PRACTICAL BUSINESS ANALYTICS REPORT (COMM053)

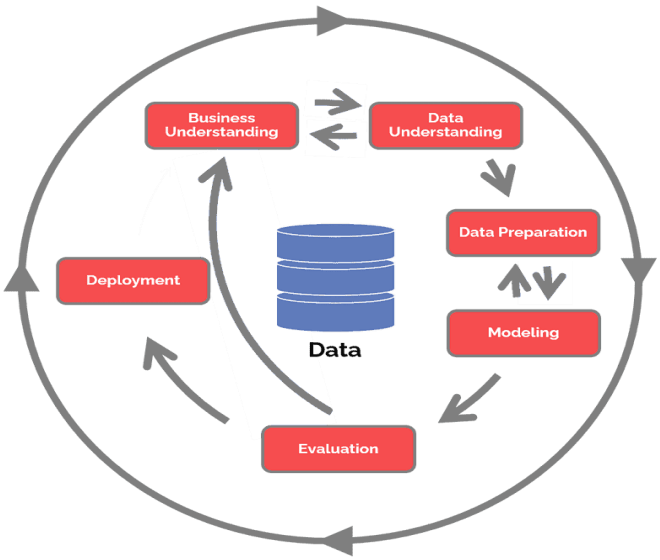
LOAN APPROVAL CLASSIFIER

Group 11

Anu Sabu - 6814704
Luke Menezes - 6803826
Mohit Singh - 6831420
Meenu Nair - 6826907
Harshita Wardhan - 6831456

Table of Content

1.CRISP DM



In our project we used the Cross Industry Standard Process for Data Mining (CRISP-DM) is a process model that serves as the base for data science process. It has six sequential phases:

1. Business understanding – What does the business need?
2. Data understanding – What data do we have / need? Is it clean?
3. Data preparation – How do we organize the data for modelling?
4. Modelling – What modelling techniques should we apply?
5. Evaluation – Which model best meets the business objectives?
6. Deployment – How do stakeholders access the results?

2.PROBLEM DEFINITION (Business understanding)

With this project we are aiming to revolutionize the way banks handle loan applications by incorporating machine learning (ML) automation. The motivation behind this revolution is emerged by the understanding the complexity of handling numerous loan applications at the same time, in this fast-paced world. With this change we are expecting to make the process more efficient, cost-effective, and faster. With a large influx of loan applications, manually reviewing each of them has become impossible. It is not only time-consuming but also prone to human errors. By implementing ML models, we aim to make this process more efficient by defining some predefined criteria with the help of machine learning models.

Along with the time, by this implementation bank can achieve significant profit by saving the cost and by accurate prediction on loan approval. The manual review process includes unnecessary expenditure which includes salary of employee including their time and energy, who are reviewing it. As humans are more prone to error there will be multiple levels in the organization to decide the application needs to approve or reject which further increase the loss. And if the prediction gone wrong, it may affect the employee eventually the bank's stability as well. Automating it through ML can eliminate these risk factors and costs. In addition to that, the introduction of ML enables a faster decision-making process. Traditional review approaches can cause delays in providing decisions to the applicants, and ML models offer a solution by assessing applications through models, which leads to a prompt response to the application. When it comes to the applicant, they might need money immediately and traditional approach may affect their plan.

One of the key benefits we seek to obtain is allow loans to eligible people who have good credit history and thus ameliorate the accuracy in decision-making. Human-based reviews are prone to errors, and maintaining consistency across a large volume of applications is really challenging. Here ML models, learns from the previous history of client which includes several features and can provide a more objective and consistent evaluation of applications, minimizing errors and improving overall accuracy.

Scalability is another goal to achieve. As ML models are scalable, it can efficiently handle fluctuations in application volumes. This adaptability makes sure the system can handle a sudden spike in the loan application without any manual support.

Another vital consideration in this project is the unbiased and transparent decision making. In this project we are generating six different machine learning algorithms, later will decide which one is working good as per the accuracy and other features. This model will make an unbiased and transparent decisions.

Furthermore, we recognize the importance of risk management in the financial industry. ML techniques provide the accurate predictive models, which will eliminate the risk associated with the loan applicant and leads to efficient decision making regarding the issuance loans.

In summary, our project aims to transform the loan application review process, making it more efficient, cost-effective, and transparent through the strategic implementation of machine learning techniques.

3. DATA UNDERSTANDING

The dataset used in this project was obtained from Kaggle, accessible through the following link: Credit Card Approval Prediction. It consists of two datasets, namely "application_record.csv" which contain personal details of applicants and credit_record.sv which includes credit details of each applicant from previous months. The linkage between the two datasets is established via the 'id' column. A thorough understanding of these datasets is important for developing a machine learning model

that predicts the creditworthiness of applicants based on their credit history and personal information.

application_record.csv includes following information.

- ID : The unique identification number assigned to each Client (INTEGER)
- CODE_GENDER : Indicates the gender of the applicant. (M/F)
- FLAG_OWN_CAR : Specifies whether the applicant owns a car. (Y/N)
- FLAG_OWN_REALTY : Indicates whether the applicant owns real estate or Property. (Y/N)
- CNT_CHILDREN : Reflects the number of children.
- AMT_INCOME_TOTAL : Indicates the annual income of the applicant.
- NAME_INCOME_TYPE : Represents the source of income.
- NAME_EDUCATION_TYPE: Indicates the education level of the client.
- NAME_FAMILY_STATUS : Provides information about the marital status of the Property.
- NAME_HOUSING_TYPE : Describes the client's living arrangement, specifying. whether they own property, rent, or reside in another type of housing.
- DAYS_BIRTH : Represents the age of the client, formatted to count backward from the current day (0), where -1 signifies yesterday.
- DAYS_EMPLOYED : Indicates the work experience, counting backward from the current day (0). A positive value implies current unemployment.
- FLAG_MOBIL : Specifies whether the client owns a mobile phone.(Y/N)
- FLAG_WORK_PHONE : Indicates whether the client possesses a work phone. (Y/N).
- FLAG_PHONE : Specifies if the client has a telephone.(Y/N)
- FLAG_EMAIL : Indicates whether the client has an email address.(Y/N)
- OCCUPATION_TYPE : Describes the occupation of the client.
- CNT_FAM_MEMBERS : Specifies the number of family members.

Credit_record.csv includes following information.

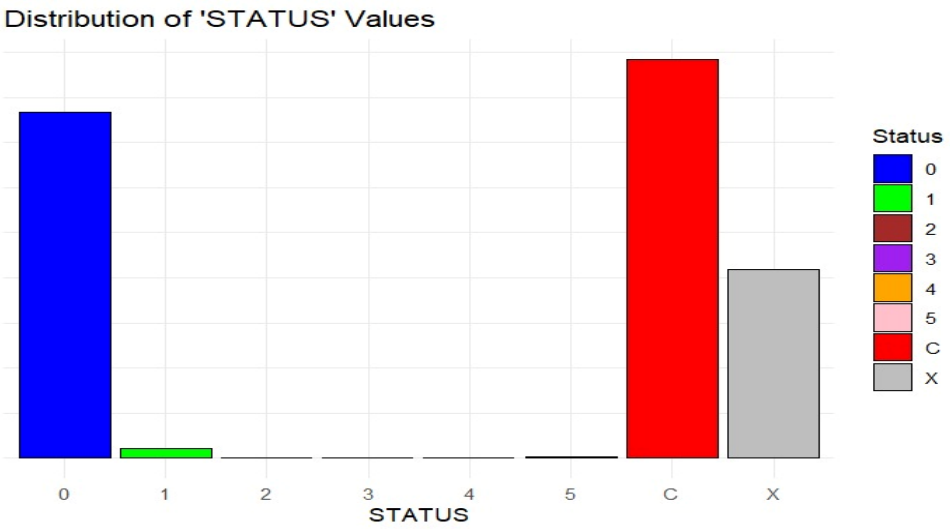
1.ID : The unique identification number assigned to each Client (INTEGER)

2.MONTHS_BALANCE : starting point of the extracted data, counting backward, with 0 indicating the current month, -1 as the previous month, and so forth.

3.STATUS : describes the credit payment status

0: No overdue status, indicating timely and regular payments.
1-29: Indicates the number of days past due, spanning from 1 to 29 days.
1: Reflects payments that are 30-59 days overdue.
2: Designates payments that are 60-89 days overdue.
3: Represents payments that are 90-119 days overdue.
4: Specifies payments that are 120-149 days overdue.
5: Denotes overdue or bad debts, including write-offs for periods exceeding 150 days.
C: Indicates that the credit was fully paid off within that month.
X: Signifies the absence of any loan activity for the given month

The "application_record" dataset comprises 438,557 records and 18 columns, while the "credit_record_csv" dataset includes 1,048,575 records and 3 columns. Both datasets share a common identifier column, "ID," which used to merge these datasets for a holistic analysis.



To enhance efficiency of the data analysis and modelling the datasets were combined into a single file. The resultant dataset encompasses 33,110 records and 16 columns, providing a comprehensive foundation for subsequent analyses and modelling endeavours.

4.DATA EXPLORATION

While exploring the dataset using various Data Exploratory Analysis techniques in R, numerous insightful observations have been uncovered. The analysis is focused on two files, namely, "Application Record.csv" and "Credit.csv," with the "ID" column serving as a common identifier across both datasets.

File 1 - Application Record.csv

Within this dataset, columns such as "DAYS_BIRTH" and "DAYS_EMPLOYED" exhibit negative values, wherein the former counts backward from the current day, considering -1 as yesterday. The latter, "DAYS_EMPLOYED," indicates unemployment when the value is positive. An examination of null values uncovered 134,203 records with missing data in the "OCCUPATION_TYPE" column.

Further analysis delved into categorical columns, including "CODE_GENDER" ,"FLAG_OWN_CAR","FLAG_OWN_REALTY" ,"NAME_INCOME_TYPE" "NAME_EDUCATION_TYPE " ,"NAME_FAMILY_STATUS," and "NAME_HOUSING_TYPE". Unique values within these categories provided insights into the distribution of gender, car ownership, real estate ownership, income type, education level, family status, and housing type among the datasets.

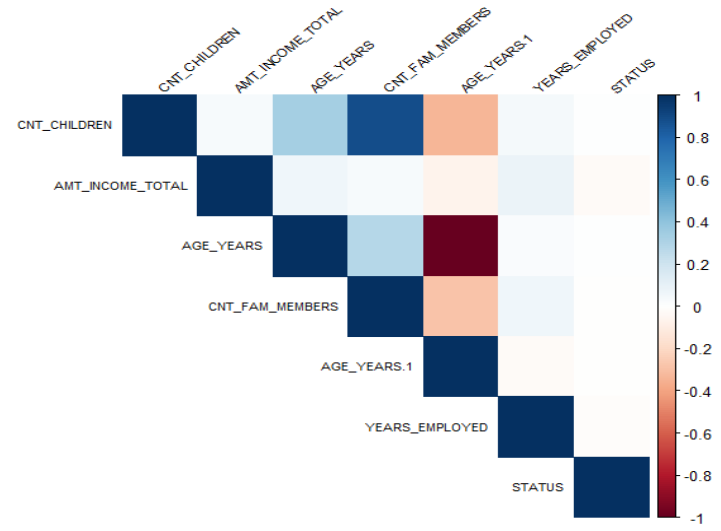
Numeric columns, such as "CNT_CHILDREN" ,"AMT_INCOME_TOTAL" ,"DAYS_BIRTH," "DAYS_EMPLOYED," "FLAG_MOBIL," "FLAG_WORK_PHONE," "FLAG_PHONE," "FLAG_EMAIL," and "CNT_FAM_MEMBERS," were also explored. Distinct values for "CNT_CHILDREN" and the range of "DAYS_BIRTH" were identified, while other binary columns like "FLAG_WORK_PHONE," "FLAG_PHONE," and "FLAG_EMAIL" revealed the distribution of positive and negative instances.

File 2 - Credit Record.csv

Moving on to the "Credit Record.csv" file, the "MONTHS_BALANCE" column signifies the temporal dimension of the data, with 0 representing the current month and negative values indicating previous months. The "STATUS" column categorizes the credit status, providing information on overdue payments, completed payments, and months without a loan. These findings lay the groundwork for a comprehensive understanding of the datasets, paving the way for subsequent analyses and insights in the report. We decided to choose STATUS as our target variable.

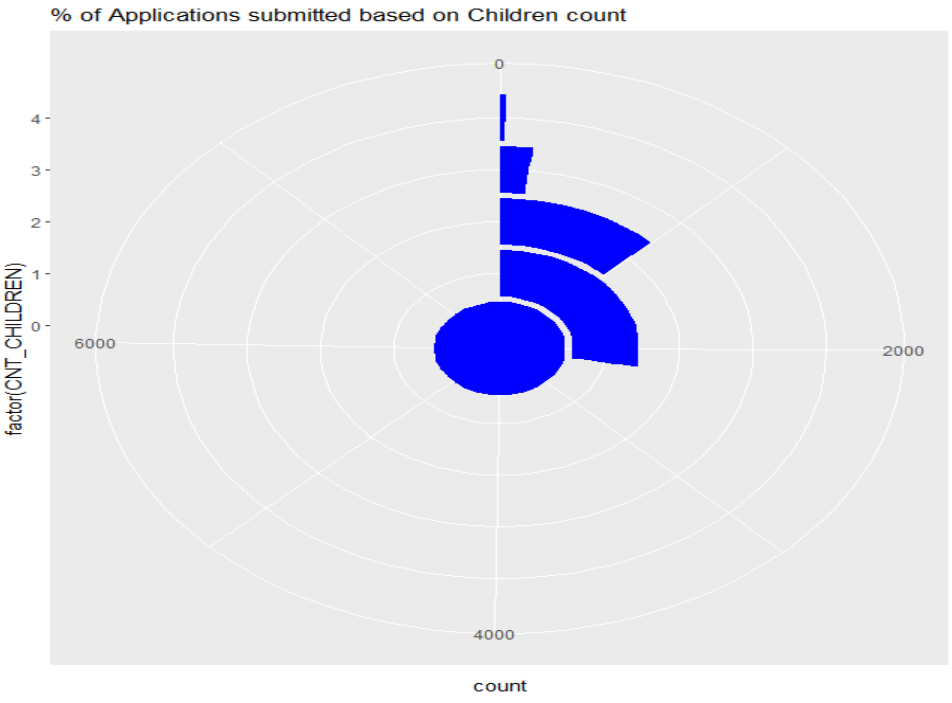
Heat Map Matrix

In the field of machine learning, a heat map serves as a visual depiction of data, portraying matrix values through a spectrum of colours.

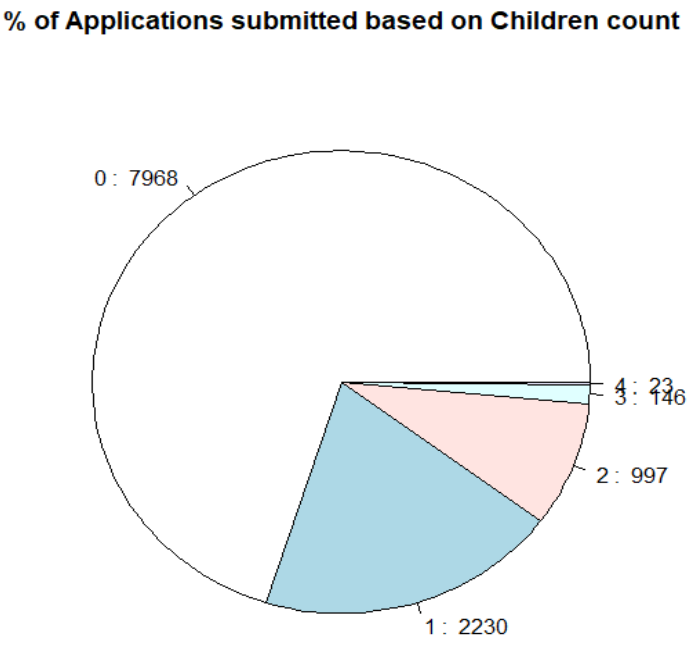


Representation of Merged Dataset

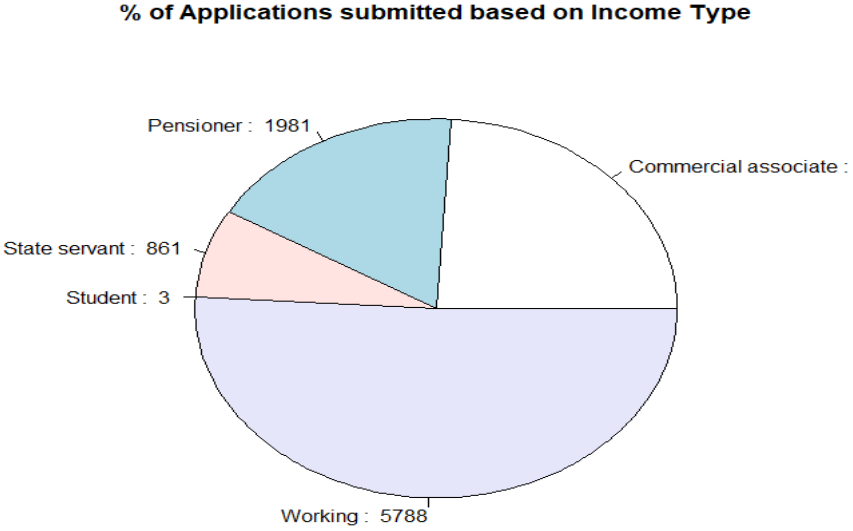
Creating Pie chart to check If the distribution of the number of children is imbalanced (e.g., most clients have no children, and a few have many children), a pie chart can visually highlight this imbalance.



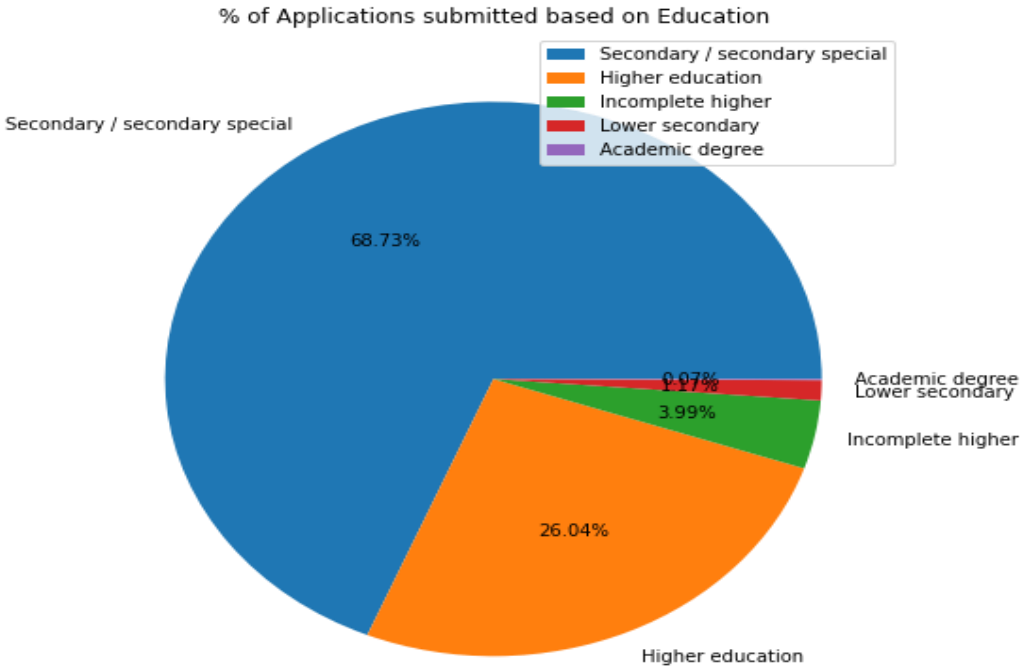
This pie chart should help identify and visualize the distinct categories or unique values within the CNT_CHILDREN variable.



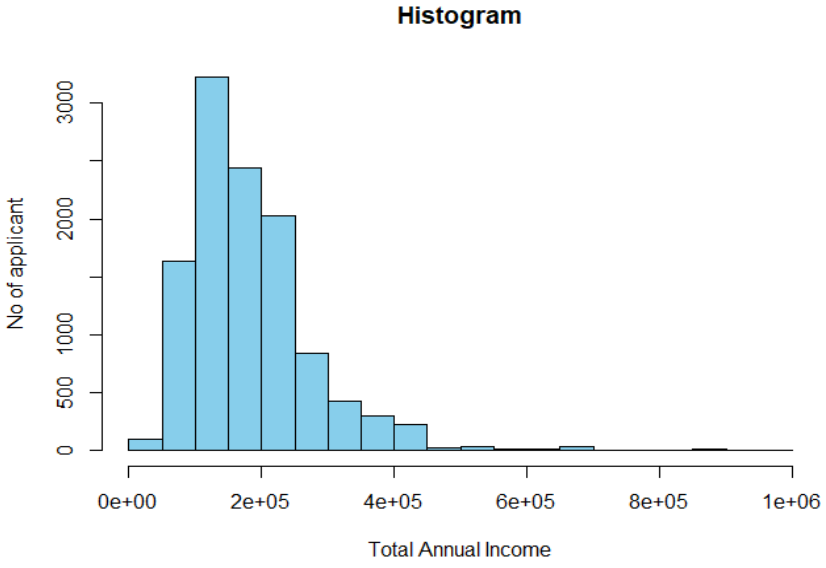
Creating a pie chart for the NAME_INCOME_TYPE variable can offer valuable insights into the distribution of income types among applicants. This pie chart is telling us about the number of applications submitted based on income type.



Creating a pie chart to represent the percentage of people who submitted applications based on their education level can be a visually appealing and informative way to convey insights. You can easily compare the distribution of applicants across various education categories, identifying which levels are more prevalent.



A histogram provides a visual representation of the distribution of total annual income within families. This can help you understand the range and pattern of income levels. Histograms can help identify outliers or extreme values in the distribution, which may have a significant impact



on the overall statistics.

5.DATA PRE-PROCESSING

In this section, we will go through the steps we took to transform the initial datasets to our final dataframe used as input for our classification models.

We start with our initial data, in the format of two CSV files from Kaggle. These have been imported into our R workspace and we can view the initial structure of both dataframes.

```{r}
glimpse(applications)
```

```
Rows: 438,557
Columns: 18
$ ID                 <int> 5008804, 5008805, 5008806, 5008808, 5008809, 5008810, 5008811, 5008812, 5008813, 5…
$ CODE_GENDER        <chr> "M", "M", "M", "F", "F", "F", "F", "F", "F", "F", "M", "M", "M", "M", "M", "M", "M…
$ FLAG_OWN_CAR       <chr> "Y", "Y", "Y", "N", "N", "N", "N", "N", "N", "N", "Y", "Y", "Y", "Y", "Y", "Y", "Y…
$ FLAG_OWN_REALTY    <chr> "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y…
$ CNT_CHILDREN       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 3…
$ AMT_INCOME_TOTAL   <dbl> 427500, 427500, 112500, 270000, 270000, 270000, 270000, 283500, 283500, 283500, 27…
$ NAME_INCOME_TYPE   <chr> "Working", "Working", "Working", "Commercial associate", "Commercial associate", "…
$ NAME_EDUCATION_TYPE <chr> "Higher education", "Higher education", "Secondary / secondary special", "Secondar…
$ NAME_FAMILY_STATUS <chr> "Civil marriage", "Civil marriage", "Married", "Single / not married", "Single / n…
$ NAME_HOUSING_TYPE  <chr> "Rented apartment", "Rented apartment", "House / apartment", "House / apartment", …
$ DAYS_BIRTH         <int> -12005, -12005, -21474, -19110, -19110, -19110, -19110, -22464, -22464, -22464, -1…
$ DAYS_EMPLOYED      <int> -4542, -4542, -1134, -3051, -3051, -3051, -3051, 365243, 365243, 365243, -769, -76…
$ FLAG_MOBIL         <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
$ FLAG_WORK_PHONE    <int> 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
$ FLAG_PHONE         <int> 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0…
$ FLAG_EMAIL         <int> 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
$ OCCUPATION_TYPE    <chr> NA, NA, "Security staff", "Sales staff", "Sales staff", "Sales staff", "Sales staf…
$ CNT_FAM_MEMBERS    <dbl> 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 5…
```

As seen above, the applications dataframe, imported from applications.csv, has 438,557 rows and 18 columns

```{r}
glimpse(credit)
```

```
Rows: 1,048,575
Columns: 3
$ ID             <int> 5001711, 5001711, 5001711, 5001711, 5001712, 5001712, 5001712, 5001712, 5001712, 500171…
$ MONTHS_BALANCE <int> 0, -1, -2, -3, 0, -1, -2, -3, -4, -5, -6, -7, -8, -9, -10, -11, -12, -13, -14, -15, -16…
$ STATUS         <chr> "X", "0", "0", "0", "C", "C", "C", "C", "C", "C", "C", "C", "C", "0", "0", "0", "0", "0…
```

The other dataframe, credit, has 1048575 rows and 3 columns. The common column between both dataframes, ID, can be observed. It can also be seen that the credit dataframe has multiple of the same ID, as it represents multiple months' data of each users' history.

To start pre-processing, we check for the null values in each dataframe. We convert these null values to NAs, which are easier to handle in R.

We find the number of NAs in each column:

```
                ID          CODE_GENDER          FLAG_OWN_CAR      FLAG_OWN_REALTY 
                 0                    0                     0                    0 
      CNT_CHILDREN     AMT_INCOME_TOTAL     NAME_INCOME_TYPE NAME_EDUCATION_TYPE 
                 0                    0                     0                    0 
NAME_FAMILY_STATUS    NAME_HOUSING_TYPE           DAYS_BIRTH       DAYS_EMPLOYED 
                 0                    0                     0                    0 
       FLAG_MOBIL      FLAG_WORK_PHONE           FLAG_PHONE          FLAG_EMAIL 
                 0                    0                     0                    0 
   OCCUPATION_TYPE      CNT_FAM_MEMBERS 
            134203                    0 
            ID MONTHS_BALANCE         STATUS 
             0              0              0 
```

As seen above, only the OCCUPATION_TYPE column has NA values. This will be handled later.

Next, we merge the two datasets into a single dataframe, on the common column ID

```
Rows: 777,715
Columns: 20
$ ID                 <int> 5008804, 5008804, 5008804, 5008804, 5008804, 5008804, 5008804, 5008804, 5008804, 5…
$ CODE_GENDER        <chr> "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M…
$ FLAG_OWN_CAR       <chr> "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y…
$ FLAG_OWN_REALTY    <chr> "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y…
$ CNT_CHILDREN       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
$ AMT_INCOME_TOTAL   <dbl> 427500, 427500, 427500, 427500, 427500, 427500, 427500, 427500, 427500, 427500, 42…
$ NAME_INCOME_TYPE   <chr> "Working", "Working", "Working", "Working", "Working", "Working", "Working", "Work…
$ NAME_EDUCATION_TYPE <chr> "Higher education", "Higher education", "Higher education", "Higher education", "H…
$ NAME_FAMILY_STATUS <chr> "Civil marriage", "Civil marriage", "Civil marriage", "Civil marriage", "Civil mar…
$ NAME_HOUSING_TYPE  <chr> "Rented apartment", "Rented apartment", "Rented apartment", "Rented apartment", "R…
$ DAYS_BIRTH         <int> -12005, -12005, -12005, -12005, -12005, -12005, -12005, -12005, -12005, -12005, -1…
$ DAYS_EMPLOYED      <int> -4542, -4542, -4542, -4542, -4542, -4542, -4542, -4542, -4542, -4542, -4542, -4542…
$ FLAG_MOBIL         <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
$ FLAG_WORK_PHONE    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
$ FLAG_PHONE         <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
$ FLAG_EMAIL         <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
$ OCCUPATION_TYPE    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
$ CNT_FAM_MEMBERS    <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
$ MONTHS_BALANCE     <int> -6, -4, -1, -5, -11, -12, -13, -14, -15, 0, -7, -2, -3, -10, -8, -9, -13, -4, -7, …
$ STATUS             <chr> "C", "C", "C", "C", "C", "C", "1", "0", "X", "C", "C", "C", "C", "C", "C", "C", "0…
```

After merging, there are 777,715 rows and 20 columns.

We can look at the distribution of our target variable, STATUS, now.

```
     0      1      2      3      4      5      C      X
290654   8747    801    286    214   1527 329536 145950
```

We aim to process this column for binary classification.

As mentioned previously, the X value is removed from the column, as it represents months where the user did not request a loan.

We filter further by taking a random single entry from every user, to prevent the classifiers from being trained on the same user data twice.

Finally, we transform the column to a binary variable. 0, 1, 2, 3, 4 and 5 are considered “bad” months, where the user did not pay or is overdue on their loan, so all those values will become 0 (no loan). The months where the value was C, which means the user paid off their loan, will become 1 (loan approved).

```
    0     1
21315 11795
```

We note that the variable is unbalanced, with many more 0s than 1s, which could cause problems when modelling.

Now that we are finished processing the target variable, we move on to the predictor variables.

We decided that 19 predictors would lead to a too complex model, so some were removed. Each removed column will be explained below:

Column Removed	Reasoning
ID	Only useful for merging dataframes. No predictive power beyond that.
CNT_CHILDREN	Effectively the same information as CNT_FAM_MEMBERS. Both represent size of family, so only one is necessary.
FLAG_MOBIL	We considered the all the mobile/phone columns to have the least predictive power on loan applications, as they were similar binary columns, so they were removed.
FLAG_WORKPHONE	We considered the all the mobile/phone columns to have the least predictive power on loan applications, as they were similar binary columns, so they were removed.

FLAG_PHONE	We considered the all the mobile/phone columns to have the least predictive power on loan applications, as they were similar binary columns, so they were removed.
OCCUPATION_TYPE	The only column with null values. With no effective way to handle them, it was best to remove the column
MONTHS_BALANCE	Only useful to track user progression. Not useful when each user has one value, so it was removed.

For the remaining columns, we start the processing by finding which of the predictors are numerical and which are not.

Numerical:

```
[1] "AMT_INCOME_TOTAL" "DAYS_BIRTH"       "DAYS_EMPLOYED"    "FLAG_EMAIL"       "CNT_FAM_MEMBERS"
[6] "STATUS"
```

Non-numerical:

```
[1] "CODE_GENDER"          "FLAG_OWN_CAR"         "FLAG_OWN_REALTY"     "NAME_INCOME_TYPE"
[5] "NAME_EDUCATION_TYPE" "NAME_FAMILY_STATUS"  "NAME_HOUSING_TYPE"
```

The non-numeric columns are all considered categorial and will be converted into factors, a data type in R for categorical values.

For the numerical columns, further testing is required to find if they are continuous or categorical. We can find this by calculating the number of unique values in each column. If there are more than 10, we can consider the column continuous.

Below we see the number of unique values in the AMT_INCOME_TOTAL, DAYS_BIRTH, DAYS_EMPLOYED, FLAG_EMAIL and CNT_FAMILY_MEMBERS columns. Note that STATUS was excluded since it was already processed.

```
[1] 258
[1] 6960
[1] 3566
[1] 2
[1] 10
```

As seen, the first three columns (AMT_INCOME_TOTAL, DAYS_BIRTH, DAYS_EMPLOYED), have far more than 10 unique values, so are considered continuous, while the other two were considered categorical.

After we found the categorical columns, we converted them to factors. For the CNT_FAMILY_MEMBERS column, which had 10 unique values, (number of family members from 0-10), we decided to group all families with 4+ members in a single value, so the model could process it easier. The NAME_EDUCATION_TYPE column also has ordered levels of education, which were added in the factor levels.

Next, we process the numerical columns, which include AMT_INCOME_TOTAL, DAYS_BIRTH, and DAYS_EMPLOYED. The latter two are stored in negative days, so we converted them to years and stored them in a positive format. After this, all three numerical columns were normalised, as models process data in a similar scale better.

We then decided to change the names of most of the columns for better viewing. The changes are described below:

Old name	New name
CODE_GENDER	GENDER
FLAG_OWN_CAR	OWN_CAR
FLAG_OWN_REALTY	OWN_REALTY
NAME_INCOME_TYPE	INCOME_TYPE
NAME_EDUCATION_TYPE	EDUCATION_TYPE
NAME_FAMILY_STATUS	FAMILY_STATUS
NAME_HOUSING_TYPE	HOUSING_TYPE
FLAG_EMAIL	EMAIL
CNT_FAM_MEMBERS	FAMILY_SIZE
AMT_INCOME_TOTAL	INCOME
DAYS_BIRTH	AGE
DAYS_EMPLOYED	YEARS_EMPLOYED
STATUS	STATUS

This concludes the pre-processing. We are left with 12 predictor variables and 1 target variable. All variables will be described in a table format below.

Name of column	Categorical / Continuous	Description
GENDER	Categorical	Gender of potential borrower. Useful to find gender bias in loan approval.
OWN_CAR	Categorical	Whether the user owns a car. Good indicator of wealth.

OWN_REALTY	Categorical	Whether the user owns housing. Good indicator of wealth.
INCOME_TYPE	Categorical	The user's kind of employment. Useful to show how financial stability.
EDUCATION_TYPE	Categorical	The user's highest level of education. More educated people may tend to get loans easier.
FAMILY_STATUS	Categorical	The user's relationship status. Married people tend to be more stable financially.
HOUSING_TYPE	Categorical	The kind of housing the user lives in. People living in houses tend to be wealthier.
EMAIL	Categorical	Whether the user has a registered email. Could also be an indicator of age.
FAMILY_SIZE	Categorical	The size of the user's family.
INCOME	Continuous	The income of the user.
AGE	Continuous	How old the user is. Older people may be more financially stable.
YEARS_EMPLOYED	Continuous	The number of years the user has been employed.
STATUS	Target Variable	The target variable. 0 means no loan, 1 means loan approved

As seen above, we end with 3 continuous predictors and 9 categorical predictors.

6.DATA MODELLING

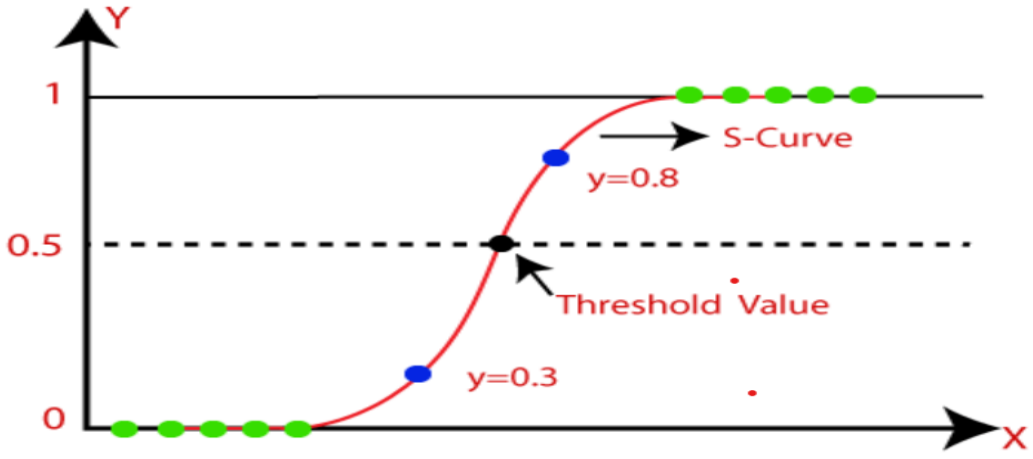
Since the target attribute (Status) is a categorical field, it is ideal to use the classification models to predict and evaluate the results. Below is the list of classification models that are trained, tested and evaluated in the process of predicting that the client is good client or bad client.

6.1LOGISTIC REGRESSION WITH K-FOLD CROSS VALIDATION-ANU

A logistic regression model is employed for data fitting, opting for K-fold cross-validation instead of the traditional hold-out approach (e.g., 70-30 or 80-20). This decision stems from the potential issues in the latter method, where the test dataset might be either too easy or too rare, leading to poor model performance. K-fold cross-validation is a more sophisticated alternative, involving random sub-sampling by using the entire dataset for both training and testing. The dataset is divided into K folds, and each fold predicts an output, with the final output being the sum of individual folds. In this case, the dataset is split into 6 folds.

Logistic regression, employed in this context, calculates the odds ratio in the presence of multiple predictor variables. Like other linear regression methods, logistic regression deals with a binomial response variable, providing insights into the relationship between a dependent binary variable and one or more independent variables of nominal, ordinal, interval, or ratio levels.

Logistic regression is a straightforward and commonly used machine learning algorithm for two-class classification, applicable to any binary categorization. Its foundational concepts are also valuable in deep learning. The primary goal is to predict the binary class using statistical methods, resulting in a dichotomous output with only two possible classes. Logistic regression assesses the likelihood of an event occurrence, offering a continuous output.



The logistic regression model utilizes the sigmoid function to map predicted values to probabilities. The sigmoid function produces an 'S'-shaped curve, restricting predicted values between 0 and 1. This curve aids in classifying the target variable into distinct classes. The sigmoid equation, y=1/(1+e^x), signifies that predicted values approach zero for significant negative x values and approach one for large positive x values. A decision boundary can be set to categorize data into classes based on a threshold.

In logistic regression, data is divided into classes, and a decision boundary is used to separate these classes. The decision boundary seeks to optimize by maximizing the distance between classes. A chosen threshold value (e.g., 0.7,0.5) classifies data based on whether the label value crosses the threshold. Logistic regression, a statistical model, excels in classifying observations across various data types, identifying influential variables for classification. The sigmoid function visually represents the algorithm's predictive nature.

Logistic Function (Sigmoid Function): The sigmoid function maps predicted values to probabilities, ensuring they fall within the range of 0 and 1. This function, resembling an 'S' curve, is crucial in logistic regression, as it shapes the output to be confined within the specified limits.

Assumptions for Logistic Regression:

1. The dependent variable must be categorical.
2. The independent variable should not exhibit multicollinearity.

Logistic Regression Equation: Derived from the linear regression equation, the logistic regression equation is refined to ensure the output falls within the range of 0 to 1. The final equation is achieved by taking the logarithm of the adjusted linear regression equation.

Types of Logistic Regression: Logistic regression can be categorized into three types based on the nature of the dependent variable:

1. Binomial: Two possible types of dependent variables (e.g., 0 or 1).
2. Multinomial: Three or more possible unordered types of the dependent variable.
3. Ordinal: Three or more possible ordered types of the dependent variable.

In summary, logistic regression proves to be a versatile and widely used algorithm for classification tasks, particularly when dealing with categorical target variables. It employs the sigmoid function to map predicted values to probabilities, ensuring a well-defined decision boundary between classes. The logistic regression equation is derived from the linear regression equation, and the algorithm's effectiveness is enhanced through techniques like K-fold cross-validation.

The following Steps were used in the Logistic Regression classifier modelling

To ensure the integrity of the data, an initial step involves creating a duplicate of the original dataframe, labeled as new_df. The subsequent training process is established with a robust 10-fold cross-validation framework using the trainControl, denoted as ctrl_lg. Subsequently, the logistic regression model is trained via the train function from

the caret package. This model incorporates the elastic net regularization technique, discernible through the "glmnet" method and "binomial" family parameters. The model's hyperparameters, specifically the mixing parameter alpha and the regularization parameter lambda, undergo tuning via a systematic grid search utilizing predefined sequences. The resulting cross-validated model is then printed to facilitate an examination of its performance metrics.

After the model training phase, predictions are generated using the trained logistic regression model on the identical dataset. To gauge the model's predictive accuracy, a confusion matrix is computed, furnishing a detailed breakdown encompassing true positive, true negative, false positive, and false negative predictions. This confusion matrix proves pivotal for a comprehensive evaluation of the model's classification performance, with key metrics such as accuracy, sensitivity, specificity, positive predictive value, negative predictive value, prevalence, and others extracted from its results.

Furthermore, the code integrates the construction and visualization of a Receiver Operating Characteristic (ROC) curve. Utilizing the logistic regression model's predicted probabilities, the ROC curve is plotted, and the Area Under the Curve (AUC) is calculated. The AUC serves as a critical metric for evaluating the model's discriminatory power. In the specific scenario outlined, the AUC is printed and interpreted. An AUC value of 0.525 suggests that the model's discriminatory ability does not significantly surpass random chance. This observation prompts contemplation of further model refinement or additional investigations to enhance predictive performance. In essence, the provided code furnishes a comprehensive workflow, encompassing model training, evaluation, and visual assessment of the logistic regression model's performance.

Confusion Matrix: The confusion matrix provides essential insights into the model's performance, allowing us to compute various performance metrics, including True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN), accuracy, sensitivity, specificity, positive predictive value, negative predictive value, prevalence, detection rate, F1-score, and balanced accuracy.

- **True Positives (TP):** The model correctly predicted class 1 in 25 instances.
- **False Positives (FP):** The model incorrectly predicted class 1 when the true class was 0, resulting in 11,703 instances.
- **True Negatives (TN):** The model correctly predicted class 0 in 21,367 instances.
- **False Negatives (FN):** The model incorrectly predicted class 0 when the true class was 1, resulting in 15 instances.

Performance Metrics:

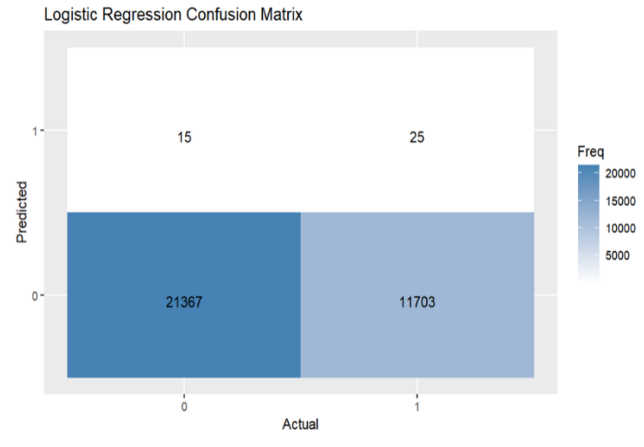
- **Accuracy:** The proportion of correctly classified instances
- **Sensitivity (Recall):** The proportion of actual positive instances correctly predicted by the model
- **Positive Predictive Value (Precision):** The proportion of predicted positive instances that are actually positive.

- **Negative Predictive Value:** The proportion of predicted negative instances that are actually negative.
- **Prevalence:** The proportion of actual positive instances
- **Rate (True Positive Rate):** The proportion of actual positive instances detected by the model.
- **F1-Score:** The harmonic mean of precision and recall
- **Balanced Accuracy:** The average of sensitivity and specificity is approximately 0.50.

Interpretation:

- The accuracy suggests that 64.61% of instances are correctly classified by the model.
- The model exhibits high sensitivity, correctly identifying nearly all positive instances.
- Specificity is extremely low, indicating challenges in correctly predicting negative instances.
- Precision is relatively low, emphasizing that positive predictions have a limited accuracy.
- The F1-score is notably low, highlighting the model's struggle to balance precision and recall.

In summary, the model demonstrates notable shortcomings, particularly in achieving a balance between precision and recall, as reflected in the low F1-score. The low specificity further indicates challenges in correctly identifying negative instances. Further model refinement or exploration of alternative rformance. approaches may be necessary for improved performance

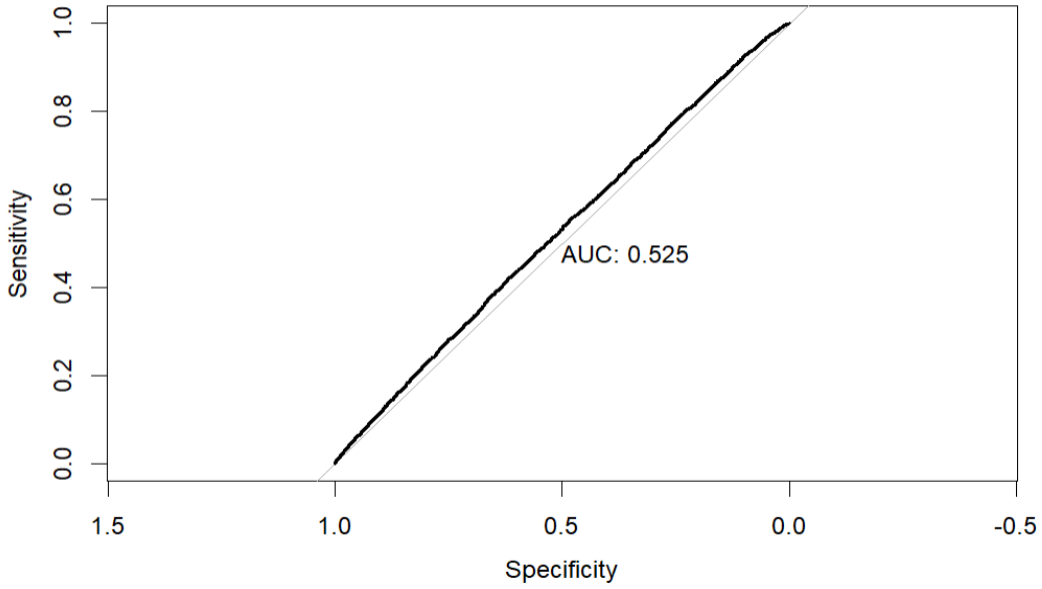


```
[1] "Confusion Matrix Of Logistic Regression:"
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 21367 11703
         1    15    25

               Accuracy : 0.6461
                 95% CI : (0.6409, 0.6512)
    No Information Rate : 0.6458
    P-Value [Acc > NIR] : 0.4568

                  Kappa : 0.0018

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.999298
            Specificity : 0.002132
         Pos Pred Value : 0.646114
         Neg Pred Value : 0.625000
             Prevalence : 0.645787
         Detection Rate : 0.645334
   Detection Prevalence : 0.998792
      Balanced Accuracy : 0.500715
```

AUC

The AUC (area under the curve) is relatively low, around 0.525, indicating that the model has limited effectiveness in distinguishing between positive and negative classes. An AUC of 0.5 would imply performance no better than random guessing.

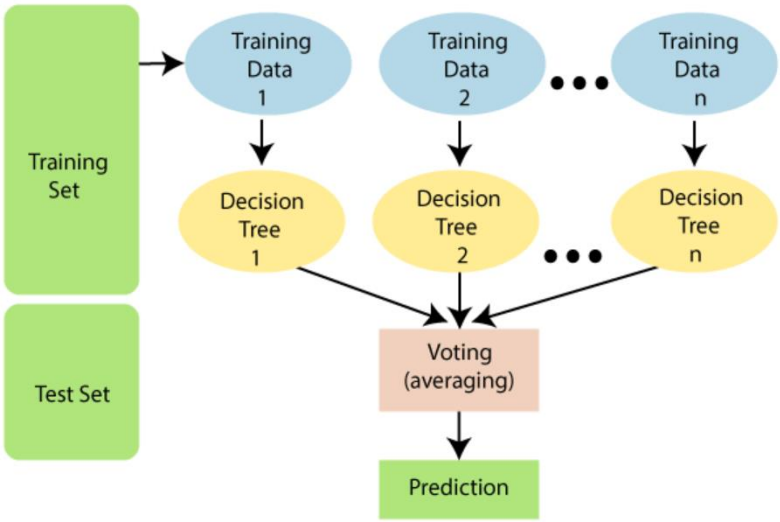
Additionally, the curve closely follows the diagonal line, suggesting suboptimal discriminative ability. Ideally, we would prefer the curve to bend towards the top left corner, indicating better overall performance. The steep ascent at the beginning of the curve highlights the model's strength in identifying true negatives (specificity), but it also reveals a challenge in accurately detecting positives (recall/sensitivity).



6.2 Random Forest-ANU

The Random Forest algorithm is a powerful ensemble learning method employed in machine learning for both classification and regression tasks. This technique leverages the collective intelligence of multiple decision trees, forming a forest of trees, to enhance classification and regression accuracy. By creating diverse subsets of the data for each tree, the algorithm addresses overfitting issues and improves accuracy by combining the outputs of all trees.

Individual decision trees within the Random Forest focus on both variables and observations in the training data. Utilizing bagging techniques, the model selects random observations and considers only those columns that demonstrate significance at the root of decision trees. This interdependence among trees enhances accuracy and mitigates overfitting. A practical rule is applied for choosing subsamples, such as taking the square root of the number of columns for classification and one-third of the columns for regression when considering two-thirds of observations for training.



Key advantages of Random Forest include its use of ensemble learning, based on a bagging algorithm, effectively addressing overfitting in decision trees. It is versatile, applicable to both classification and regression problems, and automated handling of missing values in the data. However, achieving higher accuracy with Random Forest requires substantial computational resources, as the model involves creating numerous trees and demands more training time.

The working principle of Random Forest involves selecting random data points, building decision trees associated with those points, and repeating the process for a specified number of trees. Predictions from each tree are combined to make final predictions based on majority votes.

In practical applications, Random Forest finds utility in diverse sectors such as banking for loan risk identification, medicine for disease trend analysis, land use classification, and marketing trend identification. The algorithm's advantages include its capability to handle large datasets, high dimensionality, and prevention of overfitting issues. However, it may not be as suitable for regression tasks.

In summary, Random Forest stands out as a robust algorithm that harnesses the strengths of ensemble learning, making it a valuable tool for various machine learning applications.

Confusion Matrix

The confusion matrix and accompanying metrics offer insights into the Random Forest model's performance in a classification task:

True Positives (TP): 6790 instances correctly predicted as Class 1.

False Positives (FP): 2230 instances incorrectly predicted as Class 1.

True Negatives (TN): 19152 instances correctly predicted as Class 0.

False Negatives (FN): 4938 instances incorrectly predicted as Class 0.

Now, let's explore the performance metrics:

Accuracy (0.7835): This metric reflects the ratio of correctly classified instances, with the model achieving 78.35% accuracy.

Sensitivity/Recall (0.8957): Sensitivity gauges the model's aptitude in correctly predicting actual positive instances. Here, the model achieved a high sensitivity of 89.57%, capturing a significant portion of true positive cases.

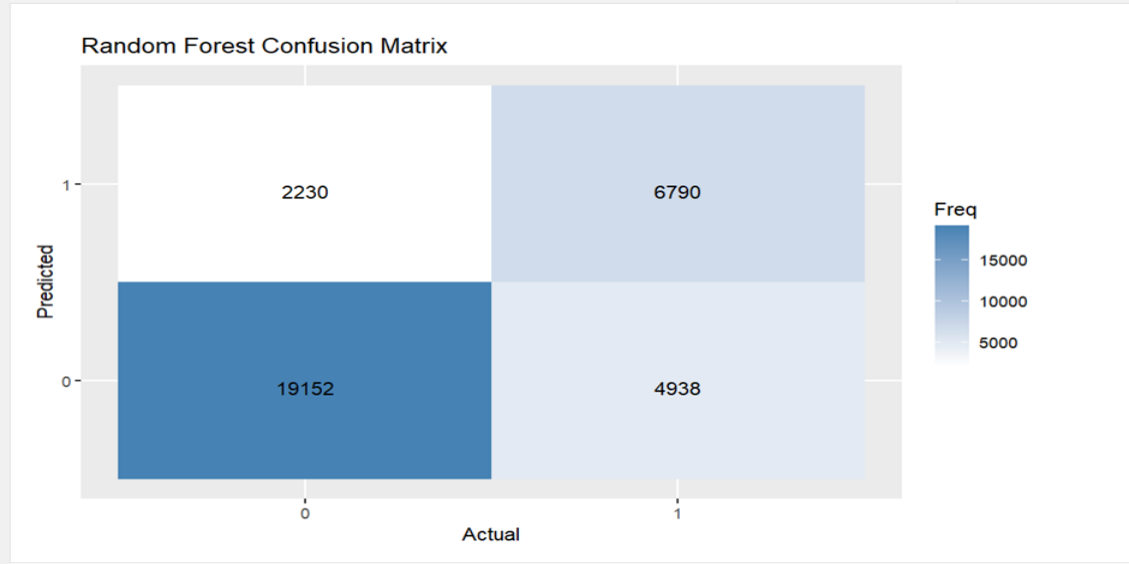
Specificity (0.5790): Specificity measures the model's accuracy in predicting actual negative instances. The model achieved a specificity of 57.90%, indicating room for enhancement in identifying negative instances accurately.

Positive Predictive Value/Precision (0.7950): Precision assesses the reliability of positive predictions. The model demonstrated a precision of 79.50%, signifying the trustworthiness of positive predictions.

Negative Predictive Value (0.7528): This metric evaluates the reliability of negative predictions, with the model achieving a Negative Predictive Value of 75.28%.

Balanced Accuracy (0.7373): Balanced Accuracy, averaging sensitivity and specificity, provides a holistic measure of overall model performance.

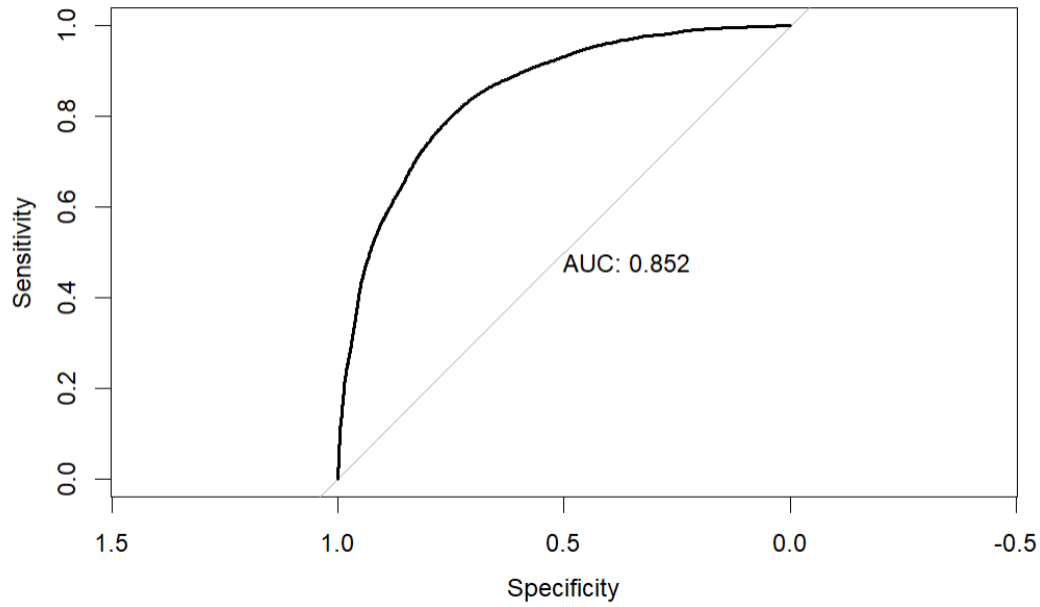
Kappa statistic (0.5008): The Kappa statistic evaluates the agreement between predicted and observed classifications, with a perfect agreement indicated by a value of 1.



```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 19152  4938
         1  2230  6790

               Accuracy : 0.7835
                 95% CI : (0.779, 0.7879)
    No Information Rate : 0.6458
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5008

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8957
            Specificity : 0.5790
         Pos Pred Value : 0.7950
         Neg Pred Value : 0.7528
             Prevalence : 0.6458
         Detection Rate : 0.5784
   Detection Prevalence : 0.7276
      Balanced Accuracy : 0.7373

       'Positive' Class : 0

[1] "Accuracy: 78.35%"
[1] "Confusion Matrix:"
          Reference
Prediction     0     1
         0 19152  4938
         1  2230  6790
```

AUC (0.852): The Area Under the Curve (AUC) from the Receiver Operating Characteristic (ROC) curve indicates strong discriminatory power. The ROC curve showcases the trade-off between sensitivity and specificity at different thresholds, and a higher AUC denotes superior performance.

In summary, the Random Forest model demonstrates strong accuracy and sensitivity, but there is room for improvement in specificity. The AUC of 0.852 indicates robust discriminatory ability, positioning the model as promising for the given classification task. The confusion matrix offers a detailed breakdown of predictions, facilitating comprehension of the model's strengths and areas for refinement.



6.3 Naive Bayes algorithm -Harshita

The Naive Bayes algorithm is a probabilistic classification method that is based on Bayes' theorem, particularly naive Bayes' assumption of independence between features. It is a simple and efficient algorithm widely used for classification tasks, especially in situations where the dataset has a large number of features. Despite its simplicity, Naive Bayes can provide competitive results in various applications.

Model Prediction:

Predictions are made on the test set using the trained Naive Bayes model.
Predictions = predict (train_df_balance , newdata = test_df_features)
predictions=predict(train_df_balanced,newdata=test_df_features)

Model Evaluation:

Accuracy is calculated as the ratio of correctly classified instances to the total instances in the test set.
Accuracy is calculated based on the predictions.

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Instances}} \times 100$$

Confusion Matrix

A confusion matrix is generated to illustrate true positive (TP), true negative (TN), false positive (FP), and false negative (FN) instances.

$$\text{Confusion Matrix} = \begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix}$$

Classification Report:

- True Positives (TP): 4234 instances correctly predicted as Class 0.
- False Positives (FP): 3351 instances incorrectly predicted as Class 1.
- True Negatives (TN): 175 instances correctly predicted as Class 1.
- False Negatives (FN): 173 instances incorrectly predicted as Class 0.

A detailed classification report is printed, including metrics such as accuracy, sensitivity, specificity, positive predictive value, negative predictive value, prevalence, and balanced accuracy.

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6234 3351
         1  173  175

               Accuracy : 0.6452
                 95% CI : (0.6357, 0.6546)
    No Information Rate : 0.645
    P-Value [Acc > NIR] : 0.4879

                  Kappa : 0.0284

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.97300
            Specificity : 0.04963
         Pos Pred Value : 0.65039
         Neg Pred Value : 0.50287
             Prevalence : 0.64502
         Detection Rate : 0.62760
   Detection Prevalence : 0.96497
      Balanced Accuracy : 0.51131

       'Positive' Class : 0
```

Overall, your report provides a thorough analysis of the Naive Bayes model's performance, facilitating a comprehensive understanding of its behavior in the given context.

$$\text{Sensitivity} = \frac{TP}{TP+FN}$$
$$\text{Specificity} = \frac{TN}{TN+FP}$$
$$\text{Precision} = \frac{TP}{TP+FP}$$
$$\text{Negative Predictive Value} = \frac{TN}{TN+FN}$$
$$\text{Balanced Accuracy} = \frac{\text{Sensitivity}+\text{Specificity}}{2}$$

- Kappa measures the agreement between predicted and observed classifications.

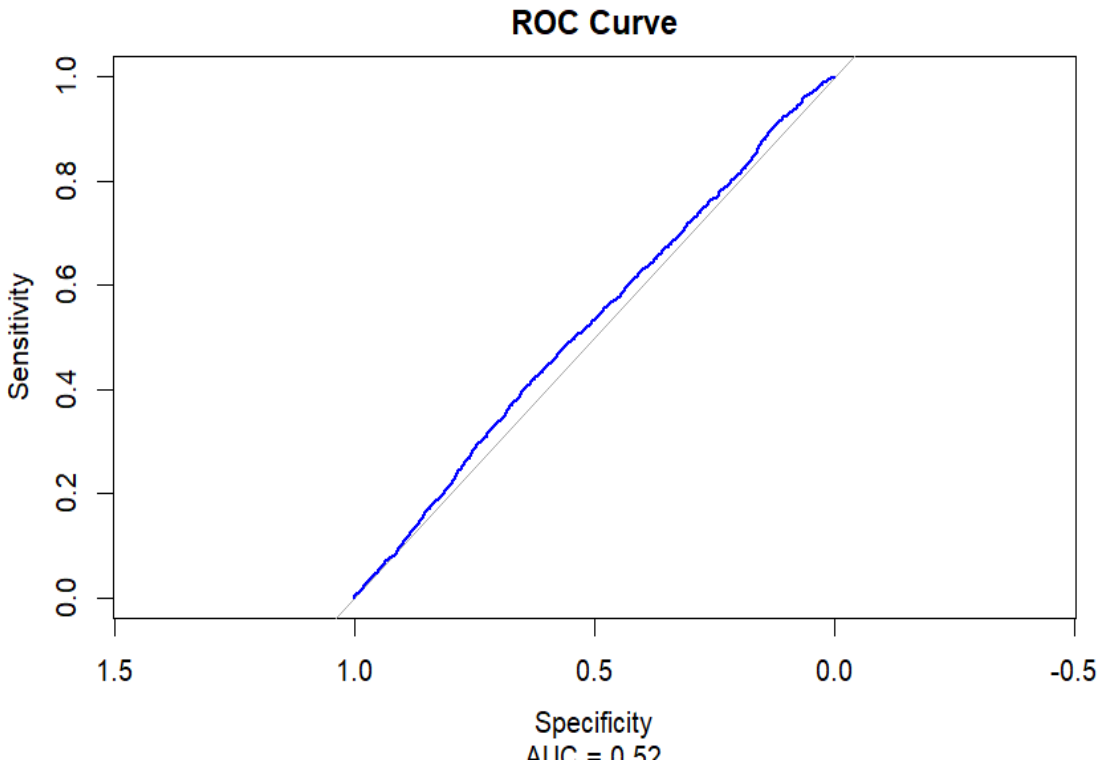
$$\text{Kappa} = \frac{\text{Observed Agreement} - \text{Expected Agreement}}{1 - \text{Expected Agreement}}$$

- AUC (Area Under the Curve) from the ROC (Receiver Operating Characteristic) curve signifies discriminatory power.

$$\text{AUC} = \text{Area Under the ROC Curve}$$

- Sensitivity: The ability to correctly predict actual positive instances (Class 0) is very high (close to 1).
- Specificity: The ability to correctly predict actual negative instances (Class 1) is very low.
- Pos Pred Value: The positive predictive value is the proportion of positive instances that were correctly predicted.
- Neg Pred Value: The negative predictive value is the proportion of negative instances that were correctly predicted.
- Prevalence: The prevalence of the positive class in the dataset.
- Balanced Accuracy: The average of sensitivity and specificity.

Interpretation:
- The Naive Bayes model performs well in terms of sensitivity, correctly identifying instances of Class 0.
- However, specificity is low, indicating challenges in correctly predicting instances of Class 1.
- The positive predictive value (precision) is moderate, and the negative predictive value is relatively high.
- The model's prevalence matches the distribution of the positive class in the dataset.
- The balanced accuracy provides an overall measure of model performance.



- Despite the simplicity of the Naive Bayes algorithm, it provides competitive results in this classification task.
- The detailed evaluation metrics offer insights into the strengths and limitations of the model, guiding further refinement if needed.

Result Printing:

The results, including model accuracy, confusion matrix, and classification report, are printed for interpretation.

64.41%, indicating the percentage of correctly classified instances.

The Classification Report provides a detailed analysis of the model's performance, including accuracy, sensitivity, specificity, precision, negative predictive value, prevalence, and balanced accuracy.

The evaluation metrics provide a comprehensive understanding of the Naive Bayes model's performance in this specific classification task, with a focus on its ability to correctly identify instances and its overall discriminatory power.

Conclusion :

Naive Bayes Model:

With an accuracy of 64.56%, the Naive Bayes model excels in sensitivity but struggles with specificity, partly due to the data imbalance in the "STATUS" column.

Addressing this imbalance through techniques like oversampling or SMOTE could enhance its overall performance.

Logistic Regression Model:

The Logistic Regression model, achieving 63.61% accuracy, grapples with a trade-off between sensitivity and specificity, exacerbated by data imbalance. Strategic adjustments, such as oversampling or weighted classes, might alleviate this imbalance and improve the model's efficacy.

```
[1] "Logistic Model Accuracy:  63.61 %"
[1] "Confusion Matrix:"
   predicted_labels
       0    1
  0 6179  228
  1 3387  139
[1] "Classification Report:"
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6179 3387
         1  228  139

               Accuracy : 0.6361
                 95% CI : (0.6265, 0.6455)
    No Information Rate : 0.645
    P-Value [Acc > NIR] : 0.9695

                  Kappa : 0.0048

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.96441
            Specificity : 0.03942
         Pos Pred Value : 0.64593
         Neg Pred Value : 0.37875
             Prevalence : 0.64502
         Detection Rate : 0.62207
   Detection Prevalence : 0.96305
      Balanced Accuracy : 0.50192

       'Positive' Class : 0
```

In summary, while both models show promise, tackling data imbalance is crucial for refining their performance across diverse classes. Exploring advanced sampling methods may further optimize their predictive capabilities

6.4 Logistic Regression using Foldout and K-Fold method-MEENU

Library CaTools is used for splitting dataset into train and test sets in the ratio 70:30. glm function which stands for Generalised Linear Model is used for fitting.

Logistic Regression uses the formula

Y = e^(βo + β1X1…. βnXn) / (1 + e^(β0 + β1X1+….. βnXn))

Where:

- X1,X2..Xn are the input value
- Y is the predicted output
- β is the bias or intercept term
- β 1 is the coefficient for the single input value (X1)

After performing Logistic Regression using Foldout method summary obtained is

```
[1] "Accuracy: 0.646632437330112"

Call:
glm(formula = STATUS ~ ., family = "binomial", data = train_df)

Coefficients:
                                        Estimate Std. Error z value Pr(>|z|)
(Intercept)                            -0.55595    0.27953  -1.989 0.046716 *
GENDERM                                 0.03858    0.03275   1.178 0.238812
OWN_CARY                                0.02625    0.03133   0.838 0.402136
OWN_REALTYY                            -0.07748    0.03027  -2.560 0.010471 *
INCOME_TYPEPensioner                    0.03623    0.06014   0.602 0.546919
INCOME_TYPEState servant                0.01597    0.05647   0.283 0.777344
INCOME_TYPEStudent                      1.16054    0.86849   1.336 0.181459
INCOME_TYPEWorking                      0.01027    0.03524   0.291 0.770721
EDUCATION_TYPESecondary / secondary special -0.45623    0.13422  -3.399 0.000676 ***
EDUCATION_TYPEIncomplete higher         -0.38486    0.15119  -2.545 0.010913 *
EDUCATION_TYPEHigher education          -0.33634    0.13656  -2.463 0.013777 *
EDUCATION_TYPEAcademic degree           0.56861    0.48495   1.173 0.240990
FAMILY_STATUSMarried                    -0.07630    0.05171  -1.476 0.140050
FAMILY_STATUSSeparated                  -0.08094    0.09186  -0.881 0.378252
FAMILY_STATUSSingle / not married       -0.04728    0.09017  -0.524 0.600056
FAMILY_STATUSWidow                      -0.15196    0.10993  -1.382 0.166858
HOUSING_TYPEHouse / apartment           0.38933    0.22347   1.742 0.081477 .
HOUSING_TYPEMunicipal apartment         0.40324    0.23617   1.707 0.087748 .
HOUSING_TYPEOffice apartment            0.42630    0.27184   1.568 0.116831
HOUSING_TYPERented apartment            0.40169    0.24894   1.614 0.106609
HOUSING_TYPEWith parents                0.43700    0.23155   1.887 0.059117 .
EMAIL1                                  0.11186    0.04811   2.325 0.020063 *
FAMILY_SIZE2                            0.05750    0.07674   0.749 0.453708
FAMILY_SIZE3                            0.02924    0.08583   0.341 0.733395
FAMILY_SIZE4+                           0.11181    0.09170   1.219 0.222727
INCOME                                 -0.88388    0.24074  -3.671 0.000241 ***
AGE                                     0.19817    0.12282   1.613 0.106646
YEARS_EMPLOYED                          0.22526    0.10789   2.088 0.036812 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 30131  on 23176  degrees of freedom
Residual deviance: 30059  on 23149  degrees of freedom
AIC: 30115
```

Model Tuning

From the summary we can find that features like Income, Education_Type Secondary/secondary perform a significant importance in modelling. OWN_REALTY, EDUCATION_TYPEIncomplete higher, EDUCATION_TYPE Higher education provide 95% significance to modelling. We cannot straight away remove the features that are not significant, there are other parameters to take into consideration. We have Null deviance, Residual deviance and AIC. Null deviance is the deviance that we get from actual value of our dataset. Our code is 30131 units deviant when it is null which means we are not using any independent variable (taking only β0). Residual deviance is the deviance when we include independent variables. When adding independent variable, the deviance comes to smaller value which is good because accuracy of the model becomes more correction should be as minimum as possible. This will be helpful if we are removing non-significant values. Basically, the

Residual deviance should not increase and AIC should decrease. By this we can tune logistic Regression Classifier. After tuning and putting a threshold of 0.399 we got an accuracy of 63.24%. After K-fold method similar accuracy was obtained.

EVALUATION AND INTERPRETATION

Even though it has an accuracy of 63.24% it is not predicting status 1 properly.

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6106 3404
         1  247  176

               Accuracy : 0.6324
                 95% CI : (0.6229, 0.6419)
    No Information Rate : 0.6396
    P-Value [Acc > NIR] : 0.9323

                  Kappa : 0.0127

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.96112
            Specificity : 0.04916
         Pos Pred Value : 0.64206
         Neg Pred Value : 0.41608
             Prevalence : 0.63959
         Detection Rate : 0.61472
   Detection Prevalence : 0.95741
      Balanced Accuracy : 0.50514

       'Positive' Class : 0
```

```r
# Calculate performance metrics
TP <- sum(comparison$Actual == 1 & comparison$Predicted == 1)
TN <- sum(comparison$Actual == 0 & comparison$Predicted == 0)
FP <- sum(comparison$Actual == 0 & comparison$Predicted == 1)
FN <- sum(comparison$Actual == 1 & comparison$Predicted == 0)

precision <- TP / (TP + FP)
recall <- TP / (TP + FN)
f1_score <- 2 * (precision * recall) / (precision + recall)
accuracy <- (TP + TN) / (TP + TN + FP + FN)

# Print performance metrics
print(paste("Precision:", precision))
print(paste("Recall:", recall))
print(paste("F1 Score:", f1_score))
print(paste("Accuracy:", accuracy))
```

```
[1] "Precision: 0.416075650118203"
[1] "Recall: 0.0491620111731844"
[1] "F1 Score: 0.0879340494629028"
[1] "Accuracy: 0.632437330111749"
```

1.CONFUSION MATRIX

- True Positive (TP): 176
- True Negative (TN): 6203
- False Positive (FP): 3410
- False Negative (FN): 247

2.Accuracy and Precision:

The overall accuracy of the model is 63.24%, which is the proportion of correctly classified instances (TP + TN) out of the total instances.

3.Kappa:

Reducing the expected agreement by chance, the Kappa statistic quantifies the degree of agreement between the model's predictions and the actual results. Since there is less agreement than would be predicted by chance in this instance, the Kappa is almost 0.

4. **Sensitivity (True Positive Rate):**

Sensitivity is 0.96. It measures correct proportion of actual positive (Class 1)

4.Specificity (True Negative Rate):

It is 0.049. It measures the proportion of actual negatives that the model correctly predicts which is very low.

5.Positive Predictive Value (Precision):

PPV is 0.645

6.Precision:

It is 0.4 Precision is the proportion of instances predicted as positive by the model that are actually positive. It is calculated as TP / (TP + FP).

The percentage of cases that the model predicts as positive but which are actually positive is known as the Positive Predictive Value (PPV) or Precision.

CONCLUSION

The model fails to identify true negatives, but it has a very low specificity (sensitivity) and identifies the majority of actual positives.

6.5 QUADRATIC DISCRIMINANT ANALYSIS-MEENU

Discriminant Analysis is a statistical technique. It aims to predict a categorical dependent variable based on one or more continuous or binary independent variable. It is often used when the dependent variable is categorical and independent variable are continuous or binary.

Concept

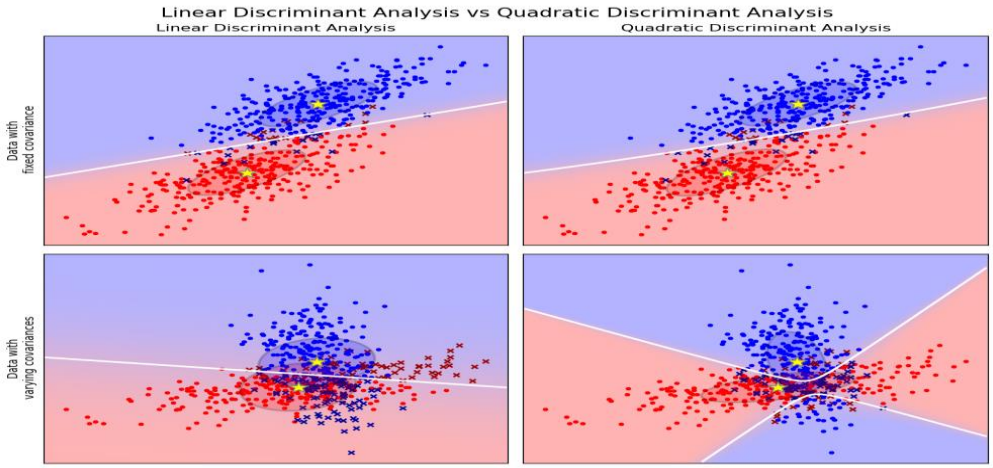
Fundamentally, QDA is intended to divide data points into discrete groups according to their characteristics. Its capacity to manage complicated datasets by employing quadratic functions to model the connection between variables accounts for its significance in machine learning. QDA is an effective technique for capturing nonlinear patterns, in contrast to other algorithms that presume linear correlations. Because of this, it is used in a variety of industries, including banking, medicine, and image recognition.

Fisher's Linear Discriminant Analysis (LDA), which sought to identify linear discriminant functions for classification tasks, is where QDA got its start. LDA's application in situations where these assumptions are unfeasible is limited by the assumption that covariance matrices across classes are equal.

By easing the requirement of identical covariance matrices and permitting each class to have a unique covariance matrix, QDA evolved as an extension of LDA. Because of its adaptability, QDA can accomplish more accuracy in classification jobs and identify more subtle linkages among datasets.

One noteworthy feature is its seamless handling of both categorical and numerical variables. Even when working with discrete variables, QDA may capture nonlinearity by adding quadratic terms into its computations.

Furthermore, rather than producing straightforward binary classifications, QDA produces probabilistic results. In other words, QDA provides us with the probability that a given data item belongs to each class rather of just classifying it. We are able to make better decisions thanks to this probabilistic technique, which offers insightful information about the uncertainty around classifications.



SELECTION OF ML MODEL

Initially there was a confusion of choosing which model for implementation among Linear Discriminant Analysis (LDA) and QDA. In our dataset, one class (Class 0) has significantly fewer observations than other (Class1). Hence QDA is chosen since it is sensitive to small sample sizes. It also allows for a quadratic decision boundary. It can better capture non-linear relationships between predictors and the response variable. While cross validating the performance of LDA and QDA, QDA showed better performance. Hence the appropriate model is selected.

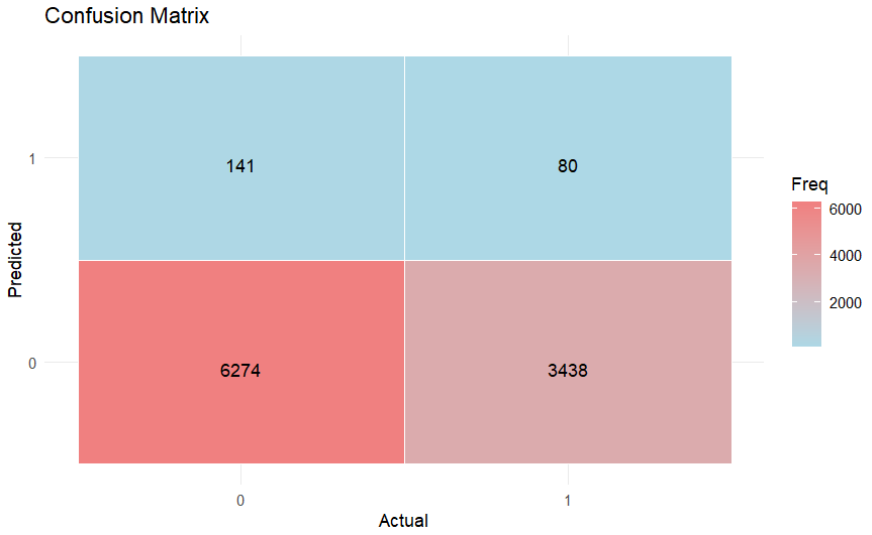
QDA doesn't have many tuning parameters to adjust. It is a parametric model that estimates class mean and variance. Therefore there are limited hyperparameter to tune.

```r
# Create a confusion matrix
conf_matrix <- confusionMatrix(data = comparison$Predicted, reference = comparison$Actual)

# Extract confusion matrix as a table
conf_matrix_table <- as.table(conf_matrix$table)

print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6274 3438
         1  141   80

               Accuracy : 0.6397
                 95% CI : (0.6302, 0.6491)
    No Information Rate : 0.6458
    P-Value [Acc > NIR] : 0.9014

                  Kappa : 0.001

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.97802
            Specificity : 0.02274
         Pos Pred Value : 0.64600
         Neg Pred Value : 0.36199
             Prevalence : 0.64583
         Detection Rate : 0.63163
   Detection Prevalence : 0.97775
      Balanced Accuracy : 0.50038

       'Positive' Class : 0
```

EVALUATION AND INTERPRETATION

1.CONFUSION MATRIX



True Positive (TP): 80

True Negative (TN): 6274
False Positive (FP): 3438
False Negative (FN): 141

2. Accuracy and Confidence Interval:

Accuracy: 0.6397 (63.97%)
95% Confidence Interval (CI): (0.6302, 0.6491)

3.Kappa:

Kappa: 0.001
The low Kappa value indicates a weaker agreement than would be predicted by chance.

4. Mcnemar's Test P-Value:

Mcnemar's Test P-Value: <2e-16
A substantial difference between the model's predictions and the actual results is shown by a significant p-value.

5. Sensitivity and Specificity:

Sensitivity (True Positive Rate): 97.8%
Specificity (True Negative Rate): 2.27%

While the specificity is quite low, suggesting a significant rate of false positives, the high sensitivity indicates a good capacity to reliably detect occurrences of the '0' class.

6. Positive Predictive Value (PPV) and Negative Predictive Value (NPV):

PPV (Precision): 64.6%
NPV: 36.2%
PPV is relatively higher, indicating that when the model predicts '0', it's correct 64.6% of the time.

7. Prevalence, Detection Rate, and Detection Prevalence:

Prevalence: 64.58%
Detection Rate: 63.16%
Detection Prevalence: 97.77%
Detection Rate is about equal to prevalence (97.77%), indicating that there isn't much of an improvement over the baseline provided by the model.

8. Balanced Accuracy:

Balanced Accuracy: 50.04%

When balanced accuracy approaches 50%, it means that overall performance of the model is not much higher than that of a random classifier.

9.F1 Score:

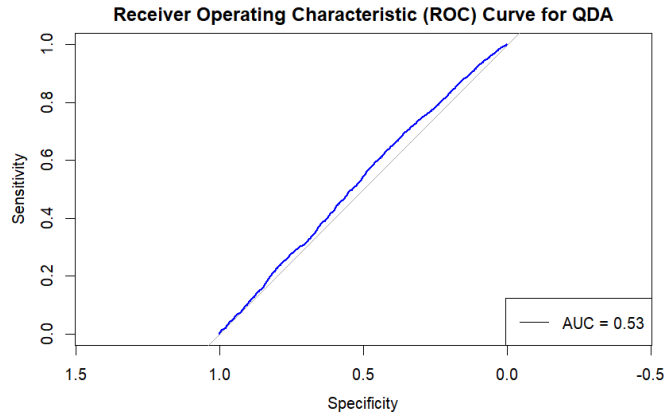
F1 Score = 0.778074

The F1 score is the harmonic mean of precision and recall, providing a balanced measure between precision and recall.

A higher F1 score indicates better balance between precision and recall. In this case, the F1 score is approximately 77.8%.

10. AUC

The QDA model's AUC of 0.53 indicates that its performance in differentiating between the positive and negative classes is not significantly superior to that of random chance, with an AUC of 0.5. Put otherwise, the predictive capacity of the model is restricted.



CONCLUSION

To sum up, there are a lot of false positives with the QDA model because of its high sensitivity and low specificity. The class imbalance affects the accuracy overall. The Mcnemar's test p-value of significance and the low Kappa suggest that there is a lack of strong agreement between the model's predictions and the observed results. To enhance the model's performance, more research or consideration of alternative methods might be required.

6.6 Decision Tree Model: Mohit

Recursive Partitioning: Decision trees recursively partition the data based on the most significant features, resulting in a tree-like structure.

In the code snippet, the Decision Tree model is trained on the 'STATUS' variable in the dataset 'df,' considering all other features as predictors. The model leverages k-fold cross-validation, enhancing its robustness and generalization to unseen data. The resulting tree is visualized using the 'plot' function, providing a clear depiction of

the decision-making process.

The overall accuracy of the Decision Tree model is calculated at 65.36%, indicating the percentage of correctly classified instances. This performance is compared to a baseline measure known as the No Information Rate, which represents the accuracy achieved by predicting the majority class. The model significantly outperforms the No Information Rate, as evidenced by a small p-value (0.0005542).

Decision Tree Model:
Recursive Partitioning: Decision trees recursively partition the data based on the most significant features, resulting in a tree-like structure.
Leaf Nodes: The final nodes (leaves) of the tree contain the predicted outcomes.

Working Principle:
Feature Selection: Decision trees select features at each node based on their ability to best separate the data.
Splitting Criteria: Nodes are split using criteria such as Gini impurity or information gain to maximize homogeneity within nodes.

Model Performance Statistics:

Accuracy: 65.36%

95% Confidence Interval: (64.85%, 65.87%)
No Information Rate: 64.50%

P-Value [Accuracy > No Information Rate]: 0.0005542
Kappa Statistic: 4.67%

Mcnemar's Test P-Value: < 2.2e-16

Sensitivity/Recall (True Positive Rate): 98.42%

Specificity (True Negative Rate): 5.29%

Positive Predictive Value (Precision): 65.38%

Negative Predictive Value: 64.79%

Prevalence: 64.50%

Detection Rate: 63.48%

Detection Prevalence: 97.10%

Balanced Accuracy: 51.86%

Interpretation:

The decision tree model achieved an accuracy of 65.36%, performing better than the No Information Rate.

The Kappa statistic indicates fair agreement between predicted and observed classifications.

Sensitivity is high at 98.42%, suggesting the model's ability to correctly predict actual positive instances.

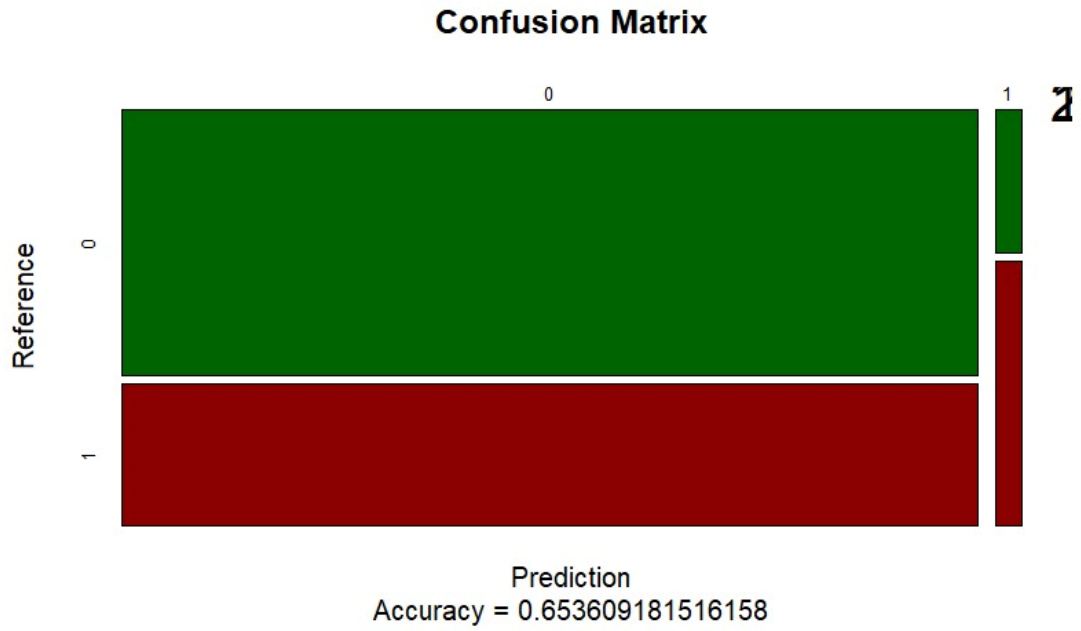
Specificity is low at 5.29%, indicating challenges in accurately predicting actual negative instances.

Positive Predictive Value (Precision) is at 65.38%, demonstrating reliability in positive predictions.

The model has a prevalence of 64.50%, with a balanced accuracy of 51.86%.

Mcnemar's Test P-Value suggests a significant difference in errors between the model and a hypothetical perfect model. Features Selection: Decision trees select features at each node based on their ability to best separate the data.

Splitting Criteria: Nodes are split using criteria such as Gini impurity or information gain to maximize homogeneity within nodes.




```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 21019 11131
         1   338   622

               Accuracy : 0.6536
                 95% CI : (0.6485, 0.6587)
    No Information Rate : 0.645
    P-Value [Acc > NIR] : 0.0005542

                  Kappa : 0.0467

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.98417
            Specificity : 0.05292
         Pos Pred Value : 0.65378
         Neg Pred Value : 0.64792
             Prevalence : 0.64503
         Detection Rate : 0.63482
   Detection Prevalence : 0.97101
      Balanced Accuracy : 0.51855

       'Positive' Class : 0
```

Logistic regression:

```
> confusion_matrix <- table(comparison$Actual, comparison$Predicted)
> predicted <- predict(logistic_model, newdata = test_df, type = "response")
> predicted_class <- ifelse(predicted > 0.45, 1, 0)
> comparison <- data.frame(Actual = test_df$STATUS, Predicted = predicted_class)
> print(mean(comparison$Actual == comparison$Predicted))
[1] 0.6412967
> confusion_matrix <- table(comparison$Actual, comparison$Predicted)
> rownames(confusion_matrix) <- c("Actual_0", "Actual_1")
> colnames(confusion_matrix) <- c("Predicted_0", "Predicted_1")
> print("Confusion Matrix:")
[1] "Confusion Matrix:"
> print(confusion_matrix)

          Predicted_0 Predicted_1
Actual_0        6296         115
Actual_1        3448          74
>
```

Logistic Regression Results (Luke)

To try and get better results from logistic regression, I attempted two evaluation methods, the standard holdout method and k-fold cross validation. K-fold is considered a more robust evaluation method, so I wanted to see if it would give better results compared to holdout. K-fold validation also removes the need to specify a decision threshold, since the training calculates it automatically.

Holdout Method

I decided to use the 70/30 split between training and testing data. This left the model with about 23000 rows to train from and about 9000 to predict on. After training the model and predicting the probabilities on the test set, I summarised the values of the probabilities to find the ideal decision threshold.

```{r}
summary(predicted)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1774  0.3403  0.3565  0.3563  0.3728  0.7222
```

As seen, the values range between 0.1 and 0.7, so I settled on 0.4 as my threshold.

The model's predictions were then compared to the test set for the evaluation metrics.

The first metric used was accuracy, which was found to be around 64%

```r
comparison <- data.frame(Actual = test_df$STATUS, Predicted = predicted_class)
print(mean(comparison$Actual == comparison$Predicted)*100)
```

```
[1] 63.70684
```

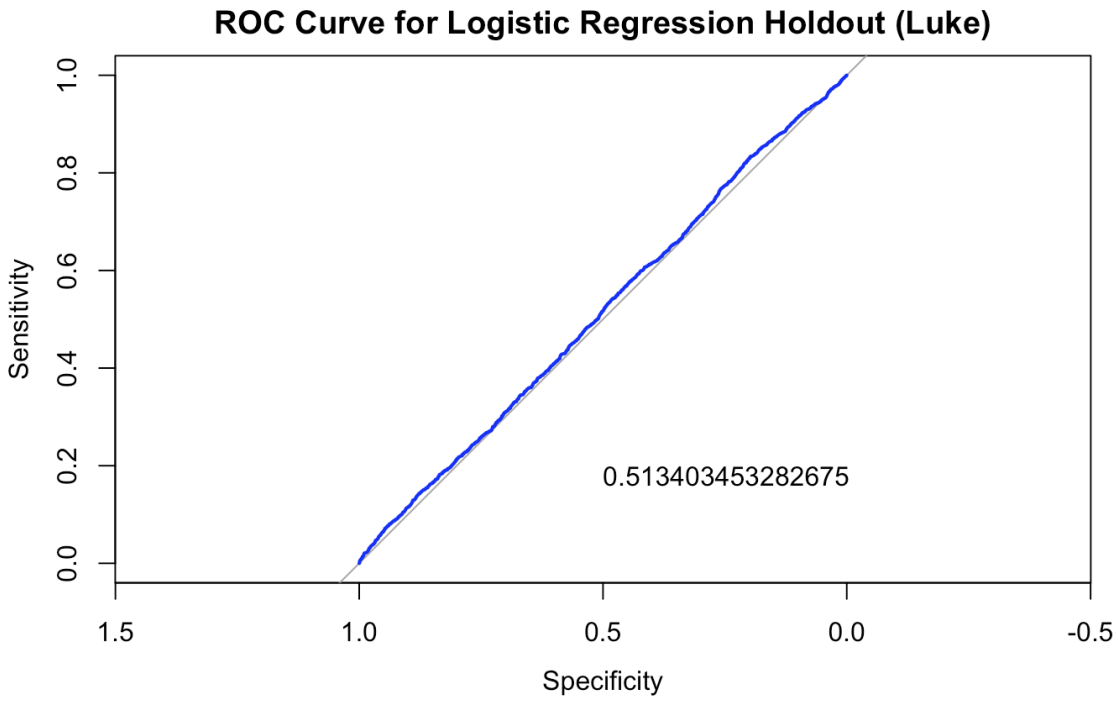
The next metric used was the confusion matrix

```r
print("Confusion Matrix:")
print(confusion_matrix)
```

```
[1] "Confusion Matrix:"

           Predicted_0 Predicted_1
Actual_0          6149         244
Actual_1          3361         179
```

As seen, the negative class (loan rejected) was predicted much more frequently with high accuracy, while the positive class (loan approved) was very rarely predicted.

The final evaluation method used is the Receiver Operator Characteristic Curve (ROC) and the Area under the curve (AUC)



The AUC was found to be just over 0.51, which is a very poor result since a random classifier would return an AUC of 0.5

K-fold Method

For the k-fold method, I decided to use 10 folds of the 33110 rows to evaluate the data.

```
Generalized Linear Model

33110 samples
   12 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 29799, 29798, 29799, 29799, 29799, 29799, ...
Resampling results:

  Accuracy   Kappa
  0.6436726  0.00105577
```

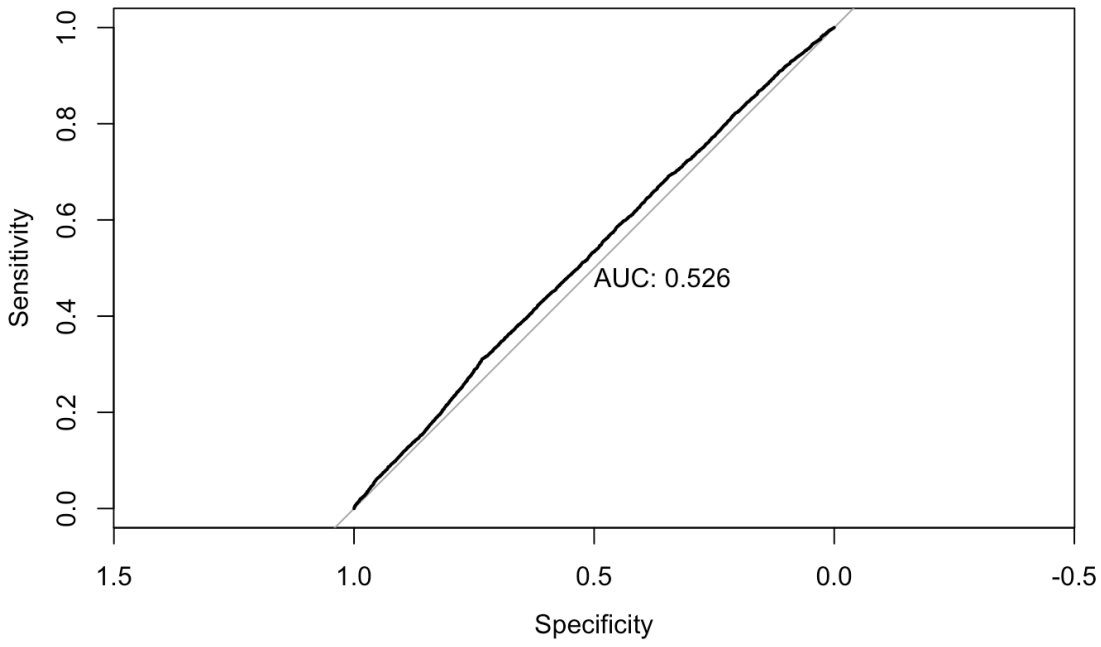
Unlike with holdout, all the data is used in the different folds for training and testing. Despite this, the accuracy was very similar to the holdout method.

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 21298 11783
         1    11    18
```

The confusion matrix was even more biased towards 0 than the holdout method, with the positive class being only predicted 18 times correctly. It seems that k-fold validation does not help with class imbalance.

```
               Sensitivity : 0.999484
               Specificity : 0.001525
            Pos Pred Value : 0.643814
            Neg Pred Value : 0.620690
                Prevalence : 0.643582
            Detection Rate : 0.643250
      Detection Prevalence : 0.999124
         Balanced Accuracy : 0.500505
```

The other statistics prove the poor performance of the classifier. The specificity value shows how poor the classifier performed on predicting 1.



The k-fold method found a slightly better AUC than the holdout method, but among the two I would still recommend the holdout due to it being less biased against the 1 class.

Support Vector Machine (SVM) Classifier

The SVM classifier attempts to create a hyperplane in N-dimensional space (for each predictor used) to separate the target variable. The model will attempt to find the maximum distance between the points of both classes to separate them. The support vectors are used to position the hyperplane so that it maximises the margin of the classifier.

The SVM classifier gets very expensive when used with a high number of predictors, as I have tried in this case, with 12 predictors, as it needs to model 12-D space. Despite this, I thought it would give better results than logistic regression, which is a simpler model.

To create an SVM classifier in R, the e1071 library is used. Due to the complexity of the model, only the holdout evaluation method is used for this classifier. Even using holdout takes several minutes to train the model, with k-fold taking significantly longer, so it was not considered.

As before, the model is trained on the training data (70% of the total data) and tested on the test data (30%)

```{r}
svm_model <- svm(STATUS ~ ., data = train_df, kernel = "linear", probability = TRUE)
```

The hyper-parameter chosen here is the kernel, which is set to "linear", which is the simplest model of the SVM kernels, meant for linear data. This was chosen to avoid making the model overly complicated.

After the predictions are made, I found the accuracy of the model.

```{r}
mean(predicted_probabilities == test_df$STATUS)
```

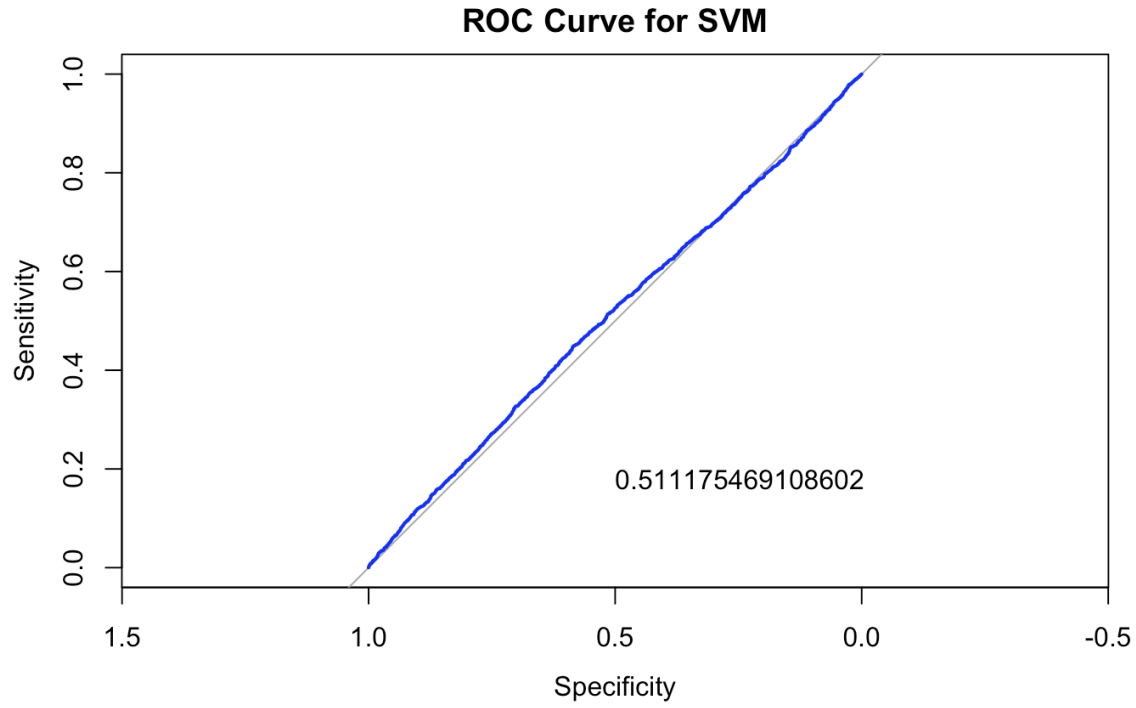
```
[1] 0.6439142
```

Despite being a more complicated model, the accuracy was very similar to the logistic regression models.

```{r}
conf_matrix <- table(predicted_probabilities, test_df$STATUS)
print("Confusion Matrix:")
print(conf_matrix)
```

```
[1] "Confusion Matrix:"

predicted_probabilities    0    1
                      0 6389 3533
                      1    4    7
```

The confusion matrix reveals the outputs are even more skewed than the previous models. The 1 class has only been predicted 7 times correctly.



The ROC curve and AUC are also very similar, with the AUC value of 0.51 being only just above a random classifier.

The models all performed poorly, especially on predicting the positive class. This may have been caused by the imbalance of the target variable, so rebalancing or oversampling would be a good next step to fix the imbalance.

LOGISTIC REGRESSION RESULTS: MOHIT

In this analysis, two machine learning models, Logistic Regression and Decision Tree, were employed to predict the 'STATUS' variable based on various features. The Logistic Regression model achieved an accuracy of 64.10%, outperforming the No Information Rate significantly (p-value = 0.0005542). It underwent a comprehensive pre-processing pipeline, including handling missing values, converting categorical variables, and normalizing features. On the other hand, the Decision Tree model utilized recursive partitioning, creating a tree-like structure based on the most significant features. The Decision Tree achieved an accuracy of 65.36%, surpassing the No Information Rate and exhibiting fair agreement with observed classifications (Kappa statistic = 4.67%). While the Logistic Regression model provides a balanced approach, the Decision Tree model offers interpretability in decision-making through its tree structure. The choice between models depends on specific objectives and preferences for interpretability. Both models contribute valuable insights, addressing the 'STATUS' prediction task with their respective strengths and considerations.

CONCLUSION

In our project we have analysed and modelled different machine learning algorithm to find whether the applicant is good or bad. If the model predicts it as 0 then it indicates the applicant is bad and we cannot grant loan to them and otherwise, the applicant is good, and we can give loan. Here everyone modelled individual Logistic regression as the common model and Random Forest, Naïve Bayes Algorithm, Quadratic Discriminant Analysis, Decision Tree, and SVM as the individual models.

The table shows the comparison and performance of the different models. From the table it is evident that Random Forest has the highest accuracy with 78.35% followed by Mohit's Logistic Regression with 65.36%.

Models	TP	FN	TN	FP	Accuracy
Logistic Regression - Anu	25	11703	21367	15	64.6%
Random Forest-Anu	6790	2230	19152	4938	78.35%
Logistic Regression - Harshita	139	3387	6179	228	63.61%
Naive Bayes - Harshita	175	3351	6234	173	64.52%
Logistic Regression - Meenu	176	247	6203	3410	63.2%
Quadratic Discriminant Analysis - Meenu	80	141	6274	3438	63.9%
Logistic Regression - Mohit	74	115	6296	3448	65.36%
Decision tree – Mohit	662	11131	21019	338	64.12%
Logistic Regression - Luke	179	244	6149	3361	63.7%
SVM - Luke	7	3533	6389	4	64.39%

Therefore, we would recommend the Random Forest model be used to classify loan applications, since it had the highest accuracy and was more accurate on the positive class, which is important for banks and other businesses that give loans. The other models struggled with the imbalanced target variable and may lead banks to reject all loan offers. In the future, a more balanced target variable would be helpful to solve this problem.

References

- https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction/discussion
- https://www.datascience-pm.com/crisp-dm-2/
- https://www.analyticsvidhya.com/blog/2021/03/introduction-to-k-fold-cross-validation-in-r/
- https://www.javatpoint.com/machine-learning
- https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
- https://medium.com/data-analytics-magazine/understand-quadratic-discriminant-analysis-qda-in-maschine-learning-a-classification-algorithm-4587747cbc4a

AUTHORS CONTRIBUTION STATEMENT

ANU SABU

I actively engaged in various project aspects, making significant contributions to meetings, data selection, and dataset comprehension. I also took part in separate data pre-processing, initially designating 'x' and 'c' as 1 and other statuses as 0 to optimize the target representation. However, due to observed low accuracy post-pre-processing, I decided to eliminate this concept to improve model performance.

During pre-processing, I handled missing values by identifying and addressing them to ensure a comprehensive and accurate dataset for model training. I implemented outlier detection techniques to enhance data robustness, crucial for preventing extreme values from unduly influencing model performance. Addressing imbalanced data, I employed techniques to balance class representation, aiming to mitigate biases caused by uneven class distributions.

I efficiently contributed to report creation, synthesizing key findings, methodologies, and results. My goal was to provide stakeholders with a clear understanding of the project's objectives and outcomes.

In modelling, I applied k-fold cross-validation to enhance Logistic Regression model robustness, allowing a comprehensive assessment across different dataset subsets. For Random Forest, a powerful ensemble learning technique, I utilized k-fold cross-validation to improve generalization capabilities and overall predictive performance.

To evaluate model performance comprehensively, I implemented confusion matrices, examining true positives, false positives, true negatives, and false negatives. Additionally, I contributed to the comparative analysis of Logistic Regression and

Random Forest models, providing insights into their respective strengths and weaknesses.
Signed: Anu Sabu

MEENU NAIR- Meenu have made a separate model on logistic Regression.
First of all, Data Pre-processing is done individually. I have merged two datasets application.csv and credit.csv. Each Customer ID has different status (X- all due is paid, C- no loans, 0: 1-29 days past due etc). I have initially selected customer ID with maximum Occurrence of Status (count of each status of corresponding ID is taken and the highest count is taken). So only unique ID is presented. Status is in turn categorized into 0 and 1. Status is our dependent variable. All the categorical columns were converted into factors and all the numerical columns were scaled. finally CODE_GENDER, FLAG_OWN_CAR, FLAG_OWN_REALTY, AMT_INCOME_TOTAL
NAME_INCOME_TYPE,NAME_EDUCATION_TYPE,
NAME_FAMILY_STATUS,NAME_HOUSING_TYPE,DAYS_BIRTH
DAYS_EMPLOYED,OCCUPATION_TYPE, CNT_FAM_MEMBERS were taken as the independent variable. totally 12 independent variable was taken. After pre-processing, training and test set was made and model is constructed. On test set it was not predicting status 0. On evaluation it was found that while choosing Customer ID with most frequent status, majority status was 1 . There was a total of 36327 1s and only130 0s. As a result, model doesn't have enough training data on 0s. So Customer ID with random Status is selected. 777715 obs. 20 variables is now reduced to 36457 obs. 20 variables. Logistic Regression was done. It gave accuracy only an accuracy of 52% even after hyperparameter tuning.
Coming to 2nd ML model I initially selected KNN, but it was not working on our dataset. After running several models, working ones were QDA and LDA. Made a choice between the two. Performed the modelling.
Signed: Meenu Nair

LUKE
Found the dataset used on Kaggle.com. Performed the initial pre-processing and cleaning, which was used for the final report, including removing columns, renaming columns, sorting columns into categorical and continuous, encoding and normalising columns, and processing the target variable.
Created models for Logistic Regression and Support Vector Machine with evaluations.
Worked on the group code submission and the Rmd notebook format, ensuring everyone's code worked correctly in both the notebook and the standard R format.

Signed: Luke Menezes

HARSHITA

Initially, team members worked individually on data cleaning, exploring various approaches. A standardized data cleaning approach was adopted based on its effectiveness across all team members' datasets. Visualizations from individual attempts, using ggplot2 and pie charts, were incorporated to understand the distribution of categorical variables.
A heatmap visually highlighted null values in the dataset, aiding in recognizing patterns of missing data.

Machine Learning Models:

Applied two machine learning models—Logistic Regression and Naive Bayes—for binary classification of the 'STATUS' variable.
Logistic Regression:
Built the model using the glm function, split the data into training and testing sets, and evaluated the accuracy.
Generated a confusion matrix and a classification report for detailed performance assessment.
Naive Bayes:
Utilized the naive Bayes function from the e1071 library, split the data, and evaluated the model's accuracy.
Produced a confusion matrix and a classification report for performance evaluation.
Conclusion:

A collaborative effort was made to refine the data cleaning process, with an adopted approach for consistency across team members. The inclusion of individual visualizations enhanced the exploratory data analysis, providing insights into variable distributions and relationships. Machine learning models were applied, and comprehensive performance metrics, including confusion matrices and classification reports, were utilized for evaluation.

Signed: Harshita Wardhan

MOHIT

In the data analysis workflow, my primary responsibility involved implementing machine learning models, specifically logistic regression and decision trees, using R and the Rpart library.
I contributed significantly to meticulous data pre-processing, ensuring dataset integrity by handling missing values, normalizing variables, and encoding categorical features. Utilizing visualization tools, I uncovered key patterns and trends in the data. The logistic regression model achieved a 64.1% accuracy rate, and the decision tree model with the accuracy of 65.3% provided additional insights.
Additionally, I played a crucial role in preparing the comprehensive report, synthesizing insights, and structuring it to cover the problem introduction, preprocessing steps, visualizations, model implementations, performance metrics, and conclusions. This systematic approach ensures effective communication of findings, facilitating informed decision-making for stakeholders based on a thorough understanding of the dataset and predictive analytics models applied.

Signed: Mohit Singh