CAPSTONE PROJECT HEALTHCARE

Project Task: Week 1

Data Exploration:

1. Perform descriptive analysis. Understand the variables and their corresponding values. On the columns below, a value of zero does not make sense and thus indicates missing value:

• Glucose

• BloodPressure

• SkinThickness

• Insulin

• BMI

1. Visually explore these variables using histograms. Treat the missing values accordingly.
2. There are integer and float data type variables in this dataset. Create a count (frequency) plot describing the data types and the count of variables.

In [1]:
```
#import libraries

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

In [2]:
```
hc_data=pd.read_csv(r'C:\Users\Admin\Desktop\Project 2\Healthcare - Diabetes\health car
```

In [3]:
```
hc_data.head()
```

Out[3]:

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

In [4]:
```
hc_data.shape
```

Out[4]:
```
(768, 9)
```

In [5]:
```
hc_data.describe()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreel
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	76
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 [6]:
```
hc_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
```

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

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

In [8]:

```
hc_data.isnull()
```

Out[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
763	False	False	False	False	False	False	False	False
764	False	False	False	False	False	False	False	False
765	False	False	False	False	False	False	False	False
766	False	False	False	False	False	False	False	False
767	False	False	False	False	False	False	False	False

768 rows × 9 columns

In [9]:
```
hc_data.isnull().any()
```

Out[9]:
```
Pregnancies                 False
Glucose                     False
BloodPressure               False
SkinThickness               False
Insulin                     False
BMI                         False
DiabetesPedigreeFunction    False
Age                         False
Outcome                     False
dtype: bool
```

In [10]:
```
hc_data.isnull().sum()
```

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

There are total 9 columns and 768 rows in dataset. The total number of null values or

missing values is Zero. But we need to check all the 0 values in every column and make necessary changes to replace them as per instructions.

In [11]:
```
hc_data['Glucose']
```

Out[11]:
```
0      148
1       85
2      183
3       89
4      137
      ...
763    101
764    122
765    121
766    126
767     93
Name: Glucose, Length: 768, dtype: int64
```

In [12]:
```
hc_data['Glucose'].value_counts()
```

Out[12]:
```
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
```

In [13]:
```
hc_data['Glucose'].value_counts()[0]
```

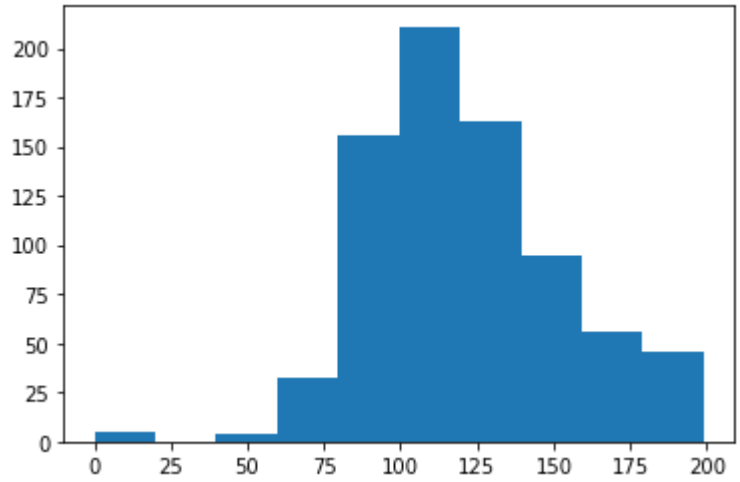
Out[13]:
```
5
```

In [14]:
```
(hc_data['Glucose']==0).sum()
```

Out[14]:
```
5
```

In [15]:
```
plt.hist(hc_data['Glucose'])
```

Out[15]:
```
(array([  5.,   0.,   4.,  32., 156., 211., 163.,  95.,  56.,  46.]),
 array([  0. ,  19.9,  39.8,  59.7,  79.6,  99.5, 119.4, 139.3, 159.2,
        179.1, 199. ]),
 <BarContainer object of 10 artists>)
```



The total number of zero values in Glucose is 5, so we use the replace zero value with mean value of that column.

In [16]:
```
Glucose_mean=hc_data['Glucose'].mean()
Glucose_mean
```

Out[16]:
```
120.89453125
```

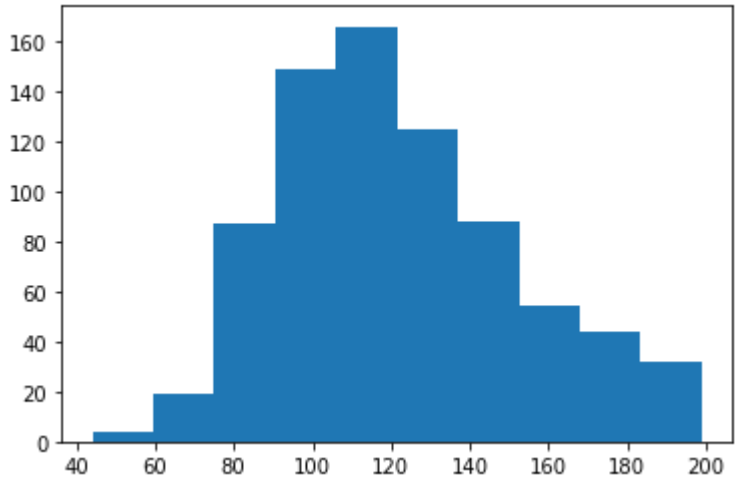
In [17]:
```
hc_data['Glucose']=hc_data['Glucose'].replace(0,Glucose_mean)
```

In [18]:
```
(hc_data['Glucose']==0).sum()
```

Out[18]:
```
0
```

In [19]:
```
plt.hist(hc_data['Glucose'])
```

Out[19]:
```
(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. ]),
 <BarContainer object of 10 artists>)
```



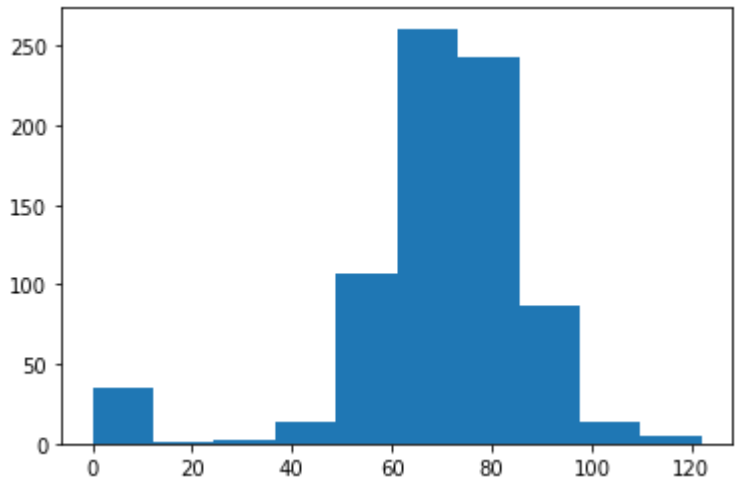
Now as you can see from above histrogram the no zero valus is present in Glucose.

In [20]:
```
(hc_data['BloodPressure']==0).sum()
```

Out[20]:
```
35
```

In [21]:
```
plt.hist(hc_data['BloodPressure'])
```

Out[21]:
```
(array([ 35.,   1.,   2.,  13., 107., 261., 243.,  87.,  14.,   5.]),
 array([  0. ,  12.2,  24.4,  36.6,  48.8,  61. ,  73.2,  85.4,  97.6,
        109.8, 122. ]),
 <BarContainer object of 10 artists>)
```



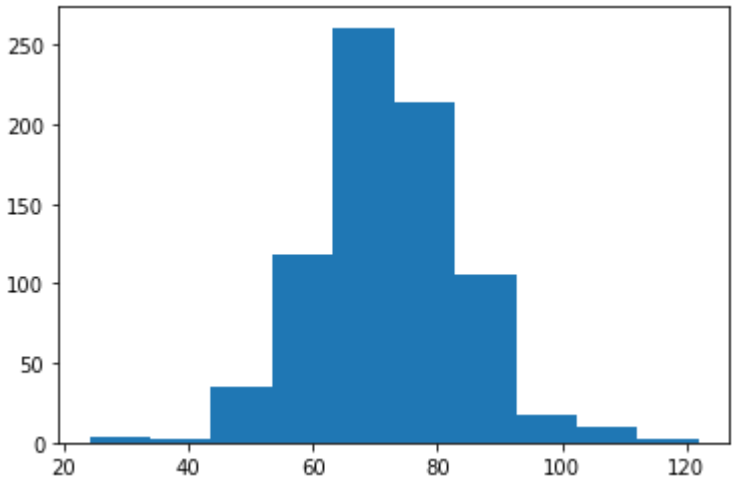
In [22]:
```
Bloodpressure_mean=hc_data['BloodPressure'].mean()
Bloodpressure_mean
```

Out[22]:
```
69.10546875
```

In [23]:
```
hc_data['BloodPressure']=hc_data['BloodPressure'].replace(0,Bloodpressure_mean)
```

In [24]:
```
plt.hist(hc_data['BloodPressure'])
```

Out[24]:
```
(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. ]),
 <BarContainer object of 10 artists>)
```



In [25]:
```
(hc_data['BloodPressure']==0).sum()
```

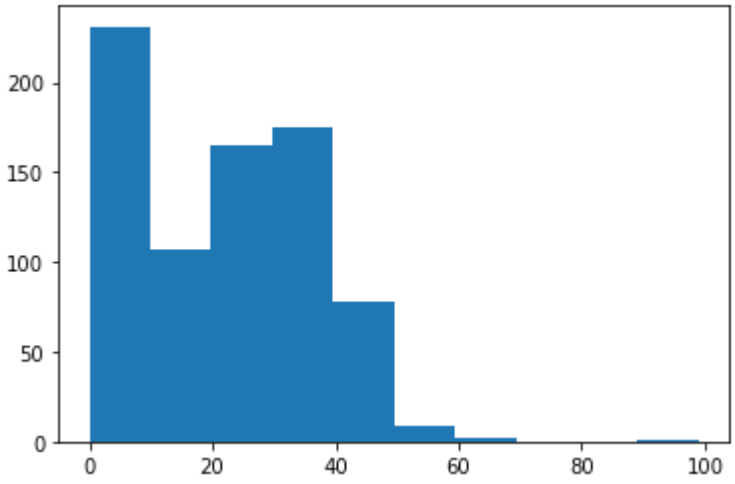
Out[25]:
```
0
```

In [26]:
```
(hc_data['SkinThickness']==0).sum()
```

Out[26]:
```
227
```

In [27]:
```
plt.hist(hc_data['SkinThickness'])
```

Out[27]:
```
(array([231., 107., 165., 175.,  78.,   9.,   2.,   0.,   0.,   1.]),
 array([ 0. ,  9.9, 19.8, 29.7, 39.6, 49.5, 59.4, 69.3, 79.2, 89.1, 99. ]),
 <BarContainer object of 10 artists>)
```



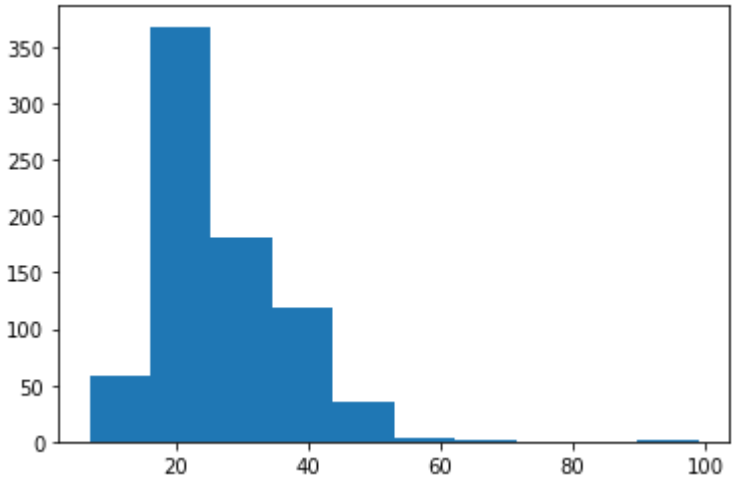
In [28]:
```
Skinthickness_mean=hc_data['SkinThickness'].mean()
Skinthickness_mean
```

Out[28]:
```
20.536458333333332
```

In [29]:
```
hc_data['SkinThickness']=hc_data['SkinThickness'].replace(0,Skinthickness_mean)
```

In [30]:
```
plt.hist(hc_data['SkinThickness'])
```

Out[30]:
```
(array([ 59., 368., 181., 118.,  36.,   4.,   1.,   0.,   0.,   1.]),
 array([ 7. , 16.2, 25.4, 34.6, 43.8, 53. , 62.2, 71.4, 80.6, 89.8, 99. ]),
 <BarContainer object of 10 artists>)
```



In [31]:
```
(hc_data['SkinThickness']==0).sum()
```

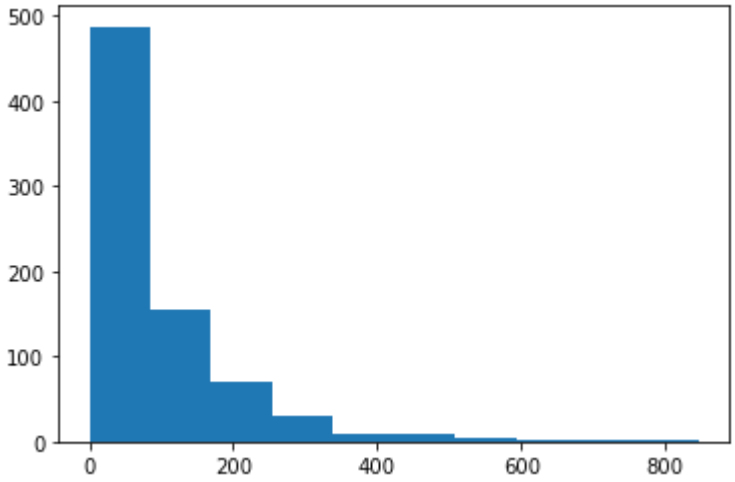
Out[31]:
```
0
```

In [32]:
```
(hc_data['Insulin']==0).sum()
```

Out[32]:
```
374
```

In [33]:
```
plt.hist(hc_data['Insulin'])
```

Out[33]:
```
(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. ]),
 <BarContainer object of 10 artists>)
```



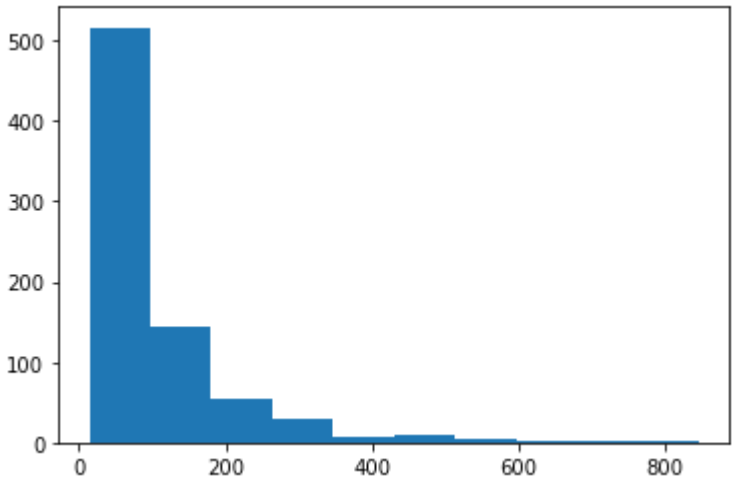
In [34]:
```
Insulin_mean=hc_data['Insulin'].mean()
Insulin_mean
```

Out[34]:
```
79.79947916666667
```

In [35]:
```
hc_data['Insulin']=hc_data['Insulin'].replace(0,Insulin_mean)
```

In [36]:
```
plt.hist(hc_data['Insulin'])
```

Out[36]:
```
(array([516., 143.,  55.,  29.,   7.,  10.,   4.,   1.,   2.,   1.]),
 array([ 14. ,  97.2, 180.4, 263.6, 346.8, 430. , 513.2, 596.4, 679.6,
        762.8, 846. ]),
 <BarContainer object of 10 artists>)
```



In [37]:
```
(hc_data['Insulin']==0).sum()
```

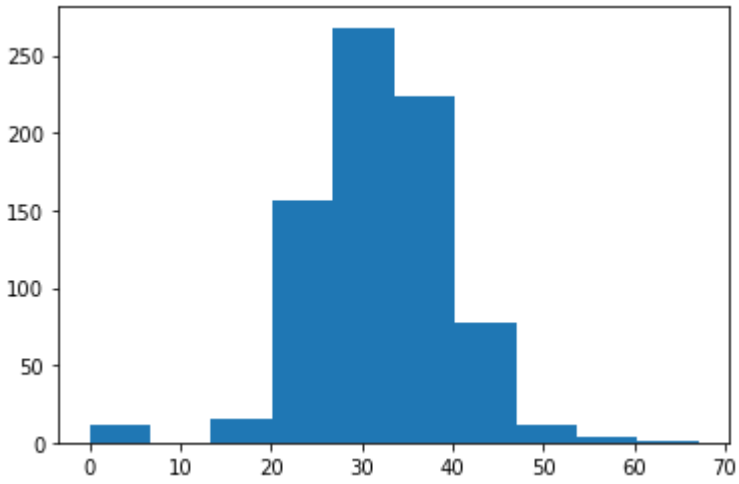
Out[37]:
```
0
```

In [38]:
```
(hc_data['BMI']==0).sum()
```

Out[38]:
```
11
```

In [39]:
```
plt.hist(hc_data['BMI'])
```

Out[39]:
```
(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 ]),
 <BarContainer object of 10 artists>)
```



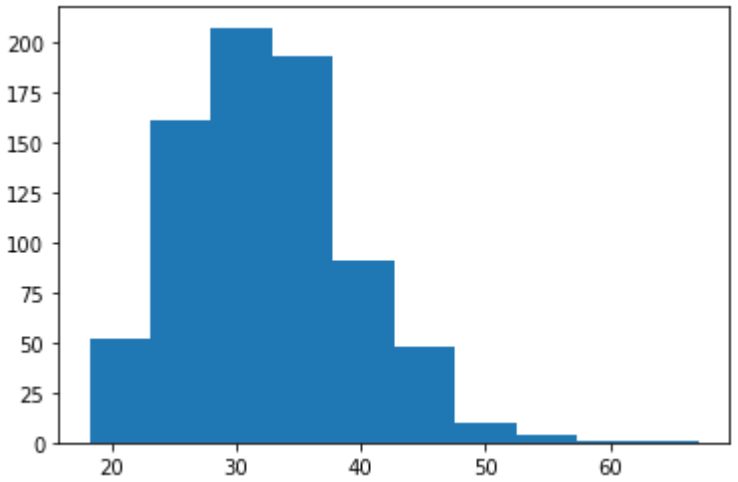
In [40]:
```
BMI_mean=hc_data['BMI'].mean()
BMI_mean
```

Out[40]:
```
31.992578124999977
```

In [41]:
```
hc_data['BMI']=hc_data['BMI'].replace(0,BMI_mean)
```

In [42]:
```
plt.hist(hc_data['BMI'])
```

Out[42]:
```
(array([ 52., 161., 207., 193.,  91.,  48.,  10.,   4.,   1.,   1.]),
 array([18.2 , 23.09, 27.98, 32.87, 37.76, 42.65, 47.54, 52.43, 57.32,
        62.21, 67.1 ]),
 <BarContainer object of 10 artists>)
```



In [43]:
```
(hc_data['BMI']==0).sum()
```

Out[43]:
```
0
```

In other 4 cases too i.e.BloodPressure, SkinThickness, Insulin, BMI. We replaced the null values with mean of their specific column as shown in their respective histogram.

In [44]:
```
import seaborn as sns
```

In [45]:
```
hc_data.dtypes
```

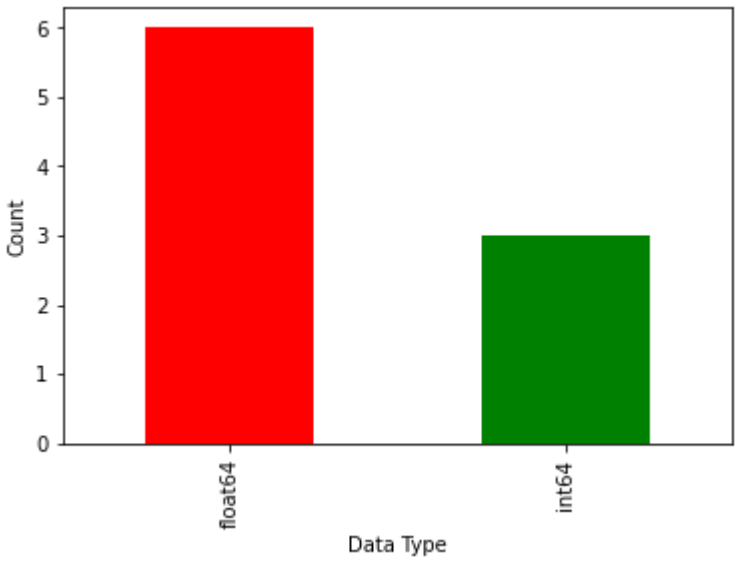
Out[45]:
```
Pregnancies                 int64
Glucose                   float64
BloodPressure             float64
SkinThickness             float64
Insulin                   float64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

In [46]:
```
hc_data.dtypes.value_counts()
```

Out[46]:
```
float64    6
int64      3
dtype: int64
```

In [47]:
```
hc_data.dtypes.value_counts().plot(kind='bar', color=['r','g'])
plt.xlabel('Data Type')
plt.ylabel("Count")
```

Out[47]:
```
Text(0, 0.5, 'Count')
```



Count Plot as shown above. It shows the no of count of types avaliable in the given dataset.

END OF WEEK 1

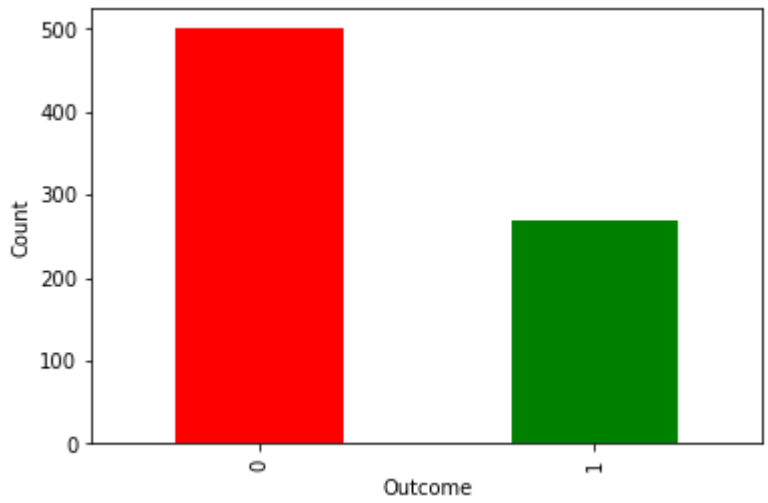
Project Task: Week 2

Data Exploration:

1. Check the balance of the data by plotting the count of outcomes by their value. Describe your findings and plan future course of action.
2. Create scatter charts between the pair of variables to understand the relationships. Describe your findings.
3. Perform correlation analysis. Visually explore it using a heat map.

In [48]:
```
hc_data['Outcome'].value_counts().plot(kind='bar', color=['r','g'])
plt.xlabel('Outcome')
plt.ylabel("Count")
```

Out[48]:
```
Text(0, 0.5, 'Count')
```



The Count of Outcome column is shown in above figure.

Creation of scatter plot to check the relation ship between variables is given below.

In [49]:

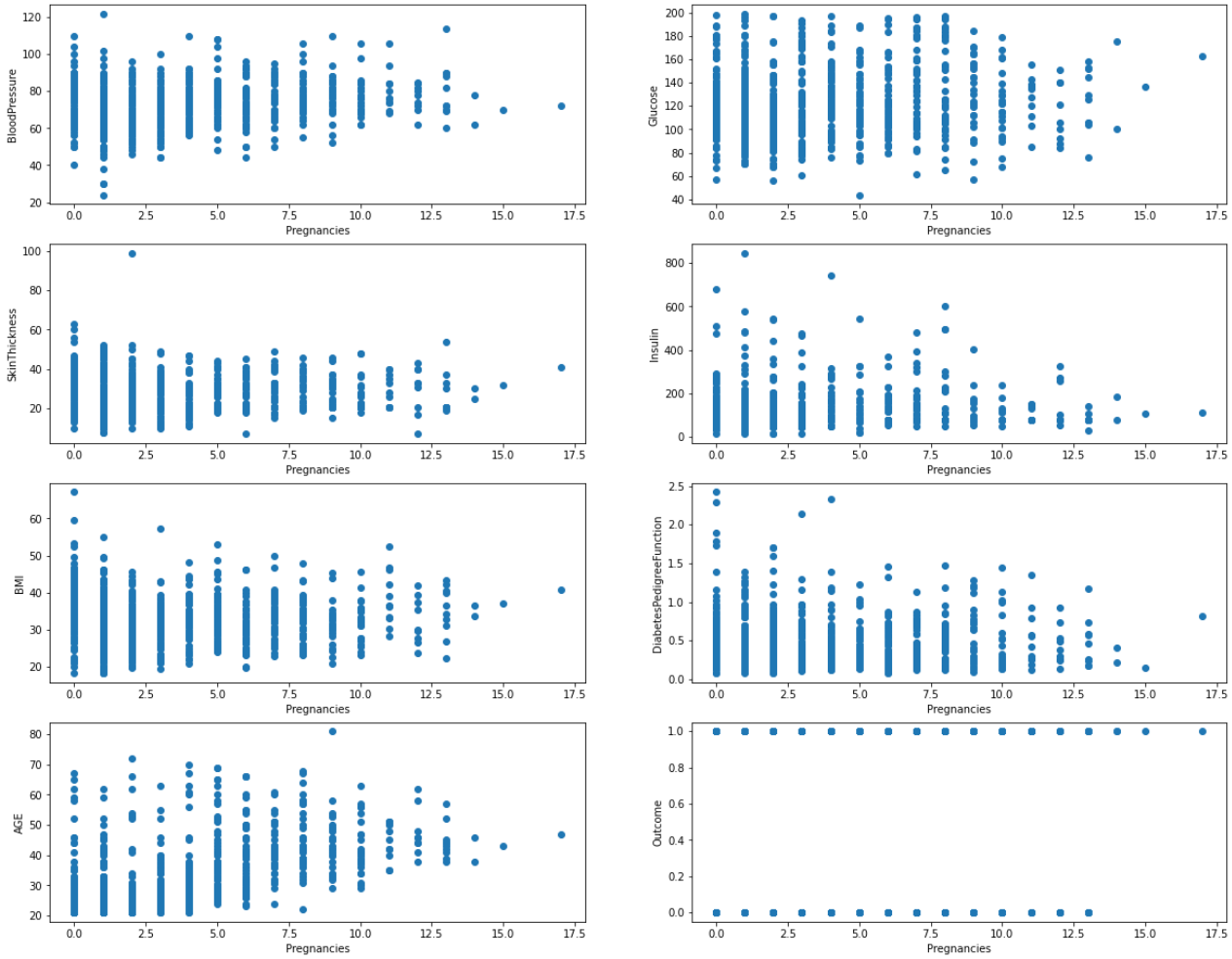
```
BP=hc_data['BloodPressure']
Gl=hc_data['Glucose']
```

In [50]:

```
PR=hc_data['Pregnancies']
BP=hc_data['BloodPressure']
Gl=hc_data['Glucose']
ST=hc_data['SkinThickness']
IN=hc_data['Insulin']
BMI=hc_data['BMI']
DPF=hc_data['DiabetesPedigreeFunction']
AGE=hc_data['Age']
OUTC=hc_data['Outcome']
```

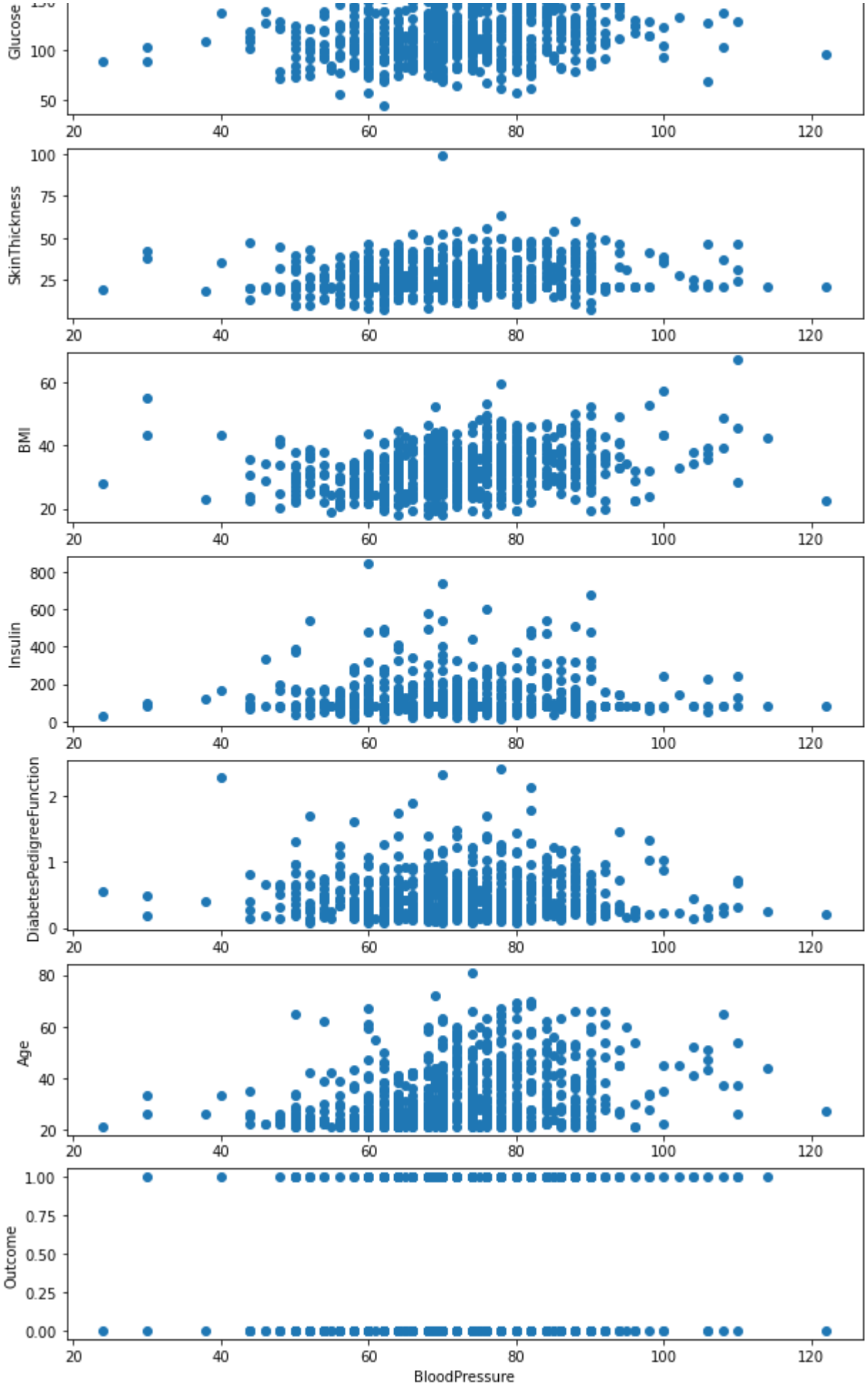
In [51]:

```
fig,ax=plt.subplots(4,2,figsize=(20,16))
ax[0,0].scatter(PR,BP)
ax[0,0].set_xlabel("Pregnancies")
ax[0,0].set_ylabel("BloodPressure")

ax[0,1].scatter(PR,Gl)
ax[0,1].set_xlabel("Pregnancies")
ax[0,1].set_ylabel("Glucose")

ax[1,0].scatter(PR,ST)
ax[1,0].set_xlabel("Pregnancies")
ax[1,0].set_ylabel("SkinThickness")

ax[1,1].scatter(PR,IN)
ax[1,1].set_xlabel("Pregnancies")
ax[1,1].set_ylabel("Insulin")

ax[2,0].scatter(PR,BMI)
ax[2,0].set_xlabel("Pregnancies")
ax[2,0].set_ylabel("BMI")

ax[2,1].scatter(PR,DPF)
ax[2,1].set_xlabel("Pregnancies")
ax[2,1].set_ylabel("DiabetesPedigreeFunction")

ax[3,0].scatter(PR,AGE)
ax[3,0].set_xlabel("Pregnancies")
ax[3,0].set_ylabel("AGE")

ax[3,1].scatter(PR,OUTC)
ax[3,1].set_xlabel("Pregnancies")
ax[3,1].set_ylabel("Outcome")
plt.show()
```



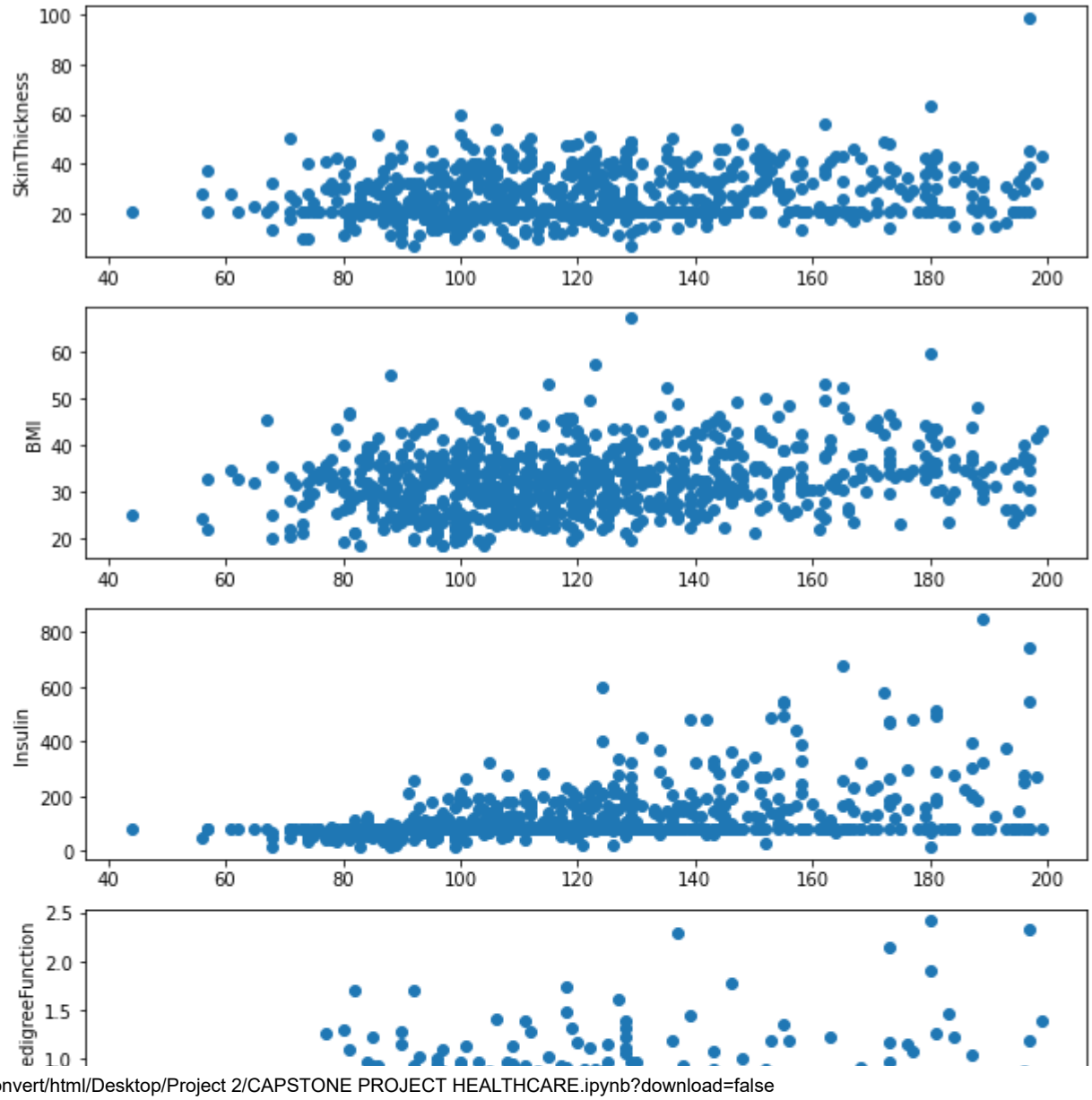
In [52]:

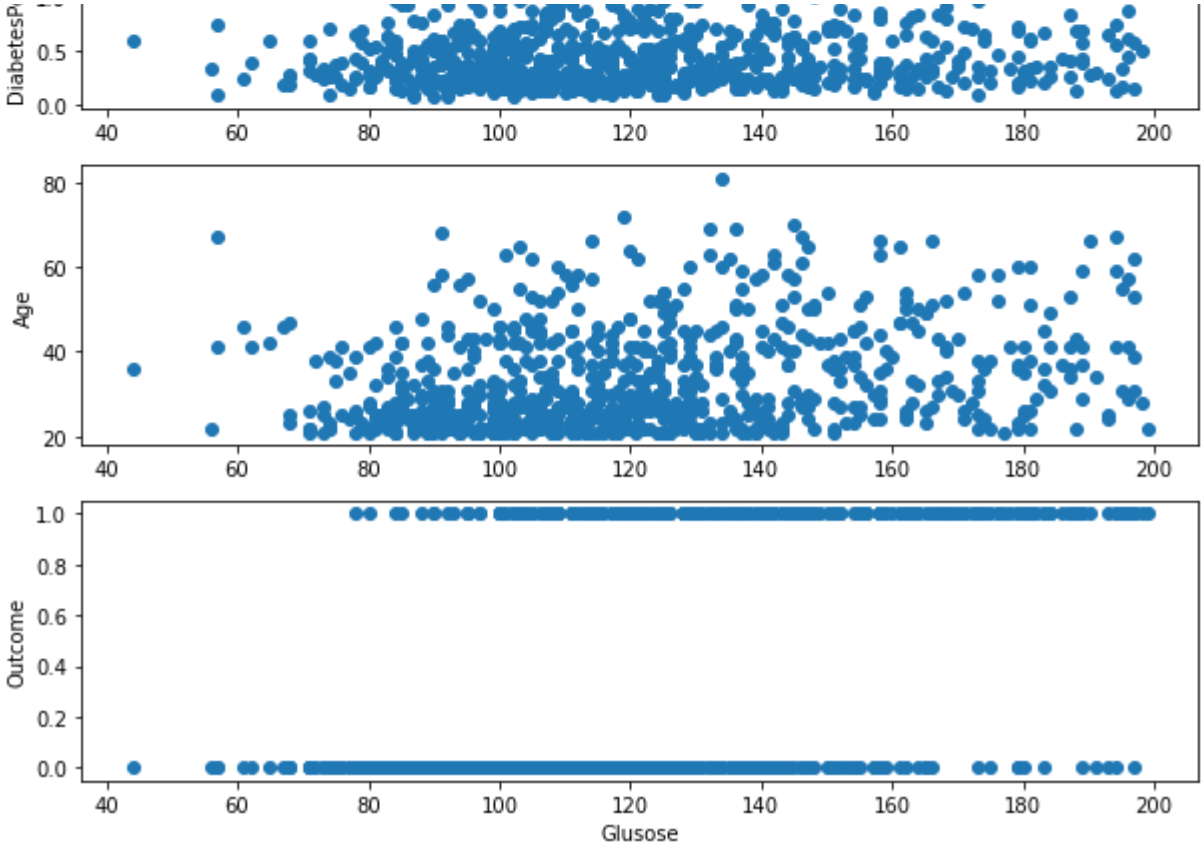
```
fig,ax=plt.subplots(7,figsize=(10,18))
ax[0].scatter(BP,Gl)
plt.xlabel("BloodPressure")
ax[0].set_ylabel("Glucose")

ax[1].scatter(BP,ST)
ax[1].set_ylabel("SkinThickness")

ax[2].scatter(BP,BMI)
ax[2].set_ylabel("BMI")

ax[3].scatter(BP,IN)
ax[3].set_ylabel("Insulin")

ax[4].scatter(BP,DPF)
ax[4].set_ylabel("DiabetesPedigreeFunction")

ax[5].scatter(BP,AGE)
ax[5].set_ylabel("Age")

ax[6].scatter(BP,OUTC)
ax[6].set_ylabel("Outcome")

plt.show()
```


Glucose
SkinThickness
BMI
Insulin
DiabetesPedigreeFunction
Age
Outcome
BloodPressure

In [53]:

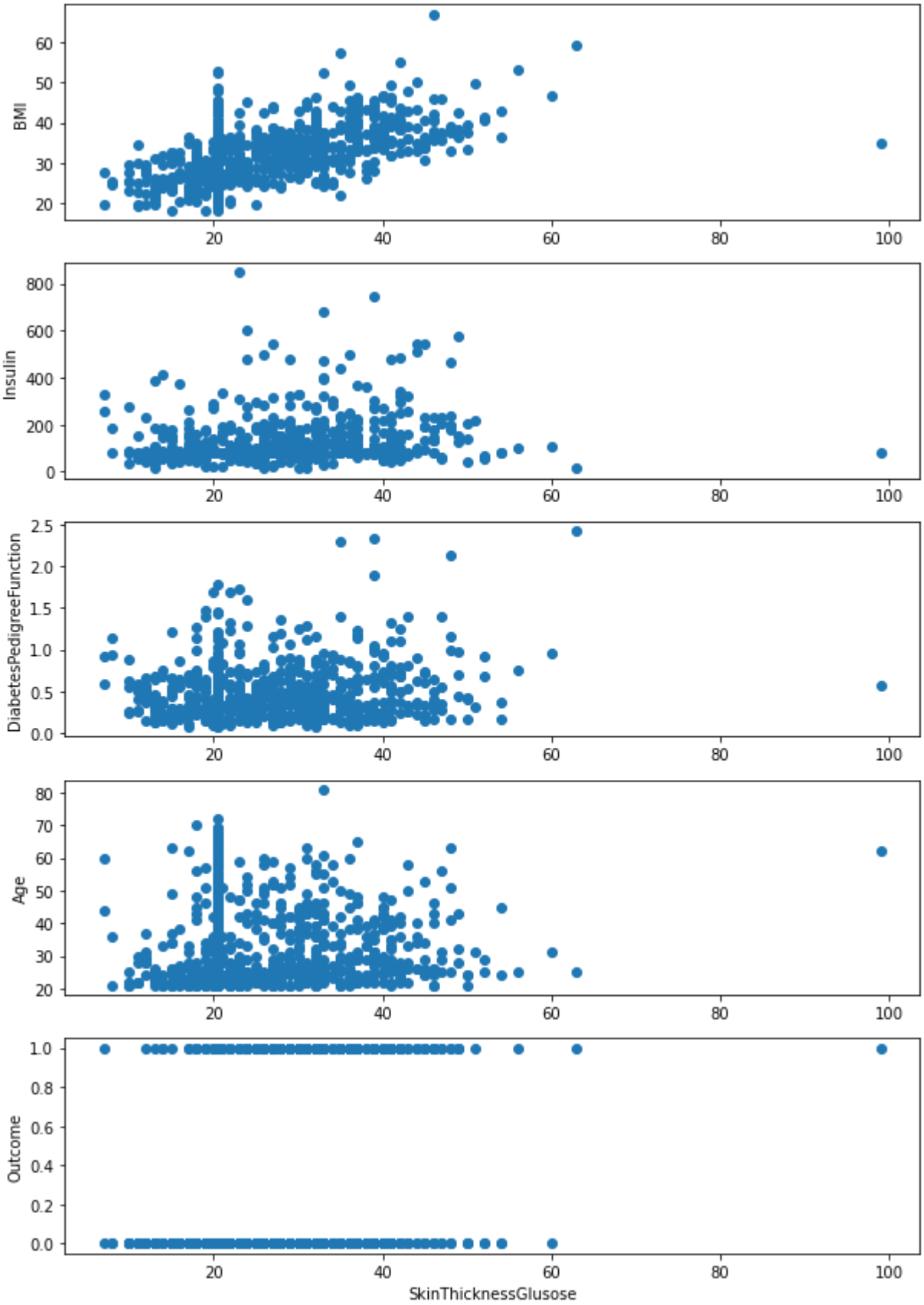
```
fig,ax=plt.subplots(6,figsize=(10,18))
ax[0].scatter(Gl,ST)
plt.xlabel("Glusose")
ax[0].set_ylabel("SkinThickness")

ax[1].scatter(Gl,BMI)
ax[1].set_ylabel("BMI")

ax[2].scatter(Gl,IN)
ax[2].set_ylabel("Insulin")

ax[3].scatter(Gl,DPF)
ax[3].set_ylabel("DiabetesPedigreeFunction")

ax[4].scatter(Gl,AGE)
ax[4].set_ylabel("Age")

ax[5].scatter(Gl,OUTC)
ax[5].set_ylabel("Outcome")

plt.show()
```





In [54]:

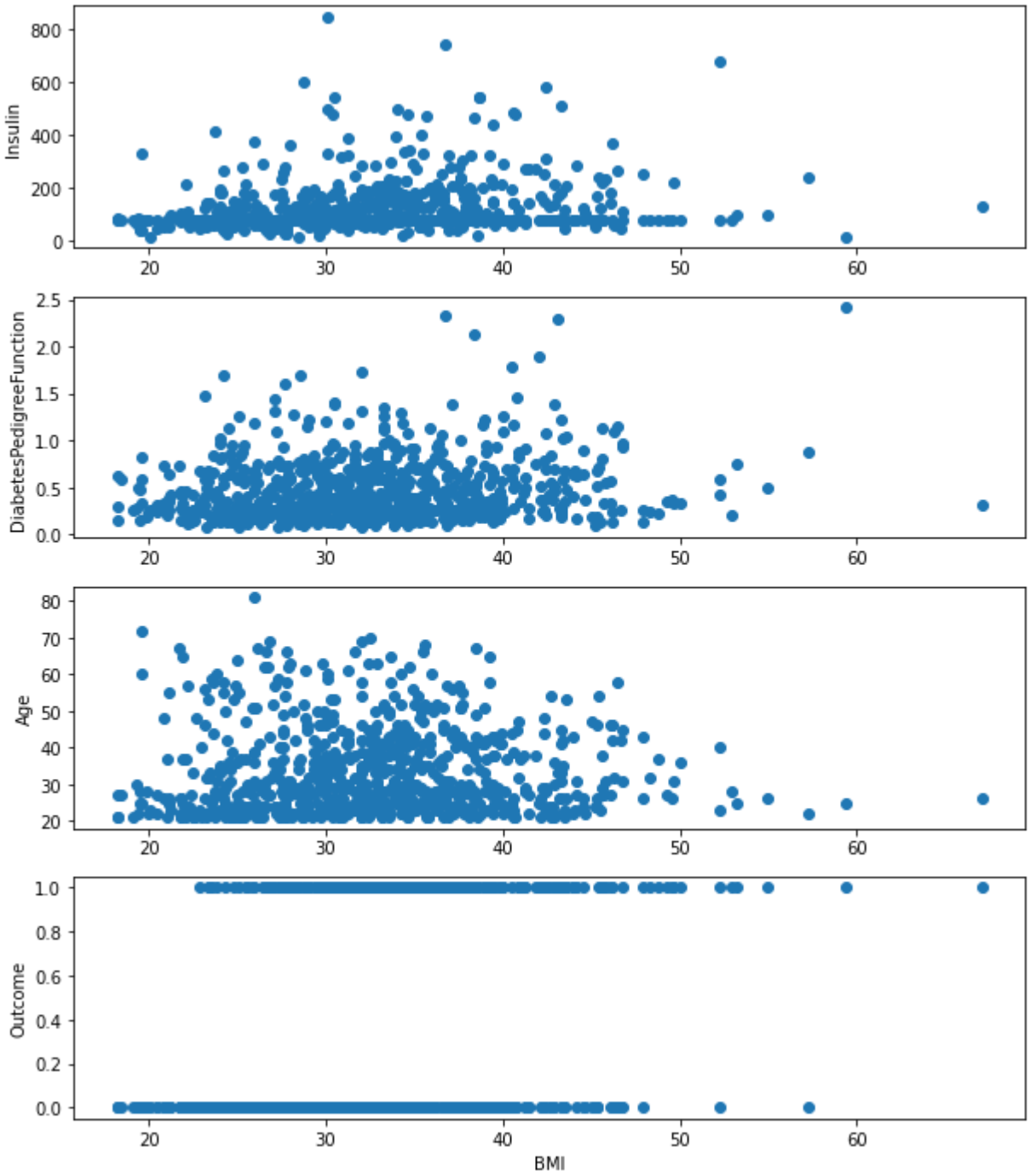
```
fig,ax=plt.subplots(5,figsize=(10,15))
ax[0].scatter(ST,BMI)
plt.xlabel("SkinThicknessGlusose")
ax[0].set_ylabel("BMI")

ax[1].scatter(ST,IN)
ax[1].set_ylabel("Insulin")

ax[2].scatter(ST,DPF)
ax[2].set_ylabel("DiabetesPedigreeFunction")

ax[3].scatter(ST,AGE)
ax[3].set_ylabel("Age")

ax[4].scatter(ST,OUTC)
ax[4].set_ylabel("Outcome")

plt.show()
```



In [55]:

```
fig,ax=plt.subplots(4,figsize=(10,12))
ax[0].scatter(BMI,IN)
plt.xlabel("BMI")
ax[0].set_ylabel("Insulin")
```

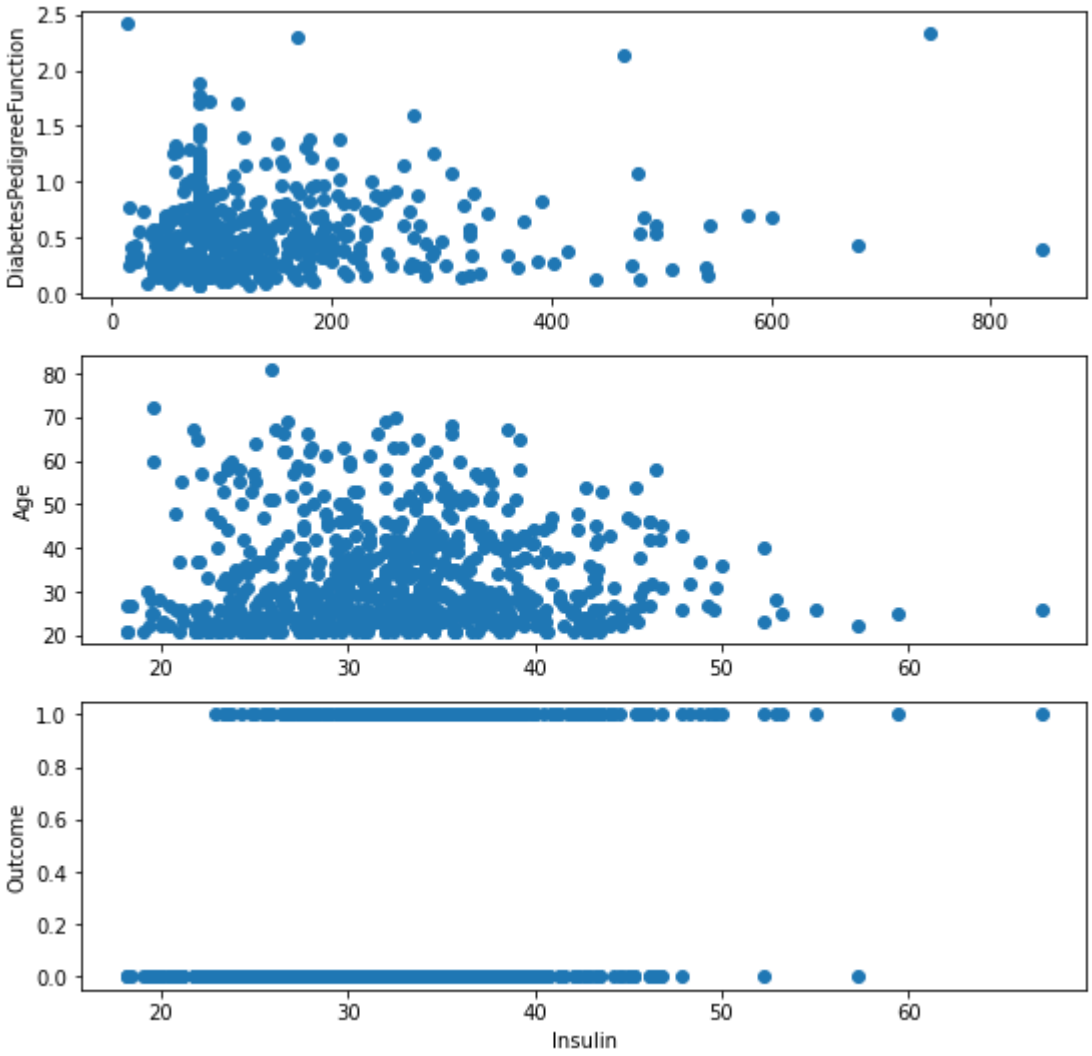
```
ax[1].scatter(BMI,DPF)
ax[1].set_ylabel("DiabetesPedigreeFunction")

ax[2].scatter(BMI,AGE)
ax[2].set_ylabel("Age")

ax[3].scatter(BMI,OUTC)
ax[3].set_ylabel("Outcome")

plt.show()
```



In [56]:

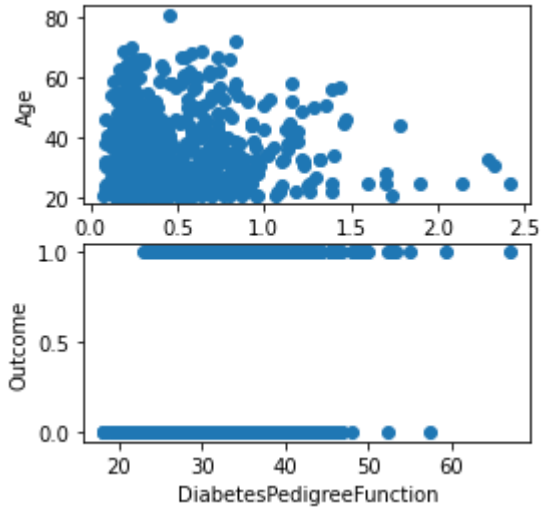
```
fig,ax=plt.subplots(3,figsize=(9,9))
ax[0].scatter(IN,DPF)
plt.xlabel("Insulin")
ax[0].set_ylabel("DiabetesPedigreeFunction")
```

```
ax[1].scatter(BMI,AGE)
ax[1].set_ylabel("Age")

ax[2].scatter(BMI,OUTC)
ax[2].set_ylabel("Outcome")

plt.show()
```

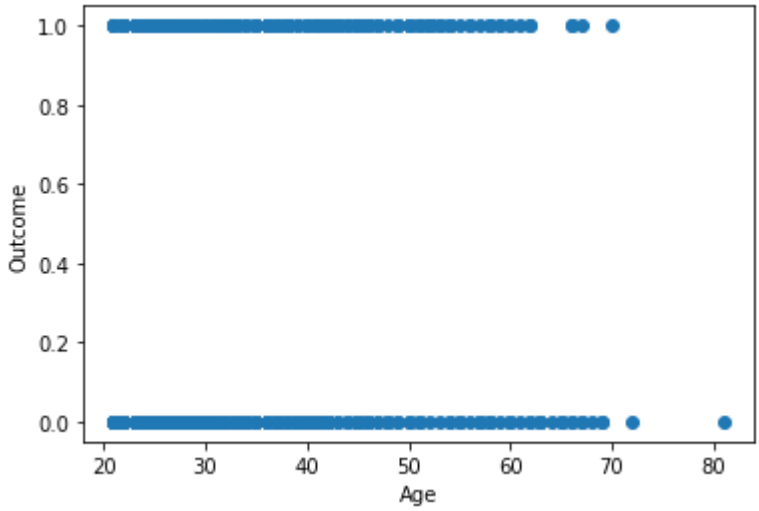


In [57]:

```
fig,ax=plt.subplots(2,figsize=(4,4))
ax[0].scatter(DPF,AGE)
plt.xlabel("DiabetesPedigreeFunction")
ax[0].set_ylabel("Age")

ax[1].scatter(BMI,OUTC)
ax[1].set_ylabel("Outcome")

plt.show()
```



In [58]:
```
plt.scatter(AGE,OUTC)
plt.xlabel("Age")
plt.ylabel("Outcome")
plt.show()
```



The scatter plot shows that their is no direct realtionship between any variable and thats why we dont need to drop any variable or column.

Creating correlation heat map as per instructions.

In [59]:
```
hc_data_corr=hc_data.corr()
hc_data_corr
```

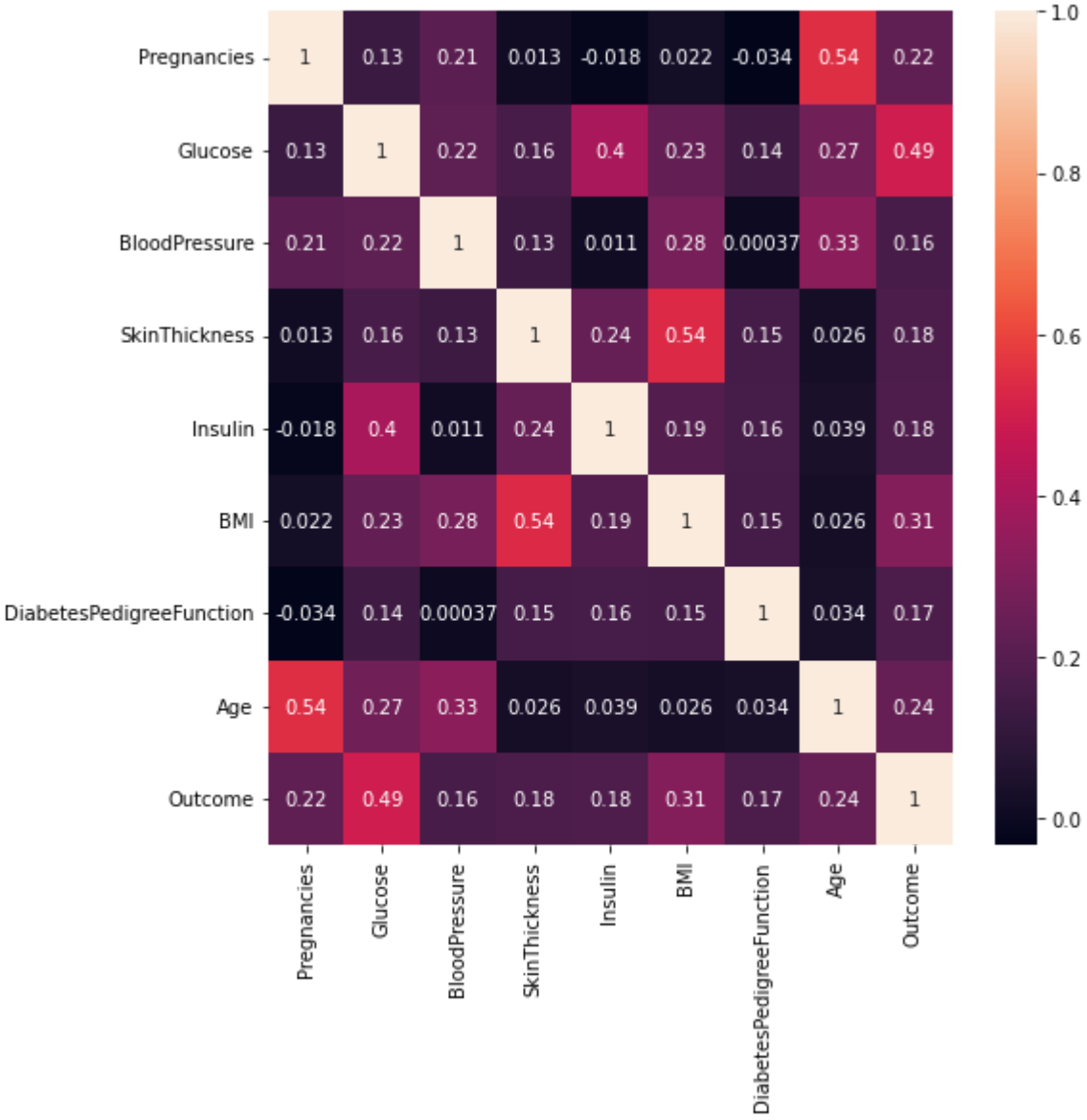
Out[59]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Dia

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Dia
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 [60]:
```
plt.subplots(figsize=(8,8))
sns.heatmap(hc_data_corr, annot=True)
```

Out[60]:
```
<AxesSubplot:>
```



The correlation table and heat map shows that thier is medium level correlation between "Preganices & Age", "Skinthickness & BMI", 'Glucose & Outcome" and "Insulin & Glucose".

END OF WEEK 2

Project Task: Week 3 & Week 4

Data Modeling:

1. Devise strategies for model building. It is important to decide the right validation framework. Express your thought process.

2. Apply an appropriate classification algorithm to build a model. Compare various models with the results from KNN algorithm.

Data Modeling:

1. Create a classification report by analyzing sensitivity, specificity, AUC (ROC curve), etc. Please be descriptive to explain what values of these parameter you have used.

1st we create a KNN Algorithm model and find the values of accuracy score, confusion matrix, classification_report,roc_curve,roc_auc_score and then value of same parameters using other classification algrotihms like Logistic Regression, Decision Forest, Random Forest. In the end of compare the values of different parameters and make a comparatove analysis between them.

KNN ALGORITHM

In [61]:
```
from sklearn.model_selection import train_test_split
```

In [62]:
```
hc_data.head()
```

Out[62]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148.0	72.0	35.000000	79.799479	33.6	0.627	50
1	1	85.0	66.0	29.000000	79.799479	26.6	0.351	31
2	8	183.0	64.0	20.536458	79.799479	23.3	0.672	32
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167	21
4	0	137.0	40.0	35.000000	168.000000	43.1	2.288	33

In [63]:
```
X=hc_data.drop(['Outcome'],axis=1)
y=hc_data['Outcome']
```

In [64]:
```
X.head()
```

Out[64]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148.0	72.0	35.000000	79.799479	33.6	0.627	5(
1	1	85.0	66.0	29.000000	79.799479	26.6	0.351	31
2	8	183.0	64.0	20.536458	79.799479	23.3	0.672	32
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167	21
4	0	137.0	40.0	35.000000	168.000000	43.1	2.288	33

In [65]:
```
y.head()
```

Out[65]:
```
0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
```

Now we use train_test_split fucntion to split the data into train and test datasets with train:test ratio of 80:20.

In [66]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.2,random_state=2)
```

In [67]:
```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[67]:
```
((614, 8), (154, 8), (614,), (154,))
```

In [68]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score,roc_c
```

In [69]:
```
knn=KNeighborsClassifier(n_neighbors=45)
knn.fit(X_train,y_train)
knn_pred=knn.predict(X_test)
```

In [70]:
```
knn_pred
```

Out[70]:
```
array([0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
      dtype=int64)
```

In [71]:
```
accuracy_score(y_test,knn_pred)
```

Out[71]:
```
0.7987012987012987
```

In [72]:
```
confusion_matrix(y_test,knn_pred)
```

Out[72]:
```
array([[97, 12],
       [19, 26]], dtype=int64)
```

In [73]:
```
print(classification_report(y_test,knn_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.89      0.86       109
           1       0.68      0.58      0.63        45

    accuracy                           0.80       154
   macro avg       0.76      0.73      0.74       154
weighted avg       0.79      0.80      0.79       154
```

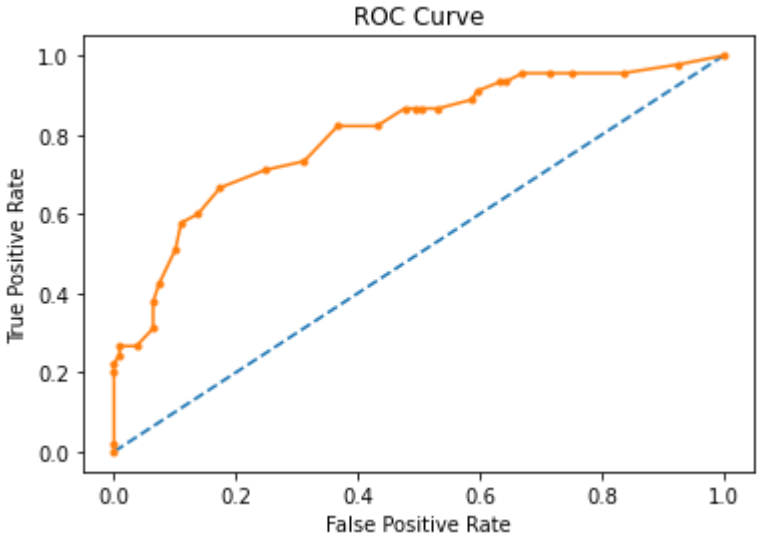
In [74]:
```
knn_prob=knn.predict_proba(X_test)
knn_prob_postive=knn_prob[:,1]
```

In [75]:
```
knn_auc=roc_auc_score(y_test,knn_prob_postive)
knn_auc
```

Out[75]:
```
0.801427115188583
```

In [76]:
```
fpr,tpr,thresh=roc_curve(y_test,knn_prob_postive)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr,tpr,marker='.')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

Out[76]:
```
Text(0, 0.5, 'True Positive Rate')
```



The AUC score and accuracy score of KNN model is 80.14% & 79.81% respectively.

LOGISTIC REGRESSION

In [77]:
```
from sklearn.linear_model import LogisticRegression
```

In [78]:
```
lr=LogisticRegression()
```

In [79]:
```
lr.fit(X_train,y_train)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[79]:
```
LogisticRegression()
```

In [80]:
```
lr.intercept_
```

Out[80]:
```
array([-6.57253935])
```

In [81]:
```
lr.coef_
```

Out[81]:
```
array([[ 1.67744010e-01,  3.60231239e-02, -3.04220070e-02,
        -1.79394656e-03, -1.51426340e-03,  7.20932793e-02,
         2.13646087e+00, -3.45069500e-03]])
```

In [82]:
```
lr_pred=lr.predict(X_test)
```

In [83]:
```
lr_pred
```

Out[83]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      dtype=int64)
```

In [84]:
```
accuracy_score(y_test,lr_pred)
```

Out[84]:
```
0.7597402597402597
```

In [85]:
```
confusion_matrix(y_test,lr_pred)
```

Out[85]:
```
array([[96, 13],
       [24, 21]], dtype=int64)
```

In [86]:
```
print(classification_report(y_test,lr_pred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.88      0.84       109
           1       0.62      0.47      0.53        45

    accuracy                           0.76       154
   macro avg       0.71      0.67      0.69       154
weighted avg       0.75      0.76      0.75       154
```

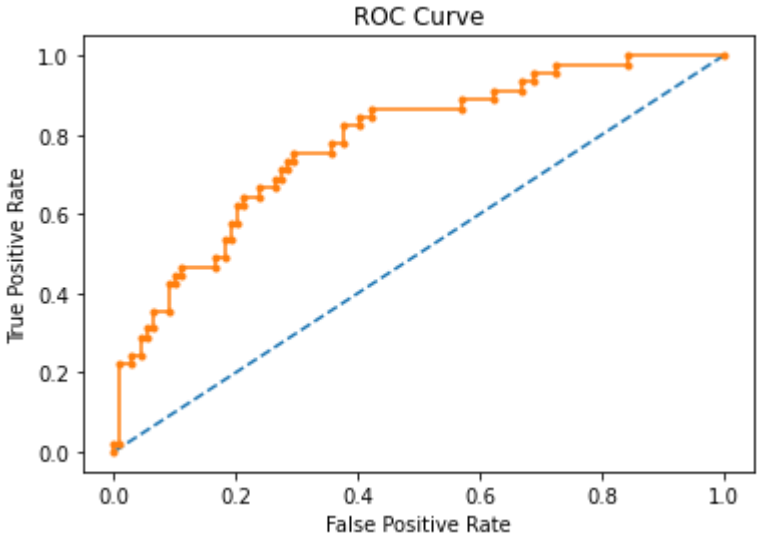
In [87]:
```
lr_prob=lr.predict_proba(X_test)
lr_prob_postive=lr_prob[:,1]
```

In [88]:
```
lr_auc=roc_auc_score(y_test,lr_prob_postive)
lr_auc
```

Out[88]:
```
0.781855249745158
```

In [89]:
```
fpr,tpr,thresh=roc_curve(y_test,lr_prob_postive)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr,tpr,marker='.')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

Out[89]:
```
Text(0, 0.5, 'True Positive Rate')
```



The AUC score and accuracy score of Logistic Regression model is 78.18% & 75.97% respectively.

DECISION TREE

In [90]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [91]:
```
dt=DecisionTreeClassifier(max_depth=5)
```

In [92]:
```
dt.fit(X_train,y_train)
```

Out[92]:
```
DecisionTreeClassifier(max_depth=5)
```

In [93]:
```
dt_pred=dt.predict(X_test)
```

In [94]:
```
dt_pred
```

Out[94]:
```
array([0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
      dtype=int64)
```

In [95]:
```
accuracy_score(y_test,dt_pred)
```

Out[95]:
```
0.7662337662337663
```

In [96]:
```
confusion_matrix(y_test,dt_pred)
```

Out[96]:
```
array([[92, 17],
       [19, 26]], dtype=int64)
```

In [97]:
```
print(classification_report(y_test,dt_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.84      0.84       109
           1       0.60      0.58      0.59        45

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

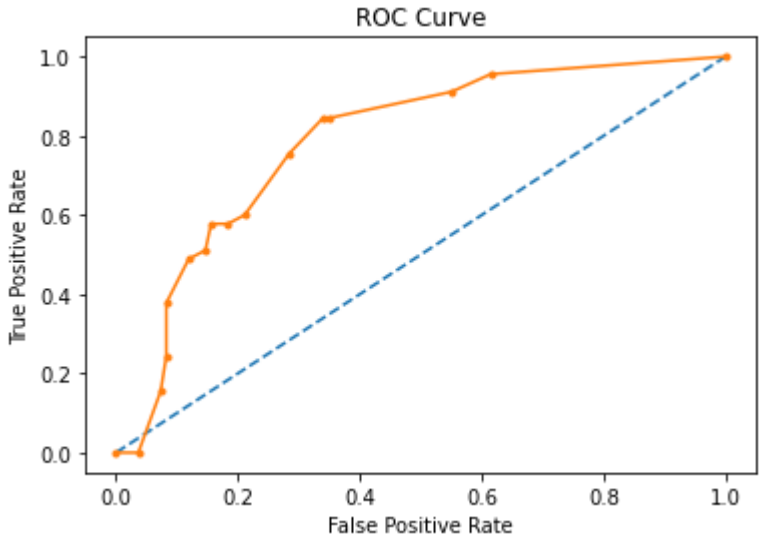
In [98]:
```
dt_prob=dt.predict_proba(X_test)
dt_prob_postive=dt_prob[:,1]
```

In [99]:
```
dt_auc=roc_auc_score(y_test,dt_prob_postive)
dt_auc
```

Out[99]:
```
0.7868501529051988
```

In [100…
```
fpr,tpr,thresh=roc_curve(y_test,dt_prob_postive)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr,tpr,marker='.')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

Out[100…
```
Text(0, 0.5, 'True Positive Rate')
```



The AUC score and accuracy score of Decision

Tree model is 79.12% & 77.27% respectively.

RANDOM FOREST

In [101…
```
from sklearn.ensemble import RandomForestClassifier
```

In [102…
```
rp=RandomForestClassifier(n_estimators=10)
```

In [103…
```
rp.fit(X_train,y_train)
```

Out[103…
```
RandomForestClassifier(n_estimators=10)
```

In [104…
```
rp_pred=rp.predict(X_test)
```

In [105…
```
rp_pred
```

Out[105…
```
array([0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0,
       1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
      dtype=int64)
```

In [106…
```
accuracy_score(y_test,rp_pred)
```

Out[106…
```
0.7467532467532467
```

In [107…
```
confusion_matrix(y_test,rp_pred)
```

Out[107…
```
array([[88, 21],
       [18, 27]], dtype=int64)
```

In [108…
```
print(classification_report(y_test,rp_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.81      0.82       109
           1       0.56      0.60      0.58        45

    accuracy                           0.75       154
   macro avg       0.70      0.70      0.70       154
weighted avg       0.75      0.75      0.75       154
```

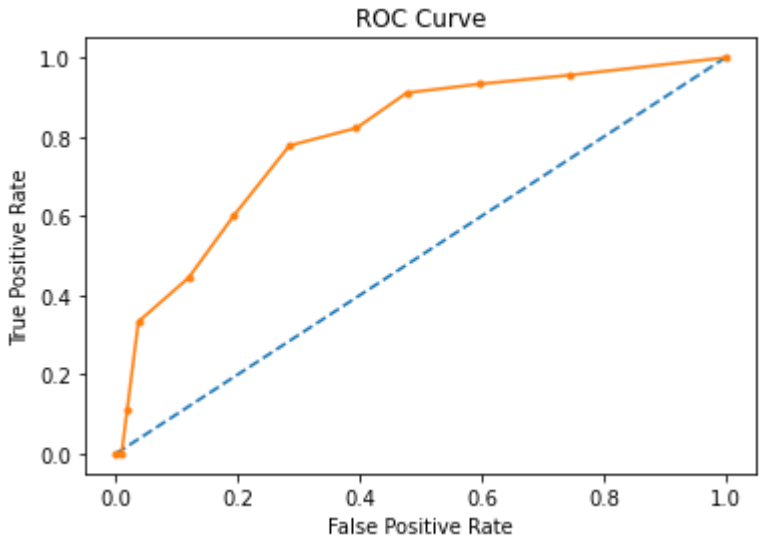
In [109…
```
rp_prob=rp.predict_proba(X_test)
rp_prob_postive=rp_prob[:,1]
```

In [110…
```
rp_auc=roc_auc_score(y_test,rp_prob_postive)
rp_auc
```

Out[110…
```
0.7976554536187563
```

In [111…
```
fpr,tpr,thresh=roc_curve(y_test,rp_prob_postive)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr,tpr,marker='.')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

Out[111…
```
Text(0, 0.5, 'True Positive Rate')
```



The AUC score and accuracy score of Random Forest model is 82.04% & 77.27% respectively.

As we can see from the above data the AUC score is maximum in the case of "Random Forest" while the accuracy score is highest in the case of "KNN Model".

END OF WEEK 3 & WEEK 4

Now we safe the new dataset into excel from for the Tableau Work

In [112…

```
hc_data.to_excel('HealthCare.xlsx') #to save data in the harddisk for the tableau work
```