Credit Card Customer Churn Analysis

Han Xu

Abstract

In this report, in addition to the general data preprocessing, summary statistics, correlation analysis and models, some insights were generated from the statistics and plots. In the exploratory data analysis, I identified the features of attrited customers and compared them with existing customers, generated some hypothesis to explain the difference, and proposed some suggestions for the bank. For the classification problem, I trained and tested models using grid search and cross-validation, and evaluated the model performance using recall score. Finally, I presented the structure and design of the web API interface implemented using flask, and the difficulties I encountered in the process.

Motivation

Credit card is something we use every day in US. Back in China, we do not use credit card that much. Instead, we use Wechat pay and Alipay, two most widely used online payment method. When I came across a credit card advertisement the other day on an online shopping app, I thought for a while whether to get one. If I decided to do so, I might stop using my current card. I was wondering, from a company's aspect, what factors can be used to determine whether a customer will decide to keep using this card or not (which could be called customer churn in business). If a bank can use the demographic and transaction features to predict whether a customer will leave their credit card service, they could take some measures (e.g., provide better services, special offers) to keep this customer. Therefore I searched for dataset with key word "credit card" on Kaggle, and found this dataset about credit card customer.

Description

Source

I downloaded the data on kaggle https://www.kaggle.com/sakshigoyal7/credit-card-customers/ in csv format.

Metadata

The metadata can be found here: https://www.kaggle.com/sakshigoyal7/credit-card-customers/metadata

License

CC0: Public Domain.

The original data has 10127 rows and 21 columns, including 10 integer variables, 7 decimal variables and 6 string variables, described as follows:

1) **CLIENTNUM**: Client number. Unique identifier for the customer holding the account.
2) **Attrition_Flag**: Internal event (customer activity) variable - if the account is closed then 1 else 0.
3) **Customer_Age**: Demographic variable - Customer's Age in Years.
4) **Gender**: Demographic variable - M=Male, F=Female.
5) **Dependent_count**: Demographic variable - Number of dependents.
6) **Education_Level**: Demographic variable - Educational Qualification of the account holder (example: high school).
7) **Marital_Status**: Demographic variable - Married, Single, Divorced, Unknown.
8) **Income_Category**: Demographic variable - Annual Income Category of the account holder.
9) **Card_Category**: Product Variable - Type of Card (Blue, Silver, Gold, Platinum).
10) **Months_on_book**: Period of relationship with bank.
11) **Total_Relationship_Count**: Total no. of products held by the customer.
12) **Months_Inactive_12_mon**: No. of months inactive in the last 12 months.
13) **Contacts_Count_12_mon**: No. of Contacts in the last 12 months.
14) **Credit_Limit**: Credit Limit on the Credit Card.
15) **Total_Revolving_Bal**: Total Revolving Balance on the Credit Card.
16) **Avg_Open_To_Buy**: Open to Buy Credit Line (Average of last 12 months).
17) **Total_Amt_Chng_Q4_Q1**: Change in Transaction Amount (Q4 over Q1).

18) **Total_Trans_Amt**: Total Transaction Amount (Last 12 months).

19) **Total_Trans_Ct**: Total Transaction Count (Last 12 months).

20) **Total_Ct_Chng_Q4_Q1**: Change in Transaction Count (Q4 over Q1).

21) **Avg_Utilization_Ratio**: Average Card Utilization Ratio.

22) **Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1**: Naive Bayes

23) **Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2**: Naive Bayes

Attrition_Flag is the target/dependent variable.

Data Prepocessing

Load the data

```
# import the libraries and data
import pandas as pd
import numpy as np
card_data = pd.read_csv("BankChurners.csv")

card_data.shape
```

```
(10127, 23)
```

Droppping unnecessary columns

Drop the first column, client number.

The last two columns are based on Naive Bayes models built on the data and are not part of the original data. Therefore, we drop the last two colums as well.

```
card_data = card_data.iloc[:,1:-2]

card_data.dtypes
```

```
Attrition_Flag              object
Customer_Age                 int64
Gender                      object
Dependent_count              int64
Education_Level             object
Marital_Status              object
Income_Category             object
Card_Category               object
Months_on_book               int64
Total_Relationship_Count     int64
```

```
Months_Inactive_12_mon        int64
Contacts_Count_12_mon         int64
Credit_Limit                float64
Total_Revolving_Bal           int64
Avg_Open_To_Buy             float64
Total_Amt_Chng_Q4_Q1        float64
Total_Trans_Amt               int64
Total_Trans_Ct                int64
Total_Ct_Chng_Q4_Q1         float64
Avg_Utilization_Ratio       float64
dtype: object
```

Now we have 9 integer variables, 5 float variables and 6 object variables.

Checking for missing values

```
card_data.isna().sum()
```

```
Attrition_Flag              0
Customer_Age                0
Gender                      0
Dependent_count             0
Education_Level             0
Marital_Status              0
Income_Category             0
Card_Category               0
Months_on_book              0
Total_Relationship_Count    0
Months_Inactive_12_mon      0
Contacts_Count_12_mon       0
Credit_Limit                0
Total_Revolving_Bal         0
Avg_Open_To_Buy             0
Total_Amt_Chng_Q4_Q1        0
Total_Trans_Amt             0
Total_Trans_Ct              0
Total_Ct_Chng_Q4_Q1         0
Avg_Utilization_Ratio       0
dtype: int64
```

No misisng values found.

Checking for duplicates

```
card_data[card_data.duplicated()].T
```

Attrition_Flag

Customer_Age

Gender

Dependent_count

Education_Level

Marital_Status

Income_Category

Card_Category

Months_on_book

Total_Relationship_Count

Months_Inactive_12_mon

Contacts_Count_12_mon

Credit_Limit

Total_Revolving_Bal

Avg_Open_To_Buy

Total_Amt_Chng_Q4_Q1

Total_Trans_Amt

Total_Trans_Ct

Total_Ct_Chng_Q4_Q1

Avg_Utilization_Ratio

No duplicate records found.

```
# get a copy of the raw data
raw_data = pd.read_csv("BankChurners.csv").iloc[:,1:-2] # copy a raw da
ta
raw_num = raw_data.select_dtypes(exclude=['object']) # raw data numeric
al columns
raw_num['Attrition_Flag'] = raw_data['Attrition_Flag']
```

Exploratory Data Analysis

Summary statistics

Display the summary statistics for numerical variables and categorical variables respectively.

```
# numerical variables
card_num = card_data.select_dtypes(exclude=['object'])
card_num.describe().T
```

```
# categorical variables
card_cat = card_data.select_dtypes(exclude=['int', 'float'])
card_cat.describe().T
```

Table 1

	count	mean	std	min	25%	50%	75%	max
Customer_Age	10127	46.32596	8.016814	26	41	46	52	73
Dependent_count	10127	2.346203	1.298908	0	1	2	3	5
Months_on_book	10127	35.92841	7.986416	13	31	36	40	56
Total_Relationship_Count	10127	3.81258	1.554408	1	3	4	5	6
Months_Inactive_12_mon	10127	2.341167	1.010622	0	2	2	3	6
Contacts_Count_12_mon	10127	2.455317	1.106225	0	2	2	3	6
Credit_Limit	10127	8631.954	9088.777	1438.3	2555	4549	11067.5	34516
Total_Revolving_Bal	10127	1162.814	814.9873	0	359	1276	1784	2517
Avg_Open_To_Buy	10127	7469.14	9090.685	3	1324.5	3474	9859	34516
Total_Amt_Chng_Q4_Q1	10127	0.759941	0.219207	0	0.631	0.736	0.859	3.397
Total_Trans_Amt	10127	4404.086	3397.129	510	2155.5	3899	4741	18484
Total_Trans_Ct	10127	64.8587	23.47257	10	45	67	81	139
Total_Ct_Chng_Q4_Q1	10127	0.712222	0.238086	0	0.582	0.702	0.818	3.714
Avg_Utilization_Ratio	10127	0.274894	0.275691	0	0.023	0.0176	0.503	0.999

Distributions

Categorical variables

```
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib
import warnings
warnings.filterwarnings("ignore")

cat = card_data.select_dtypes(include=['object']).columns
fig, ax = plt.subplots(figsize =(18, 18))

for i in (enumerate(cat)):
    plt.subplot(4, 2, i[0]+1)
    if card_data[i[1]].value_counts().count() > 2:
        ax = sns.countplot(y = i[1], data = card_data, order=card_data
[i[1]].value_counts().index)
        pct = card_data[i[1]].value_counts(ascending=False, normalize=T
rue).values * 100
        ax.grid(False)
        ax.grid(b=None)
        ax.xaxis.set_ticks_position('none')
        ax.yaxis.set_ticks_position('none')
        ax.set_xticklabels('')
```

```
    else:
        _, texts, pcts = plt.pie(
            card_data[i[1]].value_counts(),
            labels=card_data[i[1]].value_counts().index,
            autopct='%1.1f%%',
            wedgeprops={'linewidth': 3.0, 'edgecolor': 'white'},
            startangle=50)
        for pcts in pcts:
          pcts.set_color('white')
        plt.title(i[1], fontsize=20)
```

From the plots above we can conclude that:

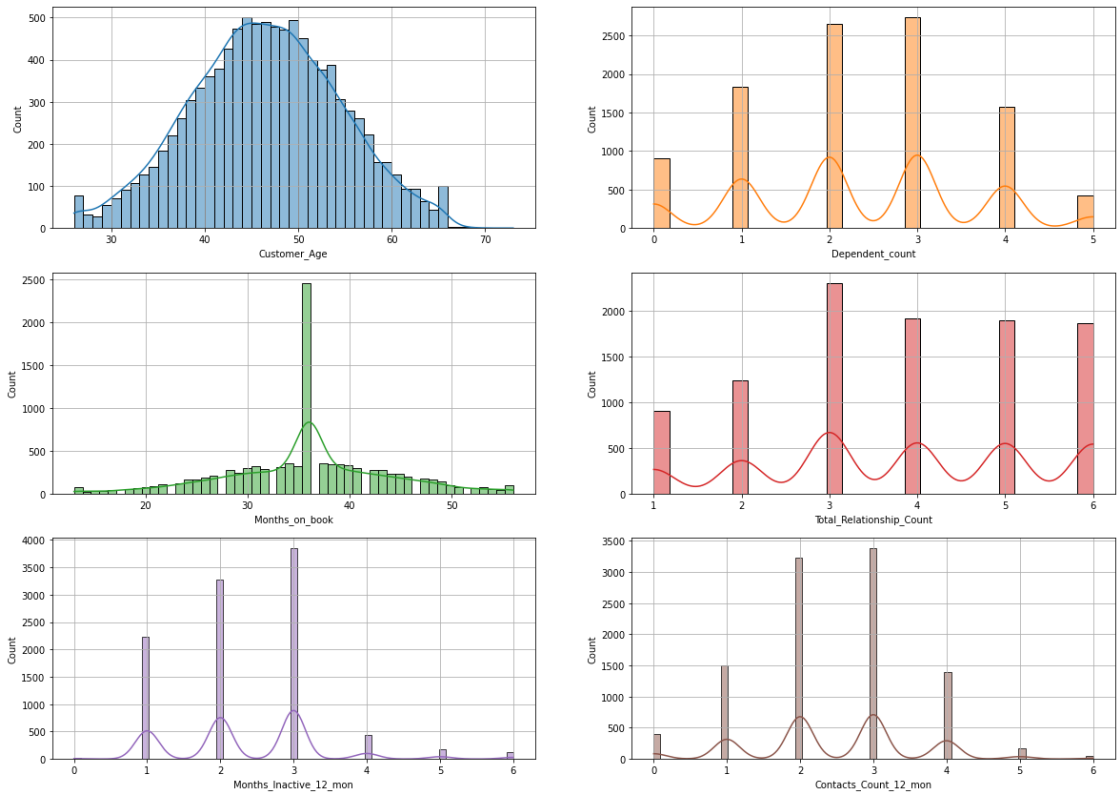
- Only 16.1% customers attrited.
- 52.9% of the customers were female.
- There are 7 types of education level in the customers, namely Unkown, Uneducated, High school, College, Graduate, Doctorate and Post-Graduate. Graduate education has the highest proportion, followed by High school, Unknown and Uneducated. Doctorate has the lowest proportion.
- There are 4 types of marital status in the customers, namely Married, Single, Unkown and Divorced. The proportion of Married and Single customers are much higher than that have Unkown marital status or are Divorced.

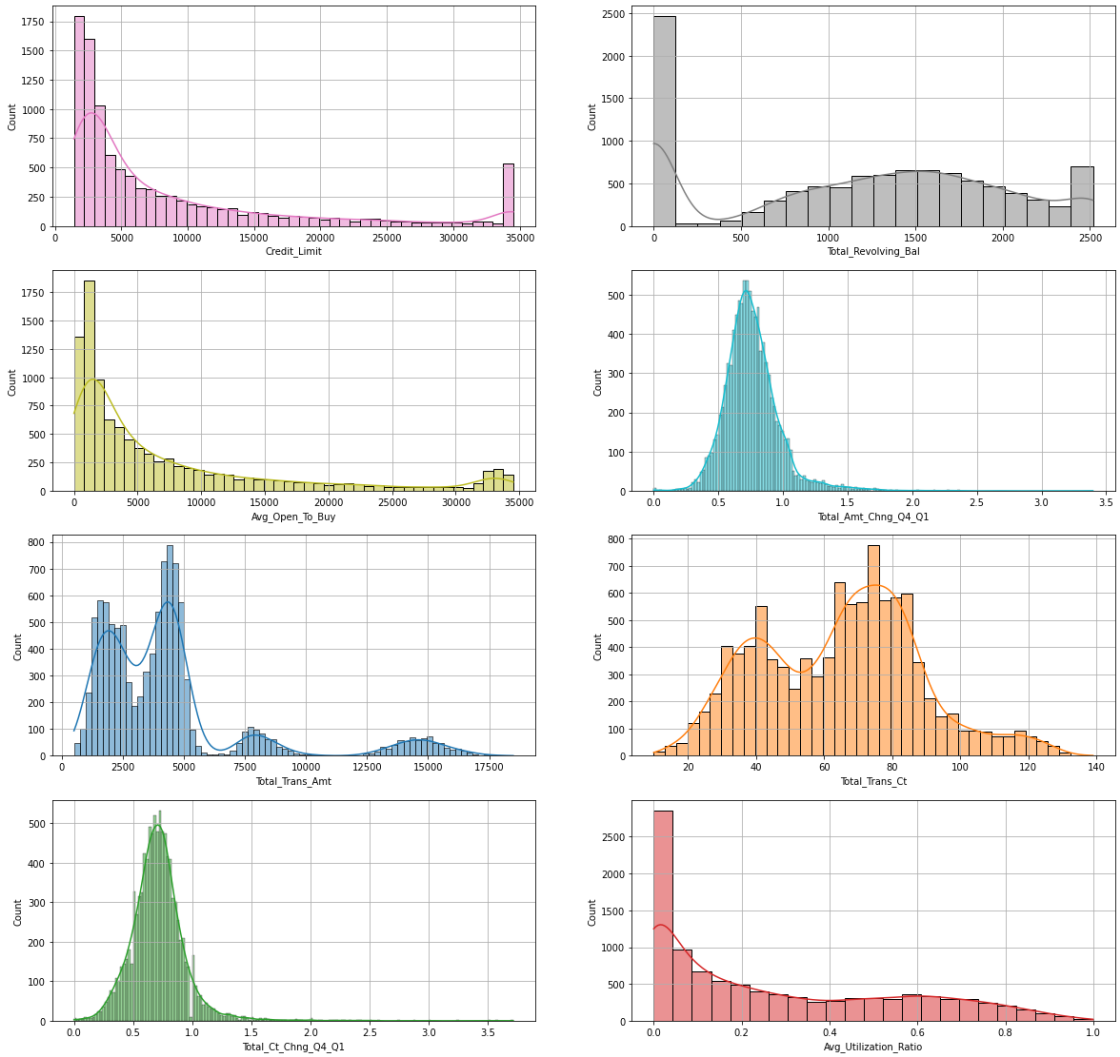
- There are 6 types of income categories, namely Less than $40k, 40k - 60k, 60k - 80k, 80k - 120k, 120k+ and Unkown. An interesting phenomenon is that the number of customers that have high or low income are more than the mid-income customers, which can be described as a “funnel shape”.

- There are 4 types of card categories, namely Blue, Silver, Gold and Platinum. The Blue card customers account for the most part.

Numercial Variables

```
numeric = card_data.select_dtypes(exclude=['object']).columns
fig, ax = plt.subplots(figsize =(20, 35))
# there are 14 numerical variables in total, use a different color for
each distribution
colors = sns.color_palette(n_colors=14)
for i in (enumerate(numeric)):
    plt.subplot(7, 2, i[0]+1)
    sns.histplot(x = i[1], data = card_data, color=colors[i[0]], kde=Tr
ue)
    plt.grid(b=None)
```





From the distributions of numerical variables we can concluded that:

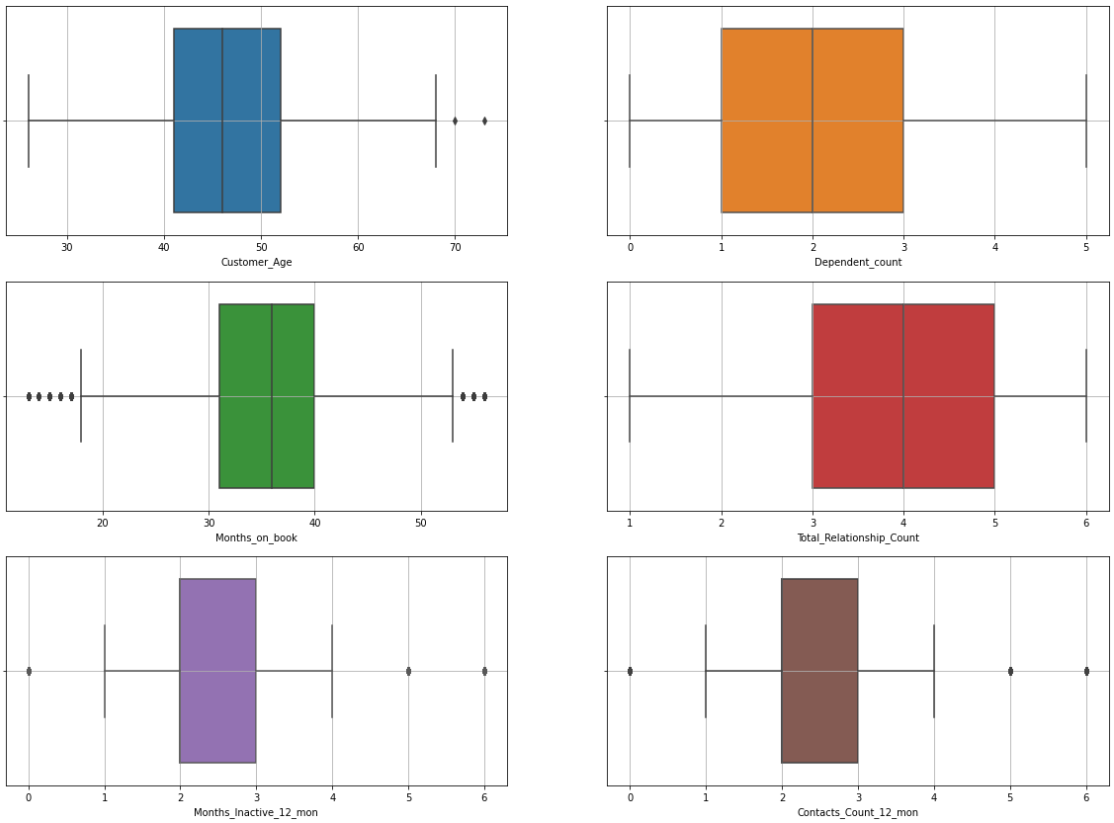
- Customer_Age is approximately normally distributed, ranging from 26 to 73 years old, with highest distribution in the middle (median 46).
- Dependent_Count (number of dependents) takes discrete values from 0 to 5, and is also approximately normal.
- Months_on_book (period of relationship with bank) takes value from 13 to 56, and there is a uncommonly high frequency in value 36. But since it is nearly in the middle of the range (not skewing the data), we can termporarily put this aside.
- Total_Relationship_Count (total number of products held by the customer) takes discrete value from 1 to 6, and is also close to normal.
- Months_Inactive_12_mon (number of months inactive in the last 12 months) takes discrete value from 0 to 6, and most of the customers are inactive for 1,

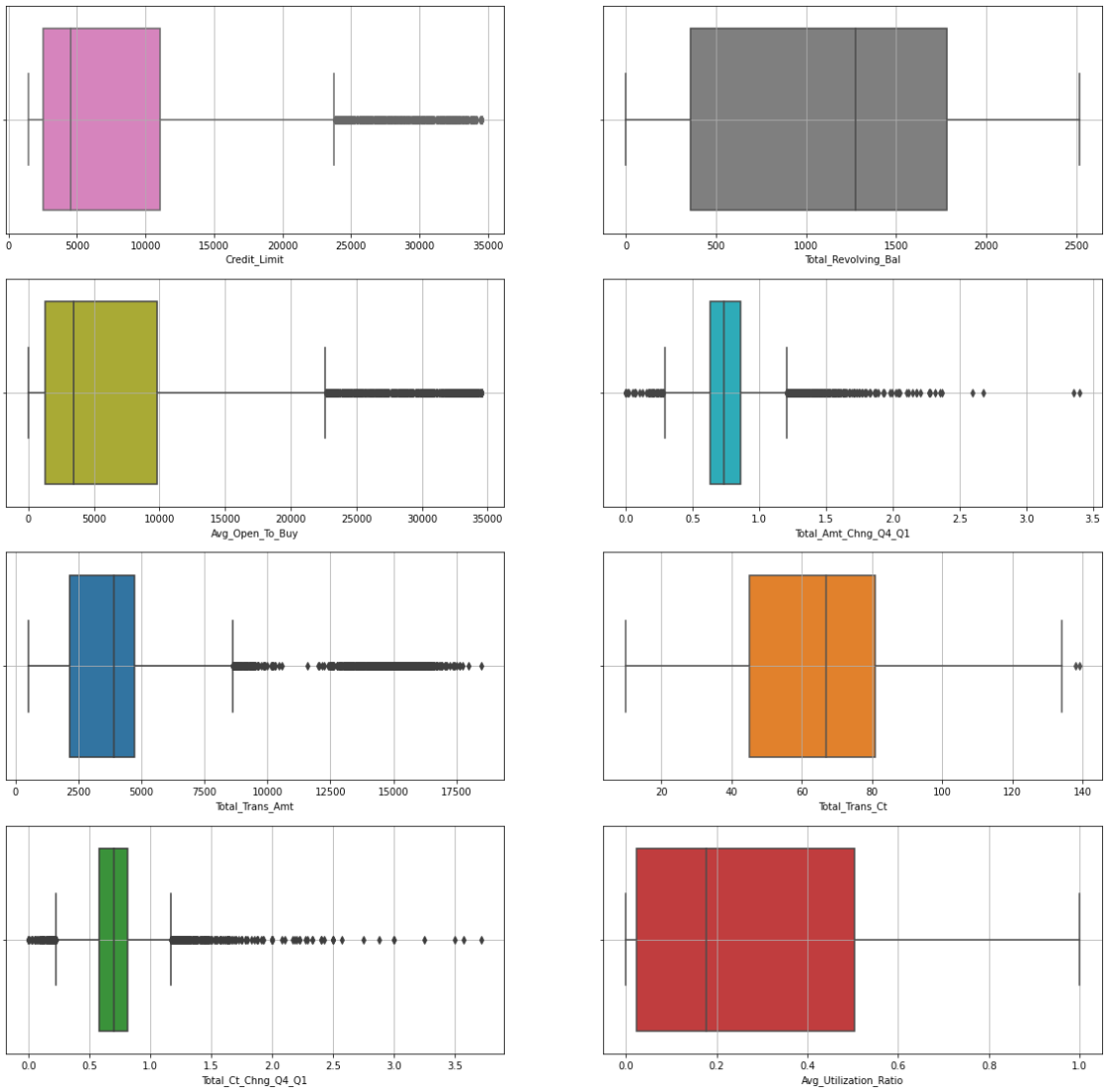
2 or 3 months. This feature looks right-skewed. Inactive month might be a good indicator of a customer's leaving.

- Contacts_Count_12_mon (number of contacts in the last 12 months) also takes discrete value from 0 to 6, and is nearly normal.
- The rest 8 features, Credit_Limit, Total_Revolving_Bal, Avg_Open_To_Buy, Total_Amt_Chng_Q4_Q1, Total_Trans_Amt, Total_Trans_Ct, Total_Ct_Chng_Q4_Q1 and Avg_Utilization_Ratio, take continuous values. Nearly all features are right-skewed (distributed towards 0). Therefore, we need to check for the skewness and outliers.

Checking for skewness and outliers

```
fig, ax = plt.subplots(figsize =(20, 35))
# there are 14 numerical variables in total, use a different color for
each distribution
colors = sns.color_palette(n_colors=14)
for i in (enumerate(numeric)):
    plt.subplot(7, 2, i[0]+1)
    sns.boxplot(x = i[1], data = card_data, color=colors[i[0]])
    plt.grid(b=None)
```





From the boxplot we can see that:

- There are no outliers in Dependent_count, Total_Relationship_Count, Tol_Revolving_Bal, and Aug_Utilization_Ratio.
- There only a few outliers in Customer_Age, Month_on_book, Months_Inactive_12_mon, Contacts_Count_12_mon, and Total_Trans_Ct.
- There are a lot of outliers in Credit_Limit, Avg_Open_to_Buy, Total_Amt_Chng_Q4_Q1, Total_Trans_Amt, and Total_Ct_Chng_Q4_Q1.

```
card_data.select_dtypes(exclude=['object']).skew().abs().sort_values(as
cending=False)
```

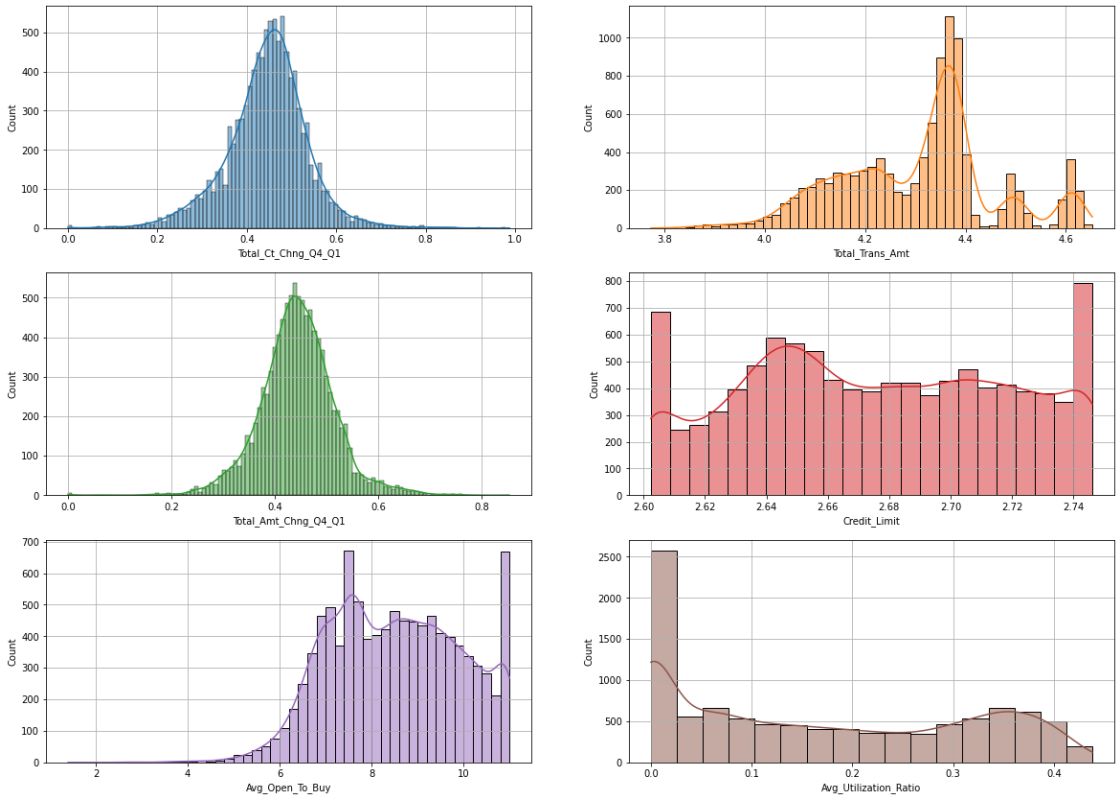
```
Total_Ct_Chng_Q4_Q1          2.064031
Total_Trans_Amt              2.041003
Total_Amt_Chng_Q4_Q1         1.732063
```

```
Credit_Limit                1.666726
Avg_Open_To_Buy             1.661697
Avg_Utilization_Ratio       0.718008
Months_Inactive_12_mon      0.633061
Total_Relationship_Count    0.162452
Total_Trans_Ct              0.153673
Total_Revolving_Bal         0.148837
Months_on_book              0.106565
Customer_Age                0.033605
Dependent_count             0.020826
Contacts_Count_12_mon       0.011006
dtype: float64
```

- The top 5 variables have skewness > 1 or < -1, and are exactly the variables with a lot of outliers (Credit_Limit, Avg_Open_to_Buy, Total_Amt_Chng_Q4_Q1, Total_Trans_Amt, and Total_Ct_Chng_Q4_Q1).
- 2 variables has skewness in (-0.5, -1) or (0.5, 1), Avg_Utilization_Ratio and Months_Inactive_12_mon.

Credit_Limit, Avg_Open_to_Buy, Total_Amt_Chng_Q4_Q1, Total_Trans_Amt, Total_Ct_Chng_Q4_Q1 and Avg_Utilization_Ratio are all continuos variables that are highly right-skewed. Thus we can use Box-Cox transformation to remove the skewness.

```
from scipy.special import boxcox1p
from scipy.stats import boxcox_normmax

skewed_col = card_data.select_dtypes(exclude=['object']).skew().abs().s
ort_values(ascending=False).head(6)
for c in skewed_col.index:
    card_data[c] = boxcox1p(card_data[c], boxcox_normmax(card_data[c] +
 1))

fig, ax = plt.subplots(figsize =(20, 35))
colors = sns.color_palette(n_colors=14)
for i in (enumerate(skewed_col.index)):
    plt.subplot(7, 2, i[0]+1)
    sns.histplot(x = i[1], data = card_data, color=colors[i[0]], kde=Tr
ue)
    plt.grid(b=None)
```




```
card_data[skewed_col.index].skew().abs().sort_values(ascending=False)
```

```
Avg_Utilization_Ratio    0.268783
Total_Ct_Chng_Q4_Q1      0.124341
Total_Trans_Amt          0.070385
Avg_Open_To_Buy          0.068015
Total_Amt_Chng_Q4_Q1     0.041141
Credit_Limit             0.022279
dtype: float64
```

After the transformation, the 6 skewed continuous variables are all within the skewness limit (0.5).

Scaling numerical variables

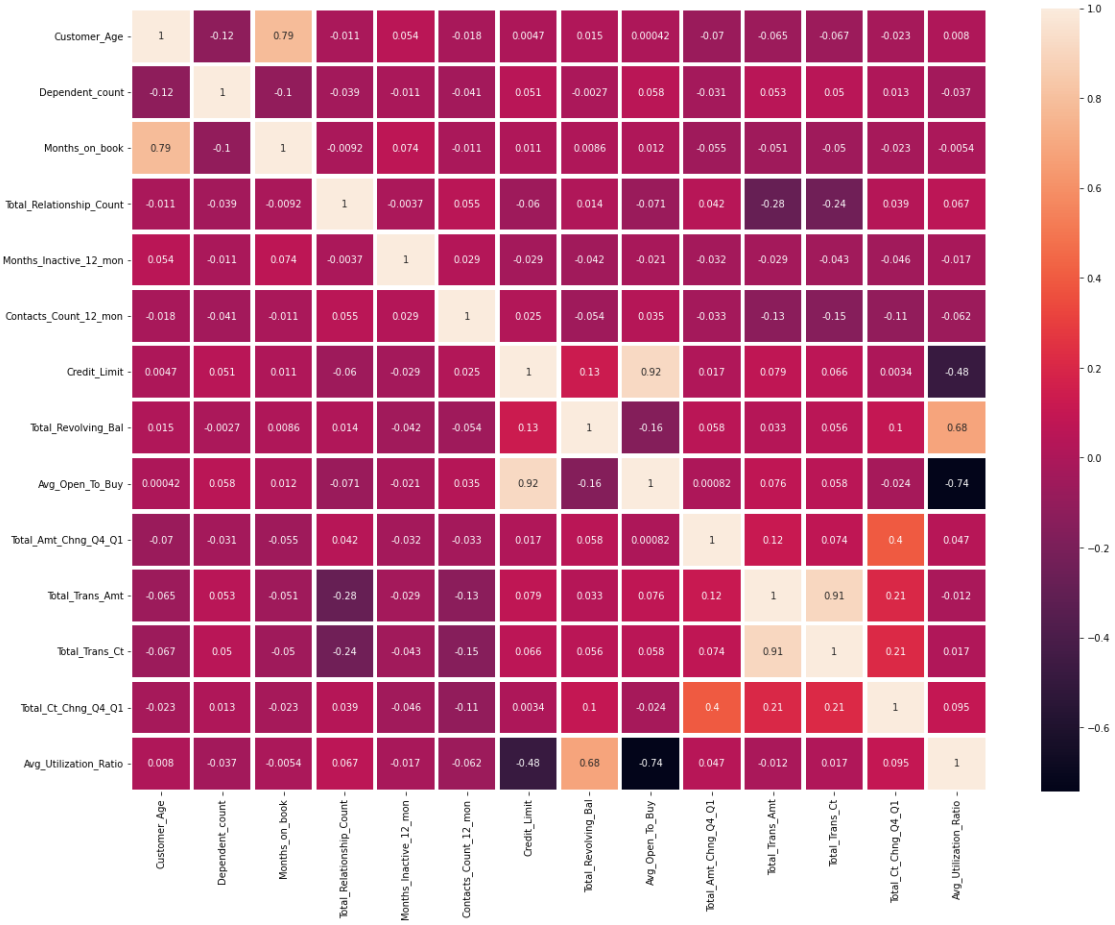
Scale all the numerical variables using min-max scaler.

```
from sklearn.preprocessing import MinMaxScaler
for c in card_data[numeric]:
    card_data[c] = MinMaxScaler().fit_transform(card_data[[c]])
```

Correlation matrix

```
corrmat = card_data.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,15))
```

```
g=sns.heatmap(card_data[top_corr_features].corr(), annot=True, linewidths=3.5)
```



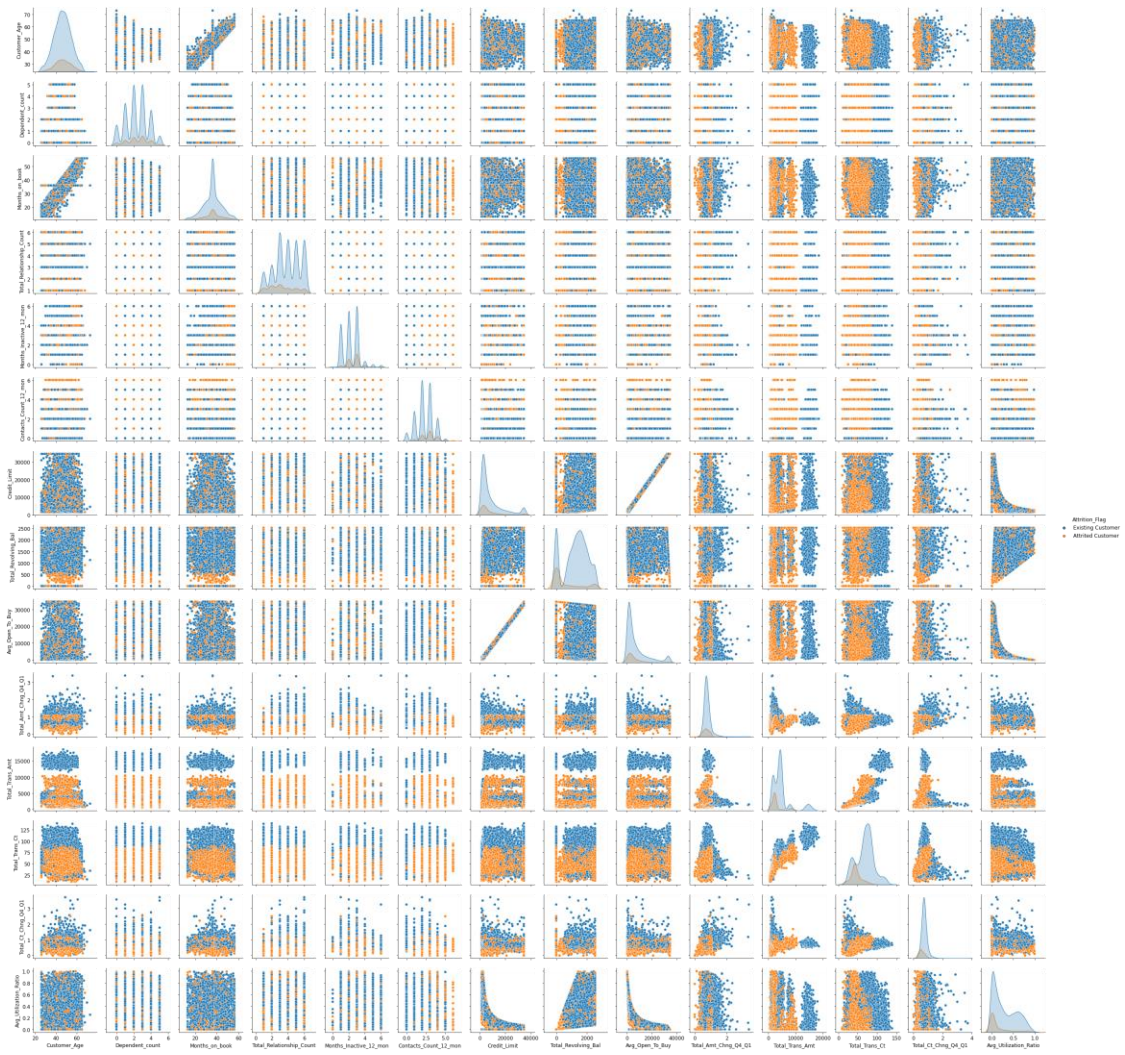
Variables that are highly correlated (correlation cooefficient > 0.5 or < -0.5):

- Months_on_book and Customer_Age (0.79)
- Avg_Utilization_Ratio and Avg_Open_To_Buy (-0.54)

Bivariate analysis

For numerical variables, we plot the

```
sns.pairplot(data=raw_num, hue="Attrition_Flag")
plt.show()
```



From the pairplot above we can tell that:

- Contacts_Count_12_mon is higher in attrited customers.
- Total_Revolving_Bal, Total_Amt_Chng_Q4_Q1, Total_Trans_Amt, Total_Trans_Ct, Total_Ct_Chng_Q4_Q1 is lower in attrited customers.
- Months_on_book is significantly postively correlated with Customer_Age.
- Avg_Open_To_Buy is significantly positively correlated with Credit_Limit.
- Total_Trans_Ct is positively correlated with Total_Trans_Amt.
- Avg_Utilization_Ratio is inversely proportional to both Credit_Limit and Avg_Open_To_Buy.

Encoding categorical variables

There are two types of categorical variables in this dataset: nominal and ordinal.

Nominal variables include: Gender and Marital_Status.

Ordinal variables include: Income_Category, Card_Category and Education_Level, which have orders in the levels.

First encode the ordinal variables.

```
Income_Category_map = {'Less than $40K' : 0, '$40K - $60K': 1,'$60K -
$80K': 2,'$80K - $120K': 3,'$120K +': 4,'Unknown': 5}
Card_Category_map = {'Blue': 0,'Silver': 1,'Gold': 2,'Platinum' : 3}
Attrition_Flag_map = {'Existing Customer': 0,'Attrited Customer': 1}
Education_Level_map = {'Uneducated': 0,'High School': 1,'College': 2,'G
raduate': 3,'Post-Graduate': 4,'Doctorate': 5,'Unknown': 6}

card_data.loc[:, 'Income_Category'] = card_data['Income_Category'].map
(Income_Category_map)
card_data.loc[:, 'Card_Category'] = card_data['Card_Category'].map(Card
_Category_map)
card_data.loc[:, 'Attrition_Flag'] = card_data['Attrition_Flag'].map(At
trition_Flag_map)
card_data.loc[:, 'Education_Level'] = card_data['Education_Level'].map
(Education_Level_map)
```

Then encode the nominal variables Gender and Marital_Status using one-hot encoding.

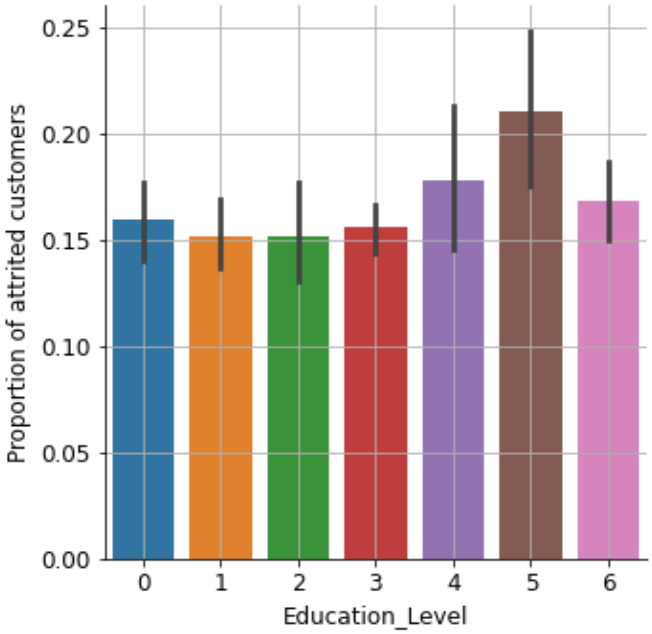
```
card_data['Female'] = np.where(card_data['Gender']=='F', 1, 0)
card_data['Married'] = np.where(card_data['Marital_Status']=='Married',
 1, 0)
card_data['Single'] = np.where(card_data['Marital_Status']=='Single', 1,
 0)
card_data['Divorced'] = np.where(card_data['Marital_Status']=='Divorced
', 1, 0)
# card_data.drop(columns=['Gender', 'Marital_Status'])

# for further analysis (the factor plot), we first keep the original Ge
nder and Marital_Status columns
# and encode them using integers
from sklearn.preprocessing import LabelEncoder
cat_cols = [x for x in card_data.columns if card_data[x].dtype == 'obje
ct']
for c in cat_cols:
    card_data.loc[:, c] = LabelEncoder().fit_transform(card_data.loc[:,
 c])
```

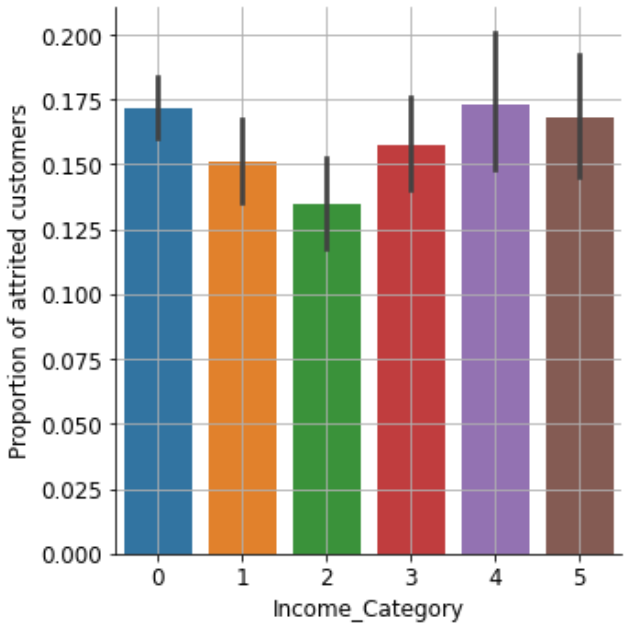
Categorical variables with target

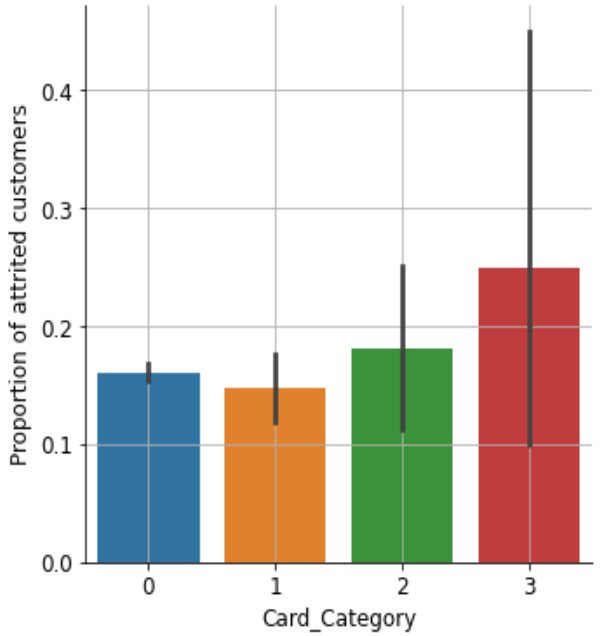
First examine the relationship between 3 ordinal variables and the target variable. Using factor plots (y-axis denotes the number of attrited customers divided by the total number of customers, x-axis is the categorical variable), we wish to answer the following question: Are education level positively / negatively correlated with

customer attrition? Are income level positively / negatively correlated with customer attrition? Which card category has the highest customer attrition rate?

```
categ = ["Education_Level","Income_Category","Card_Category"]
for i in (range(3)):
    sns.factorplot(data=card_data, x=categ[i], y ='Attrition_Flag', kin
d='bar')
    plt.xlabel(categ[i])
    plt.ylabel('Proportion of attrited customers')
    plt.grid(b=None)
```





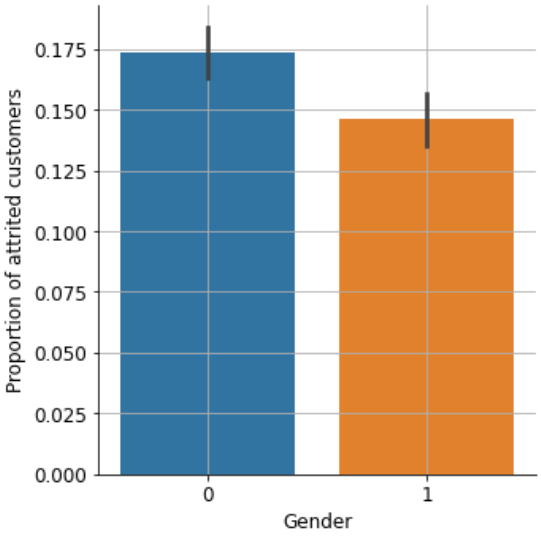


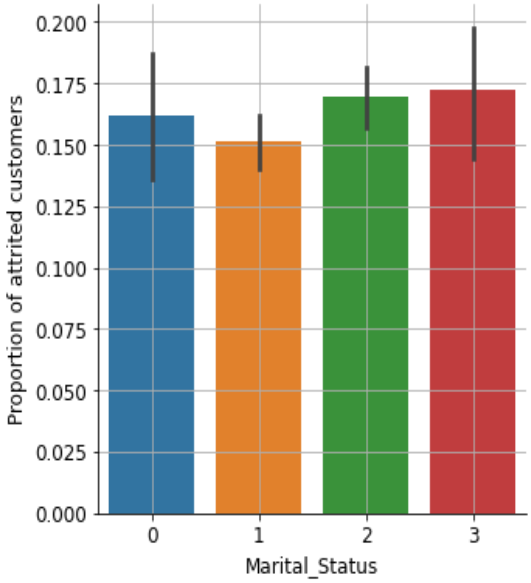
- From the factor plot of education level and the target variable, we can conclude that Doctorate (5 after encoding) education level customers have the highest proportion of attrition, followed by Post-Graduate (4 after encoding). Based on this, we could propose a hypothesis that customers with higher education level are more likely to leave the credit card service. One explanation is banks are more willing to provide credit card service for customers with higher education level, who then have more choices in banks. Anyways, banks can use more data to test this hypothesis.

- However, the same logic cannot be applied to the income levels. We can see that the mid-income customers have the lowest attrition rate. The overall rate has a 'V' shape, higher on the two sides and lower in the middle. One possible explanation could be that customers with lowest income have a higher probability of unable to pay the loans and are forced to leave the service; on the other hand, cusomers with the highest income have more choices in banks and are thus also likely to leave the current serice. Therefore, the middle-income customers become the most stable population.

- The Platinum card customers have a higher attrition rate than customers with other card types. This tells the bank owners that Platinum card services should be improved to retain customers.

```
categ2 = ["Gender","Marital_Status"]
for i in (range(2)):
    sns.factorplot(data=card_data, x=categ2[i], y ='Attrition_Flag', ki
nd='bar')
    plt.xlabel(categ2[i])
    plt.ylabel('Proportion of attrited customers')
    plt.grid(b=None)
```





Gender encoding: female is encoded as 0 and male 1.

Marital status encoding: married is encoded as 1, single 2, unknown 3, divorced 0.

- Are men more likely to attrite than women?
- Are there significant differences between the attrition rate among difference marital status? (What kind of customers are more stable, single or married?)
- Women customers have a higher attrition proportion than men customers. (Might because they are bigger fan of credit cards and do more research and comparisons on different card providers.)

- Married customers are more stable than single and divorced customers. (Maybe because married customers are just more stable, or they are too busy to think about changing cards.)

Sample imbalance

Since we have more non-attrited customers than attrited customers, the dataset is imbalanced. We could use oversampling method to address this issue.

There is a oversampling method called SMOTE, which we apply to our dataset.

```
from imblearn.over_sampling import SMOTE
X = card_data.drop('Attrition_Flag', axis=1)
y = card_data['Attrition_Flag']
smote = SMOTE(random_state=6)
X_smote, y_smote = smote.fit_resample(X, y)
print(f"Shape of X before SMOTE: {X.shape}
Shape of X after SMOTE: {X_smote.shape}")
print('\nBalance of positive and negative classes (%):')
y_smote.value_counts(normalize=True) * 100
```

```
Shape of X before SMOTE: (10127, 23)
Shape of X after SMOTE: (17000, 23)

Balance of positive and negative classes (%):
1    50.0
0    50.0
Name: Attrition_Flag, dtype: float64
```

Conclusions of EDA

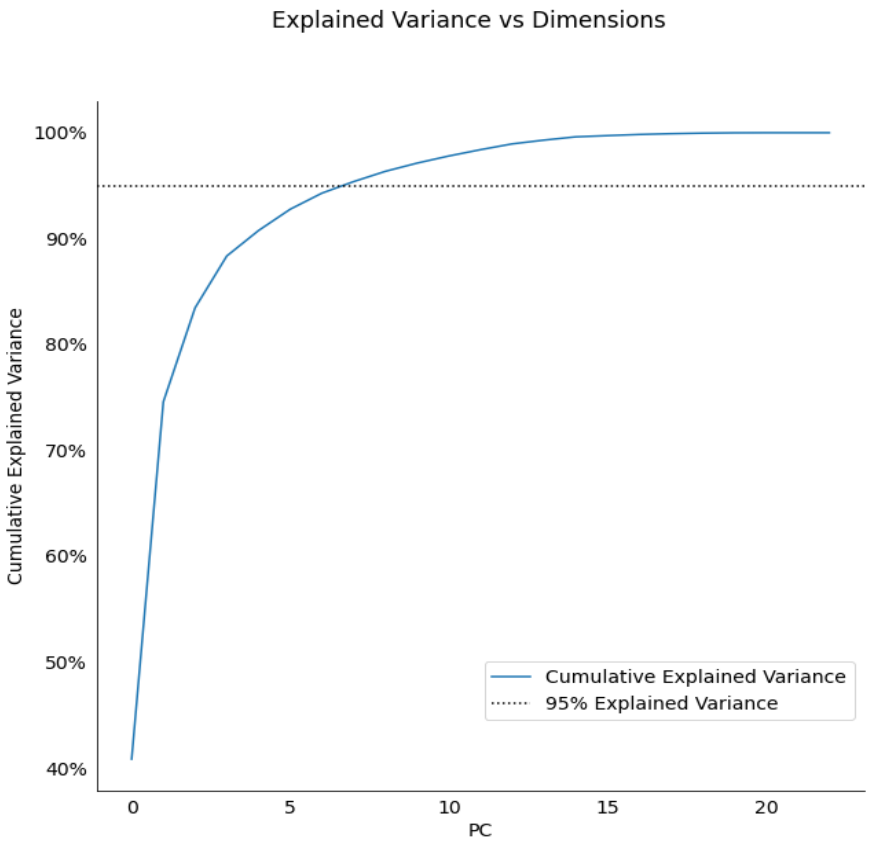
- 16.1% customers attrited.
- Attrited customers have higher number of contacts in the last 12 months.
- Attrited customers have lower total revolving balance on the credit card, change in transaction amount (Q4 over Q1), total transaction amount (last 12 months), total transaction count (last 12 months), and change in transaction count (Q4 over Q1).
- Doctorate (5 after encoding) education level customers have the highest proportion of attrition, followed by Post-Graduate (4 after encoding).
- Mid-income customers and Married customers have the lowest attrition rate.
- The Platinum card customers have a higher attrition rate than customers with other card types.
- Women customers have a higher attrition proportion than men customers.

Dimensionality Reduction

PCA

Since there are too many variables, we use PCA to reduce the dimensionality of the dataset.

```
from sklearn.decomposition import PCA
import matplotlib.ticker as mtick
import matplotlib as mpl

pca = PCA()
pca.fit(X)

cumsum = np.cumsum(pca.explained_variance_ratio_)*100
d = [n for n in range(len(cumsum))]

fig, ax = plt.subplots(figsize =(10, 10))
fig.patch.set_facecolor('white')

ax.plot(d,cumsum, label='Cumulative Explained Variance')
ax.axhline(y = 95, color='black', linestyle=':', label = '95% Explained
 Variance')
ax.legend(loc='best')
ax.xaxis.set_ticks_position('none')
ax.yaxis.set_ticks_position('none')
for i in ['top','right']:
    ax.spines[i].set_visible(False)

ax.yaxis.set_major_formatter(mtick.PercentFormatter())
arrowprops = dict(arrowstyle="->", connectionstyle="angle3,angleA=0,ang
leB=-90")

plt.legend(bbox_to_anchor = (1, 0.2))

plt.suptitle('Explained Variance vs Dimensions')
plt.ylabel('Cumulative Explained Variance')
plt.xlabel('PC');
```



We can see from the elbow plot that 8 variables can retain 95% of the information in the independent variables.

```
pca = PCA(.95)
pca.fit(X)

X_pca = pd.DataFrame(pca.transform(X))
print(f"Shape of X before PCA: {X.shape}
Shape of X after PCA: {X_pca.shape}")
```

```
Shape of X before PCA: (10127, 23)
Shape of X after PCA: (10127, 8)
```

Models

Logistic regression

Logistic model and works good for binary classification problems, and is highly interpretable.

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score
from sklearn.metrics import confusion_matrix
```

```
X = card_data.loc[:, ~card_data.columns.isin(['Attrition_Flag', 'Gender
', 'Marital_Status'])]
y = card_data['Attrition_Flag']

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=
666, stratify=y, train_size=0.8)
```

The dataset is imbalanced and we need to adjust the class_weight to ensure better performance.

```
from sklearn.linear_model import LogisticRegression
lgr = LogisticRegression(class_weight='balanced')
```

GridSearch & Parameter Tuning

- Use GridSearch for parameter tuning on penalty and C in regularization.
- We are more interested in the customer who attrited, so recall score is an appropriate metric in cross validation and measuring generalization ability.

```
C = np.arange(0.001,0.01,0.001)
penalty = ['l1', 'l2']

param_grid = dict(C=C, penalty=penalty)

grid_search = GridSearchCV(lgr, param_grid, scoring='recall', cv=5)
grid_result = grid_search.fit(X_train, y_train)

result_lgr = pd.DataFrame(grid_result.cv_results_)
result_lgr.sort_values(by='mean_test_score', ascending=False)[:4]

best_lgr = grid_search.best_estimator_
y_pred = best_lgr.predict(X_test)
print(round(recall_score(y_test, y_pred),2))
```

```
0.77
```

- The best model has a recall score of 0.77 on testing dataset.

Interpreting the result

```
Coeffient = best_lgr.coef_.flatten()
pd.DataFrame({'Coeffient':Coeffient}, index=X.columns).sort_values(by='
Coeffient', ascending=False)
```

Table 2

	Coeffient
Contacts_Count_12_mon	0.925239
Months_Inactive_12_mon	0.682052
Female	0.178925

Card_Category	0.106254
Dependent_count	0.086931
Single	0.037373
Education_Level	0.032485
Income_Category	-0.01166
Divorced	-0.02146
Months_on_book	-0.02183
Customer_Age	-0.03241
Avg_Open_To_Buy	-0.10238
Married	-0.11876
Credit_Limit	-0.21327
Total_Amt_Chng_Q4_Q1	-0.3108
Total_Trans_Amt	-0.67364
Avg_Utilization_Ratio	-0.74872
Total_Revolving_Bal	-0.75846
Total_Relationship_Count	-0.88275
Total_Ct_Chng_Q4_Q1	-0.89471
Total_Trans_Ct	-1.81582

The coefficients are ranked from very positive to very negative.

- Five features that are most positively correlated with our target variable are: Contacts_Count_12_mon, Months_Inactive_12_mon, Female, Card_Category, and Dependent_count.
- Five features that are most negatively correlated with our target variable are: Total_Trans_Ct, Total_Ct_Chng_Q4_Q1, Total_Relationship_Count, Total_Revolving_Bal, and Avg_Utilization_Ratio.

XGBoost

```
import xgboost as xgb
from sklearn.metrics import classification_report, roc_auc_score, confu
sion_matrix, plot_confusion_matrix, roc_curve
xgb_model = xgb.XGBClassifier(random_state=6, use_label_encoder=False,
n_jobs=-1)

XGBClassifier(n_jobs=-1, random_state=6, use_label_encoder=False)
```

- Use grid search to tune the parameters: learning rate, max depth and number of estimators.
- Also use recall score as the evaluation metric of the model performance.

```
lr = [0.05,0.1]
max_depth=np.arange(5,10)
n_estimators=[500, 1000, 1500]
```

```
#gamma=np.arange(0,0.1,0.01)

param_grid = dict(learning_rate=lr, max_depth=max_depth, n_estimators=n
_estimators)

grid_search = GridSearchCV(xgb_model, param_grid, scoring='recall', cv=
5)
grid_result = grid_search.fit(X_train, y_train)

result_lgr = pd.DataFrame(grid_result.cv_results_)
result_lgr.sort_values(by='mean_test_score', ascending=False)[:4]

best_xgb = grid_search.best_estimator_
y_pred = best_xgb.predict(X_test)
print(round(recall_score(y_test, y_pred),2))
```

```
0.9
```

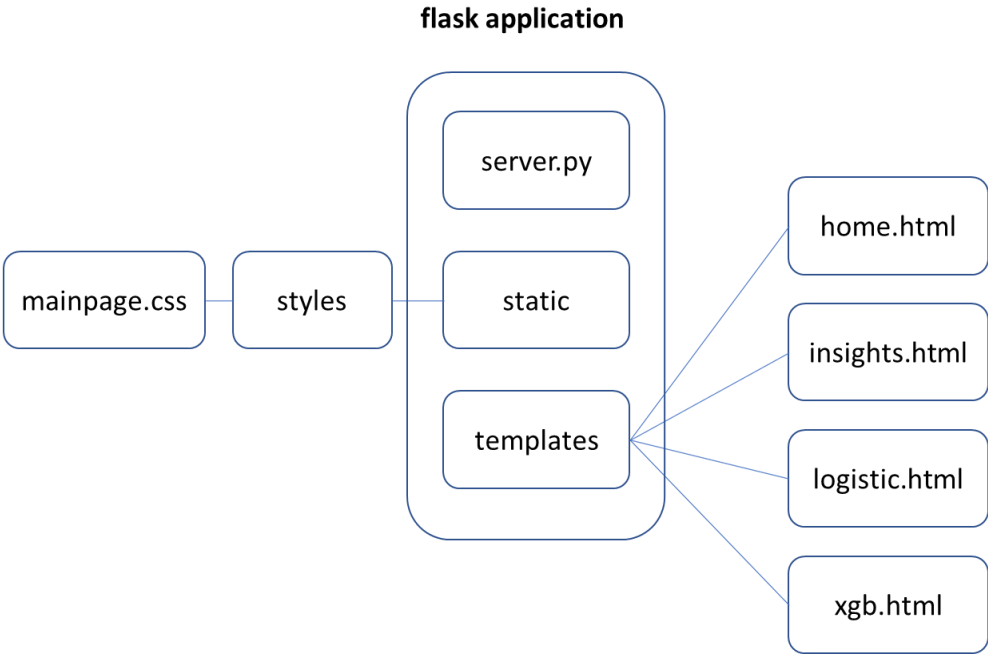
The best recall score given by the XGBoost estimator is 0.9. the result given by XGBoost is much higher than that given by logistic regression.

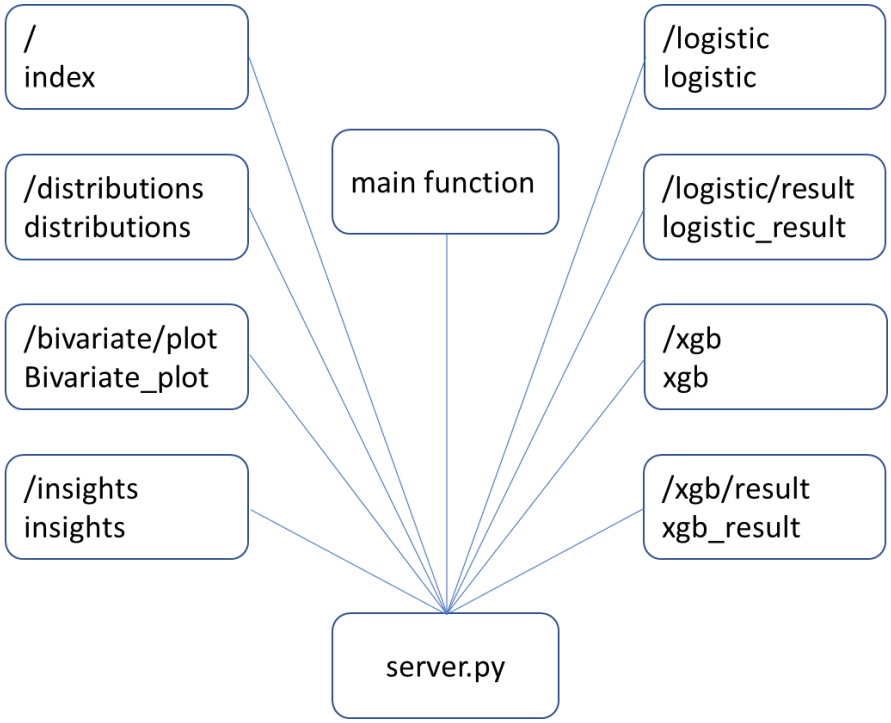
XGBoost is based on decision-tree, and is an ensembled machine learning algorithm that uses gradient boosting. It performs well in both efficiency, accuracy. But the drawback is that it might not be that interoperable. Therefore, as a complement to the XGBoost algorithm, I used the easy and highly interoperable logistic regression as well.

Web API Interface

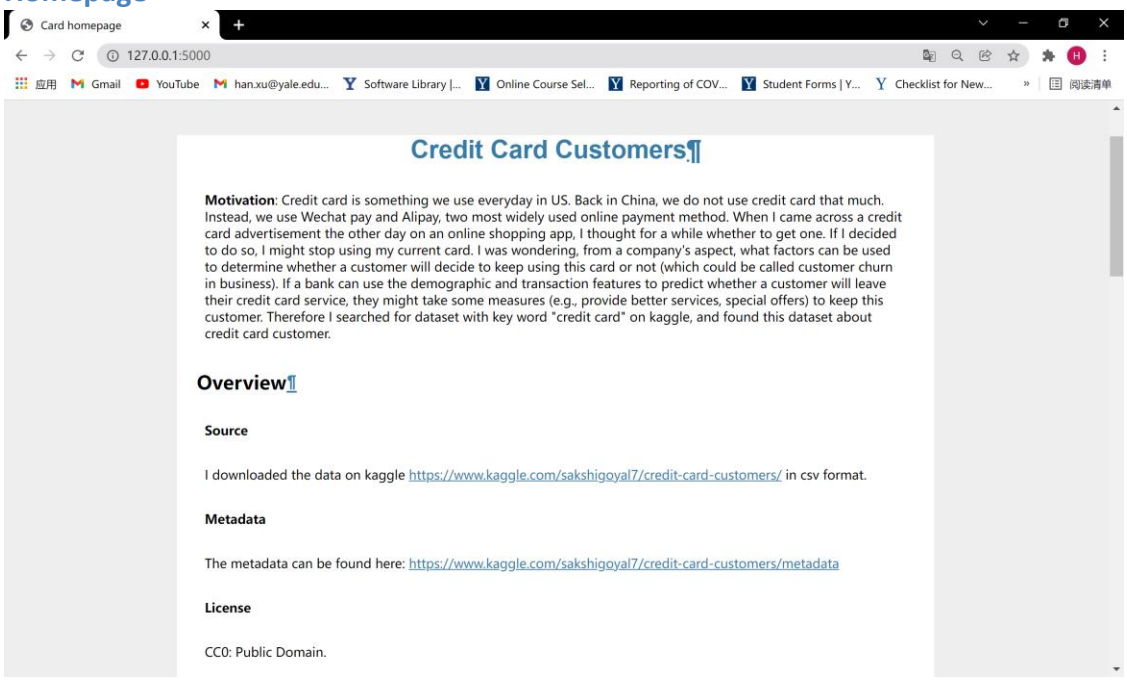
- I used flask to build a web app to show the interactive plots and designed buttons and multiple selection forms to get the variables the user selects.
- I used plotly to draw the plots, including distribution plots and bivariate scatter plots.
- For the model part, I used a textbox to enquire the parameters from the user (route "/logistic" and "/xgb"), and return the recall score as a result (route "/logistic/result" and route "xgb/result").
- For the logistic regression, the user can enter the C (the inverse of regularization strength), the parameter penalty (l1 or l2) and the number of folds for cross-validation; for the XGBoost, the user can enter learning rate, maximum depth, number of estimators and the number of folds for cross-validation.

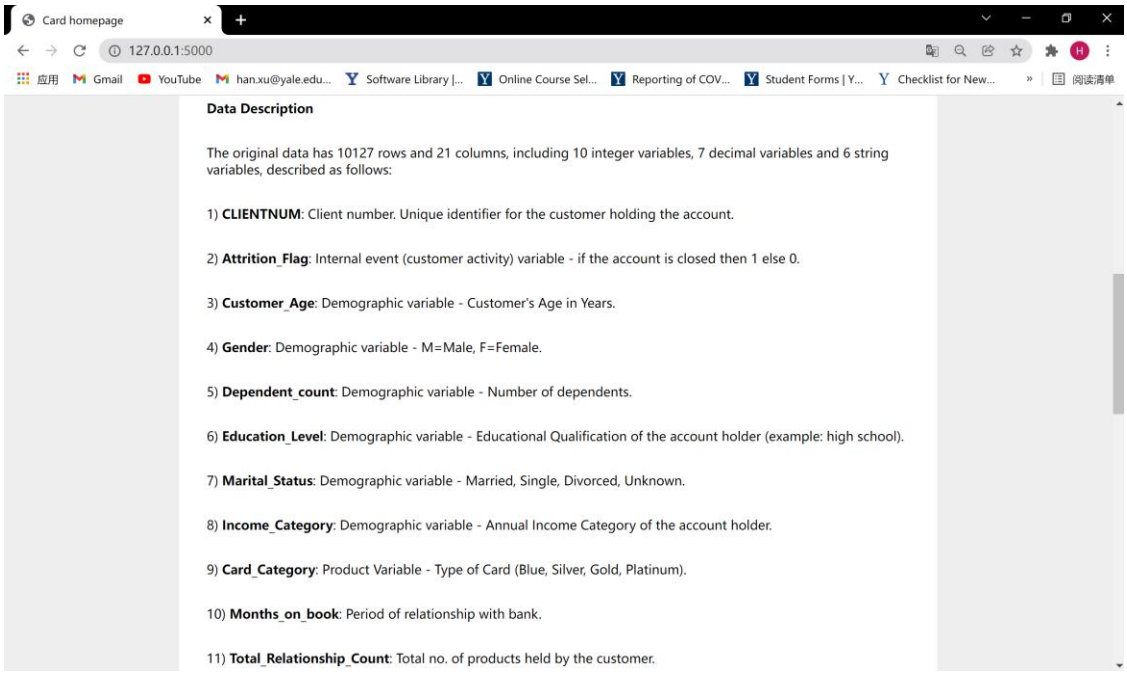
Files, routes and functions

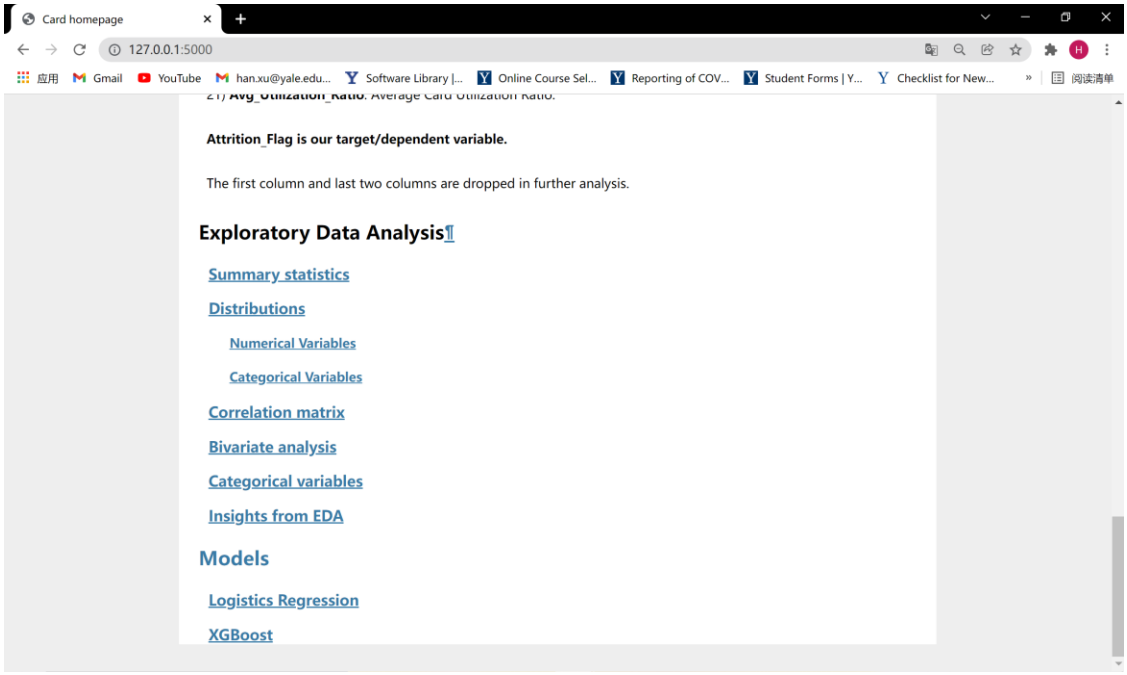




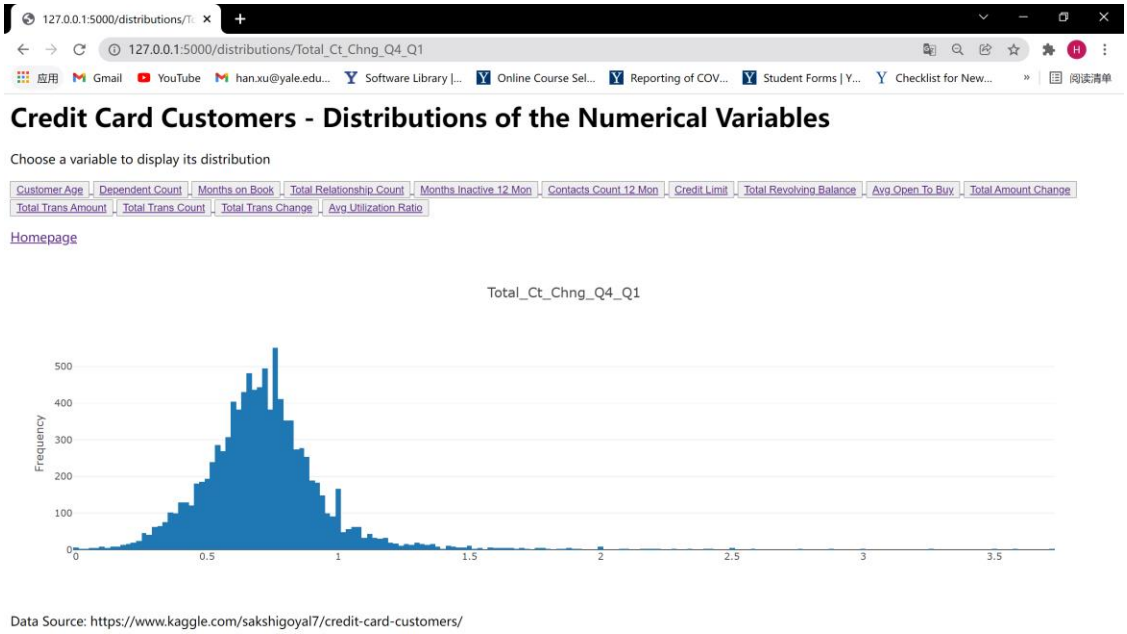
Homepage

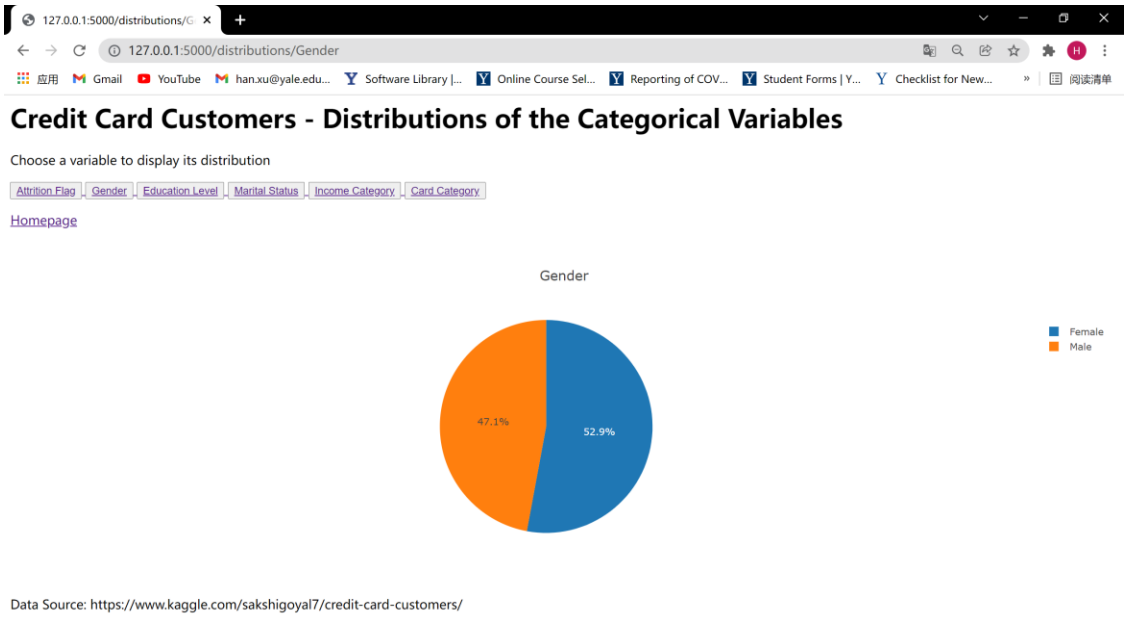

Card homepage
127.0.0.1:5000
Credit Card Customers¶
Motivation: Credit card is something we use everyday in US. Back in China, we do not use credit card that much. Instead, we use Wechat pay and Alipay, two most widely used online payment method. When I came across a credit card advertisement the other day on an online shopping app, I thought for a while whether to get one. If I decided to do so, I might stop using my current card. I was wondering, from a company's aspect, what factors can be used to determine whether a customer will decide to keep using this card or not (which could be called customer churn in business). If a bank can use the demographic and transaction features to predict whether a customer will leave their credit card service, they might take some measures (e.g., provide better services, special offers) to keep this customer. Therefore I searched for dataset with key word "credit card" on kaggle, and found this dataset about credit card customer.
Overview¶
Source
I downloaded the data on kaggle https://www.kaggle.com/sakshigoyal7/credit-card-customers/ in csv format.
Metadata
The metadata can be found here: https://www.kaggle.com/sakshigoyal7/credit-card-customers/metadata
License
CC0: Public Domain.


Card homepage
127.0.0.1:5000
Data Description
The original data has 10127 rows and 21 columns, including 10 integer variables, 7 decimal variables and 6 string variables, described as follows:
1) CLIENTNUM: Client number. Unique identifier for the customer holding the account.
2) Attrition_Flag: Internal event (customer activity) variable - if the account is closed then 1 else 0.
3) Customer_Age: Demographic variable - Customer's Age in Years.
4) Gender: Demographic variable - M=Male, F=Female.
5) Dependent_count: Demographic variable - Number of dependents.
6) Education_Level: Demographic variable - Educational Qualification of the account holder (example: high school).
7) Marital_Status: Demographic variable - Married, Single, Divorced, Unknown.
8) Income_Category: Demographic variable - Annual Income Category of the account holder.
9) Card_Category: Product Variable - Type of Card (Blue, Silver, Gold, Platinum).
10) Months_on_book: Period of relationship with bank.
11) Total_Relationship_Count: Total no. of products held by the customer.

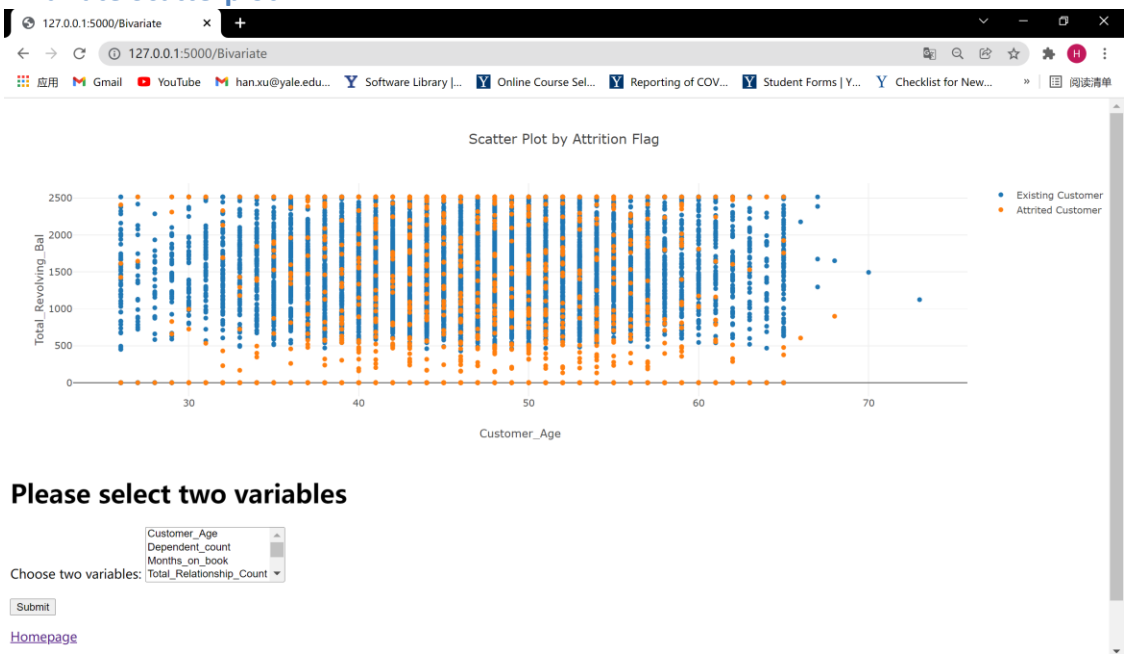


Distributions

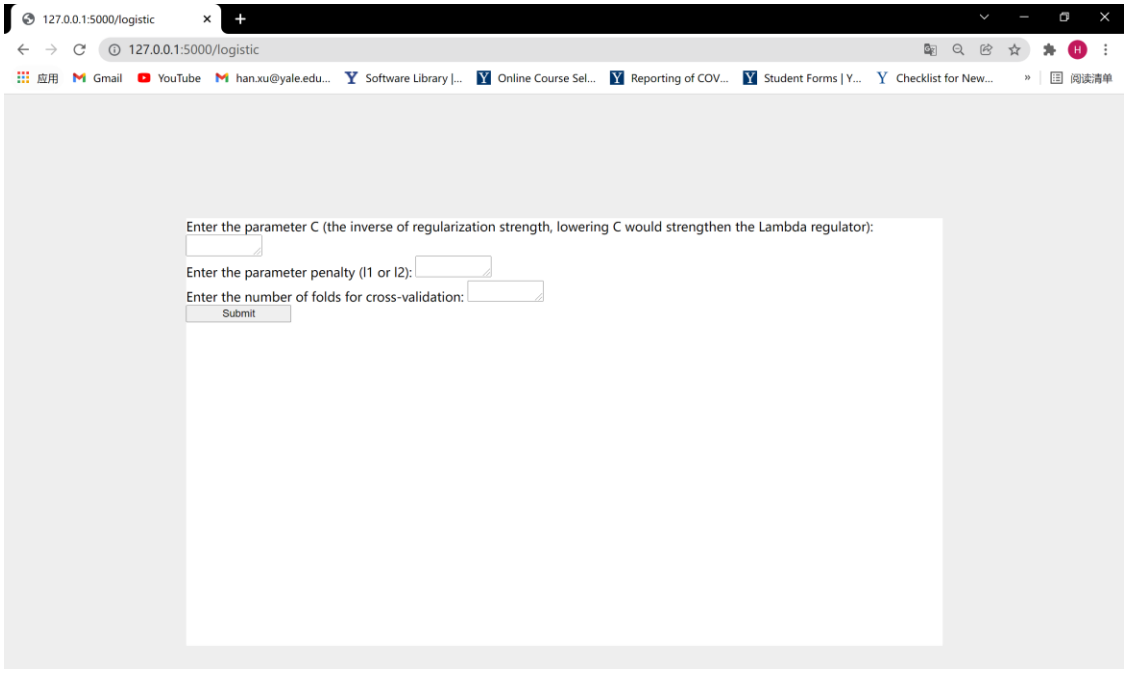


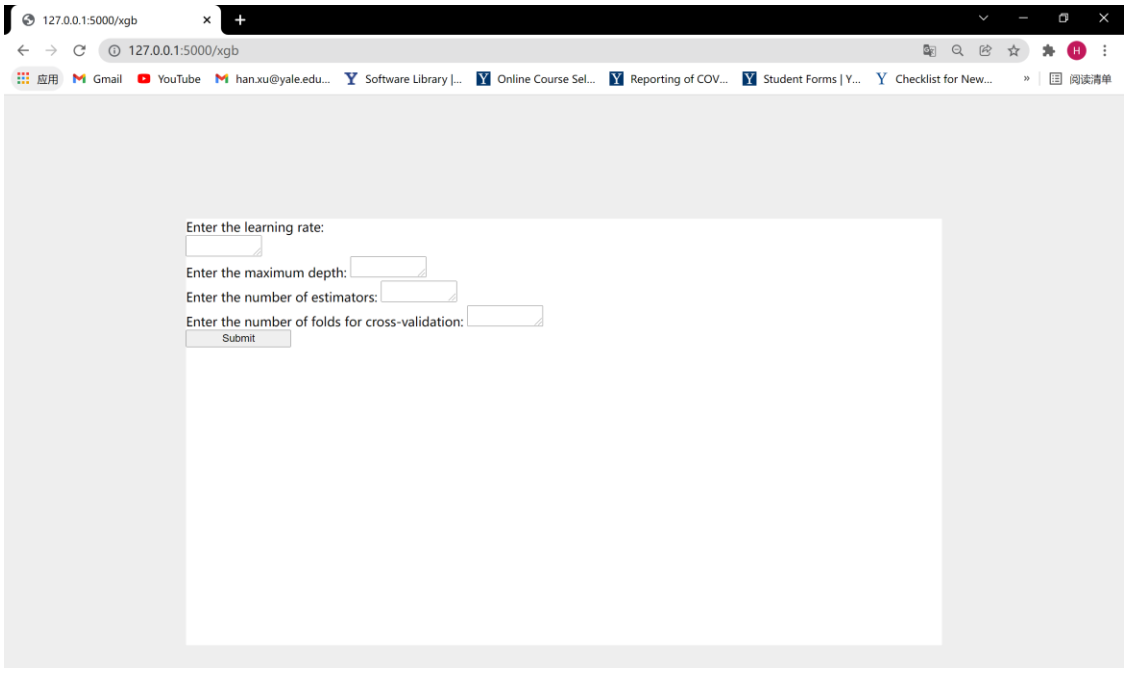


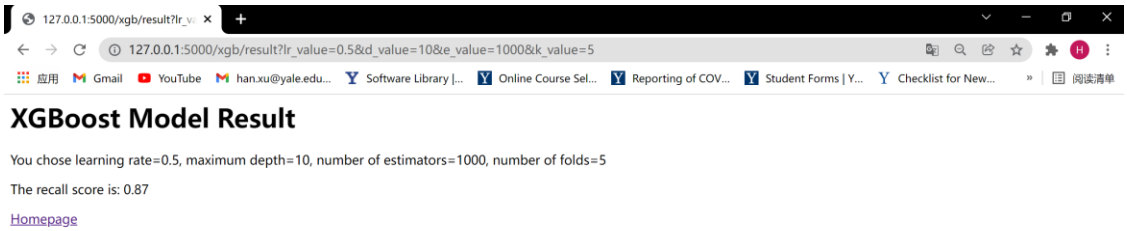
Bivariate Scatterplot



Models


127.0.0.1:5000/logistic
127.0.0.1:5000/logistic
Gmail
YouTube
Software Library |...
Online Course Sel...
Reporting of COV...
Student Forms | Y...
Checklist for New...
Enter the parameter C (the inverse of regularization strength, lowering C would strengthen the Lambda regulator):
Enter the parameter penalty (l1 or l2):
Enter the number of folds for cross-validation:
Submit


127.0.0.1:5000/xgb
127.0.0.1:5000/xgb
Gmail
YouTube
Software Library |...
Online Course Sel...
Reporting of COV...
Student Forms | Y...
Checklist for New...
Enter the learning rate:
Enter the maximum depth:
Enter the number of estimators:
Enter the number of folds for cross-validation:
Submit



Difficulties

When I was working on the distribution plot, I set a new route for every different variable and plot. But when I moved on to the bivariate plot, I realized that it would be impossible to write a new route and function for every two variables, it would be too much work since we have 20 variables in total. Then I thought that maybe I could use a multiple selection form to acquire two variables that the user is interested in, and show the scatter plot of those two variables.

Then I went into trouble getting multiple values as a result from the form. I first tried to add "[]" at the end of the variable name, but it didn't work. I checked for a lot of stack overflow and then found a solution of using "request.form.getlist('vars')". But the console kept yelling "IndexError: list index out of range", suggesting that the list was empty, and something must went wrong when passing the variables. It took me a long time to figure out that it was the request method that went wrong. We should use 'POST' method instead of 'GET' when passing values using a multiple selection form.

Conclusions and Recommendations

The best model performance is given by XGBoost, which gave a recall score of 0.9. We can apply the trained model on any new data to predict whether a customer will attrite or not. Total Transaction Count is most negatively correlated with attrition flag, and the coefficient given by the logistic regression model is -1.82. If a customer has little activities in the account for the last 12 months, the bank should consider providing some promotions to keep this customer. The most stable customers are mid-income, married, and with lower education level, which the bank could focus more on absorbing, rather than keeping. In addition, the bank provide better service for customers with platinum card type, since their attrition rate is significantly higher than customers with other card types, and the total number of platinum customers is much smaller.