In [13]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn import metrics
from matplotlib import style
```

In [14]:
```
dataset = pd.read_csv("health care diabetes.csv")
```

In [15]:
```
dataset.head()
```

Out[15]:

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

In [16]:
```
dataset.tail()
```

Out[16]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
763	10	101	76	48	180	32.9	0.171
764	2	122	70	27	0	36.8	0.340
765	5	121	72	23	112	26.2	0.245
766	1	126	60	0	0	30.1	0.349
767	1	93	70	31	0	30.4	0.315

In [17]:
```
dataset.shape
```

Out[17]:
```
(768, 9)
```

In [18]:
```
dataset.isnull().sum()
```

Out[18]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [19]:
```
dataset.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
```

In [20]:
```
dataset.describe()
```

Out[20]:

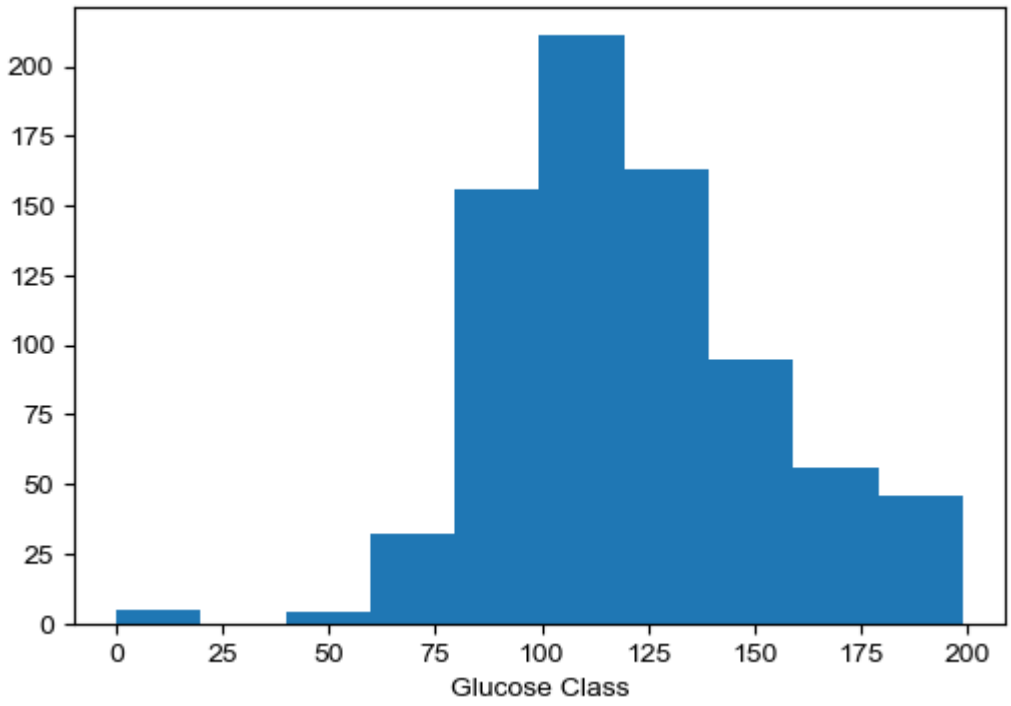
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPed
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	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	

In [24]:
```
print("Standard Deviation of each variables are ==> ")
dataset.apply(np.std)
```

Out[24]:
```
Standard Deviation of each variables are ==>
Pregnancies                 3.367384
Glucose                    31.951796
BloodPressure              19.343202
SkinThickness              15.941829
Insulin                   115.168949
BMI                         7.879026
DiabetesPedigreeFunction    0.331113
Age                        11.752573
Outcome                     0.476641
dtype: float64
```

In [26]:
```
plt.figure(figsize=(6,4),dpi=100)
plt.xlabel('Glucose Class')
plt.hist(dataset['Glucose'])
sns.set_style(style='darkgrid')
print("Mean of Glucose level is :-", dataset['Glucose'].mean())
print("Datatype of Glucose Variable is:",dataset['Glucose'].dtypes)
```

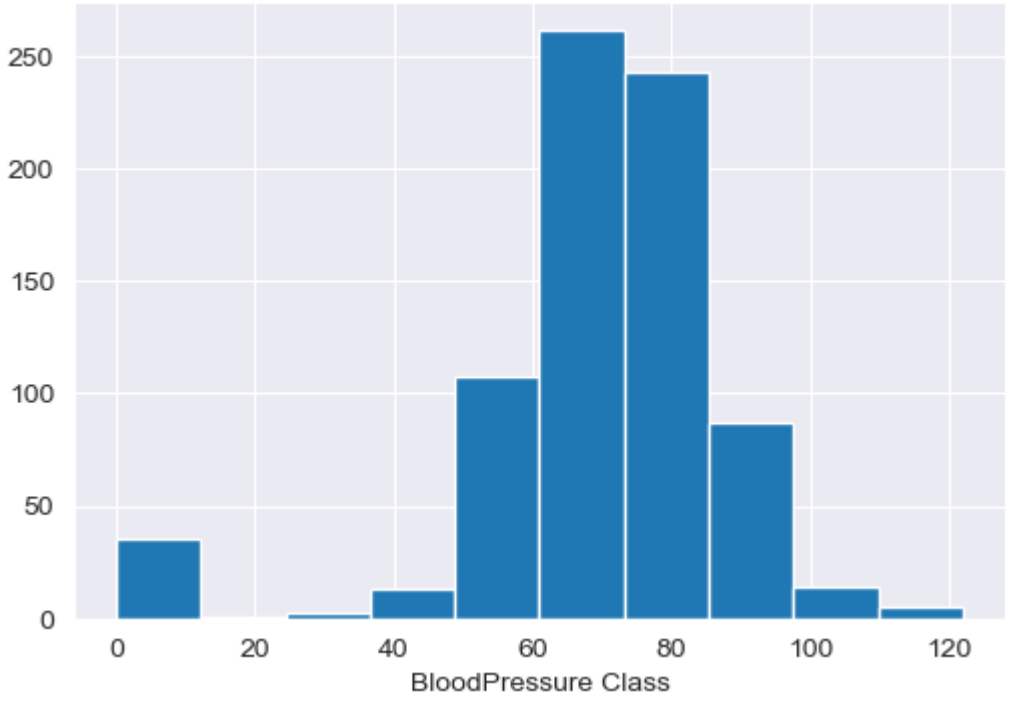
```
Mean of Glucose level is :- 120.89453125
Datatype of Glucose Variable is: int64
```



In [27]:
```
dataset['Glucose']=dataset['Glucose'].replace(0,dataset['Glucose'].mean())
```

In [28]:
```
plt.figure(figsize=(6,4),dpi=100)
plt.xlabel('BloodPressure Class')
plt.hist(dataset['BloodPressure'])
sns.set_style(style='darkgrid')
print("Mean of BloodPressure level is :-", dataset['BloodPressure'].mean())
print("Datatype of BloodPressure Variable is:",dataset['BloodPressure'].dtypes)
```

```
Mean of BloodPressure level is :- 69.10546875
Datatype of BloodPressure Variable is: int64
```

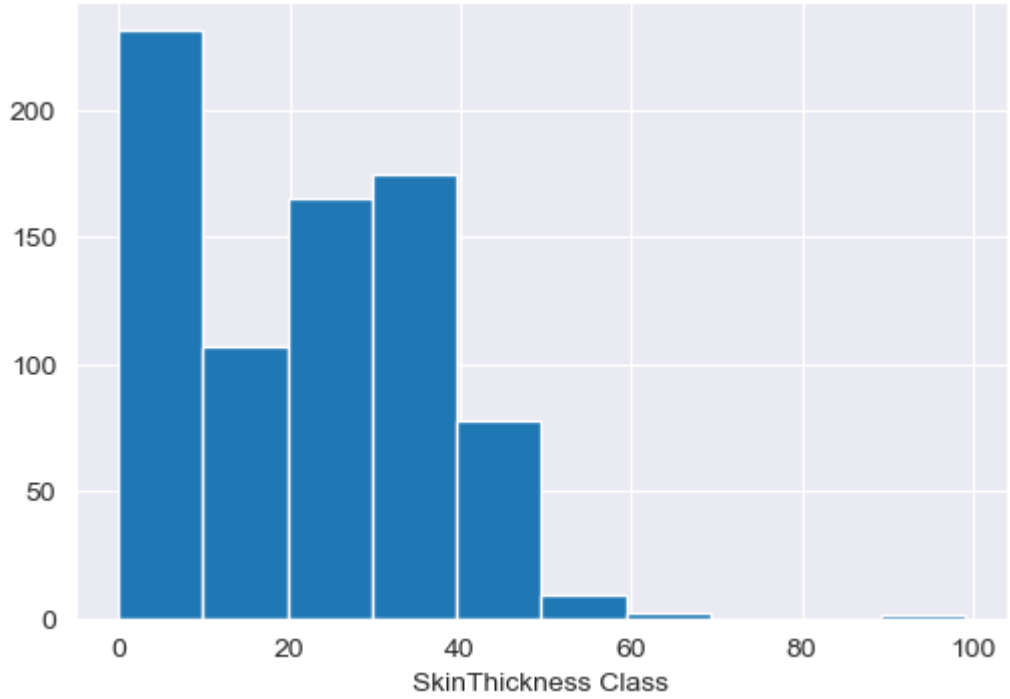


In [29]:
```
dataset['BloodPressure']=dataset['BloodPressure'].replace(0,dataset['BloodPressure'
```

In [30]:
```
plt.figure(figsize=(6,4),dpi=100)
plt.xlabel('SkinThickness Class')
plt.hist(dataset['SkinThickness'])
```

```
sns.set_style(style='darkgrid')
print("Mean of SkinThickness is :-", dataset['SkinThickness'].mean())
print("Datatype of SkinThickness Variable is:",dataset['SkinThickness'].dtypes)
```

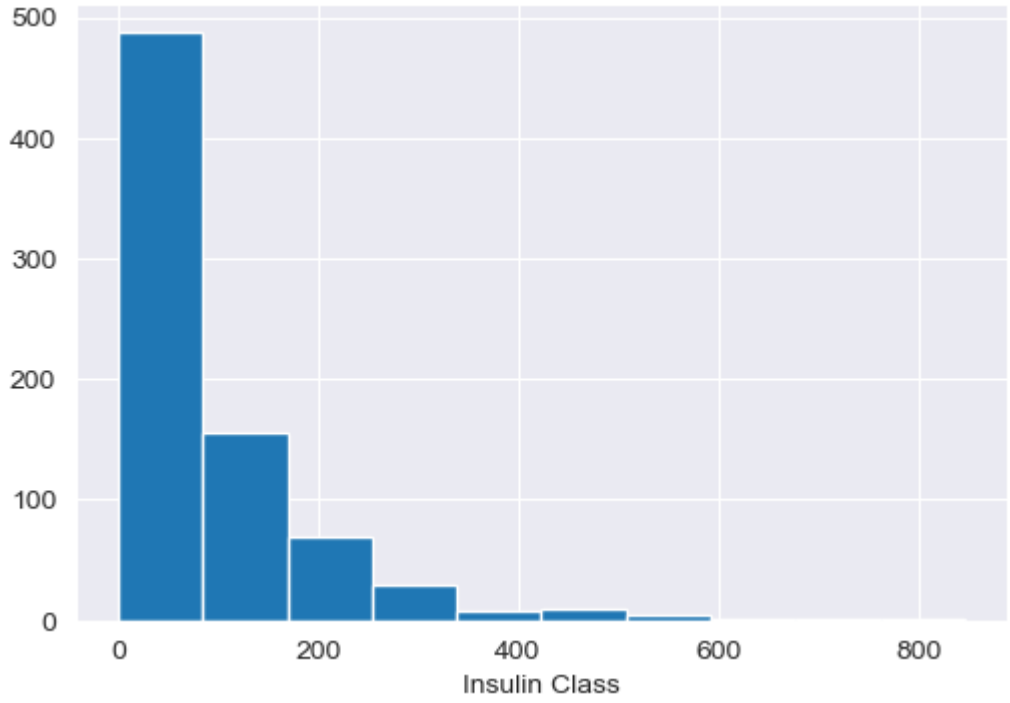
```
Mean of SkinThickness is :- 20.536458333333332
Datatype of SkinThickness Variable is: int64
```



In [31]:
```
dataset['SkinThickness']=dataset['SkinThickness'].replace(0,dataset['SkinThickness'
```

In [32]:
```
plt.figure(figsize=(6,4),dpi=100)
plt.xlabel('Insulin Class')
plt.hist(dataset['Insulin'])
sns.set_style(style='darkgrid')
print("Mean of Insulin is :-", dataset['Insulin'].mean())
print("Datatype of Insulin Variable is:",dataset['Insulin'].dtypes)
```

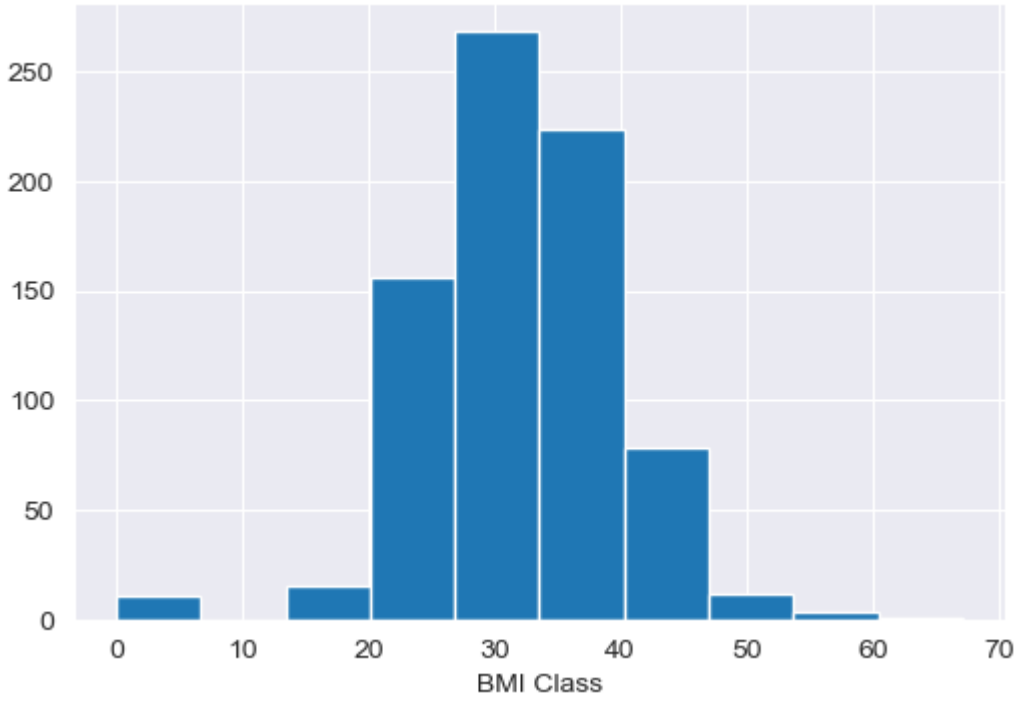
```
Mean of Insulin is :- 79.79947916666667
Datatype of Insulin Variable is: int64
```



In [33]:
```
dataset['Insulin']=dataset['Insulin'].replace(0,dataset['Insulin'].mean())
```

In [34]:
```
plt.figure(figsize=(6,4),dpi=100)
plt.xlabel('BMI Class')
plt.hist(dataset['BMI'])
sns.set_style(style='darkgrid')
print("Mean of BMI is :-", dataset['BMI'].mean())
print("Datatype of BMI Variable is:",dataset['BMI'].dtypes)
```

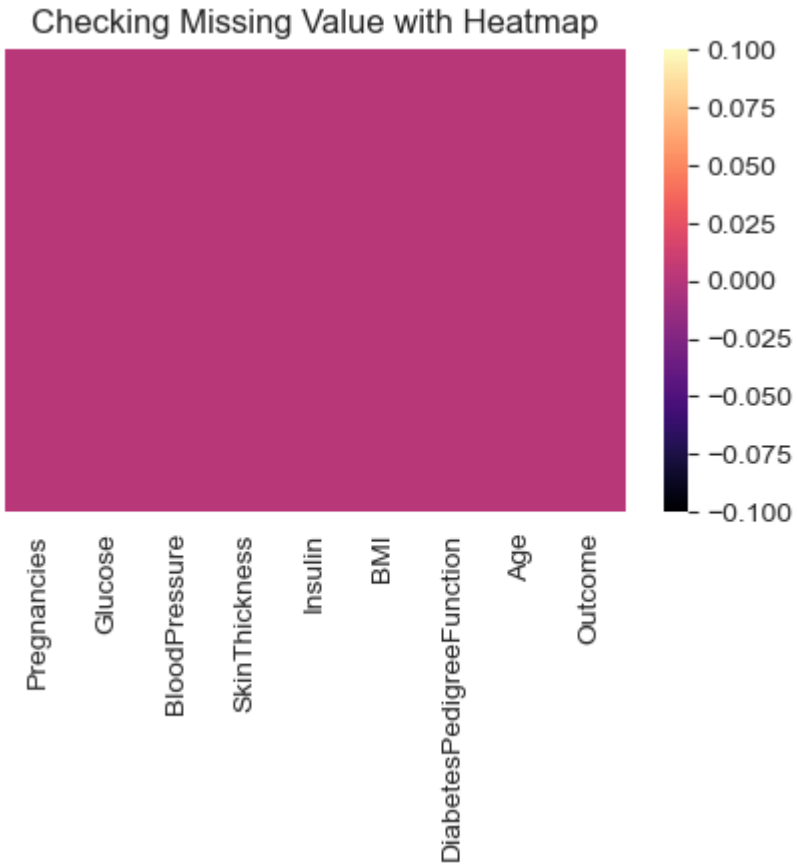
```
Mean of BMI is :- 31.992578124999998
Datatype of BMI Variable is: float64
```



In [35]:
```
dataset['BMI']=dataset['BMI'].replace(0,dataset['BMI'].mean())
```

In [36]:
```
plt.figure(figsize=(5,3),dpi=100)
plt.title('Checking Missing Value with Heatmap')
sns.heatmap(dataset.isnull(),cmap='magma',yticklabels=False)
```

Out[36]:
```
<Axes: title={'center': 'Checking Missing Value with Heatmap'}>
```



In [38]:
```
dataset.columns
```

Out[38]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [39]:
```
dataset['Glucose'].value_counts().head(10)
```

Out[39]:
```
Glucose
99.0     17
100.0    17
111.0    14
129.0    14
125.0    14
106.0    14
112.0    13
108.0    13
95.0     13
105.0    13
Name: count, dtype: int64
```

In [40]:
```
dataset['BloodPressure'].value_counts().head(10)
```

Out[40]:
```
BloodPressure
70.000000    57
74.000000    52
78.000000    45
68.000000    45
72.000000    44
64.000000    43
80.000000    40
76.000000    39
60.000000    37
69.105469    35
Name: count, dtype: int64
```

In [42]:
```
dataset['SkinThickness'].value_counts().head(10)
```

Out[42]:
```
SkinThickness
20.536458    227
32.000000     31
30.000000     27
27.000000     23
23.000000     22
33.000000     20
28.000000     20
18.000000     20
31.000000     19
19.000000     18
Name: count, dtype: int64
```

In [43]:
```
dataset['Insulin'].value_counts().head(10)
```

Out[43]:
```
Insulin
79.799479     374
105.000000     11
130.000000      9
140.000000      9
120.000000      8
94.000000       7
180.000000      7
100.000000      7
135.000000      6
115.000000      6
Name: count, dtype: int64
```

In [44]:
```
dataset['BMI'].value_counts().head(10)
```

Out[44]:
```
BMI
32.000000    13
31.600000    12
31.200000    12
31.992578    11
32.400000    10
33.300000    10
30.100000     9
32.800000     9
32.900000     9
30.800000     9
Name: count, dtype: int64
```

In [45]:
```
dataset['Age'].value_counts().head(10)
```

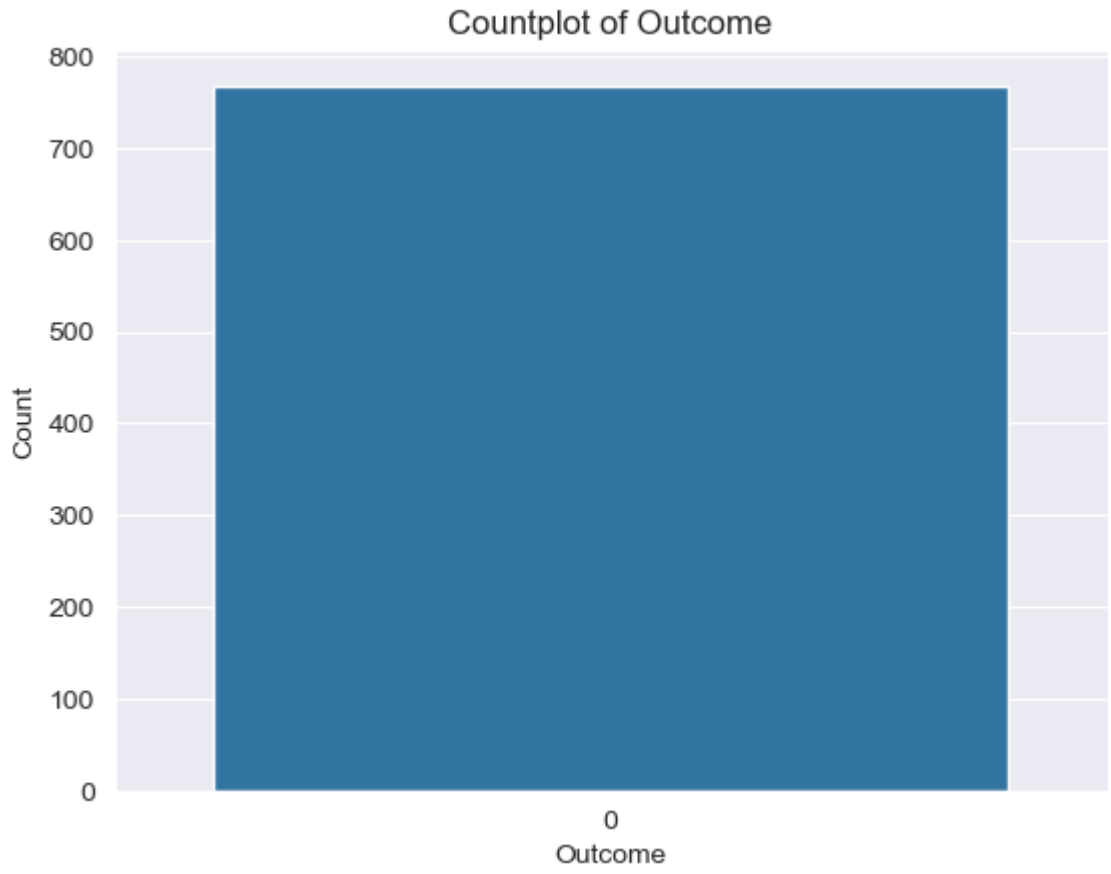
Out[45]:
```
Age
22    72
21    63
25    48
24    46
23    38
28    35
26    33
27    32
29    29
31    24
Name: count, dtype: int64
```

In [46]:
```
dataset['Outcome'].value_counts().head(10)
```

Out[46]:
```
Outcome
0    500
1    268
Name: count, dtype: int64
```

In [47]:
```
sns.set_style('darkgrid')
sns.countplot(dataset['Outcome'])
plt.title("Countplot of Outcome")
plt.xlabel('Outcome')
plt.ylabel("Count")
print("Count of class is:\n",dataset['Outcome'].value_counts())
```

```
Count of class is:
 Outcome
0    500
1    268
Name: count, dtype: int64
```



In [48]:
```
sns.pairplot(dataset)
plt.title('Scatter plot between variables')
```

```
C:\Users\Deviare User\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWar
ning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[48]:
```
Text(0.5, 1.0, 'Scatter plot between variables')
```

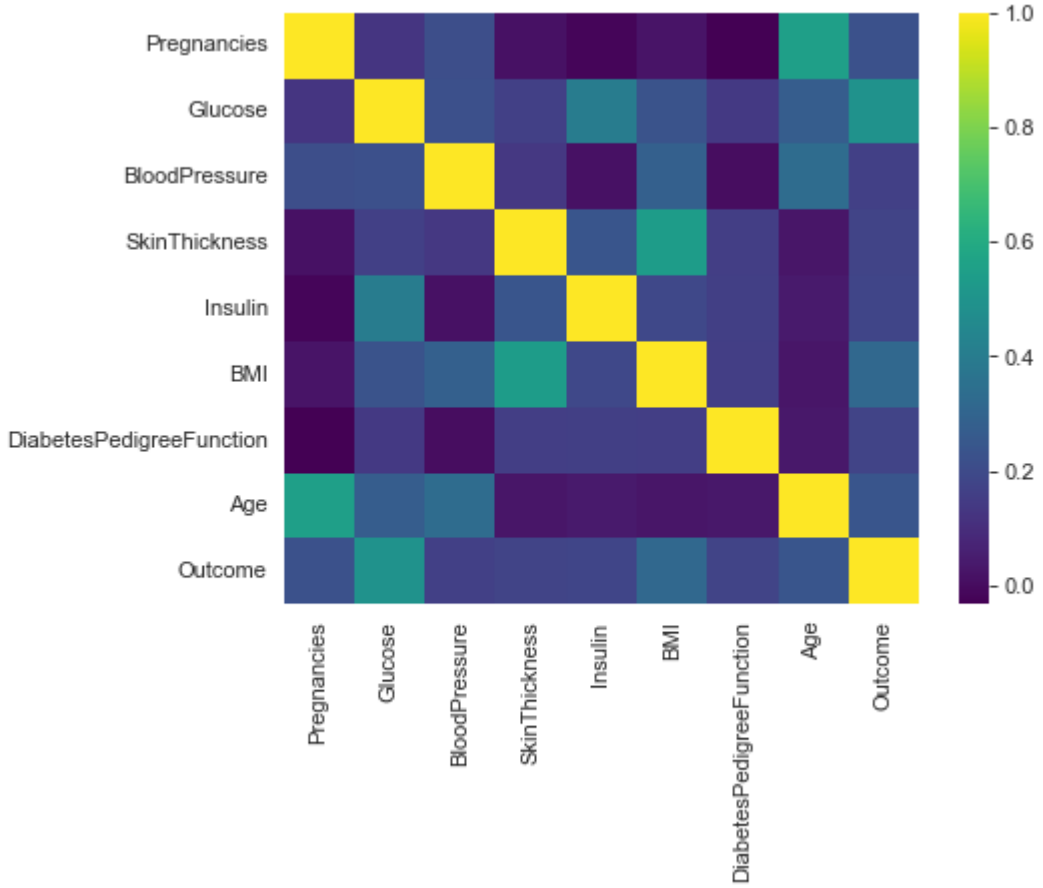
In [49]:
```
dataset.corr()
```

Out[49]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BM
Pregnancies	1.000000	0.127964	0.208984	0.013376	-0.018082	0.021546
Glucose	0.127964	1.000000	0.219666	0.160766	0.396597	0.231478
BloodPressure	0.208984	0.219666	1.000000	0.134155	0.010926	0.281231
SkinThickness	0.013376	0.160766	0.134155	1.000000	0.240361	0.535703
Insulin	-0.018082	0.396597	0.010926	0.240361	1.000000	0.189856
BMI	0.021546	0.231478	0.281231	0.535703	0.189856	1.000000
DiabetesPedigreeFunction	-0.033523	0.137106	0.000371	0.154961	0.157806	0.153508
Age	0.544341	0.266600	0.326740	0.026423	0.038652	0.025748
Outcome	0.221898	0.492908	0.162986	0.175026	0.179185	0.312254

In [50]:
```
plt.figure(dpi=80)
sns.heatmap(dataset.corr(),cmap='viridis')
```

Out[50]:
```
<Axes: >
```



In [51]:
```
x=dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values
```

In [52]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
```

In [53]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(614, 8)
(154, 8)
(614,)
(154,)
```

In [54]:
```
from sklearn.preprocessing import StandardScaler
```

In [55]:
```
Scale=StandardScaler()
x_train_std=Scale.fit_transform(x_train)
x_test_std=Scale.transform(x_test)
```

In [56]:
```
norm=lambda a:(a-min(a))/(max(a)-min(a))
```

In [58]:
```
dataset_norm=dataset.iloc[:,:-1]
```

In [59]:
```
dataset_normalized=dataset_norm.apply(norm)
```

In [61]:
```
x_train_norm,x_test_norm,y_train_norm,y_test_norm=train_test_split(dataset_normaliz
```

In [62]:
```
from sklearn.neighbors import KNeighborsClassifier
knn_model = KNeighborsClassifier(n_neighbors=25)
```

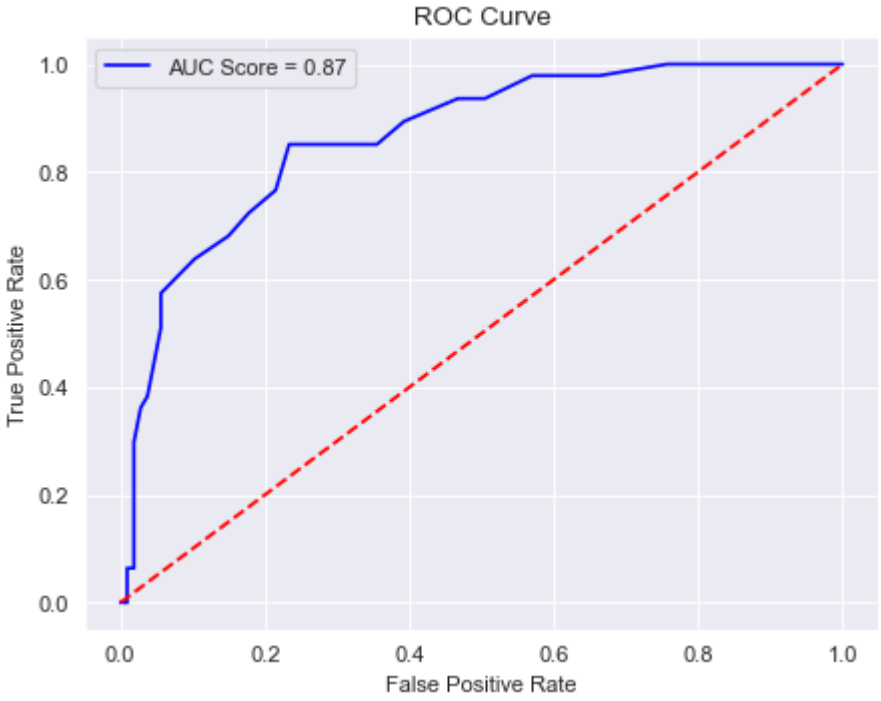
```
#Using 25 Neighbors just as thumb rule sqrt of observation
knn_model.fit(x_train_std,y_train)
knn_pred=knn_model.predict(x_test_std)
```

In [63]:
```
print("Model Validation ==>\n")
print("Accuracy Score of KNN Model::")
print(metrics.accuracy_score(y_test,knn_pred))
print("\n","Classification Report::")
print(metrics.classification_report(y_test,knn_pred),'\n')
print("\n","ROC Curve")
knn_prob=knn_model.predict_proba(x_test_std)
knn_prob1=knn_prob[:,1]
fpr,tpr,thresh=metrics.roc_curve(y_test,knn_prob1)
roc_auc_knn=metrics.auc(fpr,tpr)
plt.figure(dpi=80)
plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_knn)
plt.plot(fpr,fpr,'r--',color='red')
plt.legend()
```

```
Model Validation ==>

Accuracy Score of KNN Model::
0.8181818181818182

 Classification Report::
              precision    recall  f1-score   support

           0       0.85      0.90      0.87       107
           1       0.73      0.64      0.68        47

    accuracy                           0.82       154
   macro avg       0.79      0.77      0.78       154
weighted avg       0.81      0.82      0.81       154



 ROC Curve
```

```
C:\Users\Deviare User\AppData\Local\Temp\ipykernel_10352\2528258805.py:16: UserWar
ning: color is redundantly defined by the 'color' keyword argument and the fmt str
ing "r--" (-> color='r'). The keyword argument will take precedence.
  plt.plot(fpr,fpr,'r--',color='red')
```

Out[63]:
```
<matplotlib.legend.Legend at 0x26fef272750>
```



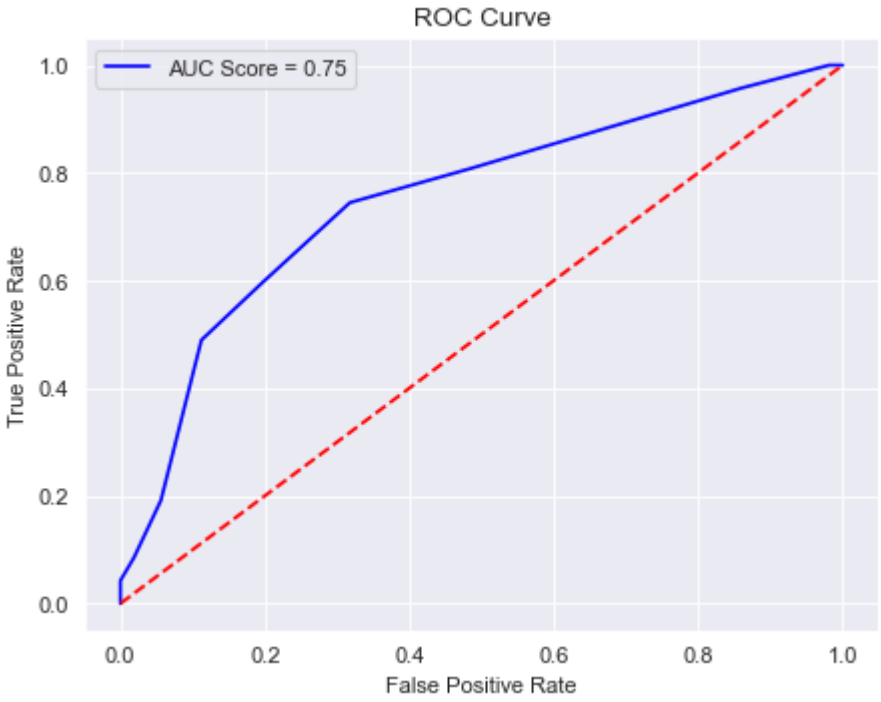
In [64]:
```
from sklearn.neighbors import KNeighborsClassifier
knn_model_norm = KNeighborsClassifier(n_neighbors=25)
knn_model_norm.fit(x_train_norm,y_train_norm)
knn_pred_norm=knn_model_norm.predict(x_test_norm)
```

In [65]:
```
print("Model Validation ==>\n")
print("Accuracy Score of KNN Model with Normalization::")
print(metrics.accuracy_score(y_test_norm,knn_pred_norm))
print("\n","Classification Report::")
print(metrics.classification_report(y_test_norm,knn_pred_norm),'\n')
print("\n","ROC Curve")
knn_prob_norm=knn_model.predict_proba(x_test_norm)
knn_prob_norm1=knn_prob_norm[:,1]
fpr,tpr,thresh=metrics.roc_curve(y_test_norm,knn_prob_norm1)
roc_auc_knn=metrics.auc(fpr,tpr)
plt.figure(dpi=80)
plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_knn)
plt.plot(fpr,fpr,'r--',color='red')
plt.legend()
```

```
Model Validation ==>

Accuracy Score of KNN Model with Normalization::
0.8311688311688312

 Classification Report::
              precision    recall  f1-score   support

           0       0.86      0.90      0.88       107
           1       0.74      0.68      0.71        47

    accuracy                           0.83       154
   macro avg       0.80      0.79      0.80       154
weighted avg       0.83      0.83      0.83       154


 ROC Curve
```

```
C:\Users\Deviare User\AppData\Local\Temp\ipykernel_10352\52957679.py:16: UserWarni
ng: color is redundantly defined by the 'color' keyword argument and the fmt strin
g "r--" (-> color='r'). The keyword argument will take precedence.
  plt.plot(fpr,fpr,'r--',color='red')
```

Out[65]:
```
<matplotlib.legend.Legend at 0x26fede8e410>
```



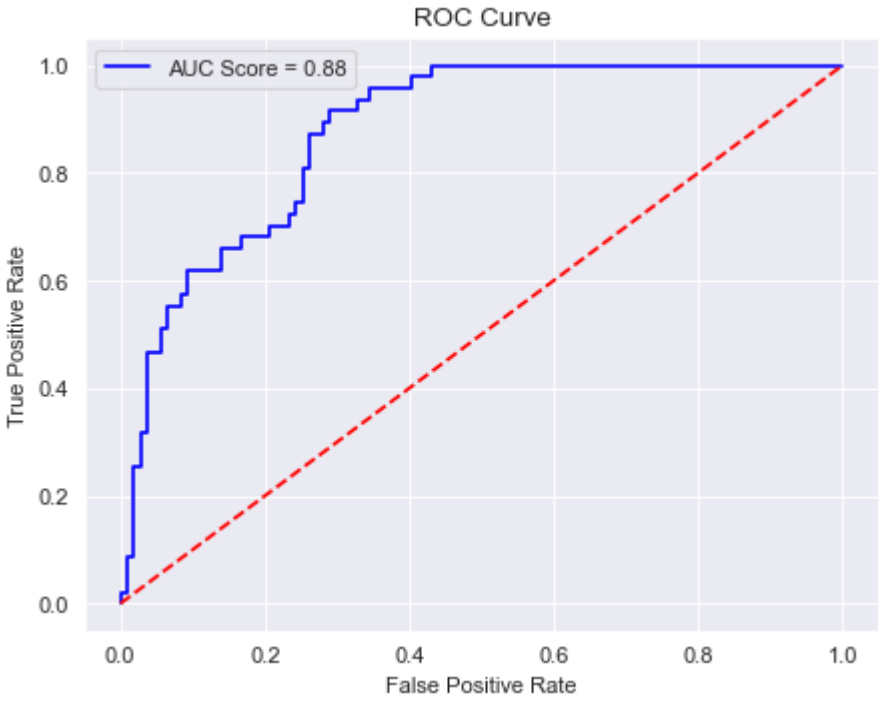
In [66]:
```
from sklearn.svm import SVC
svc_model_linear = SVC(kernel='linear',random_state=0,probability=True,C=0.01)
svc_model_linear.fit(x_train_std,y_train)
svc_pred=svc_model_linear.predict(x_test_std)
```

In [67]:
```
print("Model Validation ==>\n")
print("Accuracy Score of SVC Model with Linear Kernel::")
print(metrics.accuracy_score(y_test,svc_pred))
print("\n","Classification Report::")
print(metrics.classification_report(y_test,svc_pred),'\n')
print("\n","ROC Curve")
svc_prob_linear=svc_model_linear.predict_proba(x_test_std)
svc_prob_linear1=svc_prob_linear[:,1]
fpr,tpr,thresh=metrics.roc_curve(y_test,svc_prob_linear1)
roc_auc_svc=metrics.auc(fpr,tpr)
plt.figure(dpi=80)
plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_svc)
plt.plot(fpr,fpr,'r--',color='red')
plt.legend()
```

```
Model Validation ==>

Accuracy Score of SVC Model with Linear Kernel::
0.8116883116883117

 Classification Report::
              precision    recall  f1-score   support

           0       0.83      0.92      0.87       107
           1       0.75      0.57      0.65        47

    accuracy                           0.81       154
   macro avg       0.79      0.75      0.76       154
weighted avg       0.81      0.81      0.80       154


 ROC Curve
```

```
C:\Users\Deviare User\AppData\Local\Temp\ipykernel_10352\994548841.py:16: UserWarn
ing: color is redundantly defined by the 'color' keyword argument and the fmt stri
ng "r--" (-> color='r'). The keyword argument will take precedence.
  plt.plot(fpr,fpr,'r--',color='red')
```

Out[67]:
```
<matplotlib.legend.Legend at 0x26fef5afe10>
```



In [68]:
```
from sklearn.svm import SVC
svc_model_rbf = SVC(kernel='rbf',random_state=0,probability=True,C=1)
svc_model_rbf.fit(x_train_std,y_train)
svc_pred_rbf=svc_model_rbf.predict(x_test_std)
```

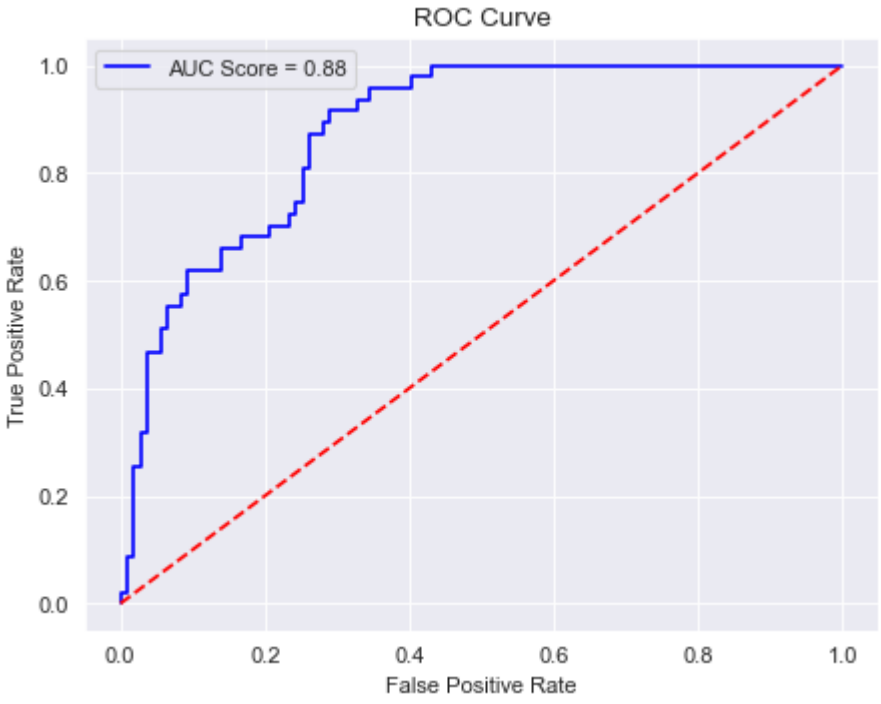
In [69]:
```
print("Model Validation ==>\n")
print("Accuracy Score of SVC Model with RBF Kernel::")
print(metrics.accuracy_score(y_test,svc_pred_rbf))
print("\n","Classification Report::")
print(metrics.classification_report(y_test,svc_pred_rbf),'\n')
print("\n","ROC Curve")
svc_prob_rbf=svc_model_linear.predict_proba(x_test_std)
svc_prob_rbf1=svc_prob_rbf[:,1]
fpr,tpr,thresh=metrics.roc_curve(y_test,svc_prob_rbf1)
roc_auc_svc=metrics.auc(fpr,tpr)
plt.figure(dpi=80)
```

```
plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_svc)
plt.plot(fpr,fpr,'r--',color='red')
plt.legend()
```

```
Model Validation ==>

Accuracy Score of SVC Model with RBF Kernel::
0.7727272727272727

 Classification Report::
              precision    recall  f1-score   support

           0       0.81      0.88      0.84       107
           1       0.66      0.53      0.59        47

    accuracy                           0.77       154
   macro avg       0.73      0.71      0.72       154
weighted avg       0.76      0.77      0.77       154



 ROC Curve
```

```
C:\Users\Deviare User\AppData\Local\Temp\ipykernel_10352\2512288102.py:16: UserWar
ning: color is redundantly defined by the 'color' keyword argument and the fmt str
ing "r--" (-> color='r'). The keyword argument will take precedence.
  plt.plot(fpr,fpr,'r--',color='red')
```

Out[69]:
```
<matplotlib.legend.Legend at 0x26fef5e7e90>
```



In [70]:
```
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression(C=0.01)
lr_model.fit(x_train_std,y_train)
lr_pred=lr_model.predict(x_test_std)
```

In [71]:
```
print("Model Validation ==>\n")
print("Accuracy Score of Logistic Regression Model::")
print(metrics.accuracy_score(y_test,lr_pred))
print("\n","Classification Report::")
print(metrics.classification_report(y_test,lr_pred),'\n')
```

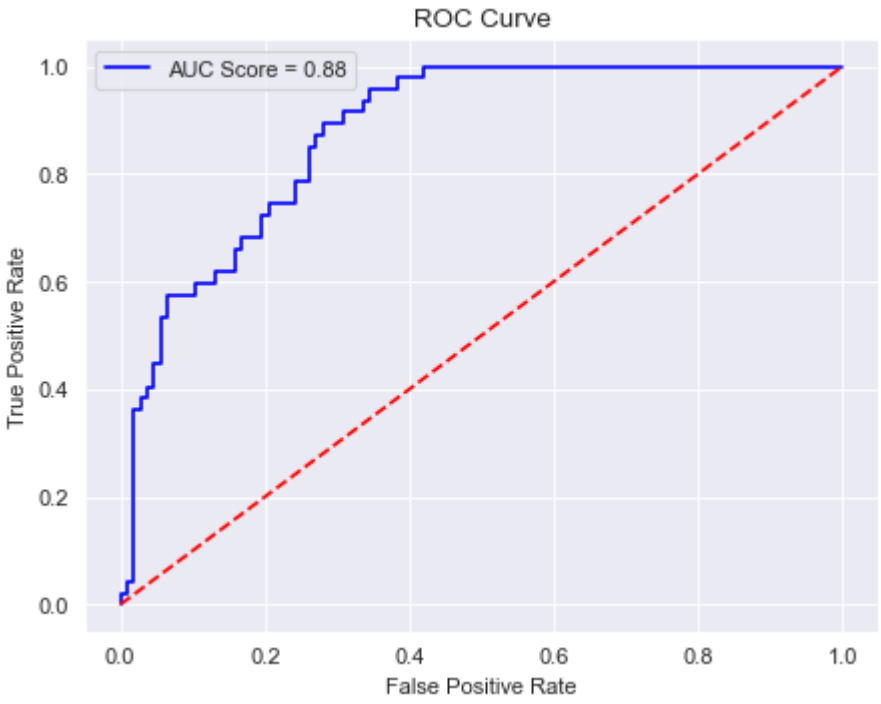
```
print("\n","ROC Curve")
lr_prob=lr_model.predict_proba(x_test_std)
lr_prob1=lr_prob[:,1]
fpr,tpr,thresh=metrics.roc_curve(y_test,lr_prob1)
roc_auc_lr=metrics.auc(fpr,tpr)
plt.figure(dpi=80)
plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_lr)
plt.plot(fpr,fpr,'r--',color='red')
plt.legend()
```

```
Model Validation ==>

Accuracy Score of Logistic Regression Model::
0.8116883116883117

 Classification Report::
              precision    recall  f1-score   support

           0       0.82      0.93      0.87       107
           1       0.78      0.53      0.63        47

    accuracy                           0.81       154
   macro avg       0.80      0.73      0.75       154
weighted avg       0.81      0.81      0.80       154
```

```
 ROC Curve
```

```
C:\Users\Deviare User\AppData\Local\Temp\ipykernel_10352\2270863612.py:16: UserWar
ning: color is redundantly defined by the 'color' keyword argument and the fmt str
ing "r--" (-> color='r'). The keyword argument will take precedence.
  plt.plot(fpr,fpr,'r--',color='red')
```

Out[71]:
```
<matplotlib.legend.Legend at 0x26fef6b5910>
```



In [72]:
```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=1000,random_state=0)
```

```
rf_model.fit(x_train_std,y_train)
rf_pred=rf_model.predict(x_test_std)
```

In [ ]:
```
print("Model Validation ==>\n")
print("Accuracy Score of Logistic Regression Model::")
print(metrics.accuracy_score(y_test,rf_pred))
print("\n","Classification Report::")
print(metrics.classification_report(y_test,rf_pred),'\n')
print("\n","ROC Curve")
rf_prob=rf_model.predict_proba(x_test_std)
rf_prob1=rf_prob[:,1]
fpr,tpr,thresh=metrics.roc_curve(y_test,rf_prob1)
roc_auc_rf=metrics.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()
```

In [ ]:

In [ ]:

In [ ]: