EDA Dataset content

Reading Material

PW SKILLS

Topics Covered

1. Understanding the Dataset

Dataset Overview

Firstly, we load the dataset and get an overview of the structure

```
import pandas as pd

# Load the Titanic dataset
data = pd.read_csv('Titanic-Dataset.csv')

# Display the first few rows of the dataset
data.head()

# Display the shape of the dataset
data.shape

# Display the data types of each column
data.dtypes
```

We took for example the Titanic Dataset. By running this code we can view the shape and data types of the dataset.

This step provides a quick look at the dataset, including the number of rows and columns and the types of data each column contains. Understanding the structure is crucial for the subsequent analysis.

Identifying Numerical and Categorical Columns

```
# Identify numerical columns
numerical_cols = data.select_dtypes(include=['int64',
'float64']).columns
print(f"Numerical Columns: {numerical_cols}")

# Identify categorical columns
categorical_cols =
data.select_dtypes(include=['object']).columns
print(f"Categorical Columns: {categorical_cols}")
```

Separating the dataset into numerical and categorical columns is crucial for understanding the types of analyses and visualizations that can be performed on each type of data.

Checking for Unique Values in Categorical Columns

```
# Check for unique values in categorical columns
for col in categorical_cols:
    print(f"{col}: {data[col].unique()}")
```

Examining the unique values in categorical columns helps identify the categories or labels present in the data. This information is essential for understanding the distribution of data within each category and for creating appropriate visualizations and analyses.

Data Dictionary

```
data_dictionary = pd.DataFrame({
    'Column Name': data.columns,
    'Data Type': data.dtypes,
    'Description': [
        'Unique ID for each passenger',
 'Survival status (0 = No, 1 = Yes)',
        'Ticket class (1, 2, or 3)',
        'Passenger\'s name',
        'Gender of the passenger',
        'Age of the passenger',
        'Number of siblings/spouses aboard',
        'Number of parents/children aboard',
        'Ticket number',
        'Ticket fare',
        'Cabin number',
        'Port of embarkation (C = Cherbourg, Q =
Queenstown, S = Southampton)'
    ]
})
```

Out[7]:

	Column Name	Data Type	Description
PassengerId	PassengerId	int64	Unique ID for each passenger
Survived	Survived	int64	Survival status (0 = No, 1 = Yes)
Pclass	Pclass	int64	Ticket class (1, 2, or 3)
Name	Name	object	Passenger's name
Sex	Sex	object	Gender of the passenger
Age	Age	float64	Age of the passenger
SibSp	SibSp	int64	Number of siblings/spouses aboard
Parch	Parch	int64	Number of parents/children aboard
Ticket	Ticket	object	Ticket number
Fare	Fare	float64	Ticket fare
Cabin	Cabin	object	Cabin number
Embarked	Embarked	object	Port of embarkation (C = Cherbourg, Q = Queens...

Insights from the Data Dictionary

1. **Column Overview:** The data dictionary provides a clear overview of each column in the dataset, helping analysts quickly understand what data is available.

2. **Data Types:** By showing the data types (e.g., numeric, object), the dictionary helps identify which columns can be used for numerical analysis and which are categorical. For example, "Survived" is a numeric column that can be treated as categorical for analysis.

3. **Descriptions:** The descriptions provide context for each column, making it easier to understand the dataset's structure and the significance of each variable. This is especially useful for new team members or stakeholders who may not be familiar with the dataset.

4. **Data Preparation:** Knowing the data types and descriptions helps in preparing the data for analysis. For instance, categorical variables like "Sex" and "Embarked" need to be encoded before modeling, while continuous variables like "Age" and "Fare" may require scaling.

5. **Identifying Missing Values:** The data dictionary can also prompt analysts to check for missing values in specific columns, such as "Age" and "Cabin," which can possibly have missing entries..

6. **Guidance for Analysis:** The data dictionary serves as a reference point for selecting appropriate statistical tests and visualizations, guiding the analysis process effectively.

2. Data Quality Check

```
# Checking Missing Values
print("Missing Values:")
print(data.isnull().sum())
```

```
# Checking for Duplicates
duplicates = data.duplicated().sum()
print(f"\nNumber of Duplicate Rows: {duplicates}")

# Checking for Incorrect Data Types
print("\nData Types:")
print(data.dtypes)

# Addressing Inconsistencies
# Check for unique values in categorical columns to
identify inconsistencies
print("\nUnique Values in Categorical Columns:")
for col in data.select_dtypes(include=['object']).columns:
    print(f"{col}: {data[col].unique()}")

# Checking for Outliers
# Using box plots to visualize potential outliers in
numerical columns
import matplotlib.pyplot as plt
import seaborn as sns


plt.figure(figsize=(10, 6))
sns.boxplot(data=data[['Age', 'Fare']])
plt.title('Box Plot for Age and Fare')
plt.show()
```

Checking Missing Values: The output will show the count of missing values for each column. For instance, if the "Age" column has many missing values, it indicates a need for imputation or further investigation.
Insight: Handling missing values is crucial for maintaining data integrity. Imputation strategies (mean, median, mode) or dropping rows/columns can be applied based on the context.

Checking for Duplicates: The number of duplicate rows will be displayed. If duplicates exist, it may indicate data collection issues.
Insight: Duplicate entries can skew the analysis results. It's essential to remove duplicates to ensure each observation is unique.

Checking for Incorrect Data Types: The data types of each column will be displayed, helping to identify any columns that may need conversion (e.g., converting "Survived" to categorical).
Insight: Correct data types are vital for accurate analysis and effective visualization. Incorrect types can lead to errors in calculations and interpretations.

Addressing Inconsistencies: Unique values in categorical columns will help identify any inconsistencies (e.g., variations in spelling or case).

Insight: Standardizing categorical values ensures consistency and accuracy in analysis. For example, ensuring that all entries for "Embarked" are uniformly formatted (e.g., "C", "Q", "S").

Checking for Outliers: The box plot visualizes potential outliers in the "Age" and "Fare" columns. Points outside the whiskers indicate outliers.

Insight: Identifying outliers is essential as they can significantly affect statistical analyses and model performance. Depending on the context, outliers may need to be investigated further, transformed, or removed.

3. Descriptive Statistics

Descriptive statistics summarize and organize the characteristics of a dataset. They provide an overview of the data through measures of central tendency, dispersion, distribution, and correlation.

Measures of Central Tendency
Measures of central tendency provide a single value that attempts to describe a set of data by identifying the central position within that set. The most common measures are the mean, **median**, and **mode**.
For example for 'Age' Column

```
# Calculate mean, median, and mode
mean_age = data['Age'].mean()
median_age = data['Age'].median()
mode_age = data['Age'].mode()[0]

print(f"Mean Age: {mean_age}")
print(f"Median Age: {median_age}")
print(f"Mode Age: {mode_age}")
```

Mean: The mean provides an average value for each numerical variable, giving a general idea of the dataset's central point. For example, if the mean age is around 29.7 years, it suggests that the average passenger was in their late twenties.

Median: The median is the middle value when the data is sorted. If the median age is 28, this indicates that half of the passengers were younger than 28, which can be more informative in skewed distributions.

Mode: The mode represents the most frequently occurring value. If the mode for the fare is 7.25, it indicates that this fare was the most common among passengers.

Measures of Dispersion
Dispersion measures describe the spread of data. Key measures include range, variance, and standard deviation.

```
# Calculate range, variance, and standard deviation
range_age = data['Age'].max() - data['Age'].min()
variance_age = data['Age'].var()
std_dev_age = data['Age'].std()

print(f"Age Range: {range_age}")
print(f"Age Variance: {variance_age}")
print(f"Age Standard Deviation: {std_dev_age}")
```

Variance: Variance measures how much the values deviate from the mean. A high variance indicates that the data points are spread out over a wider range of values.

Standard Deviation: This is the square root of the variance and provides a measure of dispersion in the same units as the data. A standard deviation of 14.5 in fares suggests significant variability in ticket prices.

Range: The range gives the difference between the maximum and minimum values, indicating the spread of the data. For instance, if the range of ages is 0 to 80, it shows that the dataset includes both very young and older passengers.

Distribution Analysis
Distribution analysis involves understanding how data is spread or distributed. Key concepts include **skewness** and **kurtosis**.

```
# Calculate skewness and kurtosis
skewness_age = data['Age'].skew()
kurtosis_age = data['Age'].kurt()

print(f"Age Skewness: {skewness_age}")
print(f"Age Kurtosis: {kurtosis_age}")
```

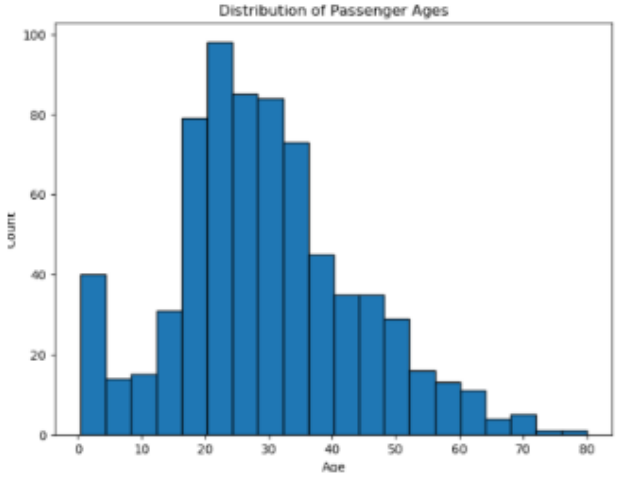
Skewness: Measures the asymmetry of the distribution. A skewness value > 0 indicates a right-skewed distribution, while < 0 indicates a left-skewed distribution.

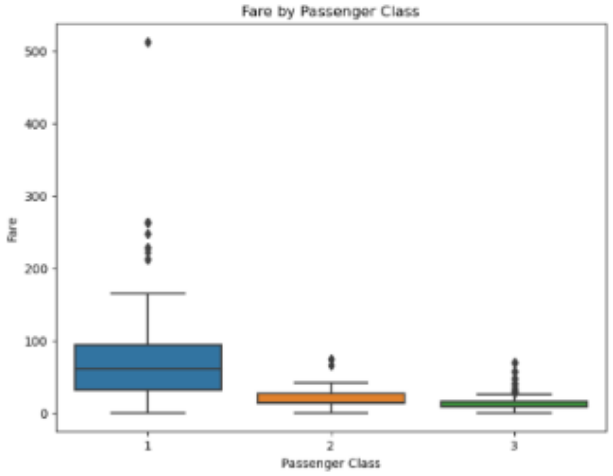
Kurtosis: Kurtosis measures the "tailedness" of the distribution. A positive kurtosis value indicates a distribution with heavy tails and a sharper peak than a normal distribution, while a negative value indicates lighter tails. For example, a kurtosis of 3 suggests that the distribution is similar to a normal distribution, while a value significantly greater than 3 indicates potential outliers.

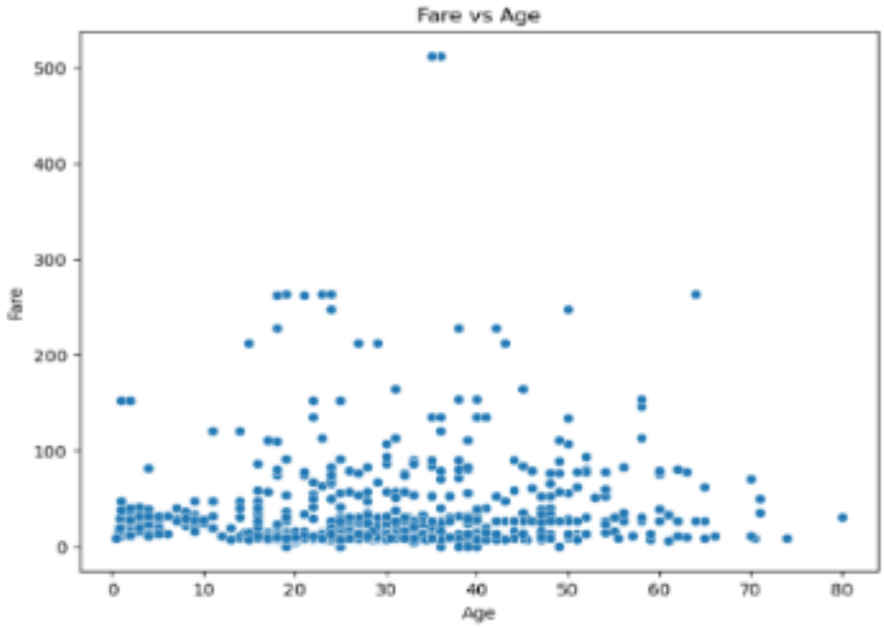
4. Data Visualization

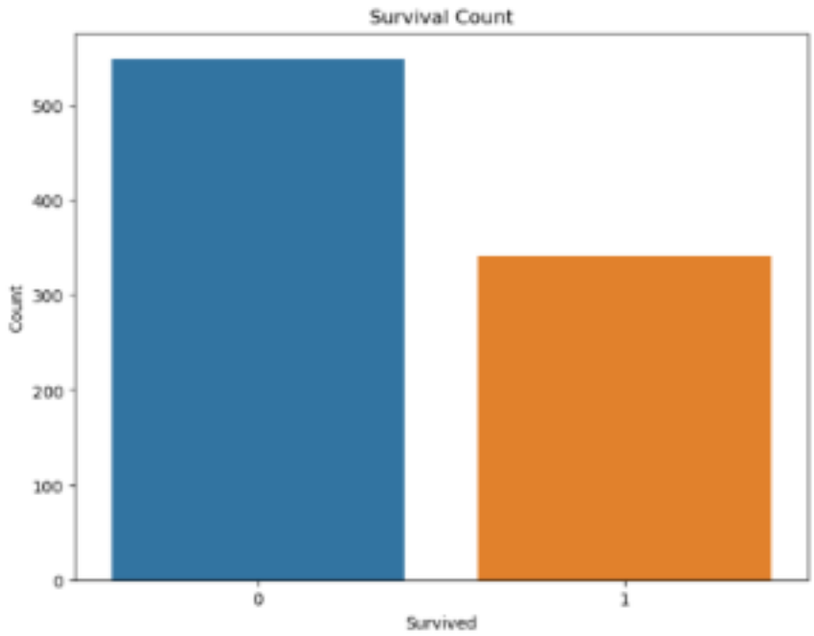
```
import matplotlib.pyplot as plt
import seaborn as sns

# Histogram and Density Plot
plt.figure(figsize=(8, 6))
data['Age'].plot(kind='hist', bins=20, edgecolor='black')
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Distribution of Passenger Ages')
plt.show()

# Box Plot
plt.figure(figsize=(8, 6))
sns.boxplot(x='Pclass', y='Fare', data=data)
plt.xlabel('Passenger Class')
plt.ylabel('Fare')
plt.title('Fare by Passenger Class')
plt.show()
```

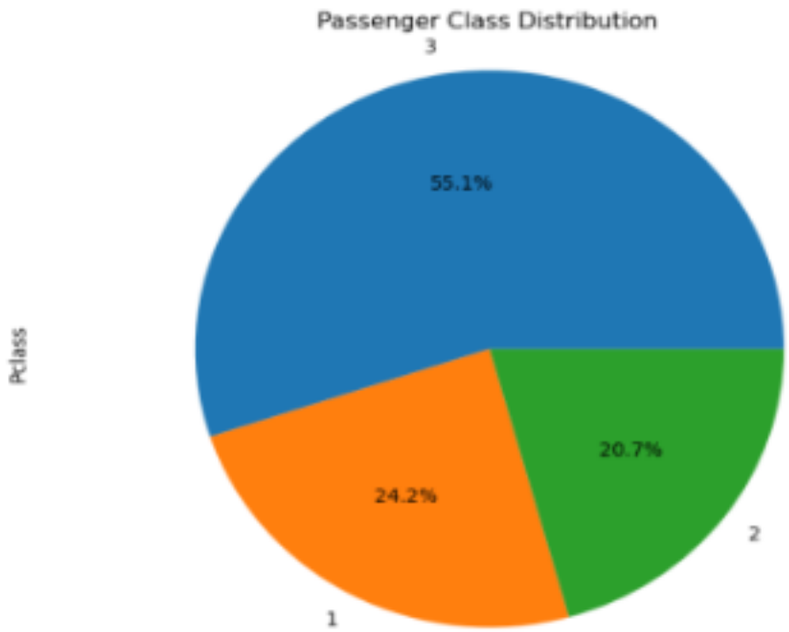
```
# Scatter Plot
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Age', y='Fare', data=data)
plt.xlabel('Age')
plt.ylabel('Fare')
plt.title('Fare vs Age')
plt.show()

# Bar Chart
plt.figure(figsize=(8, 6))
sns.countplot(x='Survived', data=data)
plt.xlabel('Survived')
plt.ylabel('Count')
plt.title('Survival Count')
plt.show()


# Pie Chart
plt.figure(figsize=(8, 6))
data['Pclass'].value_counts().plot.pie(autopct='%1.1f%%')
plt.title('Passenger Class Distribution')
plt.axis('equal')
plt.show()

# Heatmap
plt.figure(figsize=(8, 6))
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True,
cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

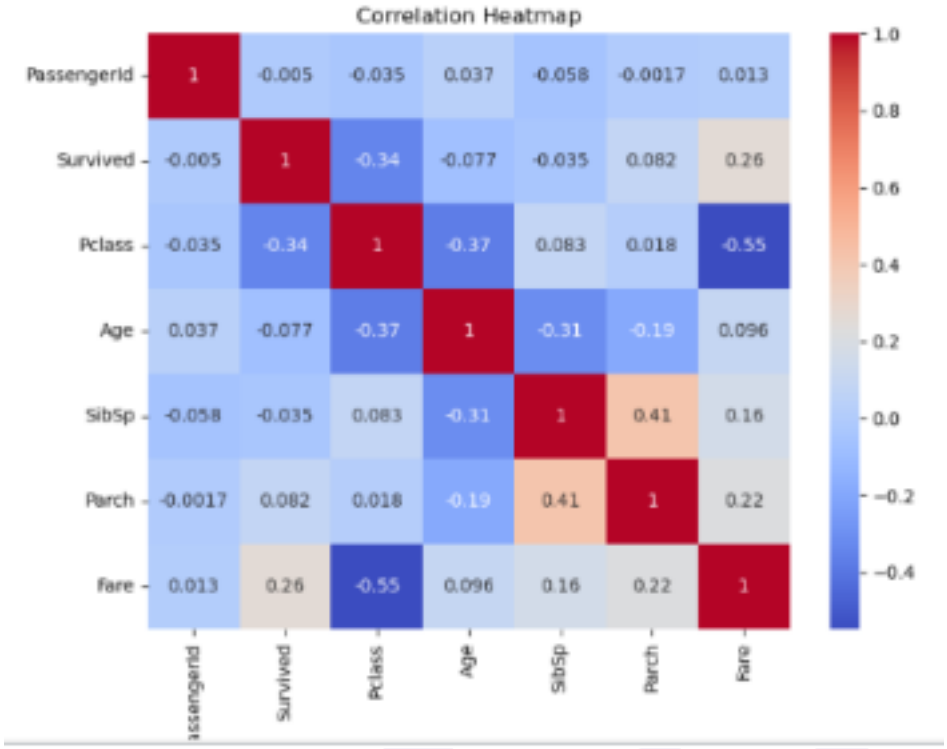





Fare vs Age
Fare
Age
0
100
200
300
400
500
0
10
20
30
40
50
60
70
80


Survival Count
Count
Survived
0
100
200
300
400
500
0
1


Passenger Class Distribution
Pclass
3
55.1%
20.7%
24.2%
2
1



Insights

Histogram and Density Plot: The age distribution is slightly right-skewed, with a higher concentration of passengers in the younger age groups.The modal age group appears to be around 24 years old, with a significant number of passengers in this age range.

Box Plot: Passengers in higher classes (1 and 2) had a higher median fare compared to those in lower classes (3).The box plot helps identify potential outliers in the fare data.

Scatter Plot: The scatter plot shows the relationship between age and fare, revealing a slight positive correlation.Older passengers tend to have higher fares, but there are also some younger passengers with high fares.

Bar Chart: The bar chart displays the count of passengers who survived and those who did not.It provides a quick visual comparison of the survival rates.

Pie Chart: The pie chart illustrates the distribution of passengers across different passenger classes.It shows that the majority of passengers were in class 3, followed by classes 2 and 1.

Heatmap: The heatmap visualizes the correlation matrix, showing the strength and direction of relationships between variables.It helps identify variables that are highly correlated, which can be useful for feature selection or identifying potential multicollinearity issues.