v5e6yql99

July 18, 2023

1 PROJECT PROPOSAL

1.0.1 TITLE OF THE PROJECT

Prediction of Diabetes

1.0.2 PROJECT BREIFING

Diabetes is a type of chronic disease which is more common among the of all age groups. Predicting this disease at an early stage can help necessary precautions and change his/her prevent the occurrence of this disease or control the disease(For people who already have the disease).

1.0.3 WHAT TO DO IN THE PROJECT

General approach would be collecting & analysing the data, Feature selection & engineering the data, Outlier detection & treatment, Missing value detection & treatment and finally model building to check training & testing accuracy. Various models will be build like logistics regression, decision tree, random forest, KNN, GNB. Accuracy score shall be compared to select the best model.

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import math
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
from sklearn.preprocessing import OrdinalEncoder, MinMaxScaler, RobustScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import␣
 ↪precision_score,accuracy_score,recall_score,confusion_matrix
from sklearn.metrics import classification_report
from sklearn import metrics
import pickle
```

[3]:
```
POD= pd.read_csv('diabetes.csv')
POD.head()
```

[3]:
```
   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

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

[5]:
```
POD.shape
```

[5]:
```
(768, 9)
```

[6]:
```
POD.columns
```

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

[8]:
```
POD.dtypes
```

[8]:
```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

[9]:
```
POD.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
```

```
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

[10]:
```
POD.isnull().sum()
```

[10]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[23]:
```
POD.describe()
```

[23]:
```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

[13]:
```
POD.nunique
```

[13]:
```
<bound method DataFrame.nunique of      Pregnancies  Glucose  BloodPressure
SkinThickness  Insulin   BMI  \
```

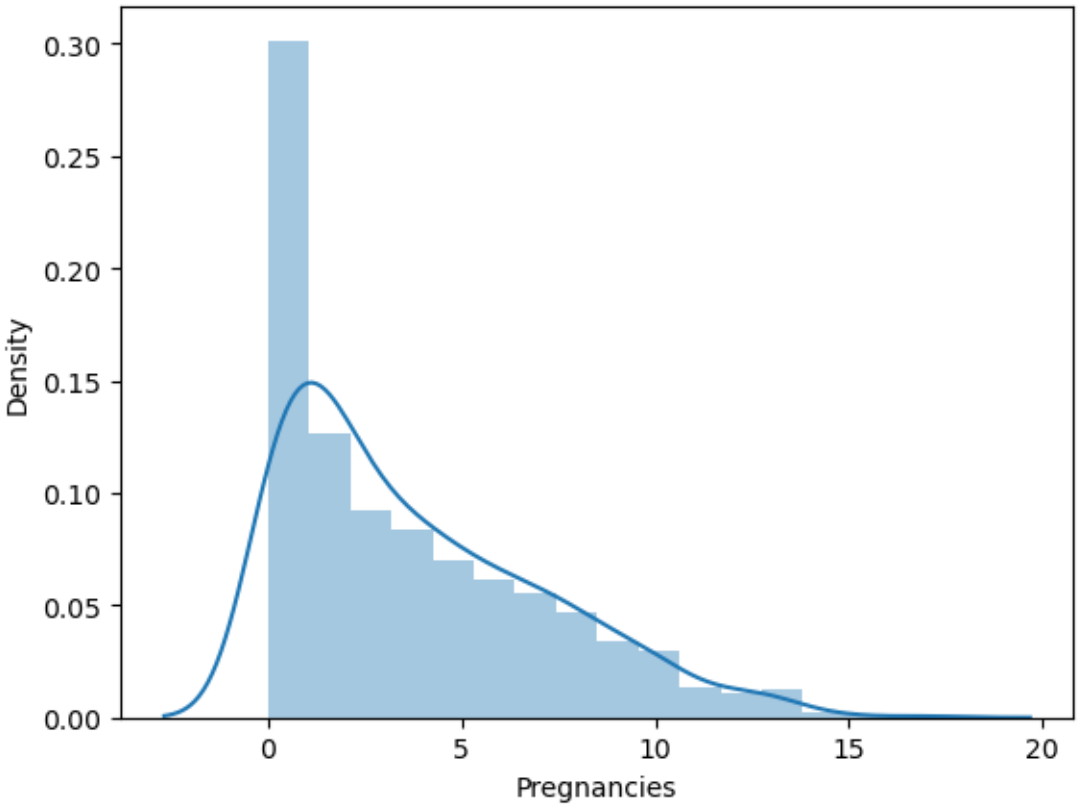
```
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              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  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        …  …      …
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]>
```

1.0.4 ANALYSING THE DATA

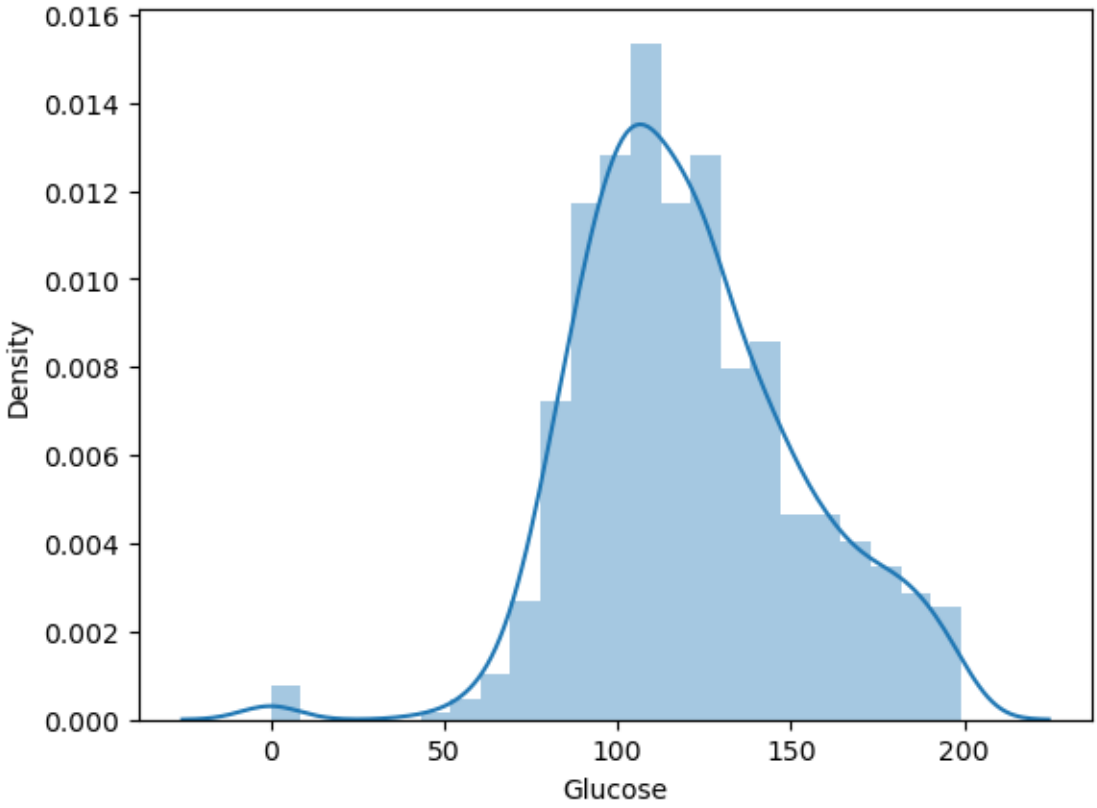
1)There are 768 rows and 9 columns 2)There are no missing values 3)Target variable is "Outcome". The variables for it are 1 and 0

1.0.5 EDA

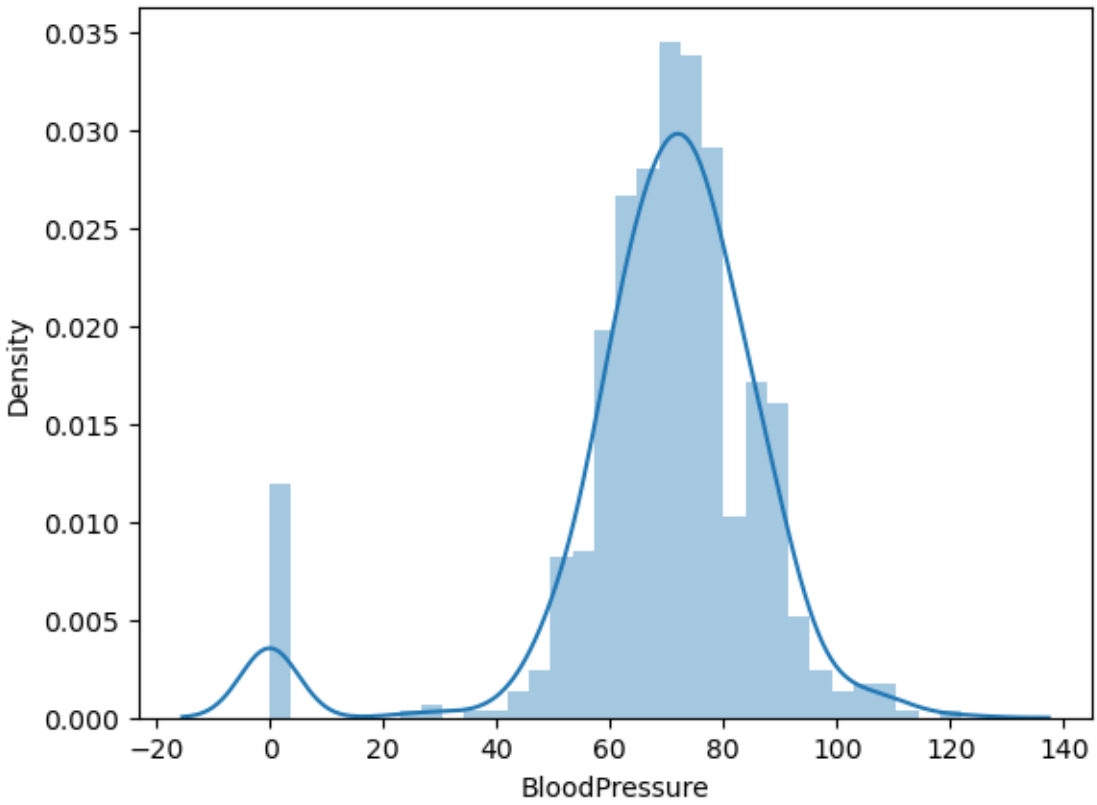
[16]:
```
for i in␣
↪["Pregnancies","Glucose","BloodPressure","SkinThickness","Insulin","BMI","DiabetesPedigreeF
↪
    sns.distplot(POD[i])
    plt.figure(figsize=[15,5])
    plt.show()
```



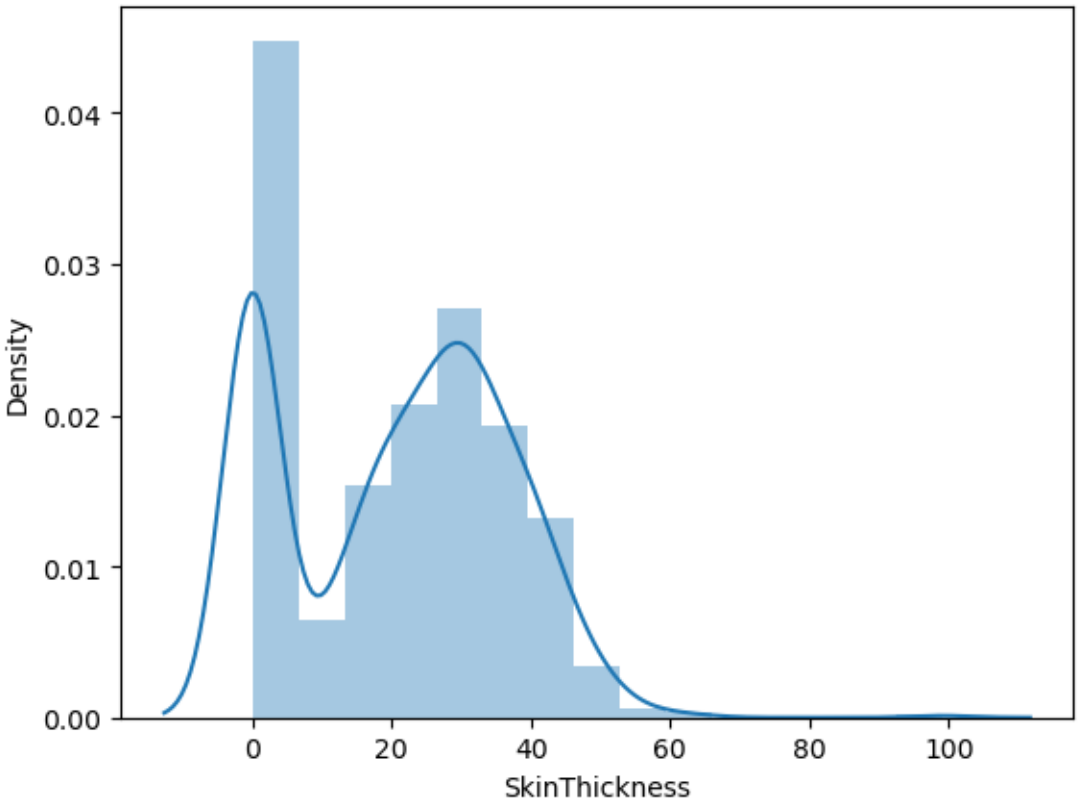

```
<Figure size 1500x500 with 0 Axes>
```



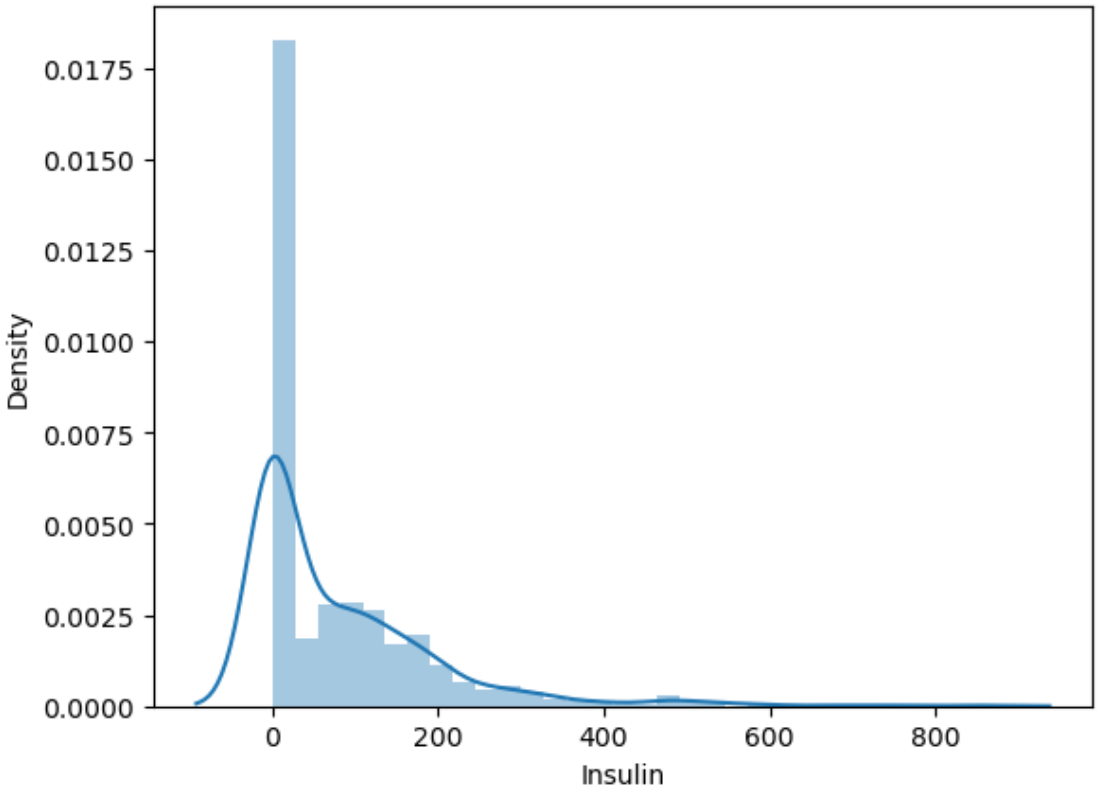

```
<Figure size 1500x500 with 0 Axes>
```



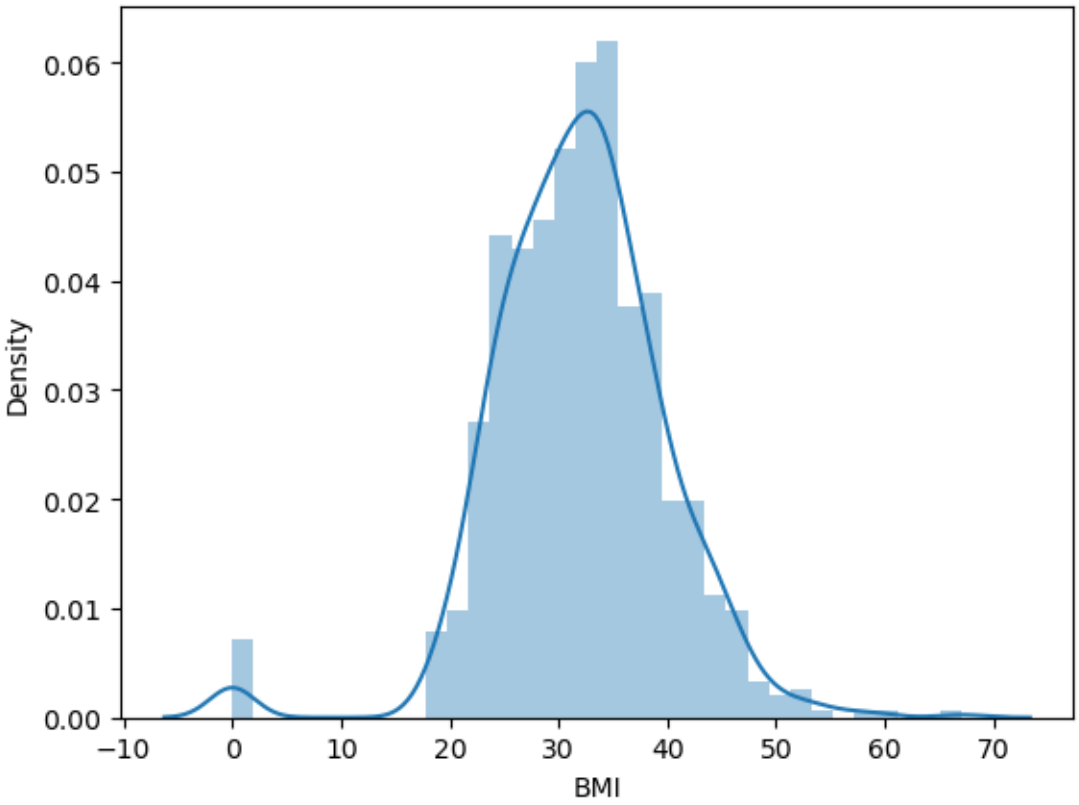

```
<Figure size 1500x500 with 0 Axes>
```



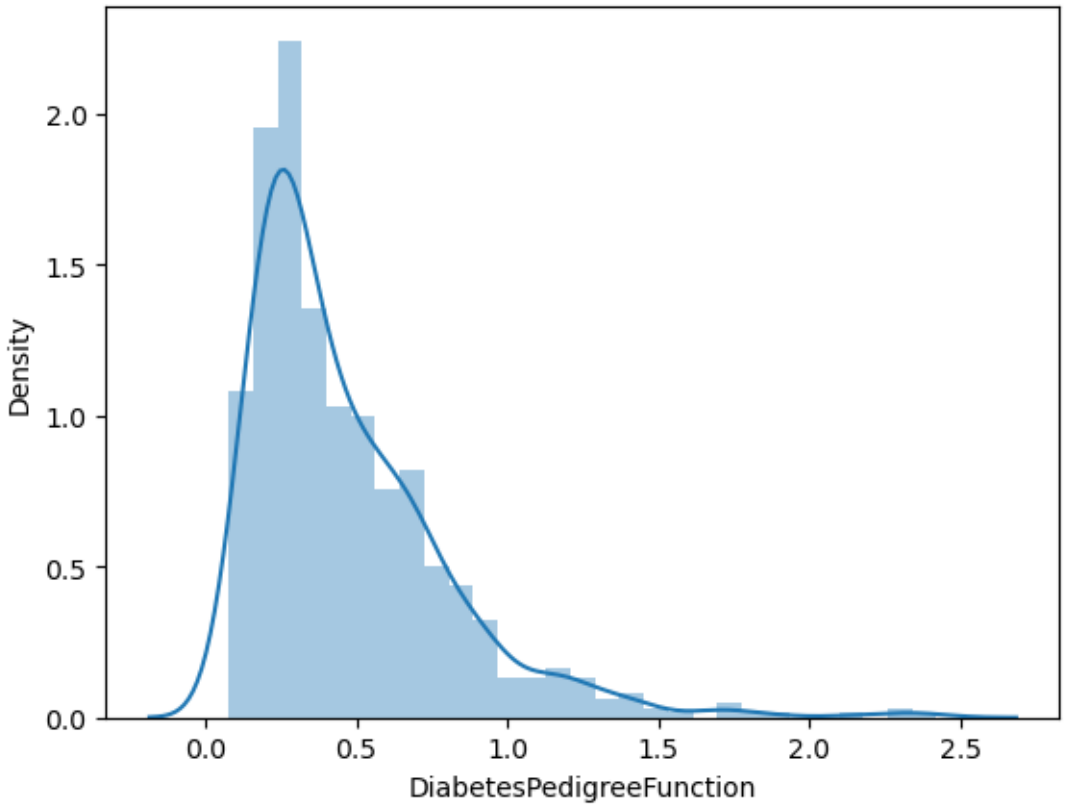

```
<Figure size 1500x500 with 0 Axes>
```



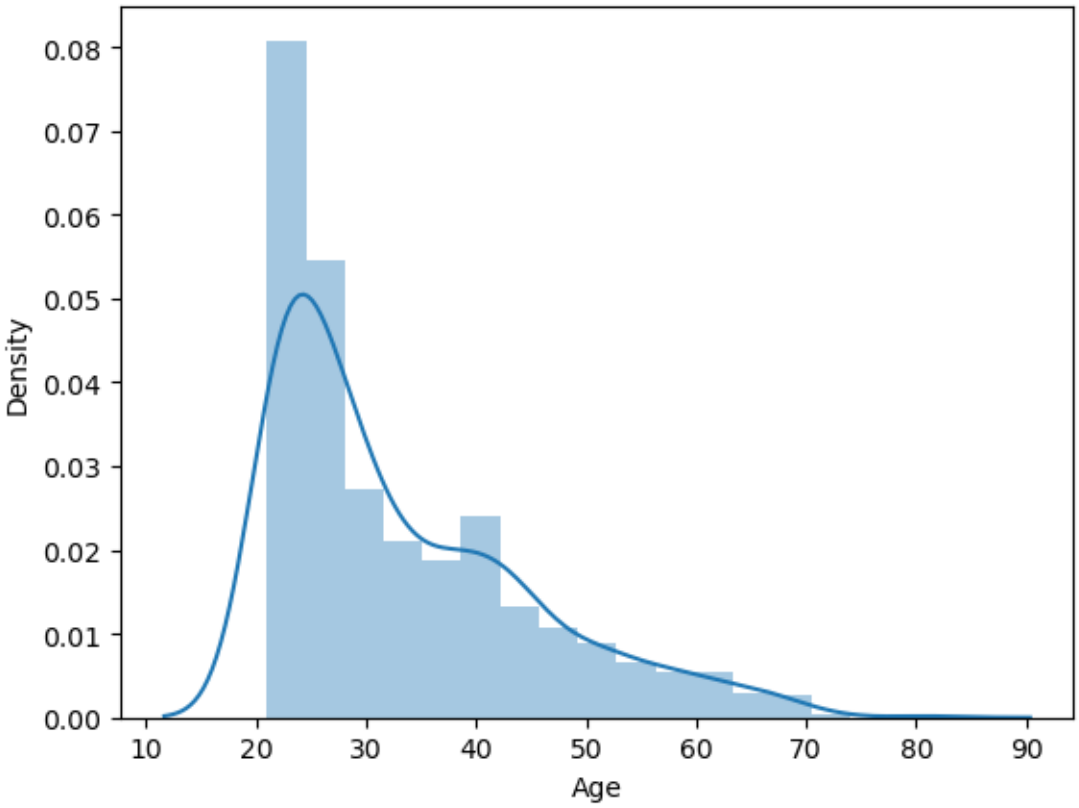

```
<Figure size 1500x500 with 0 Axes>
```



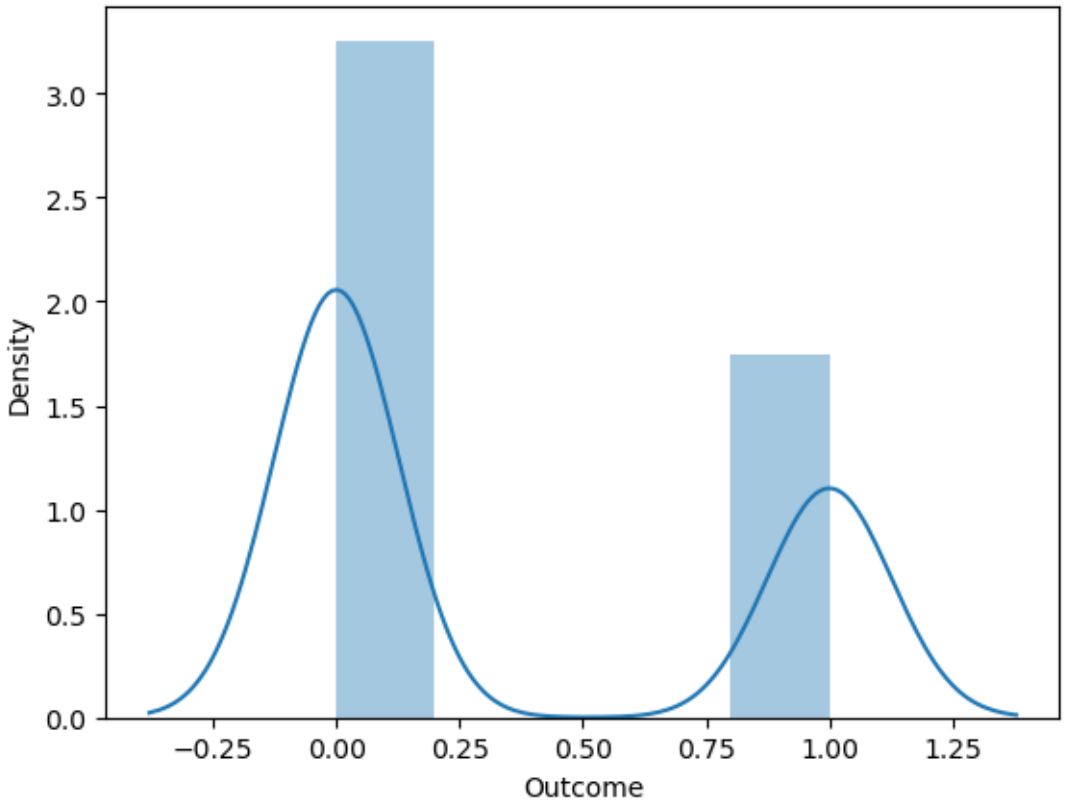

```
<Figure size 1500x500 with 0 Axes>
```




```
<Figure size 1500x500 with 0 Axes>
```



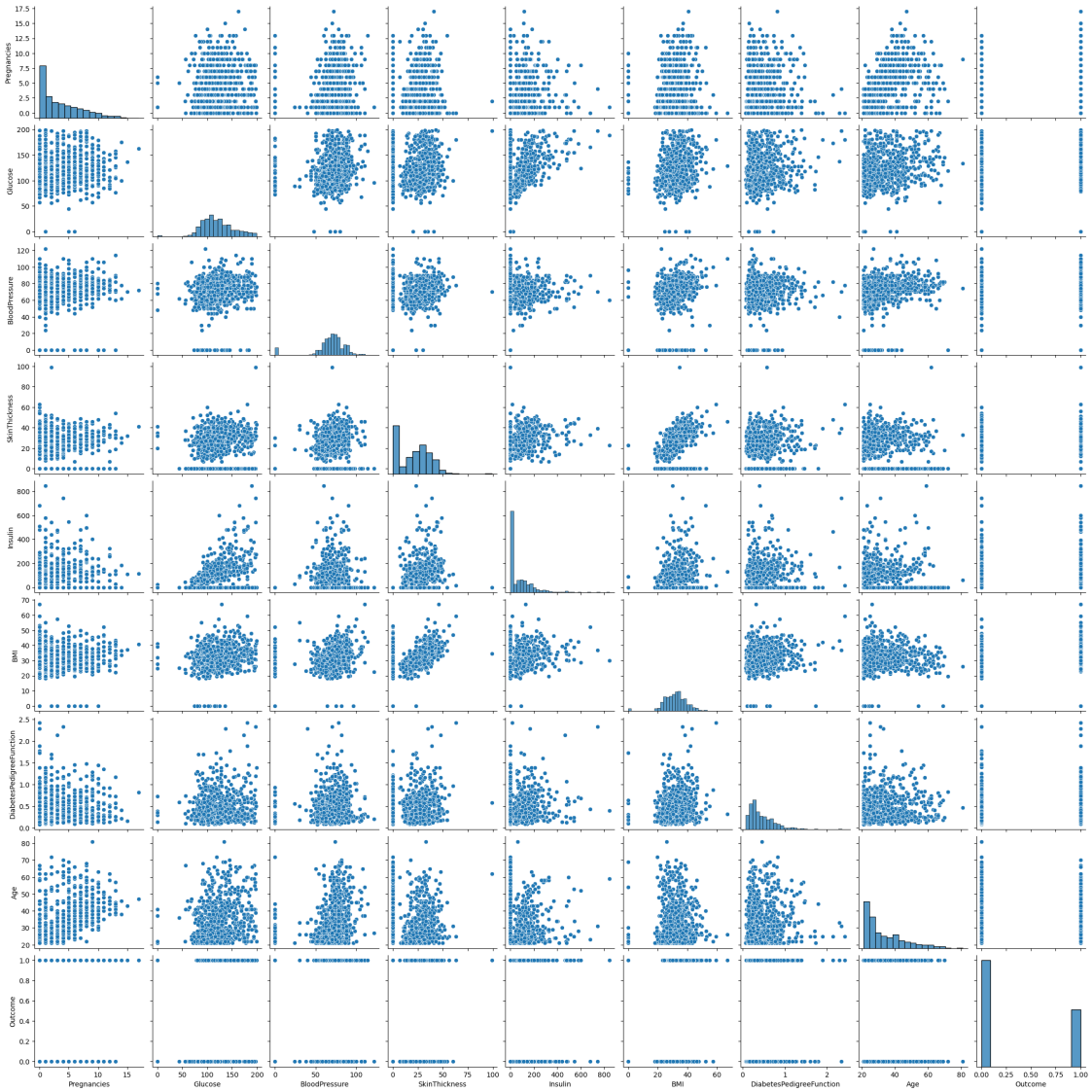

```
<Figure size 1500x500 with 0 Axes>
```




```
<Figure size 1500x500 with 0 Axes>
```

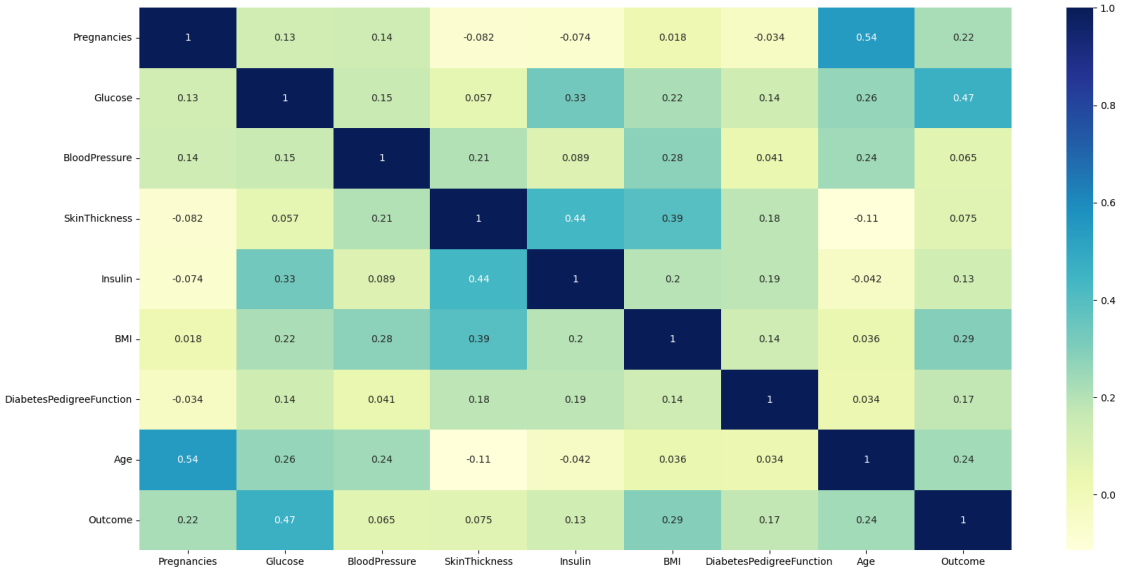
[17]:
```
sns.pairplot(POD)
```

[17]:
```
<seaborn.axisgrid.PairGrid at 0x1a835350250>
```

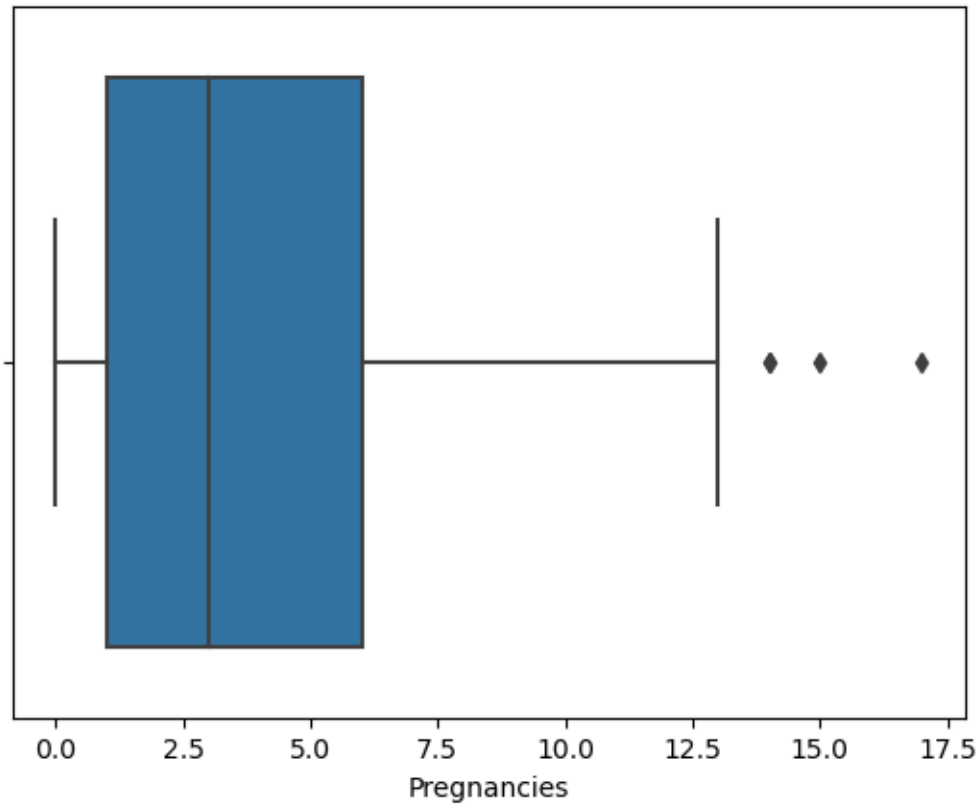


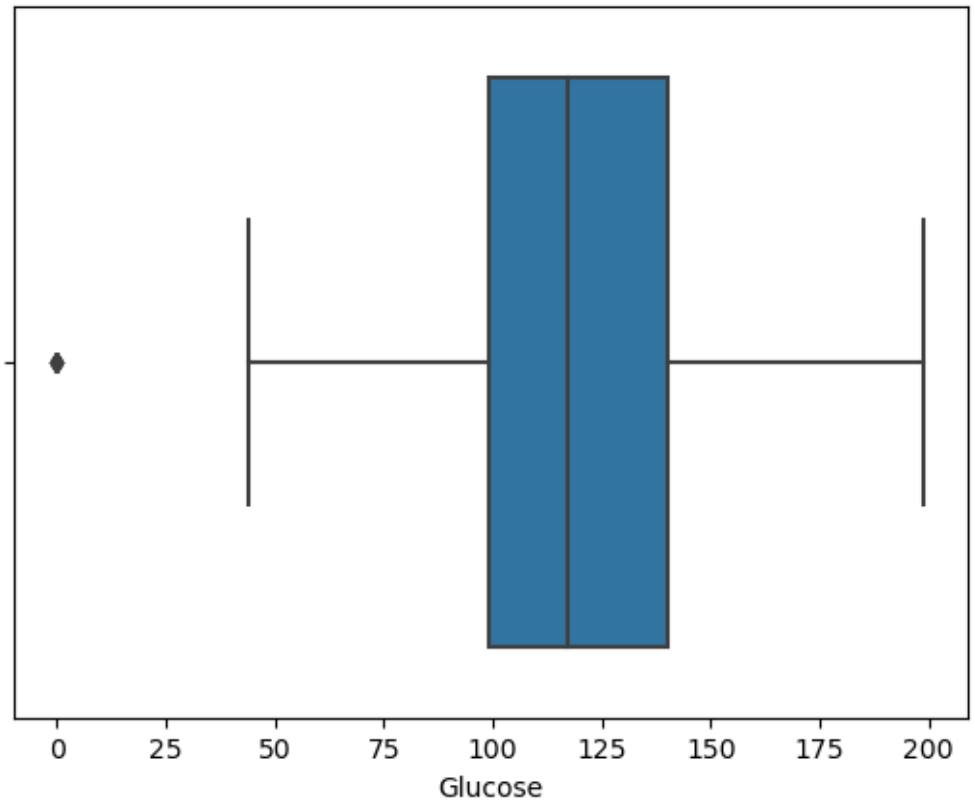
[18]:
```
plt.figure(figsize=(20,10))
sns.heatmap(POD.corr(),annot=True, cmap='YlGnBu')
```

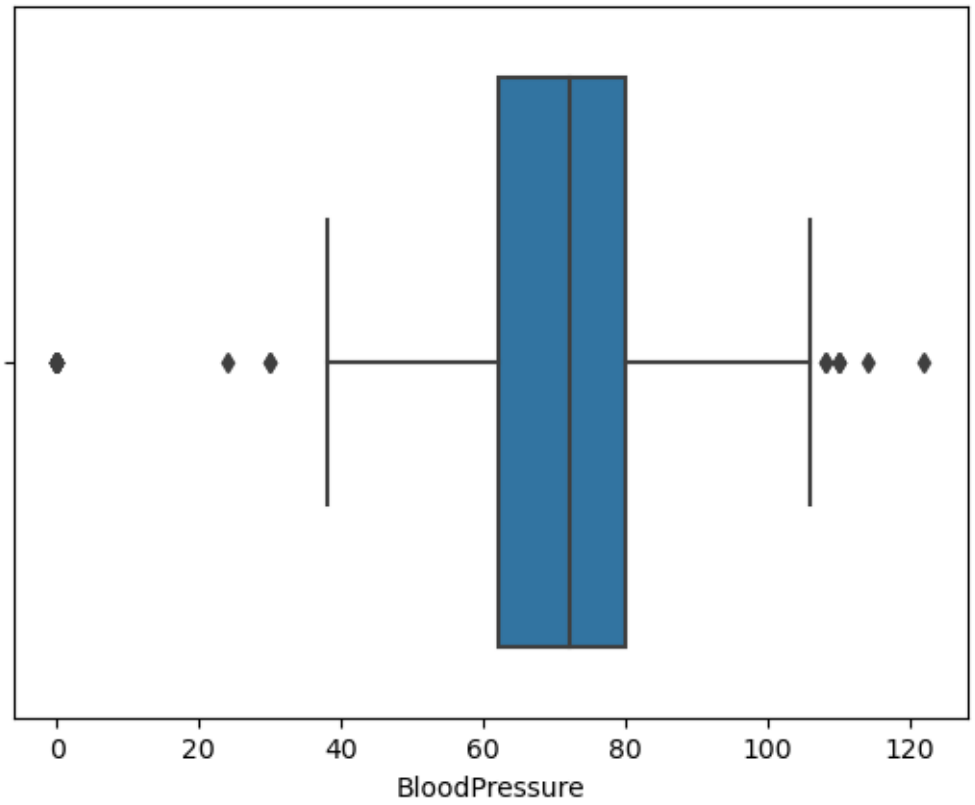
[18]: <AxesSubplot:>

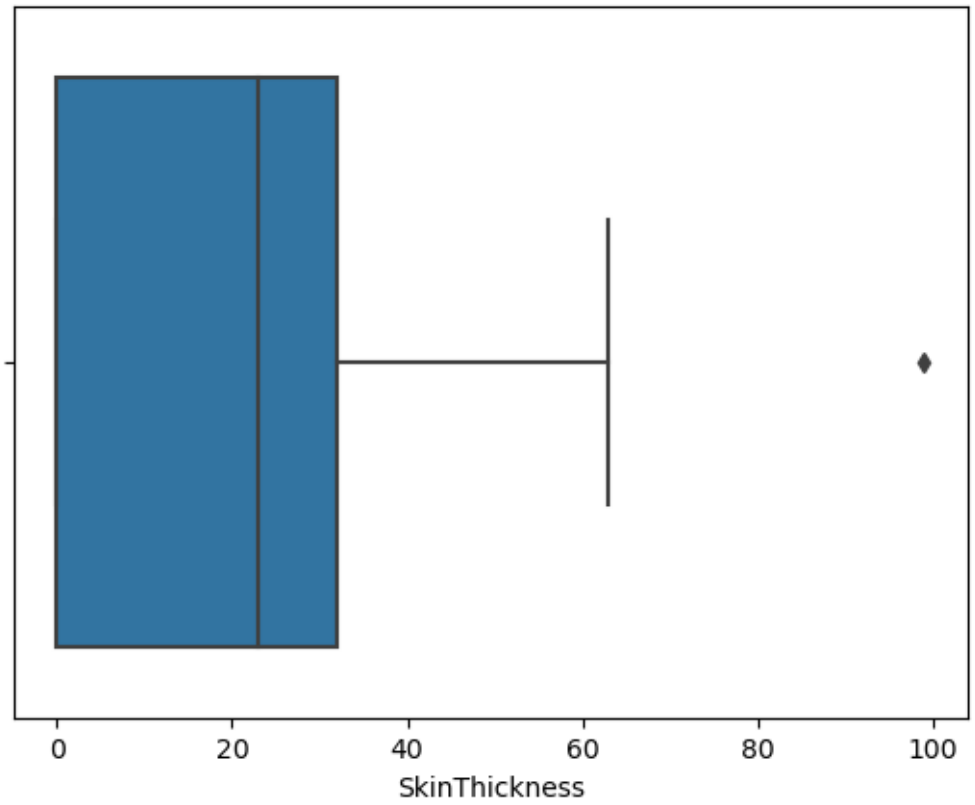


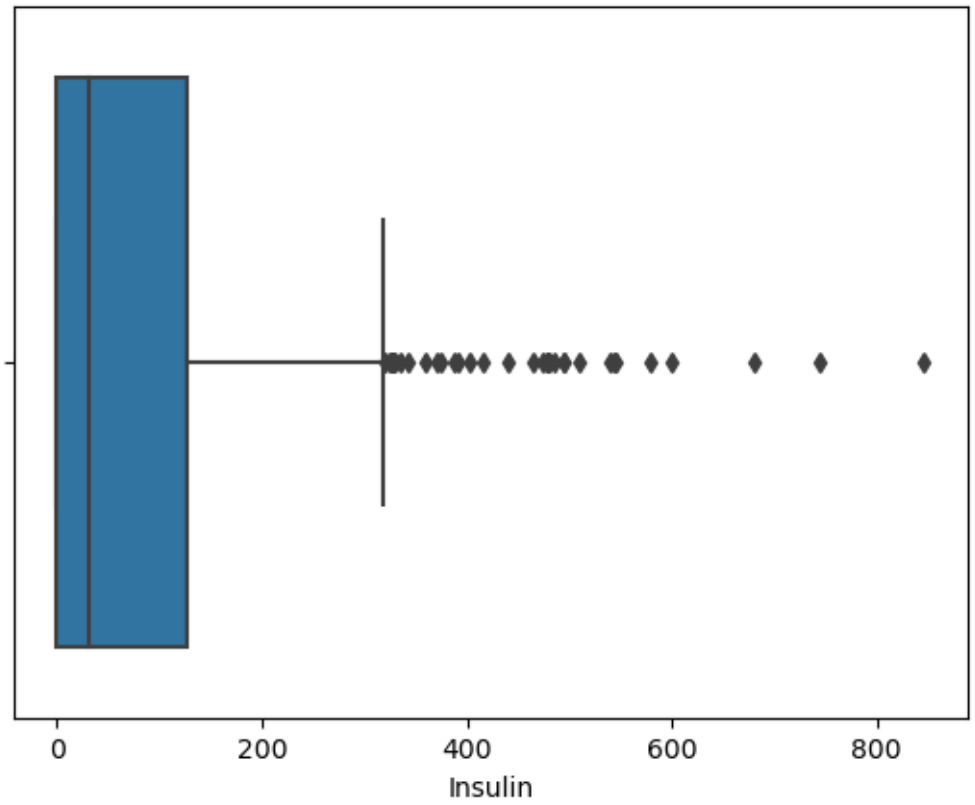
[21]:
```
for i in␣
 ↪["Pregnancies","Glucose","BloodPressure","SkinThickness","Insulin","BMI","DiabetesPedigreeF
 ↪
    sns.boxplot(data=POD, x=i)
    plt.show()
```

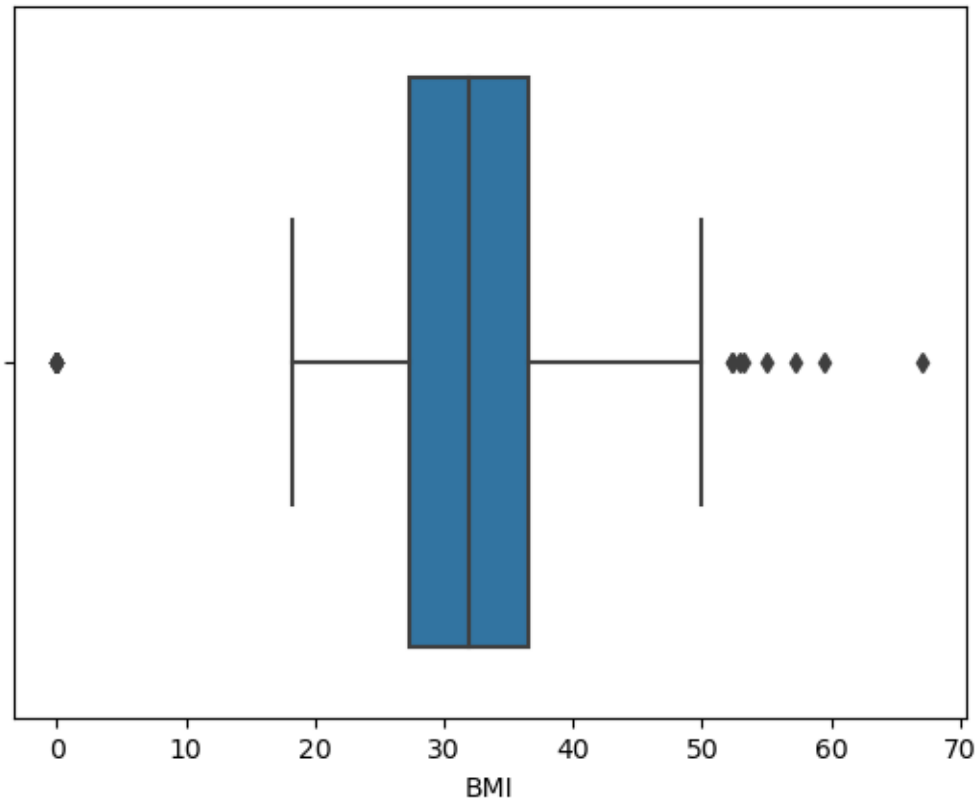

0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Pregnancies

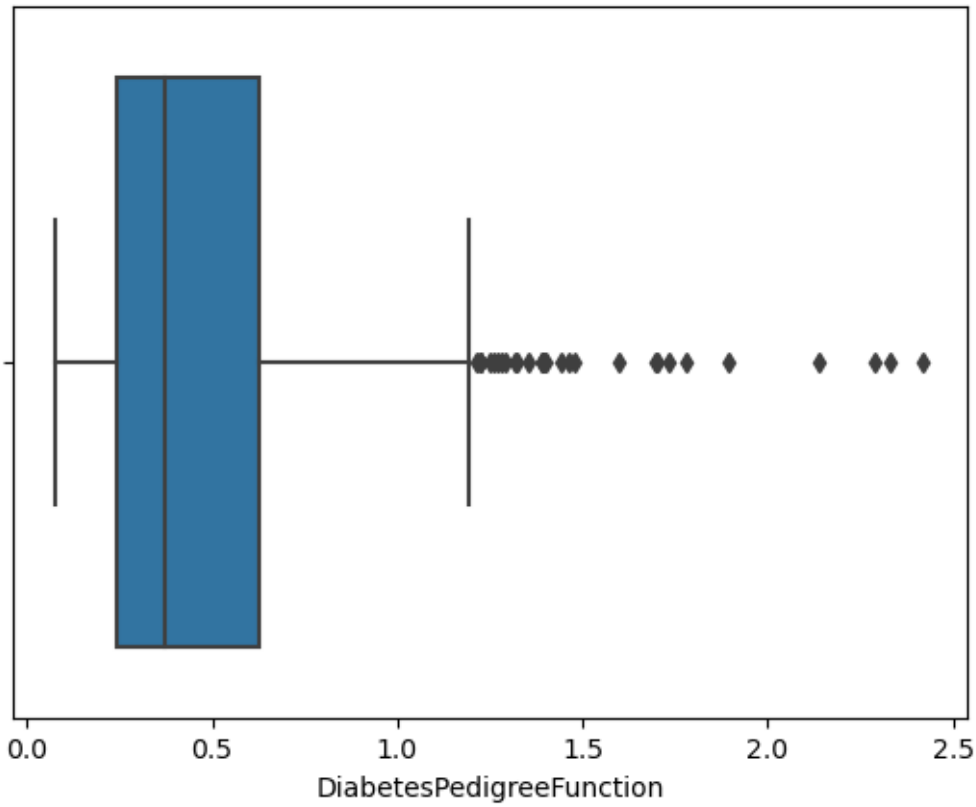

0
25
50
75
100
125
150
175
200
Glucose

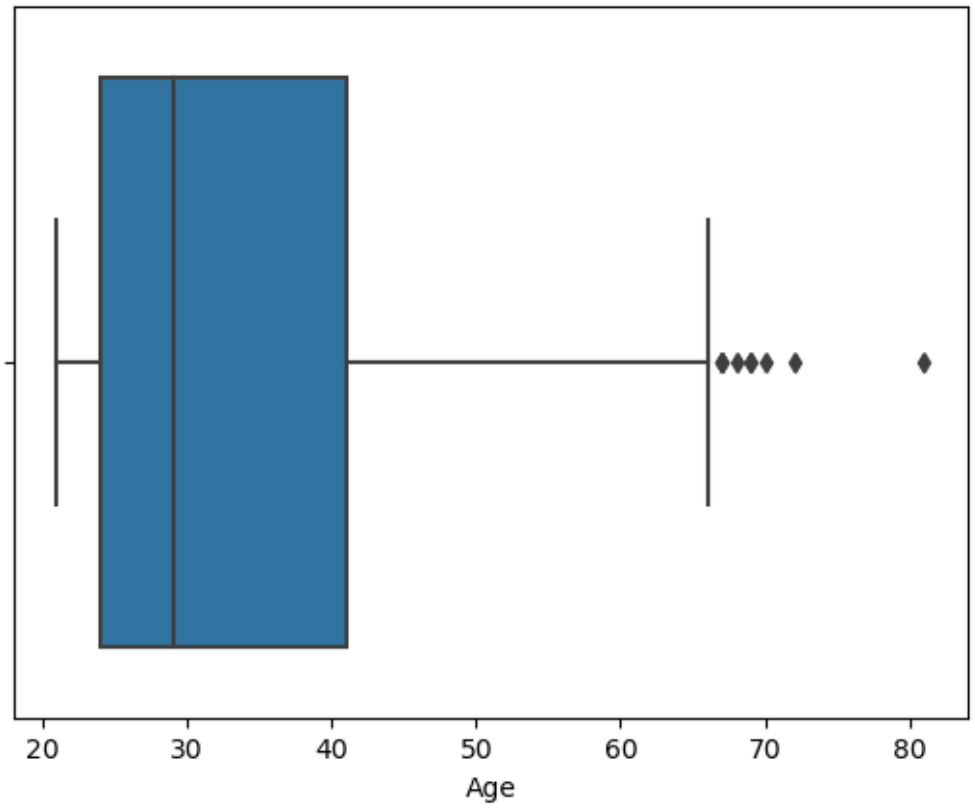

0
20
40
60
80
100
120
BloodPressure

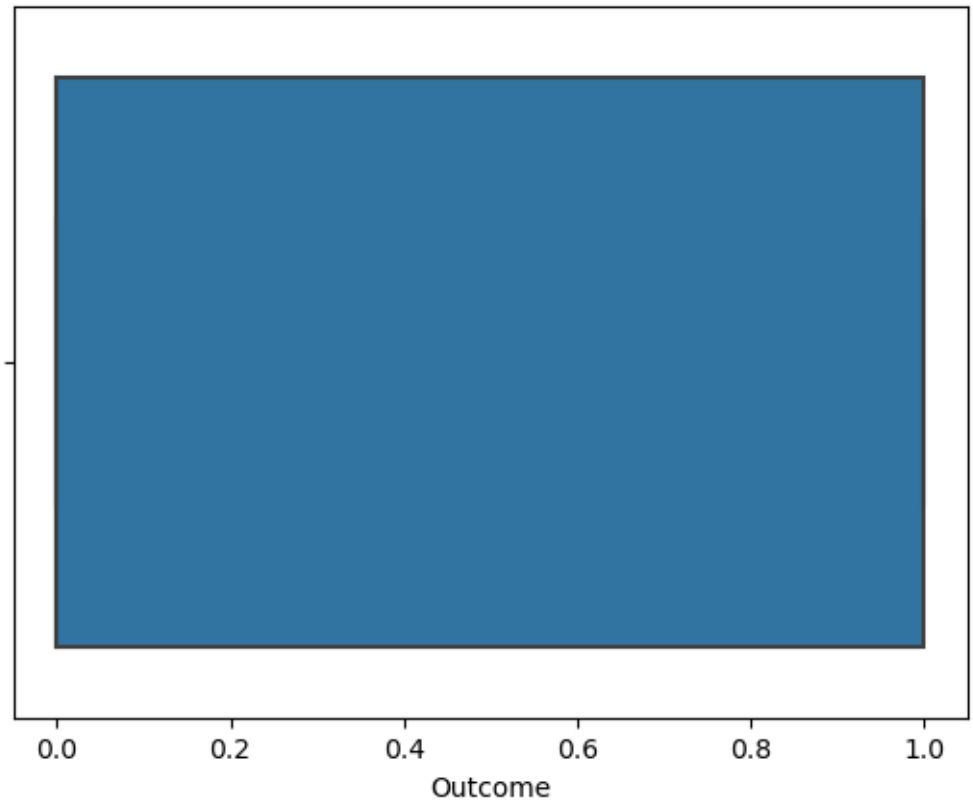

0
20
40
60
80
100
SkinThickness


0
200
400
600
800
Insulin


0
10
20
30
40
50
60
70
BMI


0.0
0.5
1.0
1.5
2.0
2.5
DiabetesPedigreeFunction


20
30
40
50
60
70
80
Age



[22]:
```
def outlierdetection(POD):
    sorted(POD)
    Q1,Q3=np.percentile(POD,[25,75])
    IQR=Q3-Q1
    lower_range=Q1-(1.5*IQR)
    Upper_range=Q3+(1.5*IQR)
    return lower_range,Upper_range
```

[24]:
```
features =␣
 ↪['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeF
x = POD[features]
y = POD.Outcome
```

[46]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split (x,y,test_size = 0.1,␣
 ↪random_state = 0)
```

2 LOGISTIC_REGRESSION_MODEL

[47]:
```
from sklearn.linear_model import LogisticRegression
```

[48]:
```
reg = LogisticRegression (max_iter = 1000)
```

[49]:
```
reg.fit(x_train, y_train)
```

[49]:
```
LogisticRegression(max_iter=1000)
```

[50]:
```
y_pred = reg.predict(x_test)
y_pred
```

[50]:
```
array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0], dtype=int64)
```

[51]:
```
x_test
```

[51]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
661	1	199	76	43	0	42.9	1.394	22
122	2	107	74	30	100	33.6	0.404	23
113	4	76	62	0	0	34.0	0.391	25
14	5	166	72	19	175	25.8	0.587	51
529	0	111	65	0	0	24.6	0.660	31
..	…	…	…	…	…	…	…	…
253	0	86	68	32	0	35.8	0.238	25
622	6	183	94	0	0	40.8	1.461	45
235	4	171	72	0	0	43.6	0.479	26
351	4	137	84	0	0	31.2	0.252	30
672	10	68	106	23	49	35.5	0.285	47

[77 rows x 8 columns]

[53]:
```
from sklearn import metrics
confusion_matrix = metrics.confusion_matrix(y_test, y_pred)
confusion_matrix
```

[53]:
```
array([[48,  3],
       [ 7, 19]], dtype=int64)
```

[52]:
```
print("Accuracy : ", metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy :  0.8701298701298701
```

[55]:
```
report=classification_report(y_test,y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.87      0.94      0.91        51
           1       0.86      0.73      0.79        26

    accuracy                           0.87        77
   macro avg       0.87      0.84      0.85        77
weighted avg       0.87      0.87      0.87        77
```

LOGISIC REGRESSION ACCURACY=87%

3 DECESION_TREE_MODEL

[56]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.model_selection import train_test_split, GridSearchCV
```

[72]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4,random_state=1)
```

[58]:
```
dt_model=DecisionTreeClassifier()
dt_model.fit(x_train,y_train)
```

[58]:
```
DecisionTreeClassifier()
```

[68]:
```
plt.figure(figsize=[20,20])
tree.plot_tree(dt_model, filled=True)
```

[68]:
```
[Text(0.40198863636363635, 0.9705882352941176, 'X[1] <= 129.5\ngini =
0.449\nsamples = 537\nvalue = [354, 183]'),
 Text(0.14879261363636365, 0.9117647058823529, 'X[5] <= 26.3\ngini =
0.329\nsamples = 357\nvalue = [283, 74]'),
 Text(0.045454545454545456, 0.8529411764705882, 'X[5] <= 9.1\ngini =
```

```
0.06\nsamples = 97\nvalue = [94, 3]'),
 Text(0.022727272727272728, 0.7941176470588235, 'X[0] <= 7.5\ngini =
0.444\nsamples = 6\nvalue = [4, 2]'),
 Text(0.011363636363636364, 0.7352941176470589, 'gini = 0.0\nsamples = 4\nvalue
= [4, 0]'),
 Text(0.03409090909090909, 0.7352941176470589, 'gini = 0.0\nsamples = 2\nvalue =
[0, 2]'),
 Text(0.06818181818181818, 0.7941176470588235, 'X[6] <= 0.669\ngini =
0.022\nsamples = 91\nvalue = [90, 1]'),
 Text(0.056818181818181816, 0.7352941176470589, 'gini = 0.0\nsamples = 76\nvalue
= [76, 0]'),
 Text(0.07954545454545454, 0.7352941176470589, 'X[6] <= 0.705\ngini =
0.124\nsamples = 15\nvalue = [14, 1]'),
 Text(0.06818181818181818, 0.6764705882352942, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.09090909090909091, 0.6764705882352942, 'gini = 0.0\nsamples = 14\nvalue
= [14, 0]'),
 Text(0.2521306818181818, 0.8529411764705882, 'X[7] <= 27.5\ngini =
0.397\nsamples = 260\nvalue = [189, 71]'),
 Text(0.14772727272727273, 0.7941176470588235, 'X[5] <= 45.4\ngini =
0.243\nsamples = 120\nvalue = [103, 17]'),
 Text(0.125, 0.7352941176470589, 'X[2] <= 12.0\ngini = 0.212\nsamples =
116\nvalue = [102, 14]'),
 Text(0.11363636363636363, 0.6764705882352942, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.13636363636363635, 0.6764705882352942, 'X[0] <= 7.0\ngini =
0.201\nsamples = 115\nvalue = [102, 13]'),
 Text(0.125, 0.6176470588235294, 'X[6] <= 1.272\ngini = 0.188\nsamples =
114\nvalue = [102, 12]'),
 Text(0.10227272727272728, 0.5588235294117647, 'X[5] <= 30.95\ngini =
0.165\nsamples = 110\nvalue = [100, 10]'),
 Text(0.09090909090909091, 0.5, 'gini = 0.0\nsamples = 43\nvalue = [43, 0]'),
 Text(0.11363636363636363, 0.5, 'X[2] <= 53.0\ngini = 0.254\nsamples = 67\nvalue
= [57, 10]'),
 Text(0.07386363636363637, 0.4411764705882353, 'X[6] <= 0.264\ngini =
0.5\nsamples = 6\nvalue = [3, 3]'),
 Text(0.0625, 0.38235294117647056, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.08522727272727272, 0.38235294117647056, 'X[4] <= 179.5\ngini =
0.375\nsamples = 4\nvalue = [3, 1]'),
 Text(0.07386363636363637, 0.3235294117647059, 'gini = 0.0\nsamples = 3\nvalue =
[3, 0]'),
 Text(0.09659090909090909, 0.3235294117647059, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.1534090909090909, 0.4411764705882353, 'X[6] <= 0.652\ngini =
0.203\nsamples = 61\nvalue = [54, 7]'),
 Text(0.13068181818181818, 0.38235294117647056, 'X[4] <= 36.5\ngini =
0.15\nsamples = 49\nvalue = [45, 4]'),
```

```
 Text(0.11931818181818182, 0.3235294117647059, 'X[2] <= 82.5\ngini =
0.32\nsamples = 20\nvalue = [16, 4]'),
 Text(0.10795454545454546, 0.2647058823529412, 'X[3] <= 40.5\ngini =
0.266\nsamples = 19\nvalue = [16, 3]'),
 Text(0.09659090909090909, 0.20588235294117646, 'X[1] <= 111.5\ngini =
0.198\nsamples = 18\nvalue = [16, 2]'),
 Text(0.08522727272727272, 0.14705882352941177, 'gini = 0.0\nsamples = 12\nvalue
= [12, 0]'),
 Text(0.10795454545454546, 0.14705882352941177, 'X[2] <= 72.0\ngini =
0.444\nsamples = 6\nvalue = [4, 2]'),
 Text(0.09659090909090909, 0.08823529411764706, 'gini = 0.0\nsamples = 3\nvalue
= [3, 0]'),
 Text(0.11931818181818182, 0.08823529411764706, 'X[7] <= 22.0\ngini =
0.444\nsamples = 3\nvalue = [1, 2]'),
 Text(0.10795454545454546, 0.029411764705882353, 'gini = 0.0\nsamples = 1\nvalue
= [1, 0]'),
 Text(0.13068181818181818, 0.029411764705882353, 'gini = 0.0\nsamples = 2\nvalue
= [0, 2]'),
 Text(0.11931818181818182, 0.20588235294117646, 'gini = 0.0\nsamples = 1\nvalue
= [0, 1]'),
 Text(0.13068181818181818, 0.2647058823529412, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.14204545454545456, 0.3235294117647059, 'gini = 0.0\nsamples = 29\nvalue
= [29, 0]'),
 Text(0.17613636363636365, 0.38235294117647056, 'X[4] <= 65.5\ngini =
0.375\nsamples = 12\nvalue = [9, 3]'),
 Text(0.16477272727272727, 0.3235294117647059, 'gini = 0.0\nsamples = 7\nvalue =
[7, 0]'),
 Text(0.1875, 0.3235294117647059, 'X[1] <= 115.0\ngini = 0.48\nsamples =
5\nvalue = [2, 3]'),
 Text(0.17613636363636365, 0.2647058823529412, 'gini = 0.0\nsamples = 3\nvalue =
[0, 3]'),
 Text(0.19886363636363635, 0.2647058823529412, 'gini = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(0.14772727272727273, 0.5588235294117647, 'X[6] <= 1.496\ngini =
0.5\nsamples = 4\nvalue = [2, 2]'),
 Text(0.13636363636363635, 0.5, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.1590909090909091, 0.5, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.14772727272727273, 0.6176470588235294, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.17045454545454544, 0.7352941176470589, 'X[0] <= 2.5\ngini =
0.375\nsamples = 4\nvalue = [1, 3]'),
 Text(0.1590909090909091, 0.6764705882352942, 'gini = 0.0\nsamples = 3\nvalue =
[0, 3]'),
 Text(0.18181818181818182, 0.6764705882352942, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.3565340909090909, 0.7941176470588235, 'X[6] <= 0.563\ngini =
```

```
0.474\nsamples = 140\nvalue = [86, 54]'),
 Text(0.26988636363636365, 0.7352941176470589, 'X[1] <= 101.5\ngini =
0.408\nsamples = 98\nvalue = [70, 28]'),
 Text(0.20454545454545456, 0.6764705882352942, 'X[2] <= 27.0\ngini =
0.208\nsamples = 34\nvalue = [30, 4]'),
 Text(0.19318181818181818, 0.6176470588235294, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.2159090909090909, 0.6176470588235294, 'X[7] <= 42.5\ngini =
0.165\nsamples = 33\nvalue = [30, 3]'),
 Text(0.20454545454545456, 0.5588235294117647, 'gini = 0.0\nsamples = 23\nvalue
= [23, 0]'),
 Text(0.22727272727272727, 0.5588235294117647, 'X[6] <= 0.383\ngini =
0.42\nsamples = 10\nvalue = [7, 3]'),
 Text(0.2159090909090909, 0.5, 'X[3] <= 24.0\ngini = 0.5\nsamples = 6\nvalue =
[3, 3]'),
 Text(0.20454545454545456, 0.4411764705882353, 'gini = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(0.22727272727272727, 0.4411764705882353, 'X[1] <= 99.0\ngini =
0.375\nsamples = 4\nvalue = [1, 3]'),
 Text(0.2159090909090909, 0.38235294117647056, 'gini = 0.0\nsamples = 3\nvalue =
[0, 3]'),
 Text(0.23863636363636365, 0.38235294117647056, 'gini = 0.0\nsamples = 1\nvalue
= [1, 0]'),
 Text(0.23863636363636365, 0.5, 'gini = 0.0\nsamples = 4\nvalue = [4, 0]'),
 Text(0.3352272727272727, 0.6764705882352942, 'X[2] <= 67.0\ngini =
0.469\nsamples = 64\nvalue = [40, 24]'),
 Text(0.29545454545454547, 0.6176470588235294, 'X[2] <= 58.0\ngini =
0.465\nsamples = 19\nvalue = [7, 12]'),
 Text(0.2727272727272727, 0.5588235294117647, 'X[1] <= 117.5\ngini =
0.245\nsamples = 7\nvalue = [6, 1]'),
 Text(0.26136363636363635, 0.5, 'gini = 0.0\nsamples = 5\nvalue = [5, 0]'),
 Text(0.2840909090909091, 0.5, 'X[6] <= 0.243\ngini = 0.5\nsamples = 2\nvalue =
[1, 1]'),
 Text(0.2727272727272727, 0.4411764705882353, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.29545454545454547, 0.4411764705882353, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.3181818181818182, 0.5588235294117647, 'X[6] <= 0.425\ngini =
0.153\nsamples = 12\nvalue = [1, 11]'),
 Text(0.3068181818181818, 0.5, 'gini = 0.0\nsamples = 11\nvalue = [0, 11]'),
 Text(0.32954545454545453, 0.5, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.375, 0.6176470588235294, 'X[5] <= 43.1\ngini = 0.391\nsamples =
45\nvalue = [33, 12]'),
 Text(0.36363636363636365, 0.5588235294117647, 'X[3] <= 27.0\ngini =
0.337\nsamples = 42\nvalue = [33, 9]'),
 Text(0.3522727272727273, 0.5, 'X[2] <= 89.0\ngini = 0.461\nsamples = 25\nvalue
= [16, 9]'),
```

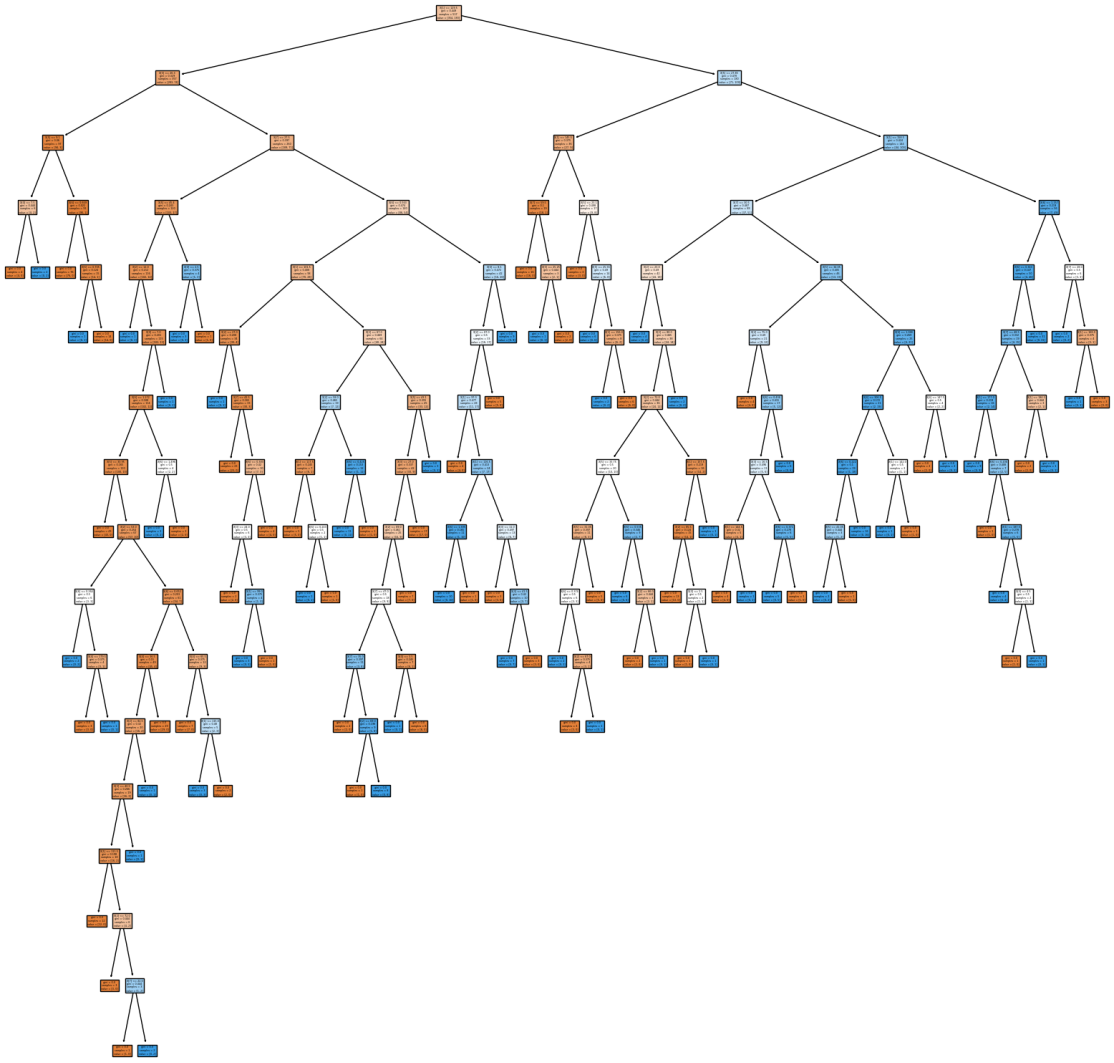
```
 Text(0.3409090909090909, 0.4411764705882353, 'X[7] <= 47.0\ngini = 0.5\nsamples
= 18\nvalue = [9, 9]'),
 Text(0.3181818181818182, 0.38235294117647056, 'X[7] <= 35.0\ngini =
0.397\nsamples = 11\nvalue = [3, 8]'),
 Text(0.3068181818181818, 0.3235294117647059, 'gini = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(0.32954545454545453, 0.3235294117647059, 'X[2] <= 69.0\ngini =
0.198\nsamples = 9\nvalue = [1, 8]'),
 Text(0.3181818181818182, 0.2647058823529412, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.3409090909090909, 0.2647058823529412, 'gini = 0.0\nsamples = 8\nvalue =
[0, 8]'),
 Text(0.36363636363636365, 0.38235294117647056, 'X[0] <= 3.5\ngini =
0.245\nsamples = 7\nvalue = [6, 1]'),
 Text(0.3522727272727273, 0.3235294117647059, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.375, 0.3235294117647059, 'gini = 0.0\nsamples = 6\nvalue = [6, 0]'),
 Text(0.36363636363636365, 0.4411764705882353, 'gini = 0.0\nsamples = 7\nvalue =
[7, 0]'),
 Text(0.375, 0.5, 'gini = 0.0\nsamples = 17\nvalue = [17, 0]'),
 Text(0.38636363636363635, 0.5588235294117647, 'gini = 0.0\nsamples = 3\nvalue =
[0, 3]'),
 Text(0.4431818181818182, 0.7352941176470589, 'X[0] <= 8.5\ngini =
0.472\nsamples = 42\nvalue = [16, 26]'),
 Text(0.4318181818181818, 0.6764705882352942, 'X[2] <= 87.0\ngini = 0.5\nsamples
= 33\nvalue = [16, 17]'),
 Text(0.42045454545454547, 0.6176470588235294, 'X[1] <= 97.0\ngini =
0.477\nsamples = 28\nvalue = [11, 17]'),
 Text(0.4090909090909091, 0.5588235294117647, 'gini = 0.0\nsamples = 4\nvalue =
[4, 0]'),
 Text(0.4318181818181818, 0.5588235294117647, 'X[1] <= 116.5\ngini =
0.413\nsamples = 24\nvalue = [7, 17]'),
 Text(0.4090909090909091, 0.5, 'X[6] <= 1.395\ngini = 0.165\nsamples = 11\nvalue
= [1, 10]'),
 Text(0.3977272727272727, 0.4411764705882353, 'gini = 0.0\nsamples = 10\nvalue =
[0, 10]'),
 Text(0.42045454545454547, 0.4411764705882353, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.45454545454545453, 0.5, 'X[3] <= 11.0\ngini = 0.497\nsamples = 13\nvalue
= [6, 7]'),
 Text(0.4431818181818182, 0.4411764705882353, 'gini = 0.0\nsamples = 3\nvalue =
[3, 0]'),
 Text(0.4659090909090909, 0.4411764705882353, 'X[3] <= 41.5\ngini =
0.42\nsamples = 10\nvalue = [3, 7]'),
 Text(0.45454545454545453, 0.38235294117647056, 'gini = 0.0\nsamples = 7\nvalue
= [0, 7]'),
 Text(0.4772727272727273, 0.38235294117647056, 'gini = 0.0\nsamples = 3\nvalue =
```

```
[3, 0]'),
 Text(0.4431818181818182, 0.6176470588235294, 'gini = 0.0\nsamples = 5\nvalue =
[5, 0]'),
 Text(0.45454545454545453, 0.6764705882352942, 'gini = 0.0\nsamples = 9\nvalue =
[0, 9]'),
 Text(0.6551846590909091, 0.9117647058823529, 'X[5] <= 27.85\ngini =
0.478\nsamples = 180\nvalue = [71, 109]'),
 Text(0.5056818181818182, 0.8529411764705882, 'X[1] <= 145.5\ngini =
0.375\nsamples = 36\nvalue = [27, 9]'),
 Text(0.48295454545454547, 0.7941176470588235, 'X[7] <= 59.5\ngini =
0.1\nsamples = 19\nvalue = [18, 1]'),
 Text(0.4715909090909091, 0.7352941176470589, 'gini = 0.0\nsamples = 16\nvalue =
[16, 0]'),
 Text(0.4943181818181818, 0.7352941176470589, 'X[5] <= 25.25\ngini =
0.444\nsamples = 3\nvalue = [2, 1]'),
 Text(0.48295454545454547, 0.6764705882352942, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.5056818181818182, 0.6764705882352942, 'gini = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(0.5284090909090909, 0.7941176470588235, 'X[5] <= 23.1\ngini =
0.498\nsamples = 17\nvalue = [9, 8]'),
 Text(0.5170454545454546, 0.7352941176470589, 'gini = 0.0\nsamples = 3\nvalue =
[3, 0]'),
 Text(0.5397727272727273, 0.7352941176470589, 'X[5] <= 25.55\ngini =
0.49\nsamples = 14\nvalue = [6, 8]'),
 Text(0.5284090909090909, 0.6764705882352942, 'gini = 0.0\nsamples = 6\nvalue =
[0, 6]'),
 Text(0.5511363636363636, 0.6764705882352942, 'X[1] <= 156.0\ngini =
0.375\nsamples = 8\nvalue = [6, 2]'),
 Text(0.5397727272727273, 0.6176470588235294, 'gini = 0.0\nsamples = 2\nvalue =
[0, 2]'),
 Text(0.5625, 0.6176470588235294, 'gini = 0.0\nsamples = 6\nvalue = [6, 0]'),
 Text(0.8046875, 0.8529411764705882, 'X[1] <= 158.5\ngini = 0.424\nsamples =
144\nvalue = [44, 100]'),
 Text(0.6661931818181818, 0.7941176470588235, 'X[7] <= 30.5\ngini =
0.487\nsamples = 88\nvalue = [37, 51]'),
 Text(0.5852272727272727, 0.7352941176470589, 'X[2] <= 23.0\ngini =
0.49\nsamples = 42\nvalue = [24, 18]'),
 Text(0.5738636363636364, 0.6764705882352942, 'gini = 0.0\nsamples = 4\nvalue =
[0, 4]'),
 Text(0.5965909090909091, 0.6764705882352942, 'X[2] <= 88.0\ngini =
0.465\nsamples = 38\nvalue = [24, 14]'),
 Text(0.5852272727272727, 0.6176470588235294, 'X[2] <= 72.0\ngini =
0.444\nsamples = 36\nvalue = [24, 12]'),
 Text(0.5454545454545454, 0.5588235294117647, 'X[5] <= 33.75\ngini =
0.5\nsamples = 20\nvalue = [10, 10]'),
 Text(0.5227272727272727, 0.5, 'X[5] <= 31.05\ngini = 0.397\nsamples = 11\nvalue
```

```
= [8, 3]'),
 Text(0.5113636363636364, 0.4411764705882353, 'X[6] <= 0.374\ngini =
0.5\nsamples = 6\nvalue = [3, 3]'),
 Text(0.5, 0.38235294117647056, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.5227272727272727, 0.38235294117647056, 'X[2] <= 67.0\ngini =
0.375\nsamples = 4\nvalue = [3, 1]'),
 Text(0.5113636363636364, 0.3235294117647059, 'gini = 0.0\nsamples = 3\nvalue =
[3, 0]'),
 Text(0.5340909090909091, 0.3235294117647059, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.5340909090909091, 0.4411764705882353, 'gini = 0.0\nsamples = 5\nvalue =
[5, 0]'),
 Text(0.5681818181818182, 0.5, 'X[6] <= 0.535\ngini = 0.346\nsamples = 9\nvalue
= [2, 7]'),
 Text(0.5568181818181818, 0.4411764705882353, 'gini = 0.0\nsamples = 6\nvalue =
[0, 6]'),
 Text(0.5795454545454546, 0.4411764705882353, 'X[2] <= 66.0\ngini =
0.444\nsamples = 3\nvalue = [2, 1]'),
 Text(0.5681818181818182, 0.38235294117647056, 'gini = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(0.5909090909090909, 0.38235294117647056, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.625, 0.5588235294117647, 'X[1] <= 157.5\ngini = 0.219\nsamples =
16\nvalue = [14, 2]'),
 Text(0.6136363636363636, 0.5, 'X[2] <= 85.5\ngini = 0.124\nsamples = 15\nvalue
= [14, 1]'),
 Text(0.6022727272727273, 0.4411764705882353, 'gini = 0.0\nsamples = 13\nvalue =
[13, 0]'),
 Text(0.625, 0.4411764705882353, 'X[0] <= 2.5\ngini = 0.5\nsamples = 2\nvalue =
[1, 1]'),
 Text(0.6136363636363636, 0.38235294117647056, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.6363636363636364, 0.38235294117647056, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.6363636363636364, 0.5, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.6079545454545454, 0.6176470588235294, 'gini = 0.0\nsamples = 2\nvalue =
[0, 2]'),
 Text(0.7471590909090909, 0.7352941176470589, 'X[5] <= 34.05\ngini =
0.405\nsamples = 46\nvalue = [13, 33]'),
 Text(0.6818181818181818, 0.6764705882352942, 'X[2] <= 75.0\ngini =
0.49\nsamples = 21\nvalue = [9, 12]'),
 Text(0.6704545454545454, 0.6176470588235294, 'gini = 0.0\nsamples = 4\nvalue =
[4, 0]'),
 Text(0.6931818181818182, 0.6176470588235294, 'X[6] <= 0.436\ngini =
0.415\nsamples = 17\nvalue = [5, 12]'),
 Text(0.6818181818181818, 0.5588235294117647, 'X[3] <= 16.5\ngini =
0.496\nsamples = 11\nvalue = [5, 6]'),
```

```
 Text(0.6590909090909091, 0.5, 'X[1] <= 144.5\ngini = 0.32\nsamples = 5\nvalue =
[4, 1]'),
 Text(0.6477272727272727, 0.4411764705882353, 'gini = 0.0\nsamples = 4\nvalue =
[4, 0]'),
 Text(0.6704545454545454, 0.4411764705882353, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.7045454545454546, 0.5, 'X[6] <= 0.371\ngini = 0.278\nsamples = 6\nvalue
= [1, 5]'),
 Text(0.6931818181818182, 0.4411764705882353, 'gini = 0.0\nsamples = 5\nvalue =
[0, 5]'),
 Text(0.7159090909090909, 0.4411764705882353, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.7045454545454546, 0.5588235294117647, 'gini = 0.0\nsamples = 6\nvalue =
[0, 6]'),
 Text(0.8125, 0.6764705882352942, 'X[6] <= 1.088\ngini = 0.269\nsamples =
25\nvalue = [4, 21]'),
 Text(0.7840909090909091, 0.6176470588235294, 'X[4] <= 306.5\ngini =
0.172\nsamples = 21\nvalue = [2, 19]'),
 Text(0.7613636363636364, 0.5588235294117647, 'X[6] <= 0.222\ngini =
0.1\nsamples = 19\nvalue = [1, 18]'),
 Text(0.75, 0.5, 'X[5] <= 41.05\ngini = 0.444\nsamples = 3\nvalue = [1, 2]'),
 Text(0.7386363636363636, 0.4411764705882353, 'gini = 0.0\nsamples = 2\nvalue =
[0, 2]'),
 Text(0.7613636363636364, 0.4411764705882353, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.7727272727272727, 0.5, 'gini = 0.0\nsamples = 16\nvalue = [0, 16]'),
 Text(0.8068181818181818, 0.5588235294117647, 'X[1] <= 142.0\ngini =
0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(0.7954545454545454, 0.5, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.8181818181818182, 0.5, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.8409090909090909, 0.6176470588235294, 'X[4] <= 147.5\ngini =
0.5\nsamples = 4\nvalue = [2, 2]'),
 Text(0.8295454545454546, 0.5588235294117647, 'gini = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(0.8522727272727273, 0.5588235294117647, 'gini = 0.0\nsamples = 2\nvalue =
[0, 2]'),
 Text(0.9431818181818182, 0.7941176470588235, 'X[6] <= 1.157\ngini =
0.219\nsamples = 56\nvalue = [7, 49]'),
 Text(0.9204545454545454, 0.7352941176470589, 'X[6] <= 0.343\ngini =
0.147\nsamples = 50\nvalue = [4, 46]'),
 Text(0.9090909090909091, 0.6764705882352942, 'X[7] <= 48.5\ngini =
0.332\nsamples = 19\nvalue = [4, 15]'),
 Text(0.8863636363636364, 0.6176470588235294, 'X[1] <= 177.0\ngini =
0.219\nsamples = 16\nvalue = [2, 14]'),
 Text(0.875, 0.5588235294117647, 'gini = 0.0\nsamples = 9\nvalue = [0, 9]'),
 Text(0.8977272727272727, 0.5588235294117647, 'X[6] <= 0.206\ngini =
0.408\nsamples = 7\nvalue = [2, 5]'),
```

```
 Text(0.8863636363636364, 0.5, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.9090909090909091, 0.5, 'X[1] <= 187.5\ngini = 0.278\nsamples = 6\nvalue
= [1, 5]'),
 Text(0.8977272727272727, 0.4411764705882353, 'gini = 0.0\nsamples = 4\nvalue =
[0, 4]'),
 Text(0.9204545454545454, 0.4411764705882353, 'X[0] <= 4.5\ngini = 0.5\nsamples
= 2\nvalue = [1, 1]'),
 Text(0.9090909090909091, 0.38235294117647056, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.9318181818181818, 0.38235294117647056, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.9318181818181818, 0.6176470588235294, 'X[1] <= 184.5\ngini =
0.444\nsamples = 3\nvalue = [2, 1]'),
 Text(0.9204545454545454, 0.5588235294117647, 'gini = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(0.9431818181818182, 0.5588235294117647, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.9318181818181818, 0.6764705882352942, 'gini = 0.0\nsamples = 31\nvalue =
[0, 31]'),
 Text(0.9659090909090909, 0.7352941176470589, 'X[7] <= 28.0\ngini = 0.5\nsamples
= 6\nvalue = [3, 3]'),
 Text(0.9545454545454546, 0.6764705882352942, 'gini = 0.0\nsamples = 2\nvalue =
[0, 2]'),
 Text(0.9772727272727273, 0.6764705882352942, 'X[1] <= 168.0\ngini =
0.375\nsamples = 4\nvalue = [3, 1]'),
 Text(0.9659090909090909, 0.6176470588235294, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.9886363636363636, 0.6176470588235294, 'gini = 0.0\nsamples = 3\nvalue =
[3, 0]')]
```



[73]:
```
prediction=dt_model.predict(x_test)
prediction
```

[73]:
```
array([0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
```

```
 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0],
dtype=int64)
```

[74]:
```
confusion_matrix = metrics.confusion_matrix(y_test, prediction)
confusion_matrix
```

[74]:
```
array([[171,  28],
       [ 39,  70]], dtype=int64)
```

[75]:
```
accuracy_score(y_test,prediction)
```

[75]:
```
0.7824675324675324
```

[76]:
```
report=classification_report(y_test,prediction)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.81      0.86      0.84       199
           1       0.71      0.64      0.68       109

    accuracy                           0.78       308
   macro avg       0.76      0.75      0.76       308
weighted avg       0.78      0.78      0.78       308
```

DESION TREE ACCURACY = 78%

4 RANDOM_FOREST_MODEL

[77]:
```
from sklearn.ensemble import RandomForestClassifier
np.random.seed(0)
```

[102]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
```

[103]:
```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
```

[103]:
```
RandomForestClassifier()
```

[104]:
```
prediction=rf.predict(x_test)
```

[105]:
```
confusion_matrix = metrics.confusion_matrix(y_test, prediction)
confusion_matrix
```

[105]:
```
array([[89, 10],
       [19, 36]], dtype=int64)
```

[106]:
```
report=classification_report(y_test,prediction)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.82      0.90      0.86        99
           1       0.78      0.65      0.71        55

    accuracy                           0.81       154
   macro avg       0.80      0.78      0.79       154
weighted avg       0.81      0.81      0.81       154
```

[107]:
```
accuracy_score(y_test,prediction)
```

[107]:
```
0.8116883116883117
```

RANDOM FOREST ACCURACY = 81%

5 K-Nearest_Neighbors_MODEL

[108]:
```
from sklearn import preprocessing
from sklearn import neighbors
from sklearn import metrics as sm
```

[109]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
```

[110]:
```
knn=neighbors.KNeighborsClassifier()
```

[111]:
```
knn.fit(x_train,y_train)
```

[111]:
```
KNeighborsClassifier()
```

[112]:
```
prediction=knn.predict(x_test)
```

[114]:
```
confusion_matrix = metrics.confusion_matrix(y_test, prediction)
confusion_matrix
```

[114]:
```
array([[128,  18],
       [ 35,  50]], dtype=int64)
```

[115]:
```
accuracy_score(y_test,prediction)
```

[115]:
```
0.7705627705627706
```

[116]:
```
score=classification_report(y_test,prediction)
print(score)
```

```
              precision    recall  f1-score   support

           0       0.79      0.88      0.83       146
           1       0.74      0.59      0.65        85

    accuracy                           0.77       231
   macro avg       0.76      0.73      0.74       231
weighted avg       0.77      0.77      0.76       231
```

KNN ACCURACY IS 77%

6 Gaussian_Naive_Bayes_MODEL

[117]:
```
from sklearn.naive_bayes import GaussianNB
```

[124]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
```

[125]:
```
gnb=GaussianNB()
gnb.fit(x_train,y_train)
```

[125]:
```
GaussianNB()
```

[126]:
```
prediction=gnb.predict(x_test)
```

[127]:
```
confusion_matrix = metrics.confusion_matrix(y_test, prediction)
confusion_matrix
```

[127]:
```
array([[85, 14],
       [21, 34]], dtype=int64)
```

[128]:
```
accuracy_score(y_test,prediction)
```

[128]:
```
0.7727272727272727
```

[129]:
```
report=classification_report(y_test,prediction)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.80      0.86      0.83        99
           1       0.71      0.62      0.66        55

    accuracy                           0.77       154
```

```
   macro avg       0.76      0.74      0.74       154
weighted avg       0.77      0.77      0.77       154
```

Gaussian Naive Bayes ACCURACY IS 77%

6.0.1 CONCLUSION

From the above machine learning models for dependent variable as "Outcome", accuracy score of each models are:

LOGISIC REGRESSION = 87%

DECISION TREE=78%

RANDOM FOREST=81%

KNN =77%

Gaussian Naive Bayes =77%

Therefore, to predict the people having diabetes best model is LOGISTIC REGRESSION with 87% accuracy