# TITANIC DATA SCIENCE

Exploratory Data Analysis

- By Allexander Agung from SMK Telkom Purokwerto

# INTRODUCTION

Hello everyone!

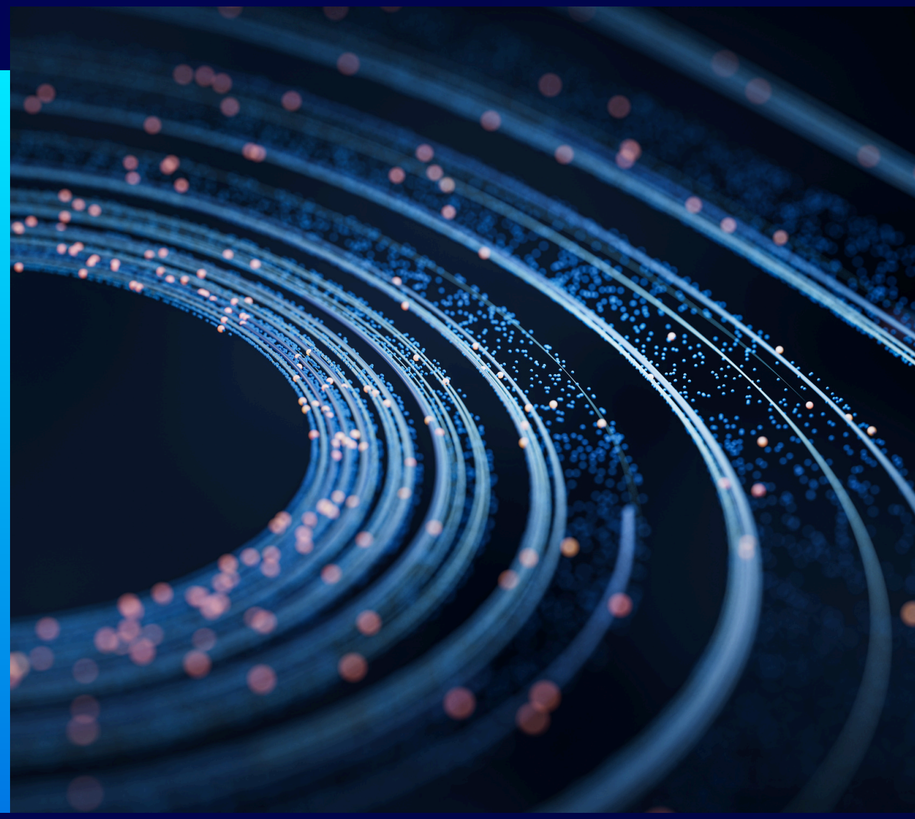
My name is Allexander Agung, and I'm a student from SMK Telkom Purwokerto, majoring in Software and Game Development (PPLG). I chose this field because I'm truly passionate about technology, especially in areas such as programming, application development, and game creation.

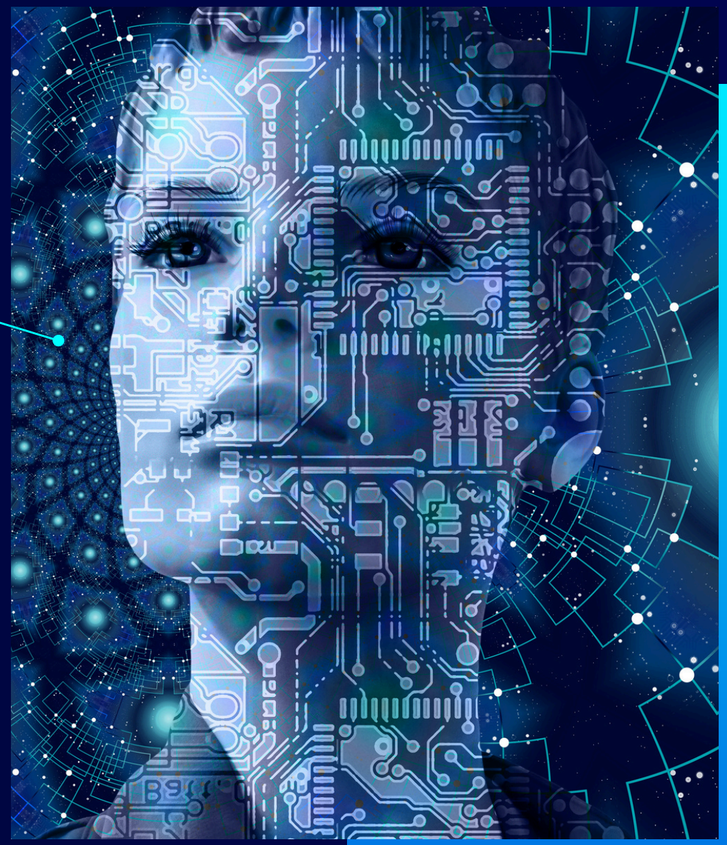
I believe that in today's digital era, having strong skills in software development is incredibly valuable. Through the PPLG program, I aim to sharpen my coding abilities, improve my application design skills, and gain a deeper understanding of the full software development process.

In the future, I aspire to become a developer who creates meaningful digital solutions and keeps learning as technology continues to evolve.

# TABLE OF CONTENT







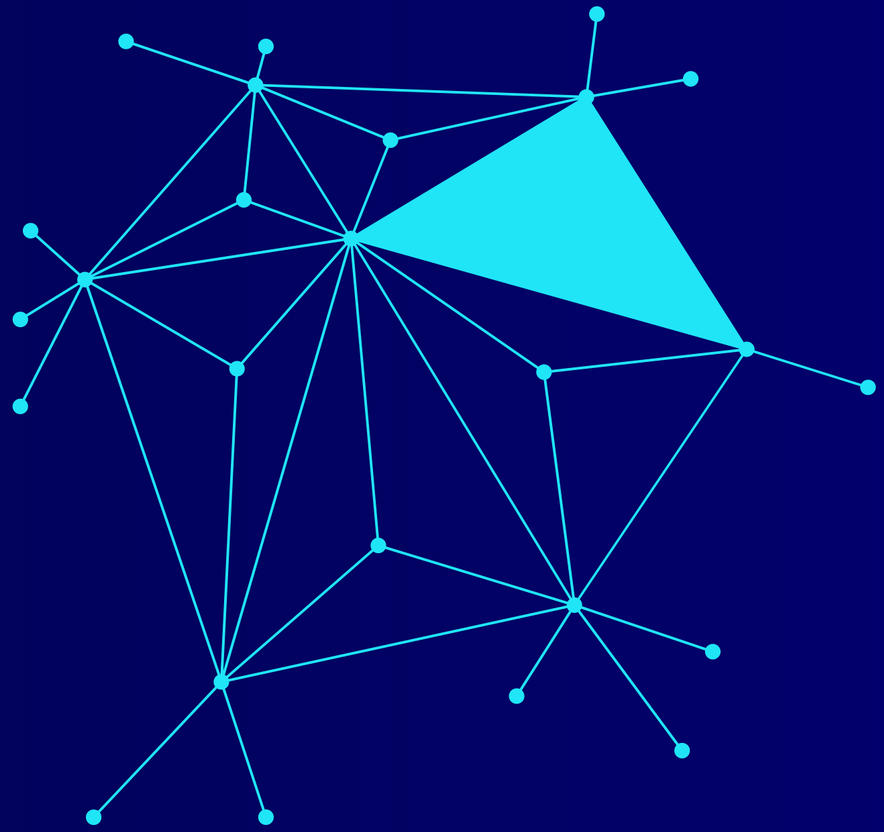
# WHAT IS EDA?

Exploratory Data Analysis (EDA) is the process of exploring and understanding data using statistics and visual tools to uncover patterns, detect outliers, find missing values, and reveal relationships between variables. It helps analysts gain insights and prepare the data for further analysis or modeling.

# PORTOFOLIO

In this portfolio, I present an Exploratory Data Analysis (EDA) of the well-known Titanic dataset, which is widely used as a benchmark in the data science community.

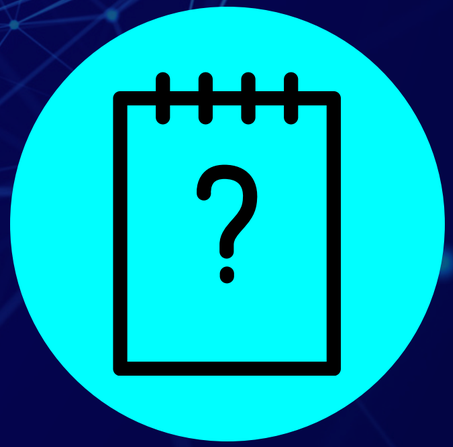
This analysis primarily aims to explore the dataset's structure, address missing values, and eliminate duplicate entries that may affect the reliability and accuracy of the insights derived



# GOAL

**Data Observation**

**Handling Duplicate Data**



**Handling Missing Values**

# LOAD DATA

At the beginning of the analysis, the Titanic dataset was loaded and initially explored using functions such as head(), tail(), and sample() to get a quick overview of the data structure.
Some of the key columns in the dataset include:

- **survived**: Indicates survival status (0 = did not survive, 1 = survived)
- **name**: The full name of the passenger
- **sex**: Gender of the passenger (male or female)
- **age**: Age of the passenger, represented in decimal format

```
# Import data
df = pd.read_excel('/content/drive/MyDrive/titanic.xlsx')
df.head() # Import data head [ Menampilan 5 data teratas dari 'titanic.xlsx' ]
```

| | 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 |

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
#digunakan untuk melihat 5 baris terakhir dari sebuah DataFrame bernama df
df.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 |

```
#digunakan untuk mengambil 5 baris acak dari DataFrame df.
df.sample(5)
```

| | survived | name | sex | age |
|---|---|---|---|---|
| 169 | 0 | Isham, Miss. Ann Elizabeth | female | 50.0 |
| 166 | 0 | Hoyt, Mr. William Fisher | male | NaN |
| 40 | 0 | Brewe, Dr. Arthur Jackson | male | NaN |
| 322 | 1 | Young, Miss. Marie Grice | female | 36.0 |
| 30 | 0 | Blackwell, Mr. Stephen Weart | male | 45.0 |

# LOAD DATA

```
#igunakan untuk menampilkan ringkasan struktur DataFrame df.
df.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
```

1. DataFrame ini punya 500 baris (dari indeks 0 sampai 499).
2. Ada 4 kolom utama, yaitu: survived, name, sex, dan age.
3. Kolom age hanya memiliki 451 nilai non-null, artinya ada 49 data yang hilang (kosong). Kolom lainnya (survived, name, sex) lengkap 500/500.

Using the df.info() function, we can understand the basic structure of the Titanic dataset. The dataset consists of 500 rows and 4 columns: survived, name, sex, and age.

Key Observations:

- All columns except age contain complete data, with 500 non-null entries.
- The age column has only 451 non-null values, indicating that 49 entries are missing.

Data Types of Each Column:

- **survived**: integer (int64)
- **name and sex**: text/string (object)
- **age**: decimal number (float64)

# STATISTICAL SUMMARY

Using the describe() function, we can generate summary statistics for both numerical and categorical columns in the Titanic dataset.

Key Observations (Numerical Columns):

The numerical columns include survived and age.

- The survived column shows binary values (0 or 1), with a mean of 0.366, meaning that approximately 36.6% of passengers survived.
- The age column has 451 non-null values, which means there are 49 missing entries.
- The average age of passengers is around 30.37 years, with ages ranging from 0.17 to 80.0.

Key Observations (Categorical Columns):

The categorical columns include name and sex.

- The name column contains 500 entries, with 499 unique values, indicating one duplicate name.
- The sex column contains two categories: male and female, with male being the most frequent (288 entries).

```
[46] #digunakan untuk menampilkan statistik deskriptif dari kolom-kolom numerik yang telah kamu kelompokkan sebelumnya dalam list numericals.
     df[numericals].describe()
```

| | survived | age |
|---|---|---|
| count | 500.000000 | 451.000000 |
| mean | 0.540000 | 35.917775 |
| std | 0.498897 | 14.766454 |
| min | 0.000000 | 0.666700 |
| 25% | 0.000000 | 24.000000 |
| 50% | 1.000000 | 35.000000 |
| 75% | 1.000000 | 47.000000 |
| max | 1.000000 | 80.000000 |

```
#Untuk menampilkan statistik deskriptif kolom kategorikal, yaitu kolom dalam list categoricals
df[categoricals].describe()
```

| | name | sex |
|---|---|---|
| count | 500 | 500 |
| unique | 499 | 2 |
| top | Eustis, Miss. Elizabeth Mussey | male |
| freq | 2 | 288 |

```
len(df.drop_duplicates()) / len(df)#Menghitung proporsi data yang unik dalam DataFrame df dengan membandingkan jumlah baris unik terhadap jumlah total baris.
```

```
0.998
```

```
# Hitung frekuensi kemunculan tiap baris duplikat
duplicate_counts = duplicates.groupby(list(df.columns)).size().reset_index(name='jumlah_duplikat')

# Urutkan berdasarkan jumlah duplikat
sorted_duplicates = duplicate_counts.sort_values(by='jumlah_duplikat', ascending=False)

# Tampilkan hasil
print("Baris duplikat yang sudah diurutkan berdasarkan jumlah kemunculannya:")
sorted_duplicates
```

```
Baris duplikat yang sudah diurutkan berdasarkan jumlah kemunculannya:
```

| | survived | name | sex | age | jumlah_duplikat |
|---|---|---|---|---|---|
| 0 | 1 | Eustis, Miss. Elizabeth Mussey | female | 54.0 | 2 |

```
[60] df = df.drop_duplicates()#digunakan untuk menghapus semua baris duplikat dalam DataFrame df.
```

```
len(df.drop_duplicates()) / len(df)#Jika output dari code di cell ini tidak bernilai 1 maka terdapat duplikat
```

```
1.0
```

# DATA DUPLICATION ANALYSIS AND HANDLING

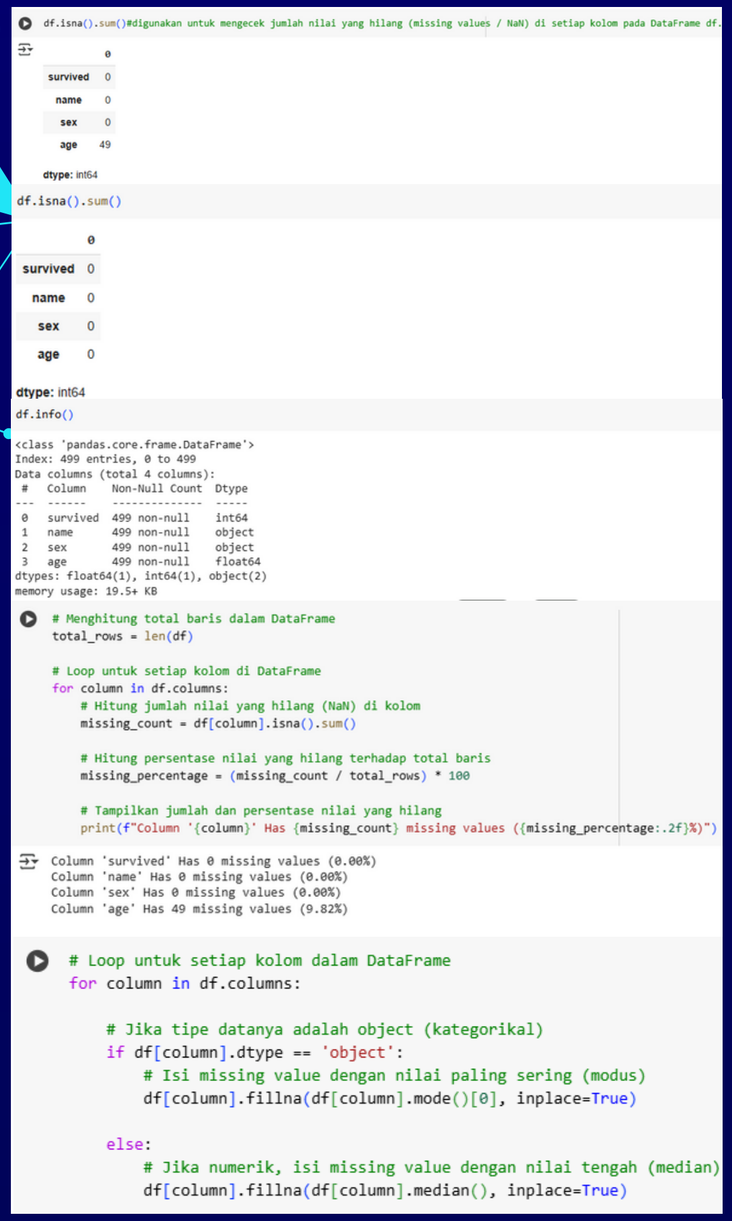
Findings:

- Before cleaning, about 99.8% of the data is unique, indicating the presence of 1 duplicate row.
- The detected duplicate row belongs to Eustis, Miss. Elizabeth Mussey with age 54.0 years, sex female, and survived = 1.
- The row appears 2 identical times in the entire dataset.

Action:

- Duplicates were removed using df.drop_duplicates().
- After cleaning, the proportion of unique data is 100%, indicating all rows are now unique.

# HANDLING MISSING VALUE



```
df.isna().sum()#digunakan untuk mengecek jumlah nilai yang hilang (missing values / NaN) di setiap kolom pada DataFrame df.
```

```
          0
survived  0
name      0
sex       0
age      49
dtype: int64
```

```
df.isna().sum()
```

```
          0
survived  0
name      0
sex       0
age       0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 499 entries, 0 to 499
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   survived  499 non-null    int64
 1   name      499 non-null    object
 2   sex       499 non-null    object
 3   age       499 non-null    float64
dtypes: float64(1), int64(1), object(2)
memory usage: 19.5+ KB
```

```
# Menghitung total baris dalam DataFrame
total_rows = len(df)

# Loop untuk setiap kolom di DataFrame
for column in df.columns:
    # Hitung jumlah nilai yang hilang (NaN) di kolom
    missing_count = df[column].isna().sum()

    # Hitung persentase nilai yang hilang terhadap total baris
    missing_percentage = (missing_count / total_rows) * 100

    # Tampilkan jumlah dan persentase nilai yang hilang
    print(f"Column '{column}' Has {missing_count} missing values ({missing_percentage:.2f}%)")
```

```
Column 'survived' Has 0 missing values (0.00%)
Column 'name' Has 0 missing values (0.00%)
Column 'sex' Has 0 missing values (0.00%)
Column 'age' Has 49 missing values (9.82%)
```

```
# Loop untuk setiap kolom dalam DataFrame
for column in df.columns:

    # Jika tipe datanya adalah object (kategorikal)
    if df[column].dtype == 'object':
        # Isi missing value dengan nilai paling sering (modus)
        df[column].fillna(df[column].mode()[0], inplace=True)

    else:
        # Jika numerik, isi missing value dengan nilai tengah (median)
        df[column].fillna(df[column].median(), inplace=True)
```

Findings:

- The Titanic dataset consists of 5 columns and 891 rows.
- Upon inspection, two columns contain missing values:
  - age with 177 missing entries (~19.77% of the total data).
  - embarked with 2 missing entries (~0.22% of the total data).

Handling:

- For object-type columns (e.g., name, sex, embarked), missing values are imputed using the mode (most frequent value).
- For numeric-type columns (e.g., age), missing values are filled using the median (middle value).

Final Result:

- After performing imputation, all missing values have been successfully handled, and the dataset contains no null entries.
- This is verified using data.isna().sum(), which returns zero missing values for all columns.

# THANK YOU

www.reallygreatsite.com - hello@reallygreatsite.com