Titanic Case Study

In [1]:
```
#Import Necessary Liabraries for predection

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [2]:
```
data_titanic = pd.read_csv(r"C:\Users\masir\Downloads\Titanic-Dataset.csv")
data_titanic.head()
```

Out[2]:

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

In [10]:
```
data_titanic.shape
```

Out[10]: (891, 11)

In [9]:
```
data_titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB
```

In [11]:
```
data_titanic.isnull().sum()
```

Out[11]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       2
dtype: int64
```

In [12]:
```
## In "Age" variable 177 is null values
## & in "Cabin" column there are 687 null values
## so dropping "Cabin" column. (50% & above Null)
```

In [4]:
```
data_titanic = data_titanic.drop(columns="Cabin", axis=1)
```

In [59]:
```
data_titanic
```

Out[59]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fa
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25(
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.28:
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92!
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.10(
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05(
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00(
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00(
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45(
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00(
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75(

891 rows × 11 columns

In [5]:
```
data_titanic.describe()
```

Out[5]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [6]:
```
## now replacing missing values in "age" with mean value
```

In [7]:
```
data_titanic["Age"].fillna(data_titanic["Age"].mean(), inplace=True)
```

In [8]:
```
data_titanic.describe()
```

Out[8]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	13.002015	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	29.699118	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [13]:
```
##working on 3rd missing value column "Embarked"
## as we know in embarked column there is no interger value so cannot go fo
```

In [14]:
```
##finding the mode value of embarked column
```

In [15]:
```
print(data_titanic["Embarked"].mode())
```

```
0    S
Name: Embarked, dtype: object
```

In [16]:
```
print(data_titanic["Embarked"].mode()[0])
```

```
S
```

In [17]:
```
data_titanic["Embarked"].fillna(data_titanic["Embarked"].mode()[0], inplace
```

In [18]:
```
data_titanic.isnull().sum()
```

Out[18]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

Exploratory Data Analysis

In [19]:
```
data_titanic.describe()
```

Out[19]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	13.002015	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	29.699118	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [20]:
```
data_titanic["Survived"].value_counts()
```

Out[20]:
```
Survived
0    549
1    342
Name: count, dtype: int64
```

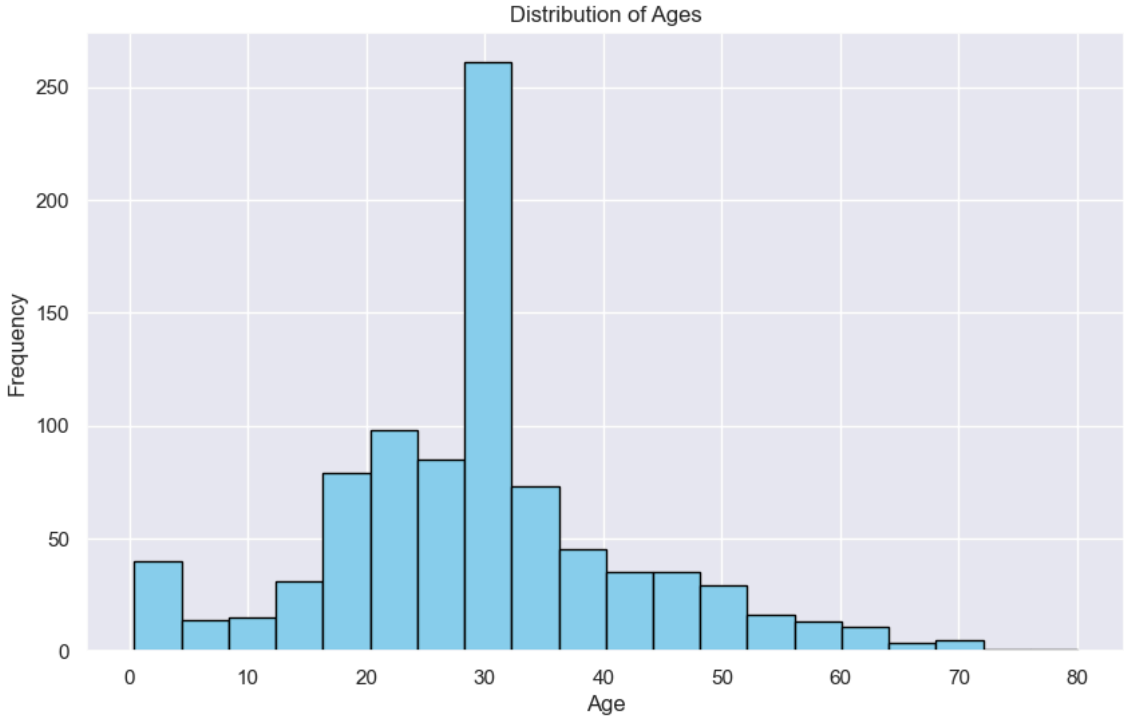
1 Data Visualisation

In [21]:
```
##checking for survived & non survived cases with countplot
```

In [22]:
```
sns.set()
```

In [25]:
```
import matplotlib.pyplot as plt

# Create histogram
plt.figure(figsize=(10, 6))
plt.hist(data_titanic['Age'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Ages')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

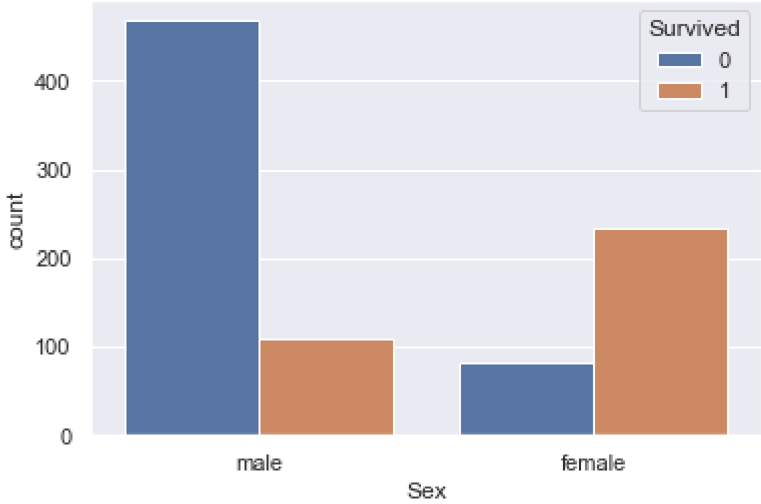


In [78]:
```
##comparing data of survivors with gender
```

In [79]:
```
sns.countplot('Sex', hue="Survived", data=data_titanic)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: Pass the following variable as a keyword arg: x. From version 0.
12, the only valid positional argument will be `data`, and passing other a
rguments without an explicit keyword will result in an error or misinterpr
etation.
  warnings.warn(
```

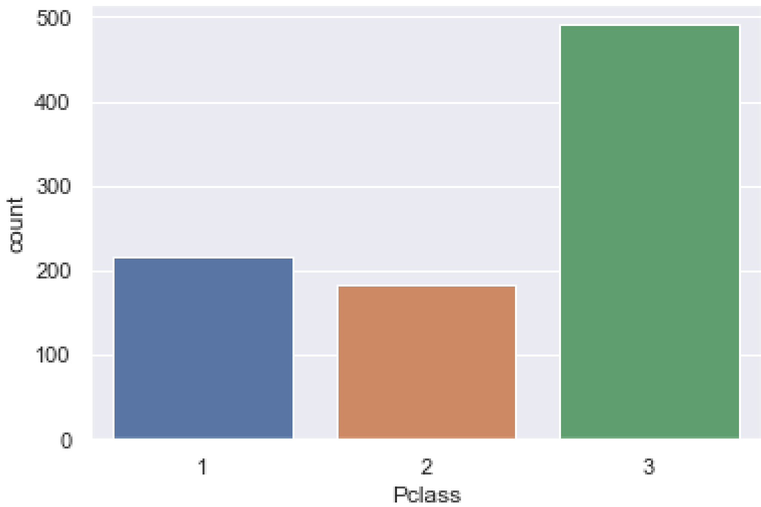
Out[79]: <AxesSubplot:xlabel='Sex', ylabel='count'>



In [80]:
```
# cheking countplot for "Pclass" column
sns.countplot('Pclass', data=data_titanic)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: Pass the following variable as a keyword arg: x. From version 0.
12, the only valid positional argument will be `data`, and passing other a
rguments without an explicit keyword will result in an error or misinterpr
etation.
  warnings.warn(
```

Out[80]: <AxesSubplot:xlabel='Pclass', ylabel='count'>



In [81]:
```
##comparing Survived (Class wise)
```

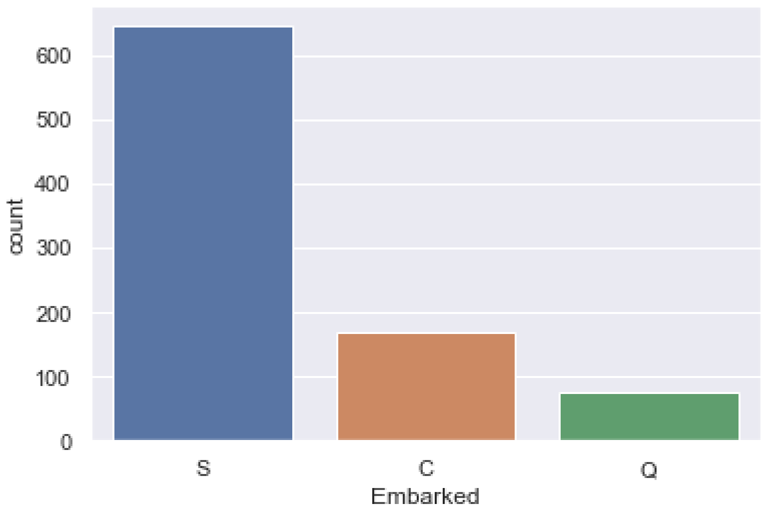
In [82]:
```
##many people were travelling in 3rd class(LOWER) in Titanic.
```

In [83]:
```
# now cheking countplot for "Embarked" column
# checking how many people started their journey from various locations.
```

In [84]:
```
sns.countplot('Embarked', data=data_titanic)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: Pass the following variable as a keyword arg: x. From version 0.
12, the only valid positional argument will be `data`, and passing other a
rguments without an explicit keyword will result in an error or misinterpr
etation.
  warnings.warn(
```

Out[84]:
```
<AxesSubplot:xlabel='Embarked', ylabel='count'>
```



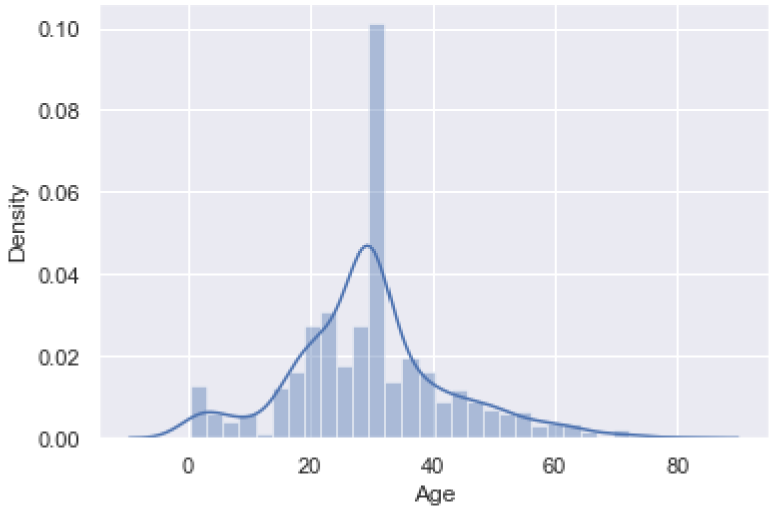
In [85]:
```
## most of the people have started their journey from Southampton (S).
```

Checking numerical attributes

In [86]:
```
sns.distplot(data_titanic['Age'])
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\distributions.py:2619: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-l
evel function with similar flexibility) or `histplot` (an axes-level funct
ion for histograms).
  warnings.warn(msg, FutureWarning)
```

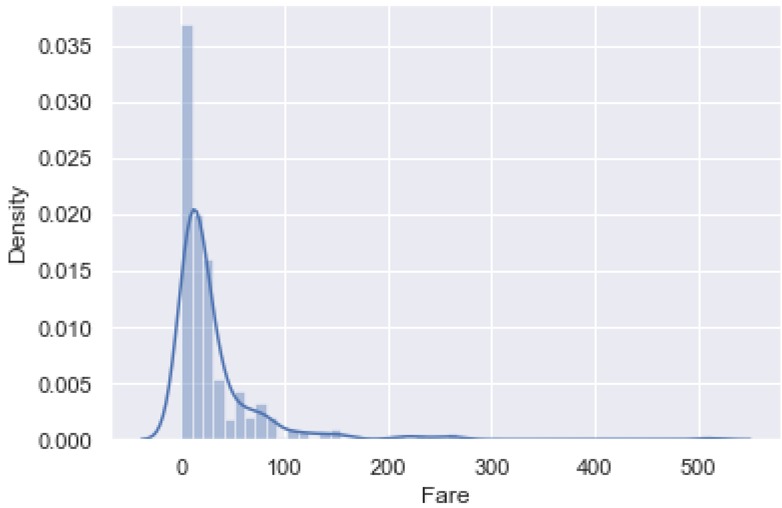
Out[86]: <AxesSubplot:xlabel='Age', ylabel='Density'>



In [87]:
```
#checking for Fare column
sns.distplot(data_titanic['Fare'])
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\distributions.py:2619: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-l
evel function with similar flexibility) or `histplot` (an axes-level funct
ion for histograms).
  warnings.warn(msg, FutureWarning)
```

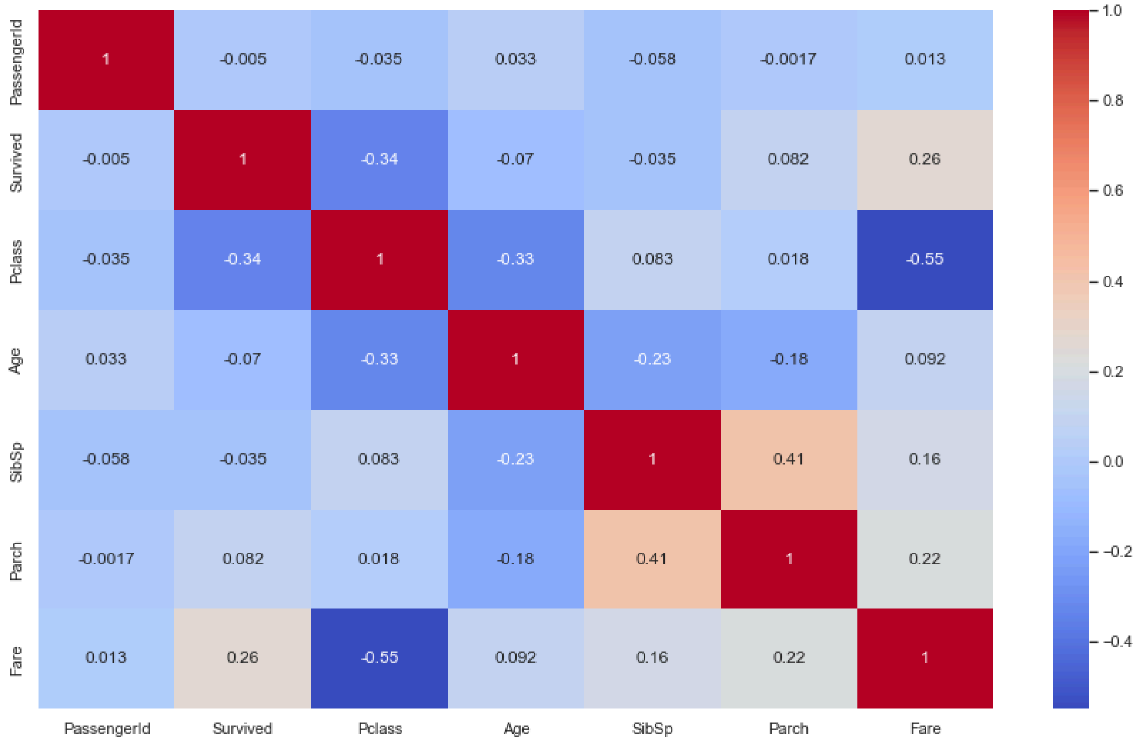
Out[87]: <AxesSubplot:xlabel='Fare', ylabel='Density'>



HeatMap to check correlation

In [88]:
```
corr = data_titanic.corr()
plt.figure(figsize=(15, 9))
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

Out[88]: <AxesSubplot:>



In [89]:
```
data_titanic.head()
```

Out[89]:

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

In [90]:
```
## drop unnecessary columns
data_titanic = data_titanic.drop(columns=['Name', 'Ticket'], axis=1)
data_titanic.head()
```

Out[90]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	male	22.0	1	0	7.2500	S
1	2	1	1	female	38.0	1	0	71.2833	C
2	3	1	3	female	26.0	0	0	7.9250	S
3	4	1	1	female	35.0	1	0	53.1000	S
4	5	0	3	male	35.0	0	0	8.0500	S

Encoding Label

In [91]:
```
#Categorical to Numerical for further modelling
```

In [92]:
```
data_titanic["Sex"].value_counts()
```

Out[92]:
```
male      577
female    314
Name: Sex, dtype: int64
```

In [93]:
```
data_titanic['Embarked'].value_counts()
```

Out[93]:
```
S    646
C    168
Q     77
Name: Embarked, dtype: int64
```

In [94]:
```
from sklearn.preprocessing import LabelEncoder
cols = ['Sex', 'Embarked']
le = LabelEncoder()

for col in cols:
    data_titanic[col] = le.fit_transform(data_titanic[col])
data_titanic.head()
```

Out[94]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	1	22.0	1	0	7.2500	2
1	2	1	1	0	38.0	1	0	71.2833	0
2	3	1	3	0	26.0	0	0	7.9250	2
3	4	1	1	0	35.0	1	0	53.1000	2
4	5	0	3	1	35.0	0	0	8.0500	2

Train_Test_Split

In [28]:
```
X = data_titanic.drop(columns = ['PassengerId','Survived'],axis=1)
Y = data_titanic['Survived']
```

In [29]:
```
print(X)
```

```
     Pclass                                               Name     Sex  \
0         3                            Braund, Mr. Owen Harris    male   
1         1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female   
2         3                             Heikkinen, Miss. Laina  female   
3         1       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female   
4         3                           Allen, Mr. William Henry    male   
..      ...                                                ...     ...   
886       2                              Montvila, Rev. Juozas    male   
887       1                       Graham, Miss. Margaret Edith  female   
888       3           Johnston, Miss. Catherine Helen "Carrie"  female   
889       1                              Behr, Mr. Karl Howell    male   
890       3                                Dooley, Mr. Patrick    male   

           Age  SibSp  Parch            Ticket     Fare Embarked  
0    22.000000      1      0         A/5 21171   7.2500        S  
1    38.000000      1      0          PC 17599  71.2833        C  
2    26.000000      0      0  STON/O2. 3101282   7.9250        S  
3    35.000000      1      0            113803  53.1000        S  
4    35.000000      0      0            373450   8.0500        S  
..         ...    ...    ...               ...      ...      ...  
886  27.000000      0      0            211536  13.0000        S  
887  19.000000      0      0            112053  30.0000        S  
888  29.699118      1      2        W./C. 6607  23.4500        S  
889  26.000000      0      0            111369  30.0000        C  
890  32.000000      0      0            370376   7.7500        Q  

[891 rows x 9 columns]
```

In [30]:
```
print(Y)
```

```
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
```

In [31]:
```
##Splitting the data into training data & Test data.
```

In [32]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, ran
```

In [33]:
```
print(X.shape, X_train.shape, X_test.shape)
```

```
(891, 9) (712, 9) (179, 9)
```

Model Training for logistic regression

In [34]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [35]:
```
model = LogisticRegression()
```

In [36]:
```
data_titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Embarked     891 non-null    object 
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB
```

In [37]:
```
data_titanic.astype({'Age':'int','Fare':'int'}).dtypes
```

Out[37]:
```
PassengerId     int64
Survived        int64
Pclass          int64
Name           object
Sex            object
Age             int32
SibSp           int64
Parch           int64
Ticket         object
Fare            int32
Embarked       object
dtype: object
```

In [43]:
```
#training the Logistic Regression model with training data
model.fit(X_train, Y_train)
```

In [109]:
```
#accuracy on training data
X_train_prediction = model.predict(X_train)
```

In [110]:
```
print(X_train_prediction)
```

```
[0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 1
 0 0 0 0 0 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 1 0 0 1 1 0 1 0 0 1
 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 1 1 1 0 1 0 0 0 0 0 1 0 0 0
 1 1 0 0 1 0 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1 0 0 1 1 1 0 0 1 0 0
 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 0 1 1 1
 0 0 0 1 0 0 0 1 0 0 1 0 0 1 1 1 0 1 0 0 0 0 0 1 1 0 1 1 1 1 0 0 0 0 0 0 0
 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0
 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 0
 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 1 1
 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0 1 1 0 0 0 1 0 1 1 1 0 0
 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1 1 1 0 1 1 0 0 0
 0 1 0 1 0 0 1 1 0 0 0 0 1 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 1 0 0
 1 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 1 0 1 0 0 1 0 0 0 1 1 0 1 0
 0 0 0 0 1 0 0 1 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 1 1 0 1 0
 0 1 0 0 1 0 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 1 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0
 0 0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0 0 1 1
 0 0 0 1 0 1 0 0 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0
 1 0 0 1 0 1 0 0 0 1 1 1 1 1 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0 0 1 0 0 0 0 0 0
 0 0 0 1 1 0 0 1 0]
```

In [112]:
```
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
print('Accuracy_score_of_training_data : ', training_data_accuracy)
```

```
Accuracy_score_of_training_data :  0.8132022471910112
```

In [113]:
```
# accuracy on test data
X_test_prediction = model.predict(X_test)
```

In [114]:
```
print(X_test_prediction)
```

```
[0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1 1
 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 1 0
 1 0 0 0 1 0 1 0 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 1 0 1 1 0 1 1 0 1 1 0 0 0 0
 0 0 0 1 1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 1 0 1 0 0
 0 1 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 0 1 0 0 1 1 1 0 0 0 0 0]
```

In [116]:
```
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
print('Accuracy_score_of_test data : ', test_data_accuracy)
```

```
Accuracy_score_of_test data :  0.7877094972067039
```