TOPIC: CREDIT CARD APPROVAL PREDICTION

Author: Do Thi Thanh Hue
ID: 11192149

I. INTRODUCTION

1. Background of the study

Credit risk is the probability that a borrower will not pay back a loan in accordance with the terms of the credit agreement. The scorecard model is a model class used in many areas such as finance, economics, and social management. Credit scorecards are a common method of risk management in the financial industry. It uses individual information or organization's profile submitted by credit card applicants to predict the likelihood of future events such as default, loan or violation of law. Based on the credit score, the financial institution or government can decide whether or not to issue a credit card to the clients.

Credit scorecards are generally based on historical data. When a big economic fluctuation occurs. Past models can lose their original predictive power. Also, manually analyzing is very error-prone and time-consuming. Currently, with the development of machine learning algorithms, in this project, we will build an automatic prediction engine for credit card approval using machine learning technology.

2. Objective of the study

The purpose of the study is to create a model that predicts whether an applicant is a "good" or "bad" customer based on the dataset of characteristics of the customer.

3. Significance of the study

Credit risk is one of the problems faced by all commercial banks. Credit risk causes banks to lose capital and lose their reputation. Therefore, credit risk management to avoid and limit credit risk is an inevitable issue that all banks must pay attention to. Proper credit risk management not only keeps a bank's business operations safe, increases business profits, improves the bank's reputation and the quality of its service, but also helps ensure the stability and development of the economy as a whole.

II. THEORETICAL BACKGROUND

In the project, I used these models: Logistic Regression, Decision Tree Classification, Random Forest Classification, Support Vector Machine Classification and XGBoost Classification.

1. Logistic Regression

Logistic regression is a simple but very effective algorithm for classification problems. Logistic regression is a method of statistical analysis used to predict data values based on previous observations of a dataset.

The purpose of logistic regression is to estimate the likelihood of an event, including determining the relationships between features used to predict the probability of outcome. So for logistic regression:
Input: Input data (assuming there are two labels, 0 and 1).
Output: The probability that the input data will be labeled 0 or 1.

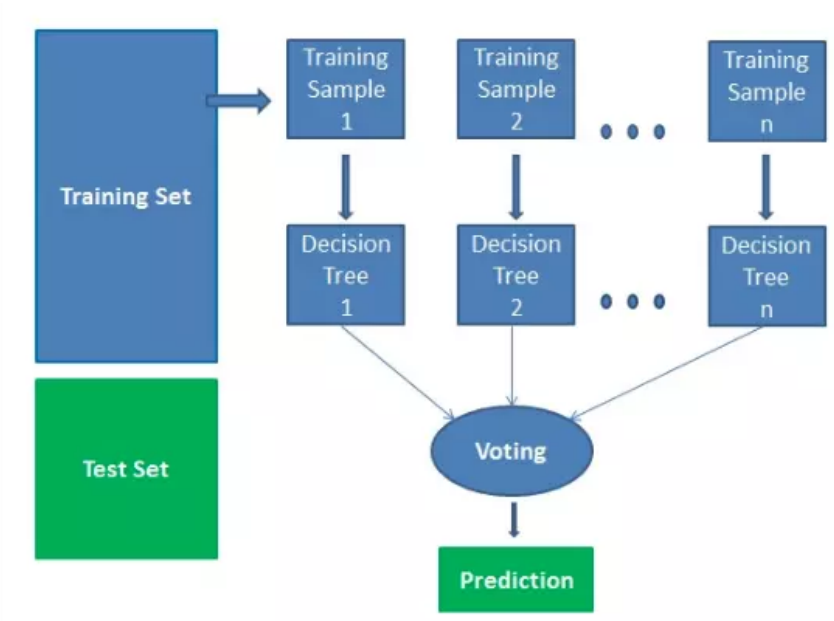
2. **Decision Tree Classification**

Decision trees are one of the most popular machine learning algorithms nowadays. A decision tree is a structured hierarchical tree used to classify objects based on a set of rules. Each node represents a function, each branch represents a rule, and each leaf represents a result (a specific value or a continuation branch). Object properties can be of various data types: Binary, Nominal, Ordinal, Quantitative while the subclass properties must be Binary or Ordinal.

Decision tree classification is a tree model in which a target variable can take a discrete set of values. In these tree structures, leaves represent class labels and branches represent the conjunctions of features that lead to these class conjunctions.

3. **Random Forest Classification**

Random forest is a supervised learning algorithm. It can be used for both regression and classification. It is also the most flexible and user-friendly algorithm. Random forest generates decision trees based on randomly selected data samples, is predicted from each tree, and chooses the best solution by voting. It also provides a great indicator of the selection of the best solution by voting. Random forests have many uses such as recommended engine, image classification, feature selection. It can be utilized to categorize loyal loan applicants, identify fraudulent activity, and predict diseases. This is on the basis of the Boruta algorithm, which selects important features in the dataset. Random forests are considered an accurate and powerful method due to the large number of decision trees involved in this process.



4. XGBoost Classification

XGBoost is a decision tree-based machine learning algorithm that uses a process called boosting to increase performance. Since its inception, XGBoost has become one of the most effective machine learning algorithms, regularly providing better results than most other algorithms. This is a common supervised learning algorithm used for classification and regression on large datasets. To avoid overfitting, it uses sequentially-built shallow decision trees to bring out accurate results and a highly scalable training method.

Big data is not a problem for XGboost because it is fast training, can be scaled for parallel computing on multiple servers, and can be accelerated using the GPU. It also includes a mechanism to automatically handle missing values. XGBoost also has a tree-pruning feature and allows users to use their evaluation criteria.

III. DATA

1. Overview

"application_record.csv " and "credit_record.csv" are the 2 files used in the assignment.

a. Dataset 'application_record.csv' (df1)

Dataset has 438557 rows and 18 columns with 348472 duplicate values.

- ID: Customer ID
- CODE_GENDER: Gender of the applicant. M is male and F is female
- FLAG_OWN_CAR: Is an applicant with a car? Y is Yes and N is No
- FLAG_OWN_REALTY: Is an applicant with realty. Y is Yes and N is No
- CNT_CHILDREN: The number of children
- AMT_INCOME_TOTAL: The total income
- NAME_INCOME_TYPE: Income category (5)
- NAME_EDUCATION_TYPE: Education level (5)
- NAME_FAMILY_STATUS: The marital status (6)
- NAME_HOUSING_TYPE: The type of house (6)
- DAYS_BIRTH: The number of the days from birth (Negative values)
- DAYS_EMPLOYED: The start day of employment (Negative values)
- FLAG_MOBIL: Is an applicant with a mobile. 1 is True and 0 is False.
- FLAG_WORK_PHONE: Is an applicant with a work phone. 1 is True and 0 is False.
- FLAG_PHONE: Is an applicant with a phone. 1 is True and 0 is False.
- FLAG_EMAIL: Is an applicant with an email. 1 is True and 0 is False.
- OCCUPATION_TYPE: The type of job (19). This column has missing values.
- CNT_FAM_MEMBERS: The number of members in family

b. Dataset 'credit_record.csv' (df2)

It includes 1048575 rows and 3 columns.

- ID: Customer ID
- MONTHS_BALANCE: The number of months from record time

- STATUS: Credit status (8)
  - 0: 1-29 days overdue
  - 1: 30-59 days overdue
  - 2: 60-89 days overdue
  - 3: 90-119 days overdue
  - 4: 120-149 days overdue
  - 5: Overdue or bad debts, write-offs for more than 150 days
  - X: No loan for the month
  - C: Paid off that month

2. Data preprocessing

2.1 Duplicate Value

Check duplicate values at df1 & df2.

```
[ ]  df1.iloc[:,1:].duplicated().sum()

     348472
```

```
[ ]  df2.duplicated().sum()

     0
```

We can see that df1 has 348472 duplicate values => Drop them.

```
[ ]  #Drop duplicates
     df1 = df1.drop_duplicates(subset=df1.columns[1:], keep='first')
```

```
[ ]  df1.shape

     (90085, 18)
```

2.2 Remove the constance feature

It's easy to see that every applicant has a mobile. So I dropped the columns 'FLAG_MOBIL' at df1.

2.3 Handle missing value

Missing values can affect accuracy of the model so we need to check and eliminate them by filling or dropping the missing values out of the dataset.

By using the function 'isnull().sum()', I found that df1 has 27477 missing values at column 'OCCUPATION_TYPE' while df2 has no. The number of missing values is too much so in this case, I decided to fill them with 'Other'.

```
#Fill missing values
df1['OCCUPATION_TYPE'].fillna(value='Other', inplace=True)
```

2.4 Final dataset

Some data columns are difficult to understand, so we'll process them by creating a new variable and then deleting the old data columns out of the dataset.

ACCOUNT_LENGTH

```
#Extract how many months account has been open for
month_df = pd.DataFrame(df2.groupby(['ID'])['MONTHS_BALANCE'].agg(min)).reset_index()
month_df.rename(columns={'MONTHS_BALANCE':'ACCOUNT_LENGTH'}, inplace=True)

# Make entries positive
month_df['ACCOUNT_LENGTH']=-month_df['ACCOUNT_LENGTH']

# Merge dataframes on ID
new_df = pd.merge(df1, month_df, how='inner', on=['ID'])
new_df
```

AGE_YEARS

```
new_df['AGE_YEARS']=-new_df['DAYS_BIRTH']//365
new_df.drop('DAYS_BIRTH', axis=1, inplace=True)
```

UNEMPLOYED

```
new_df['UNEMPLOYED'] = new_df['DAYS_EMPLOYED'].apply(lambda x : 1 if x>0 else 0)
new_df.drop('DAYS_EMPLOYED', axis=1, inplace=True)
```

To identify credit users into 2 groups: 'LOW RISK' & 'HIGH RISK', we do this in the 'STATUS' column. Because there must be 3% of users at risk, we set a customer as 'HIGH RISK' if during any month they are late on payments by 60 days or more, and 'LOW RISK' otherwise. We set 'HIGH RISK' as 1 and 'LOW RISK' as 0.
=> Replacing C, X, 0, 1 with 0 and the other value: 2, 3, 4, 5 with 1.

TARGET

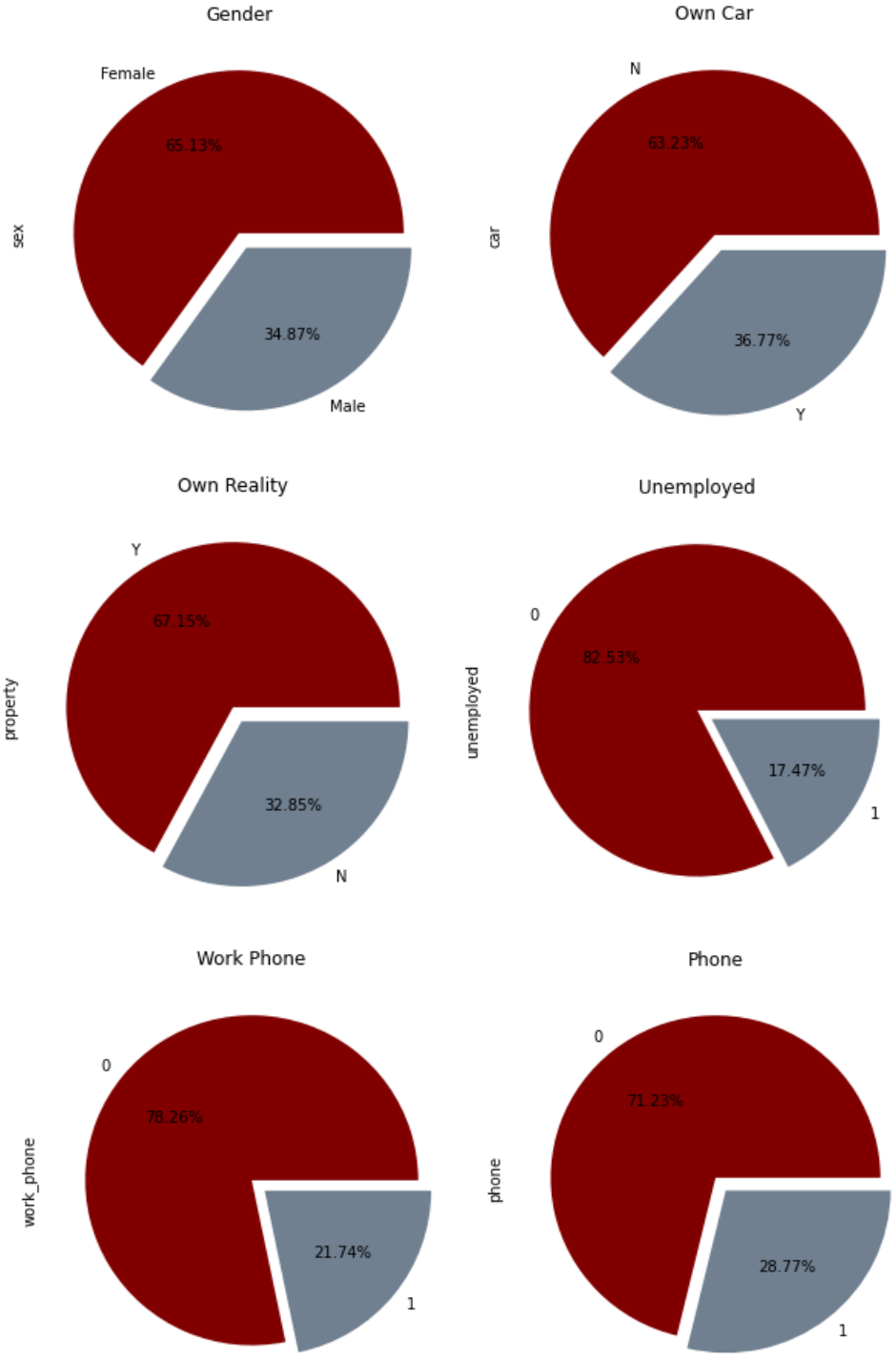
```
df2['TARGET'] = df2['STATUS']
df2['TARGET'].replace('X', 0, inplace=True)
df2['TARGET'].replace('C', 0, inplace=True)
df2['TARGET'] = df2['TARGET'].astype(int)
df2['TARGET']=  df2['TARGET'].apply(lambda x : 1 if x>=2 else 0)
```

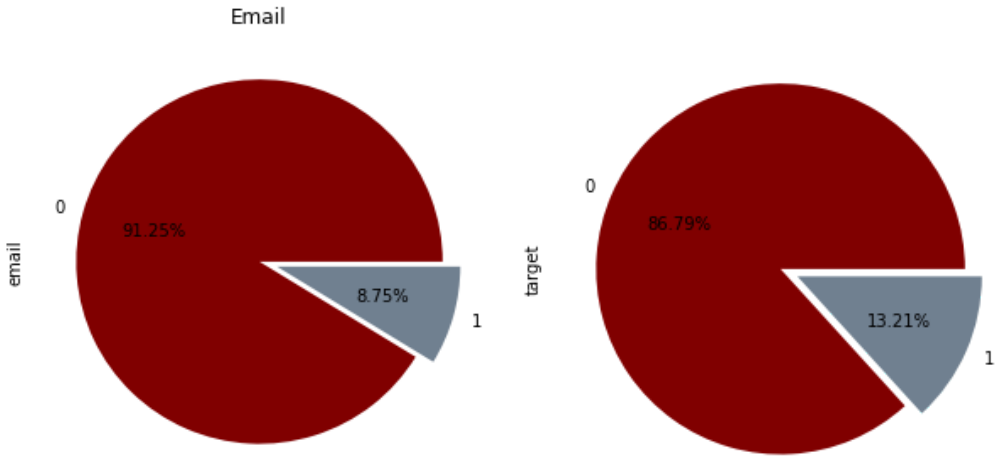
After that, I combine them to get a complete dataset. The new dataset (final_df) has 19 columns and 9709 rows.

2.5 Visualization

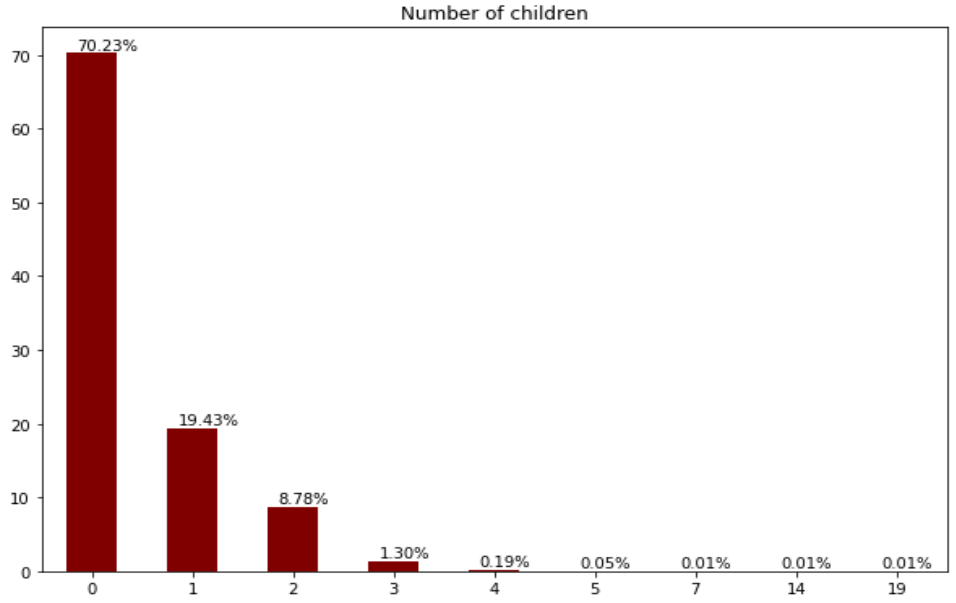
To help everyone observe and understand the data easily, the below is a graph showing the percentage of values for each variable. The results are shown on the figure for quicker visualization. The report uses three types of graphs: Pie chart, bar chart and distribution graphs.

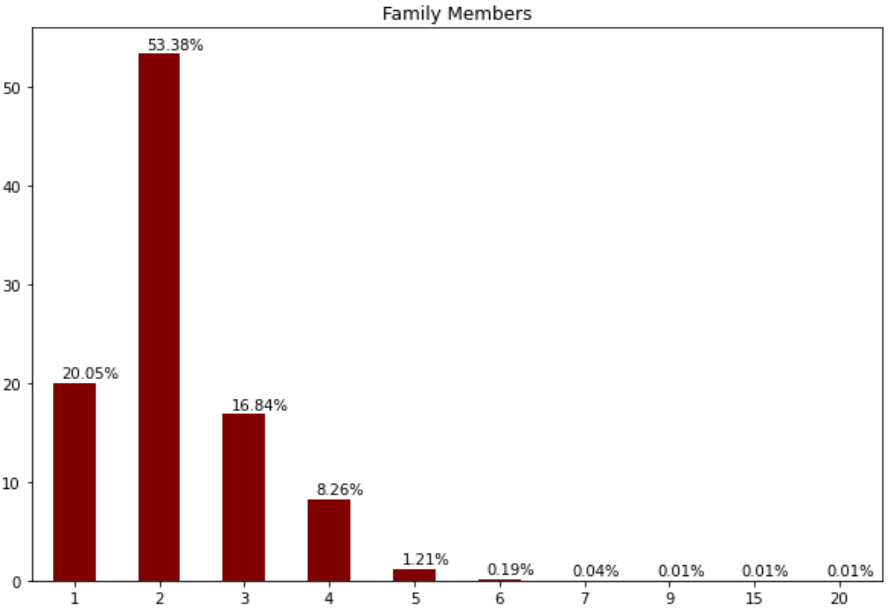
a. Pie chart

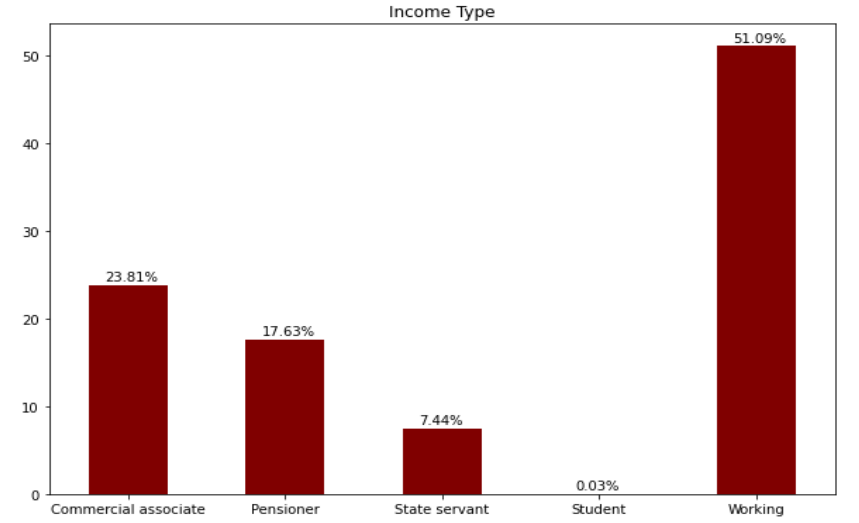


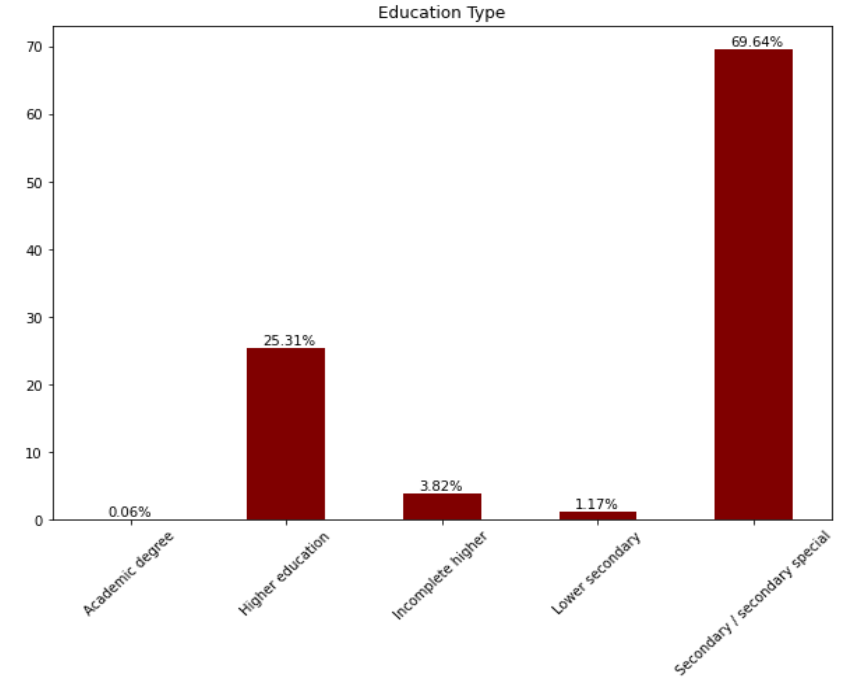

Email
email
0
91.25%
8.75%
1
target
0
86.79%
13.21%
1

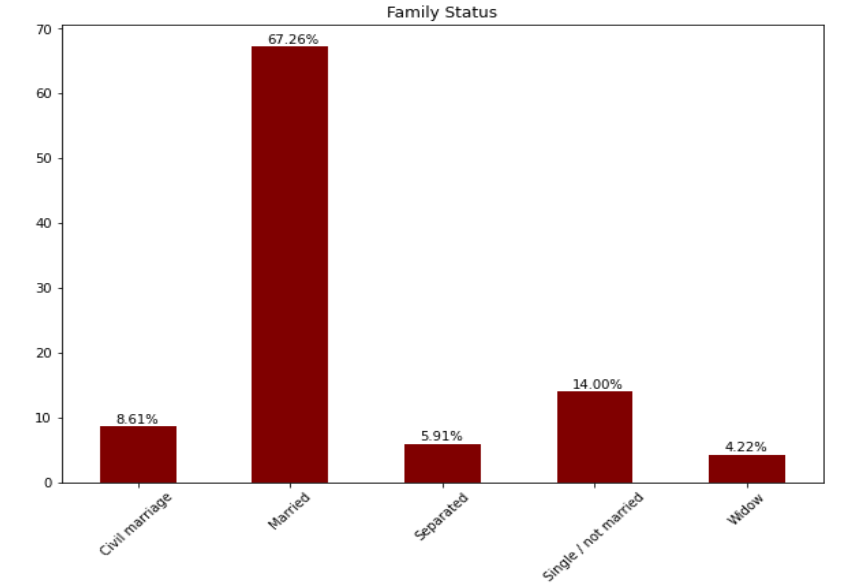
b. Bar graph

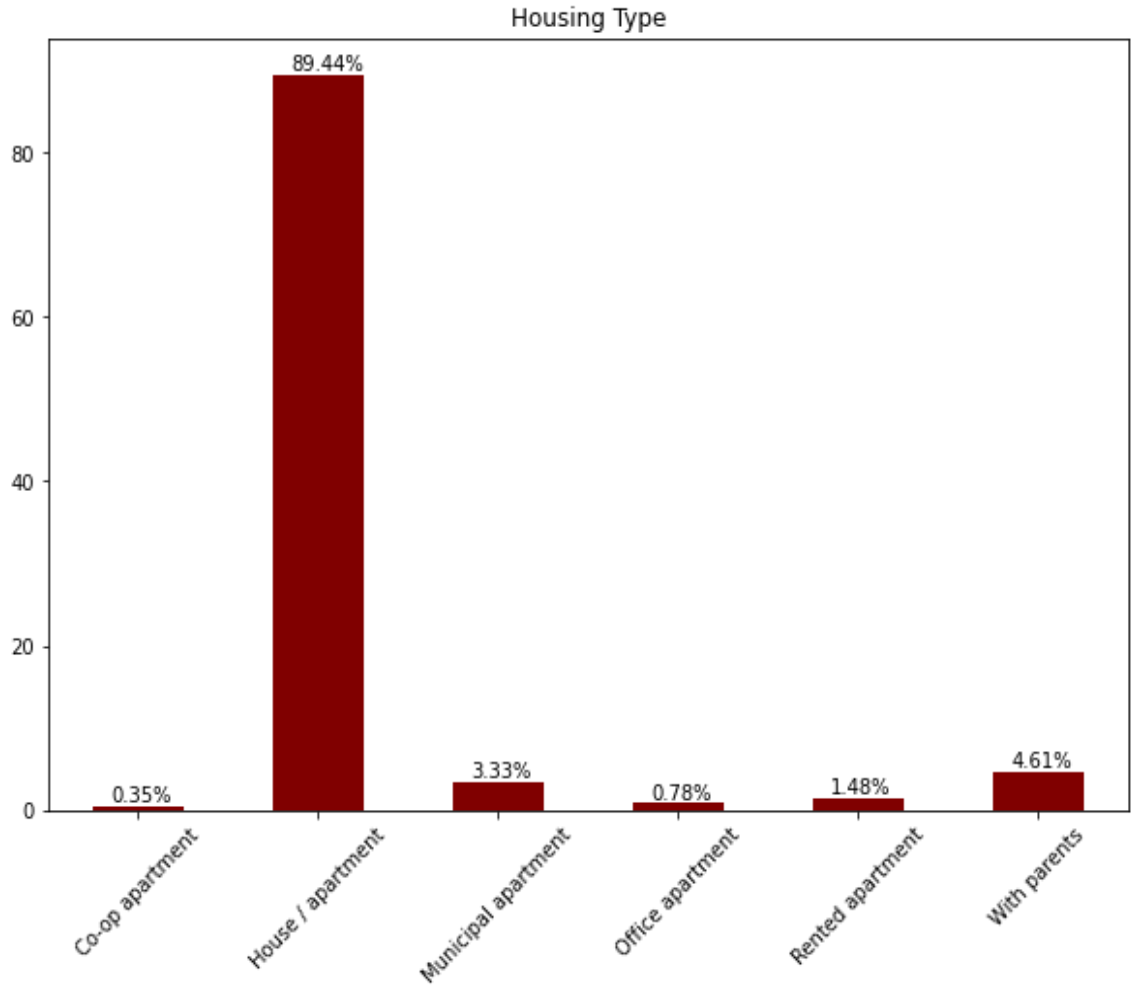

Number of children
70.23%
19.43%
8.78%
1.30%
0.19%
0.05%
0.01%
0.01%
0.01%
0
1
2
3
4
5
7
14
19

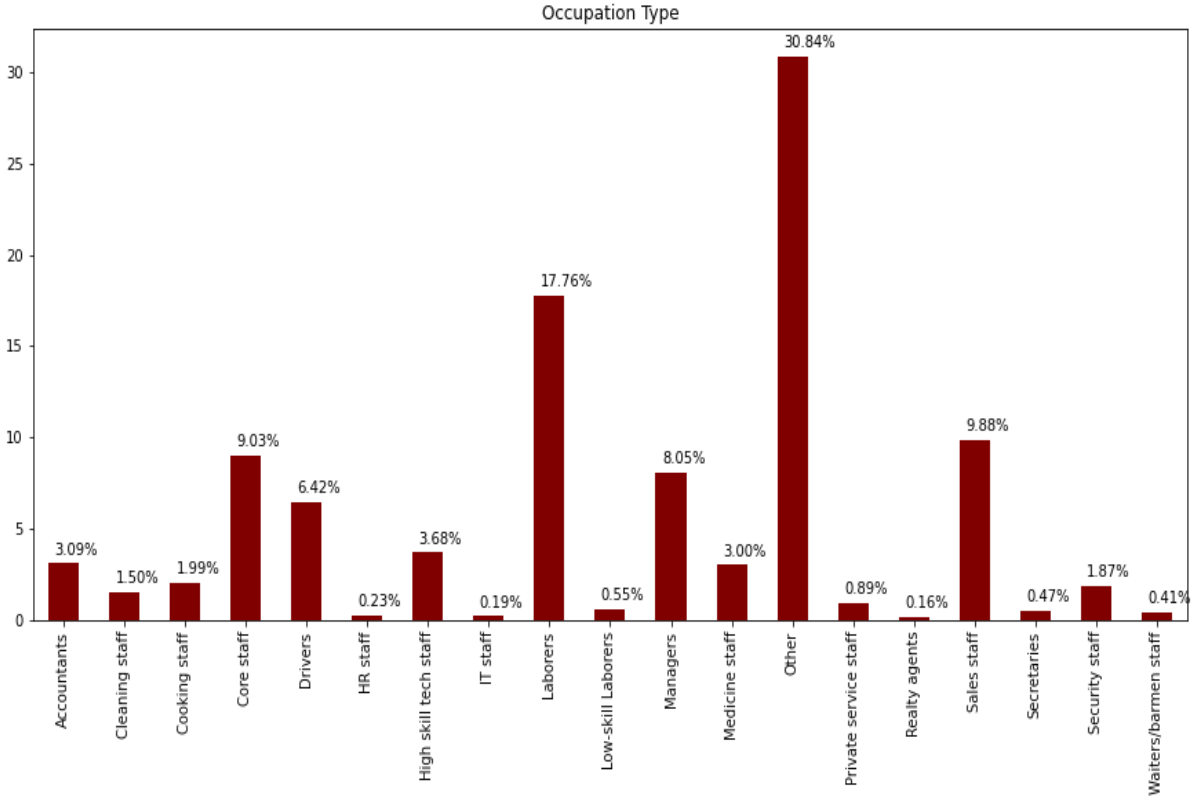

Family Members
53.38%
20.05%
16.84%
8.26%
1.21%
0.19%
0.04%
0.01%
0.01%
0.01%
1
2
3
4
5
6
7
9
15
20


Income Type
51.09%
23.81%
17.63%
7.44%
0.03%
0
10
20
30
40
50
Commercial associate
Pensioner
State servant
Student
Working

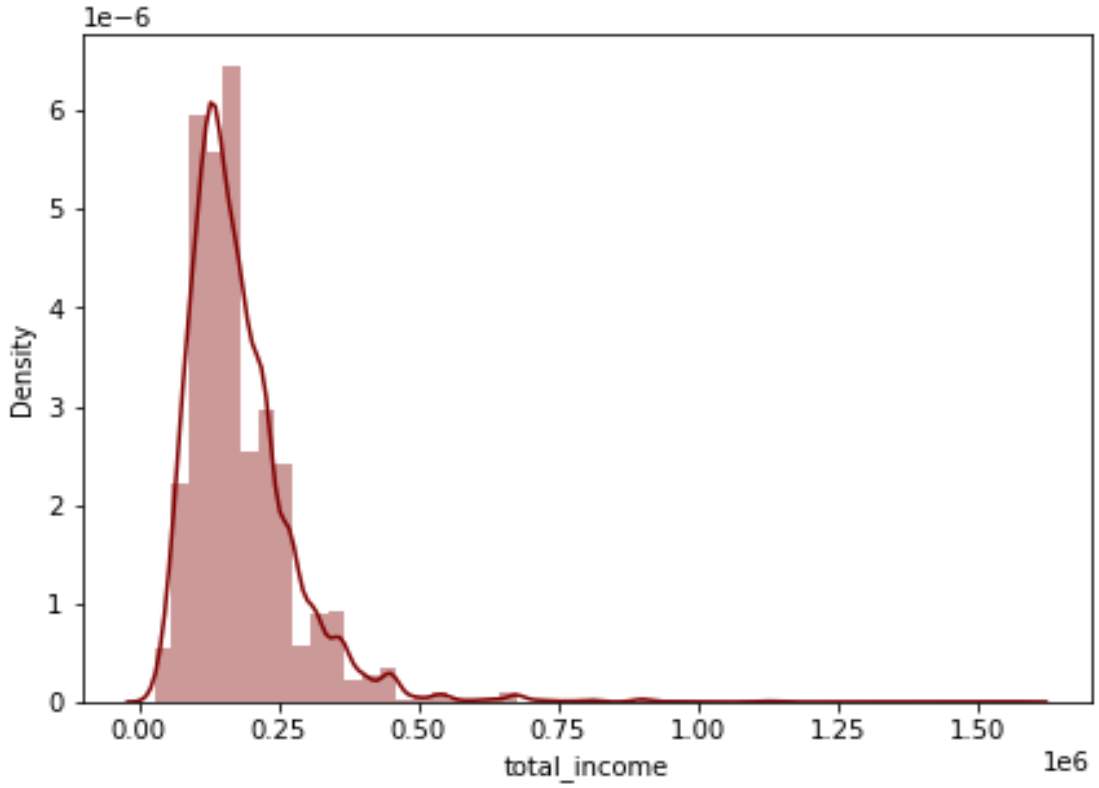

Education Type
69.64%
25.31%
3.82%
1.17%
0.06%
0
10
20
30
40
50
60
70
Academic degree
Higher education
Incomplete higher
Lower secondary
Secondary / secondary special

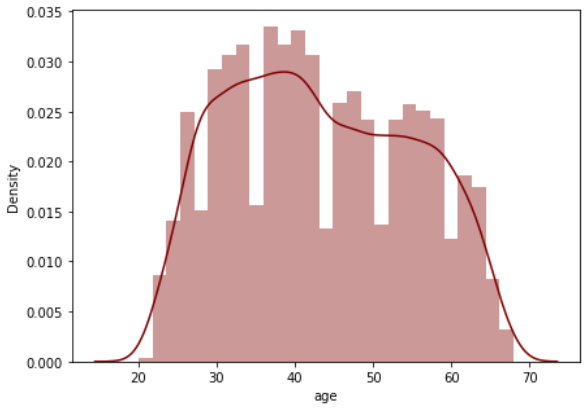

Family Status
67.26%
14.00%
8.61%
5.91%
4.22%
0
10
20
30
40
50
60
70
Civil marriage
Married
Separated
Single / not married
Widow

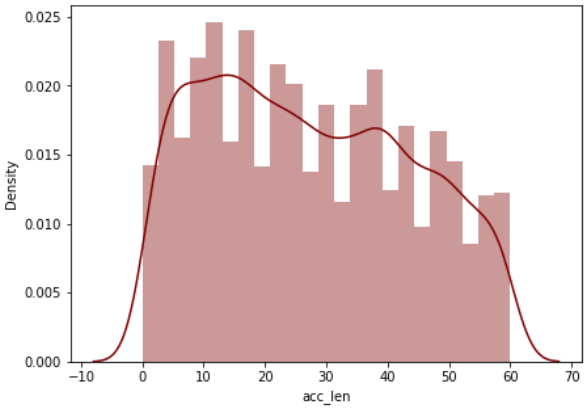

Housing Type
89.44%
80
60
40
20
0
0.35%
3.33%
0.78%
1.48%
4.61%
Co-op apartment
House / apartment
Municipal apartment
Office apartment
Rented apartment
With parents


Occupation Type
30.84%
25
20
15
10
5
0
17.76%
9.88%
9.03%
8.05%
6.42%
3.68%
3.09%
3.00%
1.99%
1.87%
1.50%
0.89%
0.55%
0.47%
0.41%
0.23%
0.19%
0.16%
Accountants
Cleaning staff
Cooking staff
Core staff
Drivers
HR staff
High skill tech staff
IT staff
Laborers
Low-skill Laborers
Managers
Medicine staff
Other
Private service staff
Realty agents
Sales staff
Secretaries
Security staff
Waiters/barmen staff

c. Displot



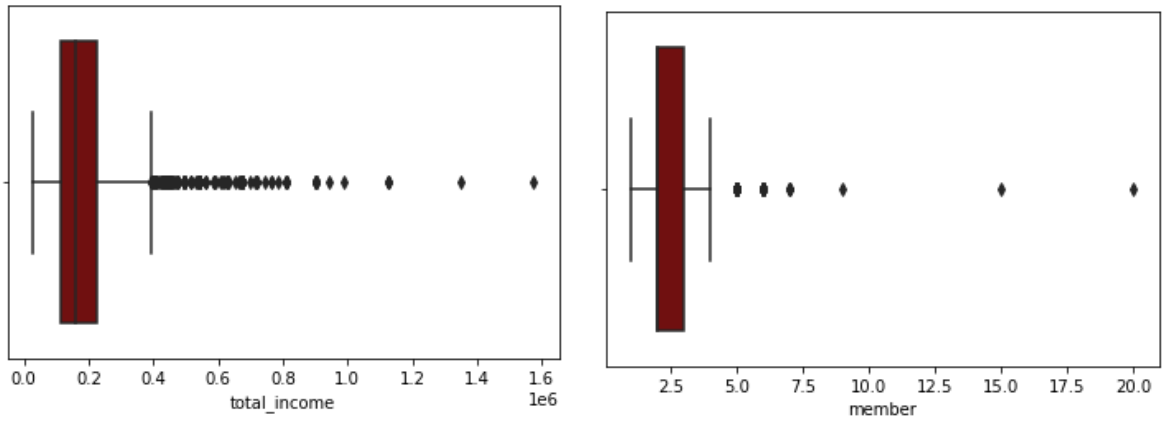




2.6 Handle outlier

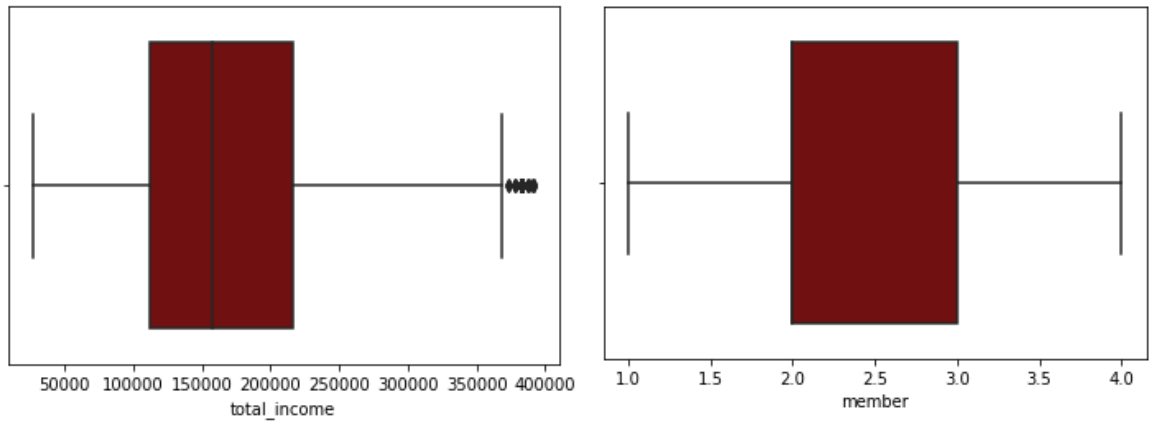
Outliers are data points that are significantly different from others, probably due to measurement variability or experimental error. ML algorithms specially pay attention to the range and distribution of attribute values. Outliers in the data can misle the training process, less accurate models, time-consuming and poorer results.

Therefore, I'll look for outliers in the dataset and modify them accordingly. To do this, I used the describe() method to calculate descriptive statistics. It not only will help you identify outliers, but also give you a better understanding of how your data is distributed. I'll also be using the boxplot matrix to visualize and detect outliers.



Here, I use def remove_outlier(df, col_name) function to eliminate outliers. After processing, there are 9257 rows in my dataset.

```
def remove_outlier(df, col_name):
  Q1 = df[col_name].quantile(0.25)
  Q3 = df[col_name].quantile(0.75)
  IQR = Q3 - Q1
  fence_low = Q1 - 1.5*IQR
  fence_high = Q3 + 1.5*IQR
  df = df.loc[(df[col_name] > fence_low) & (df[col_name] < fence_high)]
  return df
```



3. Model

3.1 Feature selection by WOE, IV

a. WOE

WOE (weight of evidence) is one of the feature engineering and feature selection techniques commonly applied in the scorecard model. This method will rank variables into strong, medium, weak, no impact, etc. based on the ability and power to predict bad debt, classify good and bad customers. The ranking criterion is the IV (information value) calculated from the WOE method. The model also generates feature values for each variable. This value measures the difference in distribution between good and bad. WOE method has different processing techniques for continuous and categorical variables.

$$WOE = ln(\frac{\%Good}{\%Bad})$$

- %Goods: Distribution of Good Customers across all bins. The sum of the column is 1.
- %Bads: Distribution of Bad Customers across all bins. The sum of column is 1.

b. IV

IV is one of the most useful techniques to select important variables in a predictive model. It helps to rank variables on their importance.

$$IV = \sum_{i=1}^{n} (\%Good_i - \%Bad_i) * WOE$$

Since WOE and IV are covariate, we can see that IV always takes a positive value. The IV value indicates how much the difference between%Good and %Bad for each bin is. If IV is high, the difference between them is large and the variables are useful for classifying records, but if the IV is small, the variables are less useful for classification problem.

Information Value	Variable Predictiveness
Less than 0.02	Not useful for prediction
0.02 to 0.1	Weak predictive Power
0.1 to 0.3	Medium predictive Power
0.3 to 0.5	Strong predictive Power
>0.5	Suspicious Predictive Power

c. Feature selection

In my project, the result in Table IV shows that there are 10 useful variables for classifying records. => Deleting the variables has no effect: email, edu_type, member, work_phone, children, car and phone because its value is less than 0.02.

variable	IV
acc_len	0.337625
income_type	0.13484
fam_stt	0.116193
job	0.093163
age	0.046388
unemployed	0.044534
property	0.038333
house_type	0.034365
sex	0.024623
total_income	0.023864
email	0.017969
edu_type	0.011877
member	0.008533
work_phone	0.003623
children	0.001035
car	0.000834
phone	0.000213

3.2 Feature encoding

Before starting model training, we need to make sure all variables are numeric because some models cannot work with label data. We can easily do this using the encoding methods: Numeric Encoding and One-hot Encoding.

3.3 Splitting the data

Now we need to split the dataset into a train and test split to start building some models.

```
X = final_df.drop(columns = 'target')
y = final_df['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

3.4 Standardize data

As with data cleaning, data standardization is also a very important step in solving machine learning problems. Many machine learning algorithms assume that the input data has a Gaussian distribution. Therefore, normalizing the data to a normal form of a Gaussian distribution with a mean of 0 and a variance equal to 1 will improve algorithms such as Linear Regression and Logistic Regression. So, in this section, I use the method `StandardScaler()`.

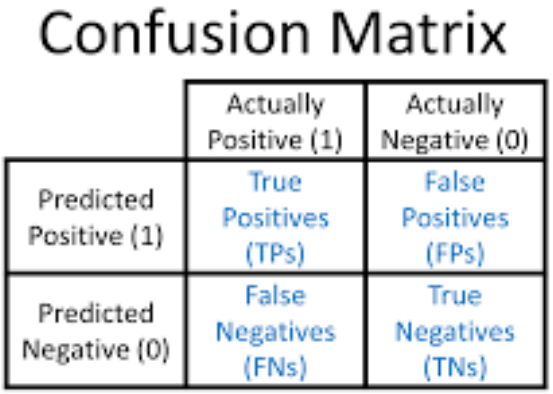
3.5 Handle imbalanced dataset

Data imbalance is one of the most common phenomena in binary classification problems such as email spam, fraud detection, and bankruptcy prediction. When severe data imbalances occur, using accuracy as a measure of model evaluation is often ineffective and is a random model that predicts everything. SMOTE (Synthetic Minority Oversampling) is a sampling method that increases the sample size of minority groups in the event of a sample imbalance.

3.6 Performance metrics

a. Confusion matrix

A table used to describe the performance of a classifier on a set of test data whose the true values are known.



b. Accuracy

The ratio of true positives and true negatives to all positive and negative observations.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{All Samples}}$$

c. Precision

The level of prediction accuracy in the predicted cases is Positive.

$$\text{Precision} = \frac{\text{True Positives}}{\text{Total } \textit{Predicted} \text{ Positives}}$$

d. Recall

The degree of predictive accuracy of cases is Positive in actual cases is Positive.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

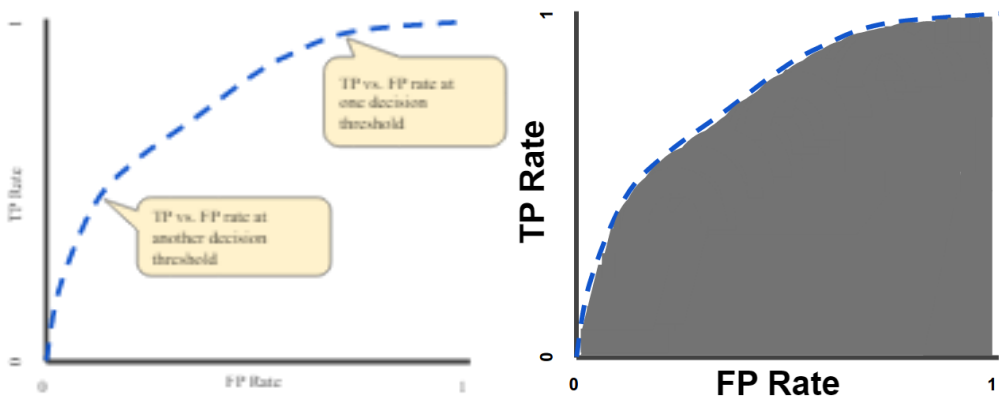
e. F1- score

The Harmonic Mean between Precision and Recall. This is an ideal surrogate metric for accuracy when the model has a high sample imbalance rate.

$$\text{F1 Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

f. ROC Curve

A graph showing the performance of a classification model at all classification thresholds.



g. AUC

AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).

IV. RESULT AND DISCUSSION

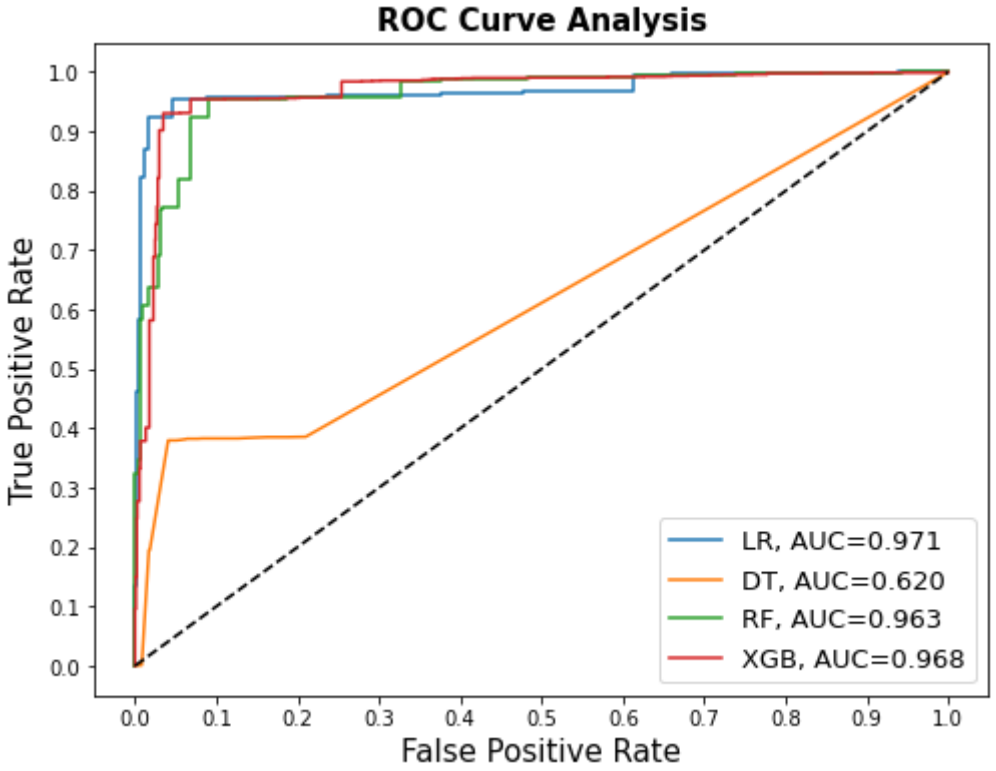
1. Result

In this section, I trained and tested 5 models: Logistic Regression, Decision Tree, Random Forest, Support Vector Machine and XGBoost. Below are the results of each model.

```
Logistic Model Accuracy :  0.9509641873278237

Confusion matrix :
[[1719   96]
 [  82 1733]]

Classification report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      1815
           1       0.95      0.95      0.95      1815

    accuracy                           0.95      3630
   macro avg       0.95      0.95      0.95      3630
weighted avg       0.95      0.95      0.95      3630

Decision Tree Model Accuracy :  0.6685950413223141

Confusion matrix :
[[1738   77]
 [1126  689]]

Classification report:
              precision    recall  f1-score   support

           0       0.61      0.96      0.74      1815
           1       0.90      0.38      0.53      1815

    accuracy                           0.67      3630
   macro avg       0.75      0.67      0.64      3630
weighted avg       0.75      0.67      0.64      3630

Random Forest Model Accuracy :  0.8044077134986226

Confusion matrix :
[[1763   52]
 [ 658 1157]]

Classification report:
              precision    recall  f1-score   support

           0       0.73      0.97      0.83      1815
           1       0.96      0.64      0.77      1815

    accuracy                           0.80      3630
   macro avg       0.84      0.80      0.80      3630
weighted avg       0.84      0.80      0.80      3630
```

```
XGBoost Model Accuracy :  0.9347107438016529

Confusion matrix :
[[1756   59]
 [ 178 1637]]

Classification report:
              precision    recall  f1-score   support

           0       0.91      0.97      0.94      1815
           1       0.97      0.90      0.93      1815

    accuracy                           0.93      3630
   macro avg       0.94      0.93      0.93      3630
weighted avg       0.94      0.93      0.93      3630
```

2. **Model comparison**



The AUC index measures the area under the ROC curve and indicates how the GOOD / BAD customers classification capacity is. The higher AUC value, the better model. We can see that Logistic Regression performs best with AUC = 0.971. The result is very high, showing that the predictive power of the model is good.

3. **Calculate the credit score**

Calculating the credit scorecard of each applicant by computing the score for each feature is the last step. It gives a hand to banks to be able to decide whether or not to lend to the clients. Scores will be scaled according to the following formula:

$$\text{Score} = (\beta.\,\text{WOE} + \frac{\alpha}{n}).\,\text{Factor} + \frac{\text{Offset}}{n}$$

Where:

βi — logistic regression coefficient for the variable Xi

α — logistic regression intercept

WoE — Weight of Evidence value for variable Xi

n — number of independent variable Xi in the model

Factor, Offset — known as scaling parameter, where

- Factor = pdo/ln(2)
- Offset = Target Score — (Factor × ln(Target Odds))

- Odds = Good/Bad

4. Conclusion & Discussion

In summary, I've analyzed, pre-processed my dataset, trained and evaluated 5 models to predict and classify Good or Bad customers. I used Accuracy, Recall, Precision, F1-score and ROC AUC to evaluate the model performances and get the Logistic Regression model performed best.

Credit scoring algorithm plays an important role in predicting and classifying problems of banks and finance. In theory, it's fairly easy to build a credit scorecard model because it's only based on logistic regression equations. However, the process of building a model requires many criteria such as interpretability and accuracy... in order to apply it in practice. In addition to its use in credit risk management, the credit scorecard model can be widened and applied in many other fields: customer ratings in marketing, business & financial institutions evaluation, national ratings... This model also has considerable potential in social management. It has been applied in many countries with positive effects.