In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
df=pd.read_csv("/home/sandhan/Downloads/diabetes.csv")
df
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0.
1	1	85	66	29	0	26.6	0.
2	8	183	64	0	0	23.3	0.
3	1	89	66	23	94	28.1	0.
4	0	137	40	35	168	43.1	2.
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.
764	2	122	70	27	0	36.8	0.
765	5	121	72	23	112	26.2	0.
766	1	126	60	0	0	30.1	0.
767	1	93	70	31	0	30.4	0.

768 rows × 9 columns

In [3]:

```
df.columns
```

Out[3]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'In
sulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [4]:

```
df.shape
```

Out[4]:

```
(768, 9)
```

In [5]:

```
df.describe()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabet
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	

In [6]:

```
df.isna().sum()
```

Out[6]:

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

In [7]:

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

In [8]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
```

In [9]:

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
```

In [10]:

```
#KNN
from sklearn.neighbors import KNeighborsClassifier
classifier=KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
y_pred
```

Out[10]:

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

In [11]:

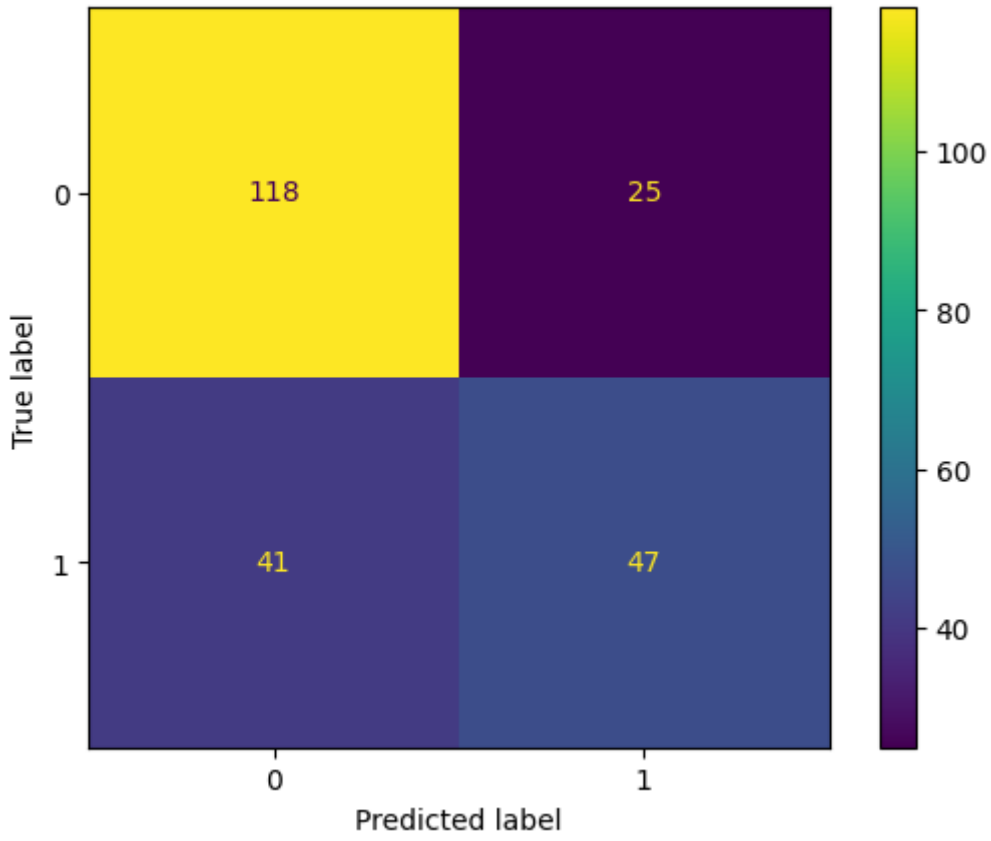
```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,C
print(classification_report(y_test,y_pred))
result=confusion_matrix(y_test,y_pred)
print(result)
print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.74      0.83      0.78       143
           1       0.65      0.53      0.59        88

    accuracy                           0.71       231
   macro avg       0.70      0.68      0.68       231
weighted avg       0.71      0.71      0.71       231

[[118  25]
 [ 41  47]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object
at 0x7f63d45a3b50>
```



In [12]:

```
score=accuracy_score(y_test,y_pred)
print(score)
```

```
0.7142857142857143
```

In [13]:

```
#Naive Bayes
from sklearn.naive_bayes import GaussianNB
model=GaussianNB()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
```

In [14]:

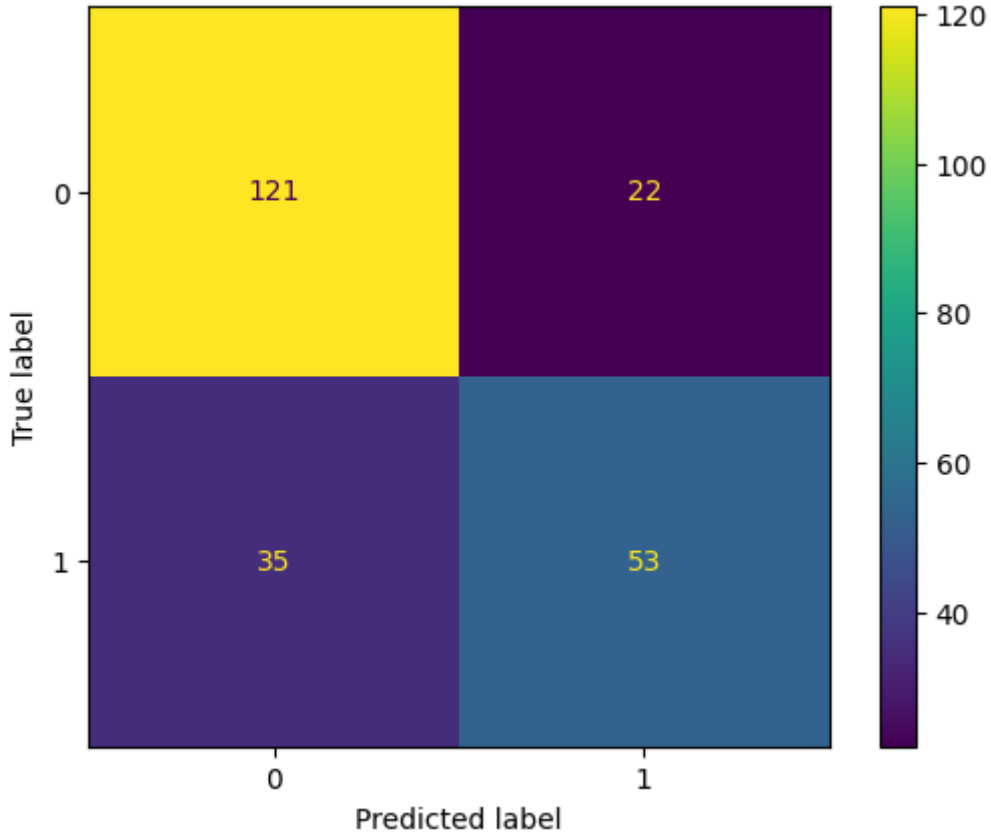
```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,C
print(classification_report(y_test,y_pred))
result=confusion_matrix(y_test,y_pred)
print(result)
print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.85      0.81       143
           1       0.71      0.60      0.65        88

    accuracy                           0.75       231
   macro avg       0.74      0.72      0.73       231
weighted avg       0.75      0.75      0.75       231

[[121  22]
 [ 35  53]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object
at 0x7f63badc5550>
```



In [15]:

```
score=accuracy_score(y_test,y_pred)
print(score)
```

```
0.7532467532467533
```

In [16]:

```
#SVM
from sklearn.svm import SVC
classifier=SVC()
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
y_pred
```

Out[16]:

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

In [17]:

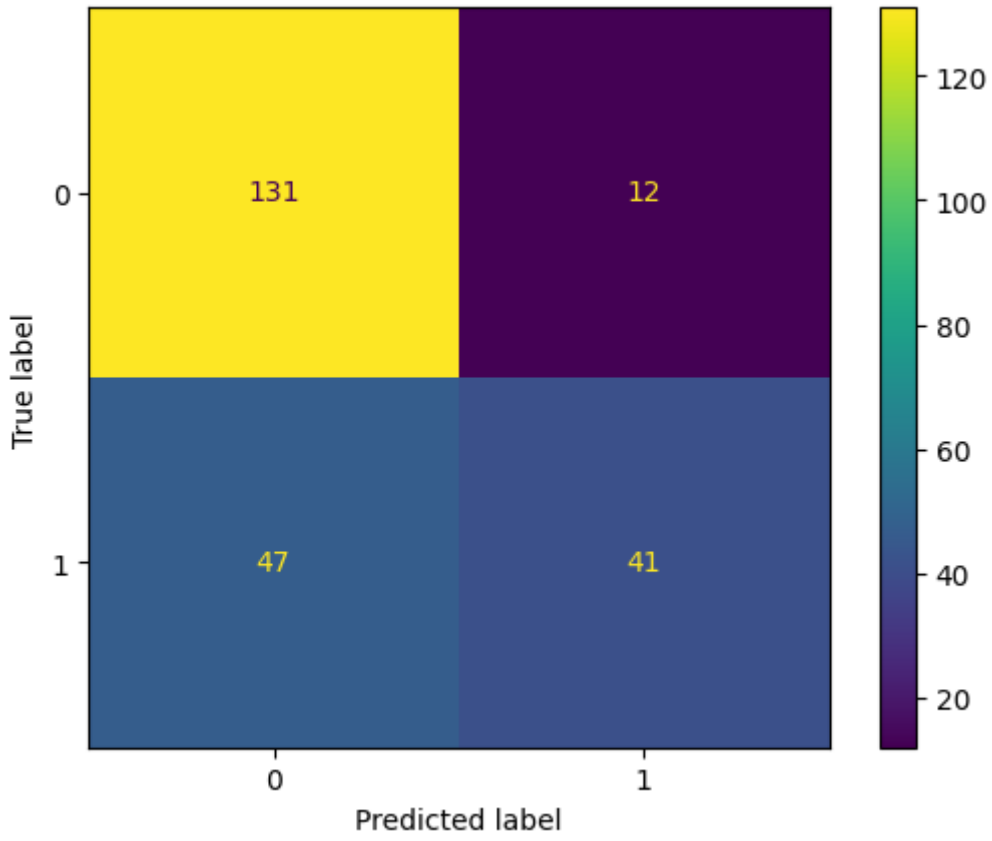
```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
print(classification_report(y_test,y_pred))
result=confusion_matrix(y_test,y_pred)
print(result)
print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.74      0.92      0.82       143
           1       0.77      0.47      0.58        88

    accuracy                           0.74       231
   macro avg       0.75      0.69      0.70       231
weighted avg       0.75      0.74      0.73       231

[[131  12]
 [ 47  41]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object
at 0x7f63d45c8700>
```



In [18]:

```
score=accuracy_score(y_test,y_pred)
print(score)
```

```
0.7445887445887446
```

In [19]:

```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier
dec=DecisionTreeClassifier()
dec.fit(x_train,y_train)
y_pred=dec.predict(x_test)
y_pred
```

Out[19]:

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

In [20]:

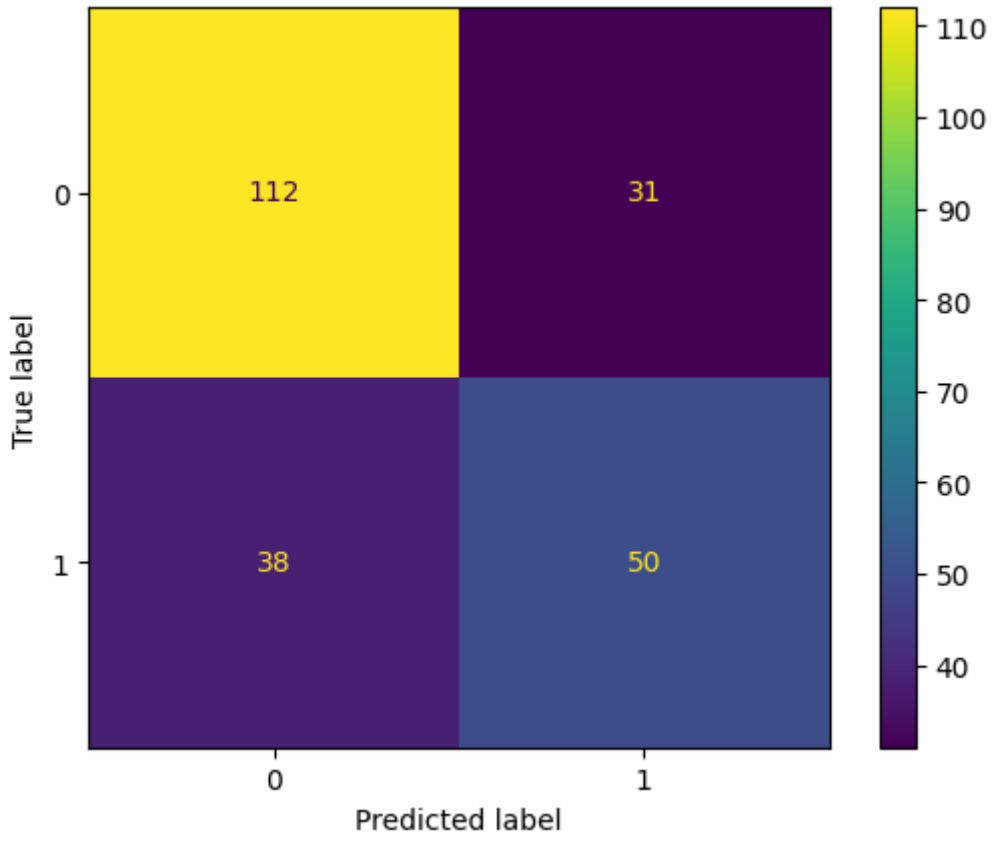
```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
print(classification_report(y_test,y_pred))
result=confusion_matrix(y_test,y_pred)
print(result)
print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.78      0.76       143
           1       0.62      0.57      0.59        88

    accuracy                           0.70       231
   macro avg       0.68      0.68      0.68       231
weighted avg       0.70      0.70      0.70       231

[[112  31]
 [ 38  50]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object
at 0x7f63bb0f9e80>
```



In [21]:

```
score=accuracy_score(y_test,y_pred)
print(score)
```

```
0.7012987012987013
```

In [22]:

```
#Random Forest
from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier(n_estimators=25,criterion='entropy')
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)
y_pred
```

Out[22]:

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

In [23]:

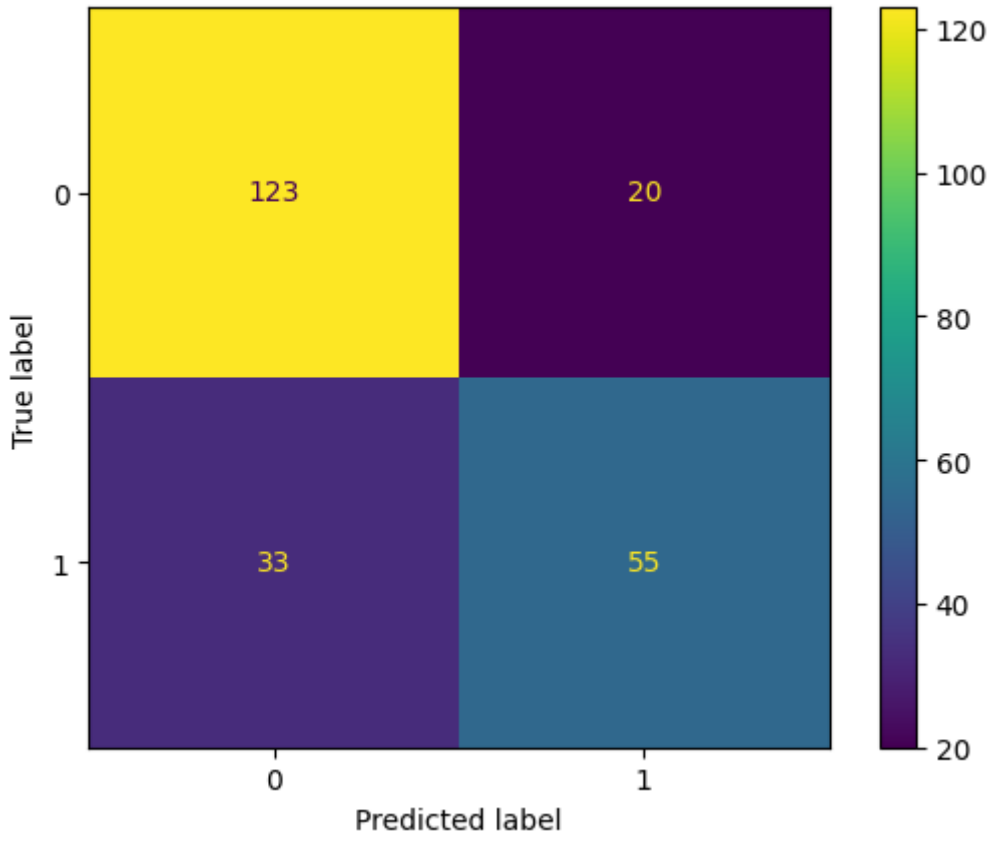
```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,C
print(classification_report(y_test,y_pred))
result=confusion_matrix(y_test,y_pred)
print(result)
print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.86      0.82       143
           1       0.73      0.62      0.67        88

    accuracy                           0.77       231
   macro avg       0.76      0.74      0.75       231
weighted avg       0.77      0.77      0.77       231

[[123  20]
 [ 33  55]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object
at 0x7f63bcd7c220>
```



In [24]:

```
score=accuracy_score(y_test,y_pred)
print(score)
```

```
0.7705627705627706
```

In [ ]:

```
#Comparing the results obtained from each of the above classification algorithms,
#we see that Random Forest algorthm has the highest accuracy of about 77% for this
#So we choose Random Forest classification algorithm for this dataset.
```