# 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.

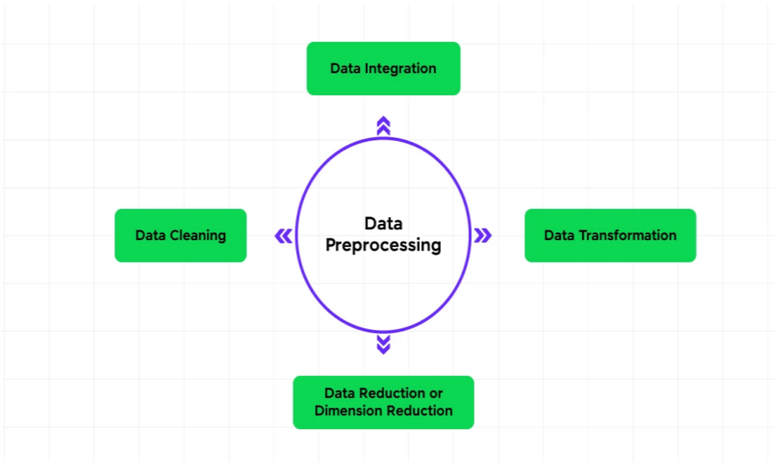
**Platform used:** Google Colab

**Theory:**

**Data Preprocessing:**

Data preprocessing is an essential step in the data analysis and machine learning pipeline. It involves data cleaning, transforming, and organizing raw data to make it suitable for further analysis or modeling. The main goals of data preprocessing are:

1. Improving data quality
2. Ensuring consistency
3. Making the data more manageable
4. Preparing the data for specific analytical or modeling techniques



Key aspects of data preprocessing include:

1. **Data cleaning:** Handling missing values, removing duplicates, and correcting errors.
2. **Data transformation:** Scaling, normalization, encoding categorical variables, and feature engineering.
3. **Data reduction:** Selecting relevant features and reducing dimensionality.
4. **Data integration:** Combining data from multiple sources.
5. **Data formatting:** Ensuring consistent data types and structures.

**Numpy & Pandas in Data Preprocessing:**

**Numpy:** NumPy is a powerful library for data preprocessing in Python. It provides an array-based computing environment that is optimized for numerical operations, making it a popular choice for scientific computing, data analysis, and machine learning. NumPy provides a wide range of mathematical functions that can be used for data preprocessing tasks such as scaling, normalization, and feature extraction.

**Pandas:** Pandas is the most commonly used library for data cleaning and preprocessing in Python due to its rich functionality for working with tabular data. Pandas provide flexible data structures like DataFrame and Series that allow data to be easily loaded, manipulated and explored.

**Basic Functions of Numpy & Pandas used for Data Preprocessing:**

**Numpy Basic Functions:**

**numpy.array():** Creates an array from a Python list or tuple.

**numpy.zeros():** Creates an array filled with zeros.

**numpy.ones():** Creates an array filled with ones.

**numpy.linspace():** Creates an array with a specified number of evenly spaced values over a specified interval.

**numpy.reshape():** Changes the shape of an array without changing its data.

**numpy.split():** Splits an array into multiple sub-arrays.

**Arithmetic:** numpy.add(), numpy.subtract(), numpy.multiply(), numpy.divide(), numpy.power(), numpy.mod().

**Aggregation:** numpy.sum(), numpy.min(), numpy.max(), numpy.mean(), numpy.std() (standard deviation), numpy.var() (variance).

**Pandas Basic Functions:**

**pd.read_csv():** Used to load data from various file formats into a DataFrame.

**df.head(), df.tail():** View the first or last n rows of a DataFrame.

**df.shape:** It displays the sample data's rows and columns (dimensions).

**df.info():** Provides a concise summary of the DataFrame, including data types, non-null values, and memory usage.

**df.describe():** Generates descriptive statistics (mean, standard deviation, percentiles, etc.) for numerical columns.

**df.loc[]:** Select rows and columns by label.

**df.iloc[]:** Select rows and columns by integer position.

**df.isnull(), df.isna():** Detects missing values (returns a boolean DataFrame).
**df.dropna():** Remove rows or columns containing missing values.
**df.fillna():** Fill missing values with a specified value or method.
**df.drop_duplicates():** Remove duplicate rows.
**df[].astype():** Change the data type of a column.
**df.value_counts():** Count unique values in a Series.

**Data Encoding:**
Data encoding in data science is the process of converting data, especially categorical data, into a numerical format that machine learning algorithms can understand and process. This is necessary because many machine learning algorithms rely on numerical computations and cannot directly handle categorical variables like "color" or "city".

**Common Data Encoding Techniques:**
**1. Label Encoding:**
Assigns a unique integer to each category. For example, "red" could be 1, "blue" 2, and "green" 3. This is useful when there's a natural order to the categories.
**2. One-Hot Encoding:**
Creates binary columns for each category, indicating the presence or absence of a category. For example, if the original column was "color" with values "red", "blue", and "green", one-hot encoding would create three new columns: "color_red", "color_blue", and "color_green", with values 1 or 0.
**3. Ordinal Encoding:**
Similar to label encoding, but specifically for categories with a meaningful order (ordinal data).
**4. Binary Encoding:**
Converts categories into binary digits, which can be useful for reducing dimensionality with high-cardinality categorical variables.
**5. Frequency Encoding:**
Assigns each category a value based on its frequency in the dataset.

**Conclusion:** Thus, we successfully performed data preprocessing and prepared the data for effective analysis and modeling using Numpy and Pandas.