Diabeties Predictor Project

using Machine Learning Algorithm

```
# Importing pandas and basic libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
data= pd.read_csv("/content/DataSet/diabetes.csv")
data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigre
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	

Next steps: Generate code with data View recommended plots

Importing Machine Learning Libraries

```
from sklearn.preprocessing import StandardScaler  #Standard Scaler
from sklearn.linear_model  import LogisticRegression #Regression Model
from sklearn.model_selection import train_test_split #Train-Test-Split
from sklearn.metrics import accuracy_score, confusion_matrix #Accuracy parameters
```

```
data.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI
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

Any null values in the set?

```
data.isnull().sum()
```

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

```
Outcome                     0
dtype: int64
```

No null values

There are some data in the columns line `Insulin` which are 0, that is not practically possible. So, we will replace the 0 with the mean value of the columns.

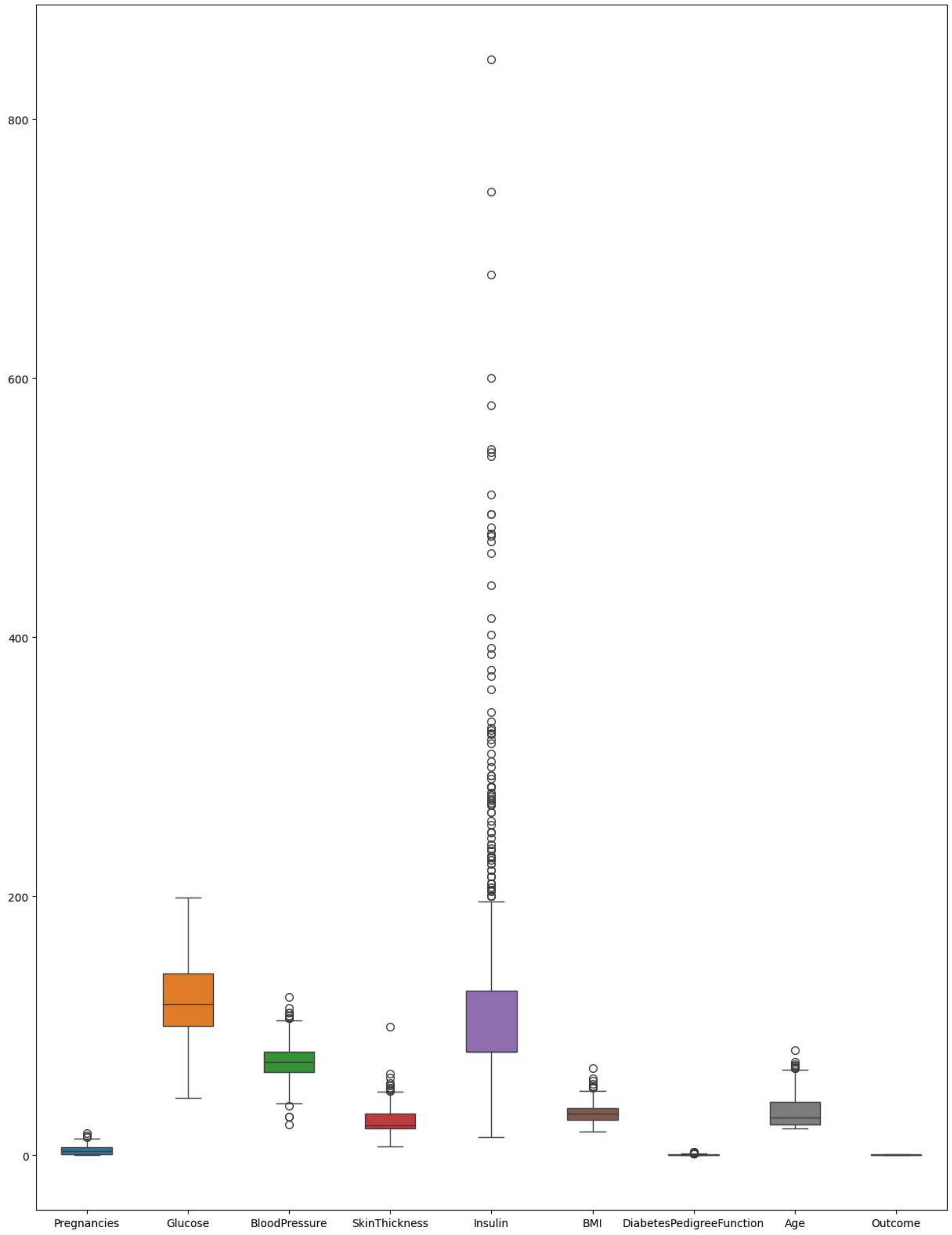
The columns name with 0 values are, `BMI, BloodPressure, Glucose, Insulin, SkinThikness`.

```
data['BMI'] = data['BMI'].replace(0,data['BMI'].mean())
data['BloodPressure'] = data['BloodPressure'].replace(0,data['BloodPressure'].mean())
data['Glucose'] = data['Glucose'].replace(0,data['Glucose'].mean())
data['Insulin'] = data['Insulin'].replace(0,data['Insulin'].mean())
data['SkinThickness'] = data['SkinThickness'].replace(0,data['SkinThickness'].mean())
```

Are there any outliers?

```
fig, ax=plt.subplots(figsize=(15,20))
sns.boxplot(data,width=0.5, ax=ax, fliersize=7)
```

```
<Axes: >
```



Separating Dependent and Independent(Prediction) data from the dataset.

```
x=data.drop(columns=['Outcome'])
y=data['Outcome'] #to be predicted
```

Separating the train, test data -

```
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size=0.25, random_state=0)
```

```
xtrain.shape, xtest.shape
```

```
((576, 8), (192, 8))
```

Scaling the data and pikiling the scaler -

```
import pickle
```

```
def scaler_standard(xtrain,xtest):
  scaler=StandardScaler()
  xtrainscaled=scaler.fit_transform(xtrain)
  xtestscaled=scaler.transform(xtest)
  #save
  file=open("/content/Model/standarscaler.pkl", 'wb')
  pickle.dump(scaler,file)
  file.close()
  return xtrainscaled, xtestscaled
```

```
xtrain_scaled, xtest_scaled=scaler_standard(xtrain,xtest)
```

```
xtrain_scaled
```

```
array([[ 1.50755225, -1.09947934, -0.89942504, ..., -1.45561965,
        -0.98325882, -0.04863985],
       [-0.82986389, -0.1331471 , -1.23618124, ...,  0.09272955,
        -0.62493647, -0.88246592],
       [-1.12204091, -1.03283573,  0.61597784, ..., -0.03629955,
         0.39884168, -0.5489355 ],
       ...,
       [ 0.04666716, -0.93287033, -0.64685789, ..., -1.14021518,
        -0.96519215, -1.04923114],
       [ 2.09190629, -1.23276654,  0.11084355, ..., -0.36604058,
        -0.5075031 ,  0.11812536],
       [ 0.33884418,  0.46664532,  0.78435594, ..., -0.09470985,
         0.51627505,  2.953134  ]])
```

Logistic Regeression Model

```
logreg=LogisticRegression()
logreg.fit(xtrain_scaled, ytrain)
```

```
▾ LogisticRegression
LogisticRegression()
```

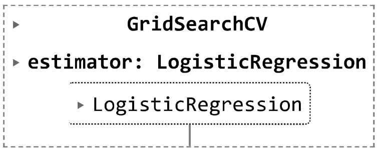
Any Hyperparameter Training?

```
# Grid Search CV
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings('ignore')
```

```
parameters={ 'penalty':['l1','l2'], 'C': np.logspace(-3,3,7), 'solver':['newton-cg', 'lbfgs', 'liblinear'],}
```

```
logreg=LogisticRegression()
```

```
c.fit(xtrain_scaled, ytrain)
```



```
▸ GridSearchCV
▸ estimator: LogisticRegression
    ▸ LogisticRegression
```

```
c.best_params_
```

```
{'C': 1.0, 'penalty': 'l2', 'solver': 'liblinear'}
```

```
c.best_score_
```

```
0.763793103448276
```

Let's do the prediction

```
ypred=c.predict(xtest_scaled)
```

```
accuracy=accuracy_score(ytest,ypred)
print("The accuracy of the model is : ", accuracy)
```

```
The accuracy of the model is :  0.796875
```

```
conmat=confusion_matrix(ytest,ypred)
conmat
```

```
array([[117,  13],
       [ 26,  36]])
```

```
tp = conmat[0][0]
fp = conmat[0][1]
fn = conmat[1][0]
tn = conmat[1][1]
```

```
accu=(tp+tn)/(tp+tn+fp+fn)
print("The accuracy of the model is : ", accu)
```