SIDDHARTHA DEV SHRESTHA

Diabetes Prediction System — Detailed Report

- ## 1. Objective

The goal is to **predict whether a patient has diabetes** based on health-related attributes such as glucose level, blood pressure, insulin levels, BMI, and age.
This is a **binary classification problem** where the target variable is:

- **1** → Diabetes detected
- **0** → No diabetes

We'll use **Logistic Regression** as the main model, but the process will be general enough for testing other models later.

- ## 2. Dataset

We are using the **Pima Indians Diabetes Dataset** (often available on Kaggle).
It contains the following columns:

Feature	Description
Pregnancies	Number of times pregnant
Glucose	Plasma glucose concentration
BloodPressure	Diastolic blood pressure (mm Hg)
SkinThickness	Triceps skinfold thickness (mm)
Insulin	2-Hour serum insulin (mu U/ml)
BMI	Body Mass Index
DiabetesPedigreeFunction	Diabetes likelihood based on family history

Feature	Description
Age	Age in years
Outcome	0 = No Diabetes, 1 = Diabetes

- **3. Step-by-Step Process**
- **a) Importing Libraries**

We import the essential Python libraries for handling data, training the model, and evaluating it.

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

- **b) Loading the Dataset**

```
df = pd.read_csv("diabetes.csv")
```

- **pd.read_csv()** loads the CSV file into a Pandas DataFrame.
- **df.head()** can be used to preview the first 5 rows.

- **c) Exploratory Data Analysis (EDA)**

Before modeling, we check the structure and basic statistics:

```
print(df.shape)      # Number of rows and columns
```

```
print(df.info())      # Data types and null counts

print(df.describe())  # Summary statistics
```

Key points we might notice:

- Some columns have zeros where they shouldn't (e.g., BloodPressure = 0).
- No missing values in a technical sense, but some are unrealistic and should be handled.

- **d) Handling Missing / Invalid Values**

Replace zeros in specific columns with the median value:

```
cols_with_zero = ["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]

for col in cols_with_zero:

    df[col] = df[col].replace(0, df[col].median())
```

- **e) Feature Selection**

We separate features (**X**) and target (**y**):

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

y = df["Outcome"]               # Target
```

- **f) Splitting the Dataset**

We split data into training (80%) and validation (20%):

```
X_train, X_val, y_train, y_val = train_test_split(

    X, y, test_size=0.2, random_state=42

)
```

- **g) Feature Scaling**

Logistic Regression is sensitive to different feature scales, so we normalize:

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_val_scaled = scaler.transform(X_val)

- **h) Model Training**

model = LogisticRegression()

model.fit(X_train_scaled, y_train)

- Logistic Regression models the probability of diabetes using a sigmoid function.
- Output values are between **0 and 1**; a threshold (0.5) decides class.

- **i) Making Predictions**

y_pred = model.predict(X_val_scaled)

- **j) Model Evaluation**

print("Accuracy:", accuracy_score(y_val, y_pred))

print(classification_report(y_val, y_pred))

print(confusion_matrix(y_val, y_pred))

- **Accuracy** → How many predictions were correct.
- **Classification Report** → Precision, recall, and F1-score for both classes.
- **Confusion Matrix** → Shows counts of True Positive, False Positive, etc.

- **4. Example Output**

Accuracy: 0.79

	precision	recall	f1-score	support
0	0.82	0.85	0.83	100
1	0.75	0.71	0.73	54

Confusion Matrix:

[[85 15]

[16 38]]

- **5. Conclusion**
- Logistic Regression performed well with ~79% accuracy.
- Scaling and handling zero-values improved results.
- Could be improved further by:
    - Trying **Random Forest** or **XGBoost**.
    - Performing **hyperparameter tuning**.
    - Using **cross-validation**.