

Identification of Sister Cities using Pollution data in India

B.Tech Project Report

July 2023 – May 2024

Submitted by

Balaji N

CH20B022

B.Tech, Chemical Engineering

Under the guidance of

Prof. Ragunathan Rengasamy

Department of Chemical Engineering

Table of Contents

1. Creating Synthetic Data	3
2. Evaluating Interpolation and Completeness Score.....	3
3. Feedback Loop to Evaluate Completeness Score and Clustering.....	4
4. Anomaly Detection Using Moving Window Median Absolute Deviation	6
5. Anomaly Detection Using Autoencoders	7
6. Interpolation Mechanism	7
7. Scoring CPCB stations	8
8. Case Study of Delhi.....	9
9. Clustering CPCB Stations in India	12
10. Conclusion.....	13
11. Appendix	14

Identification of Sister Cities using Pollution data in India

1. Creating Synthetic Data

📄 **File Name: create_data.ipynb**

📄 Synthetic time series data was generated to test our custom completeness score (Feed Back Loop)

📄 Time series generated using 4 different trends. Added seasonality and noise to generate new time series (Refer function generate_data)

📄 Inputs

- Number of Series to be generated
- Timesteps (set to 365×5 i.e. to simulate hourly data for 5 years)
- Noise

📄 Output (generated 40 series)

- Shape: 40 x 1826
- Labels: 1-4 (Suggest the parent curve it belongs)

2. Evaluating Interpolation and Completeness Score

File Name: Imputation_Scoring.ipynb

We used a feedback loop to evaluate our custom completeness scoring mechanism and interpolation mechanism.

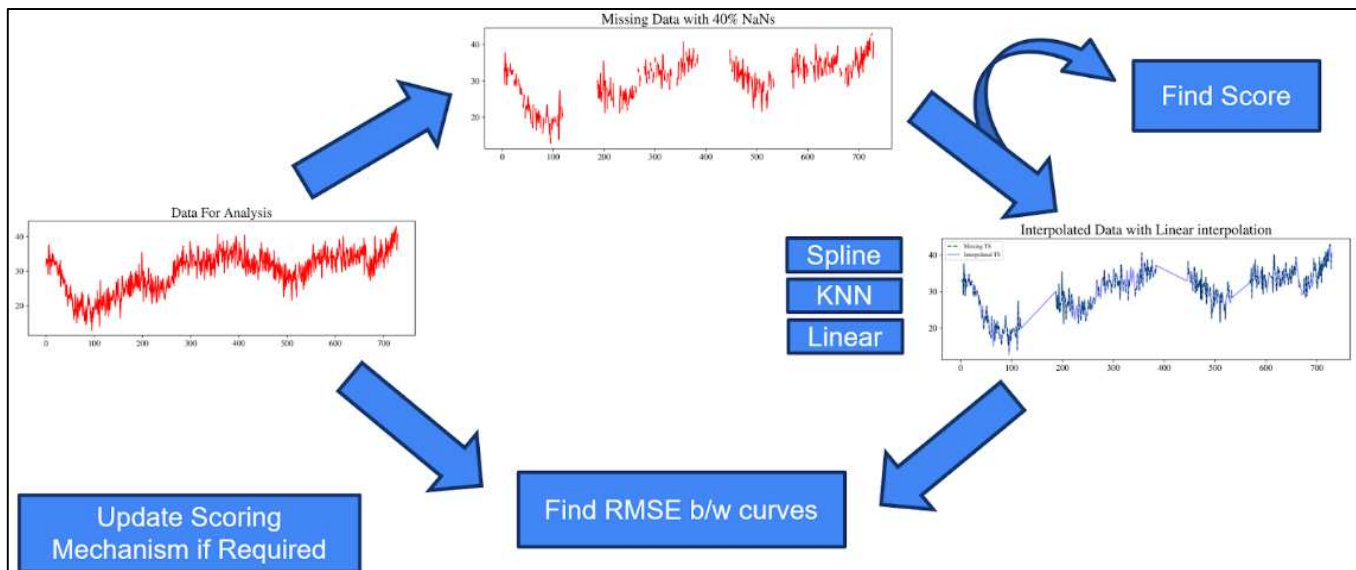
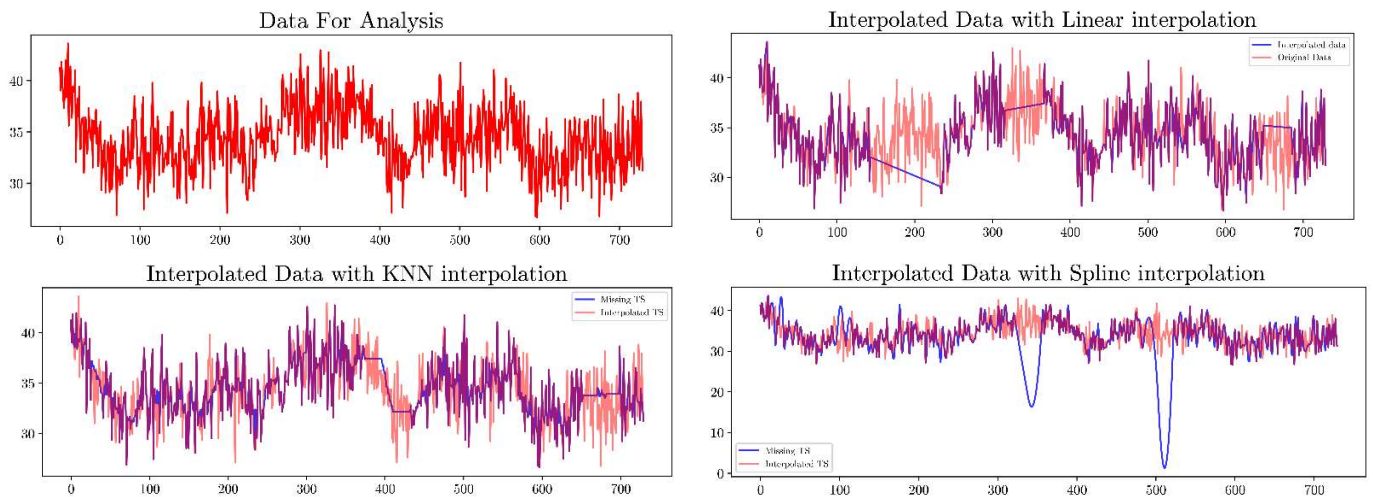


Figure 1 : Feedback loop to evaluate scoring mechanism

Outputs for some interpolation mechanisms have been shown above. Linear interpolation and KNN interpolation performed much better than spline interpolation mechanism.



We defined our own custom completeness Score. Usually, completeness is just measured as a percentage for incomplete records. But often we have to interpolate these missing values before using the data. It is easier to interpolate when the continuous missing block length is smaller. Thus, we have our own custom completeness metric function that takes continuous missing block length also into account:


$$Completeness = (1 - \% NaN's) * (1 - \frac{Mean(Missing\ block\ Lengths)}{len(data)})$$

% NaN: Denotes total number of incomplete records to total number of records in the dataset.

Mean Significant missing block length (MSMBL): Denotes mean length of continuous missing blocks in a dataset. This parameter captures how good/bad data quality will be after we perform interpolation.

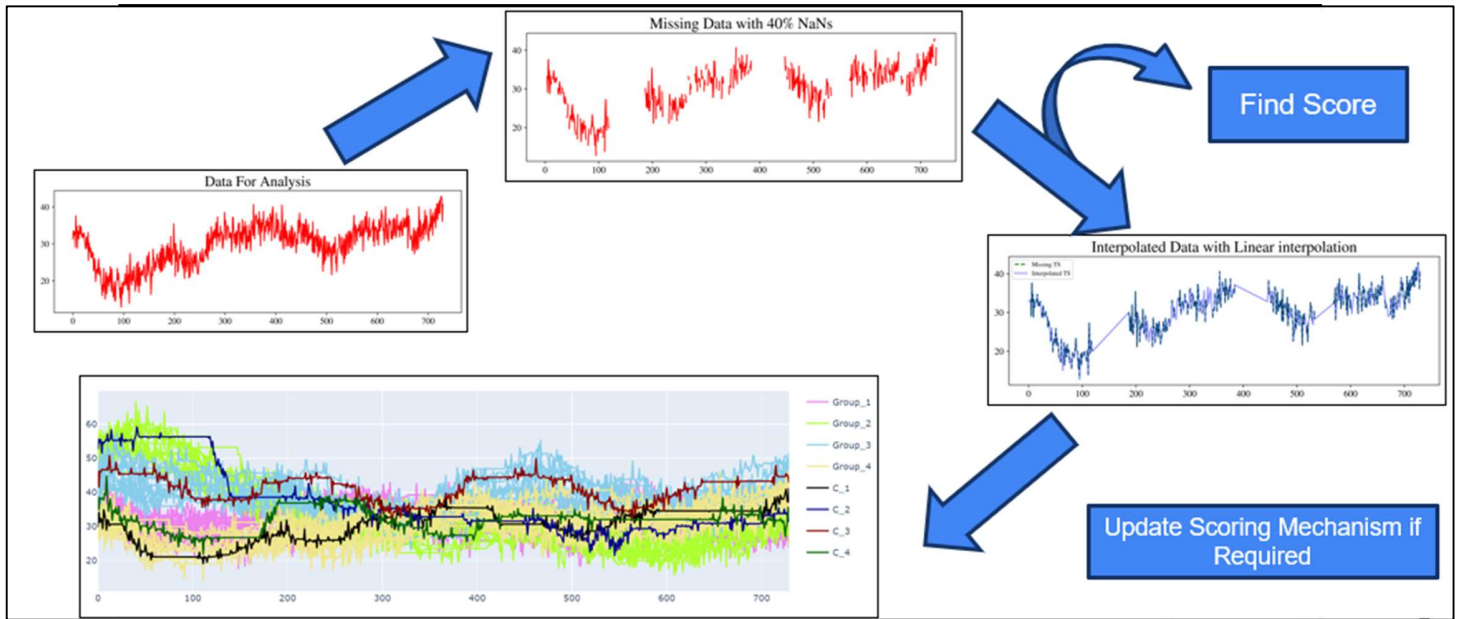
NOTE: We used tuning parameter k in our completeness score to define mean significant missing block length. For more details, please refer Appendix section 1.

3. Feedback Loop to Evaluate Completeness Score and Clustering

 **File Name:** Clustering_Feedback_Scoring.ipynb

Aim: If completeness Score is low then clustering should be poor. Try various qualities of data and different formulations of completeness scores (as discussed in the meeting)

Relationship b/w completeness and RMSE of interpolated data and original data was shown in section 2. So, our completeness score is working. We need to see if we can establish relationship b/w completeness and clustering metrics(rand-index).



Synthetic data generated in section 1 was used to evaluate our clustering mechanism. Datasets of various qualities were generated. Linear Interpolation and KNN were used to interpolate missing values. Missing blocks on synthetic data were generated using 2 methods:

- Continuous: A entire block of data was missing in the dataset.
- Selective: Data were missing in small blocks. (mimics real world case)

	NaNs	Score_20	Score_80	F1_Score	NMI	ARI	Interpolation	NaN_Method
0	0.2	0.797664	0.797664	0.980031	0.948440	0.944123	Linear	Selective
1	0.3	0.697916	0.697916	1.000000	1.000000	1.000000	Linear	Selective
2	0.5	0.498048	0.498048	0.924603	0.841108	0.789162	Linear	Selective
3	0.6	0.398058	0.397420	1.000000	1.000000	1.000000	Linear	Selective
4	0.9	0.098530	0.098298	0.808779	0.714202	0.610997	Linear	Selective

Figure 3: Clustering results on synthetic Data

No clear trend was visible b/w clustering metrics and completeness scores. Possible reason can be: Misclassification in clustering will happen only when significant part of the trend curve is interpolated in such a manner that it mismatches with the original trend curve. Also our clustering mechanism might be just too good and is able to identify current labels irrespective of large missing blocks. Thus, we weren't able to establish the required relationship on synthetic data. But our completeness score is working and same was proved in section 2.

4. Anomaly Detection Using Moving Window Median Absolute Deviation

📄 File Name: **Anomaly_Detection_MAD.ipynb**

📄 Reference: [Anomaly Detection on Hydrological Time Series Data](#)

Outliers: Defining outliers was tricky as our process is bound to vary rapidly and sudden spikes may be due to various reasons like bursting crackers, fire close to cpcb station etc.. Thus, we define outliers to be points that have a sudden spike and the spike dies down quickly. Any spike that dies within 4 hours is anomaly (4hrs is a tuneable parameter as per requirement).

Inputs:

- 📄 Window Size (=9, i.e. +/- 4 hrs)
- 📄 Threshold (=3, tuneable)
- 📄 Min_Mad = $0.1 \cdot \text{std_dev}(\text{Pollutant})$

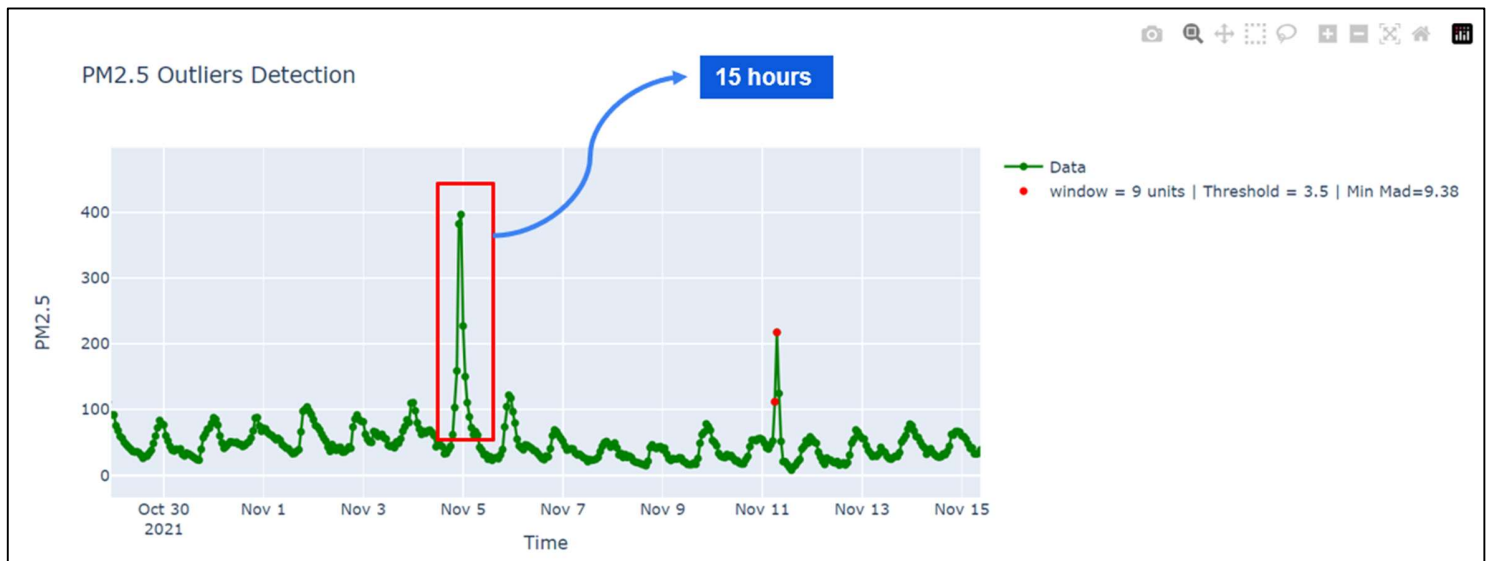
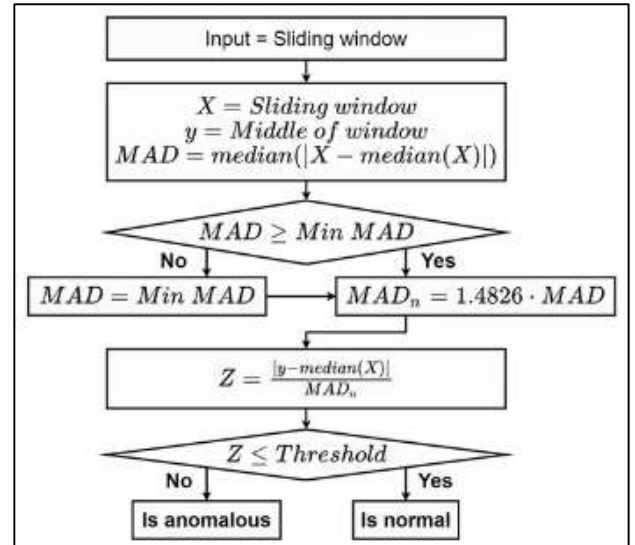


Figure 4 : Outlier Detection using MAD moving window approach

We can see that spike around November 11 has been captured by our model as expected. The spike in PM2.5 value around 5th November takes around 15 hours to die, thus the points in that spike hasn't been classified as anomaly.

5. Anomaly Detection Using Autoencoders

📄 **File Name:** LSTM_autoencoders.ipynb

📄 **Reference:** [Keras Documentation](#)

We have shown results for PM_{2.5} and this approach can be extended to other parameters too. We are using data at hourly resolution and our autoencoder takes a window of size 24 (i.e. one day). Our model contains LSTM layers which are used to encode data and again it is decoded back to a vector of 24 timesteps. If the reconstruction loss is greater than threshold then it is considered to be an anomaly.

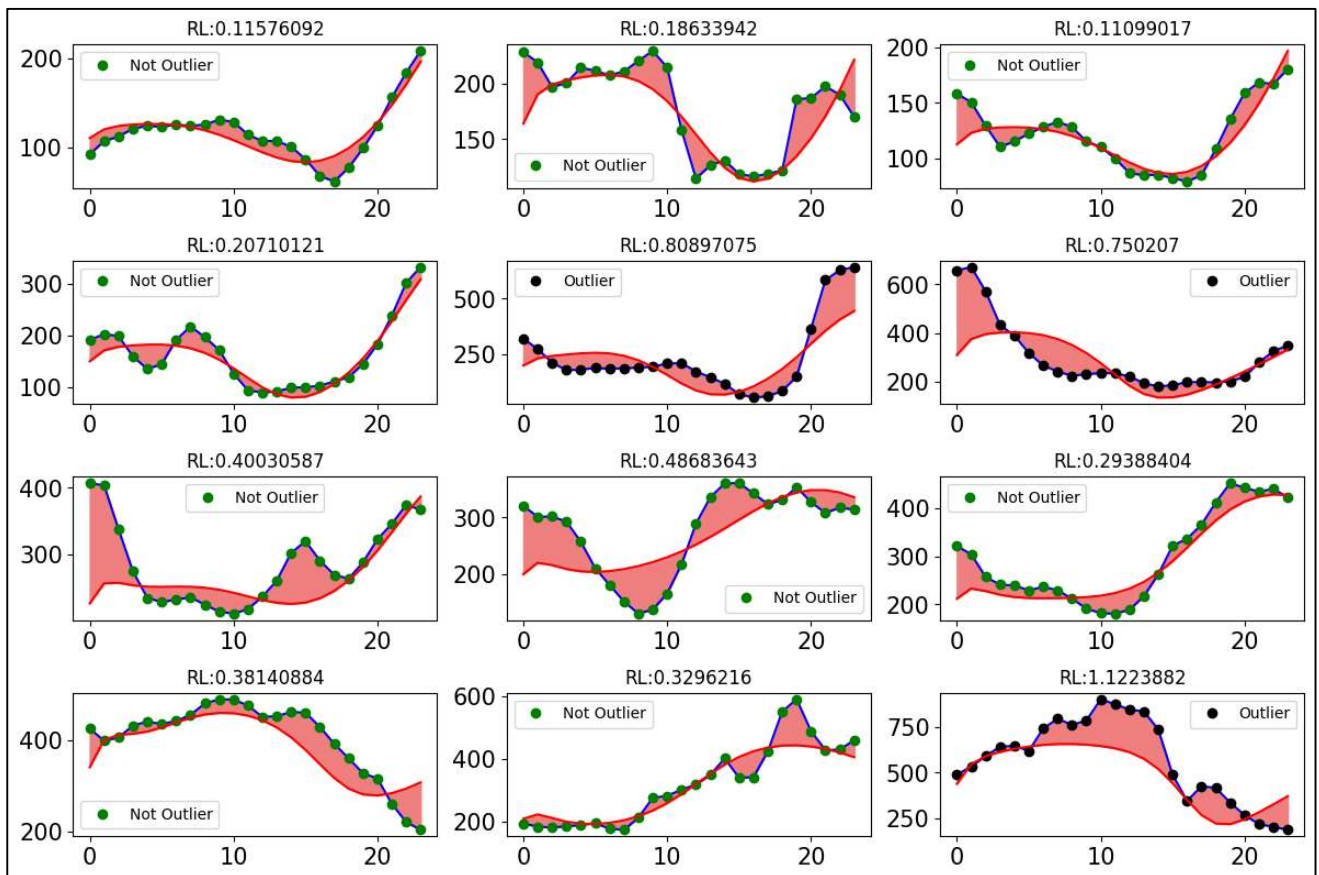


Figure 5 : Outlier Detection using Autoencoders

6. Interpolation Mechanism

📄 **File Name:** Imputation_mech.ipynb

There is no best interpolation mechanism and it depends on data property. Pollution dataset is bound to change rapidly because of the process itself. Thus, identifying best interpolation mechanism can be tricky. Linear Interpolation, KNN and MICE interpolators were tested on

cpcb station dataset by generating synthetic NaN's and the mechanism that gave least RMSE between original data and imputed data is the best mechanism.

In some stations one of the values of $PM_{2.5}$ or PM_{10} were available. $PM_{2.5}$ and PM_{10} are very well correlated. If one of the values is available then it is easier to estimate the value of other variable thus MICE did a good job in interpolating $PM_{2.5}$ and PM_{10} . In all other cases linear interpolation performed better than KNN and MICE interpolator. Thus best interpolation mechanism is:

- 📊 MICE Interpolation: $PM_{2.5}$, PM_{10}
- 📊 Linear Interpolation: NO_2 , SO_2 , O_3 , NH_3

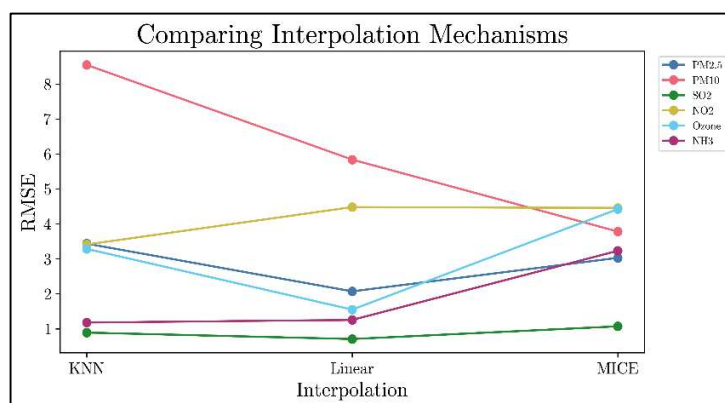


Figure 6: Identification of best Interpolation mechanism

7. Scoring CPCB stations

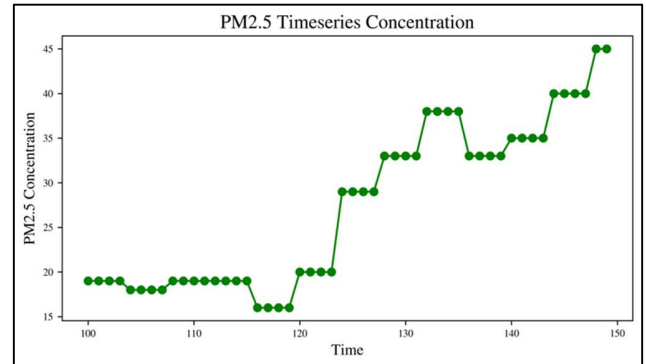
- 📊 **File Name:** Clustering in Delhi.ipynb
- 📊 **Output File containing Scores:** Stations_Score.csv
- 📊 **Reference:** <https://firsteigen.com/blog/6-key-data-quality-metrics-you-should-be-tracking/>

We used 5 attributes to score each CPCB station

1. **Accuracy:** This attribute helps us to understand if behaviour of data is within expected bounds. Anomalies may appear in the system due to several factors, such as malicious activities, hardware failures, inaccuracies in data collection, or adversarial attacks. Anomalous data records can be found using anomaly detection techniques like autoencoders, Moving window MAD approach etc...
2. **Completeness:** It measures the number of incomplete records and it's usually expressed in percentage. We have our own custom completeness function which was shown above
3. **Consistency:** Consistency checks are done using integrity constraints. The integrity constraints vary depending on the domain and the application conditions. . For example, in some domains, data observations cannot be negative or their values should fall within a

particular range. After defining the integrity rules for the data values, the consistency score can be calculated as the fraction of data values that do not meet the integrity constraints.

4. **Uniqueness:** This metric is used to track duplicate data. Counting any data twice will unduly weigh the results. It is important to either merge duplicate records or delete them. Some CPCB stations don't sample PM2.5 and PM10 at 15 min resolution and it is sampled at hourly resolution. Thus, they duplicate the values in order to fill data at 15 min resolution. Figure on right shows one such CPCB station where PM2.5 data has been duplicated. Thus, it is advisable to use data at hourly resolution.



5. **Validity:** It measures how well data conforms to the standards, i.e. if wrong data format is entered in a field, then it becomes unusable for analysis. It is important to check if a given field has correct data types. e.g. Pin code or phone number fields cannot have letters. Invalid data should be identified and corrected before using them. All fields in CPCB dataset are valid with respect to datatypes.

8. Case Study of Delhi

📄 **File Name:** clustering_India.ipynb

There are 40 CPCB stations in India. Delhi encompasses diverse regions including industrial, commercial, residential etc... Our aim is to see if our clustering algorithm is able to capture this relationship between stations. There are some stations in Delhi that don't measure at least one of

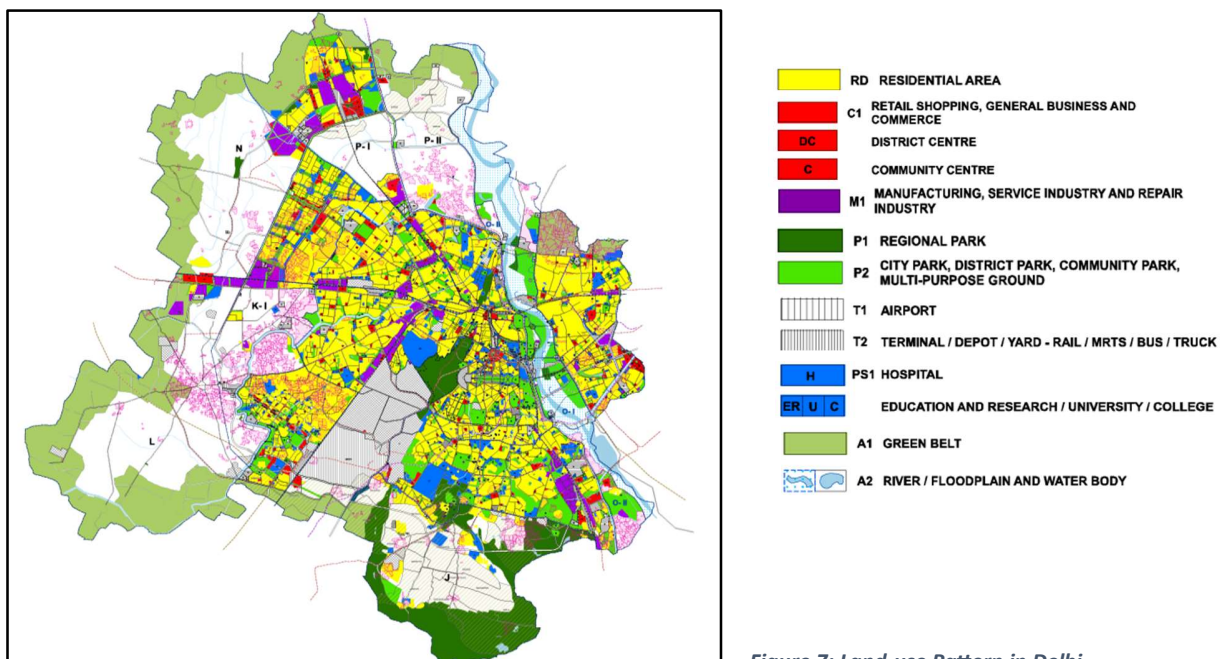


Figure 7: Land-use Pattern in Delhi

the six pollutants required for our analysis. Thus, we have dropped those stations and we have 30 stations in Delhi.

Until now, we haven't filtered out stations in India using our scoring mechanism. Our aim is to investigate whether stations with low overall scores have been erroneously classified into incorrect clusters. Various factors could contribute to this occurrence. For instance, a high volume of missing values in the dataset or numerous anomalies might distort the representation of air quality in the region, potentially leading to misclassification of the station. Both K-Means and Hierarchical Clustering were implemented. Hierarchical clustering gave us better sister cities using land use pattern as ground truth

Clustering Mechanism	Rand-Index
Hierarchical	0.63
K-Means	0.56

Table 1 : Clustering Results in Delhi

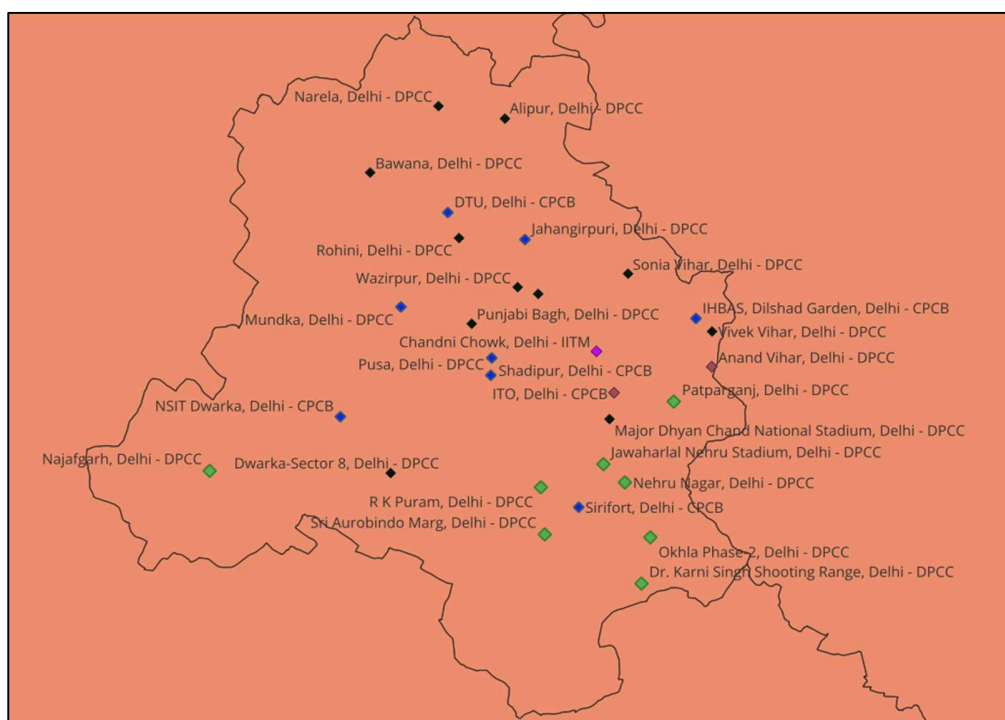


Figure 8: Sister cities in Delhi (Hierarchical Clustering)

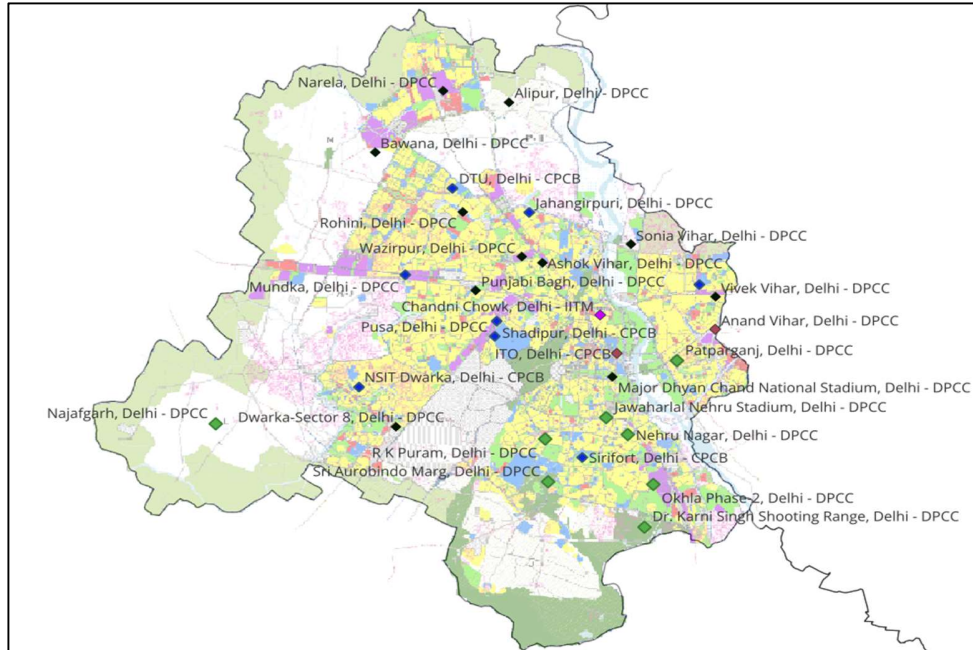


Figure 9: Sister Cities with Superimposed Land-use Pattern

Data Quality scores of all 30 stations were calculated. We will analyse if there is any relationship between our clustering results and data quality overall score.

Chandini Chowk (site_5393) is clustered into a separate cluster. Overall Score for this station is 0.000048 which is very low as overall score ranges between 0 to 1. The reason for low overall score is lot of missing values in the dataset (i.e. low completeness score). By land use pattern Chandini Chowk is an industrial area and should have got clustered under label 4.

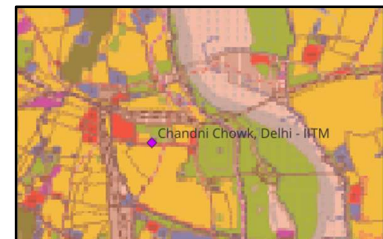


Figure 10: Land Use of Chandini Chowk

NSIT Dwaraka station also has a low overall data quality score (~0.1). From land use pattern, NSIT Dwaraka station should be clustered in residential area but it got clustered into a commercial area.

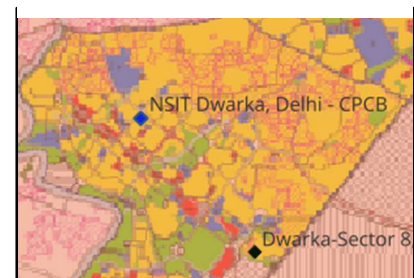



Figure 11: NSIT Land use pattern map

In the above discussed cases like Chandini Chowk and NSIT-Dwaraka misclassification is because the overall score is less and the interpolated dataset is not able to reflect the air quality in that region. This is one of the major reasons for possible mis-classification.

Typically, it is crucial to apply a specific threshold on the overall quality score to filter out stations. This ensures that only stations with reliable data are considered, providing an accurate reflection of air quality within a given region.

9. Clustering CPCB Stations in India

 **File Name:** clustering_India.ipynb

There are 513 stations in India and only 358 measure all the parameters that we need for analysis. From previous section threshold of 0.1 is used to filter out air quality monitoring stations in India (for reliable data quality). After applying the above filter, the number of stations reduced to 102. These 102 stations were grouped into 3 clusters as low, medium or highly polluted regions using hierarchical clustering. All these 3 clusters are well separated which is shown in figures 25, 26, 27. Ground truth was not available for stations across India thus we haven't evaluated our clustering mechanism for India.

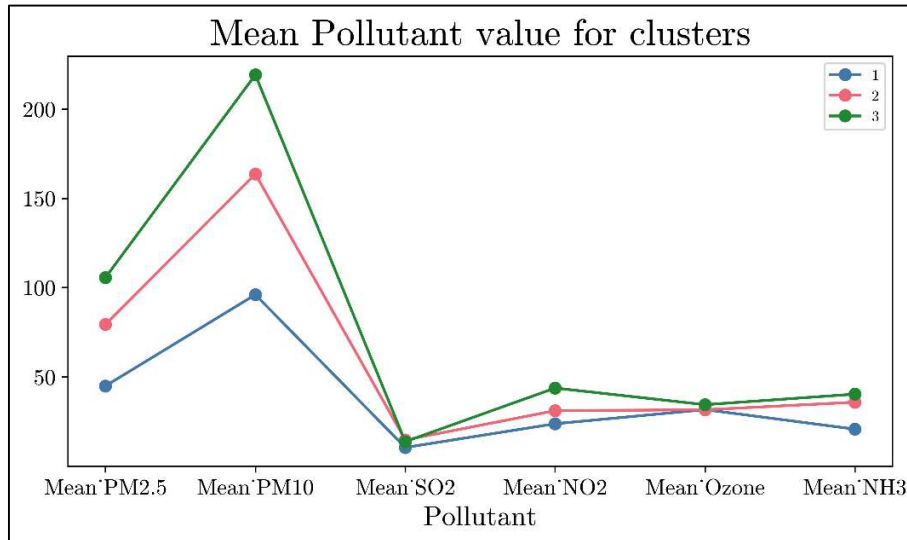


Figure 13: Cluster wise mean pollutant values of CPCB stations in India

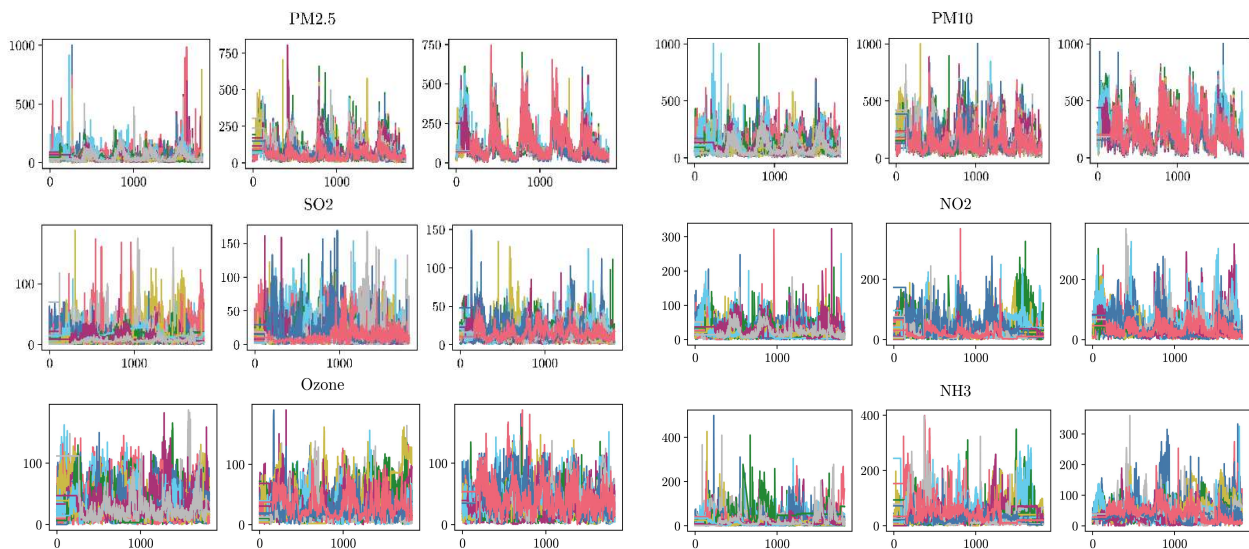


Figure 12: Cluster wise pollution pattern for various parameters

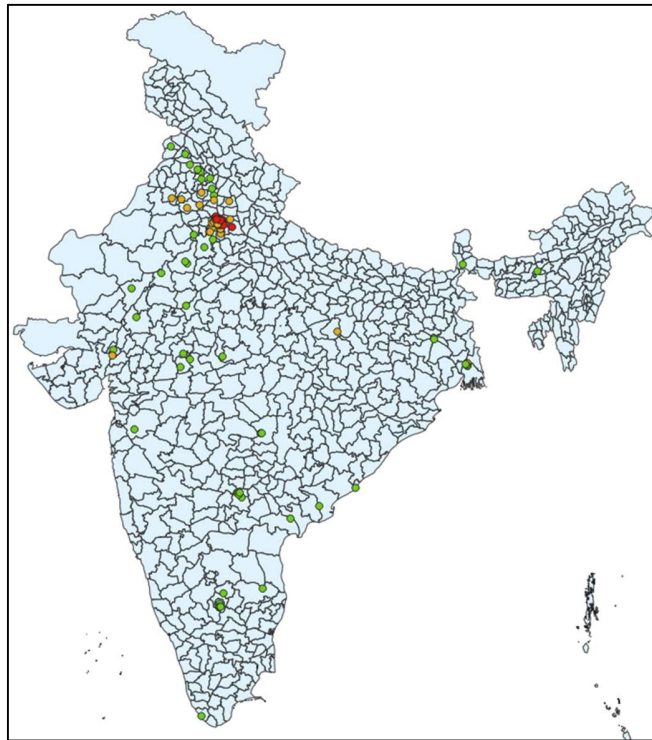


Figure 14 : Hierarchical clustering of air quality stations in India

10. Conclusion

In brief, this study introduced a novel scoring mechanism tailored for assessing the reliability of continuous datasets. By assessing metrics such as accuracy, validity, uniqueness, completeness, and consistency, researchers gain insights into data quality. This scoring mechanism was used to filter out stations before clustering to avoid any possible misclassifications.

In this paper, it has been systematically illustrated that stations with low scores have been incorrectly assigned to clusters. The underlying reason is attributed to failure of data quality to accurately reflect the air quality within the respective region. This could be attributed to various factors such as a high frequency of missing values, anomalies etc... Stations above a certain threshold on overall data quality score were filtered out and clustered using hierarchical clustering.

11. Appendix

Let continuous missing array length be = [100, 90, 88, 84, 65, 40, 3, 1, 1, 1].

The proportion of missing values is given by:

$$\text{Missing Values Proportion} \propto \left(1 - \frac{(\text{No of Null Values})}{\text{len}(\text{data})}\right)$$

Here, len(data) represents the length of your dataset.

K is a tuning parameter for Contiguous Missing Value Blocks. Let $K = 0.8$. Any continuous missing block length more than $k \times$ largest missing continuous block length would be considered significant. K can be tuned as per use case. In our case any value more than $100 \times 0.8 = 80$ would be considered significant missing block length

The proportion of Contiguous Missing Value Blocks is expressed as:

$$\text{Contiguous Missing Value Block Proportion} \propto \left(1 - \frac{\text{mean}(\text{significant_missing_blocks})}{\text{len}(\text{data})}\right)$$

Now, let's calculate the score:

$$\text{Score} = \left(1 - \frac{473}{1000}\right) \left(1 - \frac{90.5}{1000}\right) = 0.479$$

Key points about the Score:

- Ranges between 0 and 1.
- A higher score implies better data quality.
- Considers both missing values and the length of contiguous missing blocks.