

Health care Project

Thobelani Mkhize



March 25, 2020

*# Importing necessary Libraries to use*

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**import** **seaborn** **as** **sns**

sns.set()

**import** **matplotlib.pyplot** **as** **plt**

In [2]:

*# Importing the dataset*

diabetes = pd.read\_csv('health care diabetes.csv')

In [3]:

*# Viewing the first five raws of the dataset*

diabetes.head()

Out[3]:

|  | **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 [7]:

*# Checking the infomation of the dataset*

diabetes.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 768 entries, 0 to 767

Data columns (total 9 columns):

Pregnancies 768 non-null int64

Glucose 768 non-null int64

BloodPressure 768 non-null int64

SkinThickness 768 non-null int64

Insulin 768 non-null int64

BMI 768 non-null float64

DiabetesPedigreeFunction 768 non-null float64

Age 768 non-null int64

Outcome 768 non-null int64

dtypes: float64(2), int64(7)

memory usage: 54.1 KB

In [8]:

*# Checking the nan values on a dataset*

diabetes.isna().sum()

Out[8]:

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

In [8]:

*# Replacing zeros with nan values*

df = diabetes.replace({'Glucose':0, 'BloodPressure':0, 'SkinThickness':0, 'Insulin':0, 'BMI':0},np.nan)

df.head()

Out[8]:

|  | **Pregnancies** | **Glucose** | **BloodPressure** | **SkinThickness** | **Insulin** | **BMI** | **DiabetesPedigreeFunction** | **Age** | **Outcome** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 6 | 148.0 | 72.0 | 35.0 | NaN | 33.6 | 0.627 | 50 | 1 |
| **1** | 1 | 85.0 | 66.0 | 29.0 | NaN | 26.6 | 0.351 | 31 | 0 |
| **2** | 8 | 183.0 | 64.0 | NaN | NaN | 23.3 | 0.672 | 32 | 1 |
| **3** | 1 | 89.0 | 66.0 | 23.0 | 94.0 | 28.1 | 0.167 | 21 | 0 |
| **4** | 0 | 137.0 | 40.0 | 35.0 | 168.0 | 43.1 | 2.288 | 33 | 1 |

In [9]:

df.isna().sum()

Out[9]:

Pregnancies 0

Glucose 5

BloodPressure 35

SkinThickness 227

Insulin 374

BMI 11

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

In [12]:

*# Now we are replacing the nan values with mean, in order to have a value into our dataset*

df1 = df.fillna(df.mean())

df1.head()

Out[12]:

|  | **Pregnancies** | **Glucose** | **BloodPressure** | **SkinThickness** | **Insulin** | **BMI** | **DiabetesPedigreeFunction** | **Age** | **Outcome** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 6 | 148.0 | 72.0 | 35.00000 | 155.548223 | 33.6 | 0.627 | 50 | 1 |
| **1** | 1 | 85.0 | 66.0 | 29.00000 | 155.548223 | 26.6 | 0.351 | 31 | 0 |
| **2** | 8 | 183.0 | 64.0 | 29.15342 | 155.548223 | 23.3 | 0.672 | 32 | 1 |
| **3** | 1 | 89.0 | 66.0 | 23.00000 | 94.000000 | 28.1 | 0.167 | 21 | 0 |
| **4** | 0 | 137.0 | 40.0 | 35.00000 | 168.000000 | 43.1 | 2.288 | 33 | 1 |

**Visualizing The Variables**

In [13]:

*# Plotting a histograme for the Glucose column*

plt.figure(figsize=(10,10))

plt.subplot(221)

plt.title('Glucose Histogram')

df1['Glucose'].hist(alpha = 0.2)

plt.subplot(222)

plt.title('BloodPressure Histogram')

df1['BloodPressure'].hist(alpha = 0.2)

plt.subplot(223)

plt.title('SkinThickness Histogram')

df1['SkinThickness'].hist(alpha = 0.2)

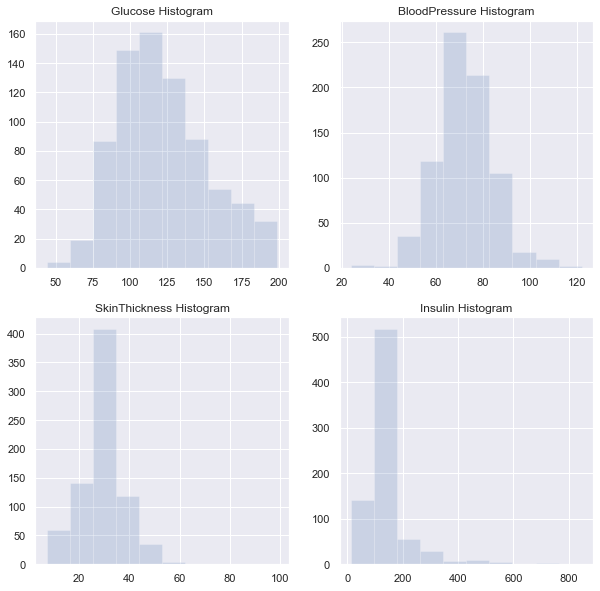
plt.subplot(224)

plt.title('Insulin Histogram')

df1['Insulin'].hist(alpha = 0.2)

Out[13]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x18e0b2c7390>



In [14]:

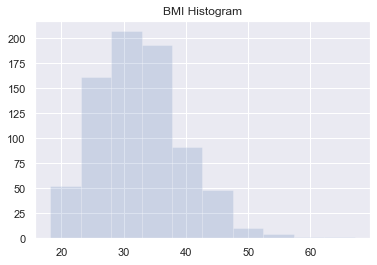
*# Plotting a histograme for the BMI column*

plt.title('BMI Histogram')

df1['BMI'].hist(alpha = 0.2)

Out[14]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x18e0b250710>



**Create a Count (Frequency) Plot**

In [16]:

*# Plotting a Count Plot for the Glucose*

*#diabetes['Glucose'].describe()*

*#sns.countplot(diabetes['Glucose'])*

Glu = df1[['Glucose']].head()

Glu

Out[16]:

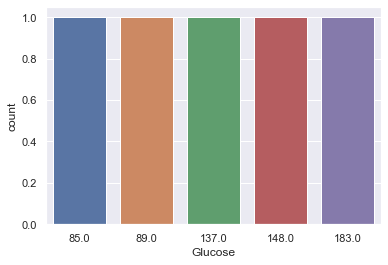
|  | **Glucose** |
| --- | --- |
| **0** | 148.0 |
| **1** | 85.0 |
| **2** | 183.0 |
| **3** | 89.0 |
| **4** | 137.0 |

In [17]:

sns.countplot(Glu['Glucose'])

Out[17]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x18e0b405e48>



In [18]:

*# Plotting a count plot for the blood pressure*

*#diabetes['BloodPressure'].describe()*

Bp = df1[['BloodPressure']].head()

Bp

Out[18]:

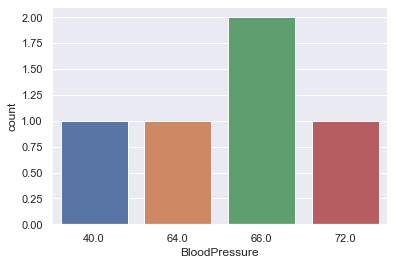
|  | **BloodPressure** |
| --- | --- |
| **0** | 72.0 |
| **1** | 66.0 |
| **2** | 64.0 |
| **3** | 66.0 |
| **4** | 40.0 |

In [19]:

sns.countplot(Bp['BloodPressure'])

Out[19]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x18e0b45d080>



In [20]:

*# Plotting a count plot for the Skin Thickness*

*# diabetes['SkinThickness'].describe()*

SThick = df1[['SkinThickness']].head()

SThick

Out[20]:

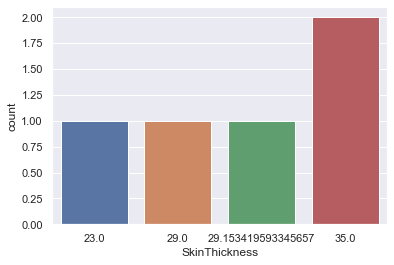
|  | **SkinThickness** |
| --- | --- |
| **0** | 35.00000 |
| **1** | 29.00000 |
| **2** | 29.15342 |
| **3** | 23.00000 |
| **4** | 35.00000 |

In [21]:

sns.countplot(SThick['SkinThickness'])

Out[21]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x18e0b444f98>



In [22]:

*# Plotting a count plot for the Insulin*

*# diabetes['Insulin'].describe()*

Ins = df1[['Insulin']].head()

Ins

Out[22]:

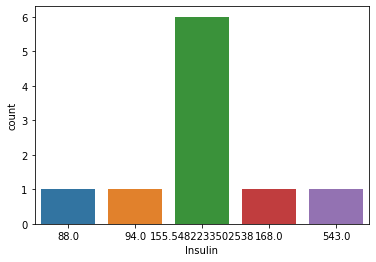
|  | **Insulin** |
| --- | --- |
| **0** | 155.548223 |
| **1** | 155.548223 |
| **2** | 155.548223 |
| **3** | 94.000000 |
| **4** | 168.000000 |

In [25]:

sns.countplot(Ins['Insulin'])

Out[25]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x27d38b12240>



In [23]:

*# Plotting the count plot for the BMI*

*# diabetes['BMI'].describe()*

BM = df1[['BMI']].head()

BM

Out[23]:

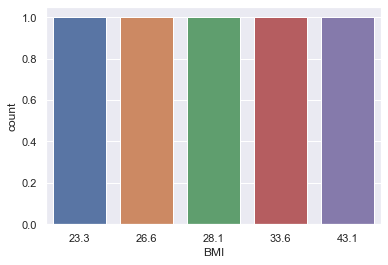
|  | **BMI** |
| --- | --- |
| **0** | 33.6 |
| **1** | 26.6 |
| **2** | 23.3 |
| **3** | 28.1 |
| **4** | 43.1 |

In [24]:

sns.countplot(BM['BMI'])

Out[24]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x18e0b5209b0>



**Data Exploration: Week Two**

In [33]:

*#Q One*

*#plt.subplot(224)*

plt.title('Count Of Outcomes for diabetes')

sns.countplot(df1['Outcome'])

plt.legend(loc = 'upper right')

plt.xlabel('Outcome')

plt.ylabel('Count')

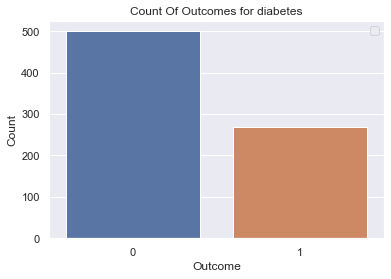
print(df1['Outcome'].value\_counts())

No handles with labels found to put in legend.

0 500

1 268

Name: Outcome, dtype: int64



In [43]:

*# Q Two*

plt.figure(figsize = (14,14))

plt.subplot(221)

plt.title('Correlation between Glucose and Insulin')

sns.scatterplot(x = 'Glucose', y = 'Insulin', data = df1, hue = 'Outcome')

plt.subplot(222)

plt.title('Correlation between BloodPressure and SkinThickness')

sns.scatterplot(x = 'BloodPressure', y = 'BMI', data = df1, hue = 'Outcome')

plt.subplot(223)

plt.title('Correlation between BloodPressure and SkinThickness')

sns.scatterplot(x = 'BloodPressure', y = 'SkinThickness', data = df1, hue = 'Outcome')

plt.subplot(224)

plt.title('Correlation between BloodPressure and Glucose')

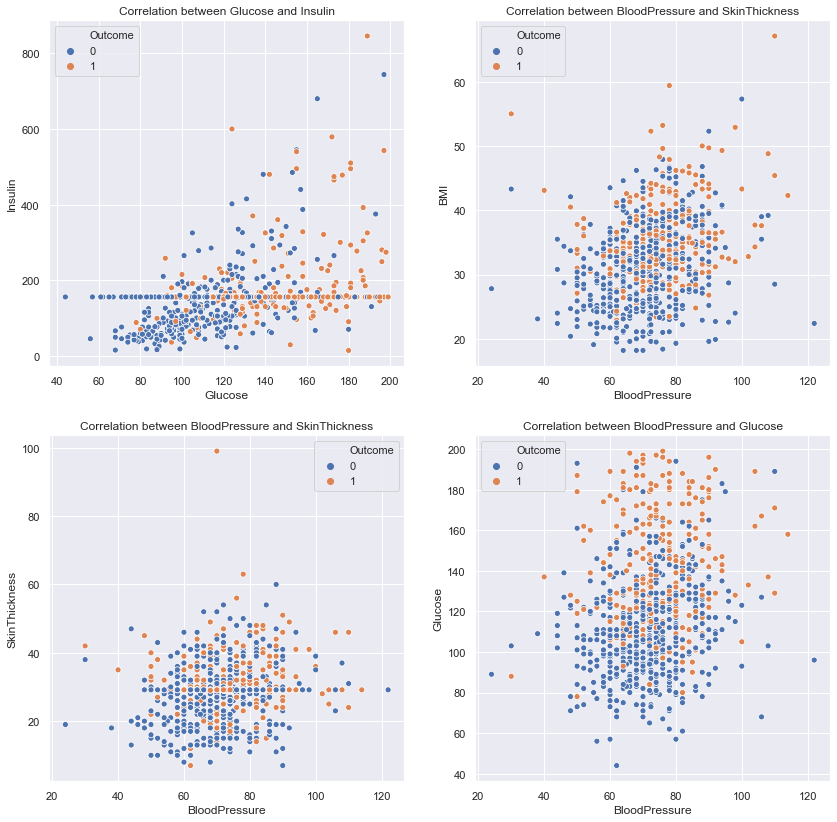
sns.scatterplot(x = 'BloodPressure', y = 'Glucose', data = df1, hue = 'Outcome')

8print(df1['Outcome'].value\_counts())

0 500

1 268

Name: Outcome, dtype: int64



In [41]:

df1.corr()

Out[41]:

|  | **Pregnancies** | **Glucose** | **BloodPressure** | **SkinThickness** | **Insulin** | **BMI** | **DiabetesPedigreeFunction** | **Age** | **Outcome** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Pregnancies** | 1.000000 | 0.127911 | 0.208522 | 0.082989 | 0.056027 | 0.021565 | -0.033523 | 0.544341 | 0.221898 |
| **Glucose** | 0.127911 | 1.000000 | 0.218367 | 0.192991 | 0.420157 | 0.230941 | 0.137060 | 0.266534 | 0.492928 |
| **BloodPressure** | 0.208522 | 0.218367 | 1.000000 | 0.192816 | 0.072517 | 0.281268 | -0.002763 | 0.324595 | 0.166074 |
| **SkinThickness** | 0.082989 | 0.192991 | 0.192816 | 1.000000 | 0.158139 | 0.542398 | 0.100966 | 0.127872 | 0.215299 |
| **Insulin** | 0.056027 | 0.420157 | 0.072517 | 0.158139 | 1.000000 | 0.166586 | 0.098634 | 0.136734 | 0.214411 |
| **BMI** | 0.021565 | 0.230941 | 0.281268 | 0.542398 | 0.166586 | 1.000000 | 0.153400 | 0.025519 | 0.311924 |
| **DiabetesPedigreeFunction** | -0.033523 | 0.137060 | -0.002763 | 0.100966 | 0.098634 | 0.153400 | 1.000000 | 0.033561 | 0.173844 |
| **Age** | 0.544341 | 0.266534 | 0.324595 | 0.127872 | 0.136734 | 0.025519 | 0.033561 | 1.000000 | 0.238356 |
| **Outcome** | 0.221898 | 0.492928 | 0.166074 | 0.215299 | 0.214411 | 0.311924 | 0.173844 | 0.238356 | 1.000000 |

In [42]:

*# Q Three*

*# Finding if variables are corelated*

*# There is a strong corelation between the Age, Pregnancies and BloodPressure as well as Glucose*

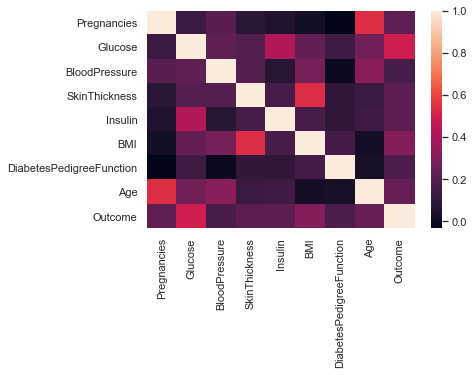
*# There is a strong corelation between the Insulin and DiabetesPedigreeFunction*

plt.figure(figsize=(6,4))

sns.heatmap(df1.corr())

Out[42]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x18e0d64a048>



**Data Modeling : Week Three**

In [44]:

**from** **sklearn.linear\_model** **import** LogisticRegression

**from** **sklearn.model\_selection** **import** train\_test\_split

**from** **sklearn.metrics** **import** confusion\_matrix

**from** **sklearn.metrics** **import** classification\_report

**from** **sklearn.metrics** **import** accuracy\_score

**from** **sklearn.metrics** **import** precision\_score

**from** **sklearn.metrics** **import** recall\_score

**from** **sklearn.metrics** **import** f1\_score

**from** **sklearn** **import** metrics

**Train and test split**

In [60]:

df1.head()

Out[60]:

|  | **Pregnancies** | **Glucose** | **BloodPressure** | **SkinThickness** | **Insulin** | **BMI** | **DiabetesPedigreeFunction** | **Age** | **Outcome** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 6 | 148.0 | 72.0 | 35.00000 | 155.548223 | 33.6 | 0.627 | 50 | 1 |
| **1** | 1 | 85.0 | 66.0 | 29.00000 | 155.548223 | 26.6 | 0.351 | 31 | 0 |
| **2** | 8 | 183.0 | 64.0 | 29.15342 | 155.548223 | 23.3 | 0.672 | 32 | 1 |
| **3** | 1 | 89.0 | 66.0 | 23.00000 | 94.000000 | 28.1 | 0.167 | 21 | 0 |
| **4** | 0 | 137.0 | 40.0 | 35.00000 | 168.000000 | 43.1 | 2.288 | 33 | 1 |

In [80]:

X = df1.iloc[:,0:7]

In [81]:

X.shape

Out[81]:

(768, 7)

In [82]:

Y = df1.iloc[:,8]

In [83]:

Y.shape

Out[83]:

(768,)

**KNN MODEL**

In [85]:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,Y , test\_size = 0.2)

In [95]:

*#Training*

**from** **sklearn** **import** neighbors

model1 = neighbors.KNeighborsClassifier()

model1.fit(X\_train,y\_train)

Out[95]:

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=None, n\_neighbors=5, p=2,

weights='uniform')

In [96]:

*#Testing*

predicted = model1.predict(X\_test)

predicted

Out[96]:

array([1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0,

1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0,

1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0,

0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,

1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0],

dtype=int64)

In [97]:

*#Evaluation*

*#Confusion Matrix*

print(metrics.confusion\_matrix(y\_test, predicted))

[[84 16]

[22 32]]

In [98]:

*#Classification Report*

print("**\n**Acuracy Score of KNN Model:")

print(metrics.accuracy\_score(y\_test, predicted))

print("**\n**Classification Report:")

print(metrics.classification\_report(y\_test, predicted))

Acuracy Score of KNN Model:

0.7532467532467533

Classification Report:

precision recall f1-score support

0 0.79 0.84 0.82 100

1 0.67 0.59 0.63 54

accuracy 0.75 154

macro avg 0.73 0.72 0.72 154

weighted avg 0.75 0.75 0.75 154

In [118]:

print("ROC Curve")

model1\_prob = model1.predict\_proba(X\_test)

model1\_prob1 = model1\_prob[:,1]

fpr,tpr,thresh = metrics.roc\_curve(y\_test,model1\_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,'r',label = 'AUC Score = **%0.2f**'%**roc\_auc\_knn**)

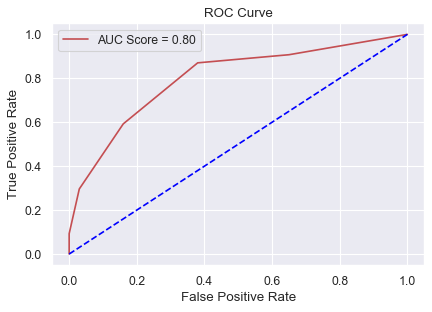
plt.plot(fpr,fpr,'b--',color='blue')

plt.legend()

ROC Curve

Out[118]:

<matplotlib.legend.Legend at 0x18e0ce1b7b8>



**Random Forest Model**

In [100]:

*#Training*

**from** **sklearn.ensemble** **import** RandomForestClassifier

model2 = RandomForestClassifier()

model2.fit(X\_train,y\_train)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: The default value of n\_estimators will change from 10 in version 0.20 to 100 in 0.22.

"10 in version 0.20 to 100 in 0.22.", FutureWarning)

Out[100]:

RandomForestClassifier(bootstrap=True, class\_weight=None, criterion='gini',

max\_depth=None, max\_features='auto', max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, n\_estimators=10,

n\_jobs=None, oob\_score=False, random\_state=None,

verbose=0, warm\_start=False)

In [101]:

*#Testing*

predicted = model2.predict(X\_test)

predicted

Out[101]:

array([1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0,

1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,

1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,

0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,

1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,

0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,

0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],

dtype=int64)

In [102]:

*#Evaluation*

*#Confusion Matrix*

print(metrics.confusion\_matrix(y\_test, predicted))

[[86 14]

[26 28]]

In [103]:

*#Classification Report*

print("**\n**Acuracy Score of RF Model:")

print(metrics.accuracy\_score(y\_test, predicted))

print("**\n**Classification Report:")

print(metrics.classification\_report(y\_test, predicted))

Acuracy Score of RF Model:

0.7402597402597403

Classification Report:

precision recall f1-score support

0 0.77 0.86 0.81 100

1 0.67 0.52 0.58 54

accuracy 0.74 154

macro avg 0.72 0.69 0.70 154

weighted avg 0.73 0.74 0.73 154

In [117]:

print("ROC Curve")

model2\_prob = model2.predict\_proba(X\_test)

model2\_prob1 = model1\_prob[:,1]

fpr,tpr,thresh = metrics.roc\_curve(y\_test,model2\_prob1)

roc\_auc\_rf = 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,'r',label = 'AUC Score = **%0.2f**'%**roc\_auc\_rf**)

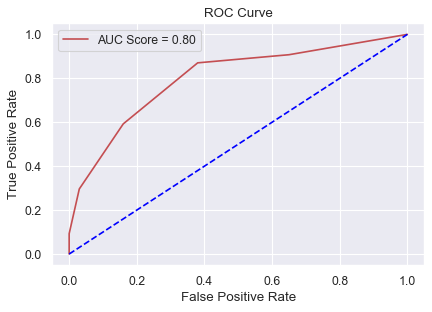
plt.plot(fpr,fpr,'b--',color='blue')

plt.legend()

ROC Curve

Out[117]:

<matplotlib.legend.Legend at 0x18e0f2474e0>



**DecisionTreeClassifier Model**

In [104]:

*#Training*

**from** **sklearn.tree** **import** DecisionTreeClassifier

model3 = DecisionTreeClassifier()

model3.fit(X\_train,y\_train)

Out[104]:

DecisionTreeClassifier(class\_weight=None, criterion='gini', max\_depth=None,

max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort=False,

random\_state=None, splitter='best')

In [105]:

*#Testing*

predicted = model3.predict(X\_test)

predicted

Out[105]:

array([1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0,

1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0,

1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1,

1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,

0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,

0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0],

dtype=int64)

In [106]:

*#Evaluation*

*#Confusion Matrix*

print(metrics.confusion\_matrix(y\_test, predicted))

[[76 24]

[24 30]]

In [107]:

*#Classification Report*

print("**\n**Acuracy Score of DecisionTree Model:")

print(metrics.accuracy\_score(y\_test, predicted))

print("**\n**Classification Report:")

print(metrics.classification\_report(y\_test, predicted))

Acuracy Score of DecisionTree Model:

0.6883116883116883

Classification Report:

precision recall f1-score support

0 0.76 0.76 0.76 100

1 0.56 0.56 0.56 54

accuracy 0.69 154

macro avg 0.66 0.66 0.66 154

weighted avg 0.69 0.69 0.69 154

In [120]:

print("ROC Curve")

model3\_prob = model3.predict\_proba(X\_test)

model3\_prob1 = model3\_prob[:,1]

fpr,tpr,thresh = metrics.roc\_curve(y\_test,model3\_prob1)

roc\_auc\_dt = 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,'r',label = 'AUC Score = **%0.2f**'%**roc\_auc\_dt**)

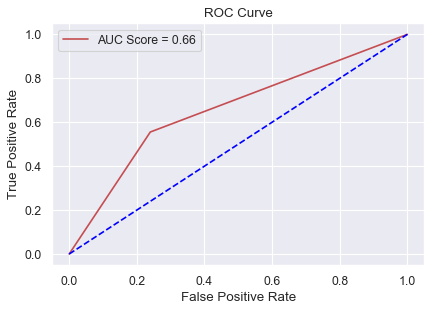
plt.plot(fpr,fpr,'b--',color='blue')

plt.legend()

ROC Curve

Out[120]:

<matplotlib.legend.Legend at 0x18e0cf79c88>



**LogisticRegression Model**

In [108]:

*#Training*

**from** **sklearn.linear\_model** **import** LogisticRegression

model4 = LogisticRegression()

model4.fit(X\_train,y\_train)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear\_model\logistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.

FutureWarning)

Out[108]:

LogisticRegression(C=1.0, class\_weight=None, dual=False, fit\_intercept=True,

intercept\_scaling=1, l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None, penalty='l2',

random\_state=None, solver='warn', tol=0.0001, verbose=0,

warm\_start=False)

In [109]:

*#Testing*

predicted = model4.predict(X\_test)

predicted

Out[109]:

array([1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,

1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,

1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,

0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],

dtype=int64)

In [110]:

*#Evaluation*

*#Confusion Matrix*

print(metrics.confusion\_matrix(y\_test, predicted))

[[94 6]

[21 33]]

In [111]:

*#Classification Report*

print("**\n**Acuracy Score of LogisticRegression Model:")

print(metrics.accuracy\_score(y\_test, predicted))

print("**\n**Classification Report:")

print(metrics.classification\_report(y\_test, predicted))

Acuracy Score of LogisticRegression Model:

0.8246753246753247

Classification Report:

precision recall f1-score support

0 0.82 0.94 0.87 100

1 0.85 0.61 0.71 54

accuracy 0.82 154

macro avg 0.83 0.78 0.79 154

weighted avg 0.83 0.82 0.82 154

In [121]:

print("ROC Curve")

model4\_prob = model4.predict\_proba(X\_test)

model4\_prob1 = model4\_prob[:,1]

fpr,tpr,thresh = metrics.roc\_curve(y\_test,model4\_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,'r',label = 'AUC Score = **%0.2f**'%**roc\_auc\_lr**)

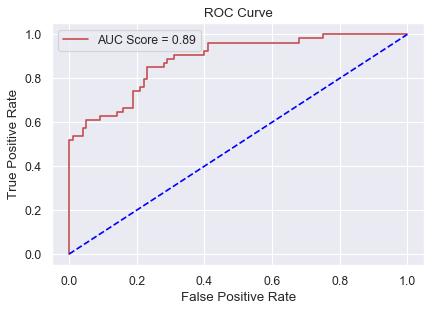
plt.plot(fpr,fpr,'b--',color='blue')

plt.legend()

ROC Curve

Out[121]:

<matplotlib.legend.Legend at 0x18e0cd2d908>



In [133]:

df1.to\_excel('Health\_Care.xlsx',index = **False**)

In [134]:

y =pd.read\_excel('Health\_Care.xlsx')

In [135]:

y

Out[135]:

|  | **Pregnancies** | **Glucose** | **BloodPressure** | **SkinThickness** | **Insulin** | **BMI** | **DiabetesPedigreeFunction** | **Age** | **Outcome** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 6 | 148.0 | 72.000000 | 35.00000 | 155.548223 | 33.600000 | 0.627 | 50 | 1 |
| **1** | 1 | 85.0 | 66.000000 | 29.00000 | 155.548223 | 26.600000 | 0.351 | 31 | 0 |
| **2** | 8 | 183.0 | 64.000000 | 29.15342 | 155.548223 | 23.300000 | 0.672 | 32 | 1 |
| **3** | 1 | 89.0 | 66.000000 | 23.00000 | 94.000000 | 28.100000 | 0.167 | 21 | 0 |
| **4** | 0 | 137.0 | 40.000000 | 35.00000 | 168.000000 | 43.100000 | 2.288 | 33 | 1 |
| **5** | 5 | 116.0 | 74.000000 | 29.15342 | 155.548223 | 25.600000 | 0.201 | 30 | 0 |
| **6** | 3 | 78.0 | 50.000000 | 32.00000 | 88.000000 | 31.000000 | 0.248 | 26 | 1 |
| **7** | 10 | 115.0 | 72.405184 | 29.15342 | 155.548223 | 35.300000 | 0.134 | 29 | 0 |
| **8** | 2 | 197.0 | 70.000000 | 45.00000 | 543.000000 | 30.500000 | 0.158 | 53 | 1 |
| **9** | 8 | 125.0 | 96.000000 | 29.15342 | 155.548223 | 32.457464 | 0.232 | 54 | 1 |
| **10** | 4 | 110.0 | 92.000000 | 29.15342 | 155.548223 | 37.600000 | 0.191 | 30 | 0 |
| **11** | 10 | 168.0 | 74.000000 | 29.15342 | 155.548223 | 38.000000 | 0.537 | 34 | 1 |
| **12** | 10 | 139.0 | 80.000000 | 29.15342 | 155.548223 | 27.100000 | 1.441 | 57 | 0 |
| **13** | 1 | 189.0 | 60.000000 | 23.00000 | 846.000000 | 30.100000 | 0.398 | 59 | 1 |
| **14** | 5 | 166.0 | 72.000000 | 19.00000 | 175.000000 | 25.800000 | 0.587 | 51 | 1 |
| **15** | 7 | 100.0 | 72.405184 | 29.15342 | 155.548223 | 30.000000 | 0.484 | 32 | 1 |
| **16** | 0 | 118.0 | 84.000000 | 47.00000 | 230.000000 | 45.800000 | 0.551 | 31 | 1 |
| **17** | 7 | 107.0 | 74.000000 | 29.15342 | 155.548223 | 29.600000 | 0.254 | 31 | 1 |
| **18** | 1 | 103.0 | 30.000000 | 38.00000 | 83.000000 | 43.300000 | 0.183 | 33 | 0 |
| **19** | 1 | 115.0 | 70.000000 | 30.00000 | 96.000000 | 34.600000 | 0.529 | 32 | 1 |
| **20** | 3 | 126.0 | 88.000000 | 41.00000 | 235.000000 | 39.300000 | 0.704 | 27 | 0 |
| **21** | 8 | 99.0 | 84.000000 | 29.15342 | 155.548223 | 35.400000 | 0.388 | 50 | 0 |
| **22** | 7 | 196.0 | 90.000000 | 29.15342 | 155.548223 | 39.800000 | 0.451 | 41 | 1 |
| **23** | 9 | 119.0 | 80.000000 | 35.00000 | 155.548223 | 29.000000 | 0.263 | 29 | 1 |
| **24** | 11 | 143.0 | 94.000000 | 33.00000 | 146.000000 | 36.600000 | 0.254 | 51 | 1 |
| **25** | 10 | 125.0 | 70.000000 | 26.00000 | 115.000000 | 31.100000 | 0.205 | 41 | 1 |
| **26** | 7 | 147.0 | 76.000000 | 29.15342 | 155.548223 | 39.400000 | 0.257 | 43 | 1 |
| **27** | 1 | 97.0 | 66.000000 | 15.00000 | 140.000000 | 23.200000 | 0.487 | 22 | 0 |
| **28** | 13 | 145.0 | 82.000000 | 19.00000 | 110.000000 | 22.200000 | 0.245 | 57 | 0 |
| **29** | 5 | 117.0 | 92.000000 | 29.15342 | 155.548223 | 34.100000 | 0.337 | 38 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **738** | 2 | 99.0 | 60.000000 | 17.00000 | 160.000000 | 36.600000 | 0.453 | 21 | 0 |
| **739** | 1 | 102.0 | 74.000000 | 29.15342 | 155.548223 | 39.500000 | 0.293 | 42 | 1 |
| **740** | 11 | 120.0 | 80.000000 | 37.00000 | 150.000000 | 42.300000 | 0.785 | 48 | 1 |
| **741** | 3 | 102.0 | 44.000000 | 20.00000 | 94.000000 | 30.800000 | 0.400 | 26 | 0 |
| **742** | 1 | 109.0 | 58.000000 | 18.00000 | 116.000000 | 28.500000 | 0.219 | 22 | 0 |
| **743** | 9 | 140.0 | 94.000000 | 29.15342 | 155.548223 | 32.700000 | 0.734 | 45 | 1 |
| **744** | 13 | 153.0 | 88.000000 | 37.00000 | 140.000000 | 40.600000 | 1.174 | 39 | 0 |
| **745** | 12 | 100.0 | 84.000000 | 33.00000 | 105.000000 | 30.000000 | 0.488 | 46 | 0 |
| **746** | 1 | 147.0 | 94.000000 | 41.00000 | 155.548223 | 49.300000 | 0.358 | 27 | 1 |
| **747** | 1 | 81.0 | 74.000000 | 41.00000 | 57.000000 | 46.300000 | 1.096 | 32 | 0 |
| **748** | 3 | 187.0 | 70.000000 | 22.00000 | 200.000000 | 36.400000 | 0.408 | 36 | 1 |
| **749** | 6 | 162.0 | 62.000000 | 29.15342 | 155.548223 | 24.300000 | 0.178 | 50 | 1 |
| **750** | 4 | 136.0 | 70.000000 | 29.15342 | 155.548223 | 31.200000 | 1.182 | 22 | 1 |
| **751** | 1 | 121.0 | 78.000000 | 39.00000 | 74.000000 | 39.000000 | 0.261 | 28 | 0 |
| **752** | 3 | 108.0 | 62.000000 | 24.00000 | 155.548223 | 26.000000 | 0.223 | 25 | 0 |
| **753** | 0 | 181.0 | 88.000000 | 44.00000 | 510.000000 | 43.300000 | 0.222 | 26 | 1 |
| **754** | 8 | 154.0 | 78.000000 | 32.00000 | 155.548223 | 32.400000 | 0.443 | 45 | 1 |
| **755** | 1 | 128.0 | 88.000000 | 39.00000 | 110.000000 | 36.500000 | 1.057 | 37 | 1 |
| **756** | 7 | 137.0 | 90.000000 | 41.00000 | 155.548223 | 32.000000 | 0.391 | 39 | 0 |
| **757** | 0 | 123.0 | 72.000000 | 29.15342 | 155.548223 | 36.300000 | 0.258 | 52 | 1 |
| **758** | 1 | 106.0 | 76.000000 | 29.15342 | 155.548223 | 37.500000 | 0.197 | 26 | 0 |
| **759** | 6 | 190.0 | 92.000000 | 29.15342 | 155.548223 | 35.500000 | 0.278 | 66 | 1 |
| **760** | 2 | 88.0 | 58.000000 | 26.00000 | 16.000000 | 28.400000 | 0.766 | 22 | 0 |
| **761** | 9 | 170.0 | 74.000000 | 31.00000 | 155.548223 | 44.000000 | 0.403 | 43 | 1 |
| **762** | 9 | 89.0 | 62.000000 | 29.15342 | 155.548223 | 22.500000 | 0.142 | 33 | 0 |
| **763** | 10 | 101.0 | 76.000000 | 48.00000 | 180.000000 | 32.900000 | 0.171 | 63 | 0 |
| **764** | 2 | 122.0 | 70.000000 | 27.00000 | 155.548223 | 36.800000 | 0.340 | 27 | 0 |
| **765** | 5 | 121.0 | 72.000000 | 23.00000 | 112.000000 | 26.200000 | 0.245 | 30 | 0 |
| **766** | 1 | 126.0 | 60.000000 | 29.15342 | 155.548223 | 30.100000 | 0.349 | 47 | 1 |
| **767** | 1 | 93.0 | 70.000000 | 31.00000 | 155.548223 | 30.400000 | 0.315 | 23 | 0 |

768 rows × 9 columns

In [ ]:

*# Q4 Visualisation Using Table*

*# Here is the link for the dashbord*

https://public.tableau.com/profile/thobelani.mkhize*#!/*