Name: Shrey Pendurkar
Class: D15C
Batch: C
Roll No: 64

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.

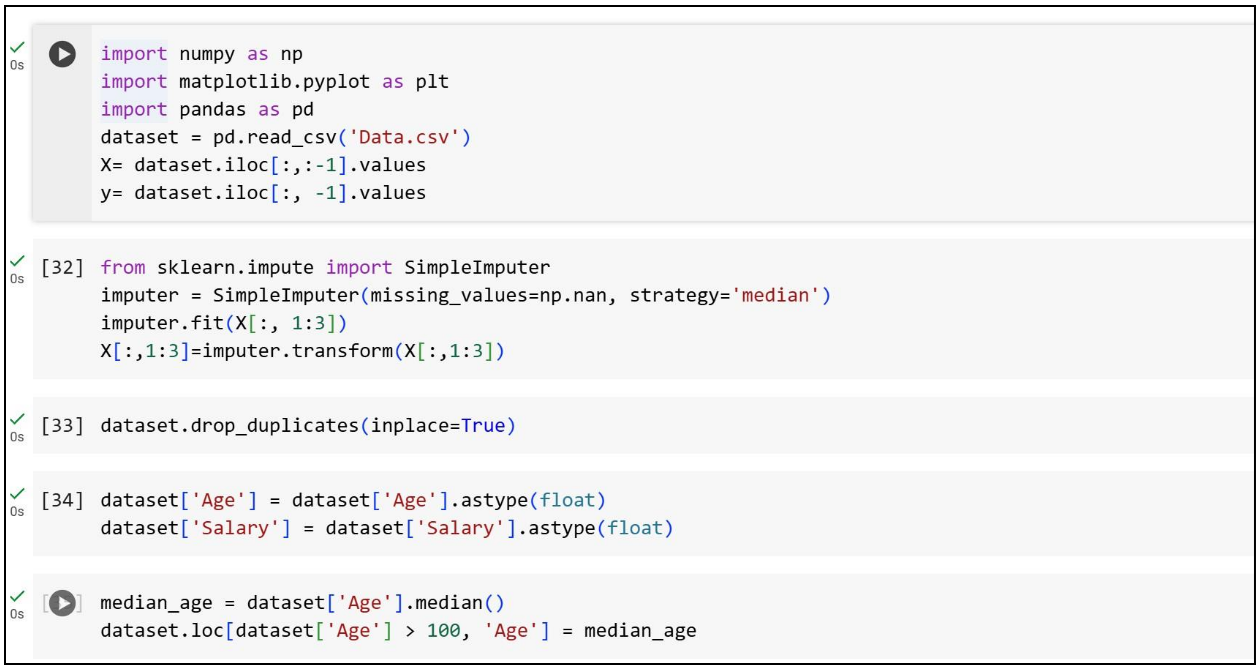
By following these steps, the dataset becomes clean, consistent, and ready for further statistical analysis or machine learning applications.

Conclusion:

Data preprocessing with Python involved handling missing values, removing duplicates, encoding categorical variables (label and one-hot encoding), fixing data types, and addressing outliers. These steps improved data quality and ensured compatibility with machine learning algorithms.

Common challenges faced included selecting appropriate imputation strategies for missing values, deciding the right encoding method for categorical data, detecting and handling outliers without losing vital information, and maintaining consistent data types throughout the process.

Output:



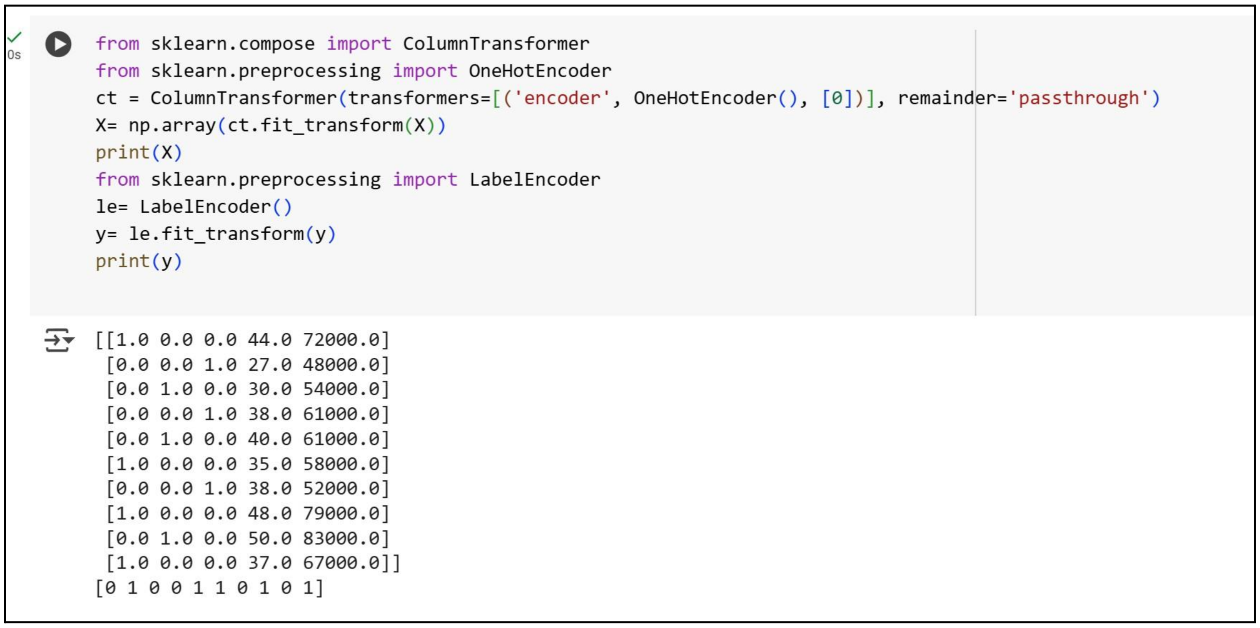
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('Data.csv')
X= dataset.iloc[:,:-1].values
y= dataset.iloc[:, -1].values
```

```
[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]
```