Titanic survival prediction

In [2]:
```
import pandas as pd
```

Explonatory Data Analysis

In [3]:
```
df=pd.read_csv("D:\datset\Titanic-Dataset.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emba
0	1	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	5	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	

In [5]:
```
df.tail()
```

Out[5]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarke
886	887	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

In [6]:
```
df.shape
```

Out[6]:
```
(891, 12)
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['PassengerId', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch',
       'Ticket', 'Fare', 'Cabin', 'Embarked', 'survived'],
      dtype='object')
```

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Pclass       891 non-null    int64  
 2   Name         891 non-null    object 
 3   Sex          891 non-null    object 
 4   Age          714 non-null    float64
 5   SibSp        891 non-null    int64  
 6   Parch        891 non-null    int64  
 7   Ticket       891 non-null    object 
 8   Fare         891 non-null    float64
 9   Cabin        204 non-null    object 
 10  Embarked     889 non-null    object 
 11  survived     891 non-null    int64  
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [9]:
```
df.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	max
PassengerId	891.0	446.000000	257.353842	1.00	223.5000	446.0000	668.5	891.0000
Pclass	891.0	2.308642	0.836071	1.00	2.0000	3.0000	3.0	3.0000
Age	714.0	29.699118	14.526497	0.42	20.1250	28.0000	38.0	80.0000
SibSp	891.0	0.523008	1.102743	0.00	0.0000	0.0000	1.0	8.0000
Parch	891.0	0.381594	0.806057	0.00	0.0000	0.0000	0.0	6.0000
Fare	891.0	32.204208	49.693429	0.00	7.9104	14.4542	31.0	512.3292
survived	891.0	0.383838	0.486592	0.00	0.0000	0.0000	1.0	1.0000

In [10]:
```
df.isna().sum()
```

Out[10]:
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
survived         0
dtype: int64
```

In [11]:
```
df['Age'].fillna(df['Age'].median() , inplace=True)
```

In [12]:
```
df.drop(columns=['PassengerId','Cabin','Name'], inplace=True)
```

In [13]:
```
df['Embarked'].value_counts()
```

Out[13]:
```
Embarked
S    644
C    168
Q     77
Name: count, dtype: int64
```

In [14]:
```
df['Embarked'].fillna('S', inplace=True) # because S is majourly used
```

In [15]:
```
df.isna().sum()
```

Out[15]:
```
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Ticket      0
Fare        0
Embarked    0
survived    0
dtype: int64
```

In [16]:
```
df.shape
```

Out[16]:
```
(891, 9)
```

Label Encoding

In [17]:
```
df.dtypes
```

Out[17]:
```
Pclass        int64
Sex          object
Age         float64
SibSp         int64
Parch         int64
Ticket       object
Fare        float64
Embarked     object
survived      int64
dtype: object
```

In [18]:
```
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
```

In [19]:
```
df['Sex']=le.fit_transform(df['Sex'])
```

In [20]:
```
df['Ticket'] =le.fit_transform(df['Ticket'])
```

In [21]:
```
df['Embarked'] =le.fit_transform(df['Embarked'])
```

selecting dependent and independent variable

In [22]:
```
x=df.iloc[:,0:8]
x
```

Out[22]:

	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	3	1	22.0	1	0	523	7.2500	2
1	1	0	38.0	1	0	596	71.2833	0
2	3	0	26.0	0	0	669	7.9250	2
3	1	0	35.0	1	0	49	53.1000	2
4	3	1	35.0	0	0	472	8.0500	2
...	...	...	...	...	...	...	...	...
886	2	1	27.0	0	0	101	13.0000	2
887	1	0	19.0	0	0	14	30.0000	2
888	3	0	28.0	1	2	675	23.4500	2
889	1	1	26.0	0	0	8	30.0000	0
890	3	1	32.0	0	0	466	7.7500	1

891 rows × 8 columns

In [23]:
```
y=df['survived']
y
```

Out[23]:
```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: survived, Length: 891, dtype: int64
```

splitting the dataset

In [24]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train, y_test= train_test_split(x,y,random_state=101,test_siz
```

In [25]:
```
x_train
```

Out[25]:

	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
733	2	1	23.0	0	0	228	13.0000	2
857	1	1	51.0	0	0	23	26.5500	2
81	3	1	29.0	0	0	311	9.5000	2
319	1	0	40.0	1	1	81	134.5000	0
720	2	0	6.0	0	1	155	33.0000	2
...	...	...	...	...	...	...	...	...
575	3	1	19.0	0	0	420	14.5000	2
838	3	1	32.0	0	0	80	56.4958	2
337	1	0	41.0	0	0	81	134.5000	0
523	1	0	44.0	0	1	7	57.9792	0
863	3	0	28.0	8	2	568	69.5500	2

712 rows × 8 columns

standardisation

In [26]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
```

In [27]:
```
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

Random forest

random forest is used for prediction

In [28]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
```

Out[28]:
```
▾ RandomForestClassifier

RandomForestClassifier()
```

In [30]:
```
y_pred=rf.predict(x_test)
y_pred
```

Out[30]:
```
array([0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 0], dtype=int64)
```

In [31]:
```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_pred))
```

```
0.8379888268156425
```

In [32]:
```
import warnings
warnings.filterwarnings("ignore", message="X does not have valid feature names
```

In [33]:
```
sample_data=[[3,1,16.0,0,0,504,9.2167,2]]
sample_data_scaled=sc.transform(sample_data)
prediction=rf.predict(sample_data_scaled)
print("final prediction : ",prediction)
```

```
final prediction :  [0]
```

In [34]:
```
sample_data=[[3,1,26.0,0,0,216,18.7875,0]]
sample_data_scaled=sc.transform(sample_data)
prediction=rf.predict(sample_data_scaled)
print("final prediction : ",prediction)
```

```
final prediction :  [1]
```