In [5]:
```
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

In [6]:
```
data_set_name = sb.get_dataset_names()
print(data_set_name)
```

```
['anagrams', 'anscombe', 'attention', 'brain_networks', 'car_crashes', 'diamonds', 'dots', 'dowjones', 'exercise', 'fligh
ts', 'fmri', 'geyser', 'glue', 'healthexp', 'iris', 'mpg', 'penguins', 'planets', 'seaice', 'taxis', 'tips', 'titanic',
'anagrams', 'anagrams', 'anscombe', 'anscombe', 'attention', 'attention', 'brain_networks', 'brain_networks', 'car_crashe
s', 'car_crashes', 'diamonds', 'diamonds', 'dots', 'dots', 'dowjones', 'dowjones', 'exercise', 'exercise', 'flights', 'fl
ights', 'fmri', 'fmri', 'geyser', 'geyser', 'glue', 'glue', 'healthexp', 'healthexp', 'iris', 'iris', 'mpg', 'mpg', 'peng
uins', 'penguins', 'planets', 'planets', 'seaice', 'seaice', 'taxis', 'taxis', 'tips', 'tips', 'titanic', 'titanic', 'ana
grams', 'anscombe', 'attention', 'brain_networks', 'car_crashes', 'diamonds', 'dots', 'dowjones', 'exercise', 'flights',
'fmri', 'geyser', 'glue', 'healthexp', 'iris', 'mpg', 'penguins', 'planets', 'seaice', 'taxis', 'tips', 'titanic']
```

In [7]:
```
df = sb.load_dataset("titanic")
```

In [8]:
```
df
```

Out[8]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

891 rows × 15 columns

In [9]:
```
df.head()
```

Out[9]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

In [11]:
```
df["sex"].value_counts(normalize=True)
```

Out[11]:
```
sex
male      0.647587
female    0.352413
Name: proportion, dtype: float64
```

In [12]:
```
df.describe()
```

Out[12]:

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [13]:
```
df["deck"].value_counts(normalize=True)
```

Out[13]:
```
deck
C    0.290640
B    0.231527
D    0.162562
E    0.157635
A    0.073892
F    0.064039
G    0.019704
Name: proportion, dtype: float64
```

In [14]:
```
df.drop(["deck"], axis=1)
```

Out[14]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	Queenstown	no	True

891 rows × 14 columns

In [15]:
```
df1=df.drop(["embarked","class","who","adult_male","deck","embark_town","alone"], axis=1)
```

In [16]:
```
df1
```

Out[16]:

	survived	pclass	sex	age	sibsp	parch	fare	alive
0	0	3	male	22.0	1	0	7.2500	no
1	1	1	female	38.0	1	0	71.2833	yes
2	1	3	female	26.0	0	0	7.9250	yes
3	1	1	female	35.0	1	0	53.1000	yes
4	0	3	male	35.0	0	0	8.0500	no
...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	no
887	1	1	female	19.0	0	0	30.0000	yes
888	0	3	female	NaN	1	2	23.4500	no
889	1	1	male	26.0	0	0	30.0000	yes
890	0	3	male	32.0	0	0	7.7500	no

891 rows × 8 columns

In [18]:
```
df1["sex"].mode()[0]
```

Out[18]:
```
'male'
```

In [19]:
```
df1["age"].mode()
```

Out[19]:
```
0    24.0
Name: age, dtype: float64
```

In [20]:
```
df1["age"].mean()
```

Out[20]:
```
29.69911764705882
```

In [21]:
```
df1.loc[:, "sex"].mode()
```

Out[21]:
```
0    male
Name: sex, dtype: object
```

In [22]:
```
df1.min()
```

Out[22]:
```
survived         0
pclass           1
sex         female
age           0.42
sibsp            0
parch            0
fare           0.0
alive           no
dtype: object
```

In [23]:
```
bool_series = pd.notnull(df1["sex"])
df1
```

Out[23]:

	survived	pclass	sex	age	sibsp	parch	fare	alive
0	0	3	male	22.0	1	0	7.2500	no
1	1	1	female	38.0	1	0	71.2833	yes
2	1	3	female	26.0	0	0	7.9250	yes
3	1	1	female	35.0	1	0	53.1000	yes
4	0	3	male	35.0	0	0	8.0500	no
...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	no
887	1	1	female	19.0	0	0	30.0000	yes
888	0	3	female	NaN	1	2	23.4500	no
889	1	1	male	26.0	0	0	30.0000	yes
890	0	3	male	32.0	0	0	7.7500	no

891 rows × 8 columns

In [24]:
```
df1.fillna(df1["age"].mean(), inplace=True)
```

In [25]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

In [26]:
```
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
label_encoder = preprocessing.LabelEncoder()
```

In [27]:
```
df1["sex"]= label_encoder.fit_transform(df1["sex"])
df1["sex"].unique()
```

Out[27]:
```
array([1, 0])
```

In [28]:
```
df1
```

Out[28]:

	survived	pclass	sex	age	sibsp	parch	fare	alive
0	0	3	1	22.000000	1	0	7.2500	no
1	1	1	0	38.000000	1	0	71.2833	yes
2	1	3	0	26.000000	0	0	7.9250	yes
3	1	1	0	35.000000	1	0	53.1000	yes
4	0	3	1	35.000000	0	0	8.0500	no
...	...	...	...	...	...	...	...	...
886	0	2	1	27.000000	0	0	13.0000	no
887	1	1	0	19.000000	0	0	30.0000	yes
888	0	3	0	29.699118	1	2	23.4500	no
889	1	1	1	26.000000	0	0	30.0000	yes
890	0	3	1	32.000000	0	0	7.7500	no

891 rows × 8 columns

In [29]:
```
df1["alive"] = label_encoder.fit_transform(df1["alive"])
df1["alive"].unique()
```

Out[29]:
```
array([0, 1])
```

In [30]:
```
df1
```

Out[30]:

	survived	pclass	sex	age	sibsp	parch	fare	alive
0	0	3	1	22.000000	1	0	7.2500	0
1	1	1	0	38.000000	1	0	71.2833	1
2	1	3	0	26.000000	0	0	7.9250	1
3	1	1	0	35.000000	1	0	53.1000	1
4	0	3	1	35.000000	0	0	8.0500	0
...	...	...	...	...	...	...	...	...
886	0	2	1	27.000000	0	0	13.0000	0
887	1	1	0	19.000000	0	0	30.0000	1
888	0	3	0	29.699118	1	2	23.4500	0
889	1	1	1	26.000000	0	0	30.0000	1
890	0	3	1	32.000000	0	0	7.7500	0

891 rows × 8 columns

In [31]:
```
x = df1.drop(["alive"], axis=1)
```

In [32]:
```
y = df1["alive"]
```

In [33]:
```
1 x
```

Out[33]:

	survived	pclass	sex	age	sibsp	parch	fare
0	0	3	1	22.000000	1	0	7.2500
1	1	1	0	38.000000	1	0	71.2833
2	1	3	0	26.000000	0	0	7.9250
3	1	1	0	35.000000	1	0	53.1000
4	0	3	1	35.000000	0	0	8.0500
...	...	...	...	...	...	...	...
886	0	2	1	27.000000	0	0	13.0000
887	1	1	0	19.000000	0	0	30.0000
888	0	3	0	29.699118	1	2	23.4500
889	1	1	1	26.000000	0	0	30.0000
890	0	3	1	32.000000	0	0	7.7500

891 rows × 7 columns

In [34]:
```
1 y
```

Out[34]:
```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: alive, Length: 891, dtype: int32
```

In [35]:
```
1 from sklearn.model_selection import train_test_split
```

In [36]:
```
1 train_x, test_x, train_y, test_y=train_test_split(x,y,test_size= 0.2, random_state=1)
```

In [37]:
```
1 train_x
```

Out[37]:

	survived	pclass	sex	age	sibsp	parch	fare
301	1	3	1	29.699118	2	0	23.2500
309	1	1	0	30.000000	0	0	56.9292
516	1	2	0	34.000000	0	0	10.5000
120	0	2	1	21.000000	2	0	73.5000
570	1	2	1	62.000000	0	0	10.5000
...	...	...	...	...	...	...	...
715	0	3	1	19.000000	0	0	7.6500
767	0	3	0	30.500000	0	0	7.7500
72	0	2	1	21.000000	0	0	73.5000
235	0	3	0	29.699118	0	0	7.5500
37	0	3	1	21.000000	0	0	8.0500

712 rows × 7 columns

In [38]:
```
1 train_y
```

Out[38]:
```
301    1
309    1
516    1
120    0
570    1
      ..
715    0
767    0
72     0
235    0
37     0
Name: alive, Length: 712, dtype: int32
```

In [39]:
```
1 test_x
```

Out[39]:

	survived	pclass	sex	age	sibsp	parch	fare
862	1	1	0	48.000000	0	0	25.9292
223	0	3	1	29.699118	0	0	7.8958
84	1	2	0	17.000000	0	0	10.5000
680	0	3	0	29.699118	0	0	8.1375
535	1	2	0	7.000000	0	2	26.2500
...	...	...	...	...	...	...	...
796	1	1	0	49.000000	0	0	25.9292
815	0	1	1	29.699118	0	0	0.0000
629	0	3	1	29.699118	0	0	7.7333
421	0	3	1	21.000000	0	0	7.7333
448	1	3	0	5.000000	2	1	19.2583

179 rows × 7 columns

In [40]:
```
1 test_y
```

Out[40]:
```
862    1
223    0
84     1
680    0
535    1
      ..
796    1
815    0
629    0
421    0
448    1
Name: alive, Length: 179, dtype: int32
```

In [41]:
```
1 from sklearn.preprocessing import MinMaxScaler
```

In [42]:
```
1 scaler = MinMaxScaler()
2 scaler
```

Out[42]:
```
▾ MinMaxScaler

MinMaxScaler()
```

In [43]:
```
1 train_x_scaled=scaler.fit_transform(train_x)
2 train_x_scaled
```

Out[43]:
```
array([[1.        , 1.        , 1.        , ..., 0.25      , 0.        ,
        0.04538098],
       [1.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.1111184 ],
       [1.        , 0.5       , 0.        , ..., 0.        , 0.        ,
        0.02049464],
       ...,
       [0.        , 0.5       , 1.        , ..., 0.        , 0.        ,
        0.14346245],
       [0.        , 1.        , 0.        , ..., 0.        , 0.        ,
        0.01473662],
       [0.        , 1.        , 1.        , ..., 0.        , 0.        ,
        0.01571255]])
```

In [44]:
```
1 cols = train_x.columns
2 cols
```

Out[44]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare'], dtype='object')
```

In [45]:
```
1 train_x_scaled = scaler.fit_transform(train_x)
2 train_x_scaled
```

Out[45]:
```
array([[1.        , 1.        , 1.        , ..., 0.25      , 0.        ,
        0.04538098],
       [1.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.1111184 ],
       [1.        , 0.5       , 0.        , ..., 0.        , 0.        ,
        0.02049464],
       ...,
       [0.        , 0.5       , 1.        , ..., 0.        , 0.        ,
        0.14346245],
       [0.        , 1.        , 0.        , ..., 0.        , 0.        ,
        0.01473662],
       [0.        , 1.        , 1.        , ..., 0.        , 0.        ,
        0.01571255]])
```

In [48]:
```
train_x_scaled = pd.DataFrame(train_x_scaled, columns=cols)
```

In [49]:
```
train_x_scaled
```

Out[49]:

	survived	pclass	sex	age	sibsp	parch	fare
0	1.0	1.0	1.0	0.367921	0.25	0.0	0.045381
1	1.0	0.0	0.0	0.371701	0.00	0.0	0.111118
2	1.0	0.5	0.0	0.421965	0.00	0.0	0.020495
3	0.0	0.5	1.0	0.258608	0.25	0.0	0.143462
4	1.0	0.5	1.0	0.773813	0.00	0.0	0.020495
...	...	...	...	...	...	...	...
707	0.0	1.0	1.0	0.233476	0.00	0.0	0.014932
708	0.0	1.0	0.0	0.377984	0.00	0.0	0.015127
709	0.0	0.5	1.0	0.258608	0.00	0.0	0.143462
710	0.0	1.0	0.0	0.367921	0.00	0.0	0.014737
711	0.0	1.0	1.0	0.258608	0.00	0.0	0.015713

712 rows × 7 columns

In [50]:
```
from sklearn.naive_bayes import GaussianNB
```

In [51]:
```
gnb= GaussianNB()
gnb.fit(train_x, train_y)
```

Out[51]:
```
GaussianNB()
```

In [52]:
```
train_predict=gnb.predict(train_x)
test_predict= gnb.predict(test_x)
```

In [53]:
```
train_predict
```

Out[53]:
```
array([1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1,
       0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 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, 1, 1, 1, 0,
       1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0])
```

In [54]:
```
test_predict
```

Out[54]:
```
array([1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1])
```

In [59]:
```
%pip install mlxtend

```

```
Defaulting to user installation because normal site-packages is not writeableNote: you may need to restart the kernel to
use updated packages.

Collecting mlxtend
  Obtaining dependency information for mlxtend from https://files.pythonhosted.org/packages/1c/07/512f6a780239ad6ce06ce2a
a7b4067583f5ddcfc7703a964a082c706a070/mlxtend-0.23.1-py3-none-any.whl.metadata (https://files.pythonhosted.org/packages/1
c/07/512f6a780239ad6ce06ce2aa7b4067583f5ddcfc7703a964a082c706a070/mlxtend-0.23.1-py3-none-any.whl.metadata)
  Downloading mlxtend-0.23.1-py3-none-any.whl.metadata (7.3 kB)
Requirement already satisfied: scipy>=1.2.1 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (1.11.1)
Requirement already satisfied: numpy>=1.16.2 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (1.24.3)
Requirement already satisfied: pandas>=0.24.2 in c:\users\admin\appdata\roaming\python\python311\site-packages (from mlxt
end) (2.1.4)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (1.3.0)
Requirement already satisfied: matplotlib>=3.0.0 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (3.7.2)
Requirement already satisfied: joblib>=0.13.2 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (1.2.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->ml
xtend) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxten
d) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->m
lxtend) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->m
lxtend) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlx
tend) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxte
nd) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.
0->mlxtend) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0
->mlxtend) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\site-packages (from pandas>=0.24.2->mlxtend)
(2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\programdata\anaconda3\lib\site-packages (from pandas>=0.24.2->mlxten
d) (2023.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn>=1.
0.2->mlxtend) (2.2.0)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplot
lib>=3.0.0->mlxtend) (1.16.0)
Downloading mlxtend-0.23.1-py3-none-any.whl (1.4 MB)
   ---------------------------------------- 0.0/1.4 MB ? eta -:--:--
   ---------------------------------------- 0.0/1.4 MB ? eta -:--:--
   - -------------------------------------- 0.1/1.4 MB 812.7 kB/s eta 0:00:02
   -- ------------------------------------- 0.1/1.4 MB 1.0 MB/s eta 0:00:02
   ------ --------------------------------- 0.2/1.4 MB 1.7 MB/s eta 0:00:01
   --------- ------------------------------ 0.3/1.4 MB 1.7 MB/s eta 0:00:01
   ------------- -------------------------- 0.5/1.4 MB 1.9 MB/s eta 0:00:01
   ----------------- ---------------------- 0.6/1.4 MB 2.0 MB/s eta 0:00:01
   -------------------- ------------------- 0.7/1.4 MB 2.1 MB/s eta 0:00:01
   ----------------------- ---------------- 0.8/1.4 MB 2.1 MB/s eta 0:00:01
   ------------------------- -------------- 1.0/1.4 MB 2.2 MB/s eta 0:00:01
   ------------------------- -------------- 1.0/1.4 MB 2.2 MB/s eta 0:00:01
   ------------------------------ --------- 1.1/1.4 MB 2.2 MB/s eta 0:00:01
   ---------------------------------- ----- 1.2/1.4 MB 2.1 MB/s eta 0:00:01
   ------------------------------------ --- 1.3/1.4 MB 2.1 MB/s eta 0:00:01
   ---------------------------------------- - 1.4/1.4 MB 2.1 MB/s eta 0:00:01
   ---------------------------------------- 1.4/1.4 MB 2.0 MB/s eta 0:00:00
Installing collected packages: mlxtend
Successfully installed mlxtend-0.23.1
```

In [60]:
```
from mlxtend.plotting import plot_confusion_matrix
```

In [62]:
```
from sklearn.metrics import f1_score, confusion_matrix, roc_auc_score, roc_curve, classification_report, accuracy_scor
```

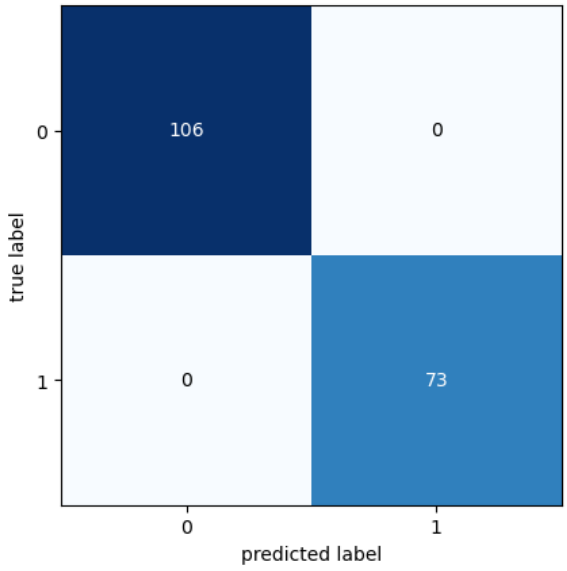
In [63]:
```
accuracy = accuracy_score(test_y, test_predict)
conf_matrix = confusion_matrix(test_y, test_predict)
accuracy
```

Out[63]: 1.0

In [64]:
```
print("Accuracy:", accuracy)
print("Confusion_Matrix: ")
print(conf_matrix)
print("\nClassification Report:")
print(classification_report(test_y, test_predict))
```

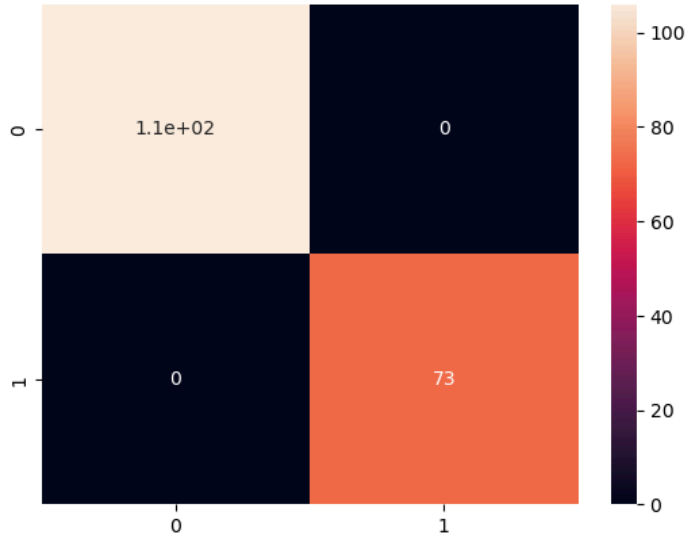
```
Accuracy: 1.0
Confusion_Matrix:
[[106   0]
 [  0  73]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       106
           1       1.00      1.00      1.00        73

    accuracy                           1.00       179
   macro avg       1.00      1.00      1.00       179
weighted avg       1.00      1.00      1.00       179
```

In [65]:
```
fig, ax = plot_confusion_matrix(conf_mat = conf_matrix)
plt.show()
```



In [66]:
```
sb.heatmap(conf_matrix, annot=True)
```

Out[66]: <Axes: >



In [ ]:
```

```