Assignment-07

**Name:**M.gnana sai likhitha

**Code:**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix, classification\_report, roc\_curve, auc

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

# Load the dataset

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.data"

column\_url = "https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.names"

# Names were extracted from the .names file manually

columns = [f"feature\_{i}" for i in range(57)] + ['target']

df = pd.read\_csv(url, header=None, names=columns)

# 1. Handle missing values (no missing in this dataset)

print(df.isnull().sum().sum())  # Just to check

# 2. Feature scaling

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

y = df['target']

scaler = StandardScaler()

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

# 3. Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

# 4. Train two models: Logistic Regression and Random Forest

log\_reg = LogisticRegression()

rf = RandomForestClassifier(random\_state=42)

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

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

# 5. Predictions

y\_pred\_lr = log\_reg.predict(X\_test)

y\_pred\_rf = rf.predict(X\_test)

# 6. Evaluation function

def evaluate\_model(y\_true, y\_pred, name):

    print(f"\nModel: {name}")

    print("Accuracy:", accuracy\_score(y\_true, y\_pred))

    print("Precision:", precision\_score(y\_true, y\_pred))

    print("Recall:", recall\_score(y\_true, y\_pred))

    print("F1 Score:", f1\_score(y\_true, y\_pred))

    print("\nConfusion Matrix:\n", confusion\_matrix(y\_true, y\_pred))

    print("\nClassification Report:\n", classification\_report(y\_true, y\_pred))

# 7. Evaluate both models

evaluate\_model(y\_test, y\_pred\_lr, "Logistic Regression")

evaluate\_model(y\_test, y\_pred\_rf, "Random Forest")

# 8. ROC Curve

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

y\_prob\_rf = rf.predict\_proba(X\_test)[:, 1]

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

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

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

plt.plot(fpr\_lr, tpr\_lr, label="Logistic Regression (AUC = {:.2f})".format(auc(fpr\_lr, tpr\_lr)))

plt.plot(fpr\_rf, tpr\_rf, label="Random Forest (AUC = {:.2f})".format(auc(fpr\_rf, tpr\_rf)))

plt.plot([0, 1], [0, 1], 'k--')

plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.title("ROC Curve")

plt.legend()

plt.show()

# 9. GridSearchCV for Random Forest

param\_grid = {

    'n\_estimators': [50, 100],

    'max\_depth': [5, 10, None]

}

grid\_search = GridSearchCV(RandomForestClassifier(random\_state=42), param\_grid, cv=3, scoring='f1')

grid\_search.fit(X\_train, y\_train)

print("\nBest Parameters (Random Forest):", grid\_search.best\_params\_)

# 10. Feature Importance (Random Forest)

importances = rf.feature\_importances\_

feature\_names = df.columns[:-1]

feat\_df = pd.DataFrame({"Feature": feature\_names, "Importance": importances})

feat\_df = feat\_df.sort\_values(by="Importance", ascending=False)

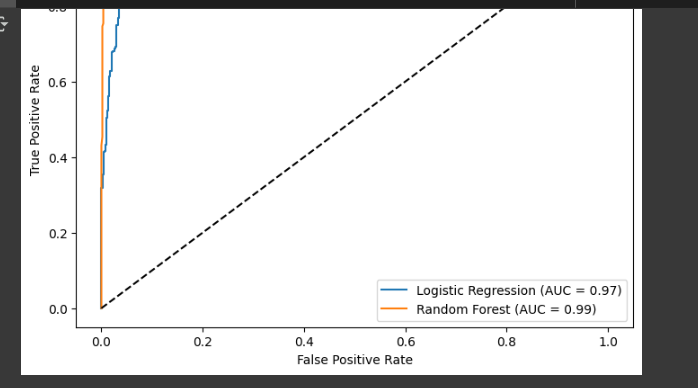
plt.figure(figsize=(10, 6))

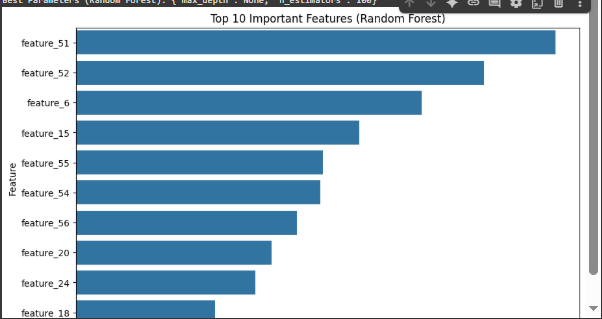
sns.barplot(x="Importance", y="Feature", data=feat\_df.head(10))

plt.title("Top 10 Important Features (Random Forest)")

plt.show()

**output:**

****

****