**PROJECT TITLE: AI BASED DIABETES PREDICTION SYSTEM**

**Phase 3: Development Part 1**

In this phase, I am building my AI based Diabetes Prediction System project by loading and preprocessing the dataset with the help of machine learning techniques.

**OBJECTIVE:**

The objective of the project to develop an AI-powered diabetes prediction system is to provide early risk assessment and personalized preventive measures for individuals, allowing them to take proactive actions to manage their health.

**DATASET:**

<https://www.kaggle.com/datasets/mathchi/diabetes-data-set>

**PROGRAM CODE FOR LOADING AND PREPROCESSING THE DATASET:**

*# Import necessary libraries*

import pandas as pd

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

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier

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

***# Step 1:******Load the dataset***

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

***# Step 2: Data Preprocessing***

*# Separate features (X) and target variable (y)*

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

y = data['Outcome']

*# Handle missing values*

X = X.fillna(X.mean())

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

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

*#* ***Standardize****/normalize the features*

scaler = StandardScaler()

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

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

***# Step 3: Model Building (Random Forest Classifier)***

clf = RandomForestClassifier(random\_state=42)

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

***# Step 4: Model Evaluation***

*# Make predictions on the test set*

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

*# Calculate evaluation 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)

roc\_auc = roc\_auc\_score(y\_test, clf.predict\_proba(X\_test)[:, 1])

***# Step 5: Display Results***

print("Accuracy:", accuracy)

print("Precision:", precision)

print("Recall:", recall)

print("F1 Score:", f1)

print("ROC AUC Score:", roc\_auc)

**EXPLANATION FOR LOADING AND PREPROCESSING:**

The provided Python code performs data loading and preprocessing for the given diabetes dataset.

**Data Loading:**

1. **import pandas as pd**: This line imports the Pandas library, a powerful library for data manipulation and analysis.

2. **data = pd.read\_csv('diabetes.csv'):** This line uses Panda’s read\_csv() function to load the diabetes dataset from the 'diabetes\_data.csv' file and store it in a Pandas DataFrame called data.

**Data Preprocessing:**

*1.* *Splitting the Data:*

**X = data.drop('Outcome', axis=1)**: This line separates the features (independent variables) from the target variable 'Outcome'. The variable X now contains all the feature columns.

**y = data['Outcome']**: This line extracts the target variable Outcome and assigns it to the variabley.

*2. Handling Missing Values*:

**X = X.fillna(X.mean())**:This line fills any missing values in the feature dataset X with the mean of the respective column. This is a simple imputation method.

*3. Splitting into Training and Testing Sets:*

**train\_test\_split** from scikit-learn is used to split the data into training and testing sets. The dataset is divided into a training set (80%) and a testing set (20%). The `random\_state` parameter ensures reproducibility.

*4. Feature Scaling (Standardization):*

**scaler = StandardScaler():** This line initializes a StandardScaler, a preprocessing technique used to standardize the feature values. It ensures that all features have the same scale, which can be essential for certain machine learning algorithms to perform optimally.

**X\_train = scaler.fit\_transform(X\_train)**: It standardizes the training features, making their distribution have a mean of 0 and a standard deviation of 1.

**X\_test = scaler.transform(X\_test)**: The same scaling transformation is applied to the testing features.

In[1]:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.preprocessing import StandardScaler

from sklearn import svm

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import ConfusionMatrixDisplay

RED = "**\033**[91m"

GREEN = "**\033**[92m"

YELLOW = "**\033**[93m"

BLUE = "**\033**[94m"

RESET = "**\033**[0m"

df = pd.read\_csv("/kaggle/input/diabetes-data-set/diabetes.csv")

*# DATA CLEANING*

print(BLUE + "**\n**DATA CLEANING" + RESET)

*# --- Check for missing values*

missing\_values = df.isnull().sum()

print(GREEN + "Missing Values : " + RESET)

print(missing\_values)

*# --- Handle missing values*

mean\_fill = df.fillna(df.mean())

df.fillna(mean\_fill, inplace=True)

*# --- Check for duplicate values*

duplicate\_values = df.duplicated().sum()

print(GREEN + "Duplicate Values : " + RESET)

print(duplicate\_values)

*# --- Drop duplicate values*

df.drop\_duplicates(inplace=True)

*# DATA ANALYSIS*

print(BLUE + "**\n**DATA ANALYSIS" + RESET)

*# --- Summary Statistics*

summary\_stats = df.describe()

print(GREEN + "Summary Statistics : " + RESET)

print(summary\_stats)

*# --- Class Distribution*

class\_distribution = df["Outcome"].value\_counts()

print(GREEN + "Class Distribution : " + RESET)

print(class\_distribution)

*# Support Vector Machine Modelling*

print(BLUE + "**\n**MODELLING" + RESET)

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

y = df["Outcome"]

*# --- Splitting the data into training and testing sets*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=42

)

*# --- Standardize Features*

scaler = StandardScaler()

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

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

*# --- init and train SVM model*

model = svm.SVC(kernel="linear")

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

*# --- Predict on test data*

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

*# --- Evaluate model performance*

accuracy = model.score(X\_test, y\_test)

print(GREEN + "Model Accuracy : " + RESET)

print(accuracy)

*# --- Classification Report and Confusion Matrix*

print(GREEN + "Classification Report : " + RESET)

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

print(GREEN + "Confusion Matrix : " + RESET)

cm = ConfusionMatrixDisplay.from\_predictions(y\_test, y\_pred)

sns.heatmap(cm.confusion\_matrix, annot=True, cmap="Blues")

plt.show()

print("Displayed")

*# SAVING THE FILE*

df.to\_csv("/kaggle/working/cleaned\_diabetes.csv", index=False)

print(BLUE + "**\n**DATA SAVING" + RESET)

print(GREEN + "Data Cleaned and Saved !" + RESET)

print("**\n**")

Out[1]:

DATA CLEANING

Missing Values :

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

Duplicate Values :

0

DATA ANALYSIS

Summary Statistics :

Pregnancies Glucose BloodPressure SkinThickness Insulin \

count 768.000000 768.000000 768.000000 768.000000 768.000000

mean 3.845052 120.894531 69.105469 20.536458 79.799479

std 3.369578 31.972618 19.355807 15.952218 115.244002

min 0.000000 0.000000 0.000000 0.000000 0.000000

25% 1.000000 99.000000 62.000000 0.000000 0.000000

50% 3.000000 117.000000 72.000000 23.000000 30.500000

75% 6.000000 140.250000 80.000000 32.000000 127.250000

max 17.000000 199.000000 122.000000 99.000000 846.000000

BMI DiabetesPedigreeFunction Age Outcome

count 768.000000 768.000000 768.000000 768.000000

mean 31.992578 0.471876 33.240885 0.348958

std 7.884160 0.331329 11.760232 0.476951

min 0.000000 0.078000 21.000000 0.000000

25% 27.300000 0.243750 24.000000 0.000000

50% 32.000000 0.372500 29.000000 0.000000

75% 36.600000 0.626250 41.000000 1.000000

max 67.100000 2.420000 81.000000 1.000000

Class Distribution :

Outcome

0 500

1 268

Name: count, dtype: int64

MODELLING

Model Accuracy :

0.7597402597402597

Classification Report :

precision recall f1-score support

0 0.81 0.82 0.81 99

1 0.67 0.65 0.66 55

accuracy 0.76 154

macro avg 0.74 0.74 0.74 154

weighted avg 0.76 0.76 0.76 154

Displayed

DATA SAVING

Data Cleaned and Saved !