```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
data=pd.read_csv("health care diabetes.csv")
```

```
data.head()
```

```
   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
```

```
data.shape
```

```
(768, 9)
```

```
data.columns
```

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

```
data.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
```

```
data.describe().T
```

```
                          count        mean        std     min
25%  \
Pregnancies               768.0    3.845052   3.369578   0.000
1.000000
Glucose                   768.0  121.681605  30.436016  44.000
99.750000
BloodPressure             768.0   72.254807  12.115932  24.000
64.000000
SkinThickness             768.0   26.606479   9.631241   7.000
20.536458
Insulin                   768.0  118.660163  93.080358  14.000
79.799479
BMI                       768.0   32.450805   6.875374  18.200
27.500000
DiabetesPedigreeFunction  768.0    0.471876   0.331329   0.078
0.243750
Age                       768.0   33.240885  11.760232  21.000
24.000000
Outcome                   768.0    0.348958   0.476951   0.000
0.000000

                                 50%        75%     max
Pregnancies                 3.000000    6.00000   17.00
Glucose                   117.000000  140.25000  199.00
BloodPressure              72.000000   80.00000  122.00
SkinThickness              23.000000   32.00000   99.00
Insulin                    79.799479  127.25000  846.00
BMI                        32.000000   36.60000   67.10
DiabetesPedigreeFunction    0.372500    0.62625    2.42
Age                        29.000000   41.00000   81.00
Outcome                     0.000000    1.00000    1.00
```

```
data["Glucose"].unique()
```

```
array([148,  85, 183,  89, 137, 116,  78, 115, 197, 125, 110, 168,
139,
       189, 166, 100, 118, 107, 103, 126,  99, 196, 119, 143, 147,
97,
       145, 117, 109, 158,  88,  92, 122, 138, 102,  90, 111, 180,
133,
       106, 171, 159, 146,  71, 105, 101, 176, 150,  73, 187,  84,
```

```
44,
       141, 114,  95, 129,  79,   0,  62, 131, 112, 113,  74,  83,
136,
        80, 123,  81, 134, 142, 144,  93, 163, 151,  96, 155,  76,
160,
       124, 162, 132, 120, 173, 170, 128, 108, 154,  57, 156, 153,
188,
       152, 104,  87,  75, 179, 130, 194, 181, 135, 184, 140, 177,
164,
        91, 165,  86, 193, 191, 161, 167,  77, 182, 157, 178,  61,
98,
       127,  82,  72, 172,  94, 175, 195,  68, 186, 198, 121,  67,
174,
       199,  56, 169, 149,  65, 190], dtype=int64)
```

```
data["Glucose"].value_counts()
```

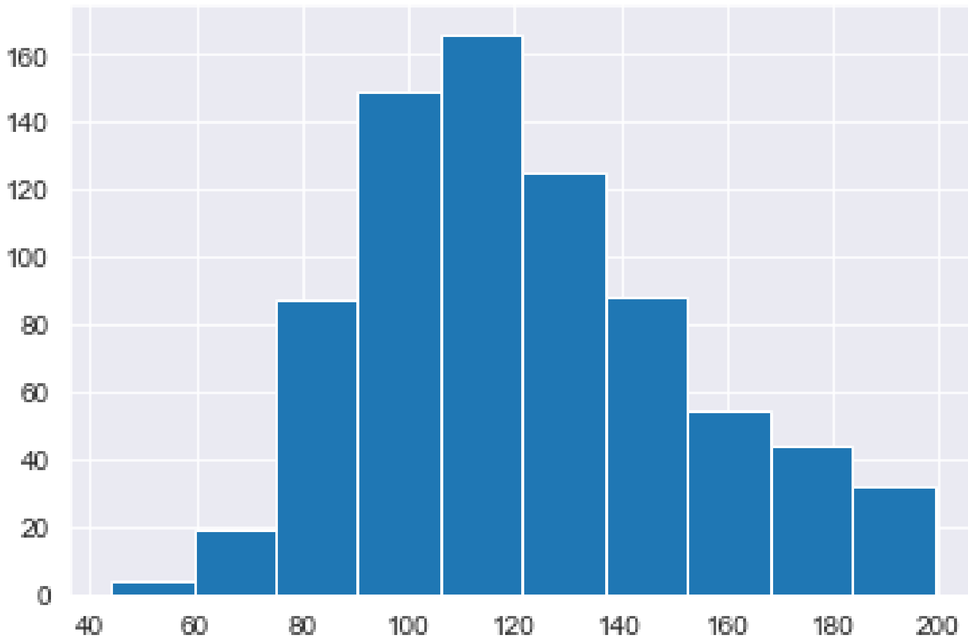
```
99     17
100    17
111    14
129    14
125    14
       ..
191     1
177     1
44      1
62      1
190     1
Name: Glucose, Length: 136, dtype: int64
```

```
data[data["Glucose"]==0].count()
```

```
Pregnancies                 5
Glucose                     5
BloodPressure               5
SkinThickness               5
Insulin                     5
BMI                         5
DiabetesPedigreeFunction    5
Age                         5
Outcome                     5
dtype: int64
```

```
sns.set_style(style='darkgrid')
plt.hist(data["Glucose"])
```

```
(array([  4.,  19.,  87., 149., 166., 125.,  88.,  54.,  44.,  32.]),
 array([ 44. ,  59.5,  75. ,  90.5, 106. , 121.5, 137. , 152.5, 168. ,
        183.5, 199. ]),
 <a list of 10 Patch objects>)
```




```
data["Glucose"].mean()
```

```
120.89453125
```

```
data["Glucose"]=data["Glucose"].replace(0,data["Glucose"].mean())
```

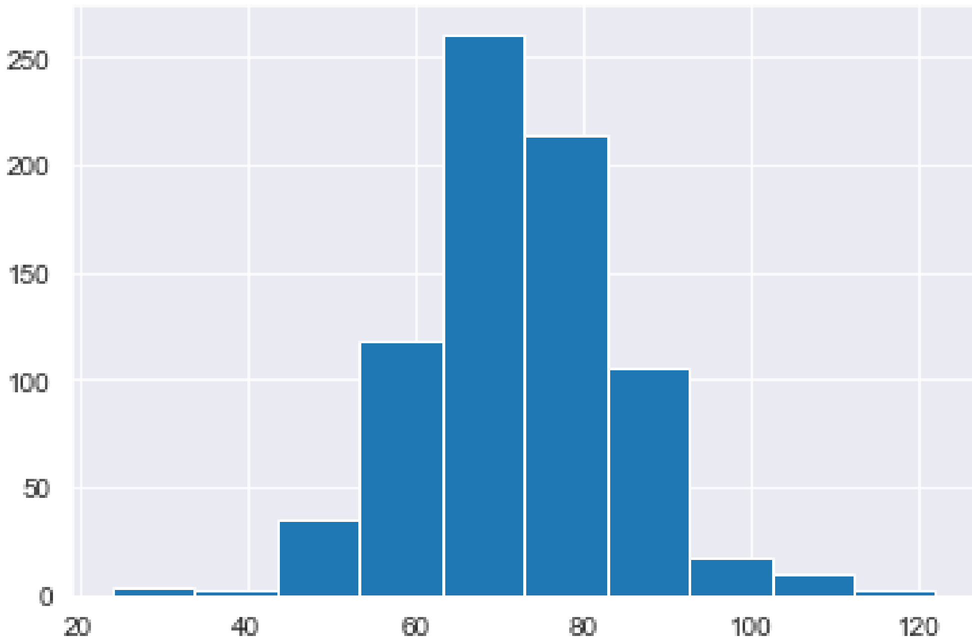
As most of data["Glucose"] is 120 and in histogram so i am repacing 0 values with mean

```
data['BloodPressure'].mean()
```

```
69.10546875
```

```
sns.set_style(style='darkgrid')
plt.hist(data['BloodPressure'])
```

```
(array([  3.,   2.,  35., 118., 261., 214., 105.,  18.,  10.,   2.]),
 array([ 24. ,  33.8,  43.6,  53.4,  63.2,  73. ,  82.8,  92.6, 102.4,
        112.2, 122. ]),
 <a list of 10 Patch objects>)
```




```
data['BloodPressure']=data['BloodPressure'].replace(0,data['BloodPress
ure'].mean())
```

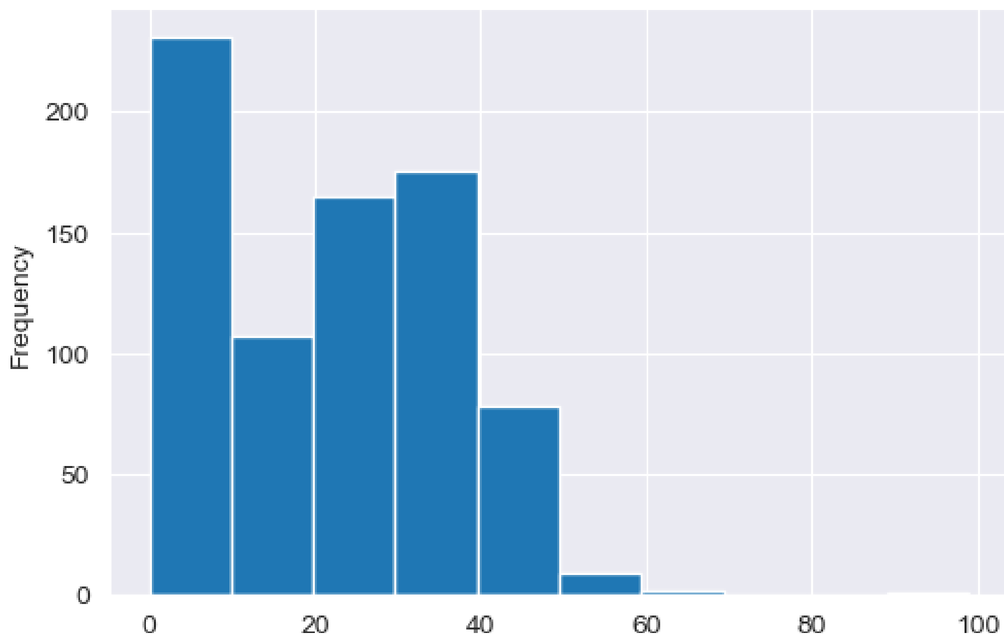
```
data["SkinThickness"].mean()
```

```
20.536458333333332
```

```
plt.figure(figsize=(6,4),dpi=100)
sns.set_style(style='darkgrid')
```

```
data["SkinThickness"].plot.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d5c337ec08>
```



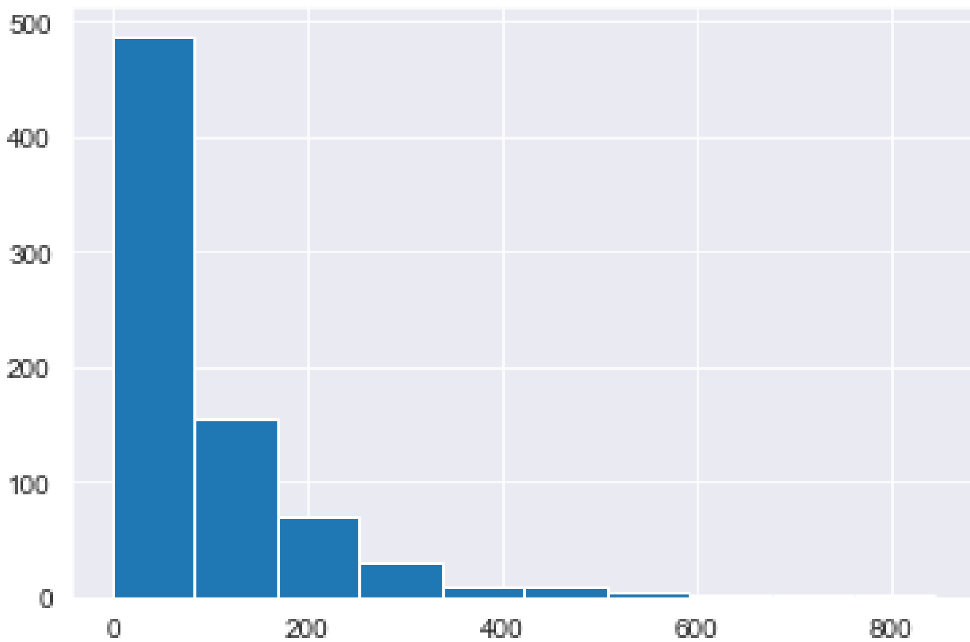

```
data['SkinThickness']=data['SkinThickness'].replace(0,data['SkinThickn
ess'].mean())
```

```
data["Insulin"].value_counts().head()
```

```
0      374
105     11
130      9
140      9
120      8
Name: Insulin, dtype: int64
```

```
sns.set_style(style='darkgrid')
plt.hist(data["Insulin"])
```

```
(array([487., 155.,  70.,  30.,   8.,   9.,   5.,   1.,   2.,   1.]),
 array([  0. ,  84.6, 169.2, 253.8, 338.4, 423. , 507.6, 592.2, 676.8,
        761.4, 846. ]),
 <a list of 10 Patch objects>)
```




```
data["Insulin"].mean()
```

```
79.79947916666667
```

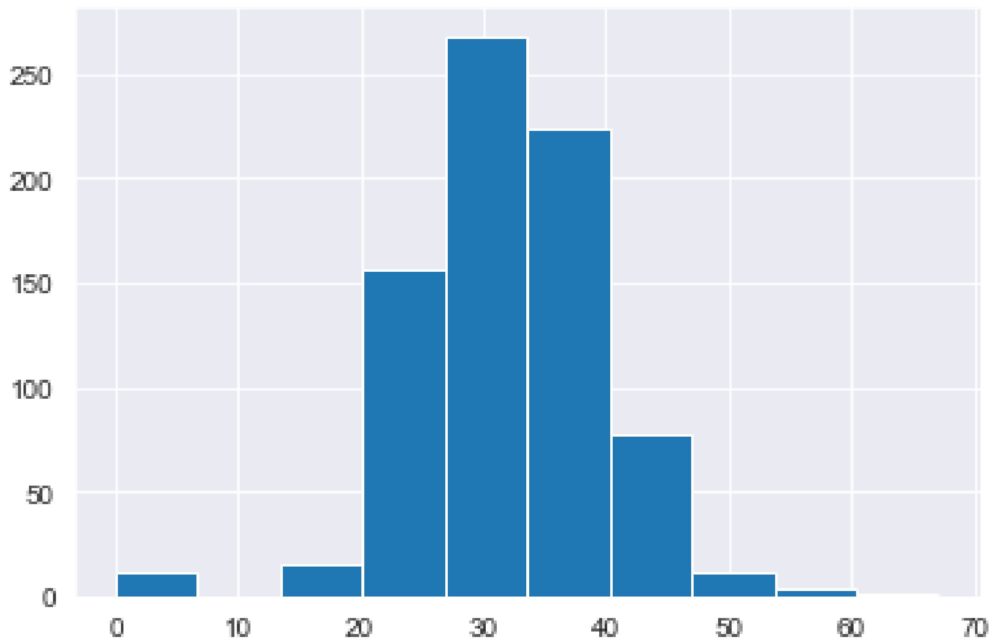
```
data["Insulin"]=data["Insulin"].replace(0,data["Insulin"].mean())
```

```
data["BMI"].value_counts().head()
```

```
32.0    13
31.6    12
31.2    12
0.0     11
32.4    10
Name: BMI, dtype: int64
```

```
sns.set_style(style='darkgrid')
plt.hist(data["BMI"])
```

```
(array([ 11.,   0.,  15., 156., 268., 224.,  78.,  12.,   3.,   1.]),
 array([ 0.  ,  6.71, 13.42, 20.13, 26.84, 33.55, 40.26, 46.97, 53.68,
        60.39, 67.1 ]),
 <a list of 10 Patch objects>)
```




```
data["BMI"].mean()
```

```
31.992578124999977
```

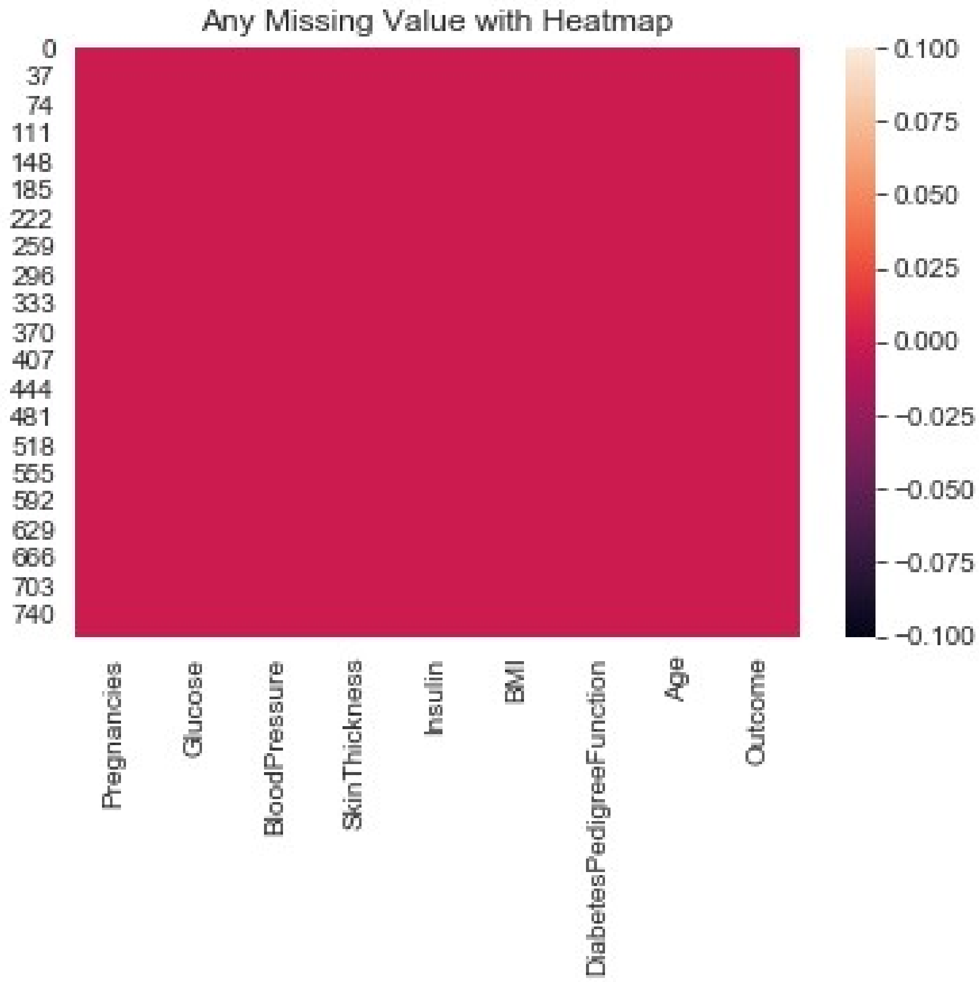
```
data["BMI"]=data["BMI"].replace(0,data["BMI"].mean())
```

```
data["Outcome"].value_counts()
```

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

```
plt.title('Any Missing Value with Heatmap')
sns.heatmap(data.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d5c4856888>
```

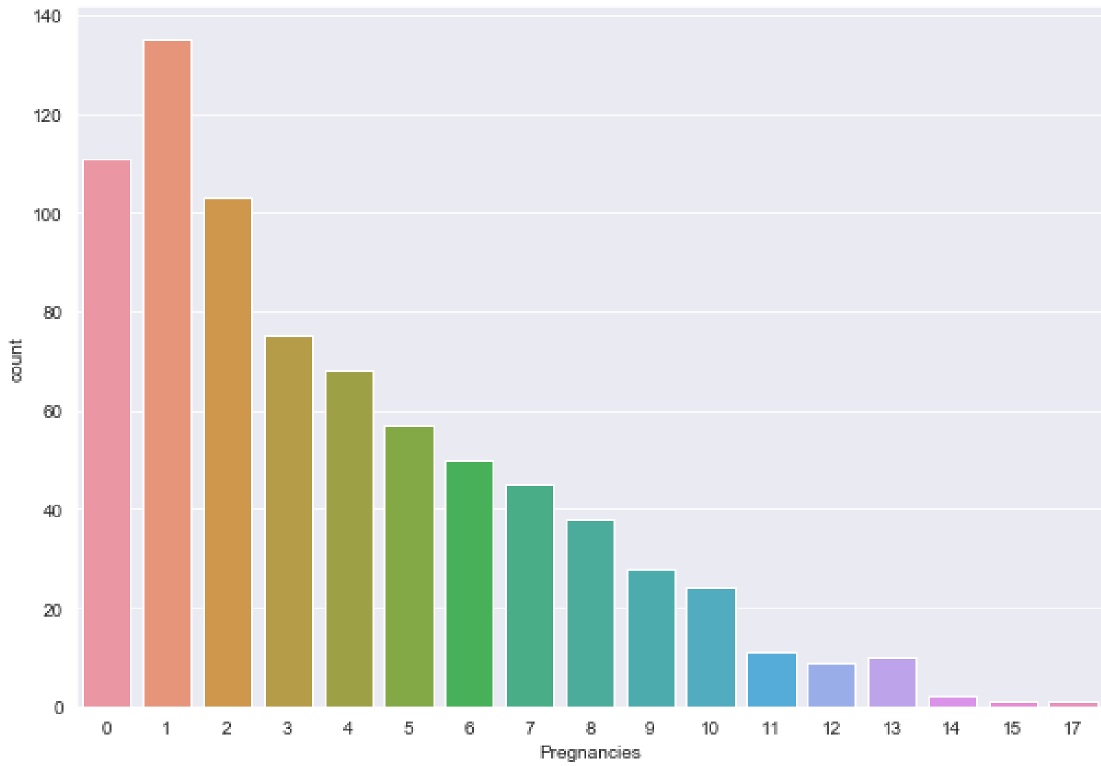



```
data.columns
```

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

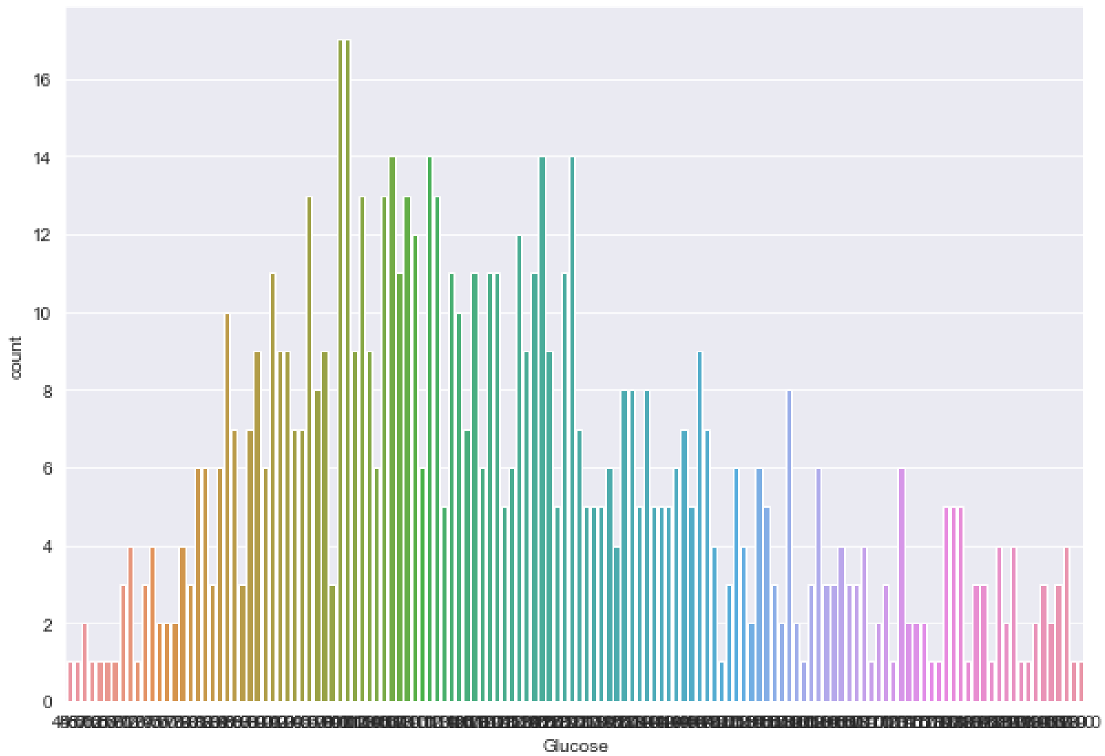
```
plt.figure(figsize=(10,7))
sns.countplot(x='Pregnancies',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d5c4e24108>
```



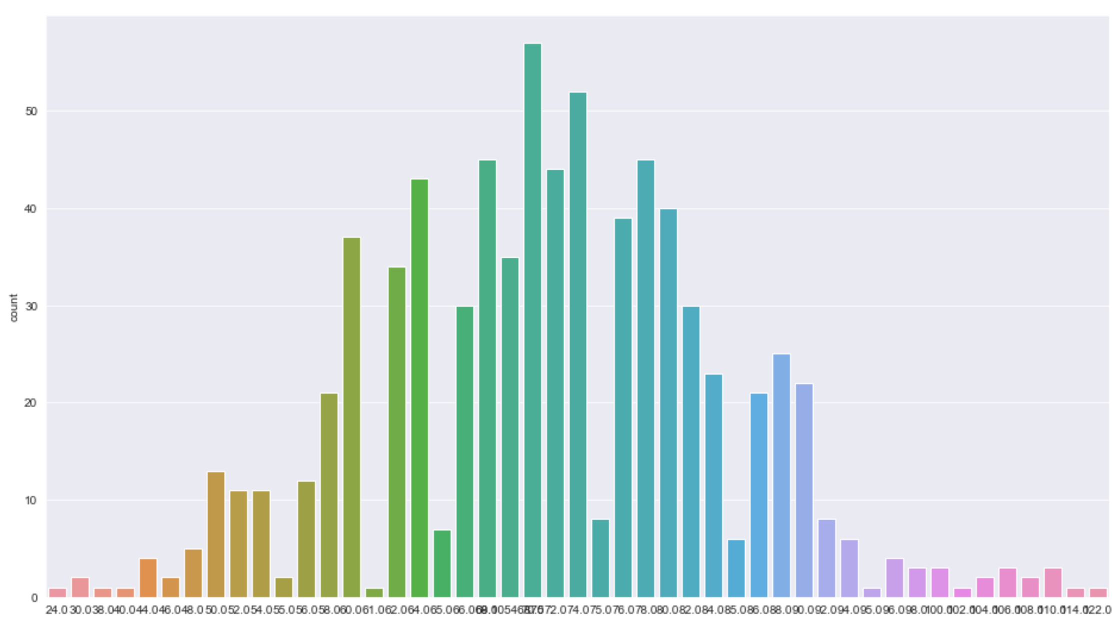

```
plt.figure(figsize=(10,7))
sns.countplot(x='Glucose',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d5c4ee91c8>
```



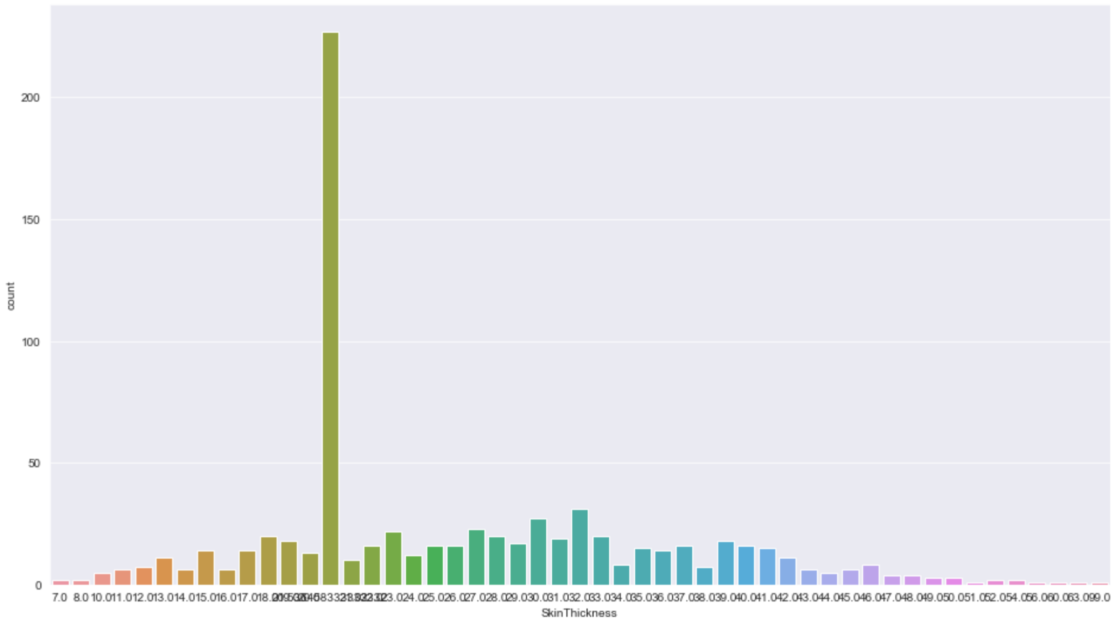

```
plt.figure(figsize=(16,9))
sns.countplot(x='BloodPressure',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d5c4e85ec8>
```



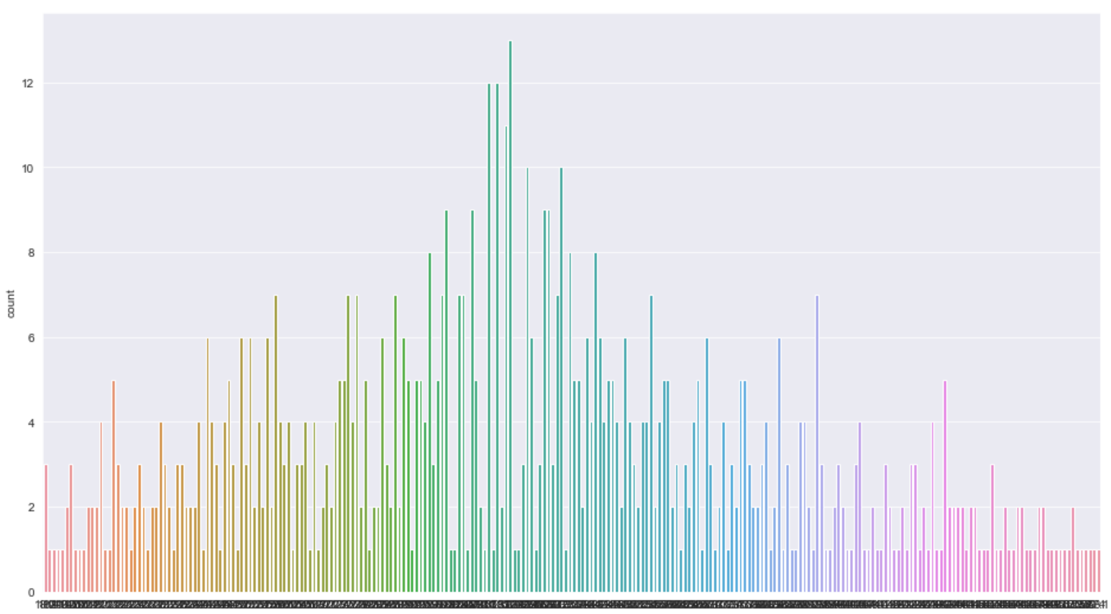

```
plt.figure(figsize=(16,9))
sns.countplot(x='SkinThickness',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d5c4ea5108>
```



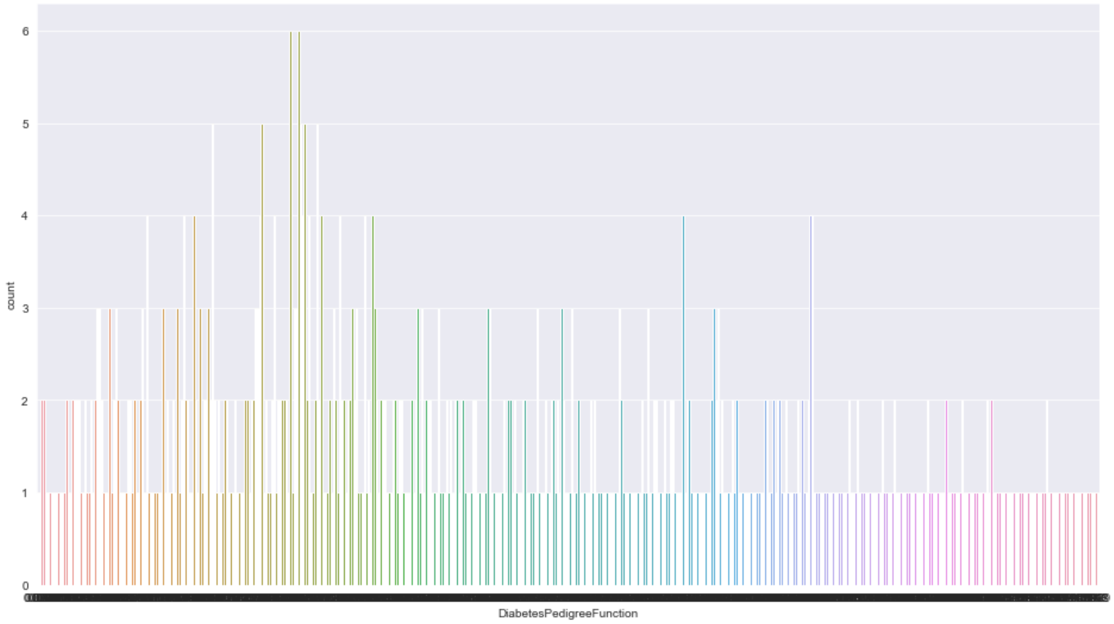

```
plt.figure(figsize=(16,9))
sns.countplot(x='BMI',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d5c63b8cc8>
```



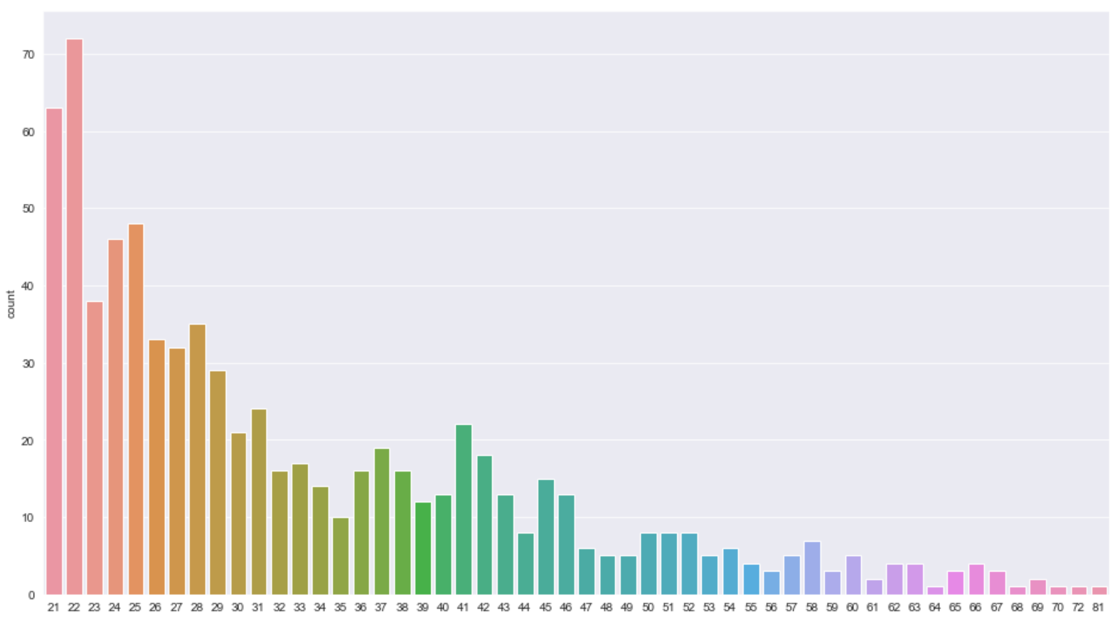

```
plt.figure(figsize=(16,9))
sns.countplot(x='DiabetesPedigreeFunction',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d5c6960b48>
```



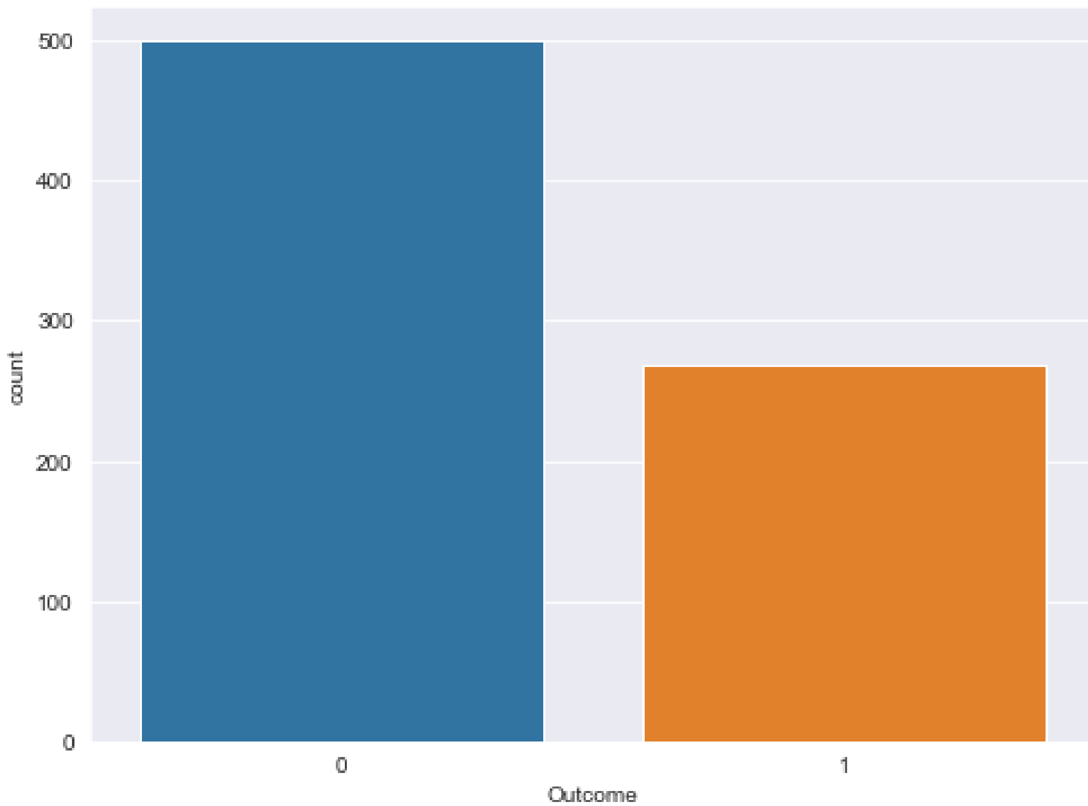

```
plt.figure(figsize=(16,9))
sns.countplot(x='Age',data=data)
```

<matplotlib.axes._subplots.AxesSubplot at 0x1d5c76ccc08>




```
plt.figure(figsize=(8,6))
sns.countplot(x='Outcome',data=data)
```

<matplotlib.axes._subplots.AxesSubplot at 0x1d5ccad70c8>



Its visible that outcome is balanced so need of doing any sampling

```
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x1d5cd53d1c8>
```

In BMI /SkinThickness and Age/Pregnancies there seems to be positive correlation

```
data.columns
```

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

Now lets see correlation among features

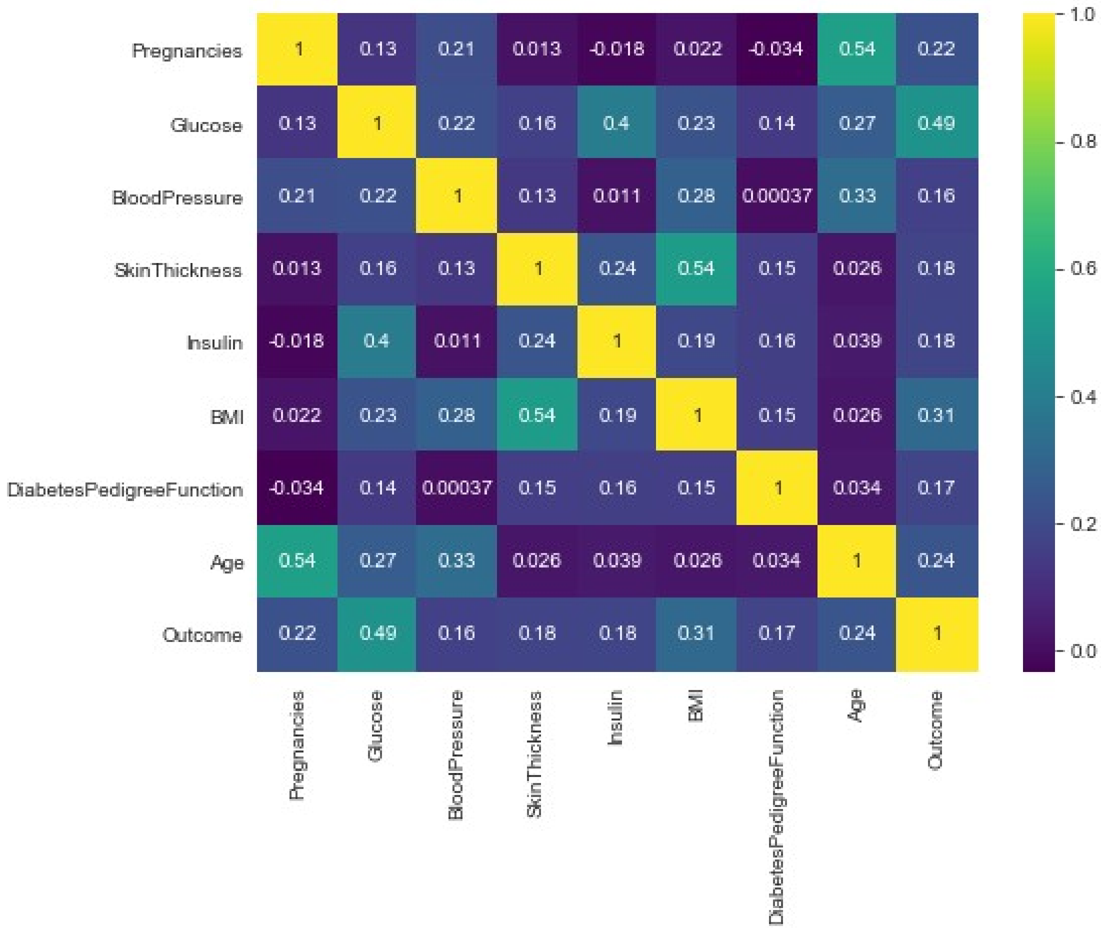
```
data.corr().sort_values(by="Outcome")   #sorting by Outcome
```

```
                          Pregnancies   Glucose  BloodPressure
SkinThickness  \
BloodPressure                0.208984  0.219666       1.000000
0.134155
DiabetesPedigreeFunction    -0.033523  0.137106       0.000371
```

```
0.154961
SkinThickness                   0.013376  0.160766       0.134155
1.000000
Insulin                        -0.018082  0.396597       0.010926
0.240361
Pregnancies                     1.000000  0.127964       0.208984
0.013376
Age                             0.544341  0.266600       0.326740
0.026423
BMI                             0.021546  0.231478       0.281231
0.535703
Glucose                         0.127964  1.000000       0.219666
0.160766
Outcome                         0.221898  0.492908       0.162986
0.175026

                           Insulin       BMI  DiabetesPedigreeFunction
\
BloodPressure             0.010926  0.281231                  0.000371

DiabetesPedigreeFunction  0.157806  0.153508                  1.000000

SkinThickness             0.240361  0.535703                  0.154961

Insulin                   1.000000  0.189856                  0.157806

Pregnancies              -0.018082  0.021546                 -0.033523

Age                       0.038652  0.025748                  0.033561

BMI                       0.189856  1.000000                  0.153508

Glucose                   0.396597  0.231478                  0.137106

Outcome                   0.179185  0.312254                  0.173844


                               Age   Outcome
BloodPressure             0.326740  0.162986
DiabetesPedigreeFunction  0.033561  0.173844
SkinThickness             0.026423  0.175026
Insulin                   0.038652  0.179185
Pregnancies               0.544341  0.221898
Age                       1.000000  0.238356
BMI                       0.025748  0.312254
Glucose                   0.266600  0.492908
Outcome                   0.238356  1.000000
```

```
plt.figure(figsize=(8,6))
sns.heatmap(data.corr(),cmap="viridis",annot=True)
```

<matplotlib.axes._subplots.AxesSubplot at 0x1d5d673d188>



Week 3

```
X=data.drop("Outcome",axis=1)
y=data["Outcome"]
```

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

```
print(X_train.shape)
print(X_test.shape)
```

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

```
print(y_train.shape)
print(y_test.shape)
```

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

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
```

```
X_std=scaler.fit_transform(X_train)
```

```
Xt_std=scaler.transform(X_test)
```

```
from sklearn.metrics import confusion_matrix, accuracy_score,
classification_report, roc_curve, RocCurveDisplay,auc
```

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
m=model.fit(X_std,y_train)
predicted= model.predict(Xt_std)
print(predicted)
```

```
[1 0 0 0 0 0 1 1 1 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 0 1 1 0 1 1 1 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 1 0 0 0
0 1
 1 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0
1 0
 0 1 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0
0 0
 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0
1 1
 0 1 0 0 0 1 0]
```

```
print(classification_report(y_pred=predicted, y_true=y_test))
```

```
              precision    recall  f1-score   support

           0       0.78      0.83      0.81       130
           1       0.59      0.52      0.55        62

    accuracy                           0.73       192
   macro avg       0.69      0.67      0.68       192
weighted avg       0.72      0.73      0.72       192
```

```
accuracy_score(y_pred=predicted, y_true=y_test)
```

```
0.7291666666666666
```

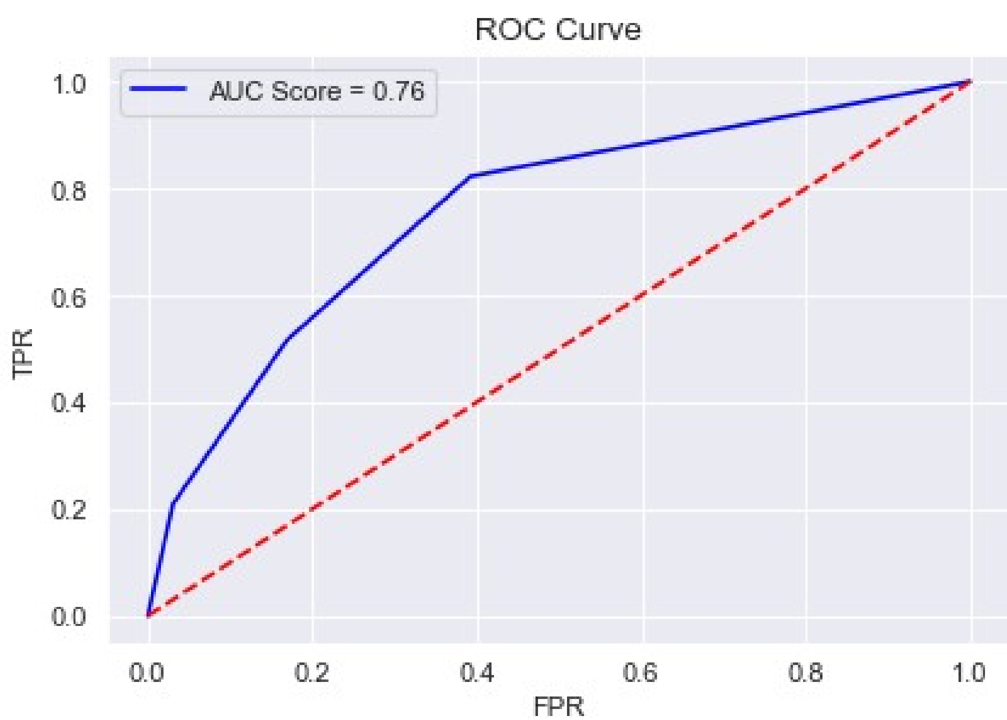
```
confusion_matrix(y_pred=predicted, y_true=y_test)
```

```
array([[108,  22],
       [ 30,  32]], dtype=int64)
```

```
pb=model.predict_proba(Xt_std)
b=pb[:,1]
```

```
fpr, tpr, thresholds = roc_curve(y_test,b)
roc_auc = auc(fpr, tpr)
plt.figure(dpi=80)
plt.title("ROC Curve")
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc)
plt.plot(fpr,fpr,'r--',color='red')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1d5d7cc6bc8>
```



SVC

```
from sklearn.svm import SVC
svc_model_linear =
SVC(kernel='linear',random_state=0,probability=True,C=0.01)
```

```
svc_model_linear.fit(X_std,y_train)
pred=svc_model_linear.predict(Xt_std)
```

```
print(classification_report(y_pred=pred, y_true=y_test))
```

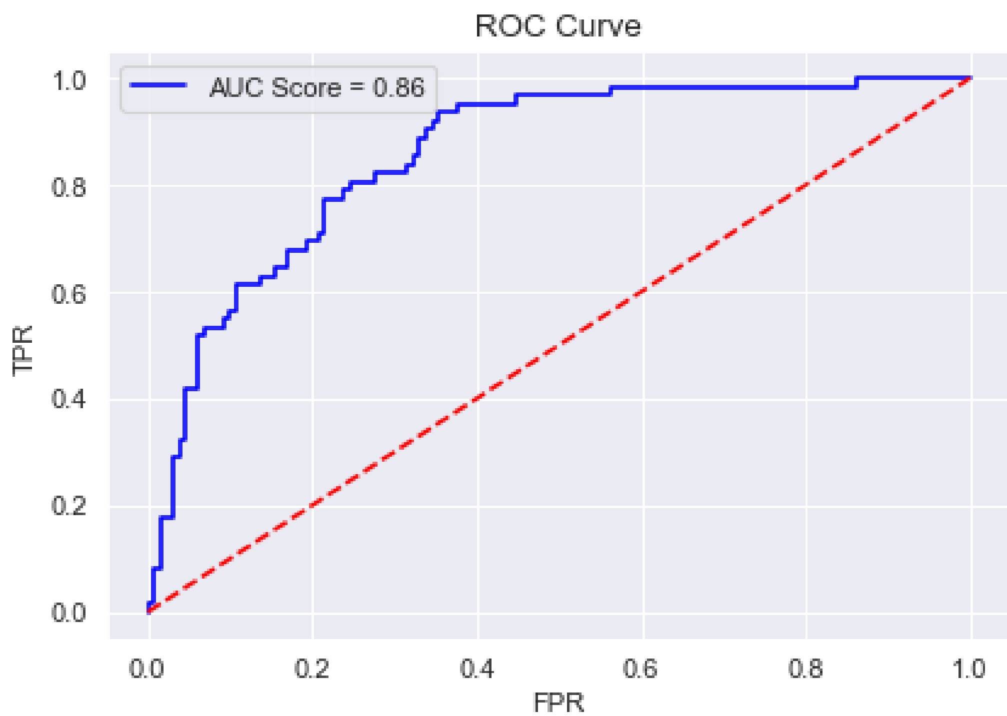
```
              precision    recall  f1-score   support

           0       0.80      0.93      0.86       130
           1       0.78      0.52      0.62        62

    accuracy                           0.80       192
   macro avg       0.79      0.72      0.74       192
weighted avg       0.79      0.80      0.78       192
```

```
print(accuracy_score(y_pred=pred, y_true=y_test))
print(confusion_matrix(y_pred=pred, y_true=y_test))
```

```
0.796875
[[121   9]
 [ 30  32]]
```

```
svc_prob=svc_model_linear.predict_proba(Xt_std)
svc_prob_linear1=svc_prob[:,1]
fpr,tpr,thresh=roc_curve(y_test,svc_prob_linear1)
roc_auc_svc=auc(fpr,tpr)
plt.figure(dpi=80)
plt.title("ROC Curve")
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_svc)
plt.plot(fpr,fpr,'r--',color='red')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1d5d7d42788>
```



LogisticRegression

```
from sklearn.linear_model import LogisticRegression

l=LogisticRegression(fit_intercept=True)

lf=l.fit(X_std,y_train)

p=lf.predict(Xt_std)

print(classification_report(y_pred=p, y_true=y_test))
```

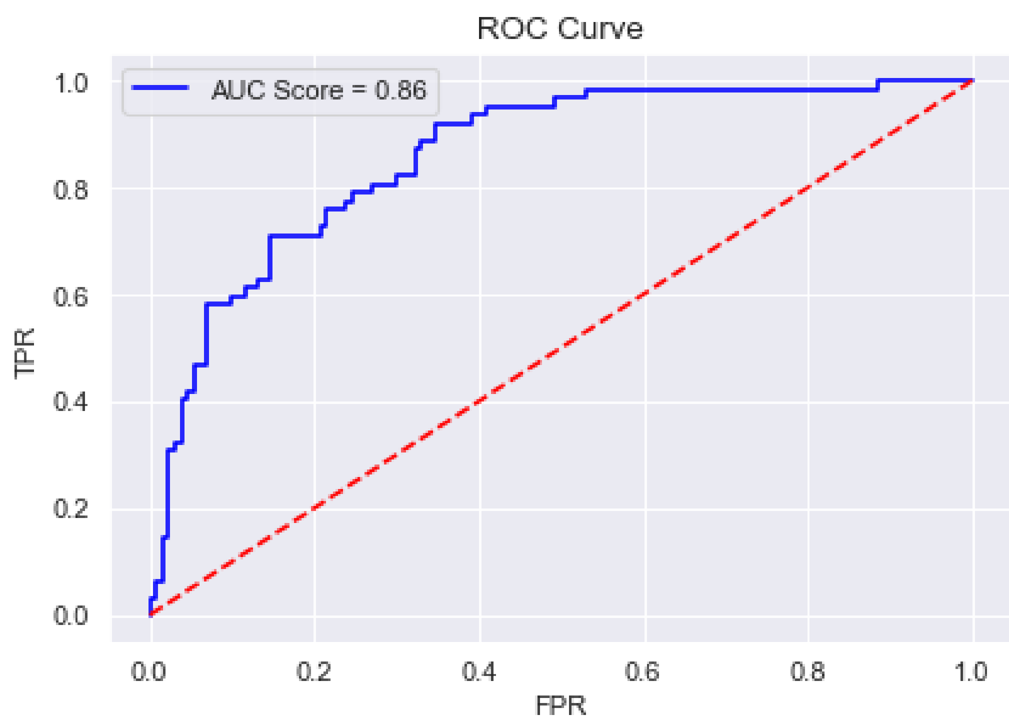
```
              precision    recall  f1-score   support

           0       0.82      0.91      0.86       130
           1       0.75      0.58      0.65        62

    accuracy                           0.80       192
   macro avg       0.78      0.74      0.76       192
weighted avg       0.80      0.80      0.79       192
```

```
print(accuracy_score(y_pred=p, y_true=y_test))
print(confusion_matrix(y_pred=p, y_true=y_test))
```

```
0.8020833333333334
[[118  12]
 [ 26  36]]
```

```
prob=lf.predict_proba(Xt_std)
prob_linear1=prob[:,1]
fpr,tpr,thresh=roc_curve(y_test,prob_linear1)
roc_auc_svc=auc(fpr,tpr)
plt.figure(dpi=80)
plt.title("ROC Curve")
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_svc)
plt.plot(fpr,fpr,'r--',color='red')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1d5d7e08f08>
```



RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=1000,random_state=0)
rf_model.fit(X_std,y_train)
rf_pred=rf_model.predict(Xt_std)
```

```
print(accuracy_score(y_test,rf_pred))
```

```
print(classification_report(y_test,rf_pred),'\n')

rf_prob=rf_model.predict_proba(Xt_std)
rf_prob1=rf_prob[:,1]
fpr,tpr,thresh=roc_curve(y_test,rf_prob1)
roc_auc_rf=auc(fpr,tpr)
plt.figure(dpi=80)
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_rf)
plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot(fpr,fpr,'r--',color='red')
plt.legend()
```

```
0.78125
              precision    recall  f1-score   support

           0       0.81      0.88      0.85       130
           1       0.70      0.56      0.62        62

    accuracy                           0.78       192
   macro avg       0.75      0.72      0.74       192
weighted avg       0.77      0.78      0.77       192


<matplotlib.legend.Legend at 0x1d5d8fb85c8>
```