In [1]:
```
# importing required libraries
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
```

Week 1

In [2]:
```
# Load data set for processing and analysis
data = pd.read_csv("C:\MySpace\Data_Science\Capstone\Project2\health care diabetes.csv")
```

In [3]:
```
# Data mining to understand the data and take necessary action w.r.t. data correction
```

In [4]:
```
data.head()
```

Out[4]:

	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 [5]:
```
data.describe()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

In [6]:
```
data.corr()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

In [7]:
```
#required Column to Update for Null (0) Values
Sel_Col=['Glucose','BloodPressure','SkinThickness','Insulin','BMI']
Sel_Col
```

Out[7]:
```
['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
```

In [8]:
```
# Count of zero in fields ,Interested columns are ['Glucose','BloodPressure','SkinThickness','Insulin','BMI']
Col_Zero_Val=data[data[Sel_Col]==0].count()
Col_Zero_Val
```

Out[8]:
```
Pregnancies                  0
Glucose                      5
BloodPressure               35
SkinThickness              227
Insulin                    374
BMI                         11
DiabetesPedigreeFunction     0
Age                          0
Outcome                      0
dtype: int64
```

In [9]:
```
#Updating required column with their respective Median value
# We are not considering Mean as there are rows present with "0" values and due to same mean values are not the correct value

for i in data.columns:
    if (i in Sel_Col):
        m=data[data[i]!=0][i].median()
#         print(i)
#         print(m)
        data[i]=data[i].apply(lambda x: m if x==0 else x)

data.head()
```

Out[9]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	125.0	33.6	0.627	50	1
1	1	85.0	66.0	29.0	125.0	26.6	0.351	31	0
2	8	183.0	64.0	29.0	125.0	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 [10]:
```
# ReChecking post updating null value with median
Col_Zero_Val=data[data[Sel_Col]==0].count()
```

```
Col_Zero_Val
```

Out[10]:

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

In [11]:

```
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    float64
 2   BloodPressure             768 non-null    float64
 3   SkinThickness             768 non-null    float64
 4   Insulin                   768 non-null    float64
 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(6), int64(3)
memory usage: 54.1 KB
```

In [12]:

```
Positive = data[data['Outcome']==1]
Positive.head(5)
```

Out[12]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	125.0	33.6	0.627	50	1
2	8	183.0	64.0	29.0	125.0	23.3	0.672	32	1

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53	1

In [13]:
```
data[Sel_Col].value_counts()
```

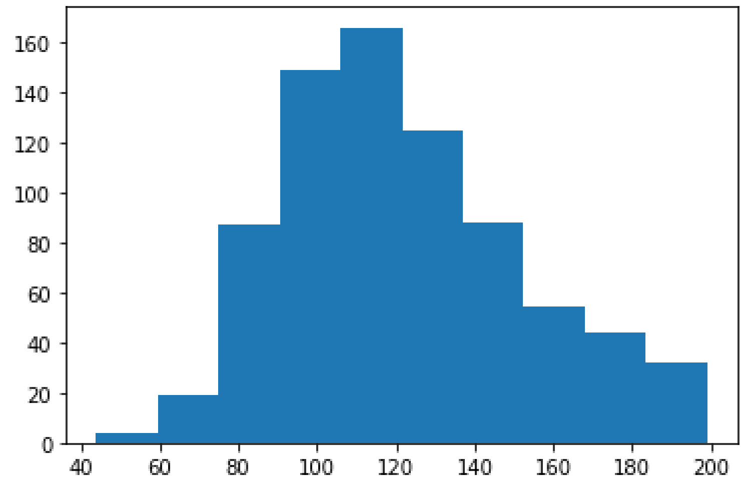
Out[13]:
```
Glucose  BloodPressure  SkinThickness  Insulin  BMI
44.0     62.0           29.0           125.0    25.0    1
56.0     56.0           28.0           45.0     24.2    1
129.0    86.0           20.0           270.0    35.1    1
         90.0           7.0            326.0    19.6    1
         92.0           49.0           155.0    36.4    1
                                                       ..
105.0    100.0          36.0           125.0    43.3    1
106.0    52.0           29.0           125.0    31.2    1
         54.0           21.0           158.0    30.9    1
         56.0           27.0           165.0    29.0    1
199.0    76.0           43.0           125.0    42.9    1
Length: 768, dtype: int64
```

In [14]:
```
data['Glucose'].value_counts().head(7)
```

Out[14]:
```
99.0     17
100.0    17
117.0    16
129.0    14
125.0    14
106.0    14
111.0    14
Name: Glucose, dtype: int64
```

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

Out[15]:
```
(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>)
```

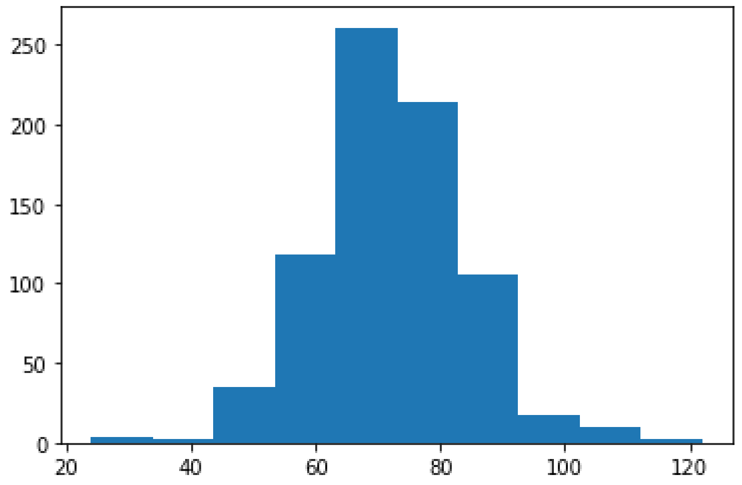


In [16]:
```
data['BloodPressure'].value_counts().head(7)
```

Out[16]:
```
72.0    79
70.0    57
74.0    52
78.0    45
68.0    45
64.0    43
80.0    40
Name: BloodPressure, dtype: int64
```

In [17]:
```
plt.hist(data['BloodPressure'])
```

Out[17]:
```
(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>)
```

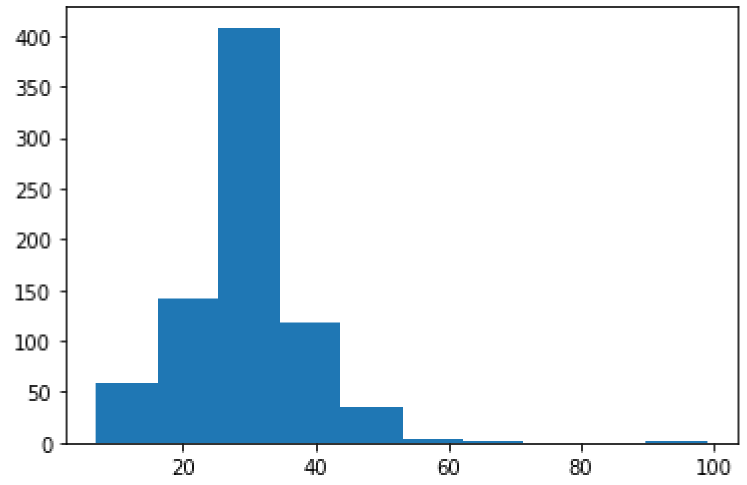


In [18]:
```
data['SkinThickness'].value_counts().head(7)
```

Out[18]:
```
29.0    244
32.0     31
30.0     27
27.0     23
23.0     22
28.0     20
33.0     20
Name: SkinThickness, dtype: int64
```

In [19]:
```
plt.hist(data['SkinThickness'])
```

Out[19]:
```
(array([ 59., 141., 408., 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>)
```

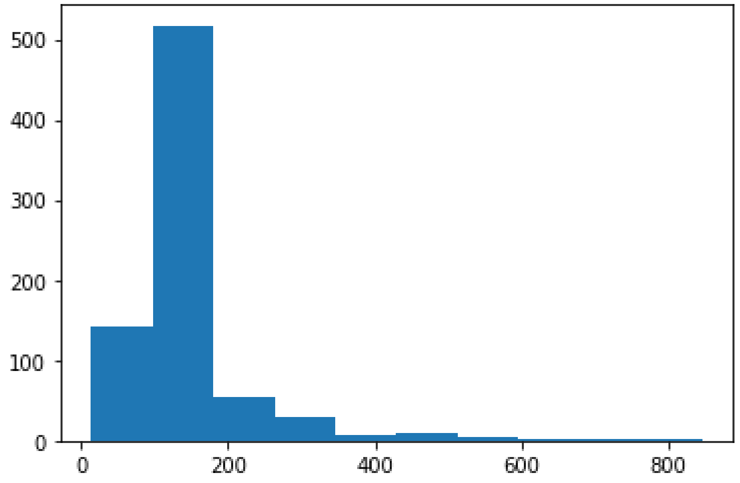


In [20]:
```
data['Insulin'].value_counts().head(7)
```

Out[20]:
```
125.0    378
105.0     11
130.0      9
140.0      9
120.0      8
94.0       7
180.0      7
Name: Insulin, dtype: int64
```

In [21]:
```
plt.hist(data['Insulin'])
```

Out[21]:
```
(array([142., 517.,  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>)
```

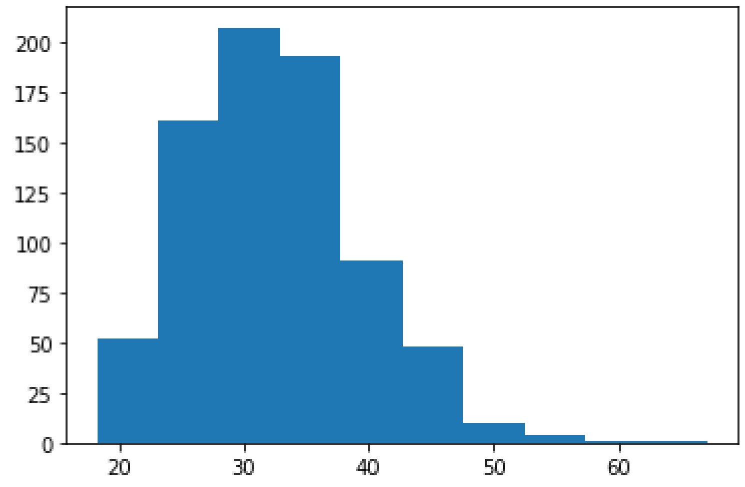


In [22]:
```
data['BMI'].value_counts().head(7)
```

Out[22]:
```
32.3    14
32.0    13
31.6    12
31.2    12
33.3    10
32.4    10
30.1     9
Name: BMI, dtype: int64
```

In [23]:
```
plt.hist(data['BMI'])
```

Out[23]:
```
(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 [24]:
```
data.describe().transpose()
```

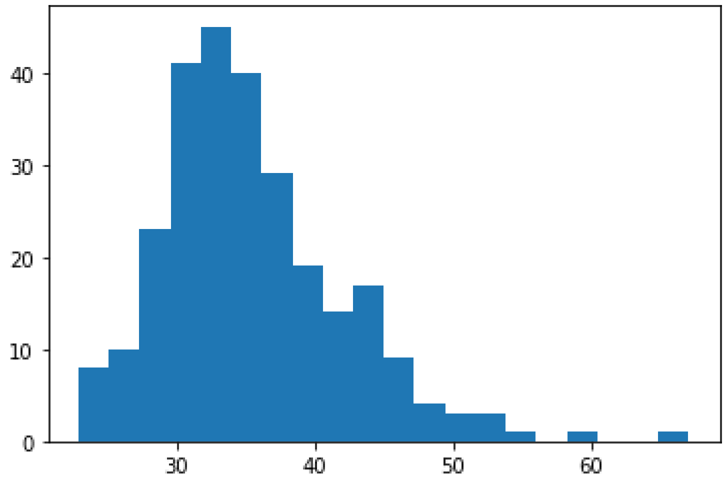
Out[24]:

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	121.656250	30.438286	44.000	99.75000	117.0000	140.25000	199.00
BloodPressure	768.0	72.386719	12.096642	24.000	64.00000	72.0000	80.00000	122.00
SkinThickness	768.0	29.108073	8.791221	7.000	25.00000	29.0000	32.00000	99.00
Insulin	768.0	140.671875	86.383060	14.000	121.50000	125.0000	127.25000	846.00
BMI	768.0	32.455208	6.875177	18.200	27.50000	32.3000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

Week 2

In [25]:
```
plt.hist(Positive['BMI'],histtype='stepfilled',bins=20)
```

Out[25]:
```
(array([ 8., 10., 23., 41., 45., 40., 29., 19., 14., 17.,  9.,  4.,  3.,
         3.,  1.,  0.,  1.,  0.,  0.,  1.]),
 array([22.9 , 25.11, 27.32, 29.53, 31.74, 33.95, 36.16, 38.37, 40.58,
        42.79, 45.  , 47.21, 49.42, 51.63, 53.84, 56.05, 58.26, 60.47,
        62.68, 64.89, 67.1 ]),
 [<matplotlib.patches.Polygon at 0x1cb83ca0>])
```

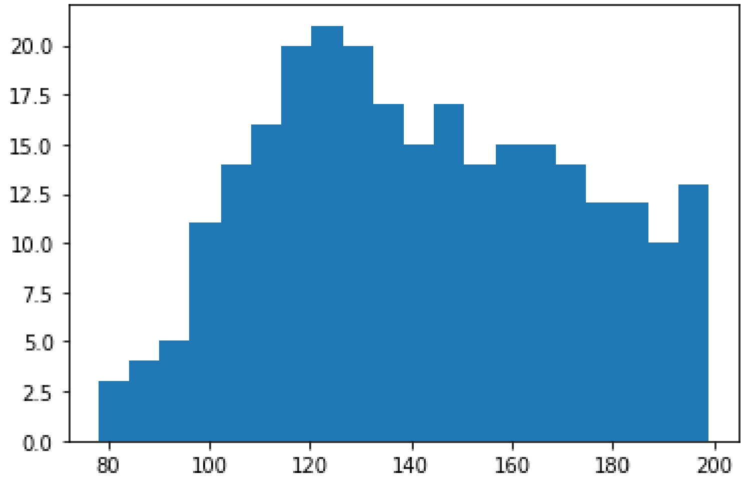


In [26]:
```
Positive['BMI'].value_counts().head(7)
```

Out[26]:
```
32.9    8
31.6    7
33.3    6
31.2    5
30.5    5
32.3    5
32.0    5
Name: BMI, dtype: int64
```

In [27]:
```
plt.hist(Positive['Glucose'],histtype='stepfilled',bins=20)
```

Out[27]:
```
(array([ 3.,  4.,  5., 11., 14., 16., 20., 21., 20., 17., 15., 17., 14.,
        15., 15., 14., 12., 12., 10., 13.]),
 array([ 78.  ,  84.05,  90.1 ,  96.15, 102.2 , 108.25, 114.3 , 120.35,
        126.4 , 132.45, 138.5 , 144.55, 150.6 , 156.65, 162.7 , 168.75,
        174.8 , 180.85, 186.9 , 192.95, 199.  ]),
 [<matplotlib.patches.Polygon at 0x1cbb5bb0>])
```

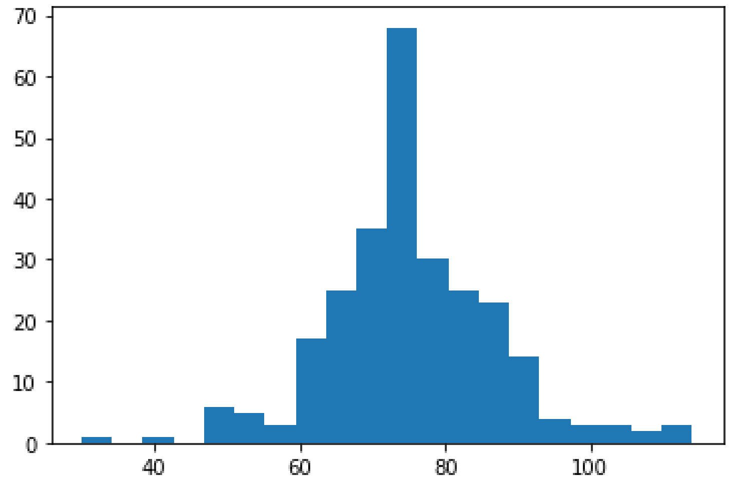


In [28]:
```
Positive['Glucose'].value_counts().head(7)
```

Out[28]:
```
125.0    7
128.0    6
158.0    6
129.0    6
115.0    6
146.0    5
181.0    5
Name: Glucose, dtype: int64
```

In [29]:
```
plt.hist(Positive['BloodPressure'],histtype='stepfilled',bins=20)
```

Out[29]:
```
(array([ 1.,  0.,  1.,  0.,  6.,  5.,  3., 17., 25., 35., 68., 30., 25.,
        23., 14.,  4.,  3.,  3.,  2.,  3.]),
 array([ 30. ,  34.2,  38.4,  42.6,  46.8,  51. ,  55.2,  59.4,  63.6,
         67.8,  72. ,  76.2,  80.4,  84.6,  88.8,  93. ,  97.2, 101.4,
        105.6, 109.8, 114. ]),
 [<matplotlib.patches.Polygon at 0x1cbf6cd0>])
```

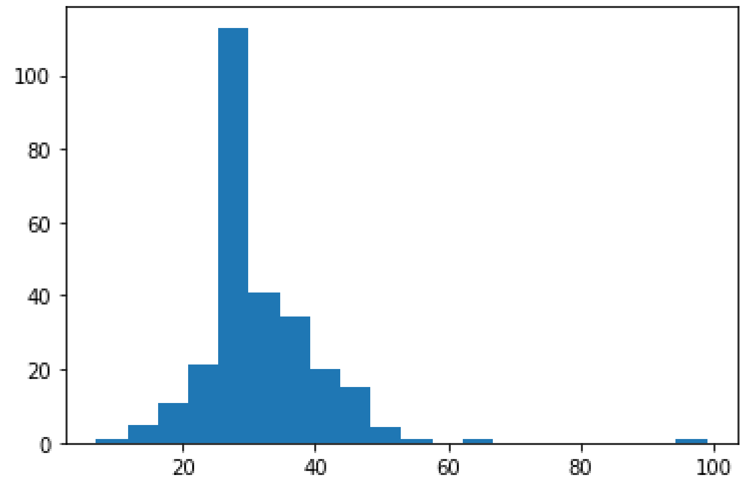


In [30]:
```
Positive['BloodPressure'].value_counts().head(7)
```

Out[30]:
```
72.0    32
70.0    23
76.0    18
78.0    17
74.0    17
82.0    13
64.0    13
Name: BloodPressure, dtype: int64
```

In [31]:
```
plt.hist(Positive['SkinThickness'],histtype='stepfilled',bins=20)
```

Out[31]:
```
(array([  1.,   5.,  11.,  21., 113.,  41.,  34.,  20.,  15.,   4.,   1.,
          0.,   1.,   0.,   0.,   0.,   0.,   0.,   0.,   1.]),
 array([ 7. , 11.6, 16.2, 20.8, 25.4, 30. , 34.6, 39.2, 43.8, 48.4, 53. ,
        57.6, 62.2, 66.8, 71.4, 76. , 80.6, 85.2, 89.8, 94.4, 99. ]),
 [<matplotlib.patches.Polygon at 0x1cc314a8>])
```

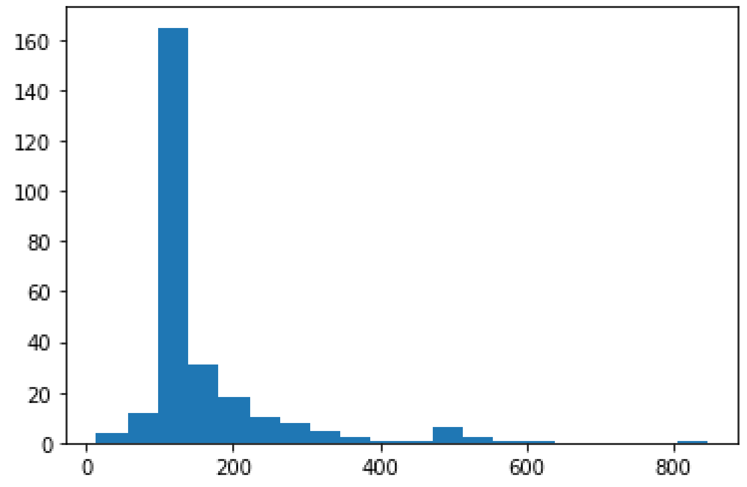


In [32]:
```
Positive['SkinThickness'].value_counts().head(7)
```

Out[32]:
```
29.0    95
32.0    14
30.0     9
33.0     9
36.0     8
37.0     8
39.0     8
Name: SkinThickness, dtype: int64
```

In [33]:
```
plt.hist(Positive['Insulin'],histtype='stepfilled',bins=20)
```

Out[33]:
```
(array([  4.,  12., 165.,  31.,  18.,  10.,   8.,   5.,   2.,   1.,   1.,
          6.,   2.,   1.,   1.,   0.,   0.,   0.,   0.,   1.]),
 array([ 14. ,  55.6,  97.2, 138.8, 180.4, 222. , 263.6, 305.2, 346.8,
        388.4, 430. , 471.6, 513.2, 554.8, 596.4, 638. , 679.6, 721.2,
        762.8, 804.4, 846. ]),
 [<matplotlib.patches.Polygon at 0x1cc65ac0>])
```



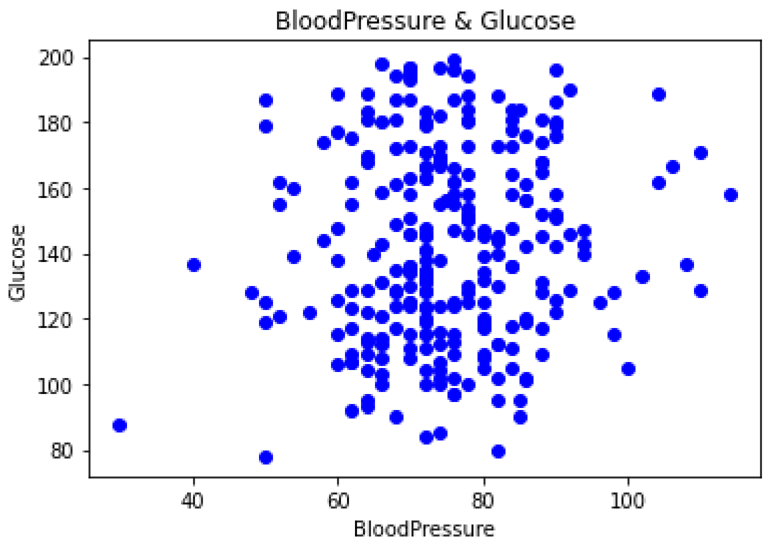
In [34]:
```
Positive['Insulin'].value_counts().head(7)
```

Out[34]:
```
125.0    140
130.0      6
180.0      4
175.0      3
156.0      3
185.0      2
225.0      2
Name: Insulin, dtype: int64
```

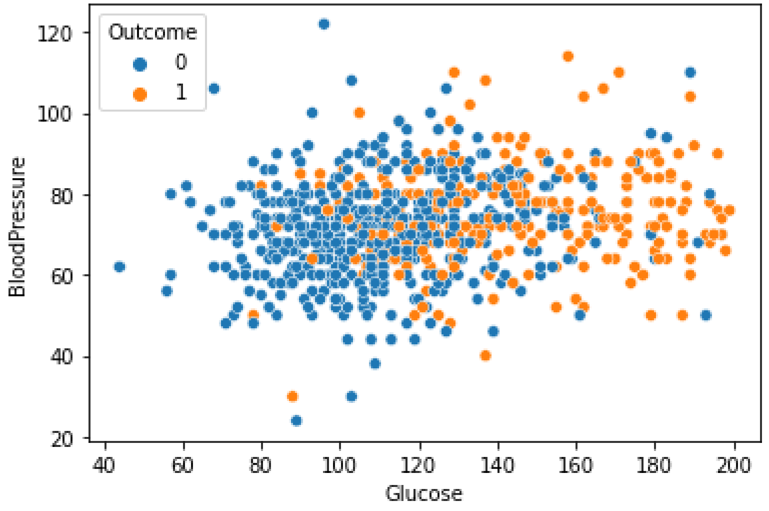
In [35]:
```
#Scatter plot
```

In [36]:
```
BloodPressure = Positive['BloodPressure']
Glucose = Positive['Glucose']
SkinThickness = Positive['SkinThickness']
Insulin = Positive['Insulin']
BMI = Positive['BMI']
```

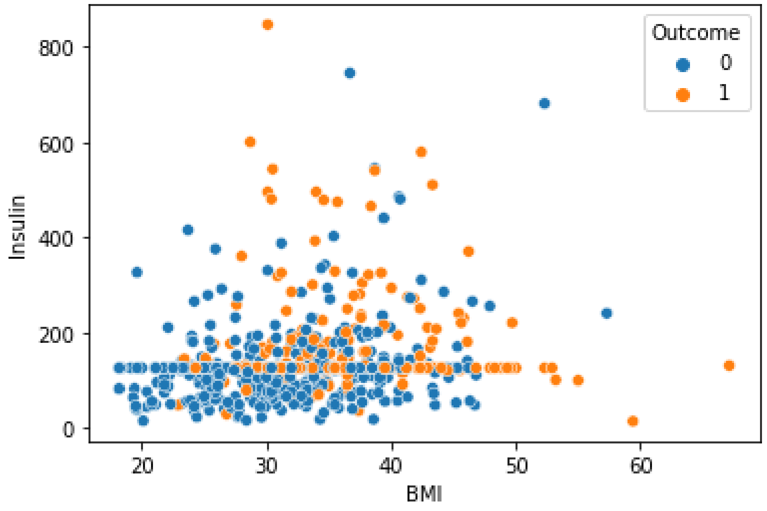
In [37]:
```
plt.scatter(BloodPressure, Glucose, color=['b'])
plt.xlabel('BloodPressure')
plt.ylabel('Glucose')
plt.title('BloodPressure & Glucose')
plt.show()
```



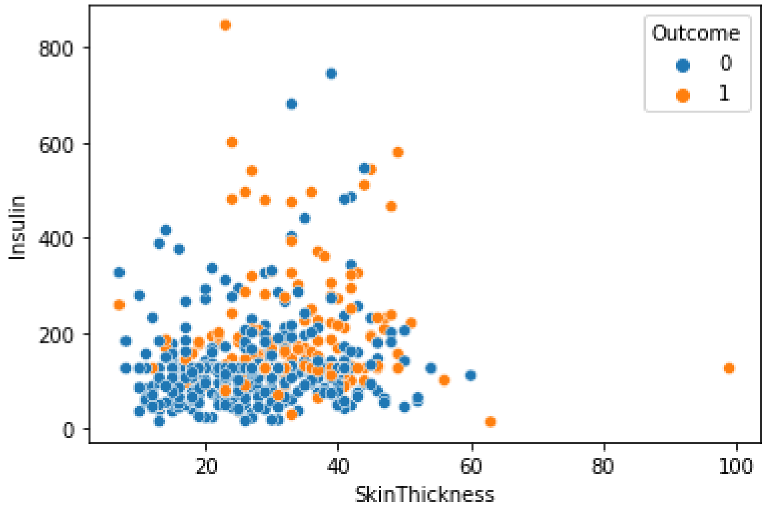
In [38]:
```
g =sns.scatterplot(x= "Glucose" ,y= "BloodPressure",
              hue="Outcome",
              data=data);
```



In [39]:
```
B =sns.scatterplot(x= "BMI" ,y= "Insulin",
              hue="Outcome",
              data=data);
```



In [40]:
```
S =sns.scatterplot(x= "SkinThickness" ,y= "Insulin",
              hue="Outcome",
              data=data);
```



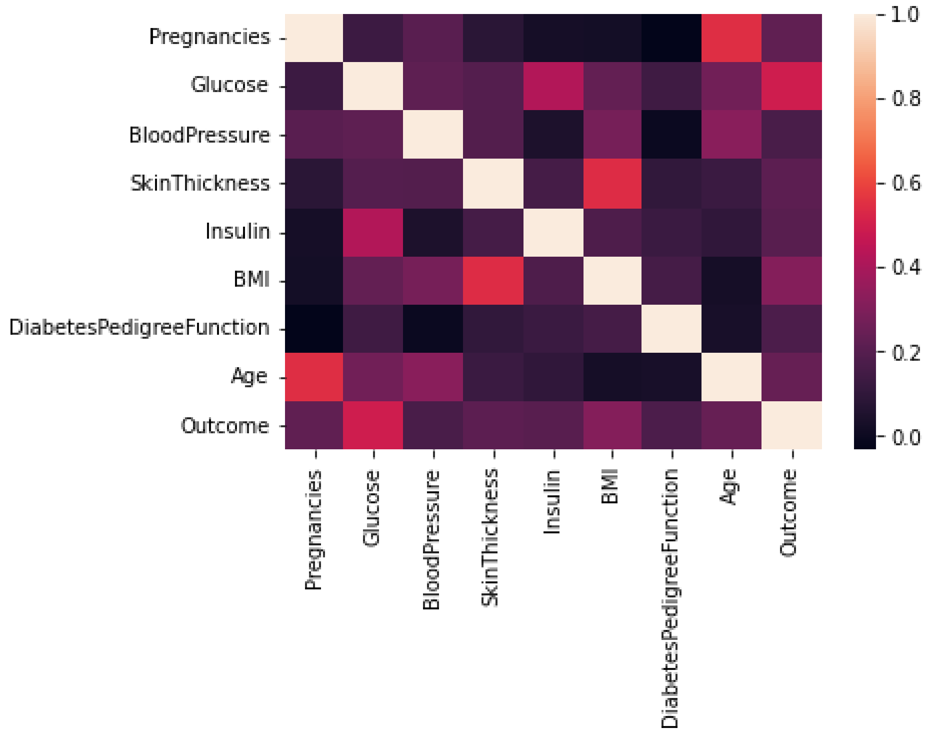
In [41]:
```
### correlation matrix
data.corr()
```

Out[41]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.128213	0.208615	0.081770	0.025047	0.021559	-0.033523	0.544341	0.221898
Glucose	0.128213	1.000000	0.218937	0.192615	0.419451	0.231049	0.137327	0.266909	0.492782
BloodPressure	0.208615	0.218937	1.000000	0.191892	0.045363	0.281257	-0.002378	0.324915	0.165723
SkinThickness	0.081770	0.192615	0.191892	1.000000	0.155610	0.543205	0.102188	0.126107	0.214873
Insulin	0.025047	0.419451	0.045363	0.155610	1.000000	0.180241	0.126503	0.097101	0.203790
BMI	0.021559	0.231049	0.281257	0.543205	0.180241	1.000000	0.153438	0.025597	0.312038
DiabetesPedigreeFunction	-0.033523	0.137327	-0.002378	0.102188	0.126503	0.153438	1.000000	0.033561	0.173844
Age	0.544341	0.266909	0.324915	0.126107	0.097101	0.025597	0.033561	1.000000	0.238356
Outcome	0.221898	0.492782	0.165723	0.214873	0.203790	0.312038	0.173844	0.238356	1.000000

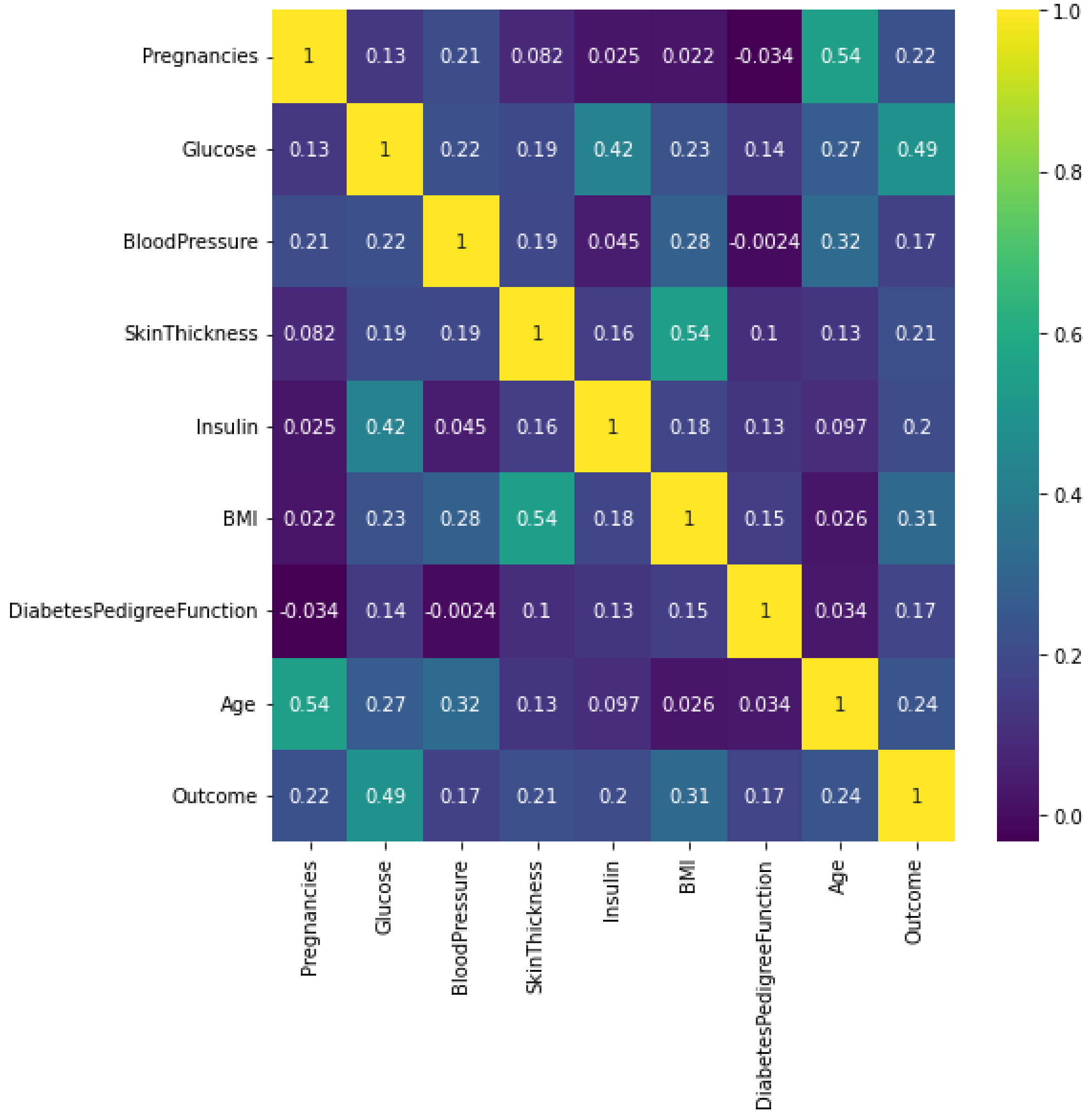
In [42]:
```
### create correlation heat map
sns.heatmap(data.corr())
```

Out[42]:
```
<AxesSubplot:>
```



```
In [43]: plt.subplots(figsize=(8,8))
         sns.heatmap(data.corr(),annot=True,cmap='viridis')  ### gives correlation value
```

```
Out[43]: <AxesSubplot:>
```



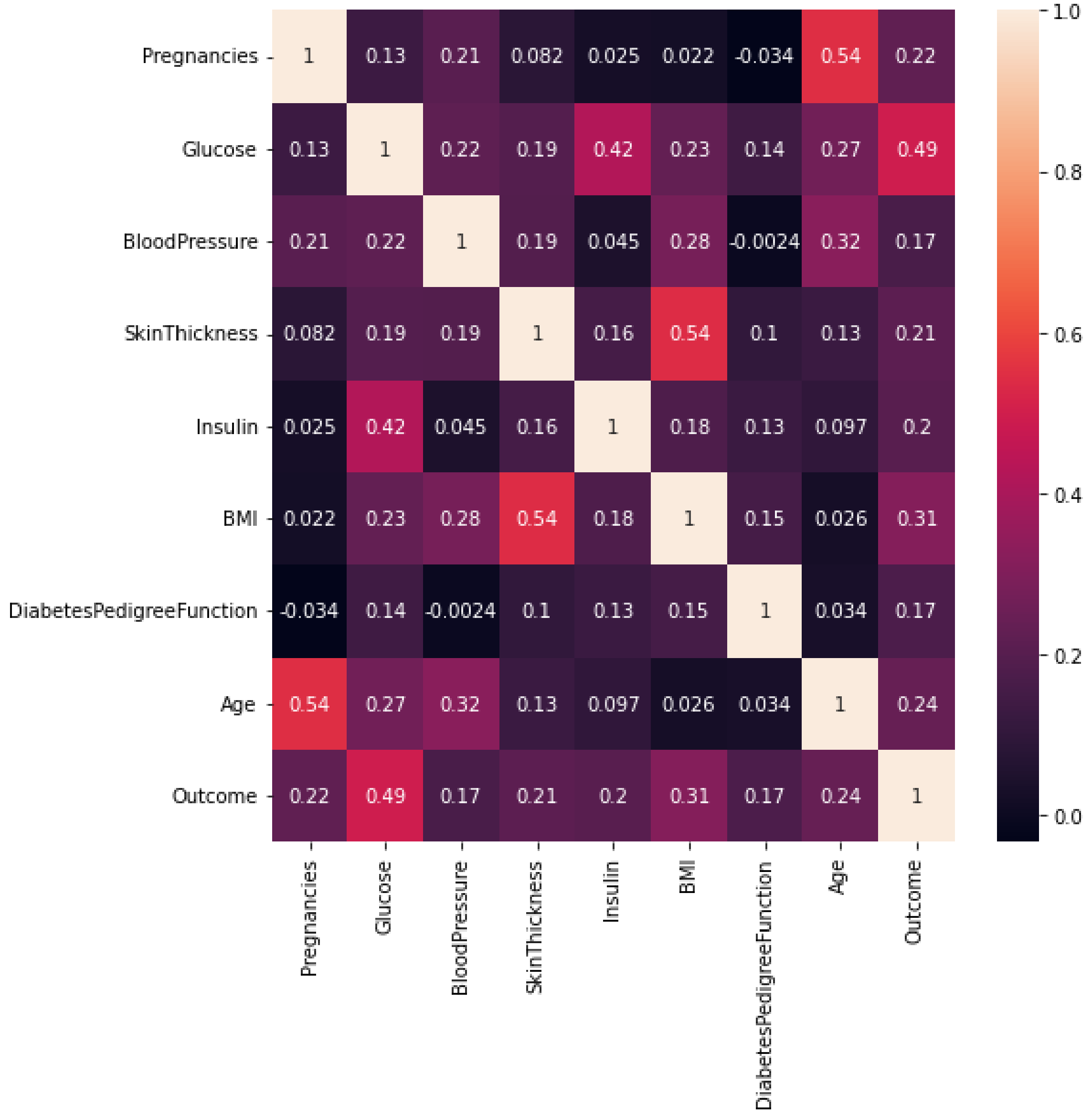
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1	0.13	0.21	0.082	0.025	0.022	-0.034	0.54	0.22
Glucose	0.13	1	0.22	0.19	0.42	0.23	0.14	0.27	0.49
BloodPressure	0.21	0.22	1	0.19	0.045	0.28	-0.0024	0.32	0.17
SkinThickness	0.082	0.19	0.19	1	0.16	0.54	0.1	0.13	0.21
Insulin	0.025	0.42	0.045	0.16	1	0.18	0.13	0.097	0.2
BMI	0.022	0.23	0.28	0.54	0.18	1	0.15	0.026	0.31
DiabetesPedigreeFunction	-0.034	0.14	-0.0024	0.1	0.13	0.15	1	0.034	0.17
Age	0.54	0.27	0.32	0.13	0.097	0.026	0.034	1	0.24
Outcome	0.22	0.49	0.17	0.21	0.2	0.31	0.17	0.24	1

In [44]:

```
plt.subplots(figsize=(8,8))
sns.heatmap(data.corr(),annot=True)  ### gives correlation value
```

Out[44]:

```
<AxesSubplot:>
```



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1	0.13	0.21	0.082	0.025	0.022	-0.034	0.54	0.22
Glucose	0.13	1	0.22	0.19	0.42	0.23	0.14	0.27	0.49
BloodPressure	0.21	0.22	1	0.19	0.045	0.28	-0.0024	0.32	0.17
SkinThickness	0.082	0.19	0.19	1	0.16	0.54	0.1	0.13	0.21
Insulin	0.025	0.42	0.045	0.16	1	0.18	0.13	0.097	0.2
BMI	0.022	0.23	0.28	0.54	0.18	1	0.15	0.026	0.31
DiabetesPedigreeFunction	-0.034	0.14	-0.0024	0.1	0.13	0.15	1	0.034	0.17
Age	0.54	0.27	0.32	0.13	0.097	0.026	0.034	1	0.24
Outcome	0.22	0.49	0.17	0.21	0.2	0.31	0.17	0.24	1

Week 3

In [45]:
```
# Logistic Regreation and model building
```

In [46]:
```
data.head(5)
```

Out[46]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	125.0	33.6	0.627	50	1
1	1	85.0	66.0	29.0	125.0	26.6	0.351	31	0
2	8	183.0	64.0	29.0	125.0	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 [47]:
```
features = data.iloc[:,[0,1,2,3,4,5,6,7]].values
label = data.iloc[:,8].values
```

In [48]:
```
pip install -U scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\prabh\appdata\local\programs\python\python38-32\lib\site-packages (1.0.2)
Requirement already satisfied: numpy>=1.14.6 in c:\users\prabh\appdata\local\programs\python\python38-32\lib\site-packages (from s
cikit-learn) (1.21.4)
Requirement already satisfied: joblib>=0.11 in c:\users\prabh\appdata\local\programs\python\python38-32\lib\site-packages (from sc
ikit-learn) (1.1.0)
Requirement already satisfied: scipy>=1.1.0 in c:\users\prabh\appdata\local\programs\python\python38-32\lib\site-packages (from sc
ikit-learn) (1.7.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\prabh\appdata\local\programs\python\python38-32\lib\site-packages
(from scikit-learn) (3.1.0)
Note: you may need to restart the kernel to use updated packages.
WARNING: You are using pip version 21.3.1; however, version 22.0.3 is available.
You should consider upgrading via the 'c:\users\prabh\appdata\local\programs\python\python38-32\python.exe -m pip install --upgrad
e pip' command.
```

In [49]:
```
#Train test split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(features,
                                                 label,
                                                 test_size=0.2,
                                                 random_state =10)
```

In [50]:
```
#Create model
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,y_train)
```

```
c:\users\prabh\appdata\local\programs\python\python38-32\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarni
ng: 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[50]:
```
LogisticRegression()
```

In [51]:
```
print(model.score(X_train,y_train))
print(model.score(X_test,y_test))
```

```
0.7833876221498371
0.7272727272727273
```

In [52]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(label,model.predict(features))
cm
```

Out[52]:
```
array([[445,  55],
       [120, 148]], dtype=int64)
```

In [53]:
```
from sklearn.metrics import classification_report
print(classification_report(label,model.predict(features)))
```

```
              precision    recall  f1-score   support

           0       0.79      0.89      0.84       500
           1       0.73      0.55      0.63       268

    accuracy                           0.77       768
   macro avg       0.76      0.72      0.73       768
```

```
weighted avg       0.77      0.77      0.76       768
```

Week 4

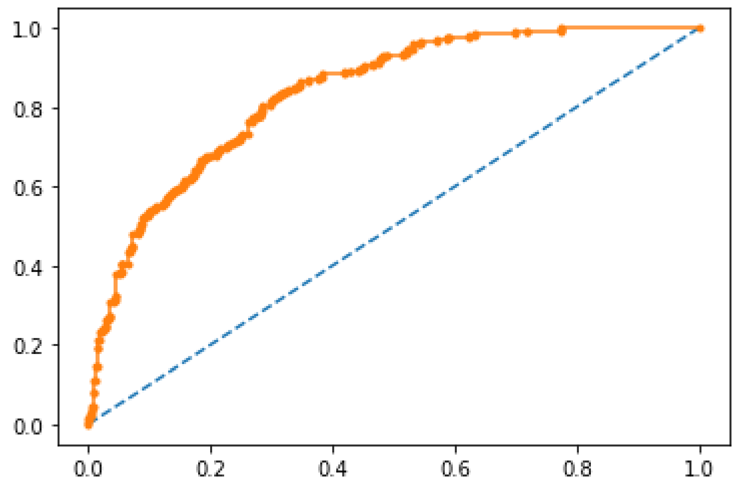
In [54]:

```
#Preparing ROC Curve (Receiver Operating Characteristics Curve)
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

# predict probabilities
probs = model.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(label, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
fpr, tpr, thresholds = roc_curve(label, probs)
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
```

```
AUC: 0.838
```

Out[54]:

```
[<matplotlib.lines.Line2D at 0x1ee90dd8>]
```



In [55]:
```
#Applying Decission Tree Classifier
from sklearn.tree import DecisionTreeClassifier
model3 = DecisionTreeClassifier(max_depth=5)
model3.fit(X_train,y_train)
```

Out[55]:
```
DecisionTreeClassifier(max_depth=5)
```

In [56]:
```
model3.score(X_train,y_train)
```

Out[56]:
```
0.8192182410423453
```

In [57]:
```
model3.score(X_test,y_test)
```

Out[57]:
```
0.7467532467532467
```

In [58]:
```
#Applying Random Forest
from sklearn.ensemble import RandomForestClassifier
model4 = RandomForestClassifier(n_estimators=11)
model4.fit(X_train,y_train)
```

Out[58]:
```
RandomForestClassifier(n_estimators=11)
```

In [59]:
```
model4.score(X_train,y_train)
```

Out[59]:
```
0.99185667752443
```

In [60]:
```
model4.score(X_test,y_test)
```

Out[60]:
```
0.7272727272727273
```

In [61]:
```
#Support Vector Classifier

from sklearn.svm import SVC
model5 = SVC(kernel='rbf',
                 gamma='auto')
model5.fit(X_train,y_train)
```

Out[61]:
```
SVC(gamma='auto')
```

In [62]:
```
# model.score(X_test,y_test).score(X_train,y_train)
```

In [63]:
```
model5.score(X_test,y_test)
```

Out[63]:
```
0.6168831168831169
```

In [64]:
```
#Applying K-NN
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier(n_neighbors=7,
                             metric='minkowski',
                             p = 2)
model2.fit(X_train,y_train)
```

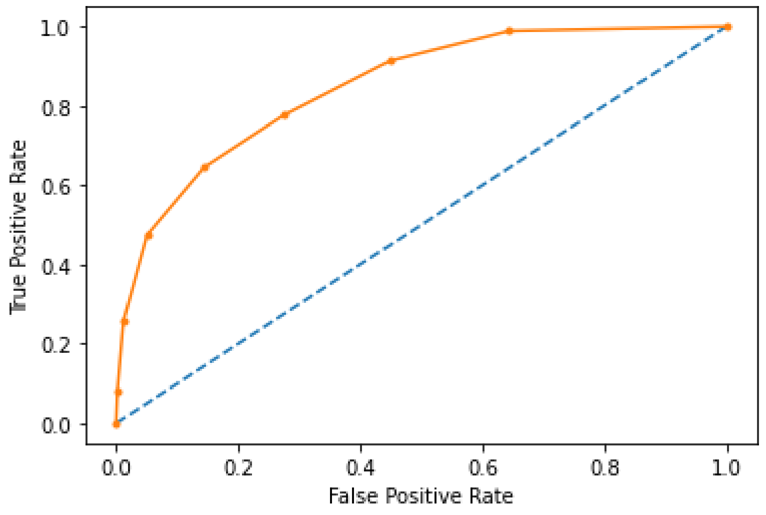
Out[64]:
```
KNeighborsClassifier(n_neighbors=7)
```

In [65]:
```
#Preparing ROC Curve (Receiver Operating Characteristics Curve)
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

# predict probabilities
probs = model2.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(label, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
fpr, tpr, thresholds = roc_curve(label, probs)
print("True Positive Rate - {}, False Positive Rate - {} Thresholds - {}".format(tpr,fpr,thresholds))
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

```
AUC: 0.848
True Positive Rate - [0.         0.07835821 0.25746269 0.4738806  0.64552239 0.7761194
 0.9141791  0.98880597 1.        ], False Positive Rate - [0.    0.002 0.012 0.05  0.144 0.274 0.45  0.644 1.   ] Thresholds - [2.
1.         0.85714286 0.71428571 0.57142857 0.42857143
 0.28571429 0.14285714 0.        ]
```

Out[65]:

```
Text(0, 0.5, 'True Positive Rate')
```



In [66]:

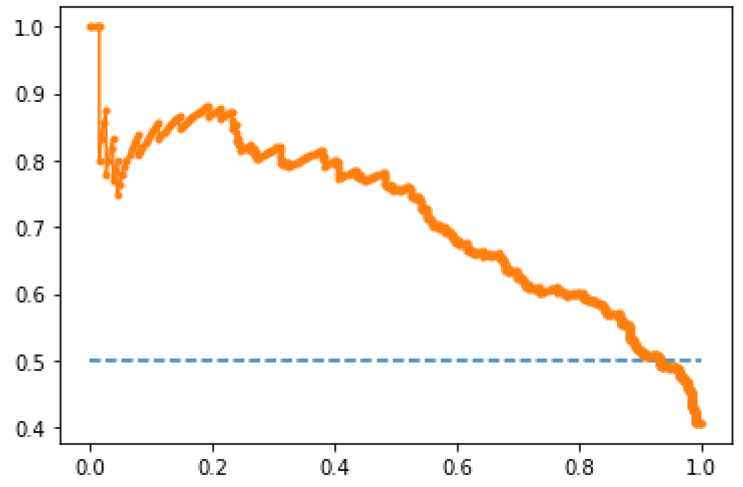
```
#Precision Recall Curve for Logistic Regression

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model.predict(features)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
# calculate F1 score
f1 = f1_score(label, yhat)
```

```
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.628 auc=0.715 ap=0.716
```

Out[66]:

```
[<matplotlib.lines.Line2D at 0x1f4efcd0>]
```



In [67]:

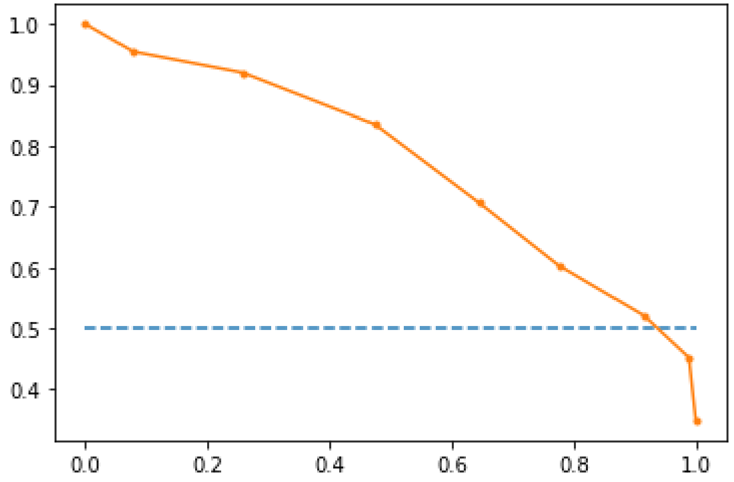
```
#Precision Recall Curve for KNN

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model2.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model2.predict(features)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
```

```
# calculate F1 score
f1 = f1_score(label, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.674 auc=0.771 ap=0.730
```

Out[67]:

```
[<matplotlib.lines.Line2D at 0x1f5254c0>]
```



In [68]:

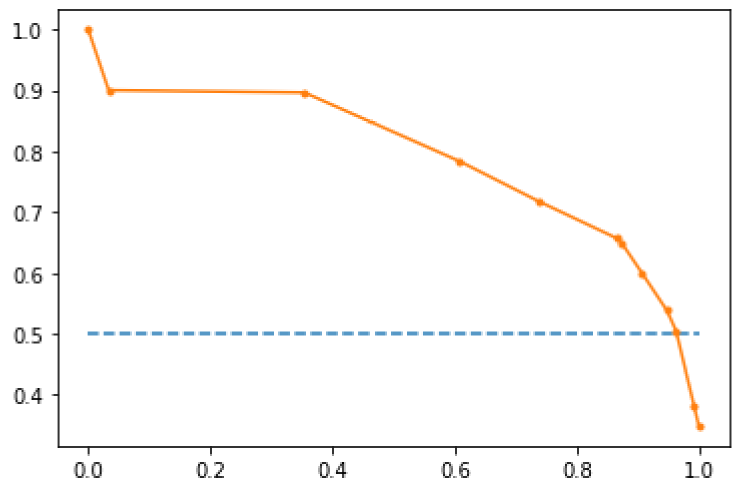
```
#Precision Recall Curve for Decission Tree Classifier

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model3.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model3.predict(features)
```

```
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
# calculate F1 score
f1 = f1_score(label, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.685 auc=0.791 ap=0.762
```

Out[68]:

```
[<matplotlib.lines.Line2D at 0x1f5581d8>]
```

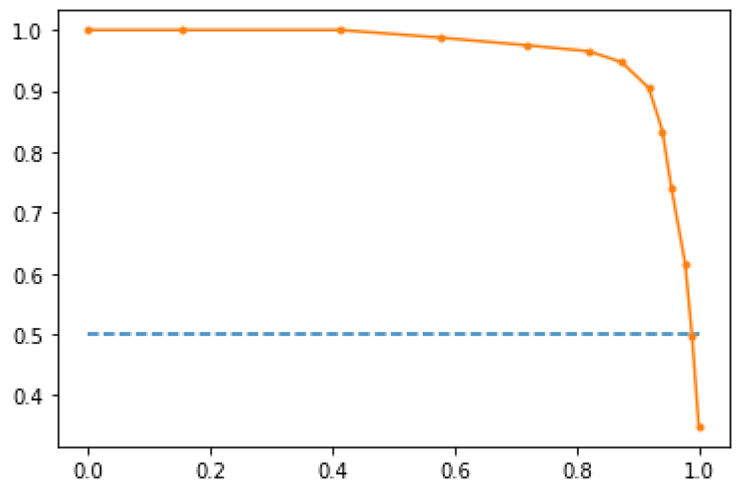


In [69]:

```
#Precision Recall Curve for Random Forest

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model4.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
```

```
# predict class values
yhat = model4.predict(features)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
# calculate F1 score
f1 = f1_score(label, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.909 auc=0.963 ap=0.955
Out[69]: [<matplotlib.lines.Line2D at 0x1f586e68>]
```


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.2
0.4
0.6
0.8
1.0