Name: Khushi Singh
Class:D15C   Roll no. 71

Experiment No.2

Aim: To perform Data Preprocessing using Python.
Perform following operations using python
1. Handling the missing values on (on age with median)
2. remove duplicates
3. encode categorical variables
4 fix datatypes (e.g salary as float)
5. Handle Outliers (eg age>100)

Theory:
In data science and machine learning, raw datasets are often incomplete, inconsistent, or contain errors. Such data cannot be directly used to build reliable models. Therefore, data preprocessing is an essential step that involves transforming raw data into a clean, structured, and usable format.
The main objectives of data preprocessing are to improve the quality of data, reduce noise, handle missing or inconsistent values, and prepare the dataset so that it can be effectively used by machine learning algorithms. Properly preprocessed data ensures that models give more accurate, efficient, and meaningful results.

Steps in Data Preprocessing

1. **Data Cleaning**
   - Handling missing values (replacing with mean, median, mode, or using predictive methods).
   - Removing duplicates.
   - Correcting inconsistent data formats.

2. **Data Transformation**
   - Converting categorical values into numerical form using encoding techniques (Label Encoding, One-Hot Encoding).
   - Scaling and normalizing numerical values so that features lie within a common range.

3. **Data Reduction**
   - Reducing dimensionality by removing irrelevant or redundant features.
   - Summarizing data without losing essential information.

4. **Data Integration**
   - Combining data from multiple sources into a single dataset.

5. **Data Discretization (if required)**
    - Converting continuous data into categorical intervals (e.g., age groups).

Importance of Data Preprocessing

- Improves the accuracy of machine learning models.
- Handles inconsistencies and noise in the dataset.
- Ensures fair comparison of features with different scales.
- Saves time and resources during model training and evaluation.

Dataset Used

In this experiment, the dataset provided (from Kaggle: *Data Preprocessing Dataset*) contains information with missing values, categorical variables, and numerical data. It is used to demonstrate various preprocessing techniques such as handling missing values, encoding categorical features, removing duplicates, fixing data types, and handling outliers.

Code and Output

```
from google.colab import files
uploaded = files.upload()

import pandas as pd
import io
```

Choose files No file chosen Upload widget is only available when the cell has been executed

```
Saving dmbi_exp2_dataset.csv to dmbi_exp2_dataset (2).csv
```

```
df = pd.read_csv(io.BytesIO(uploaded['dmbi_exp2_dataset (2).csv']))
df.head()
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

```
df = df.copy()
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')

median_age = df['Age'].median()
df['Age'] = df['Age'].fillna(median_age)
df['Age'].isna().sum(), df['Age'].describe()
```

```
(np.int64(0),
 count    10.000000
 mean     38.700000
 std       7.257946
 min      27.000000
 25%      35.500000
 50%      38.000000
 75%      43.000000
 max      50.000000
 Name: Age, dtype: float64)
```

```
before = df.shape[0]
df = df.drop_duplicates()
after = df.shape[0]
print(f"Removed {before - after} duplicate rows")
df.duplicated().sum()
```

```
Removed 0 duplicate rows
np.int64(0)
```

```
cat_cols = df.select_dtypes(include=['object']).columns.tolist()
print("Categorical columns:", cat_cols)

df = pd.get_dummies(df, columns=cat_cols, drop_first=True)
df.head()
df.dtypes
df.columns
```

```
Categorical columns: []
Index(['Age', 'Salary', 'Country_Germany', 'Country_Spain', 'Purchased_Yes'], dtype='object')
```

```
if 'Salary' in df.columns:
    df['Salary'] = (
        df['Salary']
        .astype(str)
        .str.replace(r'[^0-9.\-]', '', regex=True)
    )
    df['Salary'] = pd.to_numeric(df['Salary'], errors='coerce')

df.dtypes
```

	0
Age	float64
Salary	float64
Country_Germany	bool
Country_Spain	bool
Purchased_Yes	bool

dtype: object

```
median_age = df['Age'].median()
outliers_mask = df['Age'] > 100
print("Ages > 100:", outliers_mask.sum())
df.loc[outliers_mask, 'Age'] = median_age
```

```
Ages > 100: 0
```

```
[ ] upper_cap = 100
    df['Age'] = df['Age'].clip(upper=upper_cap)
    (df['Age'] > 100).sum(), df['Age'].describe()
```

```
(np.int64(0),
 count    10.000000
 mean     38.700000
 std       7.257946
 min      27.000000
 25%      35.500000
 50%      38.000000
 75%      43.000000
 max      50.000000
 Name: Age, dtype: float64)
```

```
print("Shape:", df.shape)
print("Nulls:\n", df.isna().sum())
print("Dtypes:\n", df.dtypes)
```

```
Shape: (10, 5)
Nulls:
 Age                0
Salary             1
Country_Germany    0
Country_Spain      0
Purchased_Yes      0
dtype: int64
Dtypes:
 Age                 float64
Salary             float64
Country_Germany       bool
Country_Spain         bool
Purchased_Yes         bool
dtype: object
```

Conclusion

This experiment highlighted the crucial role of data preprocessing in transforming raw, inconsistent data into a clean and structured form suitable for analysis. By addressing missing values, encoding categorical variables, correcting data types, removing duplicates, and handling outliers, the dataset was significantly improved in quality and reliability. Effective preprocessing not only enhances the accuracy of machine learning models but also ensures that insights derived from the data are valid and trustworthy. Overall, this experiment demonstrated that preprocessing is the foundation for successful data analysis and model building.