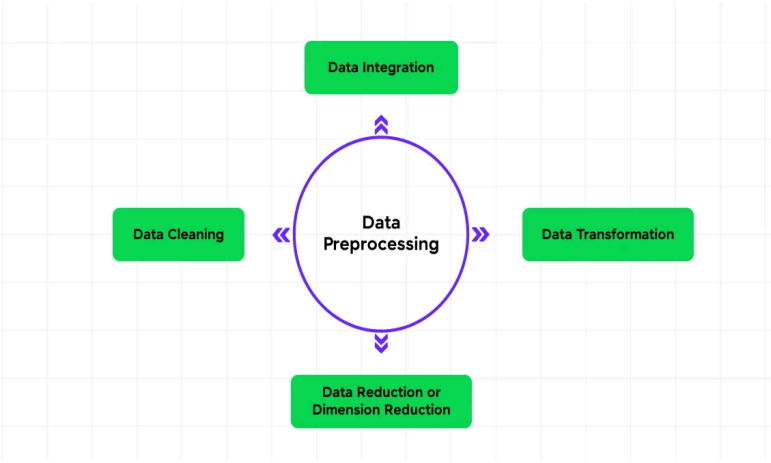
Experiment No. 1

Aim: To preprocess (Imputation, Label encoding and data cleaning) and prepare data using NumPy and Pandas in Python for effective analysis and modeling.

Software tools: Google Colab , Python Libraries(Numpy,Pandas)

Theory:

Python is the most extensively used and also the most preferred language by Data Scientists around the world. Data management and preprocessing are essential steps in the data analysis and machine learning pipeline. These steps involve preparing and organizing your data to make it suitable for analysis, modeling, and other data-related tasks.For these steps, we should import Python libraries for data preprocessing in Machine Learning.The predefined Python libraries can perform specific data preprocessing jobs.



Python libraries used for this data preprocessing in Machine Learning are :-

NumPy – NumPy is the fundamental package for scientific calculation in Python. Hence, it is used for inserting any type of mathematical operation in the code. Using NumPy, we can also add large multidimensional arrays and matrices in our code.

NumPy Basics :

Arrays: Creating and manipulating arrays, array indexing and slicing, array shape, and reshaping.

Array Operations: Mathematical operations, aggregation functions (sum, mean, etc.), and broadcasting.

np.array() → Create an array.

np.arange() → Create an array with evenly spaced values.
np.linspace() → Create an array with linearly spaced values.
np.zeros(), np.ones() → Create arrays filled with zeros/ones.
np.eye() → Identity matrix.
np.reshape() → Change array shape without changing data.
np.transpose() → Transpose array dimensions.
np.concatenate() → Join arrays.
np.split() → Split an array into multiple sub-arrays.
np.mean(), np.median(), np.std(), np.var() → Basic statistics.
np.min(), np.max(), np.sum(), np.prod() → Aggregate operations.
np.dot(), np.matmul() → Matrix multiplication.
np.unique() → Get unique values.
np.isnan(), np.isinf() → Check NaN or infinite values.

Pandas – Pandas is an excellent open-source Python library for data manipulation and analysis. It is extensively used for importing and managing the datasets. It packs in high-performance, easy-to-use data structures and data analysis tools for Python.

Pandas Basics :
Series: A one-dimensional labeled array capable of holding any data type.
DataFrame: A two-dimensional labeled data structure with columns of potentially different types.

pd.DataFrame() → Create a DataFrame.
df.head(), df.tail() → View first/last rows.
df.info(), df.describe() → Summary info and statistics.
df.shape, df.columns, df.index → Get dimensions and labels.
df.dtypes → Data types of columns.
df.isnull().sum() → Count missing values.
df.fillna(), df.dropna() → Handle missing data.
df.rename(), df.drop() → Rename or drop rows/columns.
df.sort_values(), df.sort_index() → Sort by values or index.

- **Preprocessing Techniques :-**

Imputation is the process of replacing missing or null values in a dataset. Missing data can occur due to errors in data collection, transmission, or entry.

Common Imputation Techniques:

- **Mean/Median/Mode Imputation**: Filling missing numerical values with the column's mean, median, or mode.
- **Forward Fill / Backward Fill**: Filling missing values using the previous or next value in the column.
- **Constant Value**: Replacing with a specific value such as 0 or "Unknown".

Encoding is the process of converting categorical (textual) data into numerical format so that it can be effectively used by machine learning algorithms. Since most models cannot interpret string or object data types, encoding is essential for feeding real-world data into algorithms.

- **Label Encoding** is a technique used to convert categorical (non-numeric**)** data into numerical form so that machine learning models can process them. It is an ordinal encoding method, meaning it implies an order or rank when used even if the original data doesn't have a natural order. Most models, especially mathematical ones like Linear Regression, SVM, or Logistic Regression, cannot work directly with text labels, they require numbers.The **LabelEncoder** class is part of **sklearn.preprocessing** and is widely used for performing label encoding.

eg:

```
from sklearn.preprocessing import LabelEncoder
import pandas as pd

# Sample data
df = pd.DataFrame({'Fruit': ['Apple', 'Banana', 'Mango', 'Banana', 'Apple']})
le = LabelEncoder()        # Create LabelEncoder object

# Fit and transform
df['Fruit_encoded'] = le.fit_transform(df['Fruit'])
print(df)
```

Conclusion :

Implemented preprocessing using Pandas and NumPy ensuring that the data is clean, consistent, and suitable for modeling. It directly impacts the accuracy and performance of analytical models and machine learning algorithms.