



# Data Science Study Menyelami Tragedi Titanic

By Nayaka Khansa Alana – Digital Skill Fair 38.0

# Table of contents

1

Introduction

2

-

3

Alur Data

4

Kesimpulan



1

# Introduction

You can enter a subtitle here if you need it

# Introduction



- Titanic adalah kapal penumpang legendaris yang tenggelam pada pelayaran perdananya tahun 1912.
- Lebih dari 1.500 dari 2.224 penumpang dan kru kehilangan nyawa.
- Tragedi ini menjadi dataset populer dalam dunia data science karena mencakup informasi demografi dan kondisi penumpang.
- Menjelajahi dan memahami pola dalam data penumpang Titanic.
- Membangun model prediksi kelangsungan hidup menggunakan:
  - **Logistic Regression**
  - **Random Forest Classifier**
- Membandingkan performa model dan mengidentifikasi fitur penting.

## Insight

- **Random Forest** secara keseluruhan lebih akurat dibandingkan **Logistic Regression** untuk dataset ini, tetapi hasil yang lebih mendalam bisa diperoleh dengan melihat **precision, recall, dan F1-score**.
- Visualisasi sangat membantu dalam memudahkan pemahaman dan memberikan **gambaran jelas** tentang bagaimana data dan model berperilaku.
- Proses **pembersihan data** (seperti mengatasi duplikat dan missing values) sangat penting agar model tidak mendapatkan data yang buruk yang dapat merusak hasilnya.

# Tools



## Python

Bahasa pemrograman tingkat tinggi yang serbaguna, termasuk analisis machine learning



## Titanic

Dataset excel Titanic

Google Colaboratory



## G - Colab

Writing, running, dan sharing kode python dalam bentuk cloud



# Alur Data

You can enter a subtitle here if you need it

3

## Import Library



- numpy**: Digunakan untuk operasi numerik dan komputasi ilmiah. Library ini menyediakan struktur data seperti array multidimensi, serta berbagai fungsi matematika dan statistik.
- pandas**: Digunakan untuk manipulasi dan analisis data, terutama dalam bentuk tabel (DataFrame). Library ini memungkinkan kamu untuk memproses data dengan cara yang lebih efisien dan mudah, seperti menyaring, mengelompokkan, dan merubah data.
- seaborn**: Sebuah library visualisasi data yang dibangun di atas matplotlib. Seaborn memudahkan pembuatan grafik statistik yang lebih informatif, dengan tema dan warna yang lebih menarik.
- matplotlib.pyplot**: Digunakan untuk membuat berbagai macam grafik (seperti plot garis, bar chart, histogram, dll). Ini adalah salah satu library visualisasi dasar di Python.



```
[ ] # === IMPORT LIBRARY ===  
  
import numpy as np  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt
```



## Setting Pandas



- **pd.set\_option("display.max\_columns", None):**

Mengatur tampilan agar pandas menampilkan semua kolom dalam DataFrame, tanpa membatasi jumlah kolom yang ditampilkan. Secara default, pandas hanya menampilkan sejumlah kolom tertentu jika jumlah kolom terlalu banyak, namun dengan pengaturan ini, semua kolom akan ditampilkan.

- **pd.set\_option("display.max\_rows", None):**

Mengatur tampilan agar pandas menampilkan semua baris dalam DataFrame, tanpa membatasi jumlah baris yang ditampilkan. Secara default, pandas hanya menampilkan sejumlah baris tertentu untuk menghindari tampilan yang terlalu panjang jika DataFrame memiliki banyak baris. Dengan pengaturan ini, seluruh baris akan ditampilkan.



```
[ ] # Setting tampilan pandas
    pd.set_option("display.max_columns", None)
    pd.set_option("display.max_rows", None)
```

## Load Data



- **file\_path = 'titanic.xlsx'**: Mendefinisikan variabel `file_path` yang berisi path atau lokasi file yang akan dimuat. Dalam hal ini, file yang dimaksud adalah file Excel dengan nama `titanic.xlsx`.

- **df = pd.read\_excel(file\_path)**: Menggunakan fungsi `read_excel()` dari `pandas` untuk membaca file Excel yang terletak di `file_path` dan memuatnya ke dalam sebuah `DataFrame` yang disimpan di variabel `df`. `DataFrame` ini akan berisi data dari file Excel tersebut.

```
[ ] # === LOAD DATA ===  
  
file_path = 'titanic.xlsx'  
df = pd.read_excel(file_path)
```

# Investigasi awal data

```
[ ] # === INVESTIGASI AWAL DATA ===  
    print("=== Head ===")  
    print(df.head())
```

```
=== 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
```

• **print("=== Head ===")**: Menampilkan teks "=== Head ===" di konsol untuk memberi tanda atau informasi bahwa bagian berikutnya adalah tampilan dari data teratas (head) dari DataFrame.

• **print(df.head())**: Fungsi head() digunakan untuk menampilkan lima baris pertama dari DataFrame df. Fungsi ini sangat berguna untuk melakukan inspeksi awal terhadap data yang dimuat, memungkinkan kamu untuk melihat sebagian kecil data dan memastikan bahwa data sudah diimpor dengan benar.

# Tail



```
print("\n=== Tail ===")  
print(df.tail())
```



```
=== 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
```

**1.print("\n=== Tail ==="):** Menampilkan teks "=== Tail ===" di konsol dengan sebuah baris kosong sebelumnya (karena \n), yang menandakan bahwa bagian berikutnya akan menampilkan data terendah (tail) dari DataFrame.

**2.print(df.tail()):** Fungsi tail() digunakan untuk menampilkan lima baris terakhir dari DataFrame df. Seperti halnya head(), tail() membantu dalam inspeksi awal dengan menunjukkan data paling bawah pada dataset, yang berguna untuk memverifikasi apakah data sudah lengkap dan sesuai.

# Sample

```
[ ] print("\n=== Sample ===")
    print(df.sample(5, random_state=42))
```

```
=== Sample ===
survived
361      1      Caldwell, Mr. Albert Francis      male  26.0
73       1      Cleaver, Miss. Alice      female  22.0
374      1      Clarke, Mrs. Charles V (Ada Maria Winfield)  female  28.0
155      1      Hays, Mrs. Charles Melville (Clara Jennings Gr...  female  52.0
104      1      Eustis, Miss. Elizabeth Mussey      female  54.0
```

• **print("\n=== Sample ===")**: Menampilkan teks "=== Sample ===" di konsol dengan sebuah baris kosong sebelumnya (karena \n), yang menandakan bahwa bagian berikutnya akan menampilkan sampel acak dari DataFrame.

• **print(df.sample(5, random\_state=42))**: Fungsi `sample()` digunakan untuk memilih sejumlah baris secara acak dari DataFrame `df`. Dalam hal ini, 5 menunjukkan bahwa kita ingin mengambil 5 baris acak. Argumen `random_state=42` memastikan bahwa hasil sampel acak ini akan konsisten (reproducible) setiap kali kode dijalankan, karena nilai `random_state` akan mengontrol generator angka acak.

# Info

```
[ ] print("\n=== Info ===")
    df.info()
```



```
=== 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
```

• **print("\n=== Info ===")**: Menampilkan teks "=== Info ===" di konsol dengan sebuah baris kosong sebelumnya (karena \n), yang menandakan bahwa bagian berikutnya akan menampilkan informasi tentang DataFrame.

• **df.info()**: Fungsi info() digunakan untuk memberikan ringkasan informasi tentang DataFrame df. Informasi yang ditampilkan meliputi:

- Jumlah total baris dan kolom.
- Nama dan tipe data dari setiap kolom.
- Jumlah nilai non-null (tidak kosong) dalam setiap kolom.
- Memori yang digunakan oleh DataFrame.

# Statistical summary

```
[ ] # === STATISTICAL SUMMARY ===  
print("\n=== Statistical Summary ===")  
print(df.describe(include='all'))
```



```
=== Statistical Summary ===  
count    survived          name    sex    age  
500.000000      500      500    451.000000  
unique      NaN          499      2      NaN  
top      NaN  Eustis, Miss. Elizabeth Mussey  male      NaN  
freq      NaN          2      288      NaN  
mean    0.540000      NaN      NaN    35.917775  
std    0.498897      NaN      NaN    14.766454  
min    0.000000      NaN      NaN    0.666700  
25%    0.000000      NaN      NaN    24.000000  
50%    1.000000      NaN      NaN    35.000000  
75%    1.000000      NaN      NaN    47.000000  
max    1.000000      NaN      NaN    80.000000
```

- `print("\n=== Statistical Summary ===")`:

Menampilkan teks "=== Statistical Summary ===" di konsol dengan sebuah baris kosong sebelumnya (karena `\n`), yang menandakan bahwa bagian berikutnya akan menampilkan ringkasan statistik dari DataFrame.

- `print(df.describe(include='all'))`: Fungsi `describe()` digunakan untuk menghasilkan ringkasan statistik dari DataFrame `df`. Dengan argumen `include='all'`, fungsi ini akan memberikan statistik untuk semua kolom, baik yang bersifat numerik maupun non-numerik.

# Cek duplikat

```
[ ] # === CEK DUPLIKAT ===  
    duplikat = df.duplicated().sum()  
    print(f"\nJumlah duplikat: {duplikat}")
```



Jumlah duplikat: 1

- **duplikat = df.duplicated().sum():**
- `df.duplicated()` mengembalikan sebuah Series boolean yang menunjukkan apakah setiap baris dalam DataFrame `df` merupakan duplikat dari baris sebelumnya (nilai True menunjukkan baris tersebut duplikat).
- `.sum()` digunakan untuk menghitung jumlah nilai True, yang berarti jumlah baris duplikat dalam DataFrame.
- **`print(f"\nJumlah duplikat: {duplikat}")`:**
- Menampilkan jumlah baris duplikat dalam DataFrame menggunakan fungsi `print()`. Variabel `duplikat` berisi hasil jumlah duplikat yang dihitung sebelumnya.



# Hapus duplikat

```
[ ] # Hapus duplikat jika ada  
df = df.drop_duplicates()
```



**df = df.drop\_duplicates():**

- Fungsi drop\_duplicates() digunakan untuk menghapus baris-baris duplikat dalam DataFrame df.
- Secara default, fungsi ini akan menghapus baris yang duplikat berdasarkan seluruh kolom. Hanya baris pertama yang akan dipertahankan, sementara baris-baris berikutnya yang identik akan dihapus.
- Hasilnya disimpan kembali ke dalam variabel df, sehingga DataFrame yang baru akan berisi data tanpa duplikat.

# Cek missing

```
[ ] # === CEK MISSING VALUES ===  
    missing = df.isnull().sum()  
    missing_percent = (missing / len(df)) * 100
```



- **missing = df.isnull().sum():**
- **df.isnull()** menghasilkan DataFrame boolean yang menunjukkan apakah setiap elemen dalam DataFrame **df** adalah nilai yang hilang (NaN).
- **.sum()** kemudian digunakan untuk menghitung jumlah nilai yang hilang (NaN) di setiap kolom. Hasilnya adalah sebuah Series yang menunjukkan jumlah nilai yang hilang untuk setiap kolom.
- **missing\_percent = (missing / len(df)) \* 100:**
- **len(df)** memberikan jumlah total baris dalam DataFrame **df**.
- Dengan membagi jumlah nilai yang hilang (**missing**) dengan jumlah total baris dan mengalikannya dengan 100, kita mendapatkan persentase nilai yang hilang (**missing values**) untuk setiap kolom.

# Missing values

```
[ ] print("\n=== Missing Values ===")
    print(missing)
    print("\n=== Persentase Missing ===")
    print(missing_percent)
```



```
=== Missing Values ===
```

```
survived      0
```

```
name          0
```

```
sex           0
```

```
age           49
```

```
dtype: int64
```

```
=== Persentase Missing ===
```

```
survived      0.000000
```

```
name          0.000000
```

```
sex           0.000000
```

```
age           9.819639
```

```
dtype: float64
```

- **print("\n=== Missing Values ===")**: Menampilkan teks "=== Missing Values ===" di konsol, memberi tahu bahwa bagian berikutnya akan menunjukkan jumlah nilai yang hilang (missing values) untuk setiap kolom dalam DataFrame.
- **print(missing)**: Menampilkan hasil dari variabel missing, yang berisi jumlah nilai yang hilang (NaN) di setiap kolom. Ini memberikan gambaran seberapa banyak data yang hilang pada setiap kolom.
- **print("\n=== Persentase Missing ===")**: Menampilkan teks "=== Persentase Missing ===" di konsol, memberi tahu bahwa bagian berikutnya akan menunjukkan persentase nilai yang hilang (missing values) untuk setiap kolom.
- **print(missing\_percent)**: Menampilkan hasil dari variabel missing\_percent, yang berisi persentase nilai yang hilang (NaN) di setiap kolom. Ini memberikan gambaran yang lebih jelas tentang sejauh mana missing values mempengaruhi setiap kolom dalam dataset.

# Tangani missing


```
[ ] # Tangani missing value pada kolom 'age' dengan mean
    df['age'] = df['age'].fillna(df['age'].mean())
```




**df['age'] = df['age'].fillna(df['age'].mean()):**

- df['age']: Merujuk pada kolom age dalam DataFrame df.
- df['age'].mean(): Menghitung nilai rata-rata (mean) dari kolom age, yang akan digunakan untuk menggantikan nilai yang hilang (NaN) di kolom tersebut.
- fillna(): Fungsi ini digunakan untuk mengganti nilai yang hilang (NaN) dalam kolom age dengan nilai yang diberikan, dalam hal ini adalah nilai rata-rata kolom age (df['age'].mean()).

# Visualisasi



```
[ ] # === VISUALISASI ===  
sns.set(style="whitegrid")
```



**sns.set(style="whitegrid"):**

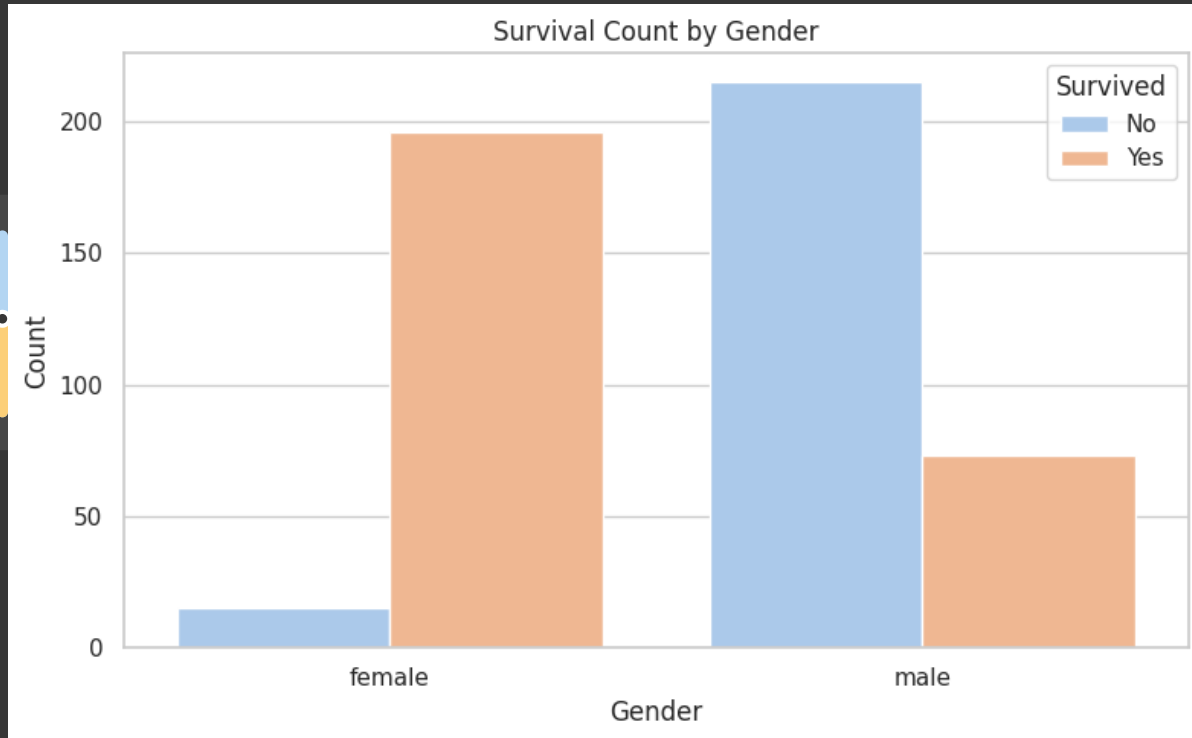
- Fungsi `sns.set()` digunakan untuk mengatur gaya (style) visualisasi yang dihasilkan oleh Seaborn.
- `style="whitegrid"` mengatur agar latar belakang plot berwarna putih dengan grid (garis bantu) yang terlihat. Gaya ini membantu untuk membuat grafik lebih bersih dan mudah dibaca, terutama saat menampilkan data yang memerlukan perbandingan angka.

## Plot 1

```
[ ] # Plot 1: Survival Count by Gender
plt.figure(figsize=(8, 5))
sns.countplot(data=df, x='sex', hue='survived', palette='pastel')
plt.title('Survival Count by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.legend(title='Survived', labels=['No', 'Yes'])
plt.tight_layout()
plt.show()
```

•Plot ini akan memberikan gambaran visual mengenai jumlah orang yang selamat (survived) berdasarkan gender, dengan kategori warna yang menunjukkan apakah seseorang selamat atau tidak.

# Plot 1



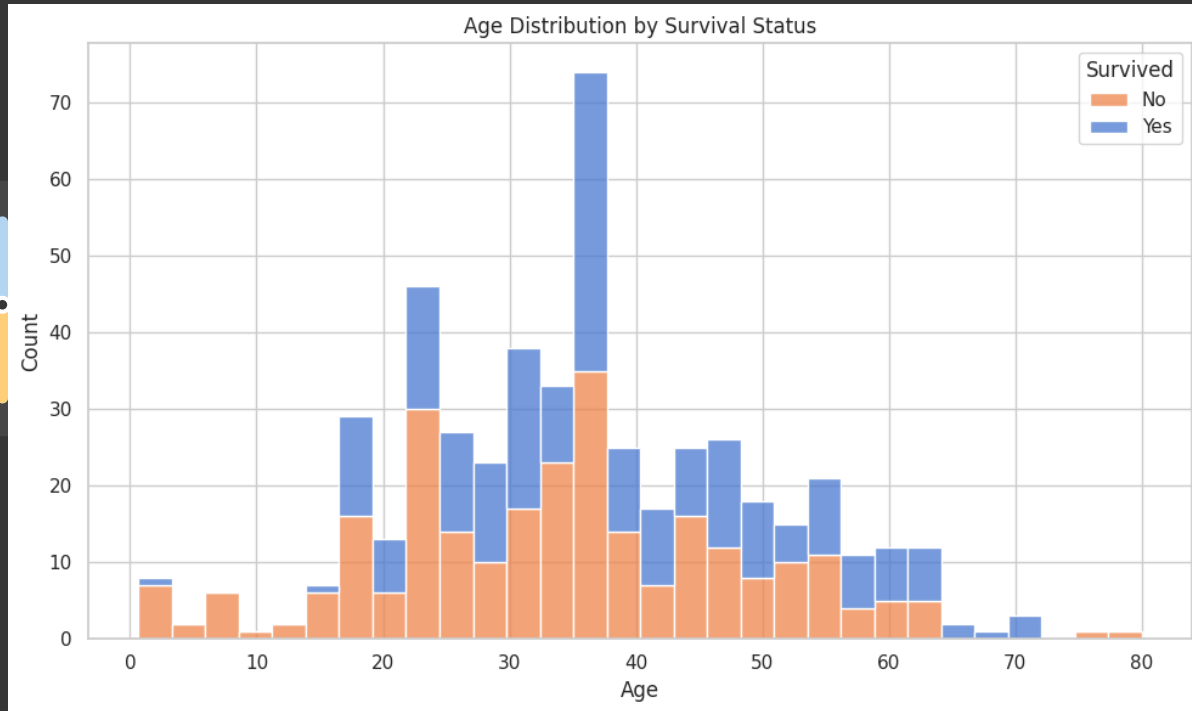
## Plot 2

```
[ ] # Plot 2: Age Distribution by Survival Status
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='age', hue='survived', multiple='stack', bins=30, palette='muted')
plt.title('Age Distribution by Survival Status')
plt.xlabel('Age')
plt.ylabel('Count')
plt.legend(title='Survived', labels=['No', 'Yes'])
plt.tight_layout()
plt.show()
```

- Plot ini menunjukkan distribusi usia berdasarkan status kelangsungan hidup, di mana histogram bertumpuk memberikan informasi tentang jumlah orang yang selamat dan yang tidak selamat pada setiap kelompok usia.



## Plot 2

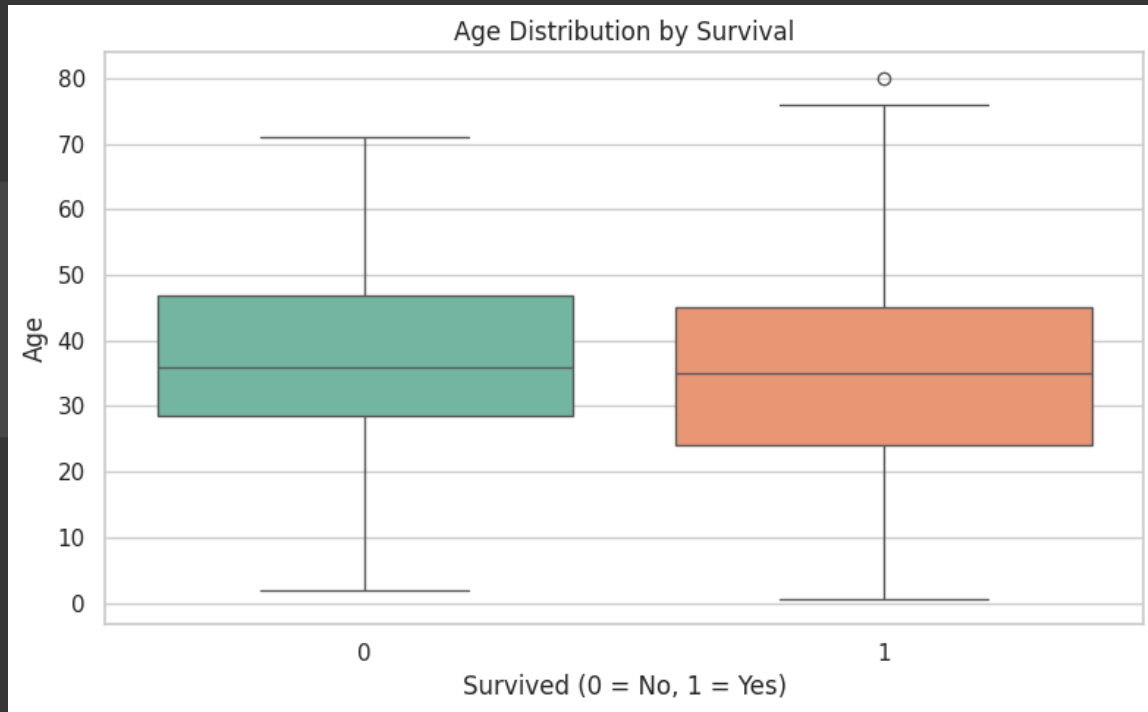


## Plot 3

```
[ ] # Plot 3: Boxplot Age vs Survival
plt.figure(figsize=(8, 5))
sns.boxplot(data=df, x='survived', y='age', palette='Set2')
plt.title('Age Distribution by Survival')
plt.xlabel('Survived (0 = No, 1 = Yes)')
plt.ylabel('Age')
plt.tight_layout()
plt.show()
```

•Plot ini menunjukkan distribusi usia berdasarkan status kelangsungan hidup. Boxplot ini memperlihatkan rentang usia, median, kuartil, dan pencilan bagi orang yang selamat (1) dan yang tidak selamat (0) dalam dataset.

## Plot 3

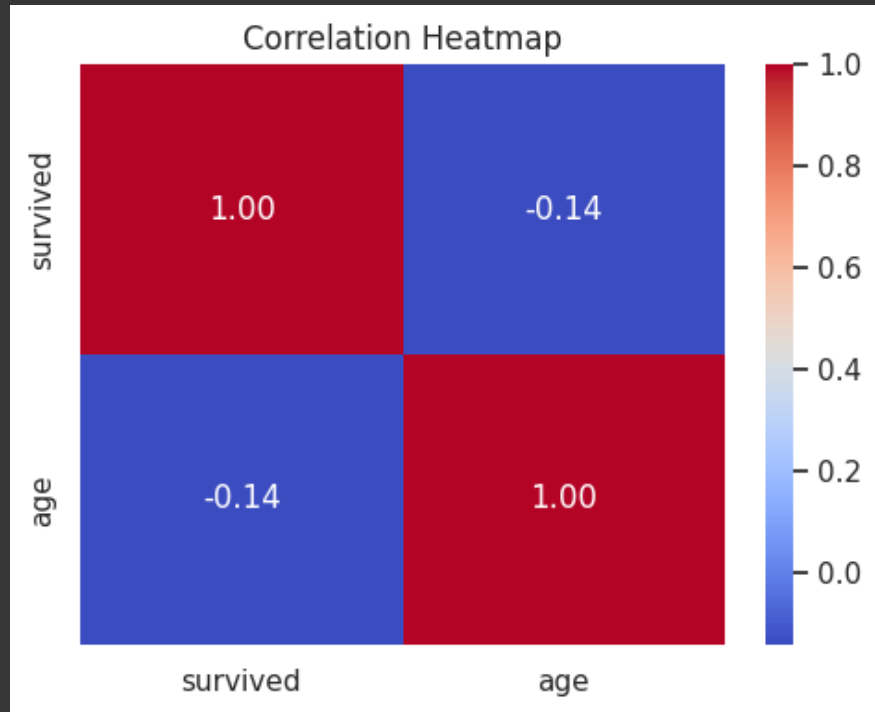


## Plot 4

```
[ ] # Plot 4: Heatmap Korelasi
plt.figure(figsize=(5, 4))
sns.heatmap(df[['survived', 'age']].corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.tight_layout()
plt.show()
```

Plot ini menghasilkan heatmap yang menggambarkan korelasi antara kolom survived dan age. Dengan melihat nilai korelasi, kita dapat mengetahui seberapa kuat hubungan linier antara usia dan kemungkinan seseorang selamat dalam dataset.

## Plot 4



## Plot 5

```
[ ] # Plot 5: Pie Chart Survival Rate
survived_counts = df['survived'].value_counts()
labels = ['Did Not Survive', 'Survived']
colors = ['lightcoral', 'lightgreen']
```

- **survived\_counts = df['survived'].value\_counts():**

- **df['survived']:** Mengambil kolom survived dari DataFrame df, yang menunjukkan apakah seseorang selamat (1) atau tidak selamat (0).

- **value\_counts():** Menghitung jumlah kemunculan setiap nilai unik dalam kolom survived (yaitu, berapa banyak yang selamat dan berapa banyak yang tidak selamat).

- **labels = ['Did Not Survive', 'Survived']:**

- Mendefinisikan label untuk setiap kategori dalam diagram pie chart, dengan label "Did Not Survive" untuk orang yang tidak selamat dan "Survived" untuk orang yang selamat.

- **colors = ['lightcoral', 'lightgreen']:**

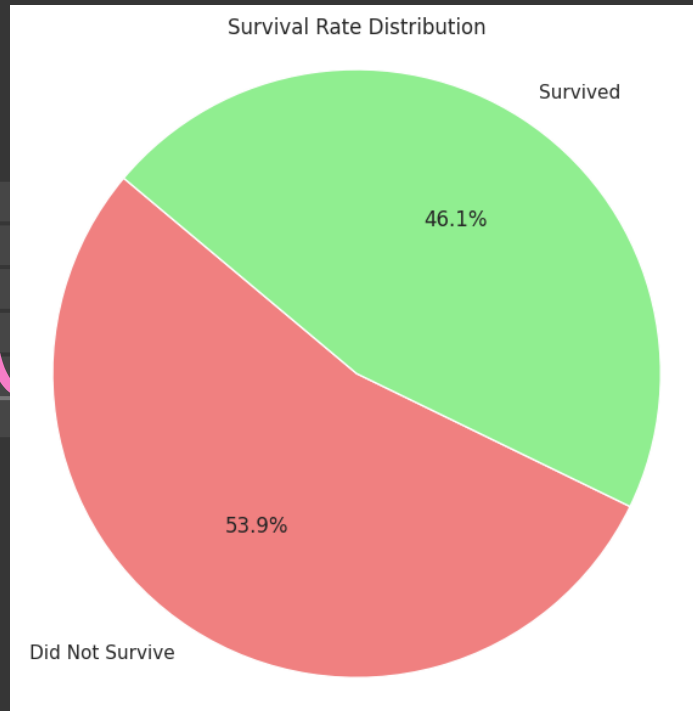
- Mendefinisikan warna untuk setiap kategori dalam pie chart. Warna lightcoral digunakan untuk orang yang tidak selamat, dan warna lightgreen digunakan untuk orang yang selamat.

## Plot 5

```
[ ] plt.figure(figsize=(6, 6))
plt.pie(survived_counts, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
plt.title('Survival Rate Distribution')
plt.axis('equal')
plt.tight_layout()
plt.show()
```

•Pie chart ini memberikan visualisasi yang jelas mengenai proporsi orang yang selamat dan yang tidak selamat dalam dataset, dengan persentase untuk masing-masing kategori.

## Plot 5





# Membuat Model Prediksi



- Digunakan untuk **membagi dataset** menjadi data **latih (train)** dan **uji (test)**.
- Berguna untuk menguji performa model setelah dilatih.
- Mengimpor **model Regresi Logistik** dari Scikit-learn, yaitu model klasifikasi yang sering digunakan untuk prediksi biner (contoh: selamat atau tidak selamat).
- Digunakan untuk **evaluasi model**:
  - `accuracy_score`: Menghitung akurasi prediksi model.
  - `confusion_matrix`: Menampilkan jumlah prediksi benar/salah untuk tiap kelas.
  - `classification_report`: Menyediakan metrik evaluasi seperti precision, recall, dan f1-score.
- Mengimpor `LabelEncoder`, digunakan untuk mengubah data kategorikal (seperti teks) menjadi angka.
- Penting ketika model machine learning hanya bisa memproses data numerik.

```
[23] from sklearn.model_selection import train_test_split
      from sklearn.linear_model import LogisticRegression
      from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
      from sklearn.preprocessing import LabelEncoder
```



# Encode



- Membuat objek LabelEncoder, yaitu alat dari Scikit-learn untuk mengubah data kategorikal menjadi angka.
- Mengubah nilai dalam kolom 'sex' dari **teks** ('male', 'female') menjadi **angka**:
  - 'female' di-encode menjadi 0
  - 'male' di-encode menjadi 1
- Transformasi ini penting karena algoritma machine learning hanya menerima input dalam bentuk numerik.

```
[25] # Encode kolom 'sex'  
le = LabelEncoder()  
df['sex'] = le.fit_transform(df['sex']) # female=0, male=1
```



# Pilih fitur dan target



- Memilih kolom 'sex' dan 'age' sebagai **fitur/input (X)** untuk model machine learning.
- Ini adalah variabel independen yang digunakan untuk memprediksi hasil.
- Menentukan kolom 'survived' sebagai **target/output (y)**, yaitu nilai yang ingin diprediksi oleh model.
- Ini adalah variabel dependen (label), dengan nilai 0 (tidak selamat) atau 1 (selamat).

```
[27] # Pilih fitur dan target  
X = df[['sex', 'age']]  
y = df['survived']
```



# Drop NaN



- Menghapus baris-baris dari X (fitur) yang mengandung nilai **NaN** (**kosong/missing**).
- Machine learning model tidak bisa bekerja dengan data yang mengandung missing values, jadi perlu dibersihkan.
- Menyesuaikan y (target) agar tetap **sinkron** dengan baris-baris X yang tersisa setelah proses dropna().
- Karena setelah menghapus baris dari X, jumlah barisnya bisa berubah, kita perlu memastikan y mencocokkan index X.

```
[28] # Drop NaN dari X dan sesuaikan y
      X = X.dropna()
      y = y[X.index]
```



# Split data



- **train\_test\_split**: Fungsi untuk membagi data menjadi dua bagian — data **latih** dan data **uji**.
- **X dan y**: Data input dan target yang ingin dibagi.
- **test\_size=0.2**: Artinya **20%** data akan digunakan sebagai **data uji**, dan **80%** sisanya sebagai **data latih**.
- **random\_state=42**: Menetapkan "seed" untuk pengacakan agar hasil pembagian selalu **konsisten** setiap kali dijalankan.

```
[29] # Split data ke train dan test  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```



# Training Model



- Membuat objek model **Regresi Logistik**, yaitu algoritma klasifikasi yang digunakan untuk memprediksi probabilitas kejadian (contohnya: selamat atau tidak selamat).
- Cocok untuk masalah klasifikasi biner seperti survived (0 atau 1).
- Melatih (**fit**) model dengan data latih (X\_train sebagai input dan y\_train sebagai target).
- Model akan **mempelajari hubungan** antara fitur (sex, age) dan label (survived), sehingga bisa digunakan untuk prediksi.

```
[30] # === TRAINING MODEL ===  
      model = LogisticRegression()  
      model.fit(X_train, y_train)
```



LogisticRegression ⓘ ?

LogisticRegression()

# Evaluasi



- Menggunakan model yang sudah dilatih untuk **memprediksi** hasil (survived) pada data uji `X_test`.
- Menghitung **akurasi** model, yaitu seberapa banyak prediksi yang benar dibandingkan total prediksi.
- Membuat **confusion matrix**, yang memperlihatkan:
  - True Positives (TP)
  - True Negatives (TN)
  - False Positives (FP)
  - False Negatives (FN)
- Menghasilkan **laporan klasifikasi** lengkap yang mencakup:
  - **Precision**: Seberapa akurat model saat memprediksi kelas tertentu.
  - **Recall**: Seberapa baik model menemukan semua instance dari kelas tersebut.
  - **F1-score**: Rata-rata harmonik dari precision dan recall.
  - **Support**: Jumlah sampel sebenarnya dari setiap kelas.

```
[31] # === EVALUASI ===  
y_pred = model.predict(X_test)  
accuracy = accuracy_score(y_test, y_pred)  
conf_matrix = confusion_matrix(y_test, y_pred)  
report = classification_report(y_test, y_pred)
```



# Hasil Evaluasi



- Menampilkan **akurasi model** dalam format 4 angka desimal.
- Akurasi adalah proporsi prediksi yang benar dibandingkan dengan total data uji.
- Menampilkan **confusion matrix**, yang menjelaskan jumlah:

- **True Negatives (TN)**: Model memprediksi tidak selamat, dan memang tidak selamat.
- **False Positives (FP)**: Model memprediksi selamat, tapi kenyataannya tidak selamat.
- **False Negatives (FN)**: Model memprediksi tidak selamat, tapi kenyataannya selamat.
- **True Positives (TP)**: Model memprediksi selamat, dan memang selamat.

- Menampilkan **ringkasan evaluasi model** berupa:
  - **Precision**: Akurasi model dalam memprediksi suatu kelas.
  - **Recall**: Kemampuan model dalam menemukan semua instance dari suatu kelas.
  - **F1-score**: Rata-rata harmonik antara precision dan recall.
  - **Support**: Jumlah data aktual untuk tiap kelas.

```
[32] # Print hasil evaluasi
      print(f"Akurasi: {accuracy:.4f}")
      print("\nConfusion Matrix:")
      print(conf_matrix)
      print("\nClassification Report:")
      print(report)
```



Akurasi: 0.8400

Confusion Matrix:

```
[[37  2]
 [14 47]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.73	0.95	0.82	39
1	0.96	0.77	0.85	61
accuracy			0.84	100
macro avg	0.84	0.86	0.84	100
weighted avg	0.87	0.84	0.84	100



# Visualisasi

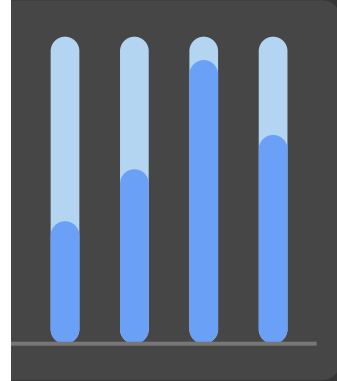
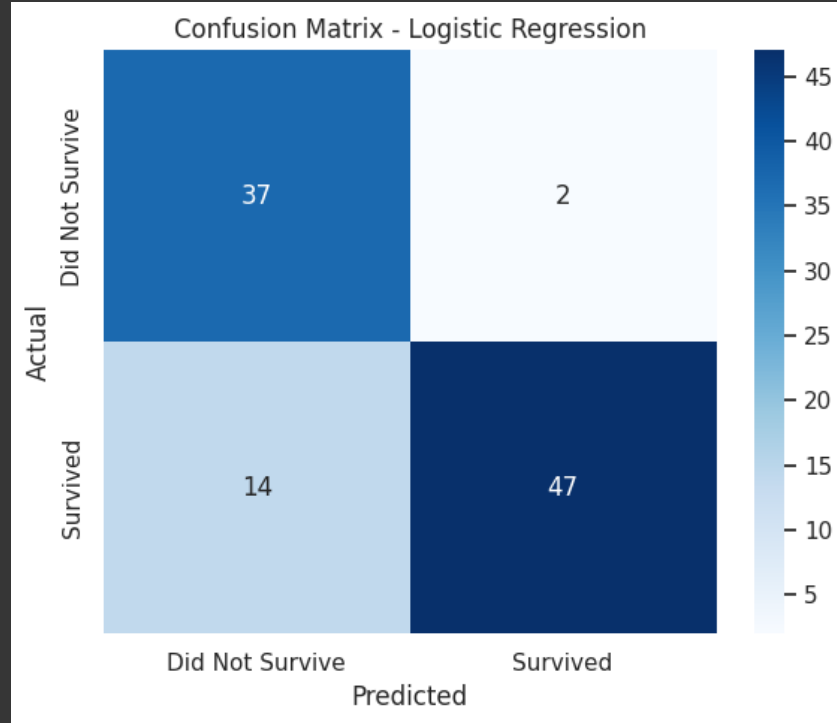


Visualisasi confusion matrix ini membantu kamu **melihat kesalahan model secara intuitif**, misalnya berapa banyak data yang diprediksi selamat padahal tidak, dan sebaliknya. Sangat berguna untuk evaluasi model klasifikasi.

```
[33] # === VISUALISASI CONFUSION MATRIX ===  
plt.figure(figsize=(6, 5))  
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',  
            xticklabels=['Did Not Survive', 'Survived'],  
            yticklabels=['Did Not Survive', 'Survived'])  
plt.xlabel('Predicted')  
plt.ylabel('Actual')  
plt.title('Confusion Matrix - Logistic Regression')  
plt.tight_layout()  
plt.show()
```



## Plot 6



## Model 2



- Mengimpor **RandomForestClassifier** dari modul `sklearn.ensemble`.
- Random Forest adalah **algoritma machine learning berbasis ensemble** yang menggunakan banyak **decision tree** dan menggabungkan hasilnya untuk meningkatkan akurasi dan mengurangi overfitting.
- Cocok untuk tugas **klasifikasi** dan **regresi**, serta mampu menangani fitur numerik maupun kategorikal.

```
[34] from sklearn.ensemble import RandomForestClassifier
```



# Random forest



- **RandomForestClassifier**: Membuat objek model Random Forest untuk klasifikasi.
  - **n\_estimators=100**: Menentukan jumlah **decision trees** dalam forest. Di sini, model menggunakan 100 pohon keputusan untuk membuat prediksi.
  - **random\_state=42**: Menetapkan nilai random seed untuk memastikan bahwa hasil pengacakan konsisten dan dapat direproduksi.
- **fit()**: Melatih model dengan data latih (X\_train sebagai fitur dan y\_train sebagai target).
- Model Random Forest akan mempelajari pola dalam data untuk memprediksi survived berdasarkan sex dan age.

```
[35] # Buat model Random Forest  
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)  
rf_model.fit(X_train, y_train)
```



RandomForestClassifier ⓘ ?  
RandomForestClassifier(random\_state=42)



# Prediksi



- **predict(X\_test)**: Menggunakan model **Random Forest** yang sudah dilatih untuk **memprediksi** target (dalam hal ini, apakah seseorang selamat atau tidak) berdasarkan data uji (X\_test).

- Hasil prediksi akan disimpan dalam variabel rf\_pred, yang berisi label prediksi untuk setiap sampel dalam X\_test.



# Prediksi

```
rf_pred = rf_model.predict(X_test)
```



# Evaluasi



- **accuracy\_score(y\_test, rf\_pred)**: Menghitung **akurasi** model Random Forest dengan membandingkan prediksi (rf\_pred) dengan nilai aktual (y\_test).
- Akurasi adalah proporsi prediksi yang benar dibandingkan dengan total data uji.
- **confusion\_matrix(y\_test, rf\_pred)**: Membuat **confusion matrix** untuk model Random Forest.
  - Matriks ini menunjukkan jumlah prediksi yang benar dan salah dalam tiap kategori (survived dan did not survive).
- **classification\_report(y\_test, rf\_pred)**: Menghasilkan **laporan klasifikasi** yang mencakup metrik seperti:
  - **Precision**: Seberapa akurat model dalam memprediksi kelas tertentu.
  - **Recall**: Seberapa baik model menemukan semua instance dari kelas tersebut.
  - **F1-score**: Rata-rata harmonik antara precision dan recall.
  - **Support**: Jumlah data aktual untuk tiap kelas.



## # Evaluasi

```
rf_accuracy = accuracy_score(y_test, rf_pred)
rf_conf_matrix = confusion_matrix(y_test, rf_pred)
rf_report = classification_report(y_test, rf_pred)
```



# Random Forest



- Menampilkan **akurasi** model Random Forest dalam format 4 angka desimal.
- Memberi gambaran umum tentang seberapa **tepat** model dalam melakukan prediksi.
- Menampilkan **confusion matrix**:
  - **TN (True Negative)**: Tidak selamat, diprediksi tidak selamat.
  - **FP (False Positive)**: Tidak selamat, diprediksi selamat.
  - **FN (False Negative)**: Selamat, diprediksi tidak selamat.
  - **TP (True Positive)**: Selamat, diprediksi selamat.
- Menampilkan **laporan klasifikasi lengkap** untuk setiap kelas (0 = tidak selamat, 1 = selamat), yang mencakup:
  - **Precision**: Keakuratan prediksi positif.
  - **Recall**: Kemampuan menangkap semua kasus positif.
  - **F1-score**: Keseimbangan precision dan recall.
  - **Support**: Jumlah data aktual untuk kelas tersebut.

```
[ ] print(f"Akurasi Random Forest: {rf_accuracy:.4f}")
    print("\nConfusion Matrix:")
    print(rf_conf_matrix)
    print("\nClassification Report:")
    print(rf_report)
```



Akurasi Random Forest: 0.8300

Confusion Matrix:

```
[[33  6]
 [11 50]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.75	0.85	0.80	39
1	0.89	0.82	0.85	61
accuracy			0.83	100
macro avg	0.82	0.83	0.82	100
weighted avg	0.84	0.83	0.83	100

# Visualisasi



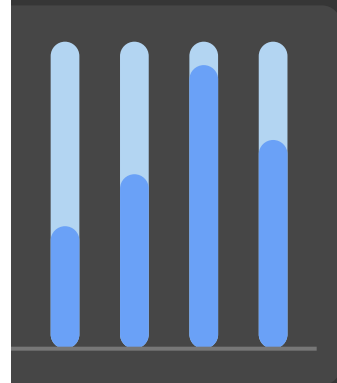
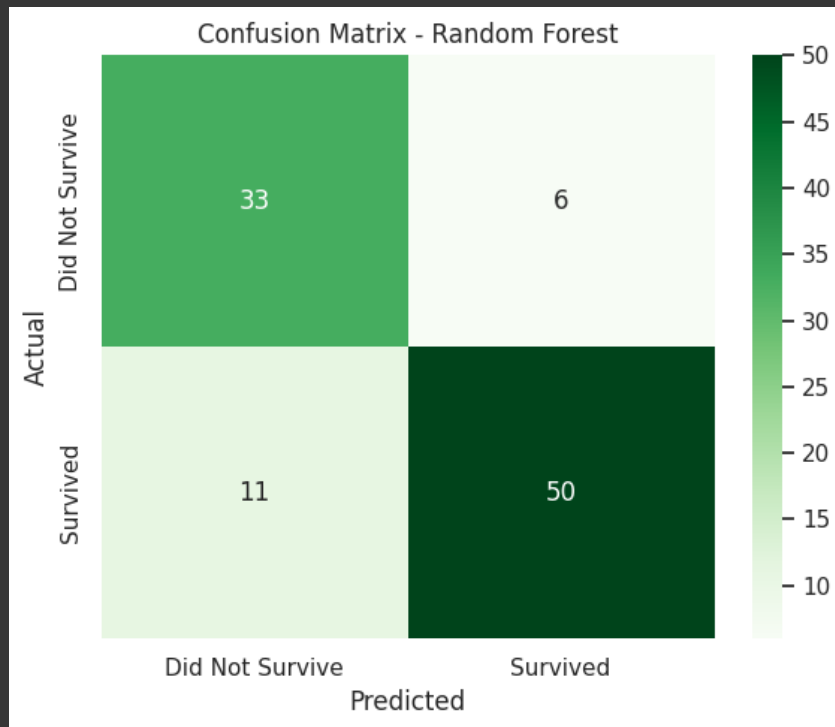
Visualisasi ini memudahkan untuk **melihat performa klasifikasi model Random Forest** secara langsung—terutama untuk mengidentifikasi kesalahan prediksi dan proporsi benar/salah antar kelas.

```
[ ] # Visualisasi Confusion Matrix Random Forest
plt.figure(figsize=(6, 5))
sns.heatmap(rf_conf_matrix, annot=True, fmt='d', cmap='Greens',
            xticklabels=['Did Not Survive', 'Survived'],
            yticklabels=['Did Not Survive', 'Survived'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Random Forest')
plt.tight_layout()
plt.show()
```





## Plot 7





- `model_names`: List berisi nama dua model yang digunakan untuk klasifikasi—Logistic Regression dan Random Forest.
- `accuracies`: List berisi nilai **akurasi** dari kedua model tersebut, yang sudah dihitung sebelumnya menggunakan `accuracy_score()`.

```
[ ] # === PERBANDINGAN AKURASI DALAM PLOT ===  
    model_names = ['Logistic Regression', 'Random Forest']  
    accuracies = [accuracy, rf_accuracy]
```

—Perbandingan akurasi

•Plot ini memungkinkan kamu untuk **memvisualisasikan perbandingan** akurasi antara kedua model (Logistic Regression dan Random Forest) secara jelas dan mudah dibaca.

```
[ ] plt.figure(figsize=(8, 5))
    sns.barplot(x=model_names, y=accuracies, palette='Set2')
    plt.ylim(0, 1)
    plt.ylabel('Accuracy')
    plt.title('Model Accuracy Comparison')
    for i, v in enumerate(accuracies):
        plt.text(i, v + 0.01, f"{v:.2f}", ha='center', fontweight='bold')
    plt.tight_layout()
    plt.show()
```

—Visualisasi

## Plot 8



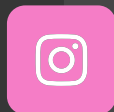
4

# Kesimpulan

You can enter a subtitle here if you need it

## Kesimpulan

Dalam analisis data Titanic, langkah pertama yang dilakukan adalah pembersihan data, termasuk penghapusan duplikat dan pengisian missing values pada kolom age. Eksplorasi data menunjukkan bahwa perempuan lebih cenderung bertahan hidup, sementara usia tidak terlalu berpengaruh terhadap kelangsungan hidup. Setelah itu, model **Logistic Regression** dan **Random Forest** diuji untuk memprediksi kelangsungan hidup. **Random Forest** menunjukkan akurasi yang lebih baik dibandingkan **Logistic Regression**, dengan performa yang lebih unggul dalam hal precision dan recall. Visualisasi perbandingan akurasi dan confusion matrix membantu memudahkan pemahaman hasil model. Secara keseluruhan, **Random Forest** lebih efektif dalam menangani data ini.



# Thanks!

Do you have any questions?  
nayakakhansaalana@gmail.com  
+62 812 9848 6620

linkedin.com/nayaka-khansa-alana  
@nykhnsln.lscn\_grdrndns1256  
github.com/nayakakhansaalana

=== SELESAI ===