

**UNIVERSITY OF ECONOMICS AND LAW**

**VNUHCM**



**THE FINAL REPORT:**

**CREDIT RISK MODELING IN R/PYTHON**

**YEAR 2023**

**<FORECASTING THE REPAYMENT ABILITY OF INDIVIDUAL CUSTOMERS>**

**Syllabus: K20414C\_ Fintech**

**Course: 222CN1001**

**Student: Lương Thị Mỹ Tâm**

**MSSV: K204141929**

**UNIVERSITY OF ECONOMICS AND LAW**

**VNUHCM**

**THE FINAL REPORT:**

**CREDIT RISK MODELING IN R/PYTHON**

**YEAR 2023**

**<FORECASTING THE REPAYMENT ABILITY OF INDIVIDUAL CUSTOMERS>**

**DECLARATION**

I hereby declare that the report "Forecasting the repayment ability of individual customers" is the result of my work under the guidance of Dr. Pham Thi Thanh Xuan, within the framework of the subject 'Credit Risk Modeling in R/Python'.

**TABLE OF CONTENTS**

[**DECLARATION**](#_heading=h.gjdgxs) **1**

[**TABLE OF CONTENTS**](#_heading=h.30j0zll) **2**

[**LIST OF TABLES/FIGURES**](#_heading=h.3znysh7) **3**

[**INTRODUCTION**](#_heading=h.tyjcwt) **6**

[1. Requirement](#_heading=h.3dy6vkm) 6

[2. Model](#_heading=h.1t3h5sf) 6

[3. Data Description](#_heading=h.4d34og8) 6

[**THE CONTENT**](#_heading=h.2s8eyo1) **7**

[1. Implementation process](#_heading=h.17dp8vu) 7

[2. Step 1: Importing data](#_heading=h.3rdcrjn) 8

[3. Step 2: Checking data](#_heading=h.26in1rg) 8

[4. Step 3: Data Processing](#_heading=h.lnxbz9) 9

[5. Step 4: Descriptive Statistics](#_heading=h.35nkun2) 11

[6. Step 5: Correlation](#_heading=h.44sinio) 17

[7. Step 6: Using Logistic Regression to predict](#_heading=h.2jxsxqh) 18

[8. Step 7: Using Decision Tree to predict](#_heading=h.4i7ojhp) 25

[9. Step 8: Using Random Forest to predict](#_heading=h.2bn6wsx) 31

[10. Step 9: Using SVM to predict](#_heading=h.1pxezwc) 36

[11. Step 10: Using Naive Bayes to predict](#_heading=h.147n2zr) 40

[12. Step 11: Using KNN to predict](#_heading=h.ihv636) 45

[13. Step 12: Improving model quality with upsample/downsample](#_heading=h.41mghml) 51

[14. Step 13: Predict new customer](#_heading=h.3fwokq0) 53

[**CONCLUSION**](#_heading=h.1v1yuxt) **54**

[**REFERENCE**](#_heading=h.4f1mdlm) **5**

# 

# **LIST OF TABLES/FIGURES**

|  | **Name of tables/figures** | **Page** |
| --- | --- | --- |
| 1 | Table 1: The Data set | 8 |
| 2 | Table 2: The information of the data set | 9 |
| 3 | Table 3: The data set after processing | 11 |
| 4 | Table 4: Descriptive Statistics of numerical columns | 13 |
| 5 | Table 5: Descriptive Statistics of categorical columns | 15 |
| 6 | Table 6: Correlation | 17 |
| 7 | Table 7: The result of Logistic Regression Model | 20 |
| 8 | Table 8: Regression report table | 22 |
| 9 | Figure 9: ROC Curve | 23 |
| 10 | Table 10: The result of Decision Tree Model | 26 |
| 11 | Figure 11: ROC Curve | 28 |
| 12 | Figure 12: Plot tree | 29 |
| 13 | Table 13: The result of Random Forest Model | 32 |
| 14 | Figure 14: ROC Curve | 34 |
| 15 | Table 15: The result of SVM Model | 37 |
| 16 | Figure 16: ROC Curve | 39 |
| 17 | Table 17: The result of Naive Bayes Model | 42 |
| 18 | Figure 18: ROC Curve | 44 |
| 19 | Table 19: The result of KNN Model | 47 |
| 20 | Figure 20: ROC Curve | 49 |
| 21 | Table 21: The result of upsampling | 50 |
| 22 | Table 22: The result of downsampling | 51 |
| 23 | Table 23: Predicting new customer | 52 |

**INTRODUCTION**

1. **Requirement**

Building the model to predict the repayment ability of individual customers based on the database created by individual learners. Because of forecasting the ability to repay on time and understanding the factors that affect the ability to repay of individual customers is extremely important and necessary. This helps commercial banks enhance their ability to identify and repay customers on time, contributing to reducing bad debts and enhancing credit risk management.

1. **Model**

Decision tree is a key tool for reasons: The results provided by the decision tree are very clear, easy to understand and analyze giving the principles of fast decision making (rejecting/approving a loan) and another rule for more complex cases helps to make quick decisions while ensuring high accuracy.

Logistic Regression, Random forest, SVM, Naive Bayes, and KNN were selected as the matching tools.

1. **Data Description**

**Target variable: Ability to repay**. It represents the ability of the customer to properly fulfill the obligation to repay the lender in full when it is due. If the customer is forecasted to have good repayment capacity, the lender will agree to lend and otherwise will refuse to lend. With 1 being overdue repayment and 0 being on-time repayment.

**Feature 1: Gender.** It indicates the gender of the customer. 1 is female and 0 is male..

**Feature 2: Loan purpose.** It shows what customers need to borrow money for. The Data set includes 5 main purposes are 1. consumption, 2. buying a house, 3. buying a car, 4. learning, and 5. investment.

**Feature 3: Marital status.** It can show the customer's current marital status including 1. single, 2. divorced, and 3. married.

**Feature 4: Electronic bill.** It shows the customer's ability to pay. The average monthly electricity bill of a customer is 516 563 VND.

**Feature 5: Loan amount.** It can indicate how much the customer needs to borrow. The average loan amount is 271 927 800 VND.

**Feature 6: Job Tenure.** It shows the number of years the customer has worked at the current company. The average year is 7 years.

**Feature 7: Age**. It shows the current age of the customer. The average age of customers is around 37 years old.

**Feature 8: Loan Term.** It indicates the length of time that the customer must repay the loan. The average loan term is 29 months.

**Feature 9: Income.** It shows how much the customer earns each month. The average income of customers is 32 748 180 VND.

**Feature 10: Collaterals**. It indicates whether the customer has assets to offer the lender to secure the loan. If the customer fails to repay the loan, the lender may foreclose the collateral to make up for the loan.

**Feature 11: Proof of income.** It shows whether the customer has proof of his financial capacity and the legitimate and consistent source of the customer's income, thereby showing whether or not the customer can repay the loan on time.

**THE CONTENT**

1. **Implementation process**

Step 1: Importing data

Step 2: Checking data

Step 3: Data processing

Step 4: Descriptive Statistics

Step 5: Correlation

Step 6: Using Logistic Regression to predict

Step 7: Using Decision Tree to predict

Step 8: Using Random Forest to predict

Step 9: Using SVM to predict

Step 10: Using Naive Bayes to predict

Step 11: Using KNN to predict

Step 12: Improving model quality with upsampling/downsampling

Step 13: Predicting new customer

1. **Step** **1: Importing data**

| data=pd.read\_excel("C:\\Users\\ASUS\\Downloads\\Lương Thị Mỹ Tâm -K204141929.xlsx")  data |
| --- |

| |  | **ID** | **Ability to repay (0: on-time, 1: overdue)** | **Gender** | **Loan Purpose** | **Marital Status** | **Electricity Bill** | **Loan Amount** | **Job Tenure (year)** | **Age** | **Loan Term (month)** | **Income** | **Collaterals** | **Proof of income** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **0** | 1 | 0 | 1. female | 1. Consumption | 3. Married | 319883 | 48000000 | 11 | 43 | 48 | 40000000 | yes | yes | | **1** | 2 | 0 | 0. male | 4. Learning | 1. Single | 150000 | 60000000 | 5 | 24 | 36 | 10000000 | yes | yes | | **2** | 3 | 1 | 1. female | 1. Consumption | 1. Single | 120000 | 32000000 | 7 | 23 | 12 | 22000000 | no | yes | | **3** | 4 | 0 | 0. male | 1. Consumption | 2. Divorced | 221520 | 27000000 | 9 | 57 | 36 | 22000000 | no | no | | **4** | 5 | 0 | 1. female | 1. Consumption | 3. Married | 320994 | 100000000 | 6 | 57 | 6 | 22000000 | no | no |   1849 rows × 13 columns |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 1: The Data set |

1. **Step 2: Checking data**

| data.info() |
| --- |

| <class 'pandas.core.frame.DataFrame'>  RangeIndex: 1849 entries, 0 to 1848  Data columns (total 13 columns):  # Column Non-Null Count Dtype --- ------ -------------- -----  0 ID 1849 non-null int64  1 Ability to repay (0: on-time, 1: overdue) 1849 non-null int64  2 Gender 1849 non-null object  3 Loan Purpose 1849 non-null object  4 Marital Status 1849 non-null object  5 Electricity Bill 1849 non-null int64  6 Loan Amount 1849 non-null int64  7 Job Tenure (year) 1849 non-null int64  8 Age 1849 non-null int64  9 Loan Term (month) 1849 non-null int64  10 Income 1849 non-null int64  11 Collaterals 1849 non-null object  12 Proof of income 1849 non-null object  dtypes: int64(8), object(5)  memory usage: 187.9+ KB |
| --- |

Table 2: The information of the data set

**Comment:**

As we can see the table above, the data set includes 13 columns and 1849 rows, of which 5 columns have the data type of object and 8 columns have the data type of number. And the data set contains no missing values.

1. **Step 3: Data Processing**

| #remove ID column  data1 = data1.iloc[:,1:]  data1  #remove of unnecessary information  for i in data1.columns[1:4]:  data1[i] = data1[i].apply(lambda x: x.split(':')[0])  data1[i] = data1[i].apply(lambda x: x.split('.')[0])  data1[i] = data1[i].apply(lambda x: int(x))  #transform yes, no into 1 and 0  data1["Collaterals"] = data1["Collaterals"].map({"no": 0, "yes": 1})  data1["Proof of income"] = data1["Proof of income"].map({"no": 0, "yes": 1})  #remove customer under 18 and above 60  data1 = data1[(data1["Age"] >= 18) & (data1["Age"] <= 60)]  data1 |
| --- |

| |  | **Ability to repay (0: on-time, 1: overdue)** | **Gender** | **Loan Purpose** | **Marital Status** | **Electricity Bill** | **Loan Amount** | **Job Tenure (year)** | **Age** | **Loan Term (month)** | **Income** | **Collaterals** | **Proof of income** | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **0** | 0 | 1 | 1 | 3 | 319883 | 48000000 | 11 | 43 | 48 | 40000000 | 1 | 1 | | **1** | 0 | 0 | 4 | 1 | 150000 | 60000000 | 5 | 24 | 36 | 10000000 | 1 | 1 | | **2** | 1 | 1 | 1 | 1 | 120000 | 32000000 | 7 | 23 | 12 | 22000000 | 0 | 1 | | **3** | 0 | 0 | 1 | 2 | 221520 | 27000000 | 9 | 57 | 36 | 22000000 | 0 | 0 | | **4** | 0 | 1 | 1 | 3 | 320994 | 100000000 | 6 | 57 | 6 | 22000000 | 0 | 0 |   1841 rows × 12 columns |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Table 3: The data set after processing

**Comment:**

In this data processing step, we will first remove the ID column so that the data set is left with only 12 columns. Since the model only works with numeric data, we need to convert the Gender column, the Loan Purpose column, and the Marital Status column to numerical data by removing the right letter content and keeping only the number on the left. In addition, we will change 'Yes' to 1 and 'No' to 0 for the collaterals column and proof of income column. Finally, we will filter out and remove customers whose age is not suitable for lending, including customers under 18 years old and over 60 years old.

=> As a result, the data set will be left with 12 columns and 1841 rows.

1. **Step 4: Descriptive Statistics**
2. **Numerical columns**

| #iloc[row, column]  num = data2.iloc[:,4:10]  cat.describe()  #Visualize  numerical\_col = ['Electricity Bill','Loan Amount','Job Tenure (year)','Age','Loan Term (month)','Income']  fig, ax = plt.subplots(6, 2, \*\*{"figsize": (20, 20)})  type\_graph = ['histplot', 'boxplot']  for i, graph in enumerate(type\_graph):  for j, numerical in enumerate(numerical\_col):  if graph == 'histplot':  sns.histplot(data=data1, x=numerical, hue='Ability to repay (0: on-time, 1: overdue)', ax=ax[j][i], kde=True)  else:  sns.boxplot(data=data1, x='Ability to repay (0: on-time, 1: overdue)', y=numerical, ax=ax[j][i]) |
| --- |

| |  | **Electricity Bill** | **Loan Amount** | **Job Tenure (year)** | **Age** | **Loan Term (month)** | **Income** | | --- | --- | --- | --- | --- | --- | --- | | **count** | 1.84E+03 | 1.84E+03 | 1841 | 1841 | 1841 | 1.84E+03 | | **mean** | 5.17E+05 | 2.72E+08 | 6.754481 | 37.513308 | 29.69799 | 3.27E+07 | | **std** | 3.18E+05 | 3.98E+08 | 2.73931 | 11.672755 | 17.592552 | 2.11E+07 | | **min** | 1.00E+05 | 2.00E+07 | 1 | 18 | 6 | 1.50E+06 | | **25%** | 3.63E+05 | 5.00E+07 | 5 | 27 | 12 | 2.00E+07 | | **50%** | 4.80E+05 | 9.00E+07 | 7 | 35 | 24 | 3.00E+07 | | **75%** | 5.96E+05 | 3.00E+08 | 8 | 47 | 48 | 4.00E+07 | | **max** | 3.20E+06 | 2.00E+09 | 15 | 60 | 72 | 2.20E+08 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Table 4: Descriptive Statistics of numerical columns

**Comment:**

* Electricity bill column: The highest electricity bill is 3.2 million VND, and the lowest is 100 000 VND. On average, a customer will have to pay 516 563 VND for electricity bill per month. Customers with higher electricity bills are more likely to pay their debts on time.
* Loan amount column: The highest loan amount is 2 billion VND, and the lowest is 20 million VND. The average loan amount is 271 927 800 VND. Customers with a larger Loan Amount are more likely to repay their loans late.
* Job tenure column: The customer is working for the current company for the longest 15 years and the lowest is 1 years. The average duration is 7 years of employment. The larger the customer with Job Tenure, the more likely they are to repay the loan on time.
* Age column: The age of customers ranges from 18 to 60 years old, the average age is 37 years old. The older customers are, the more likely they are to pay their debts on time.
* Loan term column: The longest loan customer is 72 months, the shortest is 6 months and the average is about 29 months.The longer the loan term, the more likely the customer will be able to repay the loan on time
* Income column: The highest and lowest income are 220 million VND and 1 million 500 VND respectively, on average each customer will have an income of about 32 million 748 thousand VND. Customers with higher incomes are more likely to repay their loans on time.

1. **Categorical columns**

| #iloc[row, column]  cat = data2.drop(data2.columns[[4,5,6,7,8,9]], axis=1)  cat.describe()  d = cat.groupby(['Ability to repay (0: on-time, 1: overdue)'])  d.describe().T |
| --- |

| |  | **Ability to repay (0: on-time, 1: overdue)** | **Gender** | **Loan Purpose** | **Marital Status** | **Collaterals** | **Proof of income** | | --- | --- | --- | --- | --- | --- | --- | | **count** | 1841 | 1841 | 1841 | 1841 | 1841 | 1841 | | **unique** | 2 | 2 | 5 | 3 | 2 | 2 | | **top** | 0 | 0 | 1 | 2 | 0 | 1 | | **freq** | 1002 | 937 | 708 | 742 | 924 | 1091 |  |  | **Ability to repay (0: on-time, 1: overdue)** | **0** | **1** | | --- | --- | --- | --- | | **Gender** | **count** | 1002 | 839 | | **unique** | 2 | 2 | | **top** | 0 | 0 | | **freq** | 504 | 433 | | **Loan Purpose** | **count** | 1002 | 839 | | **unique** | 5 | 5 | | **top** | 5 | 1 | | **freq** | 304 | 426 | | **Marital Status** | **count** | 1002 | 839 | | **unique** | 3 | 3 | | **top** | 3 | 2 | | **freq** | 501 | 380 | | **Collaterals** | **count** | 1002 | 839 | | **unique** | 2 | 2 | | **top** | 1 | 0 | | **freq** | 712 | 634 | | **Proof of income** | **count** | 1002 | 839 | | **unique** | 2 | 2 | | **top** | 1 | 0 | | **freq** | 832 | 580 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Table 5: Descriptive Statistics of categorical columns

**Comment:**

* Ability to repay column: includes 2 values 0 and 1, the value of 0 is paying the debt on time with the most number of 1002 and late payment is 839.
* Gender column: includes 2 values of 0 and 1. The male gender accounts for the largest number with 937 male customers. And the number of male customers paying on time and the number of male customers paying late are also higher than female customers.
* Loan purpose column: includes 5 values from 1 to 5. Customers who borrow money for consumption purpose are the most with 708 customers. Borrowers for investment purpose have the highest ability to repay on time with 1002 customers and customers who borrow for consumption are the most likely to pay late with 426 customers.
* Marital Status column: includes 3 values from 1 to 3. The number of divorced customers coming to borrow is the largest with 742 customers. Married customers will be most likely to repay on time with 501 people and divorced customers will be most likely to be late with 380 people.
* Collaterals column: includes 2 values 1 and 0 respectively yes and no. Borrowers without collateral are the most with 924 people. Customers with collateral will have a better repayment capacity than customers without collateral. And customers who do not have collateral will pay their debts overdue.
* Proof of income column: includes 2 values 1 and 0 respectively yes and no. The majority of borrowers are those with proof of full income with 1091 people. Customers with proof of income are 832 more likely to repay their loans on time than customers without proof of income.

1. **Step 5: Correlation**

| pd.DataFrame(data1.corr().iloc[:,0])   |  | **Ability to repay (0: on-time, 1: overdue)** | | --- | --- | | **Ability to repay (0: on-time, 1: overdue)** | 1 | | **Gender** | -0.013047 | | **Loan Purpose** | -0.237894 | | **Marital Status** | -0.292156 | | **Electricity Bill** | -0.003737 | | **Loan Amount** | 0.109487 | | **Job Tenure (year)** | -0.313829 | | **Age** | -0.160721 | | **Loan Term (month)** | -0.355304 | | **Income** | -0.182608 | | **Collaterals** | -0.464413 | | **Proof of income** | -0.528739 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Table 6: Correlation

**Comment:**

Based on the correlation table of the independent variables with the target variable, we see the following relationships:

* Loan Amount column show a positive correlation but it is weak and insignificant.
* The Gender column, The electricity bill column, Age column, and Income column have a negative correlation but the coefficient is quite low. It shows us that female customers have the electricity bill is higher, the higher the age and income, the more likely to pay the debt on time.
* The loan purpose column, Marital status column, and Job tenure column have an negative relationship. It shows that customers who are married, have worked for a long time at the current company, and borrowed for investment or study purposes will have better repayment capacity and otherwise.
* The Loan Term column, Collaterals column, and Proof of Income column are negatively related to the target variable and have a quite strong correlation. It shows that the longer the loan period and have collateral and proof of income, the more likely it is to repay the loan on time.

=> From here, we can see that there are many variables with weak correlation coefficients, which do not have much impact on the target variable so we will filter out the 5 most important variables to run the model to improve the quality of the model results better. The model will run variables including 5 independent variables Marital status column, Collaterals column, Proof of income column, Loan Term column, Loan purpose column, and 1 target variable is the ability to repay column.

1. **Step 6: Using Logistic Regression to predict**
2. **Theory:**

Logistic regression is a classification technique that helps to predict the probability of an outcome that can only have two values. It is a statistical technique that estimates the likelihood of an event occurring based on a given set of independent variables. The dependent variable, which represents the outcome, is a probability and is therefore bounded between 0 and 1. The model uses a logit transformation to convert the odds, which is the ratio of the probability of success to the probability of failure. Logistic regression has a wide range of applications including Fraud detection, Disease prediction,…

* Strengths:
* It is a simple and efficient algorithm that is easy to implement and interpret.
* It does not make any assumptions about the distribution of classes in the feature space.
* It can be easily extended to handle multiple classes through multinomial regression.
* It provides a measure of how appropriate a predictor (coefficient size) is, but also its direction of association (positive or negative) .
* It is a fast algorithm for classifying unknown records.
* Weaknesses:
* It should not be used when the number of observations is less than the number of features, as this can lead to overfitting.
* The algorithm assumes linearity between the dependent and independent variables.
* It can only be used to predict discrete functions, meaning that the dependent variable must be a discrete set.
* Logistic regression cannot solve non-linear problems because it has a linear decision surface, and linearly separable data is rare in real-world scenarios.

1. **Prediction:**

| #Train set  LR\_classifier = LogisticRegression()  LR\_classifier.fit(X\_train, y\_train.ravel())  y\_pred = LR\_classifier.predict(X\_train)  print(confusion\_matrix(y\_train,y\_pred))  print(classification\_report(y\_train,y\_pred))  print('Logistic Regression accuracy: ', accuracy\_score(y\_train, y\_pred))  #Test set  LR\_classifier = LogisticRegression()  LR\_classifier.fit(X\_train, y\_train.ravel())  y\_pred = LR\_classifier.predict(X\_test)  print(confusion\_matrix(y\_test,y\_pred))  print(classification\_report(y\_test,y\_pred))  print('Logistic Regression accuracy: ', accuracy\_score(y\_test, y\_pred)) |
| --- |

| **Train set**  [[625 168]  [155 524]]  precision recall f1-score support  0 0.80 0.79 0.79 793  1 0.76 0.77 0.76 679  accuracy 0.78 1472  macro avg 0.78 0.78 0.78 1472  weighted avg 0.78 0.78 0.78 1472  Logistic Regression accuracy: 0.7805706521739131  **Test set**  [[158 51]  [ 33 127]]  precision recall f1-score support  0 0.83 0.76 0.79 209  1 0.71 0.79 0.75 160  accuracy 0.77 369  macro avg 0.77 0.77 0.77 369  weighted avg 0.78 0.77 0.77 369  Logistic Regression accuracy: 0.7723577235772358 |
| --- |

Table 7: The result of Logistic Regression Model

**Comment:**

- **Logistic Regression model result for the training set:**

There are 793 customers who are predicted to pay their debts on time, but in fact, there are 168 customers who are late paying their debts. And there are 679 customers who are predicted to be overdue, but in fact, 155 of them are on time. For class 0, the precision is 0.80, which means that 80% of the instances predicted as class 0 by the model are actually class 0. The recall is 0.79, which means that 79% of the actual class 0 instances were correctly predicted by the model. The f1-score is 0.79, which balances both precision and recall. For class 1, the precision is 0.76, which means that 76% of the instances predicted as class 1 by the model are actually class 1. The recall is 0.77, which means that 77% of the actual class 1 instances were correctly predicted by the model. The f1-score is 0.76. These metrics suggest that the model is performing reasonably well in predicting both classes, with slightly better performance in predicting class 0. Accuracy of the model is 0.7806 which is also quite high.

=> We can use this model to predict the test set.

- **Logistic Regression model result for the test set:**

There are 209 customers who are predicted to pay their debts on time, but in fact, there are 51 customers who are late paying their debts. And there are 160 customers who are predicted to be overdue, but in fact, 33 of them are on time. For class 0, the precision is 0.83, which means that 83% of the instances predicted as class 0 by the model are actually class 0. The recall is 0.76, which means that 76% of the actual class 0 instances were correctly predicted by the model. The f1-score is 0.79, which balances both precision and recall. For class 1, the precision is 0.71, which means that 71% of the instances predicted as class 1 by the model are actually class 1. The recall is 0.79, which means that 79% of the actual class 1 instances were correctly predicted by the model. These metrics suggest that the model is performing reasonably well in predicting both classes, with slightly better performance in predicting class 0. Accuracy of the model on the test set of 0.7724 and does not deviate too much from the accuracy of the training set.

1. **Regression report table:**

| import statsmodels.api as SM  model = SM.Logit(y\_train, X\_train).fit()  model.summary() |
| --- |

| Optimization terminated successfully.  Current function value: 0.466018  Iterations 6   | Logit Regression Results | | | |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Dep. Variable:** | y | **No. Observations:** | 1472 |  |  |  | | **Model:** | Logit | **Df Residuals:** | 1467 |  |  |  | | **Method:** | MLE | **Df Model:** | 4 |  |  |  | | **Date:** | Fri, 05 May 2023 | **Pseudo R-squ.:** | 0.3248 |  |  |  | | **Time:** | 18:32:35 | **Log-Likelihood:** | -685.98 |  |  |  | | **converged:** | TRUE | **LL-Null:** | -1015.9 |  |  |  | | **Covariance Type:** | nonrobust | **LLR p-value:** | 1.74E-141 |  |  |  | |  |  |  |  |  |  |  | |  | **coef** | **std err** | **z** | **P>|z|** | **[0.025** | **0.975]** | | **x1** | -0.2855 | 0.08 | -3.555 | 0 | -0.443 | -0.128 | | **x2** | -0.8345 | 0.073 | -11.39 | 0 | -0.978 | -0.691 | | **x3** | -0.2282 | 0.078 | -2.914 | 0.004 | -0.382 | -0.075 | | **x4** | -0.4431 | 0.086 | -5.178 | 0 | -0.611 | -0.275 | | **x5** | -0.8785 | 0.078 | -11.265 | 0 | -1.031 | -0.726 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Table 8: Regression report table

**Comment:**

Logistic regression model results show that the value of Pseudo R squared is 0.3248 and according to McFadden, the value of Pseudo R squared must be in the range of 0.2 to 0.4, so the fit of our model is excellent. The independent variables are statistically significant (p\_value < 0.05) and have a correlation with the ability to repay.

1. **ROC Curve:**

| y\_pred\_prob\_test = LR\_classifier.predict\_proba(X\_test)[:, 1]  fpr, tpr, thres = roc\_curve(y\_test, y\_pred\_prob\_test)  roc\_auc = auc(fpr, tpr)  \_plot\_roc\_curve(fpr, tpr, thres, roc\_auc) |
| --- |

|  |
| --- |

Figure 9: ROC Curve

**Comment:**

Looking at the graph, we can see that the ROC curve is quite far from the diagonal and the ROC Curve (area = 0.84) is the model that explains 84% of the data set. With this result, the model has good performance in distinguishing between the two classes.

1. **Step 7: Using Decision Tree to predict**
2. **Theory:**

A decision tree is a type of non-parametric supervised learning that can be used for both classification and regression. It has a tree-like structure with a root node, branches, internal nodes, and leaf nodes arranged in a hierarchical manner. A decision tree works by using a divide-and-conquer strategy to identify the optimal split points within a tree. In a decision tree, the data is split into subsets in a top-down, recursive manner until all or most of the records have been assigned to specific class labels. The tree starts with a root node that has no incoming branches. The branches extending from the root node lead to internal nodes, also called decision nodes. These nodes evaluate the available features to form homogeneous subsets, which are represented by leaf nodes or terminal nodes. The leaf nodes represent all possible outcomes within the data set. Decision trees have a wide range of applications including Assessing prospective growth opportunities, Using demographic data to find prospective clients, Data mining, and data classification.

* Strength:
* [Decision trees can handle both continuous and categorical variables.](https://www.geeksforgeeks.org/decision-tree/)
* [They provide a clear indication of which features are most important for prediction or classification.](https://www.geeksforgeeks.org/decision-tree/)
* [Decision trees can be easily understood, interpreted, and visualized.](https://www.educba.com/decision-tree-advantages-and-disadvantages/)
* [They require less data preparation compared to other machine learning algorithms.](https://www.educba.com/decision-tree-advantages-and-disadvantages/)
* [Decision trees are non-parametric, which means that they do not make any assumptions about the distribution of the data or the structure of the classifier.](https://www.educba.com/decision-tree-advantages-and-disadvantages/)
* Weakness:
* [Decision trees can be prone to overfitting, especially when the tree is too complex.](https://www.educba.com/decision-tree-advantages-and-disadvantages/)
* [They may have difficulties handling numerical variables with millions of records due to the time complexity of operating this operation.](https://www.educba.com/decision-tree-advantages-and-disadvantages/)
* [Decision trees may not perform as well on regression tasks compared to classification tasks.](https://www.geeksforgeeks.org/decision-tree/)

**b) Prediction:**

| #Train set  DT\_classifier = DecisionTreeClassifier()  DT\_classifier.fit(X\_train, y\_train.ravel())  y\_pred = DT\_classifier.predict(X\_train)  print(confusion\_matrix(y\_train,y\_pred))  print(classification\_report(y\_train,y\_pred))  print('Decision Tree accuracy: ', accuracy\_score(y\_train, y\_pred))  #Test set  DT\_classifier = DecisionTreeClassifier()  DT\_classifier.fit(X\_train, y\_train.ravel())  y\_pred = DT\_classifier.predict(X\_test)  print(confusion\_matrix(y\_test,y\_pred))  print(classification\_report(y\_test,y\_pred))  print('Decision Tree accuracy: ', accuracy\_score(y\_test, y\_pred)) |
| --- |

| **Train set**  [[707 86]  [ 39 640]]  precision recall f1-score support  0 0.95 0.89 0.92 793  1 0.88 0.94 0.91 679  accuracy 0.92 1472  macro avg 0.91 0.92 0.91 1472  weighted avg 0.92 0.92 0.92 1472  Decision Tree accuracy: 0.9150815217391305  **Test set**  [[179 30]  [ 6 154]]  precision recall f1-score support  0 0.97 0.86 0.91 209  1 0.84 0.96 0.90 160  accuracy 0.90 369  macro avg 0.90 0.91 0.90 369  weighted avg 0.91 0.90 0.90 369  Decision Tree accuracy: 0.9024390243902439 |
| --- |

Table 10: The result of Decision Tree Model

**Comment:**

- **Decision Tree model result for the training set:**

There are 793 customers who are predicted to pay their debts on time, but in fact, there are 86 customers who are late paying their debts. And there are 679 customers who are predicted to be overdue, but in fact, 39 of them are on time. For class 0, the precision is 0.95, which means that 95% of the instances predicted as class 0 by the model are actually class 0. The recall is 0.89, which means that 89% of the actual class 0 instances were correctly predicted by the model. The f1-score is 0.92, which balances both precision and recall. For class 1, the precision is 0.88, which means that 88% of the instances predicted as class 1 by the model are actually class 1. The recall is 0.94, which means that 94% of the actual class 1 instances were correctly predicted by the model. The f1-score is 0.91 Accuracy of the model is 0.9151 which is very high.

=> We can use this model to predict the test set.

- **Decision Tree model result for the test set:**

There are 209 customers who are predicted to pay their debts on time, but in fact, 30 of them are late payments. And there are 160 customers who are predicted to be overdue, but in fact, 6 of them are on time. For class 0, the precision is 0.97, which means that 97% of the instances predicted as class 0 by the model are actually class 0. The recall is 0.86, which means that 86% of the actual class 0 instances were correctly predicted by the model. The f1-score is 0.91, which balances both precision and recall. For class 1, the precision is 0.84, which means that 84% of the instances predicted as class 1 by the model are actually class 1. The recall is 0.96, which means that 96% of the actual class 1 instances were correctly predicted by the model. The f1-score is 0.90. Accuracy of the model on the test set is 0.9024, not too much deviating from the accuracy of the training set and higher than the accuracy of the Logistic Regression model.

**c) ROC Curve:**

| y\_pred\_prob\_test = DT\_classifier.predict\_proba(X\_test)[:, 1]  fpr, tpr, thres = roc\_curve(y\_test, y\_pred\_prob\_test)  roc\_auc = auc(fpr, tpr)  \_plot\_roc\_curve(fpr, tpr, thres, roc\_auc) |
| --- |

|  |
| --- |

Figure 11: ROC Curve

**Comment:**

Based on the chart above, we can see that the ROC curve is very far from the diagonal and the ROC Curve (area = 0.97) is the model that explains 97% of the data set. This indicates that the model is able to effectively separate positive and negative instances and has a relatively low rate of false classifications. With this result, the Decision Tree model has better performance than the Logistic Regression model.

**d)Plot Tree:**

| X\_plot\_tree = data1[features]  clf = DecisionTreeClassifier(criterion =’gini’, max\_depth=5)  clf.fit(X\_plot\_tree.values, y.ravel())  print('Note: Class 0 is on-time repayment and class 1 is overdue repayment ')  fig = plt.figure(figsize=(50,20))  \_ = tree.plot\_tree(clf,  feature\_names=features,  class\_names=['On-time','Overdue'],  filled=True) |
| --- |

|  |
| --- |

Figure 12: Plot tree

**Comment:**

From the decision tree image, we can build the lending principles for individual customers to credit officers as follows:

* Customers with proof of income, married and Loan term from over 21 months to 54 months will be accepted for loans.
* Customers who meet all 3 criteria are single, do not have proof of income and collateral, will not be accepted for loans.
* Customers who do not have clear proof of income but have collateral, whose purpose for the loan is for consumption, buying a house or buying a car, with a Loan Term greater than 9 months and are married will tend to pay more late. However, the difference is not significant compared to the number of customers who pay on time, just more than 1 customer. This case can cause confusion and may not allow for an immediate loan decision. Other factors may need to be considered to make a better decision.

1. **Step 8: Using Random Forest to predict**
2. **Theory:**

Random forest is a type of ensemble learning that builds multiple decision trees during training. For classification tasks, the algorithm outputs the class that is the mode of the classes predicted by the individual trees. For regression tasks, it outputs the mean prediction of the individual trees. Rather than relying on just one decision tree, the algorithm uses the majority vote from the predictions of all the trees in the forest to determine the final output. Having a larger number of trees in the forest can increase accuracy and help prevent overfitting. The theory behind Random Forest involves using a divide-and-conquer strategy to identify the optimal split points within a tree. This is done through a greedy search algorithm. The process of splitting the data into subsets is repeated recursively in a top-down manner. This continues until all or most of the records have been assigned to specific class labels. Random Forests has many applications in many areas including In Finance, It can be used to assess high credit-risk customers. In Healthcare, Random Forest can be used to predict the likelihood of disease or illness for a certain population. In E-Commerce, Random Forest can be used for product demos, pricing optimization, and ranking search.

* Strength:
* [Random forests are able to handle both continuous and categorical variables.](https://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-random.html)
* [They clearly indicate the most important fields for prediction or classification.](https://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-random.html)
* [Random forests are easy to understand, interpret, and visualize.](https://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-random.html)
* [They require less data preparation compared to other machine learning algorithms.](https://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-random.html)
* [Random forests are non-parametric, meaning they make no assumptions about the spatial distribution and the classifier structure.](https://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-random.html)
* Weakness:
* [Random forests can be prone to overfitting, especially when the tree is too complex.](https://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-random.html)
* [They may have difficulties handling numerical variables with millions of records due to the time complexity of operating this operation.](https://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-random.html)

1. **Prediction:**

| #Train set  RF\_classifier = RandomForestClassifier()  RF\_classifier.fit(X\_train, y\_train.ravel())  y\_pred = RF\_classifier.predict(X\_train)  print(confusion\_matrix(y\_train,y\_pred))  print(classification\_report(y\_train,y\_pred))  print('Random Forest accuracy: ', accuracy\_score(y\_train, y\_pred))  #Test set  RF\_classifier = RandomForestClassifier()  RF\_classifier.fit(X\_train, y\_train.ravel())  y\_pred = RF\_classifier.predict(X\_test)  print(confusion\_matrix(y\_test,y\_pred))  print(classification\_report(y\_test,y\_pred))  print('Random Forest accuracy: ', accuracy\_score(y\_test, y\_pred)) |
| --- |

| **Train set**  [[690 103]  [ 22 657]]  precision recall f1-score support  0 0.97 0.87 0.92 793  1 0.86 0.97 0.91 679  accuracy 0.92 1472  macro avg 0.92 0.92 0.92 1472  weighted avg 0.92 0.92 0.92 1472  Random Forest accuracy: 0.9150815217391305  **Test set**  [[179 30]  [ 6 154]]  precision recall f1-score support  0 0.97 0.86 0.91 209  1 0.84 0.96 0.90 160  accuracy 0.90 369  macro avg 0.90 0.91 0.90 369  weighted avg 0.91 0.90 0.90 369  Random Forest accuracy: 0.9024390243902439 |
| --- |

Table 13: The result of Random Forest Model

**Comment:**

- **Random Forest model result for the training set:**

* There are 793 customers who are predicted to pay their debts on time, but in fact 97 of them are late payments. And there are 679 customers who are predicted to be overdue, but in fact, 28 of them are on time. for class 0, the precision is 0.96, which means that 96% of the instances predicted as class 0 by the model are actually class 0. The recall is 0.88, which means that 88% of the actual class 0 instances were correctly predicted by the model. The f1-score is 0.92, which balances both precision and recall. For class 1, the precision is 0.87, which means that 87% of the instances predicted as class 1 by the model are actually class 1. The recall is 0.96, which means that 96% of the actual class 1 instances were correctly predicted by the model. The f1-score is 0.91. Accuracy of the model is 0.9151 which is equal to the accuracy of Decision Tree.

=> We can use this model to predict the test set.

- **Random Forest model result for the test set:**

There are 209 customers who are predicted to pay their debts on time, but in fact, 30 of them are late payments. And there are 160 customers who are predicted to be overdue, but in fact, 6 of them are on time. For class 0, the precision is 0.97, which means that 97% of the instances predicted as class 0 by the model are actually class 0. The recall is 0.86, which means that 86% of the actual class 0 instances were correctly predicted by the model. The f1-score is 0.91, which balances both precision and recall. For class 1, the precision is 0.84, which means that 84% of the instances predicted as class 1 by the model are actually class 1. The recall is 0.96, which means that 96% of the actual class 1 instances were correctly predicted by the model. The f1-score is 0.90. Accuracy of the model on the test set is 0.9024, not too much deviating from the accuracy of the training set and higher than the accuracy of the Logistic Regression model but equal to the accuracy of the Decision Tree model.

**c) ROC Curve:**

| y\_pred\_prob\_test = RF\_classifier.predict\_proba(X\_test)[:, 1]  fpr, tpr, thres = roc\_curve(y\_test, y\_pred\_prob\_test)  roc\_auc = auc(fpr, tpr)  \_plot\_roc\_curve(fpr, tpr, thres, roc\_auc) |
| --- |

|  |
| --- |

Figure 14: ROC Curve

**Comment:**

Based on the chart above, we can see that the ROC curve is very far from the diagonal and the ROC curve (area = 0.97) is the model that explains 97% of the data set. With this result, the Random Forest model has a better performance than the Logistic Regression model and is on par with the Decision Tree model.

1. **Step 9: Using SVM to predict**
2. **Theory:**

Support Vector Machine (SVM) is a supervised machine learning algorithm that can be applied to both classification and regression problems. It operates by identifying the optimal hyperplane that separates data points in n-dimensional space into distinct classes. SVMs are effective in high-dimensional spaces and remain effective even when the number of dimensions exceeds the number of samples. They use a subset of training points, called support vectors, in the decision function, making them memory efficient. They are also versatile, as different kernel functions can be used in the decision function. Some of the fields where SVMs are used the most include Image-based analysis and classification tasks (such as facial feature extraction and recognition), and Geospatial data-based applications (such as inversion problem), Text-based applications (such as email spam classification), Computational biology, and SVMs can also be used for regression and outlier detection. The applications of SVMs are not limited to these categories, as it is a versatile algorithm with many uses.

* Strength:
* SVMs are effective in high-dimensional spaces and remain effective even when the number of dimensions exceeds the number of samples.
* [SVMs use a subset of training points, called support vectors, in the decision function, making them memory efficient](https://scikit-learn.org/stable/modules/svm.html).
* [SVMs are also versatile, as different kernel functions can be used in the decision function](https://scikit-learn.org/stable/modules/svm.html).
* Weakness:
* When the number of features significantly exceeds the number of samples, it is important to avoid over-fitting by carefully selecting kernel functions and the regularization term.
* SVMs do not directly provide probability estimates; these must be calculated using an expensive five-fold cross-validation process.

**b) Prediction:**

| #Train set  svc\_classifier = SVC(kernel = 'linear', random\_state = 42)  svc\_classifier.fit(X\_train, y\_train.ravel())  y\_pred = svc\_classifier.predict(X\_train)  print(confusion\_matrix(y\_train,y\_pred))  print(classification\_report(y\_train,y\_pred))  print('SVC accuracy: ', accuracy\_score(y\_train, y\_pred))  #Test set  svc\_classifier = SVC(kernel = 'linear', random\_state = 42, probability=True)  svc\_classifier.fit(X\_train, y\_train.ravel())  y\_pred = svc\_classifier.predict(X\_test)  print(confusion\_matrix(y\_test,y\_pred))  print(classification\_report(y\_test,y\_pred))  print('SVC accuracy: ', accuracy\_score(y\_test, y\_pred)) |
| --- |

| **Train set**  [[660 133]  [222 457]]  precision recall f1-score support  0 0.75 0.83 0.79 793  1 0.77 0.67 0.72 679  accuracy 0.76 1472  macro avg 0.76 0.75 0.75 1472  weighted avg 0.76 0.76 0.76 1472  SVC accuracy: 0.7588315217391305  **Test set**  [[172 37]  [ 37 123]]  precision recall f1-score support  0 0.82 0.82 0.82 209  1 0.77 0.77 0.77 160  accuracy 0.80 369  macro avg 0.80 0.80 0.80 369  weighted avg 0.80 0.80 0.80 369  SVC accuracy: 0.7994579945799458 |
| --- |

Table 15: The result of SVM Model

**Comment:**

- **SVM model result for the training set:**

* There are 793 customers who are predicted to pay their debts on time, but in fact 133 of them are late payments. And there are 679 customers who are predicted to be overdue, but in fact, 222 of them are on time. for class 0, the precision is 0.75, which means that 75% of the instances predicted as class 0 by the model are actually class 0. The recall is 0.83, which means that 83% of the actual class 0 instances were correctly predicted by the model. The f1-score is 0.79, which balances both precision and recall. For class 1, the precision is 0.77, which means that 77% of the instances predicted as class 1 by the model are actually class 1. The recall is 0.67, which means that 67% of the actual class 1 instances were correctly predicted by the model. The f1-score is 0.72. Accuracy of the model is 0.7588.

=> We can use this model to predict the test set.

- **SVM model result for the test set:**

* There are 209 customers who are predicted to pay their debts on time, but in fact, 37 of them are late payments. And there are 160 customers who are predicted to be overdue, but in fact, 37 of them are on time. For class 0, the precision is 0.82, which means that 82% of the instances predicted as class 0 by the model are actually class 0. The recall is 0.82, which means that 82% of the actual class 0 instances were correctly predicted by the model. The f1-score is 0.82, which balances both precision and recall. For class 1, the precision is 0.77, which means that 77% of the instances predicted as class 1 by the model are actually class 1. The recall is 0.77, which means that 77% of the actual class 1 instances were correctly predicted by the model. The f1-score is 0.77. Accuracy of the model on the test set of 0.7994 is higher than the accuracy of the training set and is higher than the accuracy of the Logistic Regression model but lower than the accuracy of the Decision Tree model and Random Forest model.

**c) ROC Curve:**

| y\_pred\_prob\_test = svc\_classifier.predict\_proba(X\_test)[:, 1]  fpr, tpr, thres = roc\_curve(y\_test, y\_pred\_prob\_test)  roc\_auc = auc(fpr, tpr)  \_plot\_roc\_curve(fpr, tpr, thres, roc\_auc) |
| --- |

|  |
| --- |

Figure 16: ROC Curve

**Comment:**

Based on the chart above, we can see that the ROC curve is very far from the diagonal and the ROC curve (area = 0.86) is the model that explains 86% of the data set. With this result, the SVM model has a better performance than the Logistic Regression model but this model is worse than the Decision Tree model and the Random Forest model.

1. **Step 10: Using Naive Bayes to predict**
2. **Theory:**

Naive Bayes is a classification algorithm that is based on Bayes’ Theorem. It is a group of algorithms that share the fundamental idea that each pair of features being classified is not dependent on one another. The fundamental assumption of Naive Bayes is that each feature contributes independently and equally to the outcome. It is assumed that no pair of features are dependent and that each feature has equal importance. Bayes’ Theorem is a mathematical formula that calculates the probability of an event based on prior knowledge of conditions that might be related to the event. It allows us to update our beliefs about the likelihood of an event occurring based on new evidence. It is expressed mathematically as P(A|B) = P(B|A) \* P(A) / P(B), where A and B are events and P(B) ≠ 0. The applications of Naive Bayes include Text classification (such as email spam filtering and sentiment analysis), Medical data classification, and Credit scoring. Naive Bayes can also be used for real-time predictions as it is an eager learner.

* Strength:
* It is simple to implement and the conditional probabilities are easy to evaluate.
* It is very fast and can be directly computed, making it useful where the speed of training is important.
* If the assumption of conditional independence between features holds true, it can produce great results.
* Weakness:
* This assumption does not always hold and in many situations, the features may exhibit some form of dependency.
* One issue that can arise is the zero probability problem, where if a word in the test data for a specific class is not present in the training data, it can result in zero class probabilities.
* It may also not perform well on data sets with many continuous features.

1. **Prediction:**

| #Train set  gnb = GaussianNB()  gnb.fit(X\_train, y\_train.ravel())  y\_pred = gnb.predict(X\_train)  print(confusion\_matrix(y\_train,y\_pred))  print(classification\_report(y\_train,y\_pred))  print('Naive Bayes accuracy: ', accuracy\_score(y\_train, y\_pred))  #Test set  gnb = GaussianNB()  gnb.fit(X\_train, y\_train.ravel())  y\_pred = gnb.predict(X\_test)  print(confusion\_matrix(y\_test,y\_pred))  print(classification\_report(y\_test,y\_pred))  print('Naive Bayes accuracy: ', accuracy\_score(y\_test, y\_pred)) |
| --- |

| **Train set**  [[595 198]  [152 527]]  precision recall f1-score support  0 0.80 0.75 0.77 793  1 0.73 0.78 0.75 679  accuracy 0.76 1472  macro avg 0.76 0.76 0.76 1472  weighted avg 0.76 0.76 0.76 1472  Naive Bayes accuracy: 0.7622282608695652  **Test set**  [[147 62]  [ 34 126]]  precision recall f1-score support  0 0.81 0.70 0.75 209  1 0.67 0.79 0.72 160  accuracy 0.74 369  macro avg 0.74 0.75 0.74 369  weighted avg 0.75 0.74 0.74 369  Naive Bayes accuracy: 0.7398373983739838 |
| --- |

Table 17: The result of Naive Bayes Model

**Comment:**

- **Naive Bayes model result for the training set:**

* There are 793 customers who are predicted to pay their debts on time, but in fact 198 of them are late payments. And there are 679 customers who are predicted to be overdue, but in fact, 152 of them are on time. For class 0, the precision is 0.80, which means that 80% of the instances predicted as class 0 by the model are actually class 0. The recall is 0.75, which means that 75% of the actual class 0 instances were correctly predicted by the model. The f1-score is 0.77, which balances both precision and recall. For class 1, the precision is 0.73, which means that 73% of the instances predicted as class 1 by the model are actually class 1. The recall is 0.78, which means that 78% of the actual class 1 instances were correctly predicted by the model. The f1-score is 0.75. Accuracy of the model is 0.7622.

=> We can use this model to predict the test set.

- **Naive Bayes model result for the test set:**

There are 209 customers who are predicted to pay their debts on time, but in fact, 62 of them are late payments. And there are 160 customers who are predicted to be overdue, but in fact, 34 of them are on time. The precision, recall, and f1-score for class 0 are 0.81, 0.70, and 0.75 respectively. This means that when the classifier predicted an instance to be of class 0, it was correct 81% of the time (precision). However, it only correctly identified 70% of the actual class 0 instances (recall). The precision, recall, and f1-score for class 1 are 0.67, 0.79, and 0.72 respectively. This means that when the classifier predicted an instance to be of class 1, it was correct 67% of the time (precision). It correctly identified 79% of the actual class 1 instances (recall). Overall, the classifier seems to have performed slightly better on class 1 instances than on class 0 instances. Accuracy of the model on the test set of 0.7398 is lower than the accuracy of the training set and is the lowest accuracy compared to the four above models.

**c) ROC Curve:**

| y\_pred\_prob\_test = gnb.predict\_proba(X\_test)[:, 1]  fpr, tpr, thres = roc\_curve(y\_test, y\_pred\_prob\_test)  roc\_auc = auc(fpr, tpr)  \_plot\_roc\_curve(fpr, tpr, thres, roc\_auc) |
| --- |

|  |
| --- |

Figure 18: ROC Curve

**Comment:**

Based on the chart above, we can see that the ROC curve is very far from the diagonal and the ROC curve (area = 0.84) is the model that explains 84% of the data set. With this result, the Naive Bayes model has the same performance as the Logistic Regression model but is the lowest performance compared to the four above models .

1. **Step 11: Using KNN to predict**
2. **Theory:**

The k-nearest neighbors algorithm (k-NN) is a type of supervised learning that was first introduced by Evelyn Fix and Joseph Hodges in 1951 and later expanded upon by Thomas Cover. It can be used for both classification and regression problems. The input for the algorithm consists of the k closest training examples from a dataset. The output depends on whether k-NN is being used for classification or regression. For classification, the output is the class membership of an object, determined by a majority vote among its k nearest neighbors. If k equals 1, the object is simply assigned to the class of its single nearest neighbor. In regression problems, the k-NN algorithm outputs the property value of an object by taking the average of the values of its k nearest neighbors. If k equals 1, the output is simply assigned to the value of its single nearest neighbor. The k-NN algorithm has been utilized within a variety of applications, largely within the classification. Some of these use cases include data preprocessing, where the algorithm can estimate missing values in a process known as missing data imputation. The k-NN algorithm is frequently used in a variety of applications such as simple recommendation systems, pattern recognition, data mining, financial market predictions, and intrusion detection, among others.

* Strength:
* It is non-parametric, meaning it does not make any assumptions about the distribution of the underlying data.
* Requires no training time.
* It is versatility, as it can be applied to both classification and regression problems.
* Weakness:
* Can be memory intensive and slow for classification and estimation.
* May not work well with large data sets or high dimensional data.
* Requires feature scaling.
* Sensitive to noise in the data set.

1. **Prediction:**

| #Train set  knn = KNeighborsClassifier(n\_neighbors=3)  knn.fit(X\_train,y\_train.ravel())  y\_pred = knn.predict(X\_train)  print(confusion\_matrix(y\_train,y\_pred))  print(classification\_report(y\_train,y\_pred))  print('KNN accuracy: ', accuracy\_score(y\_train, y\_pred))  #Test set  knn = KNeighborsClassifier(n\_neighbors=3)  knn.fit(X\_train,y\_train.ravel())  y\_pred = knn.predict(X\_test)  print(confusion\_matrix(y\_test,y\_pred))  print(classification\_report(y\_test,y\_pred))  print('KNN accuracy: ', accuracy\_score(y\_test, y\_pred)) |
| --- |

| **Train set**  [[737 56]  [ 85 594]]  precision recall f1-score support  0 0.90 0.93 0.91 793  1 0.91 0.87 0.89 679  accuracy 0.90 1472  macro avg 0.91 0.90 0.90 1472  weighted avg 0.90 0.90 0.90 1472  KNN accuracy: 0.9042119565217391  **Test set**  [[186 23]  [ 17 143]]  precision recall f1-score support  0 0.92 0.89 0.90 209  1 0.86 0.89 0.88 160  accuracy 0.89 369  macro avg 0.89 0.89 0.89 369  weighted avg 0.89 0.89 0.89 369  KNN accuracy: 0.8915989159891599 |
| --- |

Table 19: The result of KNN Model

**Comment:**

- **KNN model result for the training set:**

* There are 793 customers who are predicted to pay their debts on time, but in fact 56 of them are late payments. And there are 679 customers who are predicted to be overdue, but in fact, 85 of them are on time. The precision, recall, and f1-score for class 0 are 0.90, 0.93, and 0.91 respectively. This means that when the classifier predicted an instance to be of class 0, it was correct 90% of the time (precision). It correctly identified 93% of the actual class 0 instances (recall). The precision, recall, and f1-score for class 1 are 0.91, 0.87, and 0.89 respectively. This means that when the classifier predicted an instance to be of class 1, it was correct 91% of the time (precision). It correctly identified 87% of the actual class 1 instances (recall). Overall, the classifier seems to have performed slightly better on class 0 instances than on class 1 instances. Accuracy of the model is 0.9042.

=> We can use this model to predict the test set.

- **KNN model result for the test set:**

There are 209 customers who are predicted to pay their debts on time, but in fact, 23 of them are late payments. And there are 160 customers who are predicted to be overdue, but in fact, 17 of them are on time. The precision, recall, and f1-score for class 0 are 0.92, 0.89, and 0.90 respectively. This means that when the classifier predicted an instance to be of class 0, it was correct 92% of the time (precision). It correctly identified 89% of the actual class 0 instances (recall). The precision, recall, and f1-score for class 1 are 0.86, 0.89, and 0.88 respectively. This means that when the classifier predicted an instance to be of class 1, it was correct 86% of the time (precision). It correctly identified 89% of the actual class 1 instances (recall). Overall, the classifier seems to have performed slightly better on class 0 instances than on class 1 instances. Accuracy of the model on the test set of 0.8915 is not too much deviating from the accuracy of the training set and is higher than the accuracy of the Logistic Regression model, the SVM model, and the Naive Bayes model but is lower than the accuracy of the Decision Tree model and the Random Forest model.

**c) ROC Curve:**

| y\_pred\_prob\_test = knn.predict\_proba(X\_test)[:, 1]  fpr, tpr, thres = roc\_curve(y\_test, y\_pred\_prob\_test)  roc\_auc = auc(fpr, tpr)  \_plot\_roc\_curve(fpr, tpr, thres, roc\_auc) |
| --- |

|  |
| --- |

Figure 20: ROC Curve

**Comment:**

Based on the chart above, we can see that the ROC curve is very far from the diagonal and the ROC curve (area = 0.96) is the model that explains 96% of the data set. With this result, the KNN model has a better performance than the Logistic Regression model, the SVM model, and the Naive Bayes model but is lower than the performance of the Decision Tree model and the Random Forest model.

**The result of initial sample:**

After running the results for 6 models, we see that the quality of all 6 models is good, in which 3 models are significantly superior including Decision Tree, Random Forest, and KNN and the accuracy of all 3 models is quite similar, in which the Decision Tree model has a slightly better result and at the same time this model also provides clear and easy-to-understand lending principles to help credit officers make loan decisions more quickly and efficiently. So, we will choose the Decision Tree model to predict new customer data.

1. **Step 12: Improving model quality with upsample/downsample**
2. **Upsampling**

| #Upsampling  DT\_classifier = DecisionTreeClassifier()  DT\_classifier.fit(X\_over\_train, y\_over\_train.ravel())  y\_over\_pred = DT\_classifier.predict(X\_over\_test)  print(confusion\_matrix(y\_over\_test,y\_over\_pred))  print(classification\_report(y\_over\_test,y\_over\_pred))  print('Decision Tree accuracy: ', accuracy\_score(y\_over\_test, y\_over\_pred)) |
| --- |

| [[170 28]  [ 4 199]]  precision recall f1-score support  0 0.98 0.86 0.91 198  1 0.88 0.98 0.93 203  accuracy 0.92 401  macro avg 0.93 0.92 0.92 401  weighted avg 0.93 0.92 0.92 401  Decision Tree accuracy: 0.9201995012468828 |
| --- |

Table 21: The result of upsampling

1. **Downsampling**

| #Downsampling  DT\_classifier = DecisionTreeClassifier()  DT\_classifier.fit(X\_under\_train, y\_under\_train.ravel())  y\_under\_pred = DT\_classifier.predict(X\_under\_test)  print(confusion\_matrix(y\_under\_test,y\_under\_pred))  print(classification\_report(y\_under\_test,y\_under\_pred))  print('Decision Tree accuracy: ', accuracy\_score(y\_under\_test, y\_under\_pred)) |
| --- |

| [[141 31]  [ 5 159]]  precision recall f1-score support  0 0.97 0.82 0.89 172  1 0.84 0.97 0.90 164  accuracy 0.89 336  macro avg 0.90 0.89 0.89 336  weighted avg 0.90 0.89 0.89 336  Decision Tree accuracy: 0.8928571428571429 |
| --- |

Table 22: The result of downsampling

**The Result of upsampling/downsampling:**

Based on the result table above, we can see that when we downsample the data set, the performance of the Decision Tree model decreases compared to the initial sample. However, when we upsample the data set, the quality of the model is improved with higher accuracy compared to the initial sample. This can be explained by the fact that when we upsample the data set, we increase the number of samples in the minority class to balance it with the majority class. This can help balance the class distribution and improve the accuracy of the model. However, it’s important to note that oversampling can also result in overfitting for some models. When we downsample the data set, we remove samples from the majority class to balance it with the minority class. This can result in losing valuable information that is important for the model to make accurate predictions. Besides that, ROC Curve of oversampling has ROC Curve Area is 0.98 is higher initial sample’s ROC Curve Area.

=> Oversampling will help improve the quality of the Decision Tree model.

1. **Step 13: Predict new customer**

| prediction =DT\_classifier.predict(data2)  print(confusion\_matrix(new\_data['Ability to repay (0: on-time, 1: overdue)'],new\_data['Prediction']))  print(classification\_report(new\_data['Ability to repay (0: on-time, 1: overdue)'],new\_data['Prediction']))  print('Accuracy: ', accuracy\_score(new\_data['Ability to repay (0: on-time, 1: overdue)'],new\_data['Prediction'])) |
| --- |

| [[34 13]  [ 1 52]]  precision recall f1-score support  0 0.97 0.72 0.83 47  1 0.80 0.98 0.88 53  accuracy 0.86 100  macro avg 0.89 0.85 0.86 100  weighted avg 0.88 0.86 0.86 100  Accuracy: 0.86 |
| --- |

Table 23: Predicting new customer

**Comment:**

* There are 47 customers who are predicted to pay their debts on time, but in fact 13 of them are late payments. And there are 53 customers who are predicted to be overdue, but in fact, 1 of them are on time. The classification report shows that the model has an accuracy of 0.86, which means that it correctly classified 86% of the samples. The f1-score for class 0 is 0.83 while for class 1 it is 0.88. The precision for class 0 is 0.97 and for class 1 it is 0.80. The recall for class 0 is 0.72 and for class 1 it is 0.98. Overall, the decision tree model performed well in terms of accuracy, precision, recall, and f1-score for new customer data set.

**CONCLUSION**

The Decision Tree model is a powerful visualization tool that helps us make efficient credit decisions, saving time and cost in assessing client risk. By processing input data and filtering out 5 independent variables to run the model, the results show that the performance of the Decision Tree model for forecasting on this data set is very good and reliable. This is demonstrated by the high evaluation indicators in the Confusion Matrix and the model’s AUC. The results of running on a new customer data set also show that its applicability in practice is feasible. When compared with the remaining 5 models, all 5 models show good predictability on the data set, with Naive Bayes having the lowest performance and the Decision Tree model having the highest performance and stability. The Random Forest model also gives similar results to the Decision Tree model, but it is more difficult to visualize and build lending principles. The KNN model also performs well. Additionally, the upsampling technique has helped improve the quality of the Decision Tree model, thereby increasing its applicability in practice to solve real problems.

From the above results, we can suggest some recommendations as follows:

The current data set has a small number of observations, so the next development step will be to use a larger data set with more observations that better covers the problem with many other variables related to the target variable. We can also run other techniques to further improve the quality of the Decision Tree model.

Banks need to promote the application of digital technology in customer appraisal to ensure the science, speed and improve the quality and efficiency of credit appraisal activities, contributing to limiting bad debts and affecting the quality of customer lending activities. The credit appraisal must strictly comply with the bank's lending policy to achieve a quality loan balance. Before approving a loan, the bank needs to accurately, fully, and objectively evaluate the customer's income and expense sources before lending to ensure the feasibility of the loan plan, regularly monitoring closely monitor the receipts and expenditures of customers after lending to ensure the ability to repay debts on time, avoid arising bad debts that affect lending efficiency.

**REFERENCE**

Agarwal, A., Sharma, P., Alshehri, M., Mohamed, A. A., & Alfarraj, O. (2021). Classification model for accuracy and intrusion detection using machine learning approach. *PeerJ Computer Science*, *7*, e437.

Abdelmoula, A. K. (2015). Bank credit risk analysis with k-nearest-neighbor classifier: Case of Tunisian banks. *Accounting and Management Information Systems*, *14*(1), 79.

Krichene, A. (2017). Using a naive Bayesian classifier methodology for loan risk assessment: Evidence from a Tunisian commercial bank. *Journal of Economics, Finance and Administrative Science*, *22*(42), 3-24.

Mukid, M. A., Widiharih, T., Rusgiyono, A., & Prahutama, A. (2018, May). Credit scoring analysis using weighted k nearest neighbor. In *Journal of Physics: Conference Series* (Vol. 1025, No. 1, p. 012114). IOP Publishing.

Okesola, O. J., Okokpujie, K. O., Adewale, A. A., John, S. N., & Omoruyi, O. (2017, December). An improved bank credit scoring model: a naïve Bayesian approach. In *2017 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 228-233). IEEE.

Pandey, P., & Bandhu, K. C. (2022). A credit risk assessment on borrowers classification using optimized decision tree and KNN with bayesian optimization. *International Journal of Information Technology*, *14*(7), 3679-3689.

Roy, A. G., & Urolagin, S. (2019). Credit risk assessment using decision tree and support vector machine based data analytics. In *Creative Business and Social Innovations for a Sustainable Future: Proceedings of the 1st American University in the Emirates International Research Conference—Dubai, UAE 2017* (pp. 79-84). Springer International Publishing.

Satchidananda, S. S., & Simha, J. B. (2006). Comparing decision trees with logistic regression for credit risk analysis. *International Institute of Information Technology, Bangalore, India*.

Wang, Y., Zhang, Y., Lu, Y., & Yu, X. (2020). A Comparative Assessment of Credit Risk Model Based on Machine Learning——a case study of bank loan data. *Procedia Computer Science*, *174*, 141-149.