Name : Bhoomi Mangesh Naik
Class : D15C
Roll No. : 60
Practical No. : 2

Aim : To perform Data Preprocessing using Python.

Introduction:

Data preprocessing is an essential step in the machine learning pipeline that involves cleaning, transforming, and reducing data to improve the quality and efficiency of the model. This process ensures that the dataset is free from inconsistencies, missing values,and noise while making it suitable for analysis and modeling.

The dataset used in this exercise consists of the following attributes:

- Name: Name of the entity
- Country: Country of the entity
- Sales: Sales revenue
- Profit: Profit earned
- Assets: Total assets
- Market Value: Market valuation of the entity

Data Cleaning - removing missing values

Missing values can arise due to incomplete data collection or data entry errors. We can handle missing values in the following ways:

- Dropping missing values: Removing rows or columns with missing data.
- Replacing with a default value: Using a predefined value (e.g., zero or "Unknown").
- Replacing with the mean/median: Filling numerical missing values with the mean or median of the column.
- Grouped Mean Imputation: Filling missing values based on groups, such as the mean of each country.

Data Cleaning - removing noisy values

Noisy data consists of outliers or inconsistent values that can affect model performance.

We can handle noisy data using:

- Boxplot Analysis: Detecting and removing outliers based on IQR.
- Z-score Method: Removing values that fall outside a threshold.

Data Transformation

Transforming data involves converting categorical variables into numerical representations and vice versa.

- One-Hot Encoding
- Label Encoding

Data Normalization

Normalization scales numeric attributes to ensure uniformity in data distribution.

Common techniques include:

- Min-Max Scaling
- Z-score Standardization

Data Reduction

Data reduction techniques simplify data representation while retaining essential information. Two common methods include:

- Attribute-Oriented Induction: Reducing attributes based on relevance.
- Numerosity Reduction: Using clustering or sampling techniques.

Code and Output :-

```
import re
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from pathlib import Path

INPUT_FILE = "Data.csv"
OUTPUT_FILE = "cleaned_data.csv"
AGE_COL = "Age"
SALARY_COL = "Salary"

def coerce_numeric(series: pd.Series) -> pd.Series:
    def _k_to_number(x: str) -> str:
        if isinstance(x, str) and re.fullmatch(r"\s*\d+(\.\d+)?\s*[kK]\s*", x):
            num = re.findall(r"\d+(\.\d+)?", x)[0]
            return str(float(num) * 1000)
        return x
    series = series.astype(str).map(_k_to_number)
    series = series.str.replace(r"[^0-9\.\-]", "", regex=True)
    series = series.replace("", pd.NA)
```

```
    return pd.to_numeric(series, errors="coerce")

def encode_categoricals(df: pd.DataFrame) -> pd.DataFrame:
    cat_cols = [c for c in df.columns if df[c].dtype == "object" or str(df[c].dtype).startswith("category")]
    binary_cols, multi_cols = [], []
    for c in cat_cols:
        n = df[c].nunique(dropna=True)
        if n == 2:
            binary_cols.append(c)
        elif n > 2:
            multi_cols.append(c)
    for c in binary_cols:
        le = LabelEncoder()
        df[c] = df[c].astype("string")
        df[c] = le.fit_transform(df[c])
    if multi_cols:
        df = pd.get_dummies(df, columns=multi_cols, drop_first=True)
    return df

def print_section(title: str):
    print("\n" + "="*len(title))
    print(title)
    print("="*len(title))

path = Path(INPUT_FILE)
if not path.exists():
    raise FileNotFoundError(f"Couldn't find '{INPUT_FILE}' in {Path.cwd()}")

print_section("Loading data")
df = pd.read_csv(path)
print(f"Rows: {len(df)}  |  Columns: {len(df.columns)}")
print("\nColumn dtypes BEFORE:\n", df.dtypes)
print("\nMissing values BEFORE:\n", df.isna().sum())

print_section("Coercing numeric columns (Age, Salary) if needed")
if AGE_COL in df.columns:
    df[AGE_COL] = coerce_numeric(df[AGE_COL])
else:
    raise KeyError(f"Expected column '{AGE_COL}' not found in CSV.")
if SALARY_COL in df.columns:
    df[SALARY_COL] = coerce_numeric(df[SALARY_COL])
else:
```

```python
    print(f"Warning: Expected column '{SALARY_COL}' not found. Skipping salary dtype fix.")

print_section("Handling missing values (Age -> median)")
age_missing_before = df[AGE_COL].isna().sum()
age_median = df[AGE_COL].median(skipna=True)
df[AGE_COL] = df[AGE_COL].fillna(age_median)
print(f"Age missing before: {age_missing_before} | Median used: {age_median}")

print_section("Removing duplicates")
dup_before = df.duplicated().sum()
df = df.drop_duplicates()
dup_after = df.duplicated().sum()
print(f"Duplicates removed: {dup_before - dup_after}")

print_section("Encoding categorical variables")
df = encode_categoricals(df)
print("Categorical encoding complete.")

print_section("Fixing datatypes")
if SALARY_COL in df.columns:
    df[SALARY_COL] = df[SALARY_COL].astype("float64")
    print(f"'{SALARY_COL}' dtype ->", df[SALARY_COL].dtype)
else:
    print(f"'{SALARY_COL}' not present after transforms; nothing to cast.")

print_section("Handling outliers (removing Age > 100)")
outliers = (df[AGE_COL] > 100).sum()
df = df[df[AGE_COL] <= 100]
print(f"Outliers (Age > 100) removed: {outliers}")

print_section("After cleaning: quick check")
print("Rows:", len(df))
print("\nColumn dtypes AFTER:\n", df.dtypes)
print("\nMissing values AFTER:\n", df.isna().sum())

df.to_csv(OUTPUT_FILE, index=False)
print_section("DONE")
print(f"Saved cleaned file -> {OUTPUT_FILE}")
```

```
(.venv) PS C:\VSCODE\DMBI> python preprocessor.py
>>

============
Loading data
============
Rows: 10  |  Columns: 4

Column dtypes BEFORE:
 Country       object
Age          float64
Salary       float64
Purchased     object
dtype: object

Missing values BEFORE:
 Country      0
Age          1
Salary       1
Purchased    0
dtype: int64

================================================
Coercing numeric columns (Age, Salary) if needed
================================================

=======================================
Handling missing values (Age -> median)
=======================================
Age missing before: 1 | Median used: 38.0

===================
Removing duplicates
===================
Duplicates removed: 0

==============================
Encoding categorical variables
==============================
Categorical encoding complete.

================
Fixing datatypes
================
'Salary' dtype -> float64
```

```
=======================================
Handling outliers (removing Age > 100)
=======================================
Outliers (Age > 100) removed: 0

============================
After cleaning: quick check
============================
Rows: 10

Column dtypes AFTER:
 Age                float64
Salary             float64
Purchased            int64
Country_Germany       bool
Country_Spain         bool
dtype: object

Missing values AFTER:
 Age               0
Salary            1
Purchased         0
Country_Germany   0
Country_Spain     0
dtype: int64

====
DONE
====
Saved cleaned file -> cleaned_data.csv
(.venv) PS C:\VSCODE\DMBI>
```

preprocessor.py   Data.csv

Data.csv

```
Country,Age,Salary,Purchased
France,44,72000,No
Spain,27,48000,Yes
Germany,30,54000,No
Spain,38,61000,No
Germany,40,,Yes
France,35,58000,Yes
Spain,,52000,No
France,48,79000,Yes
Germany,50,83000,No
France,37,67000,Yes
```

Conclusion :

Data preprocessing is a crucial step in any data science or machine learning workflow. By performing preprocessing in Python using libraries like **Pandas, NumPy, and Scikit-learn**, we ensure that raw data is cleaned, transformed, and made suitable for analysis. Handling missing values, encoding categorical variables, feature scaling, and normalization improves data quality and consistency. This process enhances the efficiency and accuracy of machine learning models, ensuring more reliable and meaningful results.