Subject: **Data Science Lab**

**Experiment – 10: To implement Mini Project.**

1. **Aim:** To implement mini project with data science concepts.
2. **Brief Theory:**

   **About Topic:**

   This mini project focuses on Diabetes Prediction using a supervised machine learning approach. The goal is to predict whether a person is likely to have diabetes based on diagnostic measurements. This is a binary classification problem where the outcome is either diabetic (1) or non-diabetic (0).

   **About Dataset:**

   The dataset used is the Pima Indians Diabetes Dataset from Kaggle. It contains 768 records and 9 features, such as:

   - Pregnancies
   - Glucose
   - Blood Pressure
   - Skin Thickness
   - Insulin
   - BMI (Body Mass Index)
   - Diabetes Pedigree Function
   - Age
   - Outcome (Target variable: 0 for non-diabetic, 1 for diabetic)

   This dataset is clean and does not contain missing values.

   **About Algorithm:**

   The algorithm used is Logistic Regression, a statistical method for binary classification. It models the probability that a given input belongs to a particular category. It uses the sigmoid function to output values between 0 and 1, which represent probabilities. Logistic regression is ideal for medical classification problems due to its interpretability and efficiency.

**8. Post-Experiments Exercise**

**B. Questions:**

Applications of the case study you have implemented:

1. Early Diagnosis: Helps in the early prediction of diabetes, allowing for preventive care and timely medical intervention.
2. Healthcare Decision Support: Assists doctors and healthcare providers in making data-driven decisions.
3. Remote Monitoring: Can be integrated into mobile health apps for real-time patient monitoring and self-diagnosis.
4. Health Insurance Risk Assessment: Insurance companies can assess diabetes risk and customize premium plans.
5. Research & Public Health: Enables researchers to understand trends and risk factors in diabetic populations.

**C. Conclusion:**

**Significance of the topic studied in the experiment**

This mini project gave me a real sense of how data science can actually make a difference in people's lives. By using machine learning to predict the chances of someone having diabetes, we're not just working with numbers—we're helping make early detection possible, which can lead to better treatment and health outcomes. It also showed me how important good data and proper analysis are when building something meaningful. Through this project, I learned how to clean and prepare data, train a model, and interpret the results, which are all essential skills for real-world data science problems. Most importantly, it reminded me that behind every prediction is a real person who could benefit from it.

## Importing Libraries

In [ ]:

```
import pandas as pd
import numpy as np
import pickle
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
```

## Importing Dataset

**Dataset Link - Kaggle Diabetes Dataset**

In [ ]:

```
diabetes_dataframe = pd.read_csv('diabetes.csv')
```

## Data Preparation

### Statistcial Measure of Dataset

In [ ]:

```
diabetes_dataframe.describe()
```

Out[ ]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | Diabet |
|---|---|---|---|---|---|---|---|
| count | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | |
| mean | 3.845052 | 120.894531 | 69.105469 | 20.536458 | 79.799479 | 31.992578 | |
| std | 3.369578 | 31.972618 | 19.355807 | 15.952218 | 115.244002 | 7.884160 | |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | |
| 25% | 1.000000 | 99.000000 | 62.000000 | 0.000000 | 0.000000 | 27.300000 | |
| 50% | 3.000000 | 117.000000 | 72.000000 | 23.000000 | 30.500000 | 32.000000 | |
| 75% | 6.000000 | 140.250000 | 80.000000 | 32.000000 | 127.250000 | 36.600000 | |
| max | 17.000000 | 199.000000 | 122.000000 | 99.000000 | 846.000000 | 67.100000 | |

### Details about Dataset

In [ ]:

```
diabetes_dataframe.shape
```

Out[ ]:

```
(768, 9)
```

In [ ]:

```
diabetes_dataframe.columns
```

Out[ ]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

"We have data for 768 patients, encompassing 9 features: 'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', and 'Outcome'."

## Cleaning Dataset

In [ ]:

```
diabetes_dataframe.isnull().sum()
```

Out[ ]:

| | 0 |
|---|---|
| **Pregnancies** | 0 |
| **Glucose** | 0 |
| **BloodPressure** | 0 |
| **SkinThickness** | 0 |
| **Insulin** | 0 |
| **BMI** | 0 |
| **DiabetesPedigreeFunction** | 0 |
| **Age** | 0 |
| **Outcome** | 0 |

**dtype:** int64

The dataset is clean, as no missing values were detected.

## Outcome Insights

In [ ]:

```
diabetes_dataframe['Outcome'].value_counts()
```

Out[ ]:

| Outcome | count |
|---|---|
| 0 | 500 |
| 1 | 268 |

**dtype:** int64

Out of 768 patient, 500 patient are non diabetic and 268 are diabetic

In [ ]:

```
diabetes_dataframe.groupby('Outcome').mean()
```

Out[ ]:

| Outcome | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesI |
|---|---|---|---|---|---|---|---|
| 0 | 3.298000 | 109.980000 | 68.184000 | 19.664000 | 68.792000 | 30.304200 | |
| 1 | 4.865672 | 141.257463 | 70.824627 | 22.164179 | 100.335821 | 35.142537 | |

Patients with diabetes (Outcome=1) have higher average values for features such as Glucose, Insulin, and BMI compared to patients without diabetes (Outcome=0).

In [ ]:

```
Data = diabetes_dataframe.drop(columns=['Outcome','SkinThickness'], axis =1)
Outcome = diabetes_dataframe['Outcome']
```

In [ ]:

```
Data
```

Out[ ]:

| | Pregnancies | Glucose | BloodPressure | Insulin | BMI | DiabetesPedigreeFunction | Age |
|---|---|---|---|---|---|---|---|
| 0 | 6 | 148 | 72 | 0 | 33.6 | 0.627 | 50 |
| 1 | 1 | 85 | 66 | 0 | 26.6 | 0.351 | 31 |
| 2 | 8 | 183 | 64 | 0 | 23.3 | 0.672 | 32 |
| 3 | 1 | 89 | 66 | 94 | 28.1 | 0.167 | 21 |
| 4 | 0 | 137 | 40 | 168 | 43.1 | 2.288 | 33 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 763 | 10 | 101 | 76 | 180 | 32.9 | 0.171 | 63 |
| 764 | 2 | 122 | 70 | 0 | 36.8 | 0.340 | 27 |
| 765 | 5 | 121 | 72 | 112 | 26.2 | 0.245 | 30 |
| 766 | 1 | 126 | 60 | 0 | 30.1 | 0.349 | 47 |
| 767 | 1 | 93 | 70 | 0 | 30.4 | 0.315 | 23 |

768 rows × 7 columns

In [ ]:

```
Outcome
```

Out[ ]:

| | Outcome |
|---|---|
| **0** | 1 |
| **1** | 0 |
| **2** | 1 |
| **3** | 0 |
| **4** | 1 |
| **...** | ... |
| **763** | 0 |
| **764** | 0 |
| **765** | 0 |
| **766** | 1 |
| **767** | 0 |

768 rows × 1 columns

**dtype:** int64

## Data Standardization

In [ ]:

```
scaler = StandardScaler()
scaler.fit(Data)

Data= scaler.transform(Data)
```

In [ ]:

```
Data
```

Out[ ]:

```
array([[ 0.63994726,  0.84832379,  0.14964075, ...,  0.20401277,
         0.46849198,  1.4259954 ],
       [-0.84488505, -1.12339636, -0.16054575, ..., -0.68442195,
        -0.36506078, -0.19067191],
       [ 1.23388019,  1.94372388, -0.26394125, ..., -1.10325546,
         0.60439732, -0.10558415],
       ...,
       [ 0.3429808 ,  0.00330087,  0.14964075, ..., -0.73518964,
        -0.68519336, -0.27575966],
       [-0.84488505,  0.1597866 , -0.47073225, ..., -0.24020459,
        -0.37110101,  1.17073215],
       [-0.84488505, -0.8730192 ,  0.04624525, ..., -0.20212881,
        -0.47378505, -0.87137393]])
```

## Data Splitting

In [ ]:

```
Data_train, Data_test, Outcome_train , Outcome_test = train_test_split(Data, Outcome, test_size = 0.1, str
```

## Training Logistic Model

In [ ]:

```
model = LogisticRegression()
model.fit(Data_train, Outcome_train)
```

Out[ ]:


## Model Evaluation

In [ ]:

```
accuracy = model.score(Data_test, Outcome_test)
print("Testing Data Accuracy is :",accuracy)
```

```
Testing Data Accuracy is : 0.7662337662337663
```

## Testing the Logistic Model

In [ ]:

```
input_df = pd.DataFrame([[0, 150, 72, 0, 33.6, 0.627, 18]],
                        columns=['Pregnancies', 'Glucose', 'BloodPressure', 'Insulin', 'BMI', 'DiabetesPed

std_data = scaler.transform(input_df)

prediction = model.predict(std_data)
probability = model.predict_proba(std_data)[:, 1]

if probability < 0.2:
    print("Probability of Having Diabetes is Very Low")
elif probability < 0.4:
    print("LowProbability of Having Diabetes is Low")
elif probability < 0.6:
    print( "Probability of Having Diabetes is Moderate")
elif probability < 0.8:
    print("Probability of Having Diabetes is High")
else:
    print("Probability of Having Diabetes is Very High")
```

```
Probability of Having Diabetes is Moderate
```