NAME : MOUNIKAA V

REG NO: 20BIT0050

ASSIGNMENT -02

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]:
```
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	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

891 rows × 15 columns

3. Perform Below Visualizations.

● Univariate Analysis-

The term **univariate analysis** refers to the analysis of one variable.

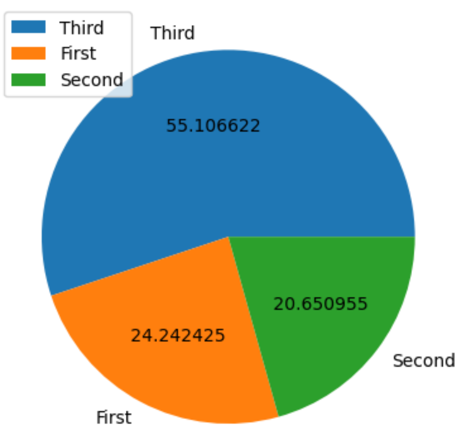
Here I'm performing analysis of the variable class and displaying it in the form of piechart

```
import matplotlib.pyplot as plt
abc=data['class'].value_counts()
abc
```

```
Third     491
First     216
Second    184
Name: class, dtype: int64
```

```
list = data['class'].unique()
plt.pie(abc, autopct='% 2f',labels=list)
plt.legend()
```

```
<matplotlib.legend.Legend at 0x18f539be740>
```



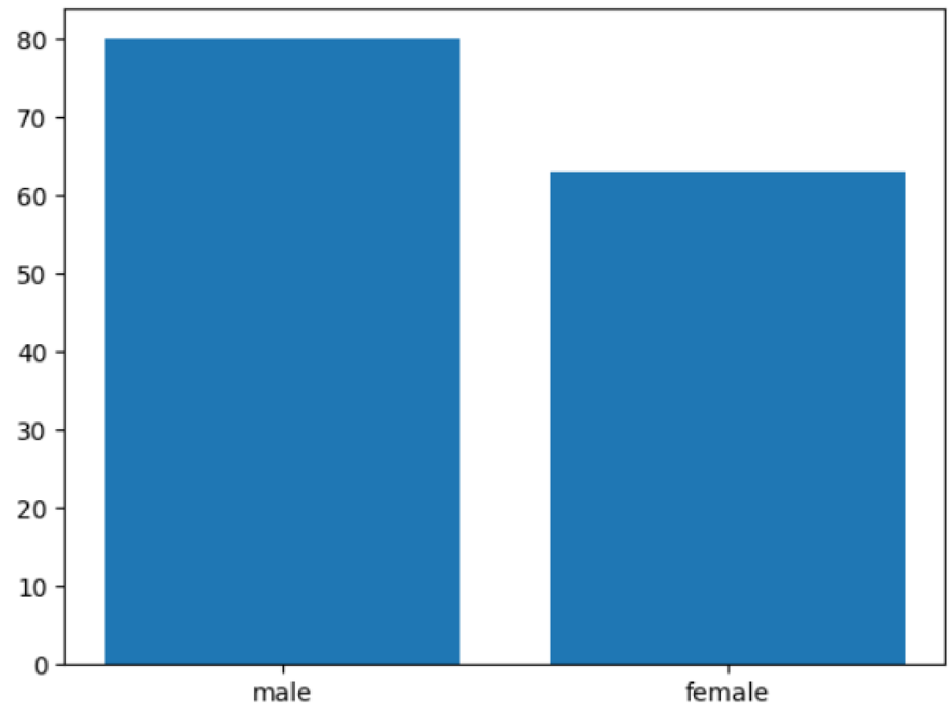
● Bi - Variate Analysis

The term bivariate analysis refers to the analysis of two variables.

Here I'm using the variables sex and age to plot a bar graph.

```
plt.bar(data['sex'],data['age'])
```

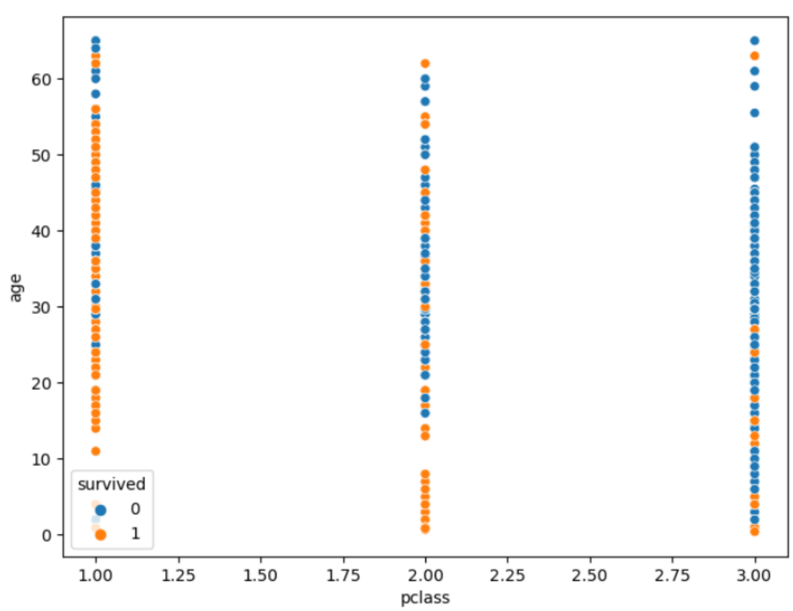
```
<BarContainer object of 891 artists>
```



● **Multi - Variate Analysis**

The term multivariate analysis refers to analysis of multiple variables.

```
import seaborn as sns
plt.figure(figsize=(8,6))
sns.scatterplot(data=data,x="pclass",y="age",hue="survived")
plt.show()
```



4. Perform descriptive statistics on the dataset.

data.describe( )

```
data.describe()
```

	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

```
data.mean()
```

```
C:\Users\91887\AppData\Local\Temp\ipykernel_23284\531903386.py:1: FutureWarning: The default val
e.mean is deprecated. In a future version, it will default to False. In addition, specifying 'nu
Select only valid columns or specify the value of numeric_only to silence this warning.
  data.mean()
```

```
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
```

```
data.median()
```

```
C:\Users\91887\AppData\Local\Temp\ipykernel_23284\4184645713.py:1: FutureWarning: The default va
e.median is deprecated. In a future version, it will default to False. In addition, specifying '
d. Select only valid columns or specify the value of numeric_only to silence this warning.
  data.median()
```

```
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
```

```
data.mode()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	24.0	0	0	8.05	S	Third	man	True	C	Southampton	no	True

```
data.var()
```

```
C:\Users\91887\AppData\Local\Temp\ipykernel_23284\445316826.py:1: FutureWarning: The default value of numeric_on
e.var is deprecated. In a future version, it will default to False. In addition, specifying 'numeric_only=None'
Select only valid columns or specify the value of numeric_only to silence this warning.
  data.var()
```

```
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
```

```
data.std()
```

```
C:\Users\91887\AppData\Local\Temp\ipykernel_23284\2723740006.py:1: FutureWarning: The default value of numeric_o
e.std is deprecated. In a future version, it will default to False. In addition, specifying 'numeric_only=None'
Select only valid columns or specify the value of numeric_only to silence this warning.
  data.std()
```

```
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.

missing values

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

```
data.info()
```

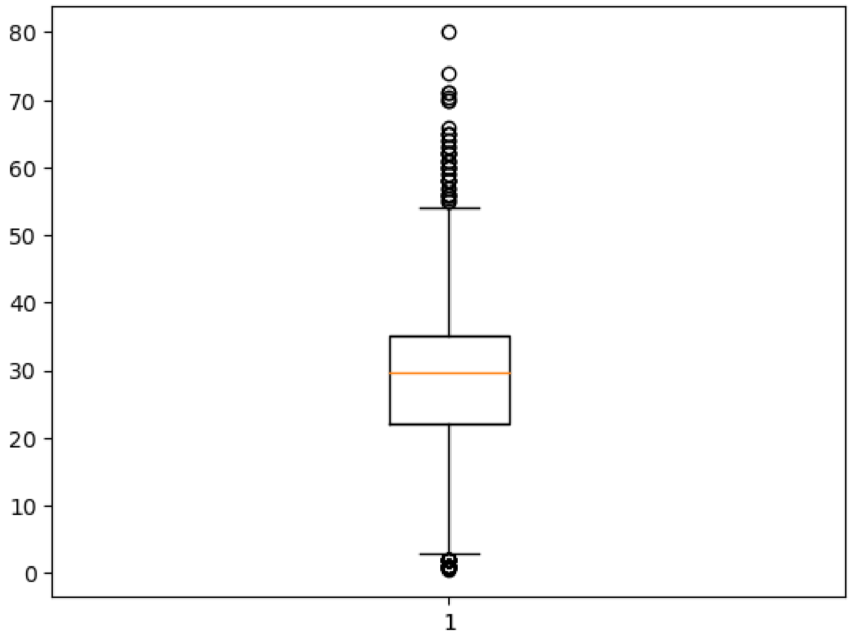
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   survived     891 non-null    int64  
 1   pclass       891 non-null    int64  
 2   sex          891 non-null    object 
 3   age          891 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
```

6. Find the outliers and replace the outliers

Finding the outliers

```
plt.boxplot(data.age)
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x18f5eace710>,
  <matplotlib.lines.Line2D at 0x18f5eace9b0>],
 'caps': [<matplotlib.lines.Line2D at 0x18f5eacec50>,
  <matplotlib.lines.Line2D at 0x18f5eaceef0>],
 'boxes': [<matplotlib.lines.Line2D at 0x18f5eace470>],
 'medians': [<matplotlib.lines.Line2D at 0x18f5eacf190>],
 'fliers': [<matplotlib.lines.Line2D at 0x18f5eacf430>],
 'means': []}
```



Replacing outliers

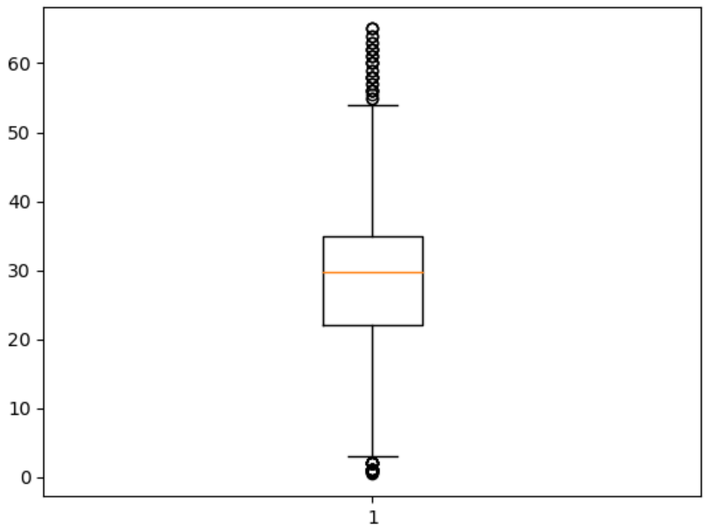
```
perc99=data.age.quantile(0.99)
perc99
```

```
65.0
```

```
data=data[data.age<=perc99]
```

```
plt.boxplot(data['age'])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x18f5ec50850>,
  <matplotlib.lines.Line2D at 0x18f5ec50af0>],
 'caps': [<matplotlib.lines.Line2D at 0x18f5ec50d90>,
  <matplotlib.lines.Line2D at 0x18f5ec51030>],
 'boxes': [<matplotlib.lines.Line2D at 0x18f5ec505b0>],
 'medians': [<matplotlib.lines.Line2D at 0x18f5ec512d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x18f5ec51570>],
 'means': []}
```



7. Check for Categorical columns and perform encoding.

Categorical variables are:

Sex

Adult_male

Alive

alone

```
data.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
```

```
data.sex=le.fit_transform(data.sex)
```

```
data.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	1	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	0	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	0	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	0	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	1	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

```
data.adult_male=le.fit_transform(data.adult_male)
data.alive=le.fit_transform(data.alive)
data.alone=le.fit_transform(data.alone)
data.sex=le.fit_transform(data.sex)
data.who=le.fit_transform(data.who)
```

```
data.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	1	22.0	1	0	7.2500	S	Third	1	1	NaN	2	0	0
1	1	1	0	38.0	1	0	71.2833	C	First	2	0	C	0	1	0
2	1	3	0	26.0	0	0	7.9250	S	Third	2	0	NaN	2	1	1
3	1	1	0	35.0	1	0	53.1000	S	First	2	0	C	2	1	0
4	0	3	1	35.0	0	0	8.0500	S	Third	1	1	NaN	2	0	1

8. Split the data into dependent and independent variables.

y- dependent variable

x- independent variable

```
y=data['fare']
y.head()
```

```
0     7.2500
1    71.2833
2     7.9250
3    53.1000
4     8.0500
Name: fare, dtype: float64
```

```
X=data.drop(columns=['fare'],axis=1)
X.head()
```

	survived	pclass	sex	age	sibsp	parch	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	1	22.0	1	0	S	Third	1	1	NaN	2	0	0
1	1	1	0	38.0	1	0	C	First	2	0	C	0	1	0
2	1	3	0	26.0	0	0	S	Third	2	0	NaN	2	1	1
3	1	1	0	35.0	1	0	S	First	2	0	C	2	1	0
4	0	3	1	35.0	0	0	S	Third	1	1	NaN	2	0	1

9. Scale the independent variables

```
from sklearn.preprocessing import StandardScaler

scale=StandardScaler()

scale.fit_transform(x)
```

```
array([[ 0.82520863, -1.55536747,  1.34530257, ...,  0.61679395,
         0.78696114, -0.78696114],
       [-1.57221121,  0.33947168,  1.34530257, ..., -1.62128697,
        -1.27071078,  1.27071078],
       [ 0.82520863, -0.57092264, -0.5713224 , ...,  0.61679395,
        -1.27071078,  1.27071078],
       ...,
       [ 0.82520863,  0.33947168,  1.34530257, ...,  0.61679395,
         0.78696114, -0.78696114],
       [-1.57221121, -0.57092264, -0.5713224 , ..., -1.62128697,
        -1.27071078,  1.27071078],
       [ 0.82520863,  0.90574462, -0.5713224 , ..., -1.62128697,
         0.78696114, -0.78696114]])
```

10. Split the data into training and testing

```
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)
```

```
X_train.shape
```

```
(706, 14)
```

```
X_train.head()
```

	survived	pclass	sex	age	sibsp	parch	embarked	class	who	adult_male	deck	embark_town	alive	alone
772	0	2	0	57.000000	0	0	S	Second	2	0	E	2	0	1
270	0	1	1	29.699118	0	0	S	First	1	1	NaN	2	0	1
722	0	2	1	34.000000	0	0	S	Second	1	1	NaN	2	0	1
205	0	3	0	2.000000	0	1	S	Third	0	0	G	2	0	0
875	1	3	0	15.000000	0	0	C	Third	0	0	NaN	0	1	1

```
X_test.shape
```

```
(177, 14)
```

```
X_test.head()
```

	survived	pclass	sex	age	sibsp	parch	embarked	class	who	adult_male	deck	embark_town	alive	alone
588	0	3	1	22.0	0	0	S	Third	1	1	NaN	2	0	1
684	0	2	1	60.0	1	1	S	Second	1	1	NaN	2	0	0
734	0	2	1	23.0	0	0	S	Second	1	1	NaN	2	0	1
341	1	1	0	24.0	3	2	S	First	2	0	C	2	1	0
574	0	3	1	16.0	0	0	S	Third	1	1	NaN	2	0	1

```
y_train
```

```
772     10.5000
270     31.0000
722     13.0000
205     10.4625
875      7.2250
         ...
842     31.0000
195    146.5208
634     27.9000
563      8.0500
690     57.0000
Name: fare, Length: 706, dtype: float64
```

```
y_test
```

```
588      8.0500
684     39.0000
734     13.0000
341    263.0000
574      8.0500
         ...
478      7.5208
367      7.2292
143      6.7500
218     76.2917
309     56.9292
Name: fare, Length: 177, dtype: float64
```