NOVA IMS
Information Management School

Financial DATA SCIENCE

Group Project 5

Carlos Caballos - 20220528

Frederico Bravo - 20221231

Neftali Herculano - 20200732

Nelson Lima - 20221539

Riccardo Gurzu - 2022123

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

Contents

List of Figures

List of Tables

Abstract

In this study, we were challenged to develop a data modelling for a credit dataset to understand the customer's loan profiles better. To do so, we went through a primary statistical analysis of the credit dataset that considered outliers' detection and treatment of missing values to understand better the data given.

Secondly, we also developed three ML models (Logistic Regression, Random Forest and Deep Neural Network) to predict the probability of default and evaluated the model's results. We also tested alternative models and compared our results. We decided to use the R version 4.2.0 (2022-04-22) for calculations and LaTeXto prepare the final report.

The findings of this study show that the Random Forest methodology represents a more reasonable coefficient of determination and a lower false-negative rate than other models.

1 Introduction

For the financial sector, particularly banks, credit risks represent a significant risk. It occurs when a firm or the debtor does not fulfil debt obligations individually. That is to say, that the probability of a lender not having received principal and interest on debt can also be taken into account. This is why board members and senior management must establish and demonstrate a clear understanding of the measurement of credit risk in their institutions. Although accounting firms and bank regulators have been going in a similar direction regarding risk modelling, they each develop their own models by relying on the best instruments at their disposal to seek reasonable values and attempt to predict potential losses. Today, banks and financial institutions are using Big Data and Machine Learning Models to help predict credit risk, which is the process of predicting who to lend money to and how much to extend credit, and credit monitoring, which tracks the borrower's credit behaviour over time. Financial institutions are also using ML to detect fraud and prevent loss.

The history of credit risk assessment and its intersection with machine learning reflects an evolution driven by the increasing complexity of financial landscapes and the growing availability of diverse data sources. Traditionally, credit risk evaluation relied on rule-based systems and statistical models with limited capacity to adapt to dynamic market conditions. Over the past few decades, the surge in computing power and the proliferation of digital data have catalyzed a paradigm shift towards machine learning in credit risk management. The use of neural networks, decision trees, and ensemble methods has become more prevalent, allowing for a more nuanced understanding of risk factors and patterns. Notably, the 2008 financial crisis underscored the shortcomings of conventional risk models, prompting a renewed focus on incorporating advanced machine learning algorithms capable of capturing nonlinear relationships and identifying hidden risks. Since then, researchers and financial institutions have continued to refine and expand machine learning applications in credit risk assessment, aiming to enhance predictive accuracy and anticipate default events with greater precision in an ever-changing financial landscape.

In conclusion, the convergence of credit risk assessment and machine learning represents a pivotal juncture in the history of financial decision-making. As institutions seek to navigate the intricate web of factors influencing creditworthiness, the adoption of machine learning methodologies stands as a testament to the ongoing quest for more robust and adaptive risk management tools. This symbiotic relationship between data-driven algorithms and the historical context of credit risk not only addresses the limitations of traditional models but also reflects a commitment to staying ahead of emerging challenges in the global financial ecosystem. By harnessing the power of machine learning, stakeholders in the financial industry aim not only to accurately predict defaults but also to foster a more resilient and responsive approach to credit risk assessment, ultimately shaping a landscape where precision, agility, and informed decision-making are paramount. This is exactly what we will be doing during this project using the a training and test dataset to develop a model that will be able to predict if the credit would or not default and understand the risk of it.

2 Primary Statistical Analysis

An Exploratory Data Analysis is the first step in any data analysis project. In the present project, 310704 observations on the training dataset and 20600 on the test dataset need to be explored in three different ways:

i. Summarizing the dataset using descriptive statistics;

ii. Visualizing the dataset using charts;

iii. Identifying missing values.

Before starting to perform statistical models, the three actions will enable us to get a sense of how values are distributed and whether there are any problems.

In the dataset, we have two types of variables, categorical and numerical:

- Categorical/factor variables: *term*, *grade*, *emp_title*, *emp_length*, *home_ownership*, *verification_status*, *issue_d*, *purpose*, *addr_state*, *earliest_crline* and *loan_status*.

- For the numerical values, we present the descriptive stats below:

On Figure 1 it is possible to understand that we have more observations, on the train dataset, classified with 0 on risk than with 1, meaning that majority of the loan didn't default. To be more precise, 207036 observations didn't default while 103668 did default.

Variable	n	mean	sd	median	min	max	range	skew	kurtosis	se
loan_amnt	310704	15518.61	9196.53	14000.00	1000.00	40000.00	39000.00	0.70	-0.27	16.50
funded_amnt	310704	15518.61	9196.53	14000.00	1000.00	40000.00	39000.00	0.70	-0.27	16.50
funded_amnt_inv	310704	15511.82	9195.22	14000.00	725.00	40000.00	39275.00	0.70	-0.27	16.50
int_rate	310704	12.57	4.70	11.99	5.32	30.99	25.67	0.75	0.27	0.01
installment	310704	452.84	264.51	387.55	14.77	1618.24	1603.47	0.95	0.55	0.47
annual_inc	310704	80539.98	92862.94	67200.00	0.00	9757200.00	9757200.00	46.99	3628.71	166.60
dti	310556	19.02	12.34	18.19	-1.00	999.00	1000.00	24.10	1587.69	0.02
delinq_2yrs	310704	0.34	0.92	0.00	0.00	21.00	21.00	5.39	49.12	0.00
inq_last_6mths	310703	0.61	0.89	0.00	0.00	5.00	5.00	1.70	3.12	0.00
open_acc	310704	11.88	5.79	11.00	0.00	81.00	81.00	1.33	3.40	0.01
pub_rec	310704	0.25	0.67	0.00	0.00	86.00	86.00	17.17	1369.85	0.00
revol_bal	310704	16052.73	23228.24	10590.00	0.00	1044210.00	1044210.00	9.62	192.60	41.67
revol_util	310491	48.50	24.81	48.00	0.00	182.80	182.80	0.06	-0.80	0.04
total_acc	310704	24.92	12.31	23.00	2.00	176.00	174.00	1.05	2.15	0.02
out_prncp	310704	723.86	3602.73	0.00	0.00	40000.00	40000.00	6.17	42.23	6.46
total_pymnt	310704	13430.48	10010.94	10741.86	0.00	59808.26	59808.26	1.11	0.73	17.96
risk	310704	0.33	0.47	0.00	0.00	1.00	1.00	0.71	-1.50	0.00

Table 1: Descriptive statistics for the Numerical Features

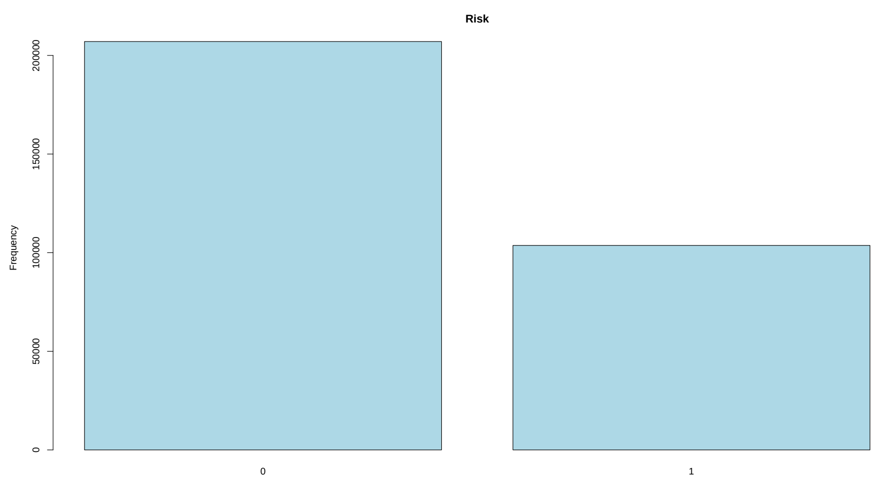

Figure 1: Target Feature Distribution

3 Data Pre-processing

Outliers can lead to vague or misleading predictions while using machine learning models. They decrease the mathematical power of statistical models, and thus, the output of the models becomes unreliable. Removing and modifying outliers using statistical detection techniques, such as Z-score, density-based spatial clustering, regression analysis, proximity-based clustering, and IQR scores, is a widely followed method. We can also use visual detection, such as box plots, to identify outliers.

Missing values are empty cells, rows, and columns that we often see in a dataset. They make the dataset inconsistent and unable to work on. Many machine learning algorithms return an error if parsed with a dataset containing null values. Detecting and treating missing values is essential while analysing and formulating data for any purpose.

3.1 Missing Values

Treating missing values and outliers is an essential step of data cleansing and preparation and should be one of the first operations done on a dataset. Only after these steps are administered is the data considered usable to build models and take insights from. We found the following missing values in the training dataset while performing remedies (we need to write down the methods we used):

- *emp_title*: need correct for missing values of 29563 observations: we considered as the most conservative approach ("Unemployed");
- *emp_lenght*: need correct for missing values of 22615 observations: we considered as the most conservative approach ($<$ 1 year);
- *dti*: need correct for missing values of 148 observations: we decided to apply the mean;
- *ing_last_6mths*: need correct for missing values of 1 observation: we decided to apply the mean;
- *revol_util*: need correct for missing values of 213 observations: we decided to apply the mean.

And we found the following missing values in the validation (test) dataset:

- Dropping the row where *loan_amnt* has missing values since 180 rows are not correctly separated;
- *emp_title*: need correct for missing values of 1224 observations: we considered as the most conservative approach ("Unemployed");
- *emp_lenght*: need correct for missing values of 1210 observations: we considered as the most conservative approach ($<$ 1 year) ;
- *dti*: need correct for missing values of 18 observations: we decided to apply the mean;
- *revol_util*: need correct for missing values of 17 observations: we decided to apply the mean.

3.2 Outliers

Moving to the outliers, the following boxplot shows a general view of the dispersion of the variables used in the training dataset:

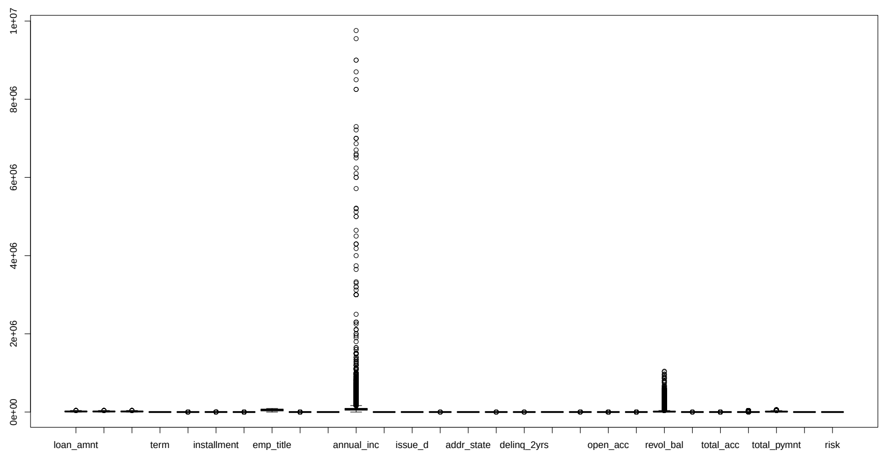

Figure 2: Variables boxplot

We decided to exclude from this checks by outliers the variables *emp_title*, *issue_d* and *earliest_cr_line* due to the high number of levels, which would make it complicated and even impossible for some of them to check for the outliers.

To deal with the outliers, we defined some thresholds for the numerical variables based on the boxplots and histograms of each feature, and all of the observations that didn't comply with that threshold will be automatically excluded. The thresholds are given in Table 2.

We applied the threshold to both the train and test dataset. By this step we conclude the pre-processing part of the project.

3.3 Feature Engineering

The feature engineering process included correlation analysis, data type conversions, mapping of categorical variables to numerical values, and the creation of new features, these steps were crucial in order to improve the model performance and providing meaningful insights from the data.

The initial step was the Numerical Columns Identification, where numerical columns were identified from the *train_dataset* using the *sapply* function with *is.numeric*, which helped in focusing on quantitative data for correlation analysis. Then the second step was the creation of a correlation matrix for the numerical columns and then visualized by using a heatmap (Figure 3), this helped to understand the relationships between different numerical features.

Variable	Threshold
int_rate	25
annual_inc	300000
dti	50
delinq_2yrs	5
inq_last_6mths	3
open_acc	30
pub_rec	3
revol_bal	60000
revol_util	110
total_acc	70
out_prncp	10000
total_pymnt	50000
credit_line _duration	20000
income_to_loan_ratio	40

Table 2: Threshold for the Outliers

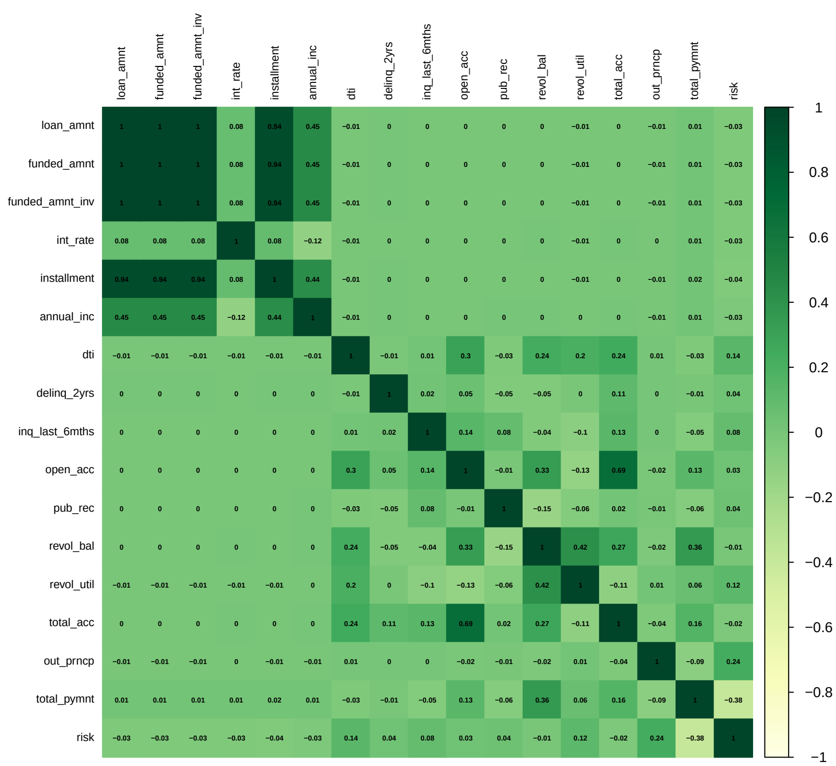

Figure 3: Correlation Heatmap

Also, features like $funded\_amnt$, $funded\_amnt\_inv$, and $installment$ were identified as highly correlated and were subsequently dropped from both training and test datasets to reduce multicollinearity, which could skew model results.

The Employment Length Conversion (*emp_length*), was where the unique values of the *emp_length* feature were inspected, then a mapping was defined to convert these categorical text values into numerical format, representing the number of years in employment, this conversion was applied to both training and test datasets. After the mapping, this feature was explicitly cast to numeric to ensure proper format. Furthermore, the loan term conversion (*term*) was when the feature was converted from a text format (e.g., "36 months") into a pure numeric format representing the number of months, this was achieved using the *gsub* function to remove non-numeric characters and then converting the result to numeric.

Some categorical features were transformed into numerical format using factorization, like: Grade, Address State, Home Ownership, Verification Status, Issue Date, and Purpose, then these features were converted into numeric by first turning them into factors and then mapping these factors to their underlying integer codes, this process was applied to both training and test datasets.

To deal with the *earliest_cr_line* feature which consists of time data, we decided to convert each observation into the difference in days between the current month (January 2024) and the observation itself, obtaining in this way a numerical variable.

A new binary feature was created to indicate whether an individual has been employed for 5 or more years, which could imply employment stability "Employment Stability" (*employment_stability*), then a new feature was calculated by dividing the annual income by the loan amount for each individual in both training and test datasets: "Income to Loan Ratio" (*income_to_loan_ratio*) which represents the proportion of income to the loan amount, which might be indicative of the ability to repay the loan.

A further cleaning process was needed in order to reduce dimensionality and focus on the most impactful variables by dropping high dispersion categorical features, these features with high dispersion or those considered less relevant, such as *emp_title* and *loan_status* were dropped from the training dataset. *emp_title* was also removed from the test dataset. The final structure display was when the structure of both the training and test datasets were displayed by using the *str*() function, providing an overview of the final dataset ready for modeling.

4 Modelling

4.1 Feature Selection

For the feature selection, we started by identifying and selecting numerical columns in the train dataset, creating a new variable called $numerical\_columns$.

Then we calculated the correlations of each numerical feature with the target variable *risk* and sorted their absolute values in descending order. This helps us understand which features have a stronger linear relationship with the target (*risk*), which is excluded from this sorted list (Figure 4).

After that, we created a horizontal bar plot to visually represent these correlations, aiding in the ulterior feature selection.

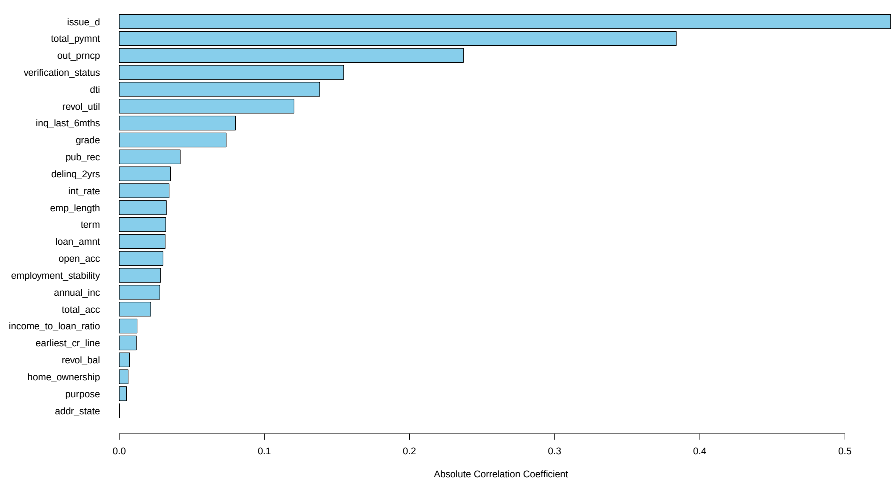

Figure 4: Correlation with the Target Feature 'risk'

We can see the highest correlations with the target variable are with the variables *total_pymnt*, *out_prncp*, *dti*, *revol_util*, and *inq_last_6mths*.

Then we proceeded to remove the weakly correlated features. We considered that 0.001 was the threshold, and only one variable (*addr_state*) with a correlation below that threshold was dropped from both training and testing datasets.

Then we dropped other variables due to different reasons:

- *emp_title*: due to high dispersion;
- *loan_status*: since it represents our target feature *risk*.

4.2 Logistic Regression

We started the modeling part by setting the seed to 123 to ensure the reproducibility of the results and splitting the data into training and test sets. A partition is created to ensure that two-thirds of the data is used for training.

Then we fit a logistic regression model to the training data with risk as the response variable and all other variables as predictors. Then we used a glm function with a binomial family and logit link function, suitable for binary classification.

Then we displayed a summary of the logistic regression model, which includes coefficient estimates, standard errors, z-values, and p-values, among other statistics.

Subsequently, we used the model to predict the risk on the test set. Predictions are probabilities, which are then converted into binary predictions using a threshold of 0.5. Then the performance of the model is evaluated using accuracy, calculated from a confusion matrix. A classification report is also printed, which includes metrics like Sensitivity (or Recall), Specificity, Positive Predictive Value (or Precision) and Negative Predictive Value, among others.

Additionally, we generated a confusion matrix and then normalized it to show proportions instead of absolute numbers, which helps in understanding the model's performance in differentiating between the classes.

	Reference	
Prediction	Non-Default	Default
Non-Default	0.8366906	0.1633094
Default	0.2617874	0.7382126

Table 3: Confusion Matrix Logistic Regression

Indicator	Values
Accuracy Rate	0.8105
Sensitivity	0.8980
Specificity	0.6213
Pos Pred Value	0.8367
Neg Pred Value	0.7382
Balanced Accuracy	0.7597

Table 4: Performance measures Logistic Regression

Finally, a Receiver Operating Characteristic (ROC) curve is plotted, and the Area Under Curve (AUC) score is calculated. The ROC curve is a plot of the true positive rate against the false positive rate at various threshold settings, and the AUC score is a measure of the model's ability to distinguish between the classes.

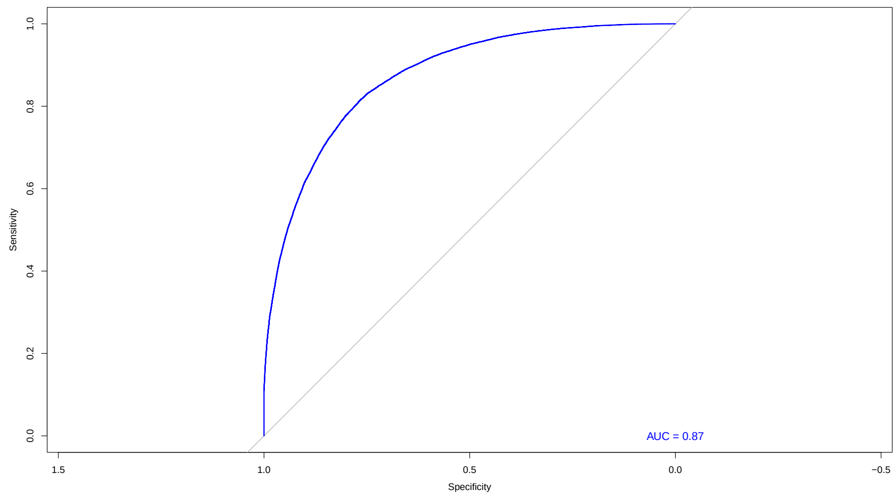

Figure 5: Logistic Regression ROC Curve

4.3 Random Forest

For the random forest model, we created one hundred trees and the result that we got for the Area Under the Curve was 0.9197253, which is higher than the 0.8731501 that we got on the logistic regression. This indicates that the random forest model should present better results.

However, when we are comparing the two models, we need to take into account other measures. Comparing the Random Forest model with the Logistic Regression, we can clearly see that the accuracy of the Random Forest model is higher, another value that is also higher is the specificity which is a very relevant measure for the problem we are solving, it allows us to correctly identify true negatives which will prevent big losses.

For this specific problem we can clearly see that we will have better results if we implement the Random Forest model instead of the Logistic Regression model.

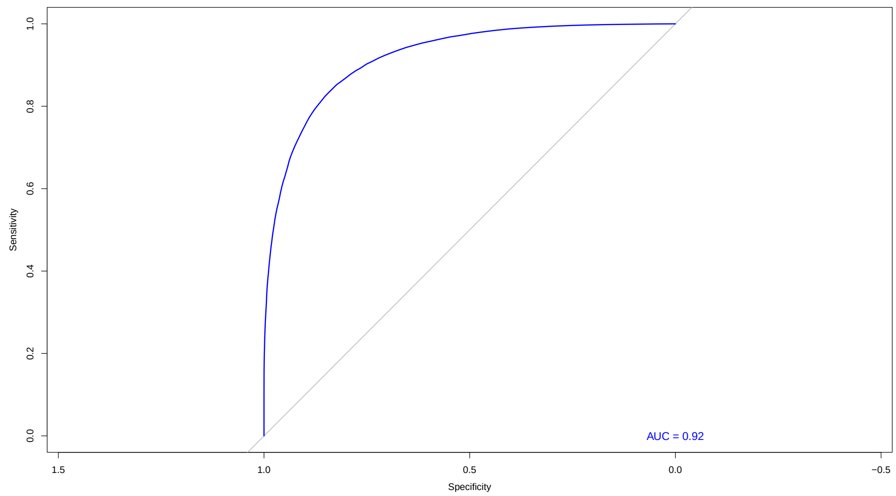

Figure 6: Random Forest ROC Curve

	Reference	
Prediction	Non-Default	Default
Non-Default	0.8552732	0.1447268
Default	0.1582959	0.8417041

Table 5: Confusion Matrix Random Forest

Indicator	Values
Accuracy Rate	0.8519
Sensitivity	0.9430
Specificity	0.6552
Pos Pred Value	0.8553
Neg Pred Value	0.8417
Balanced Accuracy	0.7978

Table 6: Performance measures Random Forest

4.4 Deep Neural Network

Finally, the Neural Network model when compared to the Random Forest model, does not outperform it in any measure, like the sensitivity and the Area Under the Curve and it underperformed in the specificity with 0.2568 when compared with 0.6552 of the Random Forrest.

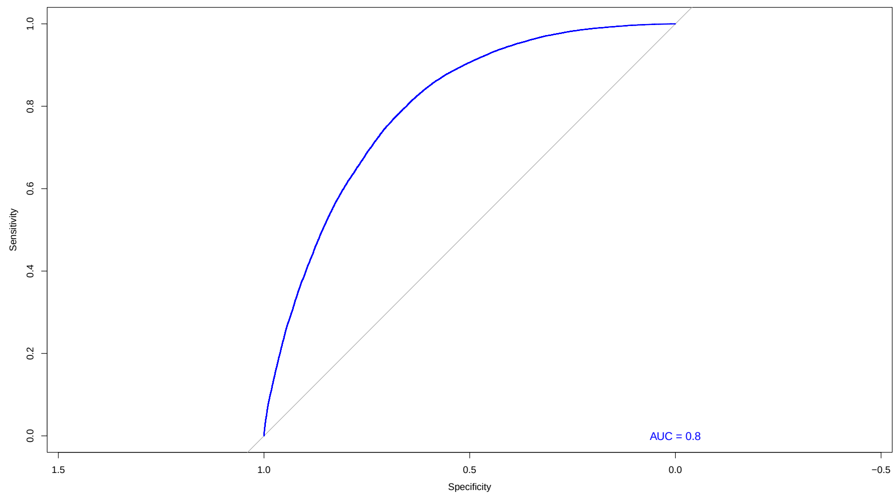

Figure 7: Deep Neural Network ROC Curve

	Reference	
Prediction	Non-Default	Default
Non-Default	0.7339719	0.2660281
Default	0.3096887	0.6903113

Table 7: Confusion Matrix Deep Neural Network

Indicator	Values
Accuracy Rate	0.7288
Sensitivity	0.9468
Specificity	0.2568
Pos Pred Value	0.7340
Neg Pred Value	0.6903
Balanced Accuracy	0.6018

Table 8: Performance measures Deep Neural Network

Comparing the three models that we tested the one with the worst results was the Neural Network, with such low specificity if we applied this model we would incur in very big losses because we would not be able to accurately predict the True Negatives. The Logistic Regression managed to outperform the Neural Network but is outperformed by the Random Forrest which has a better AUC and most importantly the specificity.

In order to have better results we choose to implement the Random Forrest model.

5 Conclusion

In this analysis, we employed three distinct Machine Learning Models (Logistic Regression, Random Forest, and a Deep Neural Network) to predict the probability of default in a credit dataset. The Logistic Regression model demonstrated a commendable overall accuracy of 81.05%, with a balanced accuracy of 75.97%. It exhibited strong sensitivity (true positive rate) of 89.80%, emphasizing its effectiveness in correctly identifying instances of default. The Random Forest model outperformed the Logistic Regression with an impressive accuracy of 85.1% and a balanced accuracy of 79.78%. It showcased high sensitivity (94.25%), indicating its proficiency in capturing true positives. However, the Deep Neural Network, while achieving an overall accuracy of 72.88%, struggled with specificity, resulting in a lower balanced accuracy of 60.18%. This suggests that the deep neural network may be more prone to false positives. In summary, the random forest model emerged as the most robust performer in predicting default probabilities, striking a balance between accuracy and sensitivity. These findings underscore the importance of selecting an appropriate model based on the specific needs and priorities of the credit risk assessment task. Future work could explore model fine-tuning and ensemble techniques to further enhance predictive performance and provide more robust risk assessment in the realm of credit evaluation.

References

[1] Adams, K., & Shin, D. (2022), *The history of credit score algorithms and how they became the lender standard.*
Available at: https://www.marketplace.org/shows/marketplace-tech/the-history-of-credit-score-algorithms-and-how-they-became-the-lender-standard/

[2] CFI Team (2022), *Credit Risk Analysis Models, Tools used to determine the probability of default of a potential borrower.*
Available at: https://corporatefinanceinstitute.com/resources/commercial-lending/credit-risk-analysis-models/#:~:text=Credit%20risk%20arises%20when%20a,debt%20extended%20to%20a%20borrower.

[3] Jain, J. & Banka, P. (2022), *Dealing with outliers and missing values in a dataset.*
Available at: https://www.neenopal.com/dealing-with-outliers-and-missing-values-in-a-dataset.html

[4] Mongelard, J. (2019), *A brief history in credit risk in banking.*
Available at: https://www.icaew.com/technical/financial-services/financial-services-faculty/fs-focus-magazine/previous-editions-of-fs-focus/fs-focus-2019-issues/november-2019/brief-history-in-credit-risk-in-banking

[5] Pilz, J., Melas, V.B., & Bathke, A. (2023). *Statistical Modeling and Simulation for Experimental Design and Machine Learning Applications.* Springer Nature, pp.245-265.

[6] Zach (2022), *How to Perform Exploratory Data Analysis in R (With Example).*
Available at: https://www.statology.org/exploratory-data-analysis-in-r/

Code

```
if (!require("psych")) install.packages("psych")
library(psych)
if (!require("dplyr")) install.packages("dplyr")
library(dplyr)
if (!require("gmodels")) install.packages("gmodels")
library(gmodels)
if (!require("tidyverse")) install.packages("tidyverse")
library(tidyverse)
if (!require("corrplot")) install.packages("corrplot")
library(corrplot)
if (!require("caret")) install.packages("caret")
library(caret)
if (!require("pROC")) install.packages("pROC")
library(pROC)
if (!require("ROCR")) install.packages("ROCR")
library(ROCR)
if (!require("randomForest")) install.packages("randomForest")
library(randomForest)
if (!require("keras")) install.packages("keras")
library(keras)
if (!require("tensorflow")) install.packages("tensorflow")
library(tensorflow)
if (!require("lubridate")) install.packages("lubridate")
library(lubridate)


# Loading Data
# ----------------------------------------------------------------------------
train_dataset <- read.csv("/Users/riccardogurzu/Desktop/predictive analytics/train_validation_kaggle.csv",
                          na.strings = c("", "n/a"),
                          stringsAsFactors = T)
# Remove the 'id' column from the dataset
train_dataset <- train_dataset[, !(names(train_dataset) %in% c("id"))]

test_dataset <- read.csv("/Users/riccardogurzu/Desktop/predictive analytics/unseen_kaggle.csv",
                         na.strings = c("", "n/a"),
                         stringsAsFactors = T)

# View datasets
View(train_dataset)
View(test_dataset)

#Check details
summary(train_dataset)
summary(test_dataset)
head(train_dataset, n=5)
tail(train_dataset, n=5)
describe(train_dataset)
describe(test_dataset)

str(train_dataset)
str(test_dataset)

```

```

# Create a bar plot of the 'risk' to check the balance of the dataset
barplot(table(train_dataset$risk),
        col = c("lightblue"),
        main = "Risk",
        ylab = "Frequency")


# ---------------------------------------------------------------------------
#Data #Pre-Processing
# ---------------------------------------------------------------------------
##Data-Type
str(train_dataset)
str(test_dataset)

# ---------------------------------------------------------------------------
##Missing Values

###Train Dataset
sapply(train_dataset, function(x) sum(is.na(x)))

###emp_title - 29563 missing values
train_dataset$emp_title = factor(train_dataset$emp_title,
                                 levels=c(levels(train_dataset$emp_title), 'Unemployed'))
train_dataset$emp_title[is.na(train_dataset$emp_title)] <- 'Unemployed'

###emp_length - 22615 missing values
train_dataset$emp_length[is.na(train_dataset$emp_length)] <- '< 1 year'

###dti - 148 missing values
mean(train_dataset$dti, na.rm = TRUE)
train_dataset$dti[is.na(train_dataset$dti)] <- mean(train_dataset$dti, na.rm = TRUE)

###inq_last_6mths - 1 missing values
mean(train_dataset$inq_last_6mths, na.rm = TRUE)
train_dataset$inq_last_6mths[is.na(train_dataset$inq_last_6mths)] <- mean(train_dataset$inq_last_6mths, na.rm = TRUE)

###revol_util - 213 missing values
mean(train_dataset$revol_util, na.rm = TRUE)
train_dataset$revol_util[is.na(train_dataset$revol_util)] <- mean(train_dataset$revol_util, na.rm = TRUE)

sapply(train_dataset, function(x) sum(is.na(x)))



###Test Dataset

str(test_dataset)
sapply(test_dataset, function(x) sum(is.na(x)))

### dropping the row where 'loan_amnt' has missing values since 180 rows are not correctly separated
### due to values in emp_title with double quotes which contain commas
```

```
test_dataset <- test_dataset[complete.cases(test_dataset$loan_amnt), ]

sapply(test_dataset, function(x) sum(is.na(x)))

###emp_title - 1224 missing values
test_dataset$emp_title = factor(test_dataset$emp_title,
                                levels=c(levels(test_dataset$emp_title),
                                ↪  'Unemployed'))
test_dataset$emp_title[is.na(test_dataset$emp_title)] <- 'Unemployed'

###emp_length - 1210 missing values
test_dataset$emp_length[is.na(test_dataset$emp_length)] <- '< 1 year'

###dti - 18 missing values
test_dataset$dti <- as.numeric(test_dataset$dti)
mean(test_dataset$dti, na.rm = TRUE)
test_dataset$dti[is.na(test_dataset$dti)] <- mean(test_dataset$dti, na.rm =
↪  TRUE)

###revol_util - 17 missing values
test_dataset$revol_util <- as.numeric(test_dataset$revol_util)
mean(test_dataset$revol_util, na.rm = TRUE)
test_dataset$revol_util[is.na(test_dataset$revol_util)] <-
↪  mean(test_dataset$revol_util, na.rm = TRUE)


sapply(test_dataset, function(x) sum(is.na(x)))


# ---------------------------------------------------------------------------

##Outliers
boxplot(train_dataset)
str(train_dataset)

numerical_columns <- sapply(train_dataset, is.numeric)

#print the boxplot for the numerical features
for (col in names(train_dataset[, numerical_columns])) {
  boxplot(train_dataset[, col],
          main = col,
          col = "skyblue",
          border = "black")
}

#print the histogram for the numerical features
for (col in names(train_dataset[, numerical_columns])) {
  hist(train_dataset[, col],
       main = col,
       col = "skyblue",
       border = "black",
       xlim = c(min(train_dataset[, col]), max(train_dataset[, col])),
       breaks = 30)
}

# for categorical features
categorical_columns <- sapply(train_dataset, is.factor)
```

```
# exclude emp_title, issue_d and earliest_cr_line due to the high number of
↪  levels
exclude_features <- c("emp_title", "issue_d", "earliest_cr_line")

# Identify the indices of the categorical columns to keep
categorical_columns <- !(names(train_dataset) %in% exclude_features) &
↪  categorical_columns

# Loop through each categorical column and plot bar charts
for (col in names(train_dataset[, categorical_columns])) {
  barplot(table(train_dataset[, col]),
          main = col,
          col = "skyblue",
          xlab = "Categories",
          ylab = "Count")
}


# Define threshold values
threshold_values <- list(
  int_rate = 25,
  annual_inc = 0.3 * 10^6,
  dti = 50,
  delinq_2yrs = 5,
  inq_last_6mths = 3,
  open_acc = 30,
  pub_rec = 3,
  revol_bal = 60000,
  revol_util = 110,
  total_acc = 70,
  out_prncp = 10000,
  total_pymnt = 50000,
  credit_line_duration = 20000,
  income_to_loan_ratio = 40
)

# Loop through each column and apply the corresponding threshold for the Train
↪  Dataset
for (column in names(train_dataset)) {
  if (column %in% names(threshold_values)) {
    train_dataset <- train_dataset[train_dataset[, column] <=
    ↪  threshold_values[[column]], ]
  }
}

# Loop through each column and apply the corresponding threshold for the Test
↪  Dataset
for (column in names(test_dataset)) {
  if (column %in% names(threshold_values)) {
    test_dataset <- test_dataset[test_dataset[, column] <=
    ↪  threshold_values[[column]], ]
  }
}

#print the boxplot for the numerical features after dropped the outliers
for (col in names(train_dataset[, numerical_columns])) {
  boxplot(train_dataset[, col],
```

```
          main = col,
          col = "skyblue",
          border = "black")
}

#print the histogram for the numerical features after dropped the outliers
for (col in names(train_dataset[, numerical_columns])) {
  hist(train_dataset[, col],
       main = col,
       col = "skyblue", border = "black",
       xlim = c(min(train_dataset[, col]), max(train_dataset[, col])),
       breaks = 30)
}

# ---------------------------------------------------------------------------
#Feature Engineering
# ---------------------------------------------------------------------------

# Select the columns with numerical data
numerical_columns <- sapply(train_dataset, is.numeric)

# Create a correlation matrix
correlation_matrix <- cor(train_dataset[, numerical_columns])
correlation_matrix

# Plot the correlation matrix heatmap
corrplot(correlation_matrix,
         method = "color",
         tl.col = "black",
         col = COL1('YlGn'),
         addCoef.col = "black",
         tl.cex = 0.7,
         number.cex = 0.4)

#drop high correlated features (>|0.7|)
train_dataset <- train_dataset[, -which(names(train_dataset) %in%
↪  c("funded_amnt", "funded_amnt_inv", "installment"))]
test_dataset <- test_dataset[, -which(names(test_dataset) %in%
↪  c("funded_amnt", "funded_amnt_inv", "installment"))]


### Feature Transformation
str(train_dataset)
str(test_dataset)

# removing the dot in total_pymnt. to match train dataset
colnames(test_dataset)[colnames(test_dataset) == 'total_pymnt.'] <-
↪  'total_pymnt'
# Convert total_pymnt into numeric
test_dataset$total_pymnt <- as.numeric(test_dataset$total_pymnt)


# Convert emp_length into numeric
unique(train_dataset$emp_length)
mapping <- c(
  "< 1 year" = 0,
  "1 year" = 1,
```

```
  "2 years" = 2,
  "3 years" = 3,
  "4 years" = 4,
  "5 years" = 5,
  "6 years" = 6,
  "7 years" = 7,
  "8 years" = 8,
  "9 years" = 9,
  "10+ years" = 10
)

train_dataset <- train_dataset %>%
  mutate(emp_length = case_when(
    emp_length %in% names(mapping) ~ mapping[as.character(emp_length)],
    TRUE ~ NA_real_
  ))
train_dataset$emp_length <- as.numeric(train_dataset$emp_length)
unique(train_dataset$emp_length)

test_dataset <- test_dataset %>%
  mutate(emp_length = case_when(
    emp_length %in% names(mapping) ~ mapping[as.character(emp_length)],
    TRUE ~ NA_real_
  ))
test_dataset$emp_length <- as.numeric(test_dataset$emp_length)
unique(test_dataset$emp_length)

# Convert term into numeric
train_dataset$term <- as.numeric(gsub("[^0-9]+", "", train_dataset$term))
test_dataset$term <- as.numeric(gsub("[^0-9]+", "", test_dataset$term))


# Transform the levels for the categoric features (except emp_title and
↪  earliest_cr_line) into numeric
unique(train_dataset$grade)
train_dataset$grade <- as.numeric(factor(train_dataset$grade,
                                         levels =
                                         ↪  levels(train_dataset$grade)))

unique(test_dataset$grade)
test_dataset$grade <- as.numeric(factor(test_dataset$grade,
                                         levels = levels(test_dataset$grade)))



unique(train_dataset$addr_state)
train_dataset$addr_state <- as.numeric(factor(train_dataset$addr_state,
                                              levels =
                                              ↪  levels(train_dataset$addr_state)))

unique(test_dataset$addr_state)
test_dataset$addr_state <- as.numeric(factor(test_dataset$addr_state,
                                              levels =
                                              ↪  levels(test_dataset$addr_state)))


unique(train_dataset$home_ownership)
```

```
train_dataset$home_ownership <-
↪  as.numeric(factor(train_dataset$home_ownership,
                                                    levels =
                                                    ↪  levels(train_dataset$home_ownership)))
unique(test_dataset$home_ownership)
test_dataset$home_ownership <- as.numeric(factor(test_dataset$home_ownership,
                                                    levels =
                                                    ↪  levels(test_dataset$home_ownership)))


unique(train_dataset$verification_status)
train_dataset$verification_status <-
↪  as.numeric(factor(train_dataset$verification_status,
                                                         levels =
                                                         ↪  levels(train_dataset$verification_status)))

unique(test_dataset$verification_status)
test_dataset$verification_status <-
↪  as.numeric(factor(test_dataset$verification_status,
                                                         levels =
                                                         ↪  levels(test_dataset$verification_status)))


unique(train_dataset$purpose)
train_dataset$purpose <- as.numeric(factor(train_dataset$purpose,
                                           levels =
                                           ↪  levels(train_dataset$purpose)))
unique(test_dataset$purpose)
test_dataset$purpose <- as.numeric(factor(test_dataset$purpose,
                                           levels =
                                           ↪  levels(test_dataset$purpose)))


# Create 'employment_stability' feature
train_dataset$employment_stability <- ifelse(train_dataset$emp_length >= 5, 1,
↪  0)
test_dataset$employment_stability <- ifelse(test_dataset$emp_length >= 5, 1,
↪  0)

# Create 'income_to_loan_ratio' feature
train_dataset$income_to_loan_ratio <-
↪  train_dataset$annual_inc/train_dataset$loan_amnt
test_dataset$annual_inc <- as.numeric(test_dataset$annual_inc)
test_dataset$income_to_loan_ratio <-
↪  test_dataset$annual_inc/test_dataset$loan_amnt


## To convert earliest_cr_line into numeric, we modify the feature computing
↪  the diff with the current month
# Convert the 'earliest_cr_line' column to a Date type with custom format
train_dataset$earliest_cr_line <- dmy(paste0("01-",
↪  train_dataset$earliest_cr_line))
unique(train_dataset$earliest_cr_line)
# Adjust two-digit years using if_else
train_dataset$earliest_cr_line <- if_else(year(train_dataset$earliest_cr_line)
↪  <= 2024,
                                          train_dataset$earliest_cr_line,
```

```
                                              train_dataset$earliest_cr_line - years(100))
unique(train_dataset$earliest_cr_line)

test_dataset$earliest_cr_line <- as.Date(test_dataset$earliest_cr_line, origin = "1899-12-30")
unique(train_dataset$earliest_cr_line)
unique(test_dataset$earliest_cr_line)

# Calculate the difference in months between each date and the reference date
current_date <- dmy("01-01-2024")
train_dataset$earliest_cr_line <- as.numeric(interval(train_dataset$earliest_cr_line, current_date) / months(1))
test_dataset$earliest_cr_line <- as.numeric(interval(test_dataset$earliest_cr_line, current_date) / months(1))
unique(train_dataset$earliest_cr_line)
unique(test_dataset$earliest_cr_line)

unique(train_dataset$issue_d)
unique(test_dataset$issue_d)

# Convert numerical labels to Date type
test_dataset$issue_d <- as.Date(test_dataset$issue_d, origin = "1899-12-30")
unique(test_dataset$issue_d)
# since levels of issue_d between train and test are different, we cannot use label encoding
# thus, we modify the issue_d feature into the diff in months between issue_d and current month

# Convert 'issue_d' to a Date type
train_dataset$issue_d <- dmy(paste0("01-", train_dataset$issue_d))
unique(train_dataset$issue_d)

current_date <- dmy("01-01-2024")

# Calculate the difference in months between each date and the current date
train_dataset$issue_d <- as.numeric(interval(train_dataset$issue_d, current_date) / months(1))
test_dataset$issue_d <- as.numeric(interval(test_dataset$issue_d, current_date) / months(1))
unique(train_dataset$issue_d)
unique(test_dataset$issue_d)


# Display the result
str(train_dataset)
str(test_dataset)


### Plot the correlation between the target feature 'Risk' and the other features

# Select the columns with numerical data
numerical_columns <- sapply(train_dataset, is.numeric)

```

```
# Calculate the correlation with the target variable
correlation_with_target <- sapply(train_dataset[, numerical_columns],
                                  function(x) cor(x, train_dataset$risk))

# Order correlations by absolute values in descending order
sorted_correlations <- sort(abs(correlation_with_target), decreasing = FALSE)

# Exclude the 'risk' variable
sorted_correlations <- sorted_correlations[!(names(sorted_correlations) %in%
↪  "risk")]

# Set larger plot margins to fit the y labels
par(mar = c(5, 10, 2, 2))

# Create a barplot with the correlations to 'risk'
barplot(sorted_correlations, horiz = TRUE, col = "skyblue",
        xlab = "Absolute Correlation Coefficient",
        las = 1)

# Set plot margins to default
par(mar = c(5, 4, 4, 2))

# Extract column names with correlation less than 0.01 and drop them
columns_to_drop <- names(correlation_with_target[abs(correlation_with_target)
↪  < 0.001]) #only addr_state meet this condition
train_dataset <- train_dataset[, !names(train_dataset) %in% columns_to_drop]
test_dataset <- test_dataset[, !names(test_dataset) %in% columns_to_drop]

#drop emp_title due to high dispersion
#drop loan_status since represent our target feature 'risk'
#drop issue_d since if we include it we get for some reason a perfect
↪  performance
train_df <- train_dataset[, -which(names(train_dataset) %in% c("emp_title",
↪  "loan_status", "issue_d"))]
test_df <- test_dataset[, -which(names(test_dataset) %in% c("emp_title",
↪  "issue_d"))]

# Display the result
str(train_df)
str(test_df)


# ---------------------------------------------------------------------------
# Logistic regression
# ---------------------------------------------------------------------------

#Breaking Data into Training and Test Sample
set.seed(123)

# Data splitting
index <- createDataPartition(train_df$risk, p = 2/3, list = FALSE)
train <- train_df[index, ]
test <- train_df[-index, ]

# Fit the logistic regression model
model_log <- glm(risk ~ .,
             data = train,
```

```
             family = binomial(link = "logit"))

# Summary of the model
summary(model_log)

# Make predictions on the test set
y_pred <- predict(model_log, newdata = test, type = "response")

# Convert probabilities to binary predictions (assuming a threshold of 0.5)
y_pred_binary <- ifelse(y_pred > 0.5, 1, 0)
y_pred_binary <- as.factor(y_pred_binary)

## Evaluate the model
test$risk <- as.factor(test$risk)
# Accuracy
accuracy <- confusionMatrix(data = y_pred_binary,
                            reference = test$risk)$overall["Accuracy"]
cat("Accuracy:", accuracy, "\n")

# Classification Report
conf_matrix <- confusionMatrix(data = y_pred_binary, reference = test$risk)
print(conf_matrix)

# Confusion Matrix
conf_matrix <- as.table(conf_matrix)
conf_matrix <- prop.table(conf_matrix, 1)

# Plot the correlation matrix
conf_matrix

#Plot ROC Curve
roccurve <- roc(test$risk ~ y_pred)
auc_score <- auc(roccurve)
cat("Logistic Regression AUC Score:", auc_score, "\n")

plot(roccurve, col='blue')
# Add AUC value as text annotation
text(0, 0, paste("AUC =", round(auc_score, 2)), col="blue", cex=1.2)



# ----------------------------------------------------------------------------
# Random Forest
# ----------------------------------------------------------------------------

#Breaking Data into Training and Test Sample
set.seed(123)

# Data splitting
index <- createDataPartition(train_df$risk, p = 2/3, list = FALSE)
train <- train_df[index, ]
test <- train_df[-index, ]

train$risk <- as.factor(train$risk)

# Fit the Random Forest model
model_rf <- randomForest(risk ~ .,
```

```
                            data = train,
                            ntree = 100)

# Make predictions on the test set
y_pred_rf <- predict(model_rf, type = "prob", test)

# Convert probabilities to binary predictions (assuming a threshold of 0.5)
y_pred_binary_rf <- ifelse(y_pred_rf[,2] > 0.5, 1, 0)
y_pred_binary_rf <- as.factor(y_pred_binary_rf)

## Evaluate the Random Forest model
test$risk <- as.factor(test$risk)

# Accuracy
accuracy_rf <- confusionMatrix(data = y_pred_binary_rf, reference =
↪  test$risk)$overall["Accuracy"]
cat("Random Forest Accuracy:", accuracy_rf, "\n")

# Classification Report
conf_matrix_rf <- confusionMatrix(data = y_pred_binary_rf, reference =
↪  test$risk)
print(conf_matrix_rf)

# Confusion Matrix
conf_matrix_rf <- as.table(conf_matrix_rf)
conf_matrix_rf <- prop.table(conf_matrix_rf, 1)

# Plot the confusion matrix
conf_matrix_rf

# Plot ROC Curve for Random Forest
roccurve_rf <- roc(test$risk ~ y_pred_rf[,2])
# AUC Score for Random Forest
auc_score_rf <- auc(roccurve_rf)
cat("Random Forest AUC Score:", auc_score_rf, "\n")
plot(roccurve_rf, col = 'blue')
# Add AUC value as text annotation
text(0, 0, paste("AUC =", round(auc_score_rf, 2)), col="blue", cex=1.2)


# ---------------------------------------------------------------------------
# Deep Neural Network
# ---------------------------------------------------------------------------

install_tensorflow(envname = "r-tensorflow")

tf$constant("Hello TensorFlow!")

#Breaking Data into Training and Test Sample
set.seed(123)

# Data splitting
index <- createDataPartition(train_df$risk, p = 2/3, list = FALSE)
train <- train_df[index, ]
test <- train_df[-index, ]

y_train <- as.numeric(train$risk)
```

```
y_test <- as.numeric(test$risk)
x_train <- train %>% select(-risk)
x_test <- test %>% select(-risk)

# Normalize hte input features
x_train <- as.matrix(apply(x_train, 2, function(x) (x-min(x))/(max(x) - min(x))))
x_test <- as.matrix(apply(x_test, 2, function(x) (x-min(x))/(max(x) - min(x))))


# Fit the Neural Network model
model_nn <- keras_model_sequential() %>%
  layer_dense(units = 64, activation = "relu", input_shape = ncol(train) - 1) %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

compile(model_nn, optimizer = "adam",
        loss = "binary_crossentropy",
        metrics = c("accuracy"))

# Train the model
history <- fit(
  model_nn,
  x = x_train,
  y = y_train,
  shuffle = T,
  epochs = 10,
  batch_size = 32,
  validation_split = 0.2
)

# Make predictions on the test set
y_pred_prob_nn <- predict(model_nn, x_test)

# Convert probabilities to binary predictions
y_pred_binary_nn <- ifelse(y_pred_prob_nn > 0.5, 1, 0)
y_pred_binary_nn <- as.factor(y_pred_binary_nn)

# Evaluate the neural network model
y_test <- as.factor(y_test)
# Accuracy
accuracy_nn <- confusionMatrix(data = y_pred_binary_nn, reference = y_test)$overall["Accuracy"]
cat("Neural Network Accuracy:", accuracy_nn, "\n")

# Classification Report
conf_matrix_nn <- confusionMatrix(data = y_pred_binary_nn, reference = y_test)
print(conf_matrix_nn)

# Confusion Matrix
conf_matrix_nn <- as.table(conf_matrix_nn)
conf_matrix_nn <- prop.table(conf_matrix_nn, 1)

# Plot the confusion matrix
conf_matrix_nn
```

```

# Plot ROC Curve for Neural Network
roccurve_nn <- roc(y_test, as.numeric(y_pred_prob_nn))
# AUC Score for Neural Network
auc_score_nn <- auc(roccurve_nn)
cat("Neural Network AUC Score:", auc_score_nn, "\n")
plot(roccurve_nn, col = 'blue')
# Add AUC value as text annotation
text(0, 0, paste("AUC =", round(auc_score_nn, 2)), col="blue", cex=1.2)


# ----------------------------------------------------------------------------
# Deployment
# ----------------------------------------------------------------------------
# Based on the evaluation metrics, Random Forest is the model which perform
↪  better

# Make predictions on the unseen file using the trained model
unseen_df <- test_df[, !(names(test_df) %in% c("id"))]
y_pred_rf <- predict(model_rf, type = "prob", unseen_df)

# Convert probabilities to binary predictions (assuming a threshold of 0.5)
y_pred_binary_rf <- ifelse(y_pred_rf[,2] > 0.5, 1, 0)
y_pred_binary_rf <- as.factor(y_pred_binary_rf)

RandomForest <- data.frame(id=test_df$id, risk=y_pred_binary_rf)
RandomForest

write.csv(RandomForest,"Group5.csv",row.names = FALSE)
```