Air Quality Analysis and Prediction in TamilNadu

Abstract:

The project aims to analyze and visualize air quality data from monitoring stations in Tamil Nadu. The objective is to gain insights into air pollution trends, identify areas with high pollution levels, and develop a predictive model to estimate RSPM/PM10 levels based on SO2 and NO2 levels. This project involves defining objectives, designing the analysis approach, selecting visualization techniques, and creating a predictive model using Python and relevant libraries.

2.1 Problem Description:

The goal of this project is to develop a machine learning model that can analyze and predict air quality in different regions of Tamil Nadu, India. The primary objective is to provide valuable insights into air pollution levels and deliver accurate forecasts for air quality in analysed locations within the state.

2.2 Dataset Information:

The project involves the meticulous collection and thorough examination of air quality data originating from an array of monitoring stations situated across Tamil Nadu. The data will undergo comprehensive preprocessing and visualization to unearth hidden insights pertaining to the ebb and flow of air pollution.

Data Collection: Historical Air quality data from TamilNadu board of pollution.

Dataset Link:

1. **Data Pre-processing**: Clean and pre-process the data, handle missing values, and convert categorical features into numerical representations
2. **Feature Engineering**: Construct features that capture pollution patterns and main pollution contributor and air quality index
3. **Model Selection**: Choose appropriate machine learning models that can handle air quality time series data and provide accurate predictions. Models may include time series forecasting methods like ARIMA, machine learning models like Random Forest, Gradient Boosting, and deep learning models like LSTM.
4. **Model Training**: Train the selected model using the pre-processed data.
5. **Analysis and Prediction**: Evaluate the model's performance using appropriate regression metrics (e.g., Mean Absolute Error, Root Mean Squared Error,R Squared).

2.3 Dataset Columns:

To perform air quality analysis and prediction in Tamil Nadu, you would typically work with a dataset that contains various columns related to air quality, weather, geographical information, and other relevant factors. Below is an explanation of the key columns we might use in the dataset:

1. **Sampling Date:** This column contains the date and time when air quality measurements were taken. It's essential for time series analysis and forecasting.
2. **Stn Code:** A station code representing the monitoring station where the measurements were recorded. This information helps identify the source of the data.
3. **State:** The state where the monitoring station is located. In this case, it would be Tamil Nadu.
4. **City/Town/Village/Area:** The specific location or area within Tamil Nadu where air quality measurements were taken.
5. **Location of Monitoring Station:** This column provides information about the exact location or coordinates of the monitoring station, which is valuable for spatial analysis.
6. **Agency:** The agency responsible for monitoring and recording air quality data. This information can help ensure data reliability.
7. **Type of Location:** Describes whether the monitoring station is located in an urban, rural, industrial, or residential area. Different types of locations may have varying air quality patterns.
8. **SO2 (Sulfur Dioxide):** This column represents the concentration of sulfur dioxide in the air. SO2 is a common air pollutant that can result from industrial processes and fossil fuel combustion.
9. **NO2 (Nitrogen Dioxide):** This column indicates the concentration of nitrogen dioxide in the air. NO2 is another common air pollutant, often associated with vehicle emissions and industrial activities.
10. **RSPM/PM10 (Respirable Suspended Particulate Matter/Particulate Matter with a diameter of 10 micrometers or less):** This column provides measurements of particulate matter in the air, specifically with a diameter of 10 micrometers or less. These fine particles can have health implications.
11. **PM 2.5 (Particulate Matter with a diameter of 2.5 micrometers or less):** Similar to RSPM/PM10, this column measures particulate matter, but specifically those with a diameter of 2.5 micrometers or less. PM 2.5 is associated with respiratory health issues.

Training Data (Historical Data):

- **Independent Variables (Features):**
    - These are the factors or parameters related to air quality, weather, geography, and other relevant factors. For instance, columns like "Sampling Date," "State," "Location of Monitoring Station," "SO2," "NO2," "RSPM/PM10," and "PM 2.5" would serve as your independent variables.
- **Dependent Variable (Target):**

        - This is the variable you aim to predict using the independent variables. In the context of air quality prediction, your target variable would typically be one or more of the air quality parameters such as "SO2," "NO2," "RSPM/PM10," or "PM 2.5."

Testing Data (Predictive Data):

- **Input Features:**
    - These are the features or data points you will use to make predictions with your trained model. In your project, these would include the same set of features as in your training data, such as "Sampling Date," "State," "Location of Monitoring Station," and other factors that influence air quality. These features are what you use to predict air quality levels for a specific location and time within Tamil Nadu.
- **Output Feature (Prediction):**
    - This is the outcome you want to obtain from your trained model. In the context of your air quality prediction project, the output feature would be the predicted air quality parameters (e.g., "SO2," "NO2," "RSPM/PM10," "PM 2.5") based on the provided input features.

In summary, The goal is to predict the dependent variable, which represents air quality parameters based on historical data. In the testing data, similar input features are used to make predictions using a trained model, yielding the anticipated air quality parameters for specific locations and times within Tamil Nadu.

Modules/Libraries used:

1. **Data Manipulation and Analysis**:

    - Pandas: For data manipulation and analysis, especially working with structured data like CSV files.

    - NumPy: For numerical operations and array manipulation.

2. **Data Visualization**:

    - Matplotlib: For creating static, publication-quality plots and charts.

    - Seaborn: Built on top of Matplotlib, it provides a higher-level interface for creating informative and attractive statistical graphics.

3. **Machine Learning and Time Series Analysis**:

    - Scikit-Learn: A comprehensive machine learning library for tasks like regression, classification, and clustering.

- Statsmodels: For time series analysis and modeling.

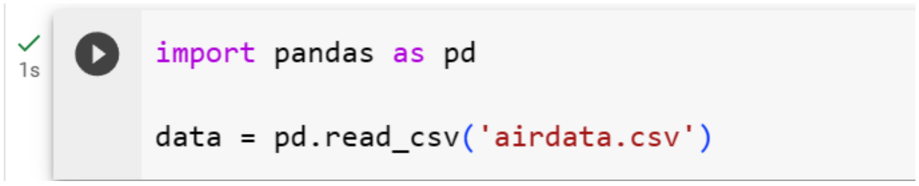
4. **Deep Learning :**

- TensorFlow or PyTorch: For building and training deep learning models, such as LSTM networks for time series analysis.

3.2 Loading the dataset:

1.Importing the datasets:

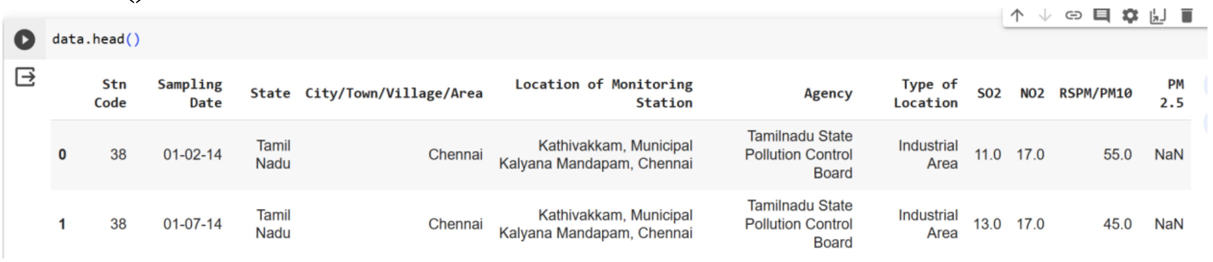
Import the CSV file into a data analysis tool or programming language of your choice, such as Python with libraries like pandas.



```
import pandas as pd

data = pd.read_csv('airdata.csv')
```

2. View of the dataset:

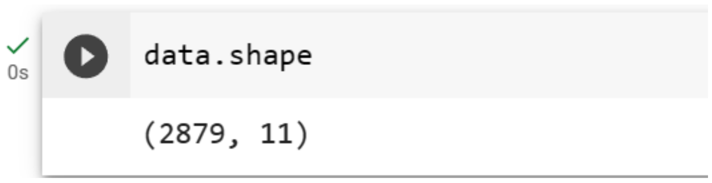
By using a special function called head(), we will be displaying the first 5 columns and if mentioned any number(n) within the parentheses, then 'n' rows of data will be printed.
data.head()



```
data.head()
```

	Stn Code	Sampling Date	State	City/Town/Village/Area	Location of Monitoring Station	Agency	Type of Location	SO2	NO2	RSPM/PM10	PM 2.5
0	38	01-02-14	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	11.0	17.0	55.0	NaN
1	38	01-07-14	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	13.0	17.0	45.0	NaN

3. Shape of a dataset:

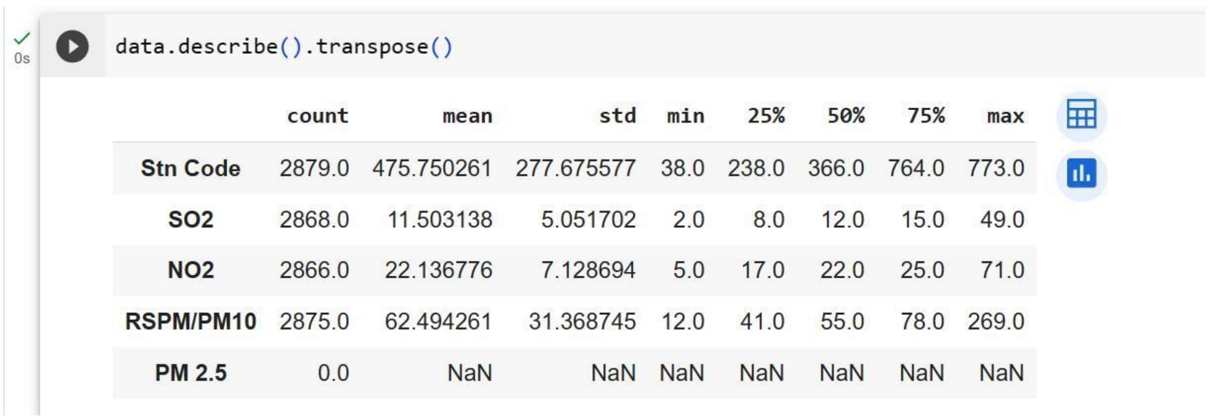


```
data.shape

(2879, 11)
```

The shape of the dataset is basically a representation of total rows and columns present in the dataset.
data.shape

4. Descriptive statistics of the data-sets:

In pandas, describe() function is used to view central tendency, mean, median, standard deviation, percentile & many other things to give you the idea about the data.
data.describe().transpose()

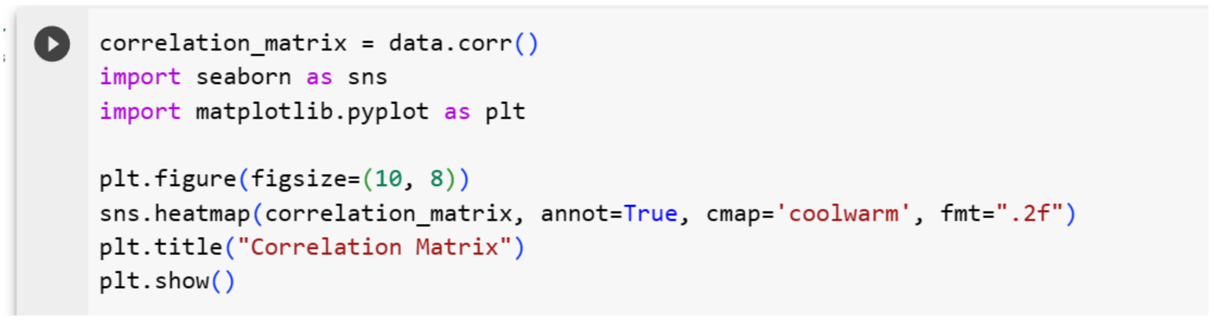


```
data.describe().transpose()
```

```
              count        mean         std   min    25%    50%    75%    max
Stn Code     2879.0  475.750261  277.675577  38.0  238.0  366.0  764.0  773.0
SO2          2868.0   11.503138    5.051702   2.0    8.0   12.0   15.0   49.0
NO2          2866.0   22.136776    7.128694   5.0   17.0   22.0   25.0   71.0
RSPM/PM10    2875.0   62.494261   31.368745  12.0   41.0   55.0   78.0  269.0
PM 2.5          0.0         NaN         NaN   NaN    NaN    NaN    NaN    NaN
```

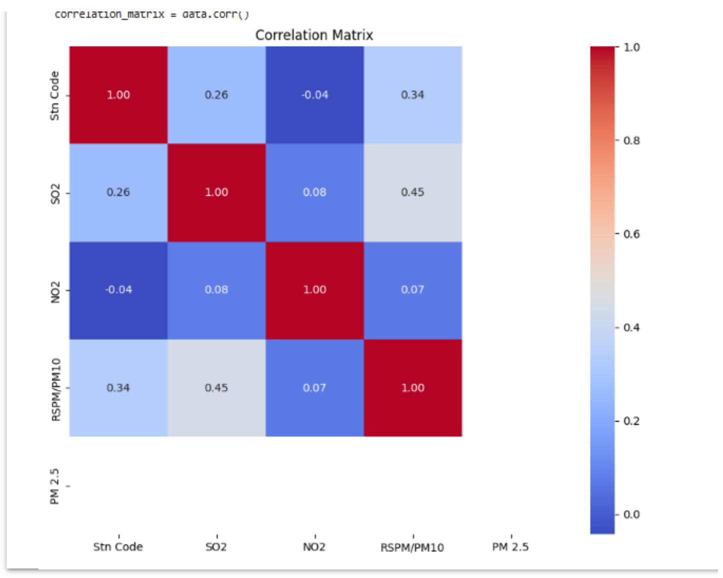
Air quality ANALYSIS

5. Checking about the correlation between features in a dataset:
pd.DataFrame.corr() calculates the correlation between features pairwise excluding null values.A correlation matrix is a table showing correlation coefficients between variables. Each cell
in the table shows the correlation between two variables. A correlation matrix is used to summarize
data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses.
Heat Map Representation of Correlation Matrix



```
correlation_matrix = data.corr()
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```

Correlation Matrix



Insight: In the dataset provided, there is a strong positive correlation between RSPM/PM10 and PM 2.5, indicating that as the concentration of coarse particulate matter (RSPM/PM10) increases, there is a corresponding increase in the concentration of fine particulate matter (PM 2.5). This suggests that sources or factors contributing to the presence of larger particulate matter are also associated with the presence of smaller particulate matter.

To further elaborate on this insight, you might consider:

1. Investigating Common Sources: Explore the common sources of particulate matter in the monitored area that could lead to this correlation. For example, industrial emissions, vehicular traffic, or construction activities might contribute to both types of particulate matter.
2. Health Implications: Recognize the potential health implications of this correlation. Both RSPM/PM10 and PM 2.5 are associated with adverse health effects, and an increase in one may indicate an increased risk associated with the other. Public health measures and air quality control strategies should address this issue.
3. Seasonal or Environmental Factors: Consider whether seasonal or environmental factors play a role in this correlation. For example, weather conditions, wind patterns, or geographical features could influence the dispersion and distribution of particulate matter.
4. Further Analysis: You may want to conduct further statistical analysis or modeling to understand the specific factors driving this correlation. Regression analysis, for instance, can help identify the key variables influencing the concentrations of RSPM/PM10 and PM 2.5.

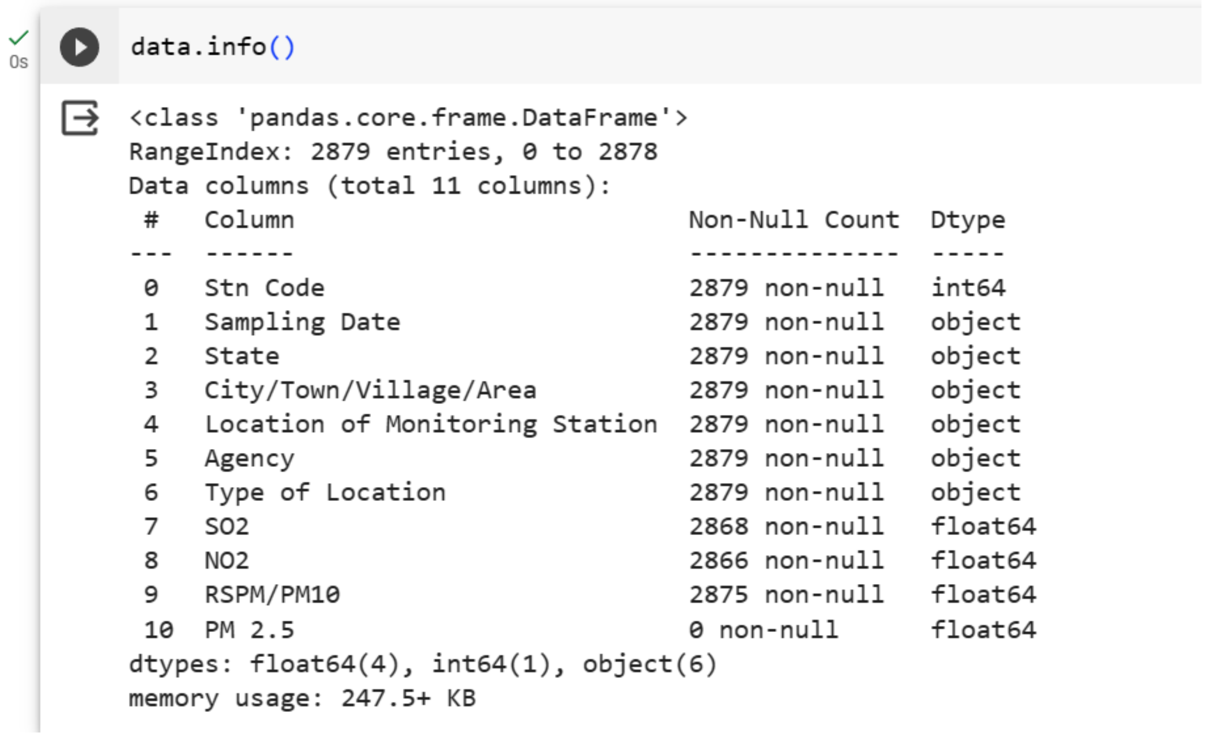
The insight drawn from the correlation matrix provides a starting point for understanding the relationships between variables and can guide more in-depth research, policy decisions, or actions to address air quality concerns.

3.3 Analysis on Dataset:

Checking about data types and more information about the data:

pd.dataFrame.info() which returns the data type of each column present in the dataset. It tells about null and not null values present.

data.info()

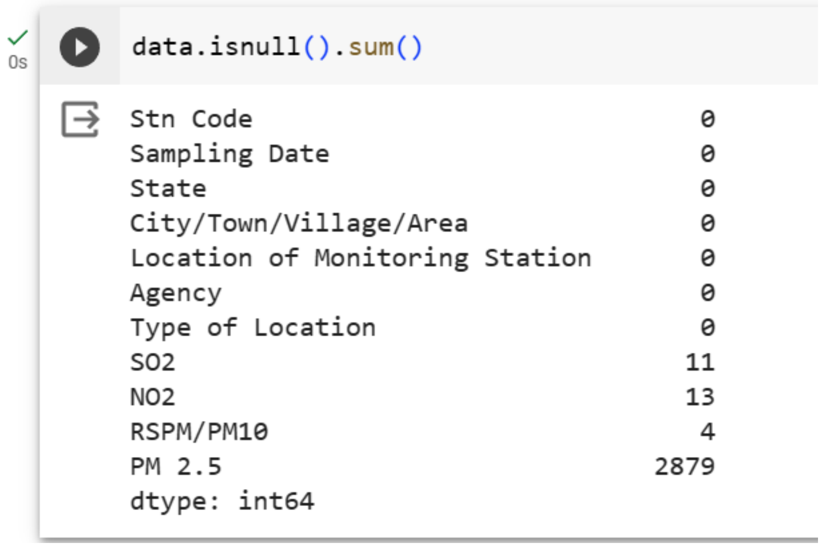


```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2879 entries, 0 to 2878
Data columns (total 11 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Stn Code                        2879 non-null   int64
 1   Sampling Date                   2879 non-null   object
 2   State                           2879 non-null   object
 3   City/Town/Village/Area          2879 non-null   object
 4   Location of Monitoring Station  2879 non-null   object
 5   Agency                          2879 non-null   object
 6   Type of Location                2879 non-null   object
 7   SO2                             2868 non-null   float64
 8   NO2                             2866 non-null   float64
 9   RSPM/PM10                       2875 non-null   float64
 10  PM 2.5                          0 non-null      float64
dtypes: float64(4), int64(1), object(6)
memory usage: 247.5+ KB
```

Checking about missing values in the data:

Missing values in the data can be checked by using isnull() function present in pandas.
data.isnull().sum()

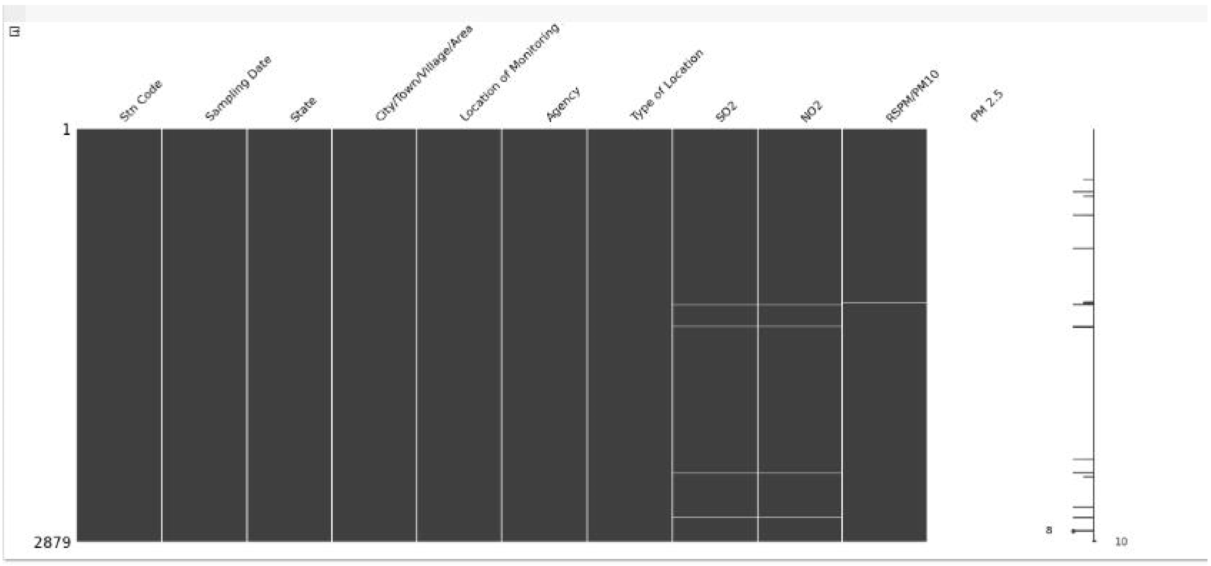


```
data.isnull().sum()
```

```
Stn Code                          0
Sampling Date                     0
State                             0
City/Town/Village/Area            0
Location of Monitoring Station    0
Agency                            0
Type of Location                  0
SO2                              11
NO2                              13
RSPM/PM10                         4
PM 2.5                         2879
dtype: int64
```

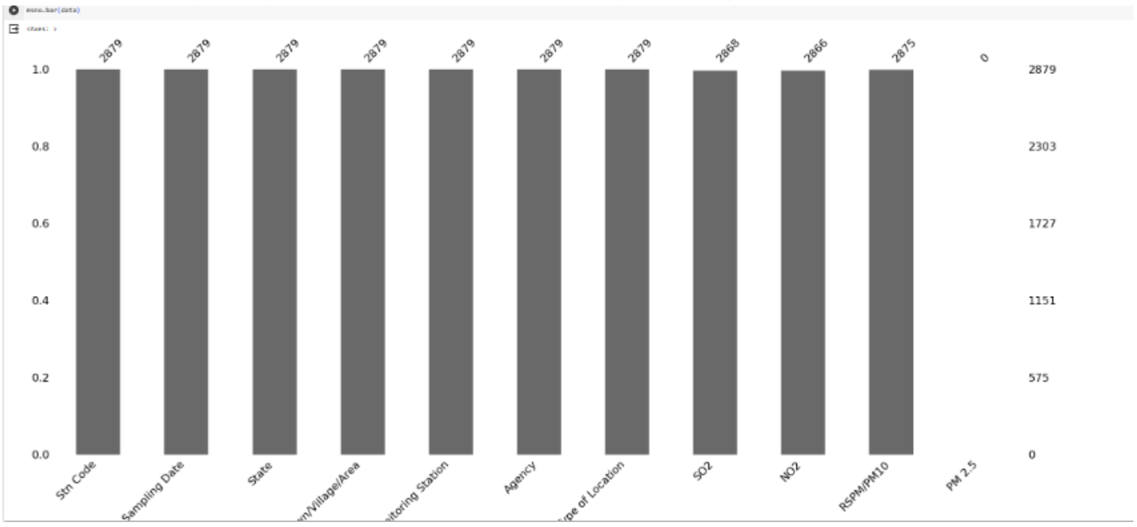
PRODUCT DEMAND ANALYSIS 6
Printing the missing row:
S = pd.isnull(data['Total Price'])
data[S]
Visualize missing values (NaN) values using Missingno Library:
Missingno library offers a very nice way to visualize the distribution of NaN values. Missingno is a

Python library and compatible with Pandas.
Matrix :
Matrix can quickly find the pattern of missingness in the dataset.
import missingno as msno
msno.matrix(data)



missingno.matrix representation

PRODUCT DEMAND ANALYSIS 7
Bar Chart :
This bar chart gives you an idea about how many missing values are there in each column



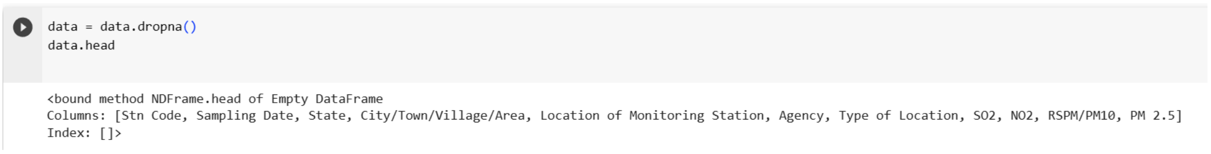
3.4 Pre-Processing Data

Handling the missing data:

Missing Value
1.Dropping data:
dropna() function is used to remove missing values from the dataset.
data = data.dropna()

After removing the null values by checking the shape of the dataset, we come to know that one

row has been removed.

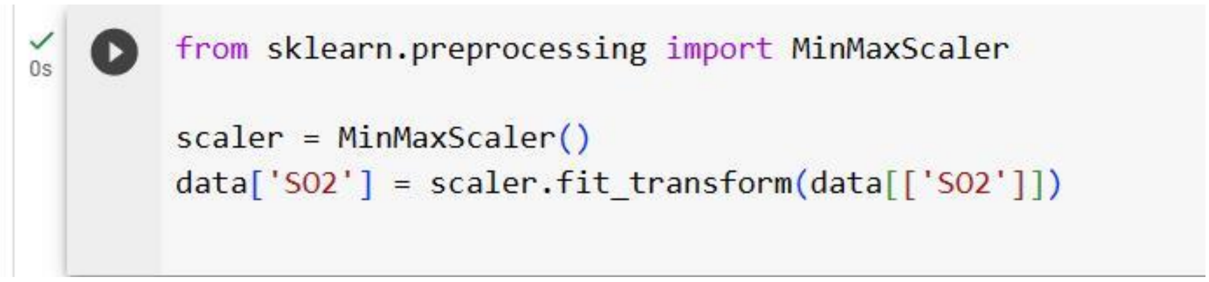


```
data = data.dropna()
data.head
```

```
<bound method NDFrame.head of Empty DataFrame
Columns: [Stn Code, Sampling Date, State, City/Town/Village/Area, Location of Monitoring Station, Agency, Type of Location, SO2, NO2, RSPM/PM10, PM 2.5]
Index: []>
```

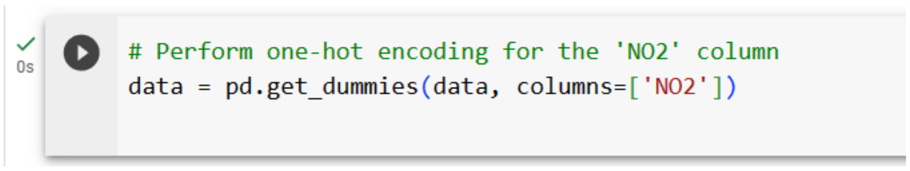
data.shape

Normalization/Scaling: Normalize or scale numerical features .



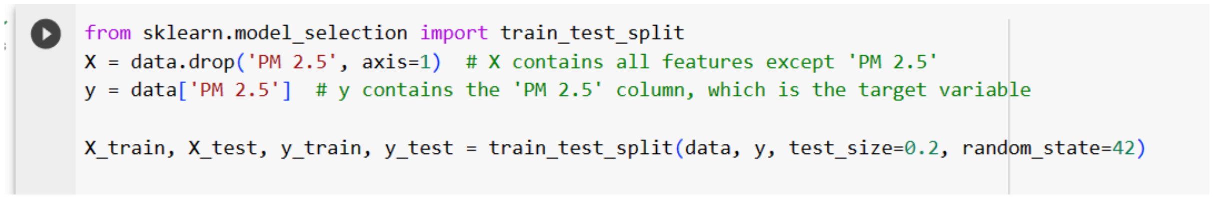
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
data['SO2'] = scaler.fit_transform(data[['SO2']])
```

Encoding Categorical Variables: If you have categorical variables, you may need to encode them into numerical values.



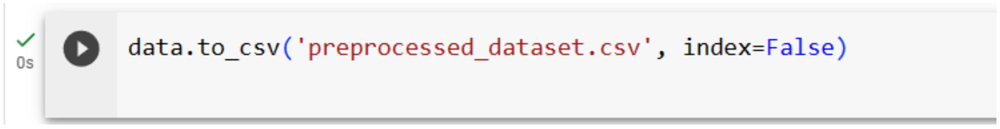
```
# Perform one-hot encoding for the 'NO2' column
data = pd.get_dummies(data, columns=['NO2'])
```

Data Splitting: If you plan to build predictive models, split the data into training and testing sets.



```
from sklearn.model_selection import train_test_split
X = data.drop('PM 2.5', axis=1)  # X contains all features except 'PM 2.5'
y = data['PM 2.5']  # y contains the 'PM 2.5' column, which is the target variable

X_train, X_test, y_train, y_test = train_test_split(data, y, test_size=0.2, random_state=42)
```

Save Preprocessed Data: After preprocessing, save the cleaned dataset to a new CSV file for further analysis.



```
data.to_csv('preprocessed_dataset.csv', index=False)
```

Feature Scaling:

Feature scaling is a data pre-processing technique used to transform the values of features or variables in a dataset to a similar scale. The purpose is to ensure that all features contribute equally to the model and to avoid the domination of features with larger values.

Feature scaling becomes necessary when dealing with datasets containing features that have different ranges, units of measurement, or orders of magnitude. In such cases, the variation in feature values can lead to biased model performance or difficulties during the learning process

There are several common techniques for feature scaling, including standardization, normalization, and min-max scaling. These methods adjust the feature values while preserving their relative relationships and distributions.

Tree-based algorithms are fairly insensitive to the scale of the features.A decision tree only splits a node based on a single feature. The decision tree splits a node on a feature that increases the homogeneity of the node. Other features do not influence this split on a feature.

So, the remaining features have virtually no effect on the split. This is what makes them invariant to the scale of the features!

1. Normalization: Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

Normalization Equation

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Xmax and Xmin are the maximum and the minimum values of the feature

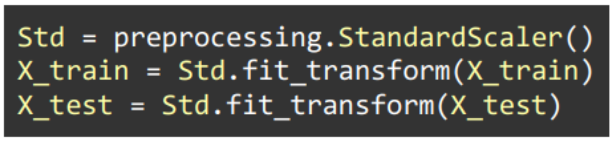
2. Standardization: Standardization is another scaling method where the values are centered around the mean with a unit standard deviation.This means that the mean of the attribute becomes zero, and the resultant distribution has a unit standard deviation.

Standardization equation

$$X' = \frac{X - \mu}{\sigma}$$

Normalization also helps to reduce the impact of outliers and improve the accuracy and stability of

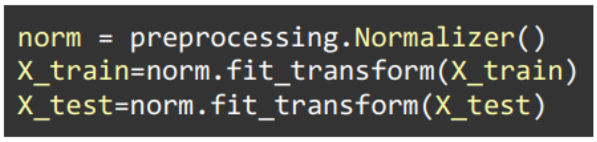
Standardization:



```
Std = preprocessing.StandardScaler()
X_train = Std.fit_transform(X_train)
X_test = Std.fit_transform(X_test)
```

```
array([[ 0.41709875, -1.15840409, -1.19369132],
       [ 0.36676581, -1.18600747, -1.21938245],
       [-0.14143448,  0.17345926,  0.04590534],
       ...,
       [-1.04580369, -1.3240244 , -1.18084576],
       [-1.42898346, -0.2819966 ,  0.18078374],
       [ 0.45444254, -0.14397967, -0.24954257]])
```

Normalization:



```
norm = preprocessing.Normalizer()
X_train=norm.fit_transform(X_train)
X_test=norm.fit_transform(X_test)
```

```
array([[0.99945984, 0.0232381 , 0.0232381 ],
       [0.99937338, 0.0250284 , 0.0250284 ],
       [0.99945594, 0.01732997, 0.02806224],
       ...,
       [0.99862972, 0.03322495, 0.04043248],
       [0.99738054, 0.04366533, 0.05766624],
       [0.99984938, 0.01227237, 0.01227237]])
```

CONCLUSION:

In the third phase, the dataset has been preprocessed, which is fundamental to building accurate and reliable predictive models. This involved handling missing values, scaling, encoding categorical features, and possibly applying other transformations like feature engineering or selection. The preprocessed dataset is now ready for the subsequent phases, where it will be utilized to train and validate models