dwon64pbc

April 19, 2025

Importing essential Python libraries for data analysis and visualization

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading the Titanic dataset using Seaborn's built-in datasets This dataset contains information about passengers on the Titanic and whether they survived

[13]:
```
data = sns.load_dataset('titanic')
```

Displaying the first 5 rows of the Titanic dataset using .head() This gives a quick overview of the dataset structure and columns.

[3]:
```
data.head()
```

[3]:

	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

Displaying concise summary information about the Titanic dataset using .info() This provides details like the number of non-null entries, data types, and memory usage.

[4]:
```
data.info()
```

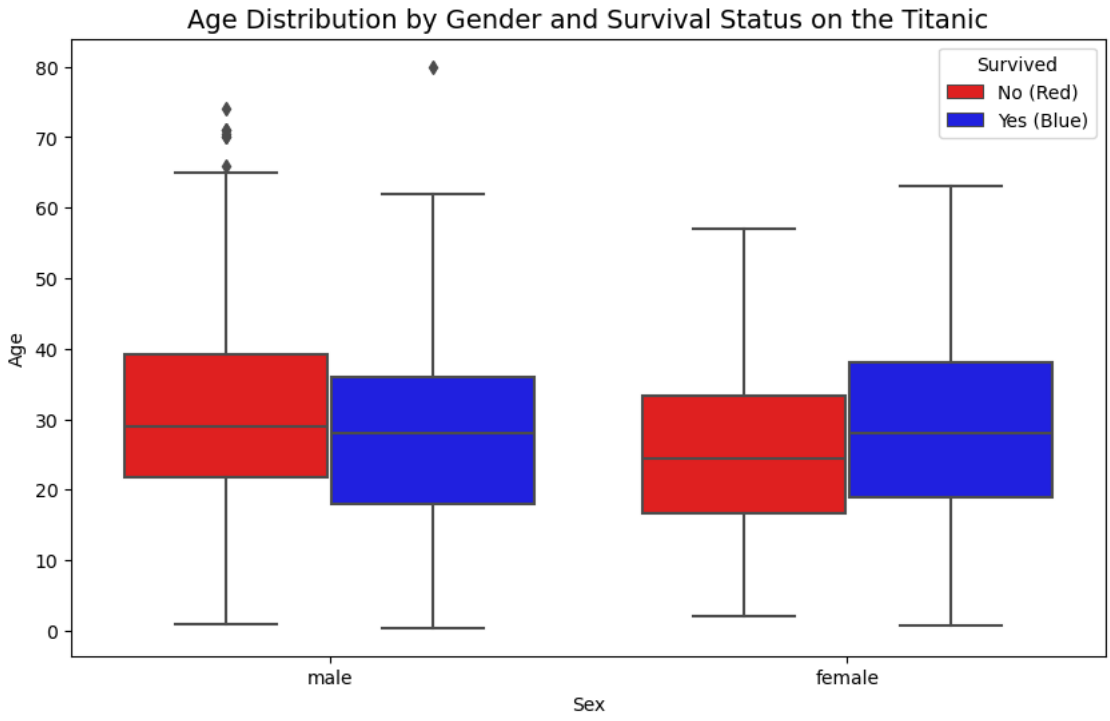
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
```

```
 #   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    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

Define a custom color palette for the box plot. The value `0` represents "Not Survived" and is assigned the color Red, while `1` represents "Survived" and is assigned the color Blue. Create a box plot to visualize the age distribution based on gender and survival status. The `plt.figure(figsize=(10, 6))` command adjusts the figure size for better readability, while the `sns.boxplot` function plots the age distribution by gender with a hue based on survival status, using the custom color palette defined earlier. Fix the legend labels for better understanding by assigning custom labels such as "No (Red)" for those who did not survive and "Yes (Blue)" for those who survived. The `plt.gca().get_legend_handles_labels()` method retrieves the legend handles, and `plt.legend` updates the legend with the correct titles. Add a title and axis labels to the plot. The `plt.title` method adds the plot's title, and `plt.xlabel` and `plt.ylabel` label the x-axis and y-axis, respectively. Display the plot using `plt.show()`, which renders the visualization on the screen.

[7]:
```
# Define highly contrasting colors
custom_palette = {0: 'red', 1: 'blue'} # 0 = Not Survived (Red), 1 =␣
 ↪Survived␣(Blue)
# Create the box plot
plt.figure(figsize=(10, 6))
sns.boxplot(x='sex', y='age', hue='survived', data=data, palette=custom_palette)
# Fix legend labels properly
legend_labels = ['No (Red)', 'Yes (Blue)']
handles, _ = plt.gca().get_legend_handles_labels()
plt.legend(handles, legend_labels, title='Survived')
# Add title and labels
plt.title('Age Distribution by Gender and Survival Status on the␣
 ↪Titanic',fontsize=14)
plt.xlabel('Sex')
plt.ylabel('Age')
# Show plot
```

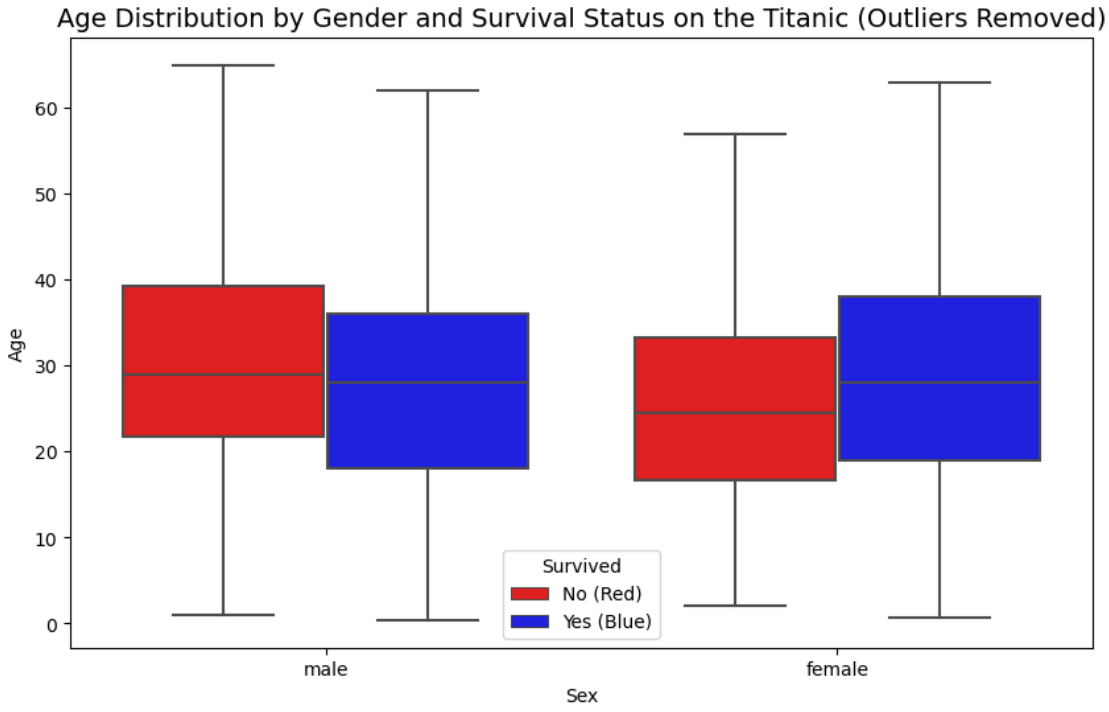
```
plt.show()
```



Define a custom color palette for the box plot. The value `0` represents "Not Survived" (Red), while `1` represents "Survived" (Blue). This palette is applied to the box plot. **Create a box plot** to visualize the age distribution based on gender and survival status, but with outliers removed by setting `showfliers=False`. The `plt.figure(figsize=(10, 6))` command ensures the plot is large enough for readability. The `sns.boxplot` function plots the age distribution by gender, with the hue based on survival status, using the custom color palette defined earlier. **Fix the legend labels** to provide clearer information. The custom legend labels are "No (Red)" for people who did not survive and "Yes (Blue)" for those who survived. This is achieved by using `plt.gca().get_legend_handles_labels()` to retrieve the legend handles and `plt.legend` to update the legend with the custom labels. **Add a title and axis labels**. The plot title, "Age Distribution by Gender and Survival Status on the Titanic (Outliers Removed)", is set using `plt.title`. Axis labels are added with `plt.xlabel` for the x-axis and `plt.ylabel` for the y-axis. **Display the plot** using `plt.show()` to render the final visualization.

[12]:
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10, 6))
sns.boxplot(x='sex', y='age', hue='survived', data=data,␣
 ↪palette=custom_palette, showfliers=False)
# Fix legend labels properly
legend_labels = ['No (Red)', 'Yes (Blue)']
handles, _ = plt.gca().get_legend_handles_labels()
```

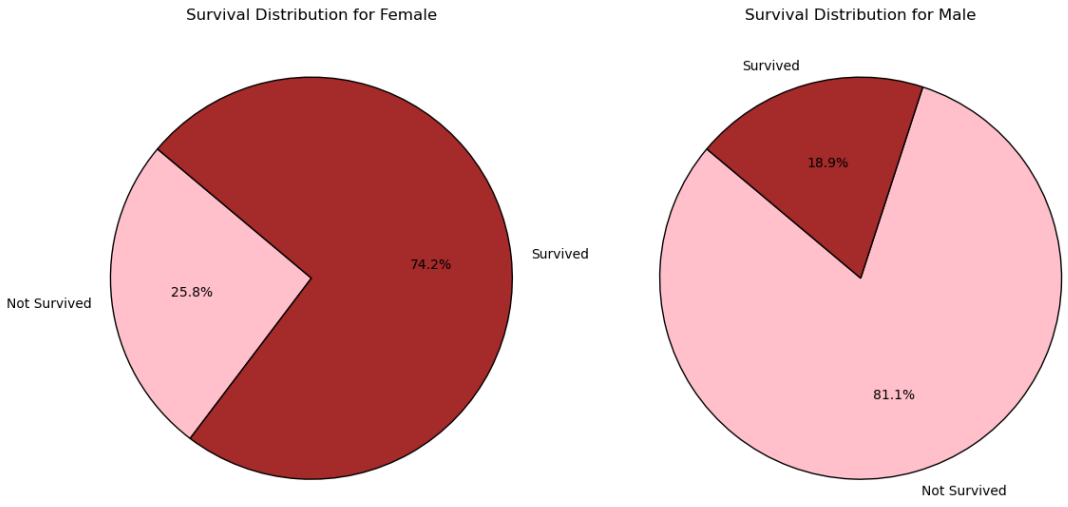
```
plt.legend(handles, legend_labels, title='Survived')
# Add title and labels
plt.title('Age Distribution by Gender and Survival Status on the Titanic␣
 ↪(Outliers Removed)', fontsize=14)
plt.xlabel('Sex')
plt.ylabel('Age')
# Show plot
plt.show()
```



Count of survival status grouped by gender: The dataset is grouped by 'sex' and 'survived' to count the number of passengers who survived and those who didn't, broken down by gender. The `.size()` method counts the occurrences, and `.unstack()` reshapes the data to make it easier for visualization. **Define colors for better visualization**: A custom color palette is defined, where 'pink' represents "Not Survived" and 'brown' represents "Survived", for better visual distinction in the pie charts. **Create the pie charts**: A figure with two subplots (1 row and 2 columns) is created using `plt.subplots()`, with the `figsize` set to (12, 6) for better readability. A `for` loop is used to create individual pie charts for male and female survival distributions. **Loop through each gender**: In each iteration, a pie chart is created for the survival status of a given gender (either male or female). The `axes[i].pie()` function is used to plot the pie chart with custom labels, percentage display, and colors. The `wedgeprops={'edgecolor': 'black'}` adds borders to the slices for clearer visualization. **Set titles for each subplot**: The title for each pie chart is set dynamically using `f'Survival Distribution for {gender.capitalize()}'`, where `gender.capitalize()` ensures the first letter is capitalized for proper formatting. **Show the plot**: The layout is adjusted using `plt.tight_layout()` to avoid overlapping of subplots, and the plot

is rendered using `plt.show()`.

[10]:
```
import matplotlib.pyplot as plt

# Count of survival status grouped by gender
survival_counts = data.groupby(['sex', 'survived']).size().unstack()

# Define colors for better visualization
colors = ['pink', 'brown']  # pink = Not survived, brown = Survived

# Create the pie charts
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

for i, gender in enumerate(survival_counts.index):
    axes[i].pie(
        survival_counts.loc[gender],
        labels=['Not Survived', 'Survived'],
        autopct='%1.1f%%',
        colors=colors,
        startangle=140,
        wedgeprops={'edgecolor': 'black'}
    )
    axes[i].set_title(f'Survival Distribution for {gender.capitalize()}')

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



[ ]: