Business Analytics Second Project

MGT 7104

Yacine Ndiaye

April 14th,2025

Instructor: Jaime Velasco Sanchez

Richmond The American University in London

Word Count: 1855

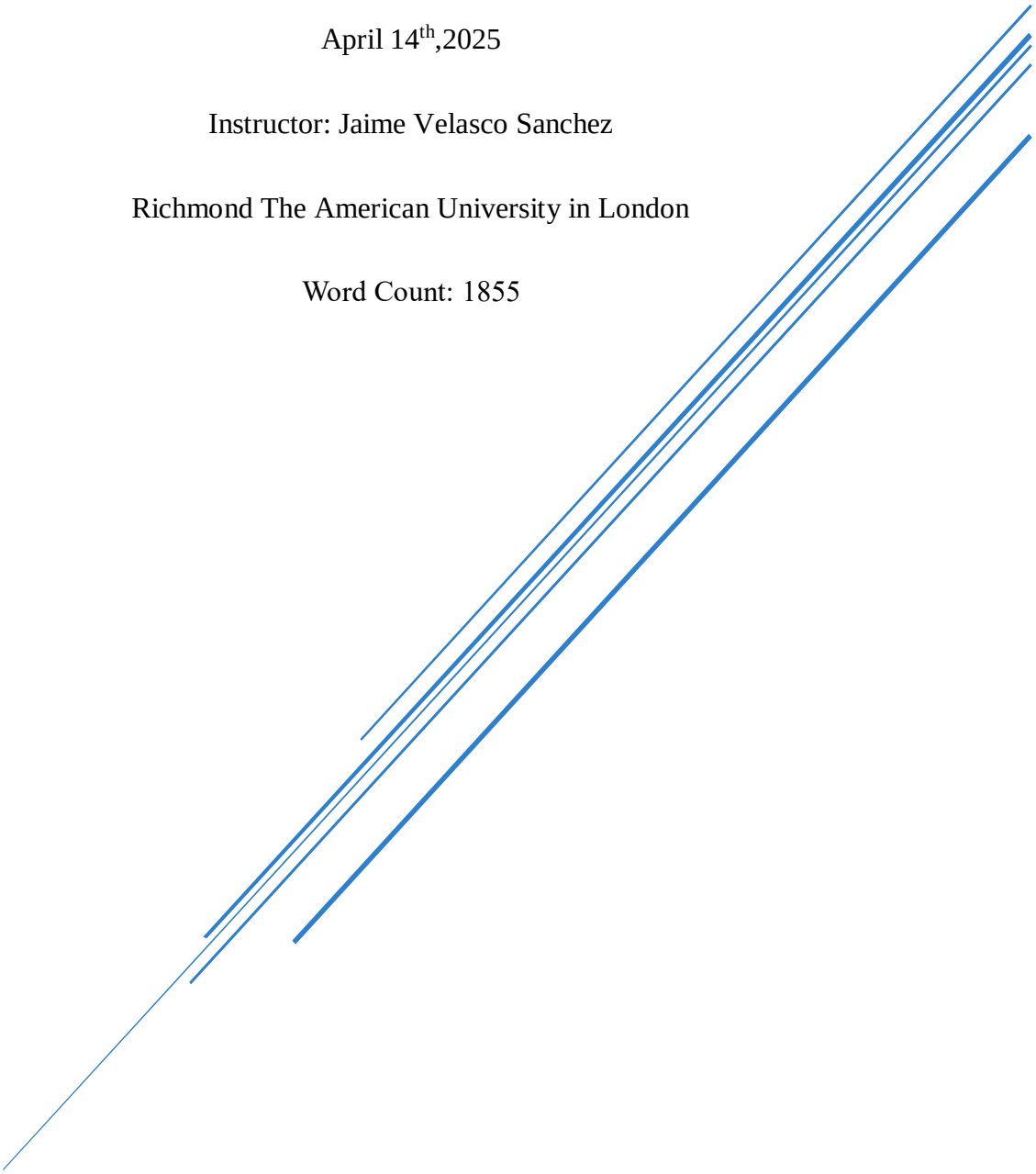
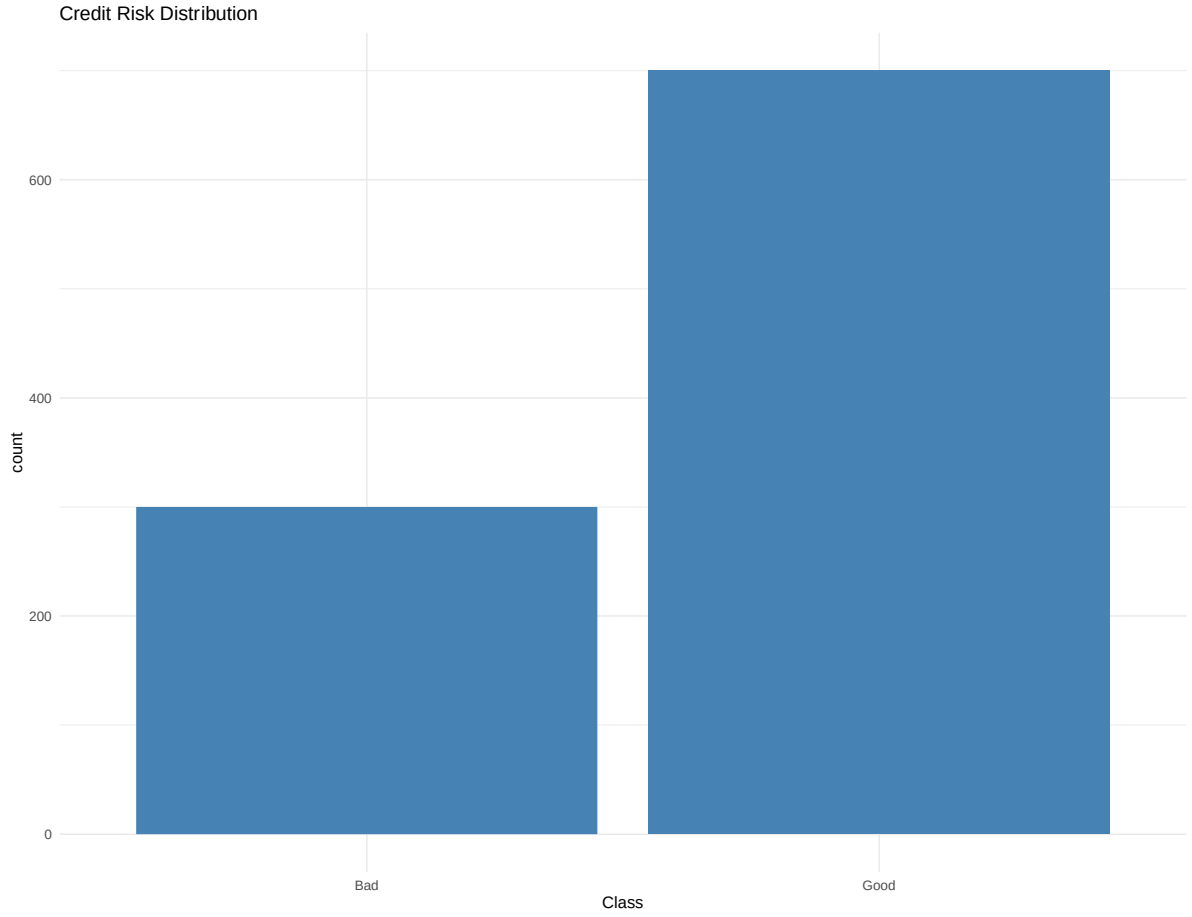


TABLE OF CONTENTS

I. GERMAN CREDIT DATASET

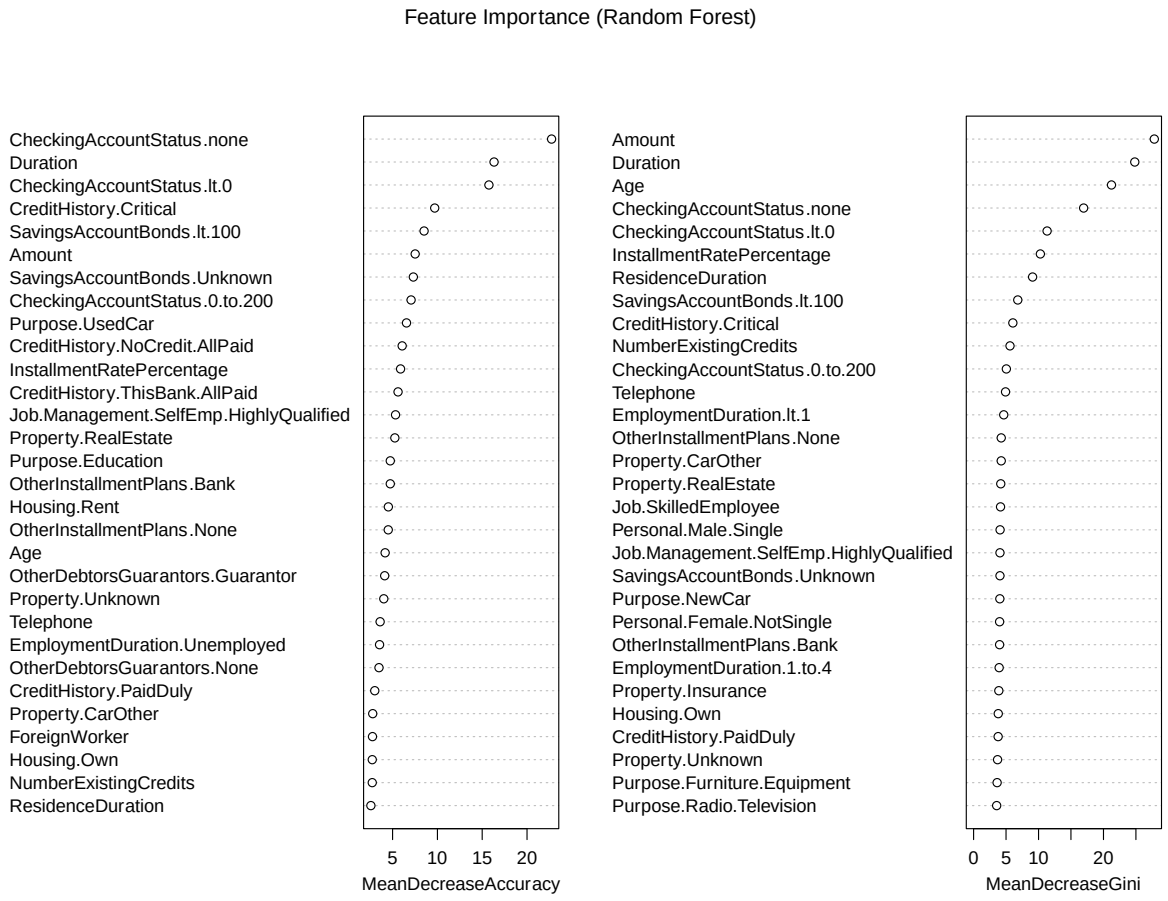
The German credit dataset is the object of analysis of this paper. It contains 1000 observations across 62 variables.

The graph below shows how many customers are into each credit risk category. A clear imbalance can be noticed with over 50% of customers being categorized as good credit risk. This will be accounted. For moving forward as it will affect model performance.



This graph on the other hand rank predictors on the basis of their contribution to the model's classification accuracy. The "mean decrease Accuracy" measures the drop in the model's overall prediction accuracy due to the random permutation of a given variable. For variable with high MDA such as "checking account status", "duration" and "credit history", their removal will greatly influence the model's prediction accuracy.

"mean decrease in Gini" on the x-axis measures how much a variable helps improve the model's ability to separate good and bad credit risk ; the most important ones are "amount", "duration" and "age".



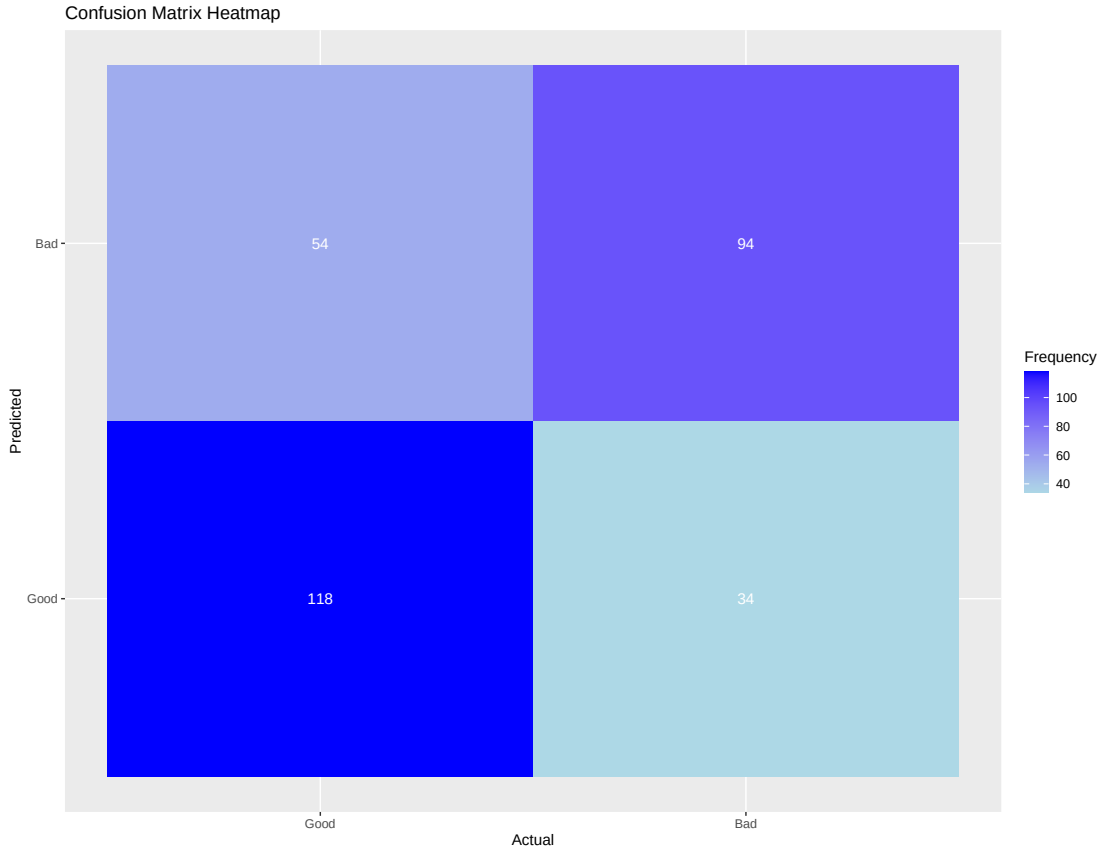
II. NAÏVE BAYES

The Naïve bayes model is a type of probabilistic classifier that bases its predictions off Bayes' theorem which helps calculate conditional probabilities along with the "naïve" assumption of the independence of a particular feature upon the presence of other features (kayvasrirelangi, 2024).

The presence of both numerical and categorical data within the dataset and the provision of interpretable probability scores which allows the categorization of a loan as "good" or "bad" suggest that the model might be a good fit for loan predictions.

Prior to the implementation of the naïve bayes model, data was split into 70% training and 30% testing sets. Packages such as ROSE (Random oversampling examples), and Laplace were used to increase accuracy by handling imbalances within the dataset.

The confusion matrix generated from the naïve bayes model compares predicted vs actual labels. The model accurately classifies 70% (0.706) of customers as "good" or "bad" credit risk suggesting a moderately good performance. Visualization of the confusion matrix shows 118 true positives or accurately predicted "good" credit risks; 94 true negatives or accurately predicted "bad credit risks" and respectively, 54 and 34 false negative and false positive that have been falsely classified as the opposite.



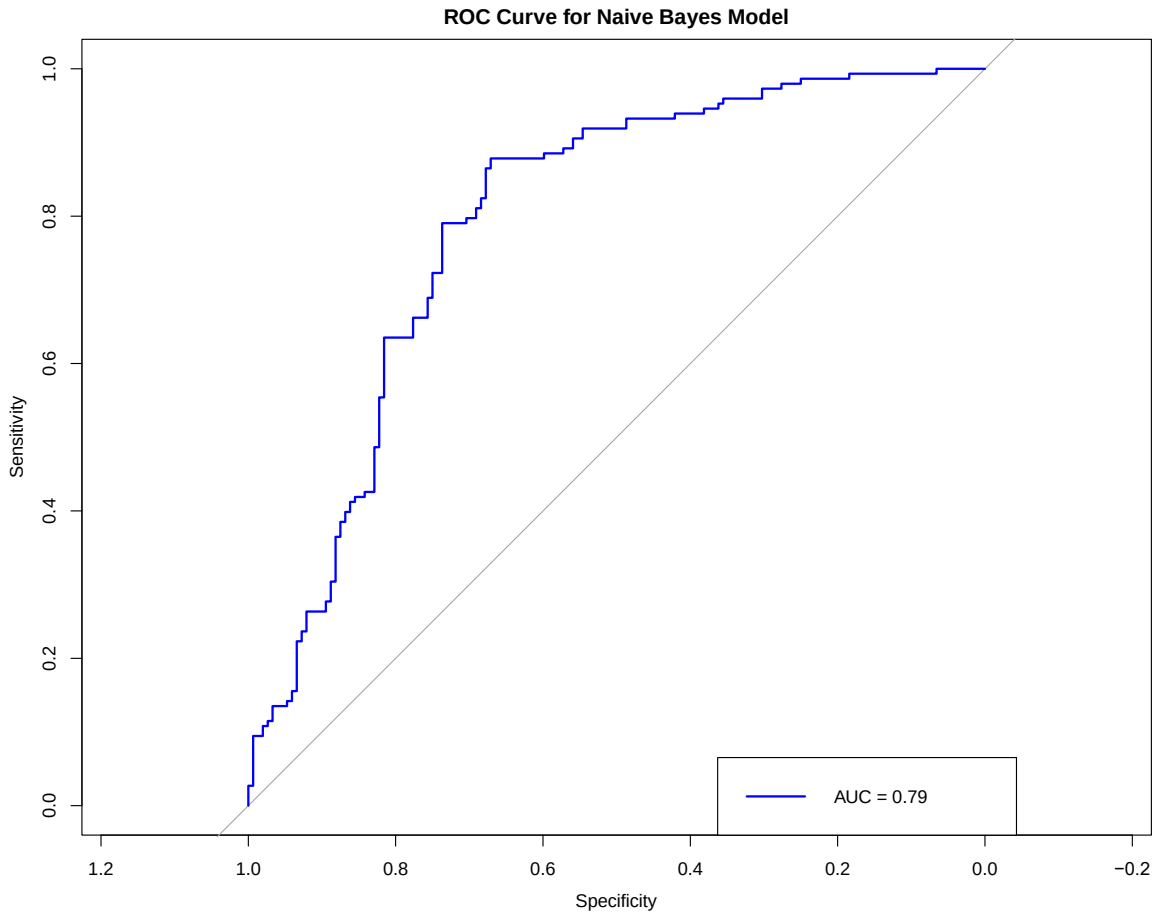
The model's performance metrics indicate a 68% precision (0.686); 77% recall (0.776) and 72% (0.728) F1 score.

Precision percentile means that the model's prediction of good credit risk is correct 68% of the time. A high precision (above 50%) means lower false positives and/or higher true negatives which maximizes the chances of approving good loans.

Recall results shows that the company correctly identifies 77% of good credit risks, ensuring that majority of credit worthy applicants are rightfully approved. Recall percentage being higher than precision suggests acceptable prediction accuracy.

The F1 score (72%) meant to showcase the balance between precision and recall for high accuracy predictions is quite acceptable. This means that the model will likely approve good credit risks and turn away bad credit risks.

The receiver operating characteristics (ROC) curve is a graphical representation of the model's ability to differentiate between categories. In this graph, sensitivity (recall) refers to true positive rate (TPR) while specificity is linked to false positive rate (FPR). the AUC line represents a random guess. AUC equals 0.79 suggesting a moderately good performance of the model.



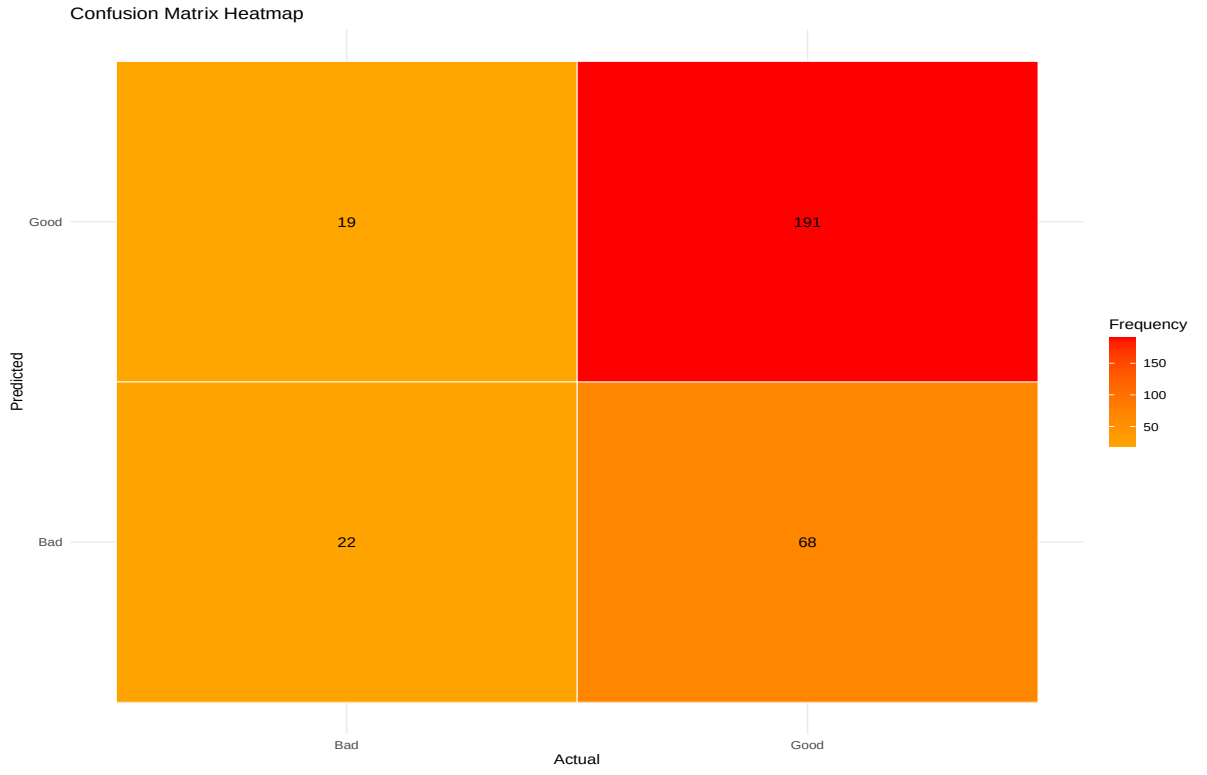
III. K-NEAREST-NEIGHBORS (KNN)

The KNN algorithm is a non-parametric machine learning algorithm mainly used for regression and classification problems. It bases its classification and predictions off the proximity of similar items by finding the nearest neighbour to a particular point and predicting class of value based on the characteristics of the neighbours of that particular point. (LaViale, 2023)

To maximize results, the dataset has been pre-processed and columns/rows with over 50% of missing values have been removed. It was then split into 70 and 30 Percent train and testing sets respectively before the implementation of KNN model. Methods such as cross-validation were used to curate estimations of the model's performance by splitting the trained data set into 10 parts and repeatedly using 9 parts to train the model and the remaining one to test it. Additionally, hyperparameter tuning was paired with the former to pick the "best K".

The original size of the dataset is 1000 samples over 62 predictors however pre-processing has reduced its size to 700 samples with 59 predictors. Accuracy was used to select the optimal model; the best K-value was chosen to promote high-accuracy (k=8). This optimal number suggests a good balance to avoid overfitting (20+) and underfitting (1-5).

The confusion matrix generated from the KNN model shows that it accurately categorizes 71% (0.71) of customer as good or bad credit risks. 191 customers are correctly predicted good credit risks (true positive) and 22 as bad credit risks (true negative). 19 and 68 customers are respectively classified as false positives and false negatives.



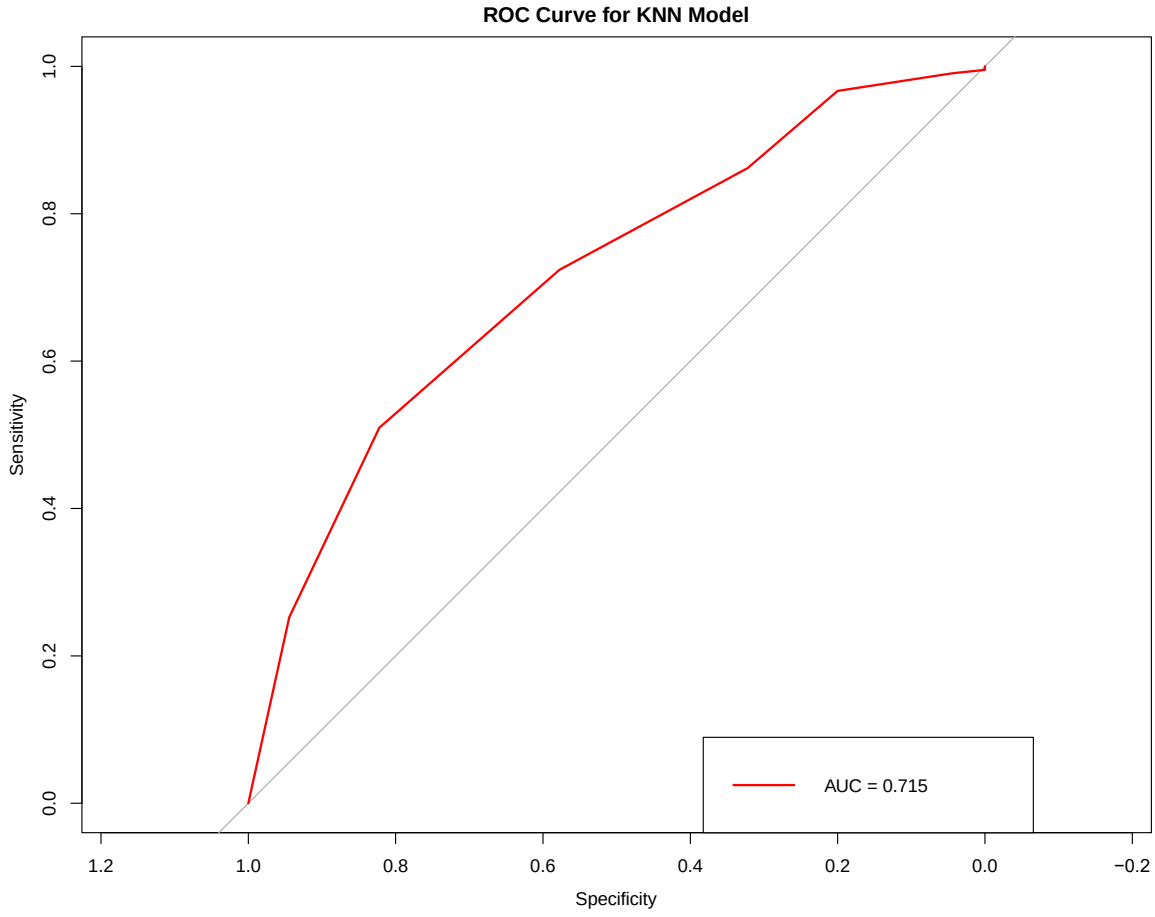
The model's performance metrics indicate a 53.7 % precision (0.537); 24.4% recall (0.244) and 33.6% (0.336) F1 score.

The precision percentage (53.7%) is relatively good, suggesting moderately accurate classification of credit risks.

Recall is exceptionally low (24.4%) implying that the model fails to detect considerable good credit customers.

The F1 score which balances out the precision and recall evaluating model accuracy is equally low (0. 336) reflecting the imbalance between the two former model accuracy evaluation tools.

The model's performance is relatively okay with considerable misclassifications as displayed by the ROC curve and Area Under the Curve (AUC). The graphical representation between sensitivity (recall) and specificity at different classification thresholds allows the visualization of the trade-off between true positive rate and false positive rate. AUC equals 0.71 meaning that the model's performance is mostly good with room for improvements.

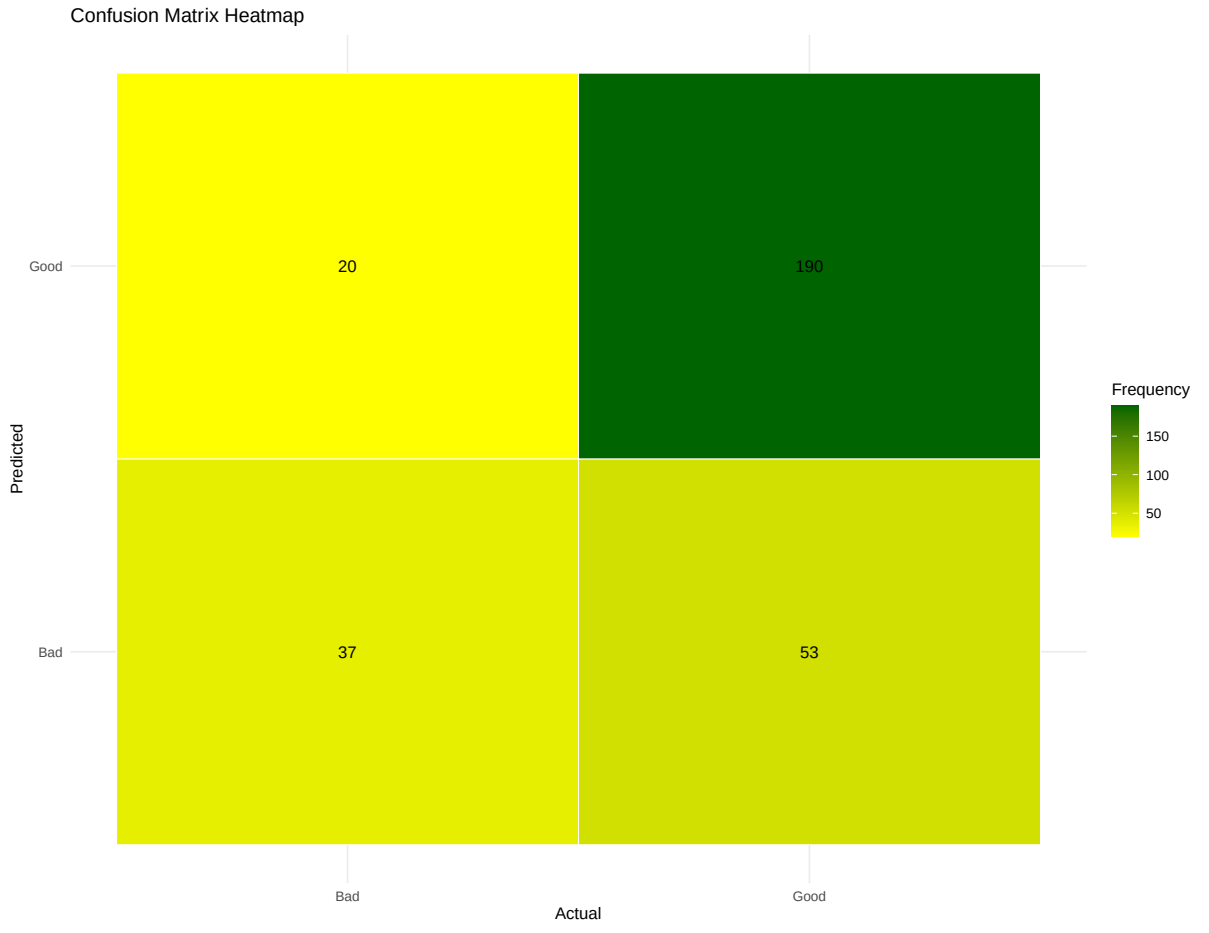


IV. SUPPORT VECTOR MACHINE (SVM)

Super vector machines are mainly used for classification and regression tasks; this model uses a decision boundary meant to separate data points into various classes in a high-dimensional space also known as hyperplane. This model is ideal for high-dimensional data and effective for binary classifications which makes it a good fit for this dataset. (Tybrewal,2023)

Prior to the application of the model, columns with little to no variance were removed from the dataset as they don't change much across observations and can reduce model performance.

The confusion matrix heatmap generated from the SVM model accurately classifies 75% (0.756) of customers. 190 customers are correctly predicted as good credit risks (true positives) and 37 as bad credit risks (true negatives). 20 and 37 customers are respectively misclassified as false positives and false negatives.



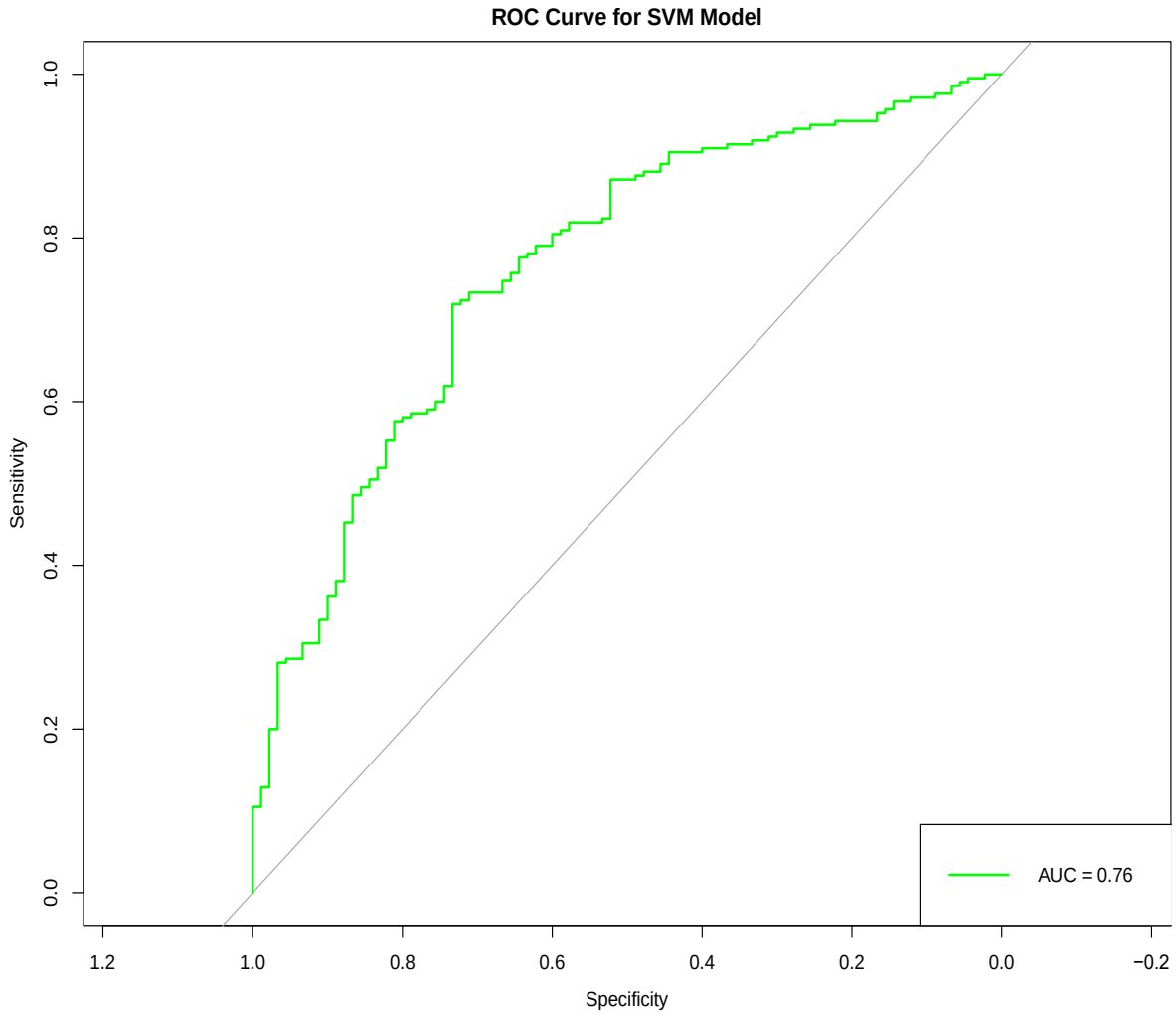
The model's performance metrics showcases a 64% (0.64) precision rate; a 41% recall (0.41) and a 0.503 F1 score.

Precision (64%) is moderately high, implying acceptable model performance with few misclassifications.

Recall (41%) is relatively low, suggesting reoccurring misclassification of bad credit customers as good.

The F1 score (0.503) which balances out the precision and recall evaluating model accuracy reflects moderate balance between precision and recall meaning that the model's performance is poor.

the ROC curve and Area Under the Curve (AUC) is the graphical representation between sensitivity (recall) and specificity at different classification thresholds. AUC is 0.76 meaning that the model's performance is moderately good albeit for some misclassifications.



V. COMPARISON OF THE MODELS

MODEL	ACCURACY	PRECISION	RECALL	F1-SCORE
NAÏVE BAYES	70%	0.686-MODERATE	0.776-HIGH	0.728-HIGH
K-NN (K=8)	71%	0.537-MODERATE	0.244-LOW	0.336-LOW
SVM	75%	0.64-MODERATE	0.41-LOW	0.503-MODERATE

Interpretation of performance metrics is relative to each model's context as they are separate machine learning methods, and the data went through different procedures for each one of them. Consequently, what is considered high, moderate and low for the metrics changes depending on each model which is why 0.50 will be deemed low in one model and moderate in another. (Grūne, 2020)

Overall, the Naïve bayes model gave the best results out of the three models. Even though SVM has the highest accuracy, the former captures more true positives (recall) and presents a more balanced trade-off between precision and recall (f1 score). The KNN's model performance was the poorest in all aspects.

VI. IMPLICATION OF THE MODELS

This section will analyse what it would mean for a business to base their decisions off any of these models.

- **Implications of the Naïve-Bayes model**

  Considering its performance, if a company were to attribute loans based off the naïve bayes model's predictions, it would mostly be accurate in granting loans to "good credit risks" and rejecting "bad credit risks" given its usefulness in handling risk-sensitive applications. However, it is very likely to generate more false positive which can only be advantageous if the company's goal is early detections (e.g., flagging potential fraud). Companies may spend more time conducting reviews which would reduce time efficiency and generate more costs for them as an increase in workload could lead to paid overtime.

- **Implications of the K-Nearest Neighbour model**

  One of this model's greatest flaws is its misleading high accuracy (71%) despite its deplorable performance. Considering that the dataset is imbalanced and "good credit" cases dominate, a lending establishment granting loans based off this model's predictions might incur a high number of loans being approved for high-risk individuals which could lead to an increase in default rates, reputational damage, fines, bankruptcy etc; inconsistency in automated decision making which would be a reflection of the model's high sensitivity to the choice of K could lead to delayed and unreliable credit related decisions being made. The negative implications caused by the use of this model are far too great and could lead to the institution incurring great financial loss and a bad reputation in a best-case scenario or closing and filing for bankruptcy in a worst-case scenario.

- **Implications of the Support Vector Machine model**

  This model had the best accuracy among the three (75%) despite a moderate bordering low performance. However, it managed imbalances within the dataset well enough that if a company were to base its loan decisions off this model's predictions, it wouldn't incur radical consequences. Implications include but are not limited to: lower credit losses per approval as most approved applicants are actually low risk, this would be ideal for a company that prioritizes growth and customer onboarding; increase in auditing challenges given the model's bias against detecting certain risk groups thus classifying considerable bad credit risks as good; requiring more time and resources to train the model along with frequent monitoring and retraining which lacks time and resource efficiency.

VII. RECOMMENDATIONS

Credit risk management is essential in mitigating future credit and trade risks as well as the financial health of the company; it involves the identification, assessment and management of the potential risks of lending money or credit extension for individuals or businesses. (Allianz, 2024)

Strategies to handle credit risk management in the context of incorporating business analytics procedure to analyse dataset would include not relying on one model and to instead combine several to leverage their strength; to employ stacking, boosting and voting ensemble methods for accuracy and overall performance improvement in addition to regular software maintenance and update to keep up with new technological findings and not fall behind competition.

Not relying solely on software for decision making and instead invest resources to use human know how to not only oversee procedures but also review machine made decisions in terms of risk identification, credit risk management and analysis. For money lending establishments, it's also important to improve internal governance practices by ensuring transparency and ethical standards toward customers; educating customers on credit categorization and implication will build trust and customer loyalty and reduce risks of defaulting.

REFERENCES

Allianz, 2024. "Credit risk management: best process and practice. Available at: https://www.allianz-trade.com/en_global/news-insights/business-tips-and-trade-advice/hub/credit-risk-management-best-practices.html Accessed: 13/05/2025

Grūne, 2020. "What's a good F1 Score?". *Getyourguide*. Available at: https://www.getyourguide.careers/posts/what-makes-a-good-f1-score . Accessed: 13/05/2025

kayvasrirelangi, 2024. "Naive Bayes classifies Explained: Assumptions, Types and Uses". *Medium*. Available at: https://medium.com/@kavyasrirelangi100/naive-bayes-classifier-explained-assumptions-types-and-uses-bef767a758a3 Accessed: 06/05/2025

LaViale, 2023. "Deep Dive on KNN: Understanding and Implementing the K-Nearest Neighbours Algorithm". *Arize*. Available at: https://arize.com/blog-course/knn-algorithm-k-nearest-neighbor/ Accessed: 07/05/2025

Tybrewal, 2023. "Support Vector Machines (SVM): An Intuitive Explanation". *Medium*. Available at: https://medium.com/low-code-for-advanced-data-science/support-vector-machines-svm-an-intuitive-explanation-b084d6238106 Accessed: 09/05/2025

R-CODE

```
#DISCLAIMER: THIS FIRST SECTION NEED TO BE RAN BEFORE ANY OTHER
SECTION
# STEP 1: LOAD LIBRARIES
install.packages("e1071")
install.packages("caTools")
install.packages("caret")
install.packages("mlbench")
install.packages("ggplot2")
install.packages("pROC")
library(mlbench)
library(caret)
library(e1071)
library(caTools)
library(ggplot2)
library(pROC)

# STEP 2: LOAD, VIEW, AND PREPARE DATA SET
# Load the dataset
data("GermanCredit")
head(GermanCredit)
str(GermanCredit)
# Convert categorical variables to factors
GermanCredit[sapply(GermanCredit, is.character)] <-
lapply(GermanCredit[sapply(GermanCredit, is.character)], factor)
ggplot(GermanCredit, aes(x = Class)) +
  geom_bar(fill = "steelblue") +
  ggtitle("Credit Risk Distribution") +
  theme_minimal()

# STEP 3: TRAIN AND TEST DATASET
# Split the dataset into training (70%) and test sets (30%)
set.seed(123)  # Set seed for reproducibility
split <- sample.split(GermanCredit$Class, SplitRatio = 0.7)
train_data <- subset(GermanCredit, split == TRUE)
test_data <- subset(GermanCredit, split == FALSE)

install.packages("randomForest")
library(randomForest)
rf_model <- randomForest(Class ~ ., data = train_data, importance = TRUE)
varImpPlot(rf_model, main = "Feature Importance (Random Forest)")


#####################################################################

install.packages("ROSE")
library(ROSE)  # Use ROSE for balancing classes
# Convert categorical variables to factors
```

```
GermanCredit[sapply(GermanCredit, is.character)] <-
lapply(GermanCredit[sapply(GermanCredit, is.character)], factor)

#NB-STEP 1: HANDLE CLASS IMBALANCE USING ROSE
set.seed(123)
# Apply ROSE for balancing dataset
smote_data <- ovun.sample(Class ~ ., data = GermanCredit, method = "both", N =
1000)$data
table(smote_data$Class)

#NB-STEP 2: SPLIT DATA INTO TRAINING AND TEST SETS
set.seed(123)  # For reproducibility
split <- sample.split(smote_data$Class, SplitRatio = 0.7)
train_data <- subset(smote_data, split == TRUE)
test_data <- subset(smote_data, split == FALSE)


#NB-STEP 3: TRAIN NAIVE BAYES MODEL WITH LAPALACE SMOOTHING AND
MAKE PREDICTIONS
# Tune Naive Bayes model with Laplace smoothing (to handle zero-frequency problems)
nb_model <- naiveBayes(Class ~ ., data = train_data, laplace = 1)
print(nb_model)
#MAKE PREDICTIONS
predictions <- predict(nb_model, newdata = test_data)


#NB-STEP 4: EVALUATE PERFORMANCE WITH CONFUSION MATRIX
conf_matrix <- confusionMatrix(predictions, test_data$Class)
print(conf_matrix)
# Precision, Recall, and F1-Score
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
f1_score <- 2 * (precision * recall) / (precision + recall)
cat("Precision: ", round(precision, 3), "\n")
cat("Recall: ", round(recall, 3), "\n")
cat("F1 Score: ", round(f1_score, 3), "\n")


#NB-STEP 5: VISUALIZE CONFUSION MATRIX
cm_table <- as.table(conf_matrix$table)
cm_df <- as.data.frame(cm_table)
colnames(cm_df) <- c("Actual", "Predicted", "Frequency")
ggplot(cm_df, aes(x = Actual, y = Predicted, fill = Frequency)) +
  geom_tile() +
  geom_text(aes(label = Frequency), color = "white") +
  scale_fill_gradient(low = "lightblue", high = "blue") +
  labs(title = "Confusion Matrix Heatmap", x = "Actual", y = "Predicted")


#NB-STEP 6: PLOT ROC CURVE AND CALCULATE AUC
```

```
# Get predicted probabilities for the positive class
probabilities <- predict(nb_model, newdata = test_data, type = "raw")[, 2]
# Create ROC curve
roc_curve <- roc(test_data$Class, probabilities)
# Plot ROC curve
plot(roc_curve, main = "ROC Curve for Naive Bayes Model", col = "blue", lwd = 2)
# Calculate AUC
auc_value <- auc(roc_curve)
legend("bottomright", legend = paste("AUC =", round(auc_value, 2)), col = "blue", lwd = 2)
cat("AUC:", round(auc_value, 3), "\n")


################################################################################
####################################

#DISCLAIMER: TRAIN STEP BY STEP IF MODEL REFUSES TO TRAIN.
# DATA PRE-PROCESSING
# Check missing values per column
col_missing <- colSums(is.na(GermanCredit))
print(col_missing)
# Check missing values per row
row_missing <- rowSums(is.na(GermanCredit))
print(table(row_missing))
# Remove columns with too many missing values (threshold 50%)
threshold <- 0.5 * nrow(GermanCredit)  # 50% missing threshold
GermanCredit <- GermanCredit[, colSums(is.na(GermanCredit)) < threshold]
# Check if any columns are left with missing values after removal
col_missing_after_removal <- colSums(is.na(GermanCredit))
print(col_missing_after_removal)
# Remove rows with missing values
# This will remove any rows where there are still missing values after column removal
GermanCredit <- GermanCredit[complete.cases(GermanCredit), ]
# Check if any missing values still exist
if (any(is.na(GermanCredit))) {
  print("There are still missing values after column and row removal!")
} else {
  print("No missing values in the dataset after removing incomplete rows and columns.")
}
# Ensure target variable is a factor
GermanCredit$Class <- as.factor(GermanCredit$Class)
# Feature Scaling: Standardize all numerical features (excluding 'Class')
numerical_cols <- setdiff(names(GermanCredit), "Class")
GermanCredit[numerical_cols] <- scale(GermanCredit[numerical_cols])


# TRAIN AND TEST PRE-PROCESSED DATA
# Split the dataset into training (70%) and test sets (30%)
set.seed(123)  # Set seed for reproducibility
split <- sample.split(GermanCredit$Class, SplitRatio = 0.7)
train_data <- subset(GermanCredit, split == TRUE)
```

```
test_data <- subset(GermanCredit, split == FALSE)
# Ensure no missing values in train and test datasets
train_data <- na.omit(train_data)
test_data <- na.omit(test_data)
# Check again for missing values in the train/test datasets
if (any(is.na(train_data)) | any(is.na(test_data))) {
  print("There are still missing values in train_data or test_data!")
} else {
  print("No missing values in train_data or test_data.")
}


# KNN-STEP 1: TRAIN KNN MODEL WITH GRID SEARCH FOR HYPERPARAMETER
TUNING
set.seed(123)  # Set seed for reproducibility
# Define control parameters for training (using 10-fold cross-validation)
trControl <- trainControl(method = "cv", number = 10, classProbs = TRUE, search = "grid")
# Define a grid of hyper parameters to tune (specifically 'k')
knn_grid <- expand.grid(k = 3:15)  # Try different values for 'k' from 3 to 15
# Train the KNN model with hyperparameter tuning
knn_model <- train(Class ~ .,
                   data = train_data,
                   method = "knn",
                   tuneGrid = knn_grid,  # Use grid search for 'k'
                   trControl = trControl)
# Print the best model details
print(knn_model)
# KNN-STEP 2: MAKE PREDICTIONS
predictions <- predict(knn_model, newdata = test_data)


# KNN-STEP 3: CONFUSION MATRIX + PERFORMANCE METRICS
conf_matrix <- confusionMatrix(predictions, test_data$Class)
print(conf_matrix)
# Extract precision, recall, and F1-score
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
f1_score <- 2 * (precision * recall) / (precision + recall)
print(paste("Precision: ", round(precision, 3)))
print(paste("Recall: ", round(recall, 3)))
print(paste("F1 Score: ", round(f1_score, 3)))
# Visualization of Confusion Matrix
cm_table <- as.table(conf_matrix$table)
cm_df <- as.data.frame(cm_table)
colnames(cm_df) <- c("Actual", "Predicted", "Frequency")
ggplot(cm_df, aes(x = Actual, y = Predicted, fill = Frequency)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Frequency), color = "black") +
  scale_fill_gradient(low = "orange", high = "red") +
  labs(title = "Confusion Matrix Heatmap", x = "Actual", y = "Predicted") +
```

```
  theme_minimal()


# KNN-STEP 4: ROC CURVE + AUC
probabilities <- predict(knn_model, newdata = test_data, type = "prob")[, 2]
# Create ROC curve
roc_curve <- roc(as.numeric(test_data$Class), probabilities)
# Plot ROC Curve
plot(roc_curve, main = "ROC Curve for KNN Model", col = "red", lwd = 2)
# Compute and display AUC score
auc_value <- auc(roc_curve)
legend("bottomright", legend = paste("AUC =", round(auc_value, 3)), col = "red", lwd = 2)
print(paste("AUC:", round(auc_value, 3)))


#####################################################################

# SVM-STEP 1: REMOVE ZERO-VARIANCE FEATURES
# Identify and remove zero-variance predictors
nzv <- nearZeroVar(train_data)
if (length(nzv) > 0) {
  train_data <- train_data[, -nzv]
  test_data <- test_data[, -nzv]
}


#SVM-STEP 2: TRAIN SVM MODEL
set.seed(123)  # Ensure reproducibility
trControl <- trainControl(method = "cv", number = 10, classProbs = TRUE)
svm_model <- train(Class ~ .,
                   data = train_data,
                   method = "svmRadial",  # Radial Basis Kernel
                   preProcess = c("center", "scale"),  # Standardization
                   tuneLength = 10,
                   trControl = trControl)
#Print model details
print(svm_model)


#SVM-STEP 3: MAKE PREDICTIONS
predictions <- predict(svm_model, newdata = test_data)


#SVM-STEP 4: CONFUSION MATRIX + PERFORMANCE METRICS
conf_matrix <- confusionMatrix(predictions, test_data$Class)
# Print confusion matrix
print(conf_matrix)
# Extract precision, recall, and F1-score
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
```

```
f1_score <- 2 * (precision * recall) / (precision + recall)
print(paste("Precision: ", round(precision, 3)))
print(paste("Recall: ", round(recall, 3)))
print(paste("F1 Score: ", round(f1_score, 3)))
#CONFUSION MATRIX VISUALIZATION
cm_table <- as.table(conf_matrix$table)
cm_df <- as.data.frame(cm_table)
colnames(cm_df) <- c("Actual", "Predicted", "Frequency")
ggplot(cm_df, aes(x = Actual, y = Predicted, fill = Frequency)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Frequency), color = "black") +
  scale_fill_gradient(low = "yellow", high = "darkgreen") +
  labs(title = "Confusion Matrix Heatmap", x = "Actual", y = "Predicted") +
  theme_minimal()


#SVM-STEP 5: ROC CURVE + AUC
# Get predicted probabilities for the positive class
probabilities <- predict(svm_model, newdata = test_data, type = "prob")[, 2]
# Create ROC curve
roc_curve <- roc(as.numeric(test_data$Class), probabilities)
# Plot ROC Curve
plot(roc_curve, main = "ROC Curve for SVM Model", col = "green", lwd = 2)
# Compute and display AUC score
auc_value <- auc(roc_curve)
legend("bottomright", legend = paste("AUC =", round(auc_value, 2)), col = "green", lwd = 2)
print(paste("AUC:", round(auc_value, 3)))
```