```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import seaborn as sns
# Standard Scaler
from sklearn.preprocessing import StandardScaler
# Robust Scaler
from sklearn.preprocessing import RobustScaler
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
import os
```

```
os.getcwd()
```

```
'/content'
```

```
dia=pd.read_csv('/content/health care diabetes.csv')
dia.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	

```
dia.shape
```

```
(768, 9)
```

```
dia.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
```

```
dia.isnull().sum()
```

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

1. Perform Descriptive Analysis

```
dia.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000

```
# Diabetic patient
Positive=dia[dia['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	

```
Positive.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI
count	268.000000	268.000000	268.000000	268.000000	268.000000	268.000000
mean	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537
std	3.741239	31.939622	21.491812	17.679711	138.689125	7.262967
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.750000	119.000000	66.000000	0.000000	0.000000	30.800000
50%	4.000000	140.000000	74.000000	27.000000	0.000000	34.250000
75%	8.000000	167.000000	82.000000	36.000000	167.250000	38.775000
max	17.000000	199.000000	114.000000	99.000000	846.000000	67.100000

```
# Non-diabetic patient
Negative=dia[dia['Outcome']==0]
Negative.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigre
1	1	85	66	29	0	26.6	
3	1	89	66	23	94	28.1	
5	5	116	74	0	0	25.6	
7	10	115	0	0	0	35.3	
10	4	110	92	0	0	37.6	

```
Negative.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Di
count	500.000000	500.0000	500.000000	500.000000	500.000000	500.000000	
mean	3.298000	109.9800	68.184000	19.664000	68.792000	30.304200	
std	3.017185	26.1412	18.063075	14.889947	98.865289	7.689855	
min	0.000000	0.0000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	93.0000	62.000000	0.000000	0.000000	25.400000	
50%	2.000000	107.0000	70.000000	21.000000	39.000000	30.050000	
75%	5.000000	125.0000	78.000000	31.000000	105.000000	35.300000	
max	13.000000	197.0000	122.000000	60.000000	744.000000	57.300000	

Non-diabetic dataset description also shows presence of outliers. Pregnancies has mean 3.298000, 75 percentile is 5 and max is 13. This shows outliers. Similarly, Insulin has mean 68.792000, median 39,75 percentile is 105 and max is 744. This shows outliers.

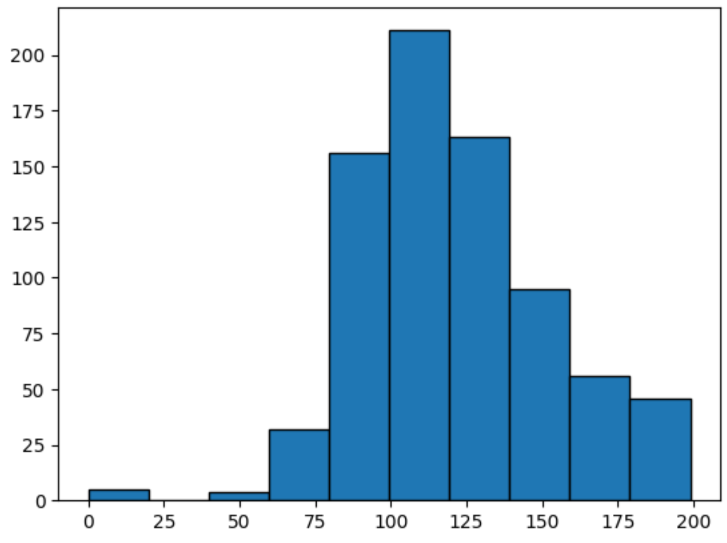
2. Visually explore these variables using histograms:

   Glucose, BloodPressure, Skin Thickness, Insulin, BMI

```
dia['Glucose'].value_counts()
```

```
    99     17
    100    17
    111    14
    129    14
    125    14
           ..
    191     1
    177     1
    44      1
    62      1
    190     1
    Name: Glucose, Length: 136, dtype: int64
```

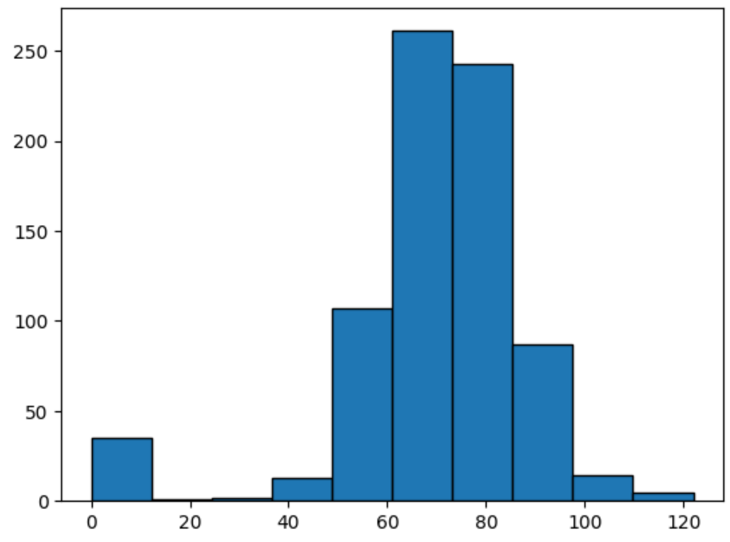
```
plt.hist(dia['Glucose'],edgecolor='black',linewidth=1.0)
plt.show()
```



```
dia['BloodPressure'].value_counts().head()
```

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

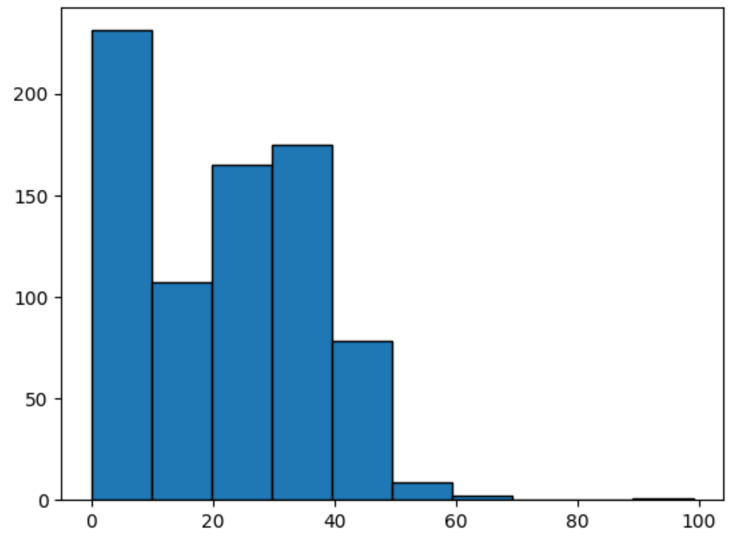
```
plt.hist(dia['BloodPressure'],edgecolor='black',linewidth=1.0)
plt.show()
```



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

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

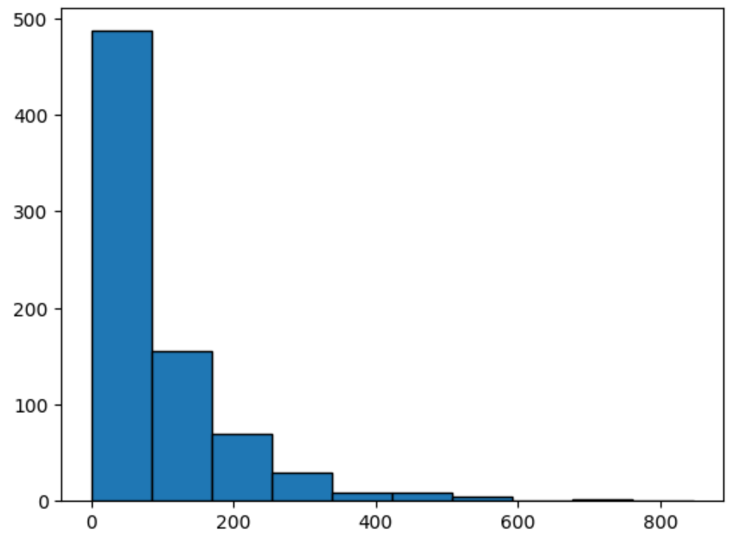
```
plt.hist(dia['SkinThickness'],edgecolor='black',linewidth=1.0)
plt.show()
```



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

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

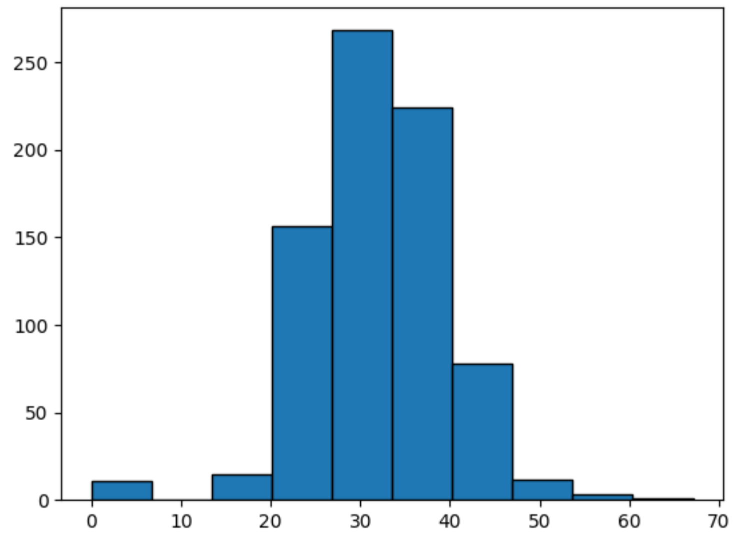
```
plt.hist(dia['Insulin'],edgecolor='black',linewidth=1.0)
plt.show()
```



```
dia['BMI'].value_counts().head()
```

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

```
plt.hist(dia['BMI'],edgecolor='black',linewidth=1.0)
plt.show()
```



Data Exploration:

(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:

```
dia['Outcome'].value_counts()
```

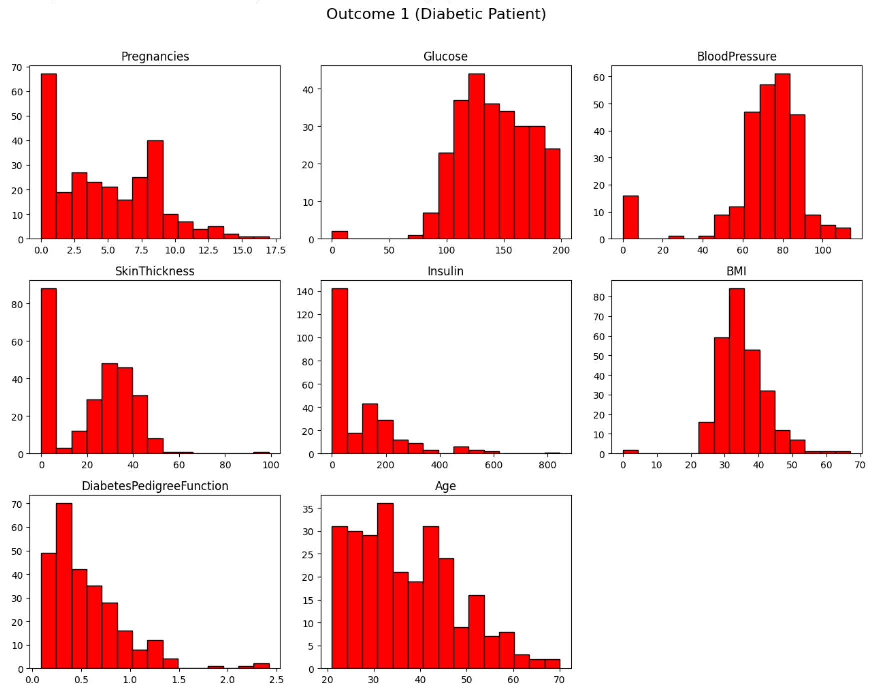
```
0    500
1    268
Name: Outcome, dtype: int64
```

```
Positive=Positive.drop('Outcome',axis=1)
Positive.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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	

```
Positive.hist(bins=15,color='r',edgecolor='black',linewidth=1.0,grid=False)
plt.tight_layout(rect=(0,0,2,2))
plt.suptitle('Outcome 1 (Diabetic Patient)',x=1,y=2.1,fontsize=16)
```

```
Text(1, 2.1, 'Outcome 1 (Diabetic Patient)')
```



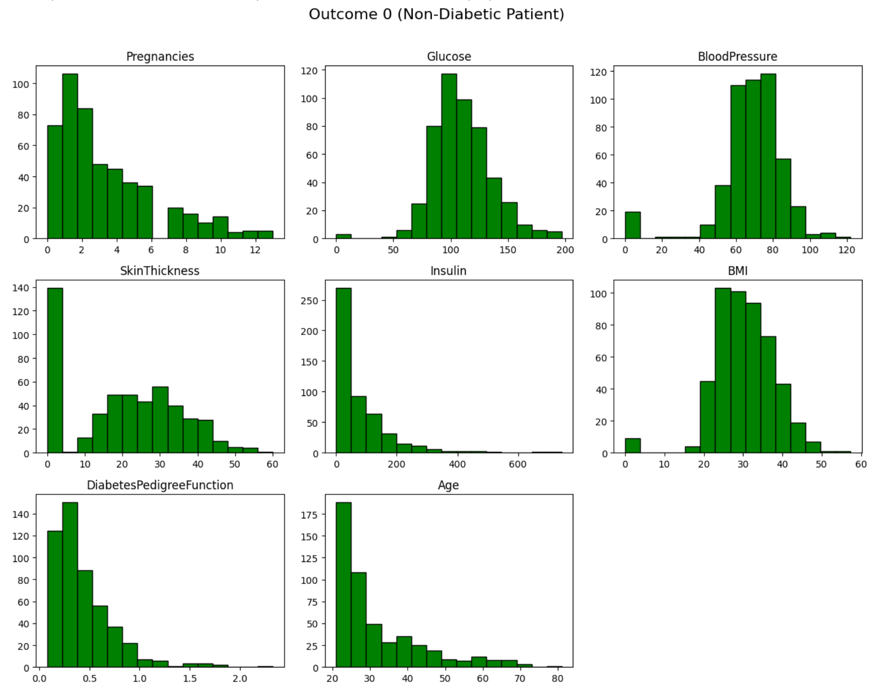
Pregnancies, Insulin, DiabetesPedigreeFunction, Age shows right skew.

```
Negative=Negative.drop('Outcome',axis=1)
Negative.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigre
1	1	85	66	29	0	26.6	
3	1	89	66	23	94	28.1	
5	5	116	74	0	0	25.6	
7	10	115	0	0	0	35.3	
10	4	110	92	0	0	37.6	

```
Negative.hist(bins=15,color='g',edgecolor='black',linewidth=1.0,grid=False)
plt.tight_layout(rect=(0,0,2,2))
plt.suptitle('Outcome 0 (Non-Diabetic Patient)',x=1,y=2.1,fontsize=16)
```

```
    Text(1, 2.1, 'Outcome 0 (Non-Diabetic Patient)')
```

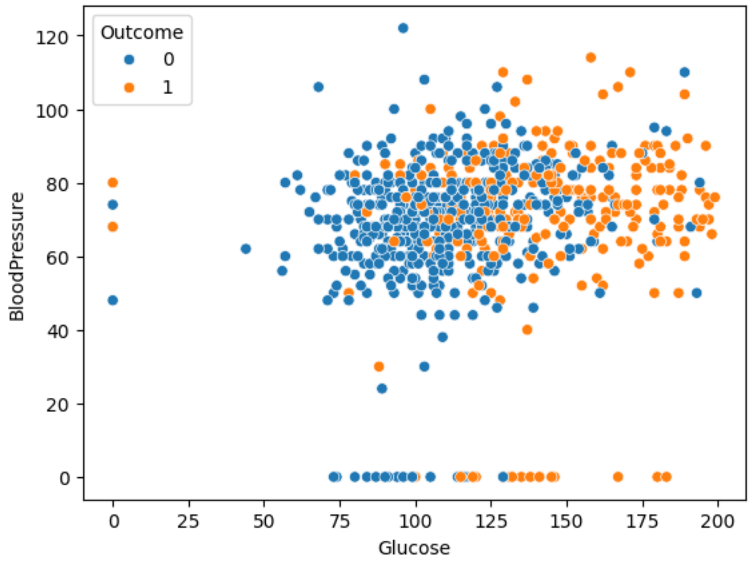


Pregnancies, Insulin, DiabetesPedigreeFunction, Age shows right skew.

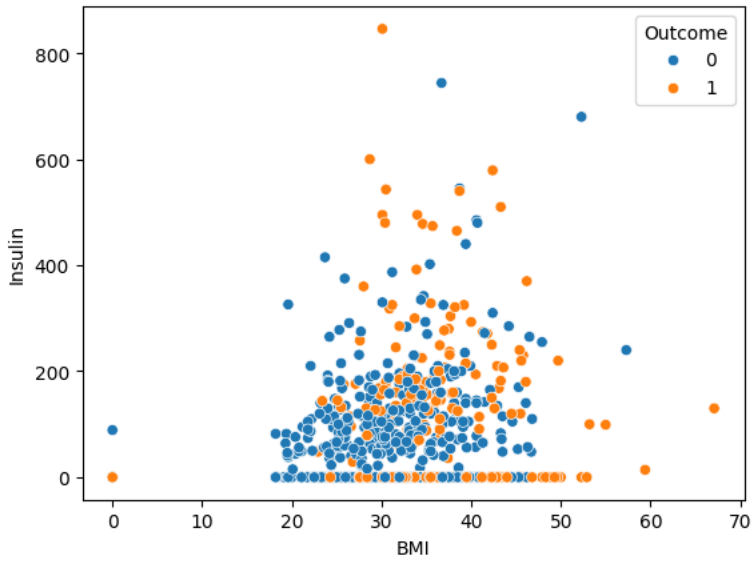
Scatter Plot

```
BloodPressure=Positive['BloodPressure']
SkinThickness=Positive['SkinThickness']
Glucose=Positive['Glucose']
Insulin=Positive['Insulin']
BMI=Positive['BMI']
Age=Positive['Age']
DiabetesPedigreeFunction=Positive['DiabetesPedigreeFunction']
```

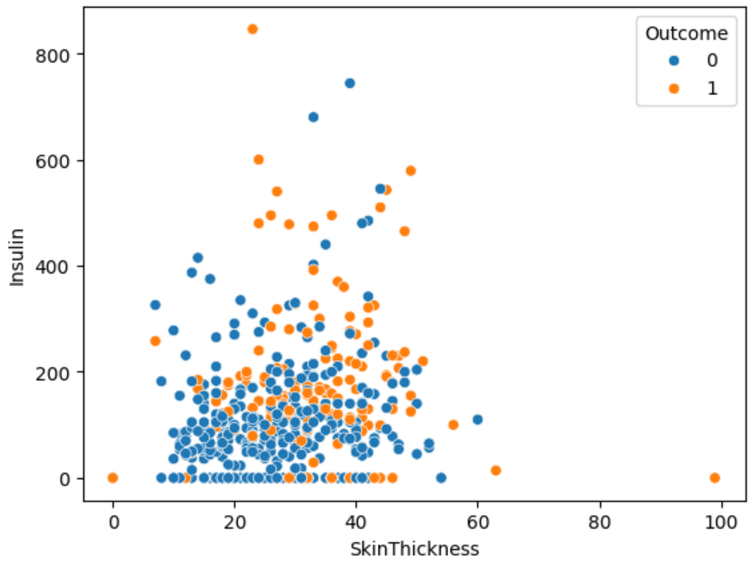
```
g=sns.scatterplot(x= "Glucose" ,y= "BloodPressure",hue='Outcome',data=dia);
```



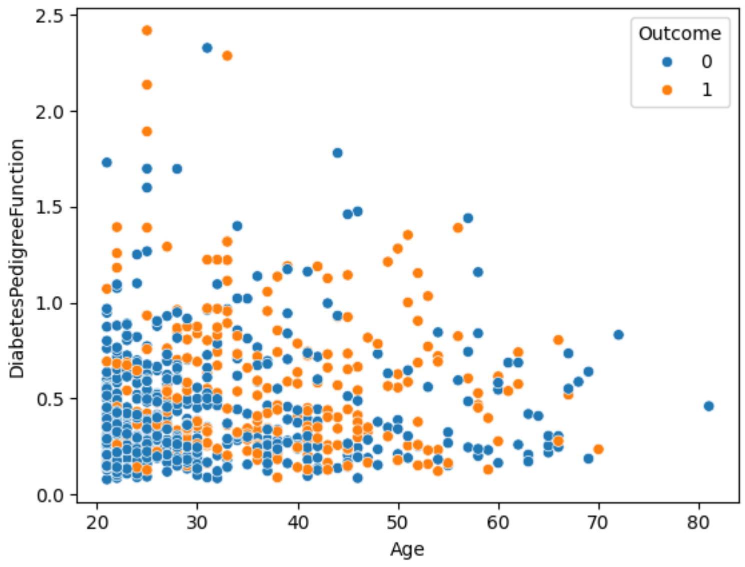
```
b =sns.scatterplot(x= "BMI" ,y= "Insulin",hue='Outcome',data=dia);
```



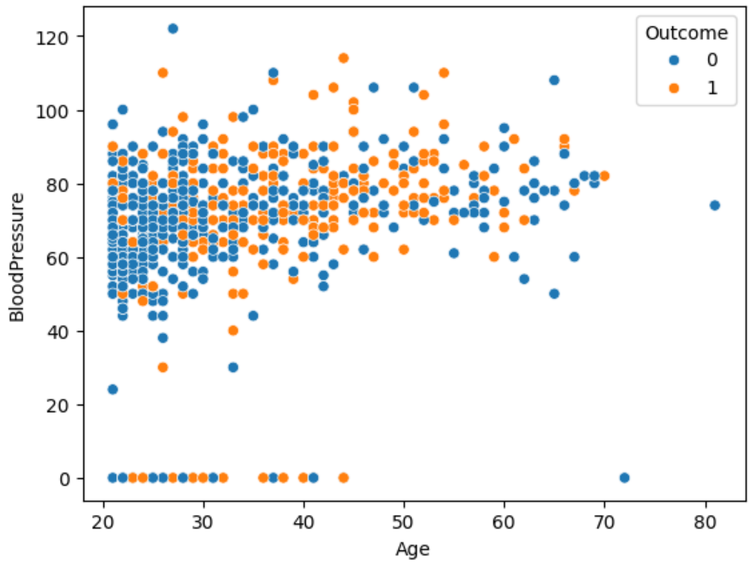
```
s=sns.scatterplot(x="SkinThickness",y="Insulin",hue='Outcome',data=dia)
```



```
s=sns.scatterplot(x="Age",y="DiabetesPedigreeFunction",hue='Outcome',data=dia)
```

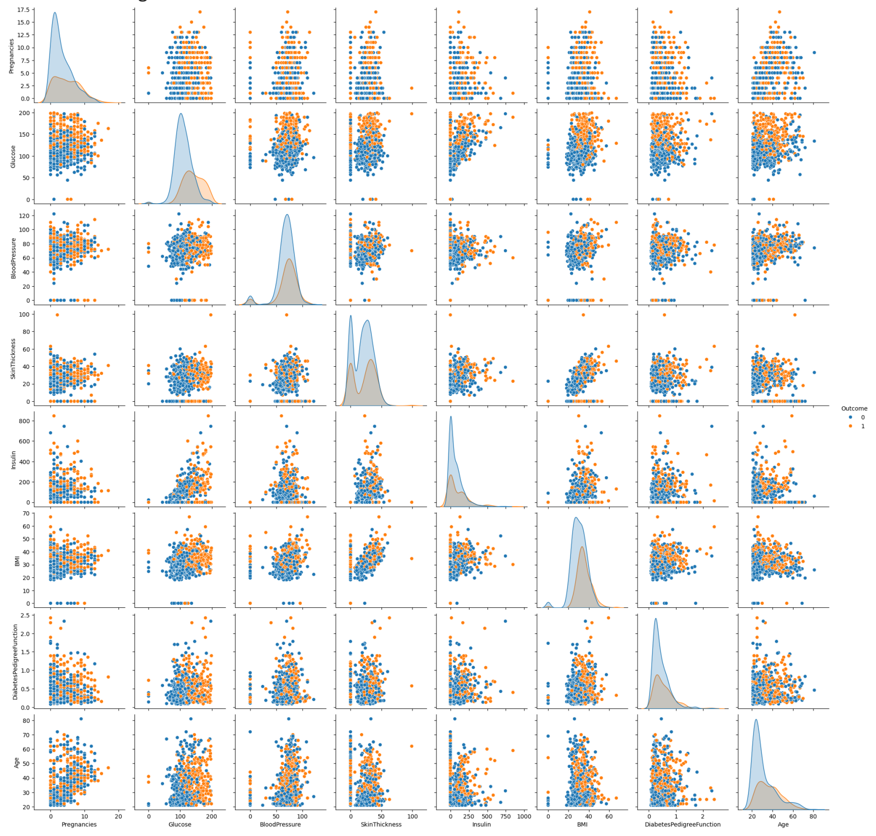


```
s=sns.scatterplot(x="Age",y="BloodPressure",hue='Outcome',data=dia)
```



```
sns.pairplot(dia,hue='Outcome')
```

```
<seaborn.axisgrid.PairGrid at 0x7df607746b00>
```



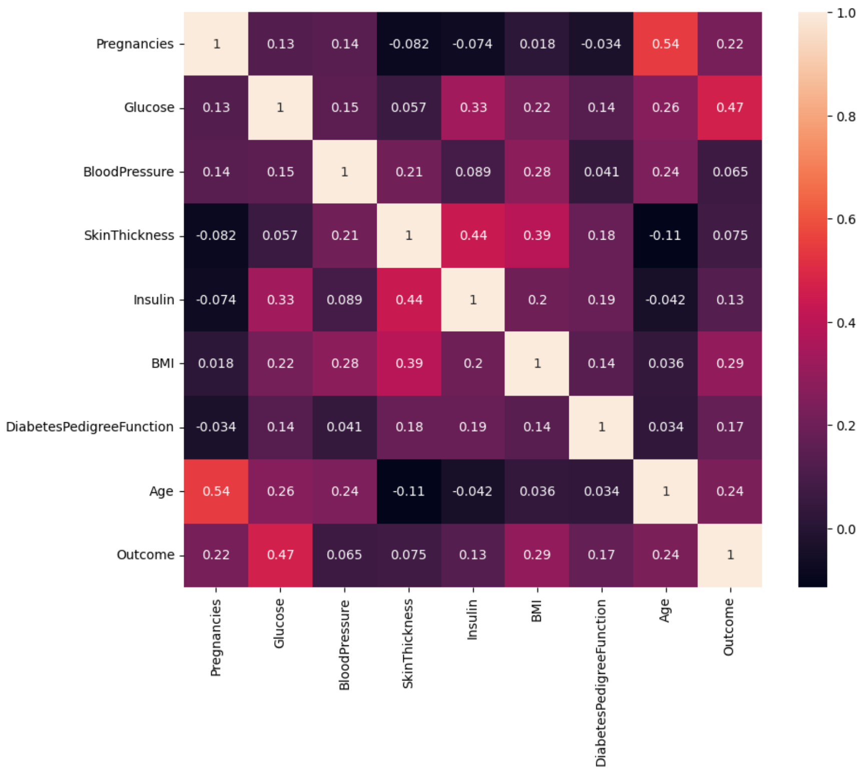
Pregnancies, Insulin, DiabetesPedigreeFunction and Age are Right skewed. Glucose, BloodPres- sure, SkinThickness, BMI are almost normally distributed.

```
# Correlation matrix
dia.corr()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insuli
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.07353
Glucose	0.129459	1.000000	0.152590	0.057328	0.33135
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.08893
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.43678
Insulin	-0.073535	0.331357	0.088933	0.436783	1.00000
BMI	0.017683	0.221071	0.281805	0.392573	0.19785
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.18507
Age	0.544341	0.263514	0.239528	-0.113970	-0.04216
Outcome	0.221898	0.466581	0.065068	0.074752	0.13054

```
# Heatmap
plt.subplots(figsize=(10,8))
sns.heatmap(dia.corr(),annot=True)
```

```
<Axes: >
```



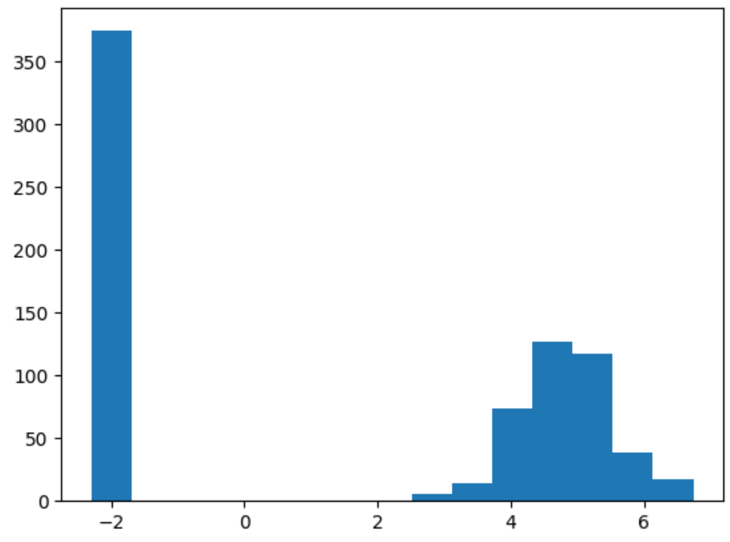
```
dia.skew()
```

```
Pregnancies                 0.901674
Glucose                     0.173754
BloodPressure              -1.843608
SkinThickness               0.109372
Insulin                     2.272251
BMI                        -0.428982
DiabetesPedigreeFunction    1.919911
Age                         1.129597
Outcome                     0.635017
dtype: float64
```

Normalization

```
dia['Insulin']=np.log(dia['Insulin']+0.1)
dia['Insulin'].hist(bins=15,grid=False)
```

```
<Axes: >
```

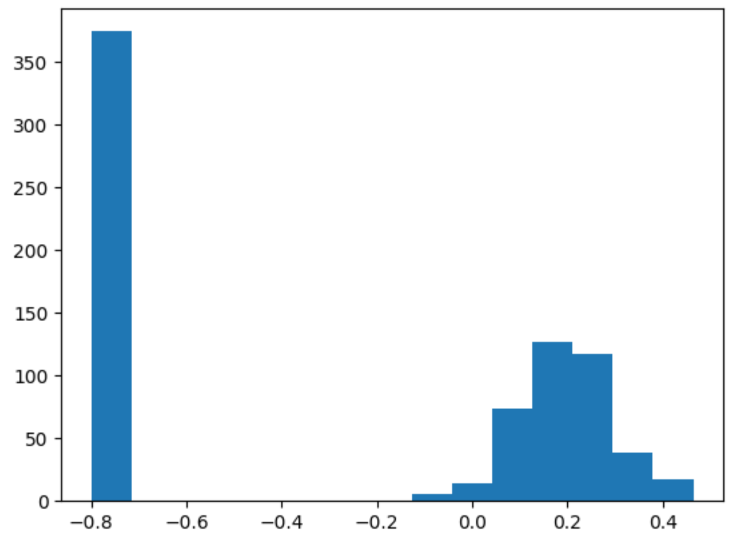


```
# Apply Robust Scaling to reduce outliers.
rs = RobustScaler(with_centering=True,
with_scaling=True,
quantile_range=(25.0, 75.0),
copy=True)
```

```
dia['Insulin']=rs.fit_transform(dia['Insulin'].values.reshape(-1,1))
```

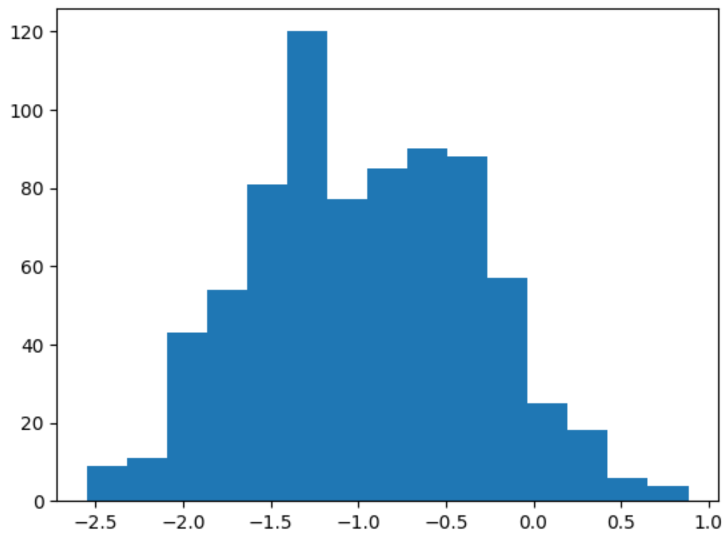
```
dia.Insulin.hist(bins=15,grid=False)
```

```
<Axes: >
```



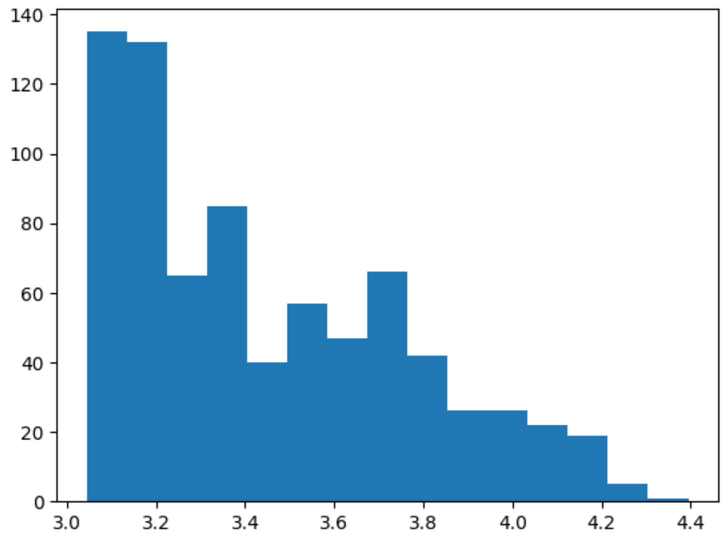
```
dia['DiabetesPedigreeFunction']=np.log(dia['DiabetesPedigreeFunction'])
dia['DiabetesPedigreeFunction'].hist(bins=15,grid=False)
```

```
<Axes: >
```



```
dia['Age']=np.log(dia['Age'])
dia['Age'].hist(bins=15,grid=False)
```

```
<Axes: >
```



```
dia.skew()
```

```
Pregnancies                 0.901674
Glucose                     0.173754
BloodPressure              -1.843608
SkinThickness               0.109372
Insulin                     0.005021
BMI                        -0.428982
DiabetesPedigreeFunction    0.114178
Age                         0.601746
Outcome                     0.635017
dtype: float64
```

```
dia['Outcome'].value_counts()
```

```
0    500
1    268
Name: Outcome, dtype: int64
```

Here, the data is imbalanced. So, it is oversampled.

Oversampling of Data

```
# Synthetic Minority Over-sampling Technique
from imblearn.over_sampling import SMOTE
```

```
smt = SMOTE(sampling_strategy='auto', random_state=9,n_jobs=-1)
X = dia.drop(['Outcome'],axis = 1)
y = dia.Outcome


X, y = smt.fit_resample(X,y)
```

Data Modeling

Logistic Regression

```
# Train-test 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=20)
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

```
((800, 8), (200, 8), (800,), (200,))
```

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

```
▾ LogisticRegression
LogisticRegression()
```

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

```
0.75375
```

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

```
0.74
```

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y,model.predict(X))
cm
```

```
array([[377, 123],
       [126, 374]])
```

```
from sklearn.metrics import classification_report
print(classification_report(y,model.predict(X)))
```

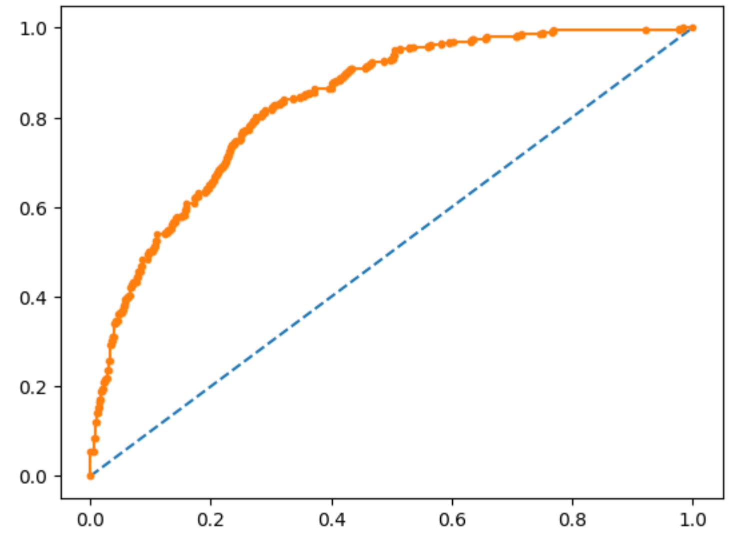
```
              precision    recall  f1-score   support

           0       0.75      0.75      0.75       500
           1       0.75      0.75      0.75       500

    accuracy                           0.75      1000
   macro avg       0.75      0.75      0.75      1000
weighted avg       0.75      0.75      0.75      1000
```

```
#Preparing ROC Curve (Receiver Operating Characteristics Curve)
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
# predict probabilities
probs = model.predict_proba(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)
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
```

```
AUC: 0.833
[<matplotlib.lines.Line2D at 0x7df6006f36d0>]
```



Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
model3=DecisionTreeClassifier(max_depth=5)
model3.fit(X_train,y_train)
```

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

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

```
0.82375
```

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

```
0.78
```

Random Forest (Ensemble Technique)

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

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

```
model4.score(X_train,y_train)
```

```
0.99375
```

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

```
0.82
```

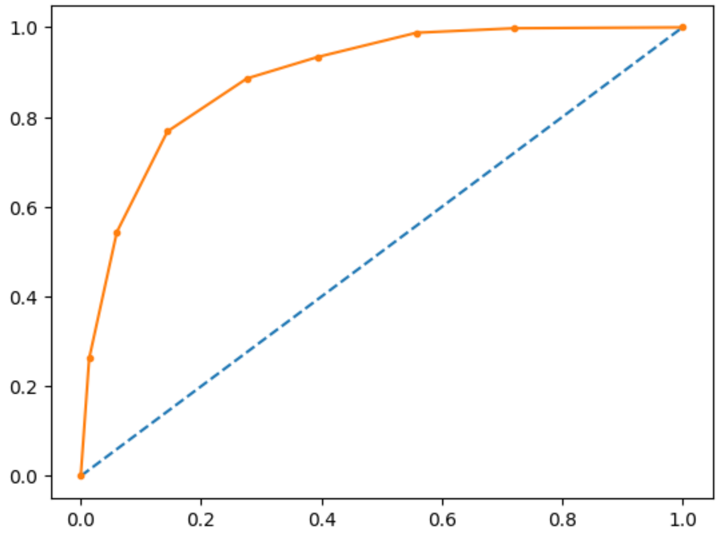
KNN

```
from sklearn.neighbors import KNeighborsClassifier
model2=KNeighborsClassifier(n_neighbors=7,metric='minkowski',p=2)
model2.fit(X_train,y_train)
```

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

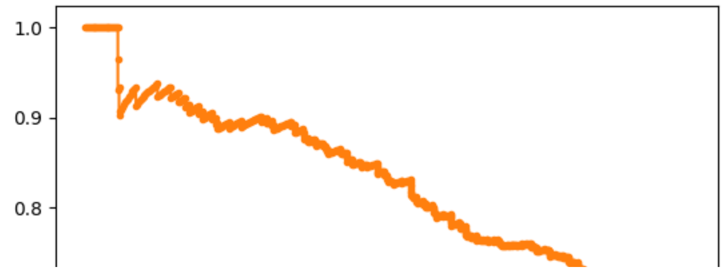
```
#Preparing ROC Curve (Receiver Operating Characteristics Curve)
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
# predict probabilities
probs = model2.predict_proba(X)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(y, 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 no skill
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
```

```
AUC: 0.890
True Positive Rate - [0.    0.262 0.544 0.768 0.886 0.934 0.988 0.998 1.   ], False F
 0.28571429 0.14285714 0.        ]
[<matplotlib.lines.Line2D at 0x7df5ff4b97b0>]
```



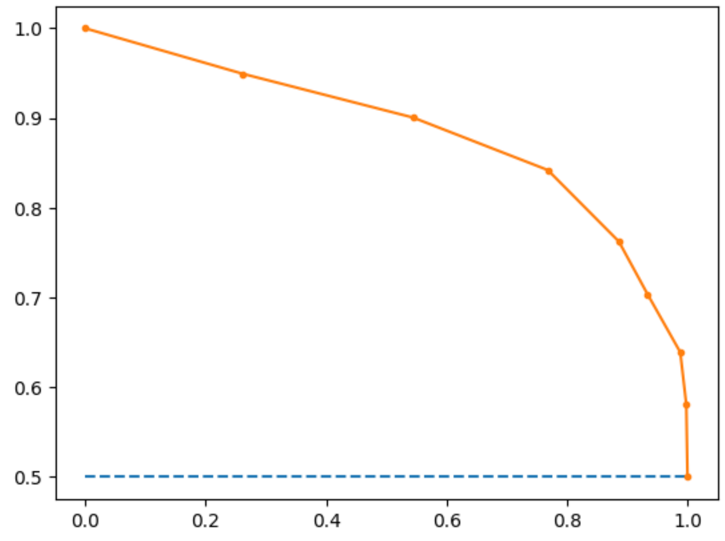
```
#Precision Recall Curve for Logistic Regression
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model.predict_proba(X)
# keep probabilities for 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=0.750 auc=0.818 ap=0.819
[<matplotlib.lines.Line2D at 0x7df5ff522950>]
```



```
#Precision Recall Curve for KNN
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model2.predict_proba(X)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# 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=0.820 auc=0.885 ap=0.856
[<matplotlib.lines.Line2D at 0x7df5feb9be20>]
```



```
#Precision Recall Curve for Decission Tree Classifier
```