Celebal Assignment – Week 4

Missing Values Graphs:

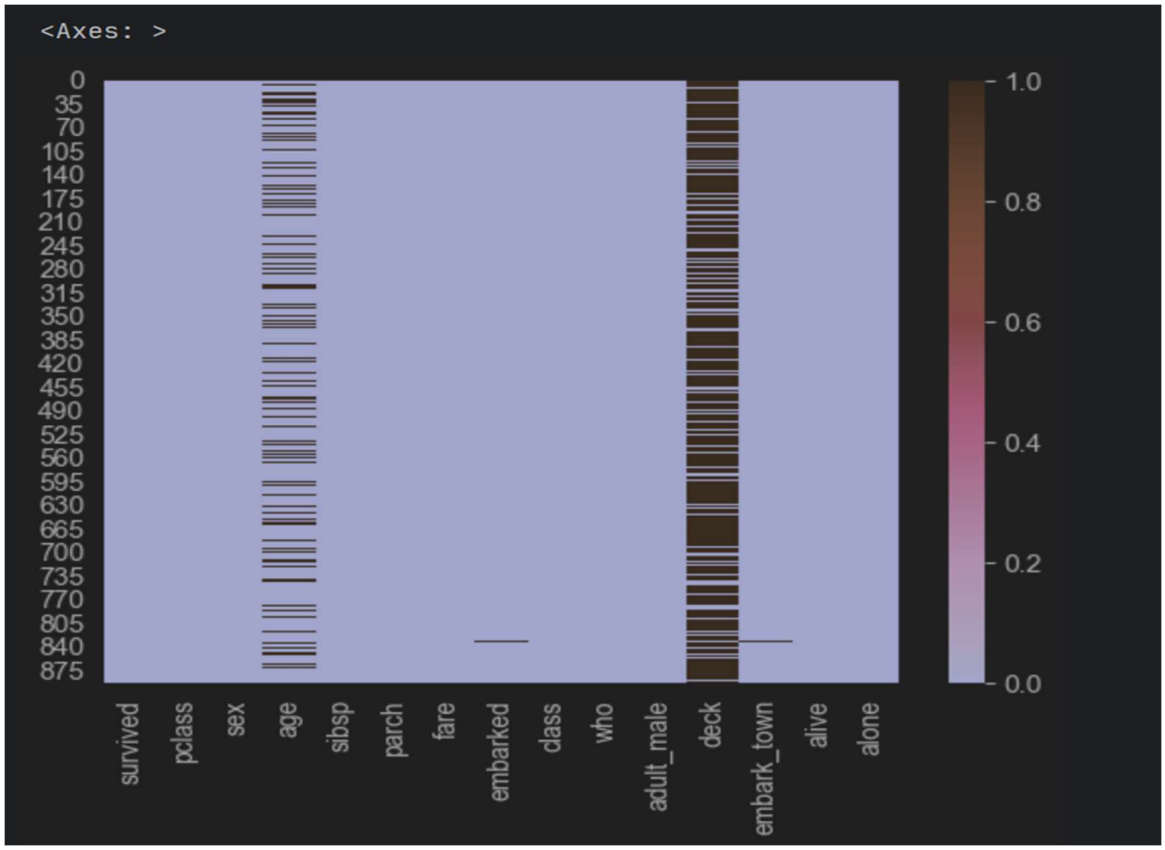
These graphs visually represent the columns in the dataset that have missing or null values. Heatmaps help identify which features (like Age, Cabin, Embarked, etc.) need data cleaning or imputation before modeling.

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

# Print the count of missing values for each column
print(missing_values)
[11]
```

```
survived         0
pclass           0
sex              0
age            177
```

```
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
```

```
# Import seaborn for data visualization
import seaborn as sns
# Visualize missing values using a heatmap
sns.heatmap(dataset.isnull())
[16]
```



Outliers Detection and Plotting:

The outliers are only detected of the columns those are having the int and float values.

This step involves identifying data points that significantly deviate from the normal range (e.g., unusually high fare or age values). Boxplots and scatter plots are commonly used to visualize and detect such outliers, which can affect model accuracy if not handled properly.

```
🔍 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: []
```

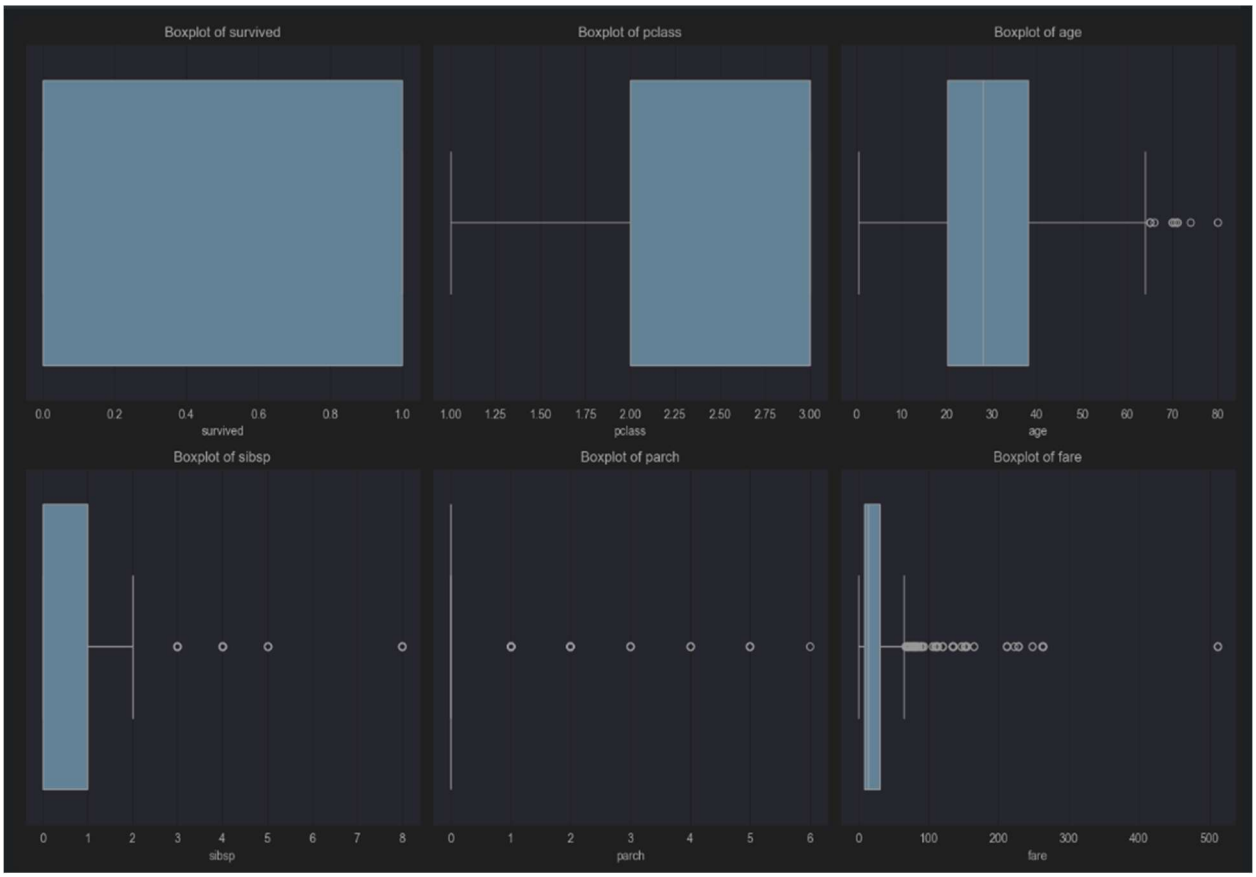
```
🔍 Column: age
Outlier Count: 11
Lower Bound: -6.69
Upper Bound: 64.81
Outlier Values: [66.  65.  71.  70.5 65.  65.  71.  80.  70.  70. ]
```

```
🔍 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]
```

```
🔍 Column: parch
Outlier Count: 213
Lower Bound: 0.00
Upper Bound: 0.00
Outlier Values: [1 2 1 5 1 1 5 2 2 1]
```

```
🔍 Column: fare
Outlier Count: 116
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]
```

```python
# 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()
```

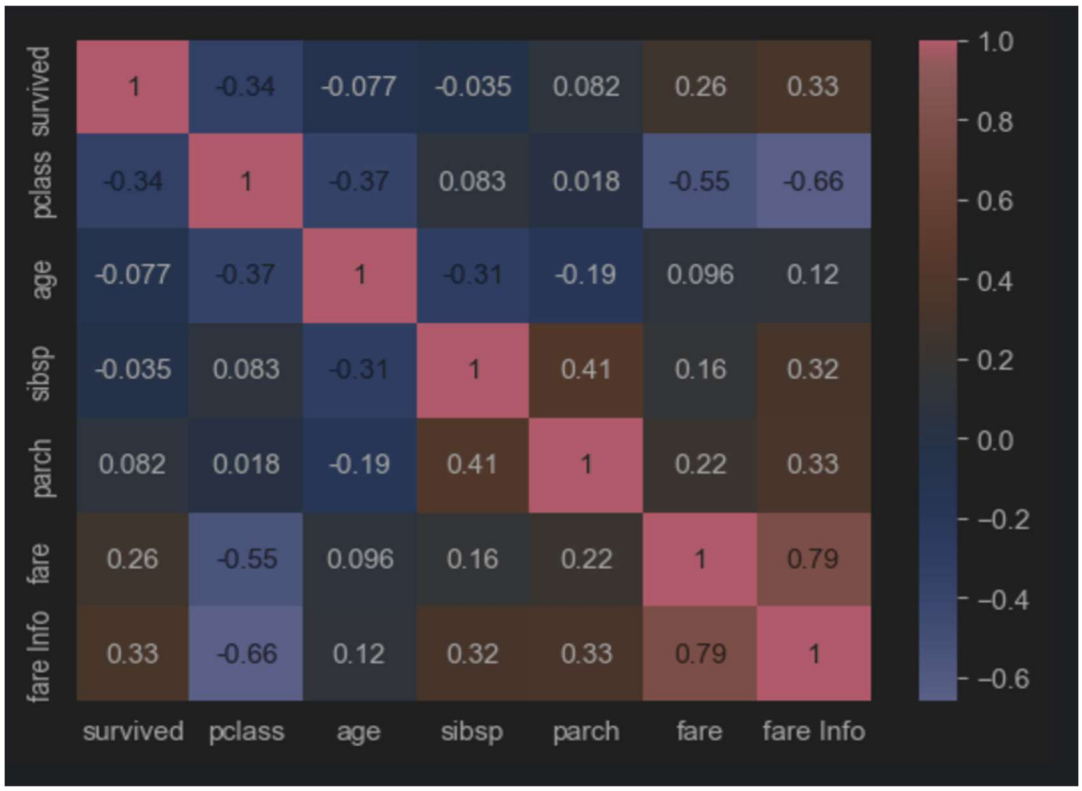


Tushar Nag

Correlation heatmap with numerical values:

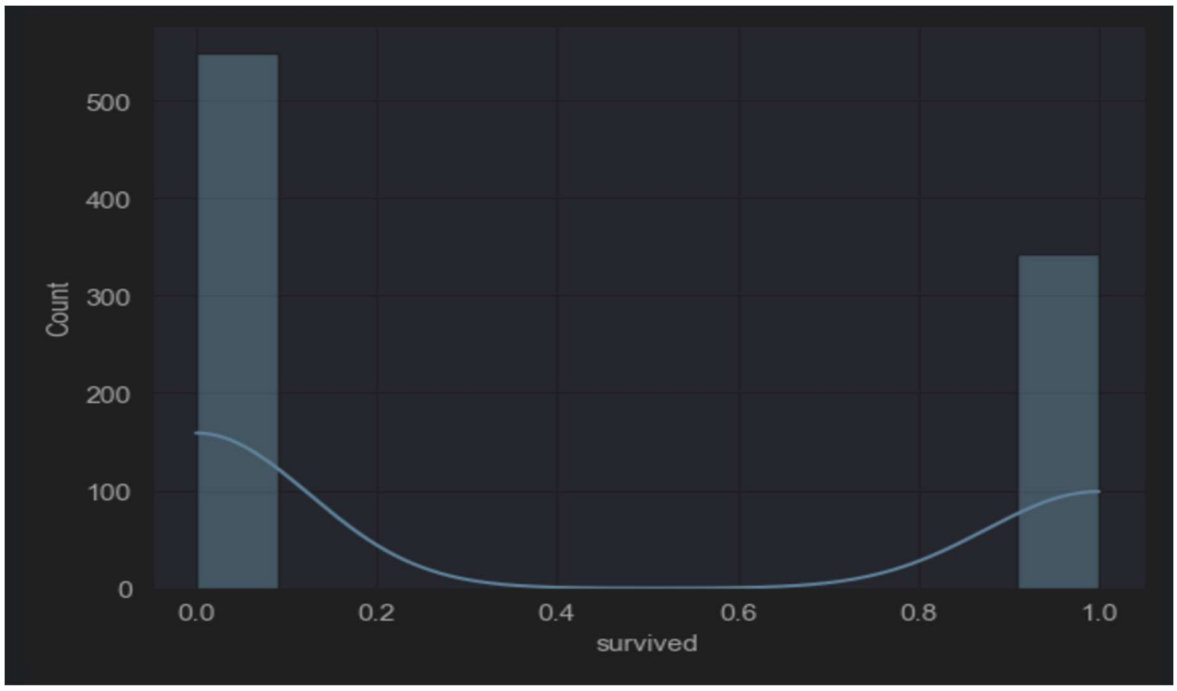
This visualization uses color gradients to represent the correlation coefficients between numerical features. It helps identify strong positive or negative relationships (e.g., between Fare and Pclass) which can influence model performance and feature engineering decisions.

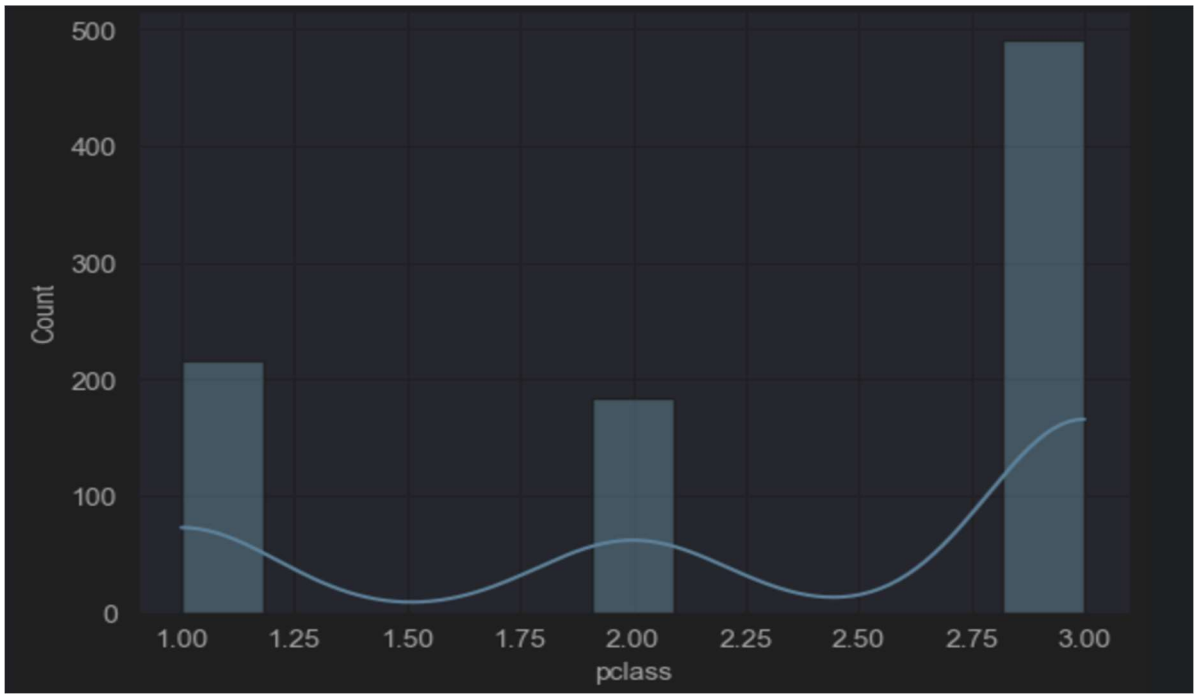
```
# Correlation heatmap with only numeric columns
corr = dataset.select_dtypes(include=['int64', 'float64']).corr()

# Create a heatmap to visualize the correlation matrix
# - annot=True shows the correlation values inside the boxes
# - cmap='coolwarm' gives a color gradient from cool (blue) to warm (red)
sns.heatmap(corr, annot=True, cmap='coolwarm')

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

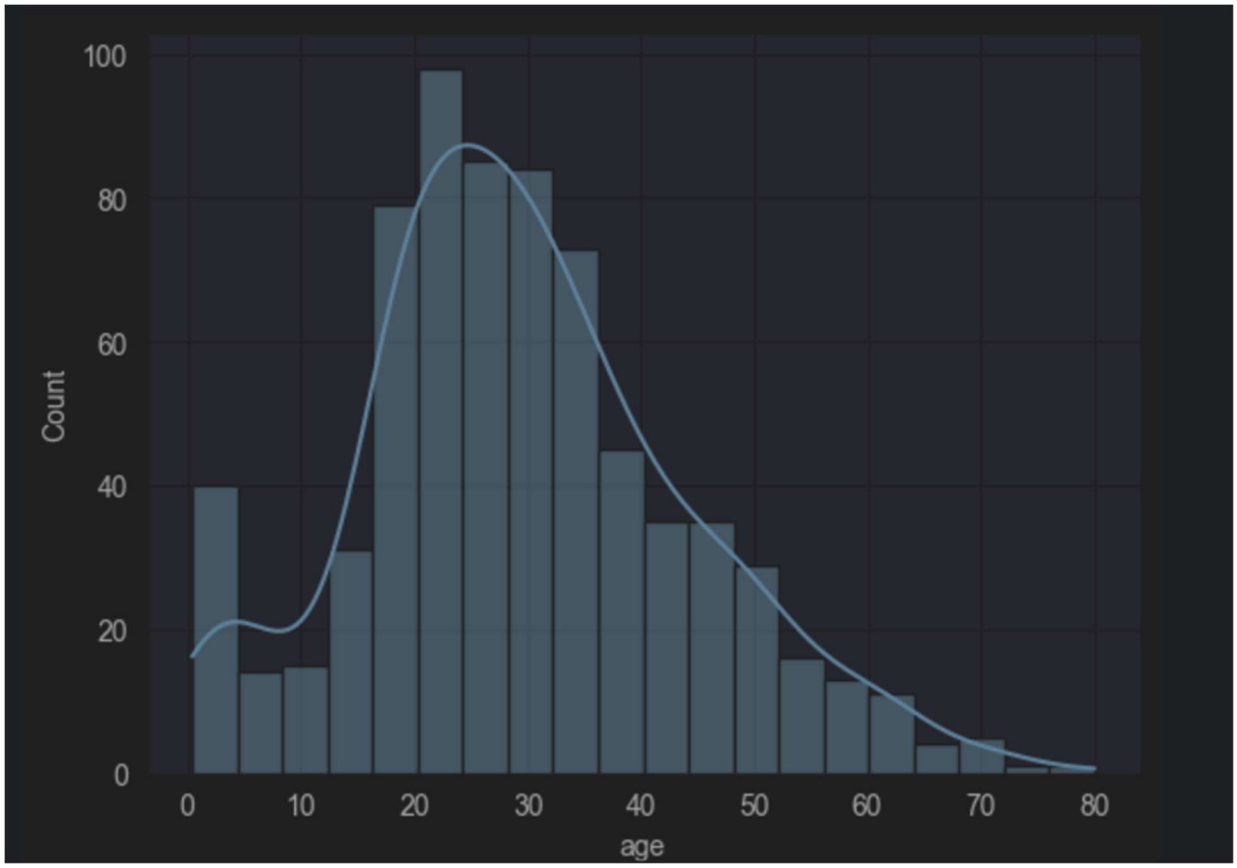


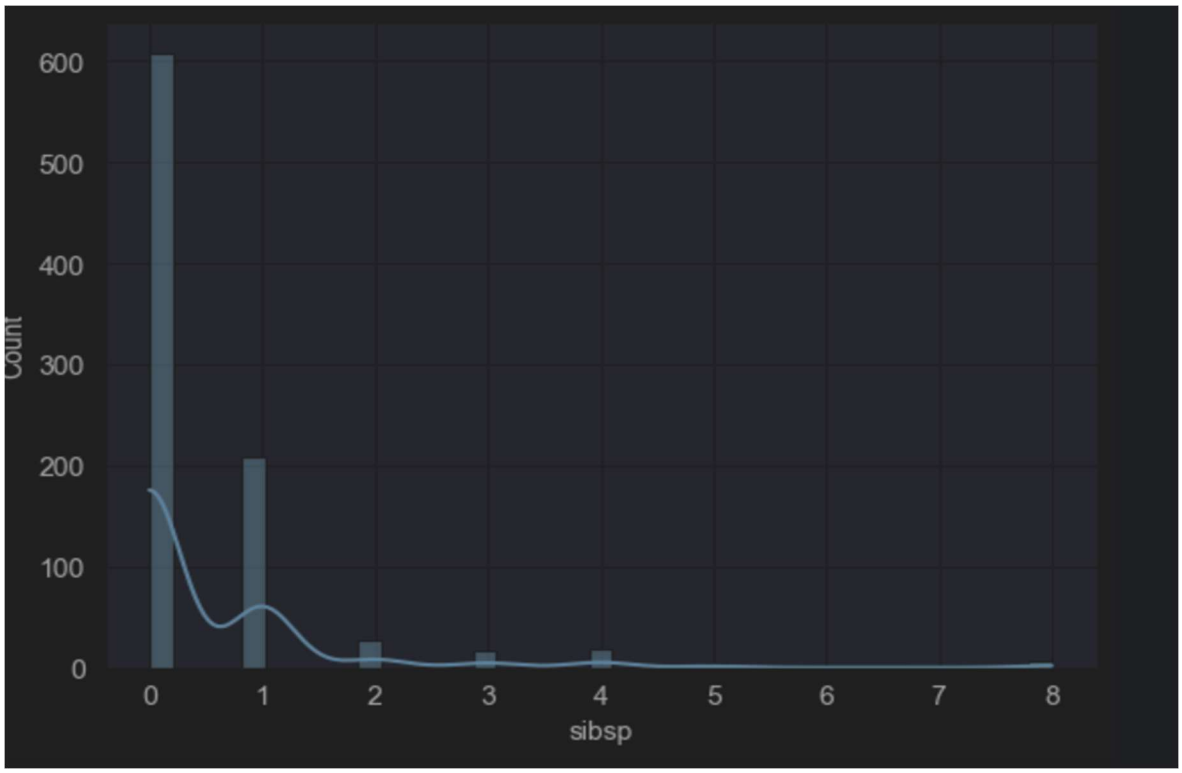
Plotting of the column frequencies:

This involves creating count plots or bar graphs to visualize how frequently each category appears in categorical columns (like Sex, Embarked, Pclass). It helps understand the distribution of data and detect any imbalances or dominant categories.

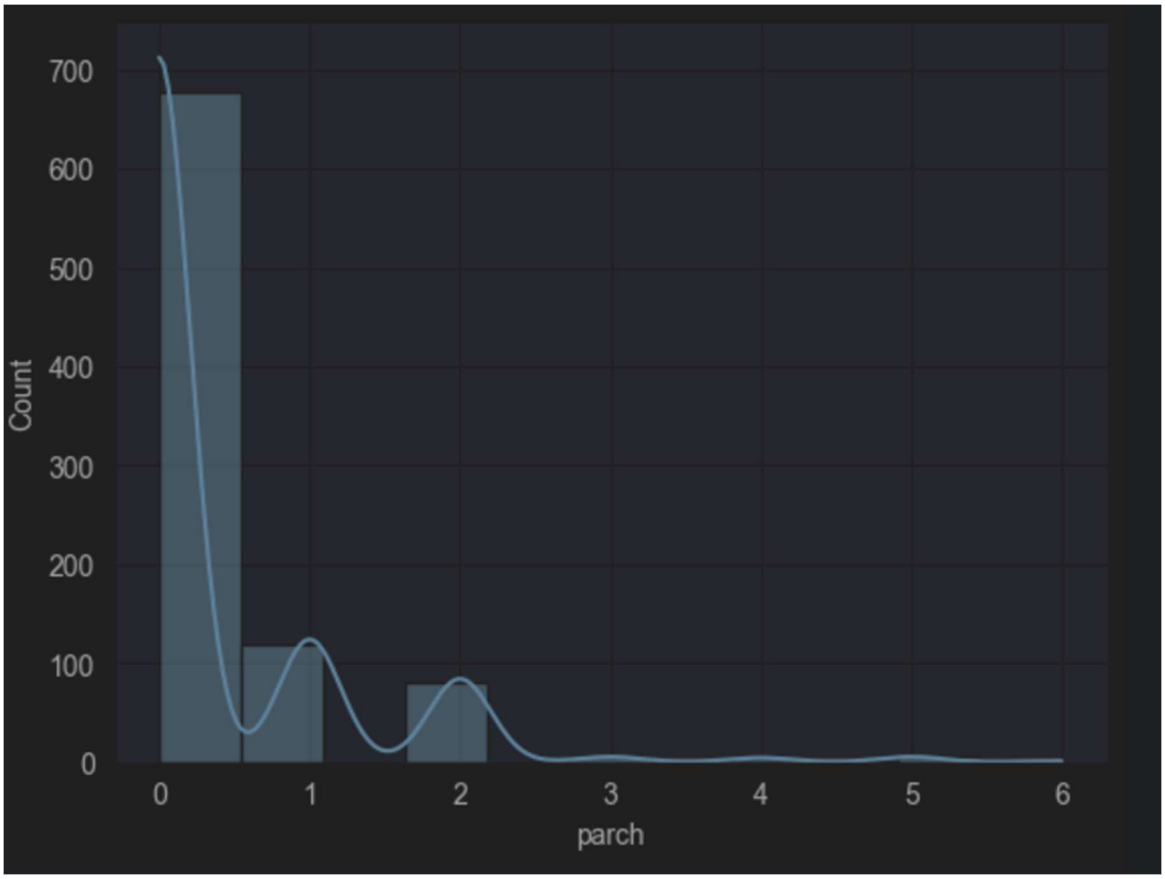


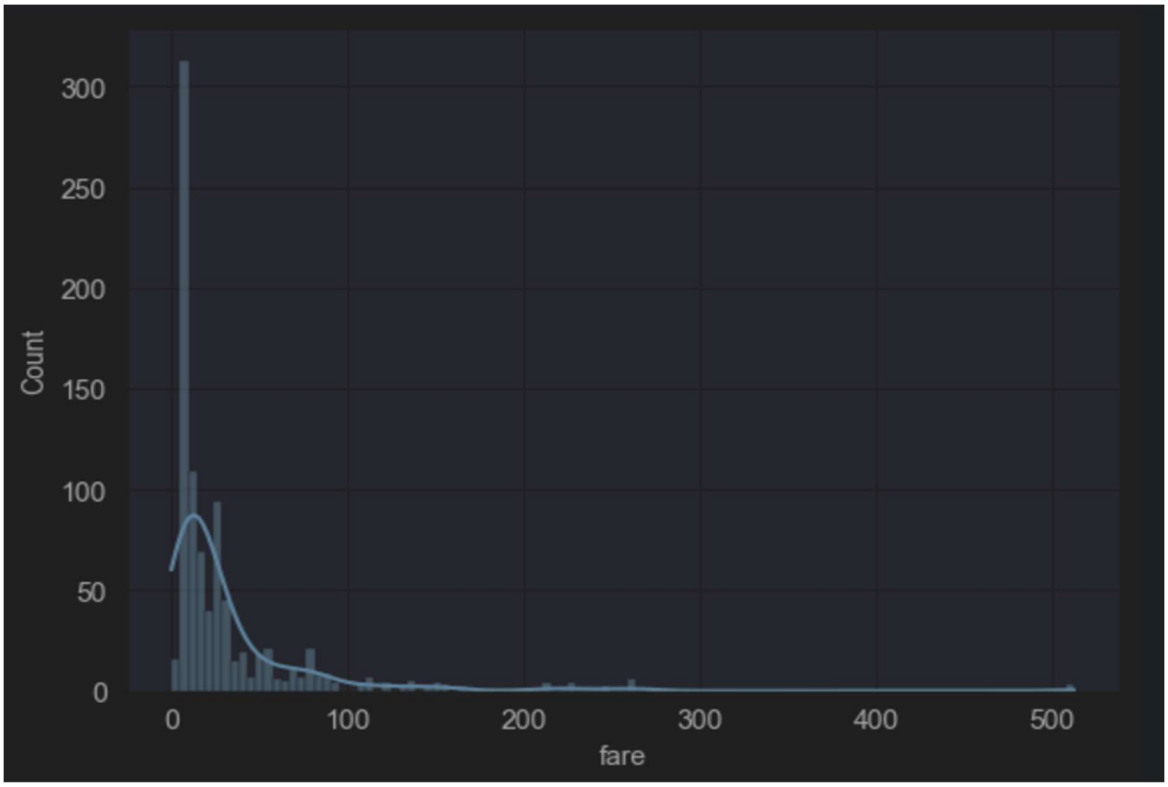



100
80
60
40
20
0
Count
0
10
20
30
40
50
60
70
80
age


600
500
400
300
200
100
0
Count
0
1
2
3
4
5
6
7
8
sibsp

Tushar Nag


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

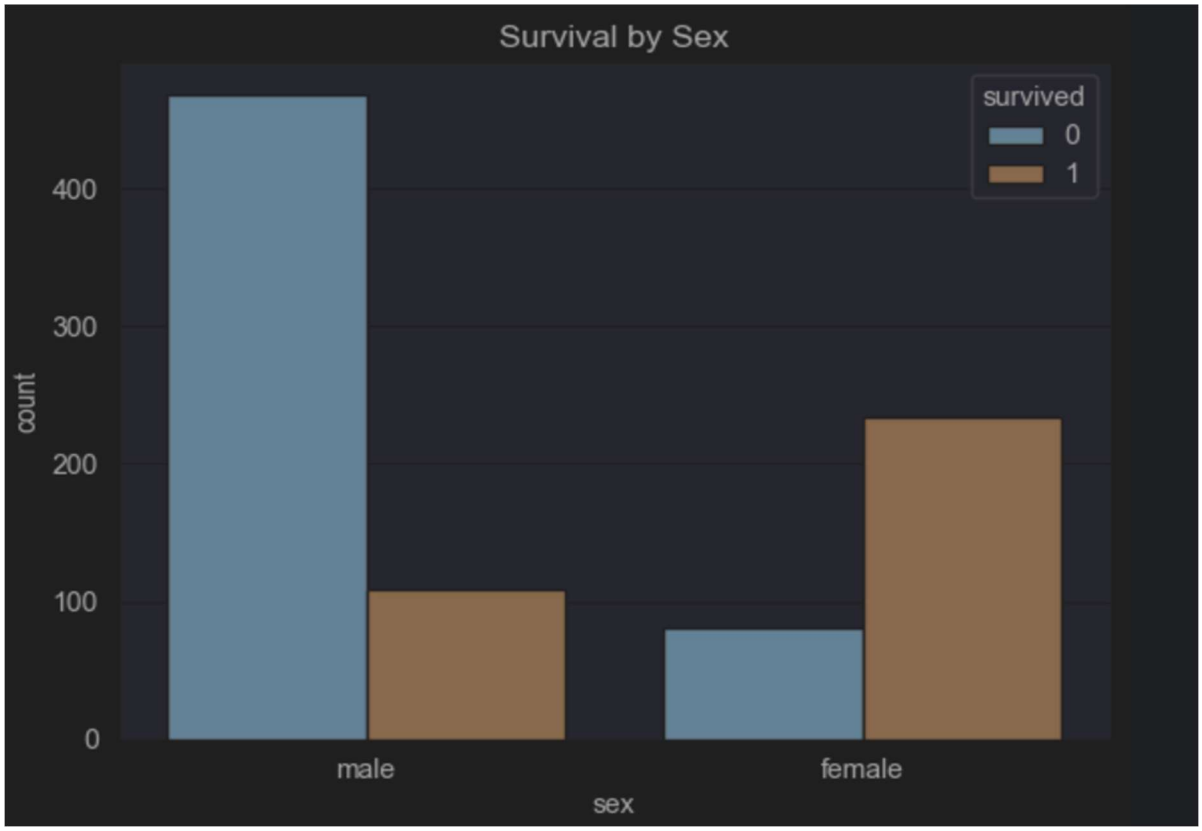

300
250
200
150
100
50
0
Count
0
100
200
300
400
500
fare

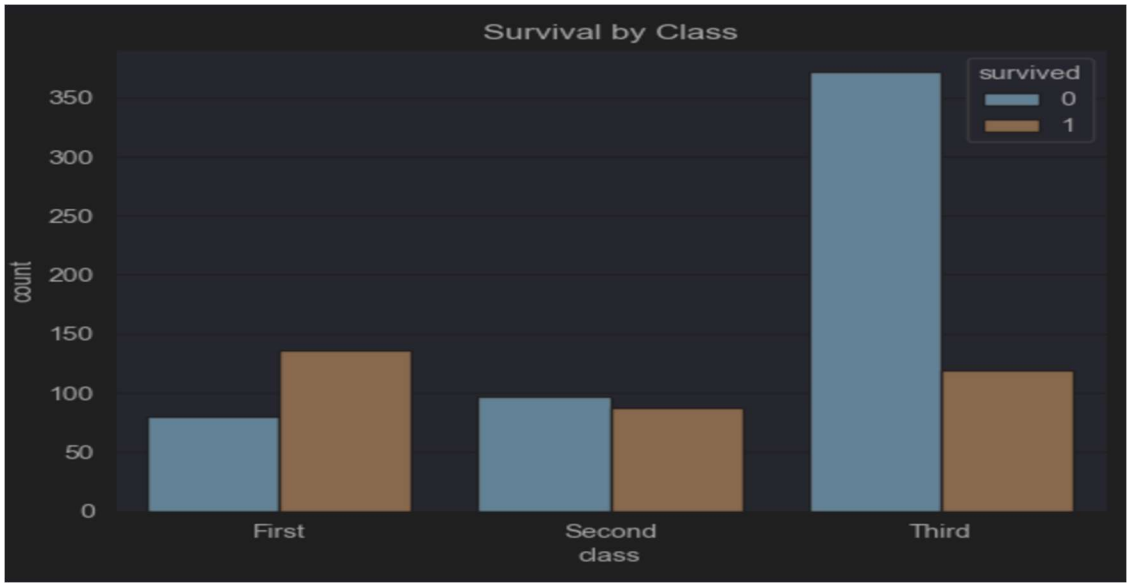
Tushar Nag

Relationship between variables:

This step explores how different features relate to each other, especially with the target variable (Survived). Visualizations like grouped bar plots, violin plots, and scatter plots reveal patterns or associations (e.g., survival rate by gender or class), which are crucial for model insights.

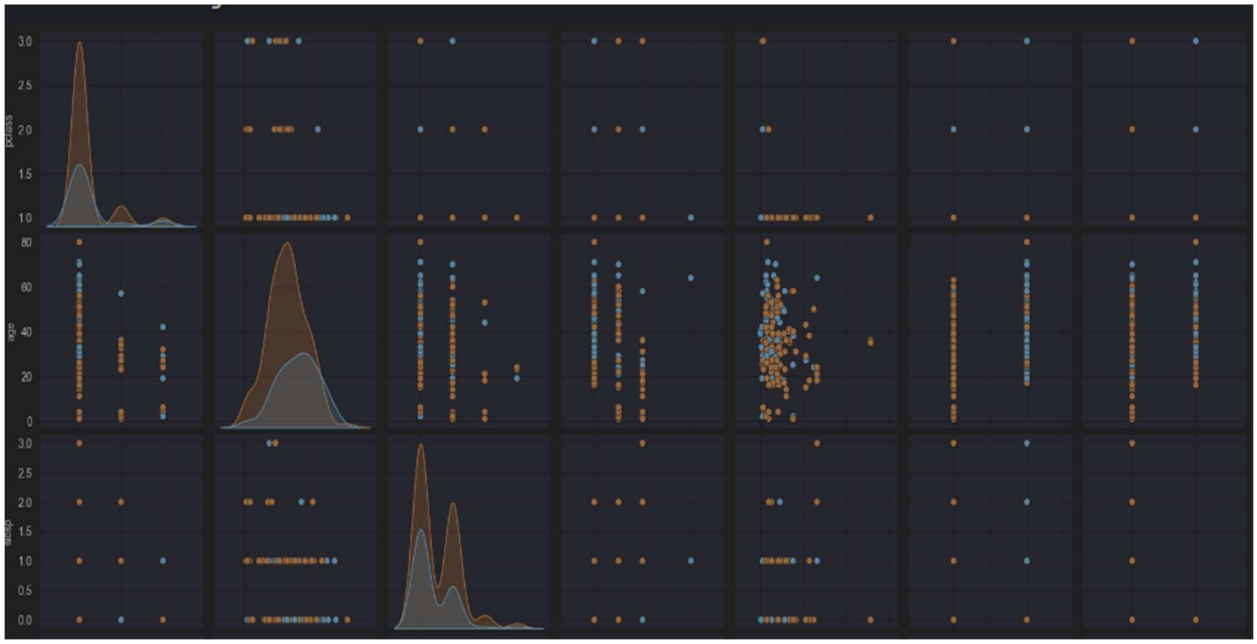
```
# Count plot to show survival distribution based on passenger sex
# hue='survived' adds a secondary category (survived or not) represented by different
 colors
sns.countplot(x='sex', hue='survived', data=dataset)
# Title of the plot
plt.title("Survival by Sex")
# Display the plot
plt.show()

# Count plot to show survival distribution based on passenger class
# Again, hue='survived' splits the bars into survived (1) and not survived (0)

sns.countplot(x='class', hue='survived', data=dataset)
# Title of the plot
plt.title("Survival by Class")

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

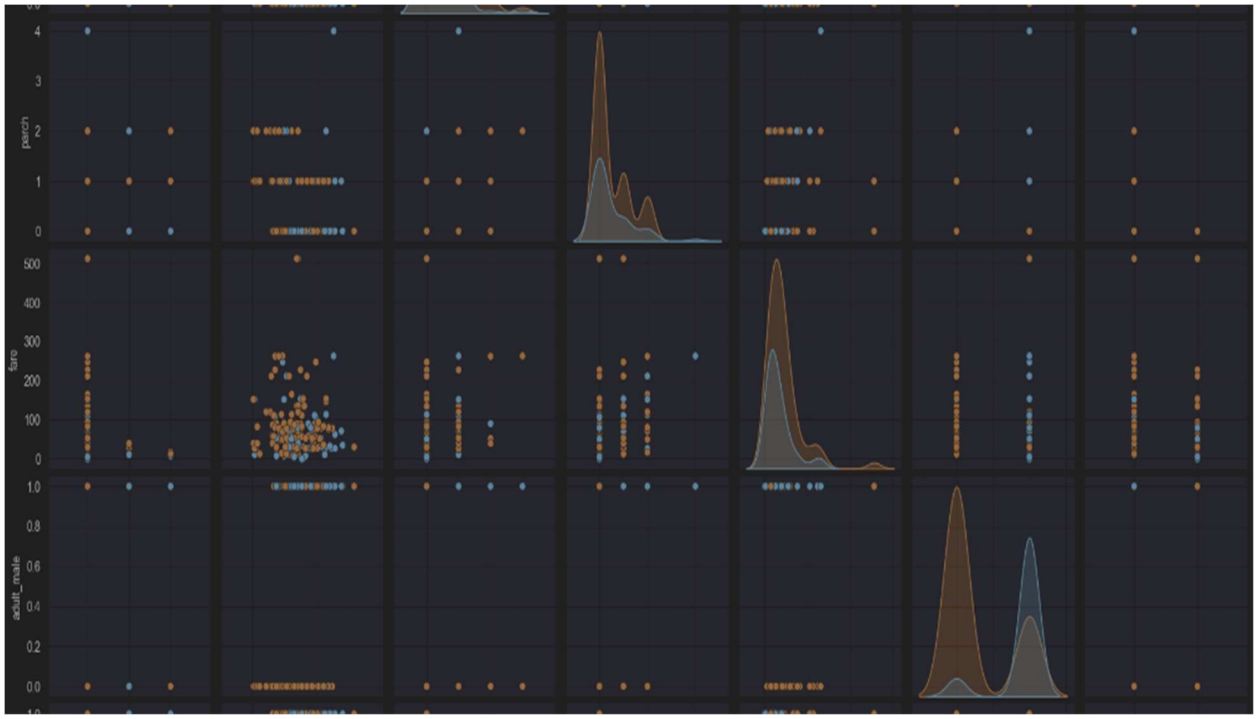


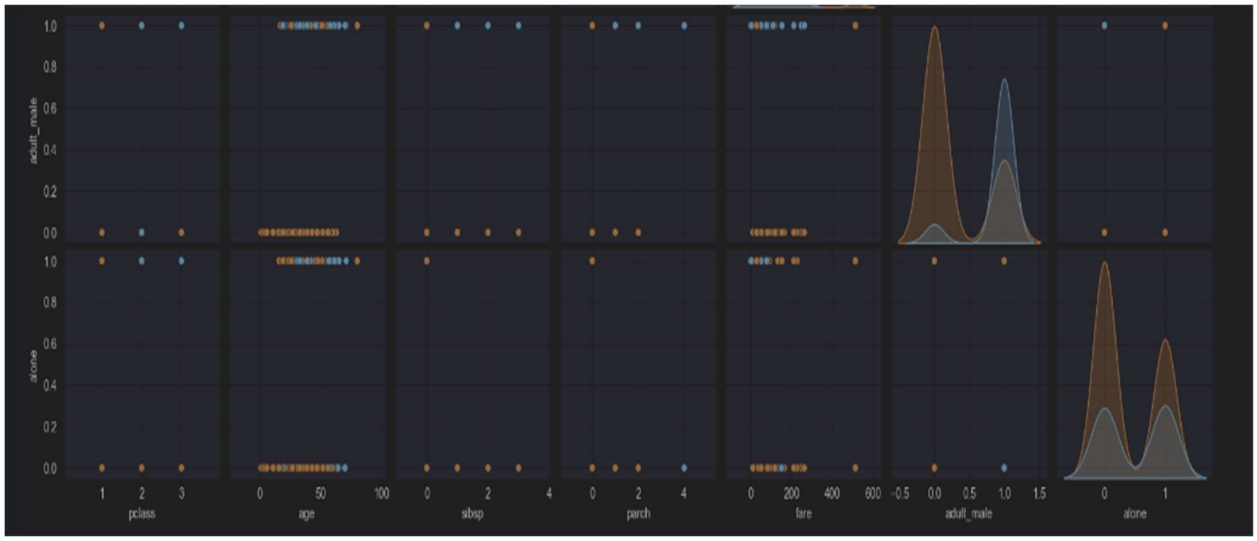


Creating Pairplot of all numerical features in the dataset:

A pairplot visualizes pairwise relationships between all numerical features using scatter plots and histograms. It helps uncover trends, clusters, and correlations, offering a quick overview of how variables like Age, Fare, and Pclass interact.




parch
fare
adult_male


adult_male
alone
pclass
age
sibsp
parch
fare
adult_male
alone

Tushar Nag

Calculating the percentage of missing values in each Columns:

This step computes what fraction (or percentage) of data is missing in each feature. It helps prioritize which columns need imputation, dropping, or special handling based on the severity of missing data.

```
# Calculate the percentage of missing (null) values in each column
null_percentage = dataset.isnull().mean() * 100
# Filter out only those columns which have missing values (> 0%)
# Convert the resulting Series into a DataFrame for better heatmap display
sns.heatmap(null_percentage[null_percentage > 0].to_frame(), annot=True,
 cmap="coolwarm")

✓ [26] 843ms
```