Assignment - 2

Name: PRABHATH SAI G B

Reg No: 20MID0137

Campus: VIT Vellore

Perform Below Tasks to complete the assignment:

1. Download the dataset: Dataset

In [20]:
```
#Importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Load the dataset.

In [2]:
```
#Loading the dataset
data=pd.read_csv('titanic.csv')
#Head of the data
data.head()
```

Out[2]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	de
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Na
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	Na
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	Na

In [4]:
```
#Shape
data.shape
```

Out[4]:
```
(891, 15)
```

In [5]:
```
#Information
data.info()
```

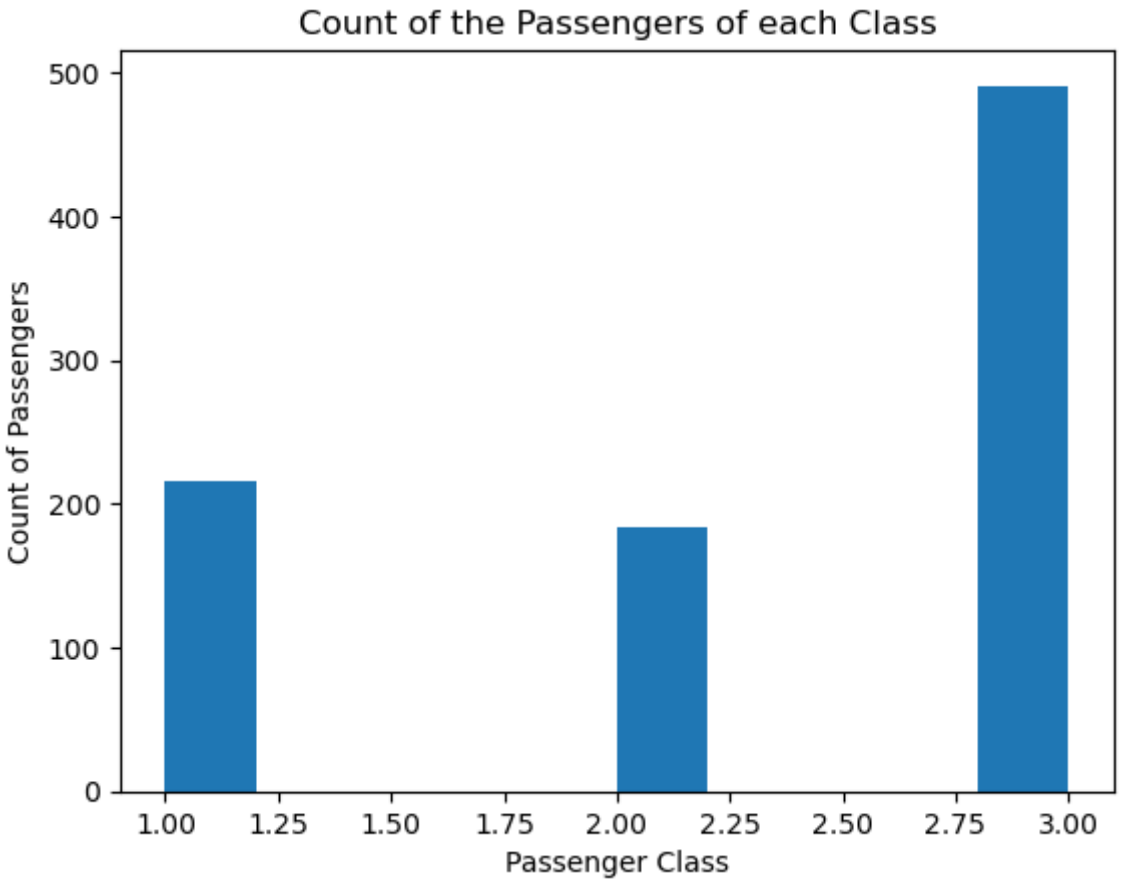
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null 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

3. Perform Below Visualizations.

• Univariate Analysis

In [10]:
```
#Histogram
plt.hist(data.pclass)
plt.xlabel('Passenger Class')
plt.ylabel('Count of Passengers')
plt.title('Count of the Passengers of each Class')
```

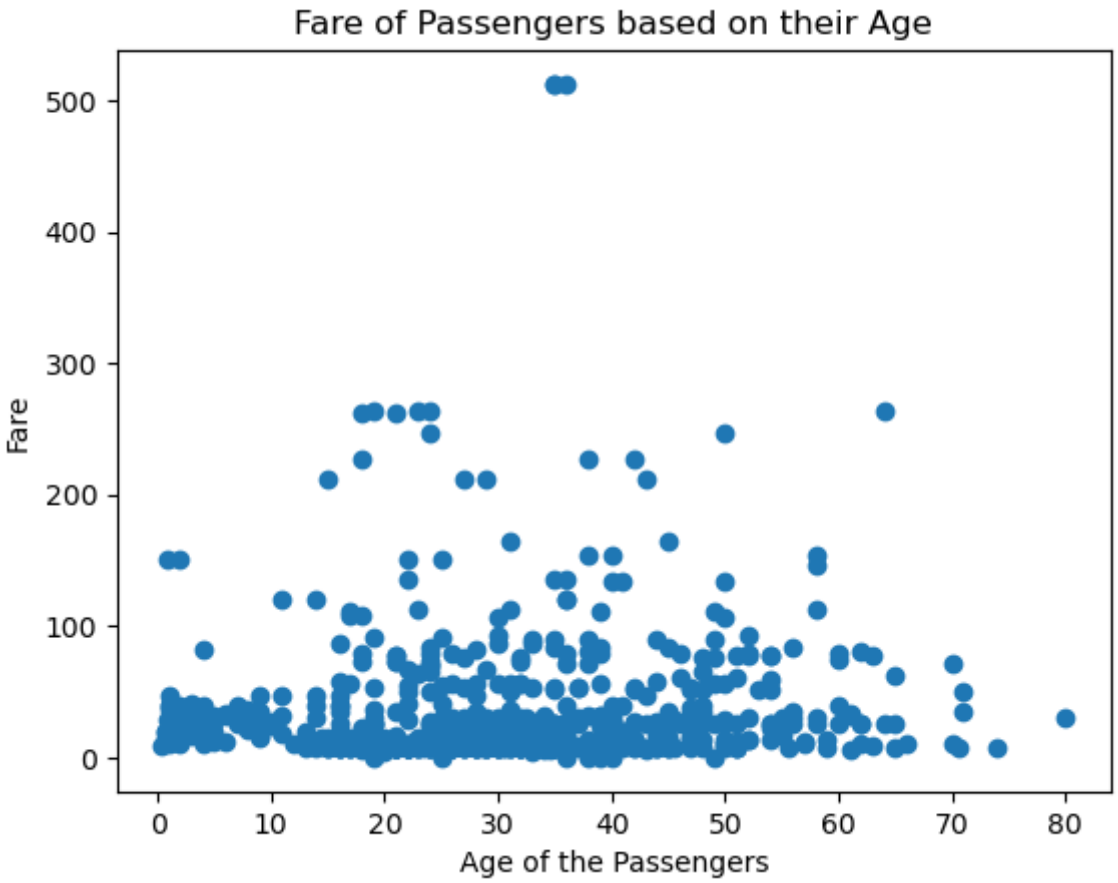
Out[10]:
```
Text(0.5, 1.0, 'Count of the Passengers of each Class')
```



• Bi - Variate Analysis

In [19]:
```
#Bar Plot
plt.scatter(data.age, data.fare)
plt.xlabel('Age of the Passengers')
plt.ylabel('Fare')
plt.title('Fare of Passengers based on their Age')
```

Out[19]:
```
Text(0.5, 1.0, 'Fare of Passengers based on their Age')
```



• Multi - Variate Analysis

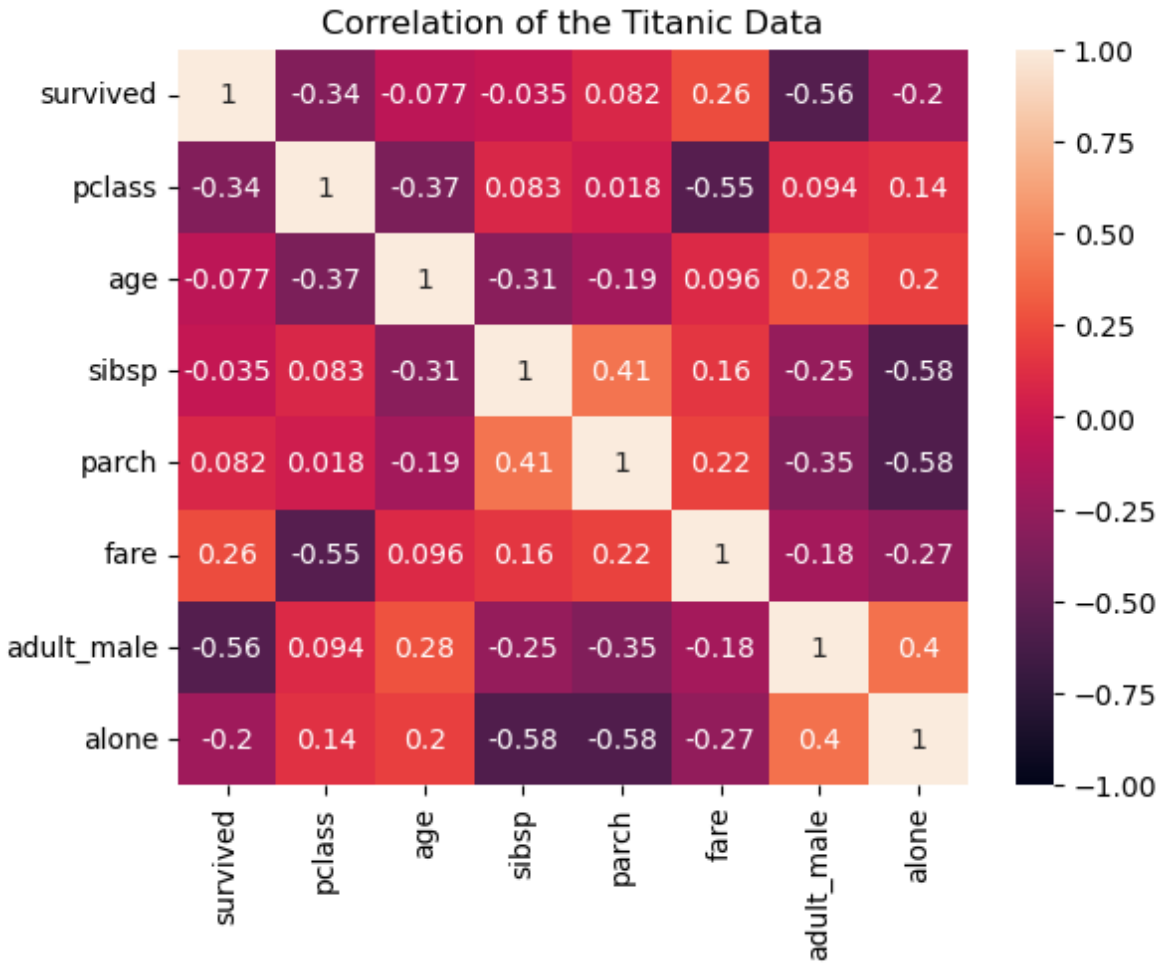
In [23]:
```
#Heat Map
data.corr()
```

Out[23]:

	survived	pclass	age	sibsp	parch	fare	adult_male	alone
survived	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307	-0.557080	-0.203367
pclass	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500	0.094035	0.135207
age	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067	0.280328	0.198270
sibsp	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651	-0.253586	-0.584471
parch	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225	-0.349943	-0.583398
fare	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000	-0.182024	-0.271832
adult_male	-0.557080	0.094035	0.280328	-0.253586	-0.349943	-0.182024	1.000000	0.404744
alone	-0.203367	0.135207	0.198270	-0.584471	-0.583398	-0.271832	0.404744	1.000000

In [28]:
```
hm=sns.heatmap(data.corr(), annot=True, vmin=-1, vmax=+1)
hm.set_title('Correlation of the Titanic Data')
```

Out[28]:
```
Text(0.5, 1.0, 'Correlation of the Titanic Data')
```



4. Perform descriptive statistics on the dataset.

In [35]:
```
#Description
data.describe()
```

Out[35]:

	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

5. Handle the Missing values.

In [43]:
```
#Checking for missing values
data.isnull().sum()
```

Out[43]:
```
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
```

In [44]:
```
#Handling the Null values by placing the mean and mode of the data
#Age Column
mean_age = data['age'].mean()
data['age'].fillna(mean_age, inplace=True)
```

In [45]:
```
#Embarked Column
mode_embarked = data['embarked'].mode()[0]
data['embarked'].fillna(mode_embarked, inplace=True)
```

In [46]:
```
#Deck Column
mode_deck = data['deck'].mode()[0]
data['deck'].fillna(mode_deck, inplace=True)
```

In [48]:
```
#Embark_town Column
mode_embark_town = data['embark_town'].mode()[0]
data['embark_town'].fillna(mode_embark_town, inplace=True)
```

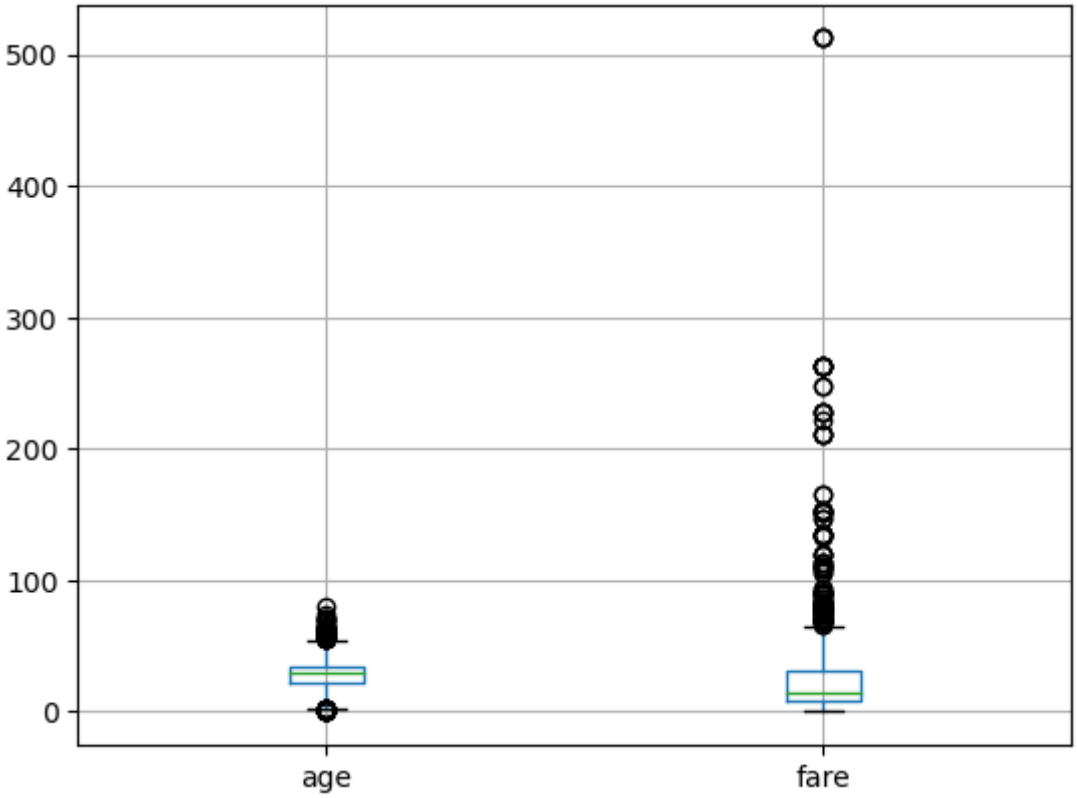
In [49]:
```
#Checking after handling null values
data.isnull().any()
```

Out[49]:
```
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 [53]:
```
#Checking outliers of the Numerical attributes using a Box Plot
num_col=['age', 'fare']
data.boxplot(num_col)
```

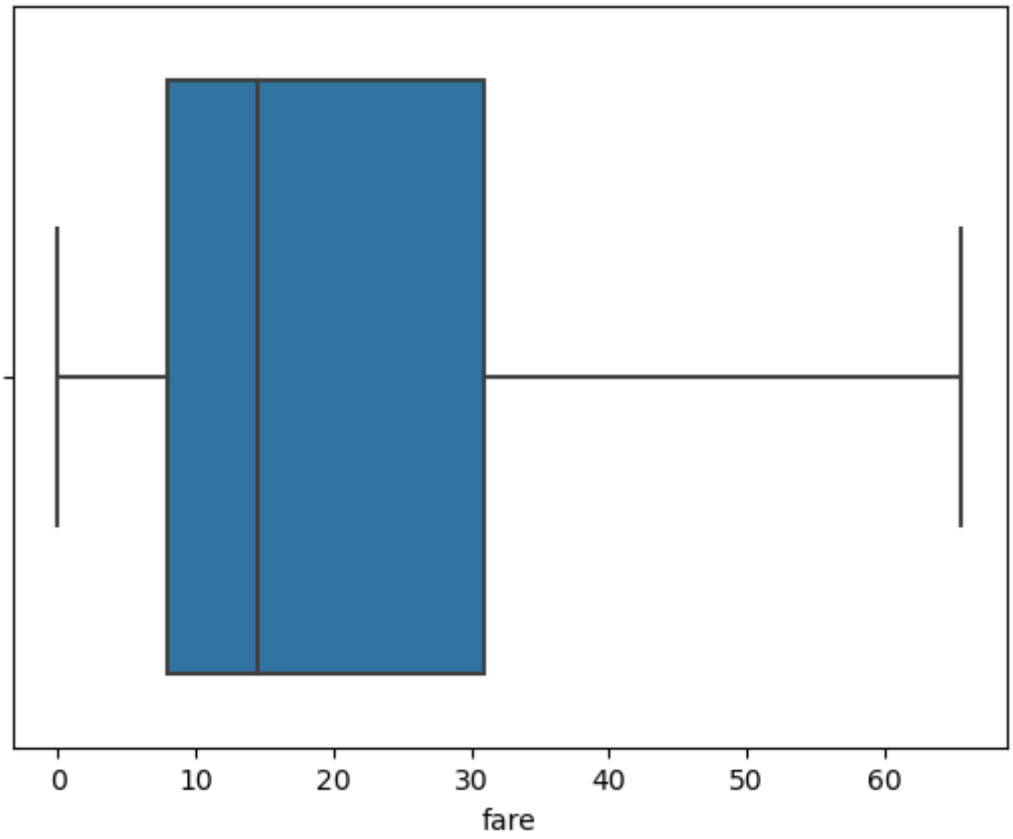
Out[53]:
```
<AxesSubplot:>
```



In [75]:
```
#Outliers identified in both the attributes
q1 = data['fare'].quantile(0.25)
q2 = data['fare'].quantile(0.75)
Inter_Quartile_Range = q2 - q1
whisker_width = 1.5
lower_whisker = q1 -(whisker_width*Inter_Quartile_Range)
upper_whisker = q2 + (whisker_width*Inter_Quartile_Range)
data['fare']=np.where(data['fare']>upper_whisker,upper_whisker,np.where(data['fare
```

In [77]:
```
#After removing outliers
sns.boxplot(data.fare)
```

Out[77]:
```
<AxesSubplot:xlabel='fare'>
```



7. Check for Categorical columns and perform encoding.

In [54]:
```
#Identify Categorical columns
categ_cols=data.select_dtypes(include=['object']).columns
print('Categorical Columns: ', categ_cols)
```

```
Categorical Columns:  Index(['sex', 'embarked', 'class', 'who', 'deck', 'embark_to
wn', 'alive'], dtype='object')
```

In [55]:
```
#Label Encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

for col in categ_cols:
    data[col]=le.fit_transform(data[col])
```

In [56]:
```
data.head()
```

Out[56]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	1	22.0	1	0	7.2500	2	2	1	True	2	
1	1	1	0	38.0	1	0	71.2833	0	0	2	False	2	
2	1	3	0	26.0	0	0	7.9250	2	2	2	False	2	
3	1	1	0	35.0	1	0	53.1000	2	0	2	False	2	
4	0	3	1	35.0	0	0	8.0500	2	2	1	True	2	

8. Split the data into dependent and independent variables.

In [57]:
```
#Survived Column is identified as the dependent variable
dep_var=data['survived']
indep_var=data.drop('survived', axis=1)
```

In [65]:
```
print('Dependent Variables: \n',dep_var.head(0))
```

```
Dependent Variables:
 Series([], Name: survived, dtype: int64)
```

In [64]:
```
print('Independent Variables: \n',indep_var.head(0))
```

```
Independent Variables:
 Empty DataFrame
Columns: [pclass, sex, age, sibsp, parch, fare, embarked, class, who, adult_male,
deck, embark_town, alive, alone]
Index: []
```

9. Scale the independent variables

In [66]:
```
from sklearn.preprocessing import StandardScaler
scale=StandardScaler()
indep_scaled=scale.fit_transform(indep_var)
```

In [68]:
```
indep_scaled_data=pd.DataFrame(indep_scaled, columns=indep_var.columns)
indep_scaled_data.head()
```

Out[68]:

	pclass	sex	age	sibsp	parch	fare	embarked	class	who
0	0.827377	0.737695	-0.592481	0.432793	-0.473674	-0.502445	0.585954	0.827377	-0.355242
1	-1.566107	-1.355574	0.638789	0.432793	-0.473674	0.786845	-1.942303	-1.566107	1.328379
2	0.827377	-1.355574	-0.284663	-0.474545	-0.473674	-0.488854	0.585954	0.827377	1.328379
3	-1.566107	-1.355574	0.407926	0.432793	-0.473674	0.420730	0.585954	-1.566107	1.328379
4	0.827377	0.737695	0.407926	-0.474545	-0.473674	-0.486337	0.585954	0.827377	-0.355242

10. Split the data into training and testing

In [69]:
```
#Seperating the Independent and Dependent variable
x=data.drop('survived', axis=1)
x.head()
```

Out[69]:

	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_tow
0	3	1	22.0	1	0	7.2500	2	2	1	True	2	
1	1	0	38.0	1	0	71.2833	0	0	2	False	2	
2	3	0	26.0	0	0	7.9250	2	2	2	False	2	
3	1	0	35.0	1	0	53.1000	2	0	2	False	2	
4	3	1	35.0	0	0	8.0500	2	2	1	True	2	

In [70]:
```
y=data['survived']
y.head()
```

Out[70]:
```
0    0
1    1
2    1
3    1
4    0
Name: survived, dtype: int64
```

In [72]:
```
#Splitting to Train and Test data
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_sta
```

In [73]:
```
print('Training Set: \n', x_train.shape, y_train.shape)
```

```
Training Set:
 (712, 14) (712,)
```

In [74]:
```
print('Test Set: \n', x_test.shape, y_test.shape)
```

```
Test Set:
 (179, 14) (179,)
```