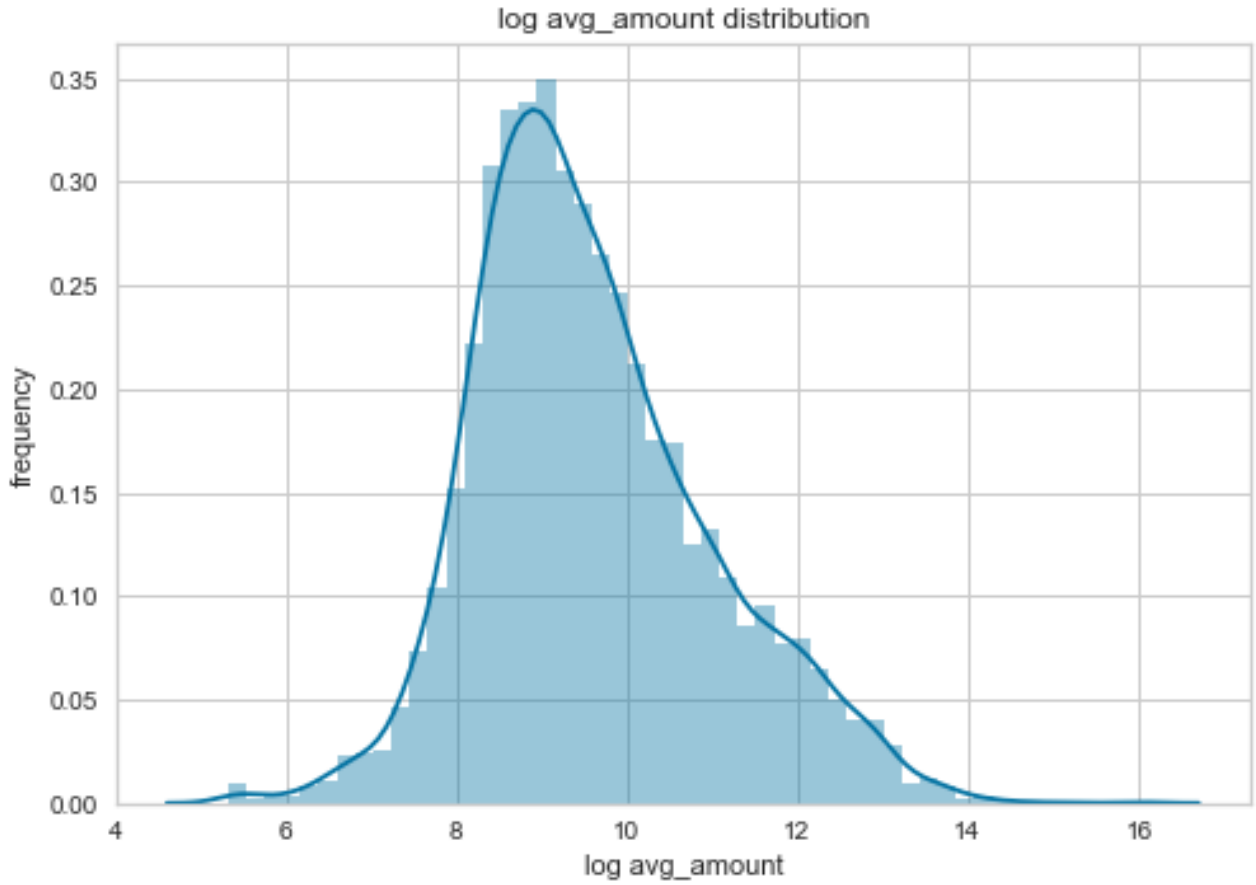

Figure 10

Then we can perform K-means with the help of the Elbow method

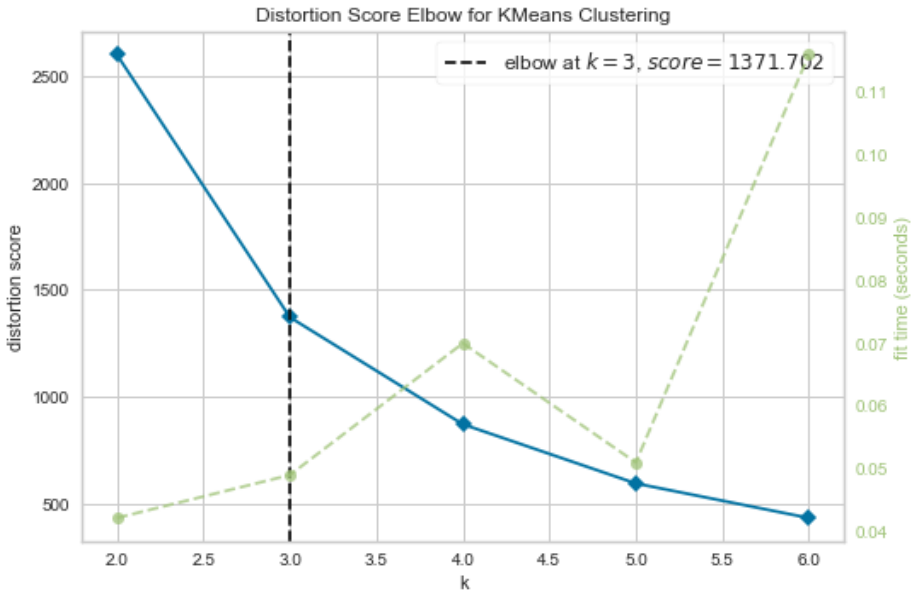

Figure 11

Figure 11 suggests the best k for K-means is 3. Finally, we get the clusters for fresh merchants

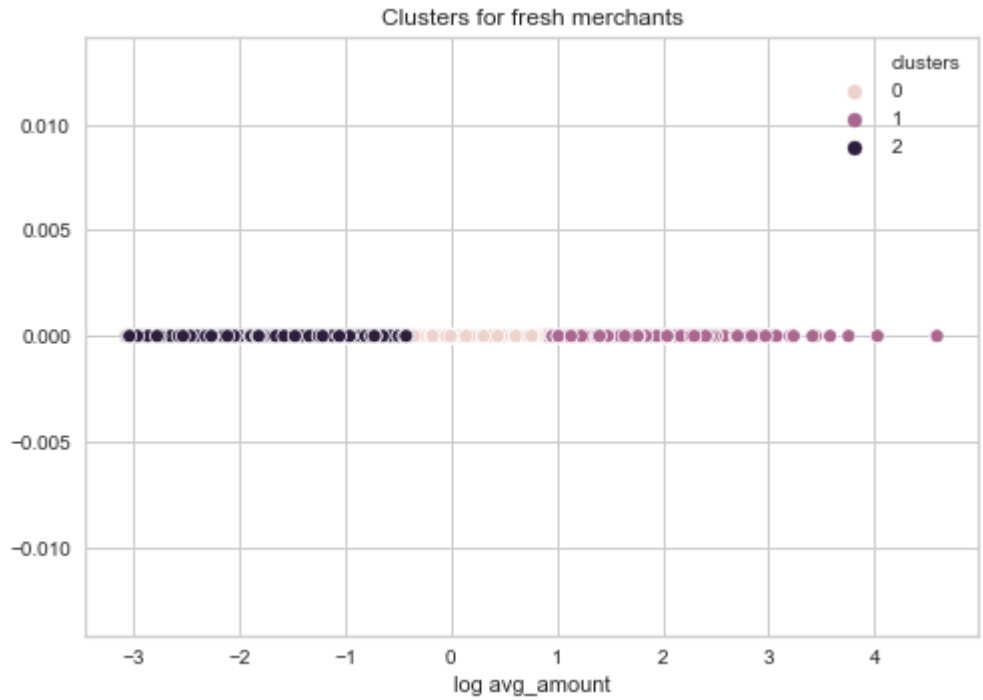

Figure 12

Regular merchant analysis

Now, we are going to analysis the regular merchants. Since we have more records on them, we can use all the features we have discovered before.

However, the correlation matrix suggests a lot of correlations between features.

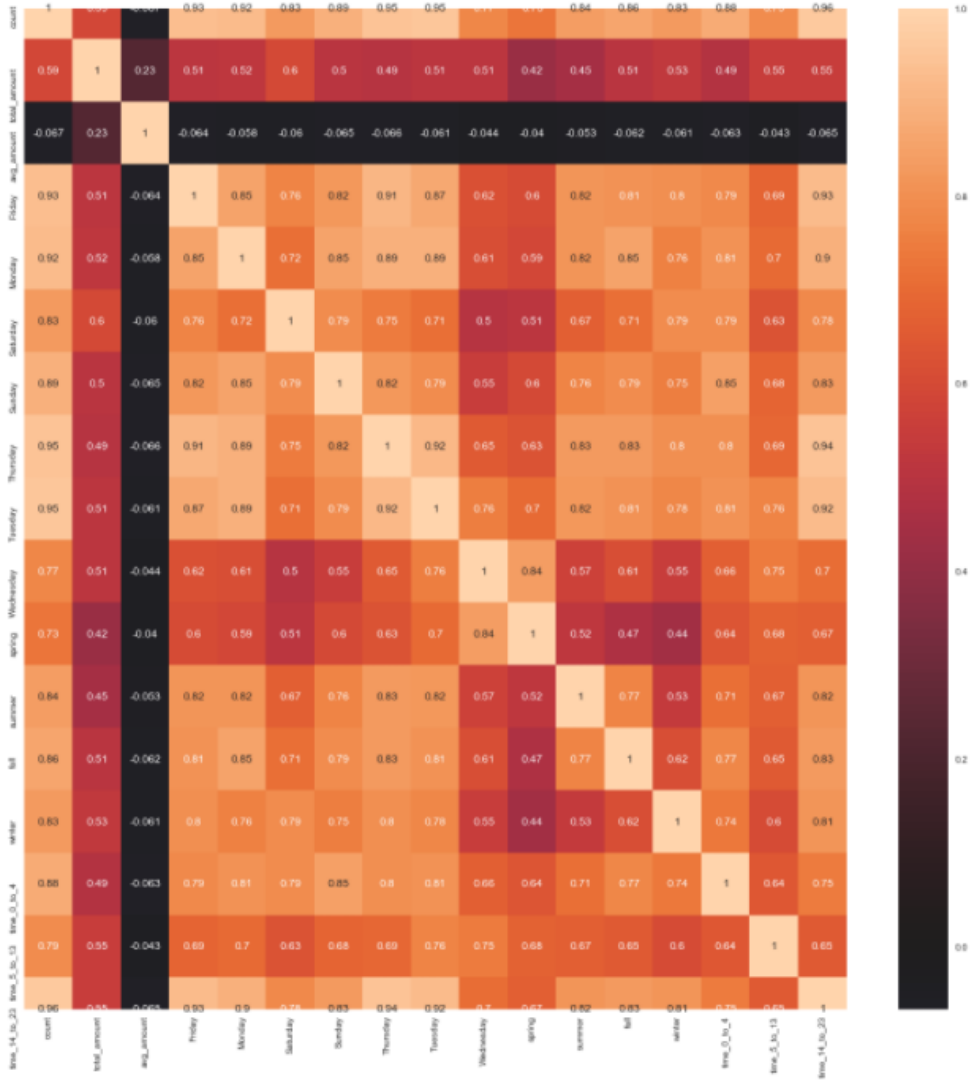

Figure 13

Since we have attribute count (# of records), and we want to discover merchant's payment behaviors along with time series, I decide to use "percentage of payment records" instead of the "absolute number of payment records" for time-related features. Thus, these features are divided by count

'Friday', 'Monday', 'Saturday', 'Sunday', 'Thursday', 'Tuesday', 'Wednesday', 'spring', 'summer', 'fall', 'winter',
'time_0_to_4', 'time_5_to_13', 'time_14_to_23'

Now there is no collinearity between features

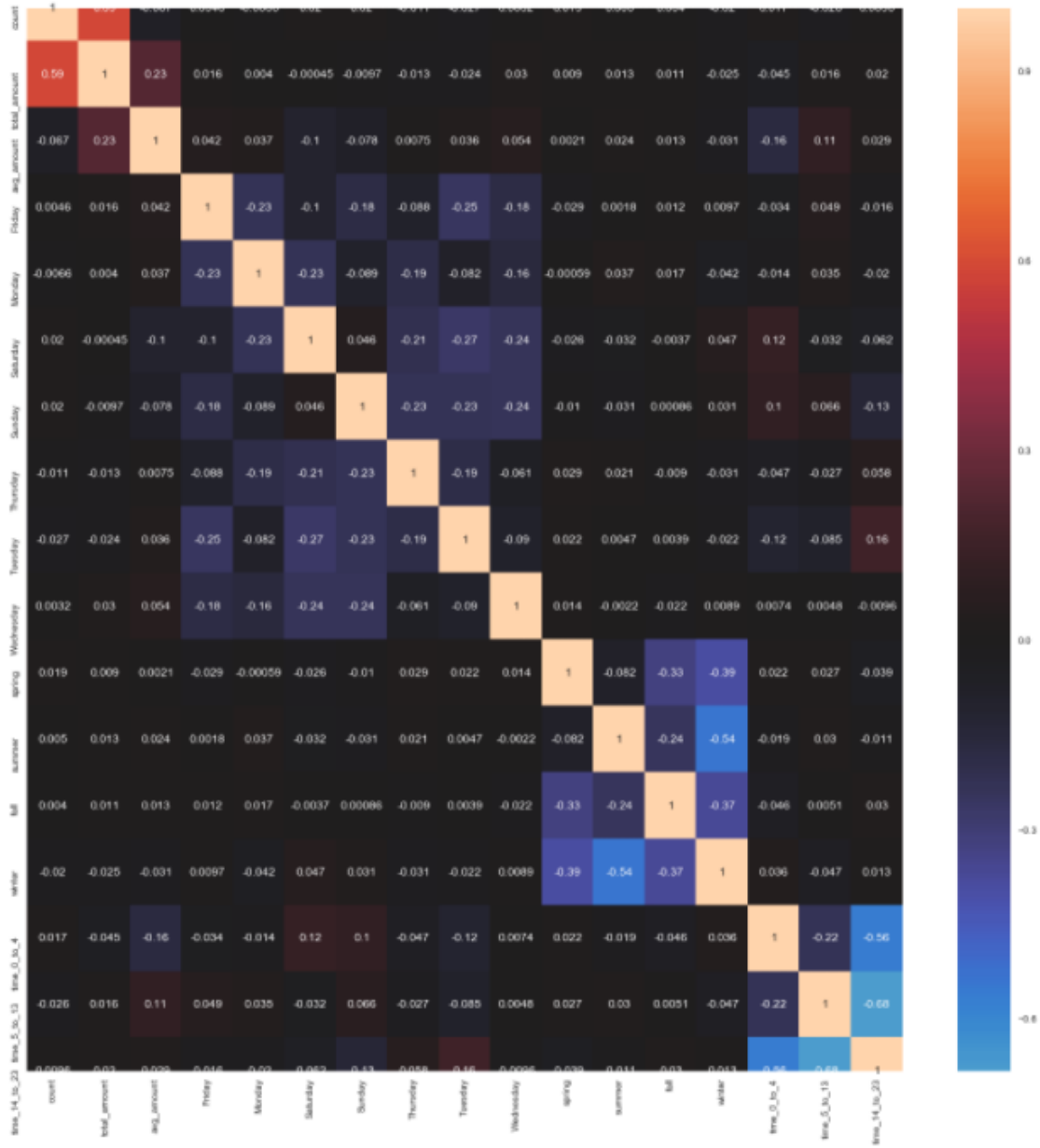

Figure 14

After performing logs to remove skewness, we need to standardize the features and perform PCA for dimension reduction. We select the top 10 principal components to cover more than 80% of the variance explanation. (described in code)

Now, we can select and train the model.

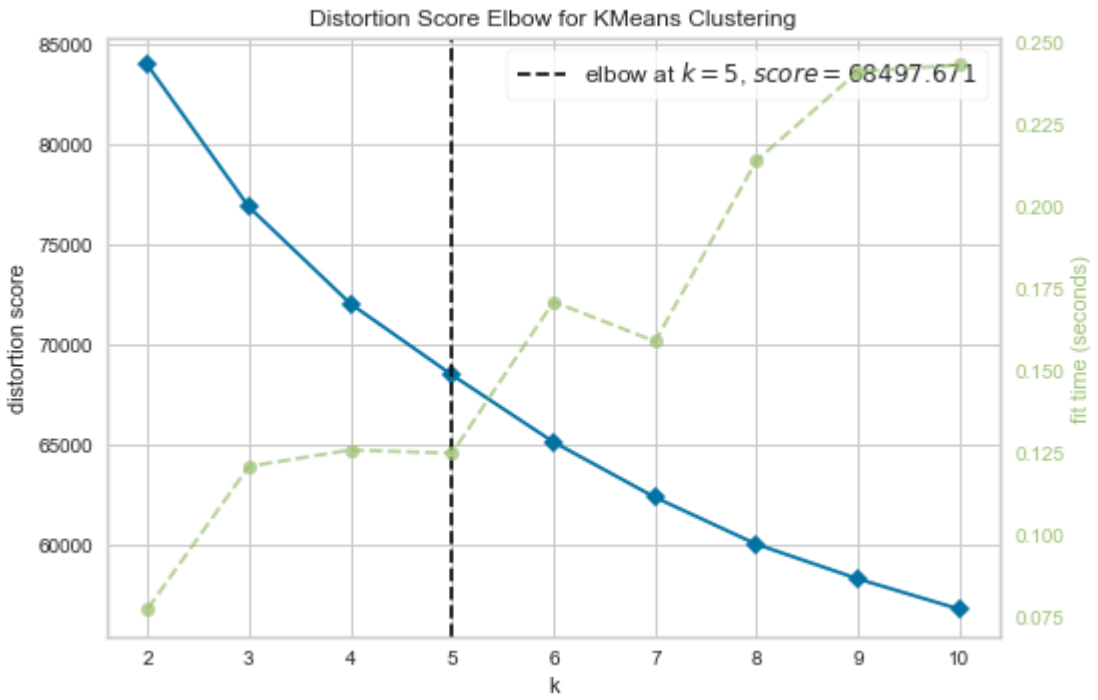

Figure 14

According to elbow method's suggestion, we choose k = 5. The 3D visualization for clusters looks like this

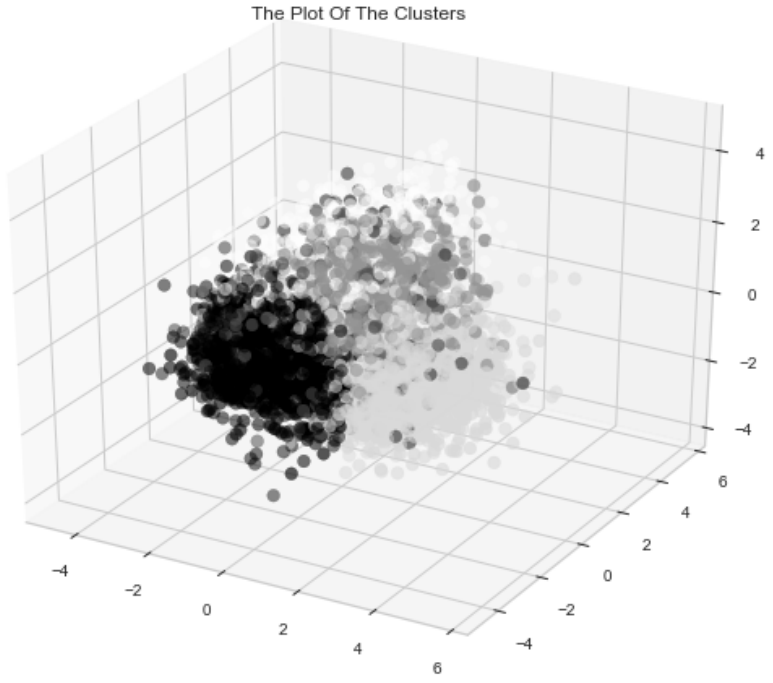

Figure 15

In general, we assign all the merchants into 8 different groups.

For fresh merchants, they are divided into 3 groups according to the average amount they have paid.

For regular merchants, they are divided into 5 groups, based on their monthly payment amounts, monthly payment behaviors, weekly payment behaviors and daily payment behaviors.

We can move further with these groups to find out their similarities and differences.

Q2. Churn

Definition

First, let's define what is churn.

Def: the payment range for a merchant (PR) is
PR = (the last payment record timestamp – the first payment record timestamp)

Def: the average period of making a payment for a merchant (APMPM)
APMPM = PR / count, count is the number of records for the merchant

Def: the churn period (CP) is the 90th percentile among all the merchant APMPMs.
CP > 90% of the APMPMs among all the merchants

Def: we call a merchant is churned when the time gap between current and his latest payment record timestamp is greater than the churn period.

In our dataset, the churn period is round to 43.

Identify churn in dataset

Then, we can get the churn merchants (the process in in the code)

The churn data

```
churn_merchant = df_group_by_merchant[df_group_by_merchant['churn'] == 1]
```

```
churn_merchant['merchant']
```

```
2        00057d4302
8        0020aefbd9
9        0026f256ac
11       0037a192b4
14       0042aadfdf
            ...
14342    ffc46fd720
14343    ffc5a319bb
14346    ffd3e45675
14347    ffe1f6b51a
14349    ffec05edb9
Name: merchant, Length: 6284, dtype: object
```

Figure 16

Finally, let's create the prediction model for churn.

Churn prediction

Since we have already defined the churn label, we can create a supervised model for prediction. Here, I will use gradient boosting.

Let's first convert some features into numbers.

```
# we want to convert the first_act and last_act into integers
for index,row in df_group_by_merchant.iterrows():
    temp_merchant = row['merchant']
    first_day = (row['first_act'] - first_date).total_seconds() / 86400
    df_group_by_merchant.loc[index,'first_act'] = int(first_day)

    last_day = (row['last_act'] - first_date).total_seconds() / 86400
    df_group_by_merchant.loc[index,'last_act'] = int(last_day)
```

```
df_group_by_merchant.dtypes
```

```
merchant              object
count                  int64
total_amount           int64
avg_amount           float64
first_act              int64
last_act               int64
total_days           float64
avg_days             float64
range_to_last_day    float64
churn                  int32
predicted_churn        int32
predicted_0_prob     float64
predicted_1_prob     float64
abs_prob_diff        float64
dtype: object
```

Figure 16

Because gradient boosting is not sensitive to outliers, we can directly look at the distribution of the label churn.

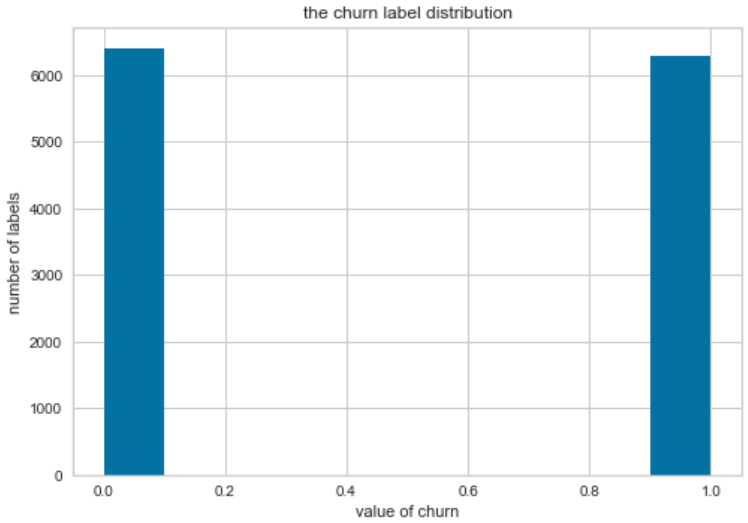

Figure 17

According to figure 17, the training data is balanced.

Moreover, we need to check the correlation between features as well.

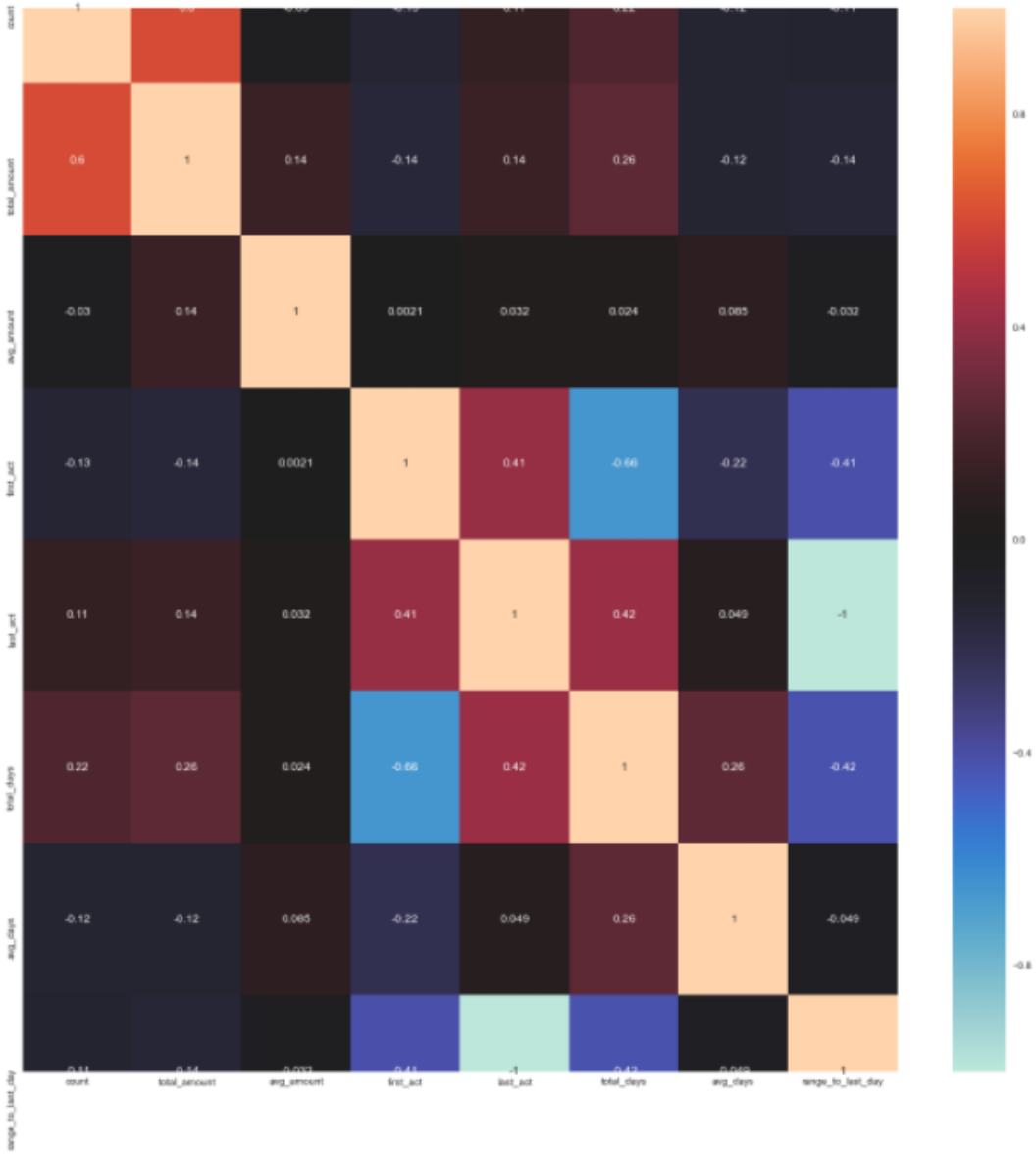

Figure 18

We need to remove either count or total_amount. Here, I remove total_amount.

We should also remove the last_act and range_to_last_day to increase the generalization of the model.

After constructing the training data, lets use cross-validation to tune the learning rate for our gradient boosting model.

```
learning_rate = [0.1, 0.2, 0.5, 0.05, 0.01]
for lr in learning_rate:
    gbc = GradientBoostingClassifier(learning_rate=lr)
    cv_results = cross_validate(gbc, X, y,scoring='roc_auc', cv=5)
    print(lr, cv_results['test_score'], np.mean(cv_results['test_score']))
```

```
0.1 [0.99642652 0.99739428 0.99757638 0.9985416  0.9974844 ] 0.9974846378078371
0.2 [0.99797941 0.99862209 0.99894777 0.99942696 0.99901551] 0.9987983481883751
0.5 [0.99860715 0.99881103 0.99925853 0.99960844 0.99940521] 0.999138072614819
0.05 [0.99216097 0.99354614 0.99400544 0.9953529  0.99268813] 0.9935507174651008
0.01 [0.95529717 0.95842376 0.95765246 0.96513804 0.96096609] 0.9594955035714957
```

Figure 18

From figure 18, we choose 0.5 to be the training rate.

Let's use the model to predict merchants who are highly likely to churn in the near future. There are 2 situations among the current active merchants.

1. The prediction of churn is 1, but the churn value according to the definition is 0.

merchant	count	total_amount	avg_amount	first_act	last_act	total_days	avg_days	range_to_last_day	churn	predicted_churn
0a4da2ae03	5	30298	6059.600000	460	689	229.452292	45.890458	40.399294	0	1
19bcda3c29	3	43078	14359.333333	651	689	38.196690	12.732230	40.173090	0	1
92643c7707	155	1455738	9391.858065	671	688	16.212280	0.104595	41.966829	0	1
fd30f4a268	3	83745	27915.000000	589	687	98.029433	32.676478	42.476250	0	1

The result is reasonable, since 3 of the 4 merchants are "fresh merchants" from the definition above in Q1, so they are not currently reliable.

For merchant 19bcda3c29, it has a extremely short ave_days, which means the regular period of payment is really short for him. 41 days of inactive could be strong evidence of being churn in the near future.

2. Both the predicted churn and churn values are 0, but the predicted probability is not very high.

merchant	count	total_amount	avg_amount	first_act	last_act	total_days	avg_days	range_to_last_day	churn	predicted_churn	predicted_0_prob
04a764b433	6	36606	6101.000000	676	688	12.037118	2.006186	41.314618	0	0	0.559688
341eef4f16	10	64505	6450.500000	222	687	465.154664	46.515466	42.153935	0	0	0.586808
633375ea72	3	21307	7102.333333	644	689	44.641308	14.880436	40.466528	0	0	0.572449
6983d0b1f6	5	11184	2236.800000	676	691	15.469421	3.093884	38.199803	0	0	0.569658
85d854dba6	58	320457	5525.120690	345	688	342.948333	5.912902	41.903368	0	0	0.577567
948c00f609	55	716198	13021.781818	463	688	224.335023	4.078819	41.910023	0	0	0.576211
b36f195561	29	468980	16171.724138	528	688	159.874618	5.512918	41.190428	0	0	0.584561
e0b65260bd	44	252899	5747.704545	374	687	313.031563	7.114354	42.387697	0	0	0.598629

The predictions also make sense. The first 4 merchants are “fresh merchants”, and the rest of the merchants still have a long period of inactive comparing with their average payment period.