INF30036 Business Analytics and Artificial Intelligence

Assignment 3

Credit Card Approval prediction

Executive Summary

The project has successfully developed a predictive model utilizing advanced machine learning techniques—Decision Trees, Random Forests, k-nearest Neighbors, Support Vector Machines, and Logistic Regression—to enhance a leading bank's credit card approval process. Aimed at optimizing creditworthiness assessments, the model significantly improves operational efficiency, reduces financial risks, and enhances customer satisfaction by speeding up the application process and ensuring transparency. Notable achievements include effective risk management through proactive default detection, automation of decision-making processes that reduce manual workload and expedite service delivery, and adherence to strict regulatory standards like the EU GDPR, ensuring compliance and ethical integrity. The model also generates valuable business insights, enabling targeted marketing strategies and product customization that drive revenue growth and competitive advantage. Recommended future enhancements include continual model updates with new data and integrating emerging technologies to maintain its effectiveness. This initiative positions the bank as a leader in innovation, aligning with strategic goals to enhance customer service and achieve sustainable growth.

Table of Contents

Introduction

In the evolving financial services landscape, the ability to accurately predict credit card eligibility through advanced analytics represents a significant competitive and operational advantage. This project aims to develop a predictive model that leverages machine learning techniques to determine the suitability of applicants for credit cards. The urgency and relevance of this task are underscored by consumers' increasing reliance on credit facilities and the corresponding need for banks to mitigate risks associated with credit offerings.

The predictive model developed in this study seeks to enhance the bank's decision-making process, refine customer segmentation, and optimise risk assessment strategies, ultimately leading to improved customer service and business efficiency. By integrating sophisticated data analysis techniques, including Decision Trees, Random Forests, k-nearest Neighbors (kNN), Support Vector Machines (SVM), and Logistic Regression, this project not only aims to predict customer behaviour but also to provide insights into the factors influencing credit card defaults. These insights are crucial for the bank to effectively tailor its products and services, ensuring both customer satisfaction and business sustainability.

Moreover, the model's development process adheres to stringent data quality checks and regulatory standards, ensuring that the predictions are both accurate and compliant with global data protection regulations, such as the EU General Data Protection Regulation (GDPR). This report documents the comprehensive steps taken from data preprocessing, exploratory analysis, model building, and rigorous evaluation to ensure that the outcomes meet and exceed business optimisation's foundational requirements.

The subsequent sections will detail the methodologies employed in defining business objectives, data preparation, model building, and evaluation, demonstrating how each step enhances the bank's analytical capabilities to effectively predict credit card approval outcomes.

1. Defining Business Objectives

The primary objective of this project, developing a robust classification model for predicting credit card eligibility, directly aligns with and contributes to each facet of Business Process Optimization, as illustrated in Figure 1 (Aldoseri et al., 2023):

Figure 1: Business Process Optimisation Benefits

Efficiency and Productivity: By automating credit evaluations, the predictive model significantly enhances operational efficiency. This automation reduces the manual workload, allowing staff to allocate more time to strategic tasks, thus boosting overall productivity (Aldoseri et al., 2023). Such efficiency is crucial in scaling operations without proportionately increasing resource allocation.

Cost Savings: The model reduces costs by minimising credit defaults through accurate risk assessments (Oreski et al., 2012). Predictive analytics enable financial institutions to identify risk factors early, potentially saving significant amounts in loss mitigation (Oreski et al., 2012). This preemptive approach is essential for maintaining fiscal health and operational sustainability.

Scalability: Integrating real-time financial data ensures that the model adapts to changing economic conditions and customer behaviours, enhancing its scalability (Farayola, 2024). This flexibility allows the bank to maintain accuracy in its credit assessments, irrespective of market dynamics, ensuring long-term relevance and utility.

Improve Customer Satisfaction: Faster processing and accurate decision-making improve customer satisfaction by reducing wait times and increasing transparency, critical factors in customer retention and acquisition (Faed et al., 2013). A positive customer experience directly correlates with higher loyalty and increased business opportunities.

Consistency and Quality: Adherence to regulatory standards such as the EU General Data Protection Regulation (GDPR) ensures that all credit evaluations are consistent and highly quality, maintaining strict compliance and ethical integrity in automated decisions (Pokholkova et al., 2024). This protects the bank legally and boosts customer trust in its processes.

Data Analysis: The model's core involves rigorous data analysis, providing insights beyond simple creditworthiness assessments to include behaviour pattern analysis and predictive customer modelling (Aldoseri et al., 2023). These insights can inform targeted marketing strategies and product development, aligning with business objectives.

Enhanced Security: Effective risk management through predictive modelling enhances security by identifying and mitigating potential threats before they materialise (Sakthiswaran Rangaraju, 2023). This proactive approach protects the bank's assets and clients' interests.

Simplify Operations: Automating the credit evaluation process reduces the complexities associated with manual processes, leading to fewer errors and a streamlined operational flow (Aldoseri et al., 2023). This simplification is vital for maintaining operational agility and responsiveness.

By thoroughly addressing these aspects, the project meets and exceeds the foundational requirements of business optimisation, offering a comprehensive tool that impacts various facets of banking operations. Implementing this model significantly improves operational efficiency, customer satisfaction, risk management, and business agility.

2. Exploring Data

LINK TO DATASET:

https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients

2.1. Original Dataset Overview

First, the original dataset was brought into the environment for a quick inspection by clicking "Import Dataset", and the first row in the dataset had already been excluded by using Excel.

```
> cccard <- data
> View(cccard)
> str(cccard)
Classes 'tbl_df', 'tbl' and 'data.frame':       30000 obs. of  25 variables:
 $ ID                        : num  1 2 3 4 5 6 7 8 9 10 ...
 $ LIMIT_BAL                 : num  20000 120000 90000 50000 50000 50000 500000 100000 140000 20000 ...
 $ SEX                       : num  2 2 2 2 1 1 1 2 2 1 ...
 $ EDUCATION                 : num  2 2 2 2 2 1 1 2 3 3 ...
 $ MARRIAGE                  : num  1 2 2 1 1 2 2 2 1 2 ...
 $ AGE                       : num  24 26 34 37 57 37 29 23 28 35 ...
 $ PAY_0                     : num  2 -1 0 0 -1 0 0 0 0 -2 ...
 $ PAY_2                     : num  2 2 0 0 0 0 0 -1 0 -2 ...
 $ PAY_3                     : num  -1 0 0 0 -1 0 0 -1 2 -2 ...
 $ PAY_4                     : num  -1 0 0 0 0 0 0 0 0 -2 ...
 $ PAY_5                     : num  -2 0 0 0 0 0 0 0 0 -1 ...
 $ PAY_6                     : num  -2 2 0 0 0 0 0 -1 0 -1 ...
 $ BILL_AMT1                 : num  3913 2682 29239 46990 8617 ...
 $ BILL_AMT2                 : num  3102 1725 14027 48233 5670 ...
 $ BILL_AMT3                 : num  689 2682 13559 49291 35835 ...
 $ BILL_AMT4                 : num  0 3272 14331 28314 20940 ...
 $ BILL_AMT5                 : num  0 3455 14948 28959 19146 ...
 $ BILL_AMT6                 : num  0 3261 15549 29547 19131 ...
 $ PAY_AMT1                  : num  0 0 1518 2000 2000 ...
 $ PAY_AMT2                  : num  689 1000 1500 2019 36681 ...
 $ PAY_AMT3                  : num  0 1000 1000 1200 10000 657 38000 0 432 0 ...
 $ PAY_AMT4                  : num  0 1000 1000 1100 9000 ...
 $ PAY_AMT5                  : num  0 0 1000 1069 689 ...
 $ PAY_AMT6                  : num  0 2000 5000 1000 679 ...
 $ defaultpaymentnextmonth: num  1 1 0 0 0 0 0 0 0 0 ...
```

Figure 2: Data Importation

```
> ## Transform categorical columns to factor data type
> cccard$SEX <- as.factor(as.character(cccard$SEX))
>
> cccard$EDUCATION <- as.factor(as.character(cccard$EDUCATION))
>
> cccard$MARRIAGE <- as.factor(as.character(cccard$MARRIAGE))
>
> cccard$PAY_0 <- as.factor(as.character(cccard$PAY_0))
> cccard$PAY_2 <- as.factor(as.character(cccard$PAY_2))
> cccard$PAY_3 <- as.factor(as.character(cccard$PAY_3))
> cccard$PAY_4 <- as.factor(as.character(cccard$PAY_4))
> cccard$PAY_5 <- as.factor(as.character(cccard$PAY_5))
> cccard$PAY_6 <- as.factor(as.character(cccard$PAY_6))
> cccard$defaultpaymentnextmonth <- as.factor(as.character(cccard$defaultpaymentnextmonth))
>
> str(cccard)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':       30000 obs. of  25 variables:
 $ ID                        : num  1 2 3 4 5 6 7 8 9 10 ...
 $ LIMIT_BAL                 : num  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                       : num  24 26 34 37 57 37 29 23 28 35 ...
 $ PAY_0                     : Factor w/ 11 levels "-1","-2","0",..: 5 1 3 3 1 3 3 3 3 2 ...
 $ PAY_2                     : Factor w/ 11 levels "-1","-2","0",..: 5 5 3 3 3 3 3 1 3 2 ...
 $ PAY_3                     : Factor w/ 11 levels "-1","-2","0",..: 1 3 3 3 1 3 3 1 5 2 ...
 $ PAY_4                     : Factor w/ 11 levels "-1","-2","0",..: 1 3 3 3 3 3 3 3 3 2 ...
 $ PAY_5                     : Factor w/ 10 levels "-1","-2","0",..: 2 3 3 3 3 3 3 3 3 1 ...
 $ PAY_6                     : Factor w/ 10 levels "-1","-2","0",..: 2 4 3 3 3 3 3 1 3 1 ...
 $ BILL_AMT1                 : num  3913 2682 29239 46990 8617 ...
 $ BILL_AMT2                 : num  3102 1725 14027 48233 5670 ...
 $ BILL_AMT3                 : num  689 2682 13559 49291 35835 ...
 $ BILL_AMT4                 : num  0 3272 14331 28314 20940 ...
 $ BILL_AMT5                 : num  0 3455 14948 28959 19146 ...
 $ BILL_AMT6                 : num  0 3261 15549 29547 19131 ...
 $ PAY_AMT1                  : num  0 0 1518 2000 2000 ...
 $ PAY_AMT2                  : num  689 1000 1500 2019 36681 ...
 $ PAY_AMT3                  : num  0 1000 1000 1200 10000 657 38000 0 432 0 ...
 $ PAY_AMT4                  : num  0 1000 1000 1100 9000 ...
 $ PAY_AMT5                  : num  0 0 1000 1069 689 ...
 $ PAY_AMT6                  : num  0 2000 5000 1000 679 ...
 $ defaultpaymentnextmonth: Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 ...
```

Figure 3: Convert to Factor

The data structure has been revised to conform to the initial variables' specification, consisting of 10 category (factor) variables, which include the goal variable "default payment next month," and 15 numerical variables.

2.2. Data quality check

According to Chai (2020), data scientists dedicate a significant portion of their time, ranging from 50% to 80%, to the task of cleaning and organizing data. This highlights the acknowledged significance of data quality and data cleansing, as recognized by Van den Broeck et al. (2005). In order to guarantee the quality of the data before conducting any analysis, the dimensions of

data quality and their respective definitions are carefully assessed and chosen based on their appropriateness to the dataset's context and the specific needs of the analysis.

This report utilizes the six dimensions of data quality assessment proposed by the Centers for Disease Control and Prevention (CDC) - completeness, uniqueness, timeliness, validity, accuracy, and consistency. These dimensions have been widely discussed and defined in the literature (Sidi et al., 2012). Assuming that the dataset is timely, legitimate, and consistent, just its consistency, completeness, and uniqueness will be examined.

2.2.1. Consistency

Based on the first examination and the data definition table, it is noted that the unidentified data in the dataset is indicated by the values "0, -2" in the variables *Pay_0* to *Pay_6*, as well as "0, 5, 6" values in *EDUCATION*, and "0" in MARRIAGE. The lack of consistency in the encoding of unidentified values has led to the need for a more standardized encoding. As a result, these missing values are now encoded as follows: -1 in Pay_0 to Pay_6, 3 in MARRIAGE, and 4 in EDUCATION.

```
> indices <- which(cccard$PAY_0 == 0 | cccard$PAY_0 == -2)
> cccard$PAY_0[indices] <- -1
> indices <- which(cccard$PAY_2 == 0 | cccard$PAY_2 == -2)
> cccard$PAY_2[indices] <- -1
> indices <- which(cccard$PAY_3 == 0 | cccard$PAY_3 == -2)
> cccard$PAY_3[indices] <- -1
> indices <- which(cccard$PAY_4 == 0 | cccard$PAY_4 == -2)
> cccard$PAY_4[indices] <- -1
> indices <- which(cccard$PAY_5 == 0 | cccard$PAY_5 == -2)
> cccard$PAY_5[indices] <- -1
> indices <- which(cccard$PAY_6 == 0 | cccard$PAY_6 == -2)
> cccard$PAY_6[indices] <- -1
> indices <- which(cccard$EDUCATION == 5 | cccard$EDUCATION == 6 | cccard$EDUCATION == 0)
> cccard$EDUCATION[indices] <- 4
> indices <- which(cccard$MARRIAGE == 0)
> cccard$MARRIAGE[indices] <- 3
```

Figure 4: Re-Encoding

```
> summary(cccard)
       ID          LIMIT_BAL          SEX          EDUCATION       MARRIAGE          AGE             PAY_0
 Min.   :    1   Min.   :  10000   1:11888   1:10585   1:13659   Min.   :21.00   -1     :23182
 1st Qu.: 7501   1st Qu.:  50000   2:18112   2:14030   2:15964   1st Qu.:28.00   1      : 3688
 Median :15000   Median : 140000             3: 4917   3:  377   Median :34.00   2      : 2667
 Mean   :15000   Mean   : 167484             4:  468             Mean   :35.49   3      :  322
 3rd Qu.:22500   3rd Qu.: 240000                                 3rd Qu.:41.00   4      :   76
 Max.   :30000   Max.   :1000000                                 Max.   :79.00   5      :   26
                                                                                 (Other):   39
     PAY_2           PAY_3           PAY_4           PAY_5           PAY_6         BILL_AMT1
 -1     :25562   -1     :25787   -1     :26490   -1     :27032   -1     :26921   Min.   :-165580
 2      : 3927   2      : 3819   2      : 3159   2      : 2626   2      : 2766   1st Qu.:   3559
 3      :  326   3      :  240   3      :  180   3      :  178   3      :  184   Median :  22382
 4      :   99   4      :   76   4      :   69   4      :   84   4      :   49   Mean   :  51223
 1      :   28   7      :   27   7      :   58   7      :   58   7      :   46   3rd Qu.:  67091
 5      :   25   6      :   23   5      :   35   5      :   17   6      :   19   Max.   : 964511
 (Other):   33   (Other):   28   (Other):    9   (Other):    5   (Other):   15
   BILL_AMT2         BILL_AMT3         BILL_AMT4         BILL_AMT5         BILL_AMT6
 Min.   :-69777   Min.   :-157264   Min.   :-170000   Min.   :-81334   Min.   :-339603
 1st Qu.:  2985   1st Qu.:   2666   1st Qu.:   2327   1st Qu.:  1763   1st Qu.:   1256
 Median : 21200   Median :  20089   Median :  19052   Median : 18105   Median :  17071
 Mean   : 49179   Mean   :  47013   Mean   :  43263   Mean   : 40311   Mean   :  38872
 3rd Qu.: 64006   3rd Qu.:  60165   3rd Qu.:  54506   3rd Qu.: 50191   3rd Qu.:  49198
 Max.   :983931   Max.   :1664089   Max.   : 891586   Max.   :927171   Max.   : 961664

    PAY_AMT1          PAY_AMT2          PAY_AMT3          PAY_AMT4          PAY_AMT5
 Min.   :     0   Min.   :      0   Min.   :     0   Min.   :     0   Min.   :     0.0
 1st Qu.:  1000   1st Qu.:    833   1st Qu.:   390   1st Qu.:   296   1st Qu.:   252.5
 Median :  2100   Median :   2009   Median :  1800   Median :  1500   Median :  1500.0
 Mean   :  5664   Mean   :   5921   Mean   :  5226   Mean   :  4826   Mean   :  4799.4
 3rd Qu.:  5006   3rd Qu.:   5000   3rd Qu.:  4505   3rd Qu.:  4013   3rd Qu.:  4031.5
 Max.   :873552   Max.   :1684259   Max.   :896040   Max.   :621000   Max.   :426529.0

    PAY_AMT6        default payment next month
 Min.   :     0.0   0:23364
 1st Qu.:   117.8   1: 6636
 Median :  1500.0
 Mean   :  5215.5
 3rd Qu.:  4000.0
 Max.   :528666.0
```

Figure 5: Re-encoded data Summary

2.2.2. Completeness

The number of the unidentified values for each variable is documented in the table below:

Table 1: Unidentified Values

Variable	Role	Type	Description	Unidentified Value
ID	ID	Integer	ID of each client.	0
LIMIT_BAL	Feature	Integer	Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.	0
SEX	Feature	Integer	Gender (1=male, 2=female).	0

EDUCATION	Feature	Integer	(1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown).	331
MARRIAGE	Feature	Integer	Marital status (1=married, 2=single, 3=others).	54
AGE	Feature	Integer	Age in years.	0
PAY_0	Feature	Integer	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).	17496
PAY_2	Feature	Integer	Repayment status in August, 2005 (scale same as above).	19512
PAY_3	Feature	Integer	Repayment status in July, 2005 (scale same as above)	19849
PAY_4	Feature	Integer	Repayment status in June, 2005 (scale same as above)	20803
PAY_5	Feature	Integer	Repayment status in May, 2005 (scale same as above)	21493
PAY_6	Feature	Integer	Repayment status in April, 2005 (scale same as above)	21181
BILL_AMT1	Feature	Integer	Amount of bill statement in September, 2005 (NT dollar).	0
BILL_AMT2	Feature	Integer	Amount of bill statement in August, 2005 (NT dollar)	0
BILL_AMT3	Feature	Integer	Amount of bill statement in July, 2005 (NT dollar)	0
BILL_AMT4	Feature	Integer	Amount of bill statement in June, 2005 (NT dollar)	0

BILL_AMT5	Feature	Integer	Amount of bill statement in May, 2005 (NT dollar)	0
BILL_AMT6	Feature	Integer	Amount of bill statement in April, 2005 (NT dollar)	0
PAY_AMT1	Feature	Integer	Amount of previous payment in September, 2005 (NT dollar)	0
PAY_AMT2	Feature	Integer	Amount of previous payment in August, 2005 (NT dollar)	0
PAY_AMT3	Feature	Integer	Amount of previous payment in July, 2005 (NT dollar)	0
PAY_AMT4	Feature	Integer	Amount of previous payment in June, 2005 (NT dollar)	0
PAY_AMT5	Feature	Integer	Amount of previous payment in May, 2005 (NT dollar)	0
PAY_AMT6	Feature	Integer	Amount of previous payment in April, 2005 (NT dollar)	0
Default payment next month	Target	Binary	Default payment (1=yes, 0=no).	0

2.2.3. Uniqueness

Uniqueness pertains to the distinctiveness of each record, ensuring that no copies exist and that any duplicates are discovered and managed accordingly. However, this dataset does not include any duplicate entries.

```
> sum(duplicated(cccard))
[1] 0
```

Figure 6: Duplicate check

2.3. Updated Data Definition Table

Variable	Role	Type	Description	Example
ID	ID	Integer	ID of each client.	150

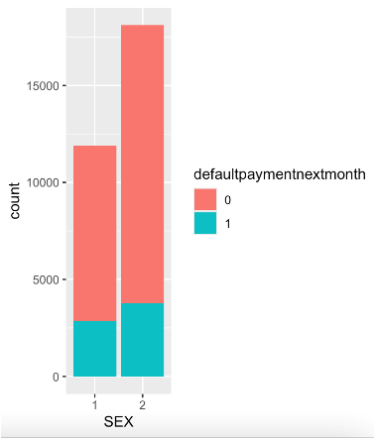
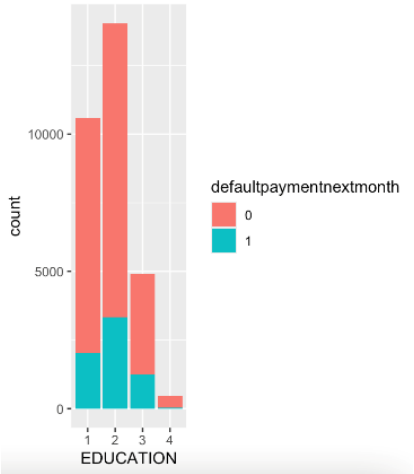
LIMIT_BAL	Predictor	Integer	Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.	180000
SEX	Predictor	Integer	Gender (1=male, 2=female).	1
EDUCATION	Predictor	Integer	(1=graduate school, 2=university, 3=high school, 4=others).	2
MARRIAGE	Predictor	Integer	Marital status (1=married, 2=single, 3=others).	1
AGE	Predictor	Integer	Age in years.	40
PAY_0	Predictor	Integer	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).	2
PAY_2	Predictor	Integer	Repayment status in August, 2005 (scale same as above).	1
PAY_3	Predictor	Integer	Repayment status in July, 2005 (scale same as above)	3
PAY_4	Predictor	Integer	Repayment status in June, 2005 (scale same as above)	2
PAY_5	Predictor	Integer	Repayment status in May, 2005 (scale same as above)	-1
PAY_6	Predictor	Integer	Repayment status in April, 2005 (scale same as above)	5
BILL_AMT1	Predictor	Integer	Amount of bill statement in September, 2005 (NT dollar).	3102
BILL_AMT2	Predictor	Integer	Amount of bill statement in August, 2005 (NT dollar)	4534

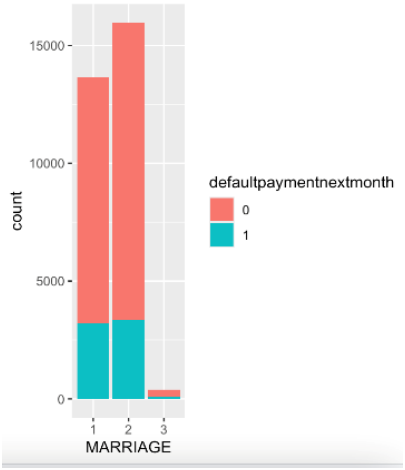
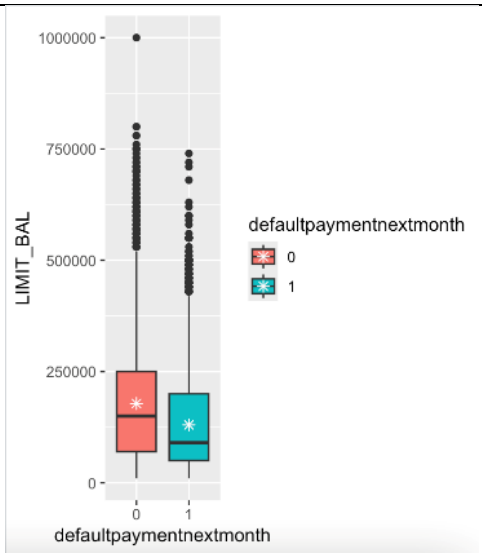
BILL_AMT3	Predictor	Integer	Amount of bill statement in July 2005 (NT dollar)	2312
BILL_AMT4	Predictor	Integer	Amount of bill statement in June 2005 (NT dollar)	1233
BILL_AMT5	Predictor	Integer	Amount of bill statement in May 2005 (NT dollar)	2344
BILL_AMT6	Predictor	Integer	Amount of bill statement in April 2005 (NT dollar)	5345
PAY_AMT1	Predictor	Integer	Amount of previous payment in September 2005 (NT dollar)	2311
PAY_AMT2	Predictor	Integer	Amount of previous payment in August 2005 (NT dollar)	3423
PAY_AMT3	Predictor	Integer	Amount of previous payment in July 2005 (NT dollar)	4355
PAY_AMT4	Predictor	Integer	Amount of previous payment in June 2005 (NT dollar)	7576
PAY_AMT5	Predictor	Integer	Amount of previous payment in May 2005 (NT dollar)	4566
PAY_AMT6	Predictor	Integer	Amount of previous payment in April 2005 (NT dollar)	2342
Default payment next month	Target	Binary	Default payment (1=yes, 0=no).	1

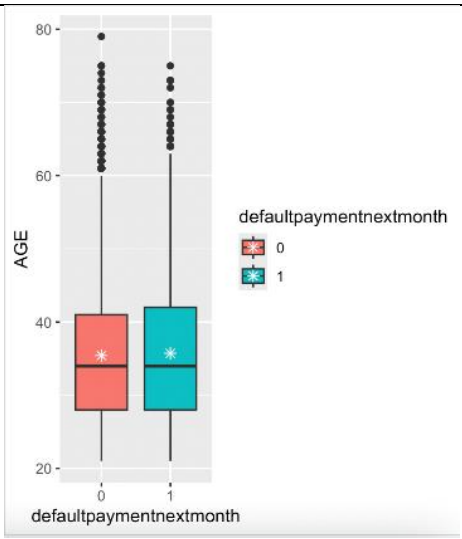
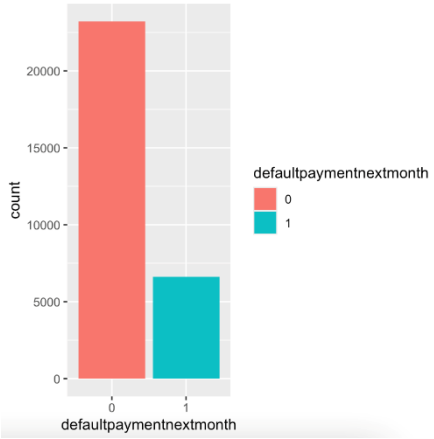
2.4. Exploratory Analysis

Exploratory Data Analysis (EDA) in business analytics is an essential process that involves the initial investigation of data to discover patterns, identify outliers, and test hypotheses. EDA plays a fundamental role in understanding the characteristics and insights of the data before applying more advanced analytics techniques such as statistical modeling (Huang et al., 2017). By exploring the data visually and statistically, EDA helps in uncovering trends, patterns, and relationships within the dataset (Ugochukwu et al., 2022). This process is

essential for gaining a better understanding of the dataset, which can lead to the elimination of errors and incorrect assumptions (Qu & Agbor, 2023).

Variable	Visualisation	Analysis
Sex	 SEX (1 = male, 2 = female) Defaultpayment (0 = no, 1 = yes)	This bar chart is produced to study the relationship between the variables SEX and default payment approval. Female records dominate the dataset, with over 18,000 compared to male's 11,800, which illustrated a female to male ratio of 3:2. While there is a higher "yes" responses for females, it is proportionally less than the sex variable. This indicates that there may be a small correlation between sex and default payment approval.
Education	 EDUCATION (1=graduate school, 2=university, 3=high school, 4=others) Defaultpayment (0 = no, 1 = yes)	This bar chart is produced to study the relationship between the variables EDUCATION and default payment approval. The majority of candidates study in graduate school and university, making up 80% of the dataset. The proportion of "yes" to each of the first three educational levels is relatively identical, indicating a possible correlation between education and default payment approval.

Marriage	 MARRIAGE (1=married, 2=single, 3=others). Defaultpayment (0 = no, 1 = yes)	This bar chart is produced to study the relationship between the variables MARRIAGE and default payment approval. The number of single records in the dataset is slightly more than married, yet the amount of "yes" responses is nearly the same for two variables. This shows that there may not be a correlation between marital status and default payment approval.
Limit_Bal	 Defaultpayment (0 = no, 1 = yes)	**No group:** • Mean: 178,099 • Median: 150,000 • Range: 990,000 • Skewness: 0.912 • Standard Deviation: 131,628 **Yes group:** • Mean: 130,109 • Median: 90,000 • Range: 730,000 • Skewness: 1.352 • Standard Deviation: 115,378 ⇨ Both groups indicated that the distribution of given credit is positively skewed. ⇨ Large standard deviation illustrated that the data is spread out sparsely.

Age	 Defaultpayment (0 = no, 1 = yes)	**No group:** • Mean: 35.41 • Median: 34 • Range: 58 • Skewness: 0.752 • Standard Deviation: 9.01 **Yes group:** • Mean: 35.73 • Median: 34 • Range: 54 • Skewness: 0.665 • Standard Deviation: 9.69
default payment nextmonth	 Defaultpayment (0 = no, 1 = yes)	This bar chart is produced to identify the number of default payment approvals in the dataset. There are over 23,300 responses with “no” while only 6,600 responses said “yes”, which indicates a major imbalance of data.

2.5. Correlation Examination

According to Akoglu (2018), correlation coefficient is a statistical measure that describes the strength and direction of a relationship between two variables. It quantifies how much one variable changes in response to changes in another variable. In his research, he mentioned the interpretation of the correlation coefficients from *The Political Science Department at Quinnipiac University* as:

Correlation Coefficient		Quinnipiac University (Politics)
+1	−1	Perfect
+0.9	−0.9	Very Strong
+0.8	−0.8	Very Strong
+0.7	−0.7	Very Strong
+0.6	−0.6	Strong
+0.5	−0.5	Strong
+0.4	−0.4	Strong
+0.3	−0.3	Moderate
+0.2	−0.2	Weak
+0.1	−0.1	Negligible
0	0	None

Figure 7: Correlation Coefficient

Based on the rankings above, we then explored the correlation coefficients of all variables in the dataset:

```
cor(ccccard_bal)
```

	defaultpaymentnextmonth
ID	-0.013832278
LIMIT_BAL	-0.154010905
SEX	-0.040560563
EDUCATION	0.033145573
MARRIAGE	-0.026799306
AGE	0.014440032
PAY_0	0.398537656
PAY_2	0.333283771
PAY_3	0.292369794
PAY_4	0.275214826
PAY_5	0.266761777
PAY_6	0.248451813
BILL_AMT1	-0.020214477
BILL_AMT2	-0.014640049
BILL_AMT3	-0.014541957
BILL_AMT4	-0.010633314
BILL_AMT5	-0.006574403
BILL_AMT6	-0.004812764
PAY_AMT1	-0.072662644
PAY_AMT2	-0.058605178
PAY_AMT3	-0.058721348
PAY_AMT4	-0.056688918
PAY_AMT5	-0.055830751
PAY_AMT6	-0.053557869
defaultpaymentnextmonth	1.000000000

Figure 8: Outlier removal

As we can see, PAY_0 has the greatest influence on target variable, which is `defaultpaymentnextmonth`, with its correlation coefficient of 0.3985, or ~4 (which is ranked at **moderate** to **strong**), followed by PAY_2, PAY_3, PAY_4, PAY_5, and PAY_6.

3. Preparing Data

3.1. Outlier removal

While inspecting the dataset, the variable "LIMIT_BAL" is highly volatile and continuous. This nature means there are many outliers and the distribution of credit values could be highly skewed. Since we are using models like Decision Tree, Random Forest, and SVM, ~~keeping~~ outliers could heavily influence our predictive analytics outcomes. Therefore, we decided to remove the outliers, using the Inter Quartile Range method.

```
#Calculate 1st, 3rd quartiles and IQR
cleanOutliers <- function(data, var_name) {
  q1 <- quantile(data[[var_name]], 0.25)
  q3 <- quantile(data[[var_name]], 0.75)
  iqr <- q3 - q1

  #Define lower and upper bounds for outlier detection
  lower_bound <- q1 - 1.5 * iqr
  upper_bound <- q3 + 1.5 * iqr

  #Identify outliers
  outliers <- data[[var_name]] < lower_bound | data[[var_name]] > upper_bound

  #Remove outliers from the original dataset
  cleaned_data <- data[!outliers, ]

  summary(cleaned_data)
  return(cleaned_data)
}

cccard <- cleanOutliers(cccard, "LIMIT_BAL")
```

Figure 9: Outlier removal

3.2. Data upscaling

As previously observed, there are 6,636 customer records responding "yes" and 23,364 records had their default payment the following month unaccepted in the target variable "defaultpaymentnextmonth", signaling the major imbalance in the dataset. With the ratio of "no" to "yes" being approximately 7:2, underfitting of data models is a possibility. Therefore, we used the ovun.sample() function to upscale the original dataset so that the number of responses are more evenly distributed.

```
target <- "defaultpaymentnextmonth"
predictors <- c("LIMIT_BAL","SEX","EDUCATION", "MARRIAGE","AGE","PAY_0","PAY_2","PAY_3","PAY_4","PAY_5","PAY_6","
BILL_AMT1",
                "BILL_AMT2","BILL_AMT3","BILL_AMT4","BILL_AMT5","BILL_AMT6","PAY_AMT1","PAY_AMT2","PAY_AMT3","PAY
_AMT4",
                "PAY_AMT5","PAY_AMT6")
formula <- as.formula(paste(target, "~", paste(predictors, collapse = "+")))
```

```
cccard_bal <- ovun.sample(formula, cccard, method = "both", p = 0.5,
                          N = 29833)$data
View(cccard_bal)
summary(cccard_bal)
```

Figure 10: Upscaling of the dataset

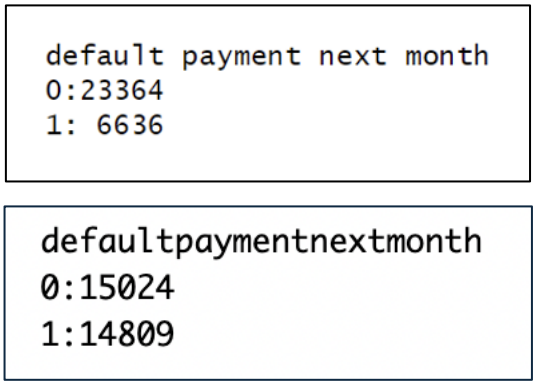


Figure 11: Number of responses for the variable defaultpaymentnextmonth before (top) and after (bottom) the upscaling.

4. Data sampling

Data sampling plays a crucial role in the field of machine learning models, as highlighted by various researchers. Bhatia et al. (2021) introduced a machine-learning-based dynamic-importance sampling method for adaptive multiscale simulations. This approach utilizes machine learning to dynamically sample the phase space explored by a macro model through microscale simulations. By doing so, it ensures a comprehensive and accurate representation of the data, ultimately improving the performance of machine learning models.

After getting the final dataset from data preparation which is cccard_bal, we created two types of datasets which were “cc_train” and “cc_test”

- set.seed(123) guarantees that each time the code is executed, the R random number generator will generate the same set of numbers.
- Use *sample()* to get a sample of the dataset. By selecting the array from 1 to nrow in “cccard_bal”, we chose randomly 70% = 0.7 (rows) of the dataset is for training. Then, we stored those 70% to a data frame trainIndex
- cc_train refers to a data frame that containts only rows from trainIndex

- cc_test refers to a data frame that contains the rest of cccard_bal = 30%

  (-trainIndex denotes that all rows not in trainIndex are being selected)

```
> set.seed(123)
> trainIndex <- sample(1:nrow(cccard_bal),0.70*nrow(cccard_bal))
> cc_train <- data[trainIndex, ]
> cc_test <- data[-trainIndex, ]
```

With respect to the total number of rows and columns divided into the two new datasets, the *dim()* function shows the dimensions of each dataset.

```
> dim(cc_train)
[1] 20883    25
> dim(cc_test)
[1] 9117   25
```

Converting the target variable to a factor is a crucial step in classification tasks in R. By leveraging factor analysis techniques, researchers can enhance the accuracy and reliability of classification models, ensuring robust predictions and actionable insights (Lantz, 2019). Therefore, we used *as.factor()* function to convert the target variable which is 'defaultpaymentnextmonth' to factor in two datasets "cc_train" and "cc_test".

```
> cc_train$defaultpaymentnextmonth <- as.factor(cc_train$defaultpaymentnextmonth)
>
> cc_test$defaultpaymentnextmonth <- as.factor(cc_test$defaultpaymentnextmonth)
```

5. Building and evaluating the models

This section proposes 5 machine learning models that help predict whether there is a default payment next month for the dataset. The 5 models are: decision tree, random forest, kNN, SVM, and logistic regression.

Each model will be trained based on historical data, with its target variable is `defaultpaymentnextmonth` and predictors are the remaining variables. Once a machine algorithm of each model is learnt, they are applied to a testing dataset. To evaluate the performance of each model, metrics like Accuracy, Precision, Recall, and F1 Score will be calculated.

5.1. Decision Tree

5.1.1. Model

Often employed in data mining, decision tree methodology is a means of creating prediction algorithms for a target variable or creating classification systems based on several variables (Song & Ying, 2015). Using this technique, a population is divided into segments that resemble branches, creating an inverted tree with a root node, internal nodes, and leaf nodes. Being non-parametric, the technique may effectively handle sizable, complex datasets without requiring a complicated parametric structure (Song & Ying, 2015).

In our case, "cc_train" dataset will be used as a basis, or training dataset, to create a decision tree model of default payment prediction. The model will be created, visualized, and summarized using the `rpart` packages:

```
tree_model <- rpart(defaultpaymentnextmonth ~., data = cc_train)
```

From the line of code, we can see that:

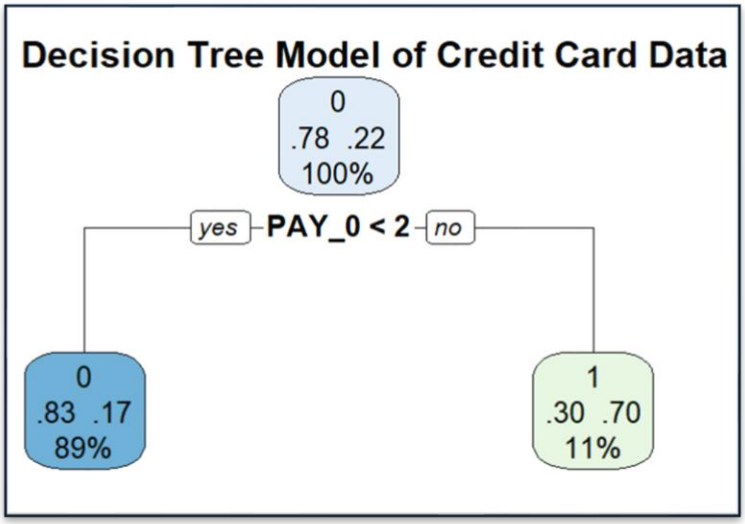
- rpart() function from the `rpart` package is used to build a decision tree model.
- The model predicts `defaultpaymentnextmonth` (whether there is a default payment next month) based on all other variables in the “cc_train” dataframe.
- The resulting decision tree model is stored in the variable `tree_model`.

The resulting model is then visualized using `rpart.plot` function:

```
rpart.plot(tree_model, extra = 104)
title(main = "Decision Tree Model of Credit Card Data")
```

(`extra=104` selects a class model with a response having more than 2 levels)

The plot:



In the graph, the root node of the model splits values using *PAY_0* as it has the strongest influence on the `defaultpaymentnextmonth` variable, with its correlation coefficient value of 0.399 (Refer to *2.5. Correlation Examination* section).

To better understand the graph, we summarize the model by using the *summary()* function:

```
summary(tree_model)
```

```
Call:
rpart(formula = defaultpaymentnextmonth ~ ., data = cc_train)
  n= 20883

         CP nsplit rel error    xerror       xstd
1 0.1911607      0 1.0000000 1.0000000 0.01290970
2 0.0100000      1 0.8088393 0.8088393 0.01192499

Variable importance
PAY_0 PAY_4 PAY_5 PAY_3 PAY_6 PAY_2
   85     4     4     3     3     2

Node number 1: 20883 observations,    complexity param=0.1911607
  predicted class=0  expected loss=0.2231959  P(node) =1
    class counts: 16222  4661
   probabilities: 0.777 0.223
```

```
Node number 2: 18676 observations
  predicted class=0  expected loss=0.166631  P(node) =0.894316
    class counts: 15564  3112
   probabilities: 0.833 0.167

Node number 3: 2207 observations
  predicted class=1  expected loss=0.2981423  P(node) =0.105684
    class counts:   658  1549
   probabilities: 0.298 0.702
```

The summary will be broken down as following:

- The decision tree was built using 20,883 observations (n = 20883).
- The complexity parameter (CP) table shows the impact of different levels of complexity on the model's error, helping to decide the optimal level of pruning.
- PAY_0 is the most important variable in the model, followed by other payment-related variables with much lower importance (refer to *2.5. Correlation Examination* section)
- For node number 1:
  - Predicted class=0: The class that the model predicts for this node.
  - The root node (node number 1) predicts class '0' with probabilities of 77.7% (16222 observation) for class 0 and 22.3% (4661 observations) for class 1.
- For node number 2:
  - P(node)=0.894316: Proportion of the entire dataset that reaches this node (89.43%).
  - class counts: 15564 (class 0), 3112 (class 1).
  - probabilities: 0.833 (class 0), 0.167 (class 1).
- For node number 3:
  - P(node)=0.105684: Proportion of the entire dataset that reaches this node (10.57%).
  - class counts: 658 (class 0), 1549 (class 1).
  - probabilities: 0.298 (class 0), 0.702 (class 1).

5.1.2. Prediction

After building the model with algorithm learnt from the training dataset, we proceeded to make predictions for the `defaultpaymentnextmonth` target variable. To make predictions, we need to apply the trained model to the "cc_test" testing dataset. The *predict()* function in the `rpart` package can help us to do this:

```
dtree.predictions <- predict(tree_model, cc_test, type = "class")
```

5.1.3. Evaluation

Once we have the predictions, we can check its performance by using a confusion matrix and evaluation metrics:

- Confusion matrix

```
actual_dt <- factor(cc_test$defaultpaymentnextmonth)
pred_dt <- factor(dtree.predictions)

confusion_matrix_dt <- table(data.frame(actual_dt, pred_dt))
confusion_matrix_dt
```

```
          pred_dt
actual_dt    0    1
        0 6847  295
        1 1347  628
```

From the matrix, we can see that: while 6847 observations of class '0' and 628 observations of class '1' were correctly predicted, 295 observations of class '0' and 1347 observations of class '1' were incorrectly predicted.

We then created a table to compare the actual vs. predicted values:

```
dtree.comparison <- cc_test
dtree.comparison$Predictions <- dtree.predictions
dtree.comparison[ , c("defaultpaymentnextmonth", "Predictions")]
```

A tibble: 9,117 × 2

defaultpaymentnextmonth <fctr>	Predictions <fctr>
0	0
0	0
1	1
0	1
1	1
0	0
1	0
1	0
0	0
1	1

951-960 of 9,117 rows Previous 1 … 94 95 96 97 98 … 100 Next

- Evaluation metrics

To calculate Accuracy, Precision, Recall, and F1 Score for the predicted model, functions like *accuracy_vec()* in the `yardstick` packages will be used:

```
dt_accuracy <- accuracy_vec(cc_test$defaultpaymentnextmonth, estimate
= as.factor(dtree.comparison$Predictions))

dt_precision <- precision_vec(truth = cc_test$defaultpaymentnextmonth,
estimate = as.factor(dtree.comparison$Predictions))

dt_recall <- recall_vec(truth = cc_test$defaultpaymentnextmonth,
estimate = as.factor(dtree.comparison$Predictions))

dt_f1_score <- f_meas_vec(truth = cc_test$defaultpaymentnextmonth,
estimate = as.factor(dtree.comparison$Predictions))
```

The results will be displayed as following:

```
cat("Accuracy of Decision Tree model:", dt_accuracy, "\n")
cat("Precision of Decision Tree model:", dt_precision, "\n")
cat("Recall of Decision Tree model:", dt_recall, "\n")
cat("F1-score of Decision Tree mode:", dt_f1_score, "\n")
```

```
Accuracy of Decision Tree model: 0.8198969
Precision of Decision Tree model: 0.8356114
Recall of Decision Tree model: 0.958695
F1-score of Decision Tree mode: 0.8929317
```

5.2. Random Forest

5.2.1. Model

According to Biau (2012), random forests are a powerful machine learning technique that leverages the strength of multiple decision trees to provide robust and accurate predictions. It operates by constructing a multitude of decision trees during training and outputting the mode of the classes (for classification) or the mean prediction (for regression) of the individual trees.

To build a random forest, we will use *ranger()* function in the `ranger` packages.

```
rf_model <- ranger(defaultpaymentnextmonth ~ .,           # Formula indicating target and predictors
                   data = cc_train,                       # Training dataset
                   mtry = floor((ncol(cc_train) - 1) / 5), # Number of variables to try at each split
                   importance = "impurity",               # Measure of variable importance
                   num.trees = 500,                       # Number of trees in the forest
                   classification = TRUE)                 # Indicate that this is a classification problem
```

The code above builds a random forest model to predict whether there is a default payment next month using the `ranger` package. It uses all columns in the “cc_train” dataset as predictors, considers a subset of predictors at each split (calculated as one-fifth of the total predictors), measures variable importance by impurity, builds 500 trees, and is set up for classification.

5.2.2. Prediction

Using the *predict()* function to apply the trained model to the testing dataset:

```
rf_pred <- predict(rf_model, data = cc_test)
```

5.2.3. Evaluation

We also use functions in the `yardstick` package to evaluate evaluation metrics for our random forest model:

```
rf_accuracy <- accuracy_vec(cc_test$defaultpaymentnextmonth, estimate = as.factor(rf_pred$predictions))

rf_precision <- precision_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(rf_pred$predictions))

rf_recall <- recall_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(rf_pred$predictions))

rf_f1_score <- f_meas_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(rf_pred$predictions))
```

The results will be displayed as following:

```
cat("Accuracy of Random Forest model:", rf_accuracy, "\n")
cat("Precision of Random Forest model:", rf_precision, "\n")
cat("Recall of Random Forest model:",rf_recall, "\n")
cat("F1-score of Random Forest mode:", rf_f1_score, "\n")
```

```
Accuracy of Random Forest model: 0.8145223
Precision of Random Forest model: 0.8399651
Recall of Random Forest model: 0.9428731
F1-score of Random Forest mode: 0.8884491
```

5.3. kNN

5.3.1. Key Characteristics

1. Instance-Based Learning: The kNN algorithm is an example of instance-based learning; it produces predictions based just on the instances of training data, without explicitly learning a model (Brownlee, 2016).
2. Lazy Learning: Because it postpones calculation until prediction time, kNN is regarded as a lazy learner. It does not build an explicit model during training but uses the entire training dataset to make predictions (Divine_inner_voice , 2023).
3. Similarity Measure: kNN uses a similarity measure, often the Euclidean distance, to determine which of the 'k' training instances is most like the one being predicted. Other distance measurements, such as the Manhattan, Minkowski, or Hamming distance, may be used depending on the kind of data (Yasmeen, 2024).
4. Classification and Regression:

- In classification, the predicted class is determined by a majority vote among the k nearest neighbors. The class with the most representatives among the k neighbors is assigned to the instance (Srivastava, 2019).

- In regression, the prediction is typically the average (or sometimes the median) of the values of the k nearest neighbors (Srivastava, 2019).

5.3.2. Model

The k-Nearest Neighbors (kNN) model was trained on the cc_train dataset with 50 nearest neighbors (k=50) to predict the default payment status (defaultpaymentnextmonth).

```
knn_model <- kknn(defaultpaymentnextmonth ~ .,
                  train = cc_train,
                  test = cc_test,
                  k = 50,
                  distance = 2)
```

For node number 1:

- Specifies the training data excluding the target column. Here, the function removes the last column, which is assumed to be the target variable.

For node number 2:

- Specifies the test data for which predictions are to be made, also excluding the target column.

For node number 3:

- Provides the target variable from the training data. The target variable contains the actual class labels for the training instances.

For node number 4:

- k = 50: Sets the number of nearest neighbors to consider for making predictions. Here, 50 neighbors are used.

5.3.3. Prediction

```
knn.predictions <- predict(knn_model, newdata = cc_test)
```

knn_model:

- This is the kNN model that was previously trained on the training dataset (cc_train). The model includes all the parameters and information learned during the training phase, such as the number of neighbors (k), distance metric, and the training data itself.

predict() function:

- The predict() function is a generic function in R used for generating predictions from various types of models, including kNN models. It takes a model object and a new dataset as inputs and returns the predicted values for the target variable based on the new data.

newdata = cc_test:

- The argument newdata = cc_test specifies that the new dataset (cc_test) is to be used for making predictions. This dataset contains the instances for which we want to predict the target variable (in this case, defaultpaymentnextmonth).

knn.predictions:

- The result of the predict() function call is stored in the variable knn.predictions. This variable will contain the predicted class labels for each instance in the cc_test dataset.

In the prediction phase, the predict() function is used to apply the trained kNN model to a new dataset (cc_test). For each instance in the cc_test dataset, the kNN model calculates the distance between this instance and all instances in the training dataset. It then identifies the k nearest neighbors (the k instances with the smallest distance values). The model uses the majority class label among these k nearest neighbors to predict the class label for the instance in the cc_test dataset.

5.3.4. kNN Comparison

After training, the model was used to make predictions on the "cc_test" dataset. The predicted values were then compared with the actual values to evaluate the model's performance.

```
knn.compare <- cc_test
knn.compare$Predictions <- knn.predictions
knn.compare[, (c("defaultpaymentnextmonth", "Predictions"))]
```

For node number 1:

- This line creates a new data frame that is a copy of the test dataset. This is done to keep the original test dataset unchanged and to add a new column for predictions to this new data frame.

For node number 2:

- This line adds a new column named Predictions to the data frame. The values in this column are the predictions made by the kNN model. The model has the predicted class labels for each instance in the test dataset.

For node number 3:

- This line adds a new columns defaultpaymentnextmonth and Predictions to the knn.compare data frame.

Comparison Table

defaultpaymentnextmonth <fctr>	Predictions <fctr>
1	0
0	0
0	0
0	0
1	0
0	0
0	0
0	0
0	0
0	0

1–10 of 9,117 rows

- 'defaultpaymentnextmonth' is the actual target variable indicating whether the payment was defaulted or not.
- Predictions is the column containing the predicted values from the kNN model.

5.3.5. Evaluation

Additionally, a confusion matrix was generated to provide a detailed breakdown of the model's performance, showing 4615 true negatives, 57 false positives, 1254 false negatives, and 73 true positives.

- **Confusion matrix**

```
actual_knn <- factor(cc_test$defaultpaymentnextmonth)
predicted_knn <- factor(knn.predictions)

confusion_matrix_knn <- table(data.frame(actual_knn, predicted_knn))
confusion_matrix_knn
```

Definitions

- **actual_knn**: These are the actual labels from the test dataset.
- **predicted_knn**: These are the labels predicted by the kNN model.

The confusion matrix has four quadrants:

True Negatives (TN):

- **Definition**: Instances where the actual value is **0** (negative) and the predicted value is also **0**.
- **Value**: 6797
- **Interpretation**: The model correctly predicted 6797 instances as non-defaults.

False Positives (FP):

- **Definition**: Instances where the actual value is **0** (negative) but the predicted value is **1**.
- **Value**: 345
- **Interpretation**: The model incorrectly predicted 345 instances as defaults when they were non-defaults.

False Negatives (FN):

- **Definition**: Instances where the actual value is **1** (positive) but the predicted value is **0**.
- **Value**: 1366
- **Interpretation**: The model incorrectly predicted 1366 instances as non-defaults when they were defaults.

True Positives (TP):

- **Definition**: Instances where the actual value is **1** (positive) and the predicted value is also **1**.
- **Value**: 609
- **Interpretation**: The model correctly predicted 609 instances as defaults.
- **Evaluation metrics**

```
knn_accuracy <- accuracy_vec(cc_test$defaultpaymentnextmonth, estimate = as.factor(knn.compare$Predictions))

knn_precision <- precision_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(knn.compare$Predicti
ons))

knn_recall <- recall_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(knn.compare$Predictions))

knn_f1_score <- f_meas_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(knn.compare$Prediction
s))
```

- **Accuracy:** The accuracy of the kNN model, which measures the proportion of correctly classified instances, was calculated to be 0.7815, indicating that approximately 78.15% of the predictions were correct.
- **Precision:** Precision, which measures the proportion of true positive predictions among all positive predictions, was 0.7863, meaning that when the model predicted a default, it was correct 78.63% of the time.
- **Recall:** The recall, which measures the model's ability to identify all actual positive instances, was exceedingly high at 0.9878, showing that the model successfully identified 98.78% of the actual defaults.
- **F1 Score:** The F1 score, which balances precision and recall, was 0.8756, showing a good balance between the two metrics.

Criteria	kNN
Precision	0.7863
Recall	0.9878
F1 Score	0.8756
Accuracy	0.7815

These metrics collectively offer a comprehensive evaluation of the kNN model's effectiveness in predicting credit card default payments. The high recall indicates that the model is highly effective at finding defaults, while the precision and F1 score reflect a balanced and reliable performance. The confusion matrix offers further insights into the distribution of predictions and actual values, highlighting areas where the model performs well and areas where improvements could be made.

5.4. SVM

5.4.1. Key Characteristics

The Support Vector Machine (SVM) model is a strong and adaptable supervised machine learning approach used for regression and classification tasks. It is well recognized for its capacity to function in high-dimensional domains with both linear and non-linear data (IBM, 2023).

1. SVM can classify data in both linear and non-linear ways. It determines the optimal hyperplane to split the data into discrete classes for linear classification. To do non-linear classification, SVM uses kernel functions to transform the data into a higher-dimensional space that may include a linear separator (Saini, 2021).

2. Finding the hyperplane that maximizes the margin between the closest points in each class—also known as **support vectors**—is the aim of the Support Vector Machine (SVM). The gap between the closest data points from each class and the hyperplane is known as the margin. Maximizing this margin helps improve the generalization ability of the classifier (Saini, 2021).

3. Kernel Trick: Without explicitly calculating the coordinates in a high-dimensional space, SVM may quickly carry out transformations and identify an ideal boundary there. The radial basis function (RBF) kernel, polynomial kernel, and linear kernel are examples of frequently used kernels (Wilimitis, 2019).

4. The SVM model has a regularization parameter called C that regulates the trade-off between decreasing classification errors and maximizing margin. While a major value of C tries to identify all training instances correctly but with a tighter margin, a small value of C generates a broader margin but permits more misclassifications (Agarwal, 2021).

5.4.2. Model

The Support Vector Machine (SVM) model was trained on the cc_train dataset to predict the likelihood of default payment for the next month. After training, the SVM model, which used a radial kernel and a cost parameter of 1, included 9311 support vectors.

```
set.seed(123)
svm_model <- svm(defaultpaymentnextmonth ~ ., data = cc_train)
svm_model
summary(svm_model)
```

For Node Number 1:

- Setting the seed ensures that the results are reproducible. The **set.seed(123)** function sets the random number generator to a specific state, allowing for consistent results when the code is run multiple times.

For Node Number 2:

- **svm**: This function is from the **e1071** package, which provides various machine learning algorithms, including SVM.
- **defaultpaymentnextmonth ~.**: This formula specifies that **defaultpaymentnextmonth** is the target variable (the variable we want to predict), and indicates that all other columns in the **cc_train** dataset are used as predictor variables.

- **data = cc_train**: This specifies that the training data is in the **cc_train** data frame.

```
Call:
svm(formula = defaultpaymentnextmonth ~ ., data = cc_train)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  9311
```

For Node Number 1:

- This indicates the function call that was used to train the SVM model.
- The formula **defaultpaymentnextmonth ~ .** specifies that **defaultpaymentnextmonth** is the target variable (the variable to predict), and means all other columns in the **cc_train** dataset is used as predictors.
- **data = cc_train** indicates that the training data is sourced from the **cc_train** dataset.

For Node Number 2:

- **SVM-Type**: **C-classification**
- This specifies that the type of SVM used is a C-classification SVM, which is a common SVM variant used for classification tasks.
- **SVM-Kernel**: **radial**
- This indicates that the kernel function used is the Radial Basis Function (RBF) kernel. The RBF kernel is popular for handling non-linear relationships in the data by mapping the input space into a higher-dimensional space.
- **Cost**: **1**
- The cost parameter, also known as **C**, controls the trade-off between achieving a low error on the training data and minimizing the model complexity to avoid overfitting. A cost of **1** is a typical default value.

For Node Number 3:

- This line indicates the total number of support vectors used by the trained SVM model.
- Support vectors are the data points that lie closest to the decision boundary and are crucial for defining the position and orientation of the decision boundary.
- In this case, the model uses 9311 support vectors out of the total training instances.

```
Call:
svm(formula = defaultpaymentnextmonth ~ ., data = cc_train)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  9311

 ( 5257 4054 )


Number of Classes:  2

Levels:
 0 1
```

For Node Number 1:

- **Distribution of Support Vectors**: **(5257 4054)**
  - This indicates the number of support vectors for each class. In this case, there are 5257 support vectors for one class and 4054 support vectors for the other class. The first number usually corresponds to the first class (e.g., **0**), and the second number to the second class (e.g., **1**).

For Node Number 2:

- This shows that the target variable has two classes, which is typical for binary classification problems. In this context, the classes represent whether a credit card payment will default (e.g., **0** for no default, **1** for default).

For Node Number 3:

- **Levels**: **0 1**
  - This indicates the levels of the target variable **defaultpaymentnextmonth**. The levels **0** and **1** correspond to the two possible outcomes (e.g., **0** for non-default, **1** for default).

5.4.3. Prediction

The model was then used to make predictions on the cc_test dataset. The predicted values were compared with the actual values to evaluate the model's performance.

```
svm_pred <- predict(svm_model, newdata = cc_test)
svm_pred
```

- **svm_model**: This is the trained SVM model obtained from the previous training step using the **svm** function.
- **newdata = cc_test**: This specifies the new data (test dataset) on which predictions are to be made. The **cc_test** dataset contains the instances for which the model will predict the target variable (**defaultpaymentnextmonth**).

```
   1    2    3    4    5    6    7    8    9   10
21   22   23
   0    0    0    0    0    0    0    0    0    0
0    0    0
  24   25   26   27   28   29   30   31   32   33
44   45   46
   0    0    1    0    0    0    0    1    0    0
0    0    0
```

- **Predicted Values**: These are the model's predictions for each instance in the test dataset. The values are:
  - **0**: Indicates that the model predicts no default for the corresponding instance.
  - **1**: Indicates that the model predicts a default for the corresponding instance.

5.4.4. Comparison

```
svm.compare <- cc_test
svm.compare$Predictions <- svm_pred
svm.compare[ , c("defaultpaymentnextmonth", "Predictions")]
```

For Node Number 1:

- This line creates a copy of the **cc_test** dataset and stores it in a new variable called **svm.compare**. This is done to add the predicted values to this dataset without altering the original test data.

For Node Number 2:

- This line adds a new column named **Predictions** to the **svm.compare** dataset. The values in this column are the predictions generated by the SVM model (**svm_pred**).

```
# A tibble: 9,117 × 2
   defaultpaymentnextmonth Predictions
   <fct>                   <fct>
 1 1                       0
 2 0                       0
 3 0                       0
 4 0                       0
 5 1                       0
 6 0                       0
 7 0                       0
 8 0                       0
 9 0                       0
10 0                       0
# ℹ 9,107 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Format:

- The output is a table with 9,117 rows and 2 columns.
- Tibbles provide a cleaner and more concise print method compared to traditional data frames.

Columns:

- **defaultpaymentnextmonth**: This column contains the actual values from the test dataset, indicating whether the payment defaulted or not.
- **Predictions**: This column contains the predicted values from the SVM model for each instance in the test dataset.

Rows:

- Each row corresponds to an individual instance from the test dataset.
- The first row shows an actual value of **1** (indicating default) and a predicted value of **0** (indicating no default).
- The second row shows an actual value of **0** (indicating no default) and a predicted value of **0**.
- And so on, for a total of 9,117 rows.

Interpretation

- **Row 1**: The actual value is **1** (default), but the predicted value is **0** (no default). This is a false negative.
- **Row 2**: The actual value is **0** (no default), and the predicted value is also **0**. This is a true negative.

- **Row 5**: The actual value is **1** (default), but the predicted value is **0** (no default). This is another false negative.

```
svm_accuracy <- accuracy_vec(cc_test$defaultpaymentnextmonth, estimate =
                               as.factor(svm.compare$Predictions))
svm_accuracy

svm_precision <- precision_vec(truth = cc_test$defaultpaymentnextmonth, estimate =
                                 as.factor(svm.compare$Predictions))
svm_precision

svm_recall <- recall_vec(truth = cc_test$defaultpaymentnextmonth, estimate =
                           as.factor(svm.compare$Predictions))
svm_recall

svm_f1_score <- f_meas_vec(truth = cc_test$defaultpaymentnextmonth, estimate =
                             as.factor(svm.compare$Predictions))
svm_f1_score
```

- The accuracy of the SVM model, which measures the proportion of correctly classified instances, was calculated to be 0.8193, indicating that approximately 81.93% of the predictions were correct.
- The precision, which indicates the proportion of true positive predictions among all positive predictions made by the model, was found to be 0.8345. This means that when the model predicted a default, it was correct 83.45% of the time.
- The recall, which measures the model's ability to identify all actual positive instances, was extremely high at 0.9580, showing that the model successfully identified 95.80% of the actual defaults.
- The F1 score, a harmonic mean of precision and recall, was calculated to be 0.8920, providing a balanced measure of the model's performance.

Criteria	SVM
Precision	0.8345
Recall	0.9580
F1 Score	0.8920
Accuracy	0.8193

These metrics collectively offer a comprehensive evaluation of the SVM model's effectiveness in predicting default payment, highlighting its strengths in both precision and recall. The high recall indicates the model's ability to identify most default cases, while the solid F1 score reflects a good balance between precision and recall, underscoring the model's overall robustness and reliability.

5.5. Logistic Regression

5.5.1. Model

Logistic regression is a widely utilized statistical method in the field of data analysis. The focus of logistic regression analysis is classification of individuals in different contexts based on a set of independent variables. Credit card default prediction involves complex data analysis to identify patterns and trends that can influence repayment behavior. Merikoski, Viitala, and Shafik (2018) emphasized the importance of predicting and preventing credit card defaults, underscoring the role of logistic regression in enhancing predictive accuracy. By leveraging logistic regression alongside other data mining techniques, financial institutions can better anticipate default risks and implement preventive measures to mitigate potential losses (Merikoski, Viitala, & Shafik, 2018)

Applying logistic regression model to our case, firstly, we used "cc_train" to build model by using *glm()* function which is used for general linear models.

```
> glm_model <- glm(defaultpaymentnextmonth ~ .,
+                  data = cc_train,
+                  family = binomial(link = "logit"))
```

- defaultpaymentnextmonth ~ . shows that "defaultpaymentnextmonth" is our target variable, and it will be predicted based on all other columns which is indicated by "."
- data = cc_train means that the dataset "cc_train" will be used for fitting the model
- family = binomial(link = "logit") declares that the model is a logistic regression model, a kind of binomial GLM in which the logit (logistic) function serves as the link function.

In this scenario, the target variable is whether a customer would miss a payment the following month "defaultpaymentnextmonth". The *coef()* function will show the link between each predictor variable and the log chances of the target variable.

```
> coef(glm_model)
  (Intercept)            ID     LIMIT_BAL           SEX     EDUCATION      MARRIAGE
-5.556556e-01 -2.317601e-06 -7.972429e-07 -1.133834e-01 -1.085874e-01 -1.645273e-01
          AGE         PAY_0         PAY_2         PAY_3         PAY_4         PAY_5
 6.569899e-03  5.781510e-01  7.179483e-02  7.568990e-02  5.426246e-02 -1.264052e-02
        PAY_6     BILL_AMT1     BILL_AMT2     BILL_AMT3     BILL_AMT4     BILL_AMT5
 2.815939e-02 -4.603569e-06 -1.750739e-07  2.910394e-06  9.562237e-07 -1.920149e-07
    BILL_AMT6      PAY_AMT1      PAY_AMT2      PAY_AMT3      PAY_AMT4      PAY_AMT5
 2.499861e-07 -1.525564e-05 -1.146970e-05 -3.543676e-06 -4.730372e-06 -2.943807e-06
     PAY_AMT6
-3.265056e-06
```

- Positive Coefficient Predictors: Variables like AGE and PAY_0 to PAY_6 indicate that greater values are linked to a higher likelihood of missing a payment the next month.
- Negative Coefficient Predictors: Variables like LIMIT_BAL, SEX, EDUCATION, MARRIAGE, and PAY_AMT1 through PAY_AMT6 indicate that a higher value is linked to a lower likelihood of defaulting.

5.5.2. Prediction

```
> glm_pred <- cc_test %>%
+   mutate(propensity = predict(glm_model, ., type = 'response'),
+          glm_est = factor(as.numeric(propensity > 0.5), labels = c("0", "1"), levels = c(0, 1)),
+          index = 1:n())
>
> glm_pred %>% select(propensity, glm_est, defaultpaymentnextmonth)
> str(glm_pred)
```

Moving to the prediction of the logistic regression model, we used *predict()* function to calculate each observation's expected probability of an event occurring in the dataset “cc_test”. To assist create a more understandable and efficient workflow, chain together numerous data manipulation actions using the %>% operator from the ‘dyplr’ package.

- propensity: It refers to the likelihood that a person will not make their payment in the upcoming month. We tried to simplify the prediction into a binary outcome by determining if a person has a greater or lesser probability of missing a payment than not. They are labelled as "1" if it is greater than or equal to, and as "0" if it is less or equal. And we also used ‘levels’ to make sure that guarantees that the interpretation of the levels is consistent throughout the analysis.

The figure below shows the predicted propensity, propensity after being simplified into a binary outcome and the real ‘defaultpaymentnextmonth’.

propensity <dbl>	glm_est <fctr>	defaultpaymentnextmonth <fctr>
0.5229970359	1	1
0.2508549811	0	0
0.1807940827	0	0
0.0814993111	0	0
0.0437180568	0	1
0.2272523238	0	0
0.1150636264	0	0
0.0253543019	0	0
0.0482003327	0	0
0.1265879878	0	0

1–10 of 9,117 rows

5.5.3. Evaluation

- Confusion matrix

```
> glm_pred %>%
+   conf_mat(defaultpaymentnextmonth, glm_est) %>%
+   pluck("table") %>%
+   addmargins()
          Truth
Prediction    0    1  Sum
       0   6956 1510 8466
       1    186  465  651
       Sum 7142 1975 9117
```

  - True Positives (TP): 465
  - True Negatives (TN): 6956
  - False Positives (FP): 1510
  - False Negatives (FN): 186
- Evaluation metrics

Unlike accuracy(), which may oversimplify the evaluation by focusing solely on individual data points, accuracy_vec() considers the vectorized representation of data, allowing for a more comprehensive analysis of the context and relationships within the dataset (Reddy et al., 2023).

```
> glm_accuracy <- accuracy_vec(truth = cc_test$defaultpaymentnextmonth, estimate = glm_pred$glm_est)
>
> glm_precision <- precision_vec(cc_test$defaultpaymentnextmonth, estimate = glm_pred$glm_est)
>
> glm_recall <- recall_vec(cc_test$defaultpaymentnextmonth, estimate = glm_pred$glm_est)
>
> glm_f1_score <- f_meas_vec(truth = cc_test$defaultpaymentnextmonth, estimate = glm_pred$glm_est, beta = 1)
```

```
cat("Accuracy of logistic regression model:", glm_accuracy, "\n")
cat("Precision of logistic regression model:", glm_precision, "\n")
cat("Recall of logistic regression model:",glm_recall, "\n")
cat("F1-score of logistic regression mode:", glm_f1_score, "\n")
```

```
Accuracy of logistic regression model: 0.8139739
Precision of logistic regression model: 0.8216395
Recall of logistic regression model: 0.9739569
F1-score of logistic regression mode: 0.8913378
```

6. Evaluation Analysis

This section assesses the efficacy of the five proposed machine learning models: Random Forest, Decision Tree, Support Vector Machine (SVM), k-Nearest Neighbours (kNN), and Logistic Regression. Accuracy, Precision, Recall, and F1 Score are our four main evaluation metrics that we use to find the optimal model for our dataset. Every metric offers distinct insights into the efficacy of the model and helps us gain understanding of different aspects of their predicting powers.

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.8198969	0.8356114	0.958695	0.8929317
Random Forest	0.8145223	0.8399651	0.9428731	0.8884491
kNN	0.7815	0.7863	0.9878	0.8756
SVM	0.8193	0.8345	0.9580	0.8920
Logistic Regression	0.8139739	0.8216395	0.9739568	0.8913378

Going in more detail of the analysis, we can find that:

- **Accuracy**
  - Highest accuracy: Decision Tree (0.8198969) and SVM (0.8193).
  - All models demonstrate high accuracy, with kNN having the lowest at 0.7815.
- **Precision**
  - Random Forest achieves the highest precision at 0.8399651.
  - When the cost of false positives is large, precision is essential. Random Forest marginally surpasses the others.

- **Recall**
  - KNN has the highest recall (0.9878).
  - Recall is crucial when the cost of false negatives is high. kNN significantly outperforms others in this regard.
- **F1 Score**
  - Decision Tree achieved the highest F1 score (0.8929317).
  - The F1 score is a measure that balances precision and recall.
  - The Decision Tree demonstrates the best balance according to the F1 score.

Considering the significance of all metrics, it is important to summarize the strengths of each model:

- **Decision Tree**: Best in Accuracy and F1 Score, with extremely good Precision and Recall.
- **Random Forest** excels in Precision, Accuracy, recall, and F1 Score.
- **kNN**: Has the highest Recall but lower in Accuracy and Precision.
- **SVM**: Very high across all measures, but not the best in any one.
- **Logistic Regression**: High Recall, competitive in other measures, but not leading.

Based on the analysis, the Decision Tree is the best overall model based on its highest Accuracy among all. It also has the highest F1 Score, which shows a good balance of Precision and Recall, as well as highly competitive metrics in Precision and Recall. However, if prioritizing Recall is key, kNN could be a valuable selection despite its lower overall Accuracy. For a balanced performance across all metrics, the Decision Tree emerges as the top choice.

Conclusion

This project has successfully developed a sophisticated predictive model that enhances the bank's ability to assess credit card eligibility, improving risk management and customer service efficiency. By integrating advanced machine learning techniques—such as Decision Trees, Random Forests, k-nearest Neighbors, Support Vector Machines, and Logistic Regression—the model not only optimizes the accuracy of credit evaluations but also streamlines the application process, thereby boosting operational efficiency and customer satisfaction. The exploratory data analysis provided deep insights into customer behaviours, informing tailored financial products and effective marketing strategies. Moreover, strict regulatory and ethical standards ensure the model's compliance and reliability. Looking forward, continuous updates

and the integration of more dynamic data sources could further refine the model's predictive accuracy. In essence, this project has achieved its goals and laid a robust foundation for enhancing the bank's strategic operations and competitive edge in the financial sector.

References

Agarwal, D. (2021, April 26). *Introduction to SVM(Support Vector Machine) Along with Python Code*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/04/insight-into-svm-support-vector-machine-along-with-code/

Agbor, E. A., & Qu, Y. (2023). Improving the Performance of Machine Learning Model Selection for Electricity Cost Forecasting in Homebased Small Businesses via Exploratory Data Analysis. *European Journal of Electrical Engineering and Computer Science*, *7*(2), 7–14. https://doi.org/10.24018/ejece.2023.7.2.481

Akoglu, H. (2018). User's guide to correlation coefficients. *Turkish journal of emergency medicine, 18*(3), 91-93.

Aldoseri , A., Khalifa , K. A. -, & Hamouda, A. (2023). *A roadmap for integrating automation with process optimization for ai-powered digital transformation*. https://www.researchgate.net/figure/Business-Process-Optimization-benefits_fig1_374784690

Bhatia, H., Carpenter, T. S., Ingólfsson, H. I., Dharuman, G., Karande, P., Liu, S., Oppelstrup, T., Neale, C., Lightstone, F. C., Van Essen, B., Glosli, J. N., & Bremer, P. T. (2021). Machine-learning-based dynamic-importance sampling for adaptive multiscale simulations. Nature Machine Intelligence, 3(5), 401–409. https://doi.org/10.1038/s42256-021-00327-w

Biau, G. (2012). Analysis of a random forests model. *The Journal of Machine Learning Research, 13*(1), 1063-1095.

Brownlee, J. (2016, September 22). *K-Nearest Neighbors for Machine Learning*. Machine Learning Mastery. https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/

Divine_inner_voice . (2023, March 7). *A Look into Lazy Learning Techniques: Machine Learning with KNN*. Medium. https://medium.com/@divine_inner_voice/a-look-into-lazy-learning-techniques-machine-learning-with-knn-e73499fc352c#:~:text=KNN%20is%20a%20lazy%20learning

Faed, A., Hussain, O. K., & Chang, E. (2013). A methodology to map customer complaints and measure customer satisfaction and loyalty. *Service Oriented Computing and Applications*, *8*(1), 33–53. https://doi.org/10.1007/s11761-013-0142-6

Farayola, O. A. (2024). Revolutionizing banking security: integrating artificial intelligence, blockchain, and business intelligence for enhanced cybersecurity. *Finance & Accounting Research Journal*, *6*(4), 501–514. https://doi.org/10.51594/farj.v6i4.990

Huang, T.-H., Huang, M. L., Nguyen, Q. V., Zhao, L., Huang, W., & Chen, J. (2017). A Space-Filling Multidimensional Visualization (SFMDVis) for Exploratory Data Analysis. *Information Sciences*, *390*, 32–53. https://doi.org/10.1016/j.ins.2015.06.031

IBM. (2023, December 27). *What is support vector machine? | IBM*. Www.ibm.com. https://www.ibm.com/topics/support-vector-machine

Lantz, B. (2019). Machine Learning with R: Expert techniques for predictive modeling. Packt Publishing Ltd.

Oreski, S., Oreski, D., & Oreski, G. (2012). Hybrid system with genetic algorithm and artificial neural networks and its application to retail credit risk assessment. *Expert Systems with Applications*, *39*(16), 12605–12617. https://doi.org/10.1016/j.eswa.2012.05.023

Orji, Ugochukwu. E., Ukwandu, E., Obianuju, Ezugwu. A., Ezema, Modesta. E., Ugwuishiwu, Chikaodili. H., & Egbugha, Malachi. C. (2022, November 1). *Visual Exploratory Data Analysis of the Covid-19 Pandemic in Nigeria: Two Years after the Outbreak*. IEEE Xplore. https://doi.org/10.1109/ITED56637.2022.10051529

Pokholkova, M., Boch, A., Hohma, E., & Christoph Lütge. (2024). Measuring adherence to AI ethics: A methodology for assessing adherence to ethical principles in the use case of AI-enabled credit scoring application. *AI and Ethics*. https://doi.org/10.1007/s43681-024-00468-9

Reddy, R. A., Rekha, J., Shirisha, M., Hussein, A. H. A., & Al-Attabi, K. (2023). Semantic Feature Extraction-Based Twitter Sentiment Analysis Using Atom Search Optimizer and Ensemble Classifier. *2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIE)*. https://doi.org/10.1109/aikiie60097.2023.10389833

Saini, A. (2021, October 12). *Support Vector Machine(SVM): A Complete guide for beginners*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

Sakthiswaran Rangaraju. (2023). Ecure by intelligence: Enhancing products with ai-driven security measures. *EPH - International Journal of Science and Engineering*, *9*(3), 36–41. https://doi.org/10.53555/ephijse.v9i3.212

Song, Y. Y., & Ying, L. U. (2015). Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry, 27*(2), 130.

Srivastava, T. (2019, March 7). *Introduction to KNN, K-Nearest Neighbors : Simplified*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

Wilimitis, D. (2019, February 21). *The Kernel Trick*. Medium. https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f

Yasmeen, R. (2024, March 16). *K-Nearest Neighbor(KNN) Algorithm in Machine Learning*. Medium. https://medium.com/@rizwanayasmeen06/k-nearest-neighbor-knn-algorithm-in-machine-learning-d38d9638d7e0

Appendices

Figure 1:

Set up all the libraries (already installed at the console part)

```
library(ggplot2)
library(GGally)
library(tidyverse)
library(dplyr)
library(purrr)
library(ROSE)
library(caret)
library(kknn)
library(rpart)
library(rpart.plot)
library(rsample)
library(ranger)
library(MLmetrics)
library(broom)
library(yardstick)
library(e1071)
```

Figure 2:

Import data

```
cccard <- data
View(cccard)
str(cccard)
```

Figure 3:

Explore data type and covert categorical columns to factor

```
# Define categorical columns
catagorical_columns <- c('SEX','EDUCATION','MARRIAGE','PAY_0','PAY_2','PAY_3','PAY_4','PAY_5','PAY_6')

## Transform categorical columns to factor data type
cccard$SEX <- as.factor(as.character(cccard$SEX))
cccard$EDUCATION <- as.factor(as.character(cccard$EDUCATION))
cccard$MARRIAGE <- as.factor(as.character(cccard$MARRIAGE))
cccard$PAY_0 <- as.factor(as.character(cccard$PAY_0))
cccard$PAY_2 <- as.factor(as.character(cccard$PAY_2))
cccard$PAY_3 <- as.factor(as.character(cccard$PAY_3))
cccard$PAY_4 <- as.factor(as.character(cccard$PAY_4))
cccard$PAY_5 <- as.factor(as.character(cccard$PAY_5))
cccard$PAY_6 <- as.factor(as.character(cccard$PAY_6))
cccard$defaultpaymentnextmonth <- as.factor(as.character(cccard$defaultpaymentnextmonth))
```

Figure 4:

Count unidentified values in dataset

```
pay_0 <- sum(cccard$PAY_0 == 0 | cccard$PAY_0 == -2)
pay_2 <- sum(cccard$PAY_2 == 0 | cccard$PAY_2 == -2)
pay_3 <- sum(cccard$PAY_3 == 0 | cccard$PAY_3 == -2)
pay_4 <- sum(cccard$PAY_4 == 0 | cccard$PAY_4 == -2)
pay_5 <- sum(cccard$PAY_5 == 0 | cccard$PAY_5 == -2)
pay_6 <- sum(cccard$PAY_6 == 0 | cccard$PAY_6 == -2)
education <- sum(cccard$EDUCATION == 5 | cccard$EDUCATION == 6)
marriage <- sum(cccard$MARRIAGE == 0)

#Count all unidentified values
count <- data.frame(pay_0,pay_2,pay_3,pay_4,pay_5,pay_6,education,marriage)

count
```

Figure 5:

*Transform value 0 and -2 from the **PAY_0** -> **PAY_6** columns to -1*

```
indices <- which(cccard$PAY_0 == 0 | cccard$PAY_0 == -2)
cccard$PAY_0[indices] <- -1
indices <- which(cccard$PAY_2 == 0 | cccard$PAY_2 == -2)
cccard$PAY_2[indices] <- -1
indices <- which(cccard$PAY_3 == 0 | cccard$PAY_3 == -2)
cccard$PAY_3[indices] <- -1
indices <- which(cccard$PAY_4 == 0 | cccard$PAY_4 == -2)
cccard$PAY_4[indices] <- -1
indices <- which(cccard$PAY_5 == 0 | cccard$PAY_5 == -2)
cccard$PAY_5[indices] <- -1
indices <- which(cccard$PAY_6 == 0 | cccard$PAY_6 == -2)
cccard$PAY_6[indices] <- -1
```

Figure 6:

*Transform the same values in **EDUCATION** and **MARRIAGE** column*

```
indices <- which(cccard$EDUCATION == 5 | cccard$EDUCATION == 6 | cccard$EDUCATION == 0)
cccard$EDUCATION[indices] <- 4
indices <- which(cccard$MARRIAGE == 0)
cccard$MARRIAGE[indices] <- 3
```

Figure 7:

Define duplicated values and remove ID column in the dataset.

```
#Define dupplicated values in the data set
sum(duplicated(cccard))

#Summary of the NA value
colSums(is.na(cccard))

#Remove first column (ID column)
cccard <- cccard[,-c(1)]
```

Figure 8:

*Box-plot for **Age** and **Limit_Bal***

```
#Box-plot for *AGE*
ggplot(data = cccard, mapping=aes(x = defaultpaymentnextmonth, y = AGE, fill = defaultpaymentnextmonth)) +
  geom_boxplot() +
  stat_summary(fun = "mean", geom = "point", shape = 8,
               size = 2, color = "white")

#Box-plot for *LIMIT_BAL*
ggplot(data = cccard, mapping=aes(x = defaultpaymentnextmonth, y = LIMIT_BAL, fill = defaultpaymentnextmonth)) +
  geom_boxplot() +
  stat_summary(fun = "mean", geom = "point", shape = 8,
               size = 2, color = "white")
```

Figure 9:

Bar chart for ***Sex, Education, Marriage****, and* ***defaultpaymentnextmont***

```
ggplot(data = cccard, mapping=aes(x = SEX, fill = defaultpaymentnextmonth)) + geom_bar()
ggplot(data = cccard, mapping=aes(x = EDUCATION, fill = defaultpaymentnextmonth)) + geom_bar()
ggplot(data = cccard, mapping=aes(x = MARRIAGE, fill = defaultpaymentnextmonth)) + geom_bar()
ggplot(data = cccard, mapping=aes(x = defaultpaymentnextmonth, fill = defaultpaymentnextmonth)) + geom_bar()
```

Figure 10:

Outlier Removal

```
#Calculate 1st, 3rd quartiles and IQR
cleanOutliers <- function(data, var_name) {
  q1 <- quantile(data[[var_name]], 0.25)
  q3 <- quantile(data[[var_name]], 0.75)
  iqr <- q3 - q1

  #Define lower and upper bounds for outlier detection
  lower_bound <- q1 - 1.5 * iqr
  upper_bound <- q3 + 1.5 * iqr

  #Identify outliers
  outliers <- data[[var_name]] < lower_bound | data[[var_name]] > upper_bound

  #Remove outliers from the original dataset
  cleaned_data <- data[!outliers, ]

  summary(cleaned_data)
  return(cleaned_data)
}

cccard <- cleanOutliers(cccard, "LIMIT_BAL")
```

Figure 11:

Upscaling data

```
target <- "defaultpaymentnextmonth"
predictors <- c("LIMIT_BAL","SEX","EDUCATION", "MARRIAGE","AGE","PAY_0","PAY_2","PAY_3","PAY_4","PAY_5","PAY_
6","BILL_AMT1",
                "BILL_AMT2","BILL_AMT3","BILL_AMT4","BILL_AMT5","BILL_AMT6","PAY_AMT1","PAY_AMT2","PAY_AMT3","PAY
_AMT4",
                "PAY_AMT5","PAY_AMT6")
formula <- as.formula(paste(target, "~", paste(predictors, collapse = "+")))
```

Figure 12:

Perform Ovun resampling then view dataset after preparing

```
cccard_bal <- ovun.sample(formula, cccard, method = "both", p = 0.5,
                          N = 29833)$data
View(cccard_bal)
summary(cccard_bal)
```

Figure 13:

Spliting dataset into a training set and a testing set (70:30)

```
set.seed(123)
trainIndex <- sample(1:nrow(cccard_bal),0.70*nrow(cccard_bal))
cc_train <- data[trainIndex, ]
cc_test <- data[-trainIndex, ]
```

Figure 14:

Convert train and test dataset to factor

```
cc_train$defaultpaymentnextmonth <- as.factor(cc_train$defaultpaymentnextmonth)

cc_test$defaultpaymentnextmonth <- as.factor(cc_test$defaultpaymentnextmonth)
```

Figure 15:

Build Decision Tree model to predict defaultpaymentnextmonth based on all other columns

```
tree_model <- rpart(defaultpaymentnextmonth ~., data = cc_train)
```

Figure 16:

Visualize decision tree model

```
rpart.plot(tree_model, extra = 106)
title(main = "Decision Tree Model of Credit Card Data")
```

Figure 17:

Make Decision Tree prediction

```
dtree.predictions <- predict(tree_model, cc_test, type = "class")
```

Figure 18:

Compare the actual values and predictions

```
dtree.comparison <- cc_test
dtree.comparison$Predictions <- dtree.predictions
dtree.comparison[ , c("defaultpaymentnextmonth", "Predictions")]
```

Figure 19:

Plot confusion matrix (Decision Tree)

```
actual_dt <- factor(cc_test$defaultpaymentnextmonth)
pred_dt <- factor(dtree.predictions)

confusion_matrix_dt <- table(data.frame(actual_dt, pred_dt))
confusion_matrix_dt
```

Figure 20:

Calculating accuracy, precision, recall, f1_score (Decision Tree)

```
dt_accuracy <- accuracy_vec(cc_test$defaultpaymentnextmonth, estimate = as.factor(dtree.comparison$Predictions))

dt_precision <- precision_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(dtree.comparison$Pred
ictions))

dt_recall <- recall_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(dtree.comparison$Prediction
s))

dt_f1_score <- f_meas_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(dtree.comparison$Predicti
ons))
```

Figure 21:

Display the metrics

```
cat("Accuracy of Decision Tree model:", dt_accuracy, "\n")
cat("Precision of Decision Tree model:", dt_precision, "\n")
cat("Recall of Decision Tree model:", dt_recall, "\n")
cat("F1-score of Decision Tree mode:", dt_f1_score, "\n")
```

Figure 22:

Build Random Forest model to predict defaultpaymentnextmonth based on all other columns

```
rf_model <- ranger(defaultpaymentnextmonth ~ .,
                   data = cc_train,
                   mtry = floor((ncol(cc_train) - 1) / 5),
                   importance = "impurity",
                   num.trees = 500,
                   classification = TRUE)
```

Figure 23:

Make random forest prediction

```
rf_pred <- predict(rf_model, data = cc_test)
rf_pred_label <- rf_pred$predictions
```

Figure 24:

Calculating accuracy, precision, recall, f1_score (Random Forest)

```
rf_accuracy <- accuracy_vec(cc_test$defaultpaymentnextmonth, estimate = as.factor(rf_pred$predictions))

rf_precision <- precision_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(rf_pred$predictions))

rf_recall <- recall_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(rf_pred$predictions))

rf_f1_score <- f_meas_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(rf_pred$predictions))
rf_f1_score
```

Figure 25:

Display the metrics

```
cat("Accuracy of Random Forest model:", rf_accuracy, "\n")
cat("Precision of Random Forest model:", rf_precision, "\n")
cat("Recall of Random Forest model:",rf_recall, "\n")
cat("F1-score of Random Forest mode:", rf_f1_score, "\n")
```

Figure 26:

Build kNN model to predict defaultpaymentnextmonth based on all other columns

- *Set K value to 50*
- *distance = 2*

```
knn_model <- kknn(defaultpaymentnextmonth ~ .,
                          train = cc_train,
                          test = cc_test,
                          k = 50,
                          distance = 2)
```

Figure 27:

Make Knn prediction

```
knn.predictions <- predict(knn_model, newdata = cc_test)
```

Figure 28:

Comparison table for KNN actual values and predictions for defaultpaymentnextmonth

```
knn.compare <- cc_test
knn.compare$Predictions <- knn.predictions
knn.compare[, (c("defaultpaymentnextmonth", "Predictions"))]
```

Figure 29:

Calculating accuracy, precision, recall, f1_score (Knn)

```
knn_accuracy <- accuracy_vec(cc_test$defaultpaymentnextmonth, estimate = as.factor(knn.compare$Predictions))

knn_precision <- precision_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(knn.compare$Predicti
ons))

knn_recall <- recall_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(knn.compare$Predictions))

knn_f1_score <- f_meas_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(knn.compare$Prediction
s))
```

Figure 30:

Display the metrics

```
cat("Accuracy of Knn model:", knn_accuracy, "\n")
cat("Precision of Knn model:", knn_precision, "\n")
cat("Recall of Knn model:",knn_recall, "\n")
cat("F1-score of Knn mode:", knn_f1_score, "\n")
```

Figure 31:

Build SVM model to predict defaultpaymentnextmonth based on all other columns

```
svm_model <- svm(defaultpaymentnextmonth ~ ., data = cc_train)
```

Figure 32:

Make SVM prediction

```
svm_pred <- predict(svm_model, newdata = cc_test)
```

Figure 33:

Comparison table for SVM actual values and predictions for defaultpaymentnextmonth

```
svm.compare <- cc_test
svm.compare$Predictions <- svm_pred
svm.compare[ , c("defaultpaymentnextmonth", "Predictions")]
```

Figure 34:

Calculating accuracy, precision, recall, f1_score (SVM)

```
svm_accuracy <- accuracy_vec(cc_test$defaultpaymentnextmonth, estimate = as.factor(svm.compare$Predictions))

svm_precision <- precision_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(svm.compare$Predicti
ons))

svm_recall <- recall_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(svm.compare$Predictions))

svm_f1_score <- f_meas_vec(truth = cc_test$defaultpaymentnextmonth, estimate = as.factor(svm.compare$Prediction
s))
```

Figure 35:

Display the metrics

```
cat("Accuracy of SVM model:", svm_accuracy, "\n")
cat("Precision of Knn model:", svm_precision, "\n")
cat("Recall of Knn model:",svm_recall, "\n")
cat("F1-score of Knn mode:", svm_f1_score, "\n")
```

Figure 36:

Build Logistic Regression model to predict defaultpaymentnextmonth based on all other columns

```
glm_model <- glm(defaultpaymentnextmonth ~ .,
                 data = cc_train,
                 family = binomial(link = "logit"))
```

Figure 37:

Make logistic regression prediction

```
glm_pred <- cc_test %>%
  mutate(propensity = predict(glm_model, ., type = 'response'),
         glm_est = factor(as.numeric(propensity > 0.5), labels = c("0", "1"), levels = c(0, 1)),
         index = 1:n())

glm_pred %>% select(propensity, glm_est, defaultpaymentnextmonth)
str(glm_pred)
```

Figure 38:

Plot confusion matrix (Logistic Regression)

```
glm_pred %>%
  conf_mat(defaultpaymentnextmonth, glm_est) %>%
  pluck("table") %>%
  addmargins()
```

Figure 39:

Calculating accuracy, precision, recall, f1_score (Logistic Regression)

```
glm_accuracy <- accuracy_vec(truth = cc_test$defaultpaymentnextmonth, estimate = glm_pred$glm_est)

glm_precision <- precision_vec(cc_test$defaultpaymentnextmonth, estimate = glm_pred$glm_est)

glm_recall <- recall_vec(cc_test$defaultpaymentnextmonth, estimate = glm_pred$glm_est)

glm_f1_score <- f_meas_vec(truth = cc_test$defaultpaymentnextmonth, estimate = glm_pred$glm_est, beta = 1)
```

Figure 40:

Display the metrics

```
cat("Accuracy of LR model:", glm_accuracy, "\n")
cat("Precision of Knn model:", glm_precision, "\n")
cat("Recall of Knn model:",glm_recall, "\n")
cat("F1-score of Knn mode:", glm_f1_score, "\n")
```