DMBI PRACTICAL 2

Aim: To Perform data preprocessing using python.

Theory:

Data preprocessing is a crucial step in the data analysis and machine learning pipeline. Raw data collected from real-world sources is often incomplete, inconsistent, and noisy, making it unsuitable for direct use in models. Preprocessing ensures that the data is clean, well-structured, and ready for further analysis or modeling.

In Python, data preprocessing is commonly done using libraries such as Pandas, NumPy, and Scikit-learn. The process typically includes the following steps:

1. **Data Cleaning**
   - Handling missing values (removal or imputation).
   - Correcting inconsistencies and removing duplicate records.
   - Detecting and handling outliers.
2. **Data Transformation**
   - Converting categorical variables into numerical form (e.g., one-hot encoding, label encoding).
   - Normalization and standardization to scale numerical values.
   - Applying mathematical transformations (e.g., log, square root) to stabilize variance.
3. **Data Integration**
   - Combining data from multiple sources into a single dataset.
   - Resolving conflicts in schema and data formats.
4. **Data Reduction**
   - Reducing dimensionality using techniques like PCA (Principal Component Analysis).
   - Feature selection to keep only relevant attributes.
   - Sampling to create smaller, representative datasets.
5. **Data Splitting**
   - Dividing the dataset into training, validation, and testing sets for model development and evaluation.

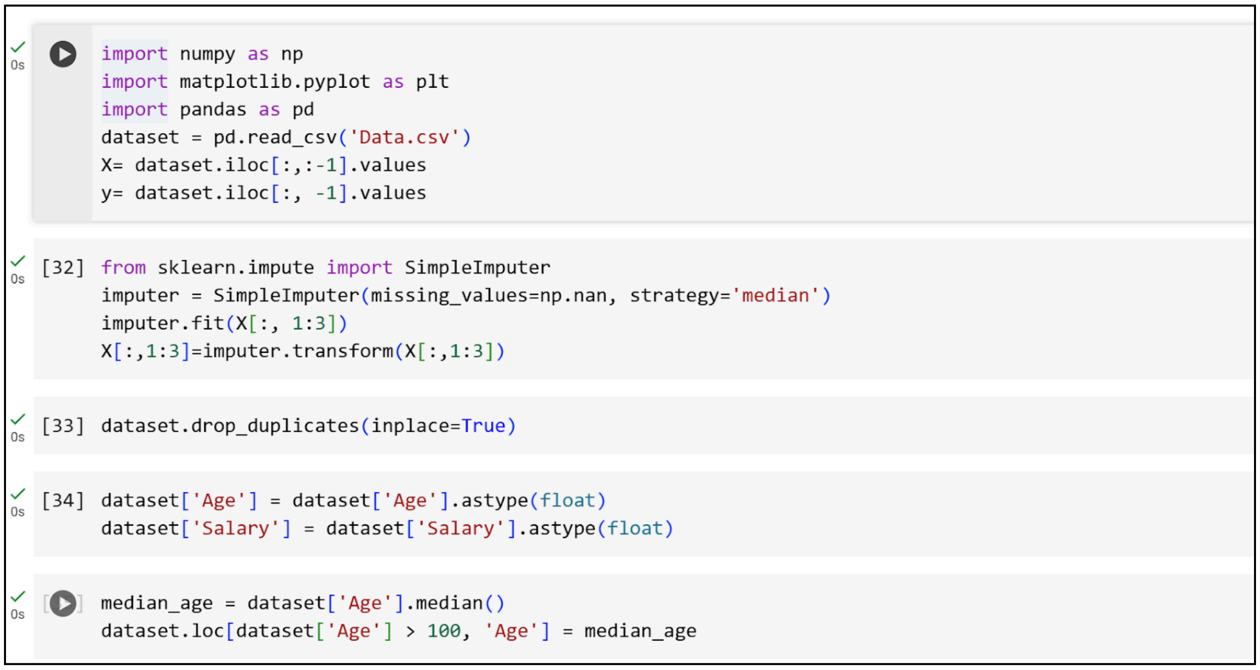
Conclusion

Through this experiment on data preprocessing using Python, I learned the importance of preparing raw data before analysis or modeling. I gained hands-on experience with handling missing values, removing duplicates, transforming categorical data, and applying normalization and standardization techniques. I also understood how crucial data integration, reduction, and splitting are for building reliable machine learning models.

During the process, I faced challenges such as dealing with inconsistent data formats, deciding the best way to handle missing values, and understanding when to use different preprocessing techniques like one-hot encoding or normalization. Overcoming these challenges helped me strengthen my problem-solving skills and build confidence in working with real-world datasets.

This journey taught me that proper preprocessing is the foundation of any successful data analysis or machine learning project, and without it, the results may be inaccurate or misleading.

Code and Output:



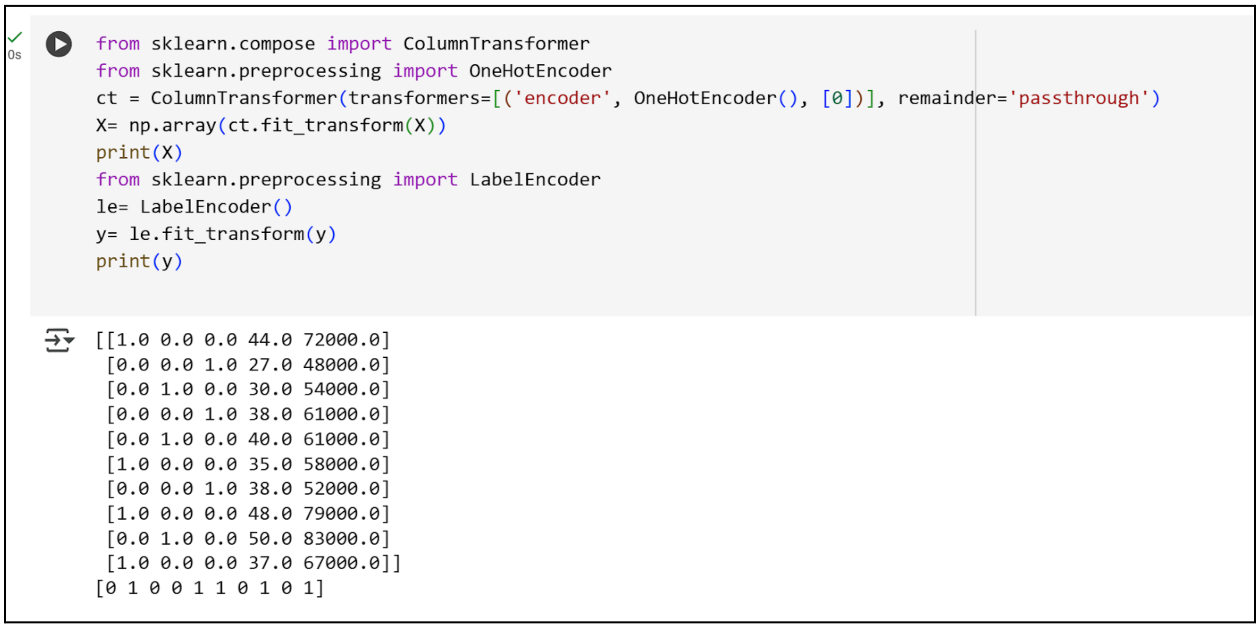
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('Data.csv')
X= dataset.iloc[:,:-1].values
y= dataset.iloc[:, -1].values
```

```
[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]
```