DMBI: EXPERIMENT - 2

Aim:

To Perform data preprocessing using python.

Theory:

Data preprocessing is the process of transforming raw data into a clean and structured form suitable for analysis or machine learning. Real-world datasets often contain missing values, duplicate records, inconsistent formats, and outliers. If these issues are not addressed, they can lead to incorrect insights and poor model performance. Preprocessing ensures that the dataset is accurate, consistent, and ready for further analysis.
In this experiment, we perform the following steps:

1. **Handling Missing Values**
   Missing values occur when no data is stored for a particular observation. They can arise due to human error, data corruption, or incomplete collection. For numerical columns like Age, a common approach is to replace missing values with the median of the available values. The median is preferred over the mean when the data contains outliers, as it is less affected by extreme values.
2. **Removing Duplicates**
   Duplicate records occur when the same observation is recorded multiple times. These can skew results and give biased conclusions. Removing duplicates ensures that each record in the dataset is unique, which is important for accurate statistical analysis.
3. **Encoding Categorical Variables**
   Categorical variables contain discrete labels (e.g., Male, Female, USA, India). Machine learning algorithms and mathematical computations require numerical input.
   a. Label Encoding assigns a unique integer to each category.
   b. One-Hot Encoding creates separate binary columns for each category, avoiding any false numeric ordering.
4. **Fixing Data Types**
   Data types must match the nature of the values they represent. For example, Salary should be stored as a float (numeric type) rather than as a string. Correct data types prevent calculation errors and ensure compatibility with data processing libraries.
5. **Handling Outliers**
   Outliers are values that deviate significantly from the rest of the data (e.g., Age > 100). They may be due to errors or rare extreme cases.
   Outliers can distort statistical measures and model training. They can be handled by removal or transformation based on domain knowledge and statistical methods (e.g., Interquartile Range method).

Order of Operations:
The preprocessing steps should be performed in a logical sequence to avoid errors and redundant computation:

- Handle missing values → avoids processing NaNs later.
- Remove duplicates → prevents duplicate processing.
- Encode categorical variables → prepares data for numeric operations.
- Fix data types → ensures correct numerical computation.
- Handle outliers → final cleaning before analysis/modeling.

By following these steps, the dataset becomes clean, consistent, and ready for further statistical analysis or machine learning applications.

Conclusion:

In this experiment, data preprocessing was performed on a raw dataset using Python. The dataset was systematically refined by handling missing values, removing duplicate entries, encoding categorical variables, correcting data types, and addressing outliers. These steps ensured that the data was transformed from an inconsistent and error-prone form into a clean and structured format suitable for further analysis or machine learning applications.

The process revealed how unprocessed data can lead to misleading insights and errors in computation if not carefully managed. Challenges such as null values, repeated records, and extreme values were resolved through appropriate preprocessing techniques applied in a logical sequence. Overall, the experiment demonstrated the importance of preprocessing in improving data quality, reducing inconsistencies, and creating a reliable foundation for accurate analysis and effective model building.

CODE & OUTPUT

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('Data.csv')
X= dataset.iloc[:,:-1].values
y= dataset.iloc[:, -1].values
```

```
[32] from sklearn.impute import SimpleImputer
     imputer = SimpleImputer(missing_values=np.nan, strategy='median')
     imputer.fit(X[:, 1:3])
     X[:,1:3]=imputer.transform(X[:,1:3])
```

```
[33] dataset.drop_duplicates(inplace=True)
```

```
[34] dataset['Age'] = dataset['Age'].astype(float)
     dataset['Salary'] = dataset['Salary'].astype(float)
```

```
median_age = dataset['Age'].median()
dataset.loc[dataset['Age'] > 100, 'Age'] = median_age
```

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough')
X= np.array(ct.fit_transform(X))
print(X)
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
y= le.fit_transform(y)
print(y)
```

```
[[1.0 0.0 0.0 44.0 72000.0]
 [0.0 0.0 1.0 27.0 48000.0]
 [0.0 1.0 0.0 30.0 54000.0]
 [0.0 0.0 1.0 38.0 61000.0]
 [0.0 1.0 0.0 40.0 61000.0]
 [1.0 0.0 0.0 35.0 58000.0]
 [0.0 0.0 1.0 38.0 52000.0]
 [1.0 0.0 0.0 48.0 79000.0]
 [0.0 1.0 0.0 50.0 83000.0]
 [1.0 0.0 0.0 37.0 67000.0]]
[0 1 0 0 1 1 0 1 0 1]
```