

Day 5 Training Report

27 June 2025

Data Collection & Cleaning — Kaggle, Handling Missing Values, Outliers, Duplicates

On Day 5, the focus shifted from Python programming to **real-world data handling** — a crucial step before applying any machine learning or AI techniques. This session covered **how to collect datasets**, particularly from **Kaggle**, and perform essential **data cleaning operations** such as handling missing values, outliers, and duplicates. Clean and well-structured data is the backbone of any successful AI project, making this day an important foundation for the upcoming modules.

1. Data Collection from Kaggle and Other Sources

Kaggle is one of the most popular platforms for **finding open-source datasets** related to various domains like healthcare, finance, education, and more.

Steps for Data Collection:

1. Create a Kaggle account and log in.
2. Search for relevant datasets using keywords.
3. Download the dataset in CSV or Excel format.
4. Load the dataset into Python using Pandas for analysis.

Example:

```
import pandas as pd

# Load a sample dataset
df = pd.read_csv('dataset.csv')
print(df.head())
```

Apart from Kaggle, students were also introduced to other sources such as **UCI Machine Learning Repository**, **GitHub repositories**, and **government open data portals**, which are valuable for finding authentic data.

2. Handling Missing Values

Real-world data often contains **missing or null values**, which can affect the accuracy of models. Various techniques were discussed to identify and handle these values:

- **Identifying missing values:**

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

- **Techniques to handle missing data:**
 - **Removing** rows/columns with excessive missing values using `dropna()`.
 - **Replacing** missing values using `fillna()` with mean, median, mode, or a constant.
 - **Imputation** methods for more advanced handling.

Example:

```
# Fill missing numerical values with the column mean
df['Age'].fillna(df['Age'].mean(), inplace=True)
```

3. Handling Duplicates

Duplicate records can lead to skewed analysis and model bias. Students learned to identify and remove duplicates using simple Pandas operations:

```
# Check for duplicates
print(df.duplicated().sum())

# Remove duplicates
df.drop_duplicates(inplace=True)
```

This ensures that the dataset remains **unique and consistent**.

4. Detecting and Handling Outliers

Outliers are data points that deviate significantly from the rest of the dataset. If not handled properly, they can impact statistical analysis and model performance.

Outlier Detection Methods:

- **Statistical methods** using the Interquartile Range (IQR):

```
Q1 = df['Column'].quantile(0.25)
Q3 = df['Column'].quantile(0.75)
IQR = Q3 - Q1
```

```
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
```

```
outliers = df[(df['Column'] < lower_bound) | (df['Column'] > upper_bound)]
```

- **Visualization methods** using boxplots or scatter plots to detect anomalies visually.
- **Handling outliers** by either **removing** them or **transforming** them based on the dataset context.