Name - Srushti Satish Sambare **D17A - 54** **ADS-2**

Aim: Apply data cleaning techniques (e.g. Data Imputation)

Theory:

Data Cleaning -
Data cleaning is the process of identifying and correcting (or removing) errors, inconsistencies, and inaccuracies in a dataset. It ensures data quality and reliability for analysis or machine learning models.

Steps in Data Cleaning -
1. Handling Missing Values: Filling in or removing missing data to avoid bias in analysis.
2. Removing Duplicates: Eliminating redundant records that may affect results.
3. Fixing Structural Errors: Correcting typos, inconsistent formatting, or incorrect labels.
4. Handling Outliers: Identifying and treating extreme values that may distort analysis.
5. Standardizing Data: Ensuring consistency in units, naming conventions, and formats.

Data Imputation -
Data imputation is the process of replacing missing values with estimated ones to retain data integrity. Missing data can arise due to various reasons such as sensor failures, human errors, or incomplete surveys.

Types of Imputation -
1. Mean/Median Imputation: Replacing missing numerical values with the mean or median of the column.
2. Mode Imputation: Filling missing categorical values with the most frequent category.
3. K-Nearest Neighbors (KNN) Imputation: Using similar observations to predict missing values.
4. Regression Imputation: Predicting missing values using a regression model based on other available features.
5. Multiple Imputation: Generating multiple estimates for missing values and averaging them for robustness.

Following is used in the code to perform Data Cleaning and Data Imputation -
1. Handling Missing Values (Data Imputation)
    a. Numerical Data:
    Used median imputation with *SimpleImputer(strategy='median')* to replace missing values in numerical columns with the median value. This is effective when data contains outliers, as the median is less affected by extreme values.
    b. Categorical Data:
    Used mode imputation with *SimpleImputer(strategy='most_frequent')* to replace missing values in categorical columns with the most frequently occurring value. This ensures that missing categories do not distort the dataset.

2. Removing Duplicates
   Used *df.drop_duplicates(inplace=True)* to eliminate duplicate rows from the dataset. This helps in maintaining data integrity and avoids redundant information.
3. Handling Outliers
   a. Interquartile Range (IQR) Method:
      i. Calculated *Q1 (25th percentile)* and *Q3 (75th percentile)* of numerical columns.
      ii. Computed the *IQR (Q3 - Q1)* to detect outliers.
      iii. Applied capping, where values below *(Q1 - 1.5 * IQR)* were set to the lower bound, and values above *(Q3 + 1.5 * IQR)* were set to the upper bound. This prevents extreme values from skewing the analysis.
4. Standardization of Numerical Data
   Used *StandardScaler()* from *sklearn.preprocessing* to scale numerical columns. This ensures that all numerical features have a mean of 0 and standard deviation of 1, improving the performance of machine learning models.
5. Comparison of Cleaned and Original Data
   Summary statistics before and after cleaning using *df.describe()*, allowing a side-by-side comparison of changes.

Code:

```python
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
# Load the CSV file
file_path = "/content/Accumulative_distribution.csv"
df = pd.read_csv(file_path)
# Display basic info and first few rows
print("Initial Data Info:")
print(df.info())
print(df.head())
```

```
Initial Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12160 entries, 0 to 12159
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   ID             12160 non-null  int64  
 1   Type           12160 non-null  object 
 2   Movie_number   12160 non-null  int64  
 3   Fly_number     12160 non-null  int64  
 4   Other_number   12160 non-null  int64  
 5   Difference_x   12160 non-null  float64
 6   Difference_y   12160 non-null  float64
 7   Distance       12160 non-null  float64
```

```
dtypes: float64(3), int64(4), object(1)
memory usage: 760.1+ KB
None
   ID          Type  Movie_number  Fly_number  Other_number  Difference_x  \
0   1  Dmelanogaster             1           1             0     -9.713147
1   2  Dmelanogaster             1           2             0    -23.854249
2   3  Dmelanogaster             1           3             0    -59.388140
3   4  Dmelanogaster             1           4             0    -62.456371
4   5  Dmelanogaster             1           5             0     -9.644838

   Difference_y   Distance
0      4.991989  10.920860
1    -16.237289  28.856104
2     63.130531  86.674189
3    -37.194611  72.692760
4      9.009750  13.198428
```

```python
# Handling Missing Values - Data Imputation
# Impute numerical columns with median
num_imputer = SimpleImputer(strategy='median')
# Impute categorical columns with most frequent value
cat_imputer = SimpleImputer(strategy='most_frequent')
num_cols = df.select_dtypes(include=[np.number]).columns
cat_cols = df.select_dtypes(include=['object']).columns
df[num_cols] = num_imputer.fit_transform(df[num_cols])
df[cat_cols] = cat_imputer.fit_transform(df[cat_cols])
# Removing Duplicates
df.drop_duplicates(inplace=True)
# Handling Outliers (Optional)
for col in num_cols:
  q1 = df[col].quantile(0.25)
  q3 = df[col].quantile(0.75)
  iqr = q3 - q1
  lower_bound = q1 - 1.5 * iqr
  upper_bound = q3 + 1.5 * iqr
  df[col] = np.where(df[col] < lower_bound, lower_bound, df[col])
  df[col] = np.where(df[col] > upper_bound, upper_bound, df[col])
# Standardizing Numerical Columns (Optional)
scaler = StandardScaler()
df[num_cols] = scaler.fit_transform(df[num_cols])
# Save the cleaned dataset
cleaned_file_path = "Cleaned_Accumulative_distribution.csv"
df.to_csv(cleaned_file_path, index=False)
print(f"Cleaned data saved at: {cleaned_file_path}")
```

```
Cleaned data saved at: Cleaned_Accumulative_distribution.csv
```

```python
df_original = pd.read_csv(file_path)
```

```python
# Compare original and cleaned data
print("Summary Statistics Before Cleaning:")
print(df_original.describe())
print("Summary Statistics After Cleaning:")
print(df.describe())
```

Before Cleaning

	ID	Movie_number	Fly_number	Other_number	Difference_x	Difference_y	Distance
count	12160.000000	12160.000000	12160.000000	12160.000000	1.216000e+04	1.216000e+04	12160.000000
mean	6080.500000	5.906250	9.500000	9.500000	-2.804774e-17	-2.337312e-17	42.227294
std	3510.433971	3.185639	5.766518	5.766518	3.647669e+01	3.870810e+01	32.335356
min	1.000000	1.000000	0.000000	0.000000	-1.113577e+02	-1.140549e+02	0.993842
25%	3040.750000	3.000000	4.750000	4.750000	-1.703750e+01	-1.833235e+01	8.792832
50%	6080.500000	6.000000	9.500000	9.500000	0.000000e+00	0.000000e+00	39.103716
75%	9120.250000	8.250000	14.250000	14.250000	1.703750e+01	1.833235e+01	70.268546
max	12160.000000	12.000000	19.000000	19.000000	1.113577e+02	1.140549e+02	114.943724

After Cleaning

	ID	Movie_number	Fly_number	Other_number	Difference_x	Difference_y	Distance
count	12160.000000	1.216000e+04	1.216000e+04	1.216000e+04	1.216000e+04	1.216000e+04	1.216000e+04
mean	0.000000	9.349247e-18	-2.191230e-19	1.168656e-17	8.764919e-19	-1.387779e-18	-3.739699e-17
std	1.000041	1.000041e+00	1.000041e+00	1.000041e+00	1.000041e+00	1.000041e+00	1.000041e+00
min	-1.731908	-1.540178e+00	-1.647509e+00	-1.647509e+00	-2.042617e+00	-2.011025e+00	-1.275234e+00
25%	-0.865954	-9.123349e-01	-8.237545e-01	-8.237545e-01	-5.106542e-01	-5.027564e-01	-1.034033e+00
50%	0.000000	2.943016e-02	0.000000e+00	0.000000e+00	3.076424e-18	-4.553649e-18	-9.660343e-02
75%	0.865954	7.357539e-01	8.237545e-01	8.237545e-01	5.106542e-01	5.027564e-01	8.672366e-01
max	1.731908	1.912960e+00	1.647509e+00	1.647509e+00	2.042617e+00	2.011025e+00	2.248914e+00

Dataset - [Insects Flight Dynamics]

Conclusion -

Hence, data cleaning and data imputation techniques were performed successfully on the Insects Flight Dynamics Dataset.