# Project Overview

A machine learning-based system for predicting diabetes risk using clinical health indicators. This project leverages on Supervised Machine Learning algorithms to provide accurate predictions and help in early detection of diabetes, supporting healthcare professionals in making informed decisions.

Industry: Healthcare / Medical Technology

## IMPORT DEPENDENCIES

```
# import the required libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
```

```
Start coding or generate with AI.
```

## DATA COLLECTION AND DATA PROCESSING

```
# Load the dataset to a pandas dataframe
diabetes_prediction = pd.read_csv('diabetes.csv')
```

```
# Check the first 5 rows of the dataset
diabetes_prediction.head()
```

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFu |
|---|---|---|---|---|---|---|---|
| **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 | |

```
Start coding or generate with AI.
```

```
# Check the shape of the dataset
diabetes_prediction.shape
```

```
(768, 9)
```

```
Start coding or generate with AI.
```

```
# Check the Statiscal measures of the dataset
diabetes_prediction.describe()
```

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | Di |
|---|---|---|---|---|---|---|---|
| **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 | |

```
Start coding or generate with AI.
```

```
# Check the Value counts of the label column
diabetes_prediction['Outcome'].value_counts()
```

```
Outcome
0    500
1    268
Name: count, dtype: int64
```

1 ---> Diabetes

0 ---> Non-Diabetes

```
Start coding or generate with AI.
```

```
# Check for missing/null values in the dataset
diabetes_prediction.isnull().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
```

```
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Start coding or generate with AI.

```
# Check the mean value of the 'label' column
diabetes_prediction.groupby('Outcome').mean()
```

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

Start coding or generate with AI.

## Seperate The Data (Features) and The Label (Outcome)

```
# Seperate the Features and the Label
x = diabetes_prediction.drop(columns= 'Outcome', axis = 1)
y = diabetes_prediction['Outcome']
```

Start coding or generate with AI.

## DATA STANDARDIZATION

- I am standardizing the data because the distribution of values in the dataset are not in a common range
- Standardizing the dataset puts the values in a common range and helps to get better prediction

```
# Standardizing the dataset
Scaler = StandardScaler()
```

```
Scaler.fit(x)
```

```
StandardScaler()
```

Start coding or generate with AI.

```
# Transform the standardized dataset
Standardized_data = Scaler.transform(x)
```

```
print(Standardized_data)
```

```
[[ 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]]
```

Start coding or generate with AI.

```
# Fit the standardized data and label to (x) and (y)
x = Standardized_data
y = diabetes_prediction['Outcome']
```

Start coding or generate with AI.

## Spliting The Data Into 'Training' And 'Test' Data

```
# create four variables and split the dataset into training and test data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, stratify=y, r
```

```
print(x.shape, x_train.shape, x_test.shape)
```

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

Start coding or generate with AI.

Start coding or generate with AI.

## Training the model ---> I am using the Support Vector Machine model

- SVM is a Classifier

```
# Put the Support Vector Machine Classifier into a new variable
classifier = svm.SVC(kernel = 'linear')
```

```
# Training the data with the Support Vector Machine model
classifier.fit(x_train, y_train)
```

```
SVC
SVC(kernel='linear')
```

Start coding or generate with AI.

Start coding or generate with AI.

## MODEL EVALUATION USING THE ACCURACY SCORE

- Accuracy score is used to evaluate our model in other to find how well our model is perfoming and how many good predictions it is making.

```
# Check the accuracy score of the trained data
x_train_prediction = classifier.predict(x_train)
training_data_accuracy = accuracy_score(x_train_prediction, y_train)
```

```
print('Accuracy Score on Training Data : ', training_data_accuracy)
```

```
Accuracy Score on Training Data :  0.7866449511400652
```

The Accuracy Score for the Train data is about 83%.

This is a very good score.

Start coding or generate with AI.

```
# Check the accuracy score of the test data
x_test_prediction = classifier.predict(x_test)
test_data_accuracy = accuracy_score(x_test_prediction, y_test)
```

```
print('Accuracy Score on Test Data : ', test_data_accuracy)
```

```
Accuracy Score on Test Data :  0.7727272727272727
```

The Accuracy Score for the Train data is about 83%.

This is a very good score.

Start coding or generate with AI.

```
Start coding or generate with AI.
```

## I build a Predictive System that predict whether a patient is 'Diabetes' or 'Non-Diabetes'

```
input_data = (7,181,84,21,192,35.9,0.586,51)

# Changing the input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

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

# Standardize the Input Data
std_data = Scaler.transform(input_data_reshaped)
print(std_data)

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

if (prediction[0]==0):
    print('The Patient Is Not Diabetic')
else:
    print('The Patient Is Diabetic')
```

```
[[0.93691372 1.88112959 0.77001375 0.02907707 0.97422544 0.49592704
  0.34466711 1.51108316]]
[1]
The Patient Is Diabetic
C:\Users\Admin\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not
  warnings.warn(
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```