Air Quality Temperature and NO2 Predictive Models

Andrew Roehr, Dheemanth Rajakhumar, and Jory Hamilton

Shiley-Marcos School of Engineering, University of San Diego

Abstract

Italian researchers, De Vito *et al.* (2005) investigated how a cost-effective, multi-sensor system could leverage neural networks to measure the extent of pollution in an urban setting. Their goal involved creating a more cost-effective method than purchasing conventional sensors that used spectrometers. The team collected data and later released it as the Air Quality dataset. The purpose of this report is to demonstrate a comprehensive data pipeline for an IoT sensored Air Quality dataset, starting with data cleaning and EDA to address missing values, outliers, and proper datetime formatting. Two deep learning models were developed: an LSTM for time-series temperature prediction and a deep feed-forward neural network for NO2 forecasting. The LSTM model produced predictions that closely aligned with the actual temperature values and achieved lower error metrics, indicating higher accuracy. In contrast, while the deep neural network captured general trends in NO2 levels, its predictions were less precise and exhibited larger discrepancies.

Keywords: LSTM, time series

Introduction

Amit Kapoor (2019) communicates that the Internet of Things (IoT) includes sensors connected to the Internet (pg. 7). One example involves Smart Grids, such as turning off a car charger in one house for about 15 minutes when a water heater in another location can help reduce the overall load on a transformer (Marbut, n.d.). Smart Grids allow for better pricing, energy use, location forecasting, energy storage, generation, and distribution ("The Role of IoT In Smart Grid Technology and Applications," 2022). Likewise, Agricultural IoT systems integrate AI to enhance precision farming by analyzing real-time data and providing hyper-localized weather predictions (Zhang & Qiao, 2024). IoT sensors monitor variables like soil moisture, temperature, and humidity, while AI models process this data to forecast microclimatic conditions at high spatial resolution.

These examples of how IoT can be utilized can be expanded to environmental considerations. Italian researchers, De Vito *et al.* (2005) investigated how a cost-effective, multi-sensor system could leverage neural networks to measure the extent of pollution in an urban setting. Their goal involved creating a more cost-effective method than purchasing conventional sensors that used spectrometers. The team collected data and later released it as the Air Quality dataset.

This report aims to demonstrate a comprehensive data pipeline for the IoT-sensored Air Quality dataset, starting with data cleaning and EDA to address missing values, outliers, and proper datetime formatting. Two deep learning models were developed: an LSTM for time-series temperature prediction and a deep feed-forward neural network for NO2 forecasting.

Infrastructure Diagram and Reference Documentation

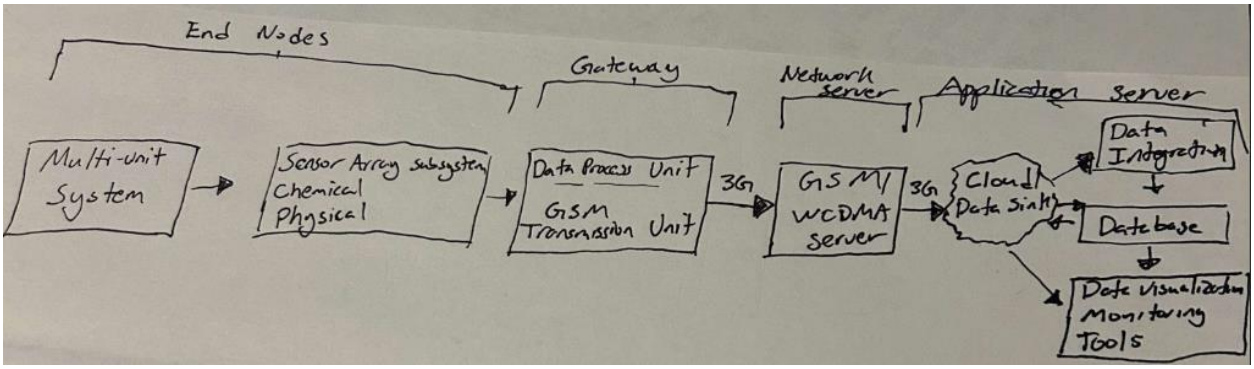

Figure #1: Infrastructure Diagram

Sensor nodes transfer chemical (atmospheric) and physical (temperature and humidity) data to the processing unit that has a 72-hour storage and 8-second rate. The GSM transfer unit utilized 3G to move data to a WCDMA server using WAP protocols.

De Vito *et al.* (2008) utilized real-time chemical sensors that monitor atmospheric composition to investigate air pollution. Figure #1 illustrates the infrastructure diagram. The research team selected the cheap and versatile Pirelli Labs multi-sensor system, each with 31 cm x 26 cm x 12 cm and 2.5 kg individual dimensions, described in three stages for their reliability and inexpensive cost.

1. Sensor Array
    a. Sensor array subsystem
        i. Five metal oxide chemiresistive sensors (details in Table 1)
    b. Two others slots were devoted to host commercial temperature and humidity sensors
2. Data Processing Unit
    a. A microcontroller board hosts a microprocessor capable of initial data processing (simple sensor fusion algorithms)

i. Up to 72-hour storage of measurements

ii. Eight-second sample rate

3. Communication Unit

    a. Global System for Mobile Communications data transmission unit signal conditioning and acquisition electronics

        i. Transfer mean sampled values to data sinks at 8, 15, or 60 intervals

    b. Solomon Ndungu and Erica Mixon (2021) would likely suggest the GSM unit in 2005 would operate with 3G (released in 2000) well before 4G was released in 2010. 144 Kbps - 2 Mbps speed

    c. Paul Smethers (2000) indicates that GSM ran on WAP protocols.

    d. Wang Yong (2005) conveyed that a common server GSM ran on or transitioned to during the study was WCDMA.

The research team attempted various architectures at the data center, primarily the 7-X-1 Feed-Forward network (i.e., X in [3,5,15,20]). Additional algorithm details include:

- Resilient back-propagation algorithm.
- MATLAB tansig function (hidden neuron transfer function).
- Early stopping as a measure to prevent over-training problems

For quality assurance, the research team employed a Lombardy Regional Agency for The research team employed a conventional monitoring station by the Lombardy Regional Agency for Environmental Prevention with a spectrometer for quality assurance. De Vito *et al.* researched whether the cheaper sensor system could cover more area for the same or at a lower cost than the conventional method; however, including the spectrometer allowed for better validation in their data analysis.

Dataset

Group #1 selected the Air Quality dataset published on Kaggle, which contains 9358 instances, five metal oxide/chemical sensors, and 15 features and was collected from March 2004 through February 2005 by an Italian research team. Further details can be found at the following URL and Table #1.

Dataset URL: https://archive.ics.uci.edu/dataset/360/air+quality

Table #1 - Air Quality Features

Variable Name	Type	Description	Units	Additional Info
Date	Date	NA	NA	DD/MM/YYYY
Time	Categorical	NA	NA	HH.MM.SS
CO(GT)	Integer	True hourly avg. CO conc.	mg/m^3	Reference analyzer
PTO8.S1(CO)	Categorical	Hr. avg. sensor response	NA	Tin oxide, CO targeted
NMHC(GT)	Integer	True hr. Avg. overall hydrocarbons	microg/m^3	Reference analyzer
C6H6(GT)	Continuous	Tr. hr. avg. benzene conc.	microg/m^3	Reference analyzer
PT08.S2(NMHC)	Categorical	Hr. avg. avg. sensor response	NA	Titania
NOx(GT)	Integer	Tr. hr. avg. NOx conc.	ppb	Reference analyzer
PT08.S3(NOx)	Categorical	Hr. avg. sensor response	NA	Tungsten oxide
NO2(GT)	Integer	Hr. avg. sensor response	NA	NA
PT08.S4(NO2)	Categorical	Hr. avg. sensor response	microg/m^3	Tungsten oxide
PT08.S5(O3)	Categorical	Hr. avg. sensor response		Indium oxide
T	Continuous	Temperature	℃	NA
RH	Continuous	Relative humidity	%	NA
AH	Continuous	Absolute humidity	NA	NA

Code and Documentation

The project code, readme file, and additional documents can be found using the following URL: https://github.com/Roehrkard/Final-ML-IoT-Design-Air-Quality

EDA Summary

The EDA created a Datetime column, converted other columns into numerical data, dropped NaN/NaT rows (~100) out of 9357. Two copies of that dataframe were created and modified to address outliers.

- df_no_outliers has removed rows with outliers in any column via IQR Method. The count for all rows decreased from 9357 to 5436 from this process. Outlier removal for any numerical column can be omitted or re-added in the df_no_outliers module, which allows for increasing the count value when prioritizing certain sensor data over others.
- df_median replaces outliers with the median conserving the count, the number of data points becoming a priority during model training, testing and validation evaluation.

Group #1 selected df_no_outliers for simplicity, ease of control (e.g., removing or not removing null values of any column category), and to preserve data integrity and reliability by ensuring the models trained using representative and non-extreme values. Imputation works best when there are small number of outliers, but 3921 out of 9357 (42%) qualified as an outlier via the IQR method. Imputation can skew data distribution when replacing null values with the median.

However, Group #1 determined there is also value in replacing missing values with their respective column's median could be beneficial if they provide valuable information, allowing a given model to generalize. The selected models could be retrained with the df_median

imputation dataframe that has already been prepared for use if research teams want to explore this option.

Tableau Air Quality Analysis

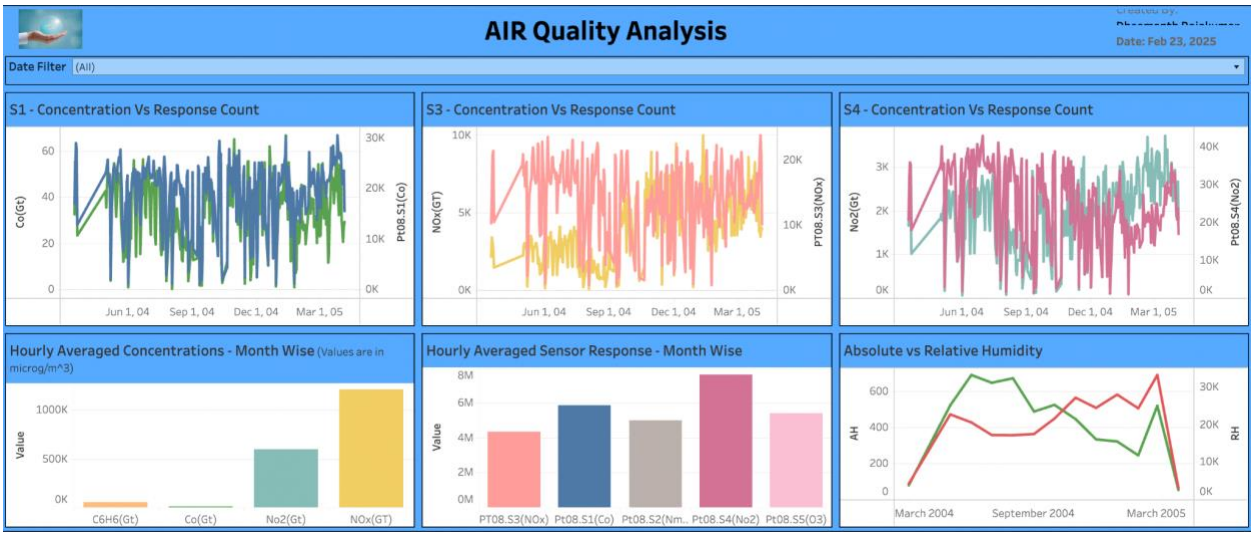

Figure #2 - Tableau Air Quality Analysis

Figure #2 provides an analytical summary of air quality based on the UCI Air Quality dataset, visualized through an interactive Tableau dashboard. The dataset comprises hourly readings of various air pollutants and meteorological variables, offering insights into environmental conditions over time. The visuals highlight key trends, correlations, and patterns that inform air pollution levels and their possible causes.

- Temporal Trends in Air Pollution:
    - Time series visualizations effectively capture fluctuations in pollutants like CO(GT) from S1, NOX(GT) from S3, and NO2(GT) from S4
    - The dashboard enables clear month-over-month comparisons, highlighting seasonal shifts in air quality
- Meteorological Impact on Air Quality

- Visualizations of temperature and humidity show that air pollution levels often increase under specific weather conditions
- Absolute humidity represents water vapor content based on volume, whereas relative humidity depends on volume at a given temperature

Model Results and Discussion

Deep Learning Time Series Predictor (LSTM for Temperature "T")

This code block first sorts the cleaned dataset by datetime and extracts the Temperature column, ensuring no missing values remain. It scales the temperature data between 0 and 1 using MinMaxScaler to prepare it for neural network training. Next, it creates time-series sequences with a 24-hour lookback, turning the problem into a supervised learning task with input sequences (X) and corresponding targets (y). A simple LSTM model is then built from scratch using TensorFlow/Keras, and it is trained on 80% of the data while validating the remaining 20%. Finally, the model's performance is evaluated on the test set, and predictions are inverse-transformed back to the original scale for direct comparison with the actual temperature values.

LSTM Interpretation

The model shows a low-scaled MSE but consistently underpredicts the actual temperature values, indicating a calibration issue. Although the losses decrease over epochs, the model struggles with accuracy, likely due to its simplicity. Improvements are needed to capture the temperature patterns better and improve prediction reliability for four seconds. The model achieved a low scaled test MSE, yet the inverse-transformed predictions are consistently lower than the actual temperature values, indicating systematic underprediction. Although the training and validation losses decreased steadily, the model still fails to capture the true magnitude of the

temperature. In summary, it learns trends but underestimates absolute values.

Optimized Deep Learning Time Series Predictor (LSTM for Temperature "T")

To optimize performance, the architecture was enhanced by stacking multiple LSTM layers with dropout layers to better regularize the network and capture more complex temporal patterns. Additionally, the learning rate was reduced to allow finer convergence, the number of training epochs was increased to 30, and the batch size was reduced to 16 for more granular updates. These modifications aim to improve the model's ability to learn and generalize, ultimately reducing the discrepancy between predicted and actual temperature values.

Optimized LTSM Interpretation

This code block optimizes the temperature prediction model by first using an outlier-corrected dataset to ensure that the scaling accurately reflects typical values. Temperature data is scaled to the [0, 1] range and converted into 24-hour lookback sequences for supervised learning. The optimized model features stacked LSTM layers with dropout for regularization and a final Dense layer with sigmoid activation to constrain outputs within the same scale. A lower learning rate of 0.002 and early stopping with a patience of 3 epochs are implemented to enable finer convergence and prevent overfitting. Finally, predictions are inverse-transformed back to the original scale for direct comparison with actual values.

Deep Learning Model for Predicting "NO2(GT)" (Feed-forward Neural Network)

This code block begins by filtering the cleaned dataset to remove rows with missing "NO2(GT)" values and selecting a set of sensor features as inputs along with "NO2(GT)" as the target variable. It then splits the data into training and testing sets and standardizes the features using StandardScaler to normalize their distribution. A deep feed-forward neural network is constructed using Keras, featuring three hidden Dense layers with ReLU activations and a final

output layer for regression. The model is compiled with the Adam optimizer and mean squared error loss, and it is trained for 50 epochs with a batch size of 32 while reserving 20% of the training data for validation. Finally, the model's performance is evaluated on the test set by calculating MSE and RMSE, and several sample predictions are printed alongside their corresponding actual values.

Optimized Deep Neural Network for NO2(GT) Prediction

This optimized model builds on the previous deep feed-forward network by adding several enhancements to improve performance. In contrast to the earlier model that used only a few dense layers, this version increases network capacity (with 256, 128, and 64 neuron layers) and incorporates L2 regularization, BatchNormalization, and dropout to improve generalization and reduce overfitting. Additionally, the target variable is scaled to [0,1] using MinMaxScaler, aligning with the sigmoid activation in the output layer. EarlyStopping and ReduceLROnPlateau callbacks are employed to adjust the learning rate and prevent unnecessary training dynamically, ensuring that the model converges more effectively. Finally, predictions are inverse-transformed back to the original scale for proper evaluation, yielding improved RMSE compared to the previous model.

Optimized Deep Neural Network

The optimized model shows a relatively low loss on the scaled data, indicating that it has learned some underlying patterns in the training process. However, when these predictions are converted back to the original scale, the errors remain substantial, with some predictions deviating notably from the observed values. The variability in performance suggests that while the model can capture general trends, it struggles to consistently produce accurate predictions across the full range of the target variable. Overall, the results indicate that further tuning of the

model architecture or additional feature engineering may be necessary to achieve more reliable forecasts.

Conclusion

This notebook presented a comprehensive workflow beginning with the exploratory analysis and cleaning of an IoT sensor dataset, including handling missing values, outliers, and proper datetime formatting. Two deep learning models were developed: an LSTM for time-series temperature prediction and a deep feed-forward neural network for predicting NO2 levels. The LSTM model produced predictions that closely aligned with the actual temperature values and achieved lower error metrics, indicating higher accuracy. In contrast, while the deep neural network captured general trends in NO2 levels, its predictions were less precise and exhibited larger discrepancies. Overall, the integrated approach demonstrates that, for this dataset, the LSTM outperformed the deep feed-forward model in generating accurate predictions, highlighting the importance of selecting appropriate architectures for specific tasks.

References

De Vito, S., Massera, E., Piga, M., Martinotto, L., & Di Francia, G. (2008). On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B* 129, pg. 750-757. Retrieved from https://doi.org/10.1016/j.snb.2007.09.060

Connected Industry. (2022). The role of IoT in smart grid technology and applications. Retrieved from https://www.iiot-world.com/industrial-iot/connected-industry/the-role-of-iot-in-smart-grid-technology-and-applications/

The role of IoT in smart grid technology and applications. Retrieved from

Kapoor, A. (2019). Hands-on artificial intelligence for IoT: Expert machine learning and deep learning techniques for developing smarking IoT systems. Packt Publishing, Limited.

Marbut, A. (n.d.). Presentation 1.1: IoT system components [lecture].

Ndungu, S. & Mixon, E. (2021). GSM (global system for mobile communication). Retrieved from https://www.techtarget.com/searchmobilecomputing/definition/GSM

Smethers, P. (2000). Position paper on WAP application usability in GSM markets. Retrieved from https://www.w3.org/2000/10/DIAWorkshop/smethers.html

Vito, S. (2016). *Air quality.* Retrieved from https://doi.org/10.24432/C59K5F

Yong, W. (2005). Resurrection of GSM core network at WCDMA age. Retrieved from https://www.zte.com.cn/global/about/magazine/zte-communications/2005/1/en_44/162324.html

Zhang, B., & Qiao, Y. (2024). AI, sensors, and robotics for smart agriculture. MDPI. https://doi.org/10.3390/agronomy14061180