```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import joblib

pip install joblib
```

```
Requirement already satisfied: joblib in c:\users\hp\appdata\local\
programs\python\python313\lib\site-packages (1.4.2)
Note: you may need to restart the kernel to use updated packages.
```

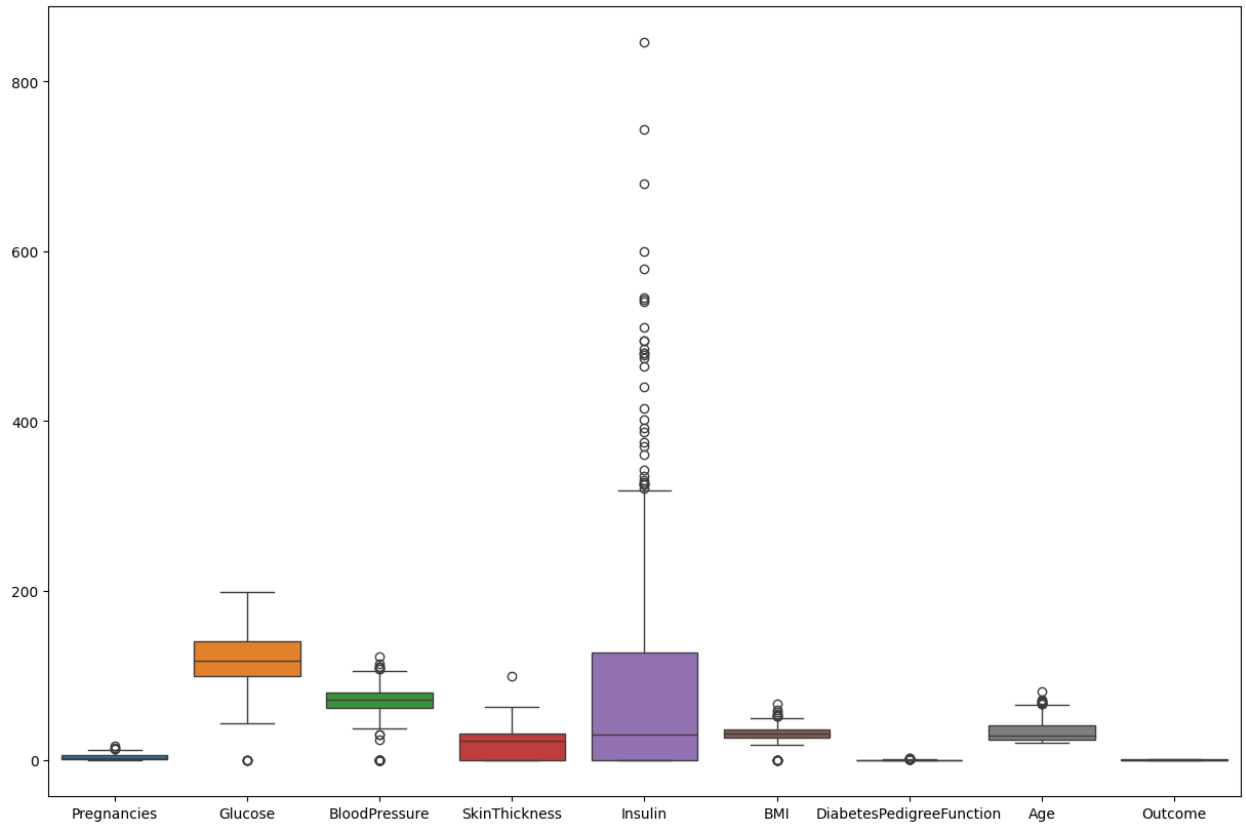
```
data = pd.read_csv("bcd.csv")

data.head()
```

```
   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
```

```
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

(data.columns)

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

data.describe().T

                           count        mean         std     min
25%  \
Pregnancies                768.0    3.845052    3.369578   0.000
1.00000
Glucose                    768.0  120.894531   31.972618   0.000
99.00000
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

plt.figure(figsize = (15,10))
sns.boxplot(data=data)

<Axes: >
```



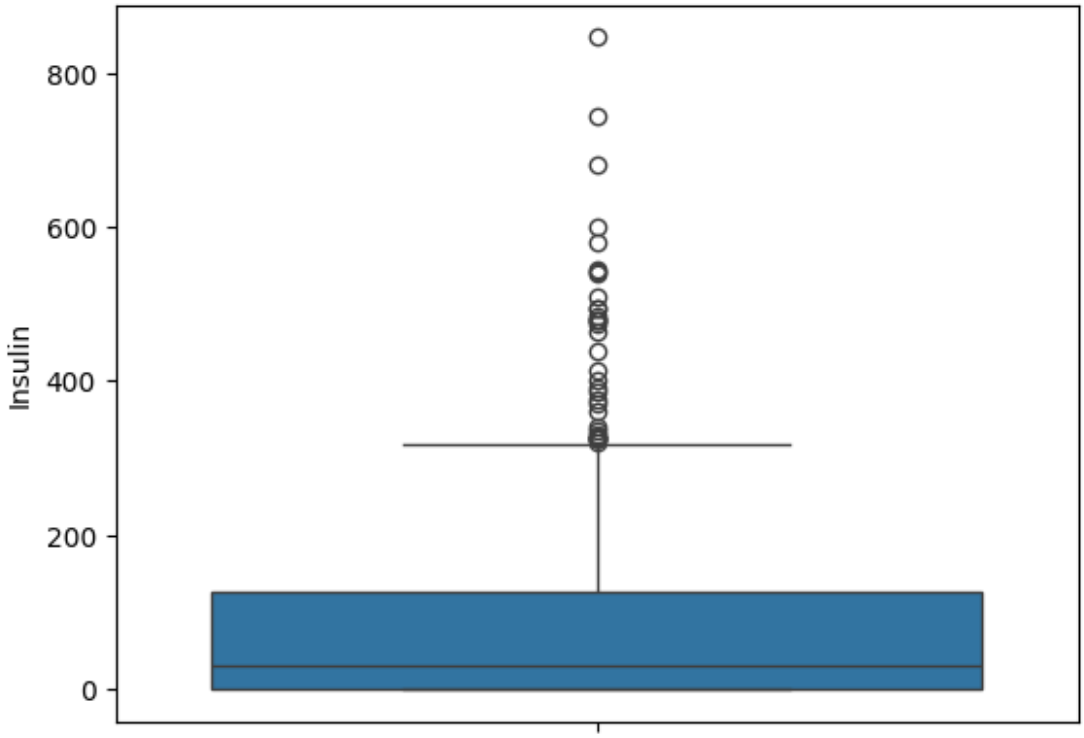

```
# to remove outlayer  to find value
def outlier(s):
   q1 = data[s].quantile(.25)
   q3 = data[s].quantile(.75)
   iqr = q3 - q1
   ul = q3+1.5*iqr
   ll = q1-1.5*iqr
   return (ll,ul)

outlier("Glucose")
```

```
(np.float64(37.125), np.float64(202.125))
```

```
for i in data.columns:
    print(f"{i} = {outlier(i)}")
```

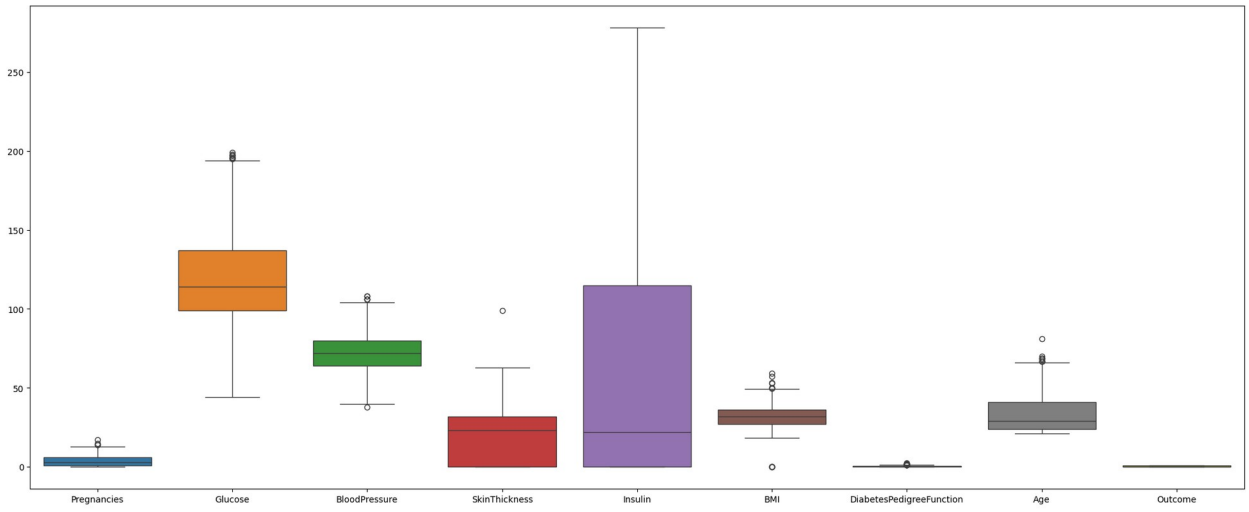
```
Pregnancies = (np.float64(-6.5), np.float64(13.5))
Glucose = (np.float64(37.125), np.float64(202.125))
BloodPressure = (np.float64(35.0), np.float64(107.0))
SkinThickness = (np.float64(-48.0), np.float64(80.0))
Insulin = (np.float64(-190.875), np.float64(318.125))
BMI = (np.float64(13.35), np.float64(50.550000000000004))
DiabetesPedigreeFunction = (np.float64(-0.32999999999999996),
np.float64(1.2))
```

```
Age = (np.float64(-1.5), np.float64(66.5))
Outcome = (np.float64(-1.5), np.float64(2.5))

data.shape

(768, 9)

data1 = data[data['Insulin']<280]

plt.figure(figsize =(10,15))

<Figure size 1000x1500 with 0 Axes>

<Figure size 1000x1500 with 0 Axes>

sns.boxplot(data = data['Insulin'])

<Axes: ylabel='Insulin'>
```

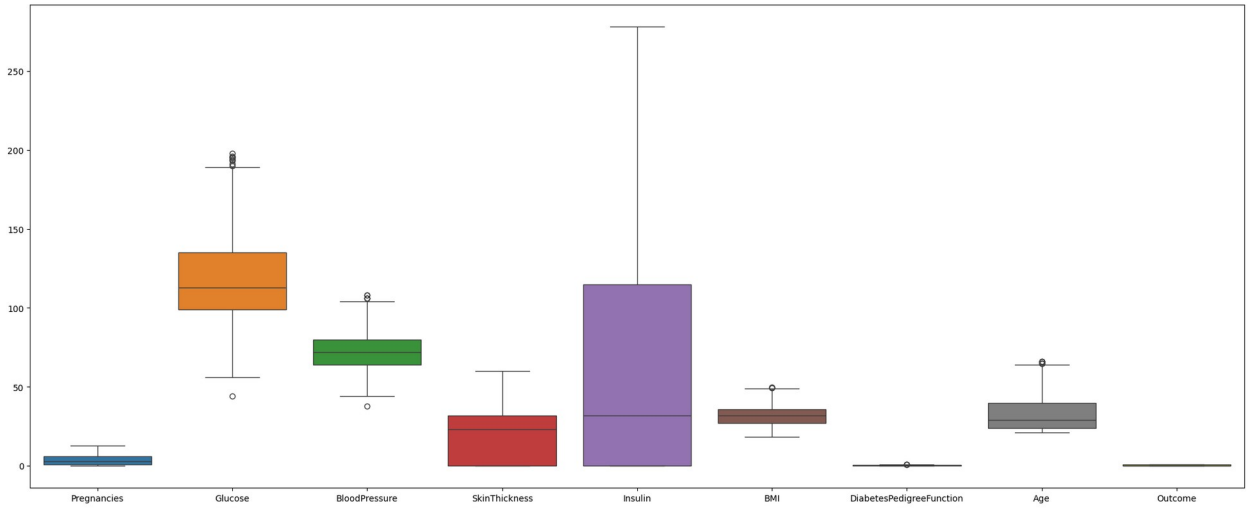



```
data2 =
data1[(data['Glucose']>40)&((data['BloodPressure']>35)&(data['BloodPre
ssure']<110))]

C:\Users\hp\AppData\Local\Temp\ipykernel_16728\498802551.py:1:
UserWarning: Boolean Series key will be reindexed to match DataFrame
index.
  data2 =
```

```
data1[(data['Glucose']>40)&((data['BloodPressure']>35)&(data['BloodPressure']<110))]

plt.figure(figsize =(25,10))
sns.boxplot(data = data2)

<Axes: >
```




```
data.shape

(768, 9)

df = data2[(data2['SkinThickness']<70)&((data['BMI']>17)&(data2['BMI']<50))&((data['Age']<67)&(data2['Pregnancies']<14))&(data2['DiabetesPedigreeFunction']<1.1)]

C:\Users\hp\AppData\Local\Temp\ipykernel_16728\339148766.py:1: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  df = data2[(data2['SkinThickness']<70)&((data['BMI']>17)&(data2['BMI']<50))&((data['Age']<67)&(data2['Pregnancies']<14))&(data2['DiabetesPedigreeFunction']<1.1)]

plt.figure(figsize =(25,10))
sns.boxplot(data = df)

<Axes: >
```




```
df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 617 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               617 non-null    int64  
 1   Glucose                   617 non-null    int64  
 2   BloodPressure             617 non-null    int64  
 3   SkinThickness             617 non-null    int64  
 4   Insulin                   617 non-null    int64  
 5   BMI                       617 non-null    float64
 6   DiabetesPedigreeFunction  617 non-null    float64
 7   Age                       617 non-null    int64  
 8   Outcome                   617 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 48.2 KB

x = df[df.columns[:-1]]
x
```

	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
5	5	116	74	0	0	25.6
..	...	...	...	...	...	...

```
763            10      101              76              48       180  32.9

764             2      122              70              27         0  36.8

765             5      121              72              23       112  26.2

766             1      126              60               0         0  30.1

767             1       93              70              31         0  30.4


     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
5                       0.201   30
..                        ...  ...
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23

[617 rows x 8 columns]

df

     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

5              5      116             74              0         0  25.6

..           ...      ...            ...            ...      ...   ...

763            10      101             76             48       180  32.9

764             2      122             70             27         0  36.8

765             5      121             72             23       112  26.2

766             1      126             60              0         0  30.1
```

```
767            1       93             70             31         0  30.4


     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
5                       0.201   30        0
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[617 rows x 9 columns]

x

     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

5              5      116             74              0        0  25.6

..           ...      ...            ...            ...      ...   ...

763           10      101             76             48      180  32.9

764            2      122             70             27        0  36.8

765            5      121             72             23      112  26.2

766            1      126             60              0        0  30.1

767            1       93             70             31        0  30.4


     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
5                       0.201   30
```

```
..                          ...  ...
763                       0.171   63
764                       0.340   27
765                       0.245   30
766                       0.349   47
767                       0.315   23

[617 rows x 8 columns]
```

```
y = df['Outcome']
y
```

```
0      1
1      0
2      1
3      0
5      0
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 617, dtype: int64
```

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test =
train_test_split(x,y,test_size=0.25,random_state=42)

x_train.shape
```

```
(462, 8)
```

```
x_test.shape
```

```
(155, 8)
```

```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()

x_train = ss.fit_transform(x_train)
x_train
```

```
array([[-0.88120816,  0.48512932,  2.56297614, ...,  0.11935477,
        -0.81334446,  1.05677328],
       [-0.88120816, -1.25338872, -0.36007833, ..., -2.17756899,
         0.88519638, -0.51703163],
       [ 1.2566967 ,  2.08456592, -0.01618957, ...,  0.10362241,
         1.29894351,  0.26987083],
       ...,
```

```
       [ 0.34045176,  0.10265535,  1.18742109, ...,  0.4182695 ,
        -0.56945142,  0.00757001],
       [-0.88120816,  1.5630105 ,  0.84353233, ...,  0.11935477,
        -0.34733454,  1.49394131],
       [-1.18662314, -0.48844078,  1.01547671, ..., -0.6515306 ,
         1.39475863,  2.54314458]], shape=(462, 8))
```

```
x_test = ss.transform(x_test)
```

```
from sklearn.linear_model import LogisticRegression
```

```
classification = LogisticRegression()
```

```
classification.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
cls_pred = classification.predict(x_test)
```

```
cls_pred
```

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

```
y_test
```

```
66     1
729    0
104    0
387    1
137    0
      ..
31     1
366    1
384    0
355    1
312    1
Name: Outcome, Length: 155, dtype: int64
```

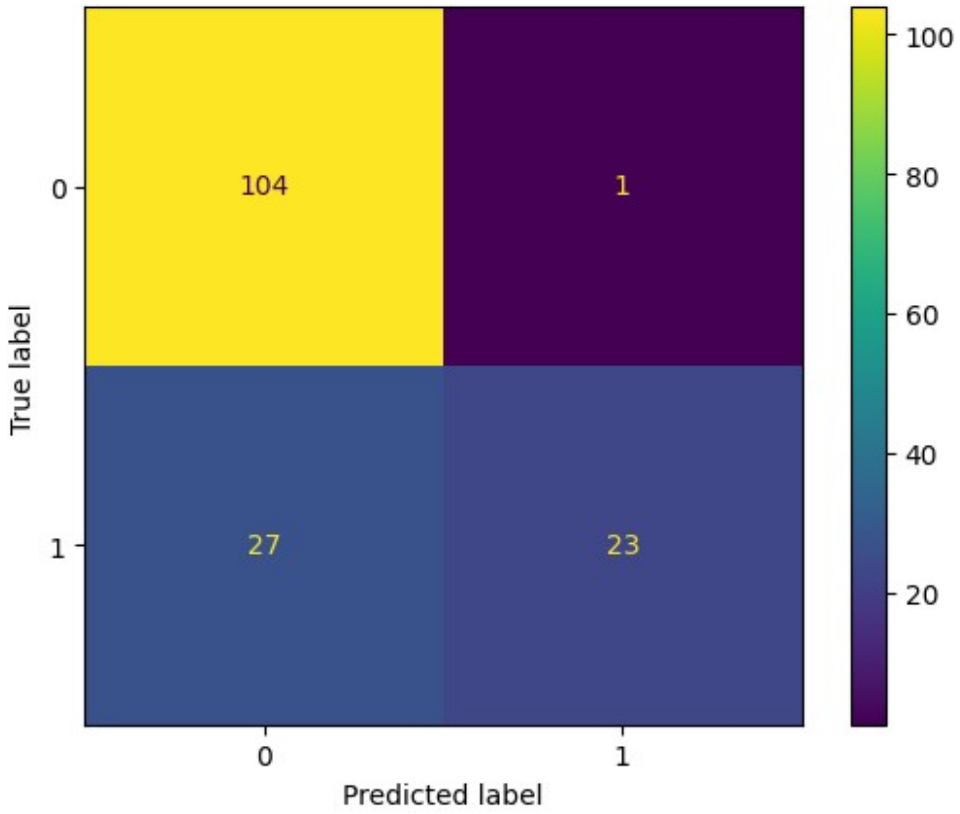
```
from sklearn.metrics import
accuracy_score,confusion_matrix,roc_curve,roc_auc_score,ConfusionMatri
xDisplay

accuracy_score(y_test,cls_pred)
```

```
0.8193548387096774
```

```
confusion_matrix(y_test,cls_pred)
```

```
array([[104,   1],
       [ 27,  23]])
```

```
cm_display =
ConfusionMatrixDisplay(confusion_matrix(y_test,cls_pred)).plot()
```



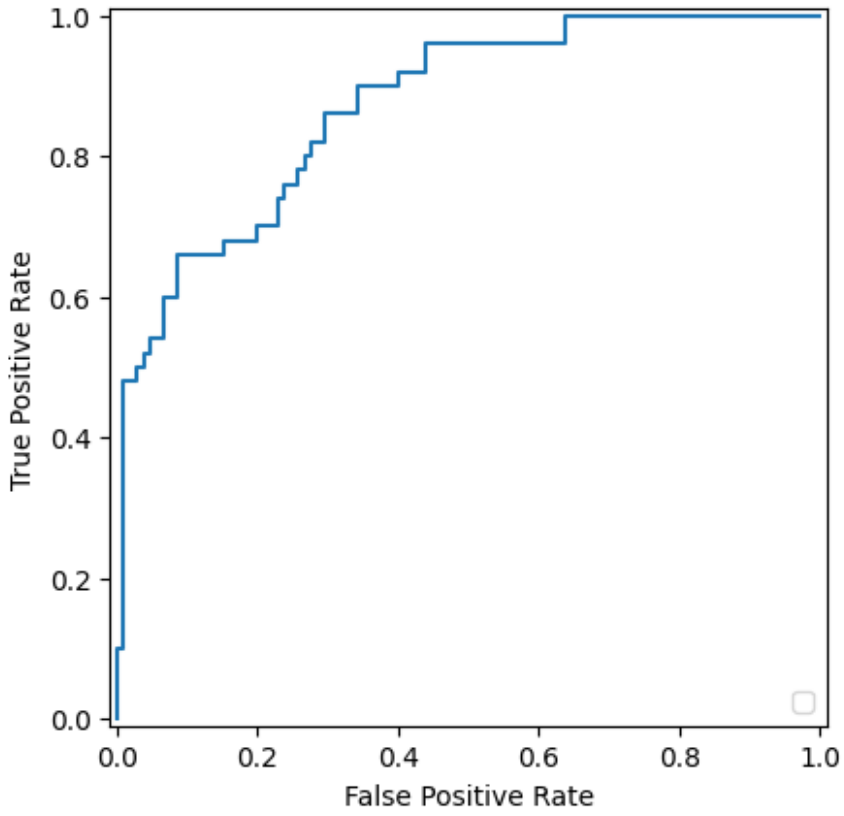

```
roc_curve(cls_pred,y_test)
```

```
(array([0.        , 0.20610687, 1.        ]),
 array([0.        , 0.95833333, 1.        ]),
 array([inf,  1.,  0.]))
```

```
from sklearn.metrics import RocCurveDisplay,roc_curve
y_score = classification.decision_function(x_test)
fpr,tpr,_ = roc_curve(y_test,y_score,pos_label =
```

```
classification.classes_[1])
roc_display = RocCurveDisplay(fpr = fpr,tpr = tpr).plot()
```

```
C:\Users\hp\AppData\Local\Programs\Python\Python313\Lib\site-packages\
sklearn\metrics\_plot\roc_curve.py:189: UserWarning: No artists with
labels found to put in legend.  Note that artists whose label start
with an underscore are ignored when legend() is called with no
argument.
  self.ax_.legend(loc="lower right")
```




```
joblib.dump(classification,"test.pkl")
```

```
['test.pkl']
```

```
xyz = joblib.load('test.pkl')
```

```
xyz
```

```
LogisticRegression()
```

```
joblib.dump(ss,"stand.pkl")
```

```
['stand.pkl']
```