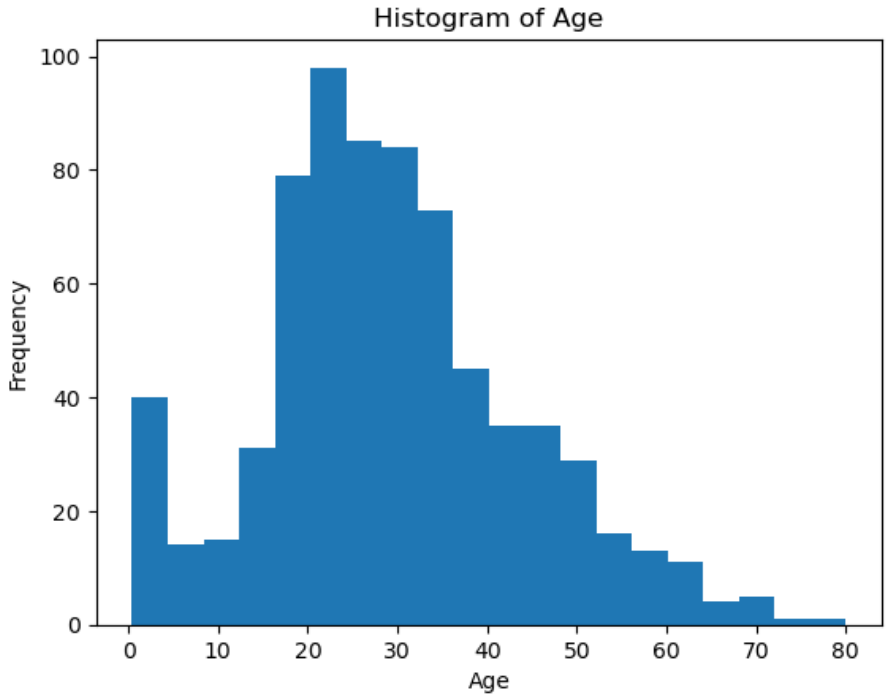
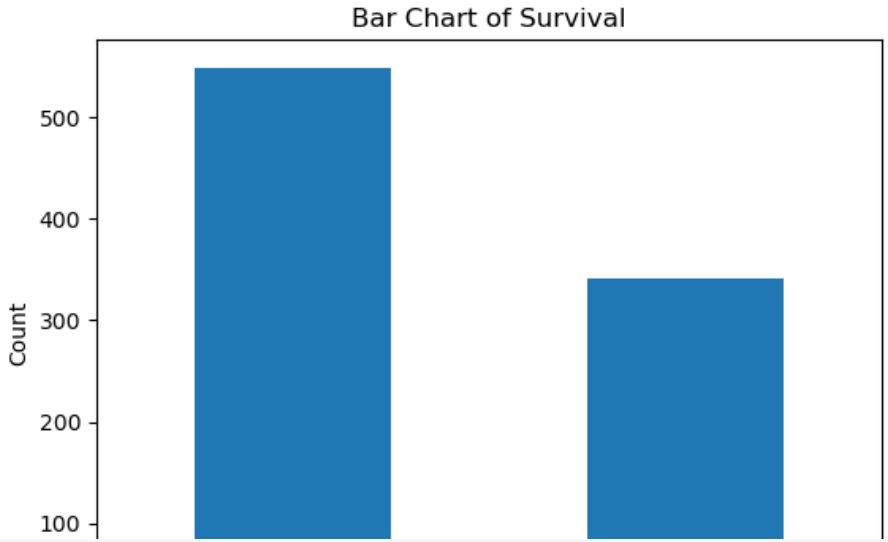
```
#load the dataset
#VEERANKI YASASWINI
#20BCI7149
import pandas as pd
data = "C:/Users/20bci7149/Downloads/titanic.csv"
df = pd.read_csv(data)
print(df)
```

```
     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]
```

```
#univariate analysis

import matplotlib.pyplot as plt



# Histogram of a numerical variable
df['age'].plot(kind='hist', bins=20)
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Histogram of Age')
```

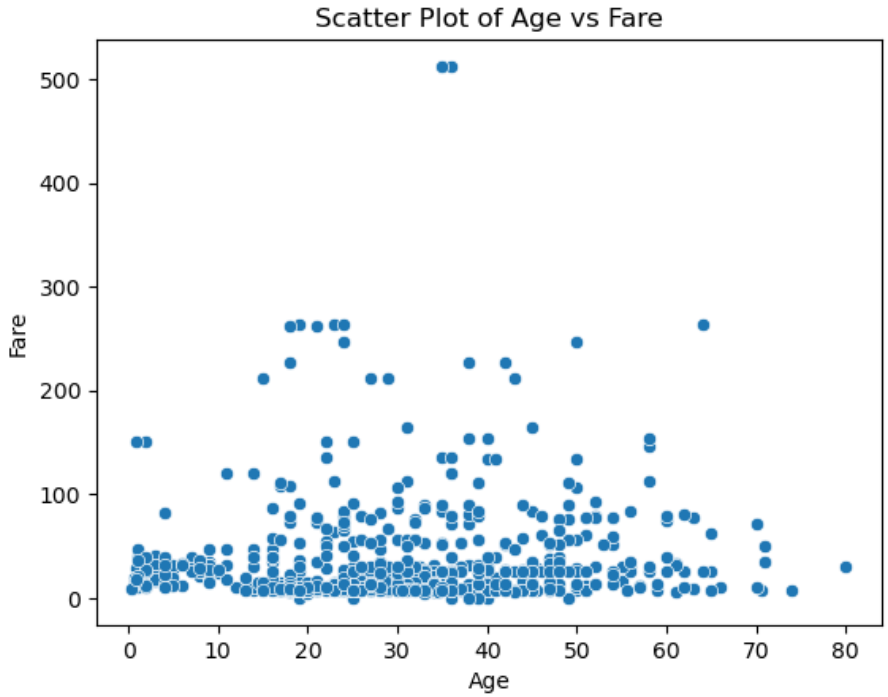
```
plt.show()

# Bar chart of a categorical variable
df['survived'].value_counts().plot(kind='bar')
plt.xlabel('Survived')
plt.ylabel('Count')
plt.title('Bar Chart of Survival')
plt.show()

# Bar chart of another categorical variable
df['sex'].value_counts().plot(kind='bar')
plt.xlabel('Sex')
plt.ylabel('Count')
plt.title('Bar Chart of Sex')
plt.show()
```

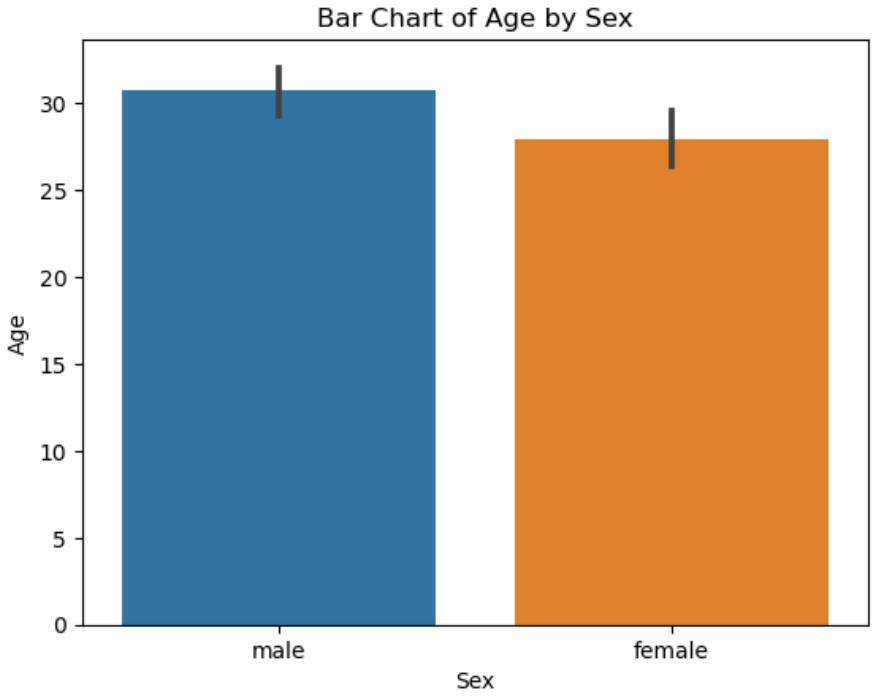




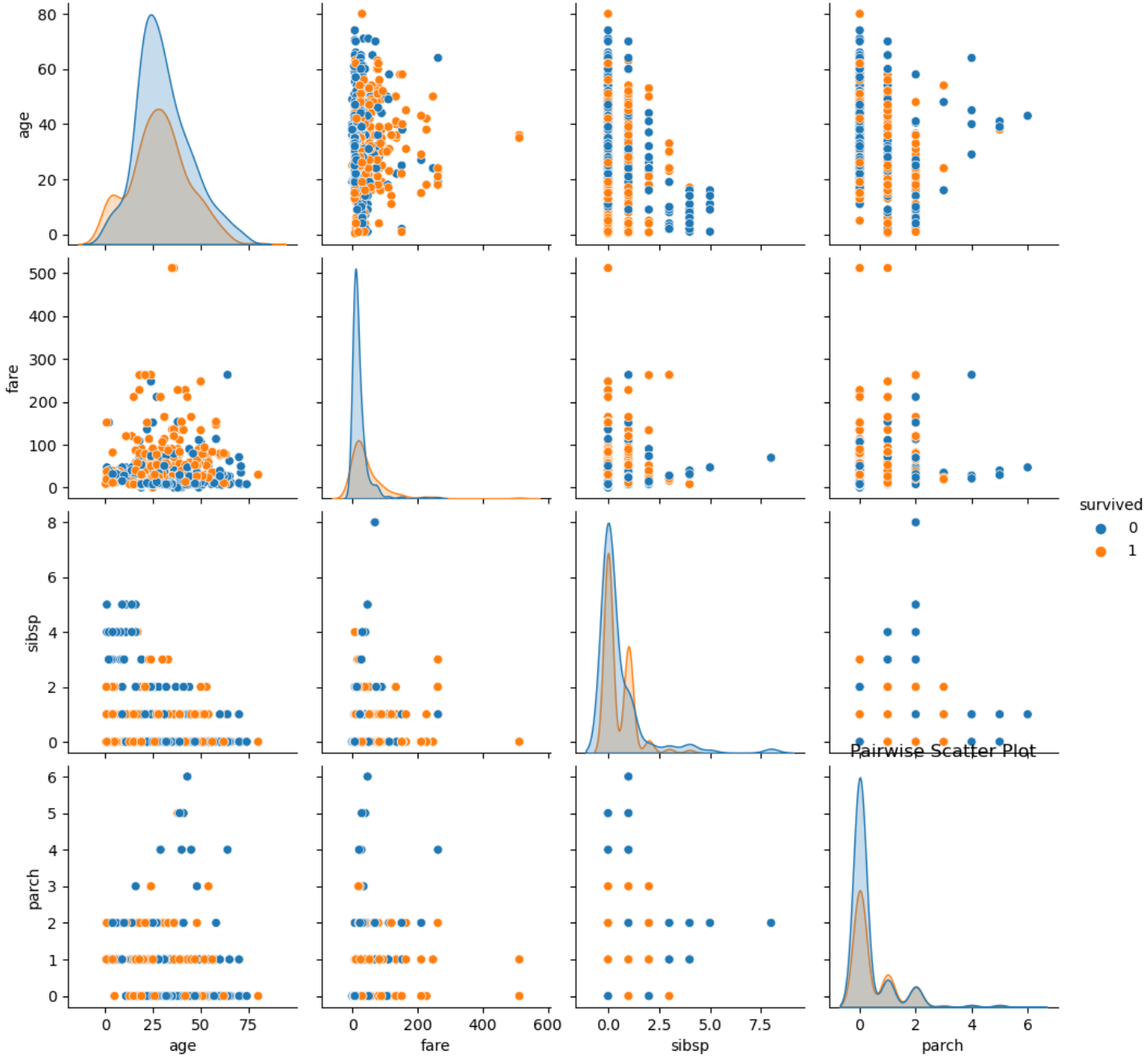

```
#bi-variate analysis

import seaborn as sns
sns.scatterplot(data=df, x='age', y='fare')
```

```
plt.xlabel('Age')
plt.ylabel('Fare')
plt.title('Scatter Plot of Age vs Fare')
plt.show()

# Bar chart of a categorical variable with respect to a numerical variable
sns.barplot(data=df, x='sex', y='age')
plt.xlabel('Sex')
plt.ylabel('Age')
plt.title('Bar Chart of Age by Sex')
plt.show()

# Heatmap of correlation between numerical variables
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

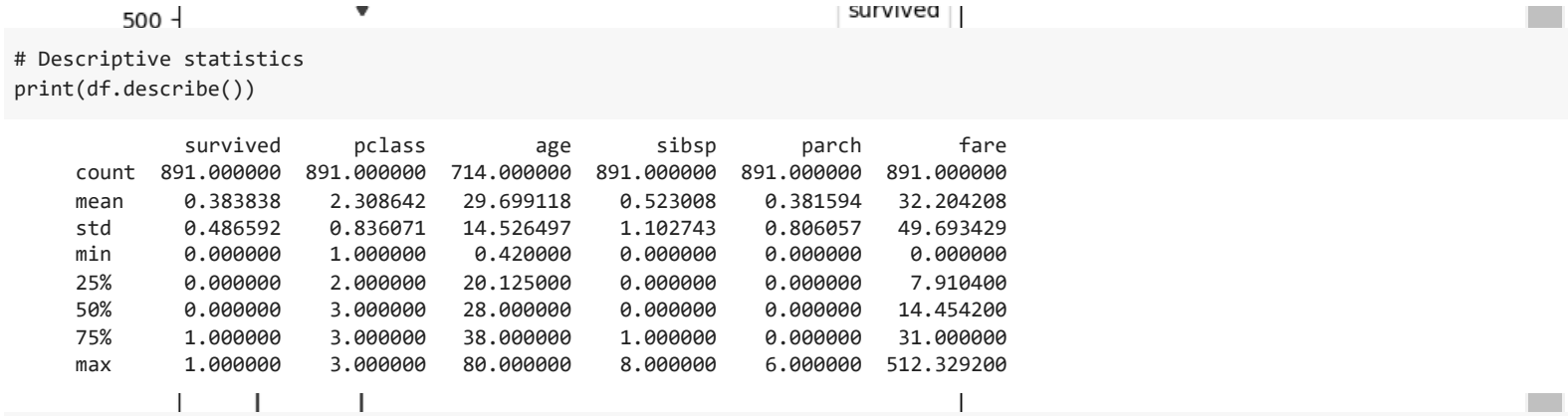

Scatter Plot of Age vs Fare
Fare
Age
0
100
200
300
400
500
0
10
20
30
40
50
60
70
80


Bar Chart of Age by Sex
Age
Sex
0
5
10
15
20
25
30
male
female

```
# multivariate-analysis

# Pairwise scatter plot of multiple numerical variables
sns.pairplot(data=df, vars=['age', 'fare', 'sibsp', 'parch'], hue='survived')
plt.title('Pairwise Scatter Plot')
plt.show()

# Box plot of a numerical variable across different categories
sns.boxplot(data=df, x='pclass', y='fare', hue='survived')
plt.xlabel('Passenger Class')
plt.ylabel('Fare')
plt.title('Box Plot of Fare by Passenger Class and Survival')
plt.show()

# Heatmap of correlation between multiple numerical variables
corr_matrix = df[['age', 'fare', 'sibsp', 'parch']].corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



Box Plot of Fare by Passenger Class and Survival




```
# Descriptive statistics
print(df.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
```

```
#check for missing values

print(df.isnull().sum())
```

```
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
```

```
# Handle missing values
# Fill missing values with mean/median/mode
df['age'].fillna(df['age'].mean(), inplace=True)
df['embarked'].fillna(df['embarked'].mode(), inplace=True)
df['deck'].fillna(df['deck'].mode()[0], inplace=True)
df['embark_town'].fillna(df['embark_town'].mode()[0], inplace=True)

# Check again for missing values after handling
print(df.isnull().sum())
```

```
survived       0
pclass         0
```

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

```
# Drop rows with missing values
df.dropna(inplace=True)
print(df.isnull().sum())
```

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

```
# Find and replace outliers using z-score method
import numpy as np

z_score_threshold = 3
z_scores = np.abs((df - df.mean()) / df.std())

outliers = (z_scores > z_score_threshold)

# Replace outliers with NaN or a specific value
df[outliers] = np.nan
```

```
# Check the replaced outliers in the DataFrame
print(df[outliers])
```

```
     survived  pclass  sex  age  sibsp  parch  fare embarked class  who  \
0         NaN     NaN  NaN  NaN    NaN    NaN   NaN      NaN   NaN  NaN
1         NaN     NaN  NaN  NaN    NaN    NaN   NaN      NaN   NaN  NaN
2         NaN     NaN  NaN  NaN    NaN    NaN   NaN      NaN   NaN  NaN
3         NaN     NaN  NaN  NaN    NaN    NaN   NaN      NaN   NaN  NaN
4         NaN     NaN  NaN  NaN    NaN    NaN   NaN      NaN   NaN  NaN
..        ...     ...  ...  ...    ...    ...   ...      ...   ...  ...
886       NaN     NaN  NaN  NaN    NaN    NaN   NaN      NaN   NaN  NaN
887       NaN     NaN  NaN  NaN    NaN    NaN   NaN      NaN   NaN  NaN
888       NaN     NaN  NaN  NaN    NaN    NaN   NaN      NaN   NaN  NaN
889       NaN     NaN  NaN  NaN    NaN    NaN   NaN      NaN   NaN  NaN
890       NaN     NaN  NaN  NaN    NaN    NaN   NaN      NaN   NaN  NaN

    adult_male deck embark_town alive alone
0          NaN  NaN         NaN   NaN   NaN
1          NaN  NaN         NaN   NaN   NaN
2          NaN  NaN         NaN   NaN   NaN
3          NaN  NaN         NaN   NaN   NaN
4          NaN  NaN         NaN   NaN   NaN
..         ...  ...         ...   ...   ...
886        NaN  NaN         NaN   NaN   NaN
887        NaN  NaN         NaN   NaN   NaN
888        NaN  NaN         NaN   NaN   NaN
889        NaN  NaN         NaN   NaN   NaN
890        NaN  NaN         NaN   NaN   NaN

[891 rows x 15 columns]
C:\Users\96036\AppData\Local\Temp\ipykernel_11320\2072163868.py:5: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future ver
  z_scores = np.abs((df - df.mean()) / df.std())
C:\Users\96036\AppData\Local\Temp\ipykernel_11320\2072163868.py:5: FutureWarning: The default value of numeric_only in DataFrame.std is deprecated. In a future vers
  z_scores = np.abs((df - df.mean()) / df.std())
```

```
# Perform encoding for each categorical column


from sklearn.preprocessing import LabelEncoder
# Identify categorical columns
categorical_columns = df.select_dtypes(include=['object']).columns

# Perform encoding for each categorical column
for column in categorical_columns:
    # Create an instance of LabelEncoder
    label_encoder = LabelEncoder()

    # Fit and transform the column
```

```
    df[column] = label_encoder.fit_transform(df[column])

# Print the encoded DataFrame
print(df)
```

```
     survived  pclass  sex        age  sibsp  parch     fare  embarked  class  \
0           0       3    1  22.000000    1.0    0.0   7.2500         2      2
1           1       1    0  38.000000    1.0    0.0  71.2833         0      0
2           1       3    0  26.000000    0.0    0.0   7.9250         2      2
3           1       1    0  35.000000    1.0    0.0  53.1000         2      0
4           0       3    1  35.000000    0.0    0.0   8.0500         2      2
..        ...     ...  ...        ...    ...    ...      ...       ...    ...
886         0       2    1  27.000000    0.0    0.0  13.0000         2      1
887         1       1    0  19.000000    0.0    0.0  30.0000         2      0
888         0       3    0  29.699118    1.0    2.0  23.4500         2      2
889         1       1    1  26.000000    0.0    0.0  30.0000         0      0
890         0       3    1  32.000000    0.0    0.0   7.7500         1      2

     who  adult_male  deck  embark_town  alive  alone
0      1        True     2            2      0  False
1      2       False     2            0      1  False
2      2       False     2            2      1   True
3      2       False     2            2      1  False
4      1        True     2            2      0   True
..   ...         ...   ...          ...    ...    ...
886    1        True     2            2      0   True
887    2       False     1            2      1   True
888    2       False     2            2      0  False
889    1        True     2            0      1   True
890    1        True     2            1      0   True

[889 rows x 15 columns]
```

```
#Split the data into dependent and independent variables.

X = df.drop("alive", axis=1)  # Independent variables (features)
y = df["alive"]  # Dependent variable

# Print the independent variables
print(X.head())

# Print the dependent variable
print(y.head())
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0    1.0    0.0   7.2500        S  Third
1         1       1  female  38.0    1.0    0.0  71.2833        C  First
2         1       3  female  26.0    0.0    0.0   7.9250        S  Third
3         1       1  female  35.0    1.0    0.0  53.1000        S  First
4         0       3    male  35.0    0.0    0.0   8.0500        S  Third
```

```
     who  adult_male deck  embark_town  alone
0    man        True    C  Southampton  False
1  woman       False    C    Cherbourg  False
2  woman       False    C  Southampton   True
3  woman       False    C  Southampton  False
4    man        True    C  Southampton   True
0     no
1    yes
2    yes
3    yes
4     no
Name: alive, dtype: object
```

```
#Scale the independent variables

from sklearn.preprocessing import MinMaxScaler

# Create an instance of MinMaxScaler
scaler = MinMaxScaler()

# Fit the scaler on the data
scaler.fit(X)

# Scale the independent variables
X_scaled = scaler.transform(X)
print(X)
```

```
     survived  pclass  sex        age  sibsp  parch     fare  embarked  class  \
0           0       3    1  22.000000    1.0    0.0   7.2500         2      2
1           1       1    0  38.000000    1.0    0.0  71.2833         0      0
2           1       3    0  26.000000    0.0    0.0   7.9250         2      2
3           1       1    0  35.000000    1.0    0.0  53.1000         2      0
4           0       3    1  35.000000    0.0    0.0   8.0500         2      2
..        ...     ...  ...        ...    ...    ...      ...       ...    ...
886         0       2    1  27.000000    0.0    0.0  13.0000         2      1
887         1       1    0  19.000000    0.0    0.0  30.0000         2      0
888         0       3    0  29.699118    1.0    2.0  23.4500         2      2
889         1       1    1  26.000000    0.0    0.0  30.0000         0      0
890         0       3    1  32.000000    0.0    0.0   7.7500         1      2

     who  adult_male  deck  embark_town  alone
0      1        True     2            2  False
1      2       False     2            0  False
2      2       False     2            2   True
3      2       False     2            2  False
4      1        True     2            2   True
..   ...         ...   ...          ...    ...
886    1        True     2            2   True
887    2       False     1            2   True
```

```
888    2       False     2           2  False
889    1        True     2           0   True
890    1        True     2           1   True

[889 rows x 14 columns]
```

```
# Split the data into training and testing sets

from sklearn.model_selection import train_test_split

# Assuming X is your independent variable matrix and y is your dependent variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("Training Data:")
print("X_train:", X_train)
print("y_train:", y_train)
print()

print("Testing Data:")
print("X_test:", X_test)
print("y_test:", y_test)
```

```
Training Data:
X_train:      survived  pclass     sex        age  sibsp  parch      fare embarked  \
331         0       1    male  45.500000    0.0    0.0   28.5000        S
733         0       2    male  23.000000    0.0    0.0   13.0000        S
382         0       3    male  32.000000    0.0    0.0    7.9250        S
704         0       3    male  26.000000    1.0    0.0    7.8542        S
813         0       3  female   6.000000    NaN    2.0   31.2750        S
..        ...     ...     ...        ...    ...    ...       ...      ...
106         1       3  female  21.000000    0.0    0.0    7.6500        S
270         0       1    male  29.699118    0.0    0.0   31.0000        S
860         0       3    male  41.000000    2.0    0.0   14.1083        S
435         1       1  female  14.000000    1.0    2.0  120.0000        S
102         0       1    male  21.000000    0.0    1.0   77.2875        S

      class    who  adult_male deck  embark_town  alone
331   First    man        True    C  Southampton   True
733  Second    man        True    C  Southampton   True
382   Third    man        True    C  Southampton   True
704   Third    man        True    C  Southampton  False
813   Third  child       False    C  Southampton  False
..      ...    ...         ...  ...          ...    ...
106   Third  woman       False    C  Southampton   True
270   First    man        True    C  Southampton   True
860   Third    man        True    C  Southampton  False
435   First  child       False    B  Southampton  False
102   First    man        True    D  Southampton  False
```