DATA MINING PROJECT

DIABETES PREDICTION

Team Members :

1) Aditya Karn
2) Rupesh Kumar
3) Harmeet Singh
4) Chirag Prakash
5) Sumit Yadav

About the Project

- The objective of this project is to classify whether someone has diabetes or not.

- Dataset consists of several Medical Predictor Variables (Independent) and one Outcome Variable (Dependent).

- The independent variables in this data set are : 'Pregnancies' , 'Glucose' , 'BloodPressure' , 'SkinThickness' , 'Insulin' ,'BMI' , 'DiabetesPedigreeFunction' , 'Age'.

- The outcome variable value is either 1 or 0 indicating whether a person has diabetes (1) or not (0).

Features

- ***Pregnancies :*** *Number of times a woman has been pregnant.*
- ***Glucose :*** *Plasma Glucose concentration of 2 hours in an oral glucose tolerance test.*
- ***BloodPressure :*** *Diastolic Blood Pressure (in mmHg).*
- ***SkinThickness :*** *Triceps skin fold thickness (in mm).*
- ***Insulin :*** *2 hour serum insulin (in µU/ml).*
- ***BMI :*** *Body Mass Index (weight in kg / (height in m)^2).*
- ***Age :*** *Age (in years).*
- ***DiabetesPedigreeFunction :*** *scores likelihood of diabetes based on family history.*
- ***Outcome :*** *0 (does not have diabetes) or 1 (has diabetes).*

Exploratory Data Analysis

In [4]:
```
df.head()
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [7]:
```
df.dtypes
```

Out[7]:
```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```


Data Cleaning
Data Visualization
Feature Selection
Handling Outliers
Training Algorithms

Data Cleaning

- The first step in data cleaning would involve removing the duplicates. Command used for it:

```
In [10]: df=df.drop_duplicates()
```

- Now we look for NULL/missing values in our dataset. We don't find any.

Command used for it

```
In [11]: df.isnull().sum()

Out[11]: Pregnancies                 0
         Glucose                     0
         BloodPressure               0
         SkinThickness               0
         Insulin                     0
         BMI                         0
         DiabetesPedigreeFunction    0
         Age                         0
         Outcome                     0
         dtype: int64
```

- Although there were no NULL/missing values, but there were some values which were practically not possible. Like 0 BP, 0 BMI etc. Therefore we came to the conclusion that missing values are represented as 0 in the dataset.

```
In [12]: print(df[df['BloodPressure']==0].shape[0])
         print(df[df['Glucose']==0].shape[0])
         print(df[df['SkinThickness']==0].shape[0])
         print(df[df['Insulin']==0].shape[0])
         print(df[df['BMI']==0].shape[0])

         35
         5
         227
         374
         11
```

Measures	Relevant Scale Type
Mean	Interval and ratio data which are not skewed.
Median	Ordinal, interval and ratio but not useful for ordinal scales having few values.
Mode	All scale types but not useful for scales having multiple values.

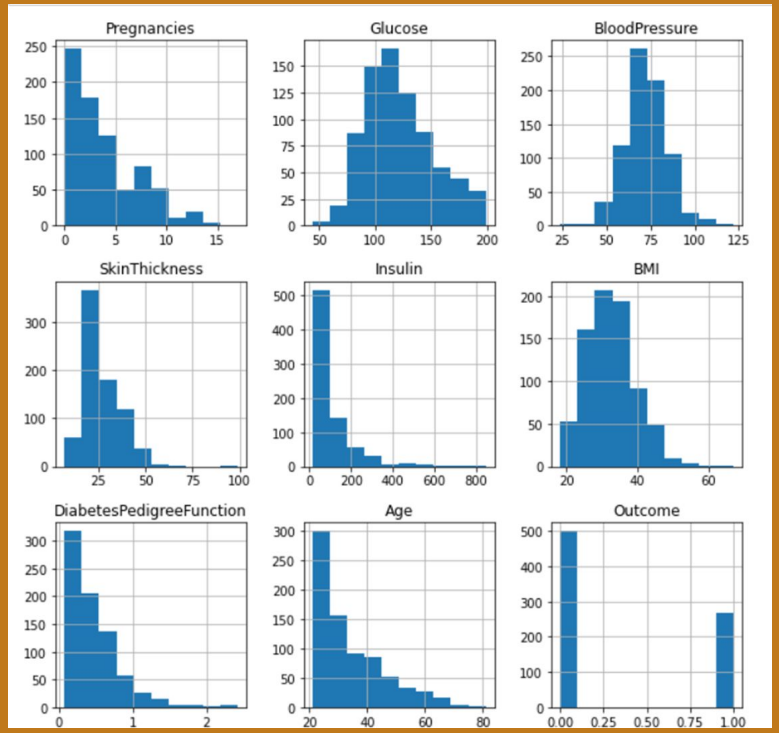
- We filled the missing values according to the table on left.
- When we visualise these attributes, we can distinguish between skewed and normal ones.

```
In [13]: df['Glucose']=df['Glucose'].replace(0,df['Glucose'].mean())
         df['BloodPressure']=df['BloodPressure'].replace(0,df['BloodPressure'].mean())
         df['SkinThickness']=df['SkinThickness'].replace(0,df['SkinThickness'].median())
         df['Insulin']=df['Insulin'].replace(0,df['Insulin'].median())
         df['BMI']=df['BMI'].replace(0,df['BMI'].median())
```

- As we will see in the next slide using data visualisation,

  Glucose and Blood Pressure are normally distributed attributes and hence the use of MEAN to substitute their values and SkinThickness, Insulin and BMI being skewly distributes, hence the use of MEDIAN.

Data Visualisation



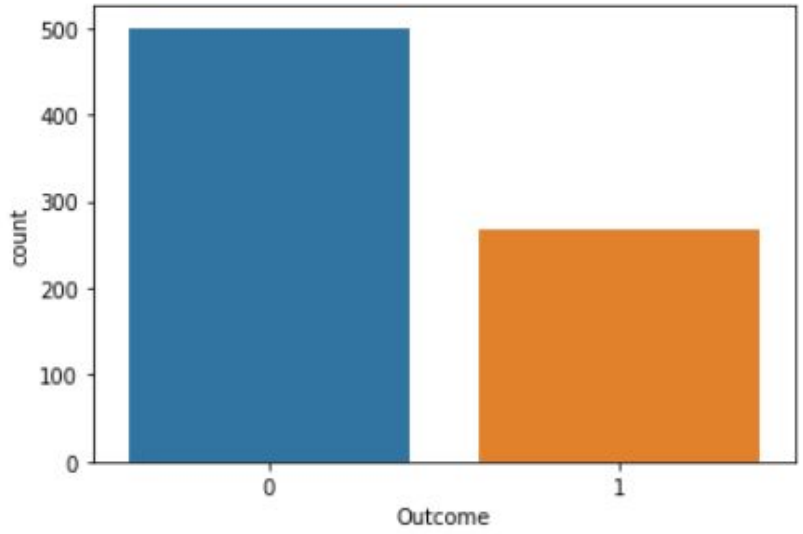
- We plotted **HISTOGRAMS** for all the attributes to fill out the missing values accordingly.

```
In [15]: df.hist(bins=10,figsize=(10,10))
         plt.show()
```

- Also, plotting **COUNTPLOT** to check if our data is balanced or not.

```
In [14]: sns.countplot('Outcome',data=df)
```

```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6c34f1cf90>
```



Feature Selection

Feature Selection is the process of reducing the number of input variables when developing a predictive model.

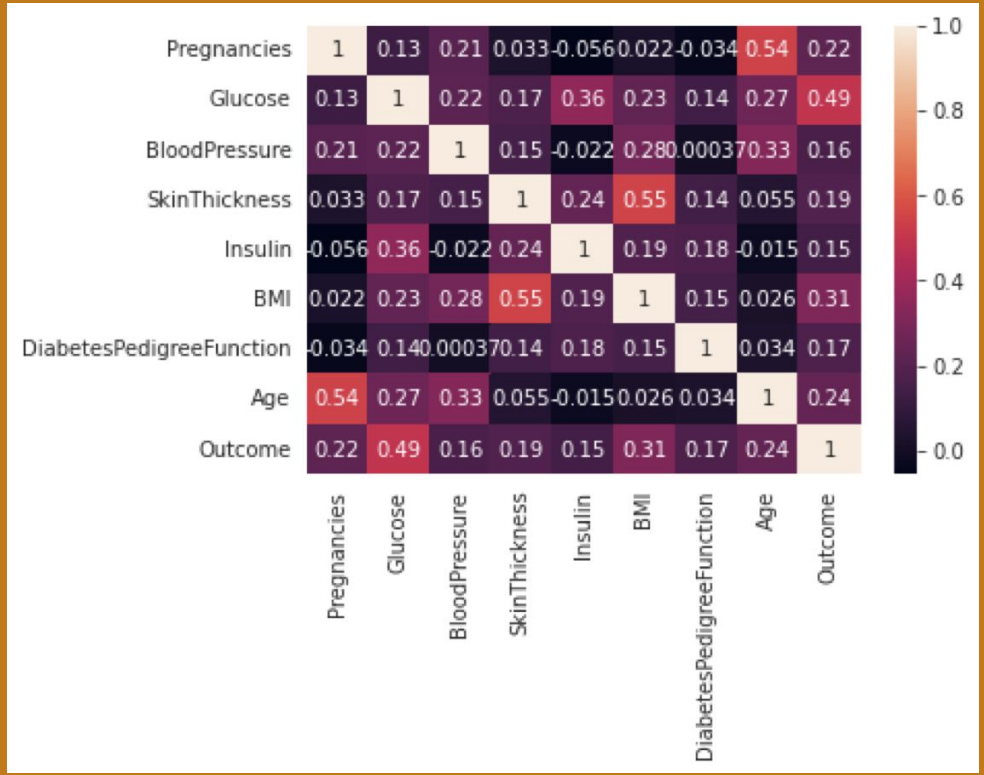
It is desirable to reduce the number of input variables to both reduce the computational cost of modelling and in some cases to improve the performance of the model.

Now we did some analysis on choosing the perfect subset of features of classification.

We did this by using Correlation Matrix via Pearson Correlation Coefficient.

In [17]:
```
corrmat=df.corr()
sns.heatmap(corrmat, annot=True)
```

Out[17]:
```
<AxesSubplot:>
```



In [18]:

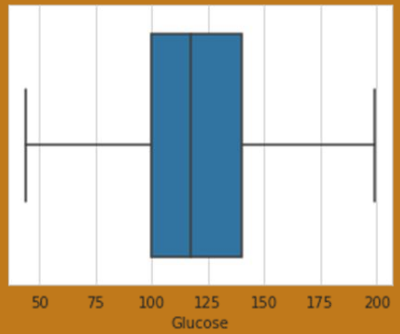
```
df_selected=df.drop(['BloodPressure','Insulin','DiabetesPedigreeFunction'],axis='columns')
```

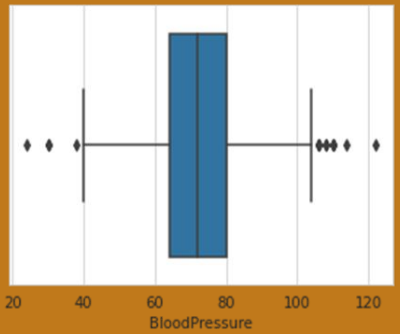
Handling Outliers

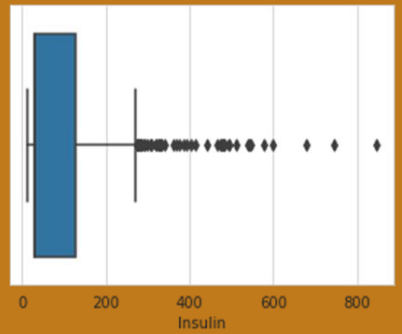
- Some input variables may have a highly skewed distribution, such as an exponential distribution where the most common observations are bunched together. Some input variables may have outliers that cause the distribution to be highly spread.
- These concerns and others, like non-standard distributions and multi-modal distributions, can make a dataset challenging to model with a range of machine learning models.
- We used Boxplots or 5 point summary to visualize this.

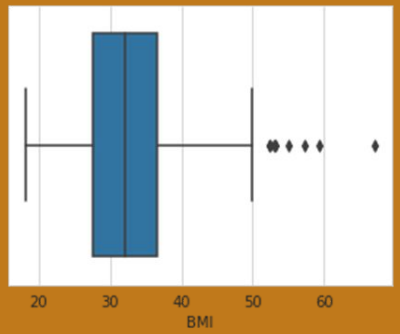
In [15]:

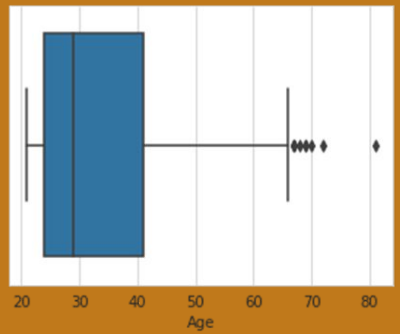
```
plt.figure(figsize=(16,12))
sns.set_style(style='whitegrid')
plt.subplot(3,3,1)
sns.boxplot(x='Glucose',data=df)
plt.subplot(3,3,2)
sns.boxplot(x='BloodPressure',data=df)
plt.subplot(3,3,3)
sns.boxplot(x='Insulin',data=df)
plt.subplot(3,3,4)
sns.boxplot(x='BMI',data=df)
plt.subplot(3,3,5)
sns.boxplot(x='Age',data=df)
plt.subplot(3,3,6)
sns.boxplot(x='SkinThickness',data=df)
plt.subplot(3,3,7)
sns.boxplot(x='Pregnancies',data=df)
plt.subplot(3,3,8)
sns.boxplot(x='DiabetesPedigreeFunction',data=df)
```

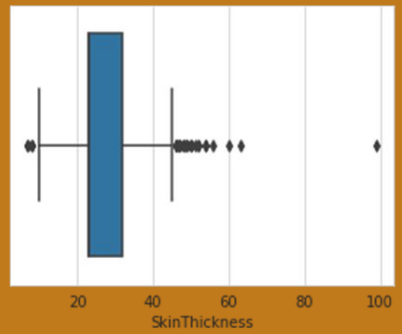

50
75
100
125
150
175
200
Glucose

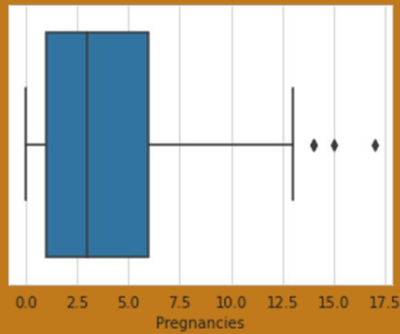

20
40
60
80
100
120
BloodPressure

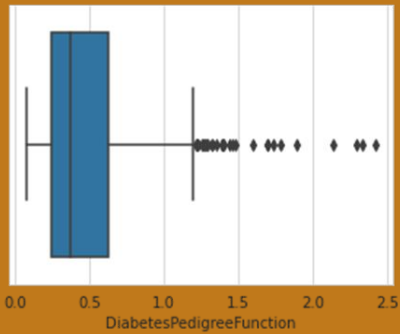

0
200
400
600
800
Insulin


20
30
40
50
60
BMI


20
30
40
50
60
70
80
Age


20
40
60
80
100
SkinThickness


0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Pregnancies


0.0
0.5
1.0
1.5
2.0
2.5
DiabetesPedigreeFunction

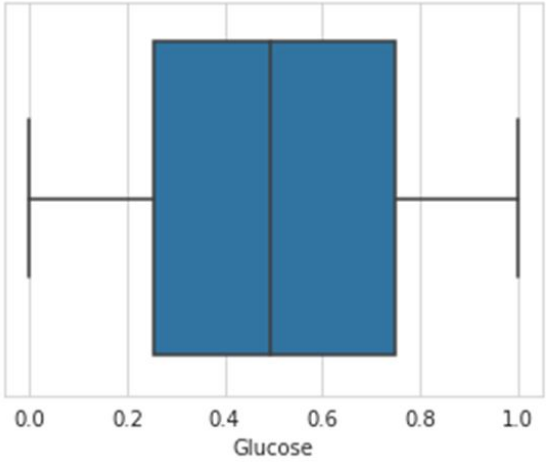
1 — What is an Outlier?

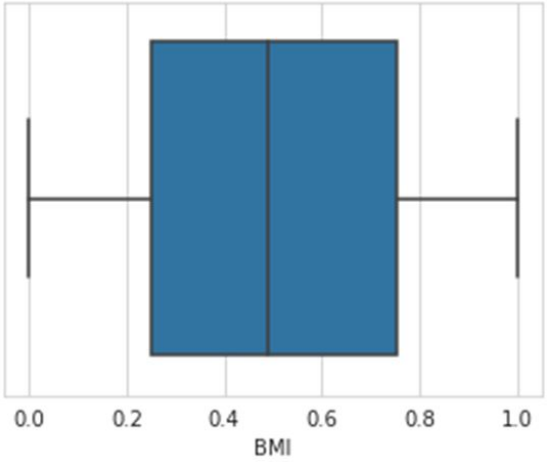
An outlier is a data point in a data set that is distant from all other observations. Outliers are unusual values in your dataset, and they can distort statistical analyses and violate their assumptions. Hence it is of utmost importance to deal with them. In this case removing outliers can cause data loss so we have to deal with it using various scaling and transformation techniques.

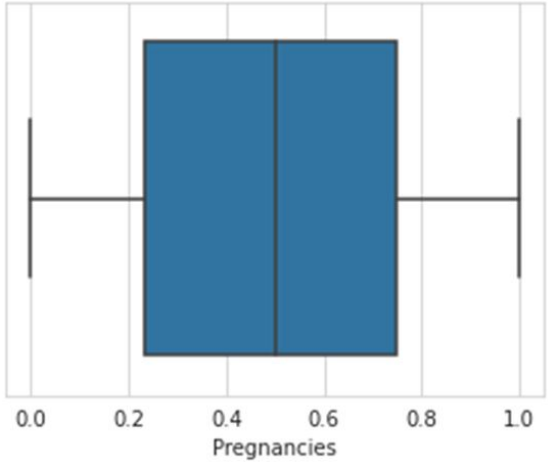
Handling Outliers

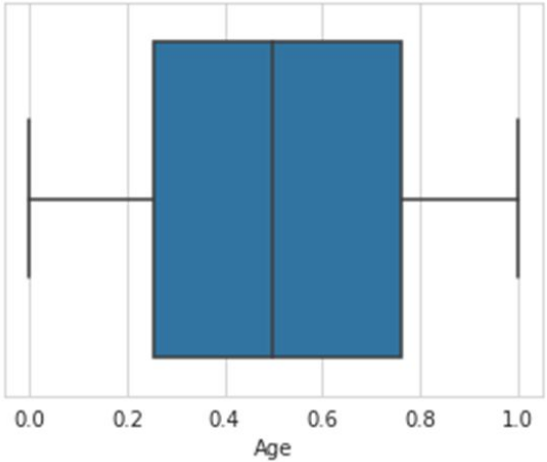
```
from sklearn.preprocessing import QuantileTransformer
x=df_selected
quantile  = QuantileTransformer()
X = quantile.fit_transform(x)
df_new=quantile.transform(X)
df_new=pd.DataFrame(X)
df_new.columns =['Pregnancies', 'Glucose','SkinThickness','BMI','Age','Outcome']
df_new.head()
# df_new.shape
```

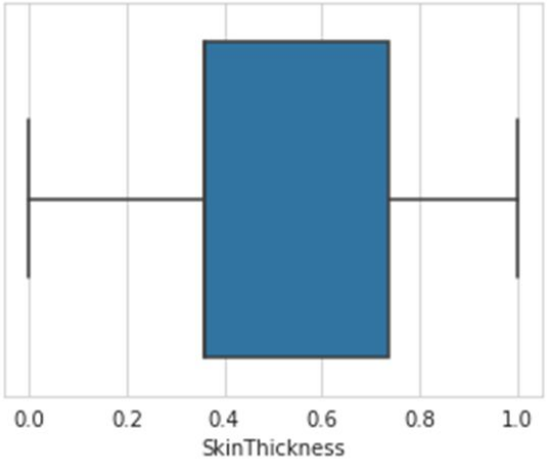
	Pregnancies	Glucose	SkinThickness	BMI	Age	Outcome
0	0.747718	0.810300	0.801825	0.591265	0.889831	1.0
1	0.232725	0.097784	0.644720	0.227510	0.558670	0.0
2	0.863755	0.956975	0.000000	0.091917	0.585398	1.0
3	0.232725	0.131030	0.505867	0.298566	0.000000	0.0
4	0.000000	0.721643	0.801825	0.926988	0.606258	1.0


0.0
0.2
0.4
0.6
0.8
1.0
Glucose


0.0
0.2
0.4
0.6
0.8
1.0
BMI


0.0
0.2
0.4
0.6
0.8
1.0
Pregnancies


0.0
0.2
0.4
0.6
0.8
1.0
Age


0.0
0.2
0.4
0.6
0.8
1.0
SkinThickness

LEARNING ALGORITHMS

Classification algorithms used

1. Decision Tree
    a. ID3
    b. CART

2. Logistic Regression

TRAIN TEST SPLIT

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test= train_test_split(x,y,test_size=0.2,random_state=0)
```

```
X_train.shape,y_train.shape
```

```
((614, 6), (614,))
```

```
X_test.shape,y_test.shape
```

```
((154, 6), (154,))
```

Split the data into

1. Training set (80%)
2. Test set (20%)

Hyperparameter Tuning for Decision Tree

Decision tree

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import f1_score, precision_score, recall_score
from sklearn.model_selection import GridSearchCV
dt = DecisionTreeClassifier(random_state=42)
```

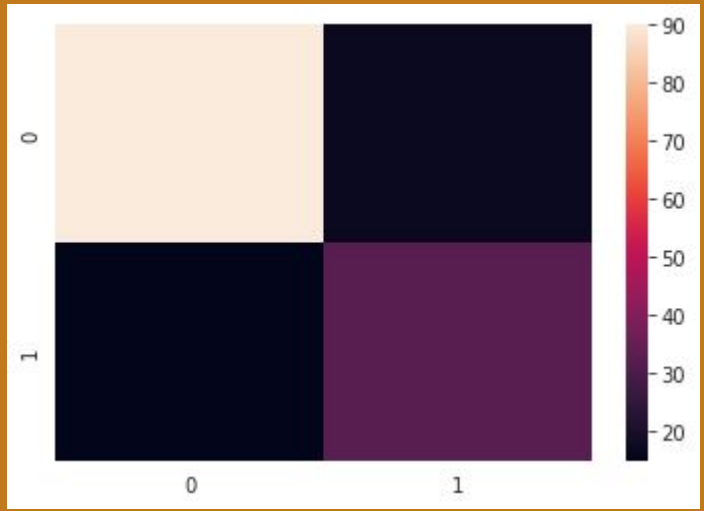
```
params = {
    'max_depth': [5, 10, 20,25],
    'min_samples_leaf': [10, 20, 50, 100,120],
    'criterion': ["gini", "entropy"]
}
```

```
grid_search = GridSearchCV(estimator=dt,
                           param_grid=params,
                           cv=4, n_jobs=-1, verbose=1, scoring = "accuracy")
```

```
best_model=grid_search.fit(X_train, y_train)
```

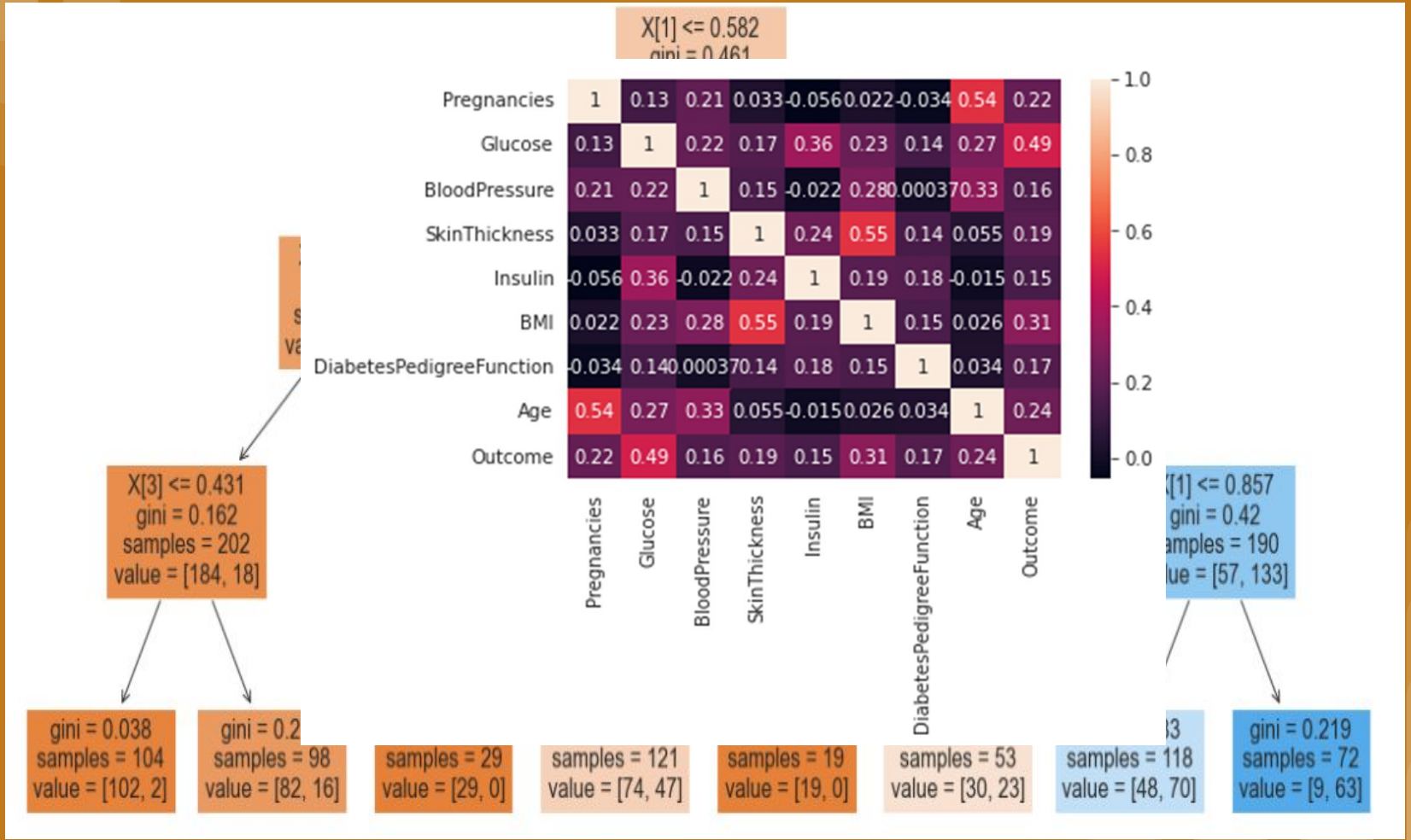
```
Fitting 4 folds for each of 40 candidates, totalling 160 fits
```

```
Classification Report is:
              precision    recall  f1-score   support

         0.0       0.86      0.84      0.85       107
         1.0       0.65      0.68      0.67        47

    accuracy                           0.79       154
   macro avg       0.76      0.76      0.76       154
weighted avg       0.79      0.79      0.79       154


 F1:
 0.6666666666666666

 Precision score is:
 0.6976744186046512

 Recall score is:
 0.6382978723404256

 Confusion Matrix:
```



True Positives >> True Negatives

True Negatives>>> False Positive and Negatives


X[1] <= 0.582
gini = 0.461
Pregnancies 1 0.13 0.21 0.033 -0.056 0.022 -0.034 0.54 0.22
Glucose 0.13 1 0.22 0.17 0.36 0.23 0.14 0.27 0.49
BloodPressure 0.21 0.22 1 0.15 -0.022 0.28 0.00037 0.33 0.16
SkinThickness 0.033 0.17 0.15 1 0.24 0.55 0.14 0.055 0.19
Insulin -0.056 0.36 -0.022 0.24 1 0.19 0.18 -0.015 0.15
BMI 0.022 0.23 0.28 0.55 0.19 1 0.15 0.026 0.31
DiabetesPedigreeFunction -0.034 0.14 0.00037 0.14 0.18 0.15 1 0.034 0.17
Age 0.54 0.27 0.33 0.055 -0.015 0.026 0.034 1 0.24
Outcome 0.22 0.49 0.16 0.19 0.15 0.31 0.17 0.24 1
1.0
0.8
0.6
0.4
0.2
0.0
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome
X[3] <= 0.431
gini = 0.162
samples = 202
value = [184, 18]
[1] <= 0.857
gini = 0.42
amples = 190
ue = [57, 133]
gini = 0.038
samples = 104
value = [102, 2]
gini = 0.2
samples = 98
value = [82, 16]
samples = 29
value = [29, 0]
samples = 121
value = [74, 47]
samples = 19
value = [19, 0]
samples = 53
value = [30, 23]
3
samples = 118
value = [48, 70]
gini = 0.219
samples = 72
value = [9, 63]

LOGISTIC REGRESSION

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import f1_score, precision_score, recall_score,accuracy_score
```

```
reg = LogisticRegression()
reg.fit(X_train,y_train)
```

```
LogisticRegression()
```

```
lr_pred=reg.predict(X_test)
```

```
print("Classification Report is:\n",classification_report(y_test,lr_pred))
print("\n F1:\n",f1_score(y_test,lr_pred))
print("\n Precision score is:\n",precision_score(y_test,lr_pred))
print("\n Recall score is:\n",recall_score(y_test,lr_pred))
print("\n Confusion Matrix:\n")
sns.heatmap(confusion_matrix(y_test,lr_pred))
```

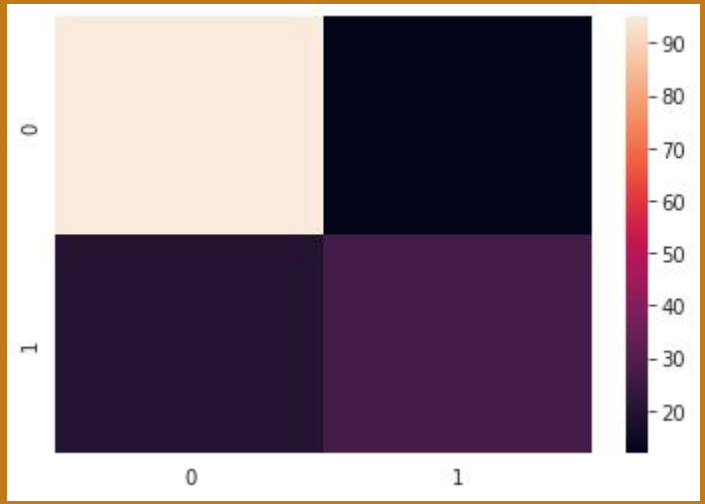
Classification Report is:

	precision	recall	f1-score	support
0.0	0.83	0.89	0.86	107
1.0	0.69	0.57	0.63	47
accuracy			0.79	154
macro avg	0.76	0.73	0.74	154
weighted avg	0.79	0.79	0.79	154

F1:
0.627906976744186

Precision score is:
0.6923076923076923

Recall score is:
0.574468085106383



The count of False Negatives and Positives is very less

INFERENCES

- The Features
  - Glucose
  - BMI

  give the maximum information about the independent variable.

- On the given dataset, Decision Tree with the following hyperparameters has the highest accuracy
  - 20 levels
  - Algorithm : CART
  - Impurity Criterion : GINI index

THANKYOU