Healthcare

Week 1

Data Exploration:

In [264]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
%matplotlib inline
```

In [265]:
```
df=pd.read_csv('health care diabetes.csv')
df.head()
```

Out[265]:

	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

1. Perform descriptive analysis.

In [266]:
```
df.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 [267]:
```
df.describe()
```

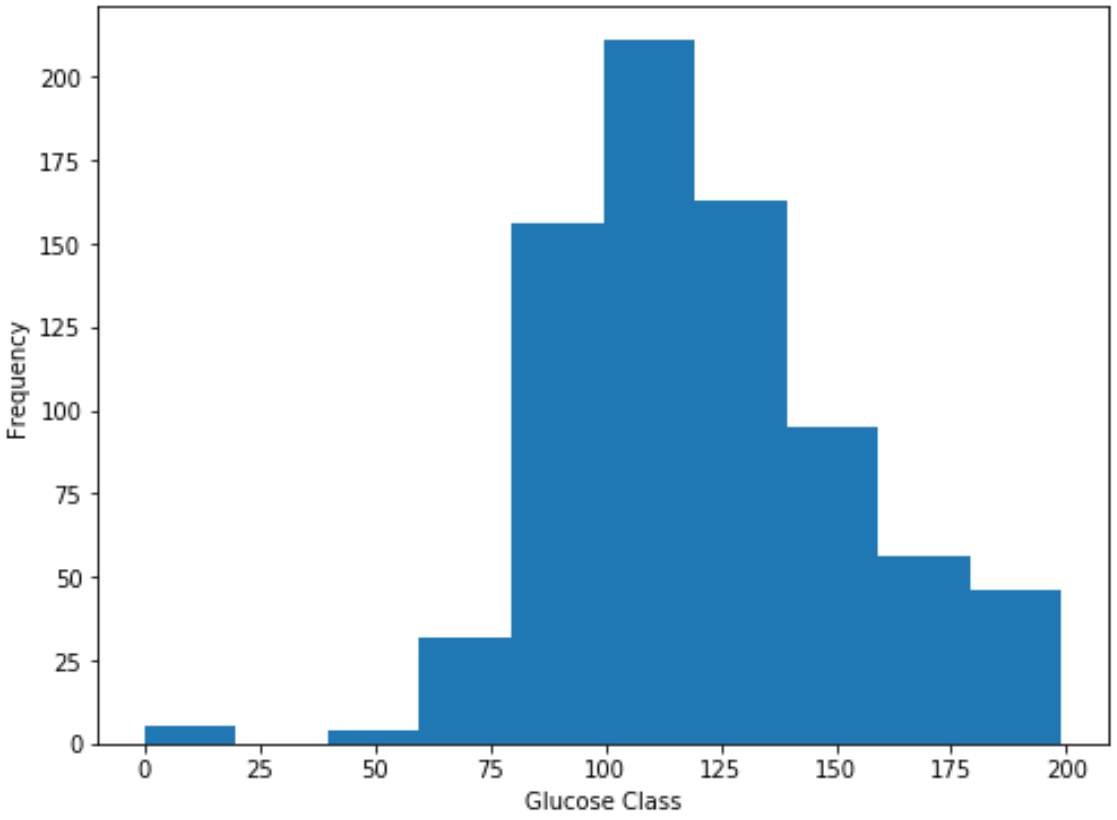
Out[267]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	7
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	

2. Visually explore these variables using histograms. Treat the missing values accordingly.

In [268]:
```
plt.figure(figsize=(8,6))
plt.xlabel('Glucose Class')
df['Glucose'].plot.hist()
print("Mean of Glucose level is :-", df['Glucose'].mean())
```

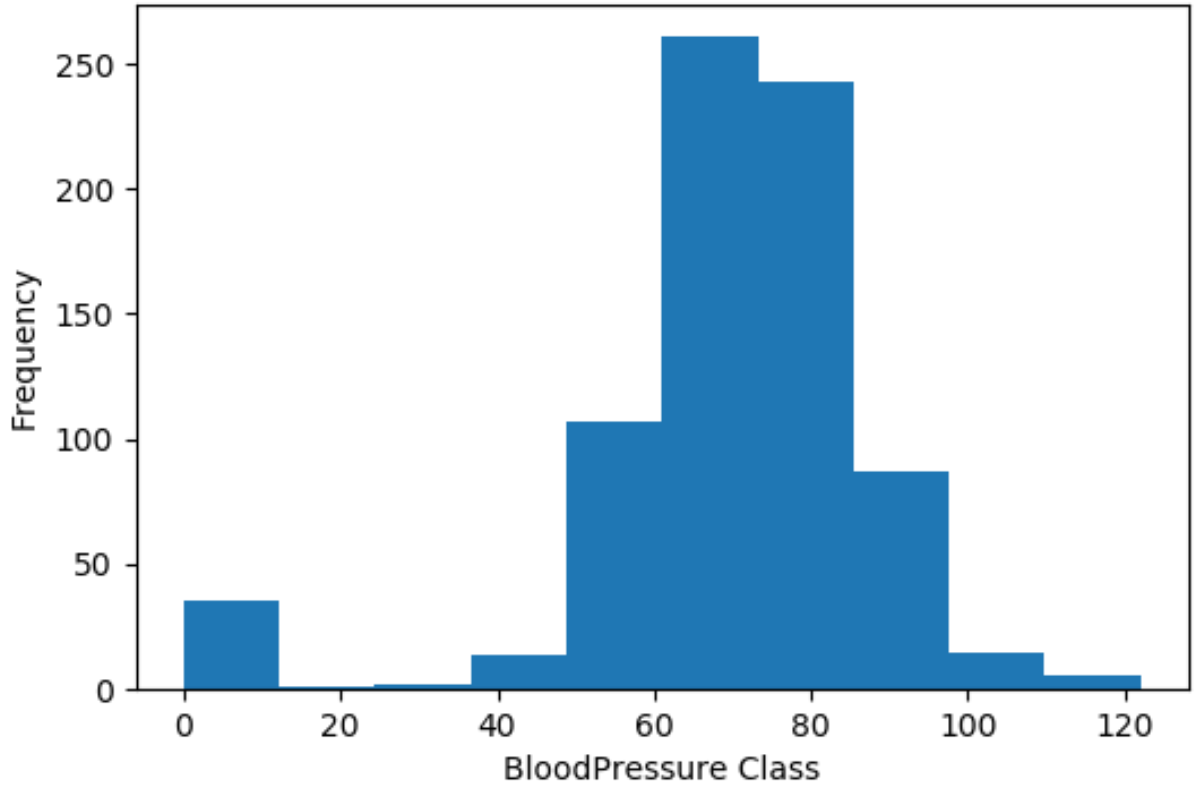
```
Mean of Glucose level is :- 120.89453125
```



In [269]:
```
df['Glucose']=df['Glucose'].replace(0,df['Glucose'].mean())
```

In [270]:
```
plt.figure(figsize=(6,4),dpi=100)
plt.xlabel('BloodPressure Class')
df['BloodPressure'].plot.hist()
print("Mean of BloodPressure level is :-", df['BloodPressure'].mean())
```

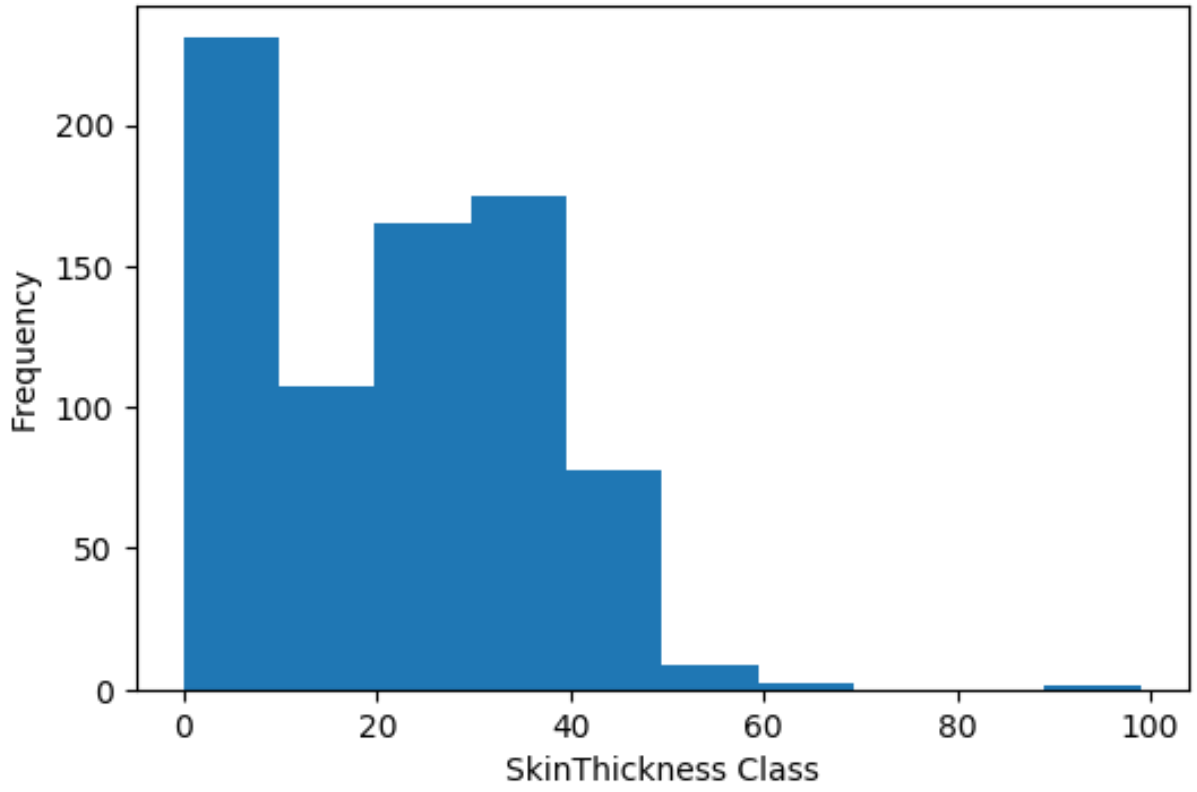
```
Mean of BloodPressure level is :- 69.10546875
```



In [271]:
```
df['BloodPressure']=df['BloodPressure'].replace(0,df['BloodPressure'].mean())
```

In [272]:
```
plt.figure(figsize=(6,4),dpi=100)
plt.xlabel('SkinThickness Class')
df['SkinThickness'].plot.hist()
print("Mean of SkinThickness is :-", df['SkinThickness'].mean())
```

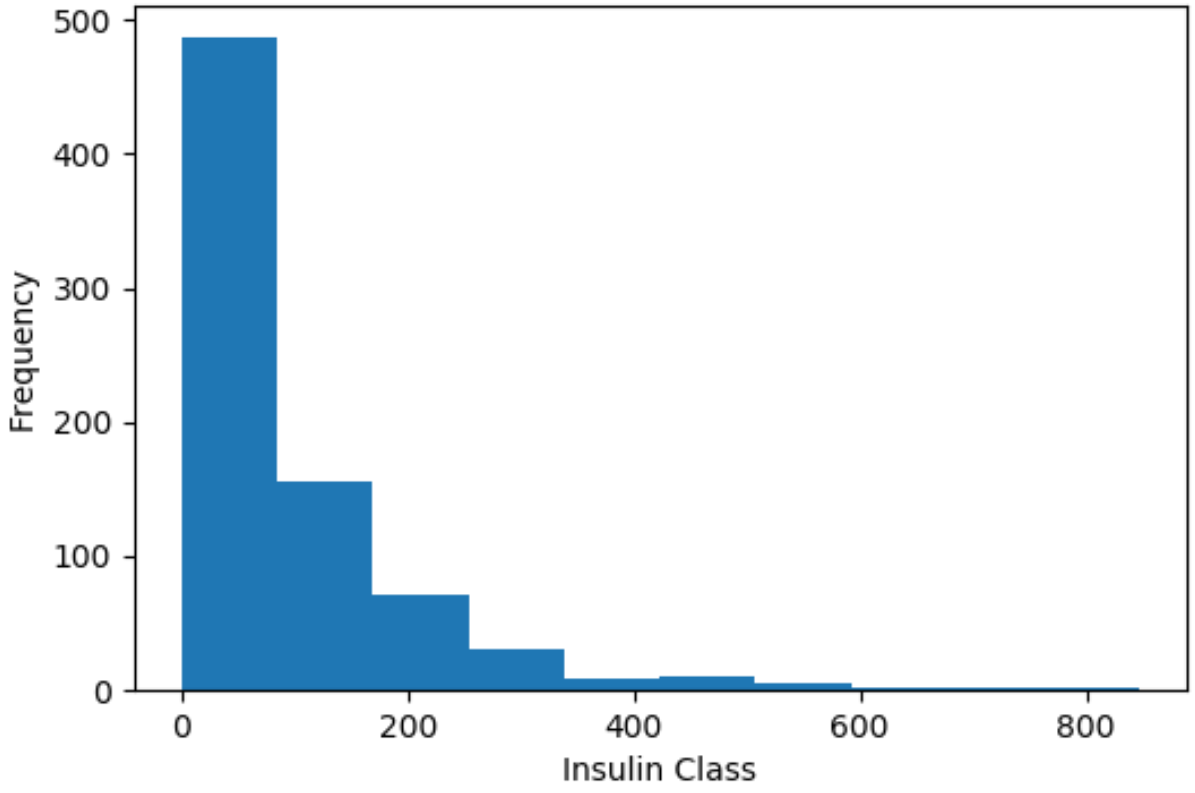
```
Mean of SkinThickness is :- 20.536458333333332
```



In [273]:
```
df['SkinThickness']=df['SkinThickness'].replace(0,df['SkinThickness'].mean())
```

In [274]:
```
plt.figure(figsize=(6,4),dpi=100)
plt.xlabel('Insulin Class')
df['Insulin'].plot.hist()
print("Mean of Insulin is :-", df['Insulin'].mean())
```

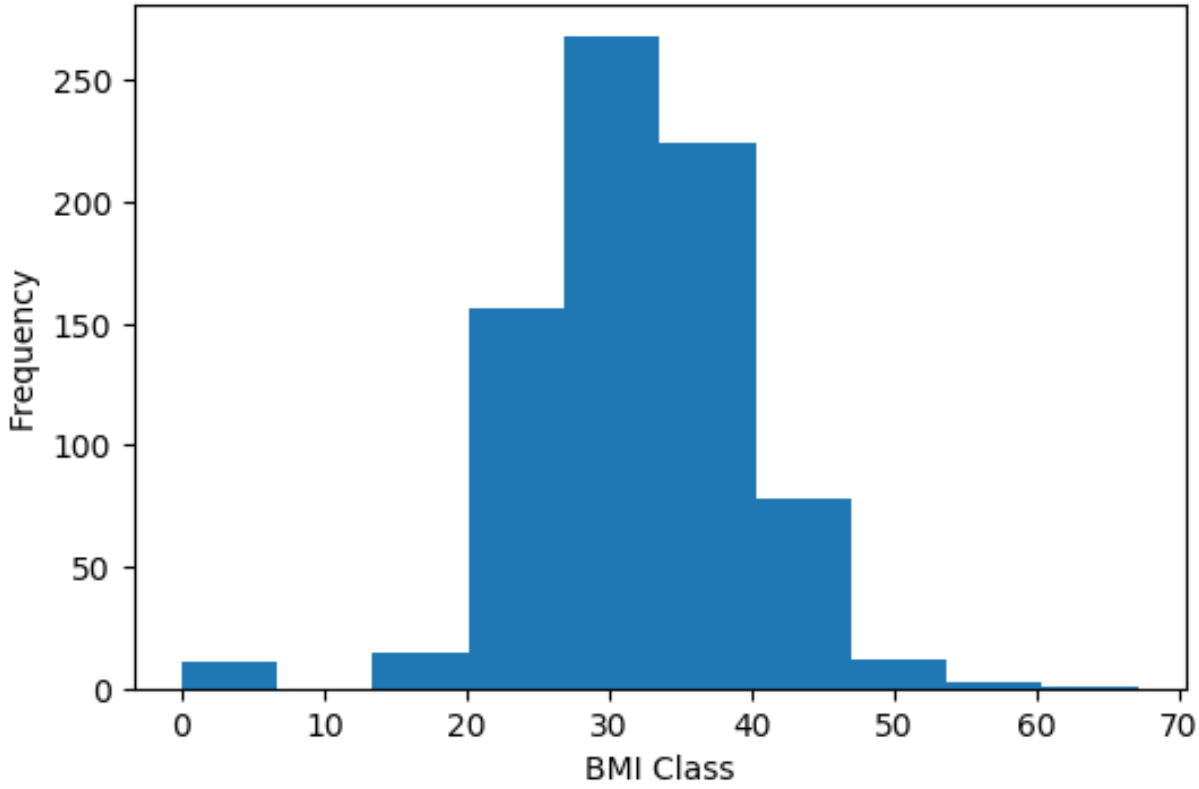
```
Mean of Insulin is :- 79.79947916666667
```



In [275]:
```
df['Insulin']=df['Insulin'].replace(0,df['Insulin'].mean())
```

In [276]:
```
plt.figure(figsize=(6,4),dpi=100)
plt.xlabel('BMI Class')
df['BMI'].plot.hist()
print("Mean of BMI is :-", df['BMI'].mean())
```

```
Mean of BMI is :- 31.992578124999977
```



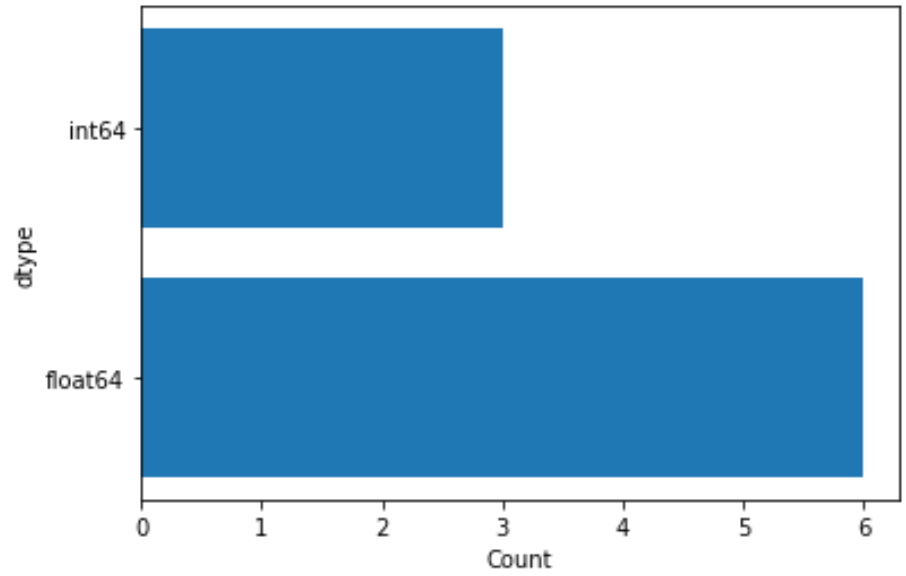
In [277]:
```
df['BMI']=df['BMI'].replace(0,df['BMI'].mean())
```

3. 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 [278]:
```
df1=pd.DataFrame(df.dtypes.value_counts(),columns = ['Count'])
df1.reset_index(level=0, inplace=True)
l=(str(df1['index'][0]),str(df1['index'][1]))
yy=df1['Count']
plt.barh(l,yy)
plt.xlabel('Count')
plt.ylabel('dtype')
```

Out[278]:
```
Text(0, 0.5, 'dtype')
```



In [279]:
```
df.head()
```

Out[279]:

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

In [280]:
```
df.tail()
```

Out[280]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101.0	76.0	48.000000	180.000000	32.9	0.171	63	0
764	2	122.0	70.0	27.000000	79.799479	36.8	0.340	27	0
765	5	121.0	72.0	23.000000	112.000000	26.2	0.245	30	0
766	1	126.0	60.0	20.536458	79.799479	30.1	0.349	47	1
767	1	93.0	70.0	31.000000	79.799479	30.4	0.315	23	0

In [281]:
```
df.to_csv('week1_treated.csv',index=False)
```

Week 2

Data Exploration:

In [282]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [283]:
```
df=pd.read_csv('week1_treated.csv')
df.head()
```

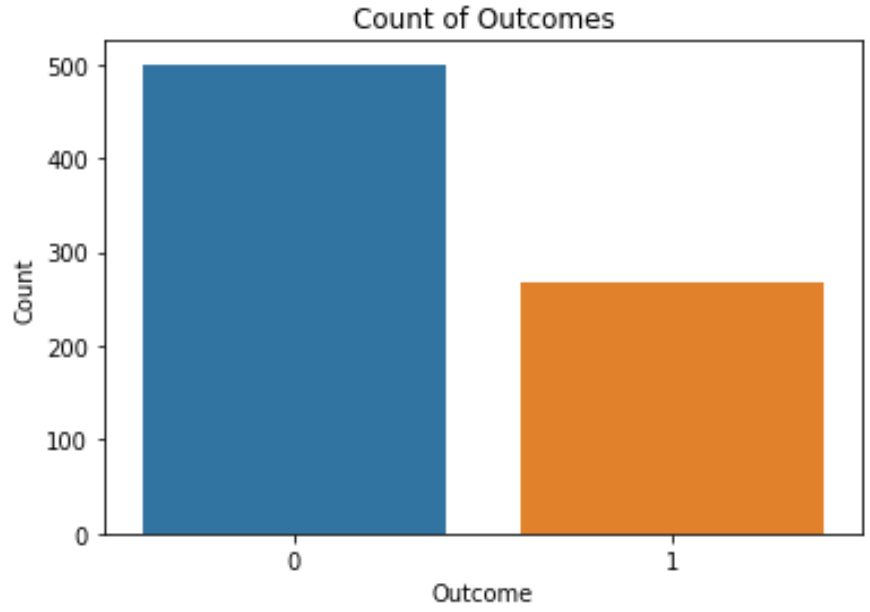
Out[283]:

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

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.

In [284]:
```
sns.countplot(df['Outcome'])
plt.title("Count of Outcomes")
plt.xlabel('Outcome')
plt.ylabel("Count")
df['Outcome'].value_counts()
```

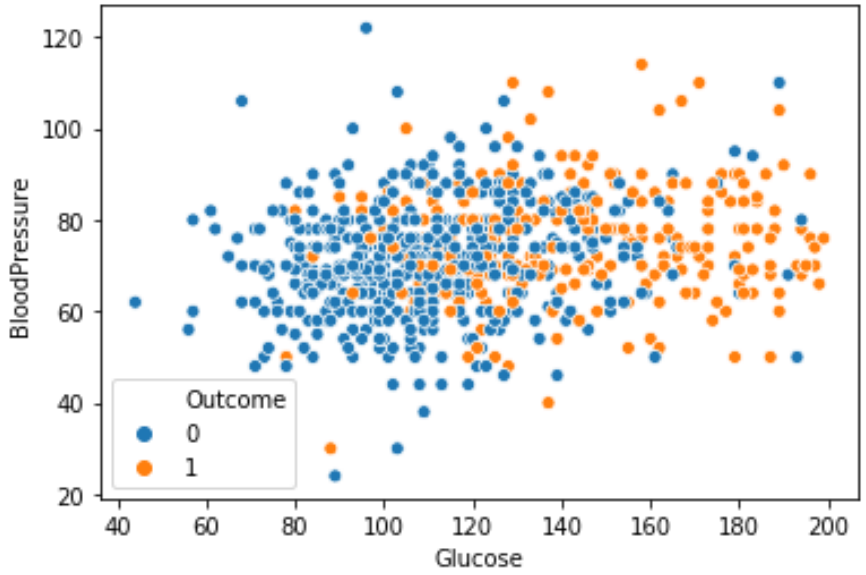
Out[284]:
```
0    500
1    268
Name: Outcome, dtype: int64
```



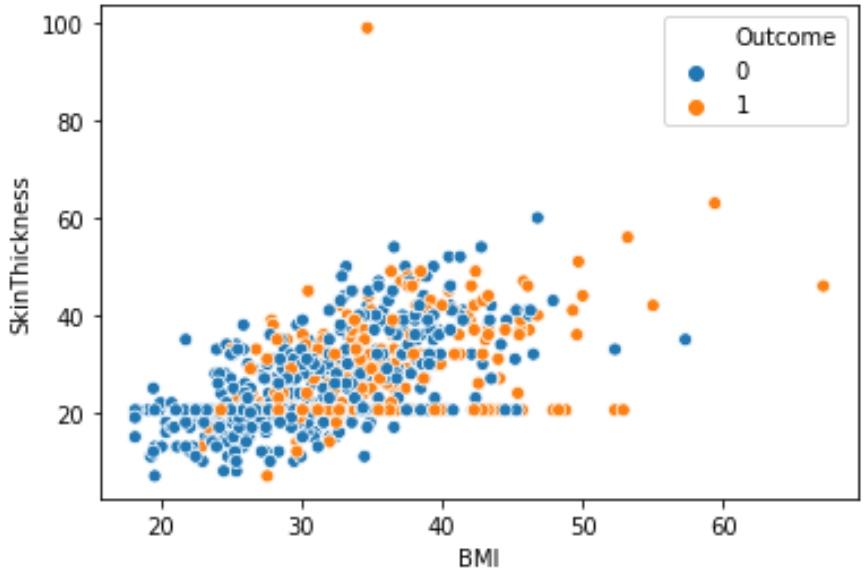
2. Create scatter charts between the pair of variables to understand the relationships. Describe your findings.

In [285]:
```
BloodPressure = df['BloodPressure']
Glucose = df['Glucose']
SkinThickness = df['SkinThickness']
Insulin = df['Insulin']
BMI = df['BMI']
Pregnancies=df['Pregnancies']
```

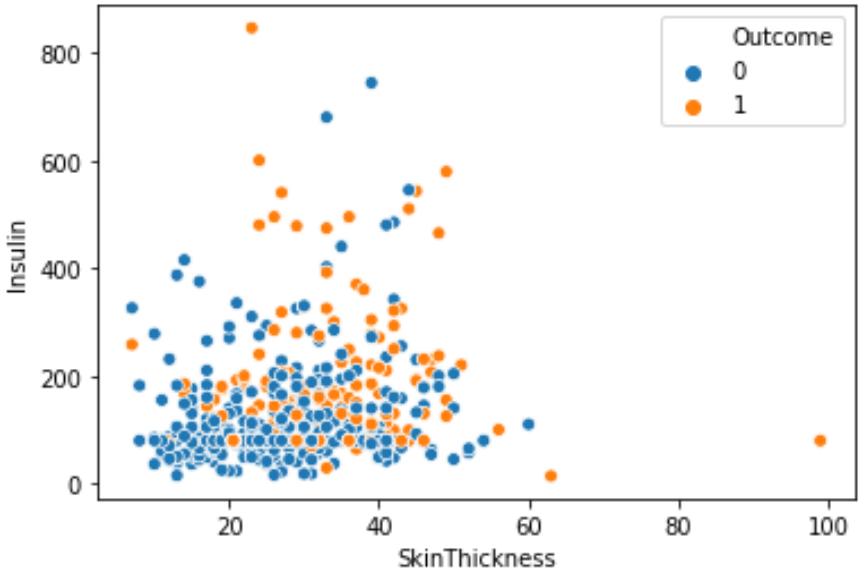
In [286]:
```
sns.scatterplot(x= "Glucose" ,y= "BloodPressure",hue="Outcome",data=df);
```



In [287]:
```
sns.scatterplot(x= "BMI" ,y= "SkinThickness",hue="Outcome",data=df);
```

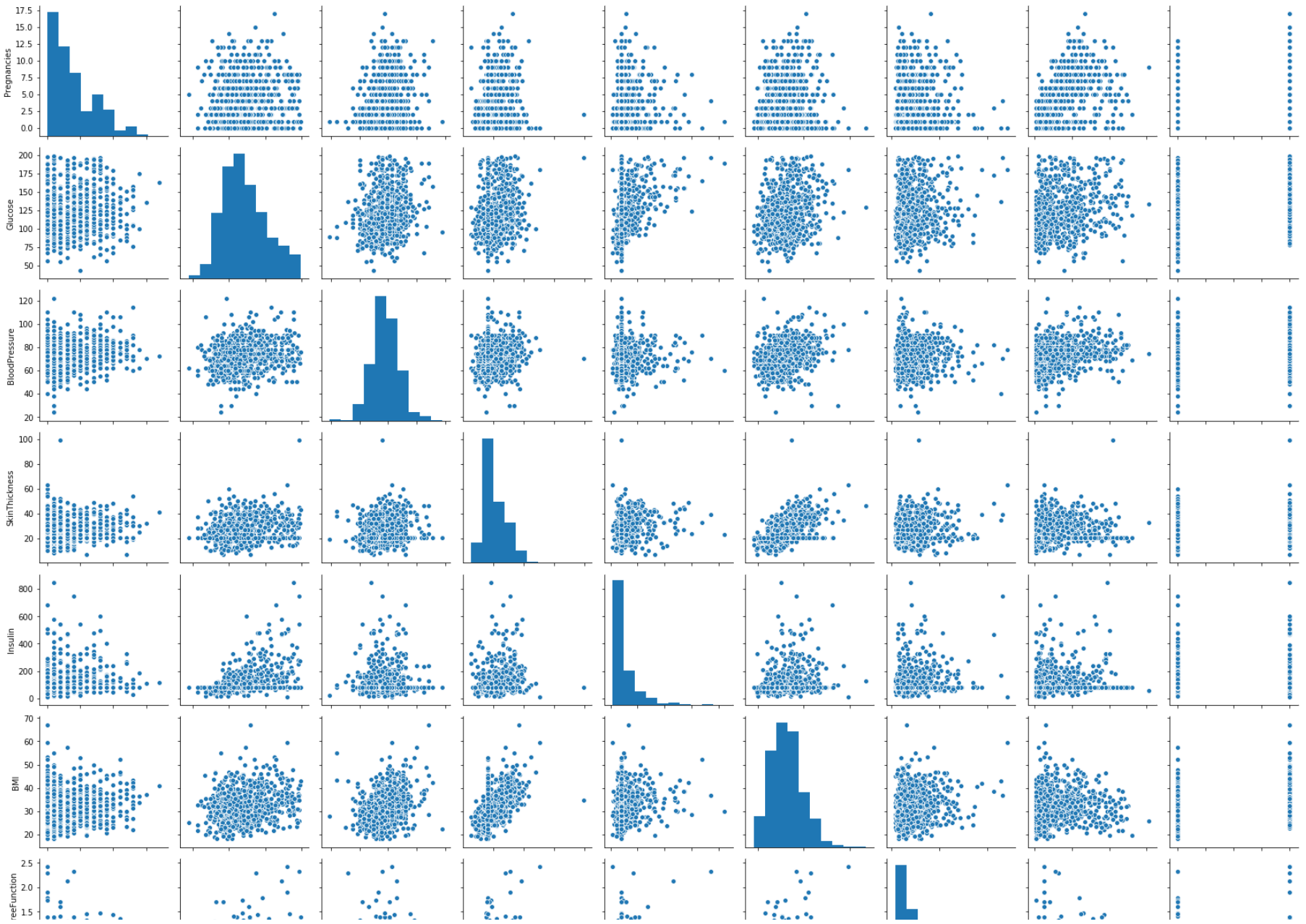


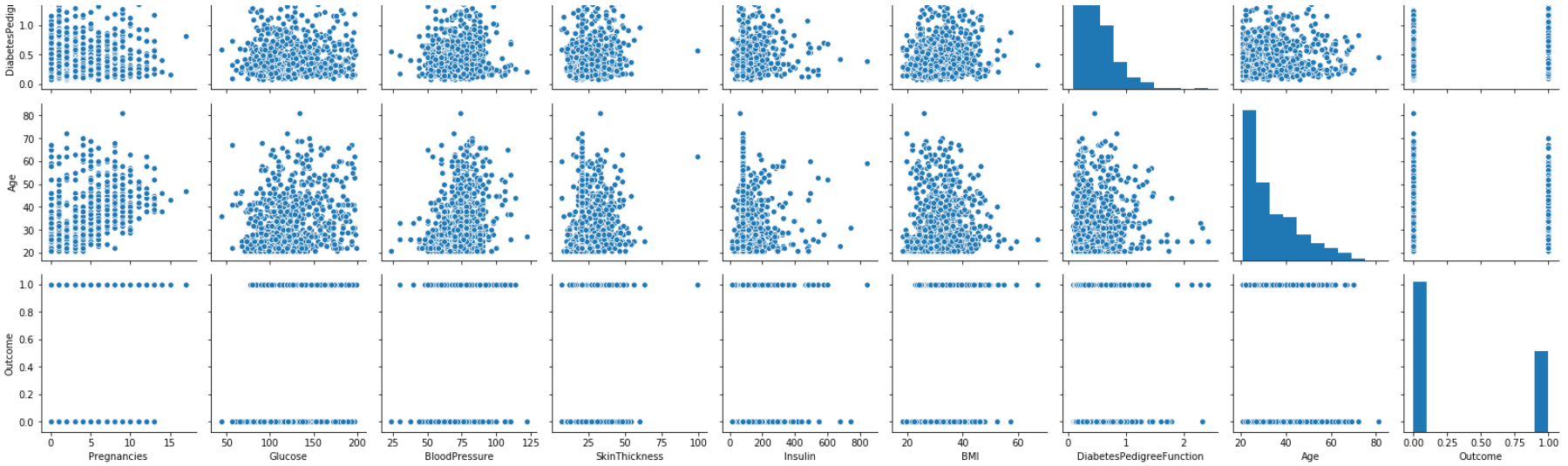
In [288]:
```
sns.scatterplot(x= "SkinThickness" ,y= "Insulin",hue="Outcome",data=df);
```



In [289]:
```
sns.pairplot(df)
```

Out[289]:
```
<seaborn.axisgrid.PairGrid at 0xa46a42f1c8>
```





3. Perform correlation analysis. Visually explore it using a heat map.

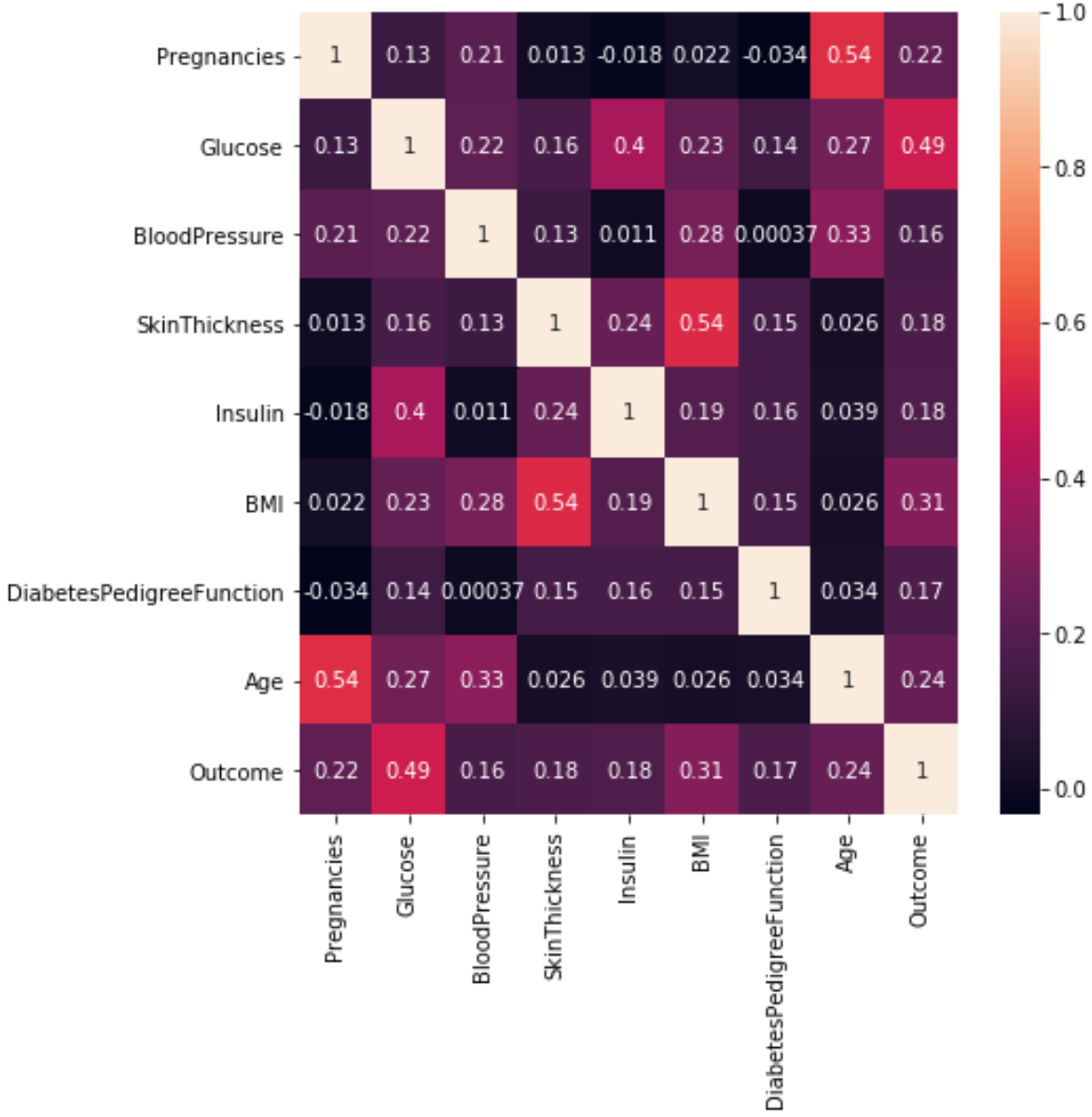
In [290]:
```
df.corr()
```

Out[290]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
Pregnancies	1.000000	0.127964	0.208984	0.013376	-0.018082	0.021546	-0.033523
Glucose	0.127964	1.000000	0.219666	0.160766	0.396597	0.231478	0.137106
BloodPressure	0.208984	0.219666	1.000000	0.134155	0.010926	0.281231	0.000371
SkinThickness	0.013376	0.160766	0.134155	1.000000	0.240361	0.535703	0.154961
Insulin	-0.018082	0.396597	0.010926	0.240361	1.000000	0.189856	0.157806
BMI	0.021546	0.231478	0.281231	0.535703	0.189856	1.000000	0.153508
DiabetesPedigreeFunction	-0.033523	0.137106	0.000371	0.154961	0.157806	0.153508	1.000000
Age	0.544341	0.266600	0.326740	0.026423	0.038652	0.025748	0.033561
Outcome	0.221898	0.492908	0.162986	0.175026	0.179185	0.312254	0.173844

In [292]:
```
plt.subplots(figsize=(7,7))
sns.heatmap(df.corr(),annot=True)
```

Out[292]: <matplotlib.axes._subplots.AxesSubplot at 0xa46da4d788>



Week 3

Data Modeling:

In [293]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
from sklearn import metrics
```

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.

In [294]:
```
df=pd.read_csv('week1_treated.csv')
df.head()
```

Out[294]:

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

In [295]:
```
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
```

In [296]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
```

In [297]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [298]:
```
from sklearn.preprocessing import StandardScaler
```

In [299]:
```
Scale=StandardScaler()
x_train_std=Scale.fit_transform(x_train)
x_test_std=Scale.transform(x_test)
```

Project Task: Week 4

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.

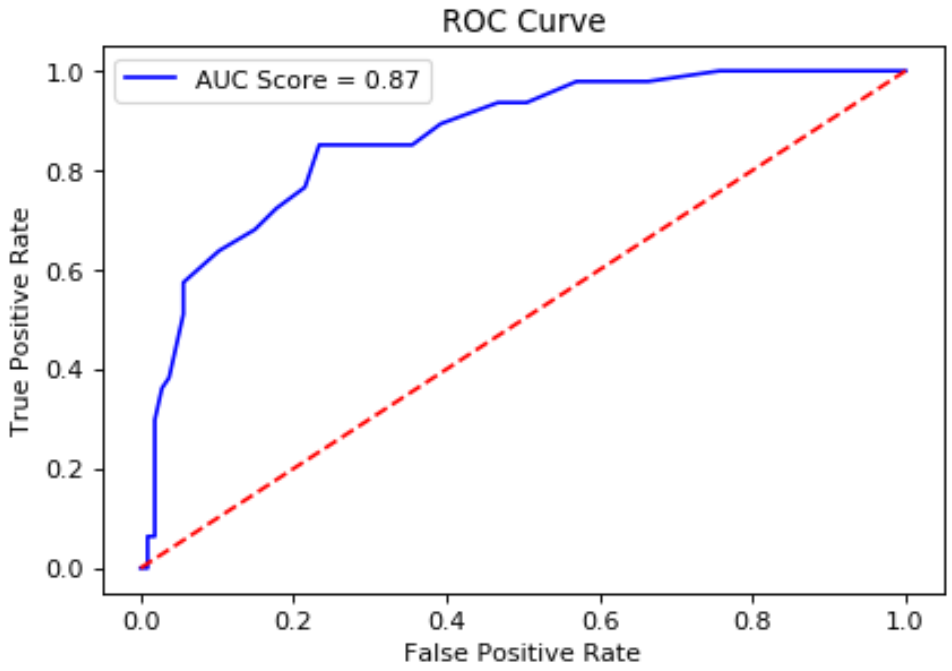
KNN

In [305]:
```
from sklearn.neighbors import KNeighborsClassifier
knn_model = KNeighborsClassifier(n_neighbors=25)
knn_model.fit(x_train_std,y_train)
knn_pred=knn_model.predict(x_test_std)
```

In [306]:
```
print("Model Validation ==>\n")
print("Accuracy Score of KNN Model::")
print(metrics.accuracy_score(y_test,knn_pred))
print("\n","Classification Report::")
print(metrics.classification_report(y_test,knn_pred),'\n')
print("\n","ROC Curve")
knn_prob=knn_model.predict_proba(x_test_std)
knn_prob1=knn_prob[:,1]
fpr,tpr,thresh=metrics.roc_curve(y_test,knn_prob1)
roc_auc_knn=metrics.auc(fpr,tpr)
plt.figure(dpi=80)
plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_knn)
plt.plot(fpr,fpr,'r--',color='red')
plt.legend()
```

```
Model Validation ==>

Accuracy Score of KNN Model::
0.8181818181818182

 Classification Report::
              precision    recall  f1-score   support

           0       0.85      0.90      0.87       107
           1       0.73      0.64      0.68        47

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



 ROC Curve
```

Out[306]: <matplotlib.legend.Legend at 0xa46f15d5c8>



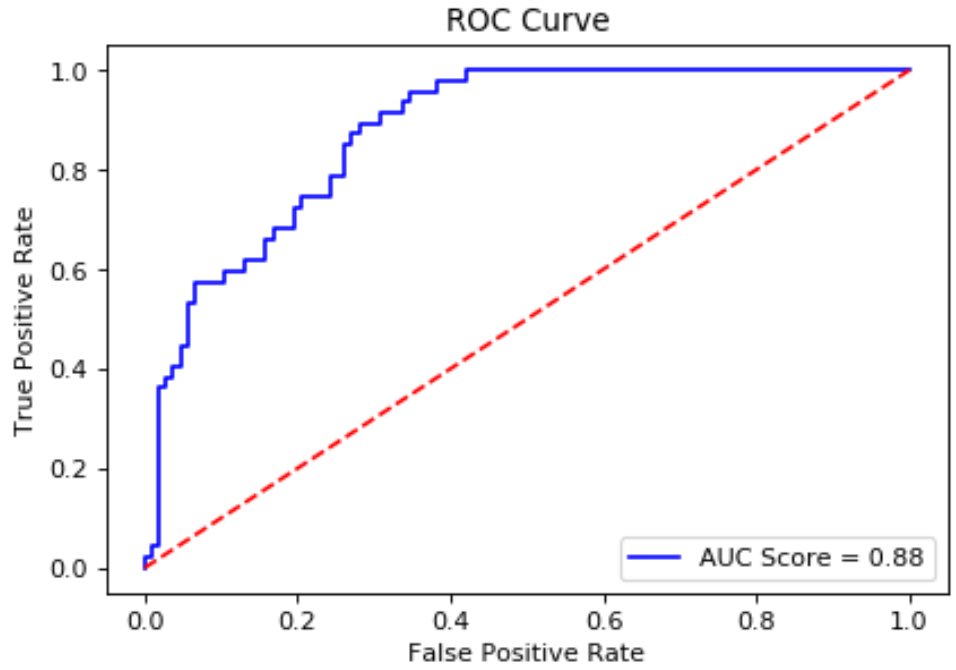
Logistic Regression

In [307]:
```
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression(C=0.01)
lr_model.fit(x_train_std,y_train)
lr_pred=lr_model.predict(x_test_std)
```

In [308]:
```
print("Model Validation ==>\n")
print("Accuracy Score of Logistic Regression Model::")
print(metrics.accuracy_score(y_test,lr_pred))
print("\n","Classification Report::")
print(metrics.classification_report(y_test,lr_pred),'\n')
print("\n","ROC Curve")
lr_prob=lr_model.predict_proba(x_test_std)
lr_prob1=lr_prob[:,1]
fpr,tpr,thresh=metrics.roc_curve(y_test,lr_prob1)
roc_auc_lr=metrics.auc(fpr,tpr)
plt.figure(dpi=80)
plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_lr)
plt.plot(fpr,fpr,'r--',color='red')
plt.legend()
```

```
Model Validation ==>

Accuracy Score of Logistic Regression Model::
0.8116883116883117

 Classification Report::
              precision    recall  f1-score   support

           0       0.82      0.93      0.87       107
           1       0.78      0.53      0.63        47

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



 ROC Curve
```

Out[308]: <matplotlib.legend.Legend at 0xa46fe01688>



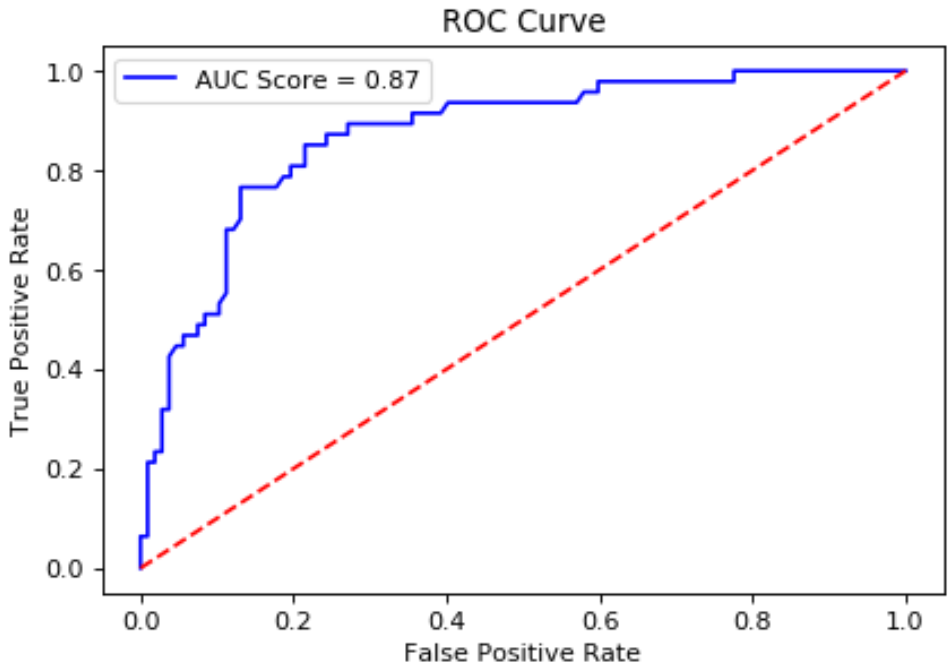
RandomForest

In [309]:
```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=1000,random_state=0)
rf_model.fit(x_train_std,y_train)
rf_pred=rf_model.predict(x_test_std)
```

In [310]:
```
print("Model Validation ==>\n")
print("Accuracy Score of Logistic Regression Model::")
print(metrics.accuracy_score(y_test,rf_pred))
print("\n","Classification Report::")
print(metrics.classification_report(y_test,rf_pred),'\n')
print("\n","ROC Curve")
rf_prob=rf_model.predict_proba(x_test_std)
rf_prob1=rf_prob[:,1]
fpr,tpr,thresh=metrics.roc_curve(y_test,rf_prob1)
roc_auc_rf=metrics.auc(fpr,tpr)
plt.figure(dpi=80)
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_rf)
plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot(fpr,fpr,'r--',color='red')
plt.legend()
```

```
Model Validation ==>

Accuracy Score of Logistic Regression Model::
0.8246753246753247

 Classification Report::
              precision    recall  f1-score   support

           0       0.88      0.87      0.87       107
           1       0.71      0.72      0.72        47

    accuracy                           0.82       154
   macro avg       0.79      0.80      0.79       154
weighted avg       0.83      0.82      0.83       154



 ROC Curve
```

Out[310]: <matplotlib.legend.Legend at 0xa471009508>



https://public.tableau.com/profile/jois.vishwesh#!/vizhome/HealthcareCapstoneVishwesh/Dashboard?publish=yes
(https://public.tableau.com/profile/jois.vishwesh#!/vizhome/HealthcareCapstoneVishwesh/Dashboard?publish=yes)