Celebal Assignment – Week 4

Missing Values Data:

Visual charts that highlight which dataset columns contain missing entries. These help in easily spotting where data cleaning or filling is required.

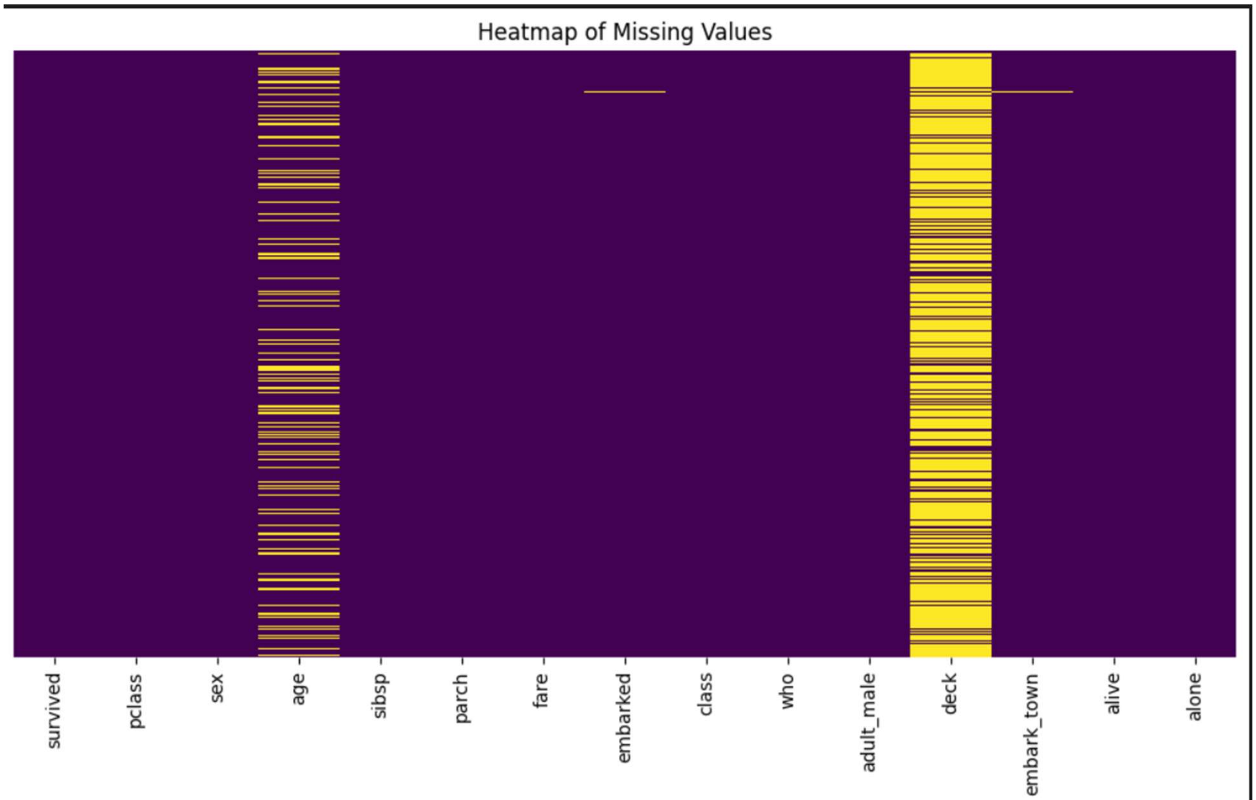
```
# Calculate the number of missing (null) values in each column
missing_values = dataset.isnull().sum()

# Print the count of missing values for each column
print(missing_values)
```

```
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
```

Heatmap of the missing values:

A visual representation using a heatmap to show where data is missing in the dataset. It provides a quick overview of the pattern and concentration of null values across rows and columns.

```
# Set the size of the plot
plt.figure(figsize=(12, 6))

# Create a heatmap to visualize missing values in the dataset
sns.heatmap(dataset.isnull(),       # True for missing values, False otherwise
            cbar=False,             # Disable the color bar for cleaner output
            cmap='viridis',         # Use 'viridis' color palette
            yticklabels=False)      # Hide row indices for simplicity

# Add a title to the plot
plt.title("Heatmap of Missing Values")

# Display the plot
plt.show()
```

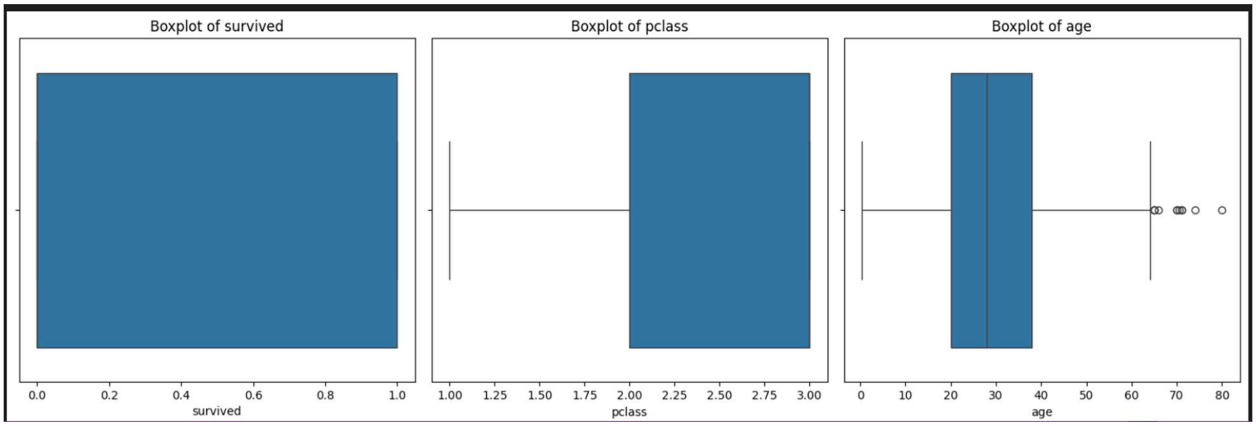


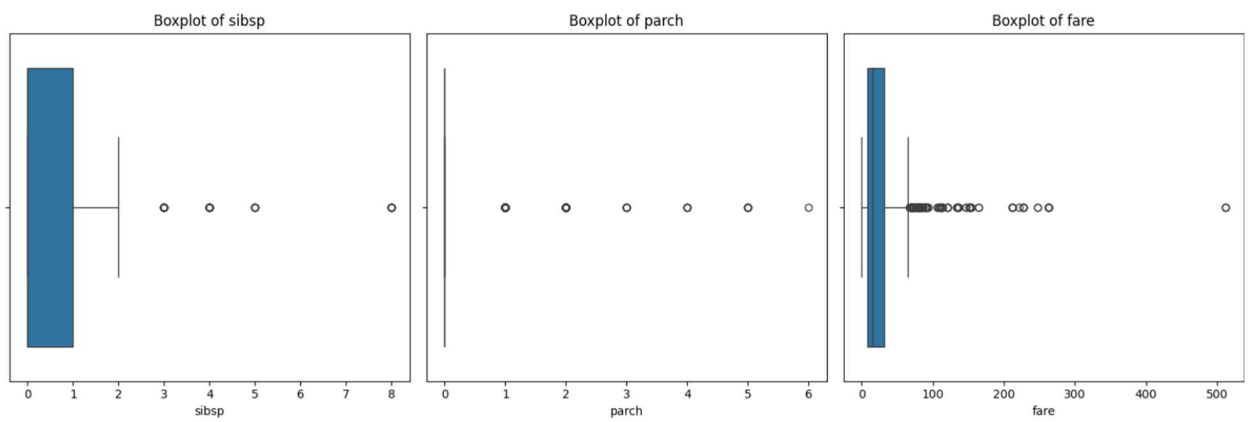
Outliers detection and plotting:

A method to find values that are far from the usual range, like extremely high fares. Visual tools like boxplots help identify and handle such anomalies.

```
# Set the overall figure size for all boxplots
plt.figure(figsize=(15, 10))

# Loop through each numerical column to create a boxplot
for i, col in enumerate(numerical_cols):
    # Create a subplot for each column (2 rows x 3 columns layout)
    plt.subplot(2, 3, i + 1)

    # Draw the boxplot for the current column
    sns.boxplot(data=dataset, x=col)

    # Set the title for the subplot
    plt.title(f'Boxplot of {col}')

# Adjust spacing between subplots to prevent overlap
plt.tight_layout()

# Display the plots
plt.show()
```

```
🔍 Column: survived
Outlier Count: 0
Lower Bound: -1.50
Upper Bound: 2.50
Outlier Values: []

🔍 Column: pclass
Outlier Count: 0
Lower Bound: 0.50
Upper Bound: 4.50
Outlier Values: []

🔍 Column: age
Outlier Count: 11
Lower Bound: -6.69
Upper Bound: 64.81
Outlier Values: [66.  65.  71.  70.5 65.  65.  71.  80.  70.  70. ]

🔍 Column: sibsp
Outlier Count: 46
Lower Bound: -1.50
Upper Bound: 2.50
Outlier Values: [3 4 3 3 4 5 3 4 5 3]
...
Lower Bound: -26.72
Upper Bound: 65.63
Outlier Values: [ 71.2833 263.     146.5208  82.1708  76.7292  80.      83.475   73.5
 263.      77.2875]
```

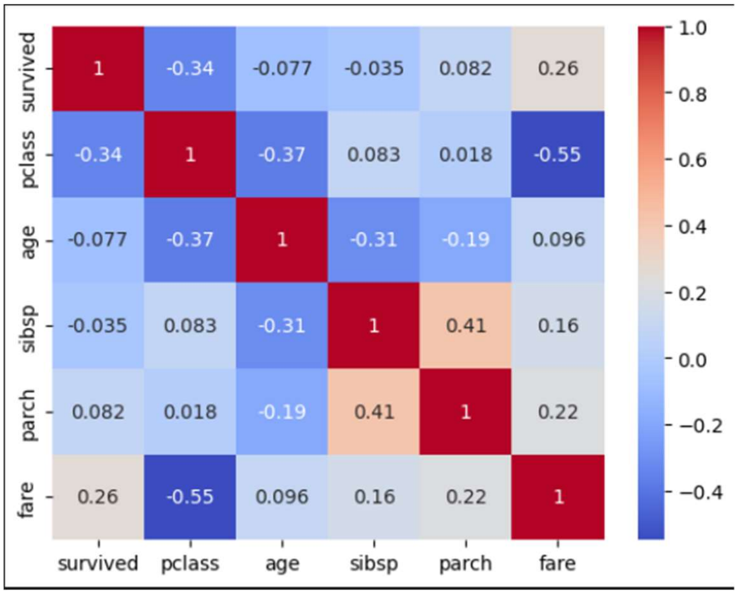




Correlation heatmap with numerical values:

A color-based chart that shows how strongly numerical columns are related to each other. Useful for understanding variable relationships and avoiding redundancy.

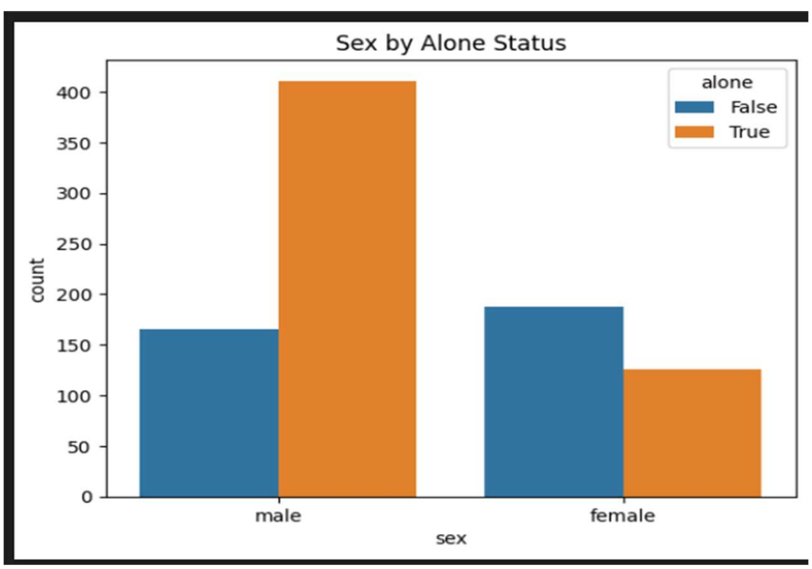
```
# Select only numeric columns and compute the correlation matrix
corr = dataset.select_dtypes(include=['int64', 'float64']).corr()

# Create a heatmap to visualize the correlation matrix
sns.heatmap(corr,
            annot=True,        # Display the correlation coefficients on the heatmap
            cmap='coolwarm')   # Use the 'coolwarm' color scheme for contrast

# Show the plot
plt.show()
```

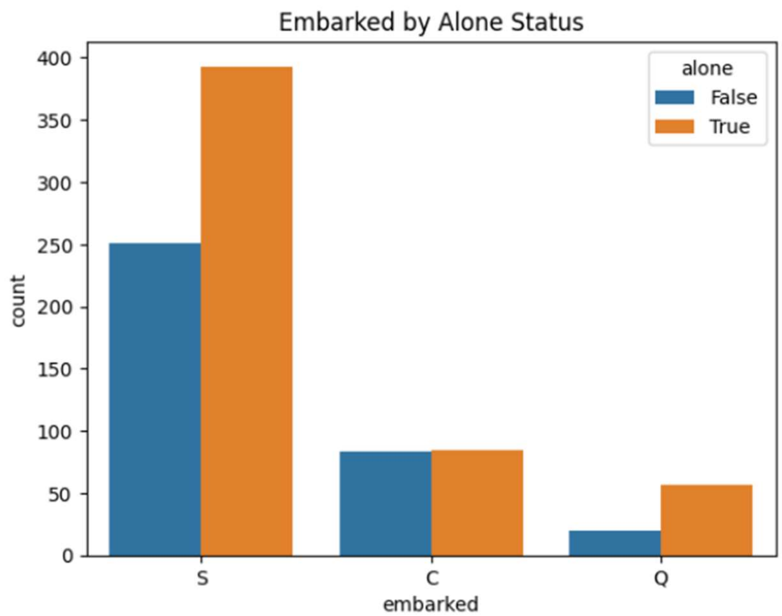


<u>Relationship between variables:</u>

Analyzes how different columns, especially with the target variable (Survived), are connected. Visual tools uncover trends and dependencies between variables.

```
# Countplot showing distribution of 'sex' grouped by whether the person was alone
sns.countplot(x='sex', hue='alone', data=dataset)
plt.title("Sex by Alone Status")  # Set the plot title
plt.show()

# Countplot showing distribution of 'embarked' grouped by whether the person was alone
sns.countplot(x='embarked', hue='alone', data=dataset)
plt.title("Embarked by Alone Status")  # Set the plot title
plt.show()
```

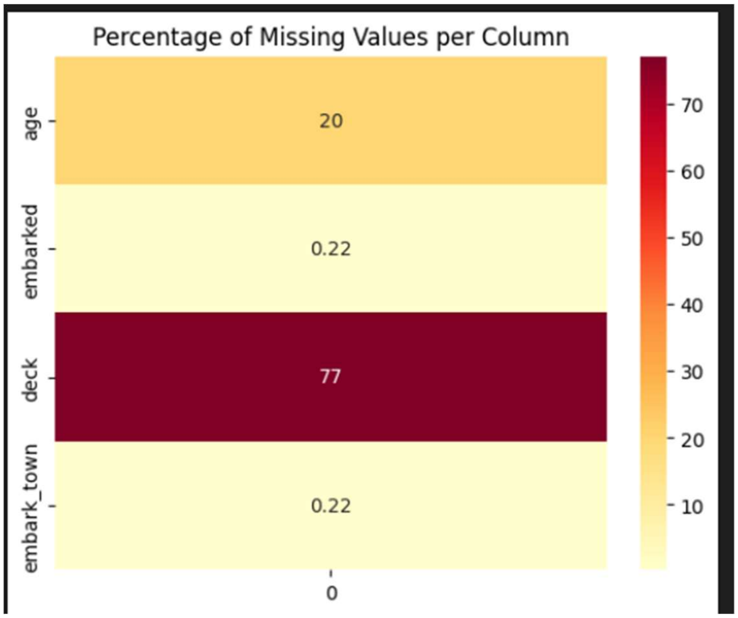




Calculating the percentage of the missing values in each columns:

Finds and displays how much data is missing in each column as a percentage. Helps determine which features need cleaning or can be dropped.

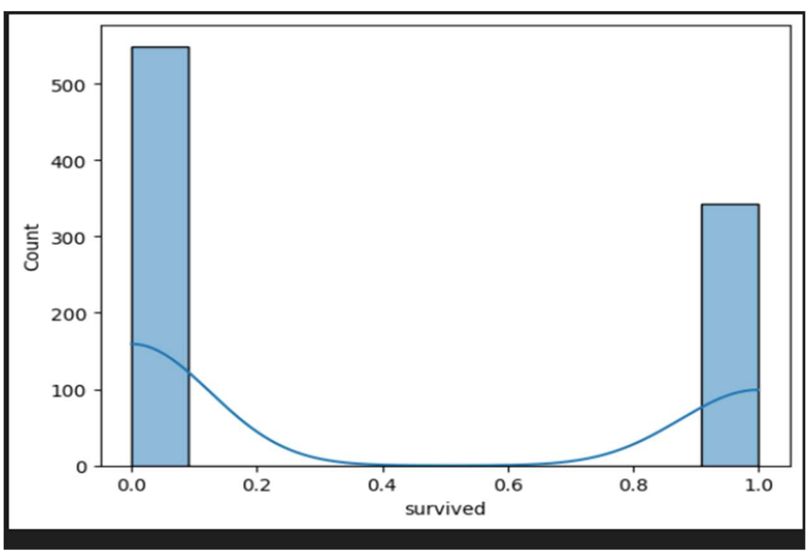
```
# Calculate the percentage of missing (null) values for each column
null_percentage = dataset.isnull().mean() * 100

# Filter out only columns with missing values (> 0%) and convert to DataFrame for heatmap
sns.heatmap(
    null_percentage[null_percentage > 0].to_frame(),  # Only columns with missing values
    annot=True,                # Display the percentage as numbers on the heatmap
    cmap="YlOrRd"              # Use the 'Yellow-Orange-Red' color map for better visualization
)

# Display the heatmap
plt.title("Percentage of Missing Values per Column")
plt.show()
```

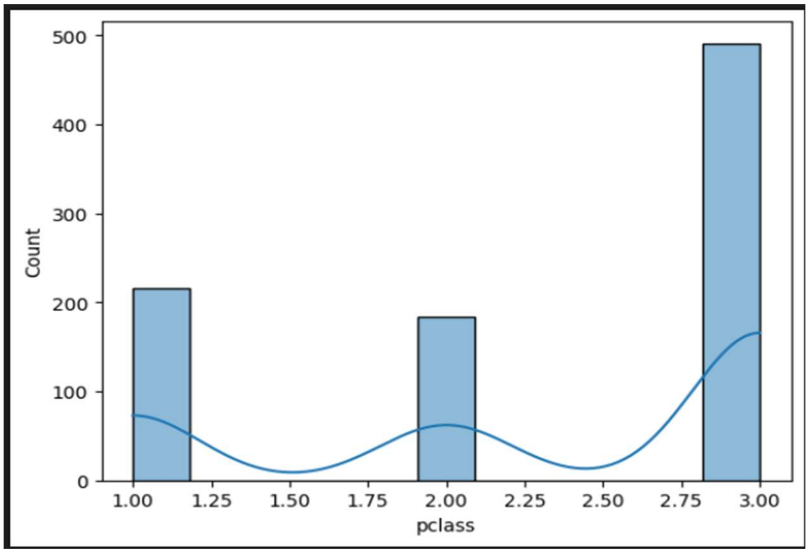


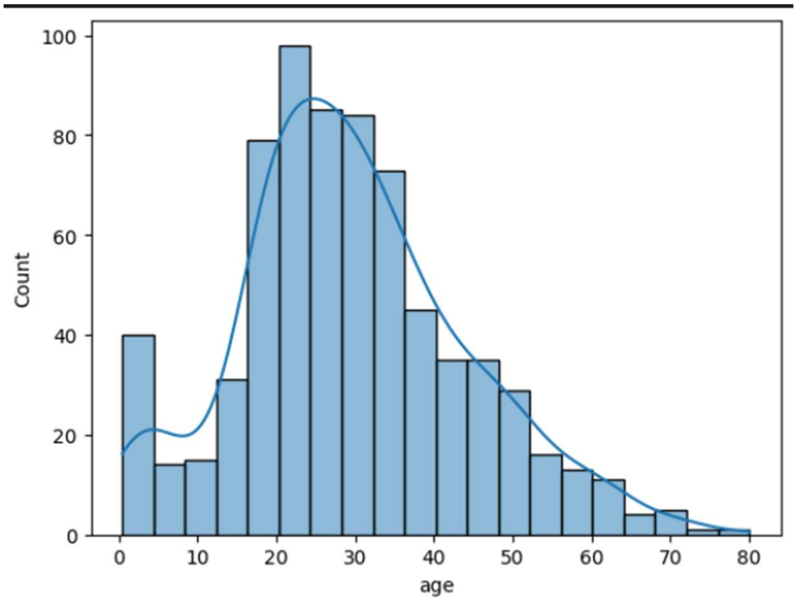
Plotting of the column frequencies:

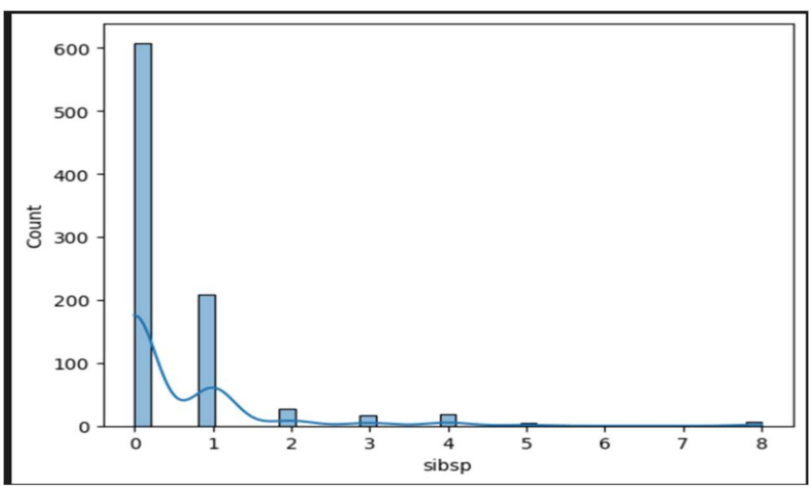
Bar graphs or count plots that show how often each category appears in columns like Sex, Embarked, etc., helping visualize data distribution.

```
# Loop through each numerical column (int64 or float64)
for col in dataset.select_dtypes(include=['int64', 'float64']).columns:
    # Plot a histogram with a KDE (Kernel Density Estimate) overlay for each numerical feature
    sns.histplot(dataset[col], kde=True)

    # Display the plot
    plt.show()
```

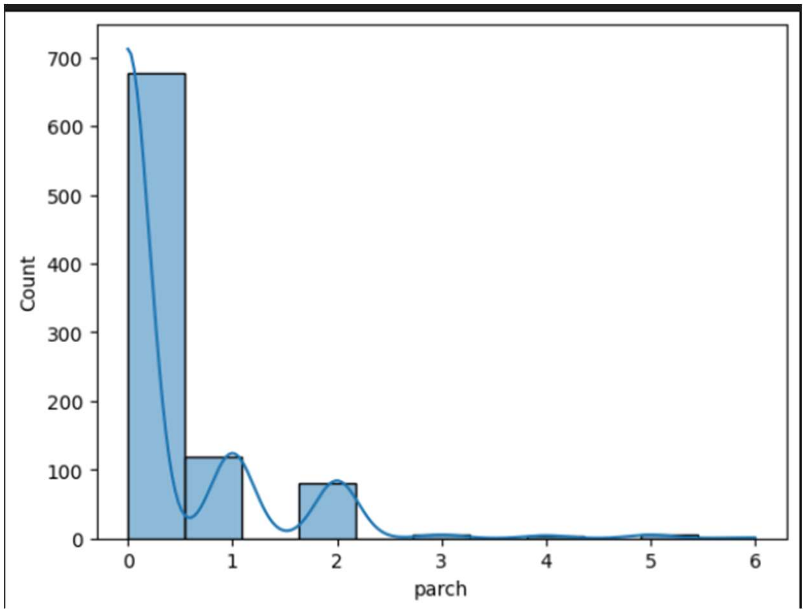




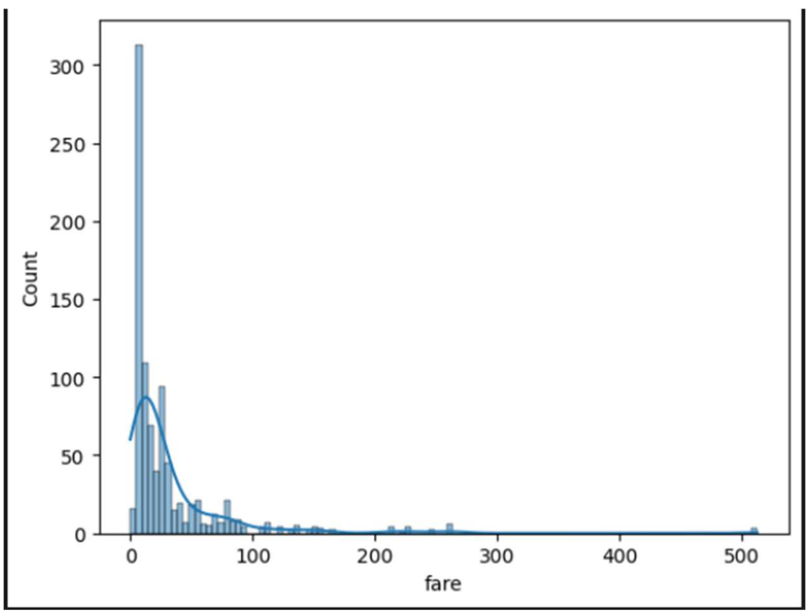




Devanshi Mittal



700
600
500
400
300
200
100
0
Count
0
1
2
3
4
5
6
parch



300
250
200
150
100
50
0
Count
0
100
200
300
400
500
fare