VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY

UNIVERSITY OF ECONOMICS AND LAW

FINAL REPORT

CREDIT RISK MODELING IN R/PYTHON

YEAR 2023

LOAN PREDICTION USING MACHINE LEARNING MODEL

Study program : **K20414C_ Fintech**
Course : **Credit risk modeling in R/Python**
Full Name : **Phạm Thị Kim Phượng**
Class : **K20414C**
ID student : **K204141927**
Email : **phuongptk20414c@st.uel.edu.vn**

VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY

UNIVERSITY OF ECONOMICS AND LAW

FINAL REPORT

CREDIT RISK MODELING IN R/PYTHON

LOAN PREDICTION USING MACHINE LEARNING MODEL

DECLARATION

I hereby declare that the report "Loan Prediction using Machine Learning Model" is the result of my work under the guidance of Dr. Pham Thi Thanh Xuan within the framework of credit risk modeling in the R/Python module.

TABLE OF CONTENTS

DECLARATION 1

LIST OF TABLES 3

LIST OF FIGURES 4

INTRODUCTION 5

I. Introducing the problem statement 5

II. Model 6

III. Dataset Description 9

CONTENTS 13

I. Implementation process 13

II. Step 1: Import libraries and load data 13

III. Step 2: Checking the data 14

IV. Step 3: Data visualization 20

V. Step 4: Pre-processing 31

VI. Step 5: Building the Models 32

1. Logistic regression 33

2. Decision tree 40

3. Random Forest 47

4. XGBoost Classifier 53

5. Ada Boost 58

6. KNN Classifier 64

VI. Compare models 69

VII. Predict new customer 73

CONCLUSION 75

REFERENCES 76

LIST OF TABLES

	Name of table	Page
1	Table 1: Head of data set	13
2	Table 2: Non-Null Count of each column	14
3	Table 3: Head of data set after delete id column	15
4	Table 4: Missing values	16
5	Table 5: Data set after drop duplicates rows	17
6	Table 6: Unique values of each column	17
7	Table 7: Descriptive statistics table of numerical features	19
8	Table 8: Descriptive statistics table of categorical features	19
9	Table 9: Data set after converting categorical features to numeric	24
10	Table 10: Prediction results of Logistic Regression	36
11	Table 11: Logit Regression Results	37
12	Table 12:Logistic Regression Feature Importance	38
13	Table 13: Decision Tree Feature Importance	43
14	Table 14: Random Forest Feature Importance	51
15	Table 15: XGBoost Feature Importance	56
16	*Table 16: AdaBoost Feature Importance*	62
17	Table 17: The model's indexs comparison table	70
18	Table 18: Head of new data	73
19	Table 19: predicted results with XGB Classifier	74

LIST OF FIGURES

	Name of figure	Page
1	Figure 1: Count of each gender	20
2	Figure 2: Count of each purppose	21
3	Figure 3: Count of each marital status	22
4	Figure 4: Count of each education level	22
5	Figure 5: Count of each collateral	23
6	Figure 6: Histogram of numerical features	23
7	Figure 7: Target distributions	26
8	Figure 8: Ability to repay with gender	27
9	Figure 9: Ability to repay with purpose of loan	28
10	Figure 10: Ability to repay with marital status	29
11	Figure 11: Ability to repay with education level	30
12	Figure 12: Ability to repay with collateral	31
13	Figure 13: Results on the training set of Logistic Regression	34
14	Figure 14: Results on the test set of Logistic Regression	36
15	Figure 15: Logistic Regression Feature Importance	39
16	Figure 16: ROC Curve for Logistic Regression model	40
17	Figure 17: Results on the training set of Decision Tree	41
18	Figure 18: Results on the test set of Decision Tree	43
19	Figure 19: Decision Tree Feature Importance	44
20	Figure 20: ROC curve of Decision Tree model	45
21	Figure 21: Plot tree before pruning	46
22	Figure 22: Plot tree after pruning	47
23	Figure 23: Results on the training set of Random Forest	49
24	Figure 24: Results on the test set of Random Forest	50
25	Figure 25: Random Forest Feature Importance	52
26	Figure 26: ROC curve of Random Forest model	53
27	Figure 27: Results on the training set of XGBoost	54
27	Figure 28: Results on the test set of XGBoost	56
29	Figure 29: XGBoost Feature Importance	57
30	Figure 30: ROC curve of XGBoost model	58
31	Figure 31: Results on the training set of Ada Boost	60
32	Figure 32: Results on the test set of Ada Boost	61
33	Figure 33: AdaBoost Feature Importance	63
34	Figure 34: ROC curve of Ada Boost model	64
35	Figure 35: K Values of KNN	65
36	Figure 36: Results on the training set of KNN	66
37	Figure 37: Results on the test set of KNN	68
38	Figure 38: ROC curve of KNN model	69
39	Figure 39: Machine learning algorithm comparison with AUC	72

INTRODUCTION

I. Introducing the problem statement

With the continuous development of the Internet in recent years, the integration of technology and finance has deepened, resulting in tremendous changes in the financial industry. The demand for various credit businesses is increasing, necessitating the establishment of a reasonable and reliable risk assessment model. In the past, banks used credit appraisal tools in the past to better understand their customers, lowering the bank's risk level in the credit management process. A credit appraisal's content includes the 5Cs: Character, Capacity, Collateral, Capital, and Conditions. And weighed as a reference for whether to lend to the user or decide whether to issue loans, this method of relying on subjective judgment is obviously inefficient, and the evaluation is very dependent on the subjective judgment ability of the risk control personnel from the perspective of internal control of the company. With the development of big data on the Internet, the traditional credit scoring model's application is severely limited. In the face of tens of thousands or even hundreds of thousands of users applying for loans, the bank needs to adopt various machine learning methods to reduce manual participation in the monitoring and testing process and use automated methods to improve credit decision making.

In recent years, there have been many research papers applying many machine learning methods in credit assessment. Aledanjro (2016) found that the traditional logistic regression did not perform as well as the previous ones when the various characteristic variables exhibited complicated nonlinear relationships. Though the Logistic Regression model may not be as good as the machine learning model in terms of prediction accuracy, it has a strong advantage in terms of variable interpretability and stability. Therefore, some scholars have improved the Logistic regression and applied it to the borrower's default behavior prediction. In 1985, for the first time, Makowski used the decision tree approach in the field of personal credit assessment. Carter proved in 1987 that the decision tree approach has a high classification accuracy in the field of personal credit assessment. Henning et al (2019) studied the factors that affect successful agricultural loan applications using logistic regression model with data from loan applications of 128 farmers in South Africa. Results indicate that loan applications that are more likely to be successful are older more experienced farmers, who can provide sufficient collateral, have more years of business with the credit provider, have an acceptable credit history, request smaller loan amounts, have lower interest expense ratio, higher production cost ratios, and have diversification strategies. According to the research paper Credit risk management in commercial banks by Konovalova et al (2016). When lending to

individuals (retail clients) the most significant factors affecting the value of the credit risk of a bank are the average income of the borrower, the loan amount, and the loan term. Research paper The prediction of loan behavior with machine learning models for secure banking by Anand et al (2022) employed multiple classification algorithms for bank loan default prediction. According to the findings, the most crucial parameters employed by their model for forecasting whether or not a customer will default on payment are the customer's employment or job experience in years and debt income.

Based on previous research papers, I selected variables and built many different models to predict the repayment ability of customers from loan applications to help banks make quick and effective lending decisions.

II. Model

Logistic Regression

Logistic regression is a widely applied statistical model that is a suitable approach for binary classification problems (Kim, Kim, Cho, Kim, & Hyun, 2018; Siddique, Akhtar, Lee, Kim, & Kim, 2017). Loans can be classified as default and no default, which matches the distribution of the logistic model.

Logistic regression provides predicted probabilities, allowing for a more nuanced understanding of the relationship between variables. It can estimate the importance of each independent variable in predicting the outcome using the magnitude of the corresponding coefficients. However, Logistic regression assumes a linear relationship between the independent variables. It may not perform well when the relationship is nonlinear or complex.

Decision trees

Decision Trees are intuitive to visualize the decision. It is a type of non-parametric supervised learning method and decision support tool used for regression and classification. Based on the dataset, it can identify and derive a sine curve with a series of if-then-else decision rules. The model is fitter if the tree is deeper since there are more complex if-then-else decision rules.

It has a flowchart structure that mimics the activity of human thinking. This algorithm uses the tree representation to solve the problem, internal nodes in the tree to represent an attribute and leaf nodes to represent class labels. We start from the root of the tree, separating the sample into groups

of homogeneous sets according to most essential splitters of input variables. Repeat this process until leaf nodes in all the branches of the tree are found. Therefore, it is easy to visualize, understand, and interpret.

This method is helpful in variable screening and feature selection, and it has the capability of accurately handle with high dimensional data with requiring little users' data preparation. Compared to other methods, this method requires little data cleaning since it is not affected by outliers. Furthermore, this method can deal with both quantitative and qualitative data, and this property is superior to other methods because other classifiers always require single attribute of the data. However, there are weaknesses exist, such as over fitting problem, which can be eliminated by using random forest. The reason the decision tree is likely to have the problem of over-fitting when we don't set a limitation of maximum depth is because the tree can keep growing until it has exactly one leaf node for every single observation, so the tree is overly complex and doesn't classify the datasets well. This method doesn't have stability because even small variation in the datasets might lead to a completely different decision tree being generated.

Random forests

Random forest is the most popular algorithm, namely, it assembles a large amounts of decision trees from training dataset. Each decision tree represents a class prediction, this method collects the votes from these decision trees and the class with most votes is considered as the final class.

The underlying reason this method outperforms than decision tree is that many uncorrelated decision trees can protect each other from individual errors to derive the ensemble predictions, thus over fitting problem can be reduced. It also has other strengths such as: the model can run efficiently on large number of datasets, and it can estimate which variables are significant in the classification as well. Random forest outperforms than linear models because it can catch non-linear relationship between the object and the features. The flaw of the method is that it can't work with sparse features because the decision trees are building blocks.

XGBoost Classifier

The XGBoost (Extreme Gradient Boosting) classifier is a powerful machine learning algorithm that has gained popularity due to its high performance and flexibility. XGBoost is known for its

excellent predictive accuracy and has often been the winning algorithm in various machine learning competitions.

XGBoost can capture complex patterns and relationships in the data, including nonlinear relationships and interactions between variables. It supports a variety of data types, including numerical and categorical variables. When dealing with large datasets or a large number of features, XGBoost can be computationally expensive. XGBoost models can become complex and difficult to interpret, especially when deep tree structures are used.

AdaBoost Classifier

Ada-boost or Adaptive Boosting is one of ensemble boosting classifier proposed by Yoav Freund and Robert Schapire in 1996. It combines multiple classifiers to increase the accuracy of classifiers. AdaBoost is an iterative ensemble method. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier. The basic concept behind Adaboost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations.

It can handle both numerical and categorical features and automatically determine their importance during the training process. However, AdaBoost is sensitive to noisy data and outliers, as it assigns higher weights to misclassified samples in each iteration and computationally expensive training.

KNN Classifier

KNN, a non-parametric and lazy learning method, is explored for regression and classification. The model structure is depended on the dataset, and there is no assumption for data distributions, therefore, it is non-parametric. This property is useful because real world datasets do not always adhere to theoretical assumptions. Lazy algorithm implies that all training datasets are used in the testing phase.

This classifier is simple, and intuitive to understand. Consequently, it's widely used for classification because of low calculation time and easy of interpretation. KNN has the following working mechanism: compute distance, find the closest neighbors, and vote for labels. In this method, K is the number of nearest neighbors, and it is a controlling variable for this method. Each dataset has its own optimal number of K. If K is small, the noise would greatly influence the result, whereas if K is large, it would be expensive to compute. Research reveals that a small K is more

flexible with high variance and low bias while a large K is not flexible with lower variance but higher bias.

The pros of the method are that the training phase of KNN is much faster than other classifiers, and it is pretty intuitive and simple since it doesn't contain any assumptions. KNN can be useful in the case of multi-class problems and regression problems. The weakness of this method is that the testing phase of K-nearest neighbor classification is costlier and slower since it requires large memory for storing the entire training datasets. What's more, KNN is sensitive to magnitudes and outliers because it directly chooses neighbors on the basis of distance. It is also not suitable for large dimensional data, and dimensions require reduction to upgrade performance. Additionally, this method is unable to deal with the missing value problem as well.

III. Dataset Description

Building a model to forecast the customer's solvency price using a decision tree model based on a self-generated data set of customers' borrowing information.

The target variable is "kha nang tra no", which refers to a borrower's ability to make payments on a loan according to the agreed-upon terms. "kha nang tra no" variable includes "0. Trả nợ đúng hạn" (on-time repayment) and "1. Trả nợ trễ hạn" (late repayment).

There are 12 features of the data set:

Feature 1: "Id" is the customer ID. A unique identifier for the customer.

Feature 2: "gioi tinh" is the customer's gender. (1 = female, 2 = male)

Feature 3: "thu nhap" is the customer's monthly income (VND unit). Income is one of the most important variables that affects a borrower's ability to repay a loan. Generally, the higher the borrower's income, the greater their ability to repay the loan.

Feature 4: "tien dien/ thang" is the customer's monthly electric bill (VND unit). The customer's monthly electric bill is an ongoing expense that they need to pay every month, and this can impact their disposable income. If the electric bill is particularly high, it can reduce the amount of money the customer has left over each month to put towards loan repayments. This can make it more difficult for them to make payments on time or in full.

Feature 5: “so tien vay” is loan amount in VND. The loan sum has a direct impact on the borrower's monthly payment. Larger monthly payments will result from a larger loan amount, making it more difficult for the borrower to manage their budget and meet their payment responsibilities.

Feature 6: “muc dich vay” is the purpose of the loan. It includes:

- “1. Vay tiêu dùng”: customer loan. These loans are generally considered to be the least risky as they are typically used to finance smaller purchases like appliances or furniture.
- “2. Vay xây nhà”: housing loan. Because housing loans are frequently secured by the value of the property, these loans are usually regarded as less risky than other types of loans. However, if the housing market crashes, the borrower may end up owing more than the property's worth, making repayment difficult.
- “3. Vay mua ô tô”: car loan. Car loans are frequently secured by the asset being bought. (i.e., the car). However, cars depreciate in value much more rapidly than houses, which can make it more difficult to repay the debt if the borrower needs to sell the car.
- “4. Vay học tập”: education loan. While these loans are typically used to fund education, they can still be risky as there is no guarantee that the borrower will be able to find a job with a high enough salary to repay the loan.
- “5. Vay đầu tư chứng khoán”: Securities investment loan. These loans are regarded as the riskiest, as they are often linked to the stock market or other fluctuating financial markets. If the borrower's assets do not perform well, they may be unable to repay the loan.

Feature 7: “gia dinh” is the customer' s marrital status. It includes:

- “1. Đã kết hôn”: Married borrowers who are married and have a dual income household may be able to repay a loan more easily because they have more income to draw from. However, if one spouse is not employed, or if the couple has a high debt-to-income ratio, it could make it more difficult to repay a loan.
- “2. Độc thân”: Single borrowers may have less income to draw from and may have to rely solely on their own income to repay a loan. This can make it more difficult to qualify for a loan or to make timely payments.
- “3. Ly hôn”: If a borrower is divorced, their ability to repay a loan may be impacted by factors such as alimony or child support payments, legal fees, and division of assets. Depending on the circumstances, this factors could either improve or worsen the borrower's ability to repay.

- “4. Góa”: Widow borrowers may have experienced a loss of income, particularly if their spouse was the primary earner. They may also be dealing with financial issues related to life insurance, pension, or Social Security benefits. This can affect their ability to repay a loan.

Feature 8: “so nam lam viec” is the years in current job. It is a categorical variable that indicates how many years the customer has been in their current job. If a borrower has a steady job with a consistent income, lenders are more likely to view them as a lower risk borrower who is able to repay the loan.

Feature 9: “tuoi” is the customer’s age. The age of the borrower may be linked to their wage level. For example, a younger borrower may be in the early stages of their job and have not yet achieved their maximum earning potential, whereas an elderly borrower may be nearing retirement and have a reduced income. A reduced salary may make loan repayment more difficult because there is less money accessible for loan payments.

Feature 10: “thoi gian vay” is the term of loan in months. A loan's term refers to the length of time the borrower is expected to repay the debt. The term of a loan can have an impact on the borrower's capacity to repay it. The lower the monthly payment, the longer the loan period. This can help the borrower manage their budget and meet their payment responsibilities. A longer loan term, on the other hand, may imply paying more in interest over the life of the loan. Longer-term loans generally have higher interest rates because the lender assumes more risk by extending the loan over a longer period of time. Furthermore, longer-term loans will have higher total costs because the borrower will pay more interest over the course of the loan.

Feature 11: “trinh do hoc van” is the customers education. It is a categorical variable and includes:

- “1. Sau đại học”
- “2. Đại học”
- “3. Cao đẳng”
- “4. Trung cấp”
- “5. THPT”

The level of education is frequently related to the level of income. Individuals with greater degrees of education may have stable jobs with consistent income, increasing their ability to repay a loan. On the other hand, higher-educated people may have a better grasp of personal finance, budgeting,

and financial planning, which can help them manage their money and make loan payments on time.

Feature 12: “tai san dam bao” is the collateral for the loan.

- 1. Có đủ TSĐB của bản thân": Have sufficient collateral of themselves
- "2. Có đủ TSĐB được bảo lãnh": Have sufficient collateral to be guaranteed
- "3. Có 1 phần TSĐB": Have a part of collateral
- "4. Không có TSĐB": No collateral

Collateral is an asset that a borrower pledges as security for a loan. Collateral can affect a borrower's ability to repay a loan. Collateral can affect a borrower's ability to get approved for a loan and the interest rate they receive. Loans that are secured by collateral are often viewed as less risky by lenders, as they have the ability to seize the collateral if the borrower defaults on the loan.

CONTENTS

I. Implementation process

Step 1: Import libraries and load data

Step 2: Checking the data

Step 3: Data visualization

Step 4: Data pre-processing

Step 5: Building the models

Step 6: Compare models

Step 7: Predict new customer

II. Step 1: Import libraries and load data

In this section I will import libraries load the data from the excel file.

Importing the required Python libraries and dataset:

```
# import libraries
import pandas as pd
import seaborn as sns
sns.set()
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

# import data
data = pd.read_excel("/content/drive/MyDrive/HỌC TẬP/NĂM 3/HK2/mô hình rủi ro tín dụng/ck/data_riskcredit_copy (1).xlsx")
data.head(5)
```

```
# Result
```

Table 1: Head of data set

index	id	gioi tinh	thu nhap	tien dien/ thang	so tien vay	muc dich vay	gia dinh	so nam lam viec	tuoi	thoi gian vay	trinh do hoc van	tai san dam bao	kha nang tra no
0	1	2. Nam	17500000	420000	150000000	1. Vay tiêu dùng	1. Đã kết hôn	2	30	12	3. Cao đẳng	2. Có đủ TSĐB được	0. Trả nợ đúng hạn

												bảo lãnh	
1	2	1. Nữ	27000000	690000	220000000	2. Vay xây nhà	1. Đã kết hôn	8	33	15	2. Đại học	1. Có đủ TSĐB của bản thân	0. Trả nợ đúng hạn
2	3	2. Nam	16000000	300000	180000000	3. Vay mua xe	2. Độc thân	4	25	13	4. Trung cấp	3. Có 1 phần TSĐB	1. Trả nợ trễ hạn
3	4	2. Nam	13000000	260000	50000000	5. Vay đầu tư chứng khoán	2. Độc thân	3	26	5	3. Cao đẳng	4. Không có TSĐB	1. Trả nợ trễ hạn
4	5	1. Nữ	16000000	350000	120000000	1. Vay tiêu dùng	3. Ly hôn	4	36	14	2. Đại học	1. Có đủ TSĐB của bản thân	0. Trả nợ đúng hạn

Comment:

The dataset consists of 3000 rows and 12 features, including “id”, “gioi tinh”, “thu nhap”, “tien dien/thang”, “so tien vay”, “muc dich vay”, “gia dinh”, “so nam lam viec”, “tuoi”, “thoi gian vay”, “trinh do hoc van”, and “tai san dam bao”. Here, the target variable is “kha nang tra no”.

III. Step 2: Checking the data

I will check for any "impurities", such as null values or duplicate rows. If any of these will appear, I will remove them from the data set. I will also plot the correlations of the class column with all the other columns.

 # Code

```
#Searching for Missing value, type of data and also known the shape of data
data.info()
```

 # Result

Table 2: Non-Null Count of each column

```
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 13 columns):
```

	Column	Non-Null Count	Dtype
0	id	3000 non-null	int64
1	gioi tinh	3000 non-null	object
2	thu nhap	3000 non-null	int64
3	tien dien/ thang	3000 non-null	int64
4	so tien vay	3000 non-null	int64
5	muc dich vay	3000 non-null	object
6	gia dinh	3000 non-null	object
7	so nam lam viec	3000 non-null	int64
8	tuoi	3000 non-null	int64
9	thoi gian vay	3000 non-null	int64
10	trinh do hoc van	3000 non-null	object
11	tai san dam bao	3000 non-null	object
12	kha nang tra no	3000 non-null	object

Comment:
The data has 3000 rows and 13 features. Seven features are integer. Six features are strings (object).

```
# Code
df = data.drop(['id'],axis = 1)

#Looking the data
df.head()
```

 # Result

Table 3: Head of data set after delete id column

index	gioi tinh	thu nhap	tien dien/ thang	so tien vay	muc dich vay	gia dinh	so nam lam viec	tuoi	thoi gian vay	trinh do hoc van	tai san dam bao	kha nang tra no
0	2. Nam	17500000	420000	150000000	1. Vay tiêu dùng	1. Đã kết hôn	2	30	12	3. Cao đẳng	2. Có đủ TSĐB được bảo lãnh	0. Trả nợ đúng hạn
1	1. Nữ	27000000	690000	220000000	2. Vay xây nhà	1. Đã kết hôn	8	33	15	2. Đại học	1. Có đủ TSĐB của bản thân	0. Trả nợ đúng hạn

2	2. Nam	16000000	300000	180000000	3. Vay mua xe	2. Độc thân	4	25	13	4. Trung cấp	3. Có 1 phần TSĐB	1. Trả nợ trễ hạn
3	2. Nam	13000000	260000	50000000	5. Vay đầu tư chứng khoán	2. Độc thân	3	26	5	3. Cao đẳng	4. Không có TSĐB	1. Trả nợ trễ hạn
4	1. Nữ	16000000	350000	120000000	1. Vay tiêu dùng	3. Ly hôn	4	36	14	2. Đại học	1. Có đủ TSĐB của bản thân	0. Trả nợ đúng hạn

Comment:
Dropping unneeded features. We don't need the id feature as it cannot be converted into any categorical value.

 # Code

```
print("Column          Null count")
for i in [df]:
    print(f"\n{i.isnull().sum().sort_values(ascending=False)}")
```

 # Result

Table 4: Missing values

Column	Null count
gioi tinh	0
thu nhap	0
tien dien/ thang	0
so tien vay	0
muc dich vay	0
gia dinh	0
so nam lam viec	0
tuoi	0
thoi gian vay	0
trinh do hoc van	0
tai san dam bao	0
kha nang tra no	0

Comment:
The data has no missing values.

 # Code

```
# drop duplicates rows
df.drop_duplicates()
df.info
```

 # Result

Table 5: Data set after drop duplicates rows

```
Int64Index: 1416 entries, 0 to 2999
Data columns (total 12 columns):
```

	Column	Non-Null Count	Dtype
0	gioi tinh	1416 non-null	object
1	thu nhap	1416 non-null	int64
2	tien dien/ thang	1416 non-null	int64
3	so tien vay	1416 non-null	int64
4	muc dich vay	1416 non-null	object
5	gia dinh	1416 non-null	object
6	so nam lam viec	1416 non-null	int64
7	tuoi	1416 non-null	int64
8	thoi gian vay	1416 non-null	int64
9	trinh do hoc van	1416 non-null	object
10	tai san dam bao	1416 non-null	object
11	kha nang tra no	1416 non-null	object

Comment:

The data set is self-generated, so there are many duplicate rows. After deleting these rows, the dataset contains 1416 rows.

 # Code

```
#Looking unique values
print(df.nunique())
```

 # Result

Table 6: Unique values of each column

Column	Unique
gioi tinh	2
thu nhap	101

tien dien/ thang	201
so tien vay	104
muc dich vay	5
gia dinh	4
so nam lam viec	28
tuoi	35
thoi gian vay	24
trinh do hoc van	5
tai san dam bao	4
kha nang tra no	2

```
# Code
# Descriptive Statistics
from copy import deepcopy
data_2 = deepcopy(df)

numerical_features = ['thu nhap', 'tien dien/ thang', 'so tien vay',
                      'so nam lam viec', 'tuoi', 'thoi gian vay']
print('Number of numerical features: ', len(numerical_features))
```

```
# Result
Number of numerical features:  6
```

```
# Code
categorical_features = [x for x in df.columns if (x not in
numerical_features
                        and x != 'kha nang tra no')]
print('Number of categorical features: ', len(categorical_features))
```

```
# Result
Number of categorical features:  5
```

```
# Code
data_2[numerical_features].describe()
```

 # Result

Table 7: Descriptive statistics table of numerical features

index	thu nhap	tien dien/ thang	so tien vay	so nam lam viec	tuoi	thoi gian vay
count	1416	1416	1416	1416	1416	1416
mean	22190105,93	593478,11	141621115,82	7,03	34,95	12,73
std	7538396,39	233165,95	92094581,81	4,38	7,61	5,16
min	10000000	170000	14500000	1	19	3
25%	16000000	400000	70000000	4	29	9
50%	20000000	600000	125000000	6	35	12
75%	28000000	800000	200000000	9	39	18
max	42000000	4335000	498000000	32	53	26

Comment:

- "thu nhap": Borrower's income is between 10 million and 42 million VND, average is about 22.2 million VND
- "tien dien/ thang": electricity bill a month is in the range of 170000 to 4335000 VND, on average 1 household will pay 593478.11 VND for electricity every month
- "so tien vay": loan amount from 14.5 million to 498 million VND, average loan size is 141.6 million VND
- "so nam lam viec": the borrower has years of employment in the current job from 1 to 32 years and the average number of years of service is 7 years
- "tuoi": The age of the youngest borrower is 19 years old, the oldest is 53 years old. The average age of the borrower is about 35 years old
- "thoi gian vay": loan period is about 3 to 26 months, an average loan lasts 12.7 months

 # Code

```
data_2[categorical_features].describe()
```

 # Result

Table 8: Descriptive statistics table of categorical features

	gioi tinh	muc dich vay	gia dinh	trinh do hoc van	tai san dam bao
count	1416	1416	1416	1416	1416
unique	2	5	4	5	4

top	2. Nam	1. Vay tiêu dùng	1. Đã kết hôn	2. Đại học	1. Có đủ TSĐB của bản thân
freq	747	393	787	633	718

Comment:

- "gioi tinh": There are 2 genders male and female in the data set, in which male gender accounts for more, nearly 52.8%.
- "muc dich vay": There are 5 borrowing purposes, of which consumer loans account for the majority of 27.8%.
- "gia dinh": Married accounted for the majority of the 4 types of marital status, accounting for 55.6%
- "trinh do hoc van": university degree accounts for the majority of educational level, about 44.7%
- "tai san dam bao": half of the loans have sufficient collateral for themselves

IV. Step 3: Data visualization

 # Code

```
sns.countplot(y=df['gioi tinh'] ,data=df)
plt.xlabel("count of each gender")
plt.ylabel("gender")
plt.show()
```

 # Result

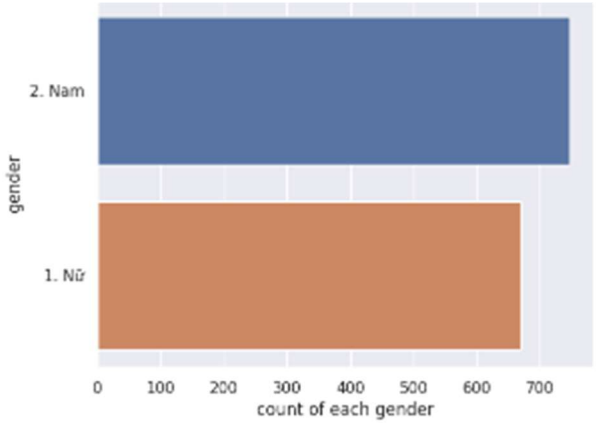


Figure 1: Count of each gender

 # Code

```
sns.countplot(y=df['muc dich vay'] ,data=df)
plt.xlabel("count of each purpose")
```

```
plt.ylabel("purpose")
plt.show()
```

 # Result

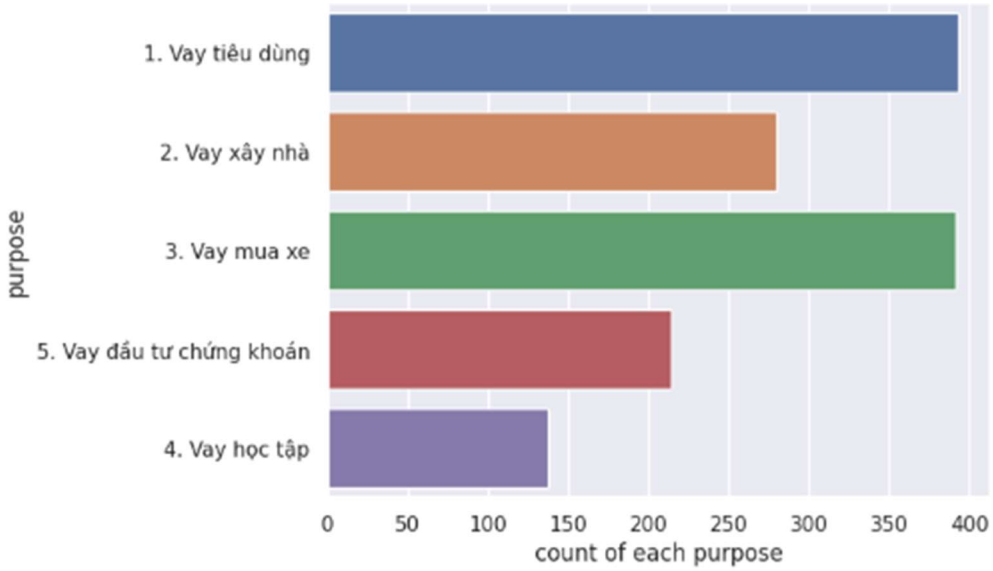


Figure 2: Count of each purppose

 # Code

```
sns.countplot(y=df['gia dinh'] ,data=df)
plt.xlabel("count of each marital status")
plt.ylabel("marital status")
plt.show()
```

 # Result

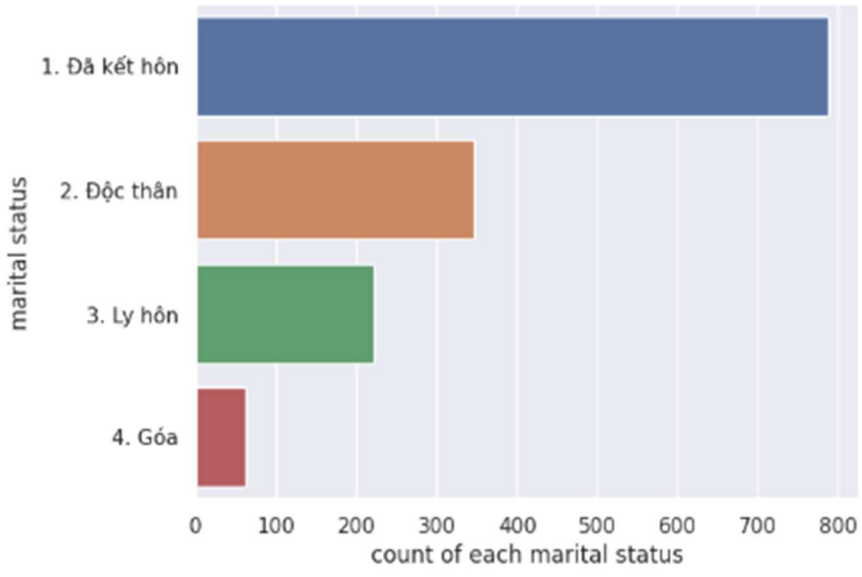


Figure 3: Count of each marital status

 # Code

```
sns.countplot(y=df['trinh do hoc van'] ,data=df)
plt.xlabel("count of each education level")
plt.ylabel("education level")
plt.show()
```

 # Result

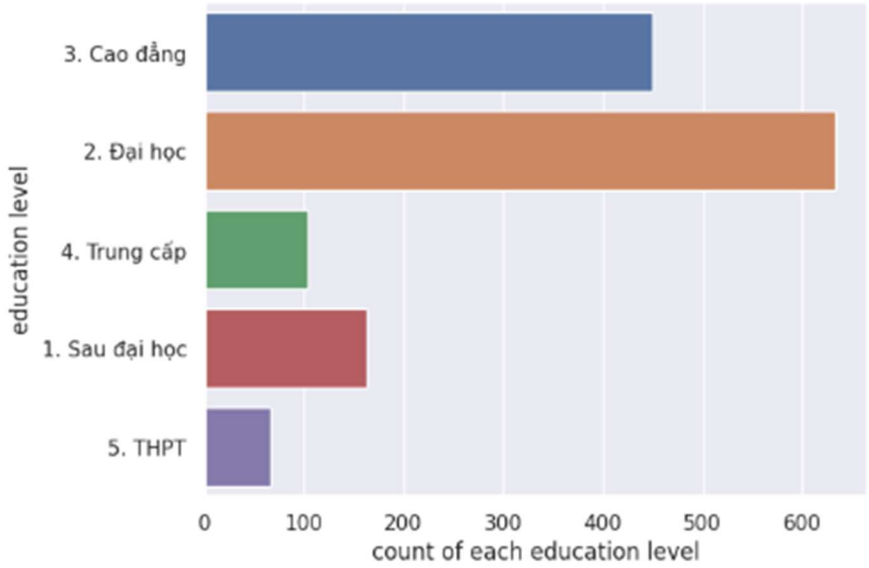


Figure 4: Count of each education level

 # Code

```
sns.countplot(y=df['tai san dam bao'] ,data=df)
plt.xlabel("count of each collateral")
plt.ylabel("collateral")
plt.show()
```

 # Result

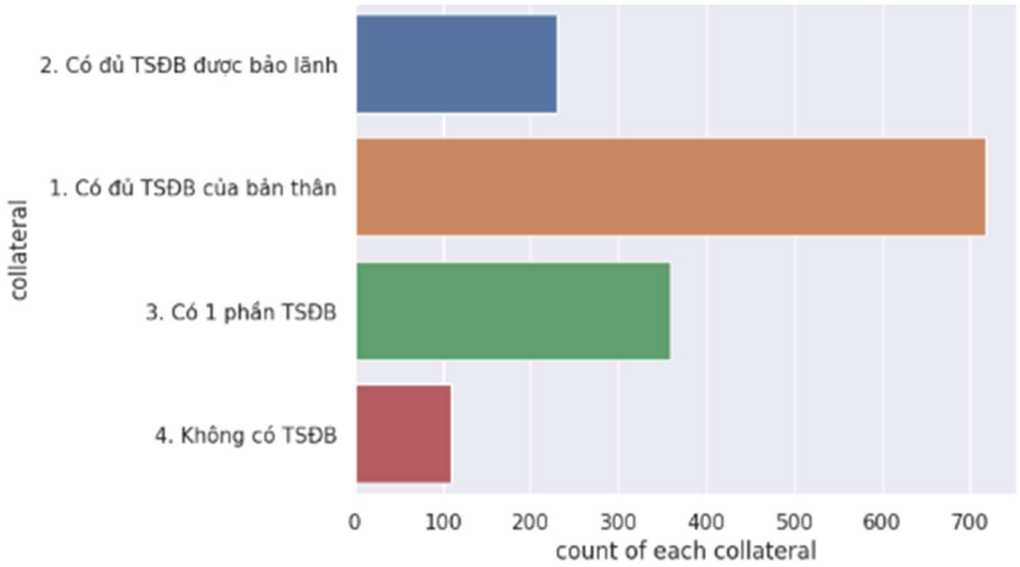


Figure 5: Count of each collateral

 # Code

```
data_2[numerical_features].hist(bins=50, figsize=(15, 10))
```

 # Result

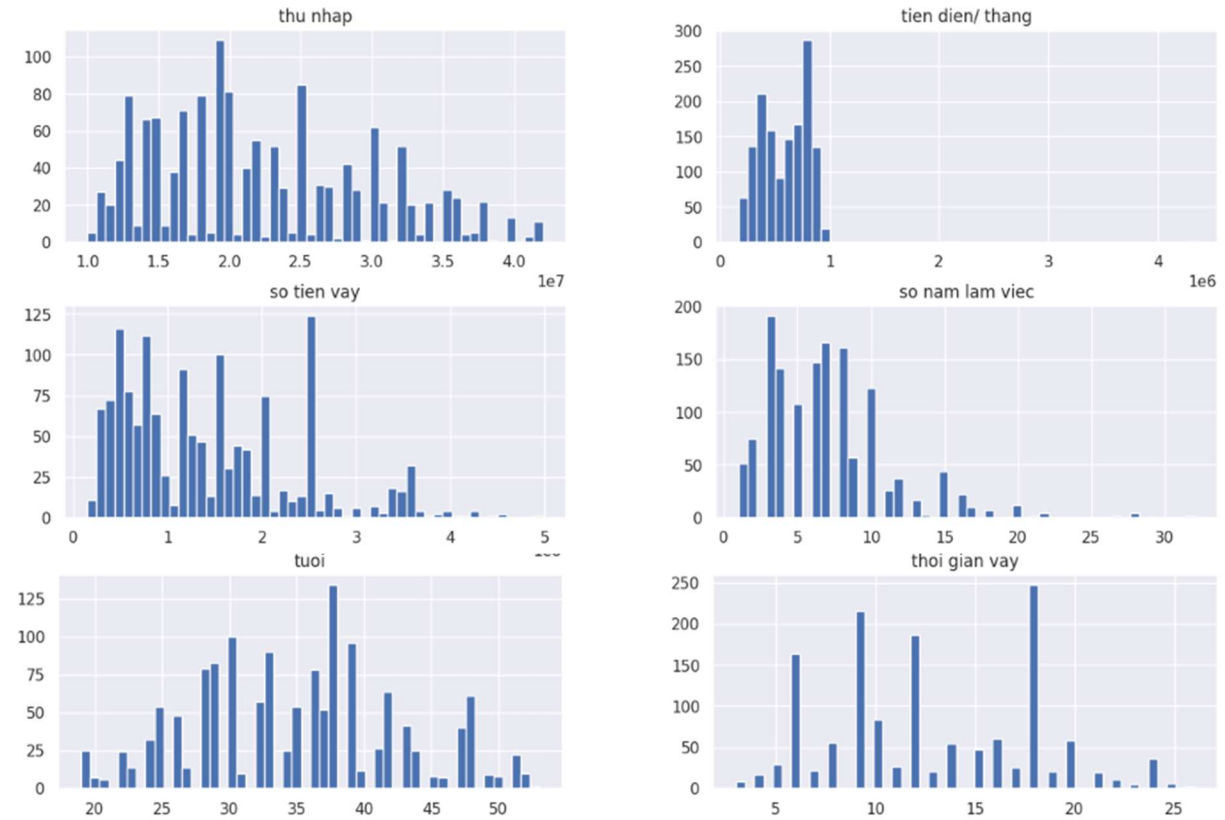


Figure 6: Histogram of numerical features

 # Code

```
#Converting categorical features to numeric

df = df.replace({"gioi tinh":{"1. Nữ":1,
                              "2. Nam":2},
                 "muc dich vay":{"1. Vay tiêu dùng":1,
                                 "2. Vay xây nhà":2,
                                 "3. Vay mua xe":3,
                                 "4. Vay học tập":4,
                                 "5. Vay đầu tư chứng khoán":5},
                 "gia dinh":{"1. Đã kết hôn":1,
                             "2. Độc thân":2,
                             "3. Ly hôn":3,
                             "4. Góa":4},
                 "trinh do hoc van":{"1. Sau đại học":1,
                             "2. Đại học":2,
                             "3. Cao đẳng":3,
                             "4. Trung cấp":4,
                             "5. THPT":5},
                "tai san dam bao":{"1. Có đủ TSĐB của bản thân":1,
                      "2. Có đủ TSĐB được bảo lãnh":2,
                      "3. Có 1 phần TSĐB":3,
                      "4. Không có TSĐB":4},
                 "kha nang tra no":{"0. Trả nợ đúng hạn":0,
                                    "1. Trả nợ trễ hạn":1}
                 })
df.info()
```

 # Result

Table 9: Data set after converting categorical features to numeric

	gioi tinh	thu nhap	tien dien/ thang	so tien vay	muc dich vay	gia dinh	so nam lam viec	tuoi	thoi gian vay	trinh do hoc van	tai san dam bao	kha nang tra no
0	2	17500000	420000	150000000	1	1	2	30	12	3	2	0
1	1	27000000	690000	220000000	2	1	8	33	15	2	1	0
2	2	16000000	300000	180000000	3	2	4	25	13	4	3	1
3	2	13000000	260000	50000000	5	2	3	26	5	3	4	1
4	1	16000000	350000	120000000	1	3	4	36	14	2	1	0

```
# Code
df["kha nang tra no"].value_counts()
```

```
# Result
0 961
1 455
Name: kha nang tra no, dtype: int64
```

```
# Code
print('Pay on time: ', round(df["kha nang tra
no"].value_counts()[0]/len(df) * 100,2), '% of data set')

print('Pay late: ', round(df["kha nang tra no"].value_counts()[1]/len(df)
* 100,2), '% of data set')
```

```
# Result
Pay on time:  67.87 % of data set
Pay late:  32.13 % of data set
```

```
# Code
plt.figure(figsize = (6,4))

sns.countplot(x=df["kha nang tra no"])
plt.title("Target Distributions", fontsize=14)
plt.show()
```

```
# Result
```

Figure 7: Target distributions

Comment:

As we can see data is imbalanced. Label 0 occupies 67.87% of the dataset, and label 1 occupies 32.13%.

```
# Code
pd.crosstab(df['gioi tinh'],df['kha nang tra no']).plot(kind="bar",
                                                        figsize=(10,5),
                                                        color=['#1CA53B','#AA1111'
])
plt.title('Kha nang ta no theo gioi tinh')
plt.xlabel('Gioi tinh (1 = Nu, 2 = Nam)')
plt.xticks(rotation=0)
plt.legend(["Tra no dung han", "Tra no tre han"])
plt.ylabel('Frequency')
plt.show()
```

```
# Result
```

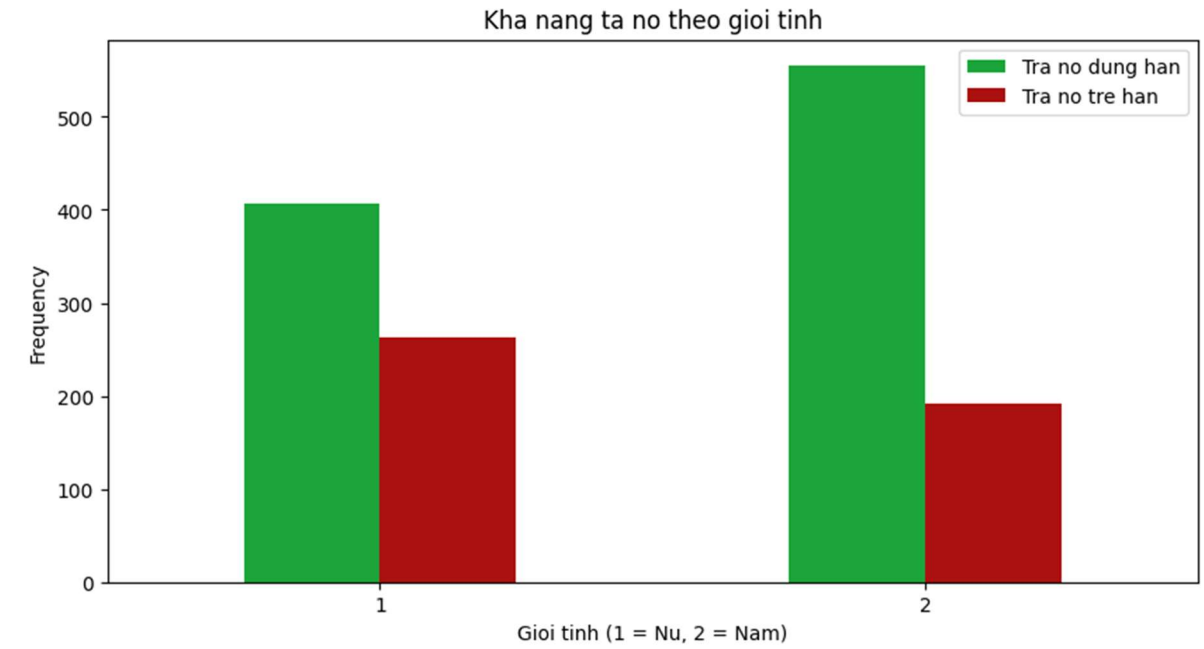


Figure 8: Ability to repay with gender

```
# Code
pd.crosstab(df['muc dich vay'],df['kha nang tra no']).plot(kind="bar",
                                                figsize=(10,5),
                                                color=['#1CA53B','#AA1111' ])
plt.title('Kha nang ta no theo muc dich vay’)
plt.xlabel('Muc dich vay')
plt.xticks(rotation=0)
plt.legend(["Tra no dung han", "Tra no tre han"])
plt.ylabel('Frequency')
plt.show()
```

 # Result

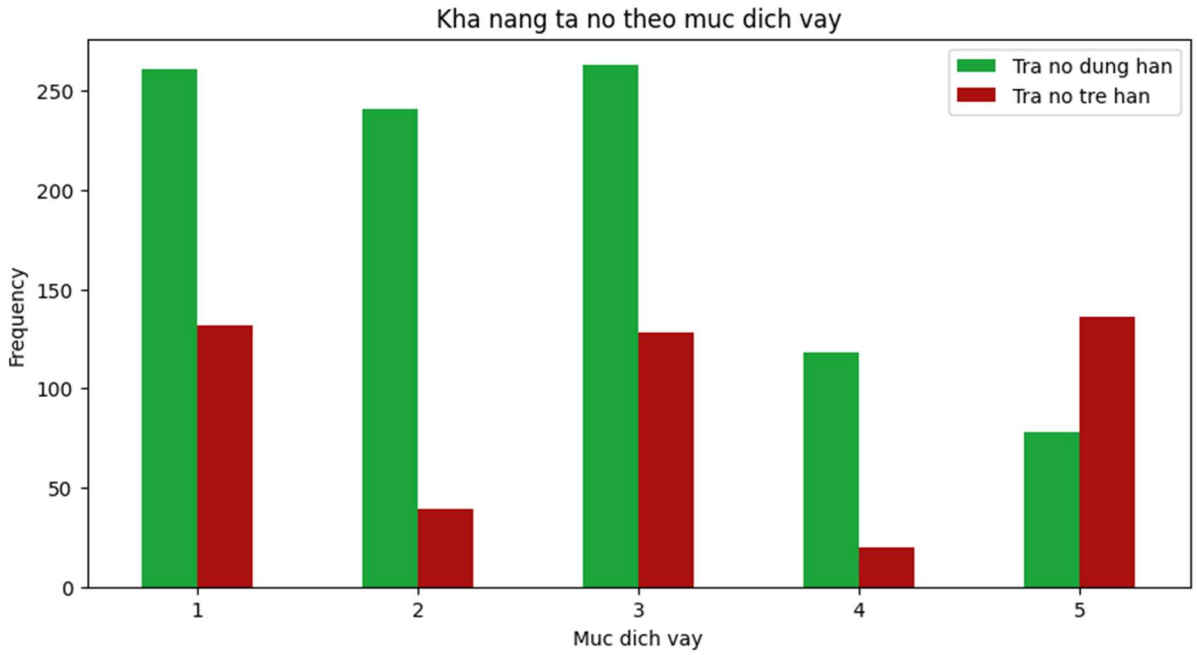


Figure 9: Ability to repay with purpose of loan

 # Code

```
pd.crosstab(df['gia dinh'],df['kha nang tra no']).plot(kind="bar",
                                                       figsize=(10,5),
                                                       color=['#1CA53B','#AA1111' ])
plt.title('Kha nang ta no theo tinh trang hon nhan')
plt.xlabel('Tinh trang hon nhan')
plt.xticks(rotation=0)
plt.legend(["Tra no dung han", "Tra no tre han"])
plt.ylabel('Frequency')
plt.show()
```

 # Result

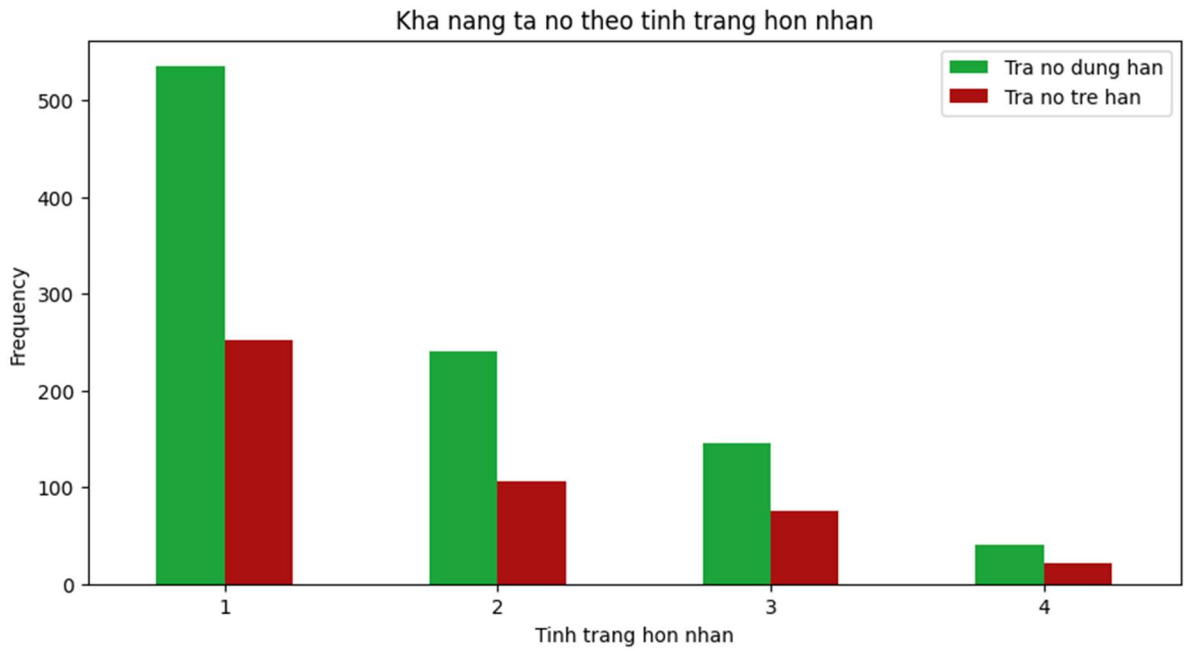


Figure 10: Ability to repay with marital status

 # Code

```
pd.crosstab(df['trinh do hoc van'],
df['kha nang tra no']).plot(kind="bar",figsize=(10,5),
                                        color=['#1CA53B','#AA1111' ])
plt.title('Kha nang ta no theo trinh do hoc van')
plt.xlabel('Trinh do hoc van')
plt.xticks(rotation=0)
plt.legend(["Tra no dung han", "Tra no tre han"])
plt.ylabel('Frequency')
plt.show()
```

 # Result

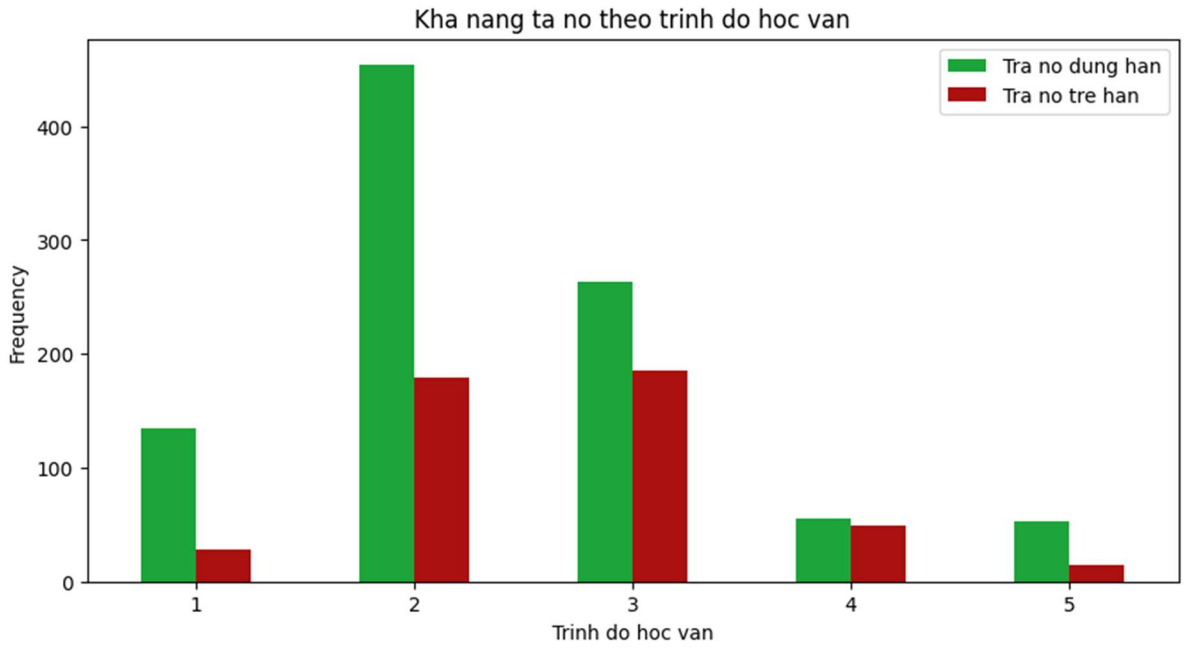


Figure 11: Ability to repay with education level

 # Code

```
pd.crosstab(df['tai san dam bao'],df['kha nang tra no']).plot(kind="bar",
                                                figsize=(10,5),
                                                color=['#1CA53B','#AA1111' ])
plt.title('Kha nang ta no theo tai san dam bao')
plt.xlabel('Tai san dam bao')
plt.xticks(rotation=0)
plt.legend(["Tra no dung han", "Tra no tre han"])
plt.ylabel('Frequency')
plt.show()
```

 # Result

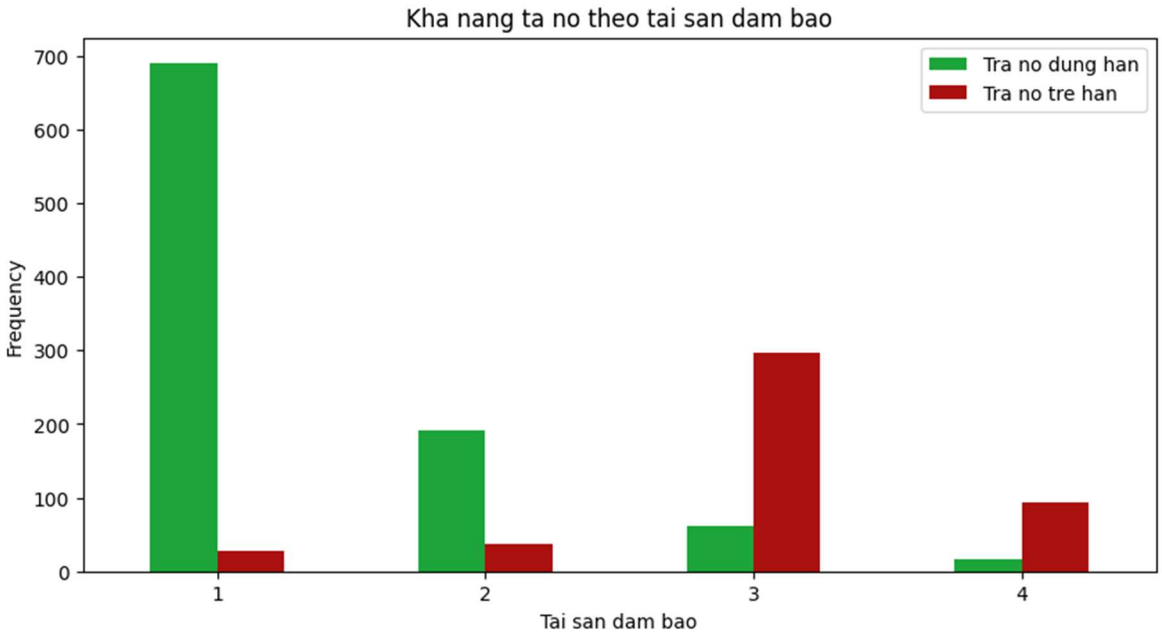


Figure 12: Ability to repay with collateral

V. Step 4: Pre-processing

In this part, I prepare data for my models. This means that I choose the columns that will be independed variables and which column the class that I want to predict. Once I am done with that, I split data into train and test sets.

```
# Code
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
from sklearn.metrics import roc_curve, auc, f1_score, precision_score,
recall_score
```

```
# Code
#ROC
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')
```

```
# Code
#Setting X and y variables to the prediction
target = ['kha nang tra no']
#features = ['gioi tinh', 'thu nhap','tien dien/ thang', 'so tien vay',
            #'muc dich vay', 'gia dinh', 'so nam lam viec','tuoi',
            #'thoi gian vay', 'trinh do hoc van', 'tai san dam bao']
features = ['tai san dam bao', 'so tien vay', 'thu nhap', 'muc dich vay',
       'thoi gian vay']

print('Target: ', target)
print('Features: ', features)
```

```
# Result
Target:  ['kha nang tra no']
Features:  ['tai san dam bao', 'so tien vay', 'thu nhap', 'muc dich vay',
'thoi gian vay']
```

```
# Code
X = df[features].values
y = df[target].values
# split data into training and validation data, for both features and
target
# The split is based on a random number generator. Supplying a numeric
value to
# the random_state argument guarantees we get the same split every time
we run model
n_state = 42
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3
, random_state=n_state)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

VI. Step 5: Building the Models

After the pre-processing on data sets, I use the 6 models which included:

- Logistic Regression
- Decision tree
- Random Forest
- XGBoost
- Ada Boost
- K-Nearest Neighbors (KNN)

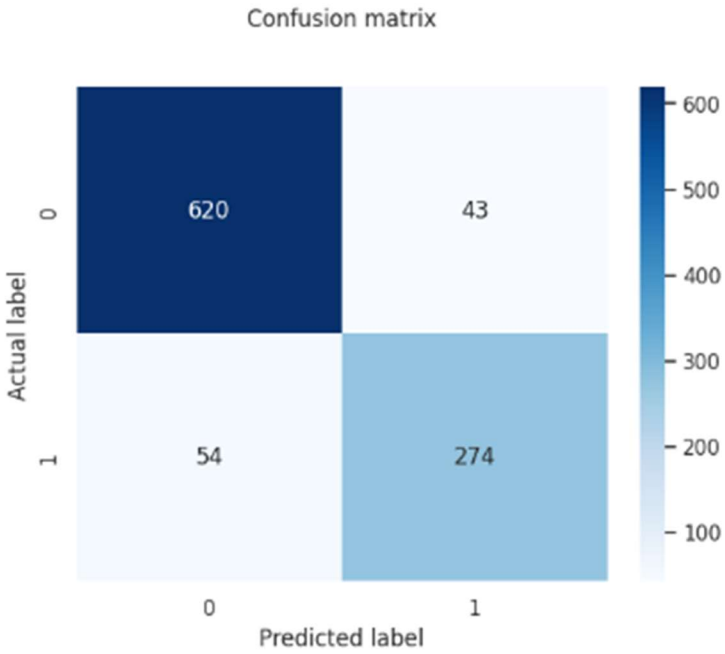
Each model will be trained and make a prediction for the train set and test set. For evaluation, I use seven metrics; Confusion Matrix, Accuracy, F1- Score, Recall, Precision, ROC area and Feature Importance. We have used Confusion matrix and these evaluation metrics as they are the correct metrics used for classification algorithms, i.e. the predicted variable's value is binary. Evaluation metrics like Accuracy, Precision, Recall, etc. definitely increase the chances of finding and removing the error algorithms are facing. This can further improve the same evaluation metrics performed on the results and lessen the scope of improvement gradually, giving better results.

1. Logistic regression

a. Predict

```
# Code
from sklearn.linear_model import LogisticRegression

# Define model
LR_classifier =
LogisticRegression(multi_class='multinomial',random_state=n_state)
# Fit model
LR_classifier.fit(X_train, y_train.ravel())
# get predicted y on train data
y_pred = LR_classifier.predict(X_train)

p = sns.heatmap(pd.DataFrame(confusion_matrix(y_train,y_pred)),
annot=True, cmap="YlGnBu" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

print('Classification report:')
print(classification_report(y_train,y_pred))
print('Logistic Regression accuracy: ', round(accuracy_score(y_train,
y_pred),4))
```

```
# Result
```



Classification report:

	precision	recall	f1-score	support
0	0.92	0.94	0.93	663
1	0.86	0.84	0.85	328
accuracy			0.90	991
macro avg	0.89	0.89	0.89	991
weighted avg	0.90	0.90	0.90	991

Logistic Regression accuracy: 0.9021

Figure 13: Results on the training set of Logistic Regression

Comment:

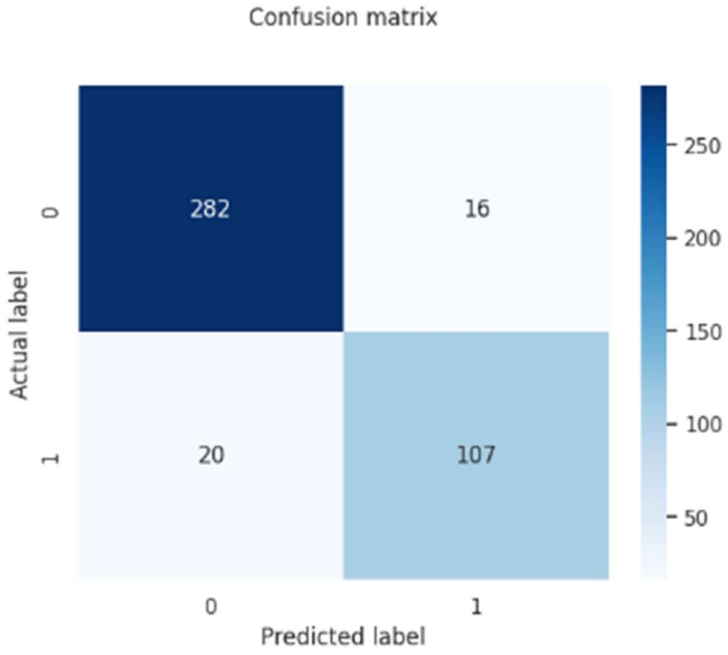
Figure 13 shows four numeric values which represent True Positive, True Negative, False Negative, False Positive. The values for True Positive and True False Positive are 620 and 274 which totals to 894 true/correct predictions and False Positive and False Negative are 43 and 54 which totals to 97 false/incorrect predictions for the confusion matrix calculated for Logistic Regression.

The model has a high precision, which indicates that the accuracy of the points found is high, and a high recall, indicating that the model does not miss a significant number of customers who are late on their loans. Furthermore, the model's high F1-score and overall accuracy of 90.21% indicate its ability to predict is well.

```
# Code
# get predicted y on test data
```

```
LR_y_pred = LR_classifier.predict(X_test)

p = sns.heatmap(pd.DataFrame(confusion_matrix(y_test,LR_y_pred)),
annot=True, cmap="Blues" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
print('Logistic Regression accuracy: ', round(accuracy_score(y_train,
y_pred),4))

print('Classification report:')
print(classification_report(y_test,LR_y_pred))
print('Logistic Regression accuracy: ', round(accuracy_score(y_test,
LR_y_pred),4))
```

 # Result



Classification report:

	precision	recall	f1-score	support
0	0.93	0.95	0.94	298
1	0.87	0.84	0.86	127
accuracy			0.92	425
macro avg	0.90	0.89	0.90	425

weighted avg	0.91	0.92	0.91	425

Logistic Regression accuracy: 0.9153

Figure 14: Results on the test set of Logistic Regression

Comment:
Figure 14 shows four numeric values which represent True Positive, True Negative, False Negative, False Positive. The values for True Positive and True False Positive are 282 and 107 which totals to 389 true/correct predictions and False Positive and False Negative are 16 and 20 which totals to 36 false/incorrect predictions for the confusion matrix calculated for Logistic Regression.

The model has a high precision, which indicates that the accuracy of the points found is high, and a high recall, indicating that the model does not miss a significant number of customers who are late on their loans. Furthermore, the model's high F1-score and overall accuracy of 91.53% indicate its ability to predict is well.

The Logistic Regression model seems to perform well, with high precision, recall, and F1-scores for both the training set and the test set.

 # Code

```
df_report = pd.DataFrame({'Ground Truth':y_test.ravel(),
'Prediction':LR_y_pred.ravel()})

df_report
```

 # Result

Table 10: Prediction results of Logistic Regression

	Ground Truth	Prediction
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
...	...	...
420	1	1
421	0	0
422	1	0
423	1	1

424	0	0

```
# Code
import statsmodels.api as SM

model_1 = SM.Logit(y_train, X_train).fit()
print(model_1.summary())
```

 # Result

Table 11: Logit Regression Results

```
Optimization terminated successfully.
Current function value: 0.366097
Iterations 7

                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  991
Model:                          Logit   Df Residuals:                      986
Method:                           MLE   Df Model:                            4
Date:                Wed, 24 May 2023   Pseudo R-squ.:                  0.4234
Time:                        08:19:44   Log-Likelihood:                -362.80
converged:                       True   LL-Null:                       -629.16
Covariance Type:            nonrobust   LLR p-value:                5.637e-114
==============================================================================
            coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1        2.3248      0.148     15.716      0.000       2.035       2.615
x2        1.0545      0.152      6.939      0.000       0.757       1.352
x3       -0.6810      0.116     -5.885      0.000      -0.908      -0.454
x4        0.2779      0.103      2.696      0.007       0.076       0.480
x5       -0.1221      0.154     -0.795      0.427      -0.423       0.179
```

Comment:

Table 11 shows that 'tai san dam bao' (x1) and 'so tien vay' (x2) have positive coefficients that indicate a positive relationship with the ability to repay, while 'thu nhap' (x3) has negative coefficients that indicate a negative relationship. 'tai san dam bao' (x1), 'so tien vay' (x2) and 'thu nhap' (x3) have P-values less than 0.05, indicating that the corresponding predictor is statistically significant in predicting the ability to repay.

b. Feature Importance

```
# Code
importance_lr = LR_classifier.coef_[0] #use coeffcient as importance
```

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

 # Result

Table 12:Logistic Regression Feature Importance

	FeatureName	Logistic Regression Feature Importance
0	tai san dam bao	1.254.513
1	so tien vay	592.693
2	thu nhap	-340.983
3	muc dich vay	135.648
4	thoi gian vay	-62.146

 # Code

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

 # Result

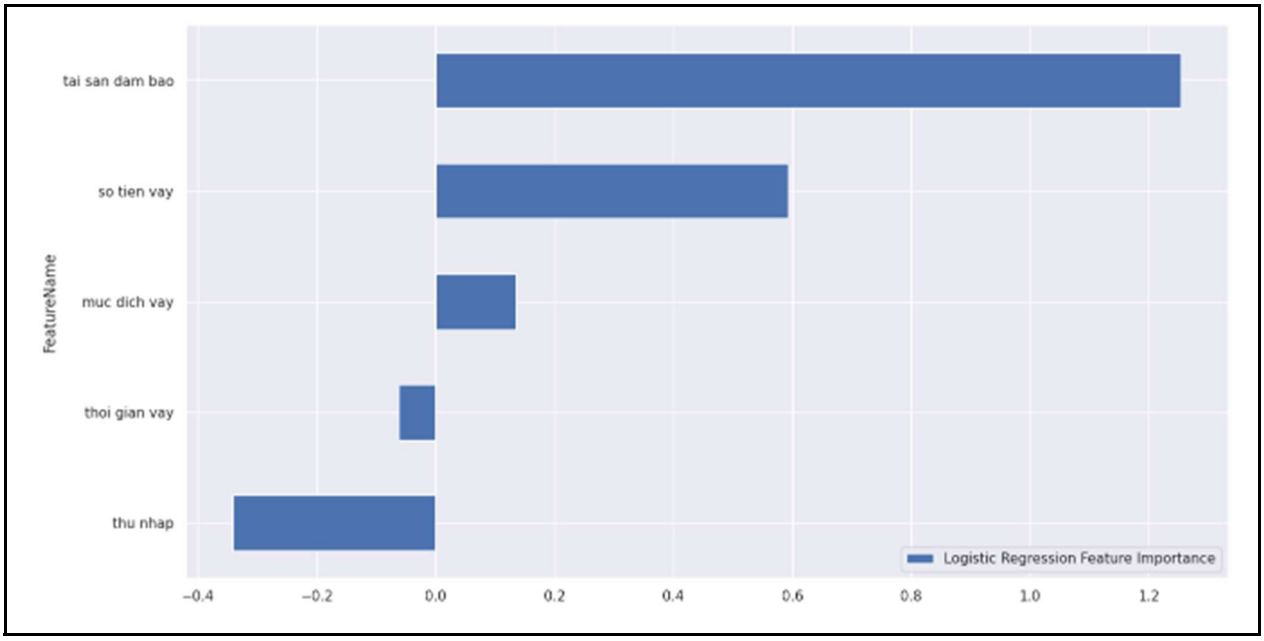


Figure 15: Logistic Regression Feature Importance

c. ROC Curve

 # Code

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

 # Result

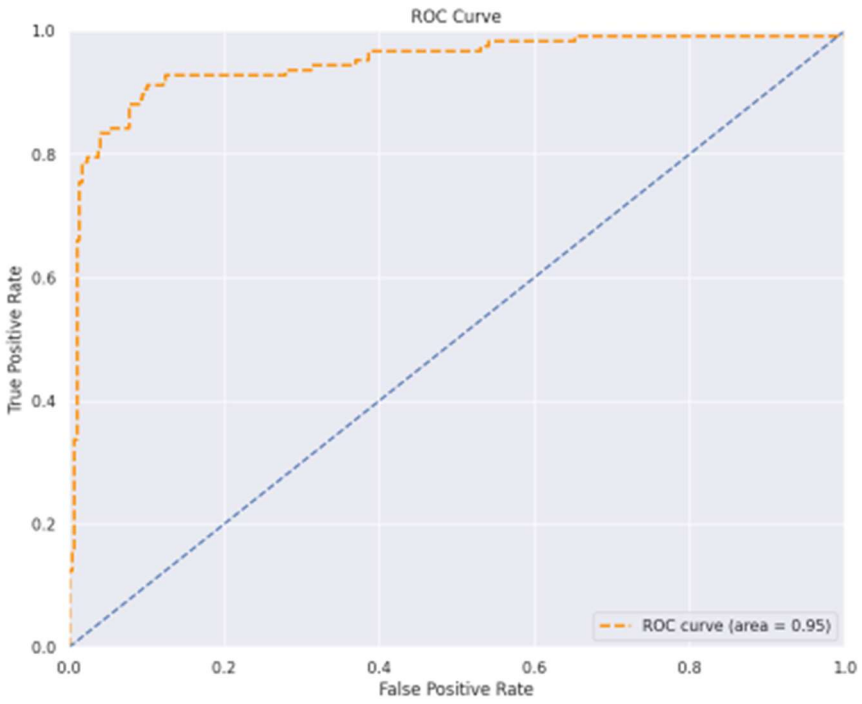


Figure 16: ROC Curve for Logistic Regression model

Comment:
The ROC line of the Logistic Regression model is close to the upper left corner of the, which shows the model's classification and forecasting ability is good. In addition, Area Under The Curve (AUC) = 0.95 (close to 1) also indicates that this model has a good performance.

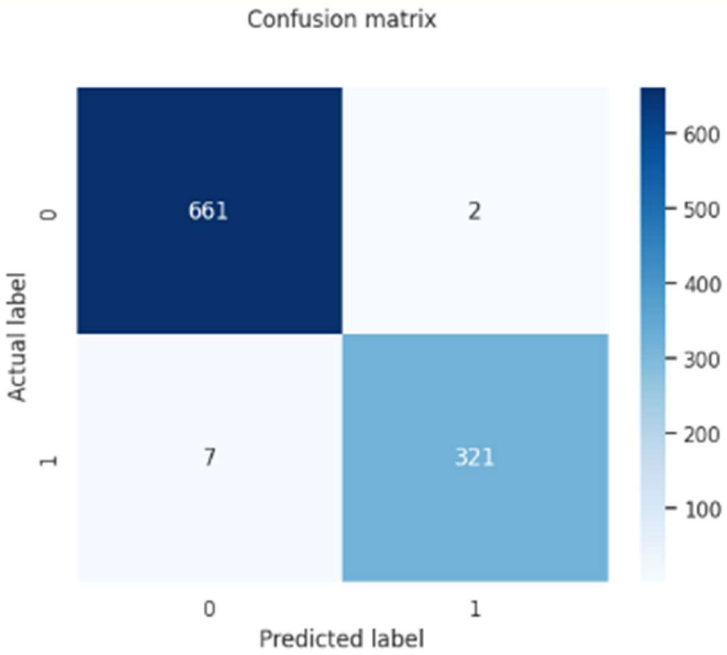
2. Decision tree

a. Prediction

```
# Code
from sklearn.tree import DecisionTreeClassifier
# Define model
DT_classifier = DecisionTreeClassifier()
# Fit model
DT_classifier.fit(X_train, y_train.ravel())
# get predicted y on train data
y_pred = DT_classifier.predict(X_train)

p = sns.heatmap(pd.DataFrame(confusion_matrix(y_train,y_pred)),
annot=True, cmap="Blues" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

print('Classification report:')
```

```
print(classification_report(y_train,y_pred))
print('Decision Tree accuracy: ', round(accuracy_score(y_train,
y_pred),4))
```

 # Result



Classification report:

	precision	recall	f1-score	support
0	0.99	1.00	0.99	663
1	0.99	0.98	0.99	328
accuracy			0.99	991
macro avg	0.99	0.99	0.99	991
weighted avg	0.99	0.99	0.99	991

Decision Tree accuracy: 0.9909

Figure 17: Results on the training set of Decision Tree

Comment:

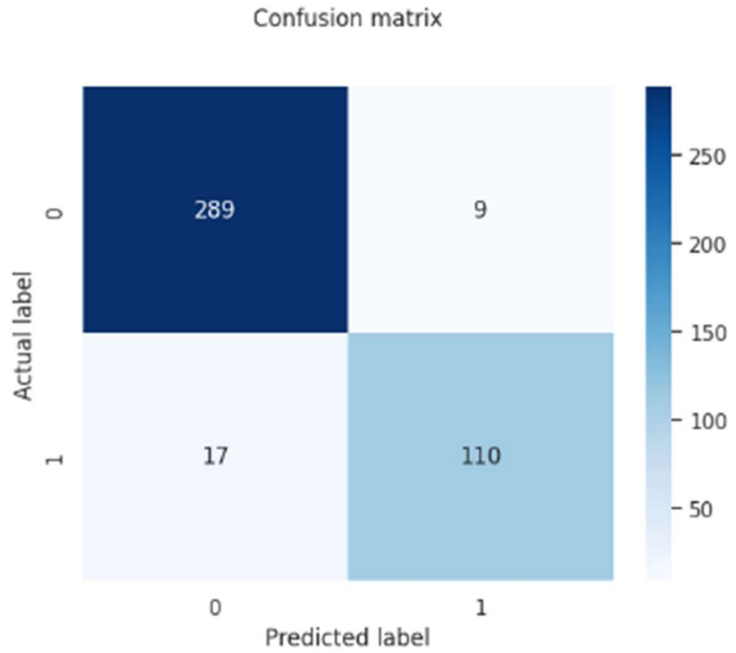
Figure 17 shows four numeric values which represent True Positive, True Negative, False Negative, False Positive. The values for True Positive and True False Positive are 661 and 321 which totals to 982 true/correct predictions and False Positive and False Negative are 2 and 7 which totals to 9 false/incorrect predictions for the confusion matrix calculated for Decision Tree.

The model has a high precision, which indicates that the accuracy of the points found is high, and a high recall, indicating that the model does not miss a significant number of customers who are late on their loans. Furthermore, the model's high F1-score and overall accuracy of 99.09% indicate its ability to predict is very good.

 # Code

```
# get predicted y on test data

DT_y_pred = DT_classifier.predict(X_test)
p = sns.heatmap(pd.DataFrame(confusion_matrix(y_test,DT_y_pred)),
annot=True, cmap="Blues" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

print('Classification report:')
print(classification_report(y_test,DT_y_pred))
print('Decision Tree accuracy: ', round(accuracy_score(y_test,
DT_y_pred),4))
```

 # Result



Classification report:

	precision	recall	f1-score	support
0	0.94	0.97	0.96	298
1	0.92	0.87	0.89	127

accuracy			0.94	425
macro avg	0.93	0.92	0.93	425
weighted avg	0.94	0.94	0.94	425

```
Decision Tree accuracy:  0.9388
```

Figure 18: Results on the test set of Decision Tree

Comment:
Figure 18 shows four numeric values which represent True Positive, True Negative, False Negative, False Positive. The values for True Positive and True False Positive are 289 and 110 which totals to 399 true/correct predictions and False Positive and False Negative are 9 and 17 which totals to 26 false/incorrect predictions for the confusion matrix calculated for Decision Tree.

The model has a high precision, which indicates that the accuracy of the points found is high, and a high recall, indicating that the model does not miss a significant number of customers who are late on their loans. Furthermore, the model's high F1-score and overall accuracy of 93.88% indicate its ability to predict is very good.

The Decision Tree model seems to perform well, with high precision, recall, and F1-scores for both the training set and the test set.

b. Feature Importance

 # 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)
```

 # Result

Table 13: Decision Tree Feature Importance

	FeatureName	Decision Tree Feature Importance
0	tai san dam bao	621.226
1	so tien vay	169.876
2	thu nhap	92.011
3	muc dich vay	71.345
4	thoi gian vay	45.542

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

Result

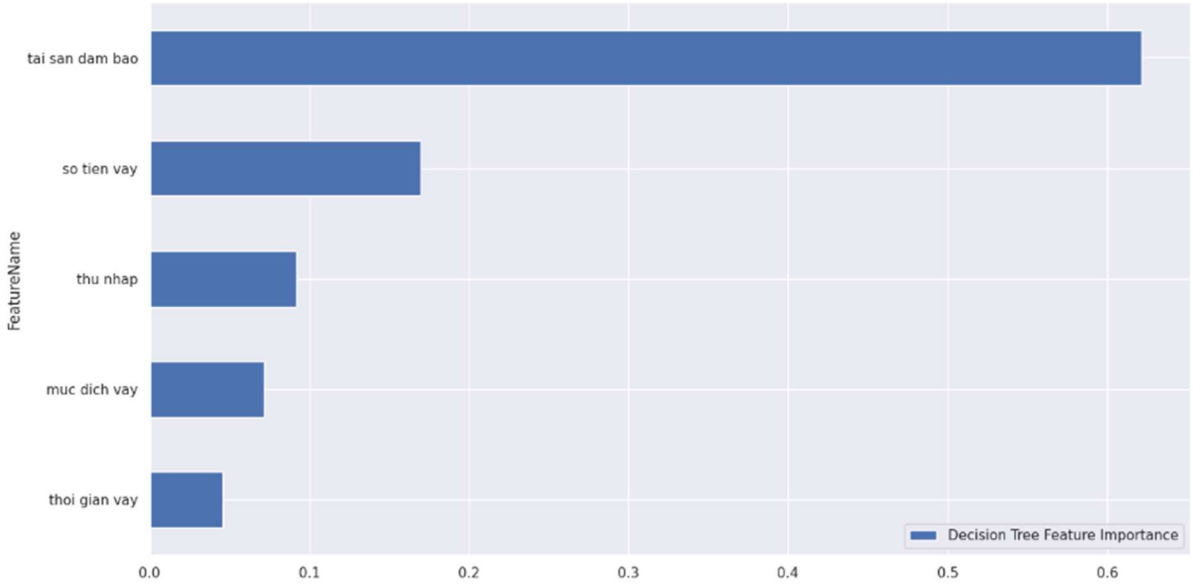


Figure 19: Decision Tree Feature Importance

c. ROC Curve

```
# Code
y_pred_prob_test_dt = DT_classifier.predict_proba(X_test)[:, 1]

fpr, tpr, thres = roc_curve(y_test, y_pred_prob_test_dt)
DT_roc_auc = auc(fpr, tpr)
_plot_roc_curve(fpr, tpr, thres, DT_roc_auc)
```

Result

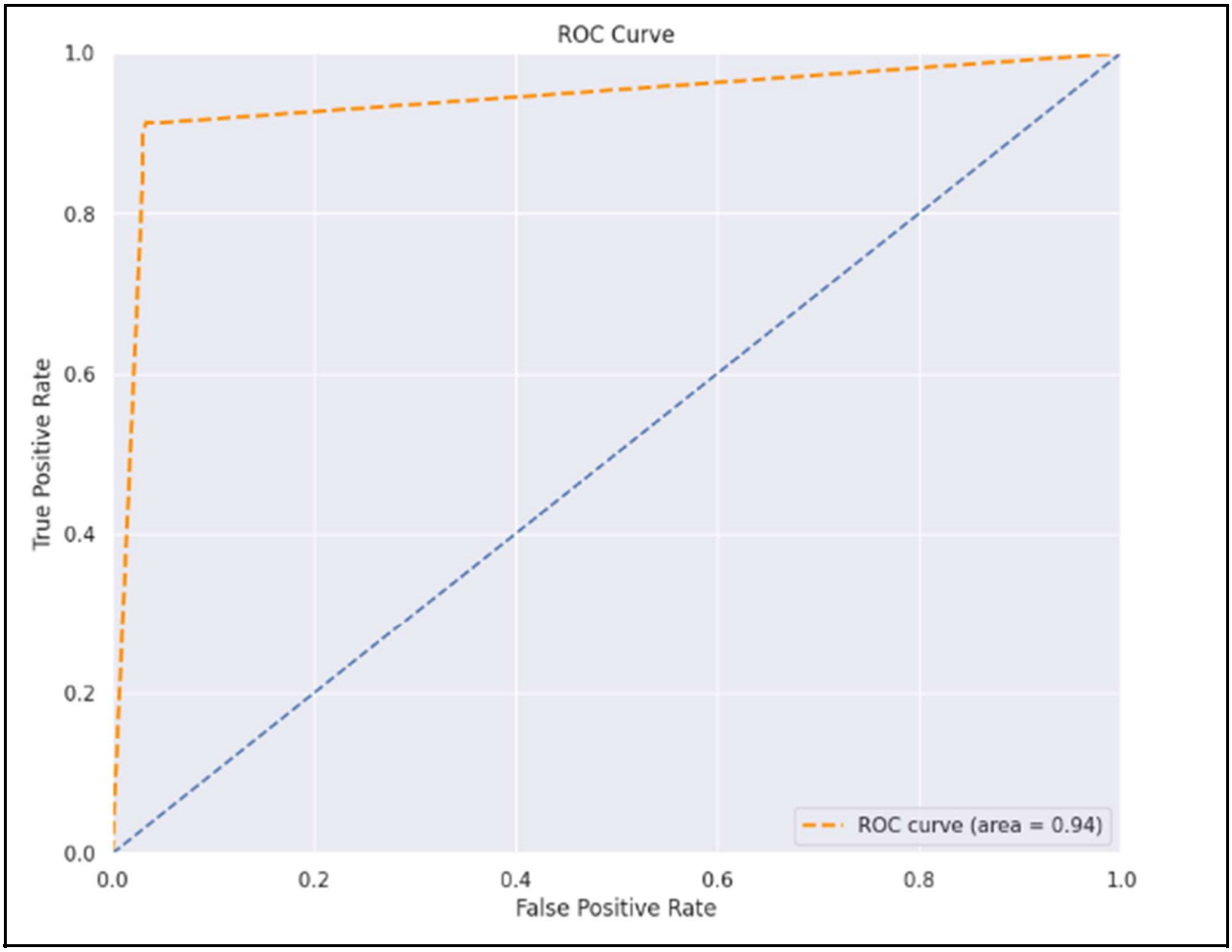


Figure 20: ROC curve of Decision Tree model

Comment:

The ROC line of the Decision Tree model is close to the upper left corner of the, which shows the model's classification and forecasting ability is good. In addition, Area Under The Curve (AUC) = 0.94 (close to 1) also indicates that this model has a good performance.

d. Plot tree

```
# 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()#(max_depth=5, min_samples_leaf=100)
#max_depth=5 , min_samples_leaf=200
clf.fit(X_plot_tree.values, y.ravel())
print('Note:Class 0 is on-time and class 1 is late payment')
```

```
export_graphviz(clf,
                out_file=dot_data,
                feature_names=features,
                class_names=['On-time','Lated'],
                filled=True,
                rounded=True,
                special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

 # Result

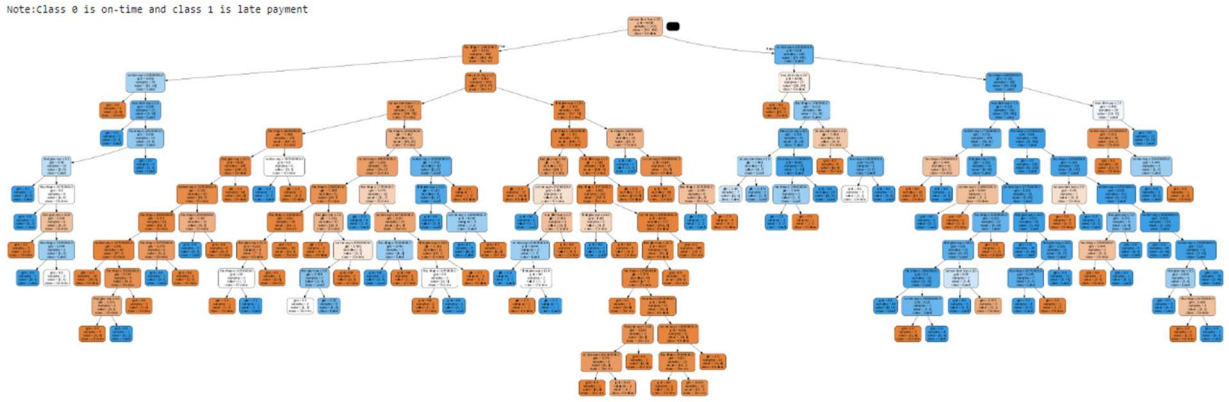


Figure 21: Plot tree before pruning

 # Code

```
# Pruning tree

dot_data = StringIO()
clf = DecisionTreeClassifier(max_depth=5, min_samples_leaf=100)
clf.fit(X_plot_tree.values, y.ravel())
print('Note:Class 0 is on-time and class 1 is late payment')
export_graphviz(clf,
                out_file=dot_data,
                feature_names=features,
                class_names=['On-time','Lated'],
                filled=True,
                rounded=True,
                special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

 # Result

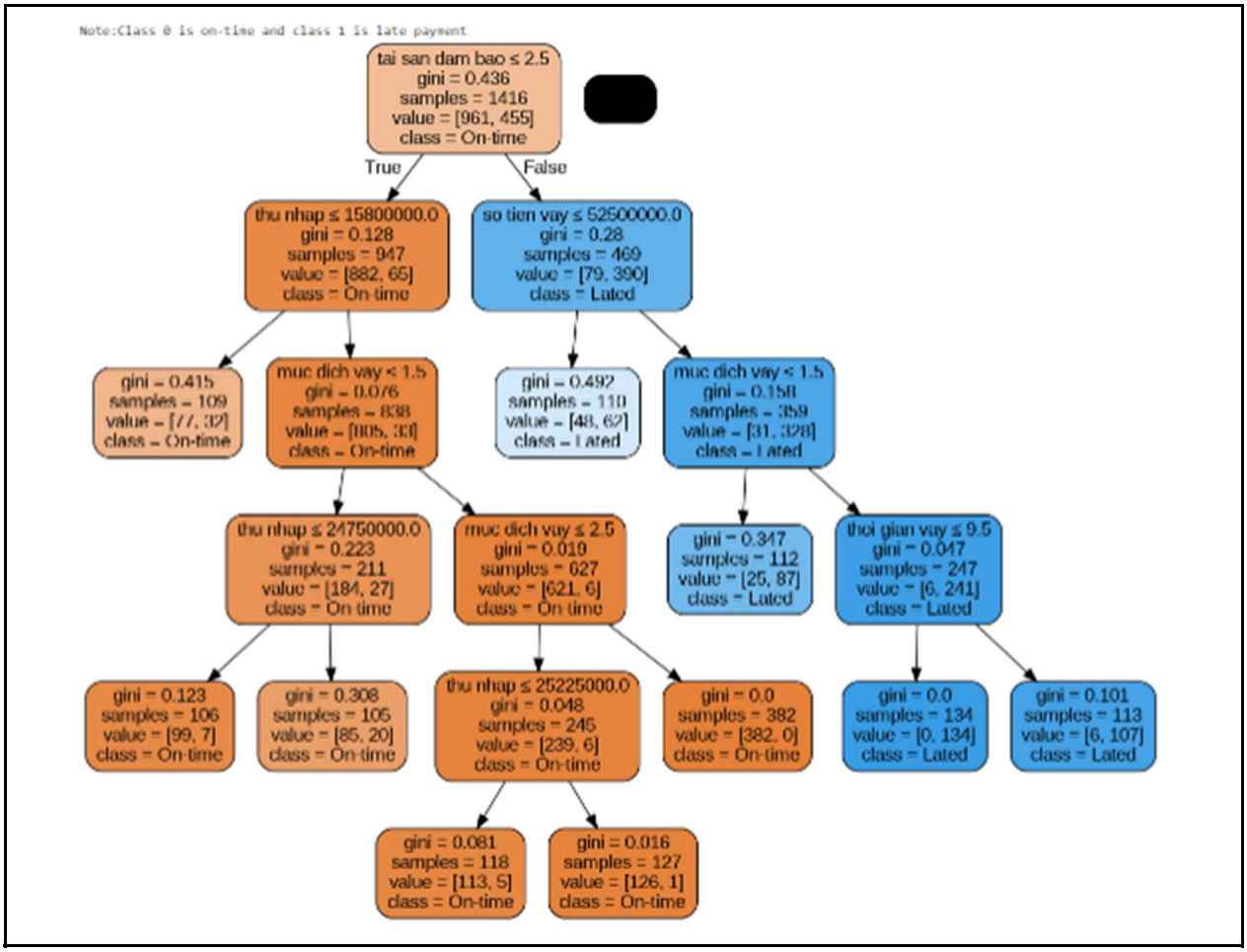


Figure 22: Plot tree after pruning

Comment:

Figure 22 shows that:

The group of borrowers with collateral is 1, 2: they have sufficient collateral for themselves or have sufficient collateral to be guaranteed; their monthly income is more than 158,000 VND; the loan is consumer loans or house loans; and they have a high rate of on-time repayment (621/627)*100 = 99%.

The group of borrowers with collateral is 3, 4 (partially secured, no collateral), the loan amount is greater than 5250,000,000 (VND), and the loan is a housing loan, car loan, study loan, or securities investment loan. Loans with a term greater than 9.5 months have a high late payment rate of (241/247)*100, or 98%.

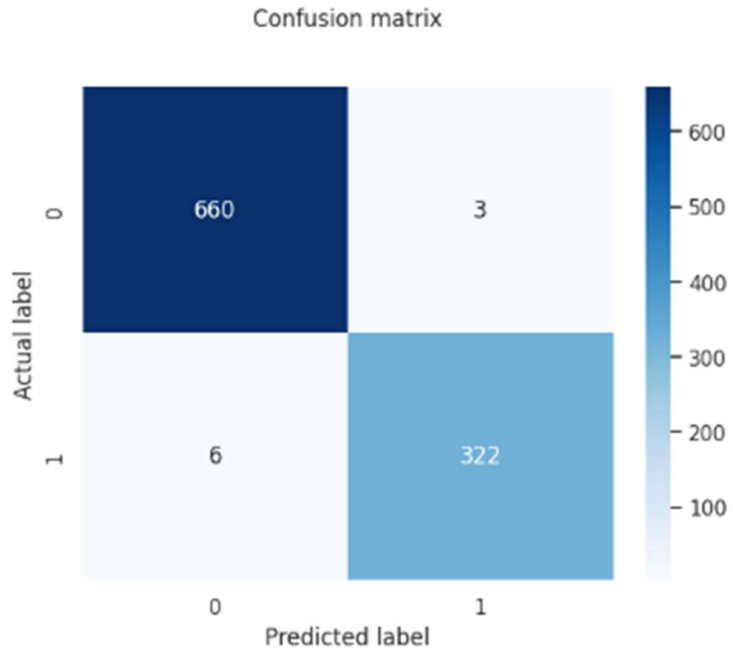
3. Random Forest

a. Prediction

 # Code

```
from sklearn.ensemble import RandomForestClassifier

# Define model
RF_classifier = RandomForestClassifier()
# Fit model
RF_classifier.fit(X_train, y_train.ravel())
# get predicted y on train data
y_pred = RF_classifier.predict(X_train)
p = sns.heatmap(pd.DataFrame(confusion_matrix(y_train,y_pred)),
annot=True, cmap="Blues" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

print('Classification report:')
print(classification_report(y_train,y_pred))
print('Random Forest accuracy: ', accuracy_score(y_train, y_pred))
```

 # Result



Classification report:

	precision	recall	f1-score	support
0	0.99	1.00	0.99	663
1	0.99	0.98	0.99	328

accuracy			0.99	991
macro avg	0.99	0.99	0.99	991
weighted avg	0.99	0.99	0.99	991

```
Random Forest accuracy:  0.9909
```

Figure 23: Results on the training set of Random Forest

Comment:

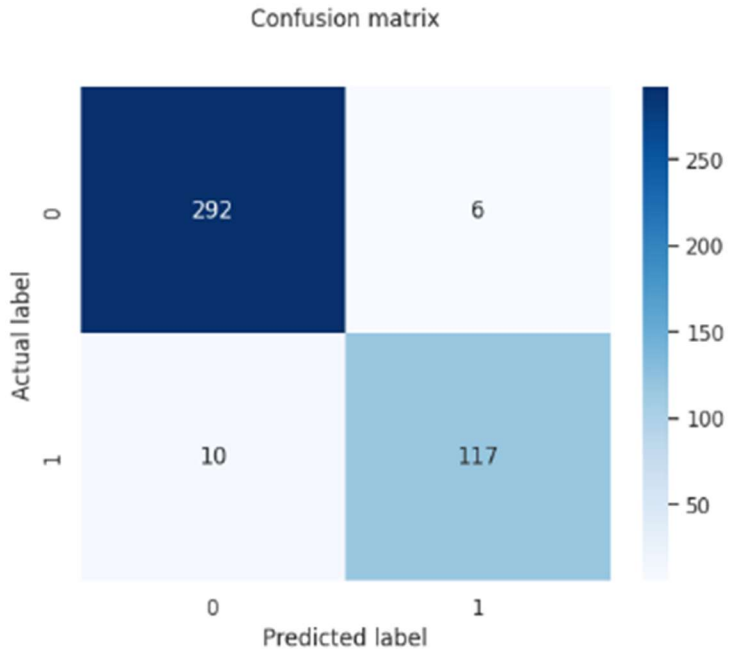
Figure 23 shows four numeric values which represent True Positive, True Negative, False Negative, False Positive. The values for True Positive and True False Positive are 660 and 322 which totals to 982 true/correct predictions and False Positive and False Negative are 3 and 6 which totals to 9 false/incorrect predictions for the confusion matrix calculated for Random Forest.

The model has a high precision, which indicates that the accuracy of the points found is high, and a high recall, indicating that the model does not miss a significant number of customers who are late on their loans. Furthermore, the model's high F1-score and overall accuracy of 99.09% (is equal to the accuracy of the decision tree model) indicate its ability to predict is very good.

 # Code

```
# get predicted y on test data

RF_y_pred = RF_classifier.predict(X_test)

p = sns.heatmap(pd.DataFrame(confusion_matrix(y_test,RF_y_pred)),
annot=True, cmap="Blues" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

print('Classification report:')
print(classification_report(y_test,RF_y_pred))
print('Random Forest accuracy: ', round(accuracy_score(y_test,
RF_y_pred),4))
```

 # Result



Classification report:

	precision	recall	f1-score	support
0	0.97	0.98	0.97	298
1	0.95	0.92	0.94	127
accuracy			0.96	425
macro avg	0.96	0.95	0.95	425
weighted avg	0.96	0.96	0.96	425

Random Forest accuracy: 0.9624

Figure 24: Results on the test set of Random Forest

Comment:
Figure 24 shows four numeric values which represent True Positive, True Negative, False Negative, False Positive. The values for True Positive and True False Positive are 292 and 117 which totals to 409 true/correct predictions and False Positive and False Negative are 6 and 10 which totals to 16 false/incorrect predictions for the confusion matrix calculated for Random Forest.

The model has a high precision, which indicates that the accuracy of the points found is high, and a high recall, indicating that the model does not miss a significant number of customers who are late on their loans. Furthermore, the model's high F1-score and overall accuracy of 96.24% indicate its ability to predict is very good.

The Random Forest model seems to perform well, with high precision, recall, and F1-scores for both the training set and the test set.

b. Feature importance

```
# 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)
```

 # Result

Table 14: Random Forest Feature Importance

	FeatureName	Random Forest Feature Importance
0	tai san dam bao	498.568
2	thu nhap	208.876
1	so tien vay	139.021
4	thoi gian vay	87.715
3	muc dich vay	65.820

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

 # Result

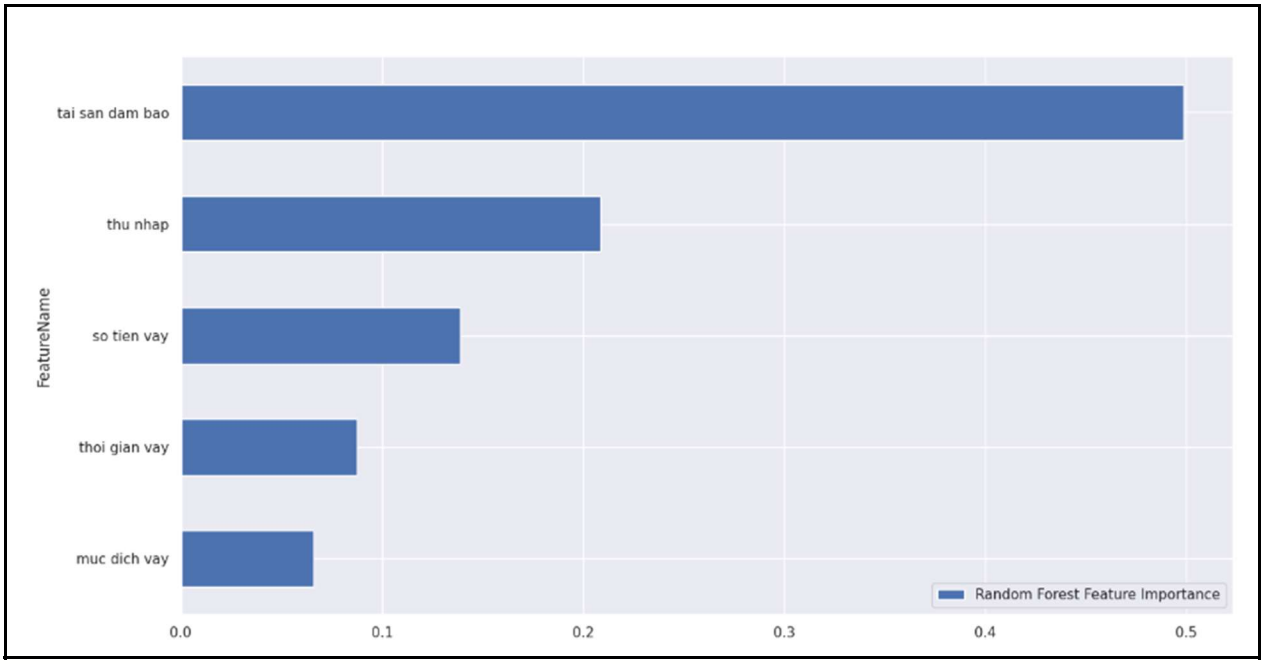


Figure 25: Random Forest Feature Importance

c. ROC Curve

```
# Code
y_pred_prob_test_rf = RF_classifier.predict_proba(X_test)[:, 1]

fpr, tpr, thres = roc_curve(y_test, y_pred_prob_test_rf)
RF_roc_auc = auc(fpr, tpr)
_plot_roc_curve(fpr, tpr, thres, RF_roc_auc)
```

```
# Result
```

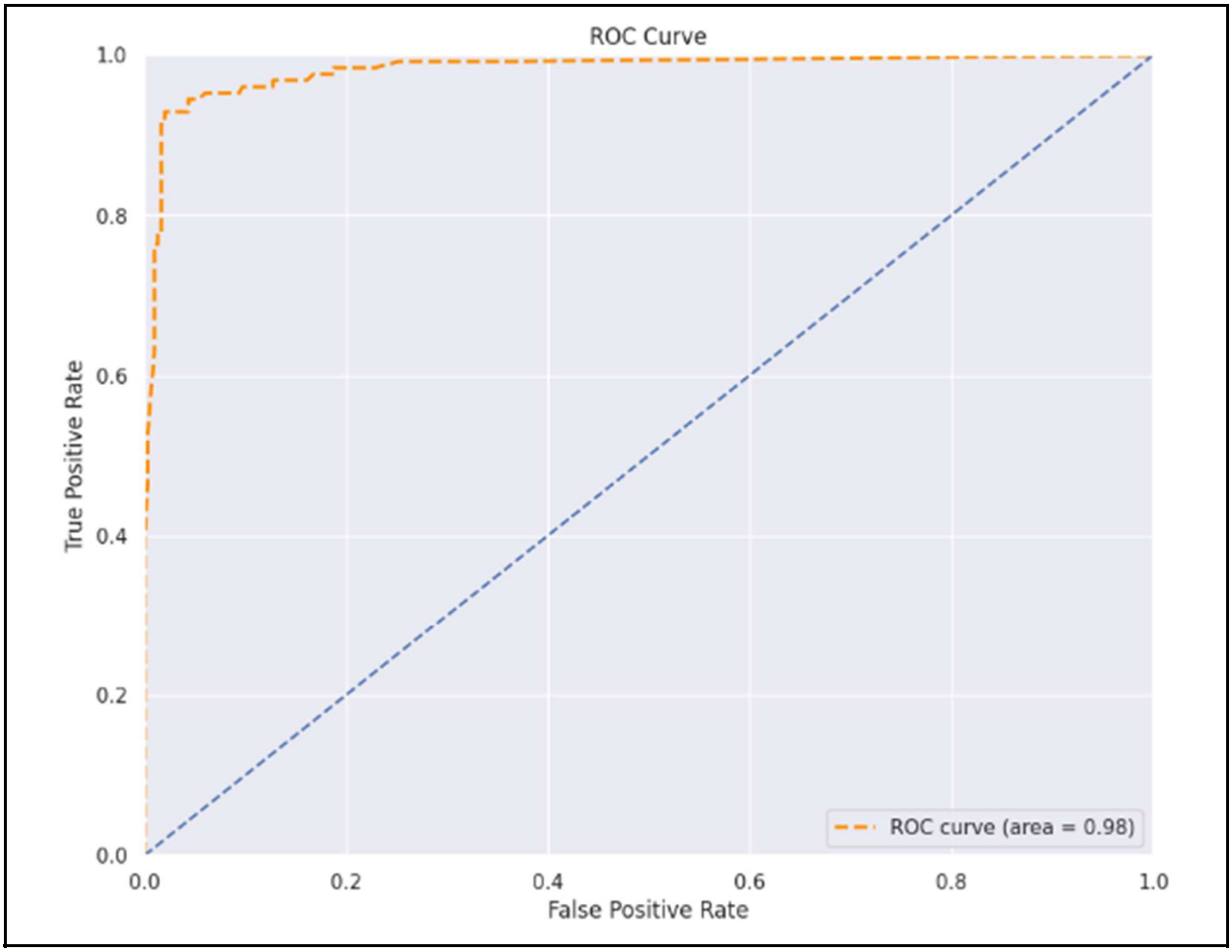


Figure 26: ROC curve of Random Forest model

Comment:

The ROC line of the Random Forest model is close to the upper left corner of the, which shows the model's classification and forecasting ability is good. In addition, Area Under The Curve (AUC) = 0.98 (close to 1) also indicates that this model has a very good performance.

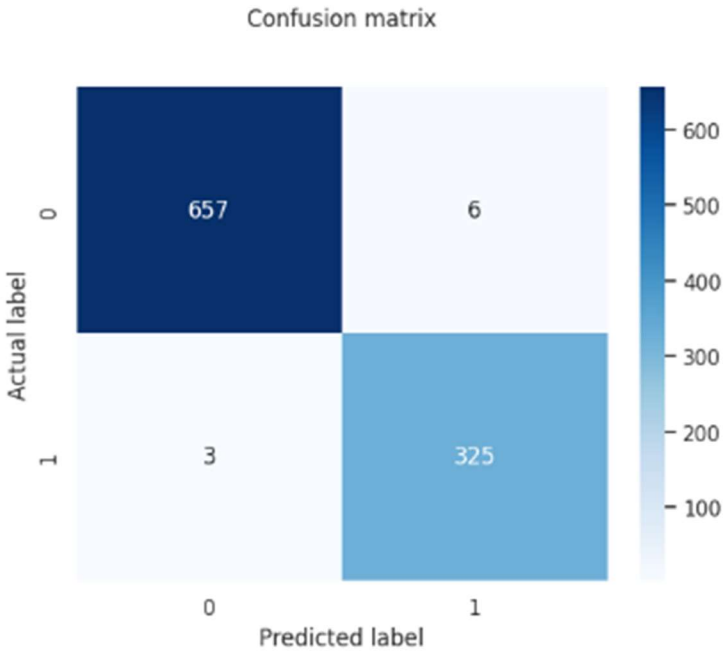
4. XGBoost Classifier

a. Prediction

```
# Code
import xgboost as xgb
from xgboost.sklearn import XGBClassifier
#Difine model
XGB_classifier = XGBClassifier()
# Fit model
XGB_classifier.fit(X_train, y_train.ravel())
# get predicted y on train data
y_pred = XGB_classifier.predict(X_train)
```

```
p = sns.heatmap(pd.DataFrame(confusion_matrix(y_train,y_pred)),
annot=True, cmap="Blues" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

print('Classification report:')
print(classification_report(y_train,y_pred))
print('XGBoost accuracy: ', round(accuracy_score(y_train, y_pred),4))
```

 # Result



Classification report:

	precision	recall	f1-score	support
0	1.00	0.99	0.99	663
1	0.98	0.99	0.99	328
accuracy			0.99	991
macro avg	0.99	0.99	0.99	991
weighted avg	0.99	0.99	0.99	991

XGBoost accuracy: 0.9909

Figure 27: Results on the training set of XGBoost

Comment:

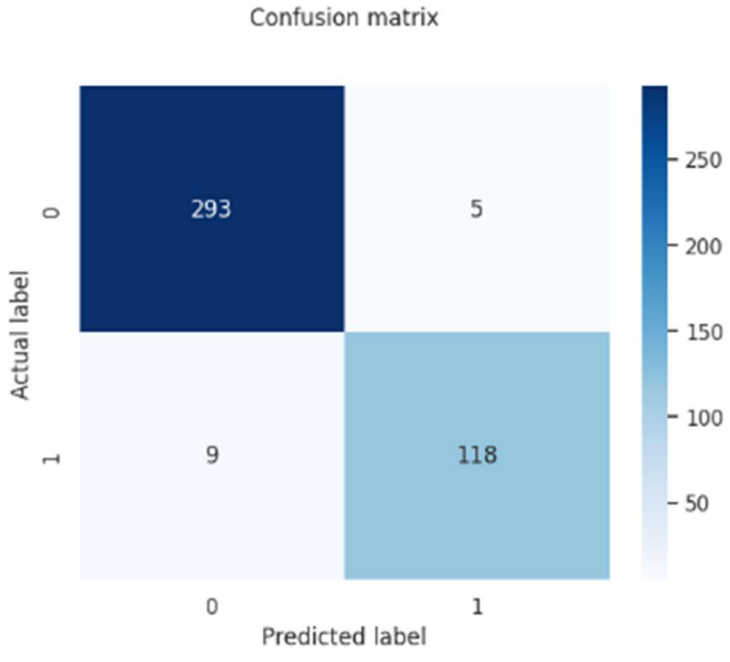
Figure 27 shows four numeric values which represent True Positive, True Negative, False Negative, False Positive. The values for True Positive and True False Positive are 657 and 325 which totals to 982 true/correct predictions and False Positive and False Negative are 6 and 3 which totals to 9 false/incorrect predictions for the confusion matrix calculated for XGBoost.

The model has a high precision, which indicates that the accuracy of the points found is high, and a high recall, indicating that the model does not miss a significant number of customers who are late on their loans. Furthermore, the model's high F1-score and overall accuracy of 99.09% (is equal to the accuracy of the decision tree model and random forest model) indicate its ability to predict is very good.

 # Code

```
# get predicted y on test data

XGB_y_pred = XGB_classifier.predict(X_test)

p = sns.heatmap(pd.DataFrame(confusion_matrix(y_test,XGB_y_pred)),
annot=True, cmap="Blues" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

print('Classification report:')
print(classification_report(y_test,XGB_y_pred))
print('XGBoost accuracy: ', round(accuracy_score(y_test, XGB_y_pred),4))
```

 # Result



Classification report:

	precision	recall	f1-score	support
0	0.97	0.98	0.98	298
1	0.96	0.93	0.94	127
accuracy			0.97	425
macro avg	0.96	0.96	0.96	425
weighted avg	0.97	0.97	0.97	425

XGBoost accuracy: 0.9671

Figure 28: Results on the test set of XGBoost

Comment:
Figure 28 shows four numeric values which represent True Positive, True Negative, False Negative, False Positive. The values for True Positive and True False Positive are 293 and 118 which totals to 411 true/correct predictions and False Positive and False Negative are 5 and 9 which totals to 14 false/incorrect predictions for the confusion matrix calculated for XGBoost.

The model has a high precision, which indicates that the accuracy of the points found is high, and a high recall, indicating that the model does not miss a significant number of customers who are late on their loans. Furthermore, the model's high F1-score and overall accuracy of 96.71% indicate its ability to predict is very good.

The XGBoost model seems to perform well, with high precision, recall, and F1-scores for both the training set and the test set.

b. Feature Importance

```
# 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)
```

 # Result

Table 15: XGBoost Feature Importance

	FeatureName	XGBoost Feature Importance

0	tai san dam bao	753.857
3	muc dich vay	87.843
1	so tien vay	70.553
2	thu nhap	50.192
4	thoi gian vay	37.555

 # Code

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

 # Result

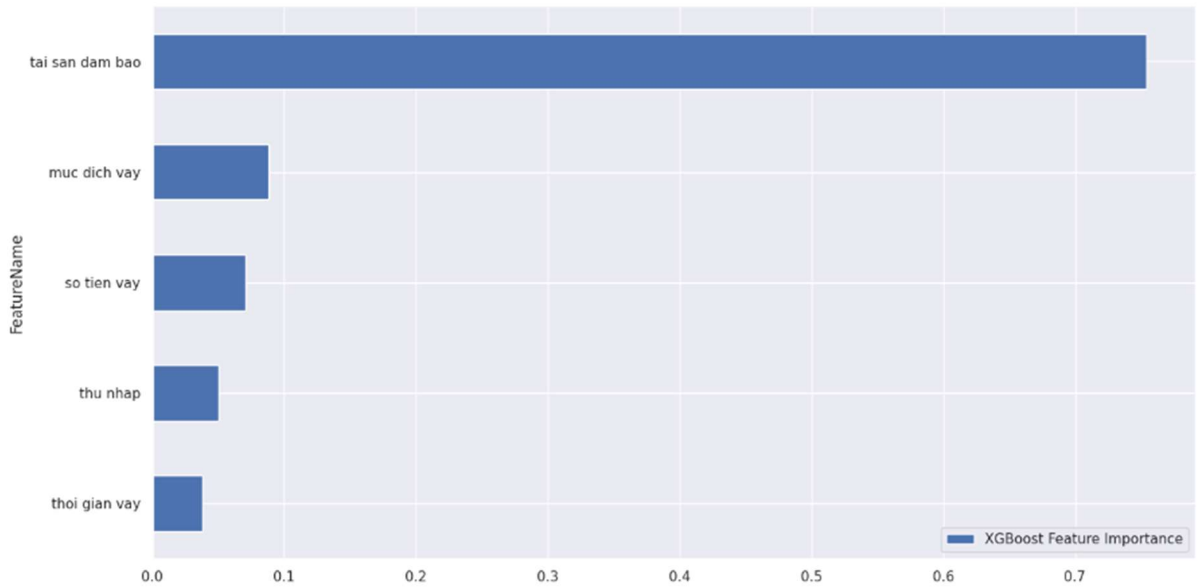


Figure 29: XGBoost Feature Importance

c. ROC Curve

 # Code

```
y_pred_prob_test = XGB_classifier.predict_proba(X_test)[:, 1]

fpr, tpr, thres = roc_curve(y_test, y_pred_prob_test)
XGB_roc_auc = auc(fpr, tpr)
_plot_roc_curve(fpr, tpr, thres, XGB_roc_auc)
```

 # Result

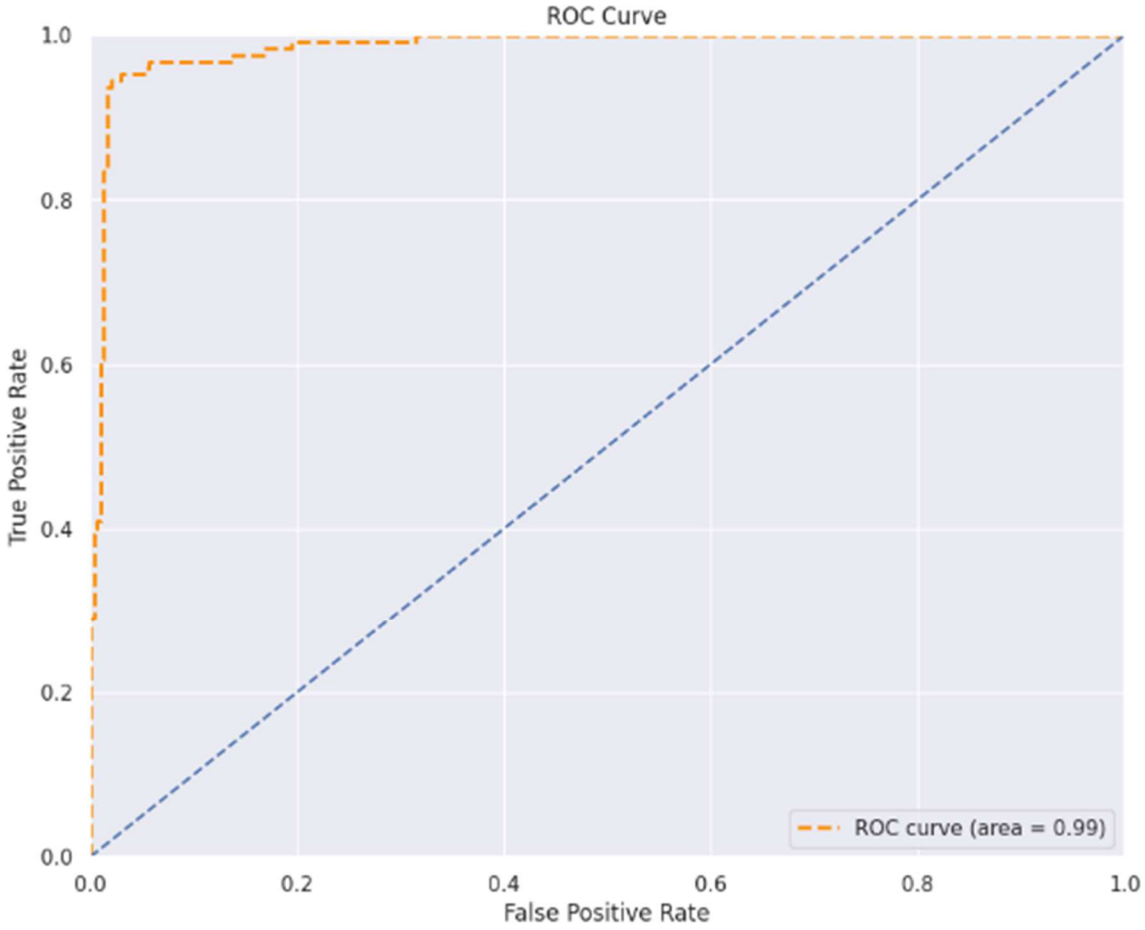


Figure 30: ROC curve of XGBoost model

Comment:
The ROC line of the XGBoost model model is close to the upper left corner of the, which shows the model's classification and forecasting ability is good. In addition, Area Under The Curve (AUC) = 0.99 (close to 1) also indicates that this model has a very good performance.

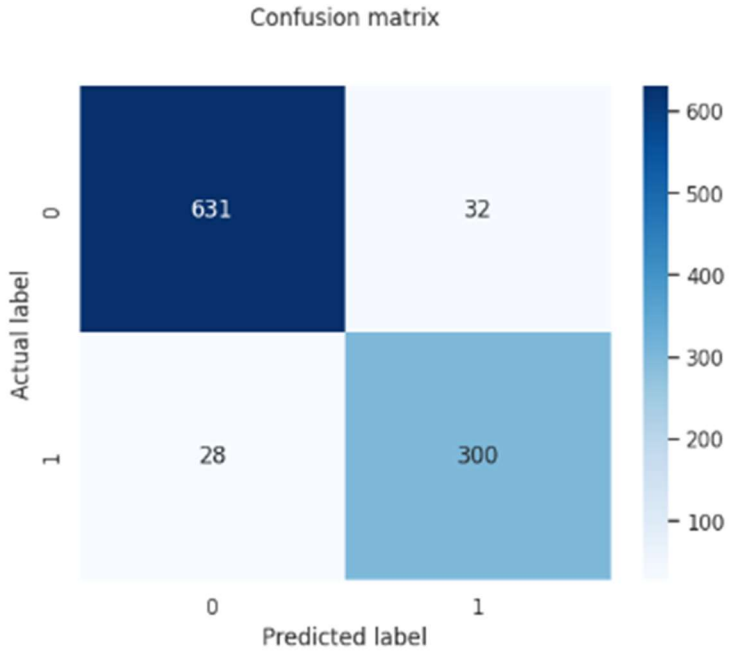
5. Ada Boost

a. Prediction

```
# Code
from sklearn.ensemble import AdaBoostClassifier

#Difine model
ada_classifier = AdaBoostClassifier()
#Fit model
ada_classifier.fit(X_train, y_train.ravel())
```

```
# get predicted y on train data
y_pred = ada_classifier.predict(X_train)

p = sns.heatmap(pd.DataFrame(confusion_matrix(y_train,y_pred)),
annot=True, cmap="Blues" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
print('Classification report:')
print(classification_report(y_train,y_pred))
print('Ada Boost accuracy: ', round(accuracy_score(y_train, y_pred),4))
```

 # Result



Classification report:

	precision	recall	f1-score	support
0	0.96	0.95	0.95	663
1	0.90	0.91	0.91	328
accuracy			0.94	991
macro avg	0.93	0.93	0.93	991
weighted avg	0.94	0.94	0.94	991

```
Ada Boost accuracy:  0.9395
```

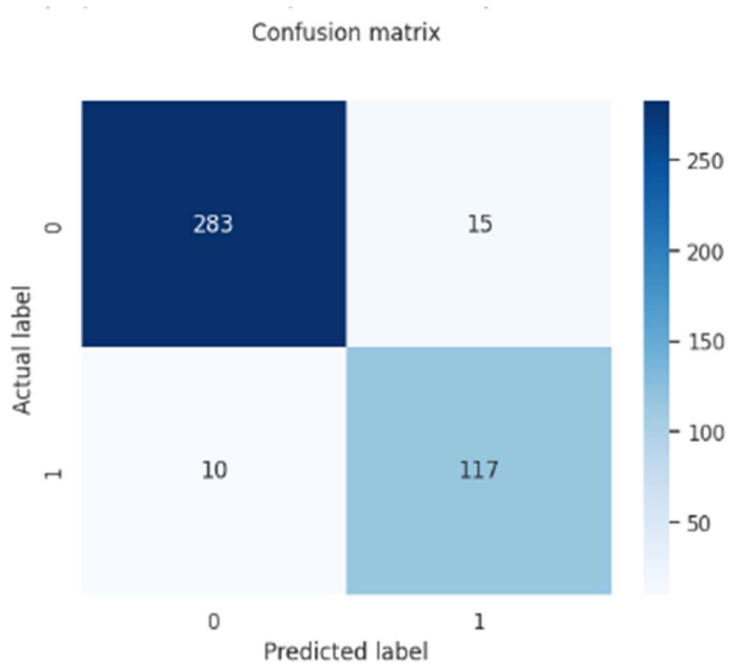
Figure 31: Results on the training set of Ada Boost

Comment:
Figure 31 shows four numeric values which represent True Positive, True Negative, False Negative, False Positive. The values for True Positive and True False Positive are 631 and 300 which totals to 931 true/correct predictions and False Positive and False Negative are 32 and 28 which totals to 60 false/incorrect predictions for the confusion matrix calculated for Ada Boost.

The model has a high precision, which indicates that the accuracy of the points found is high, and a high recall, indicating that the model does not miss a significant number of customers who are late on their loans. Furthermore, the model's high F1-score and overall accuracy of 93.95% indicate its ability to predict is good.

```
# Code
# get predicted y on test data
Ada_y_pred = ada_classifier.predict(X_test)

p = sns.heatmap(pd.DataFrame(confusion_matrix(y_test,Ada_y_pred)),
annot=True, cmap="Blues" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

print('Classification report:')
print(classification_report(y_test,Ada_y_pred))
print('Ada Boost accuracy: ', round(accuracy_score(y_test,
Ada_y_pred),4))
```

```
# Result
```



Classification report:

	precision	recall	f1-score	support
0	0.97	0.95	0.96	298
1	0.89	0.92	0.90	127
accuracy			0.94	425
macro avg	0.93	0.94	0.93	425
weighted avg	0.94	0.94	0.94	425

Ada Boost accuracy: 0.9412

Figure 32: Results on the test set of Ada Boost

Comment:
Figure 32 shows four numeric values which represent True Positive, True Negative, False Negative, False Positive. The values for True Positive and True False Positive are 283 and 117 which totals to 400 true/correct predictions and False Positive and False Negative are 15 and 10 which totals to 25 false/incorrect predictions for the confusion matrix calculated for Ada Boost.

The model has a high precision, which indicates that the accuracy of the points found is high, and a high recall, indicating that the model does not miss a significant number of customers who are late on their loans. Furthermore, the model's high F1-score and overall accuracy of 94.12% indicate its ability to predict is good.

The Ada Boost model seems to perform well, with high precision, recall, and F1-scores for both the training set and the test set.

b. Feature Importance

```
# 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)
```

 # Result

Table 16: AdaBoost Feature Importance

	FeatureName	AdaBoost Feature Importance
1	so tien vay	0.38
2	thu nhap	0.32
0	tai san dam bao	0.12
4	thoi gian vay	0.12
3	muc dich vay	0.06

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

 # Result

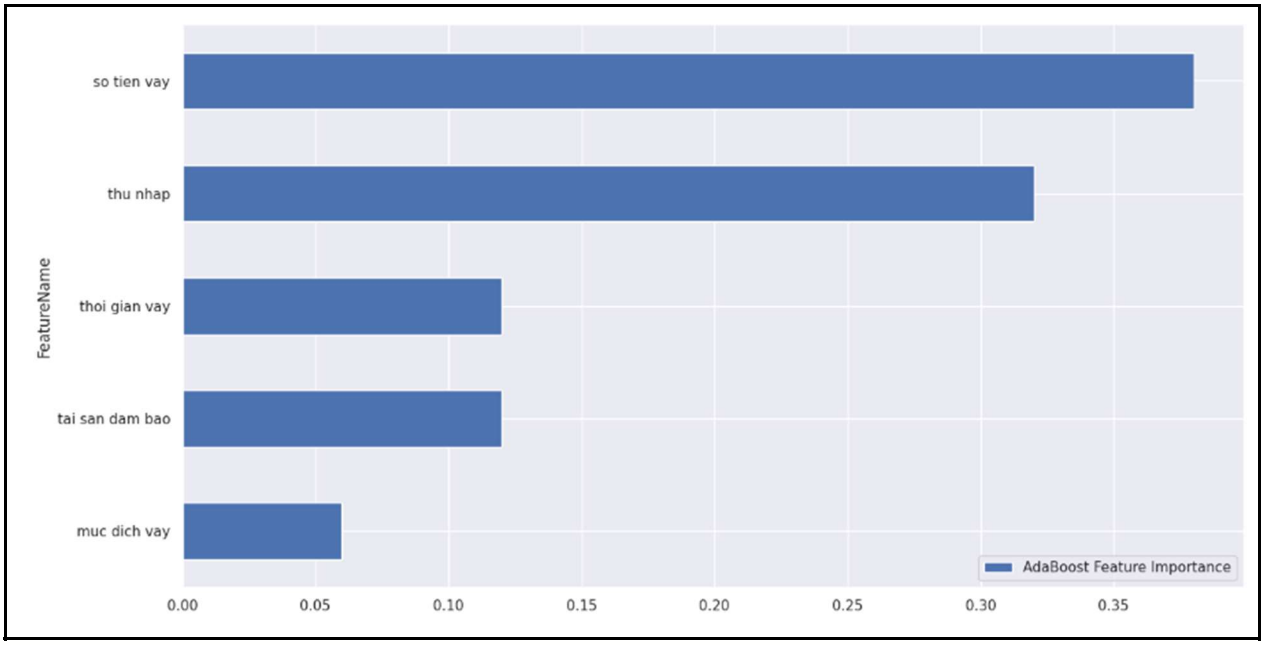


Figure 33: AdaBoost Feature Importance

c. ROC Curve

```
# Code
y_pred_prob_test = ada_classifier.predict_proba(X_test)[:, 1]

fpr, tpr, thres = roc_curve(y_test, y_pred_prob_test)
Ada_roc_auc = auc(fpr, tpr)

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

```
# Result
```

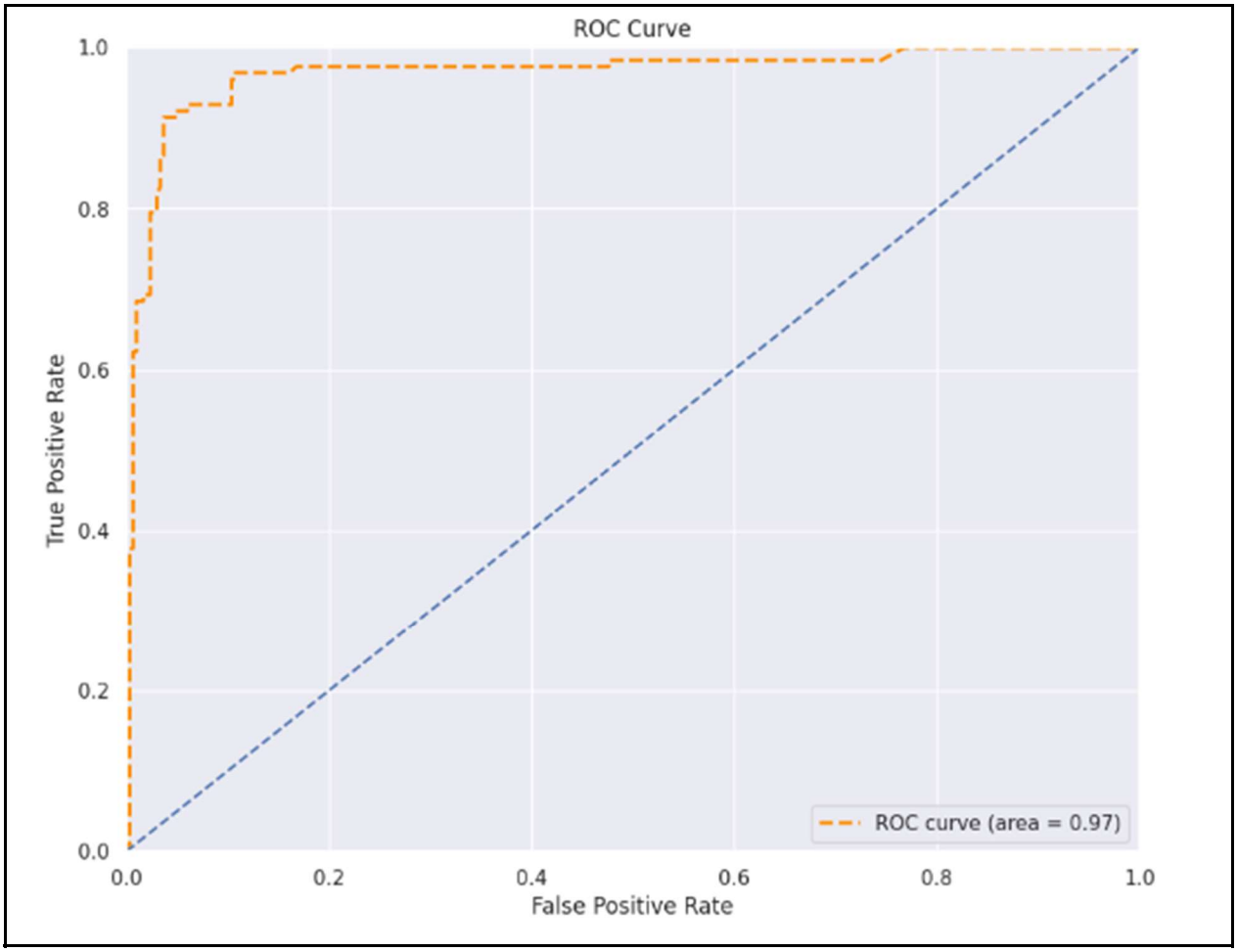


Figure 34: ROC curve of Ada Boost model

Comment:

The ROC line of the Ada Boost model is close to the upper left corner of the, which shows the model's classification and forecasting ability is good. In addition, Area Under The Curve (AUC) = 0.97 (close to 1) also indicates that this model has a very good performance.

6. KNN Classifier

a. Prediction

```
# Code
from sklearn.neighbors import KNeighborsClassifier

k_values = [i for i in range (1,15)]
scores = []
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
```

```
    score = cross_val_score(knn, X, y, cv=5)
    scores.append(np.mean(score))
sns.lineplot(x = k_values, y = scores, marker = 'o')
plt.xlabel("K Values")
plt.ylabel("Accuracy Score")
```

 # Result

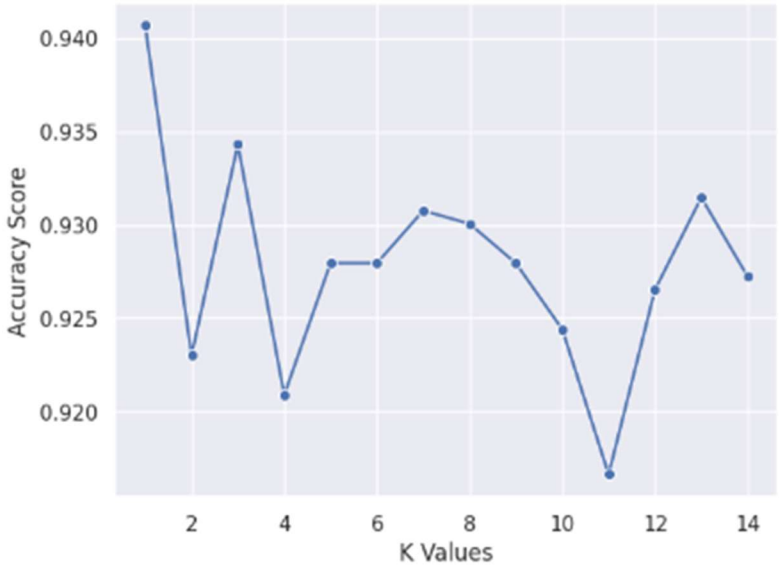


Figure 35: K Values of KNN

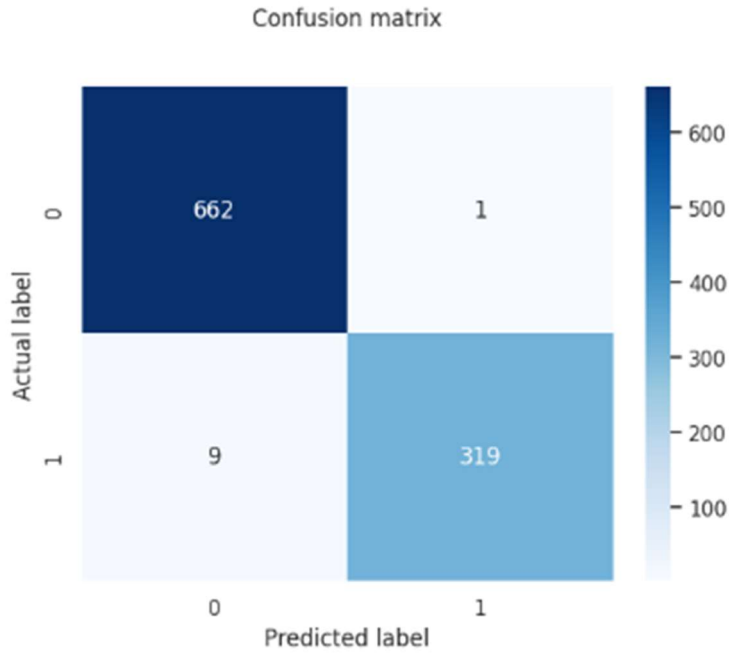
The best result is captured at k = 1 hence 1 is used for the final model.

 # Code

```
#Difine model

knn_classifier = KNeighborsClassifier(1)
#Fit model
knn_classifier.fit(X_train,y_train)
# get predicted y on train data
y_pred = knn_classifier.predict(X_train)

p = sns.heatmap(pd.DataFrame(confusion_matrix(y_train,y_pred)),
annot=True, cmap="Blues" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

print('Classification report:')
print(classification_report(y_train,y_pred))
```

```
print('KNN accuracy: ', round(accuracy_score(y_train, y_pred),4))
```

 # Result




```
Classification report:
```

	precision	recall	f1-score	support
0	0.99	1.00	0.99	663
1	1.00	0.97	0.98	328
accuracy			0.99	991
macro avg	0.99	0.99	0.99	991
weighted avg	0.99	0.99	0.99	991

```
KNN accuracy:  0.9899
```

Figure 36: Results on the training set of KNN

Comment:

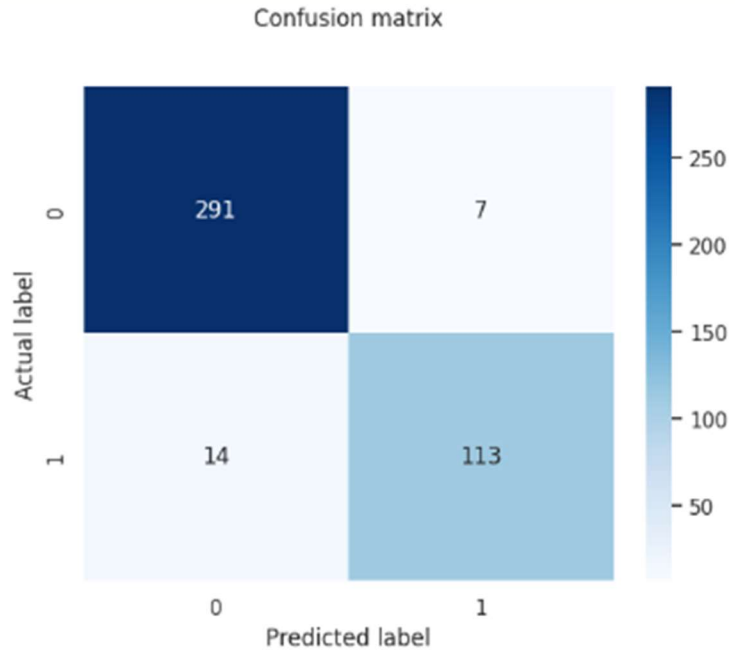
Figure 36 shows four numeric values which represent True Positive, True Negative, False Negative, False Positive. The values for True Positive and True False Positive are 662 and 319 which totals to 981 true/correct predictions and False Positive and False Negative are 1 and 9 which totals to 10 false/incorrect predictions for the confusion matrix calculated for KNN.

The model has a high precision, which indicates that the accuracy of the points found is high, and a high recall, indicating that the model does not miss a significant number of customers who are late on their loans. Furthermore, the model's high F1-score and overall accuracy of 98.99% indicate its ability to predict is good.

 # Code

```
# get predicted y on test data
knn_y_pred = knn_classifier.predict(X_test)

p = sns.heatmap(pd.DataFrame(confusion_matrix(y_test,knn_y_pred)),
annot=True, cmap="Blues" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

print('Classification report:')
print(classification_report(y_test,knn_y_pred))
print('KNN accuracy: ', round(accuracy_score(y_test, knn_y_pred),4))
```

 # Result



Classification report:

	precision	recall	f1-score	support
0	0.95	0.98	0.97	298
1	0.94	0.89	0.91	127
accuracy			0.95	425

macro avg	0.95	0.93	0.94	425
weighted avg	0.95	0.95	0.95	425

```
KNN accuracy:  0.9506
```

Figure 37: Results on the test set of KNN

Comment:
Figure 37 shows four numeric values which represent True Positive, True Negative, False Negative, False Positive. The values for True Positive and True False Positive are 291 and 113 which totals to 404 true/correct predictions and False Positive and False Negative are 7 and 14 which totals to 21 false/incorrect predictions for the confusion matrix calculated for KNN.

The model has a high precision, which indicates that the accuracy of the points found is high, and a high recall, indicating that the model does not miss a significant number of customers who are late on their loans. Furthermore, the model's high F1-score and overall accuracy of 95.06% indicate its ability to predict is good.

The KNN model seems to perform well, with high precision, recall, and F1-scores for both the training set and the test set.

b. ROC curve

```
# Code
y_pred_prob_test = knn_classifier.predict_proba(X_test)[:, 1]

fpr, tpr, thres = roc_curve(y_test, y_pred_prob_test)
knn_roc_auc = auc(fpr, tpr)

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

```
# Result
```

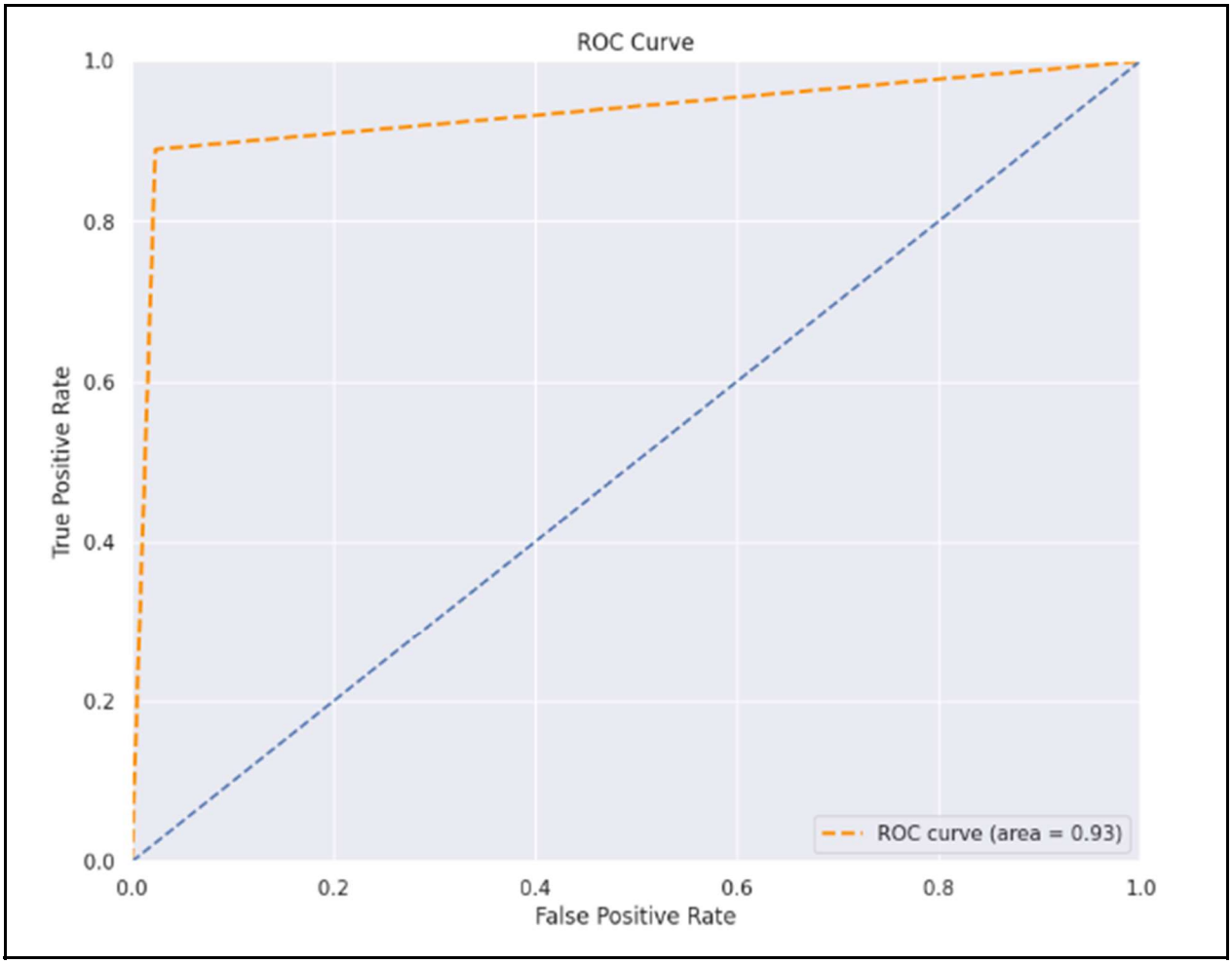


Figure 38: ROC curve of KNN model

Comment:

The ROC line of the KNN model is close to the upper left corner of the, which shows the model's classification and forecasting ability is good. In addition, Area Under The Curve (AUC) = 0.93 (close to 1) also indicates that this model has a good performance.

VI. Compare models

```
# Code
d={

'': ['Logistic Regression','Decision Tree','Random Forest','XGBoost','Ada
Boost','KNN'],
'Accuracy': [LR_classifier.score(X_test, y_test),
DT_classifier.score(X_test,y_test),RF_classifier.score(X_test, y_test),
             XGB_classifier.score(X_test,y_test),ada_classifier.score(X_t
est, y_test),knn_classifier.score(X_test, y_test)],
'Precision': [precision_score(y_test, LR_y_pred), precision_score(y_test,
DT_y_pred),precision_score(y_test, RF_y_pred),
```

```
            precision_score(y_test, XGB_y_pred),precision_score(y_test,
Ada_y_pred),precision_score(y_test, knn_y_pred)],
'Recall': [recall_score(y_test, LR_y_pred), recall_score(y_test,
DT_y_pred),recall_score(y_test, RF_y_pred),
           recall_score(y_test, XGB_y_pred),recall_score(y_test,
Ada_y_pred),recall_score(y_test, knn_y_pred)],
'F1': [f1_score(y_test, LR_y_pred), f1_score(y_test,
DT_y_pred),f1_score(y_test, RF_y_pred),
       f1_score(y_test, XGB_y_pred),f1_score(y_test,
Ada_y_pred),f1_score(y_test, knn_y_pred)],
'ROC AUC': [LR_roc_auc, DT_roc_auc, RF_roc_auc, XGB_roc_auc, Ada_roc_auc,
knn_roc_auc]
}

results=pd.DataFrame(data=d).round(4).set_index('')
results
```

 # Result

Table 17: The model's indexs comparison table

	Accuracy	Precision	Recall	F1	ROC AUC
Logistic Regression	9.153	8.699	8.425	8.560	9.484
Decision Tree	9.412	9.322	8.661	8.980	9.414
Random Forest	9.647	9.516	9.291	9.402	9.856
XGBoost	9.671	9.593	9.291	9.440	9.851
Ada Boost	9.412	8.864	9.213	9.035	9.680
KNN	9.506	9.417	8.898	9.150	9.331

 # Code

```
# Define the models to evaluate

models = [
    ('LR', LogisticRegression()),
    ('DT', DecisionTreeClassifier()),
    ('RF', RandomForestClassifier()),
    ('XGB', XGBClassifier()),
    ('ADB', AdaBoostClassifier()),
    ('KNN', KNeighborsClassifier()),
]

# 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
plt.figure(figsize = (15,8))
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()
```

 # Result

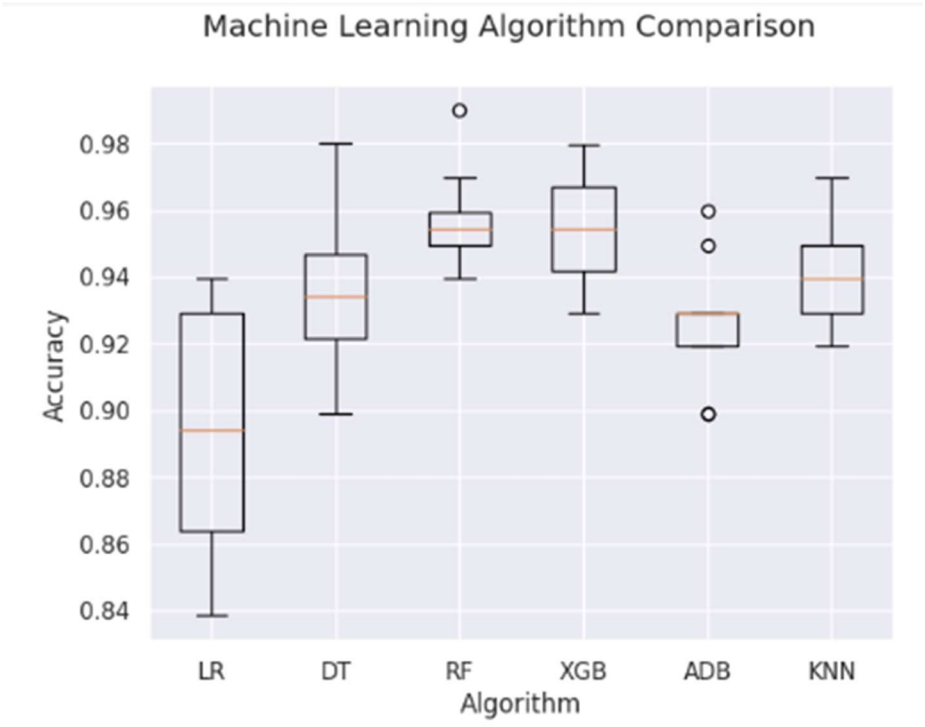


Figure 39: Machine learning algorithm comparison with accuracy

 # Code

```
def _plot_roc_curve(fpr, tpr, thres, roc_auc, label):

    plt.plot(fpr, tpr, label=f'{label} (AUC = {roc_auc:.2f})')

fig, ax = plt.subplots(figsize=(10, 6))
fig.suptitle('Machine Learning Algorithm Comparison')

# Plot ROC curves
_plot_roc_curve(fpr, tpr, thres, LR_roc_auc, 'Logistic Regression')
_plot_roc_curve(fpr, tpr, thres, DT_roc_auc, 'Decision Tree')
_plot_roc_curve(fpr, tpr, thres, RF_roc_auc, 'Random Forest')
_plot_roc_curve(fpr, tpr, thres, XGB_roc_auc, 'XGBoost')
_plot_roc_curve(fpr, tpr, thres, Ada_roc_auc, 'AdaBoost')
_plot_roc_curve(fpr, tpr, thres, knn_roc_auc, 'KNN')

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')

plt.show()
```

 # Result

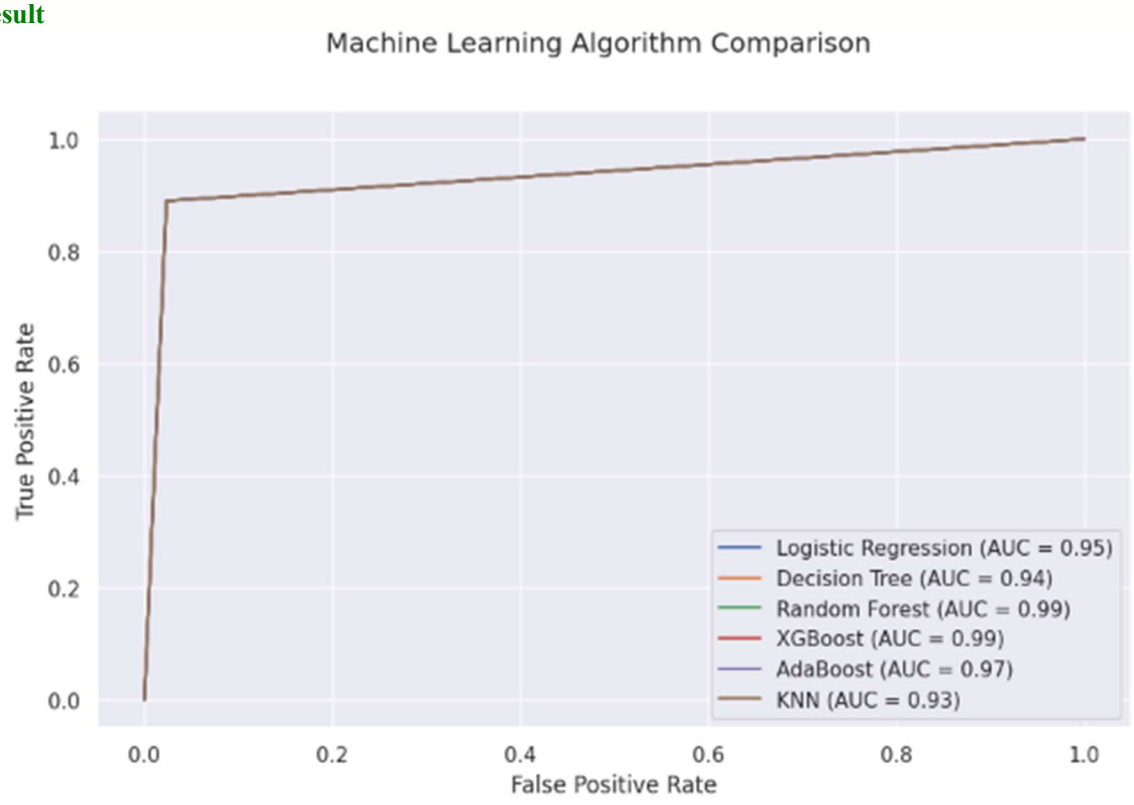


Figure 39: Machine learning algorithm comparison with AUC

Comment:

According to the comparison above, XGBoost consistently performs well across all evaluation metrics, with high scores in accuracy, precision, recall, F1-score, and ROC AUC. Random Forest also performs strongly, often matching or closely following XGBoost. Decision Tree, Ada Boost, and KNN generally have slightly lower scores across most metrics, while Logistic Regression consistently has the lowest scores.

VII. Predict new customer

I use the XGBoost Classifier model to predict a customer's ability to repay a loan against new, unseen data.

```
# Code
#import new data
new_data = pd.read_excel('/content/drive/MyDrive/HỌC TẬP/NĂM 3/HK2/mô
hình rủi ro tín dụng/ck/New data.xlsx')
new_data.head()
```

 # Result

Table 18: Head of new data

id	gioi tinh	thu nhap	tien dien/ thang	so tien vay	muc dich vay	gia dinh	so nam lam viec	tuoi	thoi gian vay	trinh do hoc van	tai san dam bao
1	0	23500000	1050000	455000000	2	1	12	34	24	2	1
2	0	16500000	955000	250000000	3	1	9	41	19	3	1
3	1	13800000	580000	100000000	1	3	8	29	12	4	3
4	0	18500000	1100000	250000000	4	1	10	36	18	2	1
5	1	20100000	690000	160000000	3	2	5	28	15	2	2

```
# Code
features = ['tai san dam bao', 'so tien vay', 'thu nhap', 'muc dich vay',

       'thoi gian vay']
X_new = new_data[features].values

#standardize the input data
std_data = scaler.transform(X_new)

prediction = XGB_classifier.predict(std_data)
```

```
results = pd.DataFrame({'Prediction': prediction})
results['Note'] = results['Prediction'].apply(lambda x: 'Will pay on-
time' if x == 0 else 'Will pay late')

print(results)
```

 # Result

Table 19: predicted results with XGB Classifier

	Prediction	Note
0	0	Will pay on-time
1	0	Will pay on-time
2	1	Will pay late
3	0	Will pay on-time
4	0	Will pay on-time
5	1	Will pay late
6	0	Will pay on-time
7	1	Will pay late
8	0	Will pay on-time
9	1	Will pay late
10	1	Will pay late
11	0	Will pay on-time
12	1	Will pay late
13	0	Will pay on-time
14	1	Will pay late

CONCLUSION

The XGBoost Classifier model and the Random Forest model are the two models that give the best prediction results. To be able to fix and perfect this loan prediction model, a combined machine learning model can be used.

In order to minimize overdue debts and bad debts and make appropriate decisions, appraisal is one of the important steps in making lending decisions to help banks prevent risks to customers. debts. Therefore, it is very necessary to detect risks in the process, customer information, and customer data warehouse to help the bank reduce the risk of overdue debt and possible bad debt. make decisions on suitable loan products, minimizing the risks affecting the development of the bank in the integration process. Banks can consider choosing a suitable prediction model for their bank.

REFERENCES

Vẹn, L. P. (2022). Risks in the individual customer credit appraisal at Vietnamese join stock commercial banks. Banking reviews. Retrieved May 20, 2023 from: https://tapchinganhang.gov.vn/nhan-dinh-mot-so-rui-ro-trong-tham-dinh-tin-dung-khach-hang-ca-nhan-tai-ngan-hang-thuong-mai-co-phan.htm

Correa Bahnsen. A. (2016). Feature Engineering Strategies for Credit Card Fraud Detection. Expert Systems with Applications, 51, 134-142.

Makowsi P. Credit scoring branches out. Credit World，1985,75(1):30-37.

Carter C, Catlett J. Assessing credit card applications using machine learning. IEEE expert, 1987,2(3):71-79.

Henning, J. I., Bougard, D. A., Jordaan, H., & Matthews, N. (2019). Factors affecting successful agricultural loan applications: the case of a South African credit provider. *Agriculture*, *9*(11), 243.

Konovalova, N., Kristovska, I., & Kudinska, M. (2016). Credit risk management in commercial banks. *Polish Journal of Management Studies*, *13*.

Anand, M., Velu, A., & Whig, P. (2022). Prediction of loan behaviour with machine learning models for secure banking. *Journal of Computer Science and Engineering (JCSE)*, *3*(1), 1-13.

Kim, K., Kim, J., Cho, K., Kim, J.-G., & Hyun, S. (2018). Analysis of the Resilience of Common-Pool Resources during Globalization: The Case of Jeju Common Ranches in Korea. Sustainability, 10(12), 4346.

Siddique, K., Akhtar, Z., Lee, H., Kim, W., & Kim, Y. (2017). Toward Bulk Synchronous Parallel-Based Machine Learning Techniques for Anomaly Detection in High-Speed Big Data Networks. Symmetry, 9(9), 197.