**Title : Energy consumption and Prediction**

**Importing Required Libraries**

A close up of a computer screen

Description automatically generated

**Pandas** used for data manipulation

**NumPy** for numerical operation

**Matplotlib.pyplot** is commonly used to create static, animated, and interactive visualizations in Python

**Seaborn** is built on top of Matplotlib and provides a higher-level interface to create attractive and informative statistical graphics

**Loading the Dataset**



The dataset hpc.txt is loaded using the read\_csv function from pandas. The dataset uses a semicolon (;) as a separator between values, which is specified using the sep=";" argument

**Inspecting the Dataset**

**df.head()** displays the first 5 rows of the dataset

A screenshot of a computer

Description automatically generated

**df.tail()** displays the last 5 rows of the dataset

A screenshot of a computer

Description automatically generated

**df.describe** gives statistical summaries of numerical columns, such as count, mean, and standard deviation.

A screenshot of a computer

Description automatically generated

**df.shape** shows the number of rows and columns

A screenshot of a computer

Description automatically generated

**df.describe(include=object)** displays the object columns

A screenshot of a computer

Description automatically generated

**df.nunique()** returns the number of unique values in each column

A screenshot of a computer

Description automatically generated

**df.info()** gives a summary of the dataset, including column names, data types

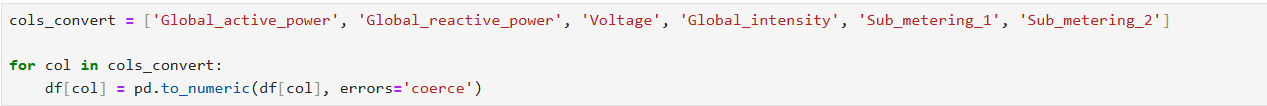
A screenshot of a computer code

Description automatically generated

**Data Preprocessing**

As we can see all columns are object type but they are float values . So ,Convert datatypes of columns which are in string type to float type

**Converting Columns to Numeric**



Columns that you want to convert into numeric data types in your DataFrame (df) are listed in the cols\_convert list.

**For loop:** The loop goes through each column in the cols\_convert list one by one.  
**pd.to\_numeric():** This pandas function attempts to convert the values in each column to a numeric data type (integer or float).  
**df[col]:** This selects the column col from the DataFrame df.

If an error arises during the conversion of non-numeric values into numbers, these very values will become a NaN as flags on proper completion instead of causing an error.

**Creating a Datetime Column**

Date and time datatype is different than float , we combine both and set to datetime datatype



Combine Date and Time columns into one column called Datetime.The string is cast into datetime format by pd.to\_datetime().  
The format :  
%d/%m/%Y is for day, month, and year and %H:%M:%S is for hours, minutes, and seconds  
This deletes both now-useless Date and Time clauses immediately from our DataFrame now that they have been merged into Datetime.  
Hence axis=1 indicates column-wise deletion, while inplace=True implies that the change will occur directly within the DataFrame.

**Checking for Missing Values**

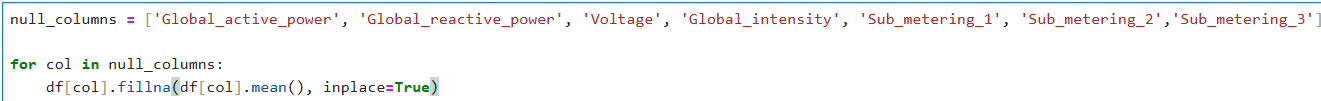
A screenshot of a computer program

Description automatically generated

True indicates missing value and False indicate no missing values.All columns except datetime have missing values.

**Handling Missing Values**

We can deal with missing values by fill with the mean , median of the column or delete row which have null values.



**null\_columns** columns that may have missing (null) values, and you want to fill those missing spots

**For loop:** This goes through each column in the null\_columns list.

**df[col].fillna(df[col].mean(), inplace=True):**

* **fillna(df[col].mean()):** This replaces Null values with Mean value
* **inplace=True:** This directly updates the Data Frame.