Air Quality Analysis In TamilNadu

To initiate our project, which involves analyzing and developing a solution, we'll start by loading and preprocessing an air quality dataset. In this Python-based example, we will utilize the Pandas library to accomplish this task. We'll assume that the dataset is provided in a CSV file containing air quality information. The code can be adjusted as needed to suit the format of your particular dataset.

LOADING AND PREPROCESSING METHODS:

- IMPORT LIBRARIES
- LOAD THE DATASET
- EXPLORE THE DATASET
- HANDLING MISSING DATA
- DATA CLEANING
- DATA TRANSFORMATION
- FEATURE ENGINEERING
- EXPLORATORY DATA ANALYSIS (EDA)
- SAVE PREPROCESSED DATASET

1. Import Libraries:

Import the necessary libraries in your chosen programming language, such as Python. Common libraries include Pandas for data manipulation, NumPy for numerical operations, Matplotlib or Seaborn for data visualization, and Scikit-Learn for machine learning tasks.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

2. Load the Dataset:

Load your dataset into a Pandas DataFrame or the appropriate data structure.

```
# Importing dataset
data = pd.read_csv('/content/air_quality_dataset.csv')
```

3. Explore the Dataset:

Begin by understanding your dataset. Check the first few rows, data types, summary statistics, and unique values of each column.

data.head() will display the top 5 rows of your DataFrame data, providing you with a quick preview of the data. If you want to see a different number of rows, you can pass the desired number as an argument to the head() method.

```
print(data.head())
```

```
  Stn Code Sampling Date       State City/Town/Village/Area  \
0        38      01-02-14  Tamil Nadu                Chennai
1        38      01-07-14  Tamil Nadu                Chennai
2        38      21-01-14  Tamil Nadu                Chennai
3        38      23-01-14  Tamil Nadu                Chennai
4        38      28-01-14  Tamil Nadu                Chennai

                    Location of Monitoring Station  \
0  Kathivakkam, Municipal Kalyana Mandapam, Chennai
1  Kathivakkam, Municipal Kalyana Mandapam, Chennai
2  Kathivakkam, Municipal Kalyana Mandapam, Chennai
3  Kathivakkam, Municipal Kalyana Mandapam, Chennai
4  Kathivakkam, Municipal Kalyana Mandapam, Chennai

                                  Agency Type of Location
SO2   NO2  \
0  Tamilnadu State Pollution Control Board  Industrial Area
 11.0  17.0
1  Tamilnadu State Pollution Control Board  Industrial Area
 13.0  17.0
2  Tamilnadu State Pollution Control Board  Industrial Area
 12.0  18.0
3  Tamilnadu State Pollution Control Board  Industrial Area
 15.0  16.0
4  Tamilnadu State Pollution Control Board  Industrial Area
 13.0  14.0

   RSPM/PM10  PM 2.5
0       55.0     NaN
1       45.0     NaN
2       50.0     NaN
3       46.0     NaN
4       42.0     NaN
```

data.info() provides details about the dataset, such as column names, data types, non-null counts, and memory usage.

```
print(data.info())
```

```
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
None
```

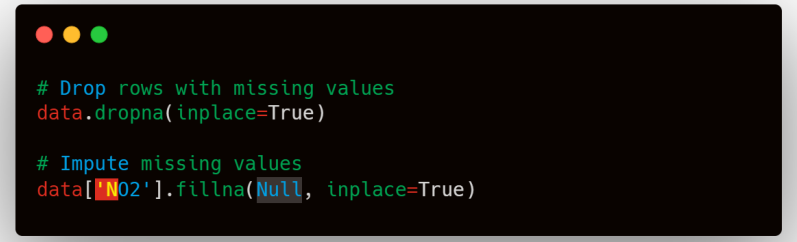
data.describe() provides statistical information such as count, mean, standard deviation, minimum, and maximum values for each numerical column in vour DataFrame.

```
print(data.describe())
```

```
          Stn Code          SO2          NO2    RSPM/PM10  PM
2.5
count  2879.000000  2868.000000  2866.000000  2875.000000
0.0
mean    475.750261    11.503138    22.136776    62.494261
NaN
std     277.675577     5.051702     7.128694    31.368745
NaN
min      38.000000     2.000000     5.000000    12.000000
NaN
25%     238.000000     8.000000    17.000000    41.000000
NaN
50%     366.000000    12.000000    22.000000    55.000000
NaN
75%     764.000000    15.000000    25.000000    78.000000
NaN
max     773.000000    49.000000    71.000000   269.000000
NaN
```

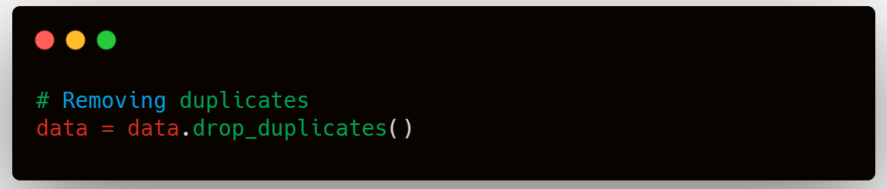
4. **Handling Missing Data:**

Identify and handle missing data in your dataset. You can either remove rows with missing data or impute missing values with appropriate techniques.



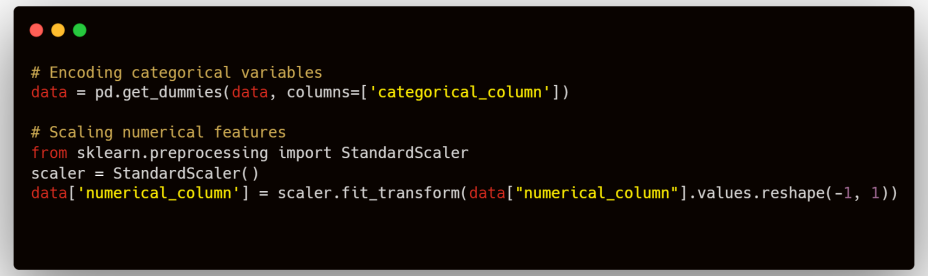
5. **Data Cleaning:**

Clean the data by addressing issues like duplicate records, inconsistent naming, and outliers. This step is specific to your dataset.



6. **Data Transformation:**

Transform the data as needed. This may include encoding categorical variables, scaling features, or converting data types.



pd.get_dummies() is a Pandas function that is used to perform one-hot encoding. It converts categorical variables into a binary/numerical format.

data is the **DataFrame** in which you want to perform one-hot encoding.

columns=['categorical_column'] specify the list of columns you want to one-hot encode.

- You import **StandardScaler** from scikit-learn's preprocessing module.
- You create an instance of the StandardScaler as scaler.
- Then, you use the fit_transform method to standardize the values in the 'numerical_column'. The **fit_transform** method both computes the mean and standard deviation needed for standardization (using the fit part) and applies the transformation to the data.
- The **reshape(-1, 1)** part is used to reshape the data from a 1D array to a 2D array with a single feature. Scikit-learn expects input data to be 2D, so this step is necessary if your 'numerical_column' is originally a 1D series.

7. **Feature Engineering:**

Create new features or transform existing features to improve model performance. Feature engineering depends on the specific problem you are solving.

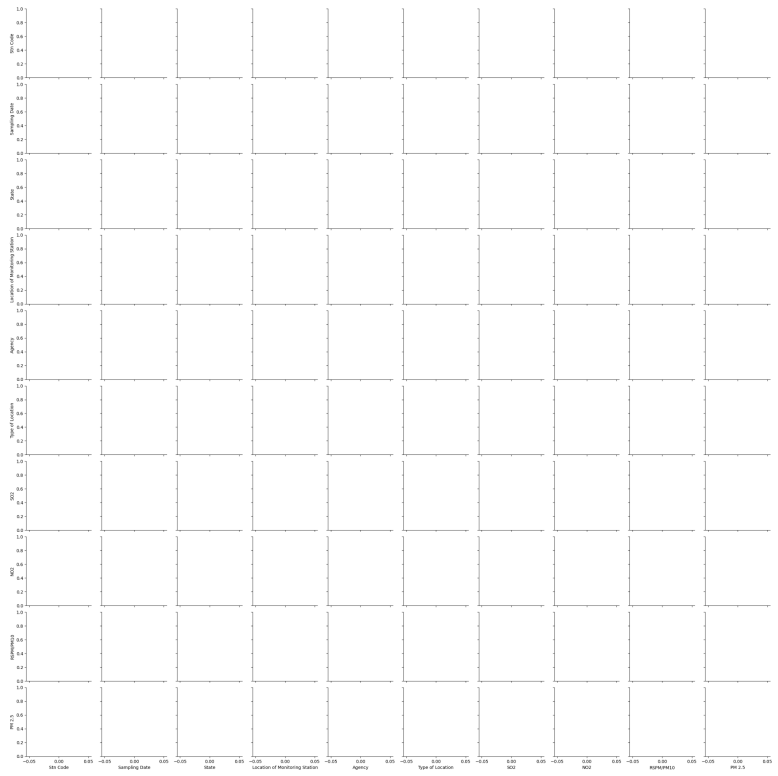
```
# Example: Create a new feature
data['new_feature'] = data['feature1'] * data['feature2']
```

- You're creating a new column in your DataFrame called 'new_feature'.
- The values in this new column are calculated by multiplying the values of 'feature1' and 'feature2'. This operation is performed element-wise, meaning each row in the 'new_feature' column will contain the product of the corresponding values in 'feature1' and 'feature2'.

8. **Exploratory Data Analysis (EDA):**

Visualize and analyze your data to gain insights. This step involves creating plots, histograms, and correlation matrices.

```
# Example EDA
sns.pairplot(data)
plt.show()
```



9. Save Preprocessed Dataset:

Once you have completed all the preprocessing steps, save the preprocessed dataset to a new file to use it for modeling.

```
data.to_csv("preprocessed_data.csv", index=False)
```