Capstone2

July 19, 2023

Data Exploration:

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

Glucose

BloodPressure

SkinThickness

Insulin

BMI

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

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.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
print('All library imported')
```

```
All library imported
```

[2]:
```
#load the data
data=pd.read_csv('health care diabetes.csv')
print ('data loaded')
```

```
data loaded
```

[3]:
```
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]:
```
#shape of data
data.shape
```

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

[5]:
```
#missing values
data.isnull().sum()
```

[5]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

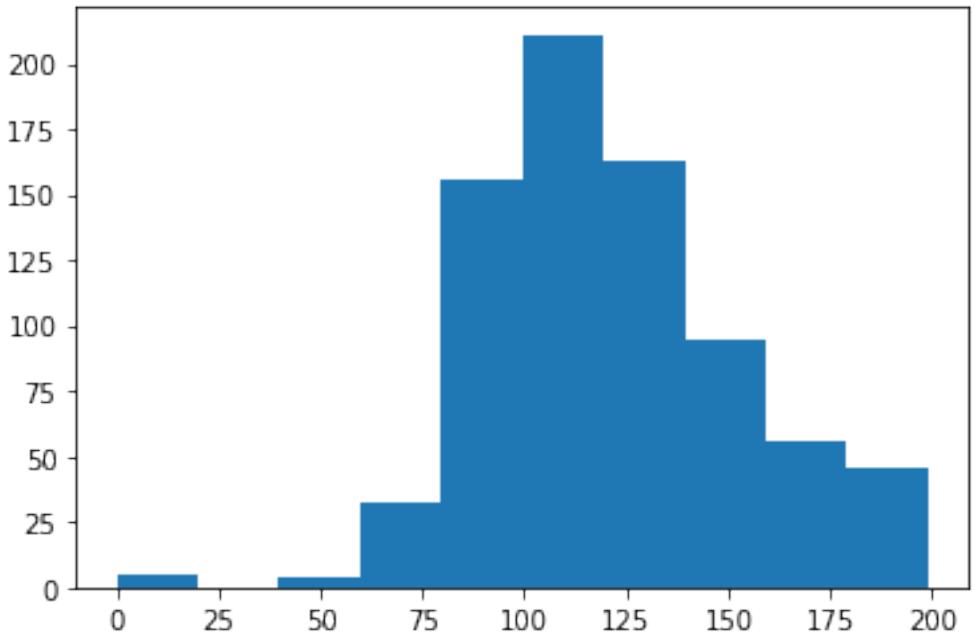
[6]:
```
data.info()
```

```
<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
```

[7]:
```
#know the target data
data['Outcome'].value_counts()
```

[7]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

[8]:
```
#create histogram distribution of the data
plt.hist(data['Glucose'])
plt.show()
```



[9]:
```
data[(data['Glucose']==0)].shape
```

[9]:
```
(5, 9)
```

[10]:
```
data['Glucose'].mean()
```

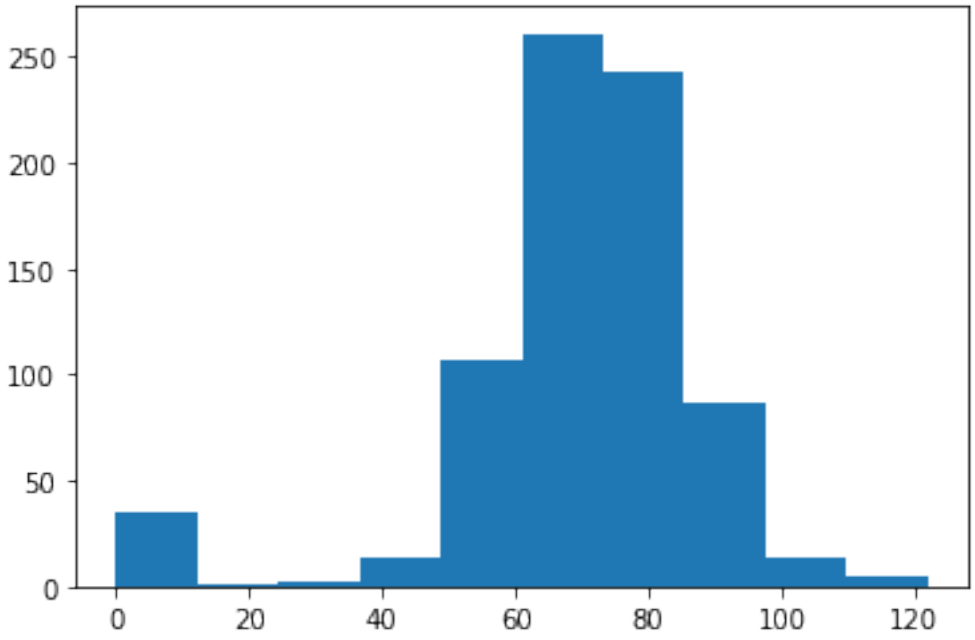
[10]:
```
120.89453125
```

[11]:
```
#fill these zeros
data.loc[data['Glucose']==0,'Glucose']=120.8945312
```

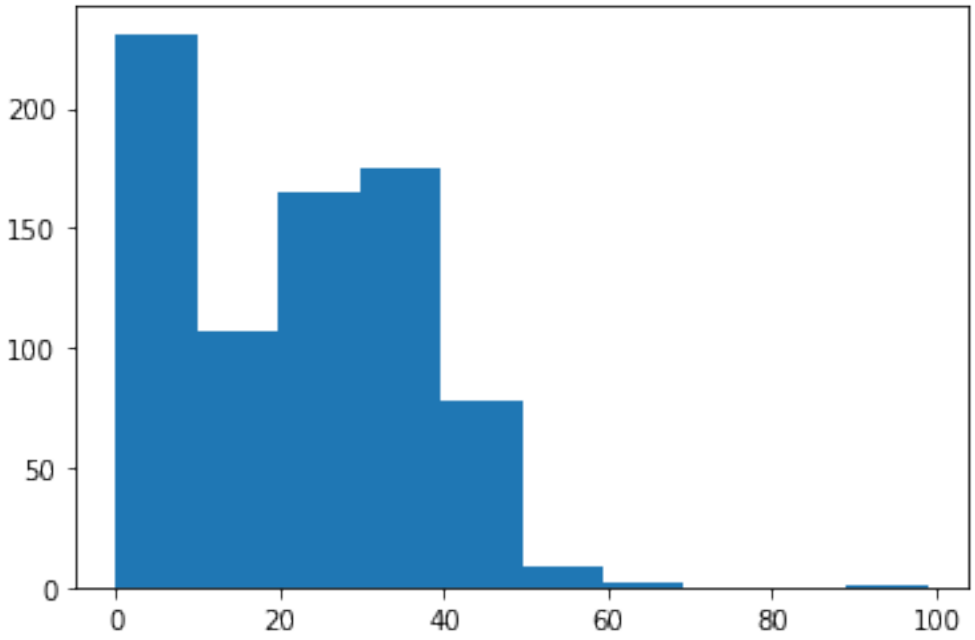
[12]:
```
data[(data['Glucose']==0)].shape
```

[12]:
```
(0, 9)
```

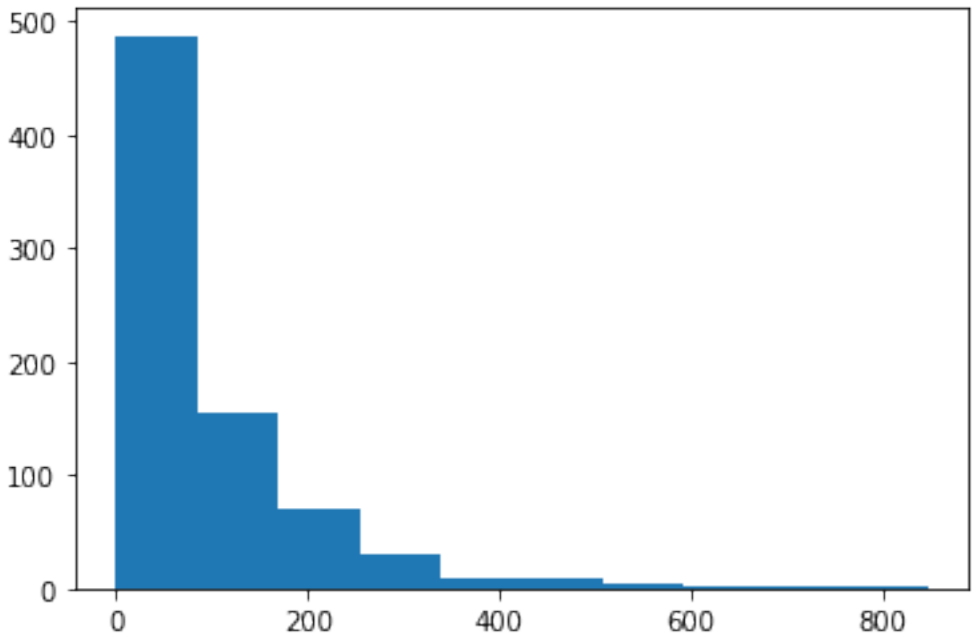
[13]:
```
#create histogram distribution of the data
plt.hist(data['BloodPressure'])
plt.show()
```



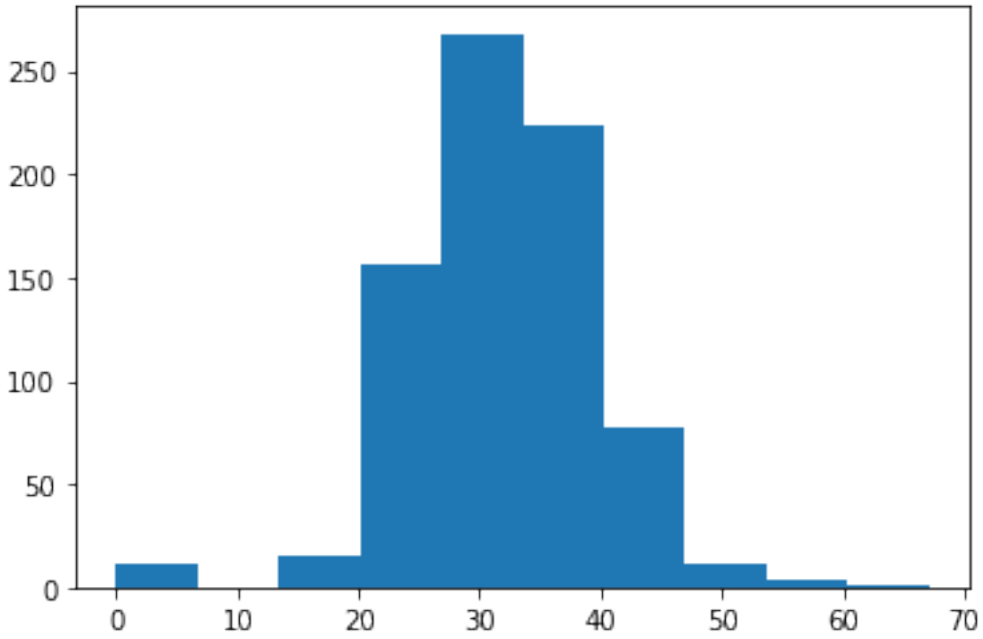
[14]:
```
#create histogram distribution of the data
plt.hist(data['SkinThickness'])
plt.show()
```



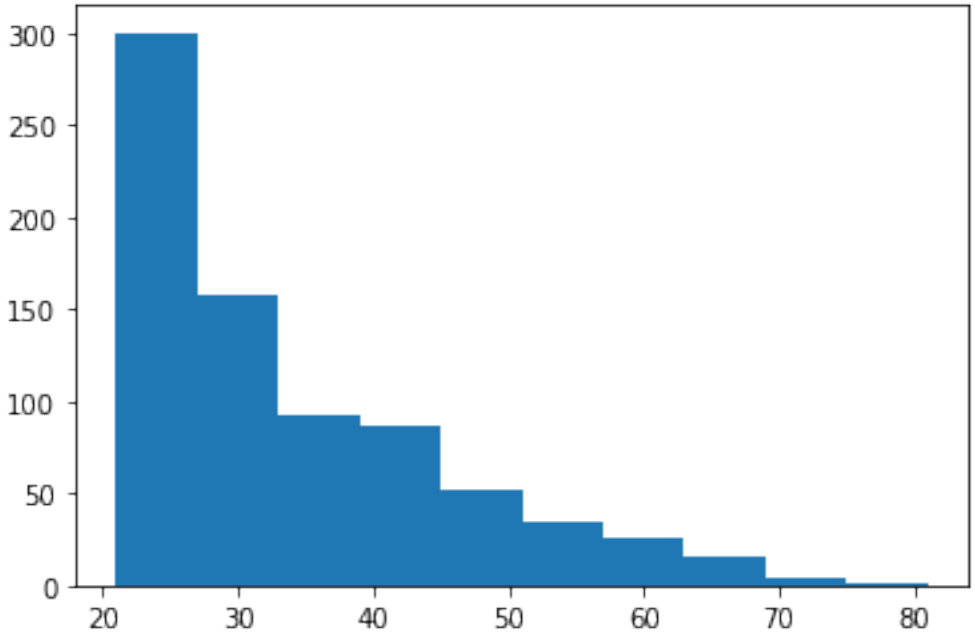
[15]:
```
# data is not normally distribute. right skeweness right side long tail.
plt.hist(data['Insulin'])
plt.show()
```



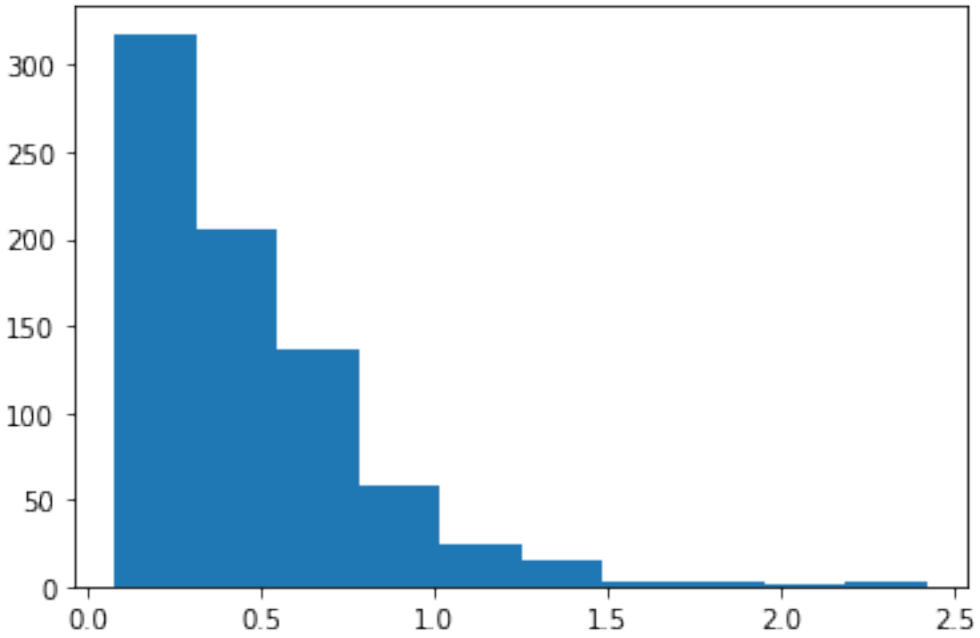
[16]:
```
plt.hist(data['BMI'])
plt.show()
```



[17]:
```
plt.hist(data['Age'])
plt.show()
```



[18]:
```
plt.hist(data['DiabetesPedigreeFunction'])
plt.show()
```



[19]:
```
data.describe().T # T transpose
```

[19]:

	count	mean	std	min	25%	\
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	
Glucose	768.0	121.681605	30.436016	44.000	99.75000	
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	
Insulin	768.0	79.799479	115.244002	0.000	0.00000	
BMI	768.0	31.992578	7.884160	0.000	27.30000	
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	
Age	768.0	33.240885	11.760232	21.000	24.00000	
Outcome	768.0	0.348958	0.476951	0.000	0.00000	

	50%	75%	max
Pregnancies	3.0000	6.00000	17.00
Glucose	117.0000	140.25000	199.00
BloodPressure	72.0000	80.00000	122.00
SkinThickness	23.0000	32.00000	99.00
Insulin	30.5000	127.25000	846.00
BMI	32.0000	36.60000	67.10
DiabetesPedigreeFunction	0.3725	0.62625	2.42
Age	29.0000	41.00000	81.00
Outcome	0.0000	1.00000	1.00

[20]:
```
#data exploration
variables = ['Glucose','BloodPressure','SkinThickness','Insulin','BMI']
for i in variables:
    data[i].replace('0',np.nan)
    data[i].fillna(data[i].median(), inplace=True)
```

[21]:
```
data.head()
```

[21]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148.0	72	35	0	33.6	
1	1	85.0	66	29	0	26.6	
2	8	183.0	64	0	0	23.3	
3	1	89.0	66	23	94	28.1	
4	0	137.0	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

[22]:
```
variables = ['Glucose','BloodPressure','SkinThickness','Insulin','BMI']
for i in variables:
    #data[i].replace(0,np.nan)
    data[i].replace(0,data[i].median(), inplace=True)
```

[23]:
```
data.head()
```

[23]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148.0	72	35	30.5	33.6	
1	1	85.0	66	29	30.5	26.6	
2	8	183.0	64	23	30.5	23.3	
3	1	89.0	66	23	94.0	28.1	
4	0	137.0	40	35	168.0	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

1 plotting count

2 create satter chart

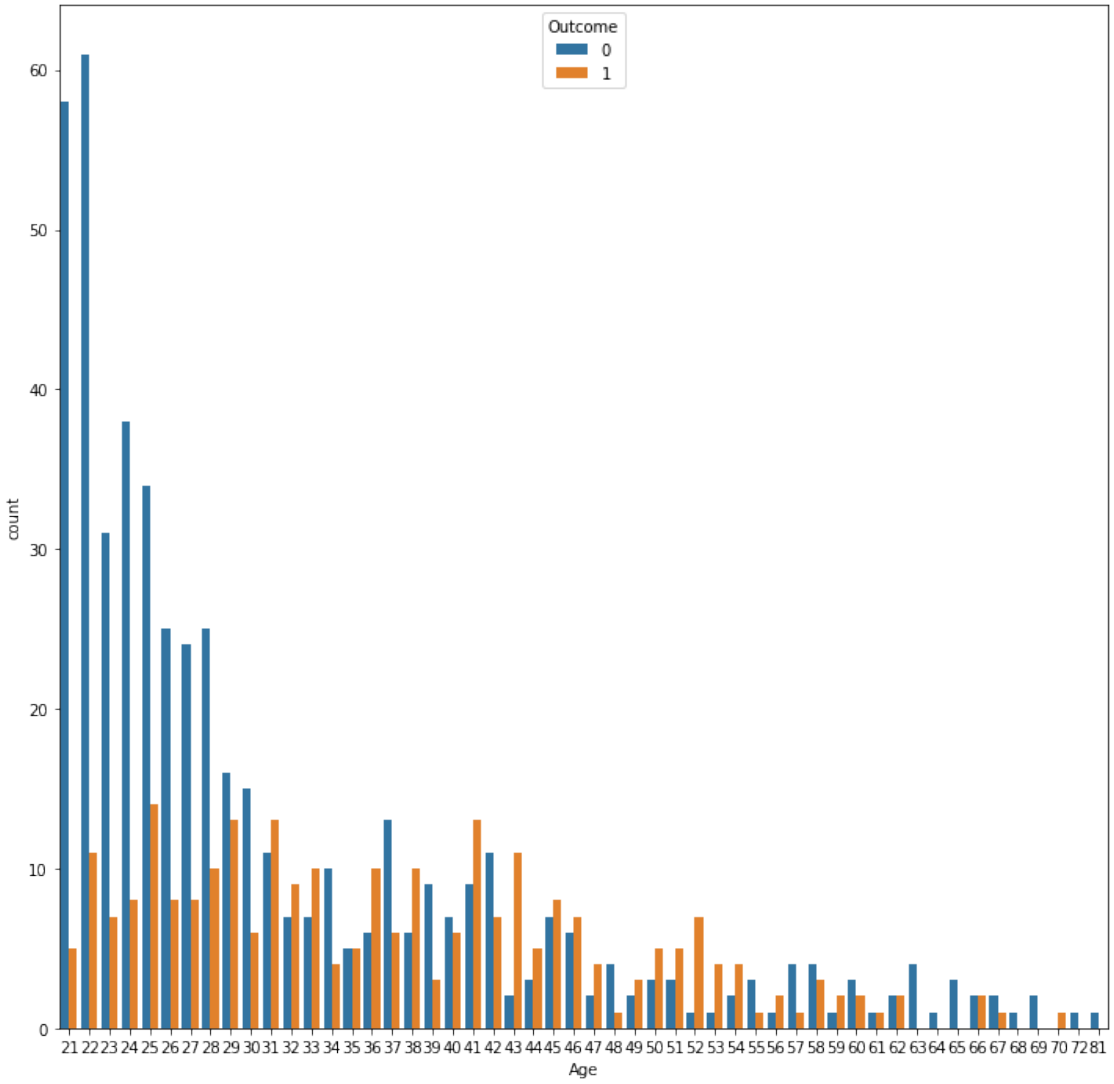
3 perform correlation analysis

[24]:
```
data.columns
```

[24]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

[25]:
```
plt.figure(figsize=(12,12))
sns.countplot(x=data['Age'],hue='Outcome',data=data)
```

[25]:
```
<AxesSubplot:xlabel='Age', ylabel='count'>
```



[26]:
```
dib_person=data[data['Outcome']==1]
```

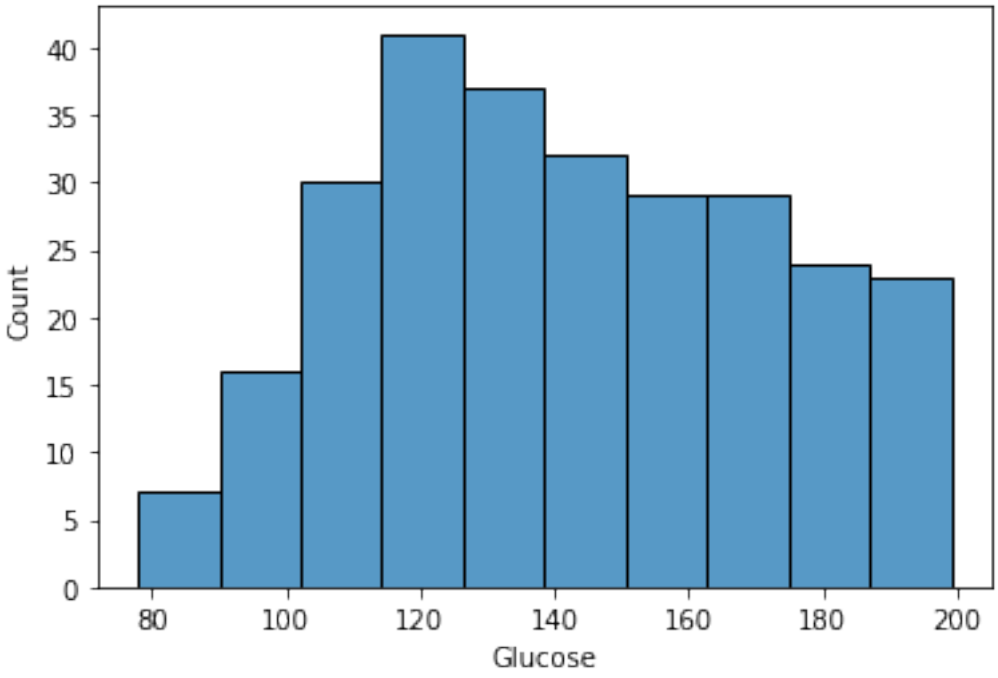
[27]:
```
dib_person
```

[27]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148.0	72	35	30.5	33.6	
2	8	183.0	64	23	30.5	23.3	
4	0	137.0	40	35	168.0	43.1	
6	3	78.0	50	32	88.0	31.0	
8	2	197.0	70	45	543.0	30.5	
..	…	…	…	…	…	…	
755	1	128.0	88	39	110.0	36.5	
757	0	123.0	72	23	30.5	36.3	
759	6	190.0	92	23	30.5	35.5	

```
761            9    170.0             74             31     30.5  44.0
766            1    126.0             60             23     30.5  30.1

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
2                       0.672   32        1
4                       2.288   33        1
6                       0.248   26        1
8                       0.158   53        1
..                        ...  ...      ...
755                     1.057   37        1
757                     0.258   52        1
759                     0.278   66        1
761                     0.403   43        1
766                     0.349   47        1

[268 rows x 9 columns]
```

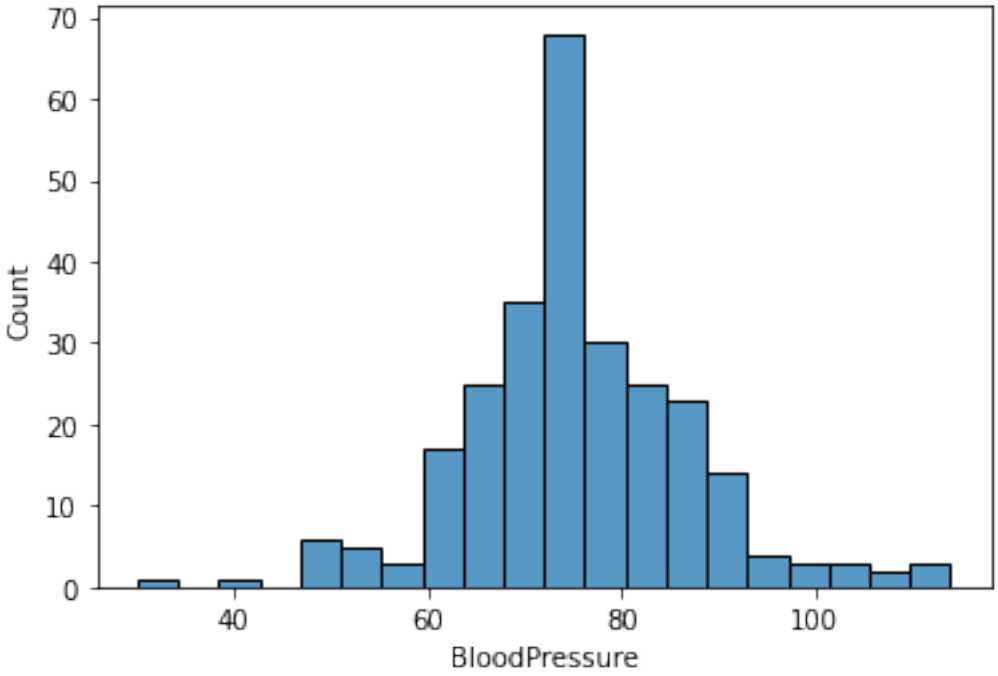
[28]:
```
sns.histplot(x=dib_person['Glucose'])
plt.show()
```



[29]:
```
dib_person['Glucose'].value_counts().head(10)
```

[29]:
```
125.0    7
128.0    6
129.0    6
158.0    6
115.0    6
181.0    5
173.0    5
162.0    5
124.0    5
146.0    5
Name: Glucose, dtype: int64
```

[30]:
```
sns.histplot(x=dib_person['BloodPressure'])
plt.show()
```



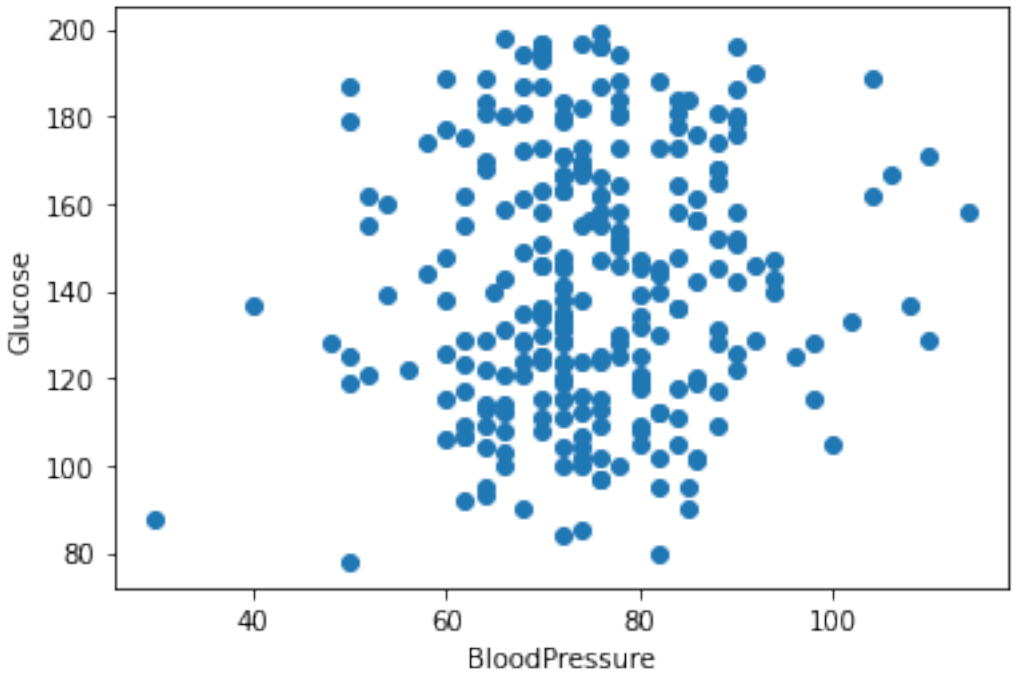
[31]:
```
dib_person['BloodPressure'].value_counts().head(10)
```

[31]:
```
72    32
70    23
76    18
74    17
78    17
64    13
80    13
82    13
```

```
84    12
68    12
Name: BloodPressure, dtype: int64
```

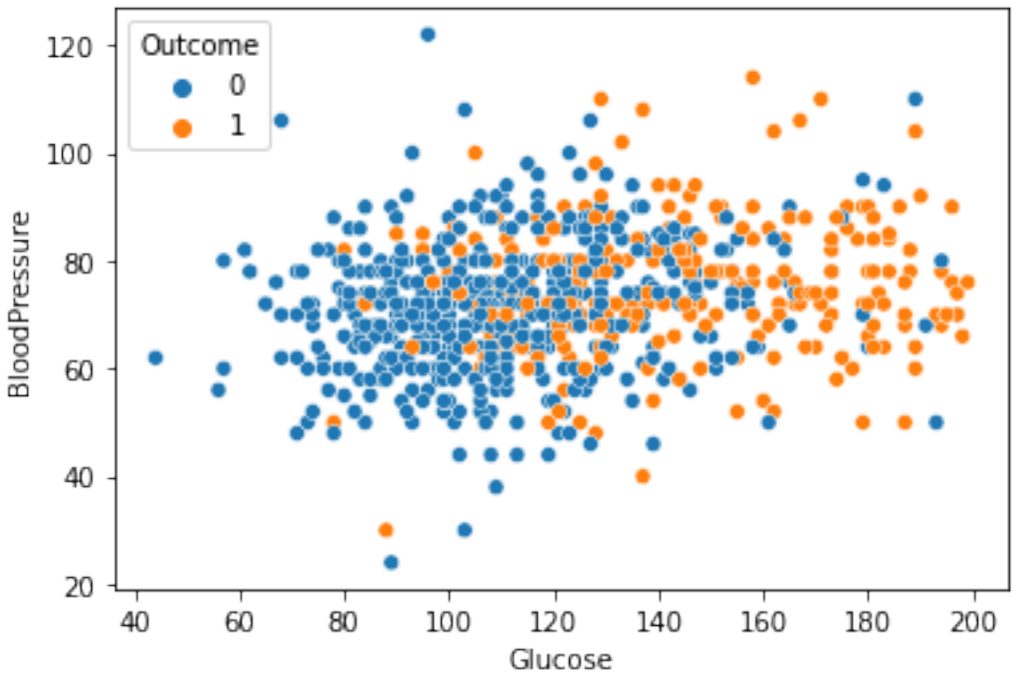
[32]:
```
#scatter plot to find data who have diabetic
plt.scatter(x=dib_person['BloodPressure'],y=dib_person['Glucose'])
plt.xlabel('BloodPressure')
plt.ylabel('Glucose')
```

[32]:
```
Text(0, 0.5, 'Glucose')
```



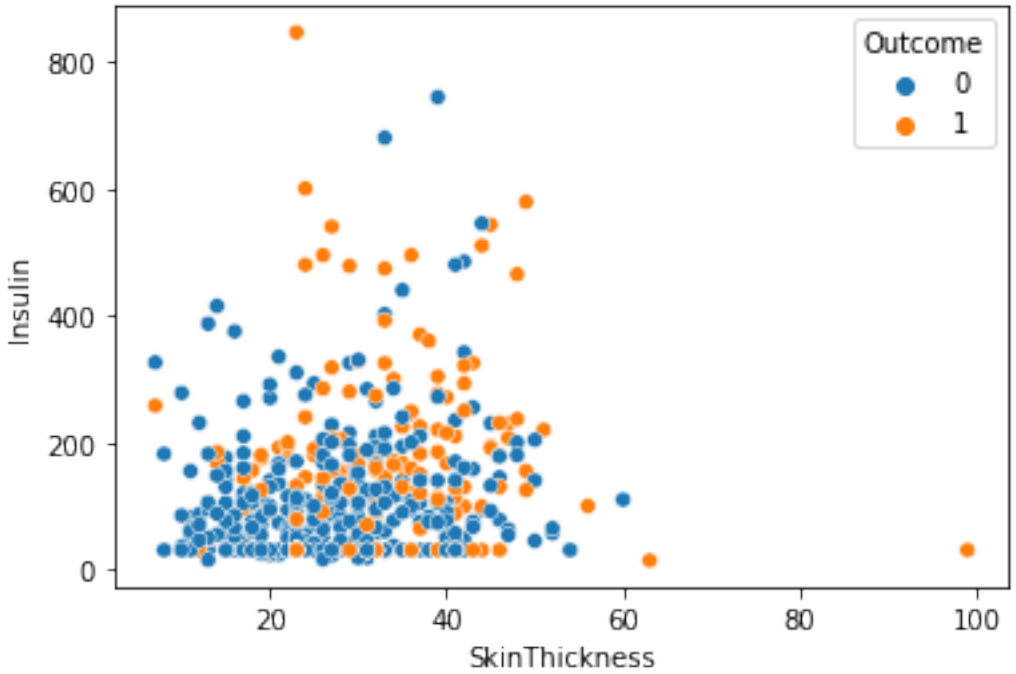
[33]:
```
sns.scatterplot(x='Glucose',y='BloodPressure',hue='Outcome',data=data)
```

[33]:
```
<AxesSubplot:xlabel='Glucose', ylabel='BloodPressure'>
```



[34]:
```
sns.scatterplot(x='SkinThickness',y='Insulin',hue='Outcome',data=data)
```

[34]: <AxesSubplot:xlabel='SkinThickness', ylabel='Insulin'>



[35]:
```
#correlation analysis.Visual
data.corr()
```

[35]:

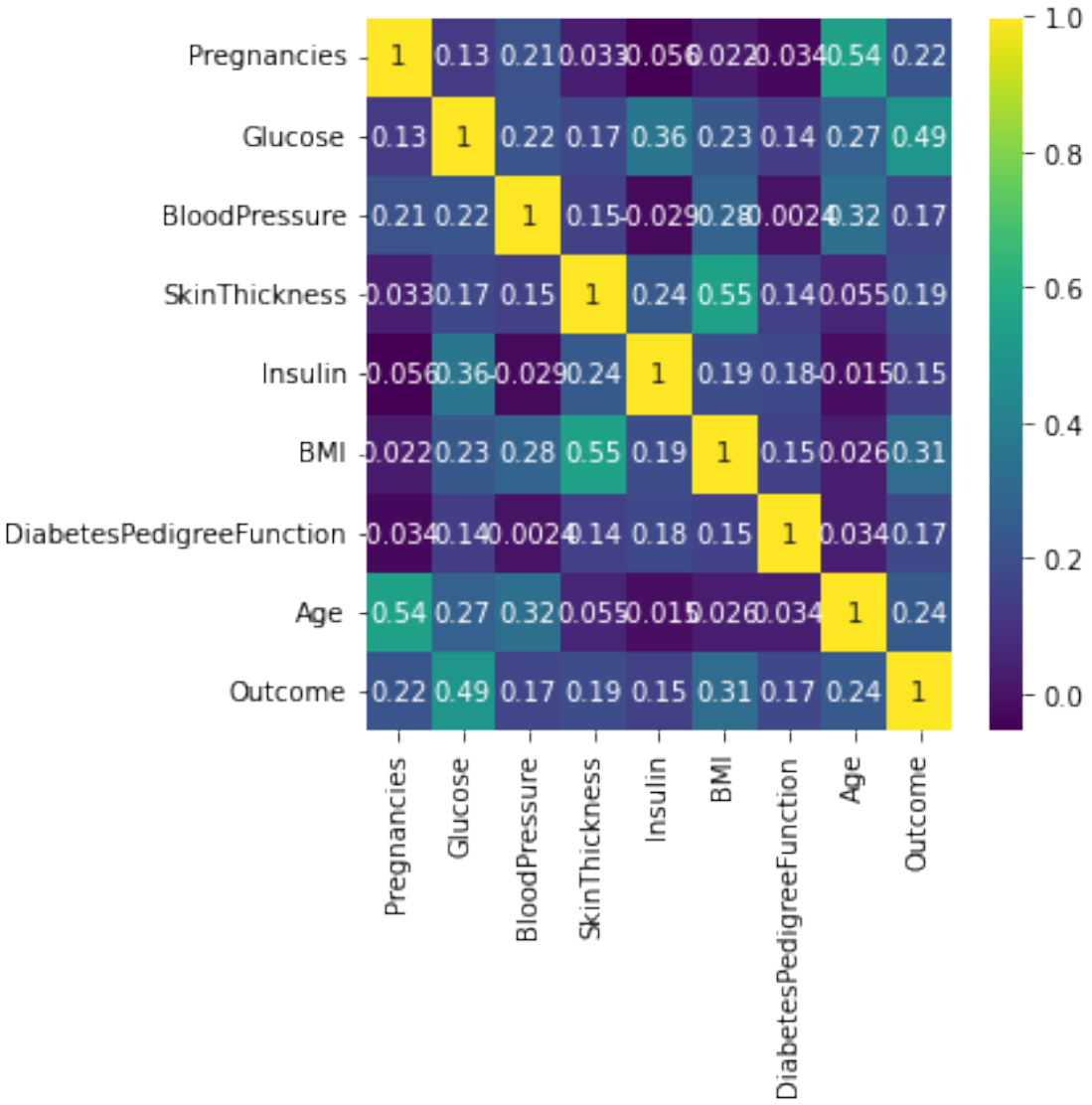
	Pregnancies	Glucose	BloodPressure	SkinThickness	\
Pregnancies	1.000000	0.127964	0.208615	0.032568	
Glucose	0.127964	1.000000	0.218623	0.172361	
BloodPressure	0.208615	0.218623	1.000000	0.147809	
SkinThickness	0.032568	0.172361	0.147809	1.000000	
Insulin	-0.055697	0.357081	-0.028721	0.238188	
BMI	0.021546	0.231469	0.281132	0.546951	
DiabetesPedigreeFunction	-0.033523	0.137106	-0.002378	0.142977	
Age	0.544341	0.266600	0.324915	0.054514	
Outcome	0.221898	0.492908	0.165723	0.189065	

	Insulin	BMI	DiabetesPedigreeFunction	\
Pregnancies	-0.055697	0.021546	-0.033523	
Glucose	0.357081	0.231469	0.137106	
BloodPressure	-0.028721	0.281132	-0.002378	
SkinThickness	0.238188	0.546951	0.142977	
Insulin	1.000000	0.189022	0.178029	
BMI	0.189022	1.000000	0.153506	
DiabetesPedigreeFunction	0.178029	0.153506	1.000000	
Age	-0.015413	0.025744	0.033561	
Outcome	0.148457	0.312249	0.173844	

	Age	Outcome
Pregnancies	0.544341	0.221898
Glucose	0.266600	0.492908
BloodPressure	0.324915	0.165723
SkinThickness	0.054514	0.189065
Insulin	-0.015413	0.148457
BMI	0.025744	0.312249
DiabetesPedigreeFunction	0.033561	0.173844
Age	1.000000	0.238356
Outcome	0.238356	1.000000

[68]:
```
plt.figure(figsize=(5,5))
sns.heatmap(data.corr(),annot=True,cmap='viridis')
```

[68]: <AxesSubplot:>



4 DATA MODELLING

[37]:
```
#DATA preporcocessing
X=data.iloc[:,:-1].values
```

[38]:
```
X
```

[38]:
```
array([[  6.   , 148.   ,  72.   , …,  33.6  ,   0.627,  50.   ],
       [  1.   ,  85.   ,  66.   , …,  26.6  ,   0.351,  31.   ],
       [  8.   , 183.   ,  64.   , …,  23.3  ,   0.672,  32.   ],
       …,
```

```
       [  5.   , 121.   ,  72.   , …,  26.2  ,   0.245,  30.   ],
       [  1.   , 126.   ,  60.   , …,  30.1  ,   0.349,  47.   ],
       [  1.   ,  93.   ,  70.   , …,  30.4  ,   0.315,  23.   ]])
```

[39]:
```
y=data.iloc[:,-1].values
```

[40]:
```
y
```

[40]:
```
array([1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0,
       1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0])
```

[41]:
```
#train set test tst split
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
↪2,random_state=10)
```

[42]:
```
X_train.shape
```

[42]: (614, 8)

[43]:
```
X_test.shape
```

[43]: (154, 8)

[44]:
```
import warnings
warnings.filterwarnings('ignore')
```

5 1.Logistic regression

[45]:
```
from sklearn.linear_model import LogisticRegression #LR is Class so we create␣
↪object
model1=LogisticRegression()
```

[46]:
```
#training
model1.fit(X_train,y_train)#training fit an be used
```

[46]: LogisticRegression()

[47]:
```
y_pred1=model1.predict(X_test)
```

[48]:
```
#train score & test score
print('Train score',model1.score(X_train,y_train))
print('Test score',model1.score(X_test,y_test))
```

```
Train score 0.7752442996742671
Test score 0.7597402597402597
```

[49]:
```
from sklearn.metrics import confusion_matrix,classification_report
```

[52]:
```
print(confusion_matrix(y_test,y_pred1))
# 0      1 (output)
#0  TN    FP
#1    FN    TP
# recall formula=TP/TP+FN
```

```
[[87  8]
 [29 30]]
```

[53]:
```
print(classification_report(y_test,y_pred1))
```

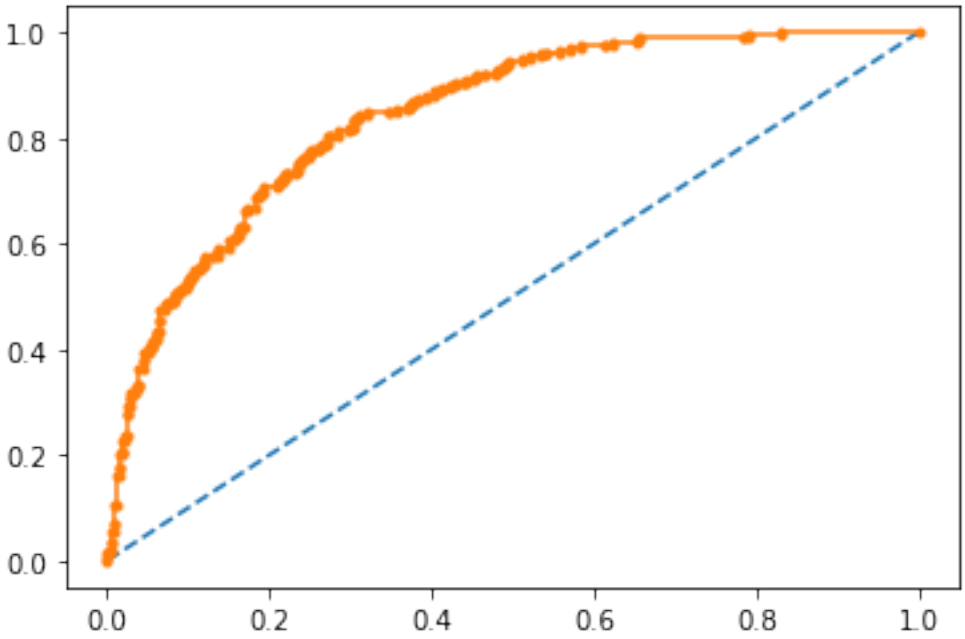
```
              precision    recall  f1-score   support

           0       0.75      0.92      0.82        95
           1       0.79      0.51      0.62        59

    accuracy                           0.76       154
   macro avg       0.77      0.71      0.72       154
weighted avg       0.77      0.76      0.75       154
```

[58]:
```
#Prepare ROC curve
from sklearn.metrics import roc_auc_score,roc_curve
prob=model1.predict_proba(X)
#prob
#select prob for the psitive outcome onnly
prob=prob[:,1]
#calculate area under the curve
auc=roc_auc_score(y,prob)
print('AUC score:',auc)
```

```
AUC score: 0.843417910447761
```

[61]:
```
#claculat roc curve
fpr,tpr,thresholds=roc_curve(y,prob)
#plot
plt.plot([0,1],[0,1],linestyle='--')
plt.plot(fpr,tpr,marker='.')
```

[61]:
```
[<matplotlib.lines.Line2D at 0x7f808516b510>]
```



[62]:
```
import joblib
joblib.dump(model1,'Logistic.pkl')
print('model1saved')
```

```
model1saved
```

[64]:
```
#Load the model
Pred_model=joblib.load('Logistic.pkl')
print('model loaded')
```

```
model loaded
```

[65]:
```
data.columns
```

[65]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

[66]:
```
Pregnancies=2
Glucose=148
BloodPressure=72
SkinThickness=40
Insulin=100
BMI=25.5
DiabetesPedigreeFunction=0.35
Age=35
```

```
output=Pred_model.predict([[Pregnancies, Glucose, BloodPressure, SkinThickness,␣
 ↪Insulin,
       BMI, DiabetesPedigreeFunction, Age]])
print('Person has',output)
```

```
Person has [0]
```

[67]:
```
#another file you predict the model
import joblib
Pred_model=joblib.load('Logistic.pkl')
print('model loaded')
```

```
model loaded
```

2.Decision Tree 3.Random forest 4.KNN 5,SVM

[ ]:
```

```