# DIABETES PREDICTION USING SUPPORT VECTOR MACHINE

## Importing Libraries

In [33]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

## Data Collection and Analysis

- PMA Diabetes Dataset

In [34]:
```
# loading diabetes dataset to a pandas dataframe
diabetes_dataset=pd.read_csv(r"C:\Users\91766\Downloads\diabetes.csv")
```

In [35]:
```
# Printing the first 5 rows of the dataset
diabetes_dataset.head()
```

Out[35]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction |
|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 |

In [36]:
```
# Number of rows and column in dataframe
diabetes_dataset.shape
```

Out[36]:
```
(768, 9)
```

In [37]:
```
# Getting statistical Measures of data
diabetes_dataset.describe()
```

Out[37]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | Diabete |
|---|---|---|---|---|---|---|---|
| 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 | |

In [40]:
```
diabetes_dataset['Outcome'].value_counts()
```

Out[40]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

- 0 represents Non Diabetic patients
- 1 represent Diabetic patients

In [41]:
```
diabetes_dataset.groupby('Outcome').mean()
```

Out[41]:

| Outcome | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | Diabe |
|---|---|---|---|---|---|---|---|
| 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 | |

- after grouping we can say from the result that patient having Age upto 31 year are mostly non Diabetic and 37 and above age are mostly diabetic patients

- Glucose level,Blood pressure,skineThickness,Insulin,BMI,DPF are found high in diabetic patients

## Separating the data and labels

In [42]:
```
x=diabetes_dataset.drop(columns='Outcome', axis=1)
y=diabetes_dataset['Outcome']
```

In [43]:
```
print(x)
print(y)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        ...  ...
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23

[768 rows x 8 columns]
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

## Data standardiazation

In [44]:
```
scaler=StandardScaler()
scaler.fit(x)
```

Out[44]:
```
StandardScaler()
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [45]:
```
standardized_data=scaler.transform(x)
standardized_data
```

Out[45]:
```
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]])
```

In [46]:
```
x=standardized_data
y=diabetes_dataset['Outcome']
```

In [47]:
```
print(x)
print(y)
```

```
[[ 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]]
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

## Train Test Split

In [48]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,stratify=y,random
```

In [49]:
```
print(x.shape)
print(xtrain.shape)
print(xtest.shape)
```

```
(768, 8)
(614, 8)
(154, 8)
```

## Training the Model

In [50]:
```
classifier=svm.SVC(kernel='linear')
```

In [51]:
```
# training the support vector machine classifier
classifier.fit(xtrain,ytrain)
```

Out[51]:
```
SVC(kernel='linear')
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

## Model Evaluation

- Accuracy Score

In [52]:
```
# Accuracy score on training data
xtrain_prediction=classifier.predict(xtrain)
training_data_accuracy=accuracy_score(xtrain_prediction,ytrain)
training_data_accuracy
```

Out[52]:
```
0.7866449511400652
```

In [53]:
```
print('Accuracy score of the training data :',training_data_accuracy)
```

```
Accuracy score of the training data : 0.7866449511400652
```

In [54]:
```
# Accuracy score on testing data
xtest_prediction=classifier.predict(xtest)
testing_data_accuracy=accuracy_score(xtest_prediction,ytest)
testing_data_accuracy
```

Out[54]:
```
0.7727272727272727
```

In [55]:
```
print('Accuracy score of the testing data :',testing_data_accuracy)
```

```
Accuracy score of the testing data : 0.7727272727272727
```

## Making a Diabetes Predictive System

In [23]:
```
input_data = (
    int(input('What is the pregnancy week of the patient (0 to 40): ')),
    int(input('What is the glucose value/level of the patient in mg (80-170):
    int(input('What is the blood pressure level in mm Hg (60-80): ')),
    int(input('What is the skin thickness of the patient in mm (2-40): ')),
    int(input('What is the insulin level of the patient in pmol/L (0-1153): ')
    float(input('What is the BMI level in kg/m2 (0-100): ')),
    float(input('What is the Diabetes Pedigree Function level (0-5): ')),
    int(input('What is the age of the patient in years (0-100): '))
)

# changing the Input data as numpy array
input_data_as_numpy_array=np.asarray(input_data)

# Reshape the array as we are predicting for one instance
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)

# standardized the input data
std_data=scaler.transform(input_data_reshaped)

prediction=classifier.predict(std_data)
print(prediction)

if prediction==0:
    print('patient is not Diabetic')
else:
    print('Patient is Diabetic')
```

```
What is the pregnancy week of the patient (0 to 40): 0
What is the glucose value/level of the patient in mg (80-170): 85
What is the blood pressure level in mm Hg (60-80): 70
What is the skin thickness of the patient in mm (2-40): 3
What is the insulin level of the patient in pmol/L (0-1153): 250
What is the BMI level in kg/m2 (0-100): 42
What is the Diabetes Pedigree Function level (0-5): 1.12
What is the age of the patient in years (0-100): 30
[0]
patient is not Diabetic

C:\Users\91766\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\base.py:439: UserWarning: X does not have valid feature names, but Standa
rdScaler was fitted with feature names
  warnings.warn(
```

**Congratulation Patient is not diabetic.....**