Machine Learning Pipeline for the UCI Bank Marketing Dataset

Introduction:

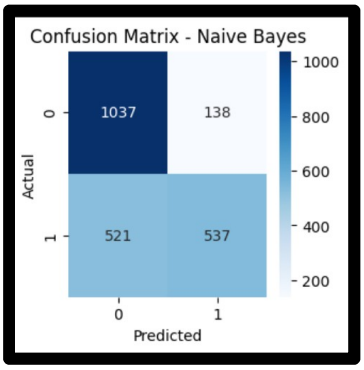
The objective of this project is to design a machine learning pipeline capable of handling a noisy real-world dataset. The dataset used is the UCI Bank Marketing dataset, which contains information about clients of a Portuguese bank and whether they subscribed to a term deposit. The dataset includes numerical and categorical features, missing values, outliers, and potentially irrelevant attributes.

This report focuses on:

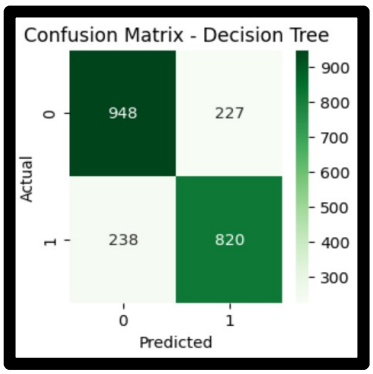
- **Data Preprocessing:** Handling missing values, outliers, encoding categorical variables, and feature scaling.

- **Model Development:** Training and evaluating Decision Tree, Naive Bayes, and Support Vector Machine models.

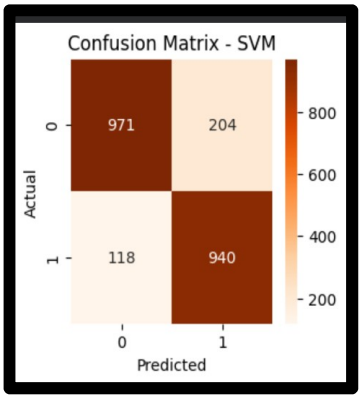
```
Naive Bayes Results
Accuracy : 0.7048813255709807
Precision: 0.7955555555555556
Recall   : 0.5075614366729678

Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.88      0.76      1175
           1       0.80      0.51      0.62      1058

    accuracy                           0.70      2233
   macro avg       0.73      0.70      0.69      2233
weighted avg       0.73      0.70      0.69      2233
```



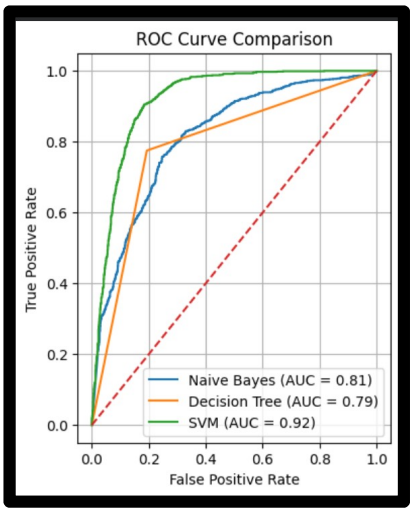

```
Decision Tree Results
Accuracy : 0.7917599641737573
Precision: 0.7831900668576887
Recall   : 0.775047258979206

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.81      0.80      1175
           1       0.78      0.78      0.78      1058

    accuracy                           0.79      2233
   macro avg       0.79      0.79      0.79      2233
weighted avg       0.79      0.79      0.79      2233
```



```
SVM Results
Accuracy : 0.8557993730407524
Precision: 0.8216783216783217
Recall   : 0.888468809073724

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.83      0.86      1175
           1       0.82      0.89      0.85      1058

    accuracy                           0.86      2233
   macro avg       0.86      0.86      0.86      2233
weighted avg       0.86      0.86      0.86      2233
```



SVM performed the best because it learnt the complex patterns and predicted most cases correctly.



SVM performed the best because its ROC curve is highest and its AUC (0.92) is the biggest. Naive Bayes performed moderately well (AUC 0.81), and Decision Tree is the weakest (AUC 0.79). So overall, SVM gives the best predictions.

To improve performance:

- **Bagging** was applied to Naive Bayes, reducing variance and making it more robust to noisy data.

```
Bagging Naive Bayes Results
Accuracy : 0.7057769816390506
Precision: 0.7961595273264401
Recall   : 0.5094517958412098

Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.88      0.76      1175
           1       0.80      0.51      0.62      1058

    accuracy                           0.71      2233
   macro avg       0.73      0.70      0.69      2233
weighted avg       0.73      0.71      0.69      2233
```

bagging provided only a tiny improvement confirming that Naive Bayes is a stable, low-variance model and does not benefit much from bagging.

- **AdaBoost** was applied to Decision Tree, focusing on misclassified samples and improving overall predictive accuracy.

```
AdaBoost + Decision Tree Results
Accuracy : 0.812807881773399
Precision: 0.803030303030303
Recall   : 0.8015122873345936

Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.82      0.82      1175
           1       0.80      0.80      0.80      1058

    accuracy                           0.81      2233
   macro avg       0.81      0.81      0.81      2233
weighted avg       0.81      0.81      0.81      2233
```

Applying AdaBoost to the Decision Tree improved the model's performance: accuracy increased from 79.2% to 81.3%, while both precision and recall for class 1 rose by about 2–3%. Overall, boosting made the classifier more accurate and balanced in identifying both classes.

Justification:

- Bagging averages multiple models, which stabilizes predictions for noisy datasets.

- Boosting emphasizes harder-to-classify samples, which can improve accuracy but may overfit if the dataset is very noisy.