PREPARED BY
Laura Tang

PREPARED FOR
NF30036- Business Analytics and Artificial Intelligence

CREDIT CARD
PREDICTION APPROVAL

Table of Contents

Introduction

Credit card qualifications and decisions are extremely vital for every bank. The decision to award a customer with a credit card must be made with a lot of caution since it determines the bank's success or failure. A vivid and thorough process must be used to ensure the right results are achieved. There are various methods that banks use to determine the credit card qualifications for their customers. One method is to use a judgmental strategy. This involves an analyst manually checking each credit card application and judging them. The use of automated systems for credit card qualifications is mainly applicable to small credit cards like customer credits. They are more appropriate since they do not require a lot of information and are not associated with huge risks.

Companies use various credit card scoring mechanisms to determine customer qualification. The scoring systems use different algorithms that have various rules and factors to consider before scoring. These algorithms check each customer’s data and apply the assigned rules to determine the customer’s capability. These rules must use customer-specific data to reduce financial risks due to defaulting. Most scoring systems use data such as the status of an existing checking A/C, duration in a month, customer’s credit history, savings account, customer’s bonds, present employment status, installment rate in percentage of disposable income, personal status, and sex, customer’s other debts, customer’s guarantors, and the customer’s other installment plans. This information helps in making careful decisions regarding the limits and customers’ suitability for a credit card.

Business Objective

Credit cards pose a huge risk to the business. The main question is how the right qualifiers can be identified. Every business is started to reduce costs, minimize risks, and maximize profits. Credit card is a potential risk to the business, hence careful considerations must be made before a business offers one. There must be a clear scoring process before a customer is provided with a credit card. This ensures that the risk of defaulters is highly minimized (Islam, Zhou & Li, 2009). Defaulting compromises with the financial base of a business. It may lead to a disruption of normal business operations and expansion needs. Defaulters may lead to bankruptcy and a complete shutdown of the business in critical and severe conditions. This paper proposes the development of a predictive model to answer this question.

The main idea behind this project is to come up with the best model for determining a customer's suitability for a credit card. This involves the development of a predictive model using the credit card approval dataset. The predictive model will be designed to use customer information to determine their eligibility for credit cards. The predictive model will be trained using the dataset given. The dataset will be divided into training data and testing data. The model will be trained using third quarters of the data from the dataset. Training must be done to act as rules for determining the eligibility process (Wah & Ibrahim, 2010). Training the model with more data ensures that all possibilities and potential dynamics of customer data are captured in the model's rules. The testing data will be used to test the system's accuracy to ascertain its effectiveness in determining the suitability of customers for credit cards. Training will be done using actual and real data. The model should not use dummy data since it may generate wrong results when applied to a real-world business. Financial information is very sensitive. Most of the information must be kept accurate to prevent wrong predictions. Potential defaulters should be easily noticeable from the model and should be captured in the predictive model's rules. The final system will be fully tested to limit errors and prevent risks.

Data Preparation and Exploration

In this section, we will prepare and explore the dataset by identifying missing, incorrect values and data types, applying suitable imputation methods and visualising data to provide insights into the given dataset.

Data Preparation

1. Missing Values

The dataset does not contain any missing values (N/A) as shown in the variables table of the dataset (see Figure 1) and R studio code developed to identify missing values in the dataset (see Figure 2). The result for the code is '0', showing that there are no missing values detected.

Variables Table

Variable Name	Role	Type	Demographic	Description	Units	Missing Values
X10	Feature	Integer		PAY_5		no
X11	Feature	Integer		PAY_6		no
X12	Feature	Integer		BILL_AMT1		no
X13	Feature	Integer		BILL_AMT2		no
X14	Feature	Integer		BILL_AMT3		no
X15	Feature	Integer		BILL_AMT4		no
X16	Feature	Integer		BILL_AMT5		no
X17	Feature	Integer		BILL_AMT6		no
X18	Feature	Integer		PAY_AMT1		no
X19	Feature	Integer		PAY_AMT2		no

Figure 1: Variables Table of the Default Credit Card Clients (UC Irvine Machine Learning Repository, 2016)

```
#Missing values
sum(is.na(credit))
```

```
> #Missing values
> sum(is.na(credit))
[1] 0
```

Figure 2: R codes to identify missing values

2. Incorrect Values

In categorical variables including EDUCATION, MARRIAGE and PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6, unknown values i.e values that are not described in the data dictionary are detected. EDUCATION contains three levels 1 = graduate school; 2 = university; 3 = high school; 4 = others; however, when using the unique() function to extract unique elements from this variable, we receive unidentified values like 0, 5 and 6 (see Figure 3). To address this data problem, we replace unknown values (0,5 and 6) with 4 = others (See Figure 4) and check the unique elements of EDUCATION again using unique() function (See Figure 5).

```
> unique(credit$EDUCATION) #There are unknown values such as 0,5,6
[1] 2 1 3 5 4 6 0
Levels: 0 1 2 3 4 5 6
>
```

Figure 3: Unique elements in EDUCATION

```
47  #Replace values of 0,5,6 into 4 as "other"
48  credit$EDUCATION[credit$EDUCATION == 0] <- 4
49  credit$EDUCATION[credit$EDUCATION == 5] <- 4
50  credit$EDUCATION[credit$EDUCATION == 6] <- 4
51  unique(credit$EDUCATION) #Check values
```

Figure 4: Replace unknown values in EDUCATION

```
> unique(credit$EDUCATION) #Check values
[1] 2 1 3 4
>
```

Figure 5: Result of unique(EDUCATION) after replacing unknown values

The variable MARRIAGE contains three levels including 1 = married, 2 = single and 3 = others. When using the unique () function on MARRIAGE, we have identified one unknown value '0' (see Figure 6). To address this, we replace this value with 3 = others (see Figure 7) and check the unique elements of MARRIAGE again (see Figure 8).

```
> unique(credit$MARRIAGE) #There are unknown value such as 0
[1] 1 2 3 0
>
```

Figure 6: Unique elements in MARRIAGE

```
#replace values of 0 into 3 as "other"
credit$MARRIAGE[credit$MARRIAGE == 0] <- 3
```

Figure 7: Replace unknown value in MARRIAGE

```
> unique(credit$MARRIAGE) #check values
[1] 1 2 3
>
```

Figure 8: Unique elements in MARRIAGE after replacing unknown value

The variables PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6 contain multiple levels -1 = pay duly; 1 = payment delay for one month, 2 = payment delay for two months; ...; 8 = payment delay for eight months; 9 = payment delay for nine months and above. However, when running the unique() function on these variables, we receive results of unknown values including '0' and '-2' (see Figure 9). To solve this data problem, we replace them with the higher and closest values that already exist in the dataset, meaning that '0' will be replaced by '1' (payment delay for one month) and '-2' will be replaced by '-1' (pay duly) (see Figure 10). After that, we will check the unique elements in these variable again using the unique() function (see Figure 11).

```
> unique(credit$PAY_0)#There are unknown value such as 0,-2
 [1]  2 -1  0 -2  1  3  4  8  7  5  6
> unique(credit$PAY_2)#There are unknown value such as 0,-2
 [1]  2  0 -1 -2  3  5  7  4  1  6  8
> unique(credit$PAY_3)#There are unknown value such as 0,-2
 [1] -1  0  2 -2  3  4  6  7  1  5  8
> unique(credit$PAY_4)#There are unknown value such as 0,-2
 [1] -1  0 -2  2  3  4  5  7  6  1  8
> unique(credit$PAY_5)#There are unknown value such as 0,-2
 [1] -2  0 -1  2  3  5  4  7  8  6
> unique(credit$PAY_6)#There are unknown value such as 0,-2
 [1] -2  2  0 -1  3  6  4  7  8  5
>
```

Figure 9: Unique elements in PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6

```
#Replace values of 0 to 1 and -2 to -1:
credit$PAY_0[credit$PAY_0 == 0] <- 1
credit$PAY_0[credit$PAY_0 == -2] <- -1

credit$PAY_2[credit$PAY_2 == 0] <- 1
credit$PAY_2[credit$PAY_2 == -2] <- -1

credit$PAY_3[credit$PAY_3 == 0] <- 1
credit$PAY_3[credit$PAY_3 == -2] <- -1

credit$PAY_4[credit$PAY_4 == 0] <- 1
credit$PAY_4[credit$PAY_4 == -2] <- -1

credit$PAY_5[credit$PAY_5 == 0] <- 1
credit$PAY_5[credit$PAY_5 == -2] <- -1

credit$PAY_6[credit$PAY_6 == 0] <- 1
credit$PAY_6[credit$PAY_6 == -2] <- -1
```

Figure 10: Replace unknow values in PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6

```
> unique(credit$PAY_0)
[1]  2 -1  1  3  4  8  7  5  6
> unique(credit$PAY_2)
[1]  2  1 -1  3  5  7  4  6  8
> unique(credit$PAY_3)
[1] -1  1  2  3  4  6  7  5  8
> unique(credit$PAY_4)
[1] -1  1  2  3  4  5  7  6  8
> unique(credit$PAY_5)
[1] -1  1  2  3  5  4  7  8  6
> unique(credit$PAY_6)
[1] -1  2  1  3  6  4  7  8  5
>
```

Figure 11: Unique elements in PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6 after replacing unknown values

3. Incorrect Data Types

To check the datatype of the variables in the dataset, we use str() function to generate a summary of the data objects including the type, length and a preview of their contents (see Figure 12).

```
$ ID                        : int  1 2 3 4 5 6 7 8 9 10 ...
$ LIMIT_BAL                 : int  20000 120000 90000 50000 50000 50000 500000 100000 140000 20000 ...
$ SEX                       : int  2 2 2 2 1 1 1 2 2 1 ...
$ EDUCATION                 : int  2 2 2 2 2 1 1 2 3 3 ...
$ MARRIAGE                  : int  1 2 2 1 1 2 2 2 1 2 ...
$ AGE                       : int  24 26 34 37 57 37 29 23 28 35 ...
$ PAY_0                     : int  2 -1 0 0 -1 0 0 0 0 -2 ...
$ PAY_2                     : int  2 2 0 0 0 0 0 -1 0 -2 ...
$ PAY_3                     : int  -1 0 0 0 -1 0 0 -1 2 -2 ...
$ PAY_4                     : int  -1 0 0 0 0 0 0 0 0 -2 ...
$ PAY_5                     : int  -2 0 0 0 0 0 0 0 0 -1 ...
$ PAY_6                     : int  -2 2 0 0 0 0 0 -1 0 -1 ...
$ BILL_AMT1                 : int  3913 2682 29239 46990 8617 64400 367965 11876 11285 0 ...
$ BILL_AMT2                 : int  3102 1725 14027 48233 5670 57069 412023 380 14096 0 ...
$ BILL_AMT3                 : int  689 2682 13559 49291 35835 57608 445007 601 12108 0 ...
$ BILL_AMT4                 : int  0 3272 14331 28314 20940 19394 542653 221 12211 0 ...
$ BILL_AMT5                 : int  0 3455 14948 28959 19146 19619 483003 -159 11793 13007 ...
$ BILL_AMT6                 : int  0 3261 15549 29547 19131 20024 473944 567 3719 13912 ...
$ PAY_AMT1                  : int  0 0 1518 2000 2000 2500 55000 380 3329 0 ...
$ PAY_AMT2                  : int  689 1000 1500 2019 36681 1815 40000 601 0 0 ...
$ PAY_AMT3                  : int  0 1000 1000 1200 10000 657 38000 0 432 0 ...
$ PAY_AMT4                  : int  0 1000 1000 1100 9000 1000 20239 581 1000 13007 ...
$ PAY_AMT5                  : int  0 0 1000 1069 689 1000 13750 1687 1000 1122 ...
$ PAY_AMT6                  : int  0 2000 5000 1000 679 800 13770 1542 1000 0 ...
$ default.payment.next.month: int  1 1 0 0 0 0 0 0 0 0 ...
```

Figure 12: Variables type and length

Through examining additional variable information on the dataset website, we have identified data problems regarding the datatypes of categorical variables including SEX, EDUCATION, MARRIAGE, PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6 and default.payment.next.month. The additional variable information demonstrates these variables as categorical values: GENDER (1 = male; 2 = female), EDUCATION (1 = graduate school; 2 = university; 3 = high school; 4 = others), MARRIAGE (1= married; 2 =

single; 3 = others), PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6 (-1 = pay duly; 1 = payment delay for one month, 2 = payment delay for two months; ...; 8 = payment delay for eight months; 9 = payment delay for nine months and above) and default.payment.next.month (1 = Yes, 0 = No). However, in the dataset these variables are represented using the 'int' (integer) data type, showing that the contents are stored as whole numbers instead of categories.

Using the integer data type for categorical data in R can lead to various issues such as misinterpretation of data, lack of ordinality and visualisation issues, thus needs to be addressed before sampling and applying predictive models.

The method to address this data problem is to use as.factor() function to change the data type of categorical variables including SEX, EDUCATION, MARRIAGE, PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6 and default.payment.next.month from 'int' to 'Factor' (see Figure 13). We will then use the str() function to check the new datatype of these variables (see Figure 14).

```
# replace data types to factors
credit$SEX <- as.factor(credit$SEX)
credit$EDUCATION <- as.factor(credit$EDUCATION)
credit$MARRIAGE <- as.factor(credit$MARRIAGE)
credit$PAY_0 <- as.factor(credit$PAY_0)
credit$PAY_2 <- as.factor(credit$PAY_2)
credit$PAY_3 <- as.factor(credit$PAY_3)
credit$PAY_4 <- as.factor(credit$PAY_4)
credit$PAY_5 <- as.factor(credit$PAY_5)
credit$PAY_6 <- as.factor(credit$PAY_6)
credit$`default.payment.next.month` <- as.factor(credit$`default.payment.next.month`)
```

Figure 13: Change data type from 'int' to 'Factor'

```
'data.frame':   30000 obs. of  25 variables:
 $ ID                        : int  1 2 3 4 5 6 7 8 9 10 ...
 $ LIMIT_BAL                 : int  20000 120000 90000 50000 50000 50000 500000 100000 140000 20000 ...
 $ SEX                       : Factor w/ 2 levels "1","2": 2 2 2 2 1 1 1 2 2 1 ...
 $ EDUCATION                 : Factor w/ 7 levels "0","1","2","3",..: 3 3 3 3 3 2 2 3 4 4 ...
 $ MARRIAGE                  : Factor w/ 4 levels "0","1","2","3": 2 3 3 2 2 3 3 3 2 3 ...
 $ AGE                       : int  24 26 34 37 57 37 29 23 28 35 ...
 $ PAY_0                     : Factor w/ 11 levels "-2","-1","0",..: 5 2 3 3 2 3 3 3 3 1 ...
 $ PAY_2                     : Factor w/ 11 levels "-2","-1","0",..: 5 2 3 3 2 3 3 3 3 1 ...
 $ PAY_3                     : Factor w/ 11 levels "-2","-1","0",..: 2 3 3 3 2 3 3 2 5 1 ...
 $ PAY_4                     : Factor w/ 11 levels "-2","-1","0",..: 2 3 3 3 3 3 3 3 3 1 ...
 $ PAY_5                     : Factor w/ 10 levels "-2","-1","0",..: 1 3 3 3 3 3 3 3 3 2 ...
 $ PAY_6                     : Factor w/ 10 levels "-2","-1","0",..: 1 4 3 3 3 3 3 2 3 2 ...
 $ BILL_AMT1                 : int  3913 2682 29239 46990 8617 64400 367965 11876 11285 0 ...
 $ BILL_AMT2                 : int  3102 1725 14027 48233 5670 57069 412023 380 14096 0 ...
 $ BILL_AMT3                 : int  689 2682 13559 49291 35835 57608 445007 601 12108 0 ...
 $ BILL_AMT4                 : int  0 3272 14331 28314 20940 19394 542653 221 12211 0 ...
 $ BILL_AMT5                 : int  0 3455 14948 28959 19146 19619 483003 -159 11793 13007 ...
 $ BILL_AMT6                 : int  0 3261 15549 29547 19131 20024 473944 567 3719 13912 ...
 $ PAY_AMT1                  : int  0 0 1518 2000 2000 2500 55000 380 3329 0 ...
 $ PAY_AMT2                  : int  689 1000 1500 2019 36681 1815 40000 601 0 0 ...
 $ PAY_AMT3                  : int  0 1000 1000 1200 10000 657 38000 0 432 0 ...
 $ PAY_AMT4                  : int  0 1000 1000 1100 9000 1000 20239 581 1000 13007 ...
 $ PAY_AMT5                  : int  0 0 1000 1069 689 1000 13750 1687 1000 1122 ...
 $ PAY_AMT6                  : int  0 2000 5000 1000 679 800 13770 1542 1000 0 ...
 $ default.payment.next.month: Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 ...
>
```

Figure 14: Summary of data type and length after changing datatypes

4. Duplicates

There is no duplicated values identified in the given dataset as when we run the duplicate() code on the dataset (see Figure 15), we receive '0' as a result, meaning that the dataset does not contain any duplicates (see Figure 16).

```
#Check duplicate
sum(duplicated(credit))
```

Figure 15: Checking duplicated values in the dataset

```
> #Check duplicate
> sum(duplicated(credit))
[1] 0
>
```

Figure 16: Result of duplicate () function

5. Summary

Data Problem		Imputation Methods

Incorrect values	EDUCATION	Replace '0', '5' and '6' with '4 = others'
		`#Replace values of 0,5,6 into 4 as "other"`
		`credit$EDUCATION[credit$EDUCATION == 0] <- 4`
		`credit$EDUCATION[credit$EDUCATION == 5] <- 4`
		`credit$EDUCATION[credit$EDUCATION == 6] <- 4`
		`unique(credit$EDUCATION) #Check values`

	MARRIAGE	Replace '0' with '3 = others'
		`#replace values of 0 into 3 as "other"`
		`credit$MARRIAGE[credit$MARRIAGE == 0] <- 3`

	PAY_0	Replace '0' with '1 = payment delay for one month' and '-2' with '-1 = payment duly'
		`credit$PAY_0[credit$PAY_0 == 0] <- 1`
		`credit$PAY_0[credit$PAY_0 == -2] <- -1`

	PAY_2	Replace '0' with '1 = payment delay for one month' and '-2' with '-1 = payment duly'
		`credit$PAY_2[credit$PAY_2 == 0] <- 1`
		`credit$PAY_2[credit$PAY_2 == -2] <- -1`

	PAY_3	Replace '0' with '1 = payment delay for one month' and '-2' with '-1 = payment duly'

		`credit$PAY_3[credit$PAY_3 == 0] <- 1` `credit$PAY_3[credit$PAY_3 == -2] <- -1`

	PAY_4	Replace '0' with '1 = payment delay for one month' and '-2' with '-1 = payment duly'
		`credit$PAY_4[credit$PAY_4 == 0] <- 1`
		`credit$PAY_4[credit$PAY_4 == -2] <- -1`

	PAY_5	Replace '0' with '1 = payment delay for one month' and '-2' with '-1 = payment duly'
		`credit$PAY_5[credit$PAY_5 == 0] <- 1`
		`credit$PAY_5[credit$PAY_5 == -2] <- -1`

	PAY_6	Replace '0' with '1 = payment delay for one month' and '-2' with '-1 = payment duly'
		`credit$PAY_6[credit$PAY_6 == 0] <- 1`
		`credit$PAY_6[credit$PAY_6 == -2] <- -1`

Incorrect data types	SEX	Change datatype from 'int' to 'Factor'
		`credit$SEX <- as.factor(credit$SEX)`
		`credit$EDUCATION <- as.factor(credit$EDUCATION)`
		`credit$MARRIAGE <- as.factor(credit$MARRIAGE)`
		`credit$PAY_0 <- as.factor(credit$PAY_0)`
		`credit$PAY_2 <- as.factor(credit$PAY_2)`
		`credit$PAY_3 <- as.factor(credit$PAY_3)`
		`credit$PAY_4 <- as.factor(credit$PAY_4)`
		`credit$PAY_5 <- as.factor(credit$PAY_5)`
		`credit$PAY_6 <- as.factor(credit$PAY_6)`
		`credit$\`default.payment.next.month\` <- as.factor(credit$\`default.payment.next.month\`)`

	EDUCATION	
	MARRIAGE	
	PAY_0	
	PAY_2	
	PAY_3	
	PAY_4	
	PAY_5	
	PAY_6	
	default.payment.next.month	

Table 1: Summary of the imputation methods applied

Data Dictionary

After cleaning the dataset, a data dictionary is developed to provide a comprehensive reference document that composes the information about data structure, data type, field size, description and defining values. This ensures the clarity and consistency of data throughout the report.

Column	*Type*	*Field size*	*Description*	*Values*
ID	String	10	The unique identifier for each borrower	Numeric
LIMIT_BAL	Integer	10	Amount of the given credit (NT dollars) including both the individual consumer credit and his/her family (supplementary) credit	Numeric (NT dollars)
SEX	Nominal	1	Gender of the borrower	1 = male, 2 = female
EDUCATION	Ordinal	1	Education level of the borrower	1 = graduate school; 2 = university; 3 = high school; 4 = others
MARRIAGE	Nominal	1	Marital status of the borrower	1 = married, 2 = single and 3 = others
AGE	Integer	3	Age of the borrower	Numeric (years)
PAY_0	Ordinal	1	Repayment status in September 2005	-1 = pay duly; 1 = payment delay for one month, 2 = payment delay for two months; ...; 8 = payment delay for eight months; 9 = payment delay for nine months and above
PAY_2	Ordinal	1	Repayment status in August 2005	Same as PAY_0
PAY_3	Ordinal	1	Repayment status in July 2005	Same as PAY_0

PAY_4	Ordinal	1	Repayment status in June 2005	Same as PAY_0
PAY_5	Ordinal	1	Repayment status in May 2005	Same as PAY_0
PAY_6	Ordinal	1	Repayment status in April 2005	Same as PAY_0
BILL_AMT1	Integer	10	Amount of bill statement in September 2005 (in NT dollars)	Numeric (NT dollars)
BILL_AMT2	Integer	10	Amount of bill statement in August 2005 (in NT dollars)	Numeric (NT dollars)
BILL_AMT3	Integer	10	Amount of bill statement in July 2005 (in NT dollars)	Numeric (NT dollars)
BILL_AMT4	Integer	10	Amount of bill statement in June 2005 (in NT dollars)	Numeric (NT dollars)
BILL_AMT5	Integer	10	Amount of bill statement in May 2005 (in NT dollars)	Numeric (NT dollars)
BILL_AMT6	Integer	10	Amount of bill statement in April 2005 (in NT dollars)	Numeric (NT dollars)
PAY_AMT1	Integer	10	Amount paid in September 2005 (in NT dollars)	Numeric (NT dollars)
PAY_AMT2	Integer	10	Amount paid in August 2005 (in NT dollars)	Numeric (NT dollars)

PAY_AMT3	Integer	10	Amount paid in July 2005 (in NT dollars)	Numeric (NT dollars)
PAY_AMT4	Integer	10	Amount paid in June 2005 (in NT dollars)	Numeric (NT dollars)
PAY_AMT5	Integer	10	Amount paid in May 2005 (in NT dollars)	Numeric (NT dollars)
PAY_AMT6	Integer	10	Amount paid in April 2005 (in NT dollars)	Numeric (NT dollars)
default payment next month	Binary	1	Default payment status of the borrower	1 = Yes, 0 = No

Table 2 Data Dictionary of the Default of Credit Card Clients dataset

Data Exploration

This section offers a visual exploration of key patterns identified in the dataset. By using various charts and graphs, we aim to provide a clear presentation of the trends and extract valuable insights to enhance the understanding of the dataset.

1. Age Distribution

The histogram chart of borrowers' age is crafted by using ggplot2 package and geom_histogram() function in R language (see Appendix A). The histogram represents the distribution of ages in the dataset. As observed in Chart 1, the histogram is skewed to the left, indicating a prevalence of younger borrowers in the dataset, typically aged between 20 and 40 years old. This dominance of younger demographics can be the result of various factors including societal and consumer behaviours, the scope of the original study and potential biases in data collection. Regarding societal and consumer behaviours factors, older demographics may be underrepresented due to lower engagement in credit-related activities as they have already navigated through significant life events like buying a house, applying for mortgages, etc.

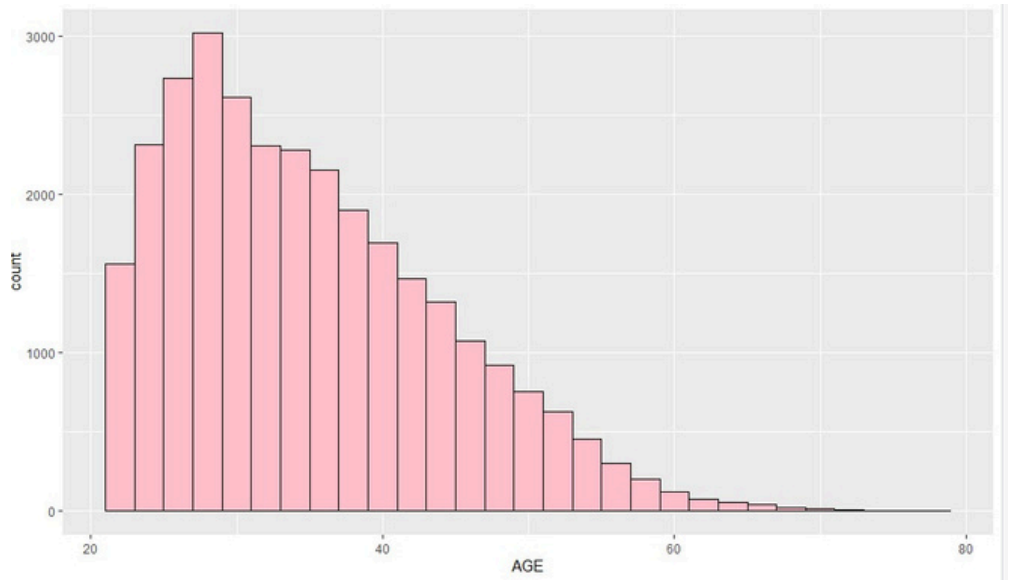


Chart 1: Histogram Chart of Borrower's Age

2. Default payment status in male and female

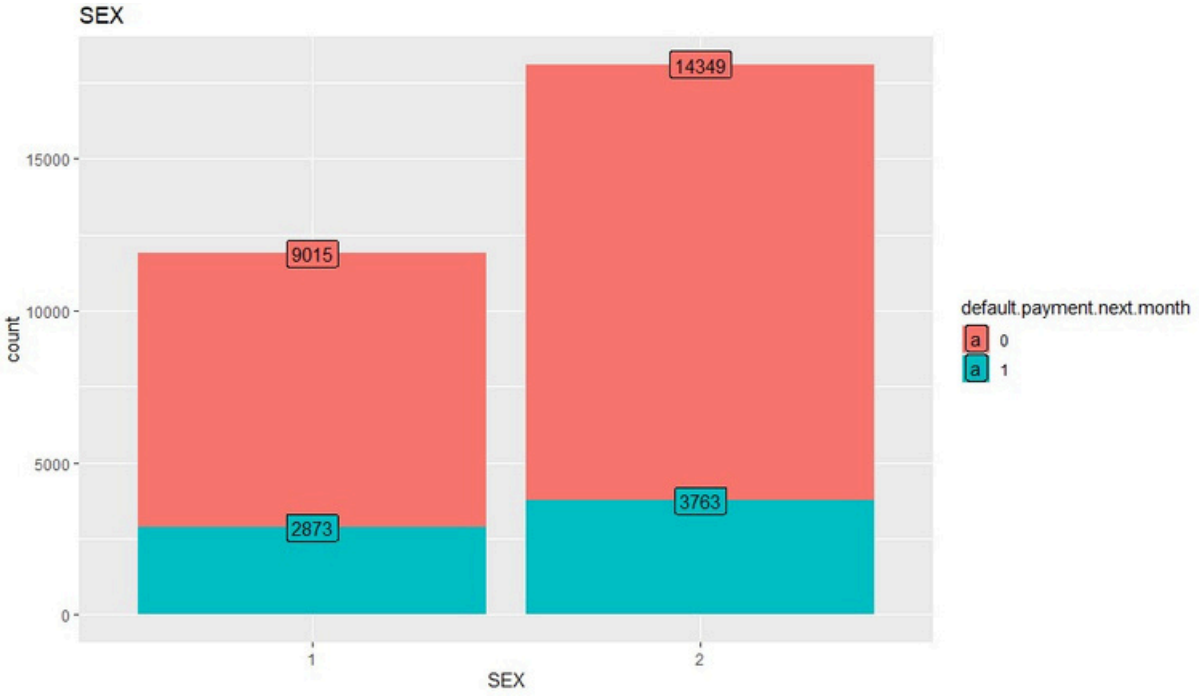


Chart 2 Default payment status in different genders

Gender	Default Rate
Male	24.17%
Female	20.77%

Table 3 Default rates in male and female

The bar chart is generated using the ggplot package and geom_bar() function in R language (See Appendix A). The plot illustrates the distribution of borrowers across gender categories and highlights the gender differences in default rates. Notably, the dataset shows a higher number of female borrowers compared to males, suggesting that females might be more involved in credit-related activities than males.

To assess the gender differences in credit default risks, we calculate default rates among male and female borrower (see Appendix A), with results summarized in Table 1. The default rate among male borrowers (24.17%) is slightly higher than that among female borrowers (20.77%) (see Table 3), indicating that male borrowers are more likely to default their payments considering the scope of the dataset.

To measure the association between SEX and default payment next month, a chi-squared test of independence is performed (see Appendix A). Two hypotheses are developed as follows:
Ho: There is no association between gender and default payment next month
H1: There is an association between gender and default payment next month

```
data:  contingency_table
X-squared = 47.709, df = 1, p-value = 4.945e-12
```

Figure 17: Chi-squared test of independence between SEX and default payment next month

Since the p-value < 0.05 (see Figure 17), we reject the null hypothesis (H0) which means that the variables 'SEX' and 'default payment next month' are not independent in the dataset. Therefore, gender might be one of the significant determinants influencing the default payment status next month.

3. Default payment status in different educational levels

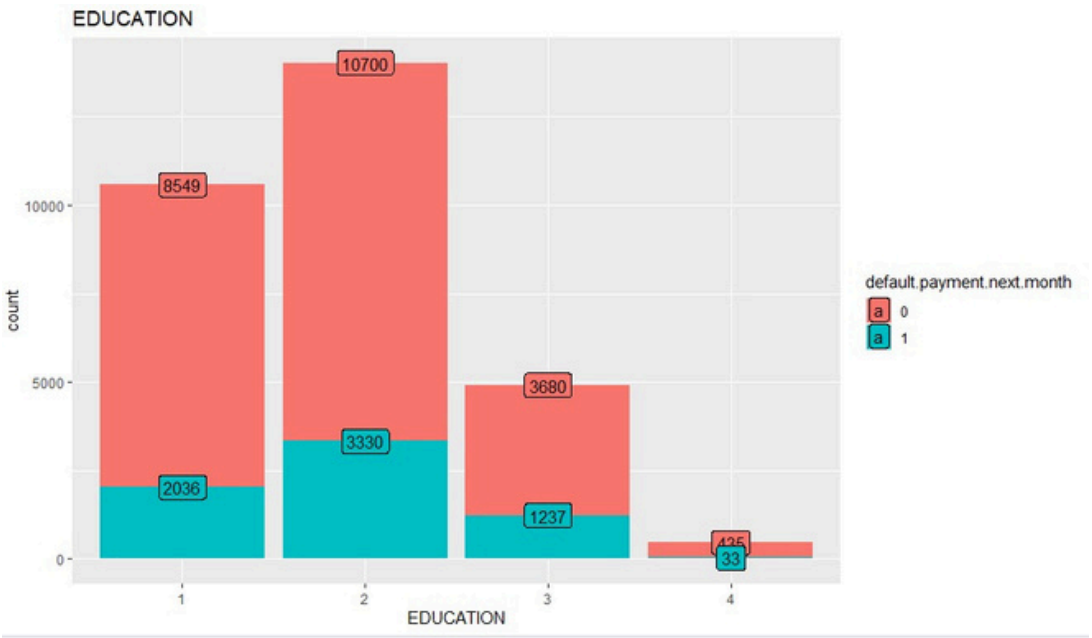


Chart 3: Default payment status in different educational levels

Education Levels	Default Rate
Graduate School	19.23%
University	23.73%
High School	25.16%
Others	7.05%

Table 4: Default rates in different education levels (see Appendix A)

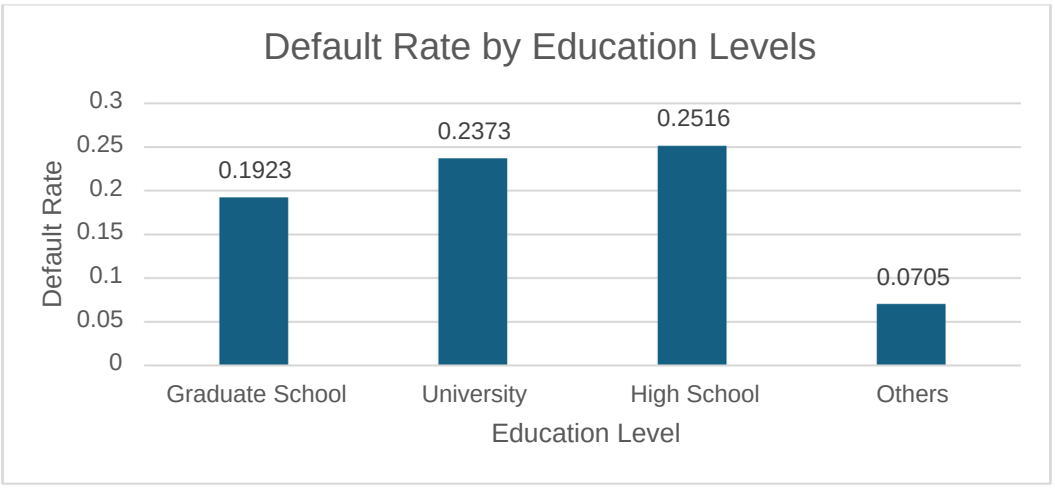


Chart 4: Default rates by Education Levels

The bar chart, created by using ggplot and geom_bar() function in R language (see Appendix A), illustrates the default payment status across four educational levels: 1 = graduate school, 2 = university; 3 = high school and 4 = others. As shown in Chart 3, the number of borrowers with a university education is the highest in the dataset, followed by those with a graduate school education and those with high school education.

According to Table 4 and Chart 4, it appears that the lower the level of education a borrower has, the more likely they are to have payment defaults. People in High School category have the highest default rate, at 25.16%, meaning approximately one in every four people receiving only High School education will end up defaulting on his/her payment. Meanwhile, people in Graduate School category have the lowest default rate among the three ('Others' excluded), at around 19.23%. This trend is consistent with findings from a study by Aalto University (2021) which reported that individuals with only comprehensive school education have a 27% likelihood of defaulting on payments, higher than that among people with any higher education levels (Aalto University, 2021). The higher default rate among people with low level of education might be attributed to limited high-paid job opportunities or lack of financial literacy (Aalto University, 2021). Limited access to well-paying jobs can result in financial constraints and difficulty in covering bills, leading to payment defaults. Furthermore, individuals with lower level of education might receive insufficient financial education in high school, resulting in challenge in managing their finances effectively.

To test the association between 'EDUCATION' and 'default payment next month', we conduct chi-squared test of independence between these variables (see Appendix A). The two hypotheses are as follows:

H0: There is no association between 'EDUCATION' and 'default payment next month' H1: There is an association between 'EDUCATION' and 'default payment next month'

```
data:  contingency_table_1
X-squared = 160.41, df = 3, p-value < 2.2e-16
```

Figure 18: Chi-squared test of independence between EDUCATION and default payment next month

According to the result of the chi-squared test (see Figure 18), p-value < 2.2e-16 < 0.05; therefore, we reject the null hypothesis (H0: There is no association between 'EDUCATION' and 'default payment next month'). Additionally, considering the high strength of association

($\chi 2 = 160.61$), we conclude that the default payment status is highly influenced by the borrowers' education level.

4. Bill Amount in July 2005 and Default Status

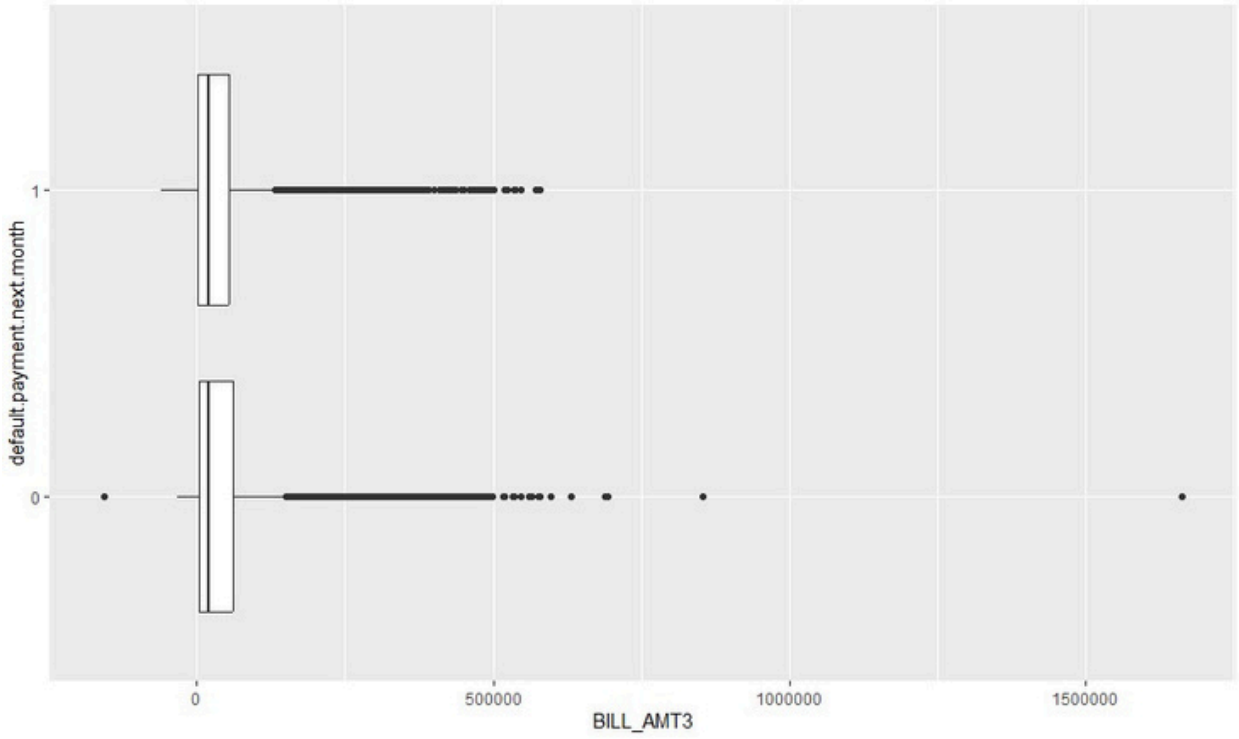


Chart 5: Box plot of bill amount in July 2005 and default payment status

The box plot generated using ggplot package and geom_boxplot() function in R language shows the distribution of the bill amount of borrowers in July 2005 across two levels of default payment statuses (Yes and No/ Defaulted and Non-defaulted) (see Appendix A). Notably, in the non -default group there are two outliers, one being far to the left and other far to the right. The former indicates that a borrower might have been eligible for a refund, while the latter shows that there is an individual with a significantly higher bill than other borrowers in July 2025 who did not choose to default. These outliers have minimal impact on the outcomes of the prediction models and the overall trends identified in the dataset. Nevertheless, it shows that an individual who have experienced a substantially high bill in July managed to avoid default on his/her payment, possibly due to financial management skills.

Decision Tree

Decision trees are highly effective tools for categorization and prediction. Decision trees describe rules that humans can understand and apply in knowledge systems like databases. A decision tree is a tree-structured classifier made up of decision nodes, leaf nodes, and edge paths. Decision trees classify cases or examples from root to leaf node (R, 2015).

Load Data and Packages

Library (caret) Library (rpart) Library (rpart.plot)

Split data into train set and test set

The training and testing dataset's outcome variable is distributed through the use of the createDataPartition with a ratio of 70% training and 30% testing, which guarantees a stratified sample. This strategy has been selected over random splitting because it helps to prevent sample bias and guarantees that both divides are true to the entire dataset.

```
> table(train_data$default.payment.next.month) / nrow(train_data)

        0         1
0.7787724 0.2212276
> table(test_data$default.payment.next.month) / nrow(test_data)

        0         1
0.7788643 0.2211357
```

Figure 19: The Result of Data Partition

Fit the decision tree model and predict the test set

During model training, errors can be avoided by making sure factor levels are legit R variable name. Cross-validation reduces overfitting while helping to evaluate the model's capacity to generalize to a different dataset. It offers a reliable estimation of the model's performance by training and validating on multiple data subsets.

```
#Make predictions on the test set
dt_predictions <- predict(decision_tree_model, test_data)
```

Figure 20: Code for predictions on the test data

Evaluation

The confusion matrix will offer a good understanding of the model's performance. Providing details of true positives, true negatives, false positives, false negatives as well as accuracy, precision, and recall, which offer insights into the model's performance and places for improvement.

Random Forest

Random Forest is a popular machine learning algorithm, developed by Leo Breiman and Adele Cutler that combines the outputs of numerous decision trees to produce a single outcome, its adaptability and ease of use boost its popularity, as it is capable of handling both classification and regression problems (IBM, 2024)

Load Data and Packages

'Random Forest' package is required for the creation of Random Forest Model. This model can manage complicated datasets and feature interactions without significant preprocessing. 'Caret' package serves the purpose of advanced data partitioning, model training and performance analysis.

Split data into train set and test set

We split the data into 70% training and 30% testing. CreateDataPartition in the caret package has been used to do this split. Guaranteeing that the ratio of classes in the target variable is identical in both training and testing datasets. This step is essential to guarantee that both subsets are representing the entire dataset and prevent training a model that is skewed towards the majority class. In datasets that are unbalanced, sampling is crucial because random samples alone may cause major issues with class representation.

```
> table(train_data$default.payment.next.month) / nrow(train_data)

        0         1
0.7787724 0.2212276
> table(test_data$default.payment.next.month) / nrow(test_data)

        0         1
0.7788643 0.2211357
>
```

Figure 21: The result of data partition

Fit the random forest model and predict the test set data

Ntree = 100 indicates the number of trees in the forest. Rasing the total of trees will improve the model's robustness, but it will raise computing cost. 100 trees are a fair balance for robust predictions.

Command 'mtry = 3' specifies the number of variables examined at every split in the tree. The default is often one-third of the total variables. Changing this can increase model accuracy and reduce overfitting.

After training, utilizing the model to predict test data is critical for determining how effective the model applies to new data. This stage is important for every predictive model because it determines the model's use in practice.

```
# Predicting on the test data
rf_prediction <- predict(random_forest_model, test_data)
```

Figure 22: Code for predictions on the test data

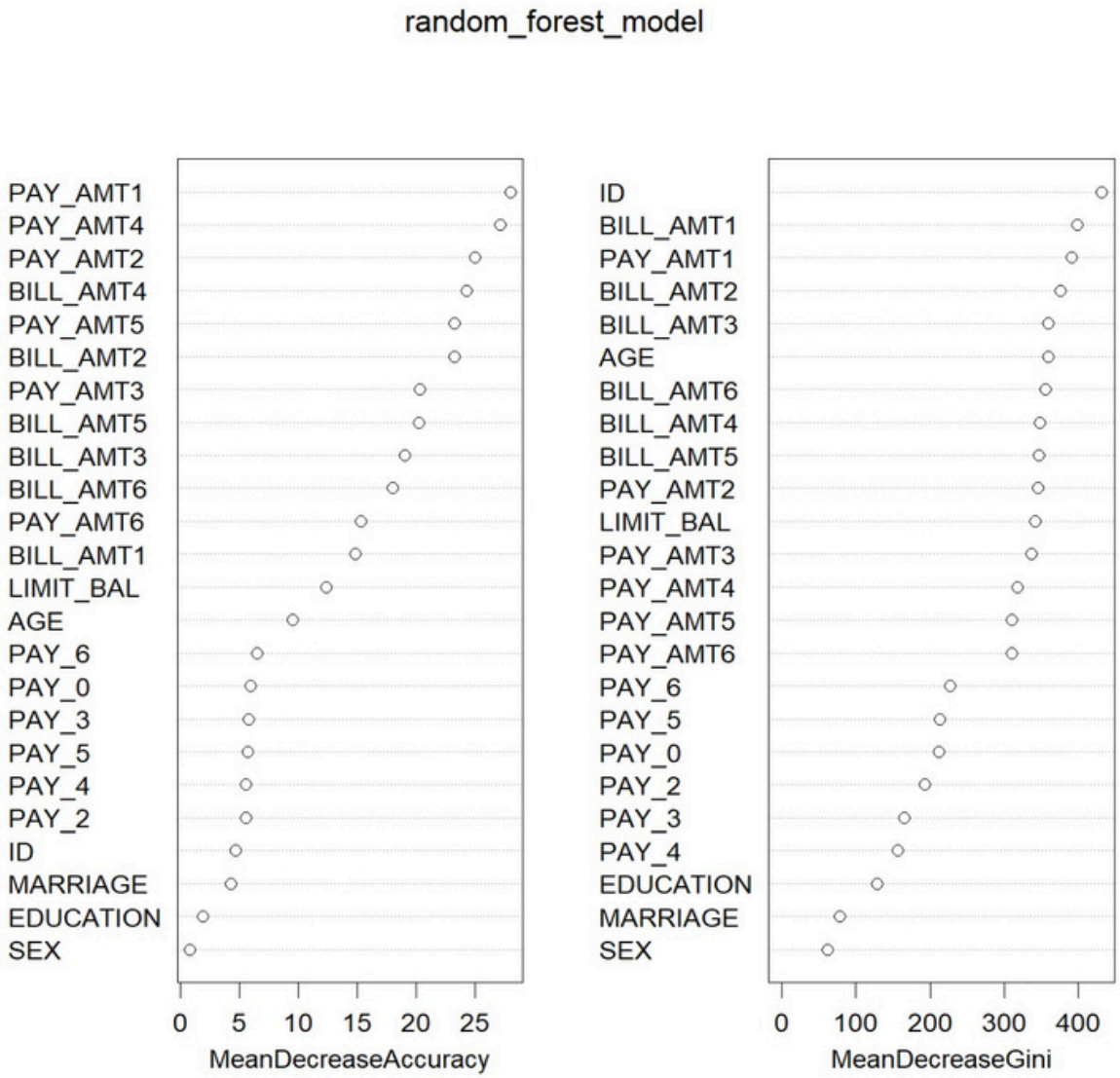


Figure 23: Random Forest Output

Evaluate the model

Using a confusion matrix to evaluate the model offers information about not only overall accuracy but also certain kinds of error (false positives and false negatives). Precision and recall provide a more detailed view of model performance in cases if classes are unbalanced.

Support Vector Machine

The Support Vector Machine (SVM) was a method employed to predict credit card payments. Below is the general process for the execution of SVM.

Load data and necessary packages

Data was cleaned prior and then loaded into the environment. The 'e1071' package is necessary for implementing SVM in R as many functions used later are dependent on the package.

Normalise the data

The data was normalised using the max-min method. If normalisation was not used, ranges may be large, especially in the large credit card dataset. Large ranges result in the suboptimal placement of the decision boundary leading to poor classification performance. This was done to ensure that biases or dominations are mitigated from the dataset as features are brought onto the same scale. This aims to improve the performance of the SVM model as it is highly affected by scale and helps the model learn more effectively.

Split data into training and test sets

Data partitioning was used to split the data into training and testing sets. To ensure the model was properly trained, data partitioning was used to split the dataset with a ratio of 70% training set and 30% testing set. As there is no historical dataset related to this one, this method enables the model's performance to be compared between two sets of data, that is the training set, and test set. Figure X shows the split across the data sets.

```
> # Check for proportion of labels in both training and test split
> prop.table(table(creditCards$default.payment.next.month))

     0      1
0.7788 0.2212
> prop.table(table(creditCards.train$default.payment.next.month))

        0         1
0.7767619 0.2232381
> prop.table(table(creditCards.test$default.payment.next.month))

        0         1
0.7835556 0.2164444
>
```

Figure 24. The result of the data partition

Fit the SVM model into the code and predict the test set of data

The model is fitted into the training set and then used to predict on the test set. The comparison of results from the two sets enables detection of overfitting or underfitting of the model. Prior issues can then be addressed by using other methods for normalisation or data splitting to improve the performance of the model. Furthermore, the threshold value was adjusted across iterations to improve performance. The performance was only affected to the hundredth but ultimately performed best with a value of 0.5.

```
# Predict on the test set
pred.test <- predict(model, creditCards.test)
predictions<-ifelse(pred.test>0.5, 1, 0)
```

Figure 25. Code for prediction on the test set.

Evaluate the model's performance

Metrics for evaluation are precision, recall and accuracy. The use of standard evaluation metrics is used for all other models used to provide a consistent and comprehensive comparison of all models' performance.

K-Nearest Neighbour (KNN)

K-Nearest Neighbours (KNN) is a type of supervised learning algorithm that used in regression and classification to predict the correct class on test and training data and calculating the distance all the data to the K number and calculate the probability of the test data and for decision making. Below will be show case the several precautions on implementation process of KNN model. (Christopher, 2021)

Load data and necessary Packages

- library(readxl)
- library(dplyr)
- library(tidyverse)
- library(class)
- library(caret)
-

The step to implement KNN model

- Step 1: Import data and data cleaning for missing value
- Step 2: Normalization dataset
- Step 3: Data Partitioning
- Step 4: Data training for building KNN model
- Step 5: To create confusion matrix and implement evaluation
-

Data Cleaning

Data cleaning is to identify and remove missing, duplicate and irrelevant value to increase the accuracy of training the data model. It's required "dplyr" packages to run data cleaning process in KNN model. (GeeksforGeeks, 2018)

Normalization Dataset

Data normalization is to calculate the min and max value to present more accurate result plotting in KNN model. Based on the figure as we all can see after normalizing the data set, it can be easier to show case the plotting of the distance of the result to K value. (*K Nearest Neighbour - Why Do You Need to Scale Data in KNN*, n.d.)

```
> summary(creditCards_norm)
       ID          LIMIT_BAL           SEX            EDUCATION         MARRIAGE
 Min.   :0.00   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.25   1st Qu.:0.0404   1st Qu.:0.0000   1st Qu.:0.1667   1st Qu.:0.3333
 Median :0.50   Median :0.1313   Median :1.0000   Median :0.3333   Median :0.6667
 Mean   :0.50   Mean   :0.1591   Mean   :0.6037   Mean   :0.3089   Mean   :0.5173
 3rd Qu.:0.75   3rd Qu.:0.2323   3rd Qu.:1.0000   3rd Qu.:0.3333   3rd Qu.:0.6667
 Max.   :1.00   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
      AGE              PAY_0            PAY_2            PAY_3            PAY_4
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.1207   1st Qu.:0.1000   1st Qu.:0.1000   1st Qu.:0.1000   1st Qu.:0.1000
 Median :0.2241   Median :0.2000   Median :0.2000   Median :0.2000   Median :0.2000
 Mean   :0.2497   Mean   :0.1983   Mean   :0.1866   Mean   :0.1834   Mean   :0.1779
 3rd Qu.:0.3448   3rd Qu.:0.2000   3rd Qu.:0.2000   3rd Qu.:0.2000   3rd Qu.:0.2000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
     PAY_5            PAY_6           BILL_AMT1         BILL_AMT2         BILL_AMT3
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.00000
 1st Qu.:0.1000   1st Qu.:0.1000   1st Qu.:0.1497   1st Qu.:0.06905   1st Qu.:0.08781
 Median :0.2000   Median :0.2000   Median :0.1663   Median :0.08634   Median :0.09737
 Mean   :0.1734   Mean   :0.1709   Mean   :0.1918   Mean   :0.11289   Mean   :0.11216
 3rd Qu.:0.2000   3rd Qu.:0.2000   3rd Qu.:0.2059   3rd Qu.:0.12696   3rd Qu.:0.11938
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.00000
   BILL_AMT4         BILL_AMT5         BILL_AMT6          PAY_AMT1          PAY_AMT2
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000000   Min.   :0.0000000
 1st Qu.:0.1623   1st Qu.:0.0824   1st Qu.:0.2619   1st Qu.:0.001145   1st Qu.:0.0004946
 Median :0.1781   Median :0.0986   Median :0.2741   Median :0.002404   Median :0.0011928
 Mean   :0.2009   Mean   :0.1206   Mean   :0.2909   Mean   :0.006483   Mean   :0.0035156
 3rd Qu.:0.2115   3rd Qu.:0.1304   3rd Qu.:0.2988   3rd Qu.:0.005731   3rd Qu.:0.0029687
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000000   Max.   :1.0000000
    PAY_AMT3          PAY_AMT4          PAY_AMT5          PAY_AMT6
 Min.   :0.0000000   Min.   :0.0000000   Min.   :0.000000   Min.   :0.0000000
 1st Qu.:0.0004352   1st Qu.:0.0004767   1st Qu.:0.000592   1st Qu.:0.0002227
 Median :0.0020088   Median :0.0024155   Median :0.003517   Median :0.0028373
 Mean   :0.0058320   Mean   :0.0077715   Mean   :0.011252   Mean   :0.0098654
 3rd Qu.:0.0050277   3rd Qu.:0.0064626   3rd Qu.:0.009452   3rd Qu.:0.0075662
 Max.   :1.0000000   Max.   :1.0000000   Max.   :1.000000   Max.   :1.0000000
```

Figure 26: Summary of Credit Card dataset

Data Partitioning

Data partitioning is the process to split into test data and train data with the ratio of 70% of train data and 30% of test data to make comparison of two data set performance on KNN model.

```
> prop.table(table(creditCards$default.payment.next.month))

     0      1
0.7788 0.2212
> prop.table(table(creditCards.train$default.payment.next.month))

       0        1
0.779381 0.220619
> prop.table(table(creditCards.test$default.payment.next.month))

        0         1
0.7774444 0.2225556
```

Figure 27: Data Partition Result

Model Building and Evaluation

Model building can clearly define the K value of two data sets in the KNN model for stakeholder better understanding the distance of the K value for helping decision making. After that, to evaluate the model may include accuracy, recall and precision value to calculate the model performance's evaluation and trustworthy of the model result.

model	0	1
0	6715	1786
1	282	217

Figure 28: Model 2x2 result

Artificial Neural Network (ANN)

Artificial Neural Network (ANN) is a machine learning type of model that can train like a human brain, it transmits and analytics information and consist of layer of interconnect "neurons" to provide the ANN model. (*Introduction to ANN | Set 4 (Network Architectures) - GeeksforGeeks*, 2018)

Load data and necessary packages

```
library(readxl)
library(ḍplyr)
library(ẗidyverse)
library(ñeuralnet)
library(m̄lbench)
library(c̄aret)
```

-
-

The step to implement ANN model

- Step 1: Import data and data cleaning for missing value
- Step 2: Normalization dataset
- Step 3: Data Partitioning
- Step 4: Data training for building ANN plotting
- Step 5: To create confusion matrix and implement evaluation

Normalization Dataset

Normalization dataset on ANN model respectively using min and max value of original vector using linear transformation of equation [u = 1, l = 0] or [u = 1, l = -1] and it will maintain all the distance ratios of the original vector after normalization it. (baeldung, 2020)

```
> summary(creditCards_norm)
       ID            LIMIT_BAL            SEX            EDUCATION         MARRIAGE
 Min.   :0.00   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.25   1st Qu.:0.0404   1st Qu.:0.0000   1st Qu.:0.1667   1st Qu.:0.3333
 Median :0.50   Median :0.1313   Median :1.0000   Median :0.3333   Median :0.6667
 Mean   :0.50   Mean   :0.1591   Mean   :0.6037   Mean   :0.3089   Mean   :0.5173
 3rd Qu.:0.75   3rd Qu.:0.2323   3rd Qu.:1.0000   3rd Qu.:0.3333   3rd Qu.:0.6667
 Max.   :1.00   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
      AGE              PAY_0            PAY_2            PAY_3            PAY_4
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.1207   1st Qu.:0.1000   1st Qu.:0.1000   1st Qu.:0.1000   1st Qu.:0.1000
 Median :0.2241   Median :0.2000   Median :0.2000   Median :0.2000   Median :0.2000
 Mean   :0.2497   Mean   :0.1983   Mean   :0.1866   Mean   :0.1834   Mean   :0.1779
 3rd Qu.:0.3448   3rd Qu.:0.2000   3rd Qu.:0.2000   3rd Qu.:0.2000   3rd Qu.:0.2000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
     PAY_5            PAY_6           BILL_AMT1         BILL_AMT2         BILL_AMT3
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.00000
 1st Qu.:0.1000   1st Qu.:0.1000   1st Qu.:0.1497   1st Qu.:0.06905   1st Qu.:0.08781
 Median :0.2000   Median :0.2000   Median :0.1663   Median :0.08634   Median :0.09737
 Mean   :0.1734   Mean   :0.1709   Mean   :0.1918   Mean   :0.11289   Mean   :0.11216
 3rd Qu.:0.2000   3rd Qu.:0.2000   3rd Qu.:0.2059   3rd Qu.:0.12696   3rd Qu.:0.11938
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.00000
   BILL_AMT4         BILL_AMT5         BILL_AMT6          PAY_AMT1          PAY_AMT2
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000000   Min.   :0.0000000
 1st Qu.:0.1623   1st Qu.:0.0824   1st Qu.:0.2619   1st Qu.:0.001145   1st Qu.:0.0004946
 Median :0.1781   Median :0.0986   Median :0.2741   Median :0.002404   Median :0.0011928
 Mean   :0.2009   Mean   :0.1206   Mean   :0.2909   Mean   :0.006483   Mean   :0.0035156
 3rd Qu.:0.2115   3rd Qu.:0.1304   3rd Qu.:0.2988   3rd Qu.:0.005731   3rd Qu.:0.0029687
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000000   Max.   :1.0000000
    PAY_AMT3           PAY_AMT4           PAY_AMT5           PAY_AMT6
 Min.   :0.0000000   Min.   :0.0000000   Min.   :0.000000   Min.   :0.0000000
 1st Qu.:0.0004352   1st Qu.:0.0004767   1st Qu.:0.000592   1st Qu.:0.0002227
 Median :0.0020088   Median :0.0024155   Median :0.003517   Median :0.0028373
 Mean   :0.0058320   Mean   :0.0077715   Mean   :0.011252   Mean   :0.0098654
 3rd Qu.:0.0050277   3rd Qu.:0.0064626   3rd Qu.:0.009452   3rd Qu.:0.0075662
 Max.   :1.0000000   Max.   :1.0000000   Max.   :1.000000   Max.   :1.0000000
```

Figure 29: Normalization Summary

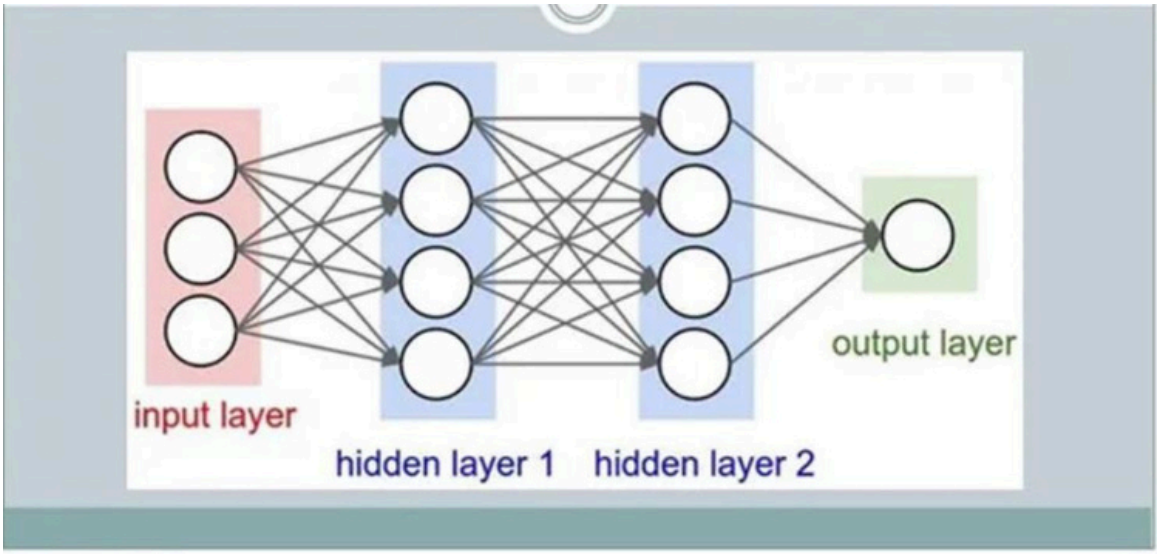
Data Partitioning

Data partitioning in ANN model is like KNN model, such as the process to split into test data and train data with the ratio of 70% of train data and 30% of test data to make comparison of two data set performance on ANN model. It can easily show the difference between two data sets.

```
> prop.table(table(creditCards$default.payment.next.month))

     0      1
0.7788 0.2212
> prop.table(table(creditCards.train$default.payment.next.month))

        0         1
0.7791429 0.2208571
> prop.table(table(creditCards.test$default.payment.next.month))

    0     1
0.778 0.222
```

Figure 30: Data Partition Result

Model Building an Evaluation



Picture 1: Examples of ANN Model Evaluation

Based on the figure, it's an example of ANN model to show case the details of ANN model analytics process, such as it will contain multiple components of input for analytics and implement multiple hidden layers of analytics to provide the output of the model, such as credit card approval analytic may involve the age, income statement and so on to set as input data and based on the input data to provide the maximum amount that are approval for credit card application.

After that, evaluation of ANN model will include accuracy, recall and precision value to measure the model performance's evaluation for better decision making on credit card approval. (Ali, 2019)

Output of ANN model

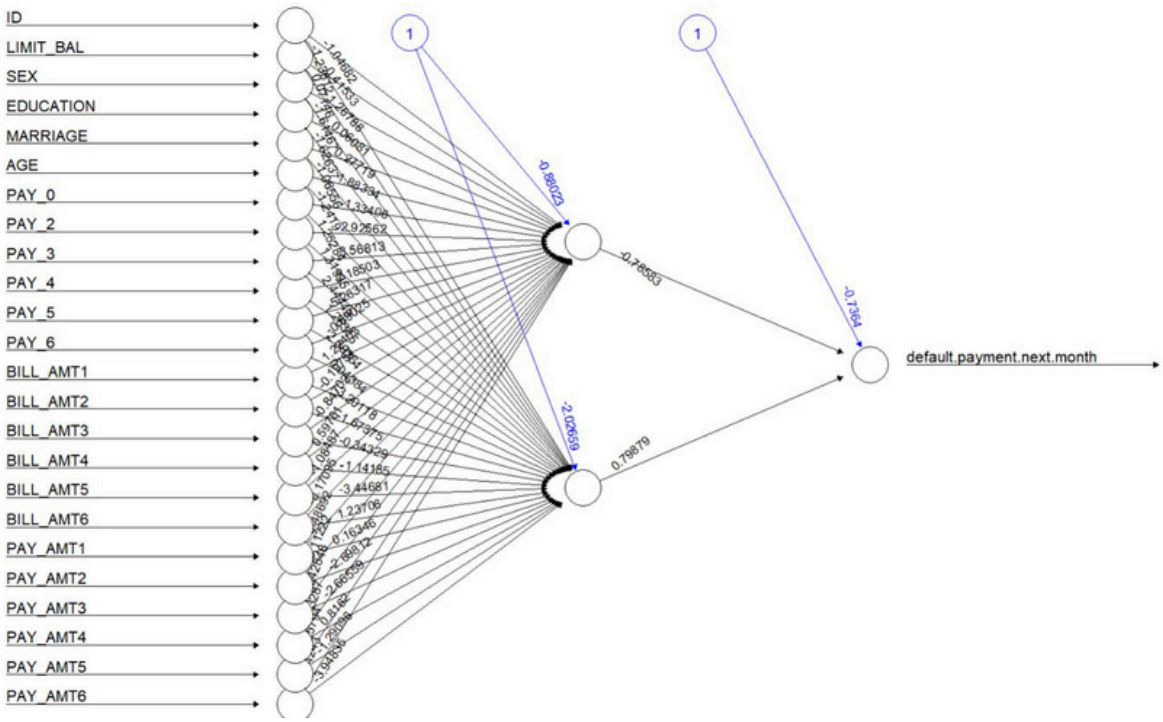


Figure 31: Output of ANN model

Evaluation of Model Performance

Through the employment of different methods, the best performing model can be selected for future predictions.

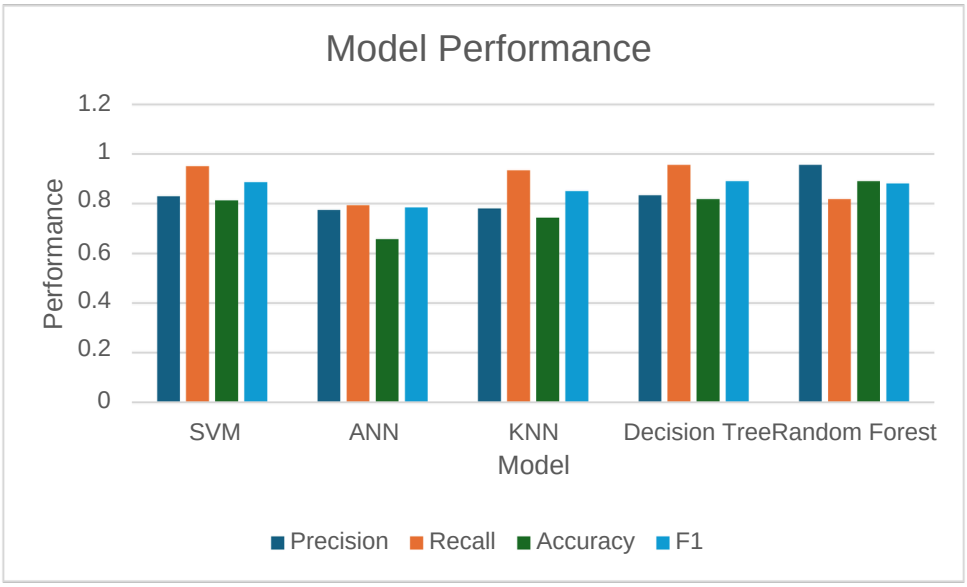


Chart 6. Evaluation chart for model performance.

The SVM model demonstrates high recall (0.9519), indicating its strong ability to identify most defaults, though its lower precision (0.8308) compared to the Random Forest and Decision Tree models results in more false positives. Its accuracy is moderate at 0.8132. The ANN model has both lower precision (0.7754) and recall (0.7945) compared to the other models, along with the lowest accuracy (0.6572), making it the least favorable choice. The KNN model has decent recall (0.9345) but lower precision (0.7814) and moderate accuracy (0.7442). The Decision Tree model features high recall (0.9568) and moderate precision (0.8344), with good accuracy (0.8184), indicating it effectively identifies most defaults but with a higher rate of false positives. Finally, the Random Forest model achieves the highest precision (0.9568) and accuracy (0.8914), though its lower recall (0.8184) means it misses more actual defaults. This model excels at correctly predicting defaults with fewer false positives, but it does not capture as many true defaults as the models with higher recall.

Model	SVM	ANN	KNN	Decision Tree	Random Forest
Precision	0.8308	0.7754	0.7814	0.8344	0.9568
Recall	0.9519	0.7945	0.9345	0.9568	0.8184
Accuracy	0.8132	0.6572	0.7442	0.8184	0.8914
F1	0.88724	0.78483	0.85112	0.89142	0.8822

Table 5 Evaluation of performance metrics

To balance precision and recall, the F1 score was calculated for each model (Table 1). The Decision Tree model has the highest F1 score (0.8915) and high recall, making it the best option for balancing the capture of defaults and minimising false positives. The Random Forest model has high precision and accuracy, with a slightly lower F1 score (0.8825), making it effective at minimizing false positives. The SVM model also performs well with a high F1 score (0.8864). Given these considerations, the Decision Tree model stands out as the best choice due to its high F1 score, indicating a strong balance between precision and recall, along with overall good accuracy.

References

Aalto University. (2021, May 13). *Low level of education the clearest common feature behind payment defaults*. Phys.org. https://phys.org/news/2021-05-clearest-common-feature-payment-defaults.html

Ali, A. (2019, December 7). *Artificial Neural Network (ANN)*. Wavy AI Research Foundation. https://medium.com/machine-learning-researcher/artificial-neural-network-ann-4481fa33d85a

baeldung. (2020, July 30). *Normalizing Inputs of Neural Networks | Baeldung on Computer Science*. Www.baeldung.com. https://www.baeldung.com/cs/normalizing-inputs-artificial-neural-network

Christopher, A. (2021, February 3). *K-Nearest Neighbor*. Medium. https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4

GeeksforGeeks. (2018, May 15). *ML | Overview of Data Cleaning*. GeeksforGeeks. https://www.geeksforgeeks.org/data-cleansing-introduction/

IBM. (2024, May 18). *IBM*. Retrieved from IBM.com: https://www.ibm.com/topics/random-forest

Introduction to ANN | Set 4 (Network Architectures) - GeeksforGeeks. (2018, July 17). GeeksforGeeks. https://www.geeksforgeeks.org/introduction-to-ann-set-4-network-architectures/

Islam, M. S., Zhou, L., & Li, F. (2009). Application of artificial intelligence (artificial neural network) to assess credit risk: a predictive model for credit card scoring.

k nearest neighbour - Why do you need to scale data in KNN. (n.d.). Cross Validated. https://stats.stackexchange.com/questions/287425/why-do-you-need-to-scale-data-in-knn

R, P. T. (2015). A Comparative Study on Decision Tree and. *International Journal of Advanced Research in Computer and Communication Engineering*, 196-199.

UCI Machine Learning Repository (2016). *Default of credit card clients dataset*. UCI Machine Learning Repository. https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients

Wah, Y. B., & Ibrahim, I. R. (2010, November). Using data mining predictive models to classify credit card applicants. In*2010 6th International Conference on Advanced Information Management and Service (IMS)* (pp. 394-398). IEEE.

Appendix

Appendix A

R codes for Cleaning Data

```
#Missing values
sum(is.na(credit))

#Check duplicate
sum(duplicated(credit))

#Check if there is any wrong value in potential columns
unique(credit$LIMIT_BAL)
unique(credit$SEX)
unique(credit$EDUCATION) #There are unknown values such as 0,5,6
unique(credit$MARRIAGE) #There are unknown value such as 0
unique(credit$AGE)

unique(credit$PAY_0)#There are unknown value such as 0,-2
unique(credit$PAY_2)#There are unknown value such as 0,-2
unique(credit$PAY_3)#There are unknown value such as 0,-2
unique(credit$PAY_4)#There are unknown value such as 0,-2
unique(credit$PAY_5)#There are unknown value such as 0,-2
unique(credit$PAY_6)#There are unknown value such as 0,-2
unique(credit$`default payment next month`)

#Replace values of 0,5,6 into 4 as "other"
credit$EDUCATION[credit$EDUCATION == 0] <- 4
credit$EDUCATION[credit$EDUCATION == 5] <- 4
credit$EDUCATION[credit$EDUCATION == 6] <- 4
unique(credit$EDUCATION) #Check values

#Replace values of 0 into 3 as "other"
credit$MARRIAGE[credit$MARRIAGE == 0] <- 3
unique(credit$MARRIAGE) #check values
```

```
#Replace values of 0 into 3 as "other"
credit$MARRIAGE[credit$MARRIAGE == 0] <- 3
unique(credit$MARRIAGE) #check values

#Replace values of 0 to 1 and -2 to -1:
credit$PAY_0[credit$PAY_0 == 0] <- 1
credit$PAY_0[credit$PAY_0 == -2] <- -1

credit$PAY_2[credit$PAY_2 == 0] <- 1
credit$PAY_2[credit$PAY_2 == -2] <- -1

credit$PAY_3[credit$PAY_3 == 0] <- 1
credit$PAY_3[credit$PAY_3 == -2] <- -1

credit$PAY_4[credit$PAY_4 == 0] <- 1
credit$PAY_4[credit$PAY_4 == -2] <- -1

credit$PAY_5[credit$PAY_5 == 0] <- 1
credit$PAY_5[credit$PAY_5 == -2] <- -1

credit$PAY_6[credit$PAY_6 == 0] <- 1
credit$PAY_6[credit$PAY_6 == -2] <- -1
```

```r
#Check values
unique(credit$PAY_0)
unique(credit$PAY_2)
unique(credit$PAY_3)
unique(credit$PAY_4)
unique(credit$PAY_5)
unique(credit$PAY_6)
str(credit)

#Replace data types to factors
credit$SEX <- as.factor(credit$SEX)
credit$EDUCATION <- as.factor(credit$EDUCATION)
credit$MARRIAGE <- as.factor(credit$MARRIAGE)
credit$PAY_0 <- as.factor(credit$PAY_0)
credit$PAY_2 <- as.factor(credit$PAY_2)
credit$PAY_3 <- as.factor(credit$PAY_3)
credit$PAY_4 <- as.factor(credit$PAY_4)
credit$PAY_5 <- as.factor(credit$PAY_5)
credit$PAY_6 <- as.factor(credit$PAY_6)
credit$`default.payment.next.month` <- as.factor(credit$`default.payment.next.month`)
str(credit)
summary(credit)

##End Cleaning
```

R codes for Data Visualisation

```r
#Explore data
summary(credit)
library(ggplot2)

#Chart 1: Histogram Cahrt of Borrower's Age
plot_5 <- ggplot(credit, aes(x = AGE)) +
  geom_histogram(bins = 30, fill = "pink", color = "black")
plot_5

#Chart 2: Default payment status in different genders
plot_9 <-ggplot(credit, mapping = aes(x = SEX, fill = `default.payment.next.month`)) +
  geom_bar() +
  ggtitle("SEX") +
  stat_count(aes(label = ..count..), geom = "label")
plot_9
##Figure 17: Chi-squared test of independence between 'SEX' and 'default payment next month'
contingency_table <- table(credit$SEX, credit$default.payment.next.month)
chi_squared_test <- chisq.test(contingency_table)
print(chi_squared_test)

#Chart 3: Default payment in different educational levels
plot_11 <-ggplot(credit, mapping = aes(x = EDUCATION, fill = `default.payment.next.month`)) +
  geom_bar() +
  ggtitle("EDUCATION") +
  stat_count(aes(label = ..count..), geom = "label")
plot_11
## Figure 18: Chi-squared test of independence between 'EDUCATION' and 'default.payment.next.month'
contingency_table_1 <- table(credit$EDUCATION, credit$default.payment.next.month)
chi_squared_test <- chisq.test(contingency_table_1)
print(chi_squared_test)

# Chart 5: Box plot of bill amount in July 2005 and default payment status
plot_10 <- ggplot(credit, aes(y = `default.payment.next.month`, x = BILL_AMT3)) +geom_boxplot()
plot_10
```

Calculation of Default Rates

Gender	Default Rate
Male	$=\frac{Number\ of\ Defaults\ in\ Male}{Total\ Number\ of\ Male} \times 100 = \frac{2,873}{11,885} \times 100 = 24.17\%$
Female	$=\frac{Number\ of\ Defaults\ in\ Female}{Total\ Number\ of\ Female} \times 100 = \frac{3,763}{18,115} \times 100 = 20.77\%$

Table A. Calculation of default rates in two genders

Education Levels	Default Rate
Graduate School	$\frac{Number\ of\ Defaults\ in\ Graduate\ School\ Category}{Total\ Number\ of\ Borrowers\ with\ Graduate\ School\ level} \times 100 = \frac{2,036}{10,585} \times 100 = 19.23\%$
University	$\frac{Number\ of\ Defaults\ in\ University\ Category}{Total\ Number\ of\ Borrowers\ with\ University\ level} \times 100 = \frac{3,330}{14,035} \times 100 = 23.73\%$
High School	$\frac{Number\ of\ Defaults\ in\ High\ School\ Category}{Total\ Number\ of\ Borrowers\ with\ High\ School\ level} \times 100 = \frac{1,237}{4,917} \times 100 = 25.16\%$
Others	$\frac{Number\ of\ Defaults\ in\ Others\ Category}{Total\ Number\ of\ Borrowers\ with\ Others\ level} \times 100 = \frac{33}{468} \times 100 = 7.05\%$

Table B. Calculation of default rates across education levels

Appendix B

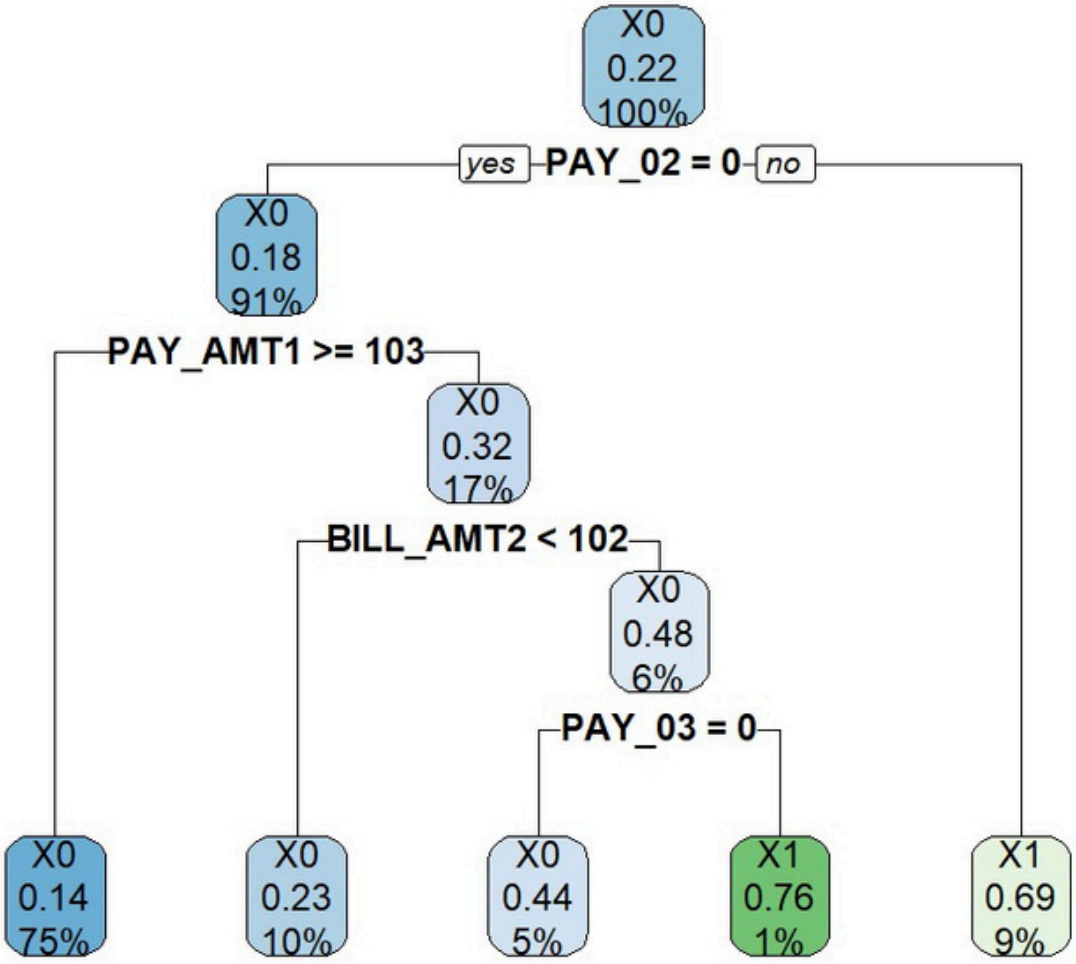
SVM

```
## SVM
#Packages for SVM
install.packages("e1071")
library(e1071)
data(creditCards)
attach(creditCards)

## Normalization of cleaned data
nor <-function(x) { (x -min(x))/(max(x)-min(x))   }
creditCards_norm <- as.data.frame(lapply(creditCards, nor))
summary(creditCards_norm)

##Split train and test data
install.packages("caret")
library(caret)
# Using data partitioning
indexes <- createDataPartition(creditCards$default.payment.next.month,times = 1,p = 0.7,list = FALSE)
creditCards.train <- creditCards[indexes,]
creditCards.test <- creditCards[-indexes,]

# Check for proportion of labels in both training and test split
prop.table(table(creditCards$default.payment.next.month))
prop.table(table(creditCards.train$default.payment.next.month))
prop.table(table(creditCards.test$default.payment.next.month))

# Fit decision model to training set
model <- svm(default.payment.next.month ~ ., data = creditCards.train)
print(model)
summary(model)

# Predict on the test set
pred.test <- predict(model, creditCards.test)
predictions<-ifelse(pred.test>0.5, 1, 0)

# Check accuracy on the test set
table(predictions, creditCards.test$default.payment.next.month)

## Evaluation
# Evaluation for performance using precision, Recall and Accuracy
conf_matrix <- confusionMatrix(factor(predictions), factor(creditCards.test$default.payment.next.month))
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
accuracy <- conf_matrix$overall["Accuracy"]
print(paste("Precision:", precision))
print(paste("Recall:", recall))
print(paste("Accuracy:", accuracy))
##End
```

Decision Tree

```r
# Load libraries
library(caret)
library(rpart)
library(rpart.plot)

set.seed(123)
#Create a stratified split
trainIndex <- createDataPartition(credit$default.payment.next.month, p = 0.7, list = FALSE)
train_data <- credit[trainIndex, ]
test_data <- credit[-trainIndex, ]
#Verify the split
table(train_data$default.payment.next.month) / nrow(train_data)
table(test_data$default.payment.next.month) / nrow(test_data)
#set up cross-validation
fitControl <- trainControl(method = "cv", number = 10, savePredictions = "final", classProbs = TRUE)
#Adjust factor levels to valid R variable names
credit$default.payment.next.month <- as.factor(credit$default.payment.next.month)
levels(credit$default.payment.next.month) <- make.names(levels(credit$default.payment.next.month))

train_data$default.payment.next.month <- as.factor(train_data$default.payment.next.month)
levels(train_data$default.payment.next.month) <- make.names(levels(train_data$default.payment.next.month))

test_data$default.payment.next.month <- as.factor(test_data$default.payment.next.month)
levels(test_data$default.payment.next.month) <- make.names(levels(test_data$default.payment.next.month))
#Check new levels
levels(train_data$default.payment.next.month)
levels(test_data$default.payment.next.month)
#Decicion Tree Model
decision_tree_model <- train(default.payment.next.month ~ ., data = train_data,
method = "rpart", trControl = fitControl)
#Model Summary
print(decision_tree_model)
#Make predictions on the test set
predictions <- predict(decision_tree_model, test_data)
#View prediction
print(predictions)
#Generate a confusion Matrix
conf_mat <- confusionMatrix(predictions, test_data$default.payment.next.month)
#print the confusion matrix and other evaluation metrics
print(conf_mat)
accuracy <- confusion_mat$overall['Accuracy']
precision <- confusion_mat$byClass['Pos Pred Value']
recall <- confusion_mat$byClass['Sensitivity']

print(paste("Precision:", precision))
print(paste("Recall:", recall))
print(paste("Accuracy:", accuracy))
#Plot the decision tree
rpart.plot(decision_tree_model$finalModel)
#End
```

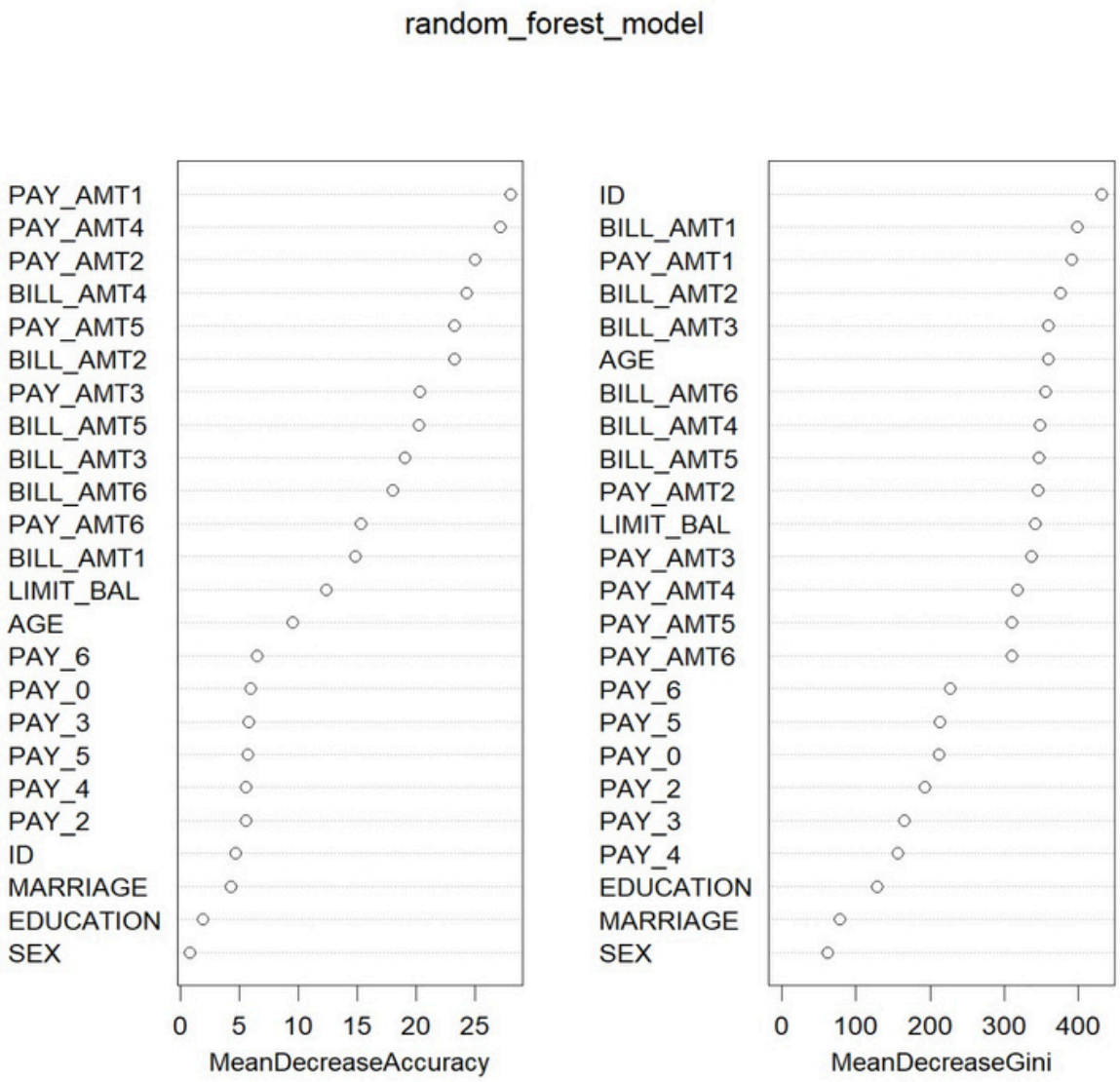
Decision Tree's Output



Random Forest

```
#Install Packages and Load Library
install.packages("randomForest")
install.packages("caret")

library(randomForest)
library(caret)

#Setting Seed
set.seed(123)

#Split data into training and testing (70% train, 30% test)
#Create a stratified split
trainIndex <- createDataPartition(credit$default.payment.next.month, p = 0.7, list = FALSE,times = 1)
train_data <- credit[trainIndex, ]
test_data <- credit[-trainIndex, ]

#Check if the split is stratified correctly
table(train_data$default.payment.next.month) / nrow(train_data)
table(test_data$default.payment.next.month) / nrow(test_data)

#Random Forest Model
random_forest_model <- randomForest(default.payment.next.month ~ ., data = train_data, ntree = 100, mtry = 3, importance = TRUE)

#Model summary
print(random_forest_model)
varImpPlot(random_forest_model)
importance(random_forest_model)
# Predicting on the test data
rf_prediction <- predict(random_forest_model, test_data)

#View the predictions
print(rf_prediction)
#Confusion Matrix and calculation of accuracy, precision, recall
confusion_mat <- confusionMatrix(rf_predictions, test_data$default.payment.next.month)
print(confusion_mat)
accuracy <- confusion_mat$overall['Accuracy']
precision <- confusion_mat$byClass['Pos Pred Value']
recall <- confusion_mat$byClass['Sensitivity']
F1 <- 2 * (precision * recall / (precision + recall))

print(paste("Precision:", precision))
print(paste("Recall:", recall))
print(paste("Accuracy:", accuracy))
print(paste("F1 Score:", F1))

print(confusion_mat$byClass)

#End
```

Random Forest’s Output



KNN

```
##the normalization function is created
nor <-function(x) { (x -min(x))/(max(x)-min(x))   }

creditCards_norm <- as.data.frame(lapply(creditCards, nor))
summary(creditCards_norm)

library(caret)
# Using data partitioning
indexes <- createDataPartition(creditCards$default.payment.next.month,times = 1,p = 0.7,list = FALSE)
creditCards.train <- creditCards[indexes,]
creditCards.test <- creditCards[-indexes,]

# Check for proportion of labels in both training and test split
prop.table(table(creditCards$default.payment.next.month))
prop.table(table(creditCards.train$default.payment.next.month))
prop.table(table(creditCards.test$default.payment.next.month))

library(class)  # Importing class library for knn
library(tidyverse)

model <- knn(creditCards.train, creditCards.test, cl = creditCards.train$default.payment.next.month, k=13)  # Assuming k=13, you can adjust it accordingly
print(model)
#####
test_labels <- creditCards.test$default.payment.next.month

# A (same as B)
tab <- table(model,creditCards.test$default.payment.next.month)
print(tab)
```

```
# Check accuracy on the test set
conf_matrix <- confusionMatrix(factor(model), factor(creditCards.test$default.payment.next.month))
precision <- conf_matrix$byClass["Pos Pred Value"]
recall <- conf_matrix$byClass["Sensitivity"]
accuracy <- conf_matrix$overall["Accuracy"]

# Print evaluation metrics
print(paste("Precision:", precision))
print(paste("Recall:", recall))
print(paste("Accuracy:", accuracy))
```

ANN

```
head(creditCards)## see the structured

##Generate a random number that is 70% of the total number of rows in dataset.
ran <- sample(1:nrow(creditCards), 0.7 * nrow(creditCards))

##the normalization function is created
nor <-function(x) { (x -min(x))/(max(x)-min(x))   }

##Run nomalization
creditCards_norm <- as.data.frame(lapply(creditCards, nor))
summary(creditCards_norm)

##Split train and test data

library(caret)
# Using data partitioning
indexes <- createDataPartition(creditCards$default.payment.next.month,times = 1,p = 0.7,list = FALSE)
creditCards.train <- creditCards[indexes,]
creditCards.test <- creditCards[-indexes,]

# Check for proportion of labels in both training and test split
prop.table(table(creditCards$default.payment.next.month))
prop.table(table(creditCards.train$default.payment.next.month))
prop.table(table(creditCards.test$default.payment.next.month))

library(neuralnet)
library(mlbench)

# fit neural network
nn=neuralnet(default.payment.next.month~ .,data=creditCards.train,
                    hidden = 2,act.fct = "logistic", linear.output = FALSE)
plot(nn)
```

```
library(class)

model <- neuralnet(default.payment.next.month ~ ., data = creditCards.train)
print(model)
summary(model)

# Predict on the test set
pred.test <- predict(model, creditCards.test)
predictions<-ifelse(pred.test>0.5,1,0)

# Check accuracy on the test set
table(predictions, creditCards.test$default.payment.next.month)


# Evaluation
library(caret)
conf_matrix <- confusionMatrix(factor(predictions), factor(creditCards.test$default.payment.next.month))
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
accuracy <- conf_matrix$overall["Accuracy"]

print(paste("Precision:", precision))
print(paste("Recall:", recall))
print(paste("Accuracy:", accuracy))
```