Credential's

Name : Mayank Anand

Registration Number : 2141001045

In [109]:
```
# Importing the required libraries

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

In [110]:
```
# To load the dataset

df=pd.read_csv("Titanic Dataset.csv")
```

Exploratory Data Analysis (EDA)

In [111]:
```
# To see first 10 columns of the dataset

df.head(10)
```

Out[111]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

In [112]:
```
# To see last 10 columns of the dataset

df.tail(10)
```

Out[112]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
881	882	0	3	Markun, Mr. Johann	male	33.0	0	0	349257	7.8958	NaN	S
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.0	0	0	7552	10.5167	NaN	S
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.5000	NaN	S
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.0500	NaN	S
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	NaN	Q
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

In [113]:
```
# To know the shape of the dataset

df.shape
```

Out[113]:
```
(891, 12)
```

In [114]:
```
# To get overall preview about the dataframe

df.info()
```

Out[114]:
```
<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   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          714 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  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [115]:
```
# To know all the column names of the dataset

df.columns
```

Out[115]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [116]:
```
# To know the datatype of each dataset

df.dtypes
```

Out[116]:
```
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
```

In [117]:
```
# To generate descriptive statistics of the dataset

df.describe()
```

Out[117]:

	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 [118]:
```
# To know the how many male and female passengers where travelling

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

Out[118]:
```
male      577
female    314
Name: Sex, dtype: int64
```

In [119]:
```
# To know how man survived that disaster

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

Out[119]:
```
0    549
1    342
Name: Survived, dtype: int64
```

In [120]:
```
# To know how many passengers where travelling in which cabin

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

Out[120]:
```
B96 B98        4
G6             4
C23 C25 C27    4
C22 C26        3
F33            3
              ..
E34            1
C7             1
C54            1
E36            1
C148           1
Name: Cabin, Length: 147, dtype: int64
```

Data Cleaning

In [121]:
```
# to know about how may data's are null in the dataframe

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

Out[121]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

Passenger ID, Name , Ticket and Cabin are irrelevant for us so we are dropping that columns from the dataset.

In [122]:
```
# Dropping irrelevant columns

df1=df.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1)
```

In [123]:
```
df1.isnull().sum()
```

Out[123]:
```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
```

We can observe form the above that Age Column has 177 missing values remaining so i would like to replace it with Mean value of that Column.

In [124]:
```
# Handling missing values
# To calculate the mean value of Age Column

age_mean = df1['Age'].mean()
```

In [125]:
```
# To replace the missing value of the age column with the Mean value of that column

df1['Age'].fillna(age_mean, inplace=True)
```

In [126]:
```
df1.isnull().sum()
```

Out[126]:
```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    2
dtype: int64
```

In [127]:
```
# Data Reduction
# To drop rows where 'Embarked' column has null values

df1.dropna(subset=['Embarked'], inplace=True)
```

In [128]:
```
df1.isnull().sum()
```

Out[128]:
```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

In [129]:
```
df1.shape
```

Out[129]:
```
(889, 8)
```

Encoding Categorical Columns

In [130]:
```
df1.dtypes
```

Out[130]:
```
Survived      int64
Pclass        int64
Sex          object
Age         float64
SibSp         int64
Parch         int64
Fare        float64
Embarked     object
dtype: object
```

We can observe form above that Sex and Embarked Column are only Categorical so we need to transform them into numerical .

In [131]:
```
df1['Sex'].value_counts()
```

Out[131]:
```
male      577
female    312
Name: Sex, dtype: int64
```

In [132]:
```
# Converting Male to 0 and Female to 1

mapping = {'male': 0, 'female':1}
df2 = df1.copy()
df2['Sex'] = df2['Sex'].map(mapping)
```

In [133]:
```
df2.head(10)
```

Out[133]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	0	22.000000	1	0	7.2500	S
1	1	1	1	38.000000	1	0	71.2833	C
2	1	3	1	26.000000	0	0	7.9250	S
3	1	1	1	35.000000	1	0	53.1000	S
4	0	3	0	35.000000	0	0	8.0500	S
5	0	3	0	29.699118	0	0	8.4583	Q
6	0	1	0	54.000000	0	0	51.8625	S
7	0	3	0	2.000000	3	1	21.0750	S
8	1	3	1	27.000000	0	2	11.1333	S
9	1	2	1	14.000000	1	0	30.0708	C

In [134]:
```
df2['Embarked'].value_counts()
```

Out[134]:
```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In [135]:
```
mapping = {'S': 0, 'C':1, 'Q':2}
df3 = df2.copy()
df3['Embarked'] = df3['Embarked'].map(mapping)
```

In [136]:
```
df3.head(10)
```

Out[136]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	0	22.000000	1	0	7.2500	0
1	1	1	1	38.000000	1	0	71.2833	1
2	1	3	1	26.000000	0	0	7.9250	0
3	1	1	1	35.000000	1	0	53.1000	0
4	0	3	0	35.000000	0	0	8.0500	0
5	0	3	0	29.699118	0	0	8.4583	2
6	0	1	0	54.000000	0	0	51.8625	0
7	0	3	0	2.000000	3	1	21.0750	0
8	1	3	1	27.000000	0	2	11.1333	0
9	1	2	1	14.000000	1	0	30.0708	1

Successfully converted the Categorical column into numbers.

Data Normalization

In [137]:
```
df3.dtypes
```

Out[137]:
```
Survived      int64
Pclass        int64
Sex           int64
Age         float64
SibSp         int64
Parch         int64
Fare        float64
Embarked      int64
dtype: object
```

From above we can observe that we can Normalise the Age and Fare Column as has a lot of variations

In [138]:
```
# Importing requried library for Normalisation

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [139]:
```
df3.head(10)
```

Out[139]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	0	22.000000	1	0	7.2500	0
1	1	1	1	38.000000	1	0	71.2833	1
2	1	3	1	26.000000	0	0	7.9250	0
3	1	1	1	35.000000	1	0	53.1000	0
4	0	3	0	35.000000	0	0	8.0500	0
5	0	3	0	29.699118	0	0	8.4583	2
6	0	1	0	54.000000	0	0	51.8625	0
7	0	3	0	2.000000	3	1	21.0750	0
8	1	3	1	27.000000	0	2	11.1333	0
9	1	2	1	14.000000	1	0	30.0708	1

In [140]:
```
# Performing Normalisation on Age and Fare Columns

df3[['Age', 'Fare']] = scaler.fit_transform(df3[['Age', 'Fare']])
```

In [141]:
```
df3.head(10)
```

Out[141]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	0	-0.590495	1	0	-0.502240	0
1	1	1	1	0.643971	1	0	0.788947	1
2	1	3	1	-0.281878	0	0	-0.486650	0
3	1	1	1	0.412509	1	0	0.422861	0
4	0	3	0	0.412509	0	0	-0.484133	0
5	0	3	0	0.000924	0	0	-0.479913	2
6	0	1	0	1.679437	0	0	0.397946	0
7	0	3	0	-2.133577	3	1	-0.223900	0
8	1	3	1	-0.204724	0	2	-0.422057	0
9	1	2	1	-1.207728	1	0	-0.040797	1