

```
In [13]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [14]: datanames=sns.get_dataset_names()
print(datanames)
```

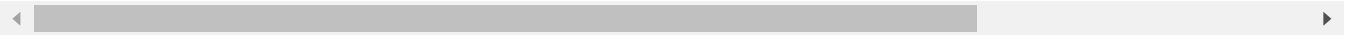
['anagrams', 'anscombe', 'attention', 'brain_networks', 'car_crashes', 'diamonds', 'dots', 'dowjones', 'exercise', 'flights', 'fmri', 'geyser', 'glue', 'healthexp', 'iris', 'mpg', 'penguins', 'planets', 'seaice', 'taxis', 'tips', 'titanic', 'anagrams', 'anagrams', 'anscombe', 'anscombe', 'attention', 'attention', 'brain_networks', 'brain_networks', 'car_crashes', 'car_crashes', 'diamonds', 'diamonds', 'dots', 'dots', 'dowjones', 'dowjones', 'exercise', 'exercise', 'flights', 'flights', 'fmri', 'fmri', 'geyser', 'geyser', 'glue', 'glue', 'healthexp', 'healthexp', 'iris', 'iris', 'mpg', 'mpg', 'penguins', 'penguins', 'planets', 'planets', 'seaice', 'seaice', 'taxis', 'taxis', 'tips', 'tips', 'titanic', 'titanic', 'anagrams', 'anscombe', 'attention', 'brain_networks', 'car_crashes', 'diamonds', 'dots', 'dowjones', 'exercise', 'flights', 'fmri', 'geyser', 'glue', 'healthexp', 'iris', 'mpg', 'penguins', 'planets', 'seaice', 'taxis', 'tips', 'titanic']

```
In [15]: df=sns.load_dataset("titanic")
df
```

Out[15]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True
...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True

891 rows × 15 columns



```
In [16]: df=df.drop('alone',axis=1)
df
```

Out[16]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True
...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True

891 rows × 14 columns

In [17]:

```
df['alive'].replace(['no','yes'],[0,1],inplace=True)
df
```

Out[17]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True
...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True

891 rows × 14 columns

In [20]:

```
from sklearn import preprocessing
enc = preprocessing.OneHotEncoder()
enc_df = pd.DataFrame(enc.fit_transform(df[['sex']]).toarray())
enc_df
```

Out[20]:

	0	1
0	0.0	1.0
1	1.0	0.0
2	1.0	0.0
3	1.0	0.0
4	0.0	1.0
...
886	0.0	1.0
887	1.0	0.0
888	1.0	0.0
889	0.0	1.0
890	0.0	1.0

891 rows × 2 columns

In [21]: df.head()

Out[21]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	de
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Ni
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Ni
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Ni

In [22]: df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 14 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    int64
dtypes: bool(1), category(2), float64(2), int64(5), object(4)
memory usage: 79.8+ KB
```

In [23]: df.describe()

Out[23]:

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

In [24]: df["sex"].value_counts(normalize=True)

Out[24]:

```
sex
male    0.647587
female  0.352413
Name: proportion, dtype: float64
```

In [25]: df["deck"].value_counts(normalize=True)

Out[25]:

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

In [30]: df1

Out[30]:

	survived	pclass	sex	age	sibsp	parch	fare	alive
0	0	3	male	22.0	1	0	7.2500	0
1	1	1	female	38.0	1	0	71.2833	1
2	1	3	female	26.0	0	0	7.9250	1
3	1	1	female	35.0	1	0	53.1000	1
4	0	3	male	35.0	0	0	8.0500	0
...
886	0	2	male	27.0	0	0	13.0000	0
887	1	1	female	19.0	0	0	30.0000	1
888	0	3	female	NaN	1	2	23.4500	0
889	1	1	male	26.0	0	0	30.0000	1
890	0	3	male	32.0	0	0	7.7500	0

891 rows × 8 columns

In [31]: `df1['sex'].mode()[0]`

Out[31]: 'male'

In [32]: `df1['age'].mode()`

Out[32]: 0 24.0
Name: age, dtype: float64

In [33]: `df1['age'].mean()`

Out[33]: 29.69911764705882

In [34]: `df1.loc[:, "sex"].mode()`

Out[34]: 0 male
Name: sex, dtype: object

In [35]: `df1.min()`

Out[35]: survived 0
pclass 1
sex female
age 0.42
sibsp 0
parch 0
fare 0.0
alive 0
dtype: object

In [36]: `boll_series = pd.notnull(df1["sex"])`
`df1`

Out[36]:

	survived	pclass	sex	age	sibsp	parch	fare	alive
0	0	3	male	22.0	1	0	7.2500	0
1	1	1	female	38.0	1	0	71.2833	1
2	1	3	female	26.0	0	0	7.9250	1
3	1	1	female	35.0	1	0	53.1000	1
4	0	3	male	35.0	0	0	8.0500	0
...
886	0	2	male	27.0	0	0	13.0000	0
887	1	1	female	19.0	0	0	30.0000	1
888	0	3	female	NaN	1	2	23.4500	0
889	1	1	male	26.0	0	0	30.0000	1
890	0	3	male	32.0	0	0	7.7500	0

891 rows × 8 columns

```
In [37]: df1.fillna(df1['age'].mean(),inplace=True)
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 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         891 non-null    float64
4   sibsp       891 non-null    int64
5   parch       891 non-null    int64
6   fare        891 non-null    float64
7   alive       891 non-null    int64
dtypes: float64(2), int64(5), object(1)
memory usage: 55.8+ KB
```

```
In [38]: from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
label_encoder = preprocessing.LabelEncoder()
```

```
In [39]: df1['sex']=label_encoder.fit_transform(df1['sex'])
df1['sex'].unique()
```

Out[39]: array([1, 0])

In [40]: df1

Out[40]:

	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 [41]: `df1['alive']=label_encoder.fit_transform(df1['alive'])`
`df1['alive'].unique()`

Out[41]: `array([0, 1], dtype=int64)`

In [42]: `df1`

Out[42]:

	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 [43]: `x=df1.drop(['alive'],axis=1)`

In [44]: `y=df1['alive']`

In [45]: `x`

Out[45]:

	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 [46]:

y

Out[46]:

```

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: int64

In [47]:

```

from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(x, y, test_size=0.2, random_state=1)
train_x

```


Out[47]:

	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 [48]: train_y

Out[48]:

3011

3091

5161

1200

5701

..

7150

7670

720

2350

370

Name: alive, Length: 712, dtype: int64

In [49]: test_x

Out[49]:

	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 [50]:

test_y

Out[50]:

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: int64

In [51]:

from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
scaler

Out[51]:

▼ MinMaxScaler
MinMaxScaler()

In [52]:

train_x_scaled=scaler.fit_transform(train_x)
train_x_scaled

```
Out[52]: 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 [53]: cols=train_x.columns
cols
```

```
Out[53]: Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare'], dtype='object')
```

```
In [54]: train_x_scaled=scaler.fit_transform(train_x)
train_x_scaled
```

```
Out[54]: 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 [55]: train_x_scaled=pd.DataFrame(train_x_scaled,columns=cols)
train_x_scaled
```

Out[55]:

	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 [56]: from sklearn.naive_bayes import GaussianNB
```

```
In [57]: gnb = GaussianNB()
         gnb.fit(train_x,train_y)
```

```
Out[57]: ▼ GaussianNB
         GaussianNB()
```

```
In [58]: train_predict=gnb.predict(train_x)
         test_predict=gnb.predict(test_x)
```

```
In [59]: train_predict
```

```
Out[59]: 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, 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, 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, 1, 0, 0, 1, 0, 0, 1, 0, 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, 1, 0, 0, 0, 1, 1, 1, 0,
                0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
                0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0,
                0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1,
                0, 0, 0, 1, 1, 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, 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, 1, 0, 0, 0,
                0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
                0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 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, 1, 0, 0, 0, 0,
                0, 0, 1, 0, 0, 0, 0, 0], dtype=int64)
```

```
In [60]: test_predict
```

```
Out[60]: 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, 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, 0, 0, 0, 1, 1, 1, 1, 0, 1, 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,
                0, 0, 1], dtype=int64)
```

```
In [75]: from mlxtend.plotting import plot_confusion_matrix
```

In [74]: `pip install mlxtend`

```

Defaulting to user installation because normal site-packages is not writeable
Collecting mlxtend
  Obtaining dependency information for mlxtend from https://files.pythonhosted.org/packages/1c/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:\programdata\anaconda3\lib\site-packages (from mlxtend) (2.0.3)
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->mlxtend) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (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->mlxtend) (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->matplotlib>=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 --:--:--
- ----- 0.0/1.4 MB 653.6 kB/s eta 0:00:03
----- 0.5/1.4 MB 4.4 MB/s eta 0:00:01
----- 1.0/1.4 MB 7.0 MB/s eta 0:00:01
----- 1.4/1.4 MB 7.6 MB/s eta 0:00:01
----- 1.4/1.4 MB 7.7 MB/s eta 0:00:00

Installing collected packages: mlxtend
Successfully installed mlxtend-0.23.1
Note: you may need to restart the kernel to use updated packages.

```

In [69]: `from sklearn.metrics import f1_score, confusion_matrix, roc_auc_score, roc_curve, clas`

In [70]: `accuracy = accuracy_score(test_y, test_predict)
conf_matrix = confusion_matrix(test_y, test_predict)
accuracy`

Out[70]: 1.0

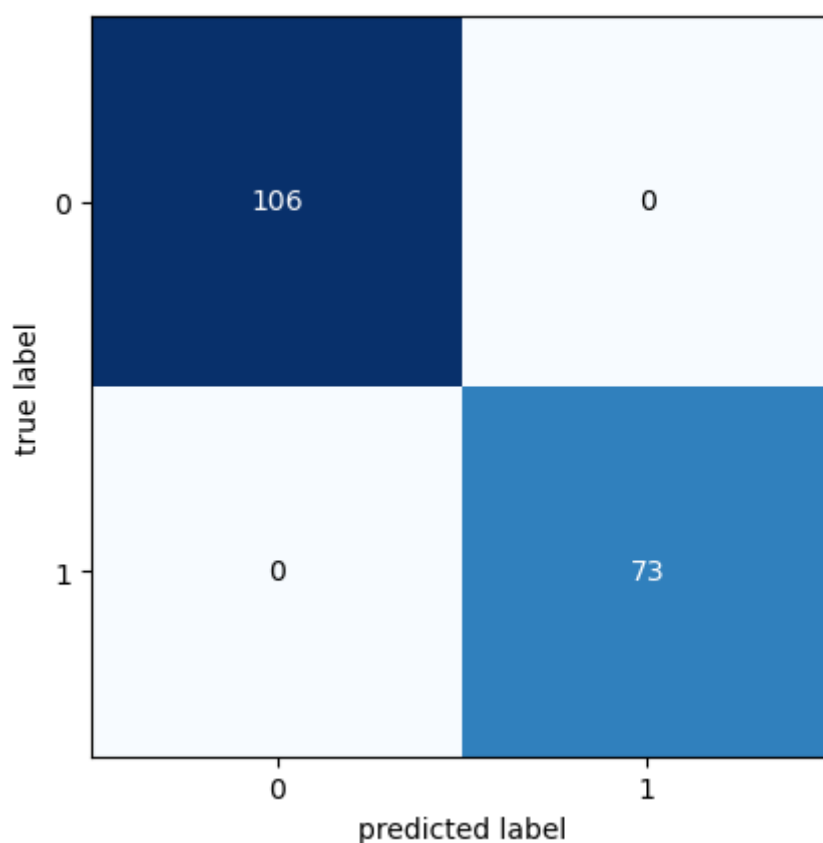
```
In [71]: print("Accuracy:",accuracy)
print("Confusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(classification_report(test_y,test_predict))
```

Accuracy: 1.0
Confusion Matrix:
[[106 0]
 [0 73]]

Classification Report:

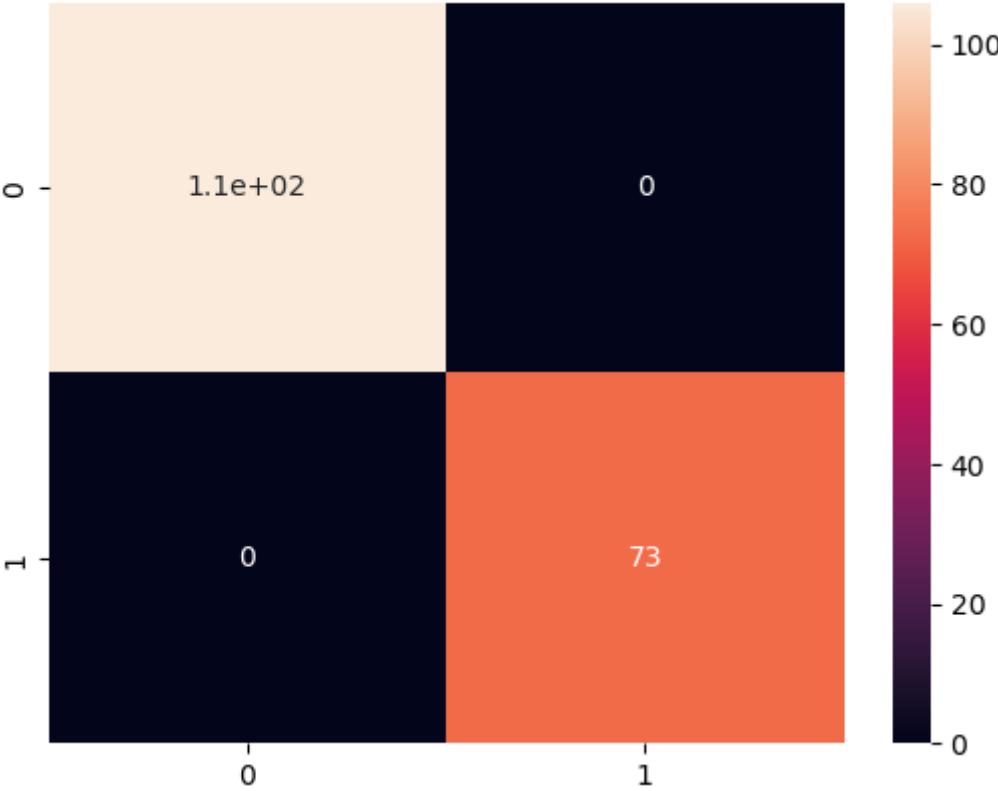
	precision	recall	f1-score	support
0	1.00	1.00	1.00	106
1	1.00	1.00	1.00	73
accuracy			1.00	179
macro avg	1.00	1.00	1.00	179
weighted avg	1.00	1.00	1.00	179

```
In [76]: fig, ax = plot_confusion_matrix(conf_mat=conf_matrix)
plt.show()
```



```
In [77]: import seaborn as sns
sns.heatmap(conf_matrix,annot=True)
```

Out[77]: <Axes: >



```
In [ ]: Name = Anurag Jadhav (13171)
```