Name: Yash Jain

Register No.:20BLC1058

ADS Assignment 2

Titanic Ship Case Study

Problem Description: On April 15, 1912, during her maiden voyage, the Titanic sank after colliding

with an iceberg, killing 1502 out of 2224 passengers and crew. Translated 32% survival rate.

� One of the reasons that the shipwreck led to such loss of life was that there were not

enough lifeboats for the passengers and crew.

� Although there was some element of luck involved in surviving the sinking, some groups of

people were more likely to survive than others, such as women, children, and the upper-

class.

The problem associated with the Titanic dataset is to predict whether a passenger survived the

disaster or not. The dataset contains various features such as passenger class, age, gender,

cabin, fare, and whether the passenger had any siblings or spouses on board. These features can

be used to build a predictive model to determine the likelihood of a passenger surviving the

disaster. The dataset offers opportunities for feature engineering, data visualization, and model

selection, making it a valuable resource for developing and testing data analysis and machine

learning skills.

Perform Below Tasks to complete the assignment:-

1. Download the dataset: Dataset

2. Load the dataset.

```
In [1]:
##ques2
import pandas as pd
df = pd.read_csv('titanic.csv')
print(df)
```

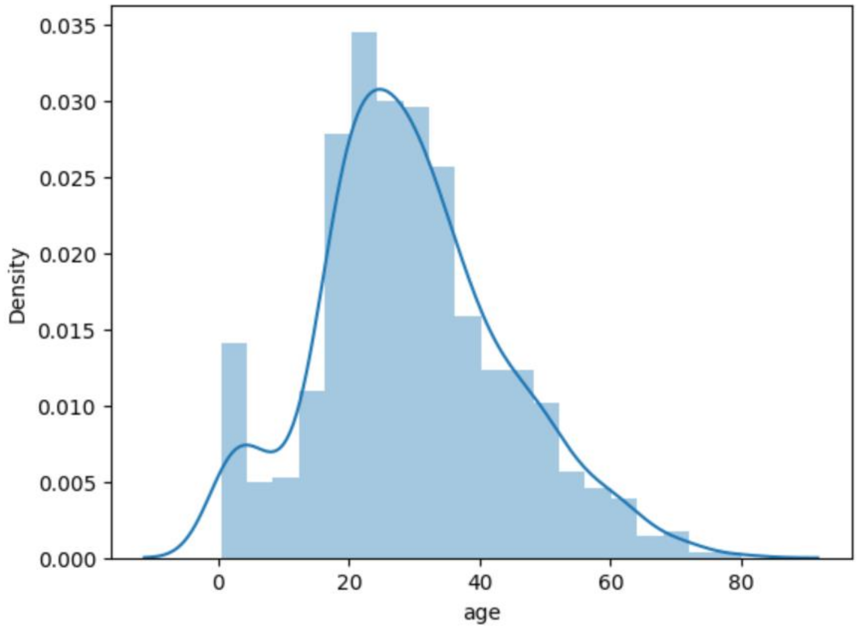
```
     survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0           0       3    male  22.0      1      0   7.2500        S   Third   
1           1       1  female  38.0      1      0  71.2833        C   First   
2           1       3  female  26.0      0      0   7.9250        S   Third   
3           1       1  female  35.0      1      0  53.1000        S   First   
4           0       3    male  35.0      0      0   8.0500        S   Third   
..        ...     ...     ...   ...    ...    ...      ...      ...     ...   
886         0       2    male  27.0      0      0  13.0000        S  Second   
887         1       1  female  19.0      0      0  30.0000        S   First   
888         0       3  female   NaN      1      2  23.4500        S   Third   
889         1       1    male  26.0      0      0  30.0000        C   First   
890         0       3    male  32.0      0      0   7.7500        Q   Third   

       who  adult_male deck  embark_town alive  alone  
0      man        True  NaN  Southampton    no  False  
1    woman       False    C    Cherbourg   yes  False  
2    woman       False  NaN  Southampton   yes   True  
3    woman       False    C  Southampton   yes  False  
4      man        True  NaN  Southampton    no   True  
..     ...         ...  ...          ...   ...    ...  
```

3. Perform Below Visualizations.

- **Univariate Analysis**

```
In [31]:
#ques3
import seaborn as sns
import matplotlib.pyplot as plt
##univariate analysis
sns.distplot(df['age'])
```

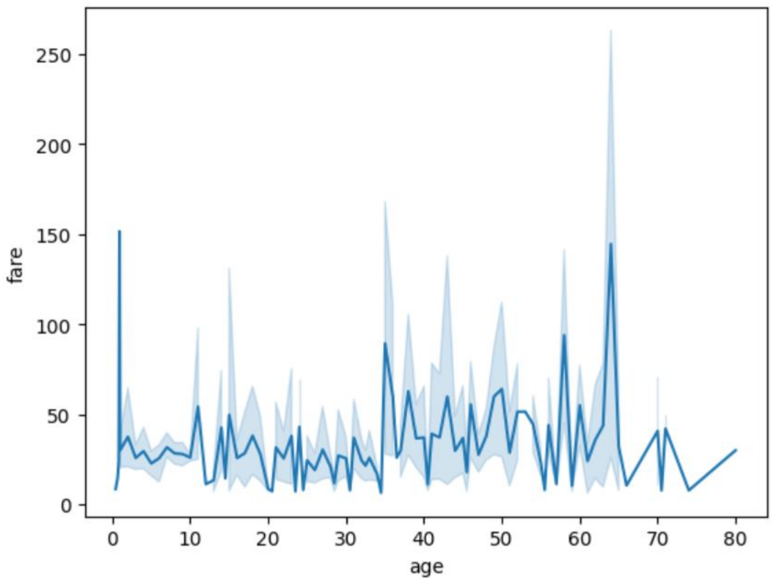
```
Out[31]: <Axes: xlabel='age', ylabel='Density'>
```



● **Bi - Variate Analysis**

```
In [32]: ##bivariate analysis

         sns.lineplot(x=df['age'], y=df['fare'])
```

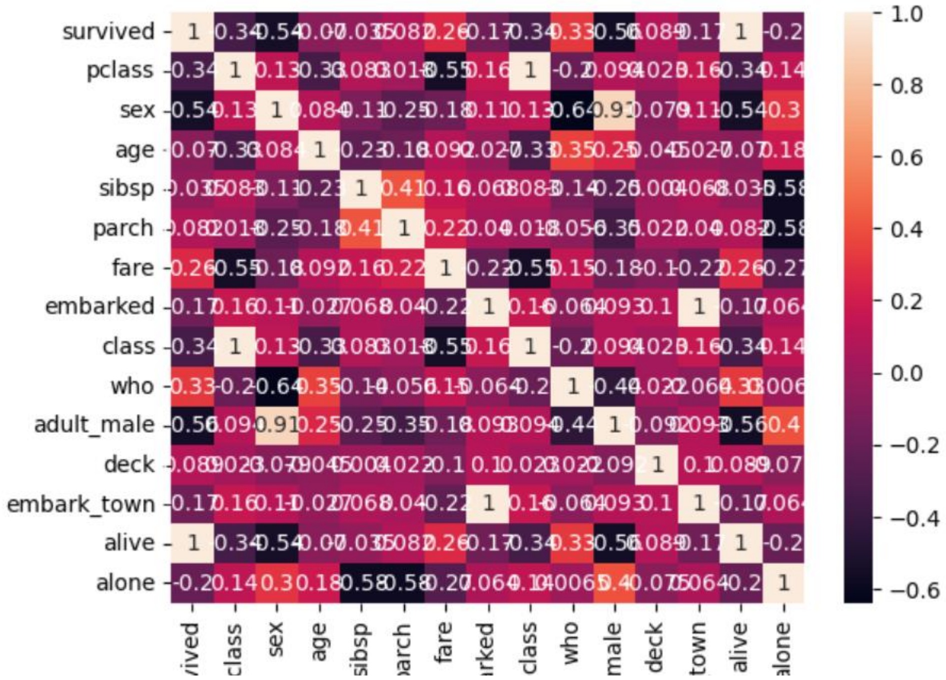
```
Out[32]: <Axes: xlabel='age', ylabel='fare'>
```



● **Multi - Variate Analysis**

```
In [76]: ##mulitivariate analysis

         ##plotted this after encoding the categorical variable
         sns.heatmap(df.corr(), annot=True)
```

```
Out[76]: <Axes: >
```



4. Perform descriptive statistics on the dataset.

```
In [34]: ##ques4

         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   survived     891 non-null    int64  
 1   pclass       891 non-null    int64  
 2   sex          891 non-null    object 
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64  
 5   parch        891 non-null    int64  
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object 
 8   class        891 non-null    object 
 9   who          891 non-null    object 
 10  adult_male   891 non-null    bool   
 11  deck         203 non-null    object 
 12  embark_town  889 non-null    object 
 13  alive        891 non-null    object 
 14  alone        891 non-null    bool   
dtypes: bool(2), float64(2), int64(4), object(7)
memory usage: 92.4+ KB
```

In [39]:
```
df.mean(numeric_only=True)
```

Out[39]:
```
survived       0.383838
pclass         2.308642
age           29.699118
sibsp          0.523008
parch          0.381594
fare          32.204208
adult_male     0.602694
alone          0.602694
dtype: float64
```

In [40]:
```
df.median(numeric_only=True)
```

Out[40]:
```
survived       0.0000
pclass         3.0000
age           28.0000
sibsp          0.0000
parch          0.0000
fare          14.4542
adult_male     1.0000
alone          1.0000
dtype: float64
```

In [35]:
```
df.columns
```

Out[35]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

In [36]:
```
df.describe()
```

Out[36]:

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [41]:
```
df.mode(numeric_only=True)
```

Out[41]:

	survived	pclass	age	sibsp	parch	fare	adult_male	alone
0	0	3	24.0	0	0	8.05	True	True

In [42]:
```
df.var(numeric_only=True)
```

Out[42]:
```
survived         0.236772
pclass           0.699015
age            211.019125
sibsp            1.216043
parch            0.649728
fare          2469.436846
adult_male       0.239723
alone            0.239723
dtype: float64
```

```
In [43]: df.std(numeric_only=True)
```

```
Out[43]: survived       0.486592
         pclass         0.836071
         age           14.526497
         sibsp          1.102743
         parch          0.806057
         fare          49.693429
         adult_male     0.489615
         alone          0.489615
         dtype: float64
```

5. Handle the Missing values.

```
In [45]: #ques5
         df.isna().any()
```

```
Out[45]: survived       False
         pclass         False
         sex            False
         age             True
         sibsp          False
         parch          False
         fare           False
         embarked        True
         class          False
         who            False
         adult_male     False
         deck            True
         embark_town     True
         alive          False
         alone          False
         dtype: bool
```

```
In [47]: df['age'].fillna(df['age'].mean(),inplace=True)
```

```
In [62]: df['embarked'].fillna(df['embarked'].mode()[0],inplace=True)
```

```
In [63]: df['embark_town'].fillna(df['embark_town'].mode()[0],inplace=True)
```

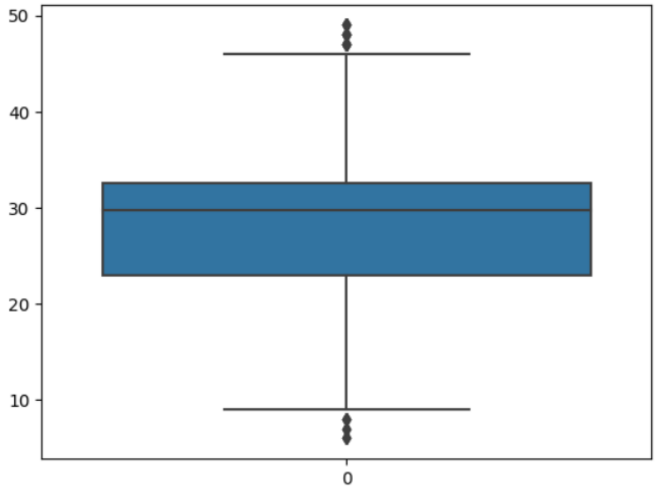
```
In [64]: df['deck'].fillna(df['deck'].mode()[0],inplace=True)
```

```
In [65]: df.isna().any()
```

```
Out[65]: survived       False
         pclass         False
         sex            False
         age            False
         sibsp          False
         parch          False
         fare           False
         embarked       False
         class          False
         who            False
         adult_male     False
         deck           False
         embark_town    False
         alive          False
         alone          False
         dtype: bool
```

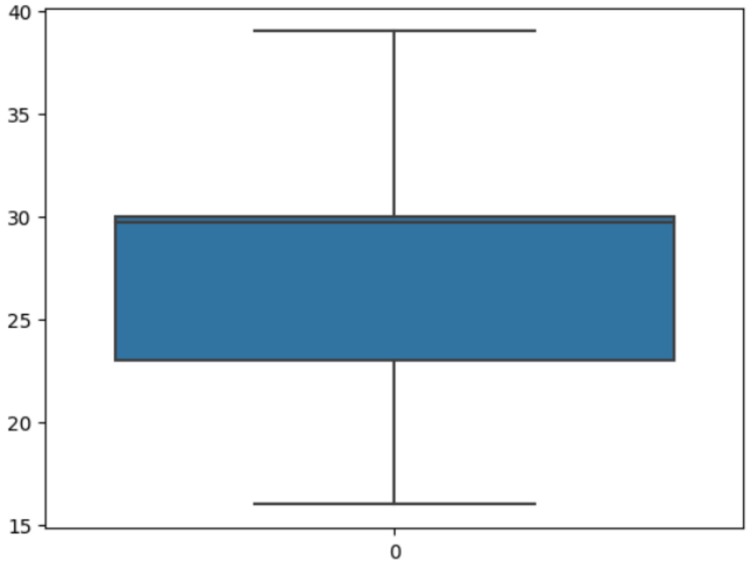
6. Find the outliers and replace the outliers

In [107]:
```
##ques6
sns.boxplot(df.age)
df=df[df.age<40]
df=df[df.age>15]
```



In [108]:
```
sns.boxplot(df.age)
```

Out[108]: <Axes: >



7. Check for Categorical columns and perform encoding.

```
In [75]:
#ques7
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

df.sex=le.fit_transform(df.sex)
df.embarked=le.fit_transform(df.embarked)
df['class']=le.fit_transform(df['class'])
df.who=le.fit_transform(df.who)
df.adult_male=le.fit_transform(df.adult_male)
df.deck=le.fit_transform(df.deck)
df.embark_town=le.fit_transform(df.embark_town)
df.alive=le.fit_transform(df.alive)
df.alone=le.fit_transform(df.alone)
df
```

Out[75]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	1	22.000000	1	0	7.2500	2	2	1	1	2	2	0	0
1	1	1	0	38.000000	1	0	71.2833	0	0	2	0	2	0	1	0
2	1	3	0	26.000000	0	0	7.9250	2	2	2	0	2	2	1	1
3	1	1	0	35.000000	1	0	53.1000	2	0	2	0	2	2	1	0
4	0	3	1	35.000000	0	0	8.0500	2	2	1	1	2	2	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	1	27.000000	0	0	13.0000	2	1	1	1	2	2	0	1
887	1	1	0	19.000000	0	0	30.0000	2	0	2	0	1	2	1	1
888	0	3	0	29.699118	1	2	23.4500	2	2	2	0	2	2	0	0
889	1	1	1	26.000000	0	0	30.0000	0	0	1	1	2	0	1	1
890	0	3	1	32.000000	0	0	7.7500	1	2	1	1	2	1	0	1

8. Split the data into dependent and independent variables.

```
In [89]:
##ques8
##independent
X=df.loc[:, df.columns != 'alive']
X
```

Out[89]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alone
0	0	3	1	22.000000	1	0	7.2500	2	2	1	1	2	2	0
1	1	1	0	38.000000	1	0	71.2833	0	0	2	0	2	0	0
2	1	3	0	26.000000	0	0	7.9250	2	2	2	0	2	2	1
3	1	1	0	35.000000	1	0	53.1000	2	0	2	0	2	2	0
4	0	3	1	35.000000	0	0	8.0500	2	2	1	1	2	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	1	27.000000	0	0	13.0000	2	1	1	1	2	2	1
887	1	1	0	19.000000	0	0	30.0000	2	0	2	0	1	2	1
888	0	3	0	29.699118	1	2	23.4500	2	2	2	0	2	2	0
889	1	1	1	26.000000	0	0	30.0000	0	0	1	1	2	0	1
890	0	3	1	32.000000	0	0	7.7500	1	2	1	1	2	1	1

891 rows × 14 columns

```
In [88]:
#dependent
Y=df.loc[:,'alive']
Y
```

```
Out[88]: 0      0
1      1
2      1
3      1
4      0
      ..
886    0
```

9. Scale the independent variables

In [90]:
```
##ques9
from sklearn.preprocessing import MinMaxScaler
scale=MinMaxScaler()
X_scaled=scale.fit_transform(X)
X_scaled
```

Out[90]:
```
array([[0.        , 1.        , 1.        , ..., 0.33333333, 1.        ,
        0.        ],
       [1.        , 0.        , 0.        , ..., 0.33333333, 0.        ,
        0.        ],
       [1.        , 1.        , 0.        , ..., 0.33333333, 1.        ,
        1.        ],
       ...,
       [0.        , 1.        , 0.        , ..., 0.33333333, 1.        ,
        0.        ],
       [1.        , 0.        , 1.        , ..., 0.33333333, 0.        ,
        1.        ],
       [0.        , 1.        , 1.        , ..., 0.33333333, 0.5       ,
        1.        ]])
```

10. Split the data into training and testing

In [91]:
```
##ques10
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=0)
```

In [92]:
```
X_train.head()
```

Out[92]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alone
140	0	3	0	29.699118	0	2	15.2458	0	2	2	0	2	0	0
439	0	2	1	31.000000	0	0	10.5000	2	1	1	1	2	2	1
817	0	2	1	31.000000	1	1	37.0042	0	1	1	1	2	0	0
378	0	3	1	20.000000	0	0	4.0125	0	2	1	1	2	0	1
491	0	3	1	21.000000	0	0	7.2500	2	2	1	1	2	2	1

In [93]:
```
X_test.head()
```

Out[93]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alone
495	0	3	1	29.699118	0	0	14.4583	0	2	1	1	2	0	1
648	0	3	1	29.699118	0	0	7.5500	2	2	1	1	2	2	1
278	0	3	1	7.000000	4	1	29.1250	1	2	0	0	2	1	0
31	1	1	0	29.699118	1	0	146.5208	0	0	2	0	1	0	0
255	1	3	0	29.000000	0	2	15.2458	0	2	2	0	2	0	0

In [94]:
```
Y_train.head()
```

Out[94]:
```
140    0
439    0
817    0
378    0
491    0
Name: alive, dtype: int64
```

In [96]:
```
Y_test.head()
```

Out[96]:
```
495    0
648    0
278    0
31     1
255    1
Name: alive, dtype: int64
```