Predicting Diabetes with Machine Learning (Pima Indians Diabetes Dataset)

Name: Anurag Majumdar

Regd. No.: 2341013076

Sec: 2N3

Branch: B. Tech - CSE

Objective:

The goal of this project is to predict whether a patient has diabetes based on their medical details using Machine Learning.
This enables early detection and can assist healthcare professionals in decision-making.

Tools & Technologies Used

Tool / Library	Purpose
Python 3.8+	Programming language
Jupyter Notebook	Interactive environment
pandas, numpy	Data loading, cleaning, and processing
matplotlib, seaborn	Data visualization
scikit-learn	Machine learning model & evaluation

Installation:

pip install pandas numpy matplotlib seaborn scikit-learn

Dataset

- **Dataset:** Pima Indians Diabetes Database
- **Size:** 768 rows × 9 columns
- **Target Column:** Outcome (1 = Diabetes, 0 = No Diabetes)

Feature	Description
Pregnancies	Number of times pregnant
Glucose	Plasma glucose concentration
BloodPressure	Diastolic blood pressure (mm Hg)
SkinThickness	Triceps skinfold thickness (mm)
Insulin	Serum insulin (mu U/ml)
BMI	Body Mass Index
DiabetesPedigreeFunction	Diabetes heredity score
Age	Age in years
Outcome	0 = No diabetes, 1 = Diabetes

Data Preprocessing

- Loaded data with pandas
- Checked for invalid/missing values
- Replaced 0 values in BMI and BloodPressure with median values
- Split features (X) and target (y)
- Standardized features using StandardScaler

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Exploratory Data Analysis (EDA)

- **Histogram of Age & BMI** → Most patients were 20–40 years old, BMI skewed toward overweight range.

- **Outcome Distribution:**
- **0 (No Diabetes):** 500+ patients
- **1 (Diabetes):** ~260 patients

Visuals made patterns clearer — more non-diabetic patients, which could affect model balance.

Model Training

- **Split Data:** 70% training / 30% testing
- **Algorithm Used:** Logistic Regression
- **Code:**

  ```
  from sklearn.linear_model import LogisticRegression
  model = LogisticRegression()
  model.fit(X_train_scaled, y_train)
  y_pred = model.predict(X_test_scaled)
  ```

Confusion Matrix

Generated confusion matrix to evaluate predictions:

	Predicted No	Predicted Yes
Actual No	TN	FP
Actual Yes	FN	TP

- **TP (True Positive):** Diabetic patient correctly classified
- **TN (True Negative):** Healthy patient correctly classified
- **FP (False Positive):** Healthy patient predicted as diabetic (false alarm)
- **FN (False Negative):** Diabetic patient predicted as healthy (dangerous miss

Key Insights

- Logistic Regression is a good **baseline model** for medical classification tasks.
- False negatives are more dangerous than false positives in healthcare — we must minimize FN even if it slightly increases FP.
- Scaling features improved accuracy.

Reflection

- **Easy Part:** Loading data, running EDA, training Logistic Regression.
- **Challenging Part:** Interpreting confusion matrix & understanding FN impact.
- **Learning:** Importance of preprocessing and balanced evaluation metrics

Future Improvements

- Apply **Random Forest** or **XGBoost** for higher accuracy
- Tune hyperparameters for better model performance
- Deploy model as a **Flask/Django web app**