ADS Assignment 2

Name:Damini N

2. Load the dataset.

In [4]:

```
import pandas as pd

# Specify the path to your CSV file
csv_file_path = ('C:/Users/Damini N/OneDrive/Desktop/ADS/titanic.csv')

# Load the dataset from the CSV file
dataset = pd.read_csv(csv_file_path)
```

In [5]:

```
# Load the dataset without a header row
dataset = pd.read_csv(csv_file_path, header=None)
```

In [6]:

```
dataset
```

Out[6]:

```
            0       1       2     3      4      5        6         7       8       9
  0  survived  pclass     sex   age  sibsp  parch     fare  embarked   class     who  adult_ma
  1         0       3    male  22.0      1      0     7.25         S   Third     man        Tr
  2         1       1  female  38.0      1      0  71.2833         C   First   woman       Fal
  3         1       3  female  26.0      0      0    7.925         S   Third   woman       Fal
  4         1       1  female  35.0      1      0     53.1         S   First   woman       Fal
...       ...     ...     ...   ...    ...    ...      ...       ...     ...     ...
887         0       2    male  27.0      0      0     13.0         S  Second     man        Tr
888         1       1  female  19.0      0      0     30.0         S   First   woman       Fal
889         0       3  female   NaN      1      2    23.45         S   Third   woman       Fal
890         1       1    male  26.0      0      0     30.0         C   First     man        Tr
891         0       3    male  32.0      0      0     7.75         Q   Third     man        Tr

892 rows × 15 columns
```

In [7]:

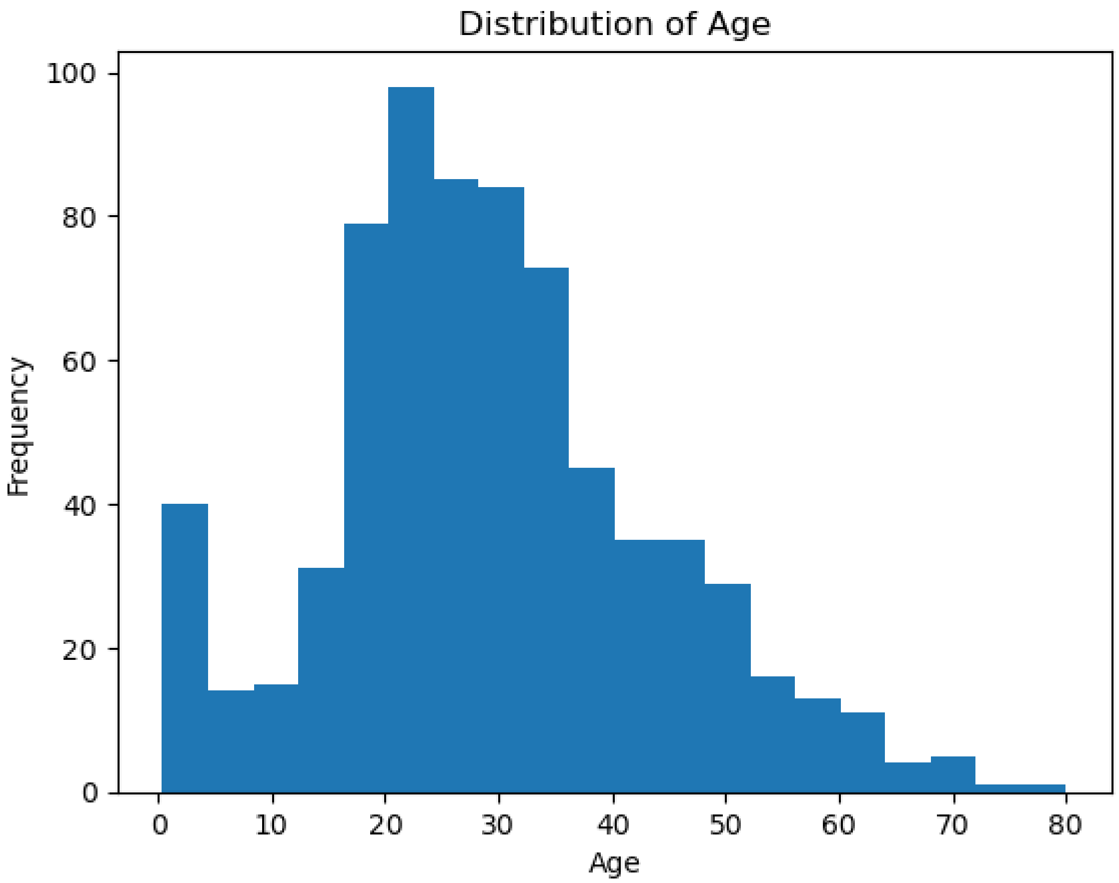
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the Titanic dataset
df = pd.read_csv('C:/Users/Damini N/OneDrive/Desktop/ADS/titanic.csv')

# Display the first few rows of the dataset
print(df.head())
```

```
   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   

     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 ● Bi - Variate Analysis ● Multi - Variate Analysis

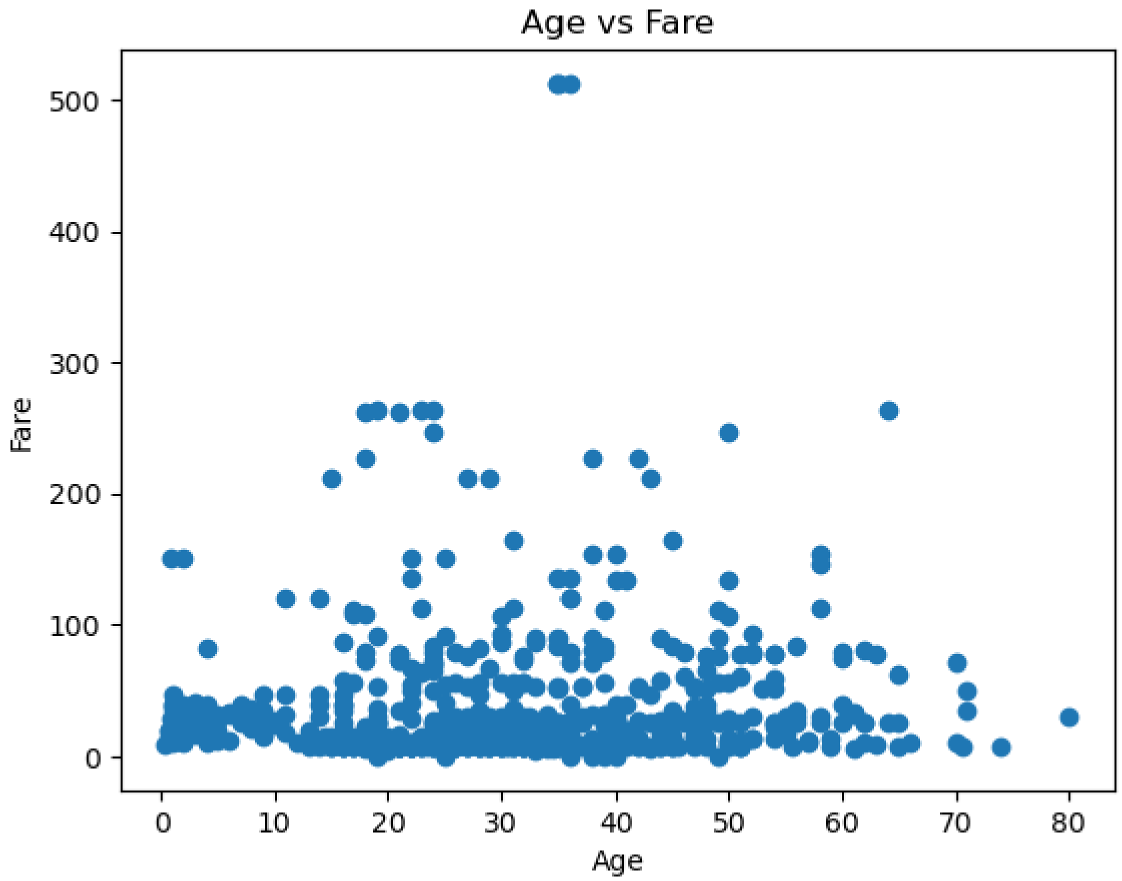
In [12]:

```
import matplotlib.pyplot as plt

# Plotting histogram for age
plt.hist(df['age'].dropna(), bins=20)
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Distribution of Age')
plt.show()
```



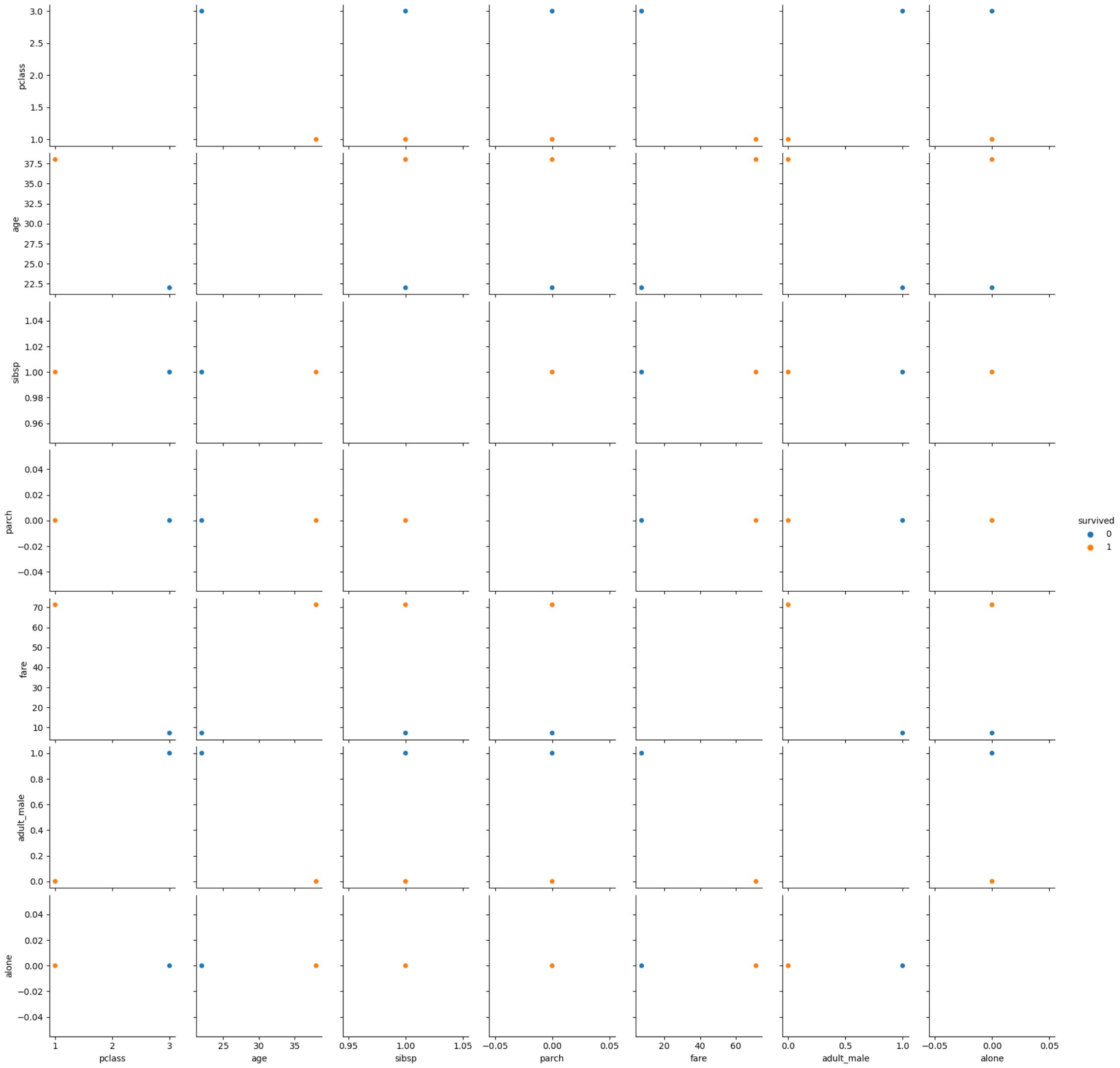
In [13]:

```
# Plotting scatter plot for age and fare
plt.scatter(df['age'], df['fare'])
plt.xlabel('Age')
plt.ylabel('Fare')
plt.title('Age vs Fare')
plt.show()
```



In [15]:

```
# Multivariate analysis using seaborn
sns.pairplot(df, hue='survived')
plt.show()
```



In [16]:

```python
import pandas as pd

# Create a DataFrame from the given dataset
data = {
    'survived': [0, 1],
    'pclass': [3, 1],
    'sex': ['male', 'female'],
    'age': [22, 38],
    'sibsp': [1, 1],
    'parch': [0, 0],
    'fare': [7.25, 71.2833],
    'embarked': ['S', 'C'],
    'class': ['Third', 'First'],
    'who': ['man', 'woman'],
    'adult_male': [True, False],
    'deck': ['', 'C'],
    'embark_town': ['Southampton', 'Cherbourg'],
    'alive': ['no', 'yes'],
    'alone': [False, False]
}

df = pd.DataFrame(data)

# Compute descriptive statistics
statistics = df.describe()

# Display the statistics
print(statistics)
```

```
       survived    pclass        age  sibsp  parch       fare
count  2.000000  2.000000   2.000000    2.0    2.0   2.000000
mean   0.500000  2.000000  30.000000    1.0    0.0  39.266650
std    0.707107  1.414214  11.313708    0.0    0.0  45.278381
min    0.000000  1.000000  22.000000    1.0    0.0   7.250000
25%    0.250000  1.500000  26.000000    1.0    0.0  23.258325
50%    0.500000  2.000000  30.000000    1.0    0.0  39.266650
75%    0.750000  2.500000  34.000000    1.0    0.0  55.274975
max    1.000000  3.000000  38.000000    1.0    0.0  71.283300
```

5. Handle the Missing values.

In [17]:

```python
df.dropna(inplace=True)
```

6. Find the outliers and replace the outliers

In [22]:

```
import pandas as pd
import numpy as np

# Create a DataFrame from the given dataset
data = {
    'survived': [0, 1],
    'pclass': [3, 1],
    'sex': ['male', 'female'],
    'age': [22, 38],
    'sibsp': [1, 1],
    'parch': [0, 0],
    'fare': [7.25, 71.2833],
    'embarked': ['S', 'C'],
    'class': ['Third', 'First'],
    'who': ['man', 'woman'],
    'adult_male': [True, False],
    'deck': ['', 'C'],
    'embark_town': ['Southampton', 'Cherbourg'],
    'alive': ['no', 'yes'],
    'alone': [False, False]
}

df = pd.DataFrame(data)

# Identify outliers using the IQR method
Q1 = df['fare'].quantile(0.25)
Q3 = df['fare'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = df[(df['fare'] < lower_bound) | (df['fare'] > upper_bound)]
print("Outliers:")
print(outliers)

# Replace outliers with a specified value
replacement_value = df['fare'].median()
df.loc[(df['fare'] < lower_bound) | (df['fare'] > upper_bound), 'fare'] = replacement_va

print("\nAfter replacing outliers:")
print(df)
```

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

After replacing outliers:
   survived  pclass     sex  age  sibsp  parch     fare embarked  class  \
0         0       3    male   22      1      0   7.2500        S  Third
1         1       1  female   38      1      0  71.2833        C  First

     who  adult_male deck  embark_town alive  alone
0    man        True       Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
```

In [23]:

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

# Create a DataFrame from the given dataset
data = {
    'survived': [0, 1],
    'pclass': [3, 1],
    'sex': ['male', 'female'],
    'age': [22, 38],
    'sibsp': [1, 1],
    'parch': [0, 0],
    'fare': [7.25, 71.2833],
    'embarked': ['S', 'C'],
    'class': ['Third', 'First'],
    'who': ['man', 'woman'],
    'adult_male': [True, False],
    'deck': ['', 'C'],
    'embark_town': ['Southampton', 'Cherbourg'],
    'alive': ['no', 'yes'],
    'alone': [False, False]
}

df = pd.DataFrame(data)

# Identify categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns.tolist()
print("Categorical columns:", categorical_cols)

# Perform one-hot encoding
encoder = OneHotEncoder(sparse=False, drop='first')
encoded_cols = pd.DataFrame(encoder.fit_transform(df[categorical_cols]))
encoded_cols.columns = encoder.get_feature_names(categorical_cols)

# Replace categorical columns with encoded columns
df.drop(columns=categorical_cols, inplace=True)
df = pd.concat([df, encoded_cols], axis=1)

# Display the encoded dataset
print("\nEncoded dataset:")
print(df)
```

```
Categorical columns: ['sex', 'embarked', 'class', 'who', 'deck', 'embark_t
own', 'alive']

Encoded dataset:
   survived  pclass  age  sibsp  parch     fare  adult_male  alone  sex_ma
le  \
0         0       3   22      1      0   7.2500        True  False        
1.0
1         1       1   38      1      0  71.2833       False  False        
0.0

   embarked_S  class_Third  who_woman  deck_C  embark_town_Southampton  \
0         1.0          1.0        0.0     0.0                      1.0   
1         0.0          0.0        1.0     1.0                      0.0   

   alive_yes  
0        0.0  
1        1.0  

C:\Users\Damini N\anaconda3\lib\site-packages\sklearn\utils\deprecation.p
y:87: FutureWarning: Function get_feature_names is deprecated; get_feature
_names is deprecated in 1.0 and will be removed in 1.2. Please use get_fea
ture_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

In [24]:

```python
import pandas as pd

# Create a DataFrame from the given dataset
data = {
    'survived': [0, 1],
    'pclass': [3, 1],
    'sex': ['male', 'female'],
    'age': [22, 38],
    'sibsp': [1, 1],
    'parch': [0, 0],
    'fare': [7.25, 71.2833],
    'embarked': ['S', 'C'],
    'class': ['Third', 'First'],
    'who': ['man', 'woman'],
    'adult_male': [True, False],
    'deck': ['', 'C'],
    'embark_town': ['Southampton', 'Cherbourg'],
    'alive': ['no', 'yes'],
    'alone': [False, False]
}

df = pd.DataFrame(data)

# Split into features (independent variables) and target (dependent variable)
X = df.drop('survived', axis=1)
y = df['survived']

# Display the independent variables
print("Independent Variables (X):")
print(X)

# Display the dependent variable
print("\nDependent Variable (y):")
print(y)
```

```
Independent Variables (X):
   pclass     sex  age  sibsp  parch     fare embarked  class    who  \
0       3    male   22      1      0   7.2500        S  Third    man   
1       1  female   38      1      0  71.2833        C  First  woman   

   adult_male deck  embark_town alive  alone  
0        True       Southampton    no  False  
1       False    C    Cherbourg   yes  False  

Dependent Variable (y):
0    0
1    1
Name: survived, dtype: int64
```

In [25]:

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Create a DataFrame from the given dataset
data = {
    'survived': [0, 1],
    'pclass': [3, 1],
    'sex': ['male', 'female'],
    'age': [22, 38],
    'sibsp': [1, 1],
    'parch': [0, 0],
    'fare': [7.25, 71.2833],
    'embarked': ['S', 'C'],
    'class': ['Third', 'First'],
    'who': ['man', 'woman'],
    'adult_male': [True, False],
    'deck': ['', 'C'],
    'embark_town': ['Southampton', 'Cherbourg'],
    'alive': ['no', 'yes'],
    'alone': [False, False]
}

df = pd.DataFrame(data)

# Select only the numeric columns to scale
numeric_cols = df.select_dtypes(include='number').columns.tolist()

# Scale the numeric columns using Min-Max scaling
scaler = MinMaxScaler()
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])

# Display the scaled dataset
print(df)
```

```
   survived  pclass     sex  age  sibsp  parch  fare embarked  class    wh
o  \
0       0.0     1.0    male  0.0    0.0    0.0   0.0        S  Third    ma
n
1       1.0     0.0  female  1.0    0.0    0.0   1.0        C  First  woma
n

   adult_male deck  embark_town alive  alone
0        True       Southampton    no  False
1       False    C    Cherbourg   yes  False
```

10. Split the data into training and testing

In [26]:

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Create a DataFrame from the given dataset
data = {
    'survived': [0, 1],
    'pclass': [3, 1],
    'sex': ['male', 'female'],
    'age': [22, 38],
    'sibsp': [1, 1],
    'parch': [0, 0],
    'fare': [7.25, 71.2833],
    'embarked': ['S', 'C'],
    'class': ['Third', 'First'],
    'who': ['man', 'woman'],
    'adult_male': [True, False],
    'deck': ['', 'C'],
    'embark_town': ['Southampton', 'Cherbourg'],
    'alive': ['no', 'yes'],
    'alone': [False, False]
}

df = pd.DataFrame(data)

# Split the dataset into features (X) and target (y)
X = df.drop('survived', axis=1)
y = df['survived']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42

# Display the shapes of the resulting sets
print("Training set - X shape:", X_train.shape)
print("Training set - y shape:", y_train.shape)
print("Testing set - X shape:", X_test.shape)
print("Testing set - y shape:", y_test.shape)
```

```
Training set - X shape: (1, 14)
Training set - y shape: (1,)
Testing set - X shape: (1, 14)
Testing set - y shape: (1,)
```

In [ ]: