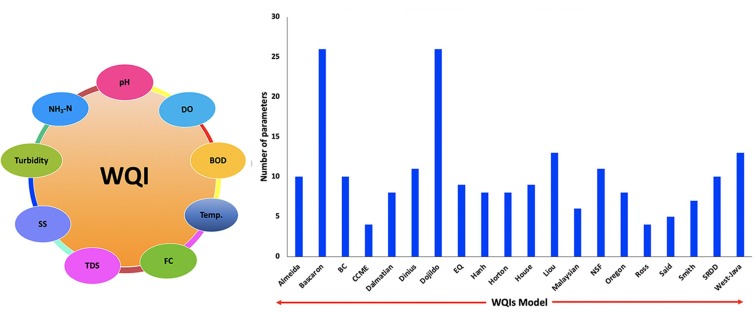
**AIR QUALITY ANALYSIS IN TAMILNADU**

**Phase 3: Development Part 1**

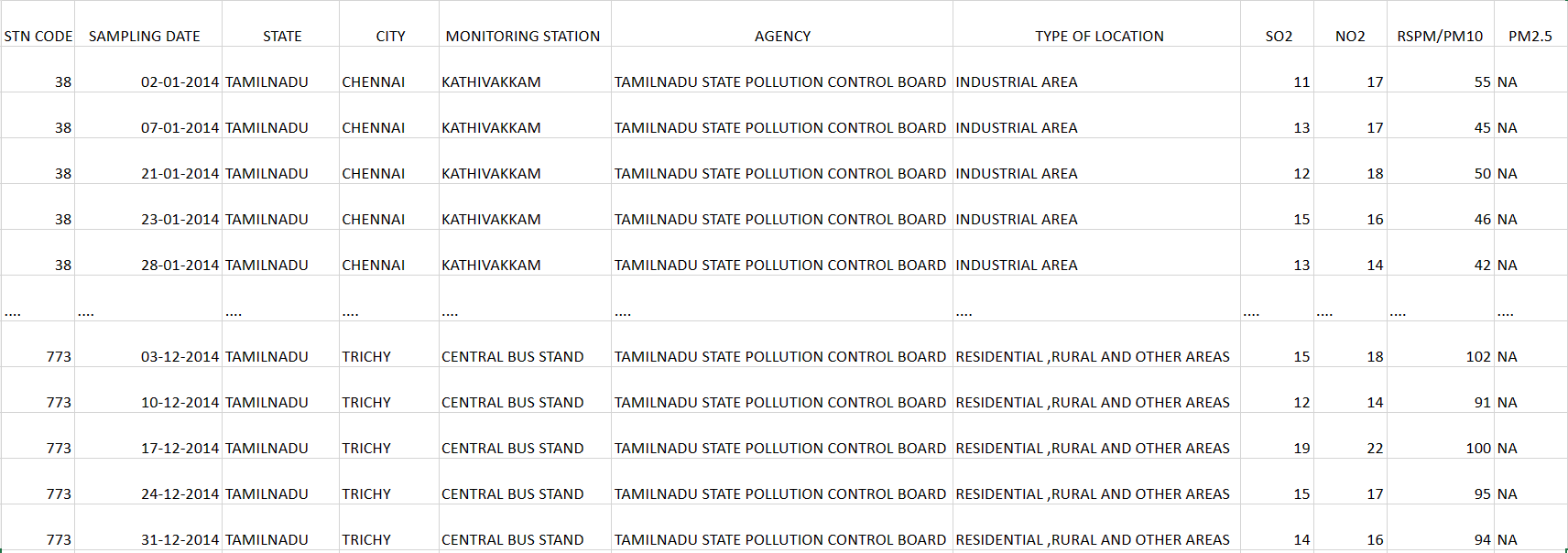


**TOPIC:Begin the analysis by loading and preprocessing the air quality dataset.**

**INTRODUCTION**

* Clean water is a fundamental and irreplaceable resource that lies at the heart of human existence and the sustainability of our planet. Its significance cannot be overstated, as it profoundly impacts every facet of our lives and the health of our environment. This vital resource is not only essential for our survival, but it also plays a pivotal role in promoting public health, economic prosperity, environmental stability, and social equity. In this discussion, we will explore the paramount importance of clean water, examining how it affects our well-being, the environment, and the future of our world.
* Ensuring access to clean water for all is a global challenge, and it involves addressing issues of water scarcity, pollution, infrastructure development, and equitable distribution. It's a critical component of sustainable development and a fundamental human right that underpins numerous aspects of society and the environment.
* This project aims to leverage the capabilities of Python, a versatile and widely-used programming language, along with popular libraries like NumPy, Pandas, Matplotlib, and Seaborn, to gather, analyze, and visualize air quality data. By harnessing the potential of these tools, we can better understand water quality patterns, identify sources of pollution, and contribute to data-driven decision-making for improving Water quality standards.

**Given Dataset:**



The above is the given dataset of Water Quality Analysis in Tamil Nadu in 2014. This dataset consists of 11 columns and 2880 rows.

**NECESSARY STEPS TO FOLLOW**

**1.Import Libraries:**

Start by importing the necessary libraries. Load your dataset into the

Pandasdata frame. And then display the output.

**Program:**

# importing pandas module for data frame

**import**pandas as pd

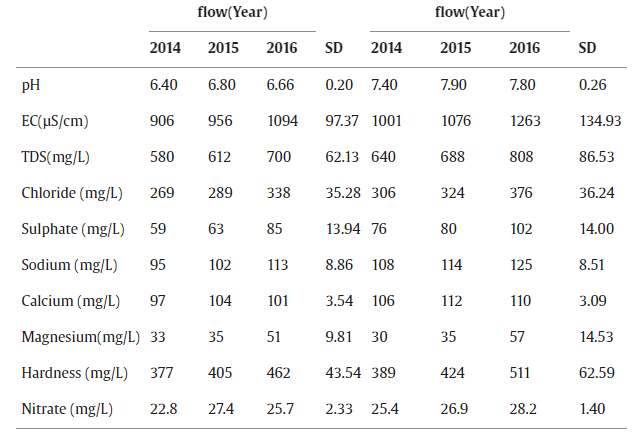
# loading dataset and storing in train variable

train**=**pd.read\_csv('water\_potability.csv')

# display top 5 data

train.head()

**Output:**



**2.Exploratory Data Analysis (EDA)**

Perform EDA to understand your data better. This includes

Checking for missing values, exploring the data's statistics, and

Visualizing it to identify patterns.

**Program:**

# Check for missing values

print(df.isnull().sum())

# Explore statistics

print(df.describe())

# Visualize the data (e.g., histograms, scatter plots, etc.)

**3.Feature Engineering:**

Depending on your dataset, you may need to create new features or

transform existing ones. This can involve one-hot encoding categorical

variables, handling date/time data, or scaling numerical features.

**Program:**

# Extract date and time components

data['hour'] = data['timestamp'].dt.hour

data['day\_of\_week'] = data['timestamp'].dt.dayofweek

# Lag features

data['pm25\_lag\_1'] = data['pm25'].shift(1)

# Rolling statistics

data['pm25\_rolling\_mean'] = data['pm25'].rolling(window=7).mean()

# Interaction features

data['pm25\_temperature\_interaction'] = data['pm25'] \* data['temperature']

**4.Split the Data:**

Split your dataset into training and testing sets. This helps you evaluate

your model’s performance later.

**Program:**

# Define your features (X) and target (y)

X = data[['Year', 'Month', 'Daily\_PM2.5\_Avg']]

y = data['AirQualityCategory']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**5.Feature Scaling:**

Apply feature scaling to normalize your data, ensuring that all

features have similar scales. Standardization (scaling to mean=0 and

std=1) is a common choice.

**Program:**

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

**6.Data Transformation:**

Scaling, normalization, and categorical variable encoding are common

preprocessing techniques. These transformations ensure that data is in

a suitable format for machine learning algorithms, guaranteeing

consistent scales across features.

**Program:**

#Transform the test data using the same scaler

X\_test\_scaled = scaler.transform(X\_test)

**LOADING THE DATASET**

* Loading the dataset using machine learning is the process of bringing the data into the machine learning environment so that it can be used to train and evaluate a model.
* The specific steps involved in loading the dataset will vary depending on the machine learning library or framework that is being used. However, there are some general steps that are common to most machine learning frameworks:

**a.Identify the dataset:**

The first step is to identify the dataset that you want to load. This dataset may be stored in a local file, in a database, or in a cloud storage Service.

**b.Load the dataset:**

Once you have identified the dataset, you need to load it into the machine learning environment. This may involve using a built-in function in the machine learning library, or it may involve writing your own code.

**c.Preprocess the dataset:**

Once the dataset is loaded into the machine learning environment, you may need to preprocess it before you can start training and evaluating your model. This may involve cleaning the data, transforming the data into a suitable format , and splitting the data into training and test sets.

**PROGRAM:**

**Essential Library Files for Analysis the Water Quality Index**

import pandas as pd

import numpy as np

import scipy

import matplotlib.pyplot as plt

# or

import seaborn as sns

import sklearn

import openpyxl

# or

import xlrd

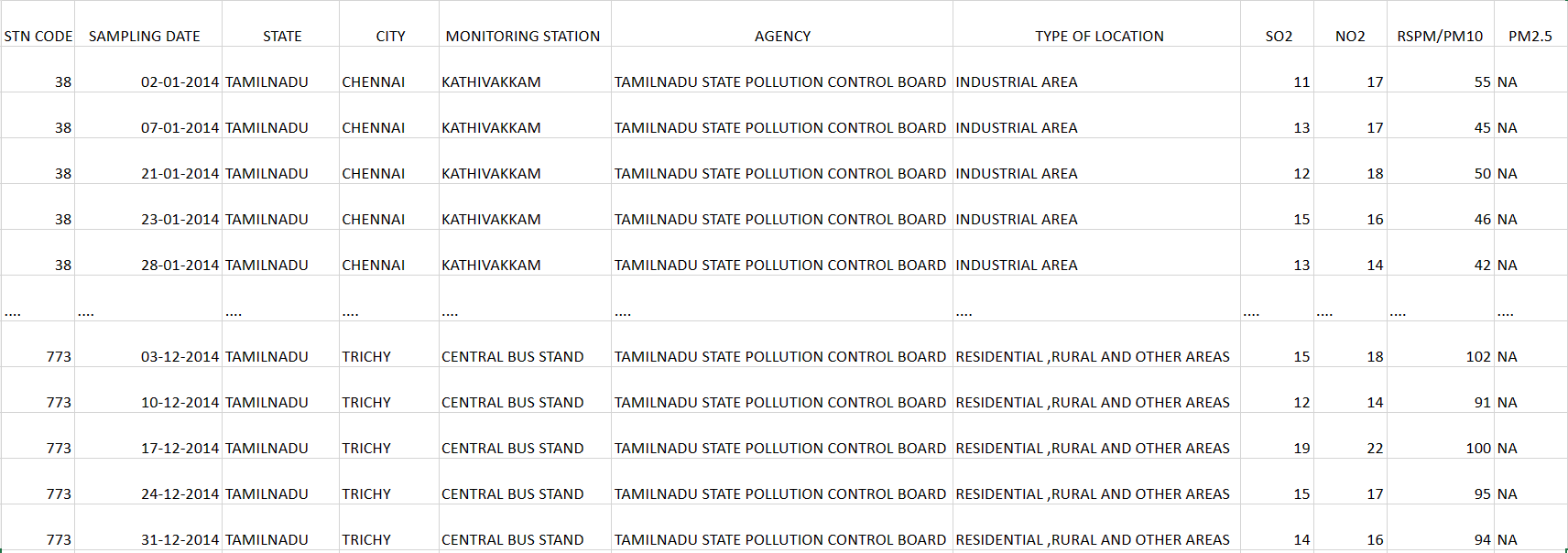
import waterquality

import statsmodels.api as sm

**Loading dataset:**

dataset = pd.read\_csv('water\_potability.csv')

**OUTPUT:**



**PREPROCESSING THE DATASET**

* Data preprocessing is the process of cleaning, transforming, and integrating data in order to make it ready for analysis.
* Preprocessing often involves feature selection, creation, and transformation. This crucial step allows us to identify the most relevant variables, reduce dimensionality, and adapt the data to better suit our modeling goals.

**Program 1:**

import pandas as pd

import numpy as np

# Sample data for water quality parameters

data = {

'DO': [8.2, 7.9, 7.6, 7.0],

'BOD': [2.1, 2.5, 2.0, 3.2],

'pH': [7.2, 7.0, 7.4, 6.8],

'TSS': [15, 18, 20, 22]

}

# Create a DataFrame

df = pd.DataFrame(data)

# Define weights for each parameter (adjust as needed)

weights = {'DO': 0.2, 'BOD': 0.2, 'pH': 0.3, 'TSS': 0.3}

# Define standard values for each parameter (for comparison)

standards = {'DO': 9, 'BOD': 3, 'pH': 7, 'TSS': 10}

# Calculate subindex for each parameter

df['DO\_subindex'] = 100 \* (1 - (abs(df['DO'] - standards['DO']) / standards['DO']))

df['BOD\_subindex'] = 100 \* (1 - (df['BOD'] / standards['BOD']))

df['pH\_subindex'] = 100 \* (1 - (abs(df['pH'] - standards['pH']) / standards['pH']))

df['TSS\_subindex'] = 100 \* (1 - (df['TSS'] / standards['TSS']))

# Calculate WQI for each row

df['WQI'] = (

(df['DO\_subindex'] \* weights['DO']) +

(df['BOD\_subindex'] \* weights['BOD']) +

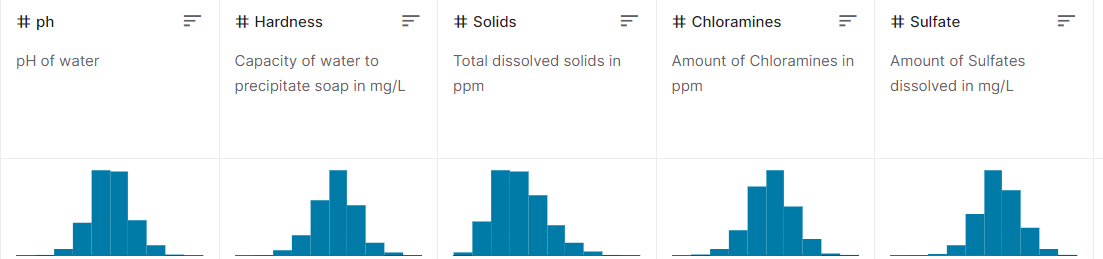
(df['pH\_subindex'] \* weights['pH']) +

(df['TSS\_subindex'] \* weights['TSS'])

) / sum(weights.values())

print(df)

**Output:**



**Program 2:**

## **Visualize Decision Tree For WQI**

INSTALL THE MANDATORY LIB FILES :

pip install scikit-learn graphviz

from sklearn import tree

from sklearn.datasets import load\_iris # Replace with your water quality data

# Sample data (replace with your water quality data)

X, y = load\_iris(return\_X\_y=True)

# Create a decision tree classifier

clf = tree.DecisionTreeClassifier()

clf = clf.fit(X, y)

import graphviz

# Visualize the decision tree

dot\_data = tree.export\_graphviz(clf, out\_file=None,

feature\_names=["Feature 1", "Feature 2", "Feature 3"], # Replace with your feature names

class\_names=["Class 0", "Class 1", "Class 2"], # Replace with your class names

filled=True, rounded=True,

special\_characters=True)

graph = graphviz.Source(dot\_data)

graph.render("wqi\_decision\_tree") # Output filename

graph.view("wqi\_decision\_tree")

**Program 3:**

*#for aqi index*  
pip install matplotlib

import matplotlib.pyplot as plt

# Sample data for water quality analysis

locations = ["Location A", "Location B", "Location C", "Location D"]

wqi\_values = [85, 72, 90, 78]

# Create a bar graph

plt.figure(figsize=(10, 6))

plt.bar(locations, wqi\_values, color='skyblue')

plt.title("Water Quality Index Analysis")

plt.xlabel("Locations")

plt.ylabel("WQI Value")

# Add data labels above the bars

for i, value in enumerate(wqi\_values):

plt.text(i, value + 2, str(value), ha='center', va='bottom')

# Display the graph

plt.tight\_layout()

plt.show()

**CONCLUSION**

* Loading and preprocessing the dataset are pivotal phases in any data analysis project, and they hold equal importance in the context of an air quality analysis project. These initial steps are instrumental in shaping the trajectory of the project and the quality of the insights that can be extracted.
* By loading the dataset, we establish the foundation for data analysis and machine learning. It allows us to confirm that the data is accessible and formatted correctly, laying the groundwork for reliable analysis. Preprocessing encompasses data cleaning, where missing values, duplicate records, and outliers are addressed. Clean data is the cornerstone of accurate analysis and modeling.
* In conclusion, dataset loading and preprocessing are not merely preliminary chores but the cornerstones of successful air quality analysis. They set the stage for rigorous, reliable, and informative analysis, offering the potential to yield actionable insights that can contribute to the betterment of Water quality, public health, and environmental well-being. The care and precision applied to these phases directly influence the strength and significance of the project's outcomes.