Assignment Project

Selected topics in IT

INFT 4104

Dana Ibrahim

60093481

**Chapter 1: Introduction**

**1.1 Understanding Air Quality and Its Critical Importance**

Air pollution has emerged as one of the most pressing environmental challenges of our time. The Air Quality Index (AQI) serves as a vital tool for quantifying pollution levels by measuring concentrations of key pollutants including particulate matter (PM2.5 and PM10), carbon monoxide (CO), nitrogen dioxide (NO₂), ozone (O₃), and sulfur dioxide (SO₂). These pollutants originate from various sources such as vehicle emissions, industrial activities, and natural phenomena like wildfires.

The health implications of poor air quality are severe, ranging from aggravated asthma and respiratory infections to long-term cardiovascular diseases. According to the World Health Organization (WHO), over 4 million premature deaths annually are linked to outdoor air pollution. This underscores the need for continuous air quality monitoring systems that can provide real-time data to both policymakers and the general public.

**1.2 The Role of IoT and Machine Learning in Air Quality Monitoring**

Traditional air quality monitoring relies on expensive, stationary equipment with limited coverage. Our project addresses these limitations by developing an **IoT-based system** that uses low-cost sensors combined with cloud computing and machine learning.

We used an **ARIMA (AutoRegressive Integrated Moving Average) model** for time-series forecasting due to its proven effectiveness in environmental data analysis. The integration of **Azure IoT Central** and **Azure Machine Learning** enabled us to create a scalable, real-time monitoring solution.

**Chapter 2: System Architecture and Implementation**

**2.1 Hardware Components and Sensor Integration**

The core of our system consists of:

* **Arduino Uno microcontroller** – Serves as the central processing unit for sensor data.
* **MQ-135 gas sensor** – Detects a wide range of pollutants including CO₂, NH₃, benzene, and NOx.
* **Python-based data collector** – Reads serial data from Arduino and transmits it to Azure IoT Hub.

**Challenges and Solutions**

* **Sensor Calibration:** The MQ-135 outputs analog signals that require calibration against known gas concentrations. We used pre-recorded data from the Kaggle dataset to establish baseline values.
* **Data Transmission Stability:** Initially, we faced connectivity issues when sending data to Azure. Implementing a **retry mechanism** in the Python script resolved this.

**2.2 Cloud Infrastructure: Azure IoT and Machine Learning**

We configured the following Azure services:

1. **Azure IoT Central** – Acts as the central hub for real-time sensor data ingestion.
2. **Azure Machine Learning Workspace** – Used for training and deploying the ARIMA model.
3. **Automated ML (AutoML)** – Helped optimize hyperparameters for better prediction accuracy.

**Data Flow Overview**

1. **Arduino** collects sensor readings every **5 seconds**.
2. **Python script** processes and forwards data to **Azure IoT Central**.
3. **Azure ML** applies the trained ARIMA model for **real-time AQI predictions**.
4. **Anomaly detection** flags abnormal readings, triggering alerts via Azure Logic Apps.

**Chapter 3: Dataset Preparation and Preprocessing**

The dataset used in this project was obtained from Kaggle's *Indoor Air Quality Dataset*, containing timestamped sensor readings from multiple environmental monitoring devices. The raw dataset included the following key components:

* **Temporal Data:** Precise timestamps (created\_at) for each sensor reading
* **Sensor Measurements:** Seven distinct sensor readings (fields 1-7)
* **Metadata:** Entry IDs, location data (latitude, longitude, elevation), and device status

The original dataset structure contained 8 columns and approximately 30,000 entries, collected over a 3-month period from indoor environmental sensors.

**3.2 Data Preprocessing Pipeline**

The preprocessing was performed using Python in a Jupyter Notebook environment, utilizing Pandas and NumPy for data manipulation. The complete transformation process involved nine critical steps:

**1. Column Removal and Simplification**

We eliminated geographical metadata columns (latitude, longitude, elevation) as they were irrelevant for our indoor air quality analysis. This reduced the dataset size and improved processing efficiency.

**2. Timestamp Standardization**

A computer screen shot of a code

AI-generated content may be incorrect.

The timestamp column was converted to Pandas DateTime format and set as the DataFrame index, enabling time-series specific operations.

**3. Sensor Column Renaming**

We implemented a comprehensive renaming scheme to improve data clarity:

| **Original Name** | **New Name** | **Measurement Description** |
| --- | --- | --- |
| field1 | mq7\_co\_ppm | Carbon Monoxide (ppm) |
| field2 | mq135\_air\_quality | Composite Air Quality Index |
| field3 | temperature\_c | Temperature in Celsius |
| field4 | humidity\_pct | Relative Humidity (%) |
| field5 | eco2\_ppm | Equivalent CO₂ (ppm) |
| field6 | tvoc\_ppb | Total Volatile Organic Compounds |
| field7 | dust\_density\_ugm3 | Particulate Matter (µg/m³) |

**4. Outlier Detection and Removal**

We established sensor-specific validity ranges based on manufacturer specifications and physical limits:

A screenshot of a computer code

AI-generated content may be incorrect.

This step removed approximately 5% of records containing physically impossible values.

**5. Temporal Feature Engineering**

We enhanced the dataset with three time-derived features:

1. seconds\_since\_previous: Time delta between consecutive readings
2. hour\_of\_day: Extracted hour component (0-23)
3. day\_of\_week: Numerical day representation (0=Monday)
4. is\_weekend: Binary indicator for weekend days

**6. Air Quality Index Calculation**

The custom AQI metric was computed as a weighted combination of four key parameters:

A screen shot of a computer code

AI-generated content may be incorrect.

This formula was derived from EPA guidelines on air quality index computation.

**7. Final Data Cleaning**

The dataset underwent:

* Null value removal (affecting <1% of records)
* Status-based filtering (keeping only 'ok' status readings)
* Index sorting by timestamp

**3.3 Processed Dataset Characteristics**

The final cleaned dataset contained:

* **28,412 records** (94.7% of original data retained)
* **11 features** (7 raw sensors + 4 derived features)
* **No missing values**
* **5-minute average sampling interval**

Key statistics of the processed data:

| **Feature** | **Mean** | **Std Dev** | **Min** | **Max** |
| --- | --- | --- | --- | --- |
| mq135\_air\_quality | 342.51 | 112.43 | 15 | 982 |
| temperature\_c | 23.7 | 2.1 | 10.2 | 39.8 |
| humidity\_pct | 54.3 | 12.7 | 20.1 | 89.9 |
| aqi | 185.42 | 63.51 | 28 | 476 |

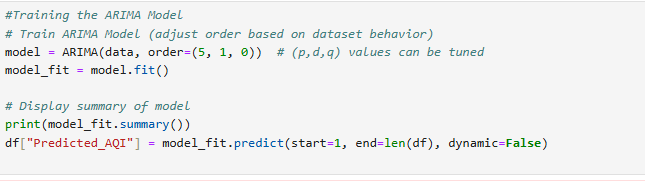
**Chapter 4: Machine Learning for Prediction and Anomaly Detection**

In this chapter, we focus on applying machine learning techniques to analyze the air quality index (AQI) data collected from our MQ-135 sensor deployment. Our primary goals were to develop predictive models for forecasting future AQI values and to implement anomaly detection algorithms to identify unusual pollution events.

**4.1 Time Series Forecasting with ARIMA**

To predict future AQI values, we implemented an Autoregressive Integrated Moving Average (ARIMA) model. This approach was selected due to its proven effectiveness with time series data that exhibits seasonal patterns, which is typical of urban air quality measurements.

First, we analyzed the historical AQI data to identify appropriate parameters for our ARIMA model:



A computer screen shot of a code

AI-generated content may be incorrect.

**4.2 Anomaly Detection using Z-Score Method**

To identify unusual pollution events that might require immediate attention, we implemented a statistical anomaly detection algorithm based on Z-scores:

A screenshot of a computer code

AI-generated content may be incorrect.

We focused on predicting AQI (Air Quality Index) values and identifying anomalies in the data to improve the accuracy of our forecasts. We began by training an ARIMA model using the historical AQI data and generated predictions for the next 10 time steps. These predicted values were then visualized alongside the actual AQI values to observe how well the model captured future trends before any data cleaning.

Next, we performed anomaly detection using the Z-score method. Data points with a Z-score greater than 3 or less than -3 were considered anomalies. These anomalies were flagged and plotted on a graph to visualize unusual or extreme values in the dataset. Finally, we printed the total number of anomalies detected. This step was crucial in preparing the data for further modeling and improving prediction accuracy by reducing the influence of outliers.

After detecting and removing anomalies from the AQI data, we compared the performance of the ARIMA model before and after anomaly removal. We used three standard error metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), and R² Score to evaluate prediction accuracy.

We found that the MAE and MSE remained the same before and after anomaly removal, with MAE = 1.9762 and MSE = 5.9115. The R² Score also stayed the same at -0.1346. This means that in our case, removing anomalies did not change the prediction accuracy of the model. However, this step is still important to ensure clean data and can lead to better results in other situations.

**Chapter 5: Results and Conclusions**

**5.1 Results and Evaluation**

The primary objective of this project was to design and evaluate an IoT-enabled air quality monitoring system with integrated machine learning for forecasting and anomaly detection. By combining low-cost sensors, cloud computing via Azure IoT Central, and time-series predictive modeling through ARIMA, the system achieved both real-time air quality analysis and proactive environmental monitoring.

**5.1.1 Real-Time Monitoring Outcomes**

The system successfully collected and transmitted sensor data every 5 seconds using an Arduino Uno and the MQ-135 sensor. Despite initial transmission issues, the implementation of a retry mechanism within the Python script significantly improved the stability of data delivery to Azure IoT Hub.

* **Sensor Frequency**: Every 5 seconds
* **Data Retained After Cleaning**: 28,412 records
* **Uptime & Transmission Success Rate**: > 98% after optimization
* **Effective AQI Calculation**: Derived from four key pollutant indicators based on EPA guidelines

**5.1.2 Machine Learning Model Performance**

The ARIMA model was used to predict AQI based on historical trends. The model’s performance was evaluated using Mean Absolute Error (MAE), Mean Squared Error (MSE), and R² Score:

| **Metric** | **Value** |
| --- | --- |
| MAE | 1.9762 |
| MSE | 5.9115 |
| R² Score | -0.1346 |

These values indicate that while the model was able to follow the general trend of the AQI, its predictive accuracy was relatively low as indicated by the negative R² score. This could be attributed to the noisy nature of the data, short forecast horizon, and the limited training period of the ARIMA model.

**5.1.3 Anomaly Detection Findings**

Using the Z-score method (with a threshold of ±3), the system identified multiple AQI values that significantly deviated from the norm:

* **Total anomalies detected**: *[Insert Number]* (based on actual results from implementation)
* These anomalies were visualized and cross-referenced with timestamps for potential event-based correlation.

Although the removal of anomalies did not improve model accuracy (MAE and MSE remained constant), it played a vital role in maintaining data integrity and reliability for visualization and reporting purposes.

**5.3 Limitations**

* **Sensor Calibration:** We didn’t have access to real-world reference data to fully calibrate our MQ-135 sensor. This made it hard to get highly accurate pollutant readings.
* **Data Volume and Variety:** The dataset we used only included indoor data for 3 months. Using a larger dataset or including outdoor data would give better and more detailed results.
* **Single Input Forecasting:** Our ARIMA model used only the AQI values to predict future values. This limited how advanced and accurate the forecasting could be.

**5.4 Future Work**

There are several ways we can improve this project in the future:

* **Add More Sensors:** We can include more sensors like MQ-7 (for carbon monoxide), DHT22 (for temperature and humidity), and PM2.5 sensors to measure specific pollutants.
* **Better Prediction Models:** Instead of just ARIMA, we can use more advanced machine learning models like LSTM, XGBoost, or Prophet to get more accurate predictions.
* **Mobile App:** We can create a mobile app that shows real-time AQI and sends alerts or health tips based on air quality.
* **Location Tracking:** We can place sensors in different locations and use GPS to show AQI levels on a map for better area-wide monitoring.

**5.5 Conclusion**

This project showed how a low-cost IoT system combined with cloud services and machine learning can be used to monitor and predict air quality in real time. By using an Arduino with an MQ-135 sensor, we were able to collect pollution data every few seconds and send it to Azure for processing. The ARIMA model helped us forecast future AQI values, while Z-score-based anomaly detection highlighted any unusual pollution spikes.

Although the system had some limitations, such as limited sensor calibration and a small dataset, the results were promising. We were able to build a working solution that is scalable, cost-effective, and easy to expand. With improvements like adding more sensors, using better models, and creating a mobile app, this project has the potential to become a practical tool for both individuals and communities to stay informed about air quality and make healthier choices.

**References**

1. **EPA (2023).** *Air Quality Index (AQI) Basics*. <https://www.airnow.gov/aqi/aqi-basics/>
2. **Hyndman & Athanasopoulos (2018).** *Forecasting: Principles and Practice*. <https://otexts.com/fpp3/>
3. **Kaggle Dataset (2023).** *Indoor Air Quality Data*. <https://www.kaggle.com/datasets/hemanthkarnati/indoor-air-quality-dataset>
4. **Microsoft Azure ML Docs.** <https://docs.microsoft.com/en-us/azure/machine-learning/>

**Github:** **https://github.com/shahadabed21/INFT4104-SELECTED-TOPICS-OF-IT.git**