TITANIC SURVIVAL PREDICTION

The objective of the dataset is to predict whether the passengers in the titanic ship were survived or not on the basis of some input values. The dataset contains 418 rows and 12 columns. PassengerId : Id of the passenger Survived : 0 = No; 1 = Yes Pclass : Passenger class Name : Name of the passenger Sex : Sex of the passenger Age : Age of the passenger SibSp : Siblings and Spouse Parch : Parents and Children Ticket : Ticket Number Fare : Passenger Fare Cabin : Cabin Embarked : Port of Embarkation

Import the libraries

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

Load the data

```
df=pd.read_csv('/content/titanic.csv')
df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
413	1305	0	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S
414	1306	1	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C
415	1307	0	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S
416	1308	0	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S

Data Exploration

```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.829
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.000

```
df.tail()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
413	1305	0	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236
414	1306	1	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758

```
df.shape
```

```
(418, 12)
```

```
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.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
```

```
df.columns
```

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

```
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	0.363636	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.481622	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	0.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	0.000000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	0.000000	3.000000	27.000000	0.000000	0.000000	14.454200
75%	1204.750000	1.000000	3.000000	39.000000	1.000000	0.000000	31.500000
max	1309.000000	1.000000	3.000000	76.000000	8.000000	9.000000	512.329200

Data Preprocessing

```
df.drop_duplicates()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q

```
df.isnull().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
```

```
df.drop(['PassengerId','Name','Ticket'],axis=1,inplace=True)
df
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	0	3	male	34.5	0	0	7.8292	NaN	Q
1	1	3	female	47.0	1	0	7.0000	NaN	S
2	0	2	male	62.0	0	0	9.6875	NaN	Q
3	0	3	male	27.0	0	0	8.6625	NaN	S
4	1	3	female	22.0	1	1	12.2875	NaN	S
...	...	...	...	...	...	...	...	...	...
413	0	3	male	NaN	0	0	8.0500	NaN	S
414	1	1	female	39.0	0	0	108.9000	C105	C
415	0	3	male	38.5	0	0	7.2500	NaN	S
416	0	3	male	NaN	0	0	8.0500	NaN	S
417	0	3	male	NaN	1	1	22.3583	NaN	C

418 rows × 9 columns

327 Cabin rows are missing values out of 418 rows. So here we can drop the column Cabin as it is irrelevant

```
df.drop(['Cabin'],axis=1,inplace=True)
df
```

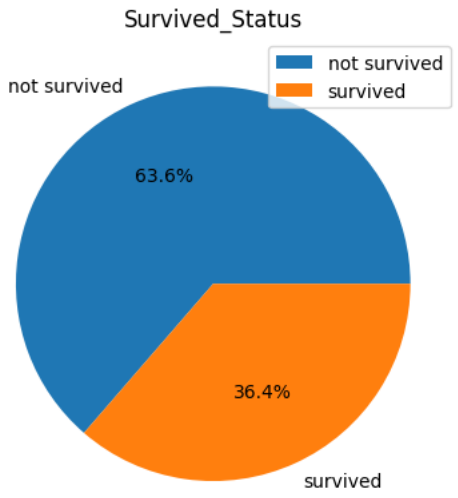
	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	34.5	0	0	7.8292	Q
1	1	3	female	47.0	1	0	7.0000	S
2	0	2	male	62.0	0	0	9.6875	Q
3	0	3	male	27.0	0	0	8.6625	S
4	1	3	female	22.0	1	1	12.2875	S
...	...	...	...	...	...	...	...	...
413	0	3	male	NaN	0	0	8.0500	S
414	1	1	female	39.0	0	0	108.9000	C
415	0	3	male	38.5	0	0	7.2500	S
416	0	3	male	NaN	0	0	8.0500	S
417	0	3	male	NaN	1	1	22.3583	C

418 rows × 8 columns

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

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

```
mylabels=['not survived','survived']
plt.pie(survived_count,labels=mylabels,autopct="%1.1f%%")
plt.title('Survived_Status')
plt.legend(loc='upper right')
plt.show()
```




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

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

```
plt.hist(df['Sex'],color='g')
```

```
(array([266.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 152.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <BarContainer object of 10 artists>)
```


```
sns.countplot(data=df,x='Sex',hue='Survived')
```

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

Converting strings to integer fromat using LabelEncoder

```
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
df['Sex']=encoder.fit_transform(df['Sex'])
```

```
df['Embarked']=encoder.fit_transform(df['Embarked'])
```

```
df
```

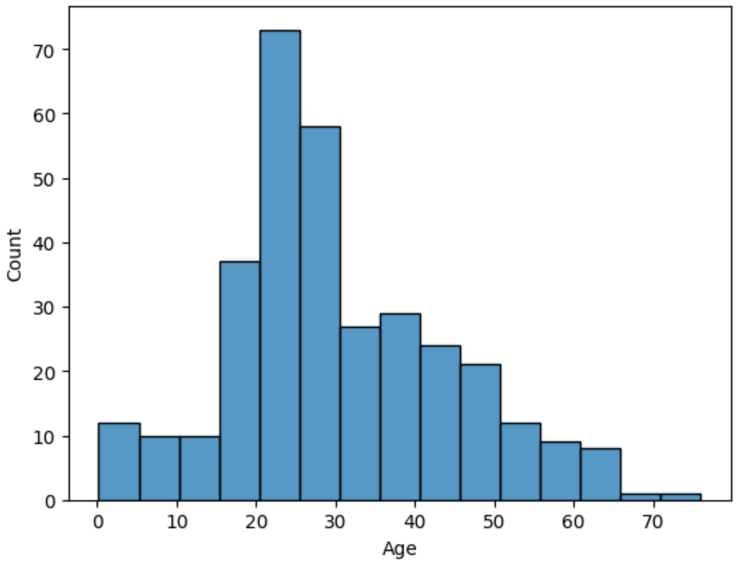
	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	34.5	0	0	7.8292	1
1	1	3	0	47.0	1	0	7.0000	2
2	0	2	1	62.0	0	0	9.6875	1
3	0	3	1	27.0	0	0	8.6625	2
4	1	3	0	22.0	1	1	12.2875	2
...	...	...	...	...	...	...	...	...
413	0	3	1	NaN	0	0	8.0500	2
414	1	1	0	39.0	0	0	108.9000	0
415	0	3	1	38.5	0	0	7.2500	2
416	0	3	1	NaN	0	0	8.0500	2
417	0	3	1	NaN	1	1	22.3583	0

418 rows × 8 columns

We converted the column sex and embarked using label encoder. Sex column converted to, male = 1 and female = 0. Embarked column converted to, Q = 1,S = 2,C = 0.

```
sns.histplot(df['Age'])
```

```
<Axes: xlabel='Age', ylabel='Count'>
```

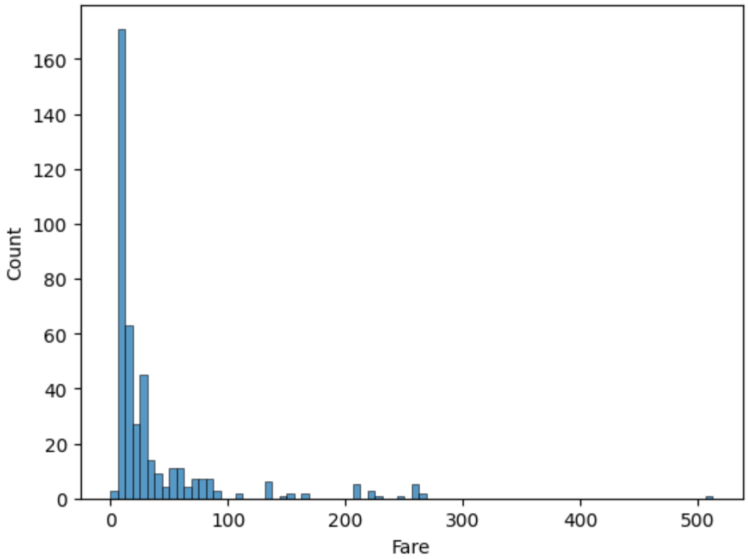


Filling the column Age with mean of that column.

```
df['Age'].fillna(df['Age'].mean(),inplace=True)
```

```
sns.histplot(df['Fare'])
```

```
<Axes: xlabel='Fare', ylabel='Count'>
```



Filling the column Fare with median of that column.

```
df['Fare'].fillna(df['Fare'].median(),inplace=True)
```

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

```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

```
df.dtypes
```

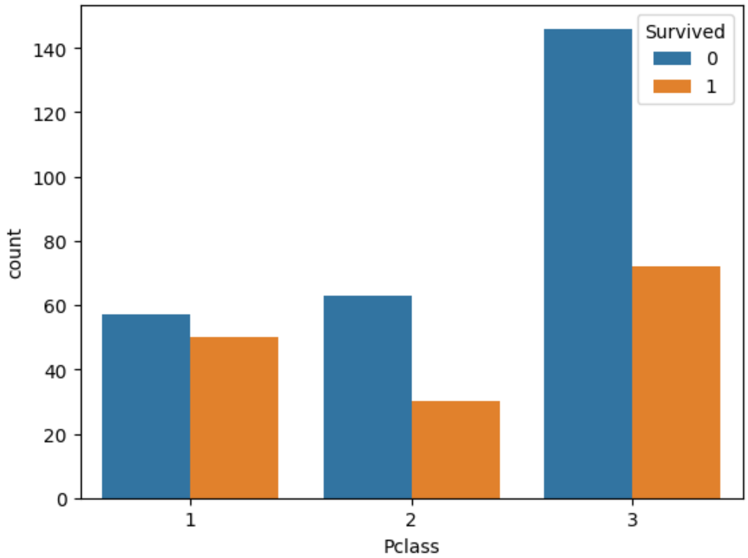
```
Survived      int64
Pclass        int64
Sex           int64
Age         float64
SibSp         int64
Parch         int64
Fare        float64
Embarked      int64
dtype: object
```

```
plt.hist(df['Age'])
```

```
(array([ 16.,  16.,  71., 183.,  43.,  37.,  25.,  17.,   9.,   1.]),
 array([ 0.17 ,  7.753, 15.336, 22.919, 30.502, 38.085, 45.668, 53.251,
        60.834, 68.417, 76.   ]),
 <BarContainer object of 10 artists>)
```

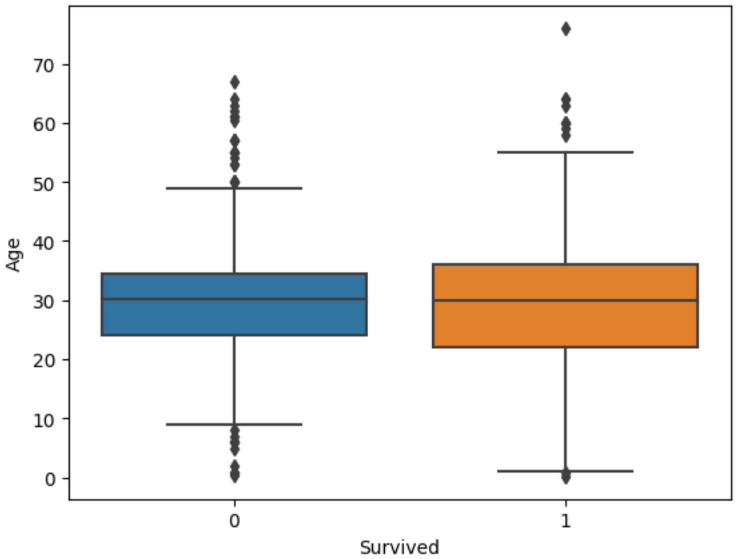

```
sns.countplot(data=df,x='Pclass',hue='Survived')
```

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




```
sns.boxplot(data=df,x='Survived',y='Age')
```

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



Seperating x and y

```
x=df.drop(['Survived'],axis=1)
y=df['Survived']
```

Spliting x and y

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)
```

Performing Normalization

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

Model creation using KNN

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

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

Model evaluation

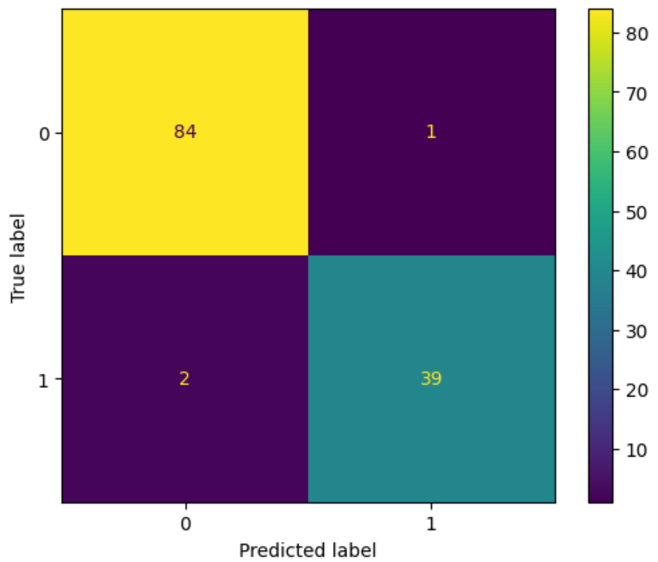
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report,ConfusionMatrixDisplay,confusion_matrix
labels=['0','1']
print("accuracy score is",accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
result=confusion_matrix(y_test,y_pred)
cmd=ConfusionMatrixDisplay(result,display_labels=labels)
cmd.plot()
print(result)
```

```
accuracy score is 0.9761904761904762
              precision    recall  f1-score   support

           0       0.98      0.99      0.98        85
           1       0.97      0.95      0.96        41

    accuracy                           0.98       126
   macro avg       0.98      0.97      0.97       126
weighted avg       0.98      0.98      0.98       126

[[84  1]
 [ 2 39]]
```



KNN model gives 98% of accuracy.