Name - Gaurang A Raorane Div - D15A Roll no - 49 Batch - C

**Experiment - 1**

**Aim :-** Data preparation using NumPy and Pandas

**Theory:-**

### **NumPy:**

**NumPy** is a powerful numerical computing library for Python. It provides support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays. Some key features of NumPy include:

**Arrays:** NumPy's primary data structure is the ndarray, a multi-dimensional array. It allows efficient and convenient manipulation of large datasets.

**Mathematical Functions:** NumPy provides a wide range of mathematical functions for performing operations on arrays. These functions are optimized for numerical computations, making them fast and efficient.

**Broadcasting:** NumPy supports broadcasting, a powerful mechanism that allows operations between arrays of different shapes and sizes, making code more concise and readable.

**Random Module:** NumPy includes a random module for generating random numbers and arrays. This is useful for simulations and random sampling.

**Linear Algebra Operations:** NumPy includes a rich set of functions for linear algebra operations, such as matrix multiplication, eigenvalue decomposition, and solving linear equations.

### **pandas:**

pandas is an open-source data manipulation and analysis library for Python. It provides data structures like Series and DataFrame that are designed for efficient and intuitive data manipulation. Key features of pandas include:

**DataFrame:** The DataFrame is a two-dimensional table with labeled axes (rows and columns). It is a powerful and flexible data structure for handling structured data.

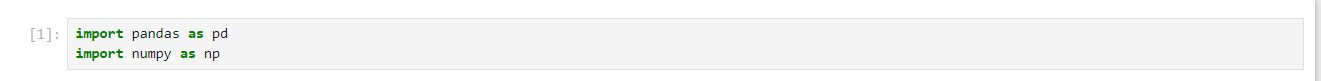
**Data Cleaning and Preprocessing:** pandas provides various functions for handling missing data, removing duplicates, and reshaping data. It allows for efficient data cleaning and preprocessing tasks.

**Data Indexing and Selection:** pandas offers powerful indexing capabilities, enabling easy access and manipulation of data. It allows indexing by both labels and integer-based positions.

**Time Series Data:** pandas includes functionality for handling time series data, making it suitable for financial and temporal analysis.

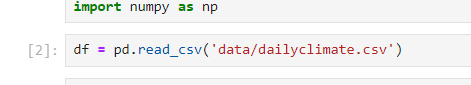
**Integration with NumPy:** pandas is built on top of NumPy, and it seamlessly integrates with NumPy arrays. This integration enables efficient data processing and analysis.

**Importing necessary libraries:-**

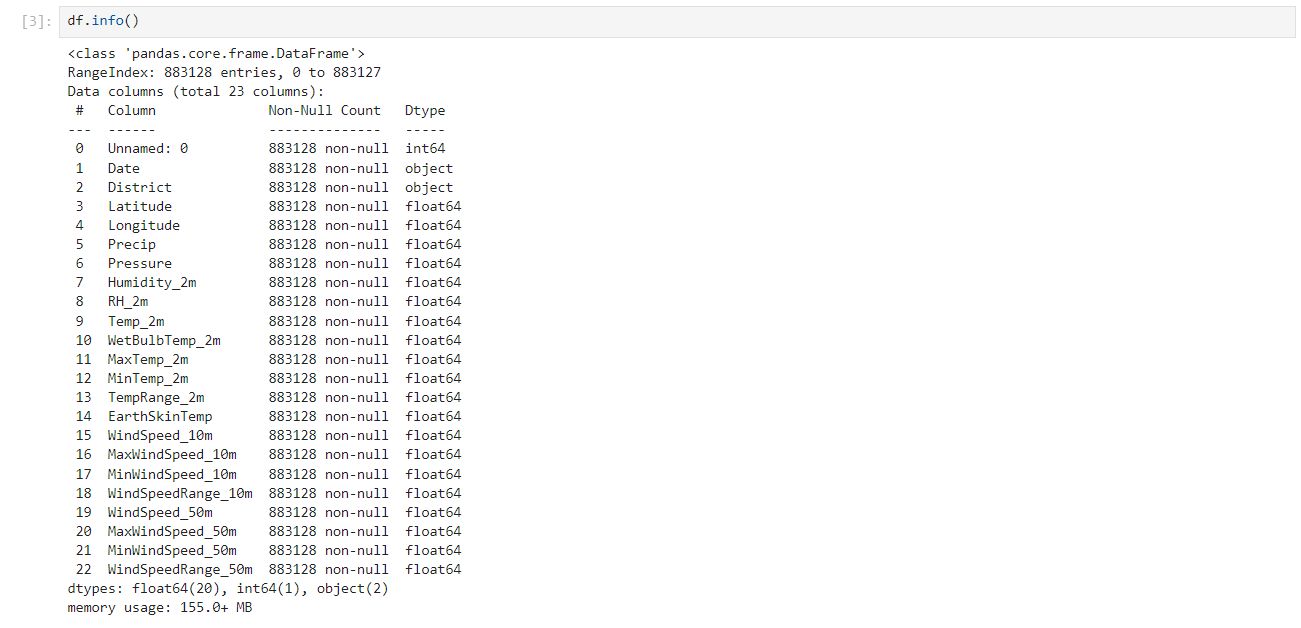
****

**Reading the data and converting to csv:-**

Website link:- https://www.kaggle.com/datasets/saimondahal/nepal-daily-climate-1981-2019

****

**Contents of the dataset:-**

****

**Additional Information about columns:-**

**Date:** The date of the recorded climate data.

**District:** The geographical region within Nepal for which the climate data is recorded. There are 62 districts in this dataset.

**Latitude and Longitude:** The geographical coordinates of the district location, specifying its position on the Earth's surface.

**Precip:** Precipitation, measured in millimeters per day (mm/day), represents the amount of rainfall or snowfall.

**Pressure**: Surface pressure, measured in kilopascals (kPa), represents the atmospheric pressure at the Earth's surface.

**Humidity\_2m:** Specific Humidity at 2 meters above the ground, measured in grams per kilogram (g/kg).

**RH\_2m:** Relative Humidity at 2 meters above the ground, measured as a percentage (%).

**Temp\_2m:** Temperature at 2 meters above the ground, measured in degrees Celsius (°C).

**WetBulbTemp\_2m:** Wet Bulb Temperature at 2 meters above the ground, measured in degrees Celsius (°C).

**MaxTemp\_2m:** Maximum Temperature at 2 meters above the ground, measured in degrees Celsius (°C).

**MinTemp\_2m:** Minimum Temperature at 2 meters above the ground, measured in degrees Celsius (°C).

**TempRange\_2m:** Temperature Range at 2 meters above the ground, measured in degrees Celsius (°C).

**EarthSkinTemp:** Earth Skin Temperature, measured in degrees Celsius (°C).

**WindSpeed\_10m:** Wind Speed at 10 meters above the ground, measured in meters per second (m/s).

**MaxWindSpeed\_10m:** Maximum Wind Speed at 10 meters above the ground, measured in meters per second (m/s).

**MinWindSpeed\_10m:** Minimum Wind Speed at 10 meters above the ground, measured in meters per second (m/s).

**WindSpeedRange\_10m:** Wind Speed Range at 10 meters above the ground, measured in meters per second (m/s).

**WindSpeed\_50m:** Wind Speed at 50 meters above the ground, measured in meters per second (m/s).

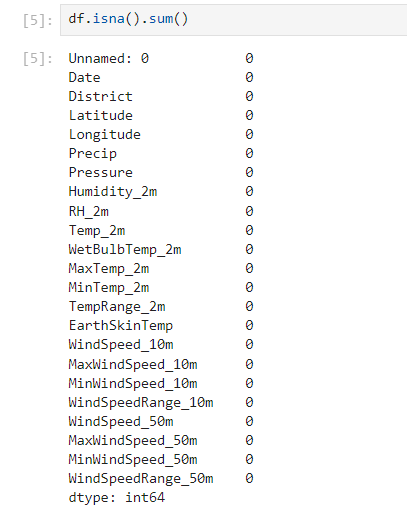
**MaxWindSpeed\_50m:** Maximum Wind Speed at 50 meters above the ground, measured in meters per second (m/s).

**MinWindSpeed\_50m:** Minimum Wind Speed at 50 meters above the ground, measured in meters per second (m/s).

**WindSpeedRange\_50m:** Wind Speed Range at 50 meters above the ground, measured in meters per second (m/s).

This dataset provides a comprehensive set of climate parameters for each day across the 62 districts of Nepal, making it valuable for climate analysis and research.

**Checking for null values:-**

****

**Methods to Fill NaN values:-**

**a. Imputation:**

- Imputation involves replacing NaN values with estimated or calculated values based on the characteristics of the dataset.

- Common imputation methods include mean, median, mode, or using more advanced techniques like regression or machine learning models.

**b. Forward Fill and Backward Fill:**

- For time series data, NaN values can be filled using the values from the previous (forward fill) or subsequent (backward fill) time steps.

**c. Customized Value Replacement:**

- NaN values can be replaced with a specific constant or a value derived from domain knowledge if it makes sense in the context of the data.

**d. Interpolation:**

- Interpolation methods can be applied to estimate missing values based on the existing values in a series. Common interpolation techniques include linear interpolation and spline interpolation.

**2. Removing NaN Values:**

**a. Row Deletion:**

- Entire rows containing NaN values can be removed. This is suitable when the missing values are relatively few and do not significantly affect the overall dataset.

**b. Column Deletion:**

- If an entire column is populated with NaN values or contributes little information, it may be reasonable to drop that column from the DataFrame.

**c. Threshold-based Removal:**

- Rows or columns can be removed based on a threshold of non-NaN values. For example, remove rows with more than a certain percentage of missing values.

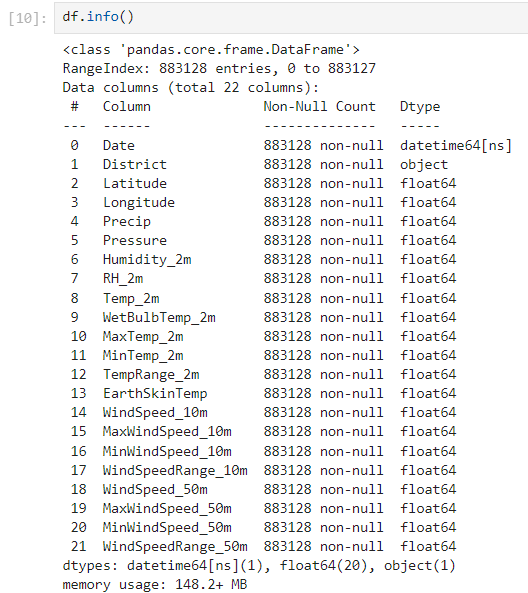
**Dropped the unnecessary columns:-**

****

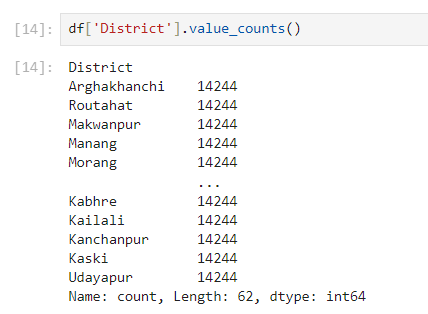
**Changed the datatype of the columns date:-**

****

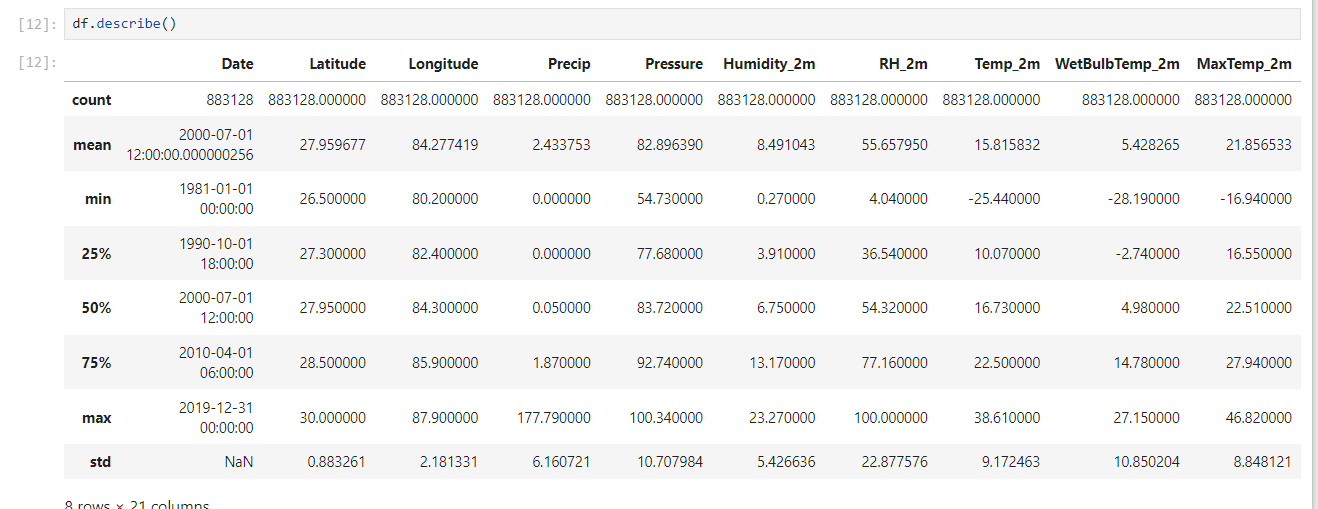
**Verifying changes made:-**

****

**Counting unique values for specific columns:-**

****

**Describing the data:-**

****

**Working of describe() function :-**

Generate descriptive statistics.

Descriptive statistics include those that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding ``NaN`` values.

Analyzes both numeric and object series, as well as ``DataFrame`` column sets of mixed data types. The output will vary depending on what is provided. Refer to the notes

below for more detail.

**Parameters:-**

percentiles : list-like of numbers, optional

The percentiles to include in the output. All should fall between 0 and 1. The default is

``[.25, .5, .75]``, which returns the 25th, 50th, and 75th percentiles.

include : 'all', list-like of dtypes or None (default), optional

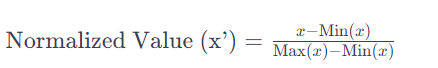
A white list of data types to include in the result.

**Creating Dummy Indicator Columns:-**

****

**Normalization:-**

Normalization is a data preprocessing technique used to scale numerical features to a standard range, making them comparable and preventing certain features from dominating others. The normalization formula is often expressed as follows:

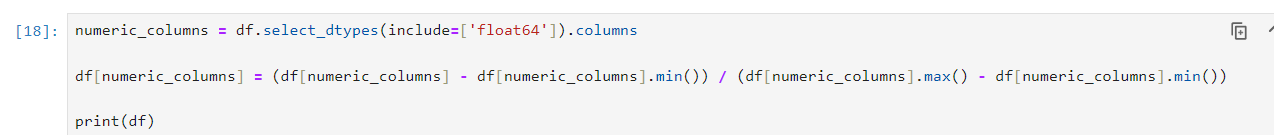
****

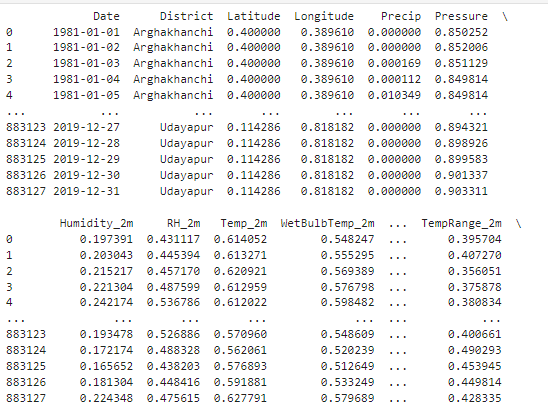
Here

* x’ is the normalized value
* x is the original value
* Min(x) is the minimum value of the feature x
* Max(x) is the maximum value of the feature x

The formula scales the values to a range between 0 and 1. The process involves subtracting the minimum value from each data point and then dividing by the range (difference between the maximum and minimum values).

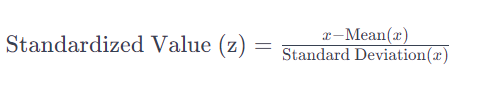
This normalization technique is known as Min-Max scaling. It preserves the relative relationships between values while ensuring that the entire range of the data is used. It is particularly useful when the features in a dataset have different scales, as it prevents features with larger scales from dominating the learning process in machine learning models.





**Standardization:-**

Standardization is a data preprocessing technique used to transform numerical features to have a mean of 0 and a standard deviation of 1. This is achieved by subtracting the mean and dividing by the standard deviation for each data point. The standardization formula is as follows:

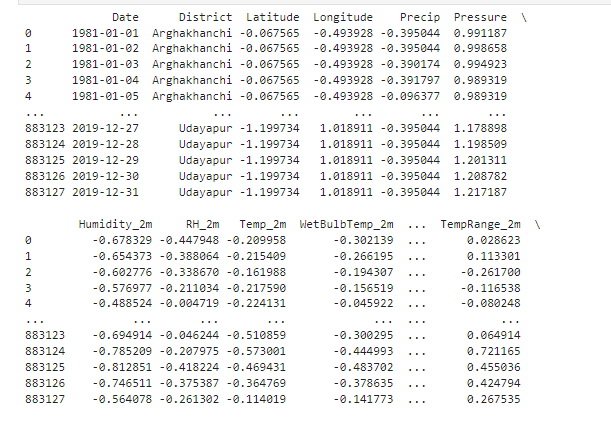


Here:

* Z is the standard deviation
* x is the original value
* Mean(x) is the mean of the feature x
* Standard Deviation(x) is the standard deviation of the feature x.

The process of standardization ensures that the transformed data has a mean of 0 and a standard deviation of 1. This is particularly beneficial for machine learning algorithms that assume a standard normal distribution of features. Standardization is useful when the features in a dataset have different scales, and it helps prevent features with larger scales from dominating the learning process.





**Outliers:-**

**Identifying Outliers using Z-Score:**

A common threshold to identify outliers is a Z-score greater than a certain value (e.g., 3 or -3). If the absolute Z-score of a data point is greater than this threshold, it is considered an outlier. The choice of threshold depends on the level of strictness desired in outlier detection.-

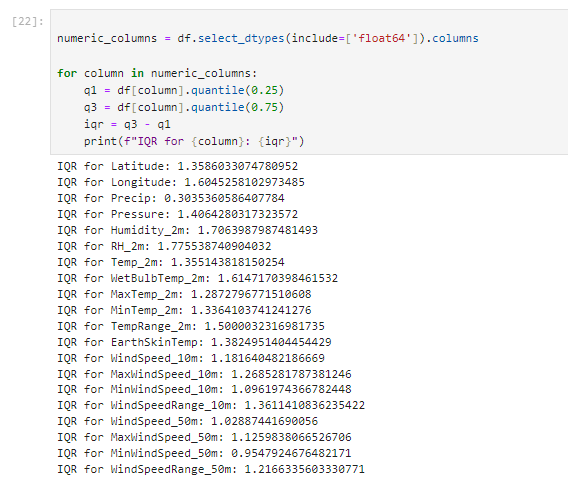
**Why Detect Outliers:**

* Influence on Statistics: Outliers can significantly affect the mean and standard deviation, leading to skewed statistics. Removing outliers helps in obtaining more representative summary statistics.
* Impact on Machine Learning Models: Outliers can distort the learning process of machine learning models, particularly those sensitive to the scale of features. Identifying and handling outliers contribute to the model's robustness.

**Handling Outliers:**

Once identified, outliers can be handled through various methods:

* Removal: Simply removing the data points identified as outliers.
* Transformation: Applying mathematical transformations to the data, such as logarithmic transformation, to reduce the impact of outliers.
* Imputation: Replacing outliers with a more representative value, often based on statistical measures.



**Conclusion :-**

In conclusion, effective data preprocessing is a fundamental aspect of any data analysis or machine learning project. It lays the foundation for robust and reliable results by addressing issues such as missing values, outliers, and inconsistent data formats. This documentation has provided a comprehensive overview of key preprocessing techniques, with a focus on practical implementation using Python's pandas, NumPy, and scikit-learn libraries.