**Name: Ufaq Zahra**

**SEMESTER: 2nd**

**ROLL NO: SU92-MPMMW-F23-012**

**Assignment NO: 2**

**# Import necessary libraries**

import pandas as pd

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

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import BaggingClassifier, AdaBoostClassifier, GradientBoostingClassifier, StackingClassifier, VotingClassifier

from sklearn.ensemble import RandomForestClassifier

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

import matplotlib.pyplot as plt

**# Load the dataset**

df = pd.read\_csv('/content/diabetes.csv')

**# Assume the last column is the target and the rest are features**

X = df.iloc[:, :-1]

y = df.iloc[:, -1]

**# Split the dataset into training and testing sets**

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

**# Standardize the features**

scaler = StandardScaler()

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

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

**# Initialize the base logistic regression model**

log\_reg = LogisticRegression()

**# Fit the logistic regression model**

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

**# Predictions**

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

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

**# Calculate metrics**

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

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

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

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

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

roc\_auc = roc\_auc\_score(y\_test, y\_pred\_prob)

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

**# Display metrics**

print(f"Accuracy: {accuracy:.4f}")

print(f"Precision: {precision:.4f}")

print(f"Recall: {recall:.4f}")

print(f"F1 Score: {f1:.4f}")

print("Confusion Matrix:")

print(conf\_matrix)

print(f"ROC AUC: {roc\_auc:.4f}")

**# Plot ROC Curve**

plt.figure()

plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (area = {roc\_auc:.4f})')

plt.plot([0, 1], [0, 1], color='red', linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic')

plt.legend()

plt.show()

**# Bagging with Logistic Regression**

bagging = BaggingClassifier(base\_estimator=LogisticRegression(), n\_estimators=50, random\_state=42)

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

y\_pred\_bagging = bagging.predict(X\_test)

print(f"Bagging Accuracy: {accuracy\_score(y\_test, y\_pred\_bagging):.4f}")

**# Boosting with Logistic Regression**

boosting = AdaBoostClassifier(base\_estimator=LogisticRegression(), n\_estimators=50, random\_state=42)

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

y\_pred\_boosting = boosting.predict(X\_test)

print(f"Boosting Accuracy: {accuracy\_score(y\_test, y\_pred\_boosting):.4f}")

**# Stacking with Logistic Regression**

estimators = [

('lr1', LogisticRegression()),

('lr2', LogisticRegression(penalty='l1', solver='liblinear')),

('lr3', LogisticRegression(penalty='l2'))

]

stacking = StackingClassifier(estimators=estimators, final\_estimator=LogisticRegression())

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

y\_pred\_stacking = stacking.predict(X\_test)

print(f"Stacking Accuracy: {accuracy\_score(y\_test, y\_pred\_stacking):.4f}")

**# Voting with Logistic Regression**

voting = VotingClassifier(estimators=estimators, voting='soft')

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

y\_pred\_voting = voting.predict(X\_test)

print(f"Voting Accuracy: {accuracy\_score(y\_test, y\_pred\_voting):.4f}")

**# Random Forest with Logistic Regression as base learner**

class CustomRandomForest:

def \_\_init\_\_(self, n\_estimators=10):

self.n\_estimators = n\_estimators

self.models = [LogisticRegression() for \_ in range(n\_estimators)]

self.feature\_subsets = []

def fit(self, X, y):

for i in range(self.n\_estimators):

# Randomly select features for each model

subset = np.random.choice(X.shape[1], size=int(np.sqrt(X.shape[1])), replace=False)

self.feature\_subsets.append(subset)

X\_subset = X[:, subset]

self.models[i].fit(X\_subset, y)

def predict(self, X):

preds = []

for i in range(self.n\_estimators):

X\_subset = X[:, self.feature\_subsets[i]]

preds.append(self.models[i].predict(X\_subset))

preds = np.array(preds)

# **Voting mechanism**

final\_pred = np.mean(preds, axis=0) > 0.5

return final\_pred.astype(int)

**# Use the custom Random Forest**

rf\_lr = CustomRandomForest(n\_estimators=10)

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

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

print(f"Random Forest with Logistic Regression Accuracy: {accuracy\_score(y\_test, y\_pred\_rf\_lr):.4f}")

**Output:**

**Accuracy: 0.7359**

**Precision: 0.6173**

**Recall: 0.6250**

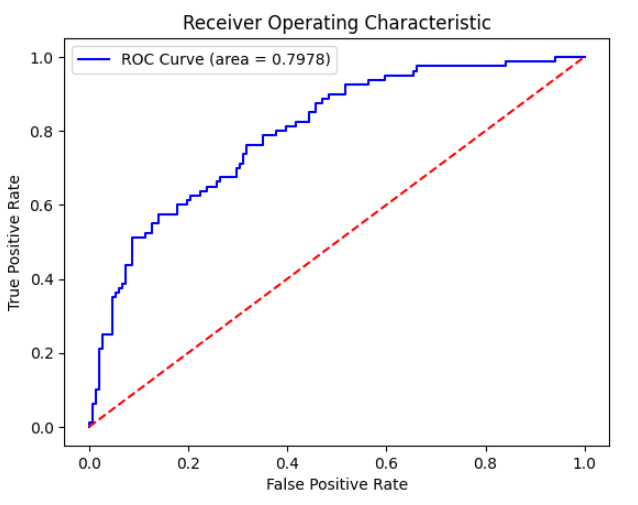
**F1 Score: 0.6211**

**Confusion Matrix:**

**[[120 31]**

**[ 30 50]]**

**ROC AUC: 0.7978**



**Bagging Accuracy: 0.7273**

**Boosting Accuracy: 0.7013**

**Stacking Accuracy: 0.7316**

**Voting Accuracy: 0.7316**