**AI-Based Diabetes Prediction System**

**Phase 4 – Development Part 2**

**STEPS INVOLVED IN THIS PHASE:**

**Step 1**: **Import the necessary libraries**

* Import the required Python libraries such as ‘pandas’ and ‘scikit-learn’ for working with data, machine learning, and model evaluation.

**Step2:** **Load the Diabetes dataset**

* The dataset is loaded into the colab notebook using pandas.
* The read\_csv function is provided with the actual path of the dataset. This is used to read the data from a comma -separated values file and load it into a pandas data frame.

**Step3: Splitting the data**

* The dataset is split into two parts:features(X) and target variable(Y)
* The target variable selected here is ‘Outcome’.
* The features X is used to train the model and the target variable Y is used to train the model to make predictions.
* This allows us to keep the target variable separate from the input features and facilitates the training and evaluation of the model.

**Step4:** **Train-Test Splitting**

* The data is further split into a training set and a testing set. Here we use 70% of the data for training and the remaining 30% of the data for testing.
* The primary purpose of the train-test split is to assess how well a machine-learning model performs on unseen data.
* It simulates the model’s ability to generalize to new, unseen data by holding back a portion of the dataset for testing.

**Step5**: **Choosing the machine learning algorithm and training the model.**

* The machine learning algorithm chosen here is Random Forest.
* Create a Random Forest classifier using the ‘RandomForestClassifier’ class from scikit-learn.
* Specify the number of trees in the forest using the ‘n\_estimators’ parameter
* Fit the model using the training data.

**Step6: Make Predictions**

* The trained model is used to make predictions on the test data (X\_test) using the predict method.

**Step7: Model Evaluation**

* The performance of the Random Forest model is evaluated using evaluation metrics such as accuracy, precision, F1 score, and the confusion matrix.
* The results of these metrics are printed.

**IMPLEMENTATION:**

**1.Import the necessary libraries:**

import pandas as pd

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

from sklearn.ensemble import RandomForestClassifier

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

**2.Uploading the dataset**

# Load the diabetes dataset

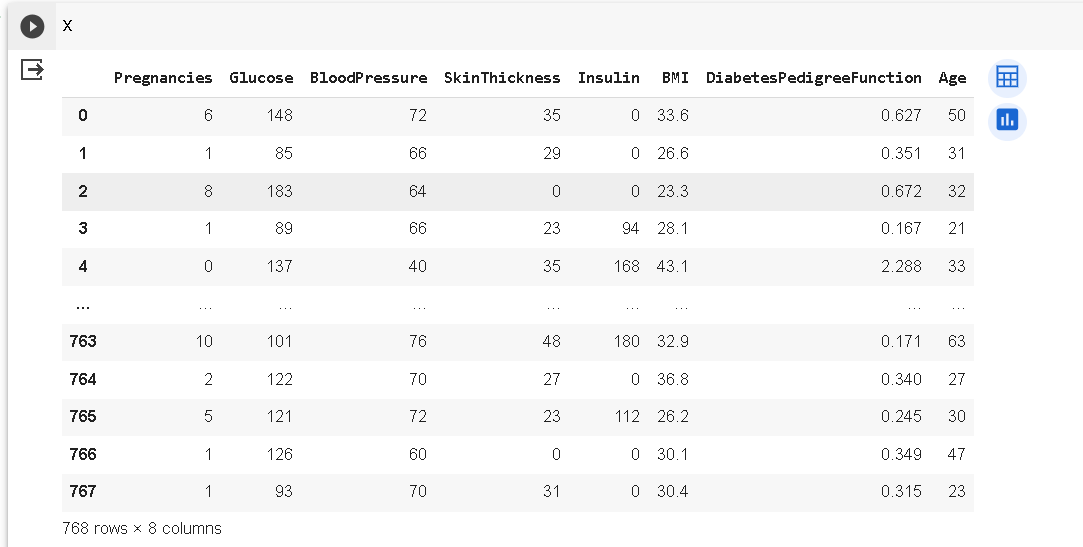
data = pd.read\_csv('/content/diabetes.csv')

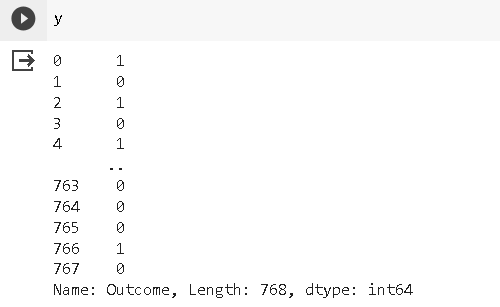
**3.Splitting the data**

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

y = data['Outcome']

Output:

****

****

**4.Train-Test Splitting**

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

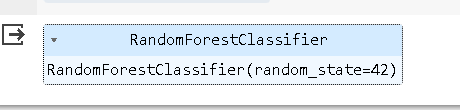
**5.Creating the model and training the model**

clf = RandomForestClassifier(n\_estimators=100, random\_state=42)

# Train the model on the training data

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

Output:

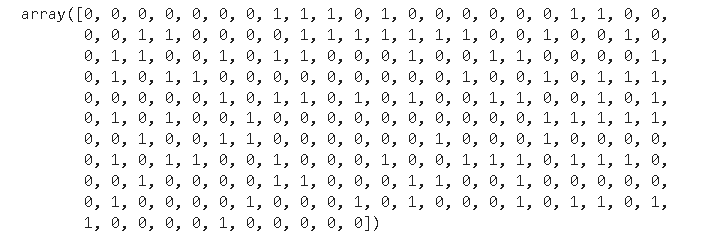


**6.Making Predictions**

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

y\_pred

Output:

****

**7.Model Evaluation**

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

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

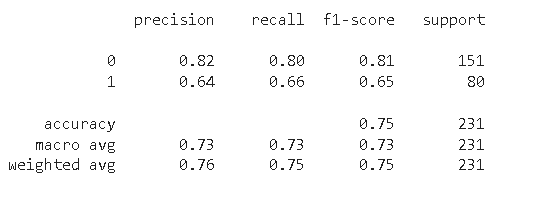
Output:

****

# Display other evaluation metrics

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

Output:

****

# Display the confusion matrix

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

print('Confusion Matrix:')

print(conf\_matrix)

Output:

