ibimbing

Digital Skill Fair 38.0

Faculty of Data : Data Science & Data Analyst

Anbar Habibah

PORTFOLIO

About Me

Hi! I’m **Anbar Habibah**, a Management student with a growing passion for data. My journey into the world of data science started with simple curiosity, exploring spreadsheets, analyzing patterns, and turning numbers into stories.

And now, I’m proud to say that this project is not just any task, it's my very first project as I step into the world of data science. I’ve started learning the fundamentals: Python for data analysis, SQL for querying data, and visualization tools to make data easier to understand.

This is my first real taste of what it means to be a data scientist, and I’m more excited than ever to keep learning, exploring, and building impactful insights from data.

Education

Universitas Negeri Makassar - Management (2021 - 2025)

Technical Skills

- Basic Python
- Basic SQL
- Google Collab
- Tableau
- Microsoft Excel/Google Sheets (Pivot Table, Formulas)

Anbar Habibah anbarhabibah2@gmail.com @nbarr_

Introduction

This mini portfolio marks my very first step into the world of data science. As someone coming from a non-technical background, this is my initial attempt at applying what I've learned in Python, data analysis, and machine learning into real projects.

This Mini portfolio contains two beginner-friendly projects that reflect core concepts in data science. The first project focuses on classifying student grades using Python logic, helping me understand the importance of conditionals and user input. The second project explores the Titanic dataset through data cleaning, feature engineering, Exploratory Data Analysis (EDA), and machine learning—offering a deeper dive into building predictive models from raw data.

This portfolio represents not only what I’ve learned, but also my growing passion for using data to uncover insights and solve meaningful problems.

Contents

Project 1

Student Grade Classification

Categorized exam scores using Python logic structure (if-elif-else)

Project 1 - Student Grade Classification

Introduction

This project demonstrates a basic classification system for student exam scores using Python. It simulates a real-world scenario where student's names, IDs, and scores are collected and then evaluated into grade categories.

- Project Objectives
  - Practice handling user input and data types.
  - Implement conditional logic using if-elif-else.
  - Automatically classify student scores based on a defined grading rubric.

- Grading Rubric

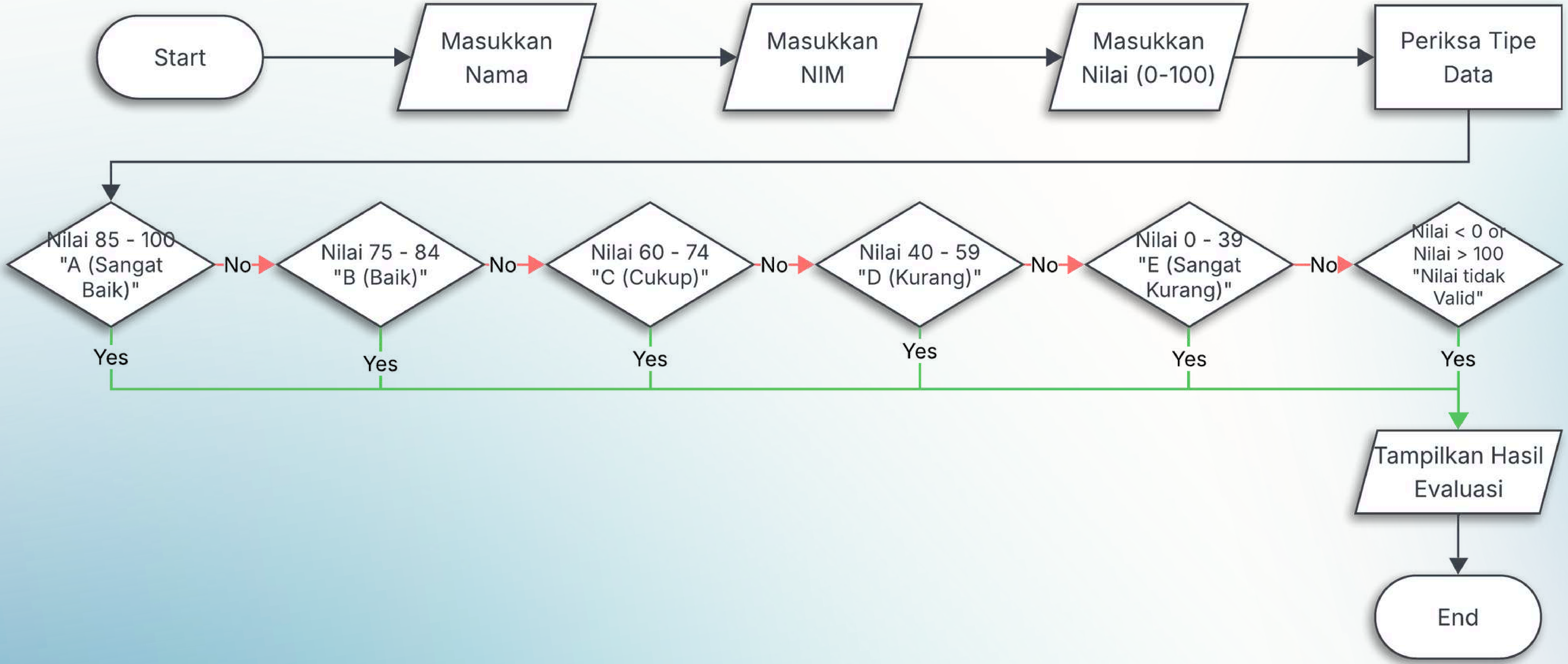
Score Range	Category
85 - 100	A (Excellent)
75 - 84	B (Good)
60 - 74	C (Average)
40 - 59	D (Poor)
< 40	E (Very Poor)

- Program Flow Overview

  Input ➡ Data Type Validation ➡ Grade Evaluation ➡ Output Display

Project 1 - Student Grade Classification

Flowchart



Project 1 - Student Grade Classification

Code Walkthrough with Insights

Collecting User Input

- Accepts input for student’s name, ID, and score.
- Explicit type casting ensures correct data handling.

```
# menginput data mahasiswa
print('============ Input Data Mahasiswa ============')
nama = str(input('Masukkan Nama Mahasiswa: '))
nim = str(input('Masukkan NIM: '))
nilai = int(input('Masukkan Nilai Ujian (0-100): '))
```

Data Type Display

- Displays each input along with its data type.
- Useful for debugging and understanding how the program interprets input.

```
# tipe data
print('\nNama: '+ nama + ' (type: ' + str(type(nama)) + ')')
print('NIM: '+ nim + ' (type: ' + str(type(nim)) + ')')
print('Nilai: '+ str(nilai) + ' (type: ' + str(type(nilai)) + ')')
```

Project 1 - Student Grade Classification

Code Walkthrough with Insights

Conditional Grade Evaluation

- Uses a decision structure to assign grade categories based on score range.
- Simple yet effective grading logic.
- This structure can be improved to include things like score weighting, bonus points, or pass/fail decisions.

```python
# evaluasi nilai
if nilai >= 85 and nilai <=100:
  kategori = 'A (Sangat Baik)'
elif nilai >= 75 and nilai < 85:
  kategori = 'B (Baik)'
elif nilai >= 60 and nilai < 75:
  kategori = 'C (cukup)'
elif nilai >= 40 and nilai < 60:
  kategori = 'D (kurang)'
elif nilai >= 0 and nilai < 40:
  kategori = 'E (sangat kurang)'
else:
  kategori = 'nilai tidak valid'
```

Output Code

- Summarizes the result in a readable format.
- Could be adapted for report generation or integration with other systems.

```python
# hasil evaluasi
print('\nHasil Evaluasi:')
print('Mahasiswa: ' + nama + ' (NIM: '+ nim +')')
print('Nilai Ujian: '+ str(nilai))
print('Kategori Nilai: '+ kategori)
print('==============================================')
```

Project 1 - Student Grade Classification

Output Display

```
============ Input Data Mahasiswa ============
Masukkan Nama Mahasiswa:
```

```
============ Input Data Mahasiswa ============
Masukkan Nama Mahasiswa: Anbar
Masukkan NIM:
```

```
============ Input Data Mahasiswa ============
Masukkan Nama Mahasiswa: Anbar
Masukkan NIM: 0000000001
Masukkan Nilai Ujian (0-100):
```

```
============ Input Data Mahasiswa ============
Masukkan Nama Mahasiswa: Anbar
Masukkan NIM: 0000000001
Masukkan Nilai Ujian (0-100): 98

Nama: Anbar (type: <class 'str'>)
NIM: 0000000001 (type: <class 'str'>)
Nilai: 98 (type: <class 'int'>)

Hasil Evaluasi:
Mahasiswa: Anbar (NIM: 0000000001)
Nilai Ujian: 98
Kategori Nilai: A (Sangat Baik)
==============================================
```

Project 1 - Student Grade Classification

ibimbing

Project Repository (Github)

SCAN ME

Click here: Github Link or

https://bit.ly/GradeClassify

Project 2

Titanic Survival Prediction

Exploratory Data Analysis (EDA) and machine learning modeling on the Titanic dataset to uncover survival patterns based on demographics and social status.

Project 2 - Titanic Survival Prediction

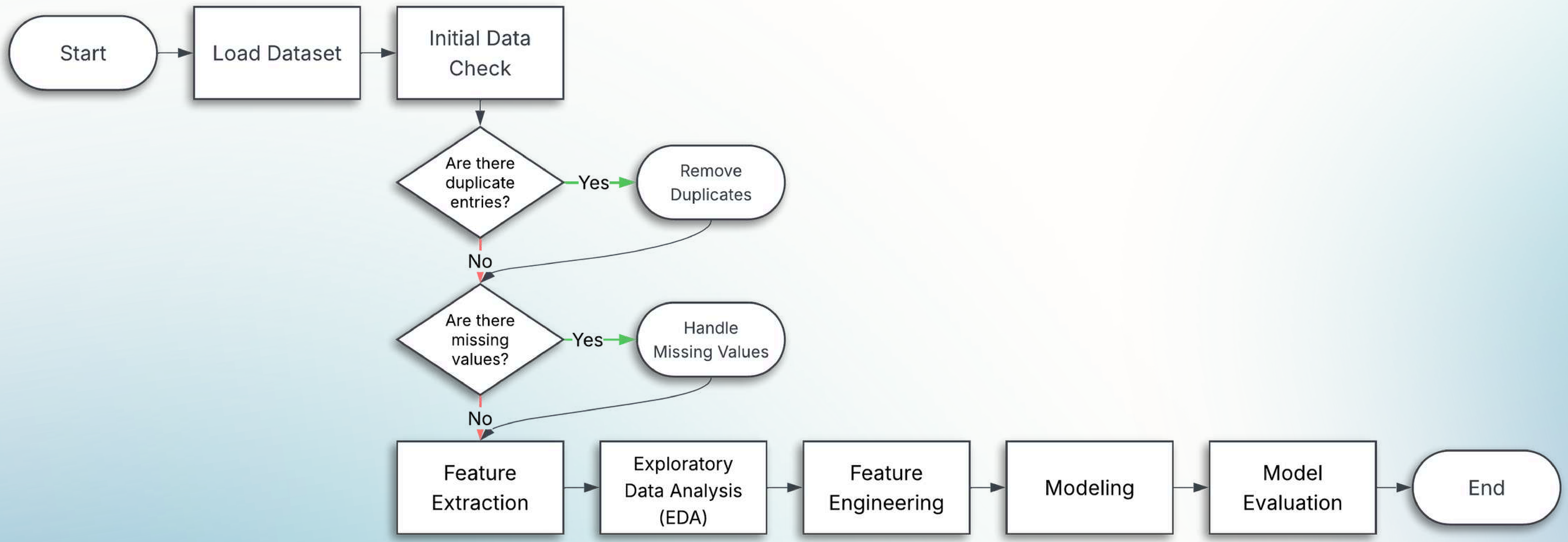
Introduction

This project focuses on predicting the survival of Titanic passengers using machine learning techniques. It explores key survival patterns based on demographic attributes and social status through comprehensive Exploratory Data Analysis (EDA) and model development.

- Project Objectives
  - Perform data cleaning and preprocessing
  - Analyze survival trends using visualizations
  - Engineer relevant features (e.g., titles, age categories)
  - Build and evaluate machine learning models
  - Compare model performances to identify the best approach
- Dataset Overview
  - Total Records: 500 rows
  - Key Columns: name, sex, age, survived
  - Missing Values: 9.82% in age
  - Feature Engineering: Added title and age_group columns

Project 2 - Titanic Survival Prediction

Flowchart



Project 2 - Titanic Survival Prediction

Data Cleaning & Feature Engineering

Data Cleaning

- Dropped duplicate entries to ensure data integrity

```
[315] len(data)
```

```
500
```

```
[316] len(data.drop_duplicates())
```

```
499
```

```
[317] len(data.drop_duplicates())/len(data)
      #jika output dari code di cell ini tidak bernilai 1 maka terdapat duplikat
```

```
0.998
```

output dari code di cell ini tidak bernilai 1, berarti ada data duplikat

```
list(data.columns)
```

```
['survived', 'name', 'sex', 'age']
```

```
[319] # mengambil baris duplikat (termasuk yang asli)
      duplicates = data[data.duplicated(keep=False)]
```

```
[320] # menampilkan list baris yang duplikat
      duplicates
```

	survived	name	sex	age
104	1	Eustis, Miss. Elizabeth Mussey	female	54.0
349	1	Eustis, Miss. Elizabeth Mussey	female	54.0

```
[323] # menghitung frekuensi kemunculan tiap baris duplikat
      duplicates_counts = duplicates.groupby(list(data.columns)).size().reset_index(name='juml

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

      # menampilkan hasil
      sorted_duplicates
```

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

```
[324] # handling drop duplicate
      data = data.drop_duplicates()
```

```
[325] len(data.drop_duplicates())/len(data)
```

```
1.0
```

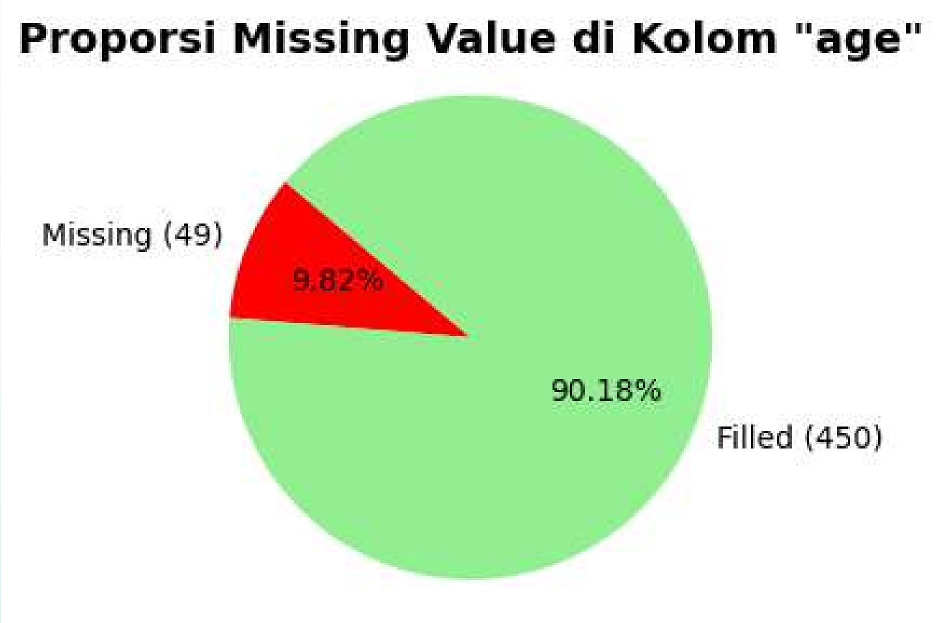
output dari code di cell ini bernilai 1, maka sudah tidak ada data duplikat atau sudah terhandle

Project 2 - Titanic Survival Prediction

Data Cleaning & Feature Engineering

Data Cleaning

- Handled missing values in age using median imputation




```
[326] data.isna().sum()
```

	0
survived	0
name	0
sex	0
age	49

dtype: int64

identified 49 missing values in the `age` column

The percentage of missing values is below 20%, so we handle missing values in numerical columns using the median and in categorical columns using the mode

```
[333] # Handling missing value for EDA, without splitting
      for column in data.columns:
          if data[column].dtype == 'object':
              # Jika kolom bertipe object, isi dengan mode
              data[column].fillna(data[column].mode()[0], inplace=True)
          else: # karena tipe data hanya object dan numerik saja tidak ada yg lain, maka pakai else berikut
              # Jika kolom bertipe numerik, isi dengan median
              data[column].fillna(data[column].median(), inplace=True)

<ipython-input-333-899fd06491ef>:8: FutureWarning: A value is trying to be set on a copy of a DataFram
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate o

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inpl


  data[column].fillna(data[column].median(), inplace=True)
<ipython-input-333-899fd06491ef>:5: FutureWarning: A value is trying to be set on a copy of a DataFram
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate o

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inpl


  data[column].fillna(data[column].mode()[0], inplace=True)

[334] data.isna().sum()
```

	0
survived	0
name	0
sex	0
age	0

Project 2 - Titanic Survival Prediction

ibimbing

Data Cleaning & Feature Engineering

Feature Engineering

- Created age group categories & Encoded

```
[337] def categorize_age(age):
          if pd.isnull(age):
              return 'unknown'
          elif age <= 1:
              return 'Baby\n(0-1)'
          elif age <= 4:
              return 'Toddler\n(2-4)'
          elif age <= 12:
              return 'Child\n(5-12)'
          elif age <= 19:
              return 'Teenager\n(13-19)'
          elif age <= 29:
              return 'Young Adult\n(20-29)'
          elif age <= 44:
              return 'Adult\n(30-44)'
          elif age <= 59:
              return 'Middle-Aged\n(45-59)'
          else:
              return 'Senior\n(60+)'

      df['age_group'] = df['age'].apply(categorize_age)
```

```
# Mapping age_group ke ordinal
age_group_map = {
    'Baby\n(0-1)': 0,
    'Toddler\n(2-4)': 1,
    'Child\n(5-12)': 2,
    'Teenager\n(13-19)': 3,
    'Young Adult\n(20-29)': 4,
    'Adult\n(30-44)': 5,
    'Middle-Aged\n(45-59)': 6,
    'Senior\n(60+)': 7,
    'unknown': -1  # Untuk yang missing age
}
df['age_group_encoded'] = df['age_group'].map(age_group_map)
```

- Encoded gender column (female = 0, male = 1)

```
[339] # Encode kolom sex: male = 0, female = 1
      df['sex_encoded'] = df['sex'].map({'female': 0, 'male': 1})
```

Project 2 - Titanic Survival Prediction

ibimbing

Data Cleaning & Feature Engineering

Feature Engineering

- Extracted titles from passenger names (e.g., Mr, Mrs, Miss) & encode

```
[341] df['title'] = df['name'].str.extract(r',\s*([^\.]+)\.', expa

      # Normalisasi title yang tidak umum
      df['title'] = df['title'].replace(['Mlle', 'Ms'], 'Miss')
      df['title'] = df['title'].replace(['Mme'], 'Mrs')
      df['title'] = df['title'].replace(['Dr', 'Rev', 'Major', 'Co

      df['title'].value_counts()
```

title	
Mr	256
Mrs	117
Miss	90
Rare	25
Master	11

```
[730] from sklearn.preprocessing import LabelEncoder

      le_sex = LabelEncoder()
      df['sex_encoded'] = le_sex.fit_transform(df['sex'])

      le_title = LabelEncoder()
      df['title_encoded'] = le_title.fit_transform(df['title'])
```

Transforming features into numerical values makes the dataset ready for further analysis or predictive modeling. This step ensures that models can recognize patterns and relationships within categorical data.

Before Encode

```
[732] df[['survived', 'name', 'sex', 'age', 'age_group', 'title']].head()
```

	survived	name	sex	age	age_group	title
0	1	Allen, Miss. Elisabeth Walton	female	29.0000	Young Adult\n(20-29)	Miss
1	1	Allison, Master. Hudson Trevor	male	0.9167	Baby\n(0-1)	Master
2	0	Allison, Miss. Helen Loraine	female	2.0000	Toddler\n(2-4)	Miss
3	0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	Adult\n(30-44)	Mr
4	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	Young Adult\n(20-29)	Mrs

After Encode

```
[733] df[['survived', 'name', 'sex_encoded', 'age_group_encoded', 'title_encoded']].head()
```

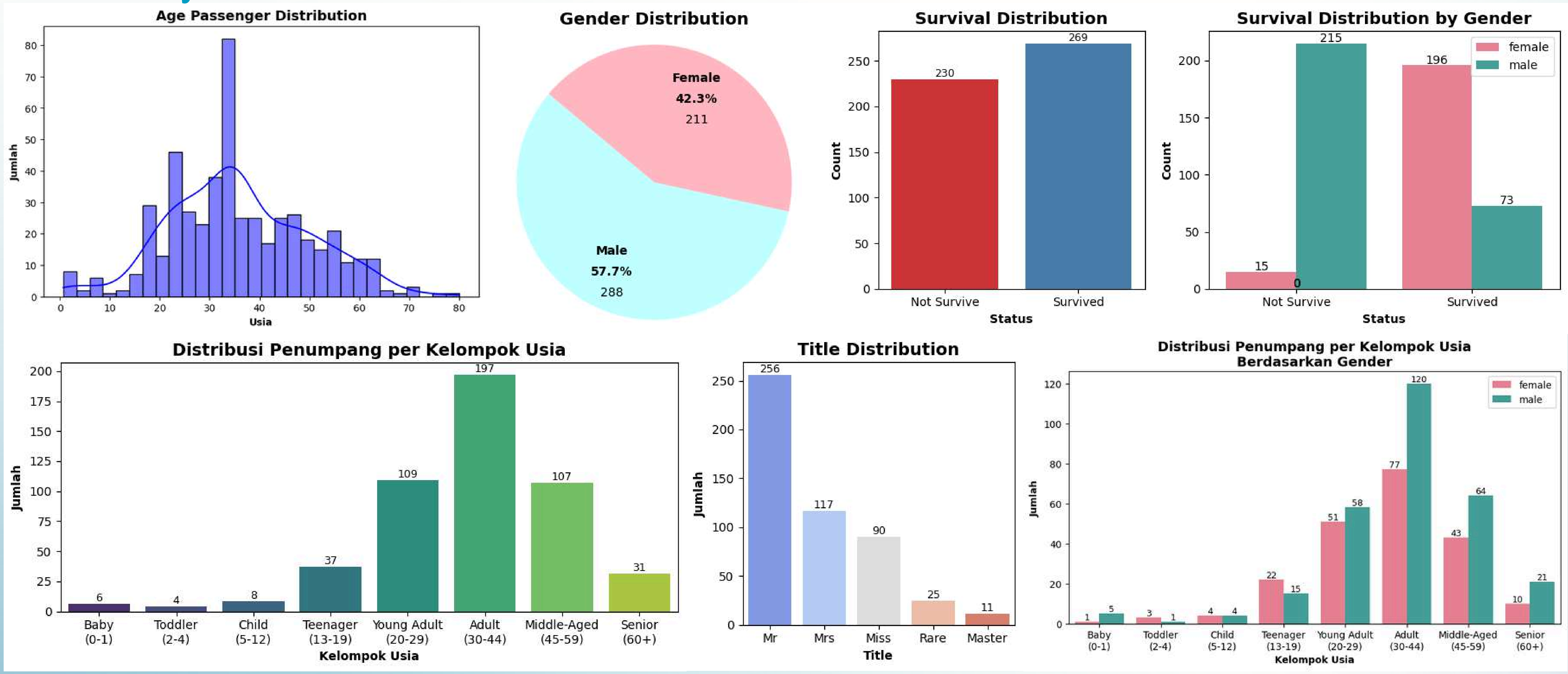
	survived	name	sex_encoded	age_group_encoded	title_encoded
0	1	Allen, Miss. Elisabeth Walton	0	4	1
1	1	Allison, Master. Hudson Trevor	1	0	0
2	0	Allison, Miss. Helen Loraine	0	1	1
3	0	Allison, Mr. Hudson Joshua Creighton	1	5	2
4	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	0	4	3

Project 2 - Titanic Survival Prediction

ibimbing

Exploratory Data Analysis (EDA)

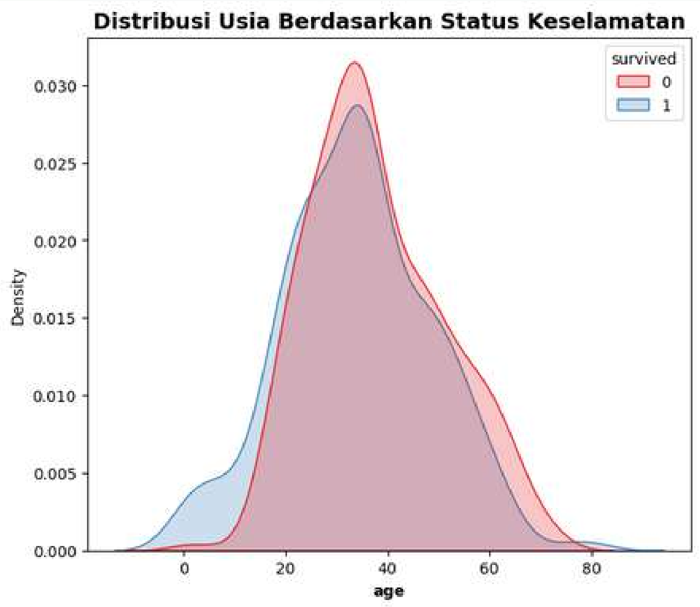
Univariate Analysis

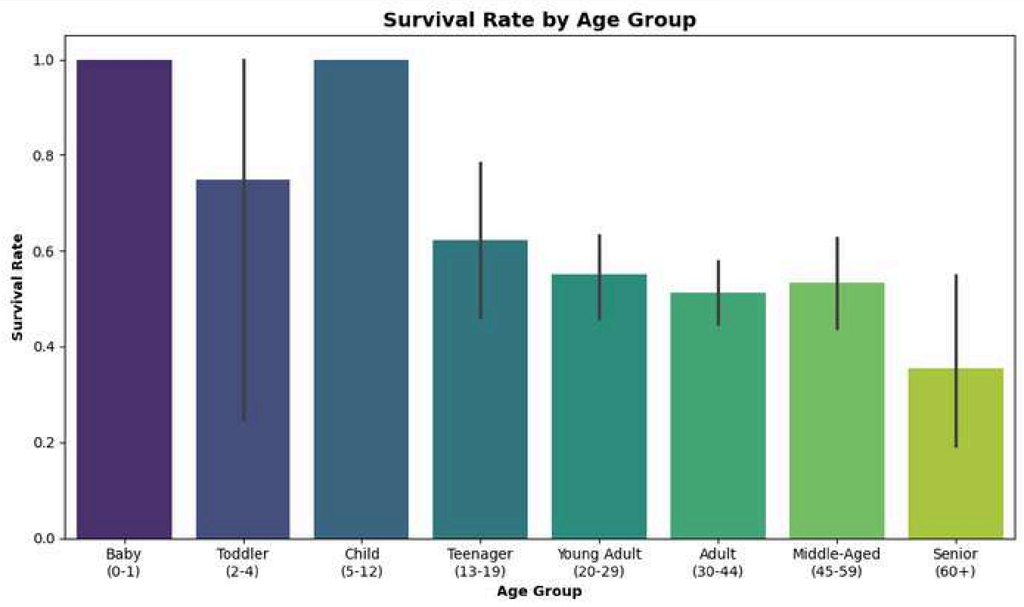


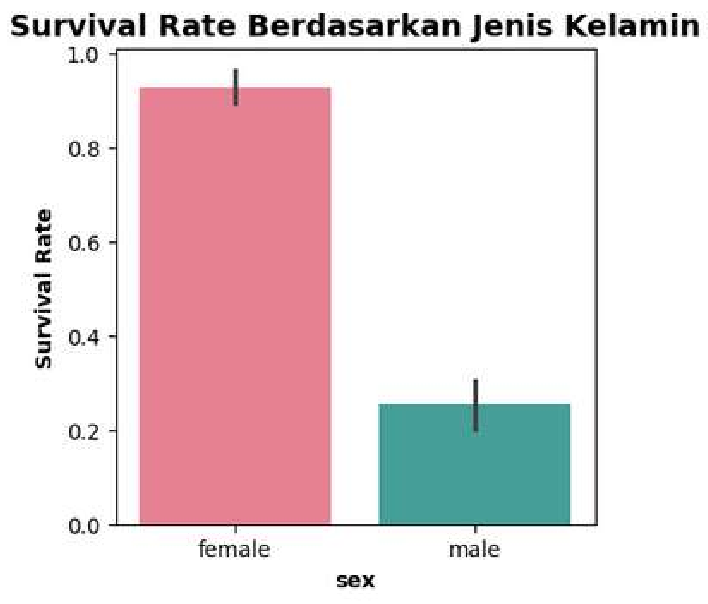
Project 2 - Titanic Survival Prediction

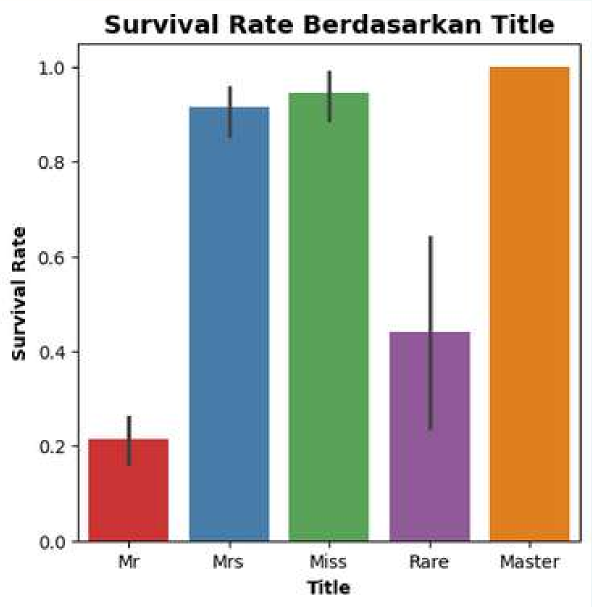
Exploratory Data Analysis (EDA)

Bivariate Analysis

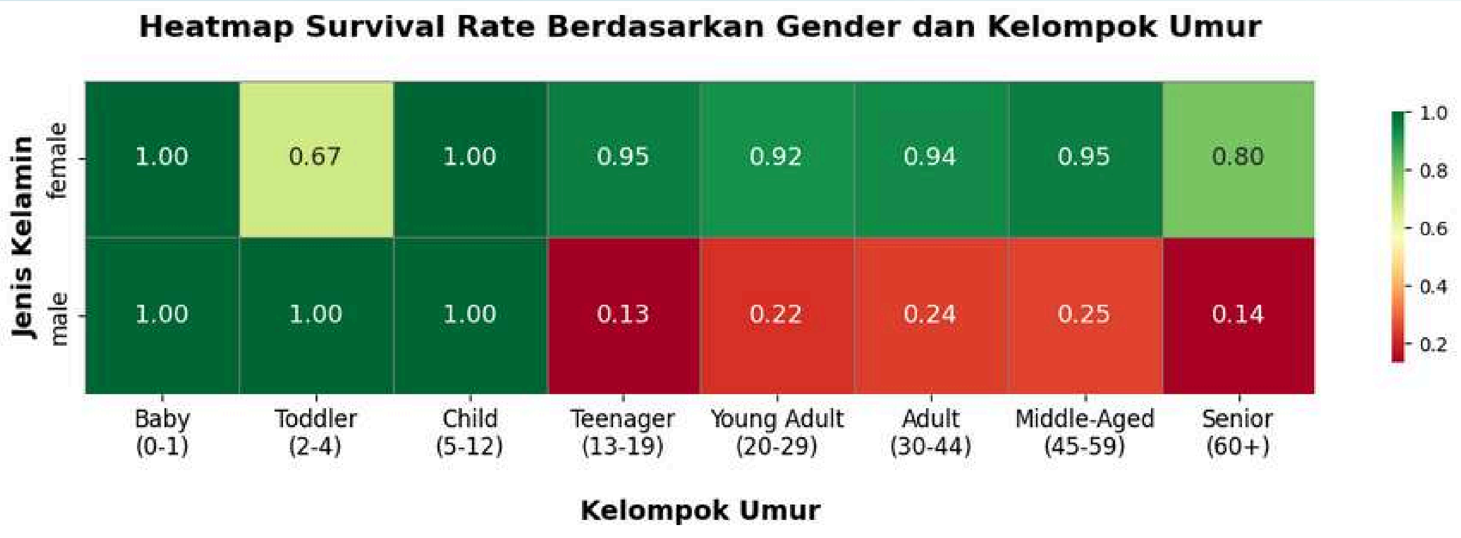




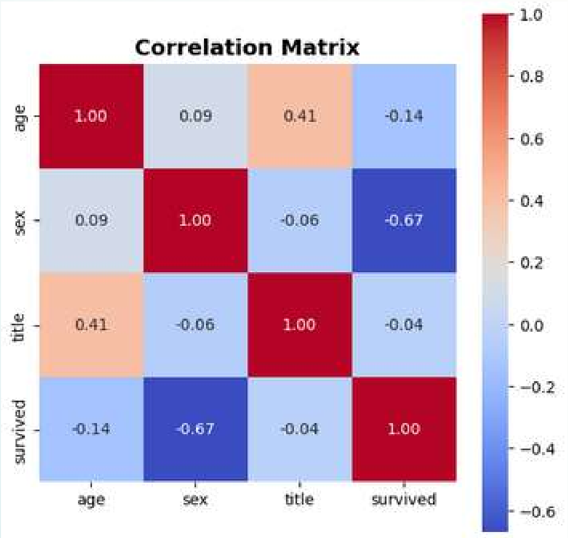




Multivariate Analysis



Correlation Matrix



Project 2 - Titanic Survival Prediction

Modeling & Evaluation

Model Comparison

```
from sklearn.metrics import classification_report
import pandas as pd

# Simpan classification report dari tiap model
models = {
    "Logistic Regression": y_pred_lr,
    "Decision Tree": y_pred_dt,
    "Random Forest": y_pred_rf,
    "XGBoost": y_pred_xgb,
    "KNN": y_pred_knn,
    "SVM": y_pred_svm}

# list untuk menyimpan hasil metrik
results = []

for name, preds in models.items():
    report = classification_report(y_test, preds, output_dict=True)
    results.append({
        "Model": name,
        "Accuracy": report["accuracy"],
        "Precision": report["weighted avg"]["precision"],
        "Recall": report["weighted avg"]["recall"],
        "F1-Score": report["weighted avg"]["f1-score"]})

# dataframe
df_results = pd.DataFrame(results)

# Tampilkan tabel perbandingan
df_results.sort_values(by="F1-Score", ascending=False)
```

	Model	Accuracy	Precision	Recall	F1-Score
4	KNN	0.87	0.887163	0.87	0.871547
0	Logistic Regression	0.85	0.874200	0.85	0.851837
3	XGBoost	0.85	0.874200	0.85	0.851837
2	Random Forest	0.85	0.874200	0.85	0.851837
5	SVM	0.85	0.874200	0.85	0.851837
1	Decision Tree	0.39	0.152100	0.39	0.218849

K-Nearest Neighbors (KNN) achieved the highest test set performance:

- Accuracy: 0.87
- F1-Score: 0.8715
- However, its cross-validation results were inconsistent (Mean Accuracy: 79.72%, Std: 0.0585), indicating sensitivity to data distribution.

Logistic Regression, Random Forest, XGBoost, and **SVM** showed similar and stable performance:

- Accuracy: 0.85
- Precision: 0.8742
- Recall: 0.85
- F1-Score: 0.8518
- These models are considered reliable for balanced classification tasks.

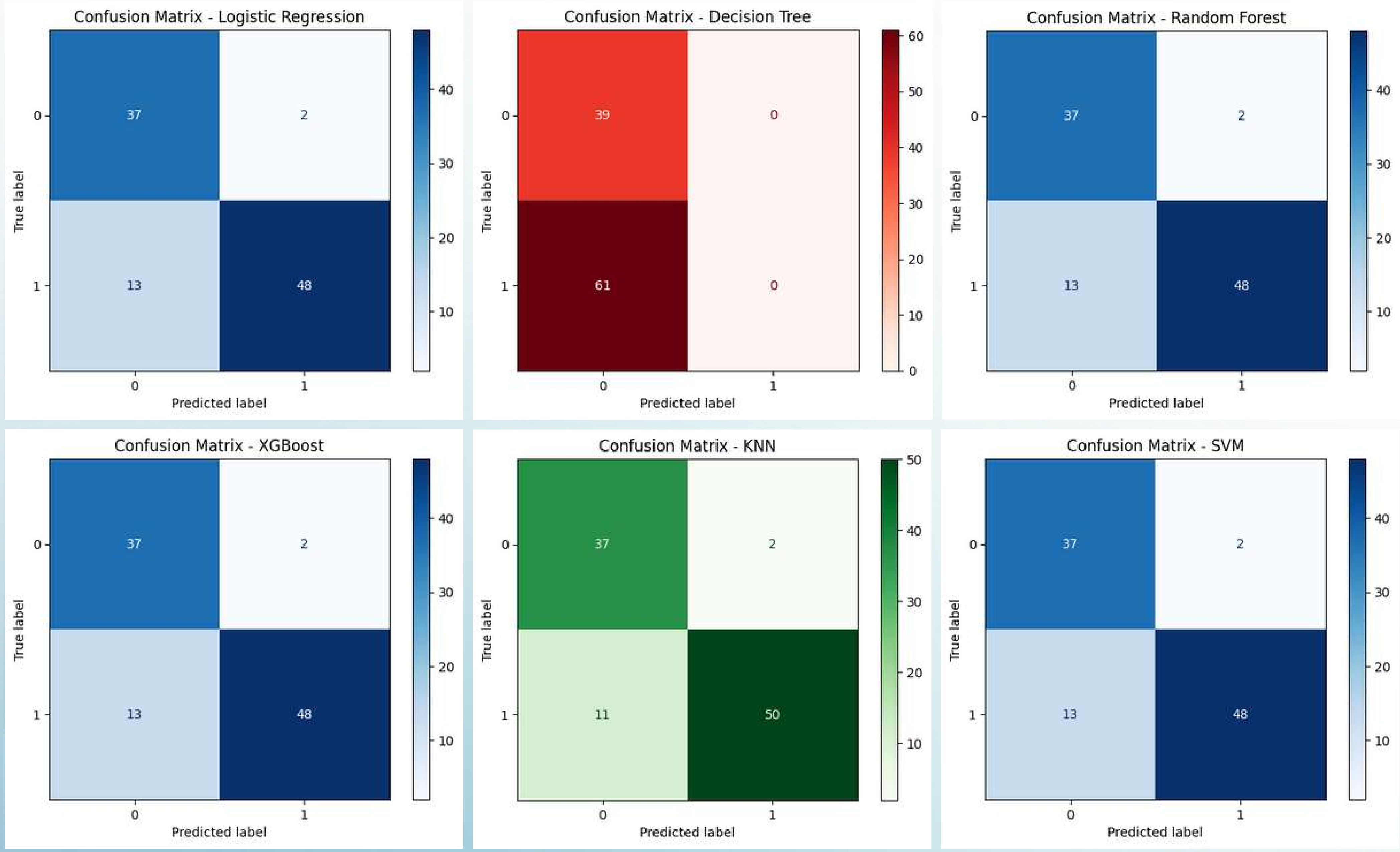
Decision Tree performed significantly worse:

- Accuracy: 0.39
- F1-Score: 0.2189
- Likely due to overfitting or inability to capture complex patterns.

Project 2 - Titanic Survival Prediction

Modeling & Evaluation

Confusion Matrix



Interpreting the Confusion Matrix

- **Top left (True Negative/TN):**
  - Model correctly predicted not survived
- **Bottom right (True Positive/TP):**
  - Model correctly predicted survived
- **Top right (False Positive/FP):**
  - Model predicted survived, but actually not
- **Bottom left (False Negative/FN):**
  - Model predicted not survived, but actually survived

Observations

- **Logistic Regression / Random Forest / XGBoost / SVM**
  - TP = 48, TN = 37
  - FN = 13, FP = 2
  - Balanced prediction, minimal misclassification
- **KNN**
  - TP = 50, FN = 11
  - Highest correct predictions of survivors
  - Slightly better in detecting survivors
- **Decision Tree**
  - TP = 0, TN = 39
  - FN = 61, FP = 0
  - Model failed to detect any survivors (only predicts one class)

Summary

- Most models predict both classes with good balance.
- KNN performs best in detecting survivors but can be unstable.
- Decision Tree performs poorly, fails to classify survivors at all.

Project 2 - Titanic Survival Prediction

ibimbing

Summary

Exploratory Data Analysis (EDA) Summary

- The Titanic dataset consists of 500 entries with 4 main columns: name, sex, age, and survived.
- The survived column indicates that approximately 54% of the passengers survived, slightly more than those who did not.
- The age column contains 49 missing values, which were handled during preprocessing.
- The data was analyzed through univariate, bivariate, and multivariate approaches:
  - The age distribution is fairly symmetrical with a mean of 35.9 years.
  - The majority of passengers were male.
  - The most common age groups were Young Adult and Adult.
- New features such as age_group and title (extracted from the name column) were successfully created and used for further analysis.
- Visualizations revealed that:
  - Female passengers had a higher survival rate than males.
  - Children and babies were more likely to survive compared to other age groups.
  - Passengers with titles such as Miss and Mrs had higher survival rates.

- Highlight
  - **The group with the highest survival rate** consisted of females with the title Miss or Mrs, particularly those in younger age groups.
  - Children and babies also had **high survival rates**, suggesting there may have been a rescue priority for younger passengers.
  - **The most vulnerable group** was adult males, especially those in the Young Adult to Middle-Aged categories.
  - Senior passengers (60+) had the **lowest survival rates**, possibly due to physical limitations during evacuation.
  - **The most significant correlation** was found between sex and survived.

Project 2 - Titanic Survival Prediction

Summary

Model Summary

- Models such as **Logistic Regression, Random Forest, XGBoost,** and **Support Vector Machine (SVM)** are considered **suitable** for further use due to their consistent and accurate performance across metrics.
- **K-Nearest Neighbors (KNN) achieved the highest test set performance but demonstrated instability in cross-validation**, indicating potential sensitivity to data distribution.
- In contrast, the **Decision Tree model is not recommended**, as its performance was significantly lower and less reliable compared to the others.

- ### Best Performing Model

  While Support Vector Machine (SVM) achieved the highest average accuracy (84.47%) in cross-validation, the best-performing model in this project is **Logistic Regression**. It was selected not only because it achieved results comparable to more complex models in terms of accuracy, precision, recall, and F1-score, but also because of its strengths in interpretability, efficiency, and simplicity. These qualities make it particularly suitable for binary classification tasks such as predicting passenger survival on the Titanic.

- ### Model Selection Rationale

  - **Simplicity & Interpretability:** Logistic Regression is easy to understand and well-suited for binary classification problems like survival prediction. It also provides direct insight into the contribution of each feature via regression coefficients.
  - **Strong Performance:** Based on evaluation results, it delivered similar accuracy, precision, recall, and F1-score as more complex models like Random Forest, XGBoost, and SVM.
  - **Efficiency:** It trains and predicts quickly, and does not require extensive parameter tuning.
  - **Feature Importance:** The model effectively highlighted important features such as sex, title, and age_group, which strongly influence the likelihood of survival.

Project 2 - Titanic Survival Prediction

ibimbing

Project Repository (Github)

SCAN ME

Click here: [Github Link] or

https://bit.ly/Titanic-EDA-ML

THANK YOU

Contact Me :

Anbar Habibah

anbarhabibah2@gmail.com