NAME: BATHALA VINOD KUMAR

PROJECT TITLE: HEALTHCARE

PHONE NUMBER: 9347766832

EMAIL ID: bathalavinodkumar419@gmil.com

DOMAIN: Data Science

Programming language used: Python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import style
import sklearn
```

```
data=pd.read_csv("/content/health care diabetes.csv")
```

```
data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigre
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	

```
data.isnull().any()
```

```
Pregnancies                 False
Glucose                     False
BloodPressure               False
SkinThickness               False
Insulin                     False
BMI                         False
DiabetesPedigreeFunction    False
Age                         False
Outcome                     False
dtype: bool
```

```
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
```

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

```
positive=data[data['Outcome']==1]
```

```
positive.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigre
0	6	148	72	35	0	33.6	
2	8	183	64	0	0	23.3	
4	0	137	40	35	168	43.1	
6	3	78	50	32	88	31.0	
8	2	197	70	45	543	30.5	

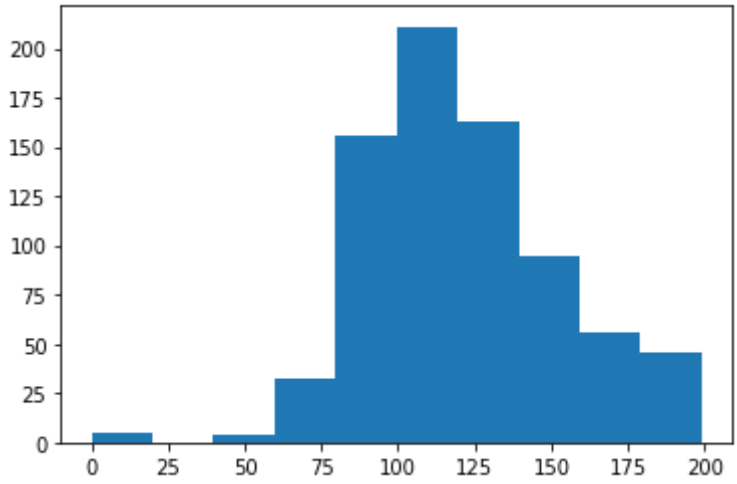
```
data['Glucose'].value_counts().head(7)
```

```
99     17
100    17
111    14
129    14
125    14
106    14
112    13
Name: Glucose, dtype: int64
```

Visually explore these variables using histograms

```
plt.hist(data['Glucose'])
```

```
(array([  5.,   0.,   4.,  32., 156., 211., 163.,  95.,  56.,  46.]),
 array([  0. ,  19.9,  39.8,  59.7,  79.6,  99.5, 119.4, 139.3, 159.2,
        179.1, 199. ]),
 <a list of 10 Patch objects>)
```

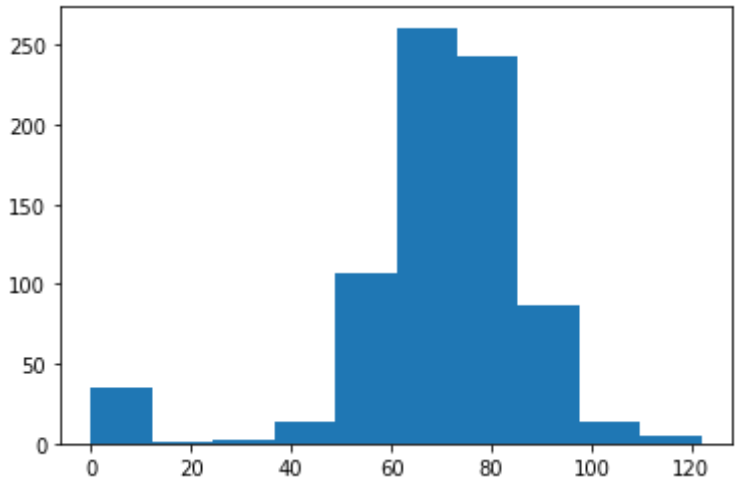



```
data['BloodPressure'].value_counts().head(7)
```

```
70    57
74    52
78    45
68    45
72    44
64    43
80    40
Name: BloodPressure, dtype: int64
```

```
plt.hist(data['BloodPressure'])
```

```
(array([ 35.,   1.,   2.,  13., 107., 261., 243.,  87.,  14.,   5.]),
 array([  0. ,  12.2,  24.4,  36.6,  48.8,  61. ,  73.2,  85.4,  97.6,
        109.8, 122. ]),
 <a list of 10 Patch objects>)
```

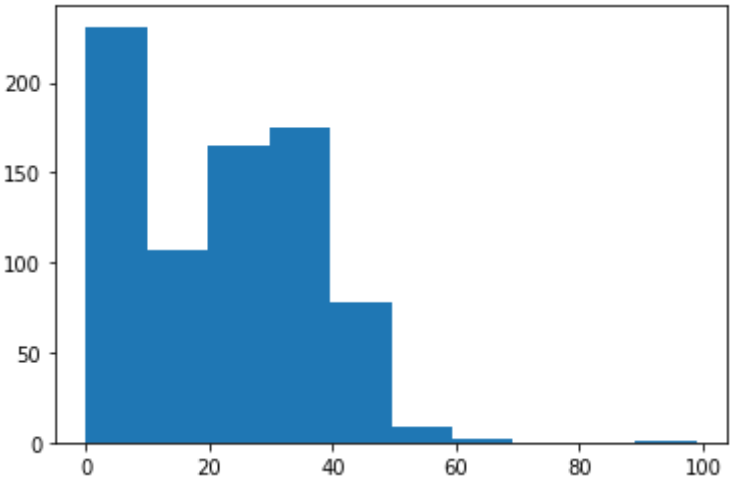



```
data['SkinThickness'].value_counts().head()
```

```
0     227
32     31
30     27
27     23
23     22
Name: SkinThickness, dtype: int64
```

```
plt.hist(data['SkinThickness'])
```

```
(array([231., 107., 165., 175.,  78.,   9.,   2.,   0.,   0.,   1.]),
 array([ 0. ,  9.9, 19.8, 29.7, 39.6, 49.5, 59.4, 69.3, 79.2, 89.1, 99. ]),
 <a list of 10 Patch objects>)
```




```
data['Insulin'].value_counts().head()
```

```
0      374
105     11
130      9
140      9
120      8
Name: Insulin, dtype: int64
```

```
plt.hist(data['Insulin'])
```

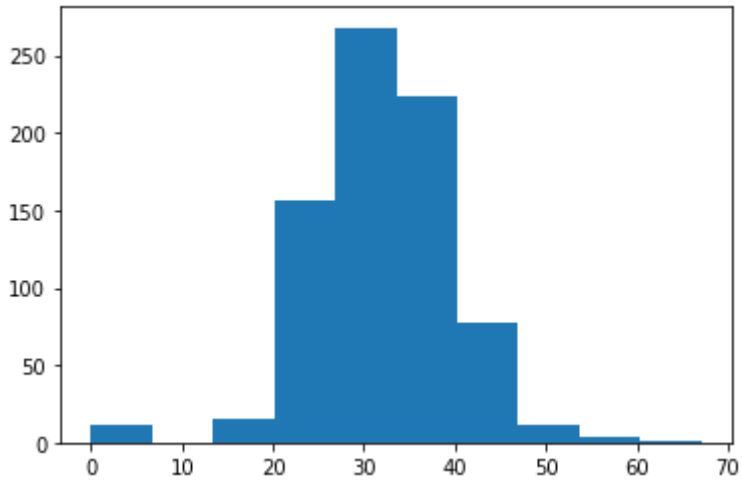
```
(array([487., 155.,  70.,  30.,   8.,   9.,   5.,   1.,   2.,   1.]),
 array([  0. ,  84.6, 169.2, 253.8, 338.4, 423. , 507.6, 592.2, 676.8,
        761.4, 846. ]),
```

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

```
32.0    13
31.6    12
31.2    12
0.0     11
32.4    10
33.3    10
30.1     9
Name: BMI, dtype: int64
```

```
plt.hist(data['BMI'])
```

```
(array([ 11.,   0.,  15., 156., 268., 224.,  78.,  12.,   3.,   1.]),
 array([ 0.  ,  6.71, 13.42, 20.13, 26.84, 33.55, 40.26, 46.97, 53.68,
        60.39, 67.1 ]),
 <a list of 10 Patch objects>)
```

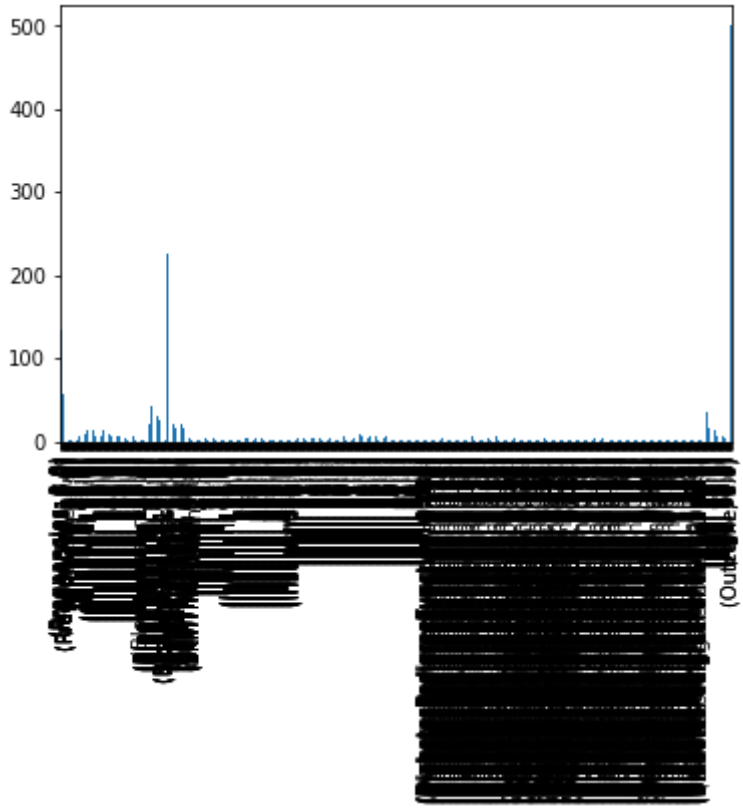



```
data.describe().transpose()
```

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.

```
data.apply(lambda x: x.value_counts()).T.stack().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e92a0d390>
```



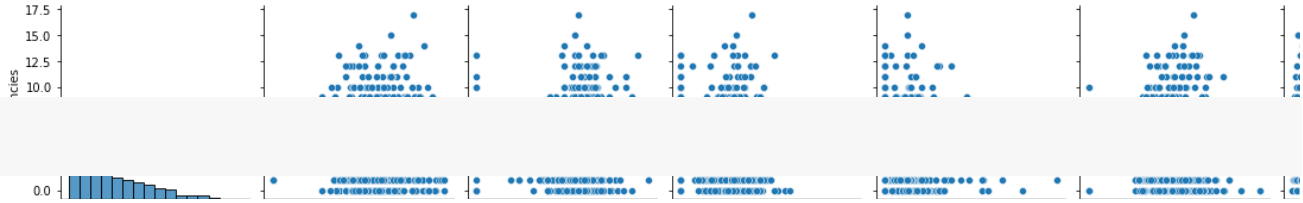

```
data.apply(pd.value_counts).T.stack().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e90cce790>
```

Creating scatter charts between the pair of variables to understand the relationships

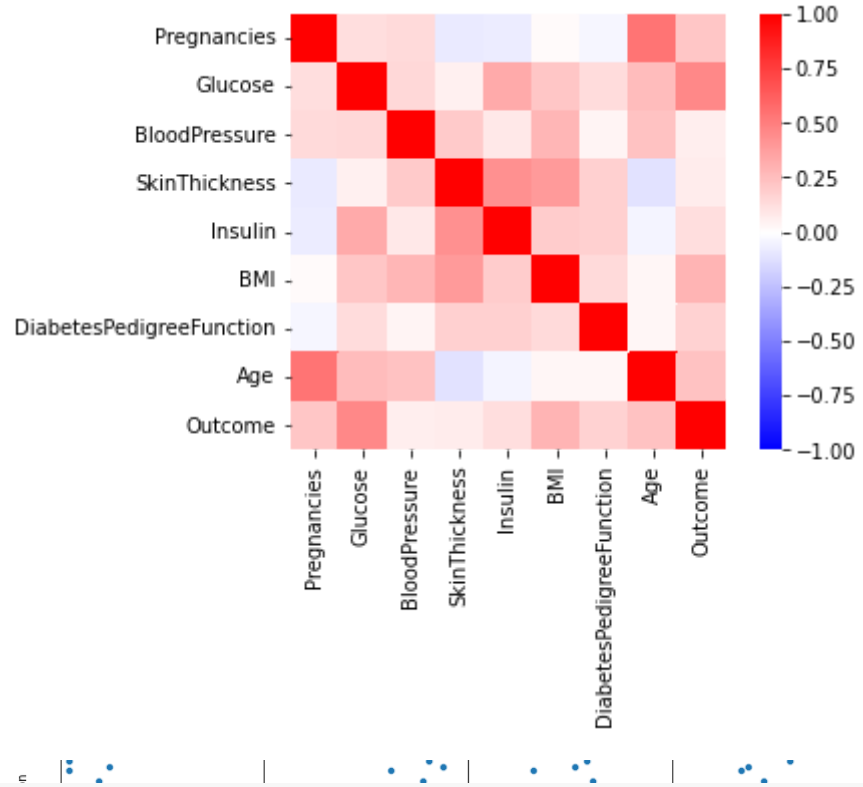
```
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x7f9e8f0d23d0>
```



Finding out correlation analysis. Visually exploring it using a heat map

```
corr=data.corr()
ax=sns.heatmap(corr,vmin=-1,vmax=1,cmap='bwr',square=True)
```




```
plt.hist(positive['BMI'],histtype='stepfilled',bins=20)
```

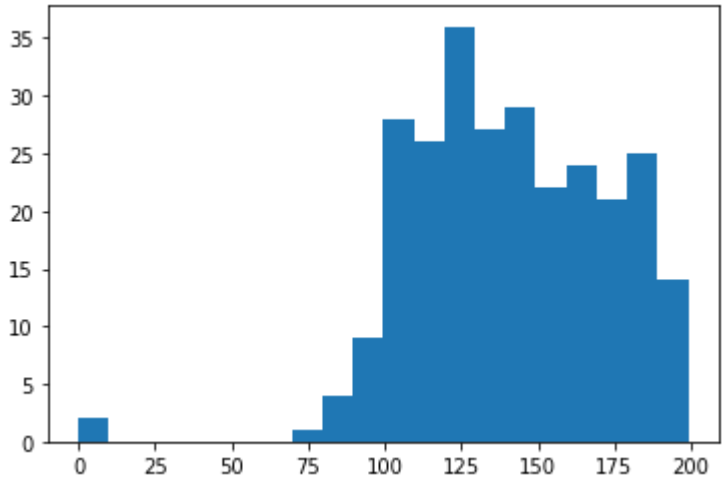
```
(array([ 2.,  0.,  0.,  0.,  0.,  0.,  3., 13., 38., 61., 61., 36., 27.,
        14.,  7.,  3.,  1.,  1.,  0.,  1.]),
```

```
positive['BMI'].value_counts().head(7)
```

```
32.9    8
31.6    7
33.3    6
31.2    5
30.5    5
32.0    5
34.3    4
Name: BMI, dtype: int64
```

```
plt.hist(positive['Glucose'],histtype='stepfilled',bins=20)
```

```
(array([ 2.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  4.,  9., 28., 26., 36.,
        27., 29., 22., 24., 21., 25., 14.]),
 array([  0.  ,   9.95,  19.9 ,  29.85,  39.8 ,  49.75,  59.7 ,  69.65,
         79.6 ,  89.55,  99.5 , 109.45, 119.4 , 129.35, 139.3 , 149.25,
        159.2 , 169.15, 179.1 , 189.05, 199.  ]),
 <a list of 1 Patch objects>)
```

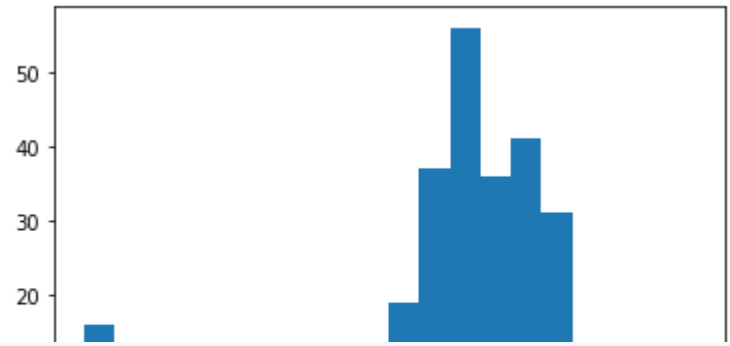



```
positive['Glucose'].value_counts().head(7)
```

```
125    7
128    6
129    6
115    6
158    6
146    5
124    5
Name: Glucose, dtype: int64
```

```
plt.hist(positive['BloodPressure'],histtype='stepfilled',bins=20)
```

```
(array([16.,  0.,  0.,  0.,  0.,  1.,  0.,  1.,  6.,  6., 19., 37., 56.,
        36., 41., 31.,  7.,  4.,  4.,  3.]),
 array([  0. ,   5.7,  11.4,  17.1,  22.8,  28.5,  34.2,  39.9,  45.6,
         51.3,  57. ,  62.7,  68.4,  74.1,  79.8,  85.5,  91.2,  96.9,
        102.6, 108.3, 114. ]),
 <a list of 1 Patch objects>)
```

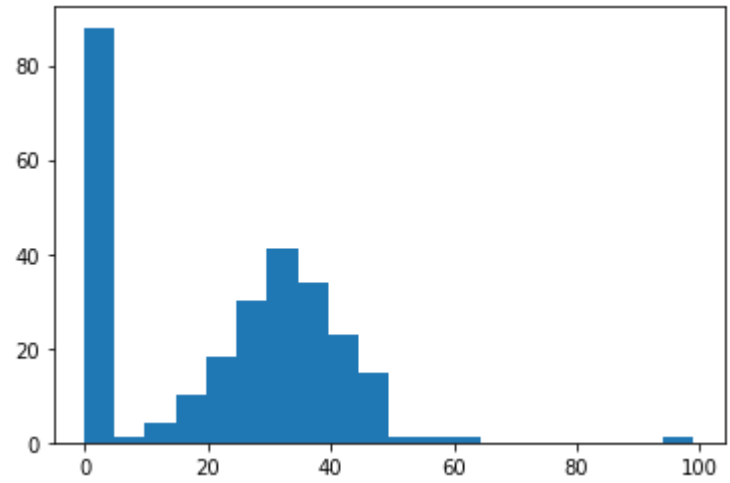



```
positive['BloodPressure'].value_counts().head(7)
```

```
70    23
76    18
78    17
74    17
72    16
0     16
80    13
Name: BloodPressure, dtype: int64
```

```
plt.hist(positive['SkinThickness'],histtype='stepfilled',bins=20)
```

```
(array([88.,  1.,  4., 10., 18., 30., 41., 34., 23., 15.,  1.,  1.,  1.,
         0.,  0.,  0.,  0.,  0.,  0.,  1.]),
 array([ 0.  ,  4.95,  9.9 , 14.85, 19.8 , 24.75, 29.7 , 34.65, 39.6 ,
        44.55, 49.5 , 54.45, 59.4 , 64.35, 69.3 , 74.25, 79.2 , 84.15,
        89.1 , 94.05, 99.  ]),
 <a list of 1 Patch objects>)
```



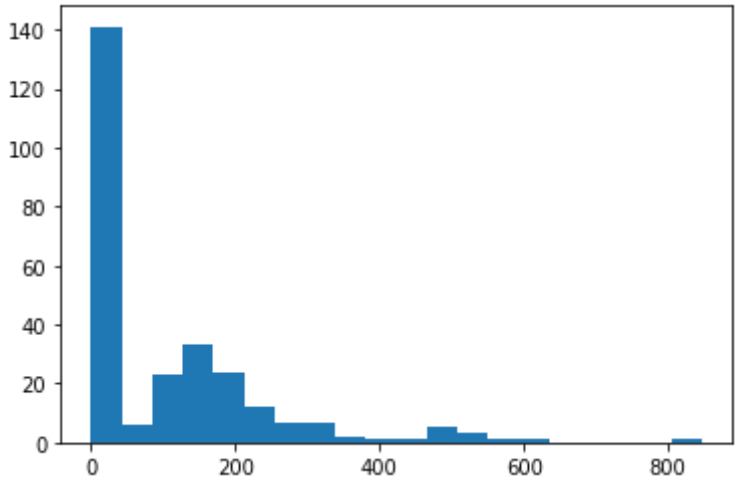

```
positive['SkinThickness'].value_counts().head()
```

```
0     88
32    14
30     9
```

```
33     9
39     8
Name: SkinThickness, dtype: int64
```

```
plt.hist(positive['Insulin'],histtype='stepfilled',bins=20)
```

```
(array([141.,   6.,  23.,  33.,  24.,  12.,   7.,   7.,   2.,   1.,   1.,
          5.,   3.,   1.,   1.,   0.,   0.,   0.,   0.,   1.]),
 array([  0. ,  42.3,  84.6, 126.9, 169.2, 211.5, 253.8, 296.1, 338.4,
        380.7, 423. , 465.3, 507.6, 549.9, 592.2, 634.5, 676.8, 719.1,
        761.4, 803.7, 846. ]),
 <a list of 1 Patch objects>)
```

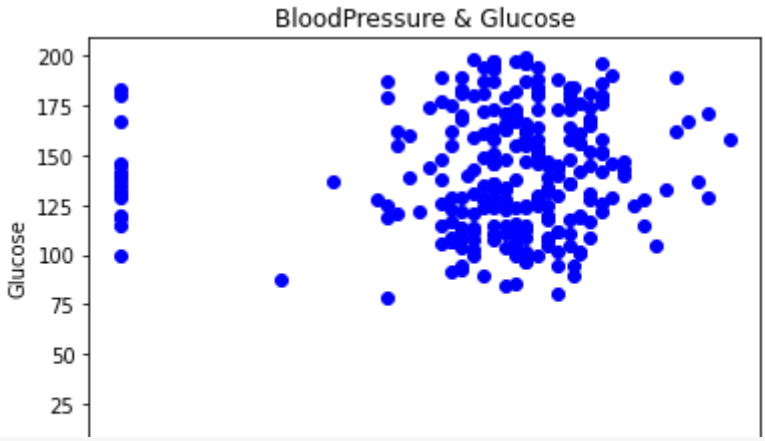



```
positive['Insulin'].value_counts().head()
```

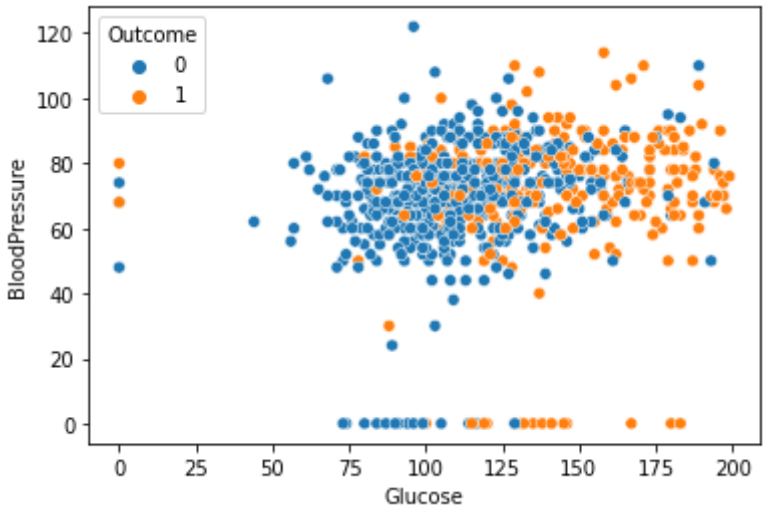
```
0      138
130      6
180      4
175      3
156      3
Name: Insulin, dtype: int64
```

```
BloodPressure=positive['BloodPressure']
Glucose=positive['Glucose']
SkinThickness=positive['SkinThickness']
Insulin = positive['Insulin']
BMI = positive['BMI']
```

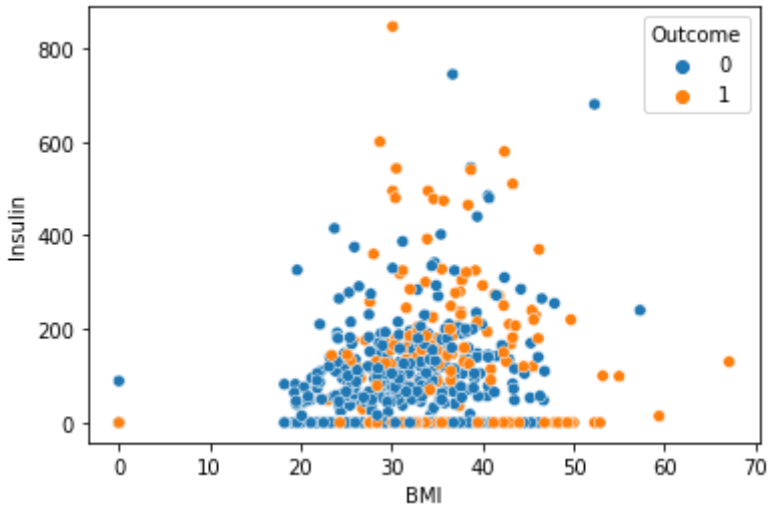
```
plt.scatter(BloodPressure,Glucose,color=['b'])
plt.xlabel('BloodPressure')
plt.ylabel('Glucose')
plt.title('BloodPressure & Glucose')
plt.show()
```



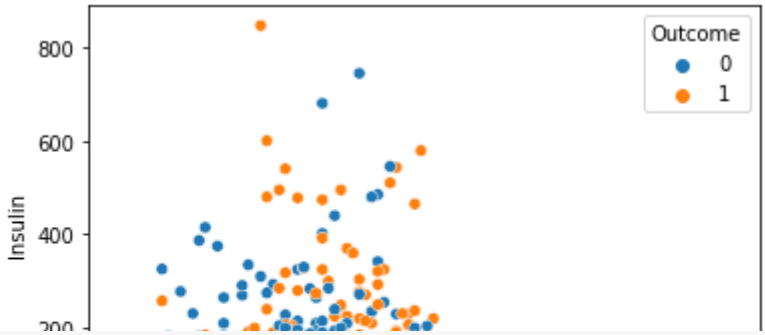

```
g=sns.scatterplot(x='Glucose',y='BloodPressure',hue='Outcome',data=data);
```




```
B= sns.scatterplot(x='BMI',y='Insulin',hue='Outcome',data=data)
```




```
S=sns.scatterplot(x='SkinThickness',y='Insulin',hue='Outcome',data=data)
```




```
data.corr()
```

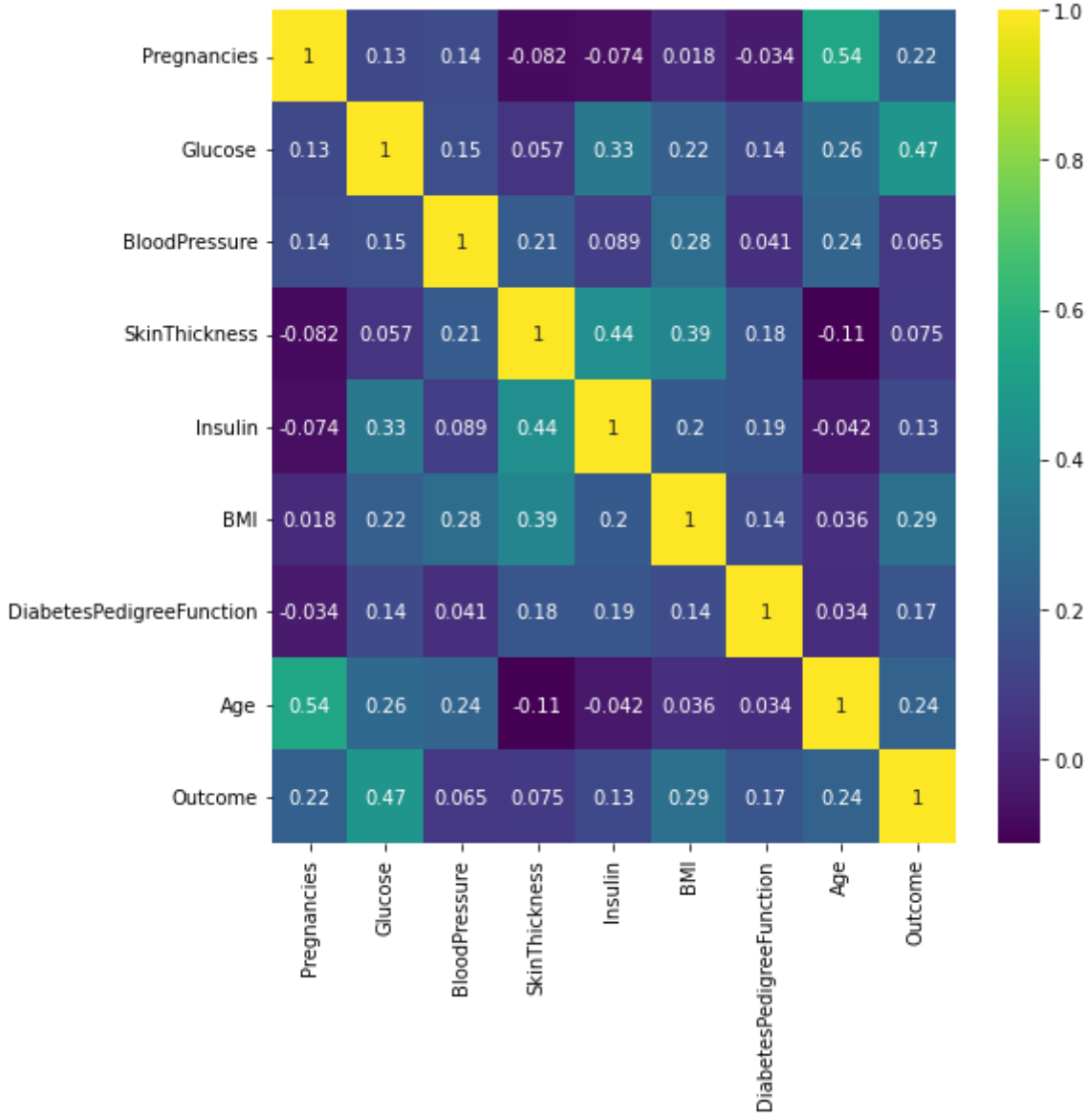
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insul
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.0735
Glucose	0.129459	1.000000	0.152590	0.057328	0.3313
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.0889
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.4367
Insulin	-0.073535	0.331357	0.088933	0.436783	1.0000
BMI	0.017683	0.221071	0.281805	0.392573	0.1978
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.1850
Age	0.544341	0.263514	0.239528	-0.113970	-0.0421
Outcome	0.221898	0.466581	0.065068	0.074752	0.1305

```
sns.heatmap(data.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e8859d3d0>
```

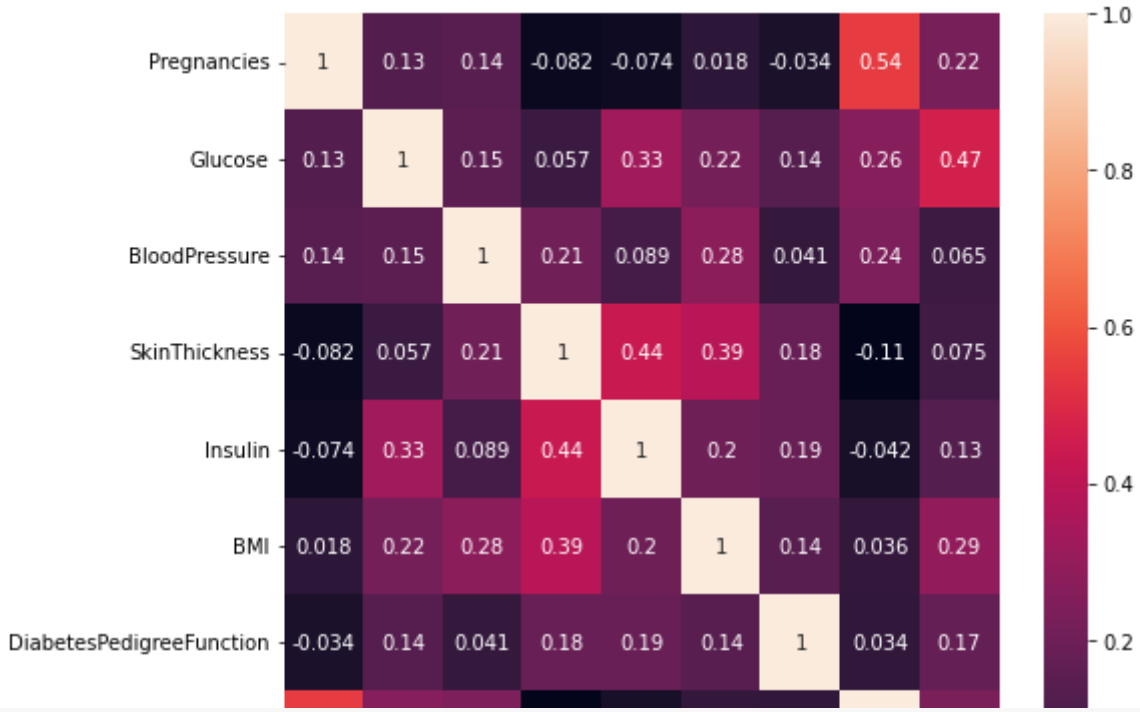
```
plt.subplots(figsize=(8,8))
sns.heatmap(data.corr(),annot=True,cmap='viridis')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e88552b90>
```




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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e88315510>
```




```
data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigre
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	

```
feature=data.iloc[:,[0,1,2,3,4,5,6,7]].values
label=data.iloc[:,8].values
```

```
##Train,Test,Split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(feature,label,test_size=0.2,random_state=12
```

```
#Create Model
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(X_train,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converg
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
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
LogisticRegression()
```

```
print(model.score(X_train,y_train))
print(model.score(X_test,y_test))
```

```
0.7850162866449512
0.7857142857142857
```

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(label,model.predict(feature))
cm
```

```
array([[439,  61],
       [104, 164]])
```

```
from sklearn.metrics import classification_report
print(classification_report(label,model.predict(feature)))
```

```
                  precision    recall  f1-score   support

               0       0.81      0.88      0.84       500
               1       0.73      0.61      0.67       268

        accuracy                           0.79       768
       macro avg       0.77      0.74      0.75       768
    weighted avg       0.78      0.79      0.78       768
```

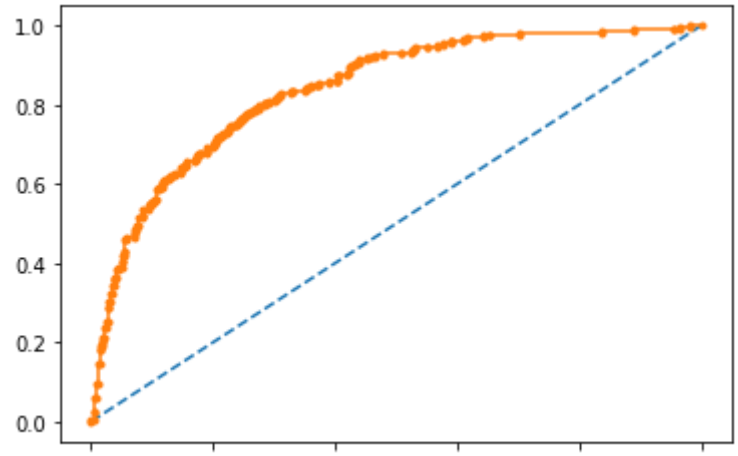
```
##Preparing ROC Curve(Receiver Operating Characteristics Curve)
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
```

```
##Predict Probabailities
probs=model.predict_proba(feature)
probs=probs[:,1]
auc=roc_auc_score(label,probs)
print('AUC %.3f' % auc)
```

```
AUC 0.839
```

```
fpr,tpr,thresholds=roc_curve(label,probs)
plt.plot([0,1], [0,1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
```

```
[<matplotlib.lines.Line2D at 0x7f9e87420350>]
```




```
##Applying Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
model3=DecisionTreeClassifier(max_depth=5)
model3.fit(X_train,y_train)
```

```
DecisionTreeClassifier(max_depth=5)
```

```
model3.score(X_train,y_train)
```

```
0.8371335504885994
```

```
model3.score(X_test,y_test)
```

```
0.7402597402597403
```

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

```
RandomForestClassifier(n_estimators=11)
```

```
model4.score(X_test,y_test)
```

```
0.7792207792207793
```

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

```
SVC(gamma='auto')
```

```
model5.score(X_test,y_test)
```

```
0.6883116883116883
```

```
##Applying KNN
from sklearn.neighbors import KNeighborsClassifier
model12=KNeighborsClassifier(n_neighbors=7,metric='minkowski',p=2)
model12.fit(X_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=7)
```

```
##Preparing ROC Curve
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
probs=model12.predict_proba(feature)
probs=probs[:,1]
auc=roc_auc_score(label,probs)
print('AUC: %.3f' % auc)
```

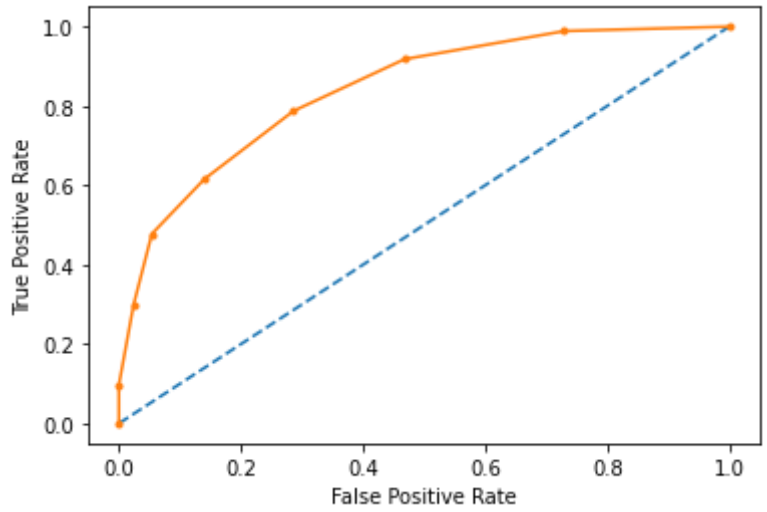
```
AUC: 0.839
```

```
fpr,tpr, thresholds=roc_curve(label,probs)
print("True Positive Rate - {}, False Positive Rate - {} Thresholds - {} ".format(tpr,fpr,
```

```
True Positive Rate - [0.         0.09701493 0.29850746 0.47761194 0.61567164 0.78731
 0.91791045 0.98880597 1.        ], False Positive Rate - [0.    0.    0.024 0.054 0
 0.28571429 0.14285714 0.        ]
```

```
plt.plot([0,1], [0,1], linestyle='--')
plt.plot(fpr,tpr,marker='.')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

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




```
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(feature)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model3.predict(feature)
# 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=0.758 auc=0.788 ap=0.777
[<matplotlib.lines.Line2D at 0x7f9e867094d0>]
```