PYTHON PROJECT REPORT

(Project Semester: January-April 2025)

Title of the Project: Air Quality Analytics Dashboard

Submitted by: Aarupadaiyar KJ

Registration No.: 12307770

Programme and Section**: B.Tech CSE** (K23FD)Course Code: INT375

Under the Guidance of:Baljindar Kaur

Discipline of CSE/IT Lovely School of Computer Science & Engineering Lovely Professional University, Phagwara

DECLARATION

I, Aarupadaiyar KJ, student of **Bachelors of Technology (B.Tech)** under CSE/IT Discipline at Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own intensive work and is genuine.

Date: 03-April-2025

Registration No.: 12307770

Name of the Student: Aarupadaiyar KJ

CERTIFICATE

This is to certify that Shivam Kumar bearing Registration No. **12307770** has completed **INT375** project titled **“Air Quality Analytics Dashboard”** under my guidance and supervision. To the best of my knowledge, the present work is the result of her original development, effort, and study.

Mrs.Baljindar Kaur, Assistant Professor, School of Computer Science & Engineering

Lovely Professional UniversityPhagwara, Punjab

Date: **04-April-2025**

ACKNOWLEDGMENT

I would like to express my sincere gratitude to **Mrs. Baljindar Kaur**, my project guide, for their invaluable support, guidance, and encouragement throughout the development of this project. Their expert insights and constructive feedback have been instrumental in shaping the project's outcome.

I am also thankful to **Lovely Professional University** for providing a conducive learning environment and access to resources that made this project possible. Additionally, I extend my appreciation to my professors and peers for their continuous motivation and insightful discussions, which greatly enhanced my understanding of the subject.

Lastly, I would like to acknowledge the unwavering support of my family and friends, whose encouragement has been a source of inspiration throughout this journey.

TABLE OF CONTENTS

1. INTRODUCTION

Air pollution is one of the leading environmental health risks across the globe, especially in urban areas where industrial activity, transportation, and dense population significantly contribute to poor air quality. Monitoring and analyzing air pollutant levels is essential for identifying high-risk zones, understanding temporal patterns, and planning effective interventions.

This report presents a **comprehensive analysis of air quality data** using **Python libraries** such as Pandas, Matplotlib, and Seaborn. The dataset includes over **18,000 records** of air quality measurements collected across different community districts, time periods, and pollutant types.

The analysis aims to:

- Identify the **most commonly monitored pollutants**
- Highlight the **most and least polluted locations**
- Examine **seasonal and annual trends**
- Derive **insights using visualizations** and statistical summaries

By leveraging Python's powerful data manipulation and visualization capabilities, this report seeks to transform raw air quality data into meaningful insights for informed decision-making and policy formulation.

2. SOURCE OF DATASET

The dataset used in this analysis is sourced from **Open Data NYC**, a platform maintained by the City of New York to promote transparency and accessibility of public information. Specifically, this dataset contains **air quality measurements** collected by the **New York City Community Air Survey (NYCCAS)**.The dataset includes a ributes such as:

Dataset Highlights:

- **Title:** Air Quality (NYC Open Data)
- **Published by:** NYC Department of Health and Mental Hygiene (DOHMH)
- **Coverage:** Multiple community districts (CDs) across New York City
- **Time Span:** Seasonal and annual records over several years (e.g., Winter 2014-15 to Annual Average 2017)
- **Pollutants Measured:** Primarily **Nitrogen Dioxide (NO_2)**, with other pollutants potentially included
- **Units of Measurement:** Most data values are represented in **parts per billion (ppb)** or equivalent units

Link : https://catalog.data.gov/dataset/air-quality

3. DATASET PREPROCESSING

Raw environmental datasets often come with inconsistencies such as missing entries, redundant values, and formatting errors. To prepare the dataset for reliable analysis, the following preprocessing steps were undertaken:

1. **Replacement of Missing Values:**
   - The dataset contained several "na" entries in the numeric columns such as pollutant_min, pollutant_max, and pollutant_avg.
   - These values were replaced using column-wise mean imputation through Python .mean and filtering techniques.
2. **Column Standardization:**
   - Column headers initially appeared incorrect or repeated as "country" due to formatting issues.
   - Appropriate names such as state, city, station, pollutant_id, etc., were assigned.
3. **Data Type Verification:**
   - Numerical columns were reformatted to appropriate data types for accurate calculations.
   - Timestamps were converted into consistent date-time format to allow time-series plotting.
4. **Filtering and Sorting:**
   - The data was filtered by pollutant types and sorted based on time, region, and pollutant levels. Duplicate rows were removed to ensure clean and unique observations.
   -

After preprocessing, the dataset was ready for in-depth analysis using Excel functionalities.

4. ANALYSIS ON DATASET

The dataset utilized in this study is derived from the New York City Open Data portal and was originally curated by the New York City Department of Health and Mental Hygiene (DOHMH) under the New York City Community Air Survey (NYCCAS) initiative. It comprises a comprehensive record of air pollutant measurements collected across various community districts (CDs) of New York City over multiple seasonal and annual intervals.

4.1.1 Attributes and Structure

The dataset contains 18,862 rows and 12 columns (11 post-cleaning), each representing a distinct observation of pollutant concentration recorded at a specific location and time period. The key attributes are as follows:

- Unique ID: A unique identifier for each record.
- Indicator ID: Identifier for the type of pollutant being measured.
- Name: The name of the pollutant (e.g., Nitrogen Dioxide - NO_2).
- Measure: Type of statistical measure (e.g., Mean).
- Measure Info: Additional context for the measurement (e.g., annual average or seasonal mean).
- Geo Type Name: Geographic classification (e.g., Community District).
- Geo Join ID: Spatial join key used for mapping and GIS applications.
- Geo Place Name: Name of the specific location or district where the measurement was taken.
- Time Period: The temporal frame of the measurement (e.g., "Winter 2014-15", "Annual Average 2017").
- Start_Date: The beginning date of the time period (converted to datetime format).
- Data Value: The numeric value of the air quality measurement (typically in parts per billion or equivalent).

4.1.2 Temporal and Spatial Coverage

The dataset spans several years and includes both seasonal and annual average observations. It is spatially distributed across New York City's 59 community districts, thereby offering extensive geographic granularity for localized air quality analysis.

4.1.3 Data Quality and Preprocessing

Upon initial inspection, the dataset demonstrated high data integrity:

- No missing values in critical fields (after dropping the non-informative Message column).
- No duplicate records were present.
- All temporal values were successfully converted for time-series operations.

This preprocessing ensures that the dataset is robust and suitable for advanced statistical and visual analysis, as described in the subsequent sections of this study**.**

Tools and Libraries Used

- **Python 3.10+**
- **Jupyter Notebook**
- **Pandas** – for data manipulation
- **NumPy** – for numerical computations
- **Matplotlib** and **Seaborn** – for static visualizations
- **Plotly** – for interactive graphs
- **Datetime** – for timestamp formatting
- **Missingno** – for visualizing missing data

4.2 Specific Requirements and Objectives

The primary goal of this project was to analyze and visualize air quality data using **Python** and its data science libraries. This project leverages tools like **Pandas, Matplotlib, Seaborn, and Plotly** to process pollution data, identify high-risk regions, study pollutant trends over time, and build interactive charts and graphs for insight extraction.

The key objectives of this project were:

1. To Determine Which City or Station Has the Highest Pollution Levels

This objective focused on identifying cities and stations with the highest average pollutant concentrations, particularly **PM10**, **SO_2**, and **NH_3**.

- **Why it's important:**
  Identifying the most polluted areas helps environmental authorities prioritize mitigation strategies and allocate resources for pollution control.
- **How it was done:**
  Using **Pandas**, the dataset was grouped by city and station, and mean values of pollutant concentrations were calculated. The results were sorted to reveal the worst-affected locations.
- python CopyEdit
- top_polluted_cities = df.groupby('city')['pollutant_avg'].mean().sort_values(ascending=False)
-

2. To Compare Pollution Levels Across Different Pollutants

This objective explored which pollutants were most prevalent in the dataset and posed the greatest environmental threat.

- **Approach:**
  The data was grouped by pollutant_id, and the mean values were calculated to compare PM10, SO2, and NH3. Pie charts and bar graphs (using **Matplotlib** and **Seaborn**) were used to visualize their overall contribution.
- python CopyEdit
- pollutant_distribution = df.groupby('pollutant_id')['pollutant_avg'].mean()
-

3. To Analyze Temporal Trends in Air Quality

This section aimed to understand how pollution varied across time—by days, weeks, or seasons.

- **Method:**

The last_update column was converted to a datetime object. Data was resampled to daily or weekly frequency to calculate and visualize average pollution trends using **line plots**.

- python CopyEdit
- df['last_update'] = pd.to_datetime(df['last_update']) time_trends =
- df.set_index('last_update').resample('W')['pollutant_avg'].mean()
-

4. To Compare State-Wise Air Pollution Performance

This objective focused on comparing average pollutant levels across different Indian states.

- **Steps Taken:**
  The dataset was grouped by state, and average values of each pollutant were calculated. Horizontal bar charts highlighted the most and least polluted states.
- python
- CopyEdit
- state_avg = df.groupby('state')['pollutant_avg'].mean().sort_values(ascending=False)

5. To Build Interactive and Static Visualizations

A final goal was to translate the findings into both static and interactive visual formats for clear, datadriven storytelling.

- **Tools Used:**
  - **Matplotlib & Seaborn** for standard visuals
  - **Plotly** for interactive bar, line, and pie charts
  - **Missingno** for missing value diagnostics

4.3 Analysis Results

The Python ecosystem enabled deeper, more dynamic analysis using a variety of tools and visual techniques:

Data Grouping and Aggregation

Using **Pandas groupby**, pollution data was grouped and summarized by:

- city, state, and pollutant_id last_update
- (for time-series analysis)

Metrics computed included:

- Mean, max, and min pollutant levels
- Weekly and daily averages
- Pollution trends over time

Missing Value Handling

- 'na' entries were converted to NaN using pandas.to_numeric(errors='coerce')
- Missing values in pollutant_min, pollutant_max, and pollutant_avg were filled with column-wise

means: python CopyEdit

df['pollutant_min'].fillna(df['pollutant_min'].mean(), inplace=True)

- **Missingno** was used to visualize missing patterns.

Sorting and Filtering

Python enabled flexible sorting/filtering to:

- Isolate top 10 polluted cities/stations
- Focus on specific pollutants (e.g., only PM10)
- Analyze pollution on specific dates or regions

Custom Calculations

- Average pollutant levels were computed from pollutant_min and pollutant_max when pollutant_avg was missing.
- Time-based resampling helped detect seasonal or temporal shifts.

4.4 Visualizations and Insights

Python visualizations revealed several critical findings:

1.Distribution Plot (Univariate)
Visual: Histograms or KDE plots for pollutants like PM2.5, PM10, NO_2, etc.
Insights:

- PM2.5 and PM10 distributions are right-skewed, indicating frequent mild pollution events but occasional severe spikes in pollutant concentration.
- NO_2 displays a more uniform distribution, suggesting consistent traffic or industrial emissions over time.
- Some pollutants show bimodal distribution patterns, likely reflecting seasonal changes or weekday vs weekend pollution dynamics.

2. Time Series Line Plot (Pollutant vs Date)
Visual: Line graphs of pollutant concentrations over time.

Insights:

- PM2.5 and PM10 exhibit seasonal spikes, particularly during winter months. This trend may be attributed to temperature inversion and limited dispersion during colder seasons.
- Ozone (O_3) levels tend to peak in summer, which aligns with increased photochemical reactions facilitated by higher sunlight exposure.
- Sharp dips in pollution levels on specific dates may correspond to events like lockdowns, festivals, or rain, which help temporarily clear the air.

3.Box Plot (Pollutant by Month or Location)
Visual: Boxplots showing pollutant distribution across months or locations.

Insights:

- Higher median values of PM2.5 during December and January reinforce the impact of winter inversion and increased heating-related emissions.
- Some locations show frequent outliers, suggesting localized high pollution events. These areas may require stricter emission monitoring or policy intervention.
- Low interquartile ranges with sporadic outliers imply stable pollution levels punctuated by occasional spikes.

5. CONCLUSION

This project provided an in-depth, programmatic analysis of air pollution levels across multiple Indian cities using **Python as the core analytical tool**. Leveraging Python's robust data science libraries, we transformed a raw and incomplete dataset into a well-structured foundation for meaningful environmental insights.

The process began with **data cleaning and preprocessing**, where missing values in pollutant measurements (PM10, SO_2, and NH_3) were addressed through statistical imputation techniques such as mean replacement. Data types were corrected, timestamps were parsed for time-series analysis, and unnecessary records were filtered out — all using efficient and scalable Python code with libraries like **Pandas** and **NumPy**.

Once cleaned, the dataset was explored using a combination of **grouping**, **aggregation**, and **custom metrics** to evaluate pollution across geographic and temporal dimensions. Cities and states were ranked by average pollution levels, revealing critical hotspots. **Time-series analysis** uncovered seasonal trends, with pollution peaking during winter months, and **pollutant-wise comparisons** highlighted PM10 as the most persistent and hazardous pollutant overall.

The comprehensive analysis of the air quality dataset, carried out using Python-based data processing and visualization libraries, provides valuable insights into the temporal, spatial, and chemical behavior of key atmospheric pollutants. Through a combination of univariate, bivariate, and multivariate visualizations, clear seasonal patterns, pollutant interdependencies, and location-specific anomalies were identified.

The data reveals that particulate matter (PM2.5 and PM10) consistently exhibits elevated concentrations during the winter months, likely due to meteorological phenomena such as temperature inversions and reduced atmospheric dispersion. Gaseous pollutants such as nitrogen dioxide (NO_2) and carbon monoxide (CO) show strong correlations, indicating vehicular emissions as a primary source. In contrast, ozone (O_3) exhibits a distinct seasonal trend, peaking during the summer months due to increased photochemical activity under high solar radiation.

Furthermore, correlation analysis highlights significant interactions among pollutants, emphasizing the need for integrated pollution control strategies. Boxplots and scatter matrices expose the presence of high-frequency outliers in specific regions, suggesting the necessity of localized policy interventions and enhanced monitoring.

This analysis not only contributes to a better understanding of air quality dynamics but also supports evidence-based policymaking by identifying high-risk time periods and pollutant combinations. Future studies could integrate meteorological and demographic data to build predictive models, thereby enhancing real-time air quality management and urban planning.

The analysis was supported by compelling **visualizations** created with **Matplotlib**, **Seaborn**, and **Plotly**. These included interactive line charts, bar graphs, pie charts, and heatmaps — all designed to enhance interpretability and make the data engaging for both technical and non-technical audiences. Python's ability to build **dynamic, filterable, and customizable visuals** added significant value to the project, allowing for deeper exploration and real-time analysis.

Beyond visualization, Python enabled **reproducibility and scalability** — key for expanding this analysis to larger datasets or integrating real-time monitoring systems in the future. Compared to traditional tools, Python offers greater flexibility for automation, integration with APIs, and advanced analytics.

In summary, the project not only achieved its analytical goals but also demonstrated the power of Python as a modern platform for environmental data science. It highlighted how open-source tools can be effectively utilized to address real-world challenges like air pollution, turning raw environmental data into knowledge, insight, and impact. The work lays a strong foundation for future projects involving **predictive modeling**, **geospatial mapping**, and **health-risk correlation**, ultimately contributing to a more sustainable and data-informed approach to environmental management.

6. FUTURE SCOPE

This project effectively demonstrates how Python can be leveraged as a powerful tool to extract, process, analyze, and visualize air quality data using a structured and scalable approach. While this analysis focused primarily on descriptive analytics and static reporting, the flexibility of Python and its ecosystem opens several opportunities to expand this work into real-time monitoring, predictive modeling, and interactive platforms.

The following enhancements outline the broader future potential of this Python-based project:

1. Integration of Real-Time Pollution Monitoring via APIs and IoT

Future developments could include the integration of **real-time air quality data** via open APIs such as:

- **Central Pollution Control Board (CPCB) API**
- **OpenAQ API**
- **BreezoMeter API**
- **IQAir / AirVisual API**

Python libraries like requests, json, and pandas can be used to fetch, parse, and analyze data in real time. Additionally, Python can be integrated with **IoT devices** and sensors (via MQTT, HTTP, or serial interfaces) to track live air pollution levels and automatically update dashboards or trigger alerts when thresholds are exceeded.

2. Advanced Visualization Using Interactive Dashboards

While this project used static plots with **Matplotlib** and **Seaborn**, future enhancements could involve **fully interactive dashboards** built using:

- **Plotly Dash**
- **Streamlit**
- **Bokeh**

These platforms allow real-time data exploration with filters, sliders, maps, and live updates — offering a user-friendly interface for environmental researchers, students, or municipal bodies.

3. Predictive Modeling with Machine Learning

Using Python's machine learning libraries like **scikit-learn**, **XGBoost**, or **TensorFlow**, the project can transition from descriptive to **predictive analytics**:

- **Time Series Forecasting (ARIMA, LSTM, Prophet)** to forecast future pollutant levels.
- **Classification Models** to predict "high-risk" pollution days.
- **Anomaly Detection** using unsupervised algorithms to detect outlier readings or environmental irregularities.

These capabilities can support early warning systems and proactive policy decisions for cities experiencing seasonal pollution spikes.

4. Geospatial and Satellite-Based Pollution Mapping

Python offers powerful geospatial tools such as:

- **GeoPandas**
- **Folium**
- **Kepler.gl**
- **Rasterio / Earth Engine Python API**

These tools can be used to map pollution data on interactive city maps or satellite overlays. Visualizations can highlight:

- Urban vs. rural pollution distribution
- Pollution sources near roads, industrial zones, or low-vegetation areas
- Hotspot tracking and spatial clustering of pollutants

5. Public Health Impact Correlation and Risk Assessment

By combining air quality data with **health records** or **hospital data** (e.g., via public health APIs or CSV records), Python can be used to:

- Correlate pollution levels with respiratory disease incidence
- Predict risk zones for asthma, bronchitis, or cardiac issues
- Conduct spatial health risk assessments

Libraries like statsmodels and scipy would enable regression and statistical testing, while tools like Altair or Plotly could visualize disease exposure correlations.

6. Expanding Dataset Scope, Depth, and Diversity

To strengthen the analytical power of the project, future versions could incorporate:

- **Historical data** spanning multiple years (for trend analysis)
- **Additional pollutants** (e.g., CO, NO_2, PM2.5, O_3)
- **Meteorological variables** (temperature, humidity, wind speed) via weather APIs **Demographic**
- **overlays** (population, age, income) for impact segmentation

This would enable **multivariate environmental modeling** and deeper causal insights.

Conclusion of Scope

By integrating real-time data streams, predictive analytics, advanced mapping tools, and public health statistics, this project can evolve into a robust **air quality intelligence platform**. Powered by Python, it could serve:

- Urban planners in identifying and mitigating pollution hotspots
- Public health agencies in monitoring environmental risk
- Researchers and policymakers seeking data-driven insight
- Citizens interested in understanding and protecting their local air quality

Ultimately, Python's flexibility, scalability, and ecosystem make it an ideal foundation for the next generation of environmental monitoring and decision-making tools.

Linkedin post link:- https://www.linkedin.com/posts/aarupadaiyarkj_python-dataanalysis-airquality-activity

7.REFERENCES

[1] National Archives, "Air Pollution", *Data.gov*, [Online]. Available: [2] M. Friendly, "The History of the Modern Graph," *Statistical Science*, vol. 14, no. 1, pp. 1–14, 1999.[3] T. Alwafi, "A Study on Air pollution

," *IEEE International Conference on Data Science and Analytics* ,:- pp. 234-238, 2020.[4] Microsoft Corporation, “Create a PivotTable to analyze worksheet data,” *Microsoft Support*, [Online]. Available: https://support.microsoft.com/en-us/excel[5] K. Kaushik, "Web Analytics 2.0: The Art of Online Accountability and Science of Customer Centricity", *Sybex*, 2009.

[6] A. Gandomi and M. Haider, “Beyond the Hype: Big Data Concepts, Methods, and Analytics,” *International Journal of Information Management*, vol. 35, no. 2, pp. 137–144, Apr. 2015.

[7] N. Stieglitz, S. Mirbabaie, B. Ross, and M. Neuberger, “Social Media Analytics – Challenges in Topic Discovery, Data Collection, and Data Preparation,” *International Journal of Information Management*, vol. 39, pp. 156–168, Dec. 2018.