EXPERIMENT NO.2

Saumyaa Namungade **Roll No: B-808**

Aim: Apply Data Cleaning techniques (Eg : Data Imputation)

Theory:

What is Data Cleaning?

Data cleaning is the process of identifying, correcting, transforming, or removing inaccurate, incomplete, inconsistent, or irrelevant data from a dataset. It is a critical step in the data preprocessing phase and ensures that data is accurate, consistent, reliable, and ready for analysis or modeling.

Clean data is essential because poor-quality data can lead to incorrect insights, flawed models, and bad business decisions.

Importance of Data Cleaning

1. Improved Data Quality
- Removes errors, inconsistencies, and noise
- Ensures accuracy, completeness, and consistency
- Makes data trustworthy for analysis

2. Better Decision-Making
- Decisions are based on reliable and up-to-date information
- Reduces the risk of misleading conclusions
- Improves confidence in analytical results

3. Increased Efficiency
- Clean data is easier and faster to analyze
- Reduces time spent fixing errors during later stages
- Enhances performance of analytical and machine learning models

4. Compliance and Regulatory Requirements
- Helps meet data quality standards and regulations
- Reduces legal risks and penalties
- Ensures ethical and responsible data use

Common Data Quality Issues
- Missing values
- Duplicate records
- Incorrect or inaccurate data
- Inconsistent formats
- Outliers or extreme values
- Irrelevant or redundant data
- Structural errors

Data Cleaning Tasks

1. Handling Missing Data

- Removing rows or columns with excessive missing values
- Replacing missing values using:
  - Mean, median, or mode
  - Forward/backward filling
  - Predictive methods (e.g., regression)

2. Removing Duplicates

- Identifying repeated records
- Keeping only unique and relevant entries

3. Correcting Inaccuracies

- Fixing incorrect values (e.g., negative age, invalid dates)
- Correcting spelling errors and typos
- Validating data against known rules or constraints

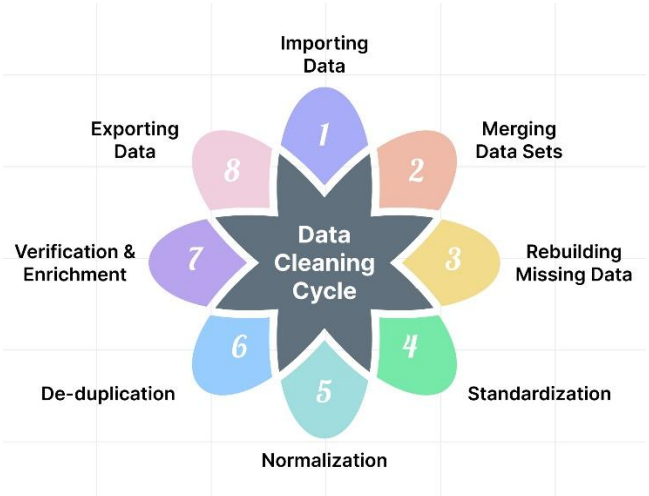
4. Standardizing Formats

- Ensuring consistent date formats (e.g., YYYY-MM-DD)
- Standardizing units (e.g., kg vs lbs)
- Converting text to consistent case (upper/lowercase)

5. Dealing with Outliers

- Identifying extreme or unusual values
- Deciding whether to:
  - Remove them
  - Transform them
  - Keep them if they are valid

Data Cleaning Cycle



1. Import Data

This is the first step where raw data is collected from various sources such as databases, files, sensors, surveys, or web sources.
The imported data may contain missing values, duplicates, inconsistencies, or errors.

2. Merge Data Sets

In this step, data from multiple sources is combined into a single dataset.
Care is taken to:

- Match common attributes
- Resolve conflicts between datasets
- Avoid duplicate records

3. Rebuild Missing Data

This step focuses on handling missing or incomplete data using data imputation techniques such as:

- Mean, median, or mode imputation
- Forward fill / backward fill
- Regression or KNN-based imputation

The goal is to replace missing values without losing important information.

4. Standardise Data

Standardisation ensures that data follows a consistent format and scale.
Examples include:

- Converting units (kg to grams)
- Uniform date formats (DD-MM-YYYY)
- Consistent text case (uppercase/lowercase)

This improves data comparability and usability.

5. Normalise Data

Normalization rescales numerical data to a common range (such as 0 to 1).
It helps:

- Improve machine learning model performance
- Reduce bias caused by large value ranges
- Maintain proportional relationships between values

6. De-duplicate Data

Duplicate records are identified and removed.
This ensures:

- No repeated entries
- Accurate counts and summaries

- Reduced storage and processing cost

7. Verify & Enrich Data

In this step, the cleaned data is:

- Verified for correctness and consistency
- Enriched by adding relevant external or derived data

Examples:

- Adding location details
- Calculated fields
- Reference data

8. Export Data

The final cleaned and processed dataset is exported to:

- Databases
- Data warehouses
- Analytics or machine learning systems

The data is now ready for analysis, reporting, or model building.

Reasons for Data Cleaning

Data cleaning is essential because:

- Raw data often contains errors and missing values
- Improves accuracy and reliability of analysis
- Enhances machine learning model performance
- Reduces misleading conclusions
- Ensures consistency across datasets
- Saves time and cost in later stages
- Supports better decision-making

Poor data quality leads to poor results, commonly stated as:

"Garbage In, Garbage Out (GIGO)"

Imputation Methods

Imputation is a data cleaning technique used to handle missing values by replacing them with estimated or substituted values instead of deleting records. Proper imputation helps maintain dataset size and improves the accuracy of analysis and machine learning models.

1. Mean Imputation

- Missing values are replaced with the mean (average) of the attribute.
- Used for numerical data.
- Simple and fast method.

Advantages:

- Easy to implement
- Maintains dataset size

Disadvantages:

- Affected by outliers
- Reduces data variability

2. Median Imputation

- Missing values are replaced with the median of the attribute.
- Suitable for skewed data.

Advantages:

- Robust to outliers
- Preserves central tendency

Disadvantages:

- Ignores relationships between variables

3. Mode Imputation

- Missing values are replaced with the most frequent value.
- Commonly used for categorical data.

Advantages:

- Simple
- Effective for nominal data

Disadvantages:

- Can introduce bias if one category dominates

4. K-Nearest Neighbors (KNN) Imputation

KNN Imputation fills missing values using the values from the most similar (nearest) data points in the dataset.

Working :

1. For each row with a missing value, the algorithm finds the K nearest rows based on other features (usually using Euclidean distance for numerical data).
2. The missing value is replaced with:
    - Mean (for numerical features) of the K neighbors
    - Mode (for categorical features) of the K neighbors

Advantages:

- Considers relationships between variables
- More accurate than simple statistical methods

Disadvantages:

- Computationally expensive
- Sensitive to choice of K

Program:

[11] ✓ 0s

```
import pandas as pd
import numpy as np
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler
```

[20] ✓ 0s

```
df = pd.DataFrame({
    'math_score':    [80, 90, np.nan, 70, 85],
    'reading_score':[78, np.nan, 88, 72, 90],
    'writing_score':[75, 85, 92, np.nan, 88]
})

print(df)
```

```
   math_score  reading_score  writing_score
0        80.0           78.0           75.0
1        90.0            NaN           85.0
2         NaN           88.0           92.0
3        70.0           72.0            NaN
4        85.0           90.0           88.0
```

[22] ✓ 0s

```
print("\nMissing Values:")
print(df.isnull().sum())
```

```
Missing Values:
math_score       1
reading_score    1
writing_score    1
dtype: int64
```

[23] ✓ 0s

```
num_cols = df.select_dtypes(include='number').columns
print("\nNumeric Columns:")
print(num_cols)
```

```
Numeric Columns:
Index(['math_score', 'reading_score', 'writing_score'], dtype='object')
```

[24] ✓ 0s

```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df[num_cols])

print("\nScaled Data:")
print(scaled_data)
```

```
Scaled Data:
[[-0.16903085 -0.54433105 -1.59111457]
 [ 1.18321596         nan  0.        ]
 [        nan  0.81649658  1.1137802 ]
 [-1.52127766 -1.36082763         nan]
 [ 0.50709255  1.08866211  0.47733437]]
```

[25] ✓ 0s

```
imputer = KNNImputer(n_neighbors=3)
imputed_scaled_data = imputer.fit_transform(scaled_data)

print("\nAfter KNN Imputation (Scaled):")
print(imputed_scaled_data)
```

```
After KNN Imputation (Scaled):
[[-0.16903085 -0.54433105 -1.59111457]
 [ 1.18321596  0.45360921  0.        ]
 [ 0.50709255  0.81649658  1.1137802 ]
 [-1.52127766 -1.36082763  0.        ]
 [ 0.50709255  1.08866211  0.47733437]]
```

[26] ✓ 0s

```
df[num_cols] = scaler.inverse_transform(imputed_scaled_data)

print("\nFinal Data After Imputation:")
print(df)
```

```
Final Data After Imputation:
   math_score  reading_score  writing_score
0        80.0      78.000000           75.0
1        90.0      85.333333           85.0
2        85.0      88.000000           92.0
3        70.0      72.000000           85.0
4        85.0      90.000000           88.0
```

```
[27]
✓ 0s
df[num_cols] = df[num_cols].round(1)
print("\nRounded Final Data:")
print(df)
```

```
Rounded Final Data:
   math_score  reading_score  writing_score
0        80.0           78.0           75.0
1        90.0           85.3           85.0
2        85.0           88.0           92.0
3        70.0           72.0           85.0
4        85.0           90.0           88.0
```

Conclusion:

In conclusion, I have successfully studied and implemented Data cleaning techniques.