# Diabetes Prediction using Machine Learning

Diabetes, is a group of metabolic disorders in which there are high blood sugar levels over a prolonged period. Symptoms of high blood sugar include frequent urination, increased thirst, and increased hunger. If left untreated, diabetes can cause many complications. Acute complications can include diabetic ketoacidosis, hyperosmolar hyperglycemic state, or death. Serious long-term complications include cardiovascular disease, stroke, chronic kidney disease, foot ulcers, and damage to the eyes.

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

## Objective

We will try to build a machine learning model to accurately predict whether or not the patients in the dataset have diabetes or not?

## ****Details about the dataset:****

The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

* **Pregnancies**: Number of times pregnant
* **Glucose**: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
* **BloodPressure**: Diastolic blood pressure (mm Hg)
* **SkinThickness**: Triceps skin fold thickness (mm)
* **Insulin**: 2-Hour serum insulin (mu U/ml)
* **BMI**: Body mass index (weight in kg/(height in m)^2)
* **DiabetesPedigreeFunction**: Diabetes pedigree function
* **Age**: Age (years)
* **Outcome**: Class variable (0 or 1)

# Importing the necessary Libraries

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

# Import the requried dataset

dataset**=**pd**.**read\_csv('C:/Users/GPT BANTWAL/Documents/diabetes.csv')

dataset

| **Pregnancies** | **Glucose** | **BloodPressure** | **SkinThickness** | **Insulin** | **BMI** | **DiabetesPedigreeFunction** | **Age** | **Outcome** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 4 | 183 | 0 | 0 | 0 | 28.4 | 0.212 | 36 | 1 |
| **1** | 5 | 162 | 104 | 0 | 0 | 37.7 | 0.151 | 52 | 1 |
| **2** | 2 | 197 | 70 | 99 | 0 | 34.7 | 0.575 | 62 | 1 |
| **3** | 13 | 158 | 114 | 0 | 0 | 42.3 | 0.257 | 44 | 1 |
| **4** | 0 | 162 | 76 | 56 | 100 | 53.2 | 0.759 | 25 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **503** | 7 | 179 | 95 | 31 | 0 | 34.2 | 0.164 | 60 | 0 |
| **504** | 0 | 113 | 76 | 0 | 0 | 33.3 | 0.278 | 23 | 1 |
| **505** | 3 | 128 | 72 | 25 | 190 | 32.4 | 0.549 | 27 | 1 |
| **506** | 1 | 119 | 88 | 41 | 170 | 45.3 | 0.507 | 26 | 0 |
| **507** | 3 | 84 | 72 | 32 | 0 | 37.2 | 0.267 | 28 | 0 |

508 rows × 9 columns

dataset**.**info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 508 entries, 0 to 507

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Pregnancies 508 non-null int64

1 Glucose 508 non-null int64

2 BloodPressure 508 non-null int64

3 SkinThickness 508 non-null int64

4 Insulin 508 non-null int64

5 BMI 508 non-null float64

6 DiabetesPedigreeFunction 508 non-null float64

7 Age 508 non-null int64

8 Outcome 508 non-null int64

dtypes: float64(2), int64(7)

memory usage: 35.8 KB

dataset**.**isnull()**.**sum()

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

x**=**dataset**.**drop(["Outcome"],axis**=**1)

x

| **Pregnancies** | **Glucose** | **Blood Pressure** | **Skin Thickness** | **Insulin** | **BMI** | **Diabetes Pedigree Function** | **Age** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 4 | 183 | 0 | 0 | 0 | 28.4 | 0.212 | 36 |
| **1** | 5 | 162 | 104 | 0 | 0 | 37.7 | 0.151 | 52 |
| **2** | 2 | 197 | 70 | 99 | 0 | 34.7 | 0.575 | 62 |
| **3** | 13 | 158 | 114 | 0 | 0 | 42.3 | 0.257 | 44 |
| **4** | 0 | 162 | 76 | 56 | 100 | 53.2 | 0.759 | 25 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **503** | 7 | 179 | 95 | 31 | 0 | 34.2 | 0.164 | 60 |
| **504** | 0 | 113 | 76 | 0 | 0 | 33.3 | 0.278 | 23 |
| **505** | 3 | 128 | 72 | 25 | 190 | 32.4 | 0.549 | 27 |
| **506** | 1 | 119 | 88 | 41 | 170 | 45.3 | 0.507 | 26 |
| **507** | 3 | 84 | 72 | 32 | 0 | 37.2 | 0.267 | 28 |

508 rows × 8 columns

y**=**dataset['Outcome']

y

0 1

1 1

2 1

3 1

4 1

..

503 0

504 1

505 1

506 0

507 0

Name: Outcome, Length: 508, dtype: int64

# Split the dataset

**from** sklearn.model\_selection **import** train\_test\_split

xtrain,xtest,ytrain,ytest**=**train\_test\_split(x,y,test\_size**=**0.2,random\_state=100)

print("xtrain:", xtrain.shape)

print("xtest:", xtest.shape)

print("ytrain:", ytrain.shape)

print("ytest:", ytest.shape)

xtrain: (614, 8)

xtest: (154, 8)

ytrain: (614,)

y­\_test: (154,)

Model Build

Decision Tree classifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification\_report,f1\_score

model=DecisionTreeClassifier()

model.fit(xtrain,ytrain)

y\_pred=model.predict(xtest)

Accuracy=classification\_report(ytest,y\_pred)

print(Accuracy)

precision recall f1-score support

0 0.73 0.81 0.77 101

1 0.54 0.42 0.47 53

accuracy 0.68 154

macro avg 0.63 0.61 0.62 154

weighted avg 0.66 0.68 0.66 154

train\_predictions=model.predict(xtrain)

test\_predictions=model.predict(xtest)

train\_accuracy=model.score(xtrain,ytrain)

print("Accuracy of the model on train data=",train\_accuracy)

test\_accuracy=model.score(xtest,ytest)

print("Accuracy of the model on test data=",test\_accuracy)

Accuracy of the model on train data= 1.0

Accuracy of the model on test data= 0.6753246753246753

model1=DecisionTreeClassifier(min\_samples\_split=10,min\_impurity\_decrease=0.005)

model1.fit(xtrain,ytrain)

print("train\_accuracy=",model1.score(xtrain,ytrain))

print("test\_accuracy=",model1.score(xtest,ytest)

train\_accuracy= 0.8550488599348535

test\_accuracy= 0.7142857142857143

Random Forest classifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report,f1\_score

model=RandomForestClassifier()

model.fit(xtrain,ytrain)

y\_pred=model.predict(xtest)

Accuracy=classification\_report(ytest,y\_pred)

print(Accuracy)

precision recall f1-score support

0 0.76 0.84 0.80 101

1 0.62 0.49 0.55 53

accuracy 0.72 154

macro avg 0.69 0.67 0.67 154

weighted avg 0.71 0.72 0.71 154

train\_predictions=model.predict(xtrain)

test\_predictions=model.predict(xtest)

train\_accuracy=model.score(xtrain,ytrain)

print("Accuracy of the model on train data=",train\_accuracy)

test\_accuracy=model.score(xtest,ytest)

print("Accuracy of the model on test data=",test\_accuracy)

Accuracy of the model on train data= 1.0

Accuracy of the model on test data= 0.7207792207792207

model1=DecisionTreeClassifier(min\_samples\_split=10,min\_impurity\_decrease=0.005)

model1.fit(xtrain,ytrain)

print("train\_accuracy=",model1.score(xtrain,ytrain))

print("test\_accuracy=",model1.score(xtest,ytest))

train\_accuracy= 0.8550488599348535

test\_accuracy= 0.7142857142857143

# confusion matrix

from sklearn.metrics import confusion\_matrix

confusion\_matrix(ytest,y\_pred)

array([[64, 11],

[20, 21]], dtype=int64)