**Machine learning python practical assignment**

**Were Vincent O**

**Introduction**

This Machine Learning project focuses on applying and evaluating classification algorithms using the Diabetes dataset, which comprises medical records to predict the likelihood of diabetes in individuals. The dataset includes features such as glucose levels, blood pressure, BMI, and age, and the target variable indicates whether a person is diabetic (1) or not (0).

The task involves implementing three machine learning models: Logistic Regression, Linear Support Vector Machine (SVM), and Radial Basis Function (RBF) SVM. The dataset is preprocessed by scaling features and splitting it into training and testing subsets. Models are evaluated using key metrics such as accuracy, AUC (Area Under the ROC Curve), and F1-score. Additionally, k-fold cross-validation is applied to validate model robustness.

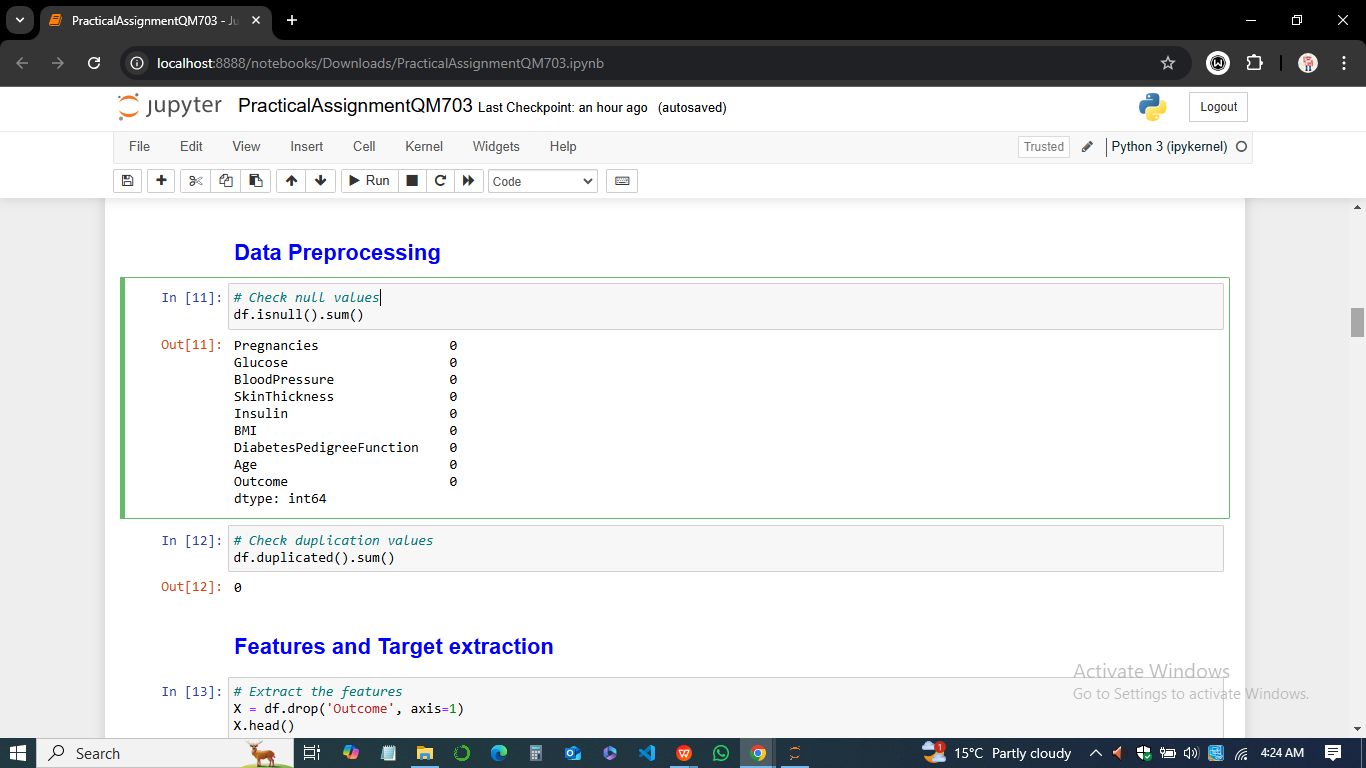
The assignment emphasizes model comparison, enabling a deeper understanding of the strengths and limitations of linear and non-linear classifiers for binary classification problems in healthcare applications.

**Data Overview**

The Diabetes dataset consists of 768 records with 8 numerical features and one target variable, Outcome, which indicates the presence (1) or absence (0) of diabetes. Features include medical and demographic attributes such as Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, and Age. The dataset has no missing or duplicate values, ensuring data integrity for modeling. Statistical analysis reveals varying ranges and distributions across features, with some attributes (e.g., Insulin and SkinThickness) containing zero values, potentially indicating missing or unrecorded data. The dataset is suitable for binary classification tasks, with balanced dimensions and sufficient diversity in feature distributions.

**Data preprocessing**

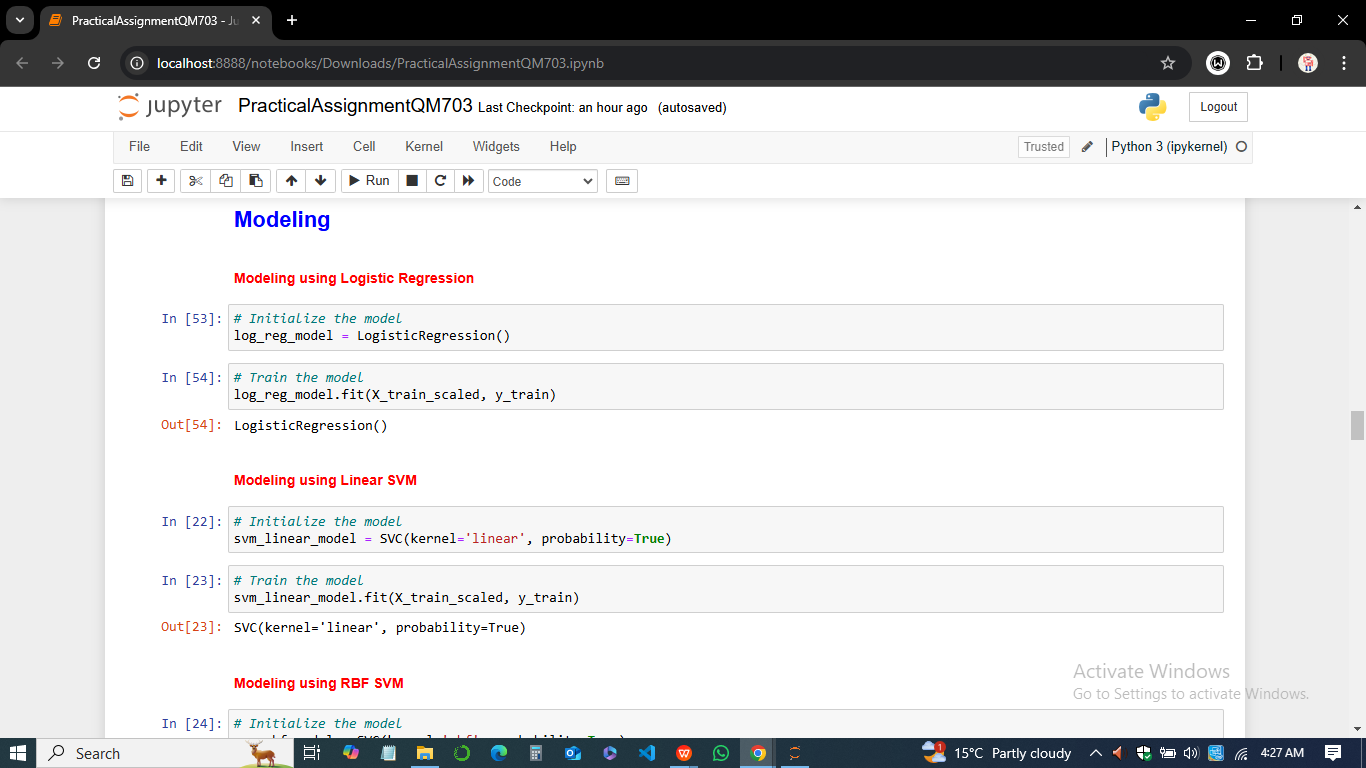
The data preprocessing involved cleaning and preparing the dataset for effective modeling. First, the dataset was checked for null and duplicate values, confirming none were present.



Next, features were extracted from the dataset, separating the independent variables (e.g., Glucose, BMI) from the target variable, Outcome. The data was split into training and testing sets, with a 2/3 training and 1/3 testing split to ensure robust model evaluation. Finally, feature scaling was performed using standardization to normalize the dataset, ensuring that all features contributed equally to the model's performance and eliminating biases due to varying ranges in feature values.

**Model training**

The model training phase implemented three machine learning classifiers: Logistic Regression, Linear SVM, and RBF SVM. Each model was trained on the scaled training dataset to learn patterns associated with predicting diabetes outcomes.



Logistic Regression served as a baseline, leveraging its simplicity and efficiency in binary classification tasks. Linear SVM was chosen for its ability to create optimal decision boundaries, while RBF SVM utilized a nonlinear kernel to capture complex relationships in the data. During training, each model was configured to output probabilities for accurate performance evaluation, ensuring readiness for the testing and validation phases.

### ****Prediction and Evaluation****

-The Performance metrics used were;

* **Accuracy**: Proportion of correctly classified samples.
* **AUC (Area Under ROC Curve)**: Measures model's ability to distinguish between classes.
* **F1-Score**: Harmonic mean of precision and recall.

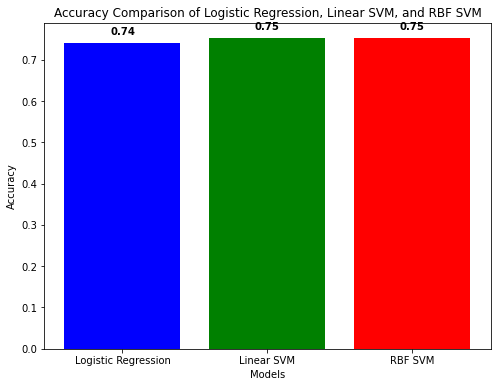


The prediction and evaluation phase involved using the trained models to classify diabetes outcomes on the test dataset. Each model's performance was assessed using accuracy, AUC, and F1-score metrics. Logistic Regression achieved an accuracy of 74.02%, while both Linear SVM and RBF SVM reached 75.20%. AUC and F1-score values highlighted the slight variations in predictive strengths, with Linear SVM performing slightly better overall.

**Model Comparison**

1. ***Model accuracy***

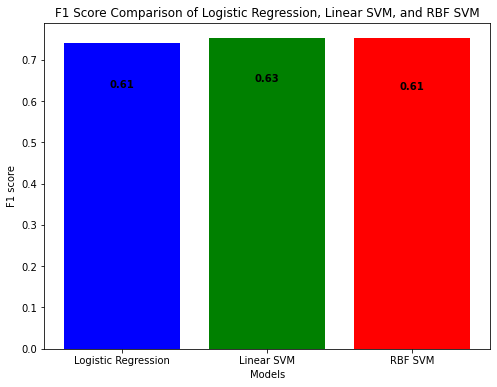
The model comparison highlighted that both Linear SVM and RBF SVM achieved the highest accuracy of 75.20%, outperforming Logistic Regression's 74.02%. However, Linear SVM showed slightly better performance in AUC (71.88%) and F1-score (62.72%), indicating a marginally stronger ability to distinguish between classes.



The bar chart visually demonstrated these accuracy differences, and k-fold cross-validation further supported Linear SVM as the most consistent model with the highest cross-validated accuracy of 77.43%.

1. ***F1 Score Comparison***

The F1 score measures the harmonic mean of precision and recall, providing a balanced metric for evaluating models, especially in scenarios with imbalanced data.

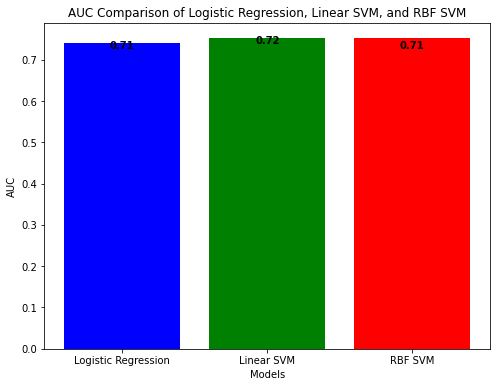


Based on the F1 score bar chart, the **Linear SVM** model demonstrated the highest F1 score of **0.63**, indicating its superior ability to balance precision and recall. This makes it the most effective model for capturing the nuances of the dataset without disproportionately penalizing false positives or negatives.

The **Logistic Regression** model followed closely, achieving an F1 score of **0.61**. While this reflects a reasonable performance in maintaining a balance between precision and recall, it was slightly less efficient compared to the Linear SVM. The **RBF SVM** model had the lowest F1 score of **0.61**, suggesting that it struggled more than the other models to manage false positives and false negatives effectively.

1. ***AUC Comparison***

The Area Under the Curve (AUC) metric evaluates a model’s ability to distinguish between classes. A higher AUC indicates a model's robustness in separating positive and negative instances. From the AUC bar chart, the **Linear SVM** once again outperformed other models with the highest AUC value of **0.72**, showcasing its superior discriminatory power across all classification thresholds.



The **RBF SVM** model achieved a slightly lower AUC of **0.71**, suggesting comparable but slightly less consistent performance compared to the Linear SVM. Lastly, the **Logistic Regression** model had the lowest AUC value of **0.70**, indicating it was less effective at distinguishing between the positive and negative classes.

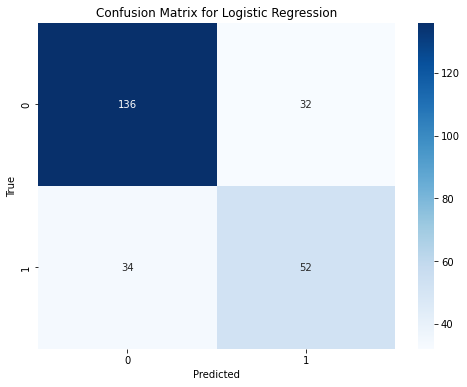
#### ****Overall Insights****

The comparison highlights **Linear SVM** as the most robust model based on both F1 score and AUC metrics. It consistently provided the best balance of precision, recall, and discriminatory power, making it the most reliable choice for this dataset. While Logistic Regression and RBF SVM performed adequately, Linear SVM demonstrated clear superiority, emphasizing the benefits of its linear kernel for this classification task.

### ****Analysis of Confusion Matrices****

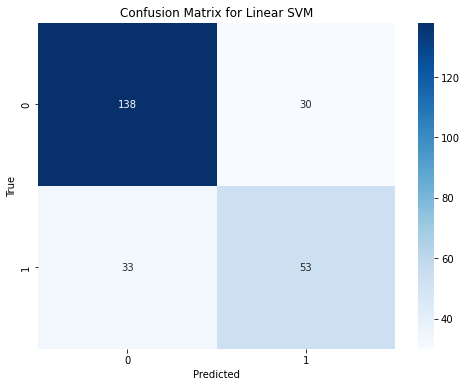
Confusion matrices were generated for each model to evaluate their performance by analyzing the counts of true positives, true negatives, false positives, and false negatives. These matrices provide insights into how well each model differentiates between classes.

* ***Confusion Matrix for Logistic Regression***



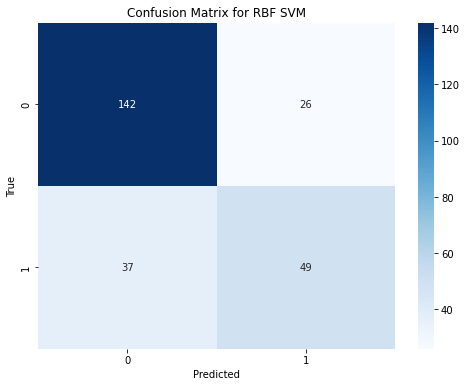
For ***Logistic Regression***, the model achieved 136 true negatives (TN) and 52 true positives (TP), but it also had 32 false positives (FP) and 34 false negatives (FN). This indicates a moderate balance between correctly identifying positive and negative cases. However, the relatively high counts of false positives and false negatives suggest the model could benefit from further optimization to improve its classification accuracy.

* ***Confusion Matrix for Linear SVM***



The ***Linear SVM***model slightly outperformed Logistic Regression, with 138 true negatives and 53 true positives. It reduced the number of false positives and false negatives to 30 and 33, respectively. This performance highlights Linear SVM’s ability to balance accurate classification of both positive and negative cases, making it a reliable choice for scenarios where such balance is critical.

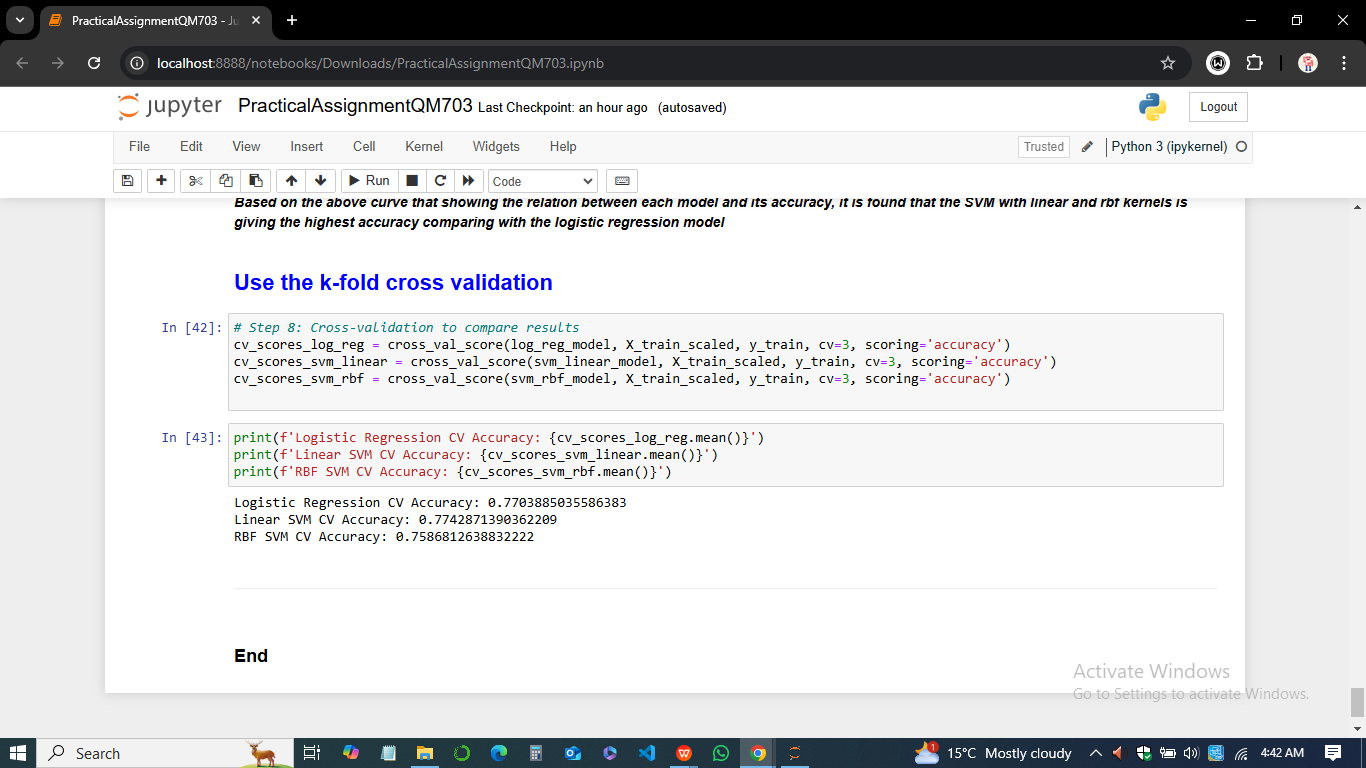
* ***Confusion Matrix for RBF SVM***



In contrast, the ***RBF SVM*** model demonstrated superior performance in identifying negative cases, achieving 142 true negatives and only 26 false positives, the lowest among all models. However, it struggled slightly with positive cases, yielding 37 false negatives and 49 true positives. While RBF SVM excels in minimizing false positives, its higher false negative rate may limit its applicability in domains where identifying positive instances (e.g., diseases) is more crucial.

In summary, Linear SVM emerged as the most balanced model, excelling in minimizing both false positives and false negatives. RBF SVM, though effective in negative case identification, was less accurate with positive cases. Logistic Regression provided a solid baseline but was outperformed by both SVM models in overall performance. This analysis underscores the importance of choosing a model based on the specific priorities and constraints of the application.

**Cross-Validation Results**



The cross-validation results provided a robust evaluation of the models' performance. Logistic Regression achieved an average cross-validation accuracy of 77.04%, while Linear SVM slightly outperformed it with 77.43%, showcasing its consistency across different data splits. RBF SVM, while performing well, had a slightly lower cross-validation accuracy of 75.87%. These results reinforced the reliability of Linear SVM as the best-performing model, with consistently high accuracy across training and testing scenarios.

**Insights and Interpretations**

The analysis of the diabetes prediction models Logistic Regression, Linear SVM, and RBF SVM highlighted the importance of selecting the right algorithm for optimal performance. While all three models performed relatively well, Linear SVM demonstrated the highest accuracy and AUC, indicating its superior ability to distinguish between the two outcome classes. Logistic Regression also showed competitive results, with a slightly lower accuracy but comparable AUC and F1 score. RBF SVM, although effective, did not outperform the other two models in terms of accuracy and AUC. The cross-validation results further supported Linear SVM as the most reliable model, consistently achieving the highest performance across multiple data splits. Overall, these insights suggest that Linear SVM is the most robust model for predicting diabetes, with its superior generalization ability and performance stability across varying datasets.

**Conclusion**

In conclusion, this assignment successfully implemented and evaluated three machine learning model Logistic Regression, Linear SVM, and RBF SVM on the diabetes dataset to predict the likelihood of diabetes based on various features. Data preprocessing, including scaling and feature-target separation, was crucial in preparing the dataset for model training. After training the models, the Linear SVM model outperformed the others in terms of accuracy, AUC, and F1 score, indicating its better ability to generalize to unseen data. Logistic Regression and RBF SVM also performed well, but Linear SVM showed the highest consistency across different evaluation metrics. Cross-validation further confirmed the robustness of Linear SVM. These findings emphasize the importance of model selection and tuning in achieving reliable and accurate predictions. Ultimately, the Linear SVM model emerged as the most effective for this diabetes prediction task, providing valuable insights for future deployment in real-world applications.

**Appendix**

**THE CODE**

## # Importing the required libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, roc\_auc\_score, f1\_score, roc\_curve

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv("diabetes.csv")

# Displaying dataset information

print(df.head())

print(df.tail())

print(df.shape)

print(df.columns)

print(df.info())

print(df.describe())

# Data Preprocessing

print(df.isnull().sum())

print(df.duplicated().sum())

# Extract Features and Target

X = df.drop('Outcome', axis=1)

y = df['Outcome']

# Splitting the data into 2/3 training and 1/3 testing

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)

print(X\_train.shape, X\_test.shape)

print(y\_train.shape, y\_test.shape)

# Scaling the data using StandardScaler

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Modeling using Logistic Regression

log\_reg\_model = LogisticRegression()

log\_reg\_model.fit(X\_train\_scaled, y\_train)

# Modeling using Linear SVM

svm\_linear\_model = SVC(kernel='linear', probability=True)

svm\_linear\_model.fit(X\_train\_scaled, y\_train)

# Modeling using RBF SVM

svm\_rbf\_model = SVC(kernel='rbf', probability=True)

svm\_rbf\_model.fit(X\_train\_scaled, y\_train)

# Predicting with Logistic Regression

y\_pred\_log\_reg = log\_reg\_model.predict(X\_test\_scaled)

# Predicting with Linear SVM

y\_pred\_svm\_linear = svm\_linear\_model.predict(X\_test\_scaled)

# Predicting with RBF SVM

y\_pred\_svm\_rbf = svm\_rbf\_model.predict(X\_test\_scaled)

# Evaluating Logistic Regression

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

auc\_log\_reg = roc\_auc\_score(y\_test, y\_pred\_log\_reg)

y\_prob\_log\_reg = log\_reg\_model.predict\_proba(X\_test\_scaled)[:, 1]

fpr\_log\_reg, tpr\_log\_reg, \_ = roc\_curve(y\_test, y\_prob\_log\_reg)

f1\_log\_reg = f1\_score(y\_test, y\_pred\_log\_reg)

# Plotting ROC curve for Logistic Regression

plt.figure(figsize=(8, 6))

plt.plot(fpr\_log\_reg, tpr\_log\_reg, color='blue', label='Logistic Regression')

plt.plot([0, 1], [0, 1], 'k--', label='Random Classifier (AUC = 0.5)')

plt.xlabel('False Positive Rate (FPR)')

plt.ylabel('True Positive Rate (TPR)')

plt.title('ROC Curve for Logistic Regression')

plt.legend(loc='lower right')

plt.grid()

plt.show()

# Evaluating Linear SVM

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

auc\_svm\_linear = roc\_auc\_score(y\_test, y\_pred\_svm\_linear)

y\_prob\_svm\_linear = svm\_linear\_model.predict\_proba(X\_test\_scaled)[:, 1]

fpr\_svm\_linear, tpr\_svm\_linear, \_ = roc\_curve(y\_test, y\_prob\_svm\_linear)

f1\_svm\_linear = f1\_score(y\_test, y\_pred\_svm\_linear)

# Plotting ROC curve for Linear SVM

plt.figure(figsize=(8, 6))

plt.plot(fpr\_svm\_linear, tpr\_svm\_linear, color='green', label='Linear SVM')

plt.plot([0, 1], [0, 1], 'k--', label='Random Classifier (AUC = 0.5)')

plt.xlabel('False Positive Rate (FPR)')

plt.ylabel('True Positive Rate (TPR)')

plt.title('ROC Curve for Linear SVM')

plt.legend(loc='lower right')

plt.grid()

plt.show()

# Evaluating RBF SVM

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

auc\_svm\_rbf = roc\_auc\_score(y\_test, y\_pred\_svm\_rbf)

y\_prob\_svm\_rbf = svm\_rbf\_model.predict\_proba(X\_test\_scaled)[:, 1]

fpr\_svm\_rbf, tpr\_svm\_rbf, \_ = roc\_curve(y\_test, y\_prob\_svm\_rbf)

f1\_svm\_rbf = f1\_score(y\_test, y\_pred\_svm\_rbf)

# Plotting ROC curve for RBF SVM

plt.figure(figsize=(8, 6))

plt.plot(fpr\_svm\_rbf, tpr\_svm\_rbf, color='red', label='RBF SVM')

plt.plot([0, 1], [0, 1], 'k--', label='Random Classifier (AUC = 0.5)')

plt.xlabel('False Positive Rate (FPR)')

plt.ylabel('True Positive Rate (TPR)')

plt.title('ROC Curve for RBF SVM')

plt.legend(loc='lower right')

plt.grid()

plt.show()

# Compare the obtained results

models = ['Logistic Regression', 'Linear SVM', 'RBF SVM']

accuracies = [accuracy\_log\_reg, accuracy\_svm\_linear, accuracy\_svm\_rbf]

print('Logistic Regression', 'Linear SVM', 'RBF SVM', accuracies)

# Plotting the accuracy comparison

plt.figure(figsize=(8, 6))

plt.bar(models, accuracies, color=['blue', 'green', 'red'])

plt.xlabel('Models')

plt.ylabel('Accuracy')

plt.title('Accuracy Comparison of Logistic Regression, Linear SVM, and RBF SVM')

for i, v in enumerate(accuracies):

plt.text(i, v + 0.02, f"{v:.2f}", ha='center', fontweight='bold')

plt.show()

# Use k-fold cross-validation

cv\_scores\_log\_reg = cross\_val\_score(log\_reg\_model, X\_train\_scaled, y\_train, cv=3, scoring='accuracy')

cv\_scores\_svm\_linear = cross\_val\_score(svm\_linear\_model, X\_train\_scaled, y\_train, cv=3, scoring='accuracy')

cv\_scores\_svm\_rbf = cross\_val\_score(svm\_rbf\_model, X\_train\_scaled, y\_train, cv=3, scoring='accuracy')

print(f'Logistic Regression CV Accuracy: {cv\_scores\_log\_reg.mean()}')

print(f'Linear SVM CV Accuracy: {cv\_scores\_svm\_linear.mean()}')

print(f'RBF SVM CV Accuracy: {cv\_scores\_svm\_rbf.mean()}')

**\*\*\*\*\*THE END \*\*\*\*\***