Task 1: Data cleaning & preprocessing

Objective: Learn how to clean and prepare raw data for ML.

Tools: - python, pandas, Numpy, Matplotlib/ Seaborn.

Data Cleaning & preprocessing using a Titanic Dataset

In this project we are learning how to clean raw data before feeding it into a machine learning model.

* Handling missing values
* Converting non numeric data into numbers.
* Normalizing values.
* Visualizing & removing outliers

Tools

VS Code
python
Libraries

Install required Libraries:

open VS → open Terminal - and type.

pip install pandas numpy matplotlib seaborn scikit-learn.

project setup

1. create a folder : Name it data_cleaning_project
2. open this folder in VS code.
3. Inside it, create a new file name it as data_cleaning.py.

Get the Dataset

Download the Titanic Dataset from Kaggle.

login to kaggle website and download the Titanic dataset zip file. after that unzip the file and store it in your project folder.

Load & Explore the data

Inside data_cleaning.py file. write the code

```
import pandas as pd.
import numpy as np.
import matplotlib.pyplot as plt.
import seaborn as sns.
```

pandas.

Data manipulation and analysis.

* It helps to read data from CSV, Excel etc.
  df = pd.read_csv() ← dataset file name. example Titanic-dataset.csv
* Allows you to clean data easily
* used for filtering, sorting, grouping data.

Reads the Titanic dataset into a DataFrame called df.

print(df.head()).

Shows the first 5 rows.

print(df.tail())

Shows the last 5 rows

print(df.info())

Shows info like data types and how many missing values exist

numpy

Numerical operations on arrays.

* It is fast and good for handling large numerical datasets.
* Useful for math operations like mean, median, standard deviation

matplotlib

Data visualization.

* To create graphs like bar charts, line plots, histograms, etc.

Seaborn

Statistical data visualization.

* It is built on matplotlib, but looks more beautiful and clean.
* Great for heatmaps, pair plots, boxplots.

print(df.isnull().sum()).

It shows how many values are missing in each column.

df['Age'].fillna(df['Age'].median(), inplace=True)

Fill the missing values using median

fillna() in pandas

we use fillna to fill the missing values.

df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True).

Fill the missing port of Embarkation with the most common value.

df.drop('cabin', axis=1, inplace=True)

Drops the Cabin Column -> it has too many missing values to fix

print(df.isnull().sum())

to check the missing values are handled or not.

missing values confure the model. By filling or removing them you ensure clean usable input.

Convert text to numbers

The titanic data contains text like male, female, S, C, etc.

But the machine learning can't read text → So you converted text to numbers.

```
df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})
```

* map({'male': 0, 'female': 1}) → Label Encoding.

This is used to convert text (categorical) data to numbers.

```
df = pd.get_dummies(df, columns=['Embarked'],
                    drop_first=True)
```

* get_dummies().

use this when there are more than 2 categories, like "Embarked" column in Titanic

It creates two new columns instead of 3. Because we used drop_first=True → it avoids duplicate info.

Normalizing the numbers (Scaling)

Some values like "Age" and "Fare" were on different scales.

* Age is between 0-80.
* Fare can go up to 500+.

So we scaled them to be on the same level

Standard Scaler is. transforms the data to. have.

* Mean = 0
* Standard Deviation = 1.

This is also called Z- Score normalization

Formula

$$Z = \frac{x - \mu}{\sigma}$$

x is the value.
μ is the mean of the column
σ is the standard deviation.

The Age & Fare Columns will now have values centered around 0, making it easier for ML models to learn from them..

from sklearn.preprocessing import StandardScaler

This imports a tool called StandardScaler from sklearn, which is used to scale or standardize numerical data.

Scaler = StandardScaler()

Creating a Scaler object. this object will help to transform the numbers in your dataset.

df[['Age', 'Fare']] = scaler.fit_transform(df[['Age', 'Fare']]).

fit_transform() does 2 things:

1. It calculates the mean and standard deviation of the data.
2. Then it transforms the data using this info.

Visualize outliers

We using boxplots to see if there are any extreme values (outliers).

```
sns.boxplot(x=df['Age'])
plt.title('Boxplot of Age')
plt.show()
sns.boxplot(x=df['Fare'])
plt.title('Boxplot of Fare')
plt.show()
```

Remove outliers

```
df = df[df['Fare'] < 300]
```

This makes your data more stable and less noisy

```
print(df.describe())
print(df.head())
```

To save cleaned data

```
df.to_csv('Cleaned_data.csv', index=False)
```