```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

```
data=pd.read_csv(r'titanic.csv')
data
```

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

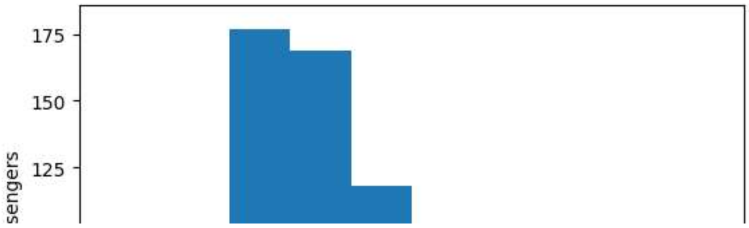
891 rows × 15 columns

```
data.info()
```

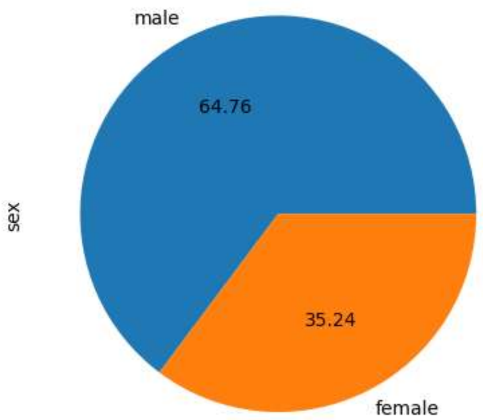
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   survived     891 non-null    int64  
 1   pclass       891 non-null    int64  
 2   sex          891 non-null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

** UNIVARIATE ANALYSIS**

```
plt.hist(data["age"])
plt.xlabel("Age of the passengers")
plt.ylabel("number of passengers")
plt.show()
```



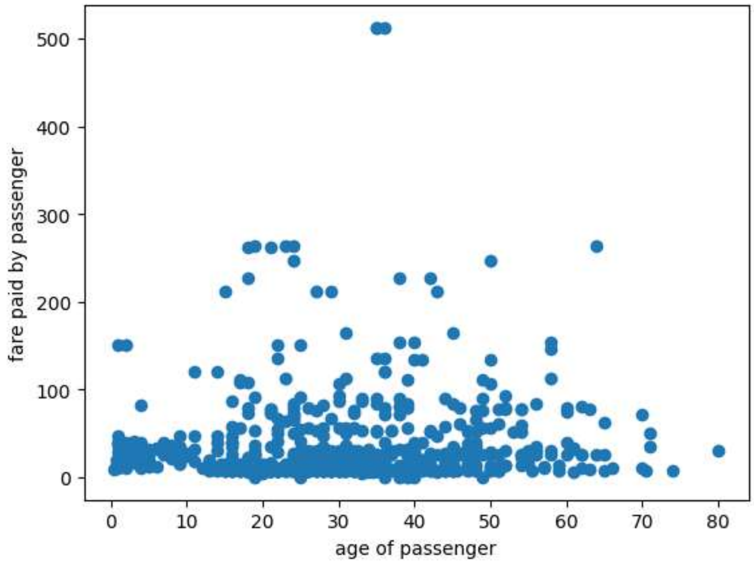

```
data['sex'].value_counts().plot(kind="pie", autopct="%.2f")
plt.show()
```



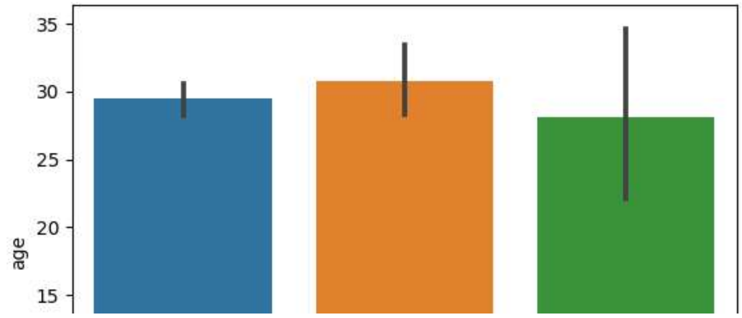
Bivariate Analysis

```
plt.scatter(data["age"], data["fare"])
plt.ylabel("fare paid by passenger")
plt.xlabel("age of passenger")
```

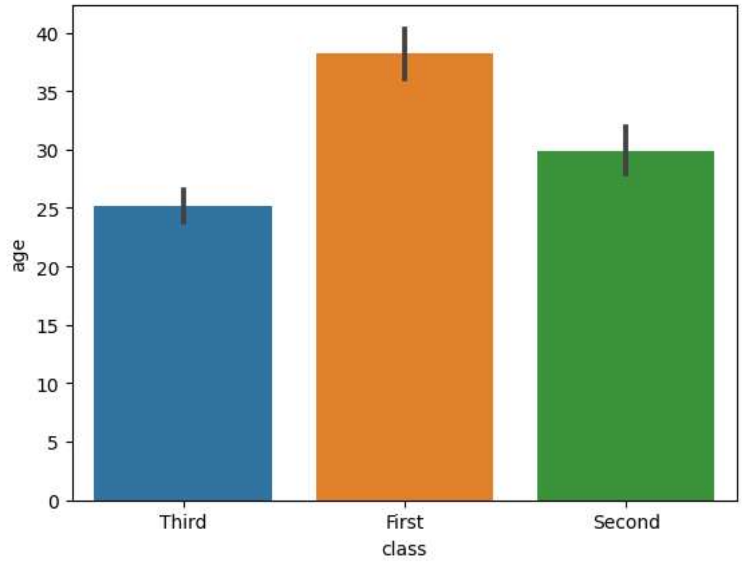
```
Text(0.5, 0, 'age of passenger')
```




```
sns.barplot(data=data ,x="embark_town", y="age")
plt.show()
```

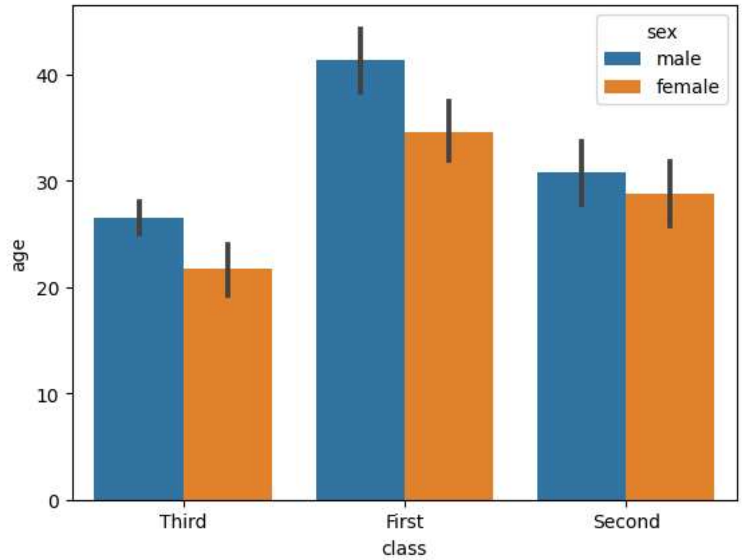



```
sns.barplot(data=data ,x="class", y="age")
plt.show()
```



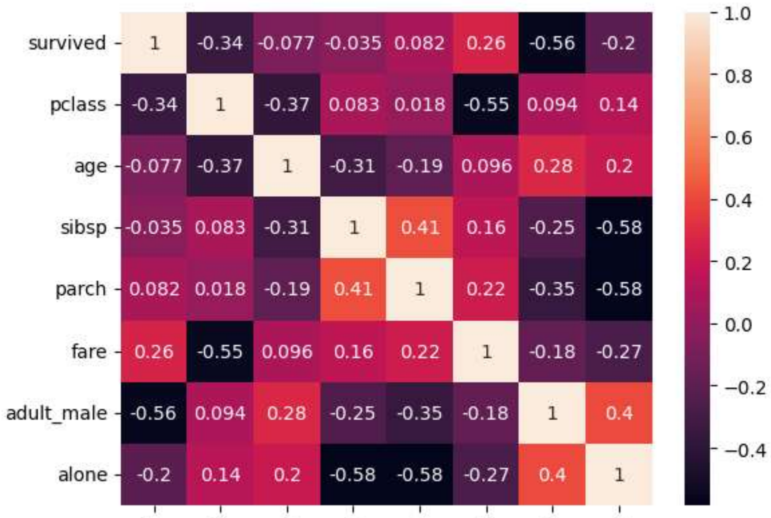
Multivariate analysis

```
sns.barplot(data=data, x='class',y='age',hue='sex')
plt.show()
```

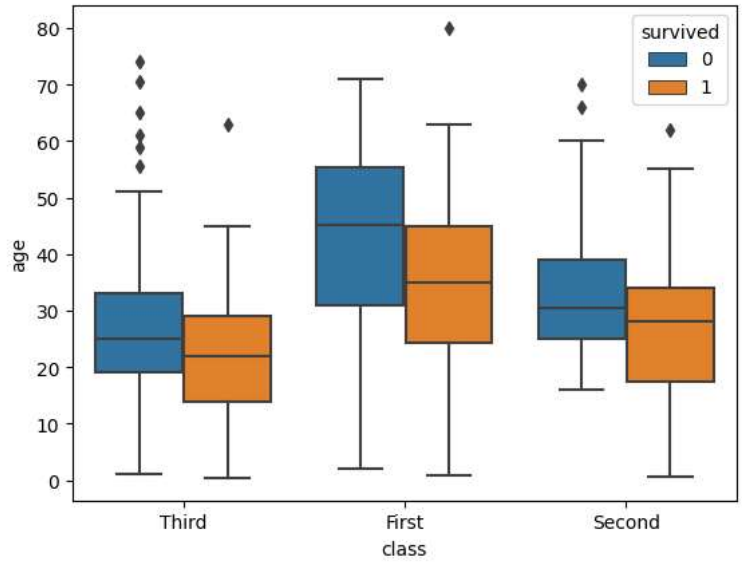



```
corr_matrix = data.corr()
sns.heatmap(corr_matrix, annot=True)
plt.show()
```

```
<ipython-input-17-62cc2e9fa4f2>:1: FutureWarning: The default value of numeric_only in DataFrame.corr i
  corr_matrix = data.corr()
```



```
sns.boxplot(data=data,x=data['class'], y=data["age"], hue=data["survived"])
plt.show()
```



Descriptive statistics on the dataset.

```
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

```
column_name = 'fare'

count = data[column_name].count()
mean = data[column_name].mean()
median = data[column_name].median()
mode = data[column_name].mode()
minimum = data[column_name].min()
maximum = data[column_name].max()
```

```
range_val = maximum - minimum
value_counts = data[column_name].value_counts()
null_counts = data[column_name].isnull().sum()
percentile_25 = data[column_name].quantile(0.25)
percentile_50 = data[column_name].quantile(0.50)
percentile_75 = data[column_name].quantile(0.75)
sum_val = data[column_name].sum()
q1 = data[column_name].quantile(0.25)
q3 = data[column_name].quantile(0.75)
iqr = q3 - q1
variance = data[column_name].var()
std_deviation = data[column_name].std()
covariance = data['pclass'].cov(data['survived'])
correlation = data['pclass'].corr(data['survived'])
abs_deviation = data[column_name].mad()
kur=data.kurt()
skew=data.skew()
```

```
<ipython-input-20-3058a9f5daf5>:23: FutureWarning: The 'mad' method is deprecated and will be removed in a future version. To compu
  abs_deviation = data[column_name].mad()
<ipython-input-20-3058a9f5daf5>:24: FutureWarning: The default value of numeric_only in DataFrame.kurt is deprecated. In a future v
  kur=data.kurt()
<ipython-input-20-3058a9f5daf5>:25: FutureWarning: The default value of numeric_only in DataFrame.skew is deprecated. In a future v
  skew=data.skew()
```

```
print("Descriptive Statistics based on age")
print("Count:", count)
print("Mean:", mean)
print("Median:", median)
print("Mode:", mode)
print("Minimum:", minimum)
print("Maximum:", maximum)
print("Range:", range_val)
print("Value Counts:")
print(value_counts)
print("Null Value Counts:", null_counts)
print("25th Percentile:", percentile_25)
print("50th Percentile:", percentile_50)
print("75th Percentile:", percentile_75)
print("Sum:", sum_val)
print("Interquartile Range (IQR):", iqr)
print("Variance:", variance)
print("Standard Deviation:", std_deviation)
print("Covariance:", covariance)
print("Correlation:", correlation)
print("Absolute Deviation:", abs_deviation)
print("Kurtosis:",kur)
print("skewness",skew)
```

```
Descriptive Statistics based on age
Count: 891
Mean: 32.204207968574636
Median: 14.4542
Mode: 0    8.05
Name: fare, dtype: float64
Minimum: 0.0
Maximum: 512.3292
Range: 512.3292
Value Counts:
8.0500     43
13.0000    42
7.8958     38
7.7500     34
26.0000    31
           ..
35.0000     1
28.5000     1
6.2375      1
14.0000     1
10.5167     1
Name: fare, Length: 248, dtype: int64
Null Value Counts: 0
25th Percentile: 7.9104
50th Percentile: 14.4542
75th Percentile: 31.0
Sum: 28693.9493
Interquartile Range (IQR): 23.0896
Variance: 2469.436845743116
```

```
Standard Deviation: 49.6934285971809
Covariance: -0.13770287141073634
Correlation: -0.33848103596101475
Absolute Deviation: 28.163691848778342
Kurtosis: survived      -1.775005
pclass        -1.280015
age            0.178274
sibsp         17.880420
parch          9.778125
fare          33.398141
adult_male    -1.827345
alone         -1.827345
dtype: float64
skewness survived      0.478523
pclass       -0.630548
age           0.389108
sibsp         3.695352
parch         2.749117
fare          4.787317
adult_male   -0.420431
alone        -0.420431
dtype: float64
```

Dealing with missing values

```
a=data.isnull().sum()
a
```

```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

```
data1=data['fare'].fillna(data['fare'].mean(), inplace=True)
data1=data.dropna(axis=1, inplace=True)
data1
```

```
data1
```

Finding the outliers and replacing the outliers

```
column_name = 'fare'
mean=data[column_name].mean()
std=data[column_name].std()
z_scores = (data[column_name] - mean) / std
outliers = data[np.abs(z_scores) > 3]
data.loc[np.abs(z_scores) > 3, column_name] = mean
```

```
data
```

	survived	pclass	sex	sibsp	parch	fare	class	who	adult_male	alive	alone
0	0	3	male	1	0	7.2500	Third	man	True	no	False
1	1	1	female	1	0	71.2833	First	woman	False	yes	False
4	0	3	male	0	0	8.0500	Third	man	True	no	True

Check for Categorical columns and perform encoding.

```
Categorical_columns = data.select_dtypes(include=['object', 'category']).columns
Encoded_data = pd.get_dummies(data, columns=Categorical_columns)

Encoded_data
```

	survived	pclass	sibsp	parch	fare	adult_male	alone	sex_female	sex_male	class_First	clas
0	0	3	1	0	7.2500	True	False	0	1	0	
1	1	1	1	0	71.2833	False	False	1	0	1	
2	1	3	0	0	7.9250	False	True	1	0	0	
3	1	1	1	0	53.1000	False	False	1	0	1	
4	0	3	0	0	8.0500	True	True	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	
886	0	2	0	0	13.0000	True	True	0	1	0	
887	1	1	0	0	30.0000	False	True	1	0	1	
888	0	3	1	2	23.4500	False	False	1	0	0	
889	1	1	0	0	30.0000	True	True	0	1	1	
890	0	3	0	0	7.7500	True	True	0	1	0	

891 rows × 17 columns

Split the data into dependent and independent variables.

```
#here alive and survived are dependent variables and rest are independent
dep={'survived','alive'}
dep_data=data[dep]
indep_data=data.drop(columns=dep)
```

```
<ipython-input-27-10332cbb73fa>:3: FutureWarning: Passing a set as an indexer is deprecated and will raise in a future version. Use
  dep_data=data[dep]
```

```
dep_data
```

	survived	alive
0	0	no
1	1	yes
2	1	yes
3	1	yes
4	0	no
...	...	...
886	0	no
887	1	yes
888	0	no
889	1	yes
890	0	no

891 rows × 2 columns

```
indep_data
```

	pclass	sex	sibsp	parch	fare	class	who	adult_male	alone
0	3	male	1	0	7.2500	Third	man	True	False
1	1	female	1	0	71.2833	First	woman	False	False
2	3	female	0	0	7.9250	Third	woman	False	True
3	1	female	1	0	53.1000	First	woman	False	False
4	3	male	0	0	8.0500	Third	man	True	True
...	...	...	...	...	...	...	...	...	...
886	2	male	0	0	13.0000	Second	man	True	True
887	1	female	0	0	30.0000	First	woman	False	True
888	3	female	1	2	23.4500	Third	woman	False	False
889	1	male	0	0	30.0000	First	man	True	True
890	3	male	0	0	7.7500	Third	man	True	True

891 rows × 9 columns

Scale the independent variables

```
indep_data=indep_data.drop(columns=["sex","class","who","adult_male","alone"],axis=1)
independent_var=indep_data.values
independent_var
```

```
array([[ 3.    ,  1.    ,  0.    ,  7.25  ],
       [ 1.    ,  1.    ,  0.    , 71.2833],
       [ 3.    ,  0.    ,  0.    ,  7.925 ],
       ...,
       [ 3.    ,  1.    ,  2.    , 23.45  ],
       [ 1.    ,  0.    ,  0.    , 30.    ],
       [ 3.    ,  0.    ,  0.    ,  7.75  ]])
```

```
sclr=StandardScaler()
scaled_data=sclr.fit_transform(independent_var)
scaled_data
```

```
array([[ 0.82737724,  0.43279337, -0.47367361, -0.66886993],
       [-1.56610693,  0.43279337, -0.47367361,  1.53800237],
       [ 0.82737724, -0.4745452 , -0.47367361, -0.64560642],
       ...,
       [ 0.82737724,  0.43279337,  2.00893337, -0.11054588],
       [-1.56610693, -0.4745452 , -0.47367361,  0.11519625],
       [ 0.82737724, -0.4745452 , -0.47367361, -0.6516377 ]])
```

```
dependent_var=dep_data.values
dependent_var
```

```
array([[0, 'no'],
       [1, 'yes'],
       [1, 'yes'],
       ...,
       [0, 'no'],
       [1, 'yes'],
       [0, 'no']], dtype=object)
```

Split the data into training and testing

```
x_train,x_test,y_train,y_test=train_test_split(scaled_data,dependent_var,test_size=0.3,random_state=10)
```

```
x_train
```

```
array([[ 0.82737724,  2.24747049,  0.76762988, -0.19239894],
       [-1.56610693,  0.43279337, -0.47367361,  1.76805256],
       [-1.56610693, -0.4745452 , -0.47367361,  0.30475071],
       ...,
       [-1.56610693, -0.4745452 , -0.47367361,  0.19116306],
       [ 0.82737724,  0.43279337, -0.47367361, -0.53129819],
       [-0.36936484, -0.4745452 , -0.47367361, -0.55686047]])
```

```
x_test
```

```
array([[ 0.82737724, -0.4745452 , -0.47367361, -0.67317798],
       [ 0.82737724, -0.4745452 , -0.47367361, -0.67576282],
       [ 0.82737724, -0.4745452 , -0.47367361, -0.64661279],
       ...,
       [ 0.82737724, -0.4745452 ,  2.00893337, -0.22212453],
       [ 0.82737724, -0.4745452 , -0.47367361, -0.64804651],
       [ 0.82737724, -0.4745452 , -0.47367361, -0.6471849 ]])
```

```
y_train
```

```
array([[0, 'no'],
       [1, 'yes'],
       [0, 'no'],
       ...,
       [0, 'no'],
       [1, 'yes'],
       [0, 'no']], dtype=object)
```

```
y_test
```

```
array([[0, 'no'],
       [0, 'no'],
       [0, 'no'],
       [1, 'yes'],
       [1, 'yes'],
       [0, 'no'],
       [0, 'no'],
       [0, 'no'],
       [1, 'yes'],
       [0, 'no'],
       [0, 'no'],
       [1, 'yes'],
       [1, 'yes'],
       [0, 'no'],
       [0, 'no'],
       [0, 'no'],
       [0, 'no'],
       [0, 'no'],
       [0, 'no'],
       [0, 'no'],
       [0, 'no'],
       [0, 'no'],
       [1, 'yes'],
       [0, 'no'],
       [0, 'no'],
       [0, 'no'],
       [0, 'no'],
       [1, 'yes'],
       [0, 'no'],
       [1, 'yes'],
       [1, 'yes'],
       [0, 'no'],
       [0, 'no'],
       [0, 'no'],
       [1, 'yes'],
       [0, 'no'],
       [0, 'no'],
       [0, 'no'],
       [0, 'no'],
       [1, 'yes'],
       [0, 'no'],
       [0, 'no'],
       [0, 'no'],
       [1, 'yes'],
       [1, 'yes'],
       [0, 'no'],
       [0, 'no'],
       [0, 'no'],
       [1, 'yes'],
       [0, 'no'],
       [1, 'yes'],
       [0, 'no'],
       [0, 'no'],
       [1, 'yes'],
       [1, 'yes'],
       [1, 'yes'],
       [0, 'no'],
       [1, 'yes'],
```