Project 2 Healthcare

October 4, 2021

```
Project 2 :Healthcare        Name: Niyojita Arun Raje    Cohort: 1 (DEC 2020)
                         WEEK 1: Data Exploration
```

[1]:
```python
import numpy as np,pandas as pd,matplotlib.pyplot as plt,seaborn as snss
```

[2]:
```python
data=pd.read_csv('G:/Simplilearn/Capstone Project/Project 2/Project 2/
↪Healthcare - Diabetes/health care diabetes.csv')
```

[3]:
```python
data.head()
```

[3]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

[4]:
```python
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
```

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

[5]:
```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

[6]:
```
data.isnull().sum()
```

[6]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[7]:
```
data['Glucose'].values==0
```

[7]:
```
array([False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
```

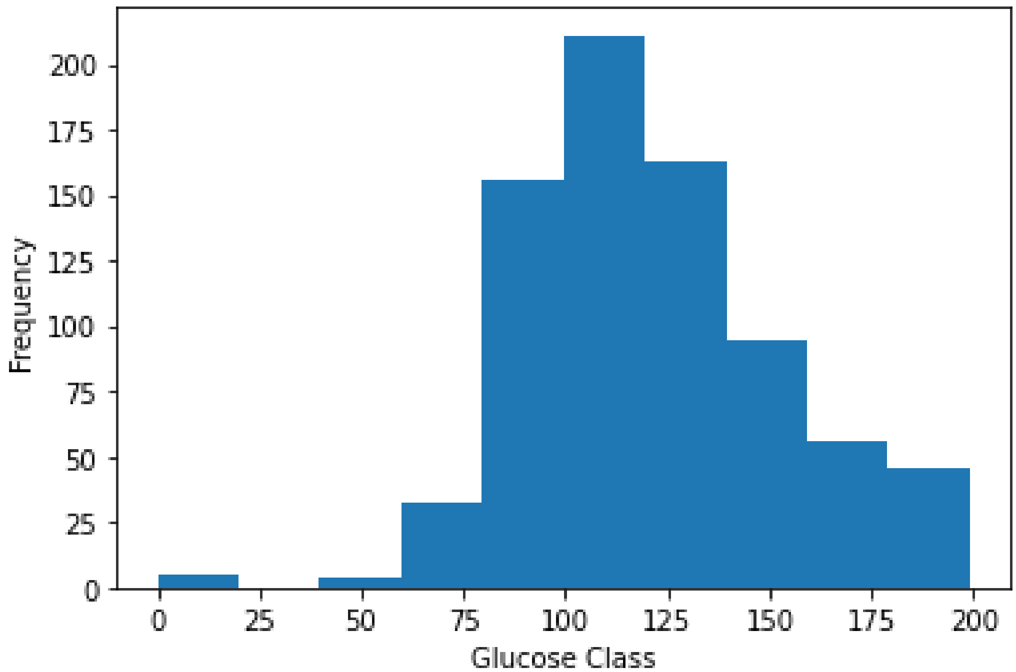
```
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False,  True, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False,  True, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
 True, False, False, False, False, False, False,  True, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
```

```
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False,  True, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False])
```

[8]:

```
plt.xlabel('Glucose Class')
data['Glucose'].plot.hist()
print("Datatype of Glucose is:",data['Glucose'].dtypes)
```

```
Datatype of Glucose is: int64
```



We can see that there are 0 value data and Glucose cannot be 0.Hence replacing 0 with mean of Glucose class

[9]:
```
data['Glucose']=data['Glucose'].replace(0,data['Glucose'].mean())
```

[10]:
```
data['BloodPressure'].values==0
```

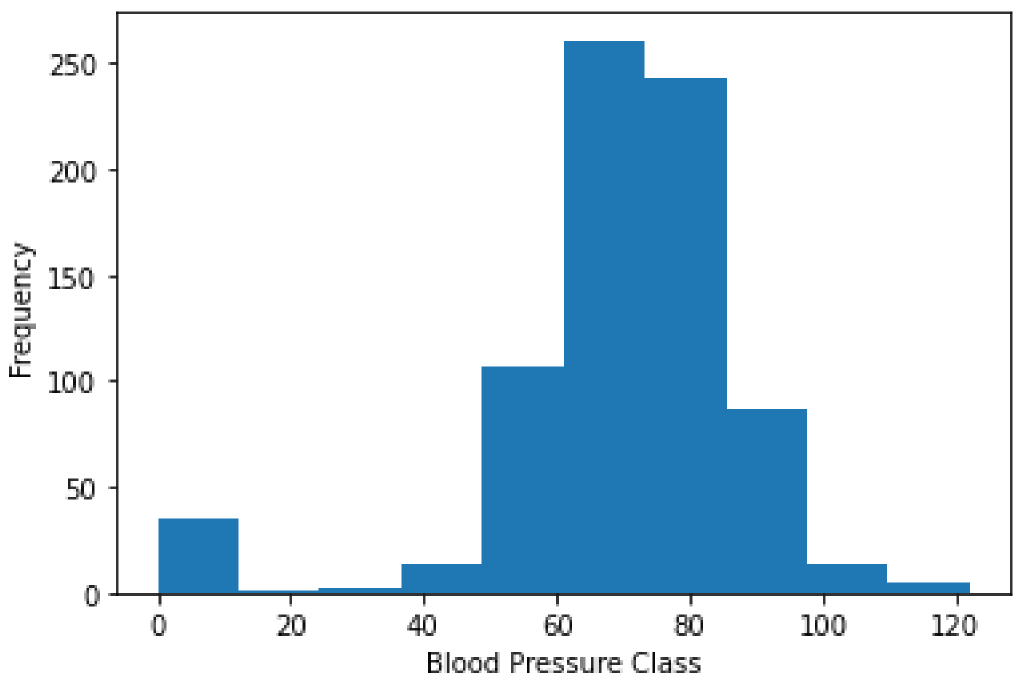
[10]:
```
array([False, False, False, False, False, False, False,  True, False,
       False, False, False, False, False, False,  True, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False,  True, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False,  True, False, False,
        True, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
```

```
False, False, False, False, False, False, False, False, False,
False,  True, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False,  True, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False,  True, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
 True, False, False, False, False,  True, False, False,  True,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False,  True, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False,  True,
False, False, False,  True, False, False, False, False, False,
False, False, False, False, False,  True, False, False, False,
False, False, False, False, False, False,  True, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False,  True, False, False, False,  True, False,
False, False, False,  True, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False,  True, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
 True, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False,  True, False,
False, False, False, False, False, False, False, False,  True,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
 True, False, False, False, False, False, False, False, False,
False, False,  True, False,  True, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
```

```
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False, False,  True, False,
       False,  True, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False,  True, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False,  True, False, False,  True, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False])
```

[11]:
```python
plt.xlabel('Blood Pressure Class')
data['BloodPressure'].plot.hist()
print("Datatype of BloodPressure is:",data['BloodPressure'].dtypes)
```

```
Datatype of BloodPressure is: int64
```



We can see that there are 0 value data and BloodPressure cannot be 0.Hence replacing 0 with mean of BloodPressure class

[12]:
```
data['BloodPressure']=data['BloodPressure'].replace(0,data['BloodPressure'].
↪mean())
```

[13]:
```
data['SkinThickness'].values==0
```

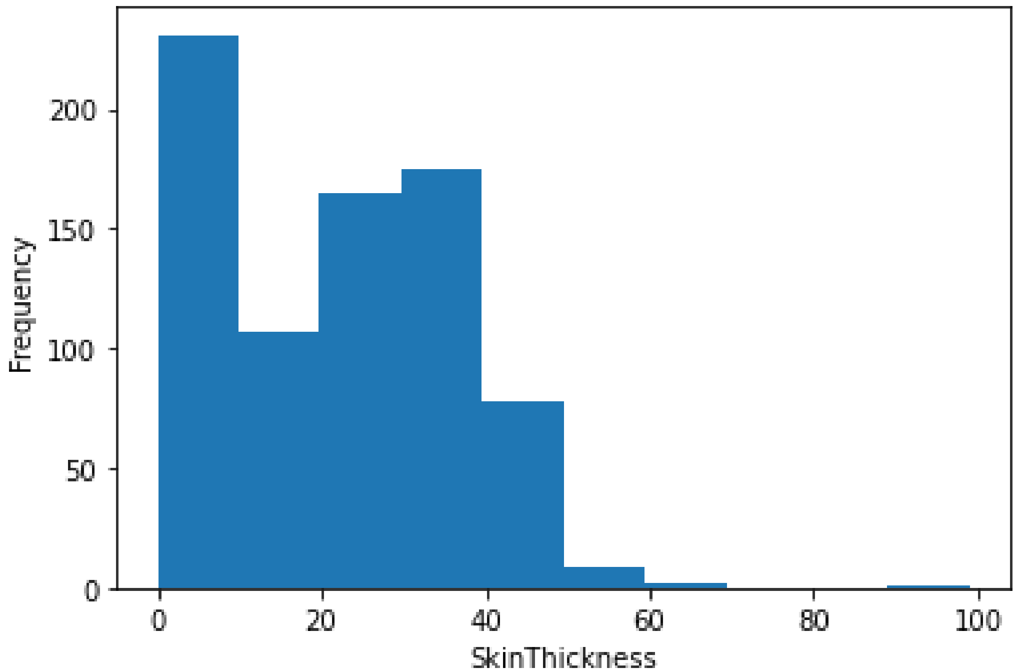
[13]:
```
array([False, False,  True, False, False,  True, False,  True, False,
        True,  True,  True,  True, False, False,  True, False,  True,
       False, False, False,  True,  True, False, False, False,  True,
       False, False,  True, False, False, False,  True, False, False,
        True, False, False, False, False,  True, False, False,  True,
       False,  True, False, False,  True, False, False, False, False,
       False, False, False, False,  True, False,  True,  True,  True,
       False,  True, False, False,  True, False, False, False, False,
        True, False, False, False,  True, False,  True, False, False,
        True, False, False,  True, False, False, False, False, False,
        True, False, False,  True, False, False, False, False, False,
       False,  True,  True,  True, False,  True, False,  True, False,
       False, False, False, False, False,  True, False,  True,  True,
        True, False, False, False, False, False,  True,  True, False,
       False, False, False,  True, False,  True, False, False, False,
       False, False, False,  True, False,  True, False, False,  True,
       False, False, False, False,  True, False, False,  True, False,
       False,  True, False, False, False, False, False, False, False,
       False, False,  True, False, False,  True,  True, False,  True,
       False, False, False, False, False,  True, False,  True,  True,
        True, False, False,  True,  True, False, False, False, False,
       False,  True, False,  True,  True, False, False,  True, False,
       False, False, False,  True, False, False, False, False, False,
        True, False, False, False, False, False, False, False, False,
       False, False, False,  True, False,  True,  True, False, False,
       False,  True, False, False, False,  True, False, False,  True,
       False,  True, False, False, False,  True, False, False,  True,
       False, False, False,  True, False, False, False,  True,  True,
       False, False, False, False, False, False, False, False, False,
        True, False, False,  True, False,  True, False,  True,  True,
       False, False,  True, False,  True, False, False, False,  True,
       False,  True, False, False,  True,  True, False, False, False,
       False, False, False, False, False, False,  True, False, False,
       False, False,  True,  True, False, False,  True,  True, False,
       False, False, False, False, False, False, False, False, False,
       False, False,  True, False,  True, False, False, False, False,
       False, False, False,  True, False, False, False, False,  True,
```

```
 True, False, False,  True,  True, False,  True, False, False,
False,  True,  True, False, False,  True, False, False,  True,
 True, False, False,  True,  True, False, False, False, False,
False,  True, False,  True, False, False,  True, False, False,
False, False, False, False, False, False, False, False, False,
 True, False, False, False, False, False, False, False, False,
False, False, False, False,  True, False, False,  True, False,
False, False,  True, False,  True,  True, False, False,  True,
False,  True,  True,  True, False, False, False, False, False,
False, False, False, False,  True, False, False, False, False,
False, False, False,  True, False, False, False,  True, False,
False,  True, False,  True, False,  True, False,  True, False,
False, False,  True, False, False, False, False, False, False,
False,  True, False,  True, False, False,  True, False, False,
False, False,  True, False, False,  True, False, False, False,
 True, False, False, False, False,  True,  True, False, False,
False, False, False, False, False, False, False,  True, False,
False, False, False,  True, False, False, False, False,  True,
 True,  True, False, False, False, False, False, False, False,
False,  True, False, False, False,  True, False, False,  True,
 True, False, False, False,  True,  True, False, False, False,
 True,  True,  True, False, False, False, False,  True, False,
 True, False,  True, False,  True,  True,  True, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False,  True, False, False, False, False,  True,
False,  True,  True, False, False, False,  True, False, False,
False, False, False,  True,  True, False, False, False, False,
False,  True,  True, False, False, False, False,  True, False,
False,  True,  True, False,  True, False, False,  True, False,
False, False,  True, False,  True, False, False,  True, False,
False,  True, False, False, False, False, False, False, False,
False, False, False,  True,  True, False, False,  True, False,
False,  True, False,  True, False,  True,  True,  True, False,
 True, False,  True, False,  True,  True,  True, False, False,
False, False,  True,  True,  True, False, False, False, False,
False, False, False, False, False,  True, False, False, False,
False,  True, False,  True, False, False, False, False, False,
False, False, False, False, False, False, False, False,  True,
 True,  True,  True,  True, False, False, False, False,  True,
 True, False,  True, False, False, False,  True,  True, False,
False,  True, False, False,  True, False,  True, False, False,
False,  True, False, False,  True, False,  True, False, False,
False, False, False,  True, False, False, False, False, False,
False, False, False, False,  True, False, False, False,  True,
 True, False,  True, False, False,  True, False, False, False,
False,  True, False, False, False,  True, False, False, False,
False, False,  True,  True, False, False, False, False, False,
```

```
        False,  True,  True,  True, False, False,  True, False, False,
        False,  True, False])
```

[14]:
```
plt.xlabel('SkinThickness')
data['SkinThickness'].plot.hist()
print("Datatype of SkinThickness is:",data['SkinThickness'].dtypes)
```

```
Datatype of SkinThickness is: int64
```



We can see that there are 0 value data and SkinThickness cannot be 0.Hence replacing 0 with mean of SkinThickness class

[15]:
```
data['SkinThickness']=data['SkinThickness'].replace(0,data['SkinThickness'].
 ↪mean())
```

[16]:
```
data['Insulin'].values==0
```

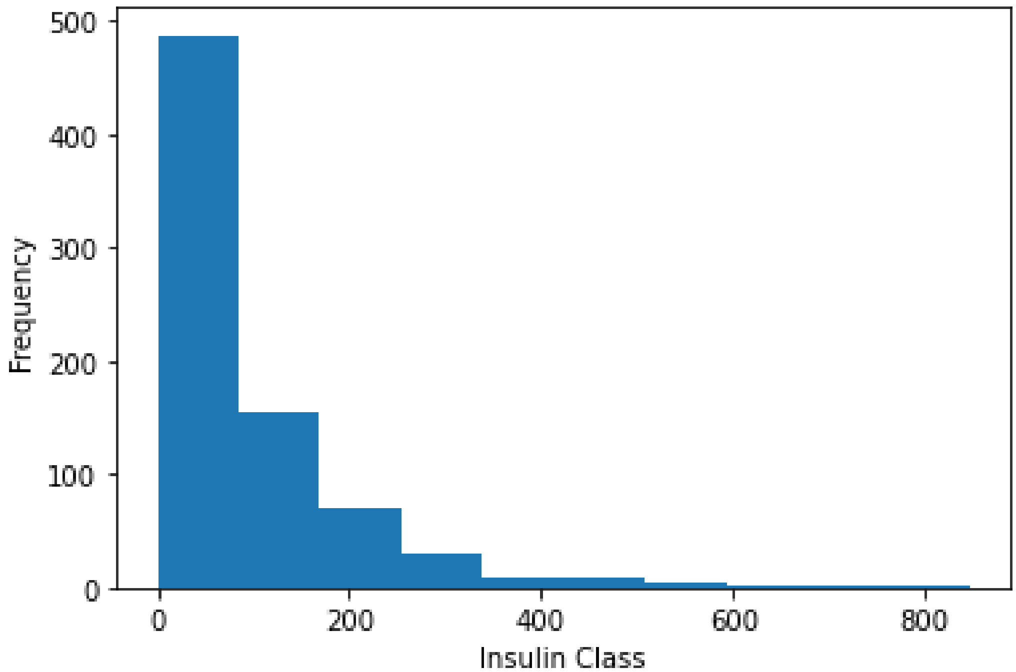
[16]:
```
array([ True,  True,  True, False, False,  True, False,  True, False,
        True,  True,  True,  True, False, False,  True, False,  True,
       False, False, False,  True,  True,  True, False, False,  True,
       False, False,  True,  True, False, False,  True,  True, False,
        True,  True,  True, False, False,  True,  True, False,  True,
        True,  True,  True,  True,  True, False, False, False, False,
       False,  True, False, False,  True, False,  True,  True,  True,
       False,  True,  True,  True,  True, False, False, False, False,
```

```
 True, False,  True,  True,  True,  True,  True,  True,  True,
 True, False,  True,  True, False,  True, False, False,  True,
 True, False, False,  True, False, False,  True, False, False,
False,  True,  True,  True, False,  True, False,  True, False,
False, False, False, False, False,  True, False,  True,  True,
 True,  True, False, False,  True, False,  True,  True, False,
False, False, False,  True, False,  True, False,  True, False,
False, False, False,  True, False,  True,  True, False,  True,
False,  True,  True, False,  True,  True, False,  True, False,
False,  True,  True, False, False, False, False,  True, False,
False,  True,  True, False,  True,  True,  True, False,  True,
False,  True, False, False, False,  True, False,  True,  True,
 True, False, False,  True,  True,  True, False, False, False,
False,  True, False,  True,  True,  True, False,  True, False,
False, False,  True,  True,  True, False, False,  True, False,
 True, False,  True,  True,  True,  True, False, False, False,
False, False,  True,  True, False,  True,  True, False, False,
False,  True,  True, False, False,  True, False, False,  True,
False,  True, False,  True,  True,  True,  True, False,  True,
False, False,  True,  True, False, False,  True,  True,  True,
False,  True, False,  True,  True,  True, False, False, False,
 True,  True,  True,  True, False,  True,  True,  True,  True,
 True, False,  True, False,  True, False,  True, False,  True,
False,  True, False, False,  True,  True, False, False, False,
False, False, False, False, False, False,  True, False, False,
False, False,  True,  True, False, False,  True,  True, False,
False, False, False, False,  True, False, False, False,  True,
False, False,  True, False,  True, False,  True,  True, False,
 True, False, False,  True, False, False,  True, False,  True,
 True, False, False,  True,  True, False,  True, False, False,
 True,  True,  True, False, False,  True, False,  True,  True,
 True,  True, False,  True,  True, False,  True, False, False,
False,  True,  True,  True, False, False,  True,  True, False,
False, False, False, False, False, False, False, False, False,
 True, False, False,  True, False, False, False, False,  True,
 True, False, False, False,  True, False, False,  True, False,
False,  True,  True,  True,  True,  True, False,  True,  True,
False,  True,  True,  True, False,  True, False, False, False,
False, False,  True,  True,  True, False, False, False, False,
 True, False, False,  True, False, False, False,  True, False,
False,  True,  True,  True,  True,  True,  True,  True,  True,
False, False,  True,  True, False, False, False, False, False,
False,  True, False,  True, False,  True,  True, False, False,
False, False,  True, False,  True,  True, False, False, False,
 True, False,  True,  True,  True,  True,  True,  True, False,
False, False,  True, False,  True, False, False,  True, False,
False, False,  True,  True, False,  True,  True, False,  True,
```

```
        True,  True, False, False, False, False,  True,  True, False,
        True,  True, False, False, False,  True,  True, False,  True,
        True, False, False, False,  True,  True, False, False, False,
        True,  True,  True,  True, False, False, False,  True, False,
        True, False,  True, False,  True,  True,  True, False, False,
       False, False,  True, False, False, False, False, False, False,
        True,  True, False,  True, False, False, False,  True,  True,
        True,  True,  True, False, False, False,  True, False, False,
       False, False, False,  True,  True, False, False, False, False,
       False,  True,  True,  True,  True,  True,  True,  True, False,
        True,  True,  True, False,  True,  True, False,  True, False,
       False, False,  True, False,  True, False,  True,  True,  True,
       False,  True,  True, False, False, False, False, False, False,
       False,  True, False,  True,  True, False,  True,  True, False,
        True,  True, False,  True, False,  True,  True,  True,  True,
        True, False,  True, False,  True,  True,  True, False, False,
       False, False,  True,  True,  True, False, False, False, False,
       False,  True, False, False, False,  True, False, False, False,
       False,  True, False,  True,  True, False, False,  True, False,
        True,  True, False, False, False,  True, False, False,  True,
        True,  True,  True,  True, False, False,  True, False,  True,
        True, False,  True,  True, False, False,  True,  True, False,
       False,  True, False, False,  True, False,  True, False,  True,
        True,  True, False,  True,  True, False,  True, False, False,
       False,  True, False,  True, False, False,  True, False,  True,
        True, False, False, False,  True,  True, False,  True,  True,
        True, False,  True, False, False,  True,  True, False,  True,
       False,  True, False, False, False,  True, False, False,  True,
       False, False,  True,  True, False,  True, False,  True, False,
        True,  True,  True,  True, False,  True,  True, False,  True,
       False,  True,  True])
```

[17]:
```
plt.xlabel('Insulin Class')
data['Insulin'].plot.hist()
print("Datatype of Insulin is:",data['Insulin'].dtypes)
```

```
Datatype of Insulin is: int64
```



We can see that there are 0 value data and Insulin cannot be 0.Hence replacing 0 with mean of Insulin class

[18]:
```
data['Insulin']=data['Insulin'].replace(0,data['Insulin'].mean())
```

[19]:
```
data['BMI'].values==0
```

[19]:
```
array([False, False, False, False, False, False, False, False, False,
         True, False, False, False, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False,
        False, False, False, False,  True, False, False, False, False,
        False, False, False, False, False, False,  True, False, False,
        False, False, False, False, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False,
         True, False, False, False, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False,
        False,  True, False, False, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False,
```

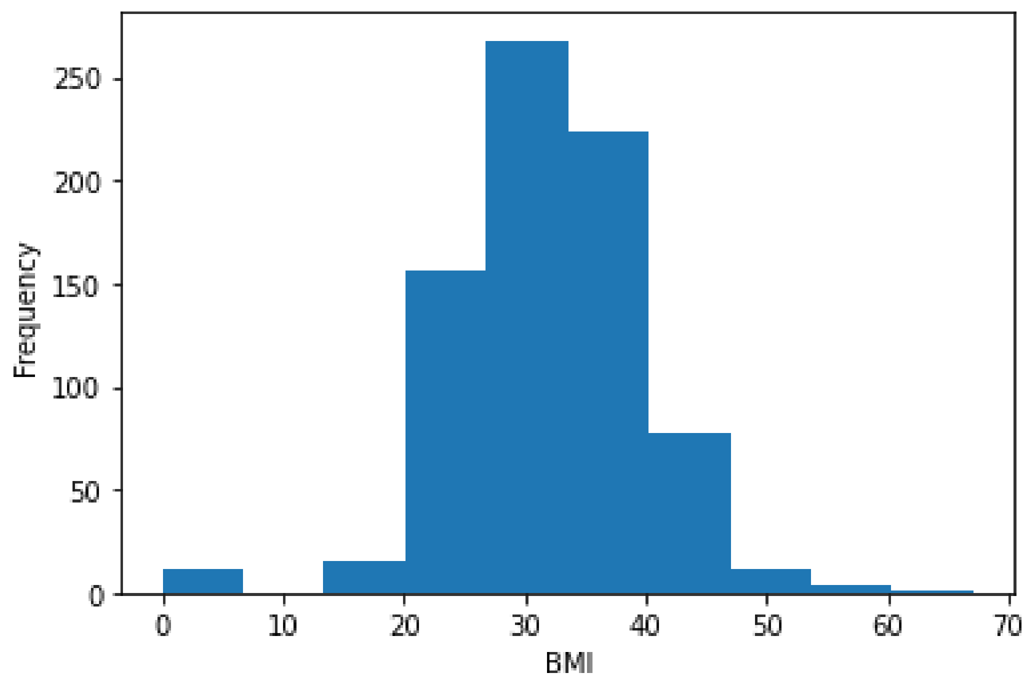
```
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False,  True, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False,  True, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False,  True,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
 True, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False,
```

```
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
        True, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False])
```

[20]:

```
plt.xlabel('BMI')
data['BMI'].plot.hist()
print("Datatype of BMI is:",data['BMI'].dtypes)
```

```
Datatype of BMI is: float64
```



We can see that there are 0 value data and BMI cannot be 0.Hence replacing 0 with mean of BMI class

[21]:
```
data['BMI']=data['BMI'].replace(0,data['BMI'].mean())
```

[22]:
```
data['Glucose'].value_counts().head(5)
```

[22]:
```
100.0    17
99.0     17
125.0    14
106.0    14
111.0    14
Name: Glucose, dtype: int64
```

[23]:
```
data['SkinThickness'].value_counts().head(5)
```

[23]:
```
20.536458    227
32.000000     31
30.000000     27
27.000000     23
23.000000     22
Name: SkinThickness, dtype: int64
```

[24]:
```
data['Insulin'].value_counts().head(5)
```

[24]:
```
79.799479     374
105.000000     11
130.000000      9
140.000000      9
120.000000      8
Name: Insulin, dtype: int64
```

[25]:
```
data['BloodPressure'].value_counts().head(5)
```

[25]:
```
70.0    57
74.0    52
68.0    45
78.0    45
72.0    44
Name: BloodPressure, dtype: int64
```

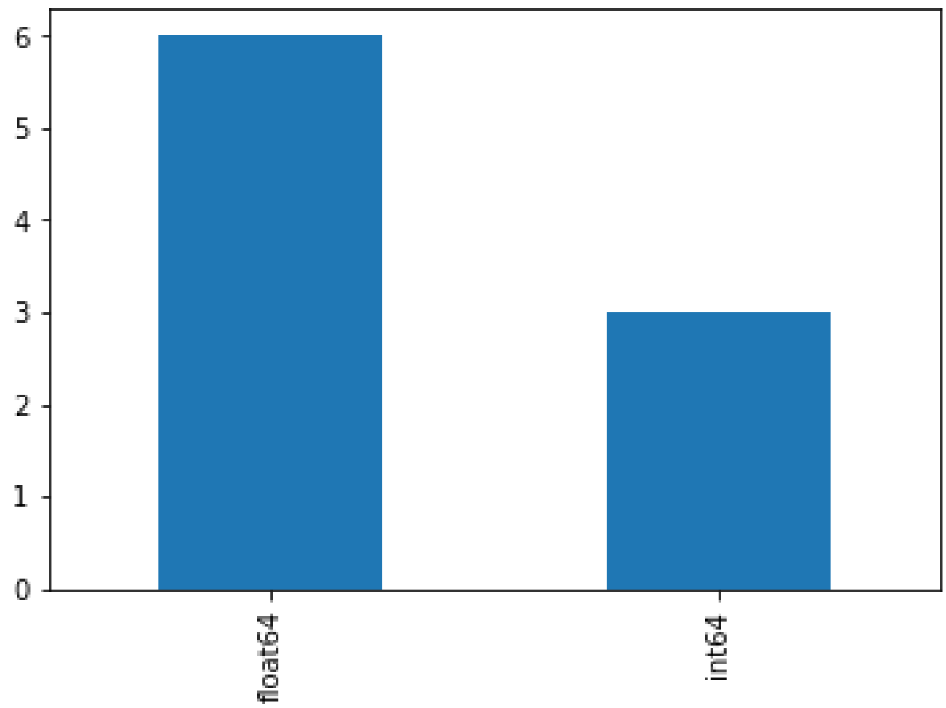
[26]:
```
data['BMI'].value_counts().head(5)
```

[26]:
```
32.000000    13
31.600000    12
```

```
31.200000    12
31.992578    11
33.300000    10
Name: BMI, dtype: int64
```

[27]:

```
data.dtypes.value_counts().plot(kind='bar')
plt.show()
```

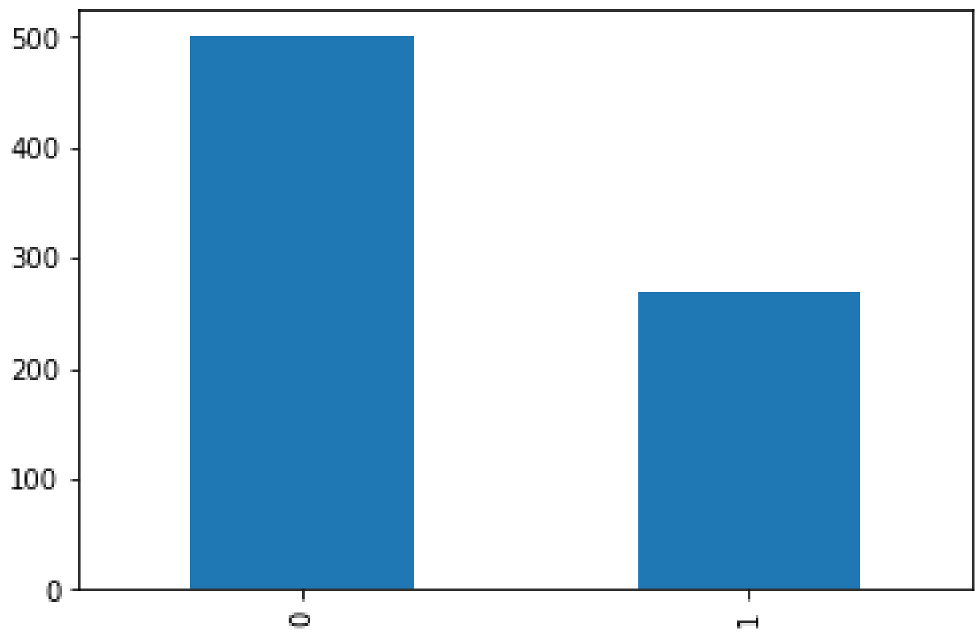


We can see 6 float columns and 3 integer columns.

WEEK 2: Data Exploration

[28]:

```
(data.Outcome).value_counts().plot(kind = 'bar')
plt.show()
```

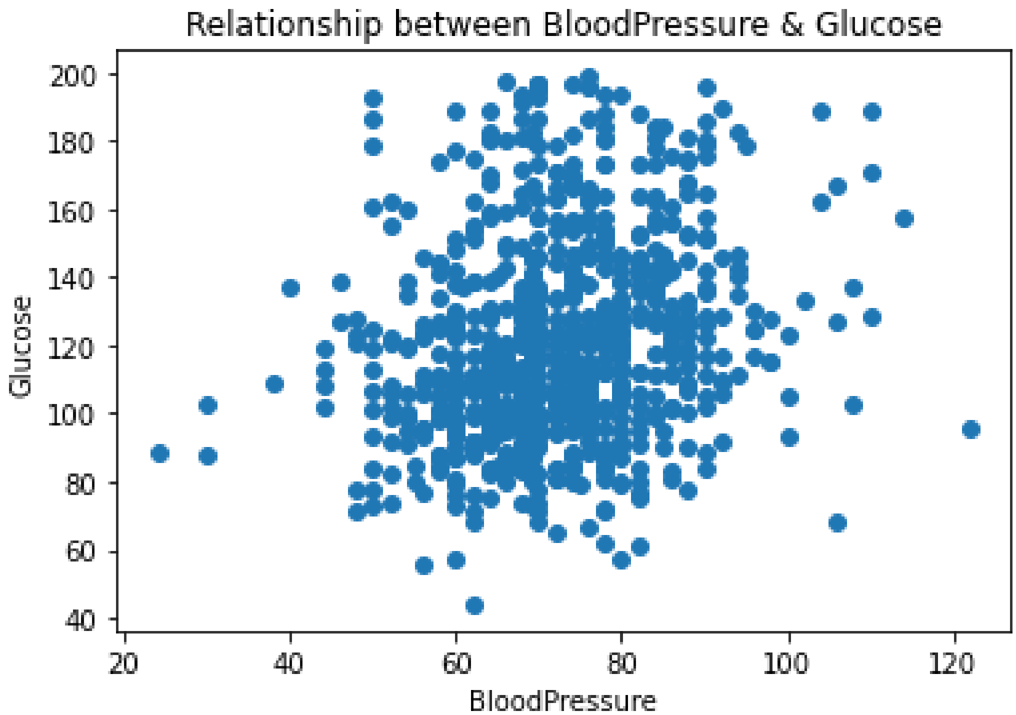


Data is Imbalanced

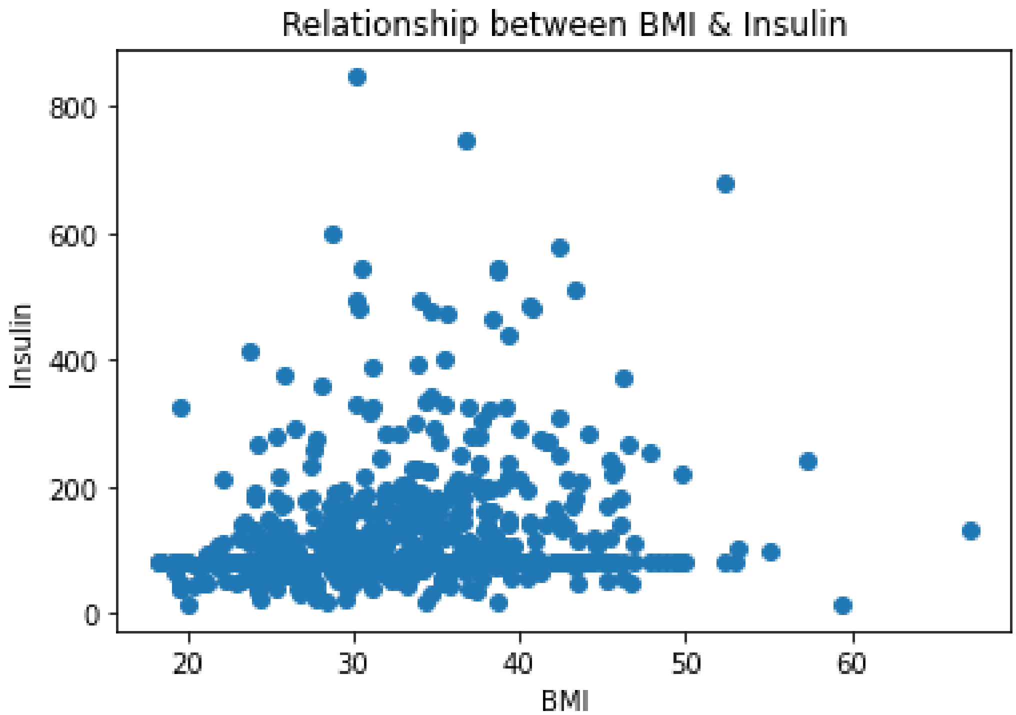
1 Scatter plot

[29]:
```
BloodPressure = data['BloodPressure']
Glucose = data['Glucose']
SkinThickness = data['SkinThickness']
Insulin = data['Insulin']
BMI = data['BMI']
```

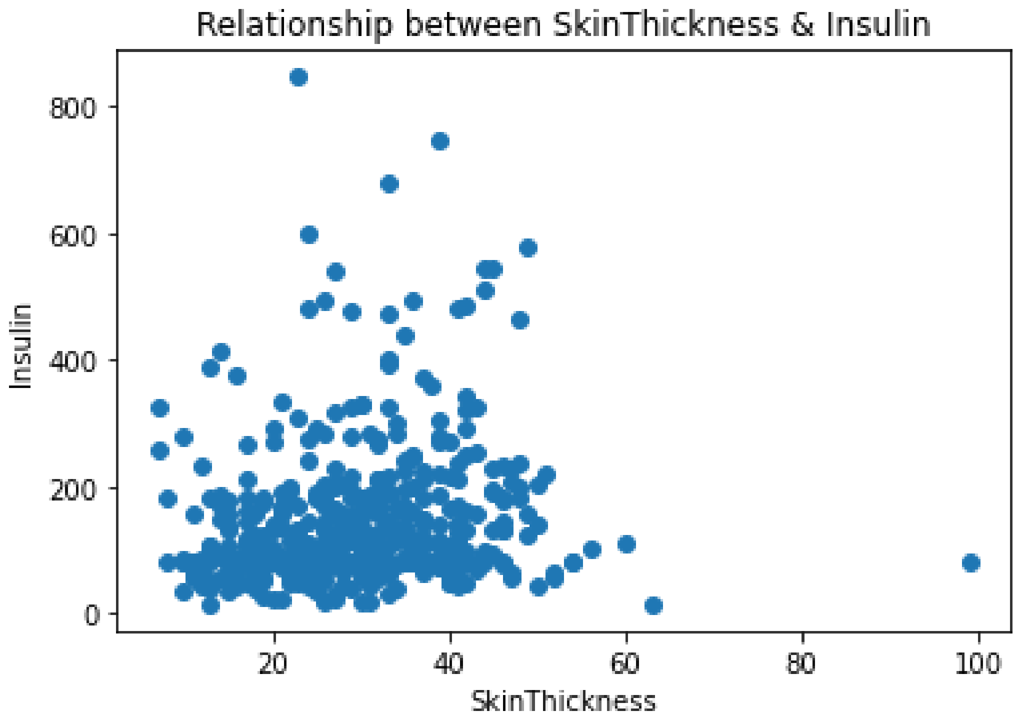
[30]:
```
plt.scatter(BloodPressure, Glucose)
plt.xlabel('BloodPressure')
plt.ylabel('Glucose')
plt.title('Relationship between BloodPressure & Glucose')
plt.show()
```



[31]:
```
plt.scatter(BMI, Insulin)
plt.xlabel('BMI')
plt.ylabel('Insulin')
plt.title('Relationship between BMI & Insulin')
plt.show()
```



[32]:
```
plt.scatter(SkinThickness, Insulin)
plt.xlabel('SkinThickness')
plt.ylabel('Insulin')
plt.title('Relationship between SkinThickness & Insulin')
plt.show()
```



[33]:
```
# correlation matrix
data.corr()
```

[33]:

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

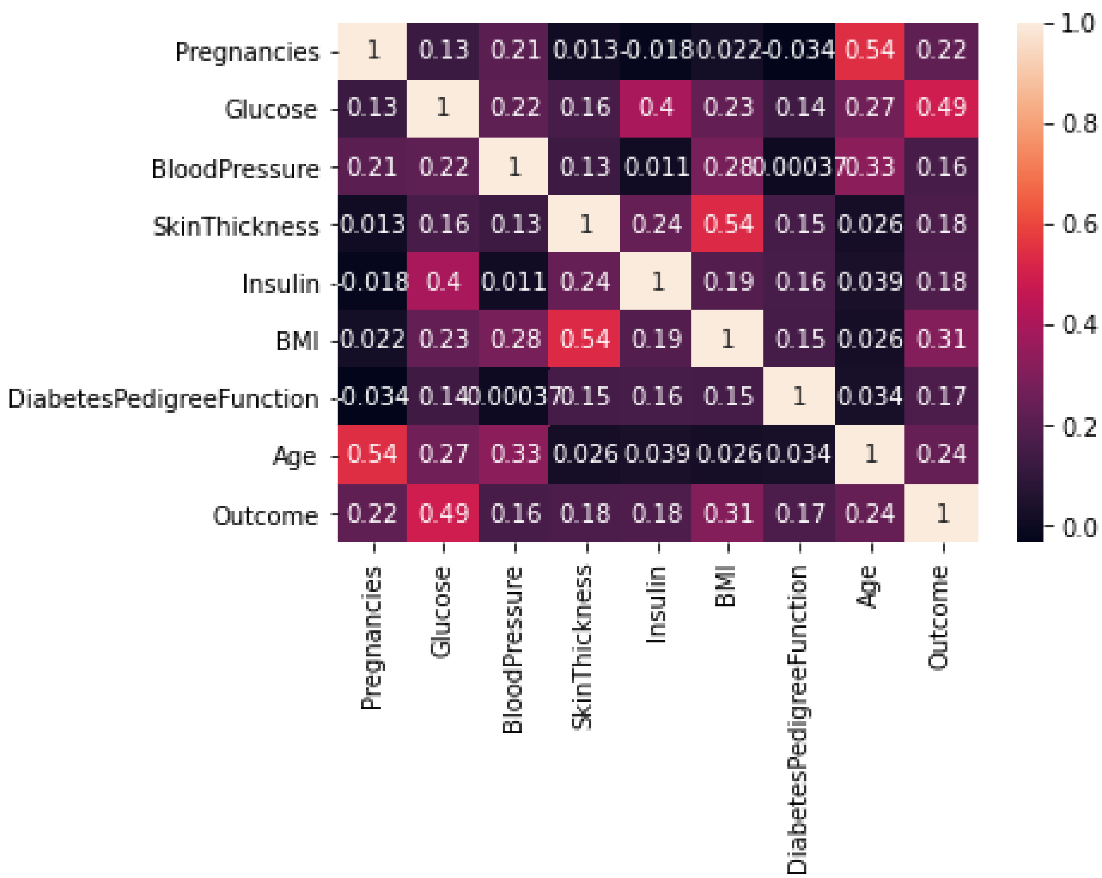
	Insulin	BMI	DiabetesPedigreeFunction	\
Pregnancies	-0.018082	0.021546	-0.033523	
Glucose	0.396597	0.231478	0.137106	
BloodPressure	0.010926	0.281231	0.000371	
SkinThickness	0.240361	0.535703	0.154961	
Insulin	1.000000	0.189856	0.157806	
BMI	0.189856	1.000000	0.153508	
DiabetesPedigreeFunction	0.157806	0.153508	1.000000	
Age	0.038652	0.025748	0.033561	

```
Outcome                   0.179185  0.312254                  0.173844

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

[34]:
```
snss.heatmap(data.corr(),annot=True)
```

[34]: <AxesSubplot:>



[35]:
```
data.corr().style.background_gradient()
```

[35]: <pandas.io.formats.style.Styler at 0x17064a4d550>

WEEK 3: Data Modeling

[36]:
```
data.head()
```

[36]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin   BMI  \
0            6    148.0           72.0      35.000000   79.799479  33.6
1            1     85.0           66.0      29.000000   79.799479  26.6
2            8    183.0           64.0      20.536458   79.799479  23.3
3            1     89.0           66.0      23.000000   94.000000  28.1
4            0    137.0           40.0      35.000000  168.000000  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

[37]:
```
x=data.iloc[:,[0,1,2,3,4,5,6,7]].values
y=data['Outcome'].values
```

[38]:
```
from sklearn.model_selection import train_test_split
```

[39]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.2)
```

LogisticRegression

[40]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train,y_train)
```

```
G:\Software\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:762:
ConvergenceWarning: 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(
```

[40]: LogisticRegression()

[41]:
```
print(model.score(x_train,y_train))
print(model.score(x_test,y_test))
```

```
0.758957654723127
0.8311688311688312
```

[42]:
```
from sklearn.metrics import classification_report
print(classification_report(y,model.predict(x)))
```

```
              precision    recall  f1-score   support

           0       0.80      0.87      0.83       500
           1       0.71      0.59      0.64       268

    accuracy                           0.77       768
   macro avg       0.76      0.73      0.74       768
weighted avg       0.77      0.77      0.77       768
```

DecisionTreeClassifier

[43]:
```
from sklearn.tree import DecisionTreeClassifier
model2 = DecisionTreeClassifier(max_depth=5)
model2.fit(x_train,y_train)
```

[43]:
```
DecisionTreeClassifier(max_depth=5)
```

[44]:
```
print(model2.score(x_train,y_train))
print(model2.score(x_test,y_test))
```

```
0.8208469055374593
0.7662337662337663
```

[45]:
```
from sklearn.metrics import classification_report
print(classification_report(y,model2.predict(x)))
```

```
              precision    recall  f1-score   support

           0       0.89      0.81      0.85       500
           1       0.69      0.81      0.75       268

    accuracy                           0.81       768
   macro avg       0.79      0.81      0.80       768
weighted avg       0.82      0.81      0.81       768
```

KNN

[46]:
```
from sklearn.neighbors import KNeighborsClassifier
model3 = KNeighborsClassifier(n_neighbors=7,
                             metric='minkowski',
                             p = 2)
```

```
model3.fit(x_train,y_train)
```

[46]: KNeighborsClassifier(n_neighbors=7)

[47]:
```
print(model3.score(x_train,y_train))
print(model3.score(x_test,y_test))
```

```
0.7899022801302932
0.7337662337662337
```

[48]:
```
from sklearn.metrics import classification_report
print(classification_report(y,model3.predict(x)))
```

```
              precision    recall  f1-score   support

           0       0.81      0.86      0.83       500
           1       0.71      0.63      0.66       268

    accuracy                           0.78       768
   macro avg       0.76      0.74      0.75       768
weighted avg       0.77      0.78      0.78       768
```

WEEK 4: Data Modeling

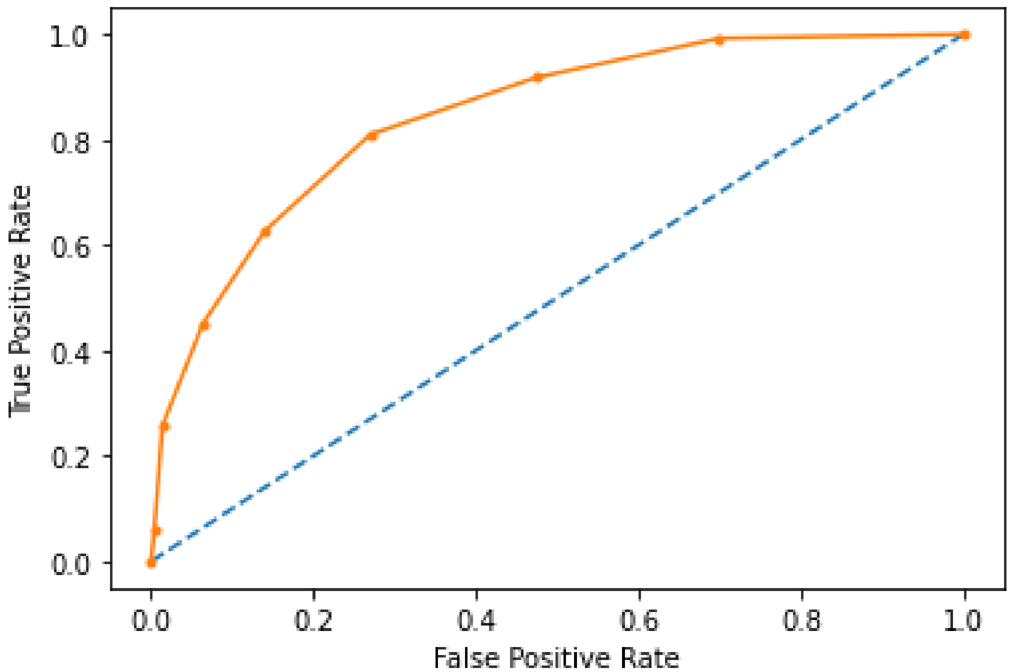
[49]:
```
# ROC Curve KNN
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

probs = model3.predict_proba(x)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(y, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
fpr, tpr, thresholds = roc_curve(y, probs)
print("True Positive Rate - {}, False Positive Rate - {} Thresholds - {}".
 ↪format(tpr,fpr,thresholds))
# plot the roc curve for the model
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

```
AUC: 0.845
True Positive Rate - [0.         0.05970149 0.25746269 0.45149254 0.62686567
0.80970149
```

```
 0.91791045 0.99253731 1.        ], False Positive Rate - [0.    0.004 0.014
0.064 0.14  0.27  0.474 0.7   1.   ] Thresholds - [2.         1.
0.85714286 0.71428571 0.57142857 0.42857143
 0.28571429 0.14285714 0.        ]
```

[49]: Text(0, 0.5, 'True Positive Rate')



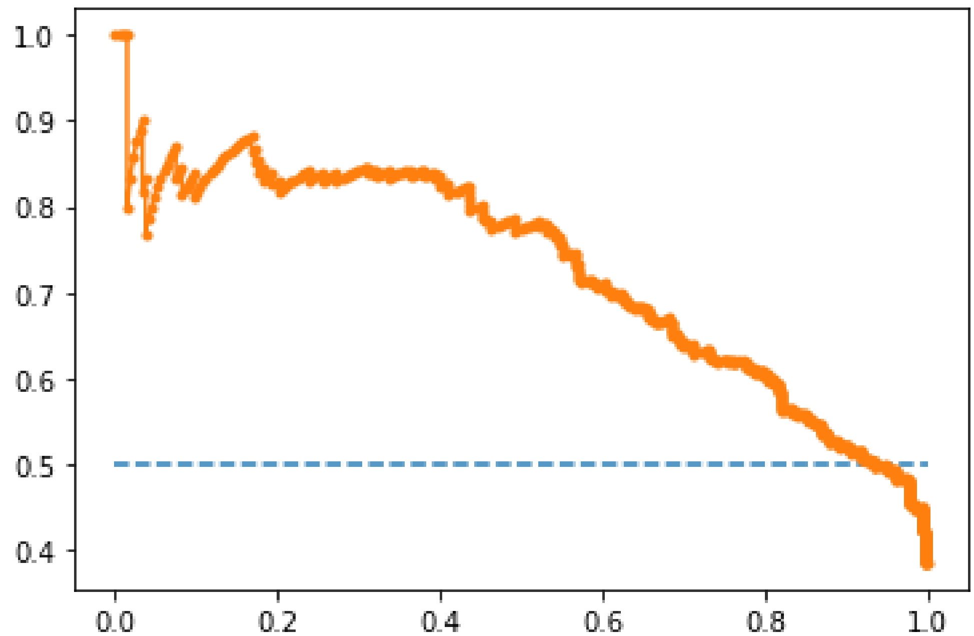
[50]:
```python
#Precision Recall Curve for Logistic Regression

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(x)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model.predict(x)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(y, probs)
# calculate F1 score
f1 = f1_score(y, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
```

```
# calculate average precision score
ap = average_precision_score(y, 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=0.643 auc=0.727 ap=0.728
```

[50]: [<matplotlib.lines.Line2D at 0x170652f4820>]



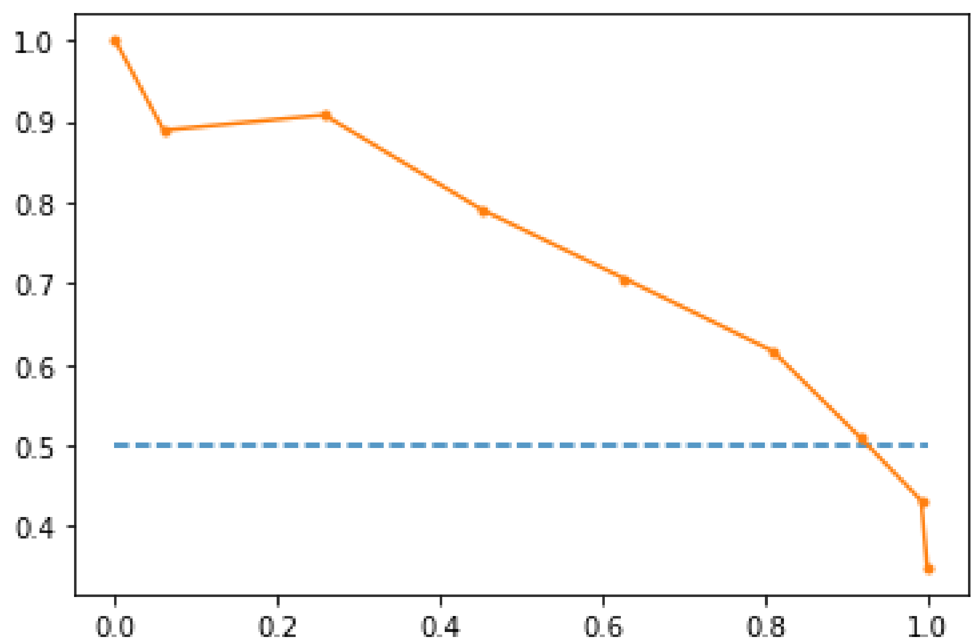
[51]:
```
#Precision Recall Curve for KNN

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 = model3.predict_proba(x)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model3.predict(x)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(y, probs)
```

```python
# calculate F1 score
f1 = f1_score(y, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(y, 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=0.664 auc=0.750 ap=0.713
```

[51]: [<matplotlib.lines.Line2D at 0x17065350a90>]



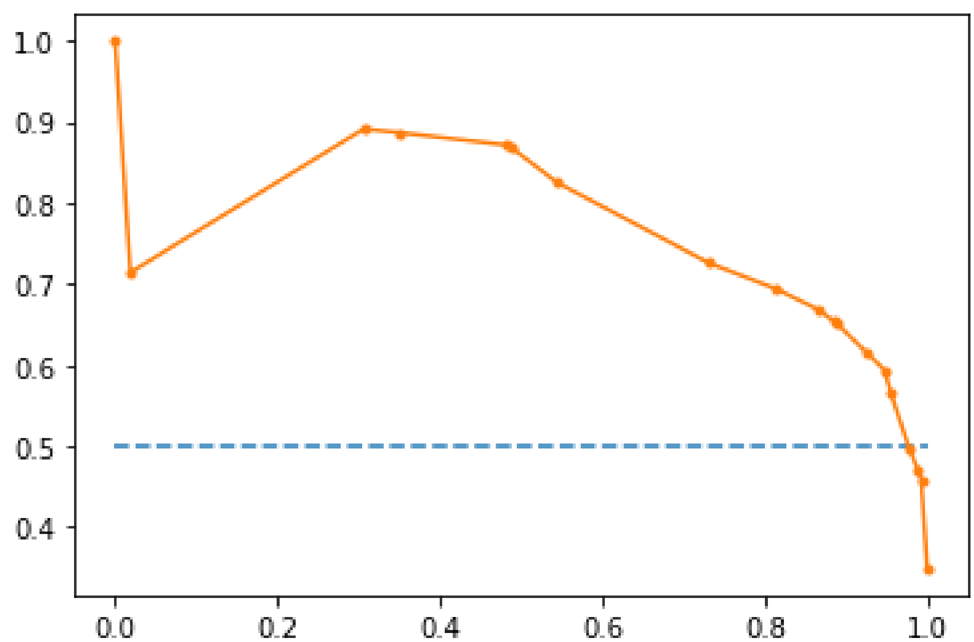
[52]:
```python
#Precision Recall Curve for Decission Tree Classifier

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 = model2.predict_proba(x)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
```

```python
# predict class values
yhat = model2.predict(x)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(y, probs)
# calculate F1 score
f1 = f1_score(y, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(y, 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=0.749 auc=0.772 ap=0.779
```

[52]:
```
[<matplotlib.lines.Line2D at 0x170653aad30>]
```



[ ]: