AISC2006 GROUP 3

Air Wise: The AQI Oracle

REPORT

Team members:

Vivek Sharma (Leader)	-	500225198
Divya Hanspal (Deputy Leader)	-	500225305
Jonathan Fred Braganza	-	500223773
Ajit Atul Kulkarni	-	500223502
Jasdeep Kaur Gill	-	500224259
Vishruti Paresh Avlani	-	500224491
Ashtami Vijayan Pillai	-	500224260
Vihar Parekh	-	500210512
Rutvik Trivedi	-	500223490
Nabila Mansoor	-	500223650

TABLE OF CONTENT

Abstract

“Air Wise: The AQI Oracle" employs cutting-edge machine learning to offer real-time air quality forecasts. Stay informed, plan outdoor activities, and safeguard your health with this advanced tool. A smart choice for a cleaner, safer environment.

We have worked on historical data as well as live data and integrated both of them with our demo.

Data Scrapping

Executive Synopsis:

The daily Air Quality Index (AQI) for areas of six Canadian cities—Hamilton (Downtown and Mountain), Ottawa (Downtown), Windsor (Downtown), and Toronto (North and West)—is thoroughly analyzed in this research. The amounts of different gases emitted into the atmosphere daily were the focus of the data, which was obtained using web scraping methods from official government websites.

Techniques:

Information from official government pages focused on monitoring the quality of the air in specified regions was scraped during the data collection procedure. Particulate matter (PM2.5 and PM10), carbon monoxide (CO), nitrogen dioxide (NO2), sulfur dioxide (SO2), and ozone (O3) are the main gases that are considered for analysis.

Data Collection: Hamilton Downtown and Mountain: The daily concentrations of PM2.5, PM10, NO2, SO2, O3, and CO for both the downtown and mountain areas were the focus of the data collected from the Hamilton Air Quality Reports.

Ottawa Downtown: Data was taken from the City of Ottawa's Air Quality Health Index (AQHI) reports, with a focus on the daily concentrations of different gases in the downtown region.

Windsor Downtown: Made use of information from the Windsor-Essex County Health Unit website, highlighting daily changes in PM, NO2, SO2, O3, and CO concentrations in the downtown region.

Toronto North and West: Data on daily air quality levels for PM2.5, PM10, NO2, SO2, O3, and CO were gathered from the Toronto Public Health Air Quality Index for both the northern and western parts of Toronto.

Evaluation:

To find patterns, variances, and possible relationships in the daily air quality levels across the designated areas within each city, the gathered data underwent thorough examination. To comprehend the relative concentrations of various gases in each location, comparative analyses were performed. The results are intended to shed light on regional variations in air quality and possible environmental effects.

Data Cleaning

Data cleaning is a pivotal phase in the development of models within data science and machine learning, encompassing essential steps to prepare datasets for analysis. Initially, this involves addressing missing data by employing techniques like imputation using means, medians, or advanced methods such as interpolation. Following this, outliers are managed by strategies like trimming or transforming data points significantly differing from the norm. Eliminating duplicate records ensures dataset accuracy, while normalization and scaling standardize numerical features for equitable contributions to the model. Categorical variables are converted into machine-interpretable formats through techniques like one-hot encoding or label encoding for effective handling. Additionally, this phase includes feature engineering to enhance predictive potential and rectify inconsistencies or formatting errors for dataset uniformity. Overall, data cleaning ensures refined and reliable datasets for model building, enabling models to derive meaningful insights without being affected by noise or errors.

Data cleaning emprises various steps like handling missing values, removing duplicates, handling outliers and so on. For handling outliers there are various techniques like z-score, IOR method and MAD method. For our project we chose turkeys fence which was like IOR method

Turkey's Fence, attributed to the statistical approach developed by John Tukey, is a method used to detect and address outliers within a dataset. This technique involves the calculation of the interquartile range (IQR), which represents the range between the dataset's first quartile (25th percentile) and the third quartile (75th percentile). Outliers are identified as data points lying beyond the range of Q1−k×IQRQ1−k×IQR to Q3+k×IQRQ3+k×IQR, where kk typically assumes values of 1.5 or 3.

Data points falling below Q1−k×IQRQ1−k×IQR or exceeding Q3+k×IQRQ3+k×IQR are considered outliers. These outliers can be managed through several methods, such as removal from the dataset, capping or flooring their values, or applying transformations to minimize their influence on subsequent analyses or models.

The strength of Turkey's Fence lies in its ability to account for the spread and variability of data, providing a robust means of identifying extreme values that could significantly impact statistical analyses or machine learning models.

Model Building

In this project, we have used 3 models:

Regular Model: We have used 2 normal model for our project

1. Decision Tree Regressor
2. Random Forest Regressor

Time-series Model: Regular models were not given result, so we have used time-series model as the data we are analyzing is time-based. We have used 1 model for time-series.

2. Long Term Short Memory

Data Storage

In this project, we have detailed the setup and functionality of an API integrated with Weather API, which collects real-time weather data for various cities in the Greater Toronto Area (GTA). The data is extracted through a Python script, and a Flask endpoint is employed to store this information in a Firestore collection. The entire system is designed to operate autonomously, ensuring 24/7 uptime and data retrieval.

The Weather API endpoint used for data retrieval is 'http://api.weatherapi.com/v1/current.json?key=#key#&q=m1v&aqi=yes'. The following key fields are collected for each city:

- Location
- Current name
- Country
- Longitude (Lon)
- Latitude (lat)
- Local time
- Local time in datetime format
- Date
- Time
- Temperature in Celsius (temp_c)
- Temperature in Fahrenheit (temp_f)

- Condition text
- Wind speed in kilometres per hour (wind_kph)
- Wind degree
- Wind direction (wind_dir)
- Pressure in millibars (pressure_mb)
- Precipitation in millimeters (precip_mm)
- Humidity
- Visibility in kilometers (vis_km)
- UV index (uv)
- Gust speed in kilometers per hour (gust_kph)
- The data is fetched using a Python script, showcasing the versatility and ease of use of the Weather API.

To automate the data storage process, a Flask endpoint was implemented. When this endpoint is accessed, it triggers a function to store the collected data into a Firestore collection. Firestore is a NoSQL document database that provides a scalable solution for real-time data storage.

This integration allows for seamless, efficient, and secure storage of weather data for analysis, reporting, or any future applications.

The weather data is collected for a comprehensive list of cities in the GTA:

1. Ajax
2. Aurora
3. Brampton
4. Brock
5. Burlington
6. Caledon
7. Clarington
8. East Gwillimbury
9. Georgina
10. Halton Hills
11. King
12. Markham

13. Milton
14. Mississauga
15. Newmarket
16. Oakville
17. Oshawa
18. Pickering
19. Richmond Hill
20. Scugog
21. Toronto
22. Uxbridge
23. Vaughan
24. Whitby
25. Whitchurch-Stouffville

This extensive list ensures a thorough representation of weather conditions across the GTA.

To achieve 24/7 operation, the Flask endpoint is triggered on system reboot. This functionality, combined with deployment on Replit, ensures that the entire system runs autonomously, providing consistent and up-to-date weather data without manual intervention. The integration of WeatherAPI, Python scripting, Flask endpoint, and Firestore storage creates a robust and autonomous system for collecting and storing real-time weather data for multiple cities in the GTA. This system is well-equipped to support various applications such as weather forecasting, analysis, and reporting. The comprehensive list of cities ensures a broad coverage of weather conditions, making it a valuable resource for users in the Greater Toronto Area.

User Interface

For user interface, we have chosen Flask as the backend for the Air Quality Index website primarily because of its lightweight and minimalistic nature, which perfectly suited the simple and focused requirements of the project.

Additionally, Flask's microframework architecture allowed for easy integration with the AI model, which was built in Python, enabling efficient communication between the web application and the model.

While Django offers a more robust set of features, its extensive framework would have been overkill for this project, making Flask the more appropriate choice for a streamlined and responsive solution.

TECHNICAL DIFFICULTIES

The first technical difficulty we faced is while data cleaning in terms of outlier removal. When we tried to remove the outliers, the original data was becoming an outlier.

The second difficulty we faced was on the Flask side, where it was creating problems while transferring data from Flask to local storage.

The last difficulty we faced was of connection in the task of deployment

CONCLUSION

To conclude the project, we are now able to connect live data as well as historical data into our deployment. We can suggest local people about the precautions taken at the time of high pollutant concentration

This suggestion is based on values of Air Quality Health Index. It depends on the quality of air.