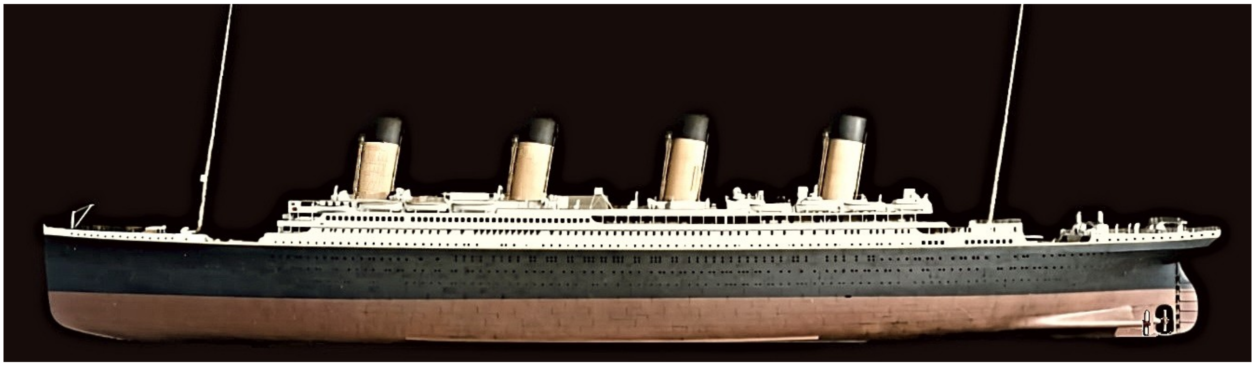
TITANIC SURVIVAL PREDICTION



About the dataset

The dataset contains information about individual passengers such as their age, gender, ticket class, fare, cabin, and whether or not they survived.

IMPORTING PYTHON LIBRARIES

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder,MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.svm import SVC
from sklearn.metrics import
classification_report,confusion_matrix,ConfusionMatrixDisplay,accuracy
_score
```

LOADING DATASET

```
df=pd.read_csv('/home/anusha/Desktop/tested.csv')
df

     PassengerId  Survived  Pclass  \
0            892         0       3
1            893         1       3
2            894         0       2
3            895         0       3
4            896         1       3
..           ...       ...     ...
413         1305         0       3
414         1306         1       1
```

```
415         1307         0       3
416         1308         0       3
417         1309         0       3

                                             Name     Sex   Age  SibSp
Parch  \
0                                Kelly, Mr. James    male  34.5      0
0
1                Wilkes, Mrs. James (Ellen Needs)  female  47.0      1
0
2                       Myles, Mr. Thomas Francis    male  62.0      0
0
3                                Wirz, Mr. Albert    male  27.0      0
0
4    Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female  22.0      1
1
..                                            ...     ...   ...    ...
...
413                            Spector, Mr. Woolf    male   NaN      0
0
414                  Oliva y Ocana, Dona. Fermina  female  39.0      0
0
415                  Saether, Mr. Simon Sivertsen    male  38.5      0
0
416                           Ware, Mr. Frederick    male   NaN      0
0
417                      Peter, Master. Michael J    male   NaN      1
1

                 Ticket      Fare Cabin Embarked
0                330911    7.8292   NaN        Q
1                363272    7.0000   NaN        S
2                240276    9.6875   NaN        Q
3                315154    8.6625   NaN        S
4               3101298   12.2875   NaN        S
..                  ...       ...   ...      ...
413           A.5. 3236    8.0500   NaN        S
414            PC 17758  108.9000  C105        C
415  SOTON/O.Q. 3101262    7.2500   NaN        S
416              359309    8.0500   NaN        S
417                2668   22.3583   NaN        C

[418 rows x 12 columns]
```

DATA PREPROCESSING

```
df.head()
```

```
   PassengerId  Survived  Pclass  \
0          892         0       3
1          893         1       3
2          894         0       2
3          895         0       3
4          896         1       3

                                           Name     Sex   Age  SibSp
Parch  \
0                              Kelly, Mr. James    male  34.5      0
0
1              Wilkes, Mrs. James (Ellen Needs)  female  47.0      1
0
2                     Myles, Mr. Thomas Francis    male  62.0      0
0
3                              Wirz, Mr. Albert    male  27.0      0
0
4  Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female  22.0      1
1

    Ticket     Fare Cabin Embarked
0   330911   7.8292   NaN        Q
1   363272   7.0000   NaN        S
2   240276   9.6875   NaN        Q
3   315154   8.6625   NaN        S
4  3101298  12.2875   NaN        S
```

```
df.tail()
```

```
     PassengerId  Survived  Pclass                          Name
Sex  \
413         1305         0       3            Spector, Mr. Woolf
male
414         1306         1       1  Oliva y Ocana, Dona. Fermina
female
415         1307         0       3  Saether, Mr. Simon Sivertsen
male
416         1308         0       3           Ware, Mr. Frederick
male
417         1309         0       3      Peter, Master. Michael J
male

      Age  SibSp  Parch              Ticket      Fare Cabin Embarked
413   NaN      0      0           A.5. 3236    8.0500   NaN        S
414  39.0      0      0            PC 17758  108.9000  C105        C
415  38.5      0      0  SOTON/O.Q. 3101262    7.2500   NaN        S
416   NaN      0      0              359309    8.0500   NaN        S
417   NaN      1      1                2668   22.3583   NaN        C
```

```
df.shape
```

```
(418, 12)
```

```
df.columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age',
'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

```
df.dtypes
```

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

The columns Name,Sex,Ticket,Cabin,Embarked are categorical datas.

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

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

Missing values are found in Age and Cabin.So we can drop Cabin column.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
```

```
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Survived     418 non-null    int64  
 2   Pclass       418 non-null    int64  
 3   Name         418 non-null    object 
 4   Sex          418 non-null    object 
 5   Age          332 non-null    float64
 6   SibSp        418 non-null    int64  
 7   Parch        418 non-null    int64  
 8   Ticket       418 non-null    object 
 9   Fare         417 non-null    float64
 10  Cabin        91 non-null     object 
 11  Embarked     418 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 39.3+ KB
```

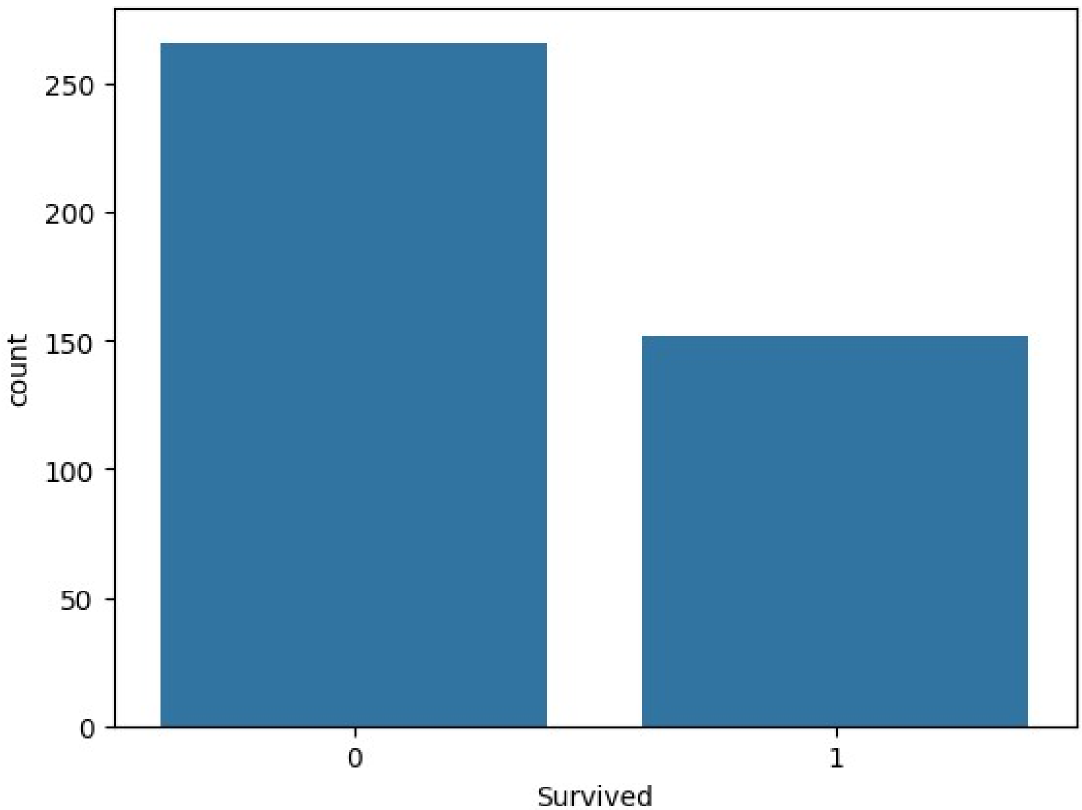
```
df['Survived'].value_counts()
```

```
Survived
0    266
1    152
Name: count, dtype: int64
```

It is a balanced dataset

DATA VISUALIZATION

```
sns.countplot(x=df['Survived'])
plt.show()
```

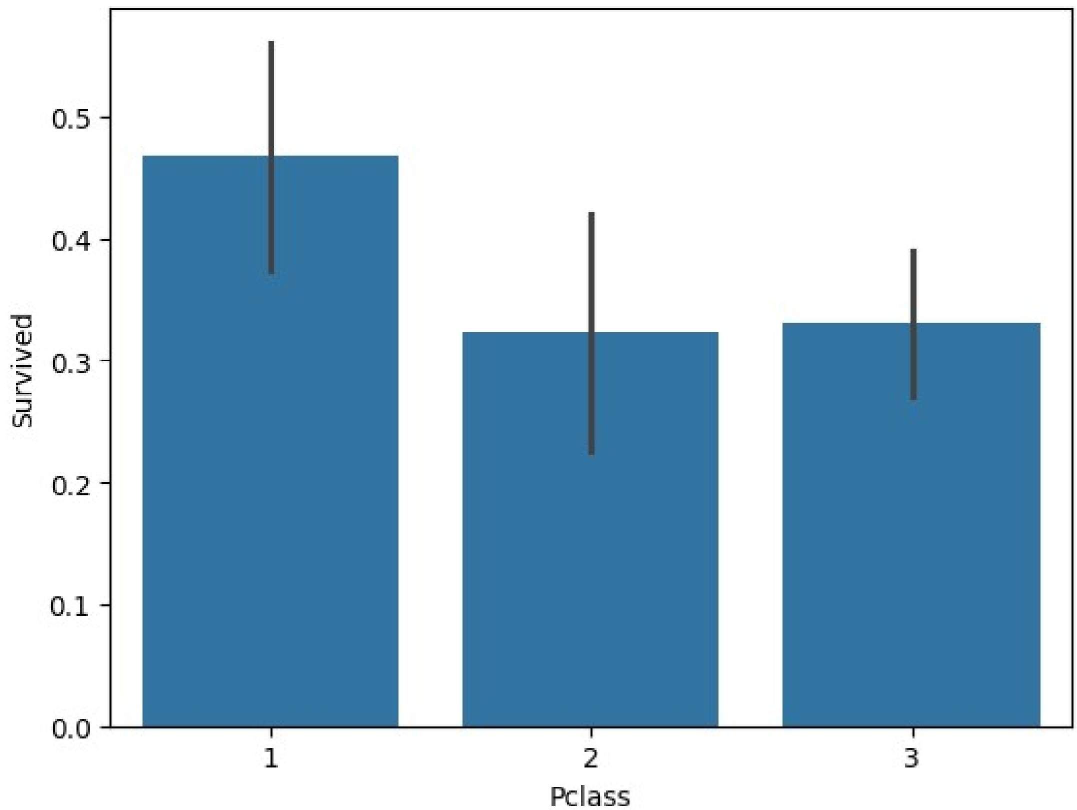



```
df['Pclass'].value_counts()
```

```
Pclass
3    218
1    107
2     93
Name: count, dtype: int64
```

```
sns.barplot(x='Pclass',y='Survived',data=df)
```

```
<Axes: xlabel='Pclass', ylabel='Survived'>
```

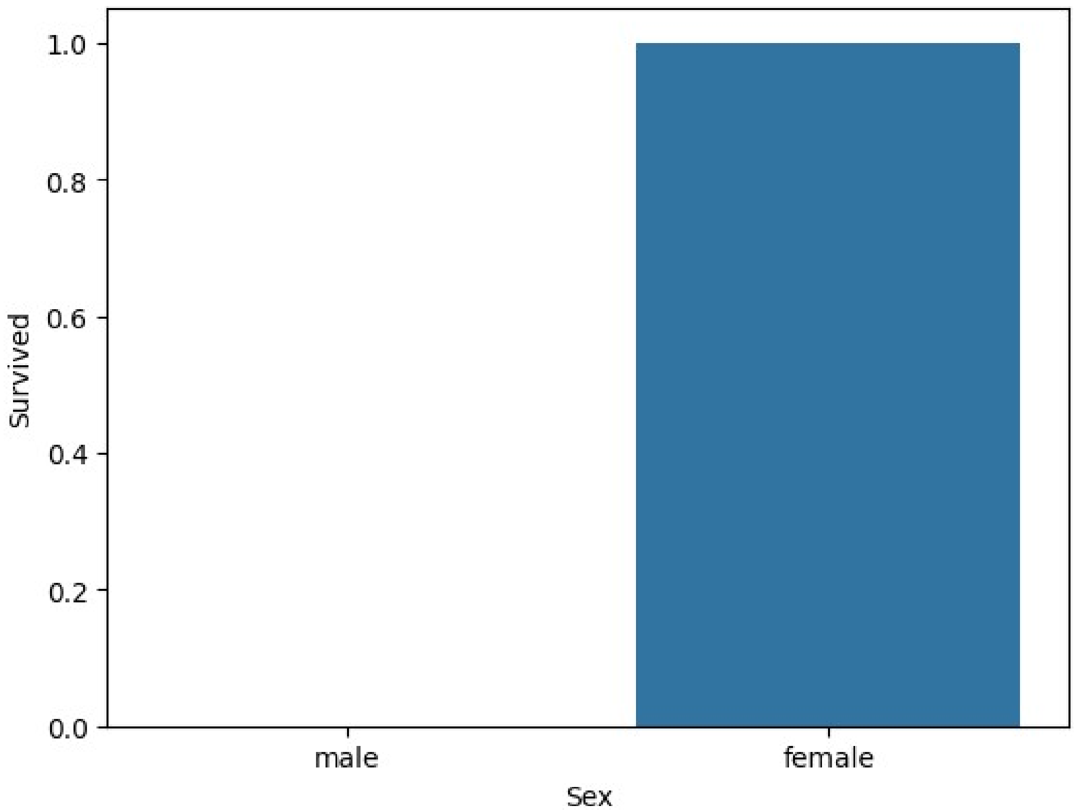



```
df['Sex'].value_counts()
```

```
Sex
male      266
female    152
Name: count, dtype: int64
```

```
sns.barplot(x='Sex',y='Survived',data=df)
```

```
<Axes: xlabel='Sex', ylabel='Survived'>
```




```
df['SibSp'].value_counts()
```

```
SibSp
0    283
1    110
2     14
3      4
4      4
8      2
5      1
Name: count, dtype: int64
```

```
df['Parch'].value_counts()
```

```
Parch
0    324
1     52
2     33
3      3
4      2
9      2
6      1
5      1
Name: count, dtype: int64
```

```
df['Ticket'].value_counts()
```

```
Ticket
PC 17608    5
CA. 2343    4
113503      4
PC 17483    3
220845      3
           ..
349226      1
2621        1
4133        1
113780      1
2668        1
Name: count, Length: 363, dtype: int64
```

Ticket can be dropped

```
df['Fare'].value_counts()
```

```
Fare
7.7500     21
26.0000    19
13.0000    17
8.0500     17
7.8958     11
           ..
7.8208      1
8.5167      1
78.8500     1
52.0000     1
22.3583     1
Name: count, Length: 169, dtype: int64
```

```
df['Cabin'].value_counts()
```

```
Cabin
B57 B59 B63 B66    3
B45                2
C89                2
C55 C57            2
A34                2
                  ..
E52                1
D30                1
E31                1
C62 C64            1
C105               1
Name: count, Length: 76, dtype: int64
```

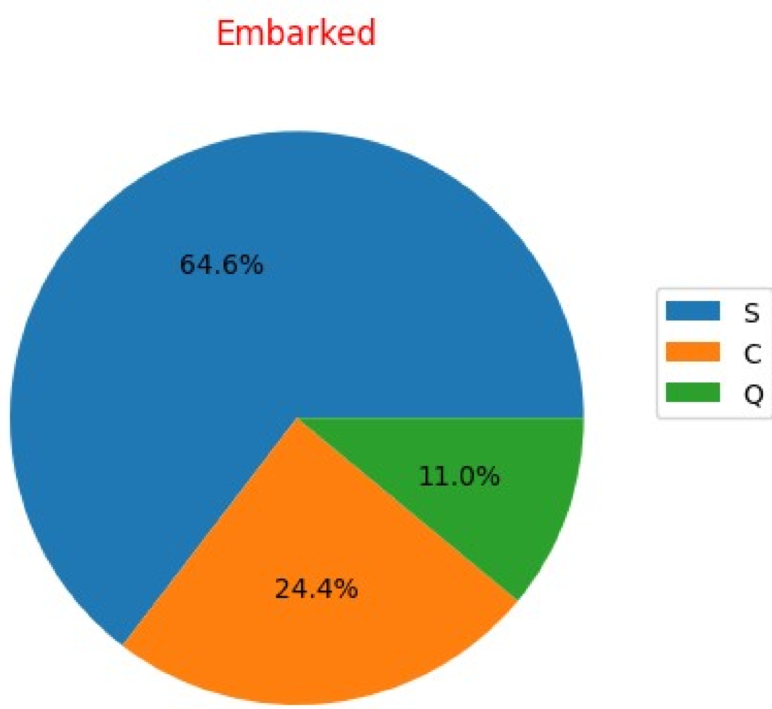
Cabin can be dropped

```
df['Embarked'].value_counts()
```

```
Embarked
S    270
C    102
Q     46
Name: count, dtype: int64
```

```
plt.pie(df['Embarked'].value_counts(),autopct='%1.1f%%')
plt.legend(df['Embarked'].value_counts().index,loc=(1, 0.5))
plt.title('Embarked',color='red')
```

```
Text(0.5, 1.0, 'Embarked')
```



MISSING VALUE HANDLING

```
df['Age']=df['Age'].fillna(df['Age'].mean())
df['Fare']=df['Fare'].fillna(df['Fare'].mean())
df.isna().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
```

```
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin        327
Embarked       0
dtype: int64
```

ENCODING

Machine learning models can only work with numerical values. For this reason, it is necessary to transform the categorical values of the relevant features into numerical ones. This process is called encoding

```
lab=LabelEncoder()
df['Sex']=lab.fit_transform(df['Sex'])
df['Embarked']=lab.fit_transform(df['Embarked'])

df.dtypes

PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex              int64
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked         int64
dtype: object
```

DROPPING UNWANTED COLUMNS

```
#PassengerId,Name,Ticket,Cabin are dropping
df.drop(['PassengerId','Name','Ticket','Cabin'],axis=1,inplace=True)
df.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	34.5	0	0	7.8292	1
1	1	3	0	47.0	1	0	7.0000	2
2	0	2	1	62.0	0	0	9.6875	1
3	0	3	1	27.0	0	0	8.6625	2
4	1	3	0	22.0	1	1	12.2875	2

```python
#correlation
df.corr()
```

```
          Survived    Pclass       Sex       Age     SibSp  
Parch  \
Survived  1.000000 -0.108615 -1.000000 -0.000011  0.099943  0.159120   

Pclass   -0.108615  1.000000  0.108615 -0.440782  0.001087  0.018721   

Sex      -1.000000  0.108615  1.000000  0.000011 -0.099943 -0.159120   

Age      -0.000011 -0.440782  0.000011  1.000000 -0.079535 -0.045073   

SibSp     0.099943  0.001087 -0.099943 -0.079535  1.000000  0.306895   

Parch     0.159120  0.018721 -0.159120 -0.045073  0.306895  1.000000   

Fare      0.191382 -0.576619 -0.191382  0.326800  0.171488  0.230001   

Embarked -0.076281  0.227983  0.076281 -0.157996  0.052708  0.054577   


              Fare  Embarked  
Survived  0.191382 -0.076281  
Pclass   -0.576619  0.227983  
Sex      -0.191382  0.076281  
Age       0.326800 -0.157996  
SibSp     0.171488  0.052708  
Parch     0.230001  0.054577  
Fare      1.000000 -0.257031  
Embarked -0.257031  1.000000  
```

SEPERATE X AND Y

```python
#x as input variable
x=df.drop('Survived',axis=1).values
x
```

```
array([[ 3.        ,  1.        , 34.5       , ...,  0.        ,
         7.8292    ,  1.        ],
       [ 3.        ,  0.        , 47.        , ...,  0.        ,
         7.        ,  2.        ],
       [ 2.        ,  1.        , 62.        , ...,  0.        ,
         9.6875    ,  1.        ],
       ...,
       [ 3.        ,  1.        , 38.5       , ...,  0.        ,
         7.25      ,  2.        ],
       [ 3.        ,  1.        , 30.27259036, ...,  0.        ,
         8.05      ,  2.        ],
```

```
       [ 3.        ,  1.        , 30.27259036, ...,  1.        ,
        22.3583    ,  0.        ]])
```

```python
#y as output variable
y=df['Survived'].values
y
```

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

TRAIN TEST SPLIT

A train_test_split function is used for spliting the datasets into a training set and a testing set. The training set is used for training the model, and the testing set is used to testing the model.

This allows us to train the models on the training set, and then test their accuracy on the unseen testing set.

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,rand
om_state=42)
x_train
```

```
array([[  1.        ,   1.        ,  36.        , ...,   0.        ,
         75.2417    ,   0.        ],
       [  3.        ,   1.        ,  30.27259036, ...,   0.        ,
          7.75      ,   1.        ],
       [  1.        ,   0.        ,  63.        , ...,   0.        ,
        221.7792    ,   2.        ],
       ...,
       [  1.        ,   1.        ,  46.        , ...,   0.        ,
         75.2417    ,   0.        ],
       [  2.        ,   1.        ,  24.        , ...,   0.        ,
         13.5       ,   2.        ],
       [  3.        ,   1.        ,  30.27259036, ...,   0.        ,
          7.75      ,   1.        ]])
```

```
x_test
```

```
array([[3.00000000e+00, 1.00000000e+00, 2.50000000e+01,
0.00000000e+00,
        0.00000000e+00, 7.22920000e+00, 0.00000000e+00],
       [1.00000000e+00, 0.00000000e+00, 3.90000000e+01,
0.00000000e+00,
        0.00000000e+00, 2.11337500e+02, 2.00000000e+00],
       [3.00000000e+00, 1.00000000e+00, 2.10000000e+01,
0.00000000e+00,
        0.00000000e+00, 7.75000000e+00, 1.00000000e+00],
       [3.00000000e+00, 1.00000000e+00, 3.50000000e+01,
0.00000000e+00,
        0.00000000e+00, 7.89580000e+00, 2.00000000e+00],
       [3.00000000e+00, 0.00000000e+00, 3.60000000e+01,
0.00000000e+00,
        2.00000000e+00, 1.21833000e+01, 2.00000000e+00],
       [2.00000000e+00, 1.00000000e+00, 5.00000000e+01,
1.00000000e+00,
        0.00000000e+00, 2.60000000e+01, 2.00000000e+00],
       [3.00000000e+00, 0.00000000e+00, 2.90000000e+01,
0.00000000e+00,
        0.00000000e+00, 7.92500000e+00, 2.00000000e+00],
       [1.00000000e+00, 1.00000000e+00, 4.90000000e+01,
0.00000000e+00,
        0.00000000e+00, 2.60000000e+01, 2.00000000e+00],
       [2.00000000e+00, 0.00000000e+00, 1.90000000e+01,
0.00000000e+00,
        0.00000000e+00, 1.30000000e+01, 2.00000000e+00],
```

```
        [3.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 8.05000000e+00, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 2.10000000e+01,
2.00000000e+00,
         0.00000000e+00, 2.41500000e+01, 2.00000000e+00],
        [1.00000000e+00, 0.00000000e+00, 5.10000000e+01,
0.00000000e+00,
         1.00000000e+00, 3.94000000e+01, 2.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 1.60000000e+01,
1.00000000e+00,
         1.00000000e+00, 8.51670000e+00, 0.00000000e+00],
        [1.00000000e+00, 0.00000000e+00, 3.90000000e+01,
0.00000000e+00,
         0.00000000e+00, 1.08900000e+02, 0.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 8.05000000e+00, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 5.64958000e+01, 2.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 2.80000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.77500000e+00, 2.00000000e+00],
        [1.00000000e+00, 1.00000000e+00, 5.50000000e+01,
0.00000000e+00,
         0.00000000e+00, 5.00000000e+01, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 1.00000000e+01,
4.00000000e+00,
         1.00000000e+00, 2.91250000e+01, 1.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 2.30000000e+01,
1.00000000e+00,
         0.00000000e+00, 1.05000000e+01, 2.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 5.70000000e+01,
0.00000000e+00,
         0.00000000e+00, 1.30000000e+01, 2.00000000e+00],
        [1.00000000e+00, 1.00000000e+00, 4.10000000e+01,
1.00000000e+00,
         0.00000000e+00, 5.18625000e+01, 2.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 3.00000000e+00,
1.00000000e+00,
         1.00000000e+00, 1.37750000e+01, 2.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 3.00000000e+01,
0.00000000e+00,
         0.00000000e+00, 1.30000000e+01, 2.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         2.00000000e+00, 1.52458000e+01, 0.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 1.85000000e+01,
```

```
0.00000000e+00,
         0.00000000e+00, 7.28330000e+00, 1.00000000e+00],
        [1.00000000e+00, 0.00000000e+00, 2.50000000e+01,
1.00000000e+00,
         0.00000000e+00, 5.54417000e+01, 0.00000000e+00],
        [1.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 2.65500000e+01, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 3.90000000e+01,
0.00000000e+00,
         2.00000000e+00, 7.22920000e+00, 0.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 3.00000000e+01,
0.00000000e+00,
         0.00000000e+00, 1.30000000e+01, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 3.20000000e+01,
0.00000000e+00,
         0.00000000e+00, 2.25250000e+01, 2.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 2.20000000e+01,
0.00000000e+00,
         0.00000000e+00, 3.96875000e+01, 2.00000000e+00],
        [1.00000000e+00, 0.00000000e+00, 3.30000000e+01,
0.00000000e+00,
         0.00000000e+00, 1.51550000e+02, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 8.05000000e+00, 2.00000000e+00],
        [2.00000000e+00, 0.00000000e+00, 2.20000000e+01,
0.00000000e+00,
         0.00000000e+00, 1.05000000e+01, 2.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 2.50000000e+01,
0.00000000e+00,
         0.00000000e+00, 1.05000000e+01, 2.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 2.40000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.75000000e+00, 1.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 1.50458000e+01, 0.00000000e+00],
        [2.00000000e+00, 0.00000000e+00, 2.90000000e+01,
1.00000000e+00,
         0.00000000e+00, 2.60000000e+01, 2.00000000e+00],
        [1.00000000e+00, 1.00000000e+00, 3.25000000e+01,
0.00000000e+00,
         0.00000000e+00, 2.11500000e+02, 0.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 2.40000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.75000000e+00, 1.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 3.02725904e+01,
1.00000000e+00,
```

```
        2.00000000e+00, 2.34500000e+01, 2.00000000e+00],
       [2.00000000e+00, 1.00000000e+00, 4.10000000e+01,
0.00000000e+00,
        0.00000000e+00, 1.50458000e+01, 0.00000000e+00],
       [1.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
        0.00000000e+00, 3.96000000e+01, 2.00000000e+00],
       [1.00000000e+00, 1.00000000e+00, 2.85000000e+01,
0.00000000e+00,
        0.00000000e+00, 2.77208000e+01, 0.00000000e+00],
       [3.00000000e+00, 1.00000000e+00, 2.10000000e+01,
0.00000000e+00,
        0.00000000e+00, 7.85420000e+00, 2.00000000e+00],
       [3.00000000e+00, 0.00000000e+00, 3.02725904e+01,
1.00000000e+00,
        9.00000000e+00, 6.95500000e+01, 2.00000000e+00],
       [2.00000000e+00, 0.00000000e+00, 2.40000000e+01,
1.00000000e+00,
        0.00000000e+00, 2.77208000e+01, 0.00000000e+00],
       [1.00000000e+00, 0.00000000e+00, 5.50000000e+01,
2.00000000e+00,
        0.00000000e+00, 2.57000000e+01, 2.00000000e+00],
       [3.00000000e+00, 1.00000000e+00, 3.60000000e+01,
0.00000000e+00,
        0.00000000e+00, 7.25000000e+00, 2.00000000e+00],
       [3.00000000e+00, 1.00000000e+00, 3.45000000e+01,
0.00000000e+00,
        0.00000000e+00, 7.82920000e+00, 1.00000000e+00],
       [3.00000000e+00, 0.00000000e+00, 4.50000000e+01,
0.00000000e+00,
        0.00000000e+00, 7.22500000e+00, 0.00000000e+00],
       [3.00000000e+00, 0.00000000e+00, 3.02725904e+01,
0.00000000e+00,
        0.00000000e+00, 8.05000000e+00, 2.00000000e+00],
       [2.00000000e+00, 1.00000000e+00, 4.70000000e+01,
0.00000000e+00,
        0.00000000e+00, 1.05000000e+01, 2.00000000e+00],
       [2.00000000e+00, 1.00000000e+00, 2.90000000e+01,
0.00000000e+00,
        0.00000000e+00, 1.38583000e+01, 0.00000000e+00],
       [2.00000000e+00, 0.00000000e+00, 2.00000000e+01,
1.00000000e+00,
        0.00000000e+00, 2.60000000e+01, 2.00000000e+00],
       [3.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
        0.00000000e+00, 8.05000000e+00, 2.00000000e+00],
       [2.00000000e+00, 1.00000000e+00, 2.60000000e+01,
0.00000000e+00,
        0.00000000e+00, 1.30000000e+01, 2.00000000e+00],
```

```
        [3.00000000e+00, 0.00000000e+00, 1.70000000e-01,
1.00000000e+00,
         2.00000000e+00, 2.05750000e+01, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 2.40000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.25000000e+00, 1.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 5.00000000e+01,
1.00000000e+00,
         0.00000000e+00, 1.45000000e+01, 2.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 2.10000000e+01,
0.00000000e+00,
         0.00000000e+00, 1.15000000e+01, 2.00000000e+00],
        [1.00000000e+00, 0.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 2.77208000e+01, 0.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 4.00000000e+01,
1.00000000e+00,
         0.00000000e+00, 2.60000000e+01, 2.00000000e+00],
        [2.00000000e+00, 0.00000000e+00, 1.50000000e+01,
0.00000000e+00,
         2.00000000e+00, 3.90000000e+01, 2.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 2.50000000e+01,
0.00000000e+00,
         0.00000000e+00, 1.05000000e+01, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 4.10000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.85000000e+00, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 2.40000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.55000000e+00, 2.00000000e+00],
        [1.00000000e+00, 1.00000000e+00, 3.90000000e+01,
0.00000000e+00,
         0.00000000e+00, 2.97000000e+01, 0.00000000e+00],
        [1.00000000e+00, 1.00000000e+00, 4.30000000e+01,
1.00000000e+00,
         0.00000000e+00, 2.77208000e+01, 0.00000000e+00],
        [1.00000000e+00, 1.00000000e+00, 5.70000000e+01,
1.00000000e+00,
         0.00000000e+00, 1.46520800e+02, 0.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 3.00000000e+01,
1.00000000e+00,
         0.00000000e+00, 2.10000000e+01, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 3.20000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.57920000e+00, 2.00000000e+00],
        [1.00000000e+00, 0.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 3.16833000e+01, 2.00000000e+00],
        [1.00000000e+00, 1.00000000e+00, 4.50000000e+01,
```

```
0.00000000e+00,
         0.00000000e+00, 2.97000000e+01, 0.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 2.20000000e+01,
2.00000000e+00,
         0.00000000e+00, 8.66250000e+00, 2.00000000e+00],
        [1.00000000e+00, 1.00000000e+00, 3.00000000e+01,
0.00000000e+00,
         0.00000000e+00, 2.60000000e+01, 2.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 2.20000000e+01,
1.00000000e+00,
         0.00000000e+00, 1.39000000e+01, 2.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 2.60000000e+01,
1.00000000e+00,
         1.00000000e+00, 2.20250000e+01, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 2.50000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.65000000e+00, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 2.20000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.79580000e+00, 2.00000000e+00],
        [1.00000000e+00, 0.00000000e+00, 4.80000000e+01,
1.00000000e+00,
         3.00000000e+00, 2.62375000e+02, 0.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 2.10000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.22500000e+00, 0.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 1.80000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.87920000e+00, 1.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 3.20000000e+01,
0.00000000e+00,
         0.00000000e+00, 1.30000000e+01, 2.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 2.40000000e+01,
2.00000000e+00,
         0.00000000e+00, 3.15000000e+01, 2.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 1.07083000e+01, 1.00000000e+00],
        [1.00000000e+00, 1.00000000e+00, 2.40000000e+01,
1.00000000e+00,
         0.00000000e+00, 8.22667000e+01, 2.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 1.90000000e+01,
0.00000000e+00,
         0.00000000e+00, 1.05000000e+01, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 7.05000000e+00, 2.00000000e+00],
        [1.00000000e+00, 1.00000000e+00, 2.50000000e+01,
0.00000000e+00,
```

```
         0.00000000e+00, 2.60000000e+01, 0.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 6.00000000e+00,
3.00000000e+00,
         1.00000000e+00, 2.10750000e+01, 2.00000000e+00],
        [1.00000000e+00, 1.00000000e+00, 4.60000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.92000000e+01, 0.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 1.40000000e+01,
0.00000000e+00,
         0.00000000e+00, 9.22500000e+00, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 2.50000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.92500000e+00, 2.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 1.00000000e+01,
5.00000000e+00,
         2.00000000e+00, 4.69000000e+01, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 3.10000000e+01,
3.00000000e+00,
         0.00000000e+00, 1.80000000e+01, 2.00000000e+00],
        [1.00000000e+00, 1.00000000e+00, 1.70000000e+01,
0.00000000e+00,
         0.00000000e+00, 4.71000000e+01, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 7.75000000e+00, 1.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 3.50000000e+01,
0.00000000e+00,
         0.00000000e+00, 1.23500000e+01, 1.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 2.40000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.77500000e+00, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 7.75000000e+00, 1.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 2.30000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.05000000e+00, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 2.70000000e+01,
0.00000000e+00,
         0.00000000e+00, 8.66250000e+00, 2.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 2.70000000e+01,
1.00000000e+00,
         0.00000000e+00, 7.92500000e+00, 2.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         4.00000000e+00, 2.54667000e+01, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 1.70000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.89580000e+00, 2.00000000e+00],
```

```
        [2.00000000e+00, 1.00000000e+00, 2.30000000e+01,
0.00000000e+00,
         0.00000000e+00, 1.05000000e+01, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 7.22500000e+00, 0.00000000e+00],
        [3.00000000e+00, 0.00000000e+00, 2.20000000e+01,
0.00000000e+00,
         0.00000000e+00, 7.72500000e+00, 1.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 7.89580000e+00, 2.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 6.10000000e+01,
0.00000000e+00,
         0.00000000e+00, 1.23500000e+01, 1.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 1.15000000e+01,
1.00000000e+00,
         1.00000000e+00, 1.45000000e+01, 2.00000000e+00],
        [1.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 0.00000000e+00, 2.00000000e+00],
        [1.00000000e+00, 1.00000000e+00, 6.00000000e+00,
0.00000000e+00,
         2.00000000e+00, 1.34500000e+02, 0.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 6.43750000e+00, 0.00000000e+00],
        [1.00000000e+00, 0.00000000e+00, 2.20000000e+01,
0.00000000e+00,
         1.00000000e+00, 6.19792000e+01, 0.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 2.60000000e+01,
1.00000000e+00,
         1.00000000e+00, 2.90000000e+01, 2.00000000e+00],
        [2.00000000e+00, 0.00000000e+00, 9.20000000e-01,
1.00000000e+00,
         2.00000000e+00, 2.77500000e+01, 2.00000000e+00],
        [2.00000000e+00, 0.00000000e+00, 2.20000000e+01,
0.00000000e+00,
         0.00000000e+00, 2.10000000e+01, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 3.02725904e+01,
0.00000000e+00,
         0.00000000e+00, 8.71250000e+00, 2.00000000e+00],
        [2.00000000e+00, 1.00000000e+00, 2.70000000e+01,
1.00000000e+00,
         0.00000000e+00, 2.60000000e+01, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 1.80000000e+01,
0.00000000e+00,
         0.00000000e+00, 8.66250000e+00, 2.00000000e+00],
        [3.00000000e+00, 1.00000000e+00, 3.02725904e+01,
2.00000000e+00,
```

```
        0.00000000e+00, 2.16792000e+01, 0.00000000e+00],
       [3.00000000e+00, 1.00000000e+00, 3.30000000e+01,
0.00000000e+00,
        0.00000000e+00, 7.85420000e+00, 2.00000000e+00],
       [1.00000000e+00, 1.00000000e+00, 2.30000000e+01,
0.00000000e+00,
        0.00000000e+00, 9.35000000e+01, 2.00000000e+00]])
```

```
y_train
```

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

```
y_test
```

```
array([0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
0,
       1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
0,
       0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1,
0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
```

```
1,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0])
```

SCALING USING MINMAXSCALER

Normalization in machine learning is the process of translating data into the range [0,1] (or any other range) or simply transforming data onto the unit sphere. Some machine learning algorithms benefit from normalization and standardization, particularly when Euclidean distance is used.

```
scaler=MinMaxScaler()
scaler.fit(x_train)
#Normalized training data
x_train=scaler.transform(x_train)
#Normalized testing data
x_test=scaler.transform(x_test)
```

MODEL CREATION

classification algorithms are

1)K Nearest Neighbors

2)Naive Bayes

3)Support Vector Machine

1) K-Nearest Neighbors algorithm(KNN)::

```
#Knn

knn=KNeighborsClassifier(n_neighbors=7)
knn.fit(x_train,y_train)

y_pred_knn=knn.predict(x_test)
y_pred_knn

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

y_test
```

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

2)Naive Bayes

```
#naive bayes
naiv=BernoulliNB()
naiv.fit(x_train,y_train)

y_pred_naiv=naiv.predict(x_test)
y_pred_naiv
```

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

3)Support Vector Machine

```
#svm
sv=SVC()
sv.fit(x_train,y_train)

y_pred_sv=sv.predict(x_test)
y_pred_sv
```

```
array([0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
0,
       1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
0,
       0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1,
0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0,
```

```
0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0])
```

PERFORMANCE EVALUATION

```
#KNN
print(classification_report(y_test,y_pred_knn))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        85
           1       1.00      1.00      1.00        41

    accuracy                           1.00       126
   macro avg       1.00      1.00      1.00       126
weighted avg       1.00      1.00      1.00       126
```

```
score_knn=accuracy_score(y_pred_knn,y_test)
print(score_knn)
```

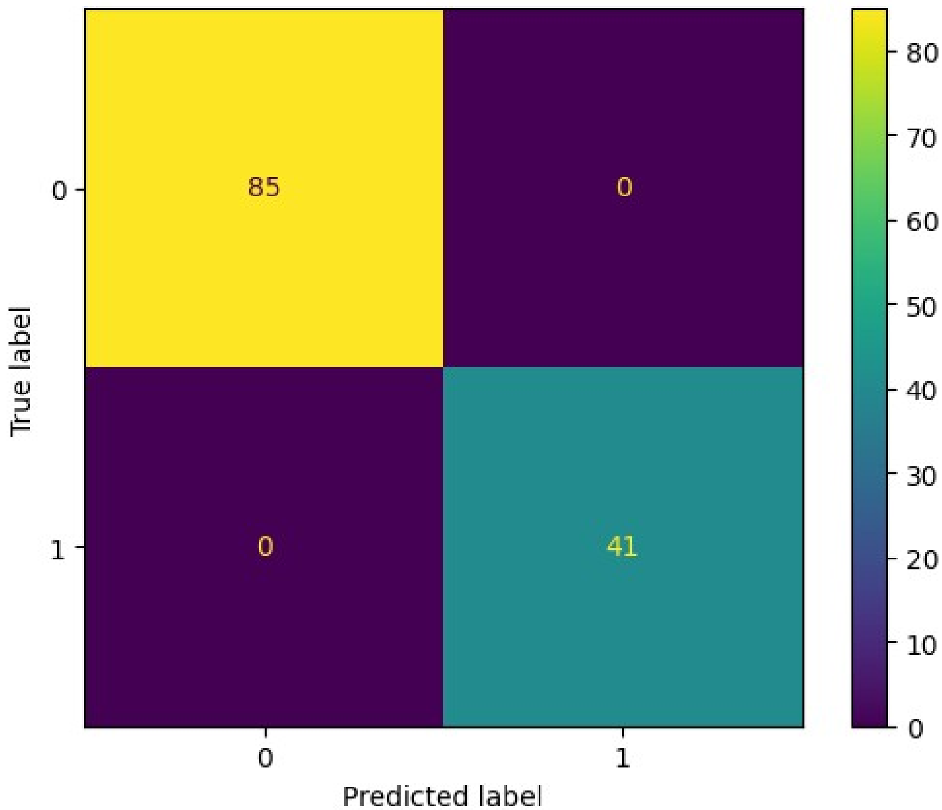
```
1.0
```

```
matx_knn=confusion_matrix(y_pred_knn,y_test)
print(matx_knn)
```

```
[[85  0]
 [ 0 41]]
```

```
label=[0,1]
cmd=ConfusionMatrixDisplay(matx_knn,display_labels=label)
cmd.plot()
```

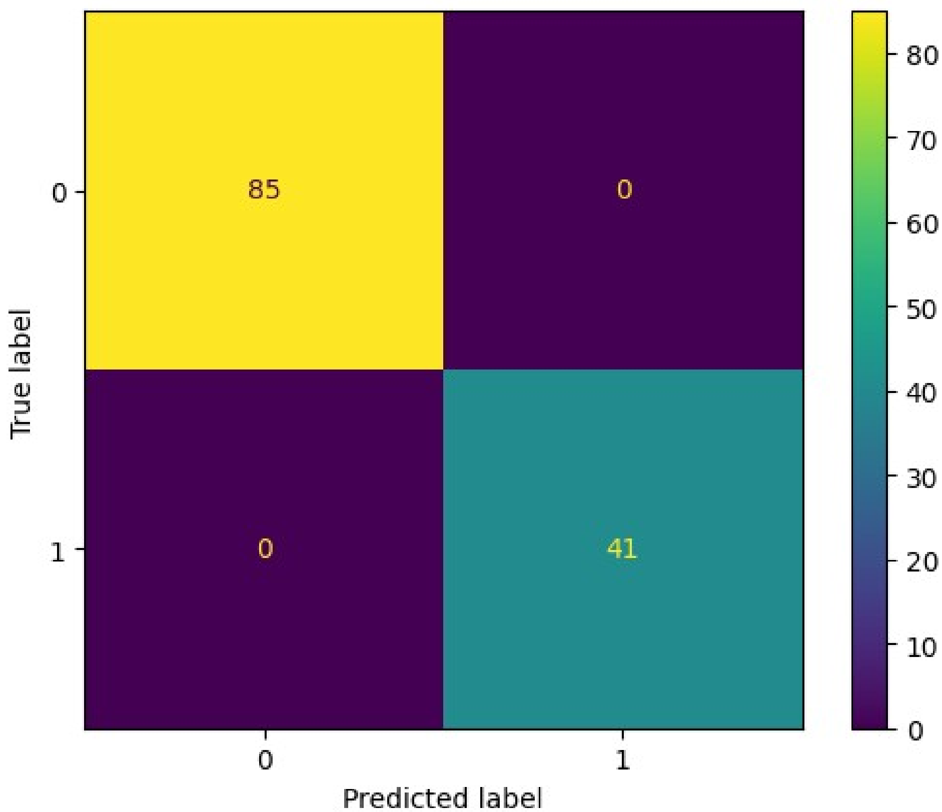
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f22692e2d90>
```




```
#Naive bayes
print(classification_report(y_test,y_pred_naiv))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        85
           1       1.00      1.00      1.00        41

    accuracy                           1.00       126
   macro avg       1.00      1.00      1.00       126
weighted avg       1.00      1.00      1.00       126


score_naiv=accuracy_score(y_pred_naiv,y_test)
print(score_naiv)

1.0

matx_naiv=confusion_matrix(y_pred_naiv,y_test)
print(matx_naiv)

[[85  0]
 [ 0 41]]
```

```
label=[0,1]
cmd=ConfusionMatrixDisplay(matx_naiv,display_labels=label)
cmd.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f22692e2ee0>
```




```
#SVM
print(classification_report(y_test,y_pred_sv))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        85
           1       1.00      1.00      1.00        41

    accuracy                           1.00       126
   macro avg       1.00      1.00      1.00       126
weighted avg       1.00      1.00      1.00       126
```

```
score_sv=accuracy_score(y_pred_sv,y_test)
print(score_sv)
```

```
1.0
```

```
matx_sv=confusion_matrix(y_pred_sv,y_test)
print(matx_sv)
```

```
[[85  0]
 [ 0 41]]
```

```
label=[0,1]
cmd=ConfusionMatrixDisplay(matx_sv,display_labels=label)
cmd.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f2269fae4f0>
```