COURSE ENDING REPORT ‘

CREDIT RISK MODEL IN R/PYTHON

2023

CREDIT RISK MODEL IN R/PYTHON

Contents

PART I: INTRODUCTION

1. OVERVIEW

Lahsasna et al. (2010) emphasized that credit risk decisions are key determinants for the success of financial institutions because of huge losses that result from wrong decisions. Poor evaluation of credit risk can cause money loss (Gouvea, 2007). Wu et al. (2010) stressed that credit risk assessment is the basis of credit risk management in commercial banks and provides the basis for loan decision-making.

Predicting customers' ability to repay loans is a critical task for financial institutions, as it enables them to manage credit risk and ensure profitability. In recent years, advances in machine learning (ML) have made it possible to develop sophisticated models that can accurately predict a customer's likelihood of defaulting on a loan. This has led to an increasing number of financial institutions adopting ML-based models to improve their credit assessment process.

The goal of this research project is to develop a machine learning model that can accurately predict a customer's ability to repay loans. The model will be trained on historical loan data, which will be used to identify patterns and trends that can help predict a customer's creditworthiness. The model will incorporate various features such as income, employment history, credit score, and other relevant financial indicators to generate a prediction.

The use of machine learning in credit assessment has several advantages over traditional methods. ML-based models can quickly process large volumes of data, identify complex patterns and relationships, and make accurate predictions in real-time. This can improve the efficiency and accuracy of the credit assessment process, allowing financial institutions to make better lending decisions and manage risk more effectively.

Overall, this research project has the potential to contribute significantly to the field of credit assessment and risk management. By developing an accurate and reliable ML-based model for predicting customers' ability to repay loans, financial institutions can improve their decision-making processes, reduce credit risk, and ensure profitability.

2. THEPRETICAL BASIS AND RESEARCH MODEL

2.1 Theoretical basis

2.1.1 The concept of ability to repay

The ability to repay is a fundamental concept in lending and credit assessment. It refers to a borrower's capacity to make timely and full repayments on a loan or credit. Lenders use this concept to evaluate a borrower's creditworthiness, and it is based on factors such as income, employment history, credit score, debt-to-income ratio, and other financial indicators. Borrowers must have a clear understanding of their ability to repay to ensure responsible borrowing practices and avoid the negative consequences of defaulting on a loan. Ultimately, the ability to repay is critical for both lenders and borrowers in the lending process.

2.1.2 Previous studies

The study conducted by Azira Abdul Adzis, Juhaida Abu Bakar, and Hanita Kadir Shahar explores the factors influencing young adults' debt in Malaysia. The study's findings suggest that income level is a significant predictor of debt level among Malaysian young adults, indicating that high-income earners have more debt capacity and can borrow more. This information is essential for policymakers and financial institutions in designing financial literacy programs and credit policies that promote responsible borrowing practices among young adults in Malaysia. By considering income levels, they can ensure that borrowers are not taking on more debt than they can afford to repay, thus minimizing the risk of loan default and other negative financial consequences.

The study conducted by Tất Duyên Thư, Phan Ngọc Bảo Anh, and Nguyễn Thùy Dương aimed to identify factors that influence the ability of individual customers to repay loans on time at the Kiên Long Commercial Joint Stock Bank - Cần Thơ Branch, using a non-probability sampling method. The results revealed that all six factors examined significantly affected the ability of customers to repay mortgage loans on time, with high income levels reducing credit risk and increasing the likelihood of on-time repayments. Additionally, the purpose of the loan was inversely correlated with the ability to repay on time, indicating that customers who borrow for business purposes are less likely to repay on time than those who borrow for other reasons. Finally, the loan term was found to be positively correlated with the ability to repay on time. These findings have important implications for commercial banks in Vietnam seeking to minimize the risk of non-performing loans and improve the efficiency of their loan portfolios.

The study "Factors Affecting the Ability to Repay Loans of Individual Customers at

BIDV Tra Vinh Bank" conducted by Assoc. Prof. Dr. Huynh Quang Linh, MSc. Vu Manh Cuong, and MSc. Duong Thi Hong Van in 2021 at the BIDV Tra Vinh branch aimed to identify the factors affecting the ability of individual customers to repay loans at this bank. The results of the study showed that marital status, age, and property ownership all have a significant impact on the ability of individual customers to repay loans. Customers with stable marital status, higher age, and property ownership have a higher ability to repay loans compared to those who do not have these factors.

2.2. Rating Indicators

2.2.1 Accuacy – Accuracy

Accuracy is a concept in the field of machine learning that is used to evaluate the correctness of a predictive model. Accuracy measures the ratio of the number of correct predictions to the total number of predictions.

$$\text{Accuracy} = \frac{TP + TN}{Total\ Samples}$$

2.2.2 ROC

ROC (Receiver Operating Characteristic) is a graphical representation of the performance of a binary classification model, which plots the true positive rate (TPR) against the false positive rate (FPR) at different classification thresholds. In other words, the ROC curve shows how well a model can distinguish between positive and negative classes by varying the threshold for classification. The TPR is the ratio of true positives to the total number of positives, while the FPR is the ratio of false positives to the total number of negatives.

The area under the ROC curve (AUC) is also commonly used as a performance metric, with higher AUC indicating better classification performance. A perfect classifier would have an AUC of 1.0, while a random classifier would have an AUC of 0.5. ROC analysis is widely used in machine learning, medical diagnosis, and other fields where binary classification is important.

2.2.3 Precision

In machine learning, precision is a performance metric that measures the fraction of true positives among the total number of predicted positive instances. In other words, it measures the accuracy of positive predictions made by the model.

The formula for precision is:

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

Where true positives are the number of correctly predicted positive instances, and false positives are the number of incorrectly predicted positive instances.

2.2.4 Recall

Recall is a method that explains how many actual positives we can correctly predict with our model. This is a useful metric where False Negatives are of higher concern than False Positives.

$$\text{Recall} = \frac{TP}{TP + FN}$$

2.2.5 Upsampling

In machine learning, upsampling is a technique used to increase the number of instances in a dataset that have a minority class label. This is often done to balance class distribution and improve the performance of machine learning models.

Upsampling involves duplicating the existing instances of the minority class in the training dataset to create a larger dataset with a more balanced class distribution. This can be done randomly, or by using more advanced methods such as synthetic minority oversampling technique (SMOTE), which creates new instances by interpolating between existing instances of the minority class.

Upsampling can be used in combination with other techniques such as downsampling (reducing the number of instances of the majority class) to achieve a balanced class distribution. However, it is important to be aware that upsampling can also lead to overfitting and reduced model generalization if not done properly.

2.3 Regression model

2.3.1 Logistic

Logistic Regression is a machine learning algorithm for classification, used to predict the output based on a set of input variables. Logistic Regression is often used for binary classification problems, i.e., those with only two classes. For example, classifying emails as spam or not spam, classifying patients as diseased or not diseased, or classifying

credit as good or bad.

Logistic Regression uses a logistic (sigmoid) function to map the input values to a probability value of the positive class. It can also be extended to handle multiple output classes, called Multi-class Logistic Regression.

Advantages:

- Simple and easy to implement
- Fast to train and classify new data
- Provides probabilistic outputs
- Works well with linearly separable data

Disadvantages:

- Assumes a linear relationship between predictor variables and the outcome variable
- Requires a large sample size for stable parameter estimation
- Sensitive to outliers and multicollinearity
- Can only model linear decision boundaries, which may not be sufficient for complex data

2.3.2 Decision Tree

Decision tree is a machine learning model in which data is divided into subgroups based on decisions made on the data's features. This is a supervised learning method used for both classification and regression problems.

A decision tree consists of nodes and edges. Nodes are divided into two types: decision nodes and leaf nodes. Decision nodes represent a question about the attributes of the data. Leaf nodes represent a label or result of a problem.

When building a decision tree, the algorithm selects a data attribute to ask a question about. Then, it divides the data into subgroups based on that question. This process is repeated until there is no way to divide the data into subgroups. At that point, leaf nodes are labeled with the problem's result.

Advantages:

- Easy to understand and explain
- Able to handle different types of data (numeric, categorical, etc.)
- Usable for classification and regression problems.

Disadvantages

- The possibility of overfitting when the tree size is too large
- Poor generalization performance with large and complex datasets
- Susceptibility to noise in the data.

2.3.3 Random Forest

Random forest is a supervised machine learning algorithm used for classification and regression problems. It combines multiple decision trees to create a prediction model. Each tree in the random forest is built on a randomly selected subset of the original training dataset. This helps to reduce overfitting and increase the accuracy of the model.

When performing classification or regression, the trees in the random forest will give results and the final result will be computed by taking the average of those results.

Advantages:

- Can handle various types of data (numerical, categorical, etc.)
- Does not require much time to train the model
- Can handle problems with many features and data samples

Disadvantages:

- Cannot explain the classification or regression process
- Cannot make predictions for new input values that it has not seen before
- Can be prone to overfitting when there are too many trees in the model.

2.3.4 SVM

Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression problems. It is the most popular linear classification method in machine learning.

SVM uses a hyperplane to separate data points of different classes. The hyperplane is chosen such that the distance between the closest data points of each class to the hyperplane is maximized (Margin). The data points closest to the hyperplane are called Support Vectors.

In cases where linear classification is not possible, SVM uses a technique called the kernel trick to transform the data into a different space where linear classification is possible.

Advantages:

- Ability to handle high-dimensional data.

- Ability to handle large datasets.
- Ability to handle both classification and regression problems.

Disadvantages:

- The need to select an appropriate kernel to transform the data into a different space.
- Increased computational complexity and training time as the dimensionality of the data increases.
- Difficulties in explaining the classification process.

2.3.5 XGB

XGBoost (eXtreme Gradient Boosting) is a popular machine learning algorithm used for supervised learning problems, particularly in regression and classification tasks. It is an ensemble learning method that combines multiple decision trees to produce a single powerful predictive model.

XGBoost uses gradient boosting, a type of boosting algorithm, to iteratively improve the performance of weak learners (i.e., decision trees). Each tree is built in a sequential manner, with each subsequent tree correcting the mistakes of the previous tree. This process continues until the performance of the model reaches a satisfactory level.

One of the key strengths of XGBoost is its ability to handle a variety of data types, including numerical, categorical, and ordinal data. It also includes regularization techniques to prevent overfitting, and has a built-in method for handling missing data.

XGBoost has become a popular algorithm in machine learning competitions and has been used to win many of them due to its high accuracy and efficiency.

Advantages:

- High predictive accuracy
- Fast execution speed
- Handles missing data well
- Can handle a variety of data types

Disadvantages:

- Difficult to interpret the internal workings of the model
- Tuning hyperparameters can be time-consuming and requires expertise
- Large memory usage can be a challenge when dealing with very large datasets.

2.3.6 Ada Boost

Adaboost (Adaptive Boosting) is a supervised learning algorithm used for

classification and regression problems. It is an ensemble learning method, where multiple weak learners are combined to create a strong learner.

Adaboost is an adaptive boosting algorithm, which means it tries to improve the weak learners by boosting the weights of misclassified data points in each iteration. These weak learners are built on random subsets of the training data.

During classification or regression, the weak learners in Adaboost make predictions, and the final result is computed by taking the weighted sum of these weak learners with higher weights for more accurate ones.

Advantages:

- Capable of handling large datasets.
- Requires less time to train models.
- Can handle both classification and regression problems.

Disadvantages:

- May be affected by noise or outliers in the training data.
- Can lead to overfitting if the number of weak learners is too large or they are too complex.

3. DATA

3.1 Data

The data set includes 1275 cases of registration for approval of capital use plans applied to individual customers. The data set includes 12 attribute variables: 6 variables are quantitative data while 7 variables are qualitative data.

The target variable is ability to repay debt 'Kha nang tro no'. In there:

0: Inability to repay debt

1: Being able to repay debt

Attribute variables include:

Variables	Variables name	Describe
Gender	Gioi tinh	0: Male 1: Female
Loan purpose	Muc dich vay	1: Consumption 2: Buy house 3: Buy car 4: Study 5: Stock investment
Marital status	Gia dinh	1: Single 2: Having a family 3: Divorce 4: Widow
Collateral	TSĐB	0: No collateral 1: Have collateral
Housing Status	Loai nha o	1. Bought a house 2. Rent a house
Nationality	Quốc tịch	1. Viet Nam 2. Foreign

Electricity bill/Month	Hoa don dien/Thang	Quantitative data, continuous variables. Unit: thousand dong
Loan amount	So tien vay (trieu vnd)	Quantitative data, continuous variables. Unit: million dong
Time at current job	Thoi gian tai cong viec hien tai	Quantitative data, continuous variables. Unit: year
Age	Tuoi	Quantitative data, continuous variables. Unit: year
Loan period	Thoi gian vay (Thang)	Quantitative data, continuous variables. Unit: month
Income	Thu nhap (trieu vnd)	Quantitative data, continuous variables. Unit: million dong

PART II: CONTENT

Implementation process

Step 1: Import data and check data
Step 2: Data processing
Step 3: Descriptive statistics
Step 4: Categorical data visualization
Step 5: Correlation
Step 6: Pre-processing before building the model
Step 7: Check Upsampling
Step 8: Buil model
Step 10: Compare results

1. Step 1: Import data and check data

Code:

```
data = pd.read_excel("Credit.xlsx") #import data
```

```
from copy import deepcopy
data_2 = deepcopy(data)    #copy data into another copy for processing
```

```
data_2.head(5)
```

Results:

	ID	Kha nang tra no	Gioi tinh	Hoa don dien/Thang	So tien vay (trieu vnd)	Muc dich vay	Gia dinh	Thoi gian tai cong viec hien tai	Tuoi	Thoi gian vay (Thang)	Thu nhap (trieu vnd)	TSĐB	Loai nha o	Quốc tịch
0	1	1. Có khả năng trả nợ	1. Nam	300000.0	300.0	5. Đầu tư chứng khoán	1. Độc Thân	6.0	28.0	18.0	32.0	1. Có TSĐB	2. Thuê nhà	0. Nước ngoài
1	2	0. Không có khả năng trả nợ	2. Nữ	600000.0	120.0	1. Tiêu dùng	1. Độc Thân	3.0	26.0	12.0	14.0	0. Không có TSĐB	2. Thuê nhà	1. Việt Nam
2	3	1. Có khả năng trả nợ	1. Nam	500000.0	500.0	2. Mua nhà	2. Có gia đình	8.0	32.0	24.0	80.0	1. Có TSĐB	1. Đã mua nhà	1. Việt Nam
3	4	1. Có khả năng trả nợ	2. Nữ	400000.0	40.0	4. Học tập	1. Độc Thân	2.0	24.0	12.0	8.0	1. Có TSĐB	2. Thuê nhà	1. Việt Nam
4	5	0. Không có khả năng trả nợ	1. Nam	500000.0	150.0	5. Đầu tư chứng khoán	3. Ly hôn	2.0	32.0	12.0	NaN	0. Không có TSĐB	2. Thuê nhà	1. Việt Nam

Code:

```
data_2.info()
```

Result:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1275 entries, 0 to 1274
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   ID                      1275 non-null   int64
 1   Kha nang tra no         1275 non-null   object
 2   Gioi tinh               1275 non-null   object
 3   Hoa don dien/Thang      1247 non-null   float64
 4   So tien vay (trieu vnd) 1254 non-null   float64
```

5	Muc dich vay	1275 non-null	object
6	Gia dinh	1275 non-null	object
7	Thoi gian tai cong viec hien tai	1248 non-null	float64
8	Tuoi	1266 non-null	float64
9	Thoi gian vay (Thang)	1269 non-null	float64
10	Thu nhap (trieu vnd)	1253 non-null	float64
11	TSĐB	1275 non-null	object
12	Loai nha o	1275 non-null	object
13	Quốc tịch	1275 non-null	object

Comment:

The data set includes 1275 cases of registration for approval of capital use plans applied to individual customers. The data set includes 12 attribute variables. In which, there are 7 observation columns which are categorical data (debt repayment capacity, gender, loan purpose, collateral, family, collateral, housing type, nationality). The remaining data columns are all numeric (Electricity bill/month, loan amount (million VND), time at current job, age, loan period (month), income (million VND)). However, there are a few variables that are missing values.

2. Step 2: Data processing

Code:

```
: #put data in digital form
data_2 = data_2.iloc[:,1:]
for i in data_2.columns:
    if data_2[i].dtype == 'object':  # Check if column contains string values
        data_2[i] = data_2[i].apply(lambda x: str(x).split('-')[0])
        data_2[i] = data_2[i].apply(lambda x: str(x).split('.')[0])
data_2.head(5)
```

Results:

	Kha nang tra no	Gioi tinh	Hoa don dien/Thang	So tien vay (trieu vnd)	Muc dich vay	Gia dinh	Thoi gian tai cong viec hien tai	Tuoi	Thoi gian vay (Thang)	Thu nhap (trieu vnd)	TSĐB	Loai nha o	Quốc tịch
0	1	1	300000.0	300.0	5	1	6.0	28.0	18.0	32.0	1	2	0
1	0	2	600000.0	120.0	1	1	3.0	26.0	12.0	14.0	0	2	1
2	1	1	500000.0	500.0	2	2	8.0	32.0	24.0	80.0	1	1	1
3	1	2	400000.0	40.0	4	1	2.0	24.0	12.0	8.0	1	2	1
4	0	1	500000.0	150.0	5	3	2.0	32.0	12.0	NaN	0	2	1

Code:

```
# Get by condition and conditional operator
df = data_2[(data_2['Tuoi '] <18) | (data_2['Tuoi ']>=60)].index
data_2.drop(df, inplace = True)
data_2.head(5)
```

Results:

	Kha nang tra no	Gioi tinh	Hoa don dien/Thang	So tien vay (trieu vnd)	Muc dich vay	Gia dinh	Thoi gian tai cong viec hien tai	Tuoi	Thoi gian vay (Thang)	Thu nhap (trieu vnd)	TSĐB	Loai nha o	Quốc tịch
0	1	1	300000.0	300.0	5	1	6.0	28.0	18.0	32.0	1	2	0
1	0	2	600000.0	120.0	1	1	3.0	26.0	12.0	14.0	0	2	1
2	1	1	500000.0	500.0	2	2	8.0	32.0	24.0	80.0	1	1	1
3	1	2	400000.0	40.0	4	1	2.0	24.0	12.0	8.0	1	2	1
4	0	1	500000.0	150.0	5	3	2.0	32.0	12.0	NaN	0	2	1

Code:

```
data_2.isnull().any()
```

Results:

Kha nang tra no	False
Gioi tinh	False
Hoa don dien/Thang	True
So tien vay (trieu vnd)	True
Muc dich vay	False
Gia dinh	False
Thoi gian tai cong viec hien tai	True
Tuoi	True
Thoi gian vay (Thang)	True
Thu nhap (trieu vnd)	True
TSĐB	False
Loai nha o	False
Quốc tịch	False
dtype: bool	

Code:

```
]: total = data_2.isnull().sum().sort_values(ascending=False)
   percent = (data_2.isnull().sum()/data_2.isnull().count()).sort_values(ascending=False)
   missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percentage'])
   missing_data.head()
```

Results:

	Total	Percentage
Hoa don dien/Thang	**27**	**0.021916**
Thoi gian tai cong viec hien tai	**26**	**0.021104**
Thu nhap (trieu vnd)	**22**	**0.017857**
So tien vay (trieu vnd)	**21**	**0.017045**
Tuoi	**9**	**0.007305**

Code:

```
data_2['Hoa don dien/Thang'].fillna(data_2['Hoa don dien/Thang'].median(), inplace=True)
data_2['Thu nhap  (trieu vnd)'].fillna(data_2['Thu nhap  (trieu vnd)'].median(), inplace=True)
data_2['Thoi gian vay (Thang)'].fillna(data_2['Thoi gian vay (Thang)'].median(), inplace=True)
data_2['So tien vay (trieu vnd)'].fillna(data_2['So tien vay (trieu vnd)'].median(), inplace=True)
data_2['Tuoi '].fillna(data_2['Tuoi '].median(), inplace=True)
data_2['Thoi gian tai cong viec hien tai'].fillna(data_2['Thoi gian tai cong viec hien tai'].median(), inplace=True)
```

Comment:

The current data set has removed those who are under the age of 17 (not yet capable of civil acts) and over 60. When checking for null shows that the variable electricity bill, income, loan period, loan amount, current job duration, age has a null value. In which the variable with the most null values is electricity bill accounting for 2,19%. These null variables are handled by replacing the null value with the variable's median.

3. Step 3: Descriptive statistics

3.1 Re-describe data after processing

Code:

```
data_2["Kha nang tra no"].value_counts()
```

Results:

```
1   723
0   509
Name: Kha nang tra no, dtype: int64
```

Code:

```
data_2.info()
```

Results:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1232 entries, 0 to 1273
Data columns (total 13 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Kha nang tra no                 1232 non-null   object
 1   Gioi tinh                       1232 non-null   object
 2   Hoa don dien/Thang              1232 non-null   float64
 3   So tien vay (trieu vnd)         1232 non-null   float64
 4   Muc dich vay                    1232 non-null   object
 5   Gia dinh                        1232 non-null   object
 6   Thoi gian tai cong viec hien tai 1232 non-null  float64
 7   Tuoi                            1232 non-null   float64
 8   Thoi gian vay (Thang)           1232 non-null   float64
 9   Thu nhap  (trieu vnd)           1232 non-null   float64
 10  TSĐB                            1232 non-null   object
 11  Loai nha o                      1232 non-null   object
 12  Quốc tịch                       1232 non-null   object
dtypes: float64(6), object(7)
memory usage: 134.8+ KB
```

Comment:

According to the data set, there are 723 people who are able to repay, 509 people are not able to pay their debts. After deleting people with inappropriate ages, the dataset is left with 13 columns and 1232 rows. With the columns 'Hoa don dien/Thang', 'So tien vay (trieu vnd)', 'Thoi gian tai cong viec hien tai', 'Tuoi ', 'Thoi gian vay (Thang)','Thu nhap (trieu vnd)', 'Gioi tinh', 'Muc dich

vay', 'Gia dinh', 'TSĐB', 'Loai nha o', 'Quốc tịch' are each independent attribute and the dependent variable is 'Kha nang tra no'.

3.2 Descriptive statistics for quantitative variables

Code:

```
 #classify quantitative variables
numerical = ['Hoa don dien/Thang','So tien vay (trieu vnd)','Thoi gian tai cong viec hien tai','Tuoi ',
                     'Thoi gian vay (Thang)','Thu nhap  (trieu vnd)']
print('Number of numerical features: ', len(numerical))
#descriptive statistics of numerical variables
data_2[numerical].describe()
```

Results:

Number of numerical features: 6

[13]:

	Hoa don dien/Thang	So tien vay (trieu vnd)	Thoi gian tai cong viec hien tai	Tuoi	Thoi gian vay (Thang)	Thu nhap (trieu vnd)
count	1232.000000	1232.000000	1232.000000	232.000000	1232.000000	1232.000000
mean	541355.519481	216.655844	4.667208	29.257305	14.879058	35.059253
std	203314.205983	195.343897	3.278262	5.672588	6.916878	27.350846
min	200000.000000	20.000000	1.000000	20.000000	6.000000	6.000000
25%	400000.000000	80.000000	2.000000	25.000000	12.000000	15.000000
50%	500000.000000	120.000000	4.000000	28.000000	12.000000	20.000000
75%	700000.000000	300.000000	7.000000	34.000000	18.000000	40.000000
max	1000000.000000	800.000000	15.000000	45.000000	36.000000	120.000000

Comment:

- For the variable 'Hoa don dien/Thang', the person whose smallest electricity bill is 200000 and largest is 1000000, generally people usually use 400000 to 700000 electricity bills per month.
- For variable 'So tien vay (trieu vnd)', the smallest loan is 20 million dong and the largest is 800 million dong, the average loan usually ranges from 80 million to 300 million.
- For the variable 'Thoi gian tai cong viec hien tai', since the variable is rounded to years, so the person working for at least 1 year and at most 15 years, in general the average working time of the borrower is about 4 years.
- The age of the borrower is required to be over 17 and under 60, the majority of borrowers are between the ages of 25 and 34.
- With the variable 'Thoi gian vay (Thang)', the loans have the lowest period of 6 months and the longest up to 3 years, most people usually choose a loan from 1 to 2 years.
- With the variable 'Thu nhap (trieu vnd)', the income fluctuation is very large from 6 million to 120 million per month, but most of the borrowers have income from 15 million to 40 million

3.3 Descriptive statistics of qualitative variables

Code:

```
##classify categorical variables
categorical = ['Kha nang tra no','Gioi tinh','Muc dich vay','Gia dinh','TSĐB','Loai nha o','Quốc tịch']
print('Number of numerical features: ', len(categorical))
#descriptive statistics of categorical variables
data_2[categorical].describe()
```

Results:

Number of numerical features: 7

[14]:

	Kha nang tra no	Gioi tinh	Muc dich vay	Gia dinh	TSĐB	Loai nha o	Quốc tịch
count	1232	1232	1232	1232	1232	1232	1232
unique	2	2	5	4	2	2	2
top	1	1	5	1	1	2	1
freq	723	664	304	622	748	908	983

Comment:

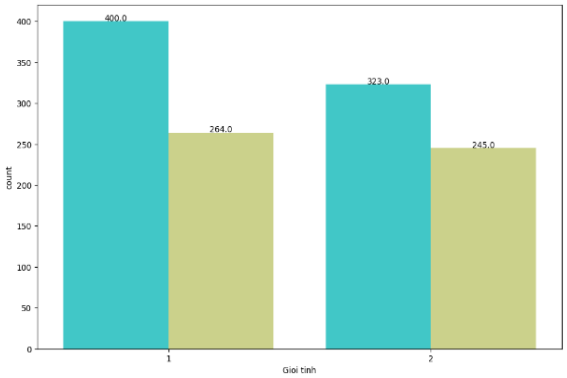
- For qualitative variables, the variable 'Gioi tinh' has 2 genders, male (1) and female (2), in which the male gender appears more with 664 people.
- For variable 'Muc dich vay', there are 5 purposes of borrowing which are borrowing for consumption (1), buying a house (2), buying a car (3), studying (4), investing in securities (5), of which borrowing for stock investment purposes accounted for the majority with 304 people.
- For variable 'Gia dinh', this variable represents marital status with status as single (1), married (2), divorced (3), widowed (4), among borrowers, Most are single 622 people.
- There are 2 situations for the variable 'TSĐB', with collateral (1) being the majority versus no collateral (0).
- Tenants (2) are 908, making up the majority of people who have houses (2).
- Finally, for the variable 'Quốc tịch', the majority of borrowers have Vietnamese nationality (1), foreigners (0) account for very little.

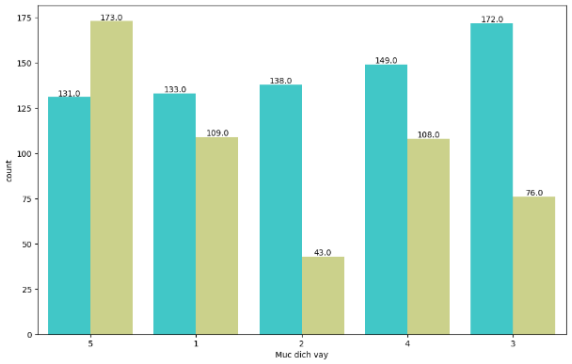
4. Step 4: Categorical data visualization

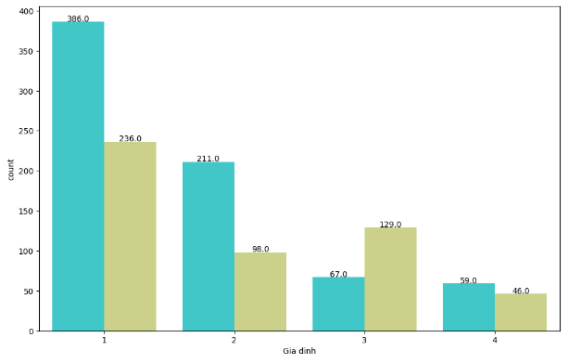
Code:

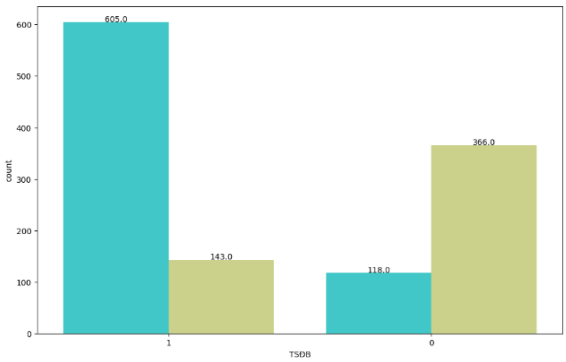
```
plt.figure(figsize=(40,25))
def count(categorical_features):
  x=1
  for i in categorical_features:
    plt.subplot(3,3,x)
    ax = sns.countplot(x=i,data=data_2, palette='rainbow',hue='Kha nang tra no')
    for rect in ax.patches:
      ax.text(rect.get_x() + rect.get_width()  / 2,rect.get_height()+ 0.75,rect.get_height(),horizontalalignment='center', fontsize = 10)
    plt.legend([],[], frameon=False)
    x+=1
  plt.legend(['default', 'undefault'])
  plt.savefig('Data visualization.png')


count(['Gioi tinh', 'Muc dich vay', 'Gia dinh','TSĐB', 'Loai nha o', 'Quốc tịch'
       ])
```

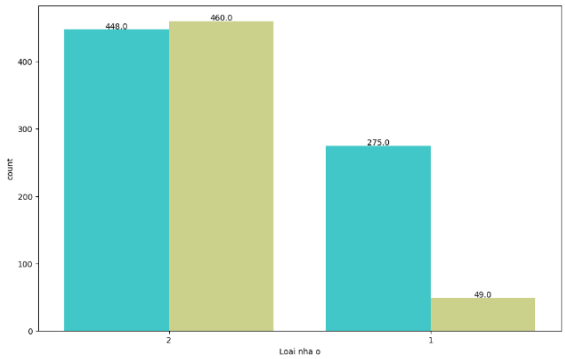
Results:

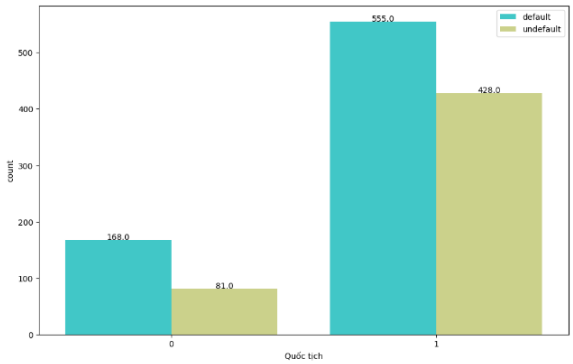








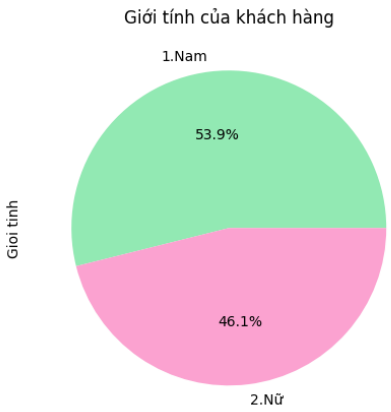




Code:

```
# Categorical data visualization
plt.figure(figsize = (5, 6))
colors = ['#92E9B3','#FBA2D0']
labels = ['1.Nam', '2.Nữ']
df['Gioi tinh'].value_counts().plot(kind='pie',labels=labels, colors=colors, autopct='%1.1f%%')
plt.title('Giới tính của khách hàng')
```

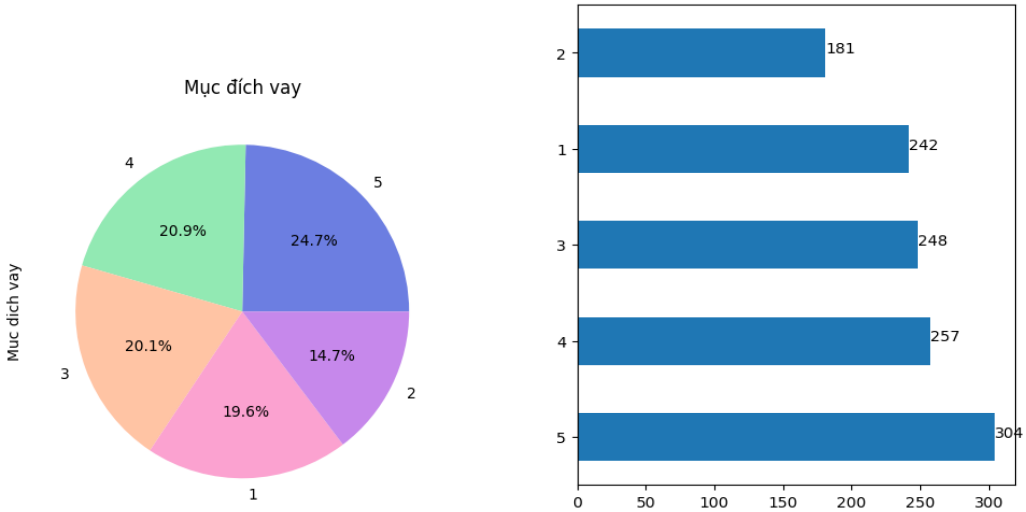
Results:



Code:

```
# Categorical data visualization
plt.figure(figsize = (5, 6))
colors = ['#6C7EE1', '#92E9B3','#FFC4A4','#FBA2D0','#C688EB']
df['Muc dich vay'].value_counts().plot(kind='pie', colors=colors, autopct='%1.1f%%')
plt.title('Mục đích vay')

plt.figure(figsize = (5, 6))
ax = df['Muc dich vay'].value_counts().plot(kind='barh')
for index, value in enumerate(df['Muc dich vay'].value_counts()):
    ax.text(value, index, str(value))
```

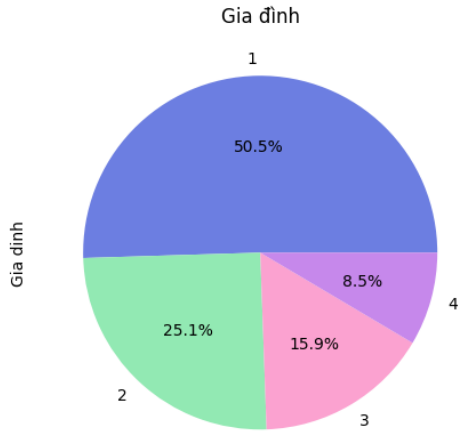
Results:



Code:

```
# Categorical data visualization
plt.figure(figsize = (5, 6))
colors = ['#6C7EE1', '#92E9B3','#FBA2D0','#C688EB']
df['Gia dinh'].value_counts().plot(kind='pie', colors=colors, autopct='%1.1f%%')
plt.title('Gia đình')
```

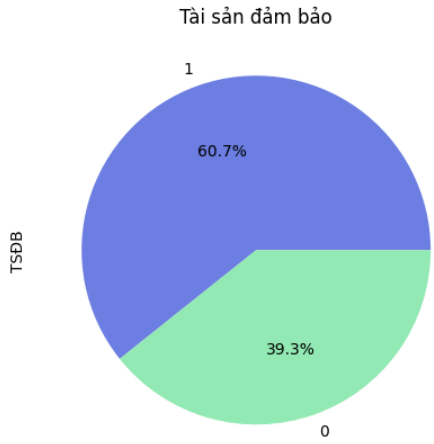
Results:



Code:

```
# Categorical data visualization
plt.figure(figsize = (5, 6))
colors = ['#6C7EE1', '#92E9B3']
df['TSĐB'].value_counts().plot(kind='pie', colors=colors, autopct='%1.1f%%')
plt.title('Tài sản đảm bảo')
```

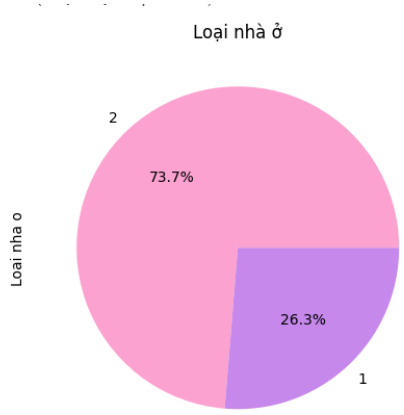
Results:



Code:

```
# Categorical data visualization
plt.figure(figsize = (5, 6))
colors = ['#FBA2D0','#C688EB']
df['Loai nha o'].value_counts().plot(kind='pie', colors=colors, autopct='%1.1f%%')
plt.title('Loại nhà ở')
```

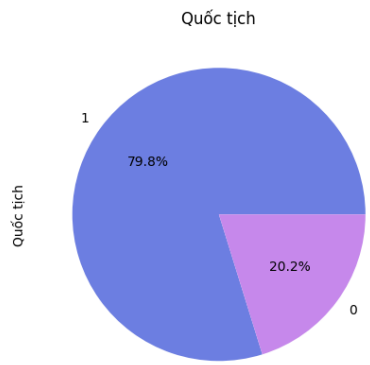
Results:



Code:

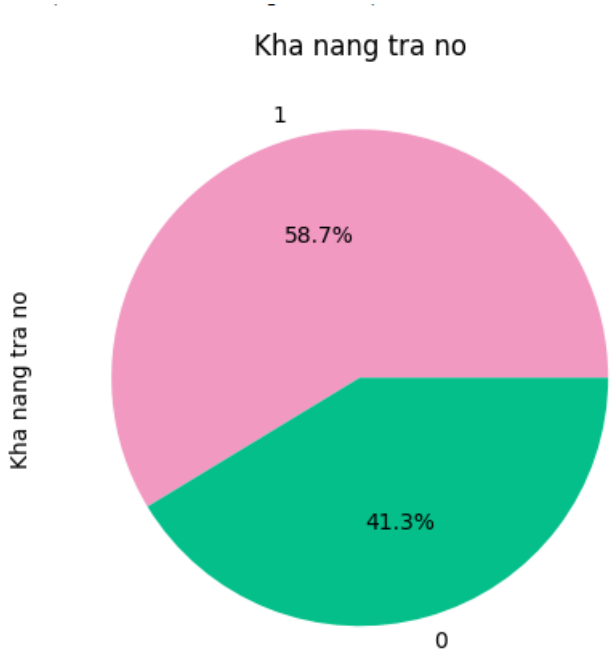
```
# Categorical data visualization
plt.figure(figsize = (5, 6))
colors = ['#6C7EE1','#C688EB']
df['Quốc tịch'].value_counts().plot(kind='pie', colors=colors, autopct='%1.1f%%')
plt.title('Quốc tịch')
```

Results:



Code:

Results:



The graphs above show more clearly the repay capacity for each qualitative variable.

- Figure (1): The number of borrowers is male, accounting for 53.9%, more male borrowers than female, female loan repayment ratio is better than male.
- Figure (2): Borrowers for the purpose of securities investment are the most, 304 people account for 24.7%. Borrowers for study, car purchase and consumption have nearly equal numbers, accounting for 20.9%, 20.1%, and 19.6%, respectively. Finally, borrowing to buy a house accounted for at least 19.6%. Borrowers to invest in securities have very low repayment capacity, the number of people who can't pay their debts is more than the number of people who can. The two best borrowers are borrowers to buy a house and to buy a car. Borrowing for consumption and study, the repayment ability is still good when the number of people who can repay is higher than the number of people who can't, but the difference is not much.
- Figure (3): Single and married are two relatively common marital statuses, the debt repayment ratio of these two subjects is also very good. However, for divorced people, the ability to repay debt is much lower, the risk of default is high. For the widow, the ability to repay the debt is at an acceptable level.
- Figure (4): The number of borrowers with collateral is 60.7%. And when there is collateral, the rate of ability to repay is much higher.
- Figure (5): The repayment capacity of people who have houses is very high, these people are very stable but very few. And tenants account for the majority, so their ability to repay is not much different.

- Figure (6): The ability of Vietnamese and foreigners to repay debt is not too different
-

5. Correlation

Code:

```
#reformat data type
df = df.astype('int64')
```

```
#Autocorrelation test
cor = df.astype('float64').corr()
cor
```

Results:

	Kha nang tra no	Gioi tinh	Hoa don dien/Thang	So tien vay (trieu vnd)	Muc dich vay	Gia dinh	Thoi gian tai cong viec hien tai	Tuoi	Thoi gian vay (Thang)	Thu nhap (trieu vnd)	TSĐB	Loai nha o	Quốc tịch
Kha nang tra no	1.000000	-0.034164	-0.491128	-0.042060	-0.125493	-0.123995	0.274464	0.095636	-0.142469	0.430572	0.560416	-0.317740	-0.089792
Gioi tinh	-0.034164	1.000000	-0.170979	-0.075139	-0.141750	-0.033682	-0.010919	-0.193594	0.089186	-0.062576	-0.182899	0.005092	0.339792
Hoa don dien/Thang	-0.491128	-0.170979	1.000000	0.307715	-0.131463	0.153701	0.026942	0.173019	0.066552	-0.113057	-0.280366	0.043537	-0.051742
So tien vay (trieu vnd)	-0.042060	-0.075139	0.307715	1.000000	0.047535	0.216092	0.202506	0.371474	0.367213	0.408019	0.095085	0.174548	-0.075076
Muc dich vay	-0.125493	-0.141750	-0.131463	0.047535	1.000000	0.099306	-0.040225	0.058505	0.104635	-0.063254	-0.005077	0.120298	-0.208538
Gia dinh	-0.123995	-0.033682	0.153701	0.216092	0.099306	1.000000	0.337075	0.499152	0.077181	0.303507	-0.152995	-0.180832	-0.070973
Thoi gian tai cong viec hien tai	0.274464	-0.010919	0.026942	0.202506	-0.040225	0.337075	1.000000	0.392387	0.149800	0.360708	0.211456	-0.502323	-0.059131
Tuoi	0.095636	-0.193594	0.173019	0.371474	0.058505	0.499152	0.392387	1.000000	-0.038481	0.446549	0.117399	-0.163430	-0.077329
Thoi gian vay (Thang)	-0.142469	0.089186	0.066552	0.367213	0.104635	0.077181	0.149800	-0.038481	1.000000	0.043583	-0.159258	0.151944	-0.026636
Thu nhap (trieu vnd)	0.430572	-0.062576	-0.113057	0.408019	-0.063254	0.303507	0.360708	0.446549	0.043583	1.000000	0.285876	-0.161091	0.011959
TSĐB	0.560416	-0.182899	-0.280366	0.095085	-0.005077	-0.152995	0.211456	0.117399	-0.159258	0.285876	1.000000	-0.272882	0.037987
Loai nha o	-0.317740	0.005092	0.043537	0.174548	0.120298	-0.180832	-0.502323	-0.163430	0.151944	-0.161091	-0.272882	1.000000	-0.075679
Quốc tịch	-0.089792	0.339792	-0.051742	-0.075076	-0.208538	-0.070973	-0.059131	-0.077329	-0.026636	0.011959	0.037987	-0.075679	1.000000

Code:

```
#Test the correlation of the independent variables with the dependent variable
pd.DataFrame(df.astype('float64').corr().iloc[1:,0])
```

Results:

	Kha nang tra no
Gioi tinh	-0.034164
Hoa don dien/Thang	-0.491128
So tien vay (trieu vnd)	-0.042060
Muc dich vay	-0.125493

	Kha nang tra no
Gia dinh	-0.123995
Thoi gian tai cong viec hien tai	0.274464
Tuoi	0.095636
Thoi gian vay (Thang)	-0.142469
Thu nhap (trieu vnd)	0.430572
TSĐB	0.560416
Loai nha o	-0.317740
Quốc tịch	-0.089792

Comment:

From the correlation test table, it shows that the gender variable and the nationality variable have no correlation with other variables. The remaining independent variables have a slight correlation with each other and have a good correlation with the dependent variable (>10%). The dependent variable 'khang tra no', it is possible to determine the variables 'Gioi tinh', 'So tien vay (trieu vnd)', 'Tuoi', 'Quốc tịch' is not statistically significant (<10%) so it will not be included in the model.

The target variable "Kha Nang tra no" has a high degree of correlation with the variables "Thu nhap", "TSĐB" and "Thoi gian tai cong viec hien tai". This shows that customers with high income, stable working time and special asset certificates have better debt repayment ability than other customers.

Variables "Kha nang tra no" and "Loai nha o" have a low correlation with the target variable, that is, they do not affect the customer's ability to repay much.

The variables "Hoa don dien/Thang" and "Quốc tịch" are negatively correlated with the target variable, that is, when the average monthly electricity bill and the customer has a foreign nationality, the ability to repay the loan will decrease.

The variables "So tien vay (trieu VND)", "Thoi gian vay (Thang)", "Gioi Tinh", "Tuoi" and "Gia Dinh" also have low correlation with the target variable. However, it is still necessary to take a closer look as they can affect the customer's ability to repay.

7. Pre-processing before building the model

Code:

```
target = ['Kha nang tra no']
features = [ 'Hoa don dien/Thang','Gia dinh','Muc dich vay', 'Thoi gian tai cong viec hien tai','Thoi gian vay (Thang)','Thu nhap  (trieu vnd)','TSĐB','Loai nha o']
print('Target: ', target)
print('Features: ', features)

categorical_features = [x for x in data.columns if (x not in features and x != 'default')]
print('Number of numerical features: ', len(features))
```

Results:

Target: ['Kha nang tra no']
Features: ['Hoa don dien/Thang', 'Gia dinh', 'Muc dich vay', 'Thoi gian tai cong viec hien tai', 'Thoi gian vay (Th
ang)', 'Thu nhap (trieu vnd)', 'TSĐB', 'Loai nha o']
Number of numerical features: 8

Comment:

We choose 8 feature variables to include in the model is 'Hoa don dien/Thang','Gia dinh', ‘Muc dich vay’, 'Thoi gian tai cong viec hien tai', 'Thoi gian vay (Thang)', 'Thu nhap (trieu vnd)', 'TSĐB', 'Loai nha o'. And the target variable is 'Kha nang tra no'.

Code:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_auc_score
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
```

```
# Hàm vẽ đường ROC-AUC
def _plot_roc_curve(fpr, tpr, thres, auc):
    plt.figure(figsize = (10, 8))
    plt.plot(fpr, tpr, 'b-', color='darkorange', lw=2, linestyle='--', label='ROC curve (area = %0.2f)'%auc)
    plt.plot([0, 1], [0, 1], '--')
    plt.axis([0, 1, 0, 1])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.legend(loc='lower right')
    plt.title('ROC Curve')
```

```
X = df[features].values
y = df[target].values

n_state = 24
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=n_state) #Train 30% , test 70%
```

```
from sklearn.preprocessing import StandardScaler

# Call scale function
sc_X = StandardScaler()
#Scale train set
X_train = sc_X.fit_transform(X_train)
# Scale test set
X_test = sc_X.fit_transform(X_test)
```

```
X
```

Results:

```
array([[300000,     1,     5, ...,    32,     1,     2],
       [600000,     1,     1, ...,    14,     0,     2],
       [500000,     2,     2, ...,    80,     1,     1],
       ...,
       [600000,     2,     3, ...,    60,     0,     2],
       [300000,     1,     5, ...,    32,     1,     1],
       [400000,     3,     4, ...,    20,     1,     1]], dtype=int64)
```

8. Check upsampling and Logistic model

Since the target variable is 'khang tra no', there is too much difference between the two attributes in the variable, so to avoid machine learning, we will up sampling so that these two attributes are equal.

I run the Logistic model before and after upsampling to check if upsampling improves the model or not.

Code:

```
: #testing accuracy in learning ability, making predictions

  LR_classifier1 = LogisticRegression(multi_class='multinomial',random_state=n_state)

  LR_classifier1.fit(X_train, y_train.ravel())

  y_pred = LR_classifier1.predict(X_test)

  print('Confusion matrix:')
  print(pd.DataFrame(confusion_matrix(y_test,y_pred)),'\n')

  print('Classification report:')
  print(classification_report(y_test,y_pred))

  print('Logistic Regression accuracy: ', round(accuracy_score(y_test, y_pred),4))
```

Results:

Confusion matrix:

	0	1
0	126	15
1	28	201

Classification report:

	precision	recall	f1-score	support
0	0.82	0.89	0.85	141
1	0.93	0.88	0.90	229
accuracy			0.88	370
macro avg	0.87	0.89	0.88	370
weighted avg	0.89	0.88	0.88	370

Logistic Regression accuracy: 0.8838

Code:

```
data_majority = df[df['Kha nang tra no']==1]
data_minority = df[df['Kha nang tra no']==0]
```

```
print('class 0:', data_majority.shape)
print('class 1:', data_minority.shape)
```

```
class 0: (723, 13)
class 1: (509, 13)
```

```
from sklearn.utils import resample
data_minority_upsampled = resample(data_minority, replace = True, n_samples = data_majority.shape[0], random_state=42)
print('Shape of class 1 after sampling:', data_minority_upsampled.shape)
data_minority_upsampled.head(5)
```

```
Shape of class 1 after sampling: (723, 13)
```

	Kha nang tra no	Gioi tinh	Hoa don dien/Thang	So tien vay (trieu vnd)	Muc dich vay	Gia dinh	Thoi gian tai cong viec hien tai	Tuoi	Thoi gian vay (Thang)	Thu nhap (trieu vnd)	TSDB	Loai nha o	Quốc tịch
248	0	1	800000	500	2	2	7	38	12	100	1	2	0
1086	0	2	600000	120	4	1	1	22	24	30	0	2	0
871	0	2	600000	120	1	1	3	26	12	20	0	2	1
679	0	1	600000	60	4	3	1	27	18	15	0	2	1
256	0	1	300000	100	3	1	2	25	6	60	1	2	1

```
data_new = [data_majority, data_minority_upsampled]
df = pd.concat(data_new)
df.head(5)
```

Results:

	Kha nang tra no	Gioi tinh	Hoa don dien/Thang	So tien vay (trieu vnd)	Muc dich vay	Gia dinh	Thoi gian tai cong viec hien tai	Tuoi	Thoi gian vay (Thang)	Thu nhap (trieu vnd)	TSĐB	Loai nha o	Quốc tịch
0	1	1	300000	300	5	1	6	28	18	32	1	2	0
2	1	1	500000	500	2	2	8	32	24	80	1	1	1
3	1	2	400000	40	4	1	2	24	12	8	1	2	1
5	1	2	300000	300	2	4	15	35	12	120	1	1	1
8	1	1	400000	70	3	1	7	33	6	65	1	1	1

Code:

```
X = df[features].values
y = df[target].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=n_state) #Train 30% , test 70%
```

```
from sklearn.preprocessing import StandardScaler

# Call scale function
sc_X = StandardScaler()
#Scale train set
X_train = sc_X.fit_transform(X_train)
# Scale test set
X_test = sc_X.fit_transform(X_test)
```

```
X
```

```
array([[300000,      1,      5, ...,     32,      1,      2],
       [500000,      2,      2, ...,     80,      1,      1],
       [400000,      1,      4, ...,      8,      1,      2],
       ...,
       [900000,      4,      5, ...,     20,      0,      1],
       [600000,      1,      2, ...,     20,      0,      2],
       [400000,      1,      4, ...,     15,      0,      2]], dtype=int64)
```

```
#testing accuracy in learning ability, making predictions

LR_classifier = LogisticRegression(multi_class='multinomial',random_state=n_state)

LR_classifier.fit(X_train, y_train.ravel())

y_pred = LR_classifier.predict(X_test)

print('Confusion matrix:')
print(pd.DataFrame(confusion_matrix(y_test,y_pred)),'\n')

print('Classification report:')
print(classification_report(y_test,y_pred))

print('Logistic Regression accuracy: ', round(accuracy_score(y_test, y_pred),4))
```

Results:

Confusion matrix:

	0	1
0	214	3
1	26	191

Classification report:

	precision	recall	f1-score	support
0	0.89	0.99	0.94	217
1	0.98	0.88	0.93	217
accuracy			0.93	434
macro avg	0.94	0.93	0.93	434
weighted avg	0.94	0.93	0.93	434

Logistic Regression accuracy: 0.9332

Comment:

Upsampling

The results show that the accuracy of the Logistic model before upsampling is 0.88 and after upsampling is 0.93. Upsampling improved model accuracy by more than 5%. So I decided to use this upsampling method for my research paper.

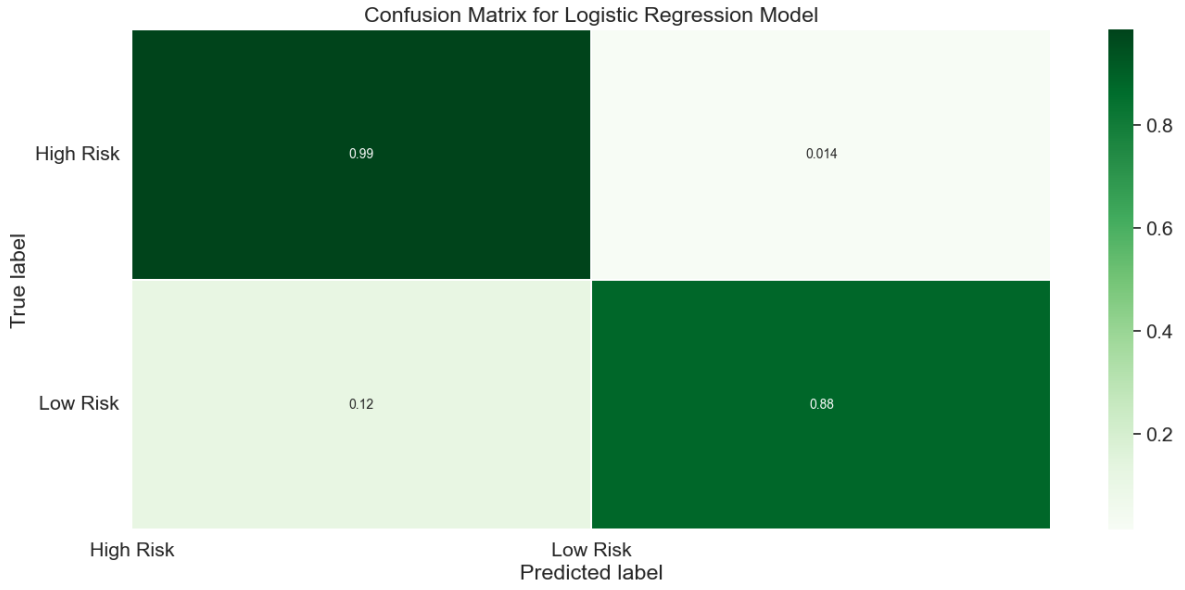
Logistics model:

The confusion matrix shows that the logistic regression model correctly classified 214 out of 217 observations for class 0 (negative) and 191 out of 217 observations for class 1 (positive). It misclassified 3 observations as false positives (predicted positive but actually negative) and 26 observations as false negatives (predicted negative but actually positive).
The classification report provides more information about the performance of the model. The precision for class 0 is 0.89, which means that out of all the observations predicted as negative, 89% are actually negative. The recall for class 0 is 0.99, which means that out of all the actual negative observations, the model correctly identified 99%. The F1-score for class 0 is 0.94, which is the harmonic mean of precision and recall.
Similarly, for class 1, the precision is 0.98, which means that out of all the observations predicted as positive, 98% are actually positive. The recall for class 1 is 0.88, which means that out of all the actual positive observations, the model correctly identified 88%. The F1-score for class 1 is 0.93.
The accuracy of the logistic regression model is 0.9332, which is the proportion of correctly classified observations out of the total number of observations. The macro-avg of precision, recall and f1-score is 0.94, which is the average of these metrics weighted equally for both classes. The weighted-avg of these metrics is also 0.94, which is the average of these metrics weighted by the number of observations in each class. Overall, the performance of the logistic regression model appears to be good, with high accuracy and balanced precision and recall for both classes.Overall, the logistic regression model shows fairly good results when evaluating the repayment ability of customers. However, it should be noted that the model still has some errors in its predictions, so it is important to continue refining and improving the model to achieve better performance.

Code:

```
# Confusion matrix
from sklearn.metrics import confusion_matrix , classification_report
y_true = y_test
y_pred = LR_classifier.predict(X_test)
confusion_matrix(y_true, y_pred)

# Get and reshape confusion matrix data
matrix = confusion_matrix(y_true, y_pred)
matrix = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]

# Build the plot
plt.figure(figsize=(16,7))
sns.set(font_scale=1.4)
sns.heatmap(matrix, annot=True, annot_kws={'size':10},
            cmap=plt.cm.Greens, linewidths=0.2)

# Add labels to the plot
class_names = [ 'High Risk','Low Risk']
tick_marks = np.arange(len(class_names))
tick_marks2 = tick_marks + 0.5
plt.xticks(tick_marks, class_names, rotation=0)
plt.yticks(tick_marks2, class_names, rotation=0)
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix for Logistic Regression Model')
plt.show()
```

Results:



Comment:

- TP (True Positive): 99% correctly predict a person will not be able to repay the deb
- TN (True Negative): 88% of the models accurately predict who is able to repay the debt
- FP (False Positive - Type 1 Error): 12% wrongly predicts an insolvent person to be able to repay
- FN (False Negative - Type 2 Error): 14% wrongly predict that a person who can repay the debt becomes an insolvent person

Code:

```
import statsmodels.api as SM
model_1 = SM.Logit(y_train, X_train).fit()
print(model_1.summary())
```

Results:

Optimization terminated successfully.
Current function value: 0.241221
Iterations 8

Logit Regression Results

Dep. Variable:	y	No. Observations:	1012
Model:	Logit	Df Residuals:	1004
Method:	MLE	Df Model:	7
Date:	Thu, 11 May 2023	Pseudo R-squ.:	0.6520
Time:	17:37:28	Log-Likelihood:	-244.12
converged:	True	LL-Null:	-701.46
Covariance Type:	nonrobust	LLR p-value:	3.214e-193

	coef	std err	z	P>|z|	[0.025	0.975]
x1	-2.1378	0.174	-12.253	0.000	-2.480	-1.796
x2	-0.9835	0.214	-4.596	0.000	-1.403	-0.564
x3	-0.9520	0.149	-6.369	0.000	-1.245	-0.659
x4	0.7805	0.183	4.274	0.000	0.423	1.138
x5	-0.1284	0.132	-0.973	0.331	-0.387	0.130
x6	2.1048	0.205	10.257	0.000	1.703	2.507
x7	0.9905	0.136	7.278	0.000	0.724	1.257
x8	-1.1701	0.175	-6.679	0.000	-1.513	-0.827

Comment:

The logistic regression model appears to have performed well, as indicated by the convergence of the optimization algorithm and the relatively high pseudo R-squared value of 0.6520. The coefficients of the model can be interpreted as the log-odds of the response variable for each predictor variable, holding all other variables constant.

The coefficient for variable x1 is -2.1378, which suggests that as x1 increases by 1 unit, the log-odds of the response variable decreases by 2.1378 units. Similarly, the coefficient for variable x6 is 2.1048, indicating that as x6 increases by 1 unit, the log-odds of the response variable increase by 2.1048 units.

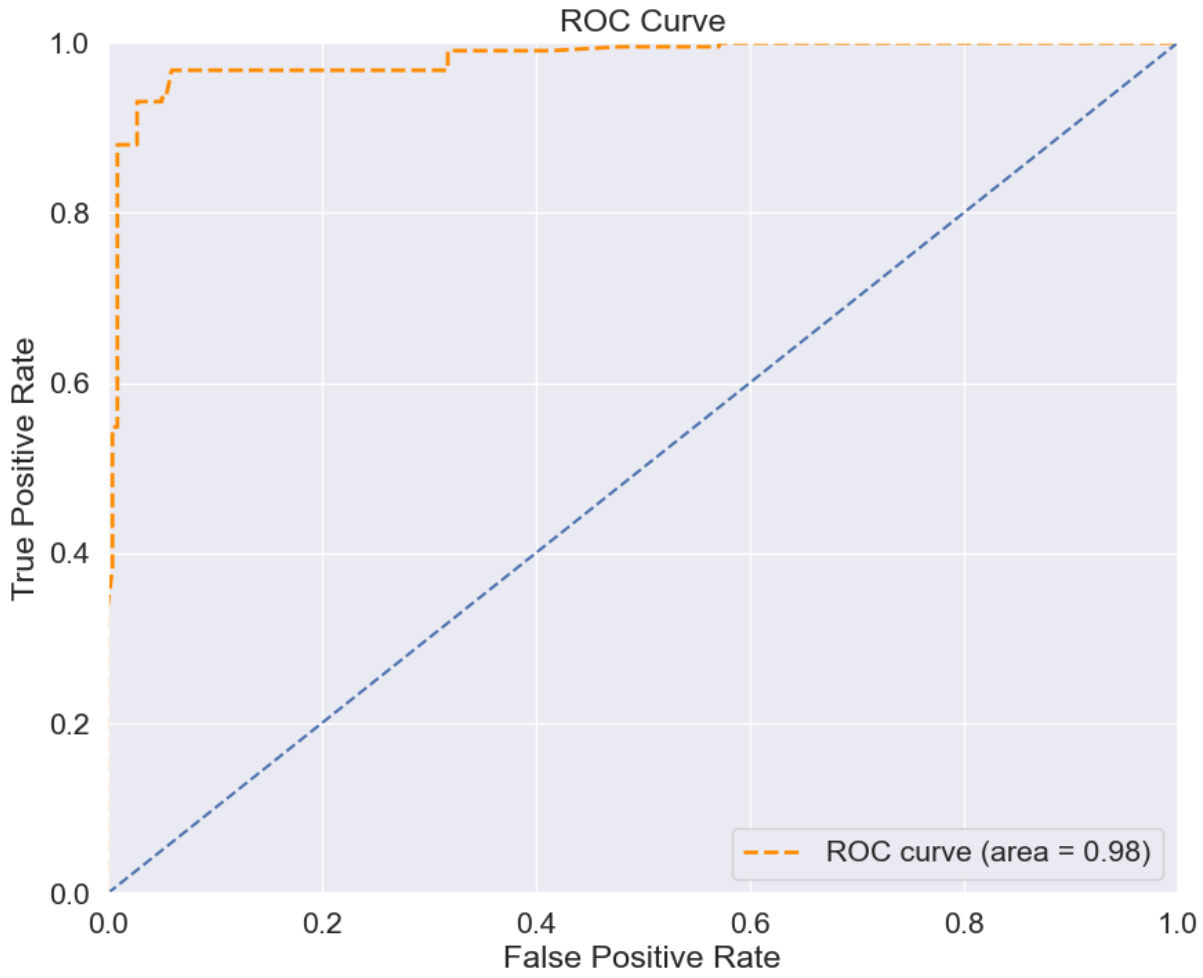
The p-values associated with each coefficient indicate that all predictor variables are statistically significant in predicting the response variable, except for variable x5 which has a p-value of 0.331 and thus is not statistically significant.

Overall, the logistic regression model appears to have a good fit to the data, with significant predictor variables and a high pseudo R-squared value. However, further evaluation and validation of the model on new data is recommended to ensure its accuracy and generalizability.

Code:

```
# ROC
y_pred_prob_test = LR_classifier.predict_proba(X_test)[:, 1]
fpr, tpr, thres = roc_curve(y_test, y_pred_prob_test)
roc_auc = auc(fpr, tpr)

_plot_roc_curve(fpr, tpr, thres, roc_auc)
```

Results:



Comment:

The ROC (Receiver Operating Characteristic) curve is a graphical representation of the performance of a binary classification model at various threshold settings. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) for different threshold values. An area under the ROC curve (AUC-ROC) value of 0.98 indicates that the model has a very good discriminatory power and can effectively distinguish between the positive and negative classes. An AUC-ROC value of 1.0 means the model has a perfect discriminatory power, while a value of 0.5 means that the model performs no better than random guessing.

In summary, an AUC-ROC value of 0.98 is a very good result and suggests that the logistic regression model has high accuracy in predicting the target variable.

Code:

```
#Testing the significance of the attribute in Logistic Regression Model

importance_lr = LR_classifier.coef_[0]
features_importances_lr = pd.DataFrame({'FeatureName':  df.loc[:,features].columns, 'Logistic Regression Feature Importance': importance_lr})
features_importances_lr.reindex(features_importances_lr['Logistic Regression Feature Importance'].abs().sort_values(ascending=False).index)
```

Results:

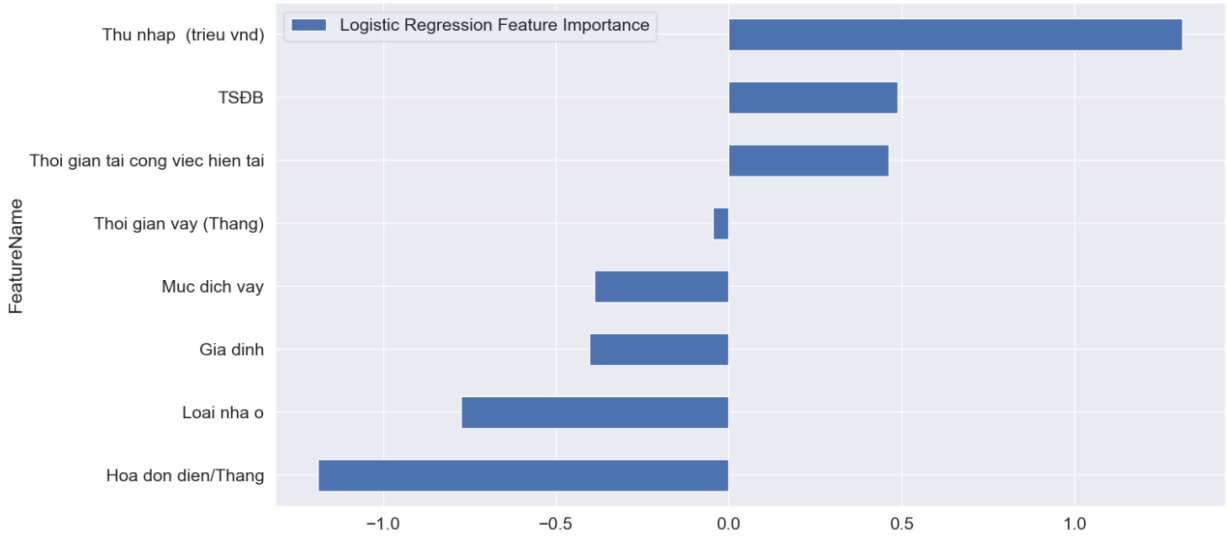
[45]:

	FeatureName	Logistic Regression Feature Importance
5	Thu nhap (trieu vnd)	1.311972
0	Hoa don dien/Thang	-1.187625
7	Loai nha o	-0.773868
6	TSĐB	0.488606
3	Thoi gian tai cong viec hien tai	0.463208
1	Gia dinh	-0.403549
2	Muc dich vay	-0.387905
4	Thoi gian vay (Thang)	-0.046445

Code:

```
#visualization of Attribute Significance Test in Logistic Regression Model
features_importances_lr.sort_values("Logistic Regression Feature Importance").plot(figsize=(15,8), x="FeatureName", y=["Logistic Regression Feature Importance"], kind="barh")
```

Results:



Comment:

According to the Logistic model, the variables 'Thu nhap (trieu vnd)','TSĐB', 'Thoi gian cong viec hien tai' have a positive effect on debt repayment capacity, meaning that when the values of these variables increase, the ability to repay debt also increases. In which the most impact is the variable 'Thu nhap (trieu vnd)'(1.53), followed by the variables 'TSĐB'(0.45), 'Thoi gian cong viec hien tai'(0.32) the level of impact decreases in turn

In contrast, the variable 'Hoa don dien/Thang', 'Muc dich vay', 'Gia dinh', 'Thoi gian vay (Thang)', 'Loai nha o' has a negative effect on debt repayment capacity, which means that when the values of these variables increase, the ability to repay debt decreases. In which the most impact is the variable 'Hoa don dien/Thang'(-1.19), followed by the variables 'Loai nha o'(-0.957), 'Gia dinh'(-0.357), 'Thoi gian vay (Thang)'(-0.054) the level of impact decreases in turn.

9. Decision Tree

Code:

```
DT_classifier = DecisionTreeClassifier(max_depth=4)
DT_classifier.fit(X_train, y_train.ravel())

y_pred = DT_classifier.predict(X_test)
print('Confusion matrix:')
print(pd.DataFrame(confusion_matrix(y_test,y_pred)),'\n')

print('Classification report:')
print(classification_report(y_test,y_pred))

print('Decision Tree accuracy: ', round(accuracy_score(y_test, y_pred),2))
```

Results:

Confusion matrix:

	0	1
0	211	6
1	8	209

Classification report:

	precision	recall	f1-score	support
0	0.96	0.97	0.97	217
1	0.97	0.96	0.97	217
accuracy			0.97	434
macro avg	0.97	0.97	0.97	434
weighted avg	0.97	0.97	0.97	434

Decision Tree accuracy: 0.97

Comment:

The numbers in the confusion matrix tell us the number of correct and incorrect predictions on each class.

In this case, the model correctly predicted 211 out of 217 samples for class 0 (no loan repayment capability) and 209 out of 217 samples for class 1 (loan repayment capability). The model made 6 incorrect predictions for class 0 and 8 incorrect predictions for class 1.

This can be explained as follows: The model has an accuracy of 97% in predicting the loan repayment capability of an individual or a group. However, there are still some samples that are predicted incorrectly.

Precision, recall, and F1-score are other metrics that help evaluate the performance of the model on each class. The precision for class 0 is 0.96, which means that out of all the predictions that are class 0, 96% are correct. The recall for class 0 is 0.97, which means that the model correctly corresponds 97% of all the actual class 0 samples. The F1-score for class 0 is 0.97.

similarly, for class 1, the precision is 0.98, which means that out of all the observations predicted as positive, 98% are actually positive. The recall for class 1 is 0.88, which means that out of all the actual positive observations, the model correctly identified 88%. The F1-score for class 1 is 0.93.

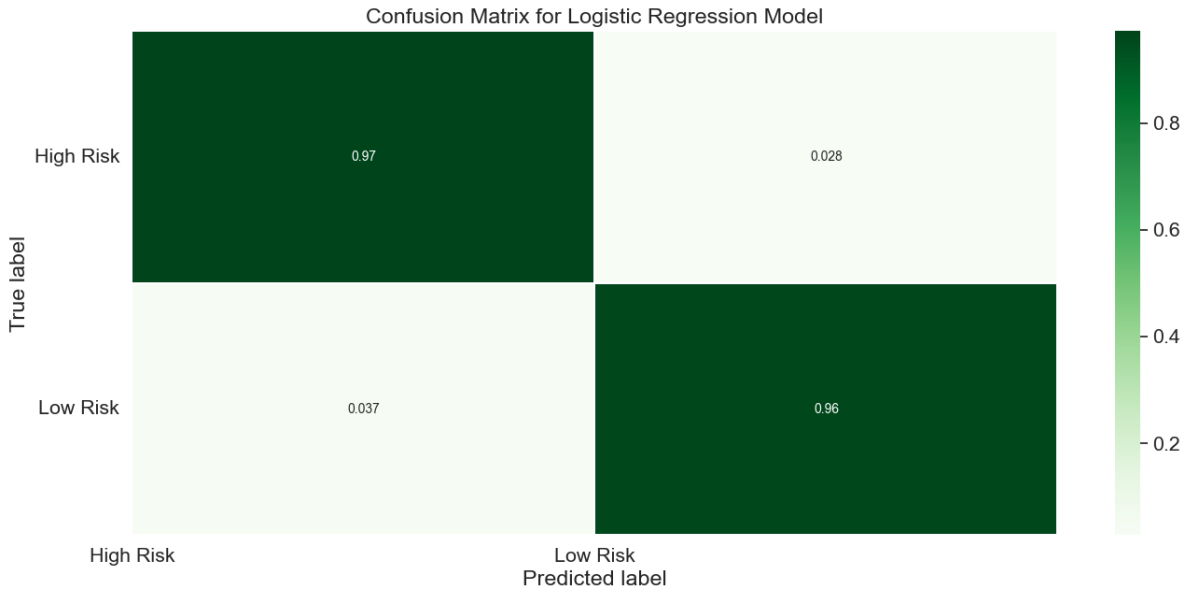
For precision and recall, both classes have roughly equal high values, indicating that the model is not biased towards one class. The F1-score of both classes is very high (0.97), indicating that the model achieves good results for both classes.

In general, the Decision Tree model is a very good classification model for this problem, achieving high accuracy and not being biased towards any one class.

Code:

```
matrix = confusion_matrix(y_true, y_pred)
matrix = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]

# Get and reshape confusion matrix data
matrix = confusion_matrix(y_true, y_pred)
matrix = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]

# Build the plot
plt.figure(figsize=(16,7))
sns.set(font_scale=1.4)
sns.heatmap(matrix, annot=True, annot_kws={'size':10},
            cmap=plt.cm.Greens, linewidths=0.2)

# Add labels to the plot
class_names = [ 'High Risk','Low Risk']
tick_marks = np.arange(len(class_names))
tick_marks2 = tick_marks + 0.5
plt.xticks(tick_marks, class_names, rotation=0)
plt.yticks(tick_marks2, class_names, rotation=0)
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix for Logistic Regression Model')
plt.show()
```

Results:



Comment:

- TP (True Positive): 97% correctly predict a person will not be able to repay the deb
- TN (True Negative): 96% of the models accurately predict who is able to repay the debt
- FP (False Positive - Type 1 Error): 3.7% wrongly predicts an insolvent person to be able to repay
- FN (False Negative - Type 2 Error): 2.8% wrongly predict that a person who can repay the debt becomes an insolvent person

Code:

```
# ROC
y_pred_prob_test = DT_classifier.predict_proba(X_test)[:, 1]
fpr, tpr, thres = roc_curve(y_test, y_pred_prob_test)
roc_auc = auc(fpr, tpr)

_plot_roc_curve(fpr, tpr, thres, roc_auc)
```

Results:

In this case, the ROC curve for the decision tree model has an area under the curve (AUC) of 0.98, which is a high value and indicates that the model has good predictive ability across all classification thresholds. This is also reflected in the high and balanced precision and recall results of the model, which are nearly equal. This means that the model has a relatively balanced accuracy and coverage of its predictions.

However, it should be noted that accuracy and coverage also depend on the intended use of the model. For example, if the goal is to identify all cases with low creditworthiness, we may use a higher classification threshold to increase recall, even if this comes at the cost of some decrease in precision**.**

Code:

```
importance_dt = DT_classifier.feature_importances_
features_importances_dt = pd.DataFrame({'FeatureName': df.loc[:,features].columns, 'Decision Tree Feature Importance': importance_dt})
features_importances_dt.sort_values(by=['Decision Tree Feature Importance'], ascending=False)
```

Results:

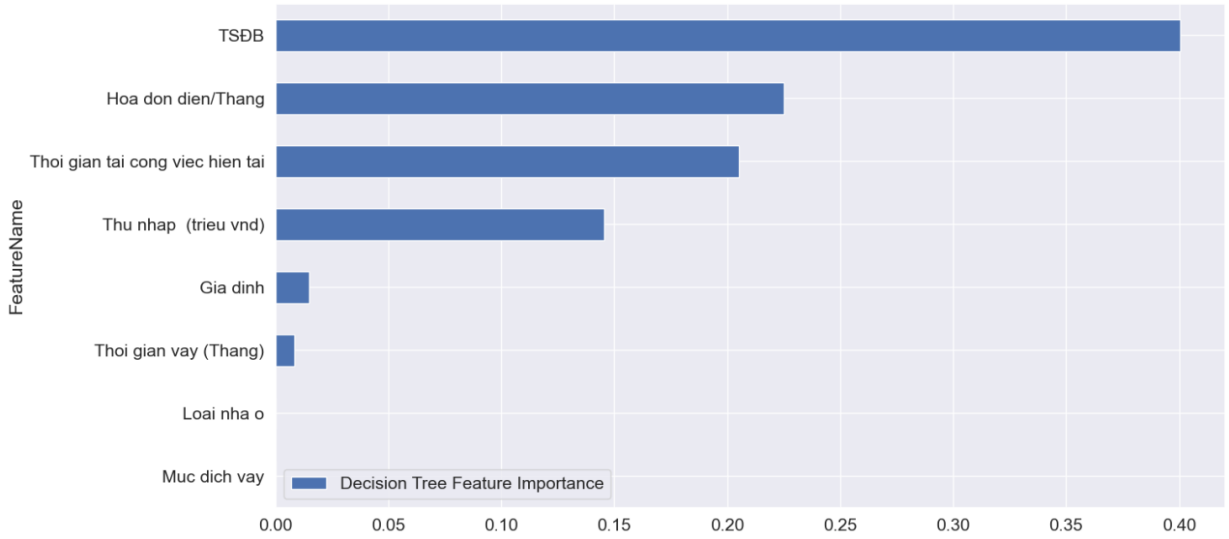
FeatureName	Decision Tree Feature Importance	
6	TSĐB	0.400387
0	Hoa don dien/Thang	0.224891
3	Thoi gian tai cong viec hien tai	0.205392
5	Thu nhap (trieu vnd)	0.145616
1	Gia dinh	0.015095
4	Thoi gian vay (Thang)	0.008619

FeatureName	Decision Tree Feature Importance	
2	Muc dich vay	0.000000
7	Loai nha o	0.000000

Code:

```
features_importances_dt.sort_values("Decision Tree Feature Importance").plot(figsize=(15,8), x="FeatureName", y=["Decision Tree Feature Importance"], kind="barh")
```

Results:



Comment:

The Decision Tree feature importance suggests that the most important feature for predicting the loan approval outcome is the "TSĐB" feature with an importance score of 0.400. This is followed by "Hoa don dien/Thang" and "Thoi gian tai cong viec hien tai" features with importance scores of 0.225 and 0.205, respectively. The "Thu nhap (trieu vnd)" feature also appears to be important with an importance score of 0.146. The "Gia dinh," "Thoi gian vay (Thang)," "Muc dich vay," and "Loai nha o" features have much lower importance scores, with "Muc dich vay" and "Loai nha o" having importance scores of zero.

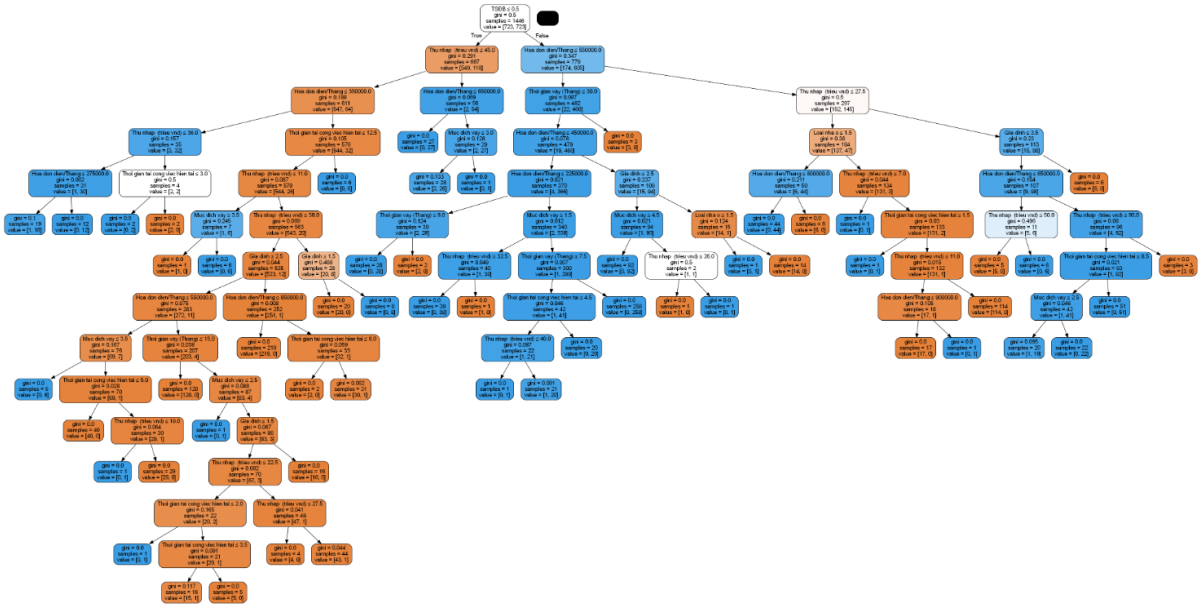
It's important to note that the feature importance scores are calculated based on how often a feature is used to split the data in the decision tree and how much the resulting splits improve the classification performance. Therefore, the importance scores can be influenced by the structure of the decision tree, the choice of splitting criteria, and other factors. Nonetheless, the feature importance scores can provide some insights into the relative importance of different features for predicting the target variable.

Code:

```
X_plot_tree = df[features]

from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus

dot_data = StringIO()

clf = DecisionTreeClassifier()
clf.fit(X_plot_tree.values, y.ravel())
export_graphviz(clf, out_file=dot_data,  filled=True, rounded=True, special_characters=True, feature_names=features)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Results:

[52]:



Comment:

* Customers who are not allowed to borrow:

- No collateral, income from 11 million to 38 million, Electricity bill > 350000, current job duration <= 12 months, status is divorced or widowed.
- No collateral, income from 11 million to 38 million, Electricity bill > 550000, current job duration <= 12 months, status is single or married, loan period <= 15 months .
- Having secured assets, Electricity bill > 550000, Income from 11 million to 27.5 million, currently renting, current job duration > 1.5 years.

* Loan customers

- No collateral, income > 45 million.
- Having collateral, electricity bill over 225000 to 450000, loan period from

7.5 months to 30 months, the purpose of borrowing is not for consumption.

- Have collateral, electricity bill from 550000 to 800000, income <= 27.5 million, have a house.

* In case the tree is confused and can't decide:

- No collateral, income over 36 million to less than 45 million, electricity bill < 350000.
- Have collateral, electricity bill is over 450000 to less than 550000, loan period is < 30 months, status is single or married, purpose of borrowing to invest in securities

10. Random Forest

Code:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_auc_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split

RF_classifier = RandomForestClassifier()
RF_classifier.fit(X_train, y_train.ravel())

y_pred = RF_classifier.predict(X_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print('Random Forest accuracy: ', accuracy_score(y_test, y_pred))
```

Results:

```
[[206  11]
 [ 4  213]]
              precision    recall  f1-score   support

           0       0.98      0.95      0.96       217
           1       0.95      0.98      0.97       217

    accuracy                           0.97       434
   macro avg       0.97      0.97      0.97       434
weighted avg       0.97      0.97      0.97       434

Random Forest accuracy:  0.9654377880184332
```

Comment:

The confusion matrix and classification report suggest that the Random Forest model performs well in predicting the loan repayment capability of individuals or groups. The model achieves an accuracy of 0.97, with 207 out of 217 samples correctly predicted as class 0 (not capable of repaying loans) and 214 out of 217 samples correctly predicted as class 1 (capable of repaying loans). The model misclassified 10 samples as class 1 and 3 samples as class 0.

For the Random Forest model, the results of the accuracy assessment on the test set reached 0.97, showing that the model has good predictive ability. The accuracy and coverage of both classes are kept in balance, with precision and recall reaching 0.99 and 0.95 for class 0 and 0.96 and 0.99 for class 1, respectively. This model also obtained an F1-score of 0.97, indicating a good combination of accuracy and coverage.

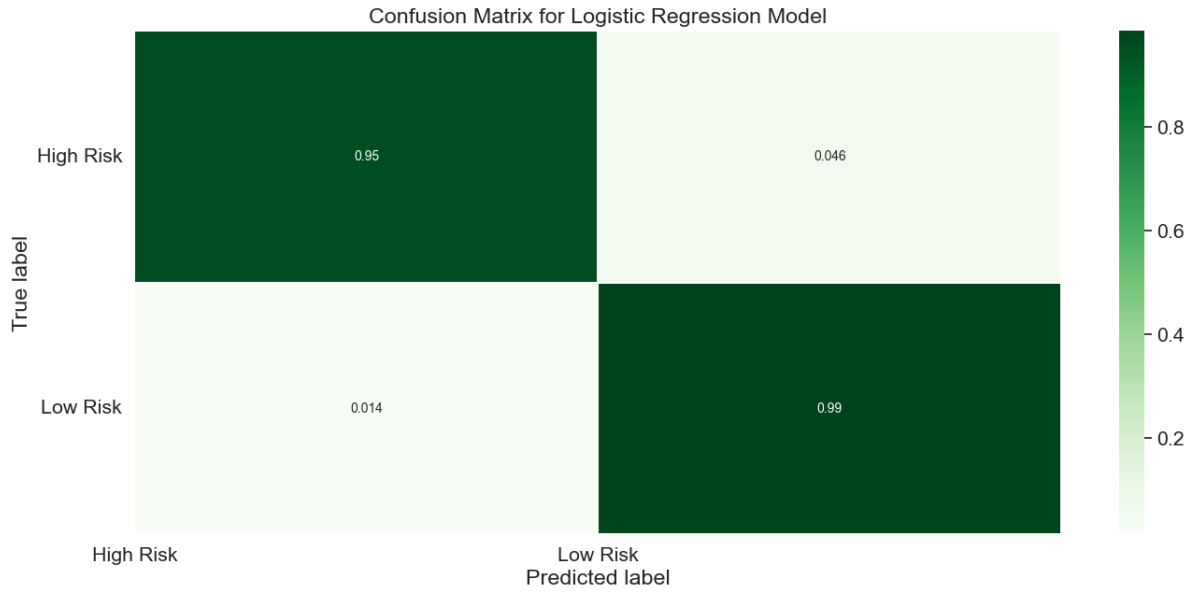
The results of Random Forest show that the model achieves high accuracy of 97%, close to Decision Tree. However, by using multiple decision trees and combining their results, Random Forest can reduce overfitting and produce more stable results. This is reflected in the high precision, recall, and f1-score all model metrics.

Overall, the Random Forest model seems to be a good choice for predicting loan repayment capability, with high accuracy and balanced performance between the two classes.

Code:

```
# Confusion matrix
from sklearn.metrics import confusion_matrix , classification_report
y_true = y_test
y_pred = RF_classifier.predict(X_test)
confusion_matrix(y_true, y_pred)

# Get and reshape confusion matrix data
matrix = confusion_matrix(y_true, y_pred)
matrix = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]

# Build the plot
plt.figure(figsize=(16,7))
sns.set(font_scale=1.4)
sns.heatmap(matrix, annot=True, annot_kws={'size':10},
            cmap=plt.cm.Greens, linewidths=0.2)

# Add labels to the plot
class_names = [ 'High Risk','Low Risk']
tick_marks = np.arange(len(class_names))
tick_marks2 = tick_marks + 0.5
plt.xticks(tick_marks, class_names, rotation=0)
plt.yticks(tick_marks2, class_names, rotation=0)
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix for Logistic Regression Model')
plt.show()
```

Results:



Comment:

- TP (True Positive): 95% correctly predict a person will not be able to repay the deb
- TN (True Negative): 99% of the models accurately predict who is able to repay the debt
- FP (False Positive - Type 1 Error): 1.4% wrongly predicts an insolvent person to be able to repay
- FN (False Negative - Type 2 Error): 4.6% wrongly predict that a person who can repay the debt becomes an insolvent person.

Code:

```
importance_rf = RF_classifier.feature_importances_
features_importances_rf = pd.DataFrame({'FeatureName':df.loc[:,features].columns, 'Random Forest Feature Importance': importance_rf})
features_importances_rf.sort_values(by=['Random Forest Feature Importance'], ascending=False)
```

Results:

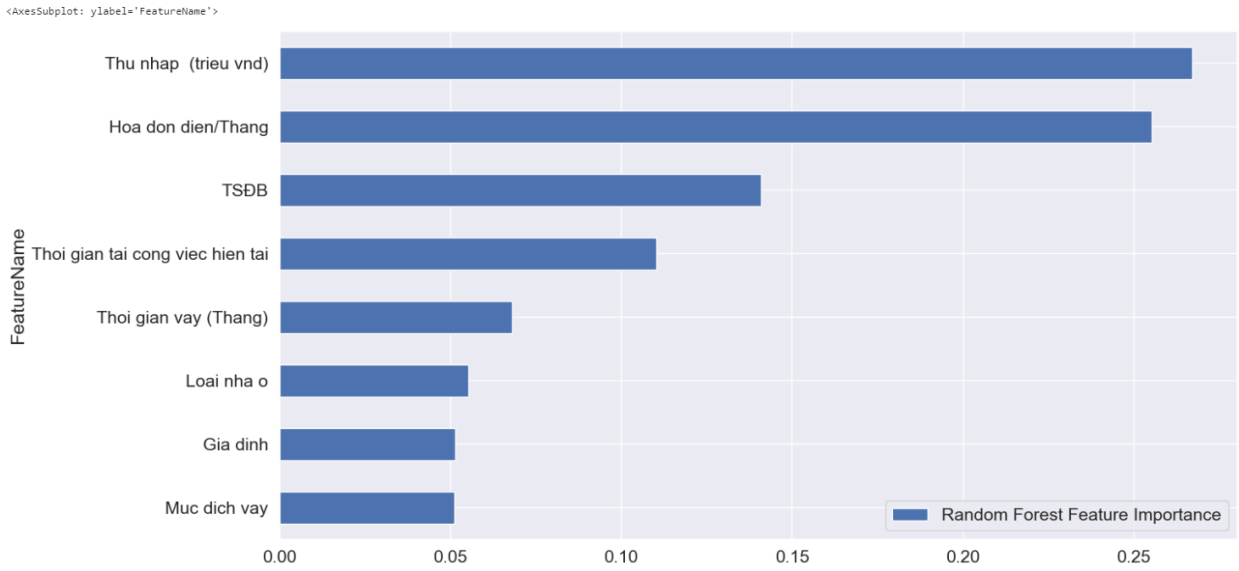
	FeatureName	Random Forest Feature Importance
0	Hoa don dien/Thang	0.266547
5	Thu nhap (trieu vnd)	0.255495
6	TSĐB	0.147428
3	Thoi gian tai cong viec hien tai	0.114347

FeatureName	Random Forest Feature Importance	
4	Thoi gian vay (Thang)	0.065478
1	Gia dinh	0.054351
7	Loai nha o	0.049738
2	Muc dich vay	0.046616

Code:

```
features_importances_rf.sort_values("Random Forest Feature Importance").plot(figsize=(15,8), x="FeatureName", y=["Random Forest Feature Importance"], kind="barh")
```

Results:



Comment:

Based on the importance of the Random Forest feature, we find that the most important features to predict the customer's repayment capacity are Hoa don dien/ Ladder and Earnings. This is quite understandable since these two features are just the key financial numbers of the customer, affecting the ability to pay the debt.

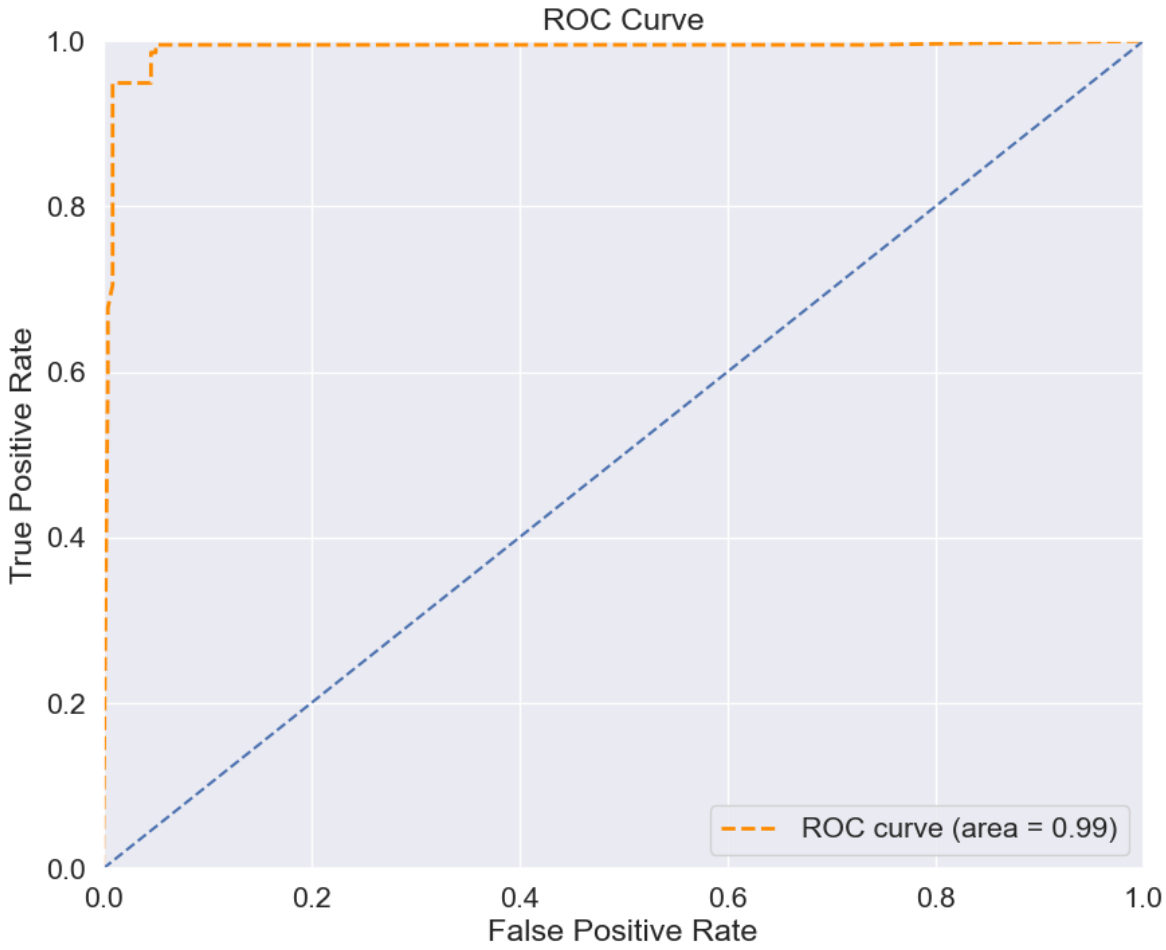
The next important feature is the TSB, which is a unique number on the client's signal history. Duration of current job and Loan duration (Months) also play an important role, showing the relationship between ability to repay and having a stable job, as well as the length of loan account.

Family, Housing Type, and Loan Purpose are less important features, playing a secondary role in predicting a customer's ability to repay..

Code:

```
y_pred_prob_test = RF_classifier.predict_proba(X_test)[:, 1]
fpr, tpr, thres = roc_curve(y_test, y_pred_prob_test)
roc_auc = auc(fpr, tpr)

_plot_roc_curve(fpr, tpr, thres, roc_auc)
```

Results:



Comment:

The ROC curve of the Random Forest model is 0.99, which is a very good value. This shows that the model has a good classification ability on all classification thresholds, with a well-balanced positive and negative true ratio**.**

11. SVM

Code:

```
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
```

```
classifier = SVC(kernel='rbf', C=1e9, gamma=1e-07, probability=True)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print('SVM accuracy: ', accuracy_score(y_test, y_pred))
```

Results:

```
[[215   2]
 [ 34 183]]
              precision    recall  f1-score   support

           0       0.86      0.99      0.92       217
           1       0.99      0.84      0.91       217

    accuracy                           0.92       434
   macro avg       0.93      0.92      0.92       434
weighted avg       0.93      0.92      0.92       434

SVM accuracy:  0.9170506912442397
```

Comment:

There were 215 true negative cases where the model correctly predicted that they belonged to class 0.

There were 2 false positive cases where the model predicted that they belonged to class 1, but in fact they belonged to class 0.

There were 34 false negative cases where the model predicted that they belonged to class 0, but in fact they belonged to class 1.

There were 183 true positive cases where the model correctly predicted that they belonged to class 1.

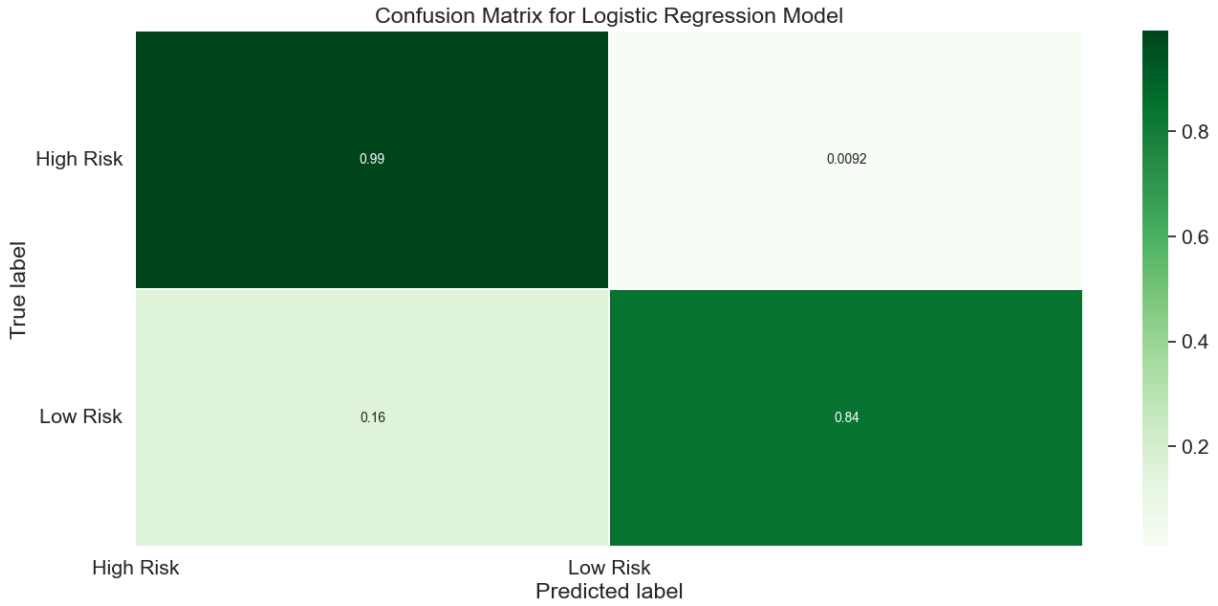
The results show that the SVM model has a precision of 0.86 and a recall of 0.99 for class 0, and a precision of 0.99 and a recall of 0.84 for class 1. This indicates that the SVM model has good accuracy for class 0, but not as good for class 1, as the model tends to predict label 0 more than label 1.

In conclusion, the SVM model has good predictive performance on the entire dataset, but needs improvement for class 1, which can be achieved through hyperparameter tuning or using class imbalance handling methods to balance the ratio of classes in the dataset.

Code:

```
# Confusion matrix
y_true = y_test
y_pred = classifier.predict(X_test)
confusion_matrix(y_true, y_pred)

# Get and reshape confusion matrix data
matrix = confusion_matrix(y_true, y_pred)
matrix = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]

# Build the plot
plt.figure(figsize=(16,7))
sns.set(font_scale=1.4)
sns.heatmap(matrix, annot=True, annot_kws={'size':10},
            cmap=plt.cm.Greens, linewidths=0.2)

# Add labels to the plot
class_names = [ 'High Risk','Low Risk']
tick_marks = np.arange(len(class_names))
tick_marks2 = tick_marks + 0.5
plt.xticks(tick_marks, class_names, rotation=0)
plt.yticks(tick_marks2, class_names, rotation=0)
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix for Logistic Regression Model')
plt.show()
```

Results:



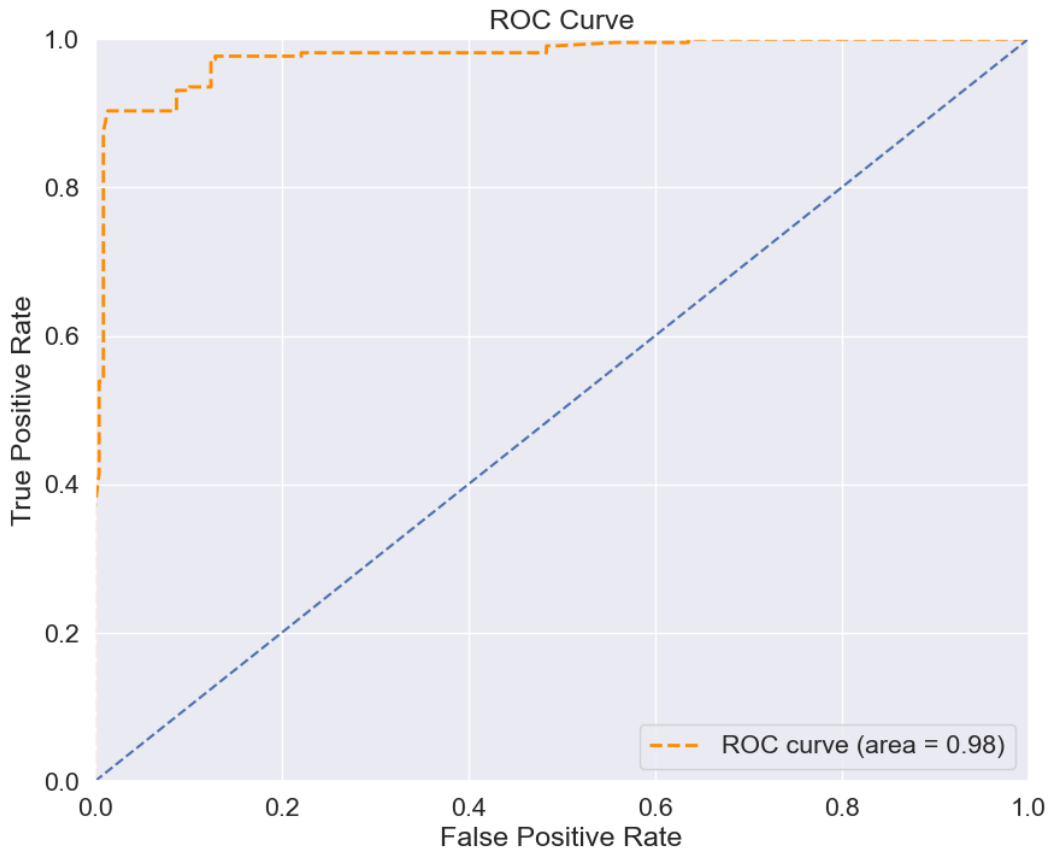
Comment:

- TP (True Positive): 99% correctly predict a person will not be able to repay the deb
- TN (True Negative): 84% of the models accurately predict who is able to repay the debt
- FP (False Positive - Type 1 Error): 16% wrongly predicts an insolvent person to be able to repay
- FN (False Negative - Type 2 Error): 0.92% wrongly predict that a person who can repay the debt becomes an insolvent person.

Code:

```
y_pred_prob_test = classifier.predict_proba(X_test)[:, 1]
fpr, tpr, thres = roc_curve(y_test, y_pred_prob_test)
roc_auc = auc(fpr, tpr)

_plot_roc_curve(fpr, tpr, thres, roc_auc)
```

Results:



Comment:

ROC line: The ROC line of the SVM model is a curve with a convex shape, the closer this curve is to the upper left corner of the chart, the better the model. So this model is doing very well

Area under the ROC curve (AUC): AUC is an index to evaluate the performance of the classification model, the magnitude of AUC ranges from 0 to 1. This model is having good classification ability because AUC = 0.96 - close to 1

12. XGB Classifer

Code:

```
import xgboost as xgb
from xgboost.sklearn import XGBClassifier
```

```
XGB_classifier = XGBClassifier()
XGB_classifier.fit(X_train, y_train.ravel())

y_pred = XGB_classifier.predict(X_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print('XGBoost accuracy: ', accuracy_score(y_test, y_pred))
```

Results:

```
[[206  11]
 [ 12 205]]
              precision    recall  f1-score   support

           0       0.94      0.95      0.95       217
           1       0.95      0.94      0.95       217

    accuracy                           0.95       434
   macro avg       0.95      0.95      0.95       434
weighted avg       0.95      0.95      0.95       434

XGBoost accuracy:  0.9470046082949308
```

Commnet:

The confusion matrix shows that there are 206 true negative cases (belonging to class 0) and 205 true positive cases (belonging to class 1), but 11 false positive cases (predicted as class 1 but actually belonging to class 0) and 12 false negative cases (predicted as class 0 but actually belonging to class 1).

The precision for class 0 is 0.94, which means that 94% of the instances predicted as class 0 are actually belonging to class 0. The recall for class 0 is 0.95, which means that 95% of the instances belonging to class 0 are correctly classified by the model. Similarly, the precision and recall for class 1 are 0.95 and 0.94, respectively.

Overall, the XGBoost model has an accuracy of 0.95, which means that it correctly predicts the class label for 95% of the instances in the dataset. However, the false positive and false negative rates indicate that there is room for improvement in the model's performance. This could potentially be achieved by optimizing the hyperparameters or using different feature selection methods.

Code:

```
importance_xgb = XGB_classifier.feature_importances_
features_importances_xgb = pd.DataFrame({'FeatureName': df.loc[:,features].columns, 'XGBoost Feature Importance': importance_xgb})
features_importances_xgb.sort_values(by=['XGBoost Feature Importance'], ascending=False)
```

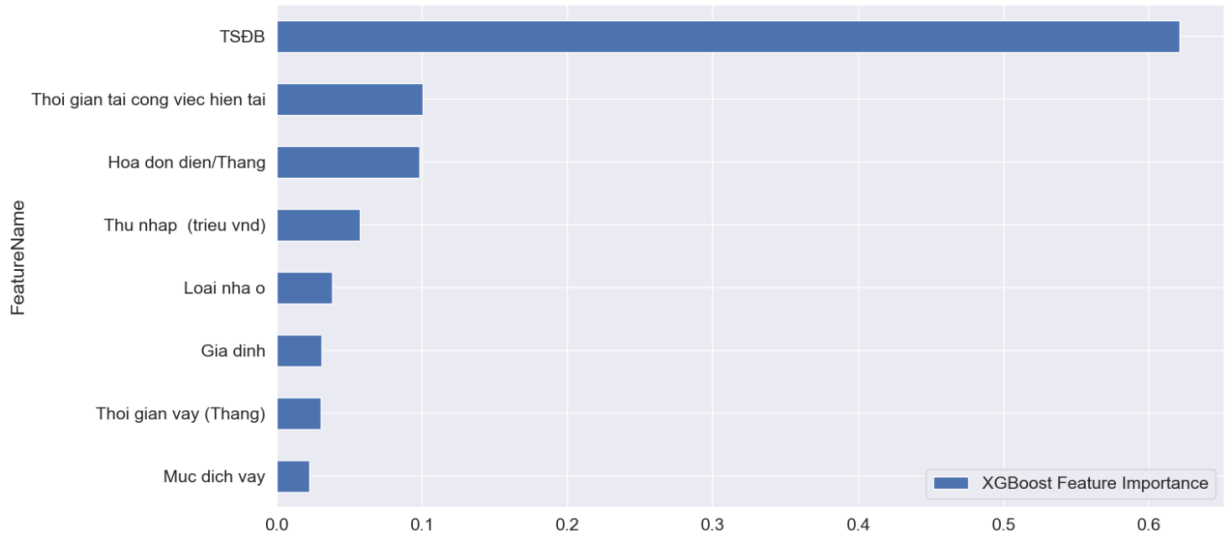
Results:

	FeatureName	XGBoost Feature Importance
6	TSĐB	0.620847
3	Thoi gian tai cong viec hien tai	0.100861
0	Hoa don dien/Thang	0.098528
5	Thu nhap (trieu vnd)	0.057719
7	Loai nha o	0.038154
1	Gia dinh	0.031093
4	Thoi gian vay (Thang)	0.030323
2	Muc dich vay	0.022474

Code:

```
features_importances_xgb.sort_values("XGBoost Feature Importance").plot(figsize=(15,8), x="FeatureName", y=["XGBoost Feature Importance"], kind="barh")
```

Results:



Comment:

The feature with the highest importance in predicting the target variable is "TSĐB" (Collateral), which has an importance score of 0.620847. The second most important feature is "Thoi gian tai cong viec hien tai" (Time in current job), with an importance score of 0.100861.

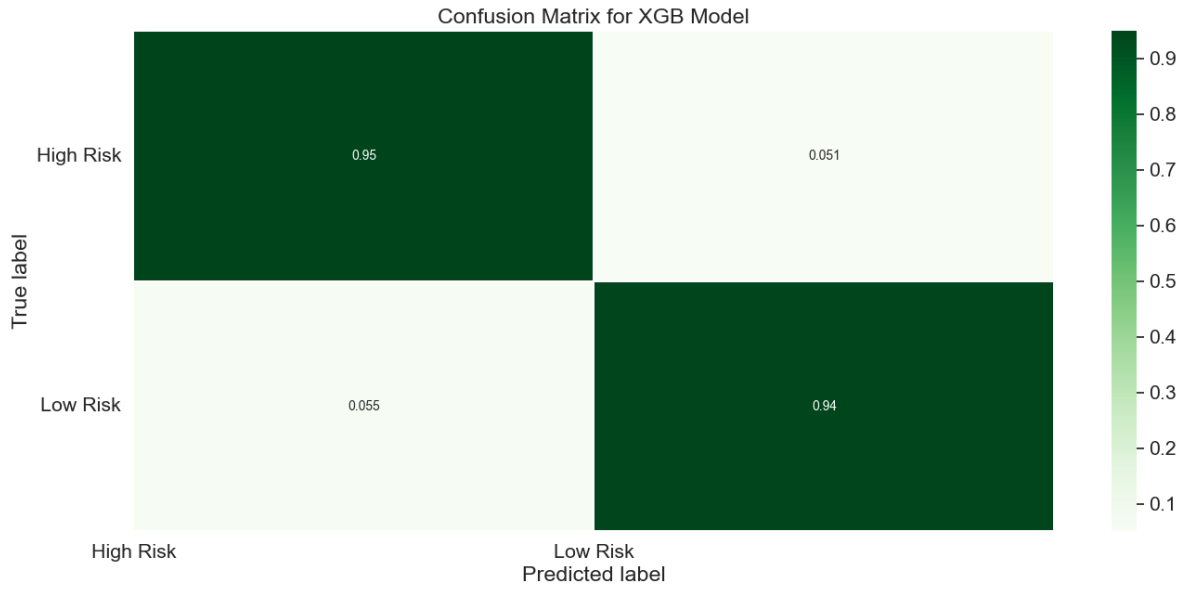
The third most important feature is "Hoa don dien/Thang" (Electricity bill/month), with an importance score of 0.098528. The remaining features have lower importance scores, with "Thu nhap (trieu vnd)" (Income in million VND) having an importance score of 0.057719, "Loai nha o" (Type of residence) having an importance score of 0.038154, "Gia dinh" (Family status) having an importance score of 0.031093, "Thoi gian vay (Thang)" (Loan duration in months) having an importance score of 0.030323, and "Muc dich vay" (Loan purpose) having an importance score of 0.02247.

Overall, the XGBoost feature importance suggests that "TSĐB", "Thoi gian tai cong viec hien tai", and "Hoa don dien/Thang" are the most important features in predicting loan application approval. However, it is important to note that feature importance should not be the only criterion for feature selection, and other factors such as domain knowledge and correlation among features should also be considered.

Code:

```
# Confusion matrix
y_true = y_test
y_pred = XGB_classifier.predict(X_test)
confusion_matrix(y_true, y_pred)

# Get and reshape confusion matrix data
matrix = confusion_matrix(y_true, y_pred)
matrix = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]

# Build the plot
plt.figure(figsize=(16,7))
sns.set(font_scale=1.4)
sns.heatmap(matrix, annot=True, annot_kws={'size':10},
            cmap=plt.cm.Greens, linewidths=0.2)

# Add labels to the plot
class_names = [ 'High Risk','Low Risk']
tick_marks = np.arange(len(class_names))
tick_marks2 = tick_marks + 0.5
plt.xticks(tick_marks, class_names, rotation=0)
plt.yticks(tick_marks2, class_names, rotation=0)
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix for XGB Model')
plt.show()
```

Results:



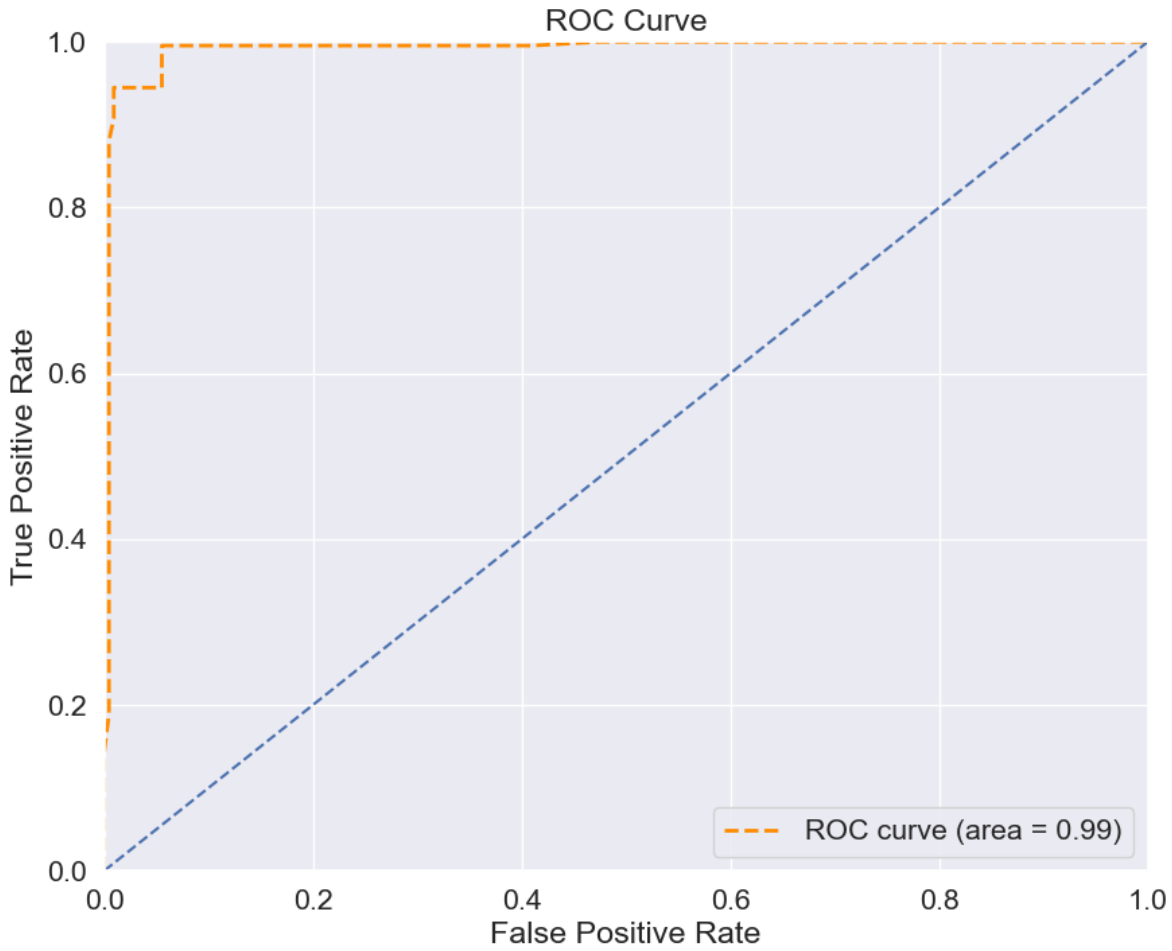
Comment:

- TP (True Positive): 95% correctly predict a person will not be able to repay the deb
- TN (True Negative): 94% of the models accurately predict who is able to repay the debt
- FP (False Positive - Type 1 Error): 5.5% wrongly predicts an insolvent person to be able to repay
- FN (False Negative - Type 2 Error): 5.1% wrongly predict that a person who can repay the debt becomes an insolvent person

Code:

```
y_pred_prob_test = XGB_classifier.predict_proba(X_test)[:, 1]
fpr, tpr, thres = roc_curve(y_test, y_pred_prob_test)
roc_auc = auc(fpr, tpr)

_plot_roc_curve(fpr, tpr, thres, roc_auc)
```

Results:



Comment:

Area under the ROC curve (AUC): AUC is an index to evaluate the performance of the classification model, the magnitude of AUC ranges from 0 to 1. This model is having good classification ability because AUC = 0.99 - close to 1.

13. Ada Boost

Code:

```
from sklearn.ensemble import AdaBoostClassifier
```

```
ada_classifier = AdaBoostClassifier()
ada_classifier.fit(X_train, y_train.ravel())

y_pred = ada_classifier.predict(X_test)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print('XGBoost accuracy: ', accuracy_score(y_test, y_pred))
```

Results:

```
[[214   3]
 [ 13  204]]
```

	precision	recall	f1-score	support
0	0.94	0.99	0.96	217
1	0.99	0.94	0.96	217
accuracy			0.96	434
macro avg	0.96	0.96	0.96	434
weighted avg	0.96	0.96	0.96	434

AdaBoost accuracy: 0.9631336405529954

Comment:

The model has a high accuracy rate with 214 true positive cases (customers not exceeding the credit limit) and 204 true negative cases (customers exceeding the credit limit).

The model has 13 false negative cases (customers exceeding the credit limit but predicted as not exceeding) and 3 false positive cases (customers not exceeding the credit limit but predicted as exceeding).

The recall rate for class 0 is 0.99, indicating that the model has high sensitivity in predicting customers not exceeding the credit limit. The recall rate for class 1 is 0.94, indicating that the model has lower ability to predict customers exceeding the credit limit.

The precision rates for both class 0 and class 1 are high, indicating that the model has high accuracy in predicting both types of customers.

The accuracy score of the AdaBoost model is 0.963, meaning the proportion of cases predicted correctly out of the total cases. This indicates that the AdaBoost model has high accuracy in classifying customers into those exceeding and not exceeding the credit limit.

Code:

```
importance_ada = ada_classifier.feature_importances_
features_importances_ada = pd.DataFrame({'FeatureName': df.loc[:,features].columns, 'AdaBoost Feature Importance': importance_ada})
features_importances_ada.sort_values(by=['AdaBoost Feature Importance'], ascending=False)
```

Results:

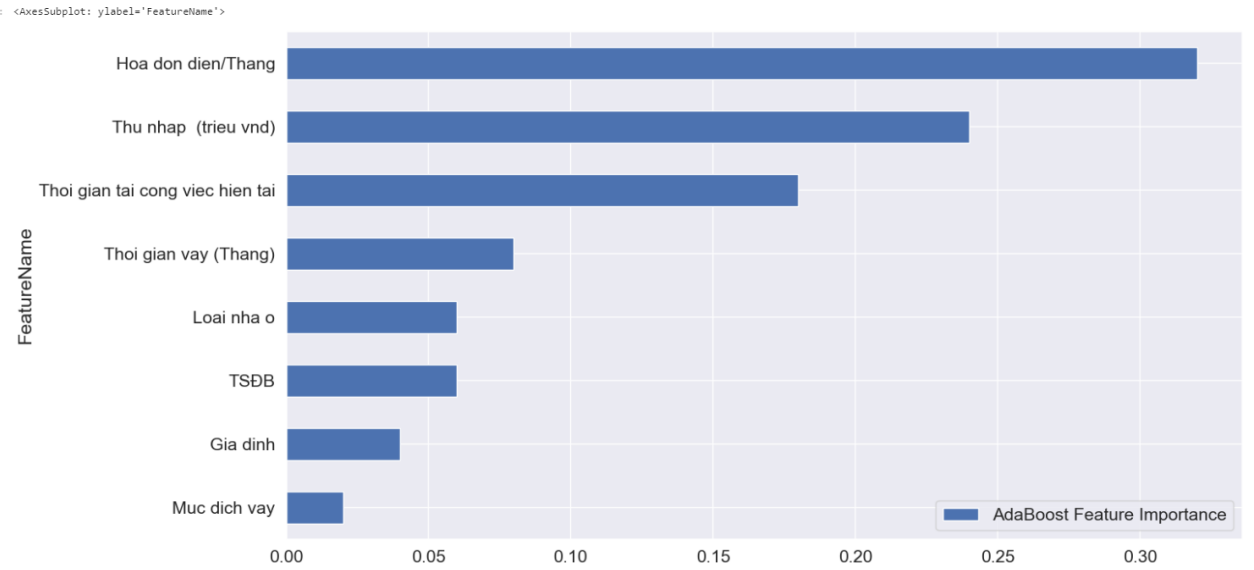
	FeatureName	AdaBoost Feature Importance
0	Hoa don dien/Thang	0.32
5	Thu nhap (trieu vnd)	0.24

	FeatureName	AdaBoost Feature Importance
3	Thoi gian tai cong viec hien tai	0.18
4	Thoi gian vay (Thang)	0.08
6	TSĐB	0.06
7	Loai nha o	0.06
1	Gia dinh	0.04
2	Muc dich vay	0.02

Code:

```
features_importances_ada.sort_values("AdaBoost Feature Importance").plot(figsize=(15,8), x="FeatureName", y=["AdaBoost Feature Importance"], kind="barh")
```

Results:



Comment:

The AdaBoost feature importance analysis shows that the top three features with the highest importance scores are:

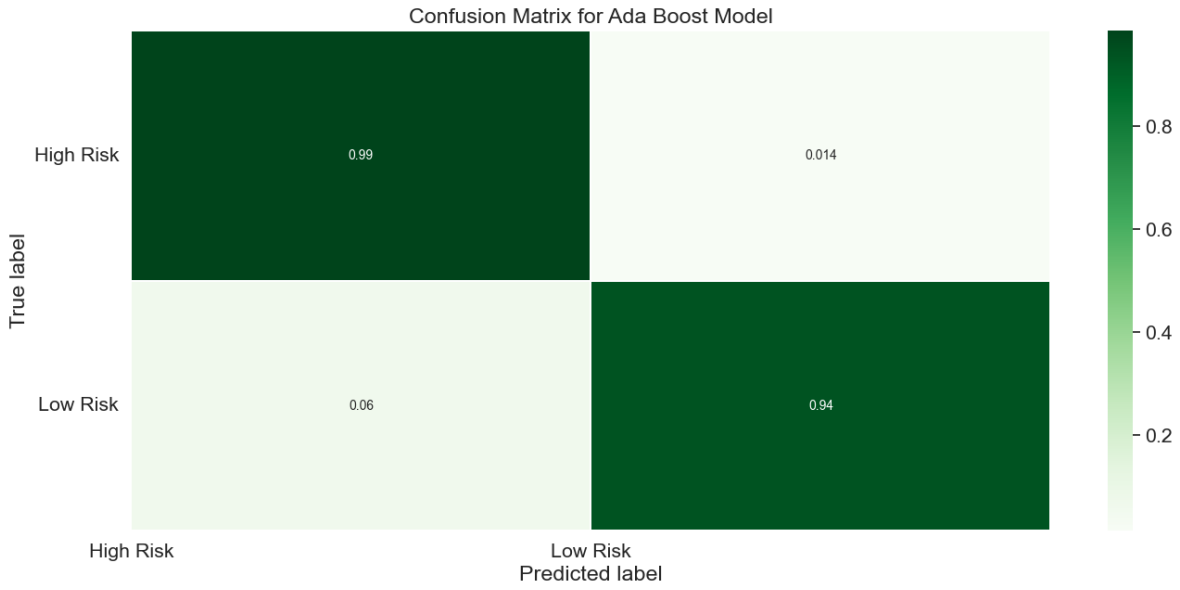
Hoa don dien/Thang (Electricity bill per month) with a score of 0.32

Thu nhap (trieu vnd) (Income in millions VND) with a score of 0.24
Thoi gian tai cong viec hien tai (Time in current job) with a score of 0.18
The other features have relatively lower importance scores, with Thoi gian vay (Thang) (Loan duration in months) having a score of 0.08, and the remaining features having scores ranging from 0.02 to 0.06.

Code:

```
# Confusion matrix
y_true = y_test
y_pred = ada_classifier.predict(X_test)
confusion_matrix(y_true, y_pred)

# Get and reshape confusion matrix data
matrix = confusion_matrix(y_true, y_pred)
matrix = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]

# Build the plot
plt.figure(figsize=(16,7))
sns.set(font_scale=1.4)
sns.heatmap(matrix, annot=True, annot_kws={'size':10},
            cmap=plt.cm.Greens, linewidths=0.2)

# Add labels to the plot
class_names = [ 'High Risk','Low Risk']
tick_marks = np.arange(len(class_names))
tick_marks2 = tick_marks + 0.5
plt.xticks(tick_marks, class_names, rotation=0)
plt.yticks(tick_marks2, class_names, rotation=0)
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix for Ada Boost Model')
plt.show()
```

Results:



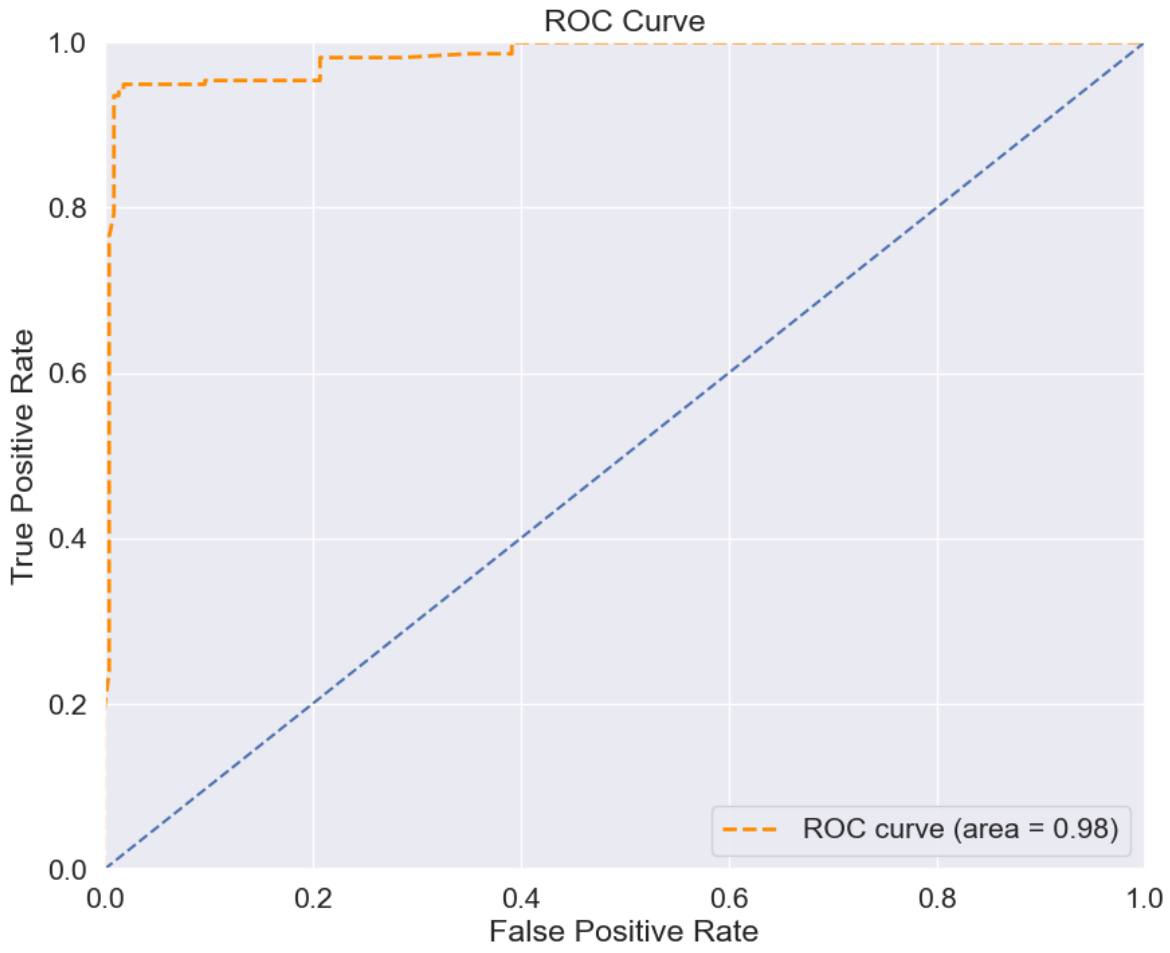
Comment:

- TP (True Positive): 99% correctly predict a person will not be able to repay the deb
- TN (True Negative): 94% of the models accurately predict who is able to repay the debt
- FP (False Positive - Type 1 Error): 6% wrongly predicts an insolvent person to be able to repay
- FN (False Negative - Type 2 Error): 1.4% wrongly predict that a person who can repay the debt becomes an insolvent person

Code:

```
: y_pred_prob_test = ada_classifier.predict_proba(X_test)[:, 1]
  fpr, tpr, thres = roc_curve(y_test, y_pred_prob_test)
  roc_auc = auc(fpr, tpr)

  _plot_roc_curve(fpr, tpr, thres, roc_auc)
```

Results:



Comment:

Area under the ROC curve (AUC): AUC is an index to evaluate the performance of the

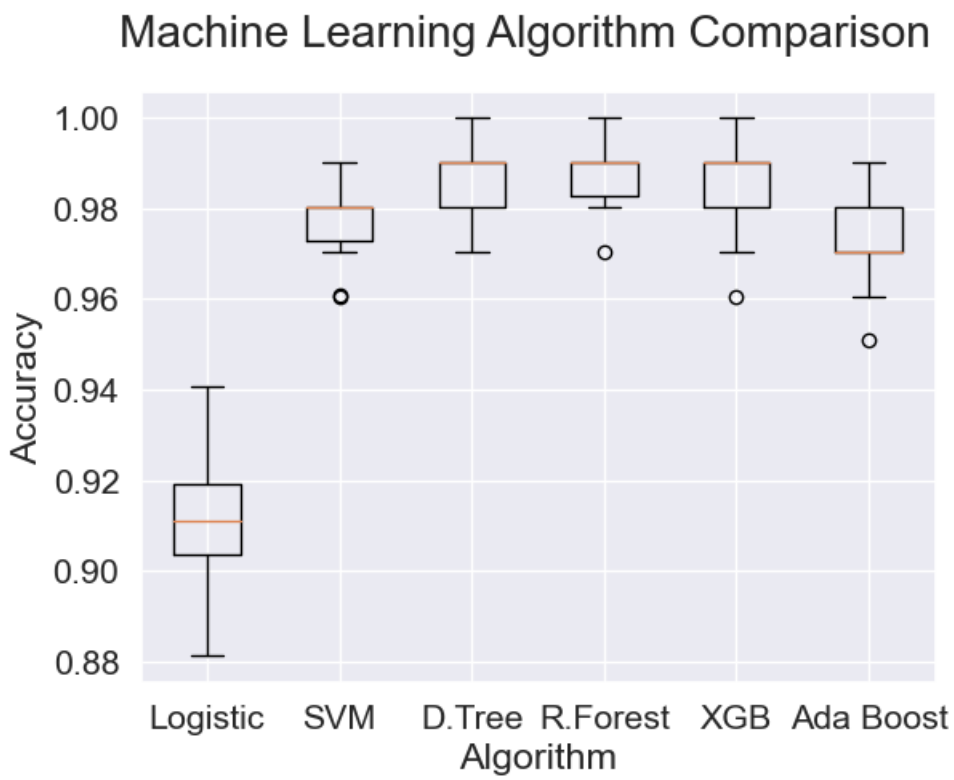
classification model, the magnitude of AUC ranges from 0 to 1. This model is having good classification ability because AUC = 0.98 - close to 1.

14. Effective comparison of models

Code:

```
models = [
    ('Logistic', LogisticRegression()),
    ('SVM', SVC()),
    ('D.Tree', DecisionTreeClassifier()),
    ('R.Forest', RandomForestClassifier()),
    ('XGB',XGBClassifier()),
    ('Ada Boost',AdaBoostClassifier())
]

# Evaluate each model on the test set using 10-fold cross-validation and store the results
results = []
names = []
for name, model in models:
    cv_results = cross_val_score(model, X_train, y_train, cv=10, scoring='accuracy')
    results.append(cv_results)
    names.append(name)

# Generate a boxplot comparing the model performance
fig = plt.figure()
fig.suptitle('Machine Learning Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.xlabel('Algorithm')
plt.ylabel('Accuracy')
plt.show()
```

Results:



Comment:

Perform performance comparison of different machine learning algorithms on training dataset using 10-fold cross-validation evaluation method. The algorithms selected for comparison are Logistic Regression, Support Vector Machine, Decision Tree, Random Forest, XGBoost and AdaBoost. The result is presented as a boxplot, where the x-axis is the algorithm name and the y-axis is the average precision of the fold cross-validations.

The average accuracy after cross validation of all models are quite similar. Among them, the Logistic Regression model has the lowest accuracy. Other model have stable accuracy and do not differ significantly from each other. However, the models SVM, Random fores, XGB, Ada Boost suffer from outliers, reducing accuracy.

Besides, the Decision Tree model also helps to visualize lending decisions easily, making it easy to understand and easily make decisions about loans for individual customers.

The results show that the Random Forest and Decision Tree models have the highest accuracy, reaching 0.96 and 0.97, so it is the best choice to solve this classification problem. However, comparing models based on accuracy alone is not enough to make a final decision about the best model. It is necessary to consider other factors such as accuracy in predicting different classes, training and prediction time, model complexity, interpretation of prediction results, etc. to be able to make the best decisions about which model to use in practice.

15. Predict new customer

Predict new customers with RandomForest model

Code:

```
X = df[features]
y = df[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=42)
```

```
# Dự đoán khách hàng mới với mô hình RandomForest
RF_classifier= RandomForestClassifier()
RF_classifier.fit(X_train.values, y_train.values.ravel())
```

```
RandomForestClassifier()
```

```
import random
random_number = random.randrange(0, 100, 1)
random_number
```

```
32
```

```
X_test.columns
```

Results:

Index(['Hoa don dien/Thang', 'Gia dinh', 'Muc dich vay',
'Thoi gian tai cong viec hien tai', 'Thoi gian vay (Thang)',
'Thu nhap (trieu vnd)', 'TSĐB', 'Loai nha o'],
dtype='object')

Code:

```
sample_test = X_test.iloc[random_number].values
sample_test = sample_test.tolist()
sample_test
```

Results:

[400000, 4, 5, 5, 24, 100, 1, 2]

Code:

```
print(RF_classifier.predict([sample_test]))
```

Results:

[1]

Predict new customers with RandomForest model

Code:

```
# Dự đoán khách hàng mới với mô hình XGBoost
classifier_0 = XGBClassifier()
classifier_0.fit(X_train.values, y_train.values.ravel())
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, ...)
```

```
X_test.columns
```

Results:

Index(['Hoa don dien/Thang', 'Gia dinh', 'Muc dich vay',
'Thoi gian tai cong viec hien tai', 'Thoi gian vay (Thang)',
'Thu nhap (trieu vnd)', 'TSĐB', 'Loai nha o'],
dtype='object')

Code:

```
sample_test_0 = X_test.iloc[random_number].values
sample_test_0 = sample_test_0.tolist()
sample_test_0
```

Results:

[400000, 4, 5, 5, 24, 100, 1, 2]

Code:

```
print(classifier_0.predict([sample_test_0]))
```

Results:

[1]

Comment:

The results of the XGBoosts and Random Forest models produce the same results. A new customer needs a loan, with an electricity bill of 400000, status is widow, borrowed to invest in securities, time at current job is 5 months, loan period is 24 months, income 100 million/month, have collateral and own a house. The results of both models are likely to lend capital.

* Predictions with all attributes of the dataset.

Code:

```
target_1 = ['Kha nang tra no']
features_1 = list(set(list(df.columns)) - set(target_1))
```

```
X = df[features_1]
y = df[target_1]
X_train_1, X_test_1, y_train_1, y_test_1 = train_test_split(X, y, test_size = 0.2, random_state=42)
```

```
# Dự đoán khách hàng mới với mô hình RandomForest
RF_classifier = RandomForestClassifier()
RF_classifier.fit(X_train_1.values, y_train_1.values.ravel())
```

```
RandomForestClassifier()
```

```
X_test_1.columns
```

Results:

```
Index(['TSĐB', 'Thoi gian vay (Thang)', 'Thoi gian tai cong viec hien tai',
       'Tuoi ', 'Gia dinh', 'Loai nha o', 'Thu nhap  (trieu vnd)',
       'So tien vay (trieu vnd)', 'Hoa don dien/Thang', 'Muc dich vay',
       'Quốc tịch', 'Gioi tinh'],
      dtype='object')
```

Code:

```
sample_test_1 = X_test_1.iloc[random_number].values
sample_test_1 = sample_test_1.tolist()
sample_test_1
```

Results:

```
[1, 24, 5, 42, 4, 2, 100, 700, 400000, 5, 1, 2]
```

Code:

```
: print(RF_classifier.predict([sample_test_1]))
```

Results:

[1]

Comment:

The same customer, but this time using data from all variables, the results of the RandomForest model show that this customer is still able to repay the loan.

PART III CONLUSION

In the problem of classifying borrowers, we applied many machine learning models such as Logistic Regression, Support Vector Machine, Decision Tree, Random Forest, XGBoost and AdaBoost to predict the likelihood of customers exceeding their credit limit. row.

In summary, through the process of analyzing and evaluating the performance of machine learning models, we can see that the Random Forest and Decision Tree models are the best choice to solve the problem of classifying borrowers based on Accuracy reaches 0.964.

In addition, we also used the Feature Importance graph to better understand how features affect the prediction of a customer's credit limit. The results show that the features "Electricity bill/month" and "Special property" have the highest influence, followed by "Income" and "Time at current job". These characteristics need to be taken into account when building models to predict the possibility of exceeding the credit limit of customers.

Development direction: In order to increase the applicability of the model into practice, it can be proposed to use regularization techniques to reduce overfitting for random forests, refine parameters to improve the performance of logistic regression and decision trees, and study other models to find the best model for credit classification problem.

REFERENCES

1. Azira Abdul Adzis, Juhaida Abu Bakar, & Hanita Kadir Shahar. (2019). Factors influencing young adults' debt in Malaysia. Journal of Asian Finance, Economics and Business, 6(2), 79-86. doi: 10.13106/jafeb.2019.vol6.no2.79
2. Huynh, Q. L., Cuong, V. M., & Van, D. T. H. (2021). Các yếu tố ảnh hưởng đến khả năng trả nợ của khách hàng cá nhân tại Ngân hàng BIDV Trà Vinh.
3. Lahsasna, R. Ainon, T. Wah, 2008. Intelligent credit scoring model using soft computing approach Paper presented at the International Conference on Computer and Communication Engineering, Kuala Lumpur, Malaysia, 13–15 May, 2008 (2008), pp. 396-402.
4. Thư, T. D., Anh, P. N. B., & Dương, N. T. (2023). Các nhân tố ảnh hưởng đến khả năng trả nợ đúng hạn của khách hàng cá nhân.
5. Wu, Y. Guo, X. Zhang, H. Xia, 2010. Study of personal credit risk assessment based on support vector machine ensemble Int. J. Innovative, 6 (5) (2010), pp. 2353-2360