SKILL UP

IBM

AIR QUALITY ANALYSIS IN TAMIL NADU

DAC_Phase2

Team Members:

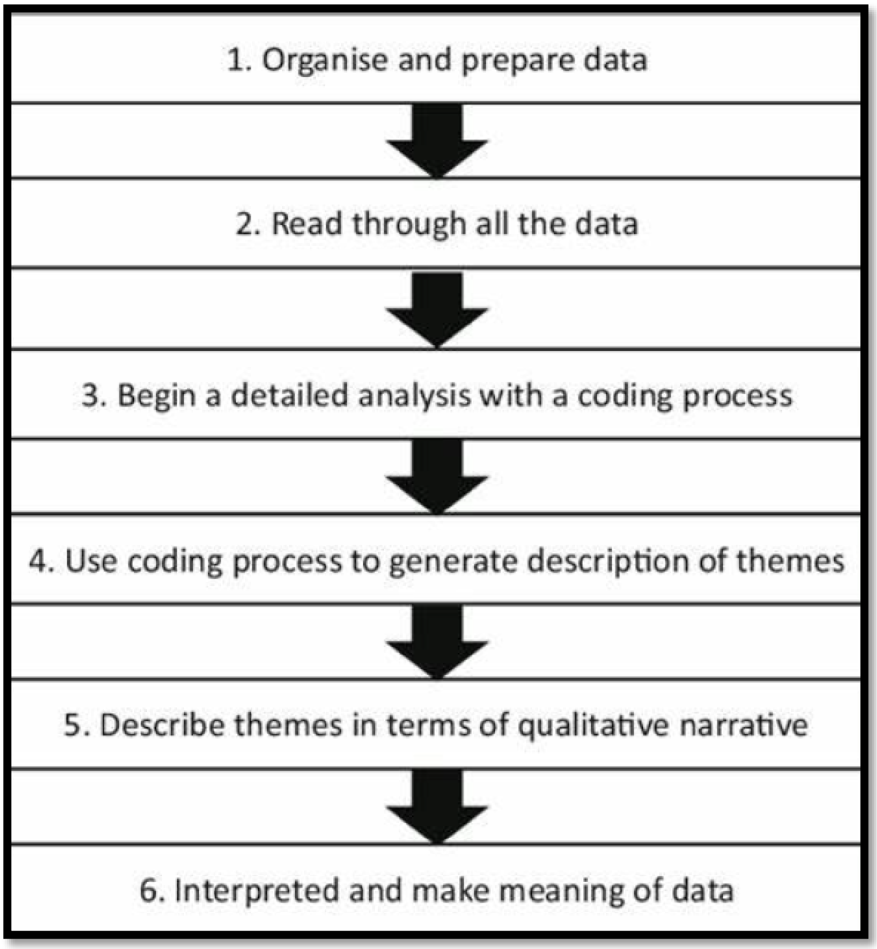
1. Nithyasri V G
2. Sakshi Rajesh Bhavsar
3. Mathew P
4. Sarfraj Ansari

AIR QUALITY ANALYSIS IN TAMIL NADU

Project Definition:

The primary objective of this project is to conduct a comprehensive assessment and visualization of air quality data collected from monitoring stations situated across Tamil Nadu. Through this project, we aim to extract valuable insights into the prevailing patterns of air pollution, identify regions exhibiting elevated levels of pollution, and establish a predictive model capable of estimating RSPM/PM10 levels. This analysis will be based on the concentrations of SO2 and NO2.

Case process diagram:


1. Organise and prepare data
2. Read through all the data
3. Begin a detailed analysis with a coding process
4. Use coding process to generate description of themes
5. Describe themes in terms of qualitative narrative
6. Interpreted and make meaning of data

Algorithm: Air Quality Data:

Import Relevant Libraries:

- import pandas as pd
- import numpy as np
- import matplotlib.pyplot as plt
- import seaborn as sns
- import scipy.stats as stats
- import geopandas as gpd
- from sklearn.model_selection import air_test_split
- from sklearn.linear_model import LinearRegression

Load Data:

Step 1: Import the pandas library

- import pandas as pd

Step 2: Read the CSV file

- df = pd.read_csv("air_quality_data.csv")

Step 3: Print the DataFrame

- print(df)

Data Preprocessing:

Handle missing values and outliers:

- df.dropna() # Drop rows with missing values
- new_df = df.drop(['State','Agency'],axis =1)
- new_df = df.dropna(subset=['SO2','NO2'])
- new_df = df.fillna(0)

Exploratory Data Analysis (EDA):

Input:- DataFrame (df) containing the dataset

Output:- Summary statistics, visualizations, and insights about the dataset

Descriptive Statistics:

Step1: Calculate basic statistics for each numerical column.

- df.describe()

Step2: Data Types and Missing Values- Identify data types and check for missing values:

- df.info()

Step3: Data Distribution Visualization

- import matplotlib.pyplot as plt
- import seaborn as sns
- # Create data visualizations
- sns.histplot(df['City/Town/Village/Area'])
- plt.show()

Data Transformation and Feature Engineering:

Create new features or transform existing ones to make the data more suitable for analysis. This might involve scaling, encoding categorical variables, or creating time-based features.

Step 1: This means that the categorical values in 'RSPM/PM10' are converted into binary columns.

- df_encoded = pd.get_dummies(df, columns=['RSPM/PM10'])
- print(df_encoded)

Step2: reads a CSV file, performs one-hot encoding on the specified column, and prints the resulting DataFrame with the encoded values.

Data Distribution Visualization

Step 1:

- import pandas as pd
- import matplotlib.pyplot as plt
- import seaborn as sns
- import numpy as np

Step 2:

Box Plots:

- Identify outliers and distribution of numerical variables:

  - sns.boxplot(x='City/Town/Village/Area', y='RSPM/PM10', data=df)

Scatter Plots:

- Visualize relationships between two numerical variables:

  - #Scatter Plots for Correlation Analysis:
  - plt.scatter(df['SO2'], df['NO2'])

Machine Learning:

If your goal is predictive modeling or classification, you can use machine learning libraries such as Scikit-Learn to train and evaluate models.

Step 1: Import Libraries

- import pandas as pd: Import the pandas library for data manipulation
- from sklearn.linear_model import LinearRegression
- Import the LinearRegression model from scikit-learn

Step 2: Load and Prepare Data

- new_df = pd.read_csv("air_quality_data.csv"):
- Read the data from the CSV file "air_quality_data.csv" and store it in the DataFrame new_df.
- new_df = df.dropna(subset=['SO2','NO2','RSPM/PM10']):
- Remove rows with missing values in the columns 'SO2', 'NO2', and 'RSPM/PM10'.

Step 3: Define Independent and Dependent Variables

- X = new_df[['SO2']]: Define the independent variable X as the 'SO2' column.
- y = new_df['RSPM/PM10']: Define the dependent variable y as the 'RSPM/PM10' column.

Step 4: Create and Train the Linear Regression Model

- model = LinearRegression(): Create an instance of the LinearRegression model.
- model.fit(X, y): Train the model using the independent variable X and dependent variable y.

Step 5: Print Coefficients and Make Predictions

- print("Intercept (b0):", model.intercept_): Print the intercept (b0) of the regression line.
- print("Slope (b1):", model.coef_[0]): Print the slope (b1) of the regression line.
- X_new = [[5]]: Define a new value of 'SO2' (5 in this case) for prediction.
- y_pred = model.predict(X_new):
- Use the model to predict the corresponding 'RSPM/PM10' value.
- print("Predicted y for X =", X_new[0][0], ":", y_pred[0]): Print the predicted value.

Visualizations and Reports:

- Creating estimating RSPM/PM10 levels with analysis will based on the concentrations of SO2 and NO2.

End.