Assignment week - 2 of Applied Data Science

Submitted by - Rahul Kumar , Registration number - 20BEC1186

Mail ID - rahulkumar2020a@vitstudent.ac.in (mailto:rahulkumar2020a@vitstudent.ac.in)

Question 1:

1. Download the dataset: Dataset

Question 2:

2. Load the dataset

In [1]:

```
import pandas as pd
data = pd.read_csv("titanic.csv")
data
```

Out[1]:

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

891 rows × 15 columns

Question 3:

3. Perform Below Visualizations.

● Univariate Analysis

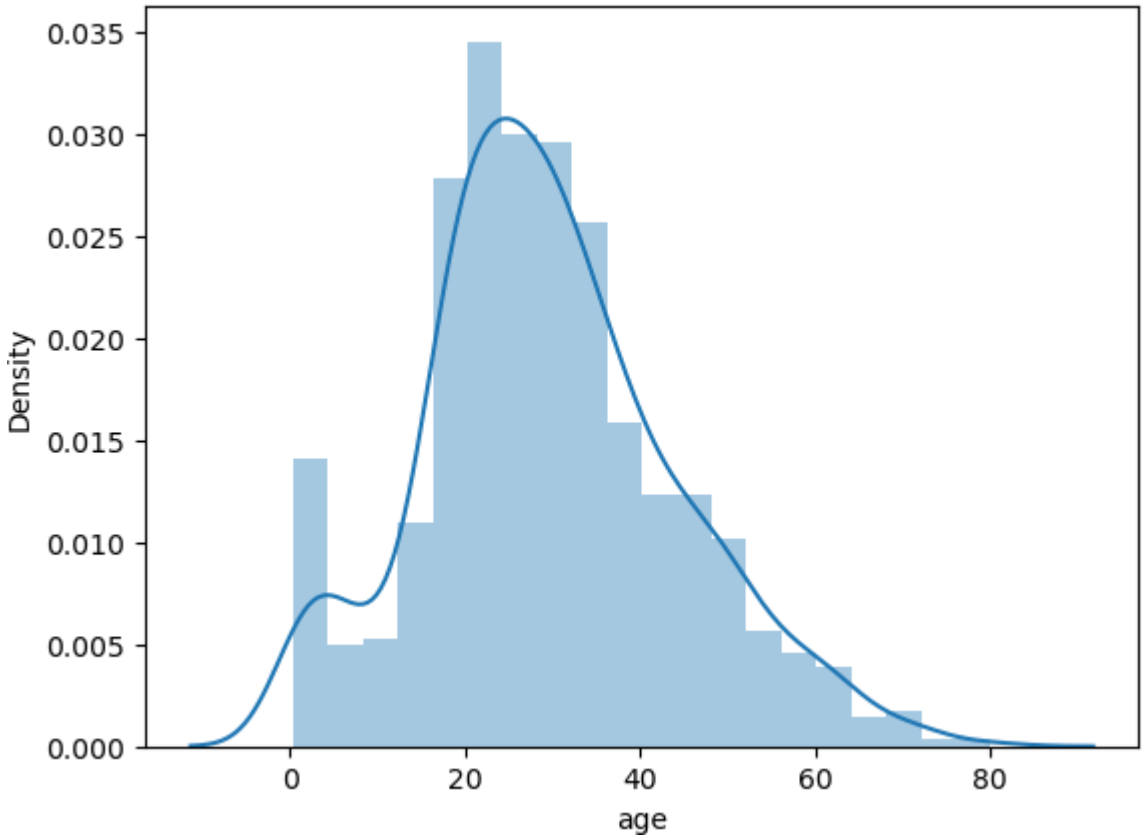
In [4]:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.distplot(data['age'])
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\distributions.py:2619: Future
Warning: `distplot` is a deprecated function and will be removed in a future ve
rsion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

Out[4]:

```
<AxesSubplot:xlabel='age', ylabel='Density'>
```



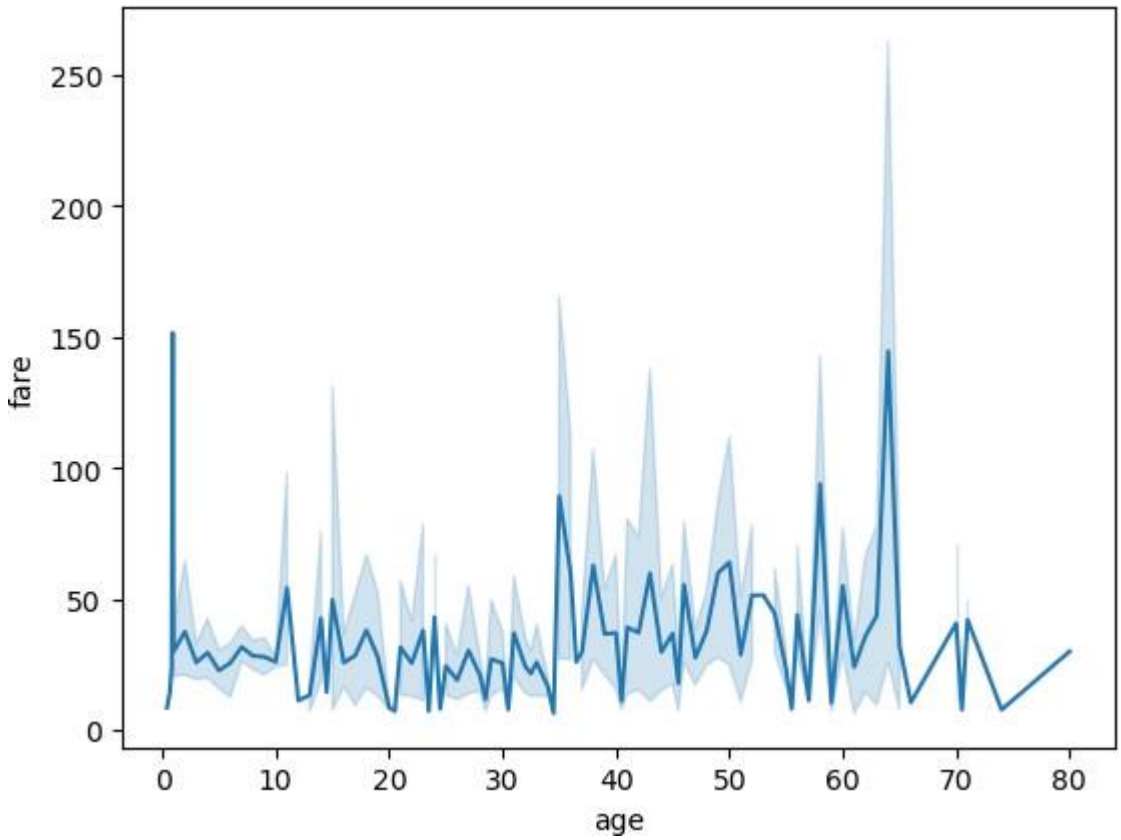
● Bi - Variate Analysis

In [5]:

```
#line plot for the bi-variate Analysis
sns.lineplot(x=data.age , y = data.fare)
```

Out[5]:

```
<AxesSubplot:xlabel='age', ylabel='fare'>
```



- **Multi - Variate Analysis**

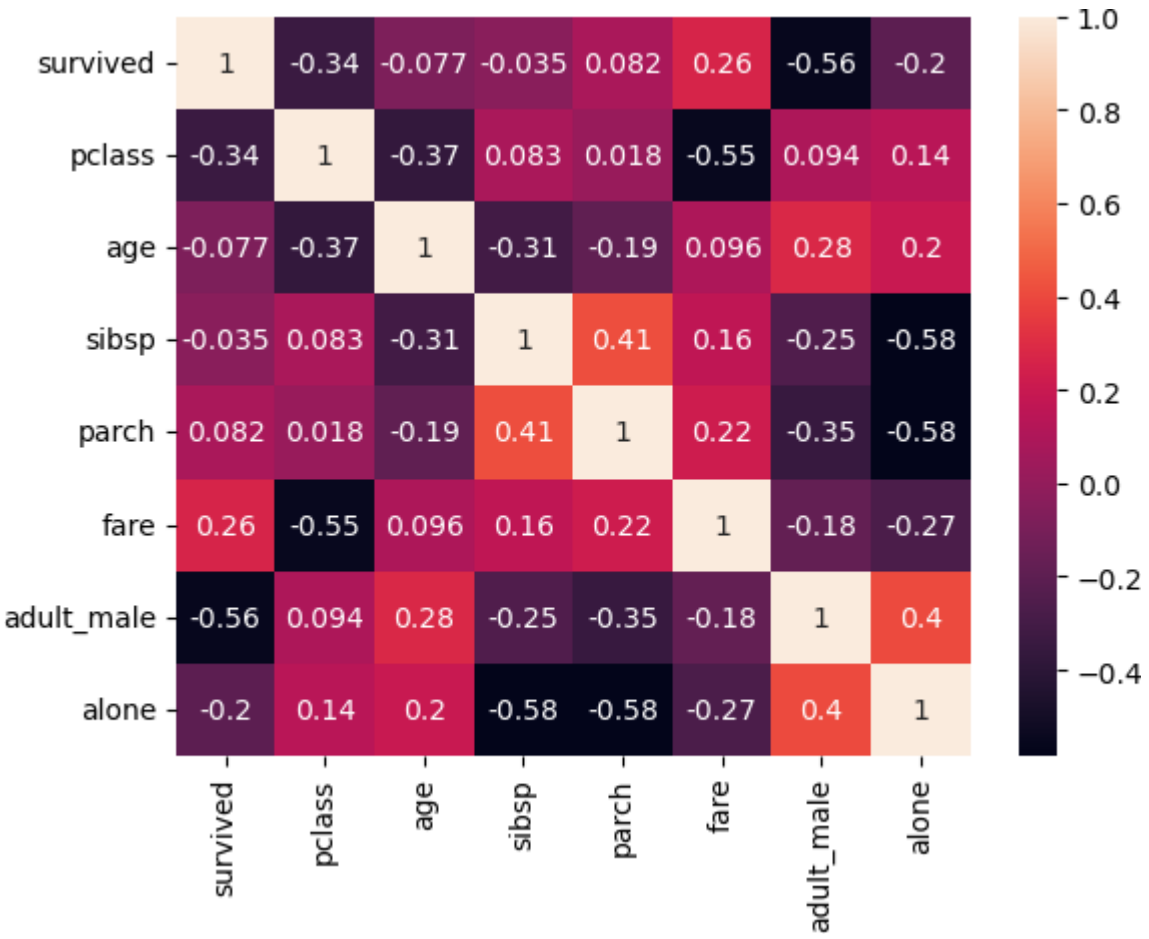
In [6]:

```
#heat plot for the multi variate Analysis
# plotted this without encoding the categorical variable

sns.heatmap(data.corr(), annot = True)
```

Out[6]:

<AxesSubplot:>



Question 4:

4. Perform descriptive statistics on the dataset.

In [7]:

```
import pandas as pd
print(data)
```

```
     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
..     ...         ...  ...          ...   ...    ...
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True

[891 rows x 15 columns]
```

In [8]:

```
# printing and getting the information about the data set

data.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 [9]:

```
#mean median and mode of every column of only numeric containg column
data.mean(numeric only = True)
```

Out[9]:

```
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 [12]:

```
# without using the only numeric function
data.median()
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_10288\4184645713.py:1: FutureWarnin
g: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=Non
e') is deprecated; in a future version this will raise TypeError. Select only
valid columns before calling the reduction.
  data.median()
```

Out[12]:

```
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 [13]:

```
data.mode(numeric_only = True)
```

Out[13]:

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

In [14]:

```
# variance
data.var(numeric_only = True)
```

Out[14]:

```
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 [16]:

```
data.std(numeric_only = True)
```

Out[16]:

```
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
```

In [15]:

```
data.columns
```

Out[15]:

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

Question 5:

5. Handle the Missing values.

In [19]:

```
data.isna()
```

Out[19]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	False	False	False	False	False	False	False	False	False	False	False	True
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	True
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	True
...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	False	True
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	True	False	False	False	False	False	False	False	True
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	False	True

891 rows × 15 columns

In [21]:

```
data.isna().any()
```

Out[21]:

```
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 [23]:

```
data.age.fillna(data.age.mean() , inplace = True)
```

In [29]:

```
data['embarked'].fillna(data['embarked'].mode()[0] , inplace = True)
```

In [31]:

```
data['embark town'].fillna(data['embark town'].mode()[0] , inplace = True)
```

In [32]:

```
data['deck'].fillna(data['deck'].mode()[0] , inplace = True)
```

In [33]:

```
data.isna().any()
```

Out[33]:

```
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
```

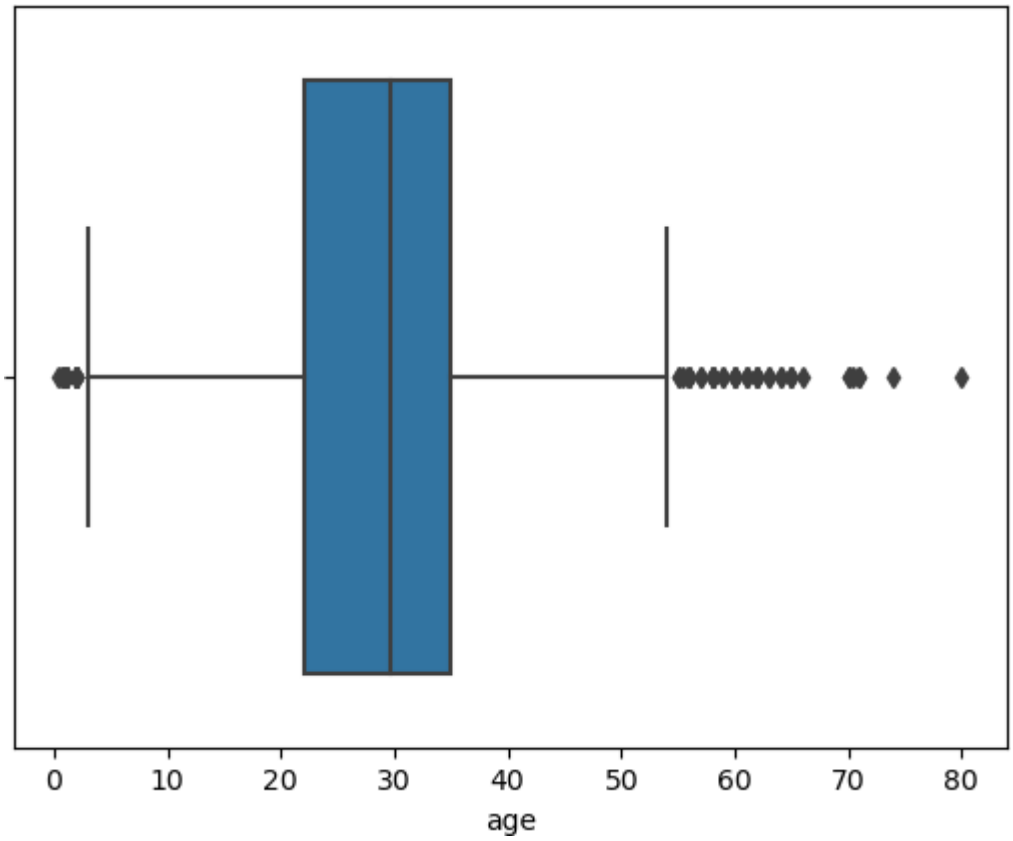
Question 6:

6. Find the outliers and replace the outliers

In [34]:

```
#boxplot will help us to find outlier
sns.boxplot(data.age)
data = data[data.age < 40]
data = data[data.age > 15]
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarn
ing: Pass the following variable as a keyword arg: x. From version 0.12, the on
ly valid positional argument will be `data`, and passing other arguments withou
t an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



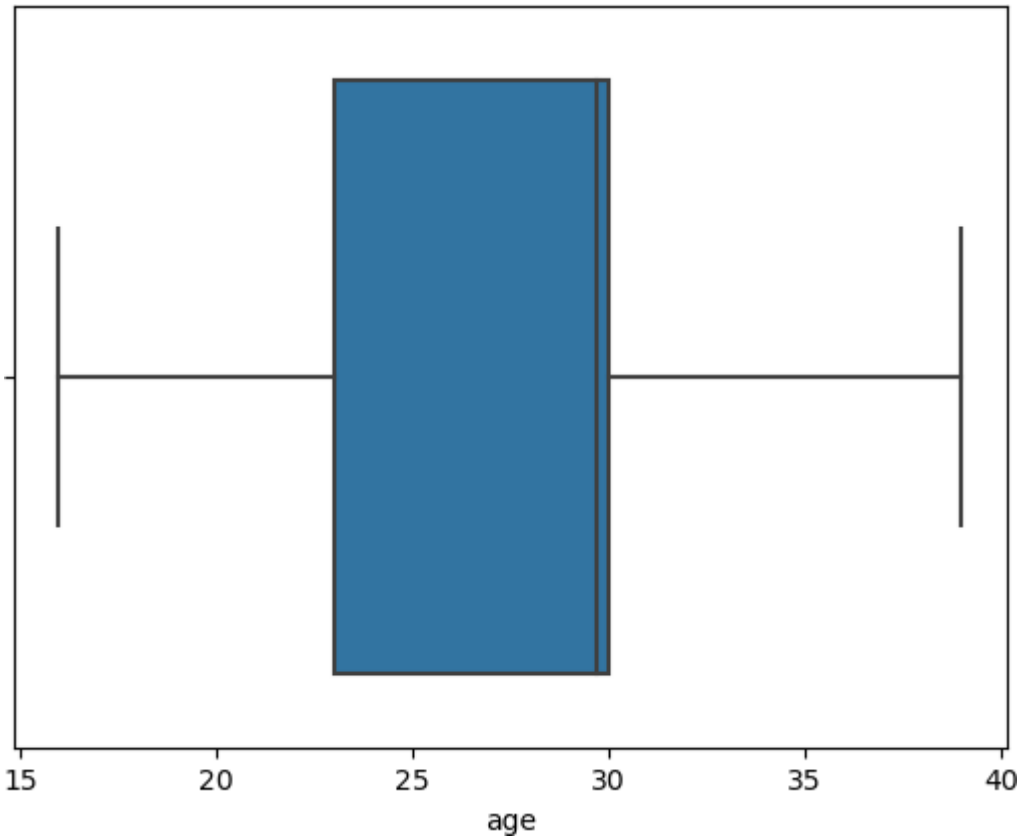
In [35]:

```
plot = sns.boxplot(data.age)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarn
ing: Pass the following variable as a keyword arg: x. From version 0.12, the on
ly valid positional argument will be `data`, and passing other arguments withou
t an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[35]:

```
<AxesSubplot:xlabel='age'>
```



Question 7:

7. Check for Categorical columns and perform encoding.

In [39]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

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

Out[39]:

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

645 rows × 15 columns

Question 8:

8. Split the data into dependent and independent variables.

In [42]:

```
#independent
X = data.loc[: , data.columns != 'alive']
X
```

Out[42]:

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

645 rows × 14 columns

In [43]:

```
Y = data.loc[:, 'alive']
Y
```

Out[43]:

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: alive, Length: 645, dtype: int32
```

Question 9:

9. Scale the independent variables

In [45]:

```
from sklearn.preprocessing import MinMaxScaler
scale = MinMaxScaler()
X_scaled = scale.fit_transform(X)
X_scaled
```

Out[45]:

```
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.        ]])
```

Question 10:

10. Split the data into training and testing

In [46]:

```
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
```

In [48]:

```
X_train.head()
```

Out[48]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	d
713	0	3	1	29.000000	0	0	9.4833	2	2	1	True	
45	0	3	1	29.699118	0	0	8.0500	2	2	1	True	
378	0	3	1	20.000000	0	0	4.0125	0	2	1	True	
795	0	2	1	39.000000	0	0	13.0000	2	1	1	True	
595	0	3	1	36.000000	1	1	24.1500	2	2	1	True	

In [49]:

```
X test.head()
```

Out[49]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	d
877	0	3	1	19.000000	0	0	7.8958	2	2	1	True	
815	0	1	1	29.699118	0	0	0.0000	2	0	1	True	
826	0	3	1	29.699118	0	0	56.4958	2	2	1	True	
399	1	2	0	28.000000	0	0	12.6500	2	1	0	False	
99	0	2	1	34.000000	1	0	26.0000	2	1	1	True	

In [50]:

```
Y train.head()
```

Out[50]:

```
713    0
45     0
378    0
795    0
595    0
Name: alive, dtype: int32
```

In [51]:

```
Y_test.head()
```

Out[51]:

```
877    0
815    0
826    0
399    1
99     0
Name: alive, dtype: int32
```