**Aim**

To build a predictive model for diabetes using Decision Tree, AdaBoost, Gradient Boosting, and XGBoost classifiers on the Pima Indians Diabetes dataset. The objective is to identify the best model based on performance metrics such as accuracy, precision, recall, F1-score, and confusion matrix. Additionally, predefined user input will be used to predict diabetes.

**Problem Statement**

Diabetes is a chronic condition characterized by high blood sugar levels. The aim is to develop a model that can predict the likelihood of diabetes based on various health parameters such as glucose levels, BMI, age, etc. Accurate prediction can lead to early detection and treatment, improving patient outcomes.

**Procedure**

1. **Data Loading and Preprocessing**
   * The dataset is loaded from an online repository.
   * The data contains several features including pregnancies, glucose levels, blood pressure, and more, with the Outcome variable indicating whether the patient has diabetes.
   * The data is checked for duplicates and missing values, which are removed to ensure data quality.
2. **Splitting Data into Training and Testing Sets**
   * The dataset is split into training (70%) and testing (30%) sets to evaluate the model's performance.
3. **Hyperparameter Tuning for Decision Tree**
   * A grid search is performed over a range of max\_depth values for the Decision Tree classifier to find the optimal depth that maximizes accuracy.
   * The best parameters and cross-validation accuracy are obtained.
4. **Model Training and Evaluation**
   * The Decision Tree classifier is trained using the best parameters from the grid search.
   * The model is evaluated on the test set using accuracy, classification report, and confusion matrix.
   * Additional ensemble methods such as AdaBoost, Gradient Boosting, and XGBoost are used to potentially improve performance.
   * Each model's performance is evaluated using the same metrics.
5. **Prediction on New Data**
   * Predefined user input representing a patient's health parameters is used to predict whether the patient has diabetes using the trained models.

**Source Code**

import pandas as pd

import numpy as np

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

import xgboost as xgb

# Load the dataset

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"

columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']

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

# Data preprocessing

df = df.drop\_duplicates()

df = df.dropna()

X = df.drop(columns=['Outcome'])

y = df['Outcome']

# Split the data 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)

# Define the parameter grid for Decision Tree

param\_grid = {

'max\_depth': np.arange(1, 20)

}

# Create a GridSearchCV object

grid\_search = GridSearchCV(estimator=DecisionTreeClassifier(random\_state=42), param\_grid=param\_grid, cv=5, scoring='accuracy')

# Fit the grid search to the data

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

# Get the best parameters and the best score

best\_params = grid\_search.best\_params\_

best\_score = grid\_search.best\_score\_

print("Best Parameters:", best\_params)

print("Best Cross-Validation Accuracy:", best\_score)

# Train the Decision Tree with the best parameters

clf = DecisionTreeClassifier(\*\*best\_params, random\_state=42)

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

# Predict on the test set

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

# Calculate the accuracy score

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

print(f"Accuracy: {accuracy:.2f}")

# Print the classification report

report = classification\_report(y\_test, y\_pred, target\_names=["No Diabetes", "Diabetes"])

print("Classification Report:")

print(report)

# Print the confusion matrix

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

print("Confusion Matrix:")

print(conf\_matrix)

# 1. AdaBoost with Decision Tree

ada\_clf = AdaBoostClassifier(estimator=clf, n\_estimators=100, random\_state=42)

ada\_clf.fit(X\_train, y\_train)

y\_pred\_ada = ada\_clf.predict(X\_test)

print("\nAdaBoost with Decision Tree Performance:")

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred\_ada):.2f}")

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred\_ada))

print("Classification Report:")

print(classification\_report(y\_test, y\_pred\_ada))

# 2. Gradient Boosting

gb\_clf = GradientBoostingClassifier(n\_estimators=100, max\_depth=best\_params['max\_depth'], random\_state=42)

gb\_clf.fit(X\_train, y\_train)

y\_pred\_gb = gb\_clf.predict(X\_test)

print("\nGradient Boosting Classifier Performance:")

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred\_gb):.2f}")

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred\_gb))

print("Classification Report:")

print(classification\_report(y\_test, y\_pred\_gb))

# 3. XGBoost

xgb\_clf = xgb.XGBClassifier(n\_estimators=100, max\_depth=best\_params['max\_depth'], random\_state=42)

xgb\_clf.fit(X\_train, y\_train)

y\_pred\_xgb = xgb\_clf.predict(X\_test)

print("\nXGBoost Classifier Performance:")

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred\_xgb):.2f}")

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred\_xgb))

print("Classification Report:")

print(classification\_report(y\_test, y\_pred\_xgb))

# Predefined user input for prediction

user\_input = {

'Pregnancies': 2,

'Glucose': 85,

'BloodPressure': 75,

'SkinThickness': 30,

'Insulin': 90,

'BMI': 28.1,

'DiabetesPedigreeFunction': 0.5,

'Age': 25

}

user\_df = pd.DataFrame([user\_input])

user\_df = user\_df.reindex(columns=X.columns, fill\_value=0)

# Predict based on user input

user\_prediction\_ada = ada\_clf.predict(user\_df)

print("\nAdaBoost Classifier Prediction:", "Diabetes" if user\_prediction\_ada[0] == 1 else "No Diabetes")

user\_prediction\_gb = gb\_clf.predict(user\_df)

print("Gradient Boosting Classifier Prediction:", "Diabetes" if user\_prediction\_gb[0] == 1 else "No Diabetes")

user\_prediction\_xgb = xgb\_clf.predict(user\_df)

print("XGBoost Classifier Prediction:", "Diabetes" if user\_prediction\_xgb[0] == 1 else "No Diabetes")

**Output**

Best Parameters: {'max\_depth': 5}

Best Cross-Validation Accuracy: 0.7504

Accuracy: 0.75

Classification Report:

precision recall f1-score support

No Diabetes 0.78 0.85 0.82 151

Diabetes 0.67 0.55 0.60 80

accuracy 0.75 231

macro avg 0.72 0.70 0.71 231

weighted avg 0.74 0.75 0.74 231

Confusion Matrix:

[[129 22]

[ 36 44]]

AdaBoost with Decision Tree Performance:

Accuracy: 0.74

Confusion Matrix:

[[116 35]

[ 24 56]]

Classification Report:

precision recall f1-score support

0 0.83 0.77 0.80 151

1 0.62 0.70 0.65 80

accuracy 0.74 231

macro avg 0.72 0.73 0.73 231

weighted avg 0.75 0.74 0.75 231

Gradient Boosting Classifier Performance:

Accuracy: 0.73

Confusion Matrix:

[[115 36]

[ 26 54]]

Classification Report:

precision recall f1-score support

0 0.82 0.76 0.79 151

1 0.60 0.68 0.64 80

accuracy 0.73 231

macro avg 0.71 0.72 0.71 231

weighted avg 0.74 0.73 0.73 231

XGBoost Classifier Performance:

Accuracy: 0.74

Confusion Matrix:

[[116 35]

[ 25 55]]

Classification Report:

precision recall f1-score support

0 0.82 0.77 0.80 151

1 0.61 0.69 0.65 80

accuracy 0.74 231

macro avg 0.71 0.73 0.72 231

weighted avg 0.75 0.74 0.74 231

AdaBoost Classifier Prediction: No Diabetes

Gradient Boosting Classifier Prediction: No Diabetes

XGBoost Classifier Prediction: No Diabetes

INFERENCE

The diabetes prediction model demonstrates a robust performance, with the Decision Tree classifier achieving the highest accuracy of 75%, effectively identifying non-diabetic individuals while highlighting the need for improved recall in detecting diabetic patients, which stands at 55%. This moderate level of predictive capability underscores the importance of targeted healthcare programs and patient education to ensure timely interventions for at-risk individuals. By investing in continuous model improvements and advanced machine learning techniques, healthcare providers can enhance diagnostic accuracy, optimize resource allocation, and ultimately improve patient outcomes, while also implementing strategies to mitigate the risk of misclassifying diabetes cases.