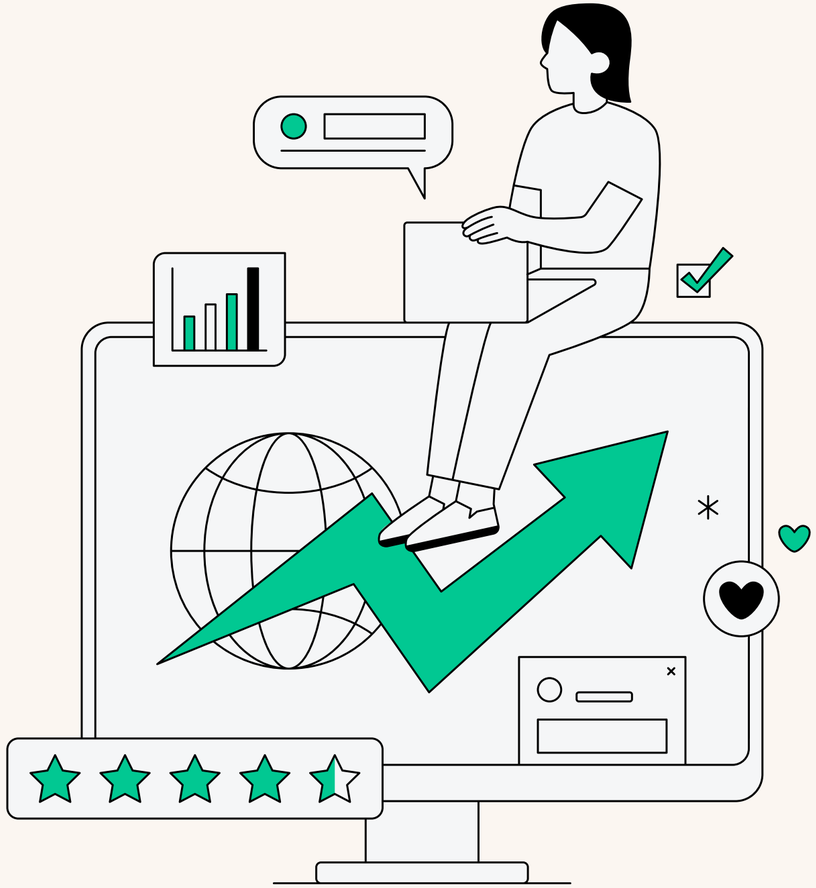
Data Science - Data Analyst

Exploratory Data Analysis

on Titanic Dataset



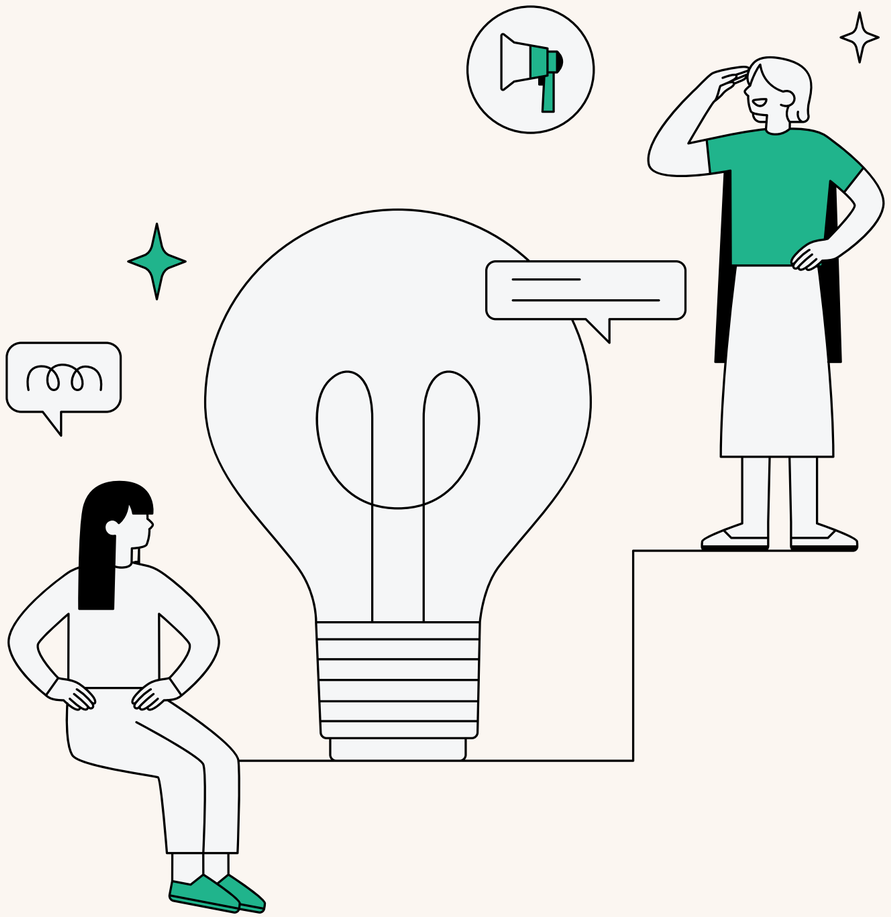
DIGITAL SKILL FAIR 38.0 - Dibimbing.id

About me

I am **Rizal Mattovani**, a 6th-semester Mathematics student at Universitas Airlangga with a strong analytical mindset and a keen interest in Data Science. With a background in mathematical modeling and programming, I aim to apply my skills in data processing, database management, and predictive analytics to solve real-world problems.

Dataset

This dataset contains information about the **passengers aboard the Titanic**, consisting of **500 rows and 4 columns**. The main objective of this dataset is to analyze survival patterns based on passenger characteristics through **Exploratory Data Analysis (EDA).**



Attributes Overview:

- **Survived**
  Indicates whether the passenger survived the Titanic disaster.
  1 = Survived
  0 = Did not survive
- **Name**
  The full name of the passenger.
- **Age**
  The age of the passenger (in years).
- **Sex**
  The gender of the passenger (male or female).

Goal of The Analysis

01.

To understand the basic structure, characteristics, and quality of the Titanic dataset through exploration and cleaning.

02.

To summarize the statistical properties and visualize the distribution of key variables.

03.

To examine the relationship between age and survived to identify potential patterns.

Data Science - Data Analyst

Result & Discussion

DIGITAL SKILL FAIR 38.0 - Dibimbing.id

Import Libraries and Dataset

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", None)
```

```
# import dataset
df = pd.read_excel('/content/drive/MyDrive/Colab Notebooks/titanic.xlsx')
data = df.copy()
```

Showing Top 5 Rows of Data

```
data.head()
```

	survived	name	sex	age
0	1	Allen, Miss. Elisabeth Walton	female	29.0000
1	1	Allison, Master. Hudson Trevor	male	0.9167
2	0	Allison, Miss. Helen Loraine	female	2.0000
3	0	Allison, Mr. Hudson Joshua Creighton	male	30.0000
4	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000

data.head()

```
data.tail()
```

	survived	name	sex	age
495	1	Mallet, Mrs. Albert (Antoinette Magnin)	female	24.0
496	0	Mangiavacchi, Mr. Serafino Emilio	male	NaN
497	0	Matthews, Mr. William John	male	30.0
498	0	Maybery, Mr. Frank Hubert	male	40.0
499	0	McCrae, Mr. Arthur Gordon	male	32.0

data.tail()

Dataset Overview: Structure and Data Types

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   survived  500 non-null    int64
 1   name      500 non-null    object
 2   sex       500 non-null    object
 3   age       451 non-null    float64
dtypes: float64(1), int64(1), object(2)
memory usage: 15.8+ KB
```

Observation:

1. The `survived` and `age` columns are **numeric**.
2. The `name` and `sex` columns are **string** (text).
3. The `survived` column is apparently binary (0 or 1).
4. The `sex` column seems to contain two distinct values (male or female).
5. No obvious defects in the data (column names match their entries); everything looks good.
6. The dataset **contains 4 columns and 500 rows.**
7. Most of the **missing values** appear in the `**age**` **column**; this will be handled later.
8. All data types seem appropriate for their respective columns.

Duplicate Handling

```
# Check for duplicate rows
duplicate_rows = data[data.duplicated()]
print("Number of duplicate rows:", duplicate_rows.shape[0])

# Display duplicate rows (if any)
print("Duplicate records found:\n", duplicate_rows)
```

```
Number of duplicate rows: 1
Duplicate records found:
      survived                              name     sex   age
349         1  Eustis, Miss. Elizabeth Mussey  female  54.0
```

Display duplicate rows (if any)

The output indicates that the dataset **contains 1 duplicate** row.

The duplicated record is:

- Passenger Name: Eustis, Miss. Elizabeth Mussey
- Sex: Female
- Age: 54.0
- Survived: 1

Remove duplicate rows

```
# Remove duplicate rows
data_cleaned = data.drop_duplicates()

# Check the number of rows after removing duplicates
print("Number of rows after removing duplicates:", data_cleaned.shape[0])
```

```
Number of rows after removing duplicates: 499
```

Missing Value Handling

```
# Check for missing values in each column
missing_values = data_cleaned.isnull().sum()
print("Missing values in each column:\n", missing_values)
```

```
Missing values in each column:
 survived     0
name          0
sex           0
age          49
dtype: int64
```

Check for missing values in each column

Calculate the percentage of missing values for each column

The output shows the percentage of missing values in each column of the dataset.

- survived, name, and sex columns have no missing values (0.00%).
- The **age column** has approximately **9.82% missing values.**

```
# Calculate the percentage of missing values for each column
missing_percentage = (data_cleaned.isnull().sum() / len(data_cleaned)) * 100
print("\nPercentage of missing values per column:\n", missing_percentage)
```

```
Percentage of missing values per column:
 survived    0.000000
name         0.000000
sex          0.000000
age          9.819639
dtype: float64
```

Missing Value Handling

```python
# Handling missing values (below 20%)
for column in data_cleaned.columns:
    if missing_percentage[column] > 0 and missing_percentage[column] < 20:
        if data_cleaned[column].dtype == 'object':
            # If column is categorical, fill missing values with the mode
            data_cleaned[column].fillna(data_cleaned[column].mode()[0], inplace=True)
        else:
            # If column is numerical, fill missing values with the median
            data_cleaned[column].fillna(data_cleaned[column].median(), inplace=True)
```

This code **handles missing values** for columns where the missing rate is **between 0% and 20%**:

- If the column is **categorical** (data type = object), the missing values are filled with the **mode** (most frequent value).
- If the column is **numerical**, the missing values are filled with the **median**.

```python
# Re-check for any remaining missing values
print("\nMissing values after handling:\n", data_cleaned.isnull().sum())
```

```
Missing values after handling:
 survived    0
name         0
sex          0
age          0
dtype: int64
```

Re-check for any remaining missing values

Statistical Summary

A **statistical summary** provides a general overview of the dataset by describing its key characteristics. It helps identify patterns, spot anomalies, and understand the distribution and relationships within the data before moving on to deeper analysis or modeling

In this step, we focus on:

- **Descriptive Statistics**

Summarizing the central tendency and spread of numerical data using metrics like mean, median, standard deviation, minimum, and maximum.

- **Distribution Analysis**

Understanding how the data values are distributed, including skewness, outliers, and range, often visualized with histograms or boxplots.

- **Correlation Analysis**

Measuring and visualizing the relationships between numerical variables to identify possible dependencies or trends.

Statistical Summary

```
data_cleaned.columns
```

```
Index(['survived', 'name', 'sex', 'age'], dtype='object')
```

```
# group column names based on type
categoricals = ['name', 'sex']
numericals = ['survived', 'age']
```

Grouping Columns by Data Type

Descriptive Summary of Categorical Features

1. The name column contains **499 entries**, with 499 unique values
2. The sex column also has 499 entries, but only 2 unique categories (**male and female**), which is expected for gender data.
3. The **most common category in the sex** column is **male** with a total frequency of 288.

```
# Syntax categorical statistical summary
data_cleaned[categoricals].describe()
```

	name	sex
count	499	499
unique	499	2
top	McCrae, Mr. Arthur Gordon	male
freq	1	288

Statistical Summary

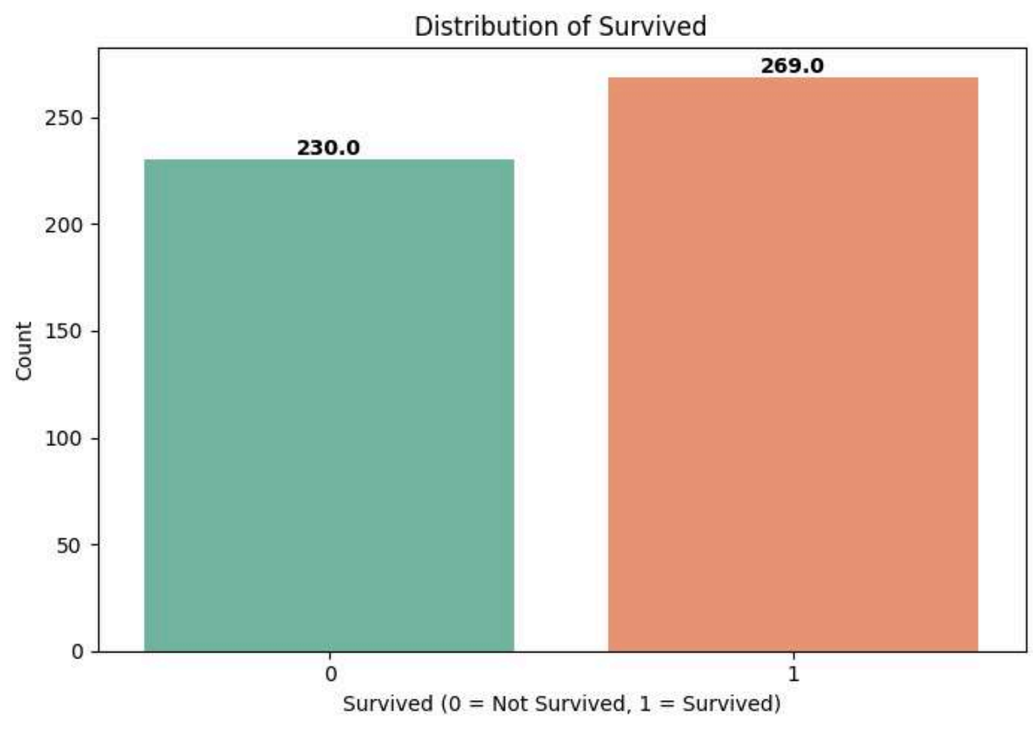
```
# Syntax numerical statistical summary
data_cleaned[numericals].describe()
```

	survived	age
count	499.000000	499.000000
mean	0.539078	35.791416
std	0.498971	14.015770
min	0.000000	0.666700
25%	0.000000	25.500000
50%	1.000000	35.000000
75%	1.000000	45.000000
max	1.000000	80.000000

Descriptive Summary of Numerical Features

1. Overall, the **minimum** and **maximum** values make sense for each column.
2. The age column has a reasonable range (min = 0.6667, max = 80.0). The median (35.0) is quite close to the mean (35.79), suggesting the **age distribution** is likely close to symmetrical but may have some slight skew due to outliers at the lower end (age 0.6667).
3. The age spread (**standard deviation** 14.02) shows a fairly wide range, which is typical for demographic data like age.

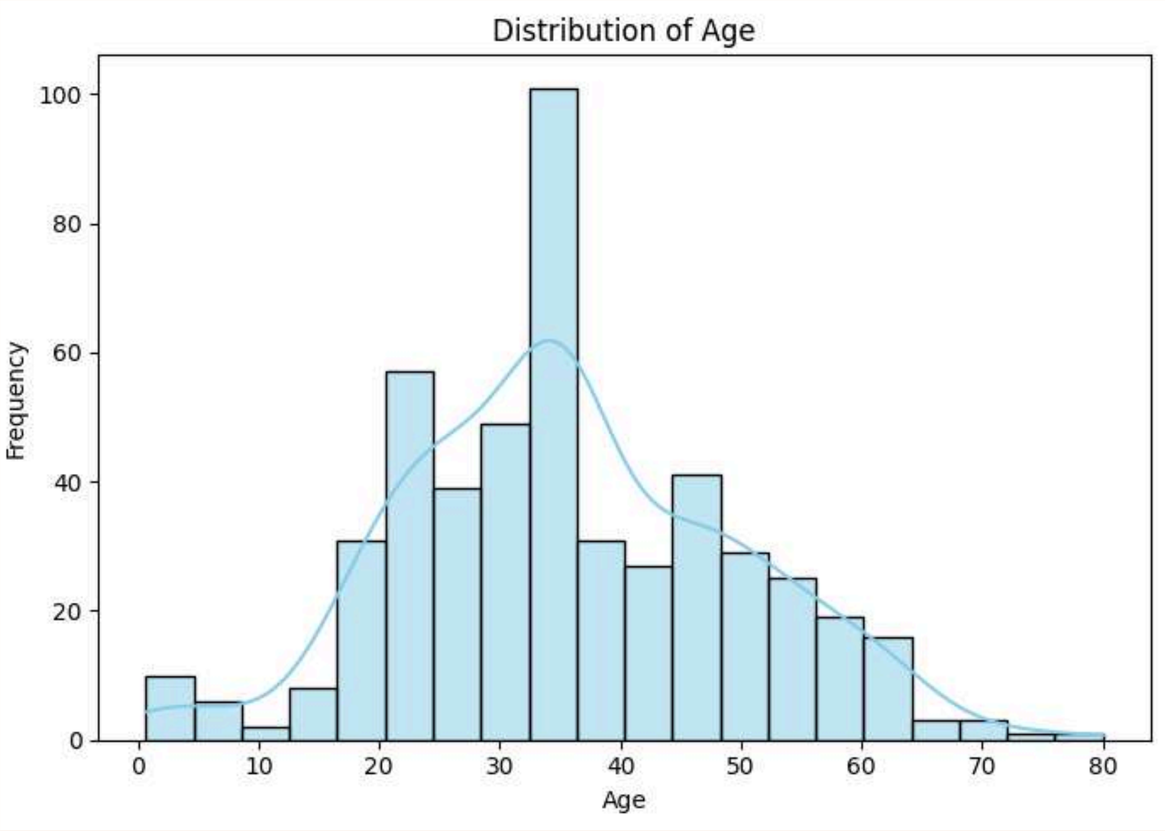
Survival Distribution



The bar chart shows the distribution of the survived column.

- A total of **269 passengers** survived (label 1).
- A total of **230 passengers** did not survive (label 0).

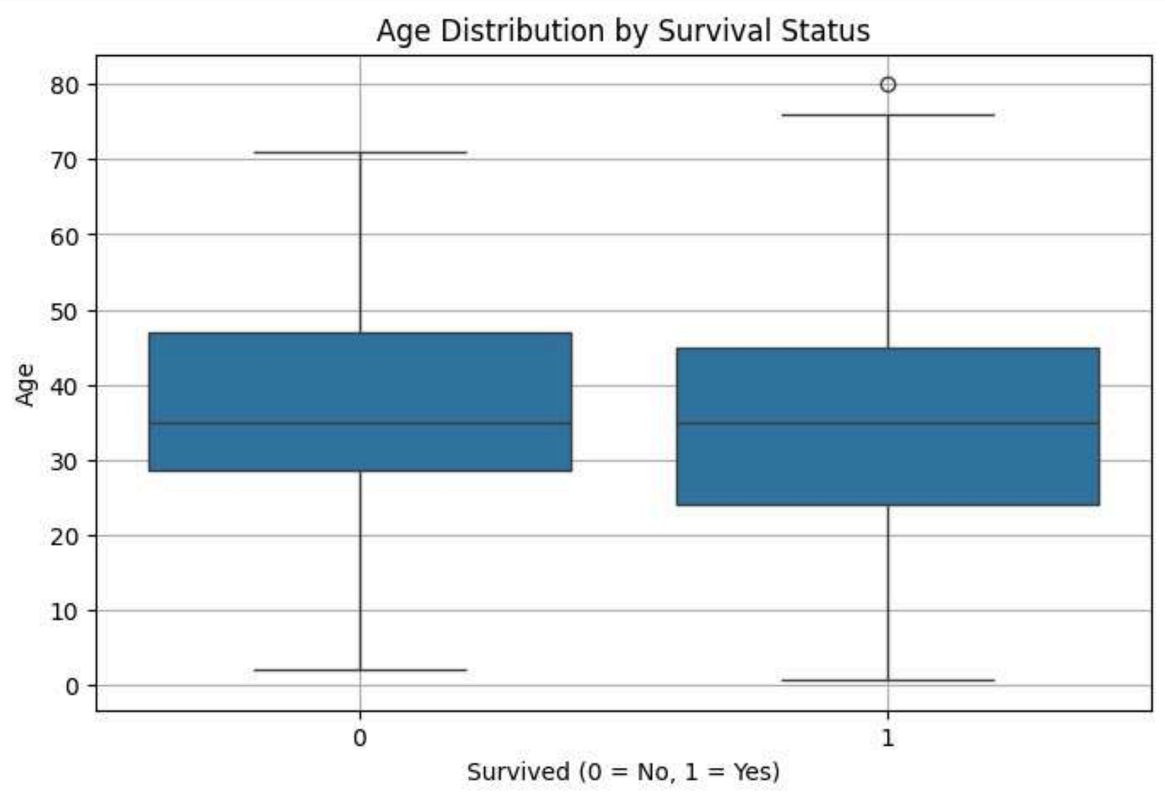
Age Distribution



The **histogram** shows the distribution of Age among passengers.

- The **majority** of passengers are between **20 to 40 years old**, with the peak around the 25–35 age range.
- **The distribution is right-skewed**, meaning there are more younger passengers compared to older ones.
- Very few passengers were above 70 years old or below 5 years old.
- The smooth blue curve (KDE plot) shows the overall age trend, which confirms the central concentration in the young adult group.

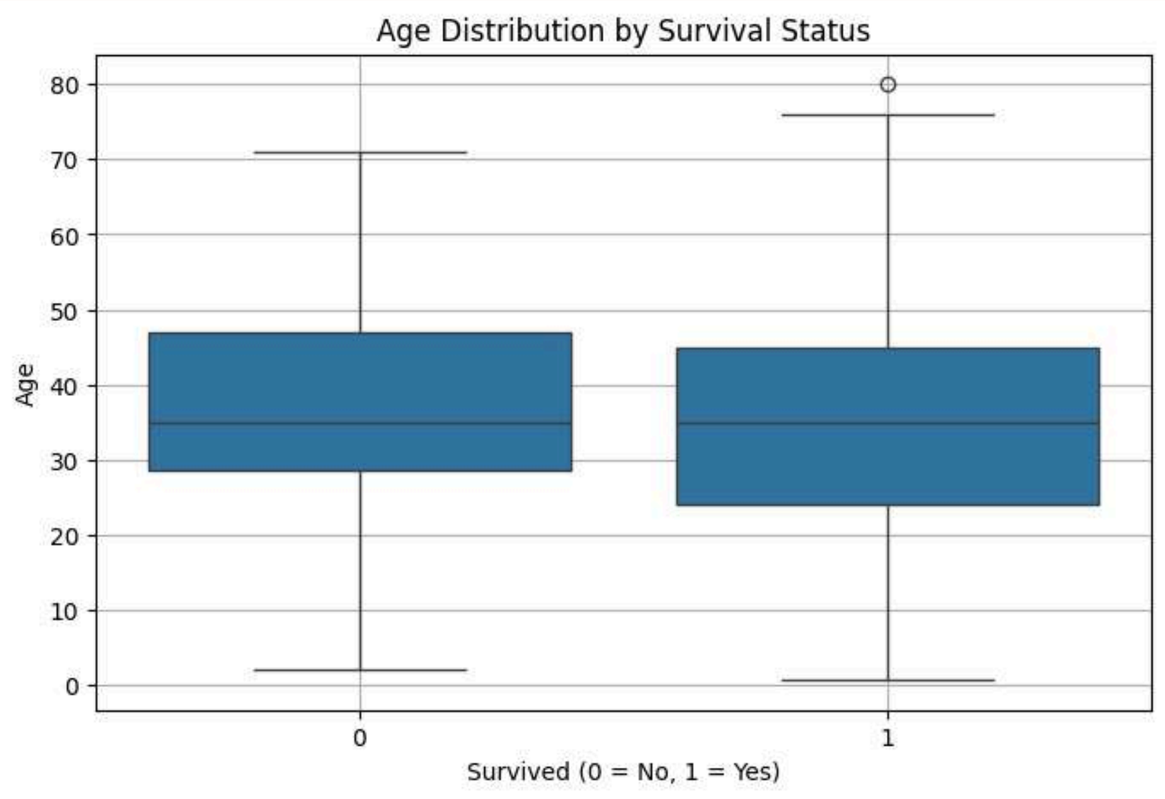
Age Distribution by Survival



Boxplot Analysis:

- The **median** age of passengers who survived (Survived = 1) and those who did not (Survived = 0) is quite **similar**, around 30–35 years.
- The age range for both groups is wide, from infants up to 80 years old (**80 is identified as an outlier** in the Survived = 1 group).
- Passengers who **survived** tend to be slightly **younger** compared to those who did not survive.

Age Distribution by Survival



Boxplot Analysis:

- The **median** age of passengers who survived (Survived = 1) and those who did not (Survived = 0) is quite **similar**, around 30–35 years.
- The age range for both groups is wide, from infants up to 80 years old (**80 is identified as an outlier** in the Survived = 1 group).
- Passengers who **survived** tend to be slightly **younger** compared to those who did not survive.

Correlation Between Age and Survival

```
# Correlation Between Age and Survival
from scipy.stats import pointbiserialr
corr, p_value = pointbiserialr(data_cleaned['survived'], data_cleaned['age'])
print("Correlation coefficient:", corr)
print("P-value:", p_value)
```

```
Correlation coefficient: -0.13939423233698792
P-value: 0.0018007061085229116
```

This analysis examines the relationship between age (numeric) and survival (binary). **Point Biserial Correlation** is used as it's suitable for one **numeric** and one **binary variable.**

Correlation Coefficient: -0.139

- Indicates a **weak negative correlation** between Age and Survival.
- As age increases, the likelihood of survival slightly decreases.

P-value: 0.0018

- Since the p-value is less than 0.05, the result is statistically significant.
- This means the relationship is not due to chance, and age does have a meaningful impact on survival.

DIGITAL SKILL FAIR 38.0 - Dibimbing.id

Thank you very much!

@_rizalmv

www.linkedin.com/in/rizal-mattovani