**MICRO PROJECT**

**DATA ANALYTICS LAB (CDL 331)**

**MODELLING BANK CUSTOMER BEHAVIUOR USING FEATUR ENGINEERING AND CLASSIFICATION**

**TECHNIQUES**

##### MICRO PROJECT REPORT

***Submitted by***

RAHUL KRISHNA KR(MAC22CD050)

VIGNESH S(MAC22CD062)

NIBRAS UL HAQUE O N(MAC22CD047)

***To***

***The APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY IN**

**COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(DATA SCIENCE)**

**MAR ATHANASIUS COLLEGE OF ENGINEERING KOTHAMANGALAM , KERALA-686 666**

**OCTOBER 2024**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(DATA SCIENCE)**

**MAR ATHANASIUS COLLEGE OF ENGINEERING KOTHAMANGALAM , KERALA-686 666**



**CERTIFICATE**

This is to certify that the report entitled **“MODELLING BANK CUSTOMER**

**BEHAVIUOR USING FEATUR ENGINEERING AND CLASSIFICATION**

**TECHNIQUES”** submitted by **Mr.** **RAHUL KRISHNA K R(MAC22CD050),**

**Mr.** **VIGNESH S(MAC22CD062) & Mr.** **NIBRAS UL HAQUE O N (MAC22CD047)** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science & Engineering for the academic year 2023-2024 is a bonafide record of the micro project presented by them under our supervision and guidance. This report in any form has not been submitted to any other university or institute for any purpose.

##### ……………………… …………………….

**Prof**. **Richu Shibu Prof. Joby Staff Staff in charge Head of the Department**

## **ACKNOWLEDGEMENT**

First and foremost, we sincerely thank God Almighty for his grace for the successful and timely completion of the micro project. We express our sincere gratitude and thanks to our Principal Dr. Bos Mathew Jos and Head of the Department Prof. Joby George for providing the necessary facilities and their encouragement and support. We owe special thanks to the faculty in charge Prof. Richu Shibu & Prof. Shabiaya M I for their corrections, suggestions and sincere efforts to coordinate the micro project under a tight schedule. We express our sincere thanks to staff members in the Department of Computer Science and Engineering who have taken sincere efforts in helping me to conduct this seminar. Finally, we would like to acknowledge the heartfelt efforts, comments, criticisms, co-operation and tremendous support given to me by our dear friends during the preparation of the seminar and also during the presentation without whose support this work would have been all the more difficult to accomplish.

**ABSTRACT**

The provided code implements an interactive platform using Streamlit to train and evaluate several machine learning classification models. The platform supports popular classification algorithms, including Random Forest, Gradient Boosting, Support Vector Machine (SVM), k-Nearest Neighbors (k-NN), Naive Bayes, Decision Tree, Logistic Regression, and Multilayer Perceptron (MLP). These models are instantiated with default or commonly-used hyperparameters, such as `n\_estimators` for Random Forest, or `hidden\_layer\_sizes` for MLP. Once the user uploads or selects a dataset, the models are trained on the training set and tested on the testing set. The platform calculates performance metrics like accuracy, Matthews Correlation Coefficient (MCC), and Receiver Operating Characteristic Area Under the Curve (ROC-AUC) to assess the predictive power of each model.

For each model, several evaluation tools are presented to help users understand model performance more deeply. The accuracy score measures the overall performance, while MCC provides a more nuanced view of prediction quality, especially in imbalanced datasets. The classification report includes metrics like precision, recall, and F1 score, which are essential for understanding class-wise performance. Additionally, a confusion matrix is generated, offering a visual breakdown of true positives, false positives, true negatives, and false negatives. The platform also plots the ROC curve for each model, which displays the trade-off between true positive and false positive rates across different threshold settings, providing insight into the model’s sensitivity and specificity.

The platform’s main functionality is to identify the best-performing model by comparing the accuracy of all models evaluated. In case of a tie, MCC is used as a tiebreaker to ensure the most reliable model is selected. The results are presented in an easy-to-read, side-by-side layout, allowing users to quickly grasp the strengths and weaknesses of each model. This tool makes it simple for users to experiment with different classifiers and evaluate their performance without needing to write additional code. By streamlining model comparison and visualization, the platform becomes a valuable asset for data scientists and machine learning practitioners, assisting them in selecting the most suitable model for their data.

**Contents**

##### iLIST OF FIGURES

1. INTRODUCTION 1
2. SYSTEM DESIGN 2

2.1Machine Learning Concepts and Tools…………………………...2

2.1.1 Data Preprocessing………………………………………2

2.1.2 Encoding Categorical Variables……………...…….2

2.1.3 Feature Selection…………………………………..2

2.1.4 Train-Test Split…………………………………...2

2.1.5 Model Training…………………………………...3

2.1.6 Model Evaluation Metrics……………….………..3

2.1.7 Visualization……………………………...……....3

2.2 Dataset

1. **PROGRAM AND RESULTS 4**
   1. PROGRAM 6
   2. RESULT 12

##### CONCLUSION 20

##### BIBLIOGRAPHY 21

**List Of Figures**

3.1.1 Confusion Matrix for Random Forest and ROC Curve……….14

3.1.2 Confusion Matrix and ROC Curve for Random Forest

and Gradient Boosting……………………………………………...15

3.1.3 Confusion Matrix and ROC Curve for SVM and K-NN……...16

3.1.4 Confusion Matrix and ROC Curve for Naive Bayes and Decision Tree………………………………………………………..17

3.1.5 Confusion Matrix and ROC Curve for Logistic

Regression and MLP………………………………………………...18

**CHAPTER 1**

**INTRODUCTION**

The code presented aims to provide a streamlined platform for comparing multiple machine learning classification algorithms. Built using Streamlit, the platform enables users to upload datasets and evaluate various models without having to manually code individual algorithms. The platform incorporates a selection of widely-used classification algorithms, including Random Forest, Gradient Boosting, Support Vector Machine (SVM), k-Nearest Neighbors (k-NN), Naive Bayes, Decision Tree, Logistic Regression, and Multilayer Perceptron (MLP). These models are pre-configured with common hyperparameters, allowing users to focus on analyzing performance metrics rather than fine-tuning model parameters.

A significant aspect of the platform is the comprehensive evaluation framework it provides. For each model, key performance metrics such as accuracy and the Matthews Correlation Coefficient (MCC) are calculated to assess the overall prediction quality. Additionally, the platform generates detailed classification reports, confusion matrices, and Receiver Operating Characteristic (ROC) curves. These tools allow users to understand how well each model performs, both in terms of overall accuracy and in terms of its ability to distinguish between different classes. By offering a visual comparison of the models, users can easily identify the strengths and weaknesses of each algorithm on their specific dataset.

The core functionality of the platform is designed to determine the best-performing model based on these metrics. After training and evaluating all models, the platform identifies the model with the highest accuracy. In case of a tie, the Matthews Correlation Coefficient (MCC) is used as a tiebreaker to select the most reliable classifier. This tool simplifies the process of model selection, allowing users to experiment with different algorithms quickly and efficiently. Overall, the platform serves as a powerful resource for anyone working with classification problems, providing an easy-to-use interface for comparing and selecting the best machine learning models.

**CHAPTER 2**

**SYSTEM DESIGN**

The system design for this machine learning project involves several components, each playing a critical role in data ingestion, processing, model training, evaluation, and visualization. The design follows a modular approach to ensure that each step of the process is isolated and can be enhanced or updated independently.

**2.1 Machine Learning Concepts And Tools**

2.1.1. Data Pre-processing

- Handling Missing Values: The code checks for missing values in the dataset.

- Normalization (Min-Max Scaling): Numerical features are normalized to a range of [0, 1] using MinMaxScaler to standardize the input data and improve the performance of machine learning algorithms.

2.1.1. Encoding Categorical Variables

- One-Hot Encoding: Categorical variables are transformed into numerical format using pd.get\_dummies, which creates dummy variables for each category. This is useful for handling non-numeric data in machine learning models.

2.1.3. Feature Selection

- Chi-Square Test (SelectKBest): Feature selection is done using the chi-squared test with the SelectKBest method to select the 10 most important features based on their statistical relationship with the target variable. This helps reduce dimensionality and improve model performance.

2.1.4. Train-Test Split:

- Splitting the Dataset: The dataset is split into training and testing sets using train\_test\_split, typically to evaluate how the model generalizes to unseen data. The test size is 30% of the dataset.

2.1.5. Model Training:

- Random Forest: A tree-based ensemble learning method that builds multiple decision trees and averages their predictions to improve accuracy and control overfitting.

- Gradient Boosting: Another ensemble technique that builds decision trees sequentially, where each tree corrects the errors of the previous ones.

- Support Vector Machine (SVM): A classification algorithm that finds the optimal hyperplane to separate different classes.

- k-Nearest Neighbors (k-NN): A simple instance-based learning algorithm where the label is predicted based on the labels of the nearest neighbors.

2.1.6. Model Evaluation Metrics:

- Accuracy Score: Measures the percentage of correct predictions.

- Confusion Matrix: Provides a summary of prediction results showing true positives, true negatives, false positives, and false negatives.

- Classification Report: Includes precision, recall, and F1-score to evaluate the performance of a classifier.

- Matthews Correlation Coefficient (MCC): A balanced measure of classification quality, especially when dealing with imbalanced datasets.

- ROC Curve (Receiver Operating Characteristic): A plot of true positive rate (sensitivity) vs. false positive rate at various threshold levels.

- ROC-AUC (Area Under the Curve): A performance metric that summarizes the ability of the model to distinguish between classes.

2.1.7. Visualization:

- Confusion Matrix Heatmap: Visualized using seaborn to make it easier to interpret the results of model predictions.

- ROC Curve: A comparison of the performance of all models using their ROC curves, where the area under the curve (AUC) is computed.

These techniques cover data preparation, feature selection, model building, model evaluation, and result vis**ualization.**

**2.3 Dataset**

The CSV file contains customer data from a bank's marketing campaign, structured with the following columns: **age** (customer's age), **job** (occupation type), **marital** (marital status), **education** (level of education), **default** (whether the customer has credit in default), **balance** (account balance in euros), **housing** (whether the customer has a housing loan), and **loan** (whether the customer has a personal loan). Additional columns include **contact** (contact communication type), **day** (last contact day of the month), **month** (last contact month of the year), **duration** (duration of the last contact in seconds), **campaign** (number of contacts during the campaign), **pdays** (number of days since the client was last contacted), **previous** (number of contacts before the campaign), and **poutcome** (outcome of the previous marketing campaign). The target column **y** indicates whether the customer subscribed to the bank's term deposit, with values like 'yes' or 'no'. This file structure is essential for the feature selection and machine learning classification tasks performed in the code.

**CHAPTER 3**

**PROGRAM AND RESULTS**

**3.1 PROGRAM**

import pandas as pd

import numpy as np

import streamlit as st

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

from sklearn.feature\_selection import SelectKBest, chi2

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import (

accuracy\_score, confusion\_matrix, classification\_report,

roc\_auc\_score, roc\_curve, matthews\_corrcoef

)

import matplotlib.pyplot as plt

import seaborn as sns

# Custom CSS to set background image

page\_bg\_img = """

<style>

[data-testid="stAppViewContainer"] {

background-image: url("https://i.postimg.cc/4d0sht5Y/black-elegant-background-with-copy-space.jpg");

background-size: cover;

background-position: center;

background-repeat: no-repeat;

}

[data-testid="stHeader"] {

background: rgba(0, 0, 0, 0); /\* Make header transparent \*/

}

</style>

"""

st.markdown(page\_bg\_img, unsafe\_allow\_html=True)

# App Title with Custom Styling

st.markdown('<h1 style="font-family: serif; color: white; text-align: center;">✨ Bank Marketing Model Interface </h1>', unsafe\_allow\_html=True)

# File Upload

uploaded\_file = st.file\_uploader("📁 Upload your CSV file", type="csv")

if uploaded\_file is not None:

data = pd.read\_csv(uploaded\_file)

# Display DataFrame

st.write("🗂 \*\*Dataset Preview\*\*:")

st.dataframe(data.head())

# Data Pre-processing

st.subheader("🛠 Data Pre-processing")

# Check for missing values

missing\_values = data.isnull().sum()

st.write("❌ \*\*Missing Values\*\*:", missing\_values)

# Apply Min-Max Normalization

numerical\_features = ['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous']

scaler = MinMaxScaler()

data[numerical\_features] = scaler.fit\_transform(data[numerical\_features])

# One-Hot Encoding

data\_encoded = pd.get\_dummies(data, drop\_first=True)

# Feature Selection

X = data\_encoded.drop("y\_yes", axis=1)

y = data\_encoded["y\_yes"]

selector = SelectKBest(score\_func=chi2, k=10)

X\_selected = selector.fit\_transform(X, y)

selected\_features = X.columns[selector.get\_support()]

st.write("📊 \*\*Selected Features:\*\*", selected\_features)

# Train-Test Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.3, random\_state=42)

# List of models

models = {

"Random Forest": RandomForestClassifier(n\_estimators=100, random\_state=42),

"Gradient Boosting": GradientBoostingClassifier(random\_state=42),

"SVM": SVC(probability=True, random\_state=42),

"k-NN": KNeighborsClassifier(n\_neighbors=5),

"Naive Bayes": GaussianNB(),

"Decision Tree": DecisionTreeClassifier(random\_state=42),

"Logistic Regression": LogisticRegression(max\_iter=1000, random\_state=42),

"MLP": MLPClassifier(hidden\_layer\_sizes=(100,), max\_iter=1000, random\_state=42)

}

# Dropdown to select individual model

selected\_model\_name = st.selectbox("🔍 Choose a model to display results individually", list(models.keys()))

# Show results for the selected model

if selected\_model\_name:

model = models[selected\_model\_name]

model.fit(X\_train, y\_train)

# Predictions

y\_pred = model.predict(X\_test)

# Metrics

accuracy = accuracy\_score(y\_test, y\_pred)

mcc = matthews\_corrcoef(y\_test, y\_pred)

st.write(f"### {selected\_model\_name} Results")

st.write(f"✅ \*\*Accuracy:\*\* {accuracy \* 100:.2f}%")

st.write(f"📏 \*\*MCC:\*\* {mcc:.2f}")

# Classification Report

st.text("📝 \*\*Classification Report:\*\*")

st.text(classification\_report(y\_test, y\_pred))

# Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

fig, ax = plt.subplots()

sns.heatmap(cm, annot=True, fmt='d', cmap='Purples', cbar=False, ax=ax)

ax.set\_xlabel('Predicted Label')

ax.set\_ylabel('True Label')

ax.set\_title(f'Confusion Matrix for {selected\_model\_name}')

st.pyplot(fig)

# ROC Curve

y\_probs = model.predict\_proba(X\_test)[:, 1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_probs)

fig, ax = plt.subplots()

ax.plot(fpr, tpr, label=f"{selected\_model\_name} (AUC = {roc\_auc\_score(y\_test, y\_probs):.2f})")

ax.plot([0, 1], [0, 1], 'k--') # Random Guess Line

ax.set\_xlabel('False Positive Rate')

ax.set\_ylabel('True Positive Rate')

ax.set\_title('ROC Curve')

ax.legend(loc='best')

st.pyplot(fig)

# Train and Evaluate All Models

if st.button("🚀 Train and Evaluate All Models"):

# Create layout for displaying models' results

cols = st.columns(2)

results = {}

for i, (name, model) in enumerate(models.items()):

# Train the model

model.fit(X\_train, y\_train)

# Predictions

y\_pred = model.predict(X\_test)

# Metrics

accuracy = accuracy\_score(y\_test, y\_pred)

mcc = matthews\_corrcoef(y\_test, y\_pred)

results[name] = (accuracy, mcc)

with cols[i % 2]: # Display in columns

st.markdown(f"### {name}")

st.write(f"✅ \*\*Accuracy:\*\* {accuracy \* 100:.2f}%")

st.write(f"📏 \*\*MCC:\*\* {mcc:.2f}")

# Classification Report

st.text("📝 \*\*Classification Report:\*\*")

st.text(classification\_report(y\_test, y\_pred))

# Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

fig, ax = plt.subplots()

sns.heatmap(cm, annot=True, fmt='d', cmap='Purples', cbar=False, ax=ax)

ax.set\_xlabel('Predicted Label')

ax.set\_ylabel('True Label')

ax.set\_title(f'Confusion Matrix for {name}')

st.pyplot(fig)

# ROC Curve

y\_probs = model.predict\_proba(X\_test)[:, 1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_probs)

fig, ax = plt.subplots()

ax.plot(fpr, tpr, label=f"{name} (AUC = {roc\_auc\_score(y\_test, y\_probs):.2f})")

ax.plot([0, 1], [0, 1], 'k--') # Random Guess Line

ax.set\_xlabel('False Positive Rate')

ax.set\_ylabel('True Positive Rate')

ax.set\_title('ROC Curve')

ax.legend(loc='best')

st.pyplot(fig)

# Determine the best model based on accuracy, and use MCC as a tiebreaker

best\_model\_name = None

best\_accuracy = 0

best\_mcc = -1 # Lower bound for MCC

for model\_name, (accuracy, mcc) in results.items():

# If accuracy is higher, update the best model

if accuracy > best\_accuracy:

best\_accuracy = accuracy

best\_mcc = mcc

best\_model\_name = model\_name

# If accuracy is the same but MCC is higher, update the best model

elif accuracy == best\_accuracy and mcc > best\_mcc:

best\_mcc = mcc

best\_model\_name = model\_name

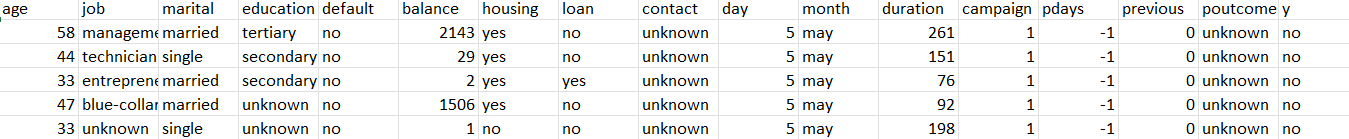
# Display best model message

st.markdown(f"## 🏆 The best model is \*\*{best\_model\_name}\*\* with an accuracy of \*\*{best\_accuracy \* 100:.2f}%\*\*!")

st.write(f"### Reason: {best\_model\_name} achieved the highest accuracy or, in case of a tie, th**e highest MCC.")**

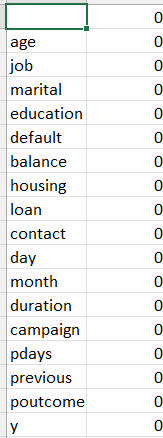
**3.2 RESULT**

**Data Preview**

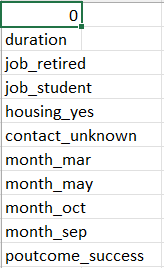


**Data Pre-processing**

**Missing Values:**



**Selected Features:**



**Random Forest Results**

Accuracy: 88.66%

MCC: 0.39

\*\* Classification Report: \*\*

Precision recall fl-score support

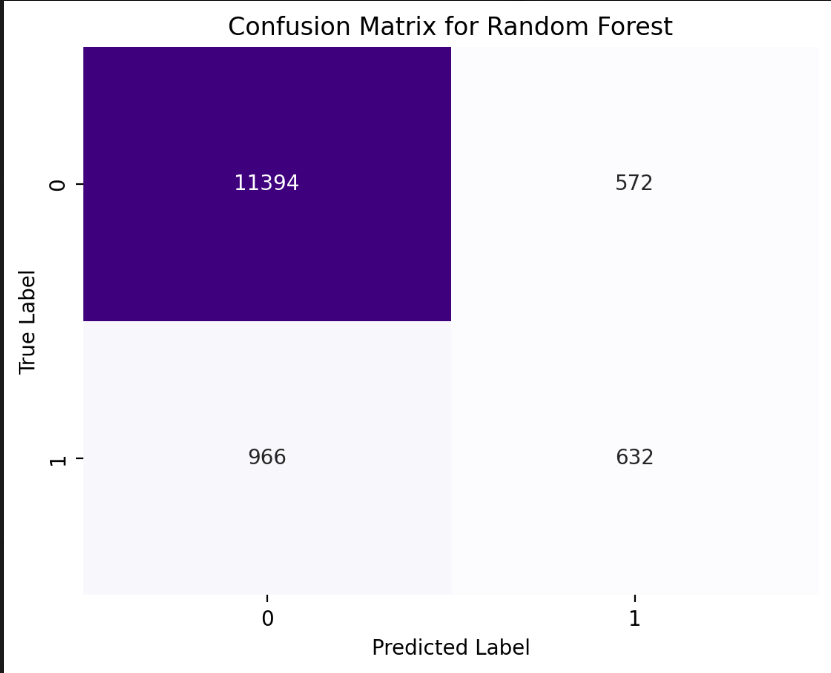
False 0.92 0.95 0.94 13564

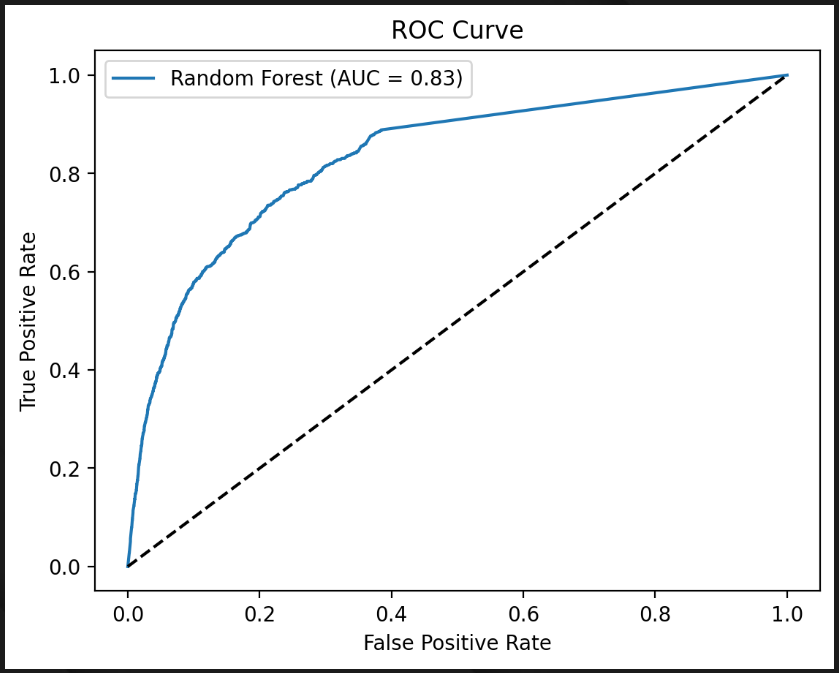
True 0.52 0.40 0.45 1598

Accuracy 0.89 13564

Macro avg 0.72 0.67 0.69 13564

Weighted avg 0.88 0.89 0.88 13564

****

****Random Forest Gradient Boosting

Accuracy: 88.66% Accuracy: 90.03%

MCC: 0.39 MCC: 0.44

\*\*Classification Report: \*\* \*\*Classification Report: \*\*

Precision recall fl-score support precision recall fl-score support

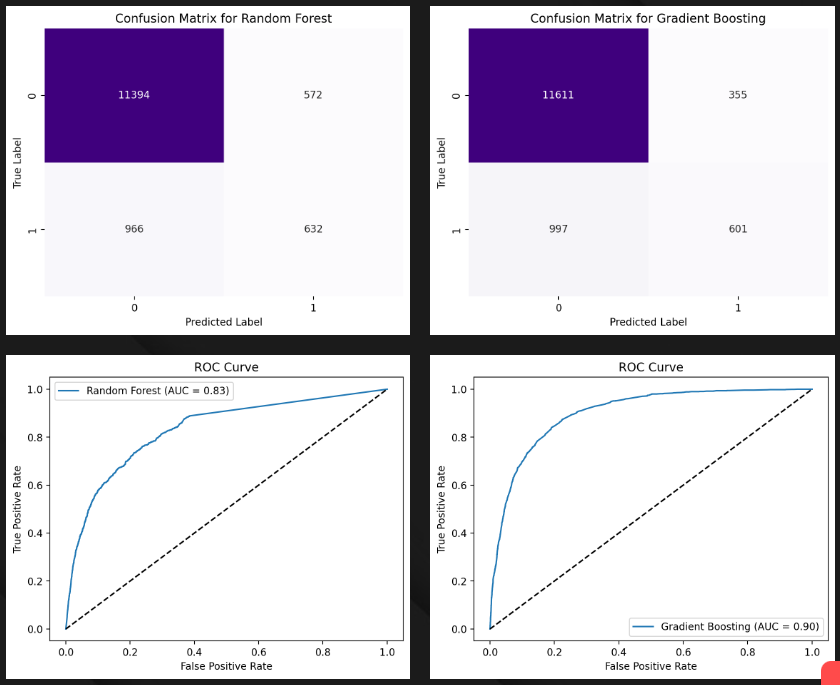
False 0.92 0.95 0.94 11966 False 0.92 0.97 0.94 11966

True 0.52 0.40 0.45 1598 True 0.63 0.38 0.47 1598

Accuracy 0.89 13564 accuracy 0.90 13564

Macro avg 0.72 0.67 0.69 13564 macro avg 0.77 0.67 0.71 13564

Weighted avg 0.88 0.89 0.88 13564 weighted avg 0.89 0.90 0.89 13564



**SNM K-NN**

Accuracy: 89.26% Accuracy: 89.29%

MCC: 0.31 MCC: 0.42

\*\*Classification Report: \*\* \*\*Classification Report: \*\*

Precision recall fl-score support precision recall fl-score support

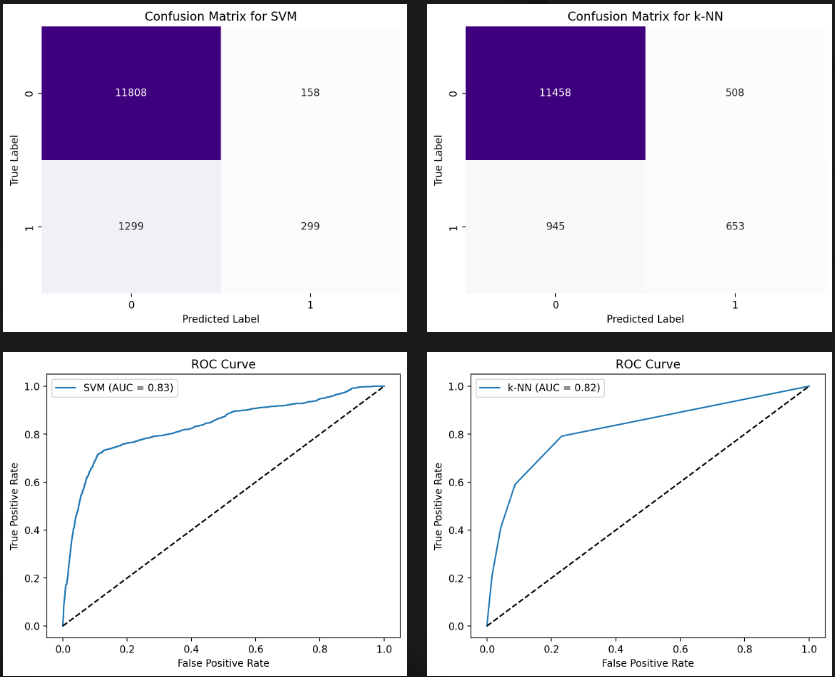
False 0.90 0.99 0.94 11966 False 0.92 0.96 0.94 11966

True 0.65 0.19 0.29 1598 True 0.56 0.41 0.47 1598

Accuracy 0.88 600 accuracy 0.89 13564

Macro avg 0.75 0.56 0.58 600 macro avg 0.74 0.68 0.71 13564

Weighted avg 0.88 0.91 0.88 600 weighted avg 0.88 0.89 0.89 13564



**Naïve Bayes Decision Tree**

Accuracy: 87.98% Accuracy: 88.79%

MCC: 0.39 MCC: 0.38

\*\*Classification Report: \*\* \*\*Classification Report: \*\*

Precision recall fl-score support precision recall fl-score support

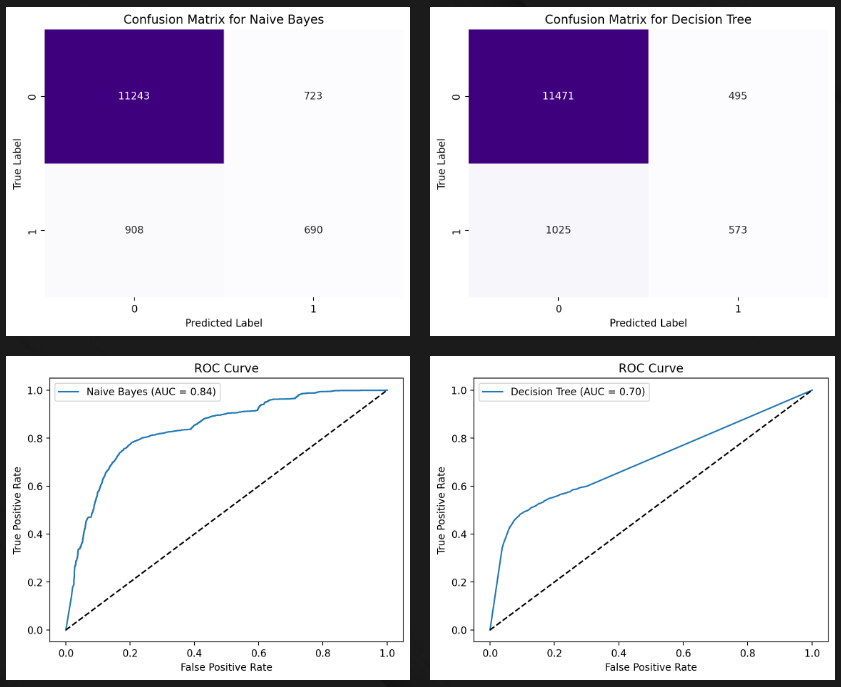
False 0.93 0.94 0.93 11966 False 0.92 0.96 0.94 11966

True 0.49 0.43 0.46 1598 True 0.54 0.36 0.43 1598

Accuracy 0.88 13564 accuracy 0.89 13564

Macro avg 0.71 0.69 0.70 13564 macro avg 0.73 0.66 068 13564

Weighted avg 0.87 0.88 0.88 13564 weighted avg 0.87 0.89 0.88 13564

****

**Logistic Regression MLP**

Accuracy: 89.90% Accuracy: 90.83%

MCC: 0.40 MCC: 0.39

\*\*Classification Report: \*\* \*\*Classification Report: \*\*

Precision recall fl-score support precision recall fl-score support

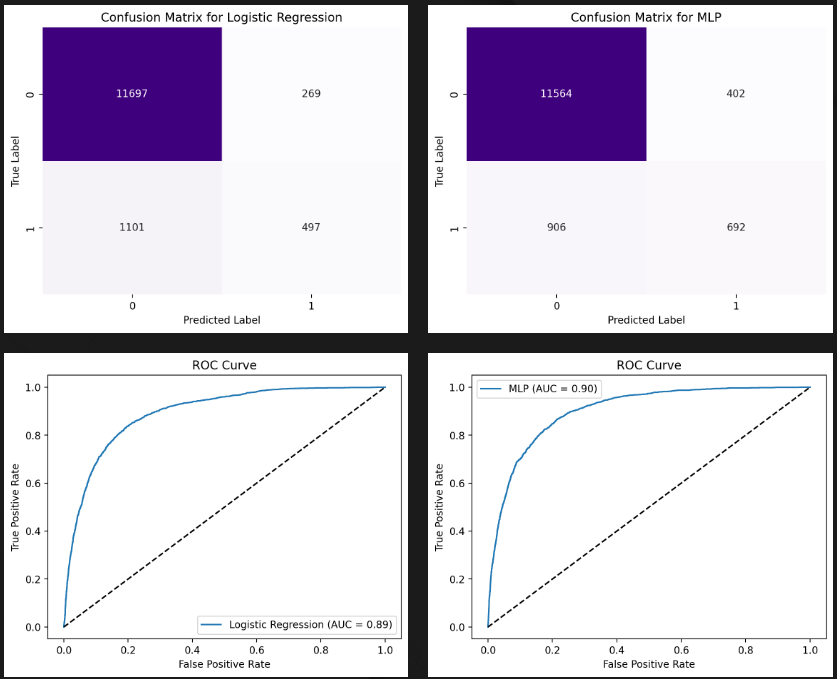
False 0.91 0.98 0.94 11966 False 0.93 0.97 0.95 11966

True 0.65 0.31 0.42 1598 True 0.63 0.43 0.51 1598

Accuracy 0.90 13564 accuracy 0.90 13564

Macro avg 0.78 0.64 0.68 13564 macro avg 0.78 0.70 0.73 13564

Weighted avg 0.88 0.90 0.88 13564 weighted avg 0.890 0.91 0.90 13564



**The best model is MLP with an accuracy of 90.83!**

**Reason: MLP achieved the highest accuracy or, in case of a tie, the highest MCC.**

**CHAPTER 4**

**CONCLUSION**

In this bank marketing model interface project, eight machine learning algorithms—Random Forest, Gradient Boosting, Support Vector Machine (SVM), k-Nearest Neighbors (k-NN), Naive Bayes, Decision Tree, Logistic Regression, and Multi-Layer Perceptron (MLP)—were evaluated to predict customer behavior. After performing feature selection and splitting the data into training and testing sets, each model was trained and tested using key metrics such as accuracy and the Matthews correlation coefficient (MCC). A range of visualization techniques, including confusion matrices and ROC curves, provided further insights into the classification performance of each model. The results indicated that the MLP model demonstrated the highest predictive capability, achieving an accuracy of 90.83%, making it the top-performing algorithm among the eight.

The MLP classifier's success can be attributed to its ability to model complex patterns in the data through its neural network structure, which outperformed other traditional algorithms such as Random Forest and Gradient Boosting. Although these ensemble models are typically strong classifiers, they did not match the MLP in this case, suggesting that the data set had nonlinear characteristics that the MLP was better equipped to handle. The use of MCC as a tiebreaker metric also reinforced the robustness of the MLP’s classification, ensuring that it balanced both precision and recall effectively. This project highlights the importance of experimenting with multiple algorithms to identify the most suitable model for a given problem, with the MLP ultimately proving to be the best fit for this bank marketing data.

**BIBLIOGRAPHY**

[https://sci2s.ugr.es/most-influential-preprocessing - :~:text=Data%20preprocessing%20includes%20data%20preparation,feature%20selection%2C%20instance%20selection%20or](https://sci2s.ugr.es/most-influential-preprocessing#:~:text=Data%20preprocessing%20includes%20data%20preparation,feature%20selection%2C%20instance%20selection%20or)

Data preprocessing: Definition, Steps and Concepts.

[https://www.tableau.com/learn/articles/what-is-data-cleaning - :~:text=tools%20and%20software-,What%20is%20data%20cleaning%3F,to%20be%20duplicated%20or%20mislabeled.](https://www.tableau.com/learn/articles/what-is-data-cleaning#:~:text=tools%20and%20software-,What%20is%20data%20cleaning%3F,to%20be%20duplicated%20or%20mislabeled.)

Data Cleaning: Definition, Benefits and How To Tableau

<https://www.javatpoint.com/machine-learning-random-forest-algorithm>

Random Forest Algorithm: Definition, steps, Advantages, Applications

<https://www.snowflake.com/guides/what-gradient-boosting/#:~:text=Gradient%20boosting%20algorithms%20work%20iteratively,predictions%20of%20all%20the%20models>

Gradient Boosting Algorithm: Definition, Concepts

https://www.techtarget.com/whatis/definition/support-vector-machine-SVM#:~:text=A%20support%20vector%20machine%20(SVM)%20is%20a%20type%20of%20supervised,data%20set%20into%20two%20groups.

Support Vector Machine (SVM) Algorithm

https://auth.geeksforgeeks.org/roadBlock\_v2.php

k-Nearest Neighbors (k-NN)