

EDA – Data Cleaning

Aim: To perform data cleaning by handling missing values, removing duplicates, converting data types, and normalizing data.

Code:

```
import pandas as pd import numpy as np from sklearn.preprocessing  
import StandardScaler, MinMaxScaler
```

```
# Step 1: Create sample dataset data  
= {  
    "ID": [1, 2, 3, 4, 5, 5],  
    "Name": ["Alice", "Bob", "Charlie", "David", None, "David"],  
    "Age": [23, 25, np.nan, 24, 22, 22],  
    "Marks": [85, 78, 90, np.nan, 95, 95],  
    "Department": ["CSE", "ECE", "ME", "CIVIL", "AI", "AI"]  
}  
  
df = pd.DataFrame(data)  
print("Original Data:") print(df)
```

```
# Step 2: Handling missing values
print("\nHandling Missing Values:")
print("Detect missing:\n", df.isnull().sum())

# Fill missing with mean for Age, fill with mode for Marks
df["Age"].fillna(df["Age"].mean(), inplace=True)
df["Marks"].fillna(df["Marks"].mode()[0], inplace=True)

# Fill missing Name with "Unknown"
df["Name"].fillna("Unknown", inplace=True)

print("\nAfter Filling Missing Values:") print(df)

# Step 3: Remove duplicates df =
df.drop_duplicates() print("\nAfter
Removing Duplicates:") print(df)

# Step 4: Data type conversion df["ID"] =
df["ID"].astype(str) # Convert ID to string
print("\nAfter Data Type Conversion:")
print(df.dtypes)
```

```

# Step 5: Normalization scaler =
MinMaxScaler() df["Marks_MinMax"]=
scaler.fit_transform(df[["Marks"]])

standard_scaler = StandardScaler() df["Age_Standardized"]
= standard_scaler.fit_transform(df[["Age"]])

print("\nAfter Normalization:") print(df)

```

Output:

```

Original Data:
 ID Name Age Marks Department
 0 1 Alice 23.0 85.0 CSE
 1 2 Bob 25.0 78.0 ECE
 2 3 Charlie NaN 98.0 ME
 3 4 David 24.0 NaN CIVIL
 4 5 None 22.0 95.0 AI
 5 5 David 22.0 95.0 AI

Handling Missing Values:
Detect missing:
 ID
Name 1
Age 1
Marks 1
Department 0
dtype: int64

After Filling Missing Values:
 ID Name Age Marks Department
 0 1 Alice 23.0 85.0 CSE
 1 2 Bob 25.0 78.0 ECE
 2 3 Charlie 23.2 98.0 ME
 3 4 David 24.0 95.0 CIVIL
 4 5 Unknown 22.0 95.0 AI
 5 5 David 22.0 95.0 AI

After Removing Duplicates:
 ID Name Age Marks Department
 0 1 Alice 23.0 85.0 CSE
 1 2 Bob 25.0 78.0 ECE
 2 3 Charlie 23.2 98.0 ME
 3 4 David 24.0 95.0 CIVIL
 4 5 Unknown 22.0 95.0 AI
 5 5 David 22.0 95.0 AI

```

```

After Data Type Conversion:
   ID      Name    Age   Marks Department
0  1      Alice  23.0    85.0        CSE
1  2       Bob   25.0    78.0        ECE
2  3     Charlie  23.2    90.0         ME
3  4      David  22.0    95.0        CSCE
4  5    Unknown  22.0    85.0        AI
5  5      David  22.0    95.0        AI
dtype: object

After Normalization:
   ID      Name    Age   Marks Department  Marks_MinMax  Age_Standardized
0  1      Alice  23.0    85.0        CSE  0.411765  -0.187867
1  2       Bob   25.0    78.0        ECE  0.000000  1.698086
2  3     Charlie  23.2    90.0         ME  0.705882  0.600000
3  4      David  22.0    95.0        CSCE 1.000000  0.751469
4  5    Unknown  22.0    85.0        AI  1.000000  1.120333
5  5      David  22.0    95.0        AI  1.000000  -1.127204

/tmp/python-input-2005816956.py:22: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method((col: value), inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df["Age"].fillna(df["Age"].mean(), inplace=True)
/tmp/python-input-2005816956.py:23: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method((col: value), inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df["Marks"].fillna(df["Marks"].mode()[0], inplace=True)
/tmp/python-input-2005816956.py:26: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method((col: value), inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df["Name"].fillna("Unknown", inplace=True)

```

Result: Successfully cleaned the dataset by handling missing values, removing duplicates, converting data types, and normalizing data.