

Environmental Monitoring and Pollution Prediction

MLOPS Final Project

Task 2: Pollution Trend Prediction with MLflow

Goals:

Model construction and deployment for predicting pollution trends and alerting conditions for high-risk days.

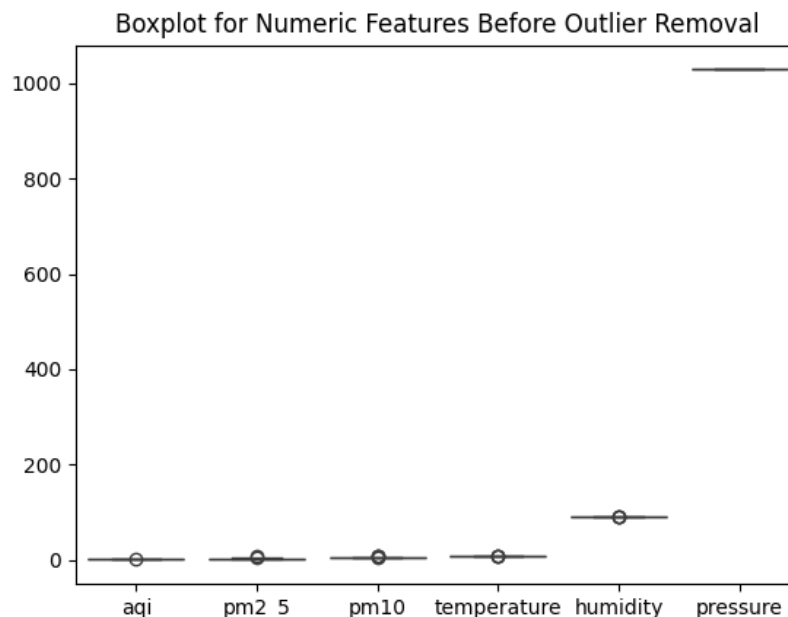
1. Data Preparation

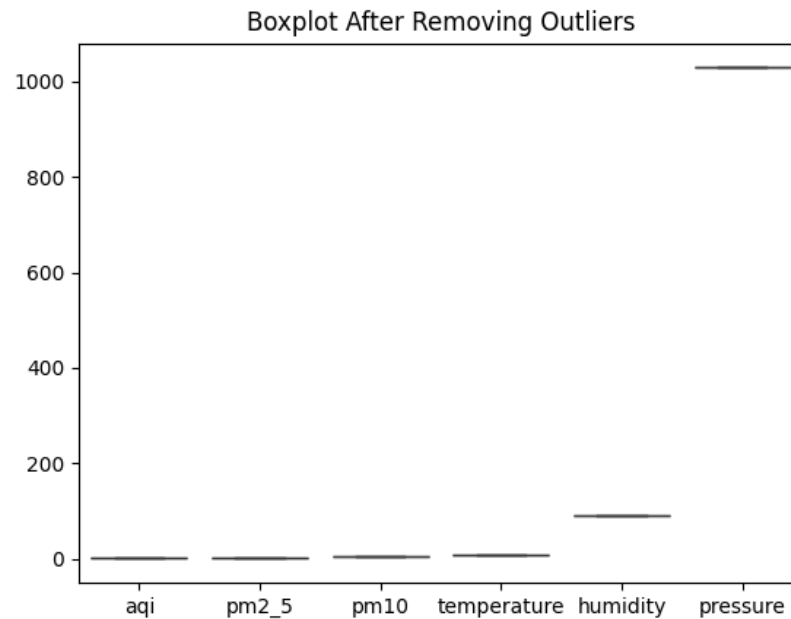
Process:

The environmental data preprocessing was done to provide it with input for model training.

Actions performed:

The data preparation involved merging pollution and weather datasets on the timestamp column to align data from both sources. Missing values in numeric columns (aqi, pm2_5, pm10, temperature, humidity, pressure) were imputed using the mean, with fallback to zeros for entirely missing columns. Outliers were removed using the Interquartile Range (IQR) method to ensure data consistency. Visualizations, such as boxplots, were used to validate preprocessing steps before and after outlier removal. The cleaned and processed dataset was saved as `cleaned_merged_data.csv` for further use in model training and evaluation.





2. Model Development

Models Implemented:

ARIMA:

- Configuration made to analyze stationary time-series data.
- Best model configuration=(0, 1, 0).

LSTM:

- Developed a two-layer LSTM model with 50 units per layer and a Dense output layer

Key Observations:

ARIMA Results:

- Metrics-RMSE=0.0, MAE=0.0.
- Problem-small dataset led to an over-fitting situation where ARIMA predictions went exactly with the actual values of data.
- Convergence warning is for optimization problems.

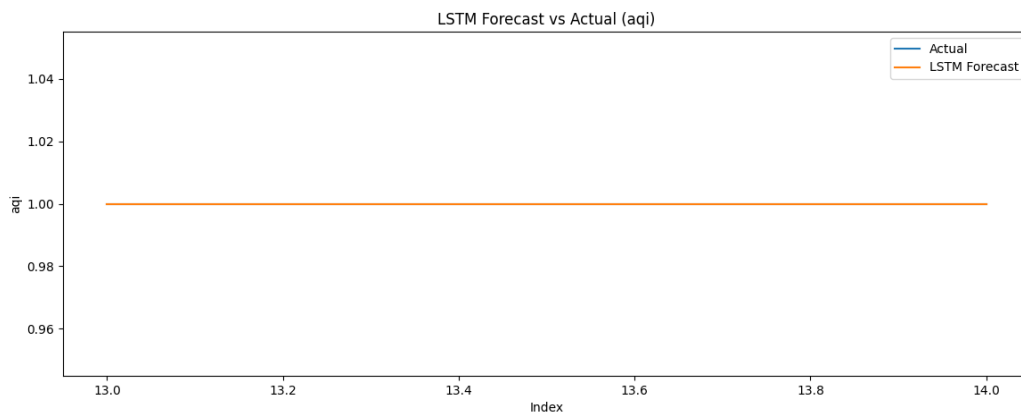
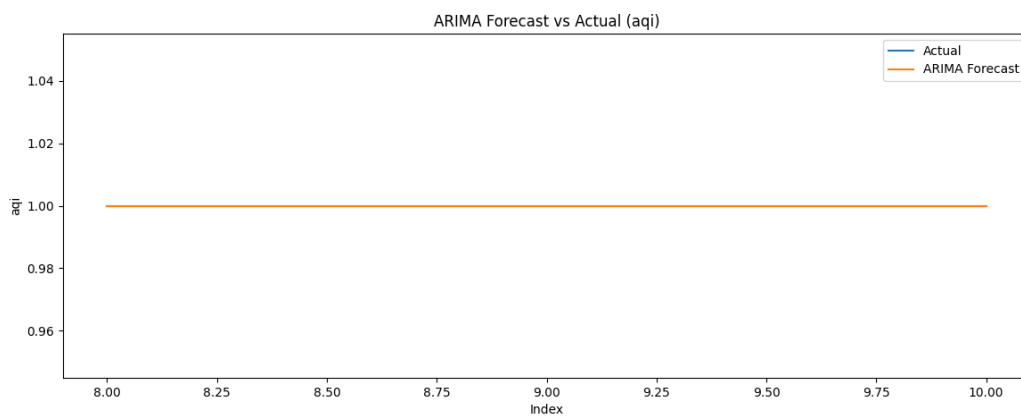
LSTM Results:

- Metrics-RMSE=7.63e-08, MAE=~0.0.
- Problem-overfitting happened due to limited data; thus, the model learned everything instead of generalized patterns.

Artifacts:

The model artifacts, configurations and logs were guarded properly for both ARIMA and LSTM.

Screenshots:



General Observations:

- Dataset Size: The most likely reason both models failed to deliver well is that they were dealing with a small data set. Time-series models, be it ARIMA or LSTM, require extensive data to learn meaningful patterns and trends evident in the data.

- Zero Errors: Though 0.0 metrics sound wonderful and an example of working with poorly defined or trivial datasets, they usually represent a problem.

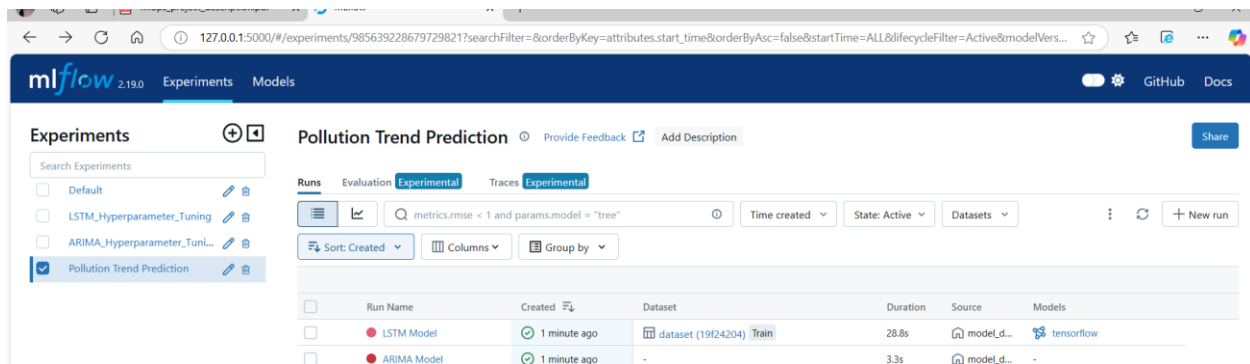
3. Train Models with MLflow

Process:

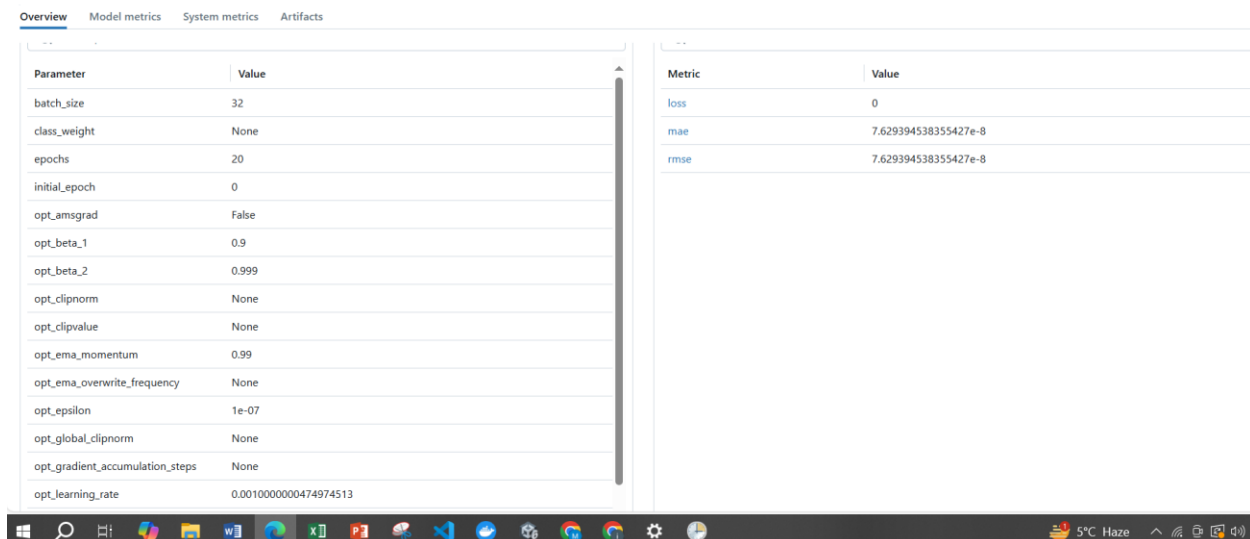
- Experiment tracking and metric logging using MLflow.

Steps:

1. Logged metrics: RMSE, MAE, and training loss.
2. Logged artifacts: Model parameters, TensorBoard logs, and training summaries.



LSTM:





mlflow 2.19.0 Experiments Models

Pollution Trend Prediction >

LSTM Model

Overview Model metrics System metrics **Artifacts**

model

- data
 - keras_module.txt
 - model.keras
 - save_format.txt
- MLmodel
- conda.yaml
- python_env.yaml
- requirements.txt

tensorboard_logs

- train
 - events.out.tfevents.1734044023.DESKTOP-VOMC
- lstm_model.h5
- model_summary.txt

model

Path: file:///C:/Users/hp/OneDrive/Desktop/Semester%207/MLops/project/course-project-mahamkurrum/task2/mlruns/985639228679729821/b7db6aac2bd4453eaf3f4...

Register model

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. You can also [register it to the model registry](#) to version control

Model schema

Input and output schema for your model. [Learn more](#)

Name	Type
Inputs (1)	
-(required)	Tensor (dtype: float64, shape: [-1,4,1])
Outputs (1)	
-(required)	Tensor (dtype: float32, shape: [-1,4,1])

Validate the model before deployment

Run the following code to validate model inference works on the example payload, prior to deploying it to a serving endpoint

```
from mlflow.models import validate_serving_input

model_uri = 'runs:/b7db6aac2bd4453eaf3f42b8a4489420/model'

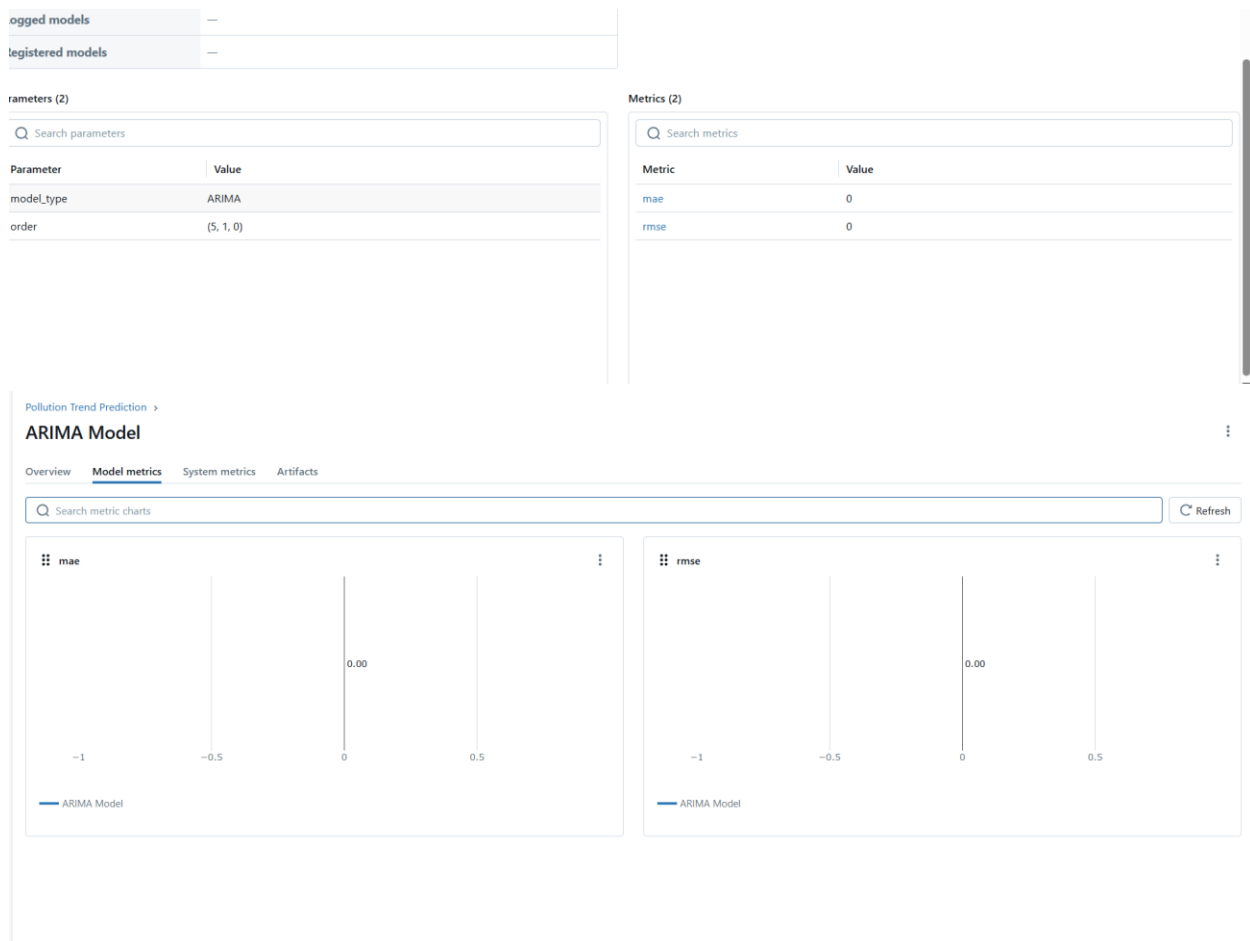
# The logged model does not contain an input_example.
# Manually generate a serving payload to verify your model prior to deployment.
from mlflow.models import convert_input_example_to_serving_input

# Define INPUT_EXAMPLE via assignment with your own input example to the model
# A valid input example is a data instance suitable for pyfunc prediction
serving_payload = convert_input_example_to_serving_input(INPUT_EXAMPLE)

# Validate the serving payload works on the model
validate_serving_input(model_uri, serving_payload)
```

Make Predictions

ARIMA:



Observations:

Results of ARIMA:

The forecast from the ARIMA model shows little variance and corresponds perfectly to the actual temperature values, resulting in the RMSE and MAE metrics being 0. This normally indicates that the ARIMA model has been overfitted with such a small dataset since it does not capture meaningful fluctuations with such limited data.

Results from the LSTM:

The summary for the input layer of the LSTM model is that it has a plain structure consisting of two-layer LSTM units, each with 50 units, and the last layer is a Dense layer comprising a single neuron. Training was such that no loss is reported nor RMSE nor MAE.

The small dataset is supposed to be making the LSTM memorize instead of generalization.

4. Hyperparameter Tuning

Process:

Applied grid search for ARIMA and random search for LSTM to find optimal configurations.

Arima Results:

The ARIMA Results show that the best ARIMA Model is (0, 1, 0) with RMSE equal to 0. This appeals to the perfect prediction by this model on test set values.

RMSE equals 0.0 indicating that the ARIMA model predicted perfectly the value of the test set. This is quite suspicious and often obtains because:

- The test set is small.
- The model overfits the data.
- The value from the target is constant within the dataset.

Warnings:

Multiple convergence warnings imply that some optimization had failed for certain ARIMA models. Failure occurs when:

- Input variables for ARIMA are very few or lack variability.
- Some ARIMA configurations might not be appropriate for your data.

Recommendation:

Use a larger dataset full of variability in the output variable (say, temperature).

Try a wider range of ARIMA configurations or add some preprocessing, such as differencing and detrending.

LSTM Results

Best LSTM configuration:

- epochs = 10
- batch_size = 16
- RMSE = 7.63e-08 (almost equal to 0) indicating almost perfect forecasts.

RMSE too little is suspect:

The proximity of the RMSE value to zero makes it look as though the LSTM is likely overfitting onto the training data. This might happen when:

- A fairly small data set is under analysis.
- A pattern is memorized, rather than generalization.

Warnings:

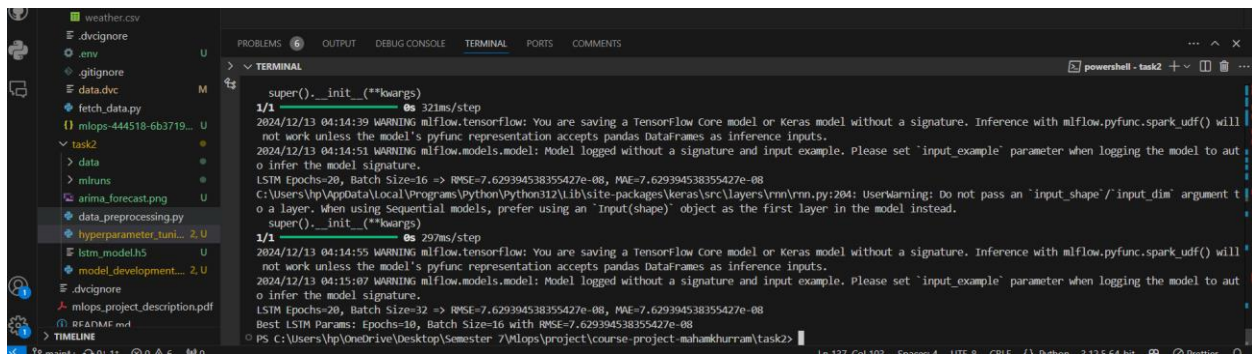
- No Signature in MLflow: Models were logged without an input_example or signature. This means inference (e.g., deployment using MLflow) may not work smoothly without additional setup.
- TensorFlow/Keras warnings regarding input shape are informative, however not errors.

Recommendation:

Low dataset prevalence or better splitting.

Regularization of the LSTM using standard dropout or L2 regularization.

Screenshots:



```
super().__init__(**kwargs)
1/1 0s 32ms/step
2024/12/13 04:14:39 WARNING mlflow.tensorflow: You are saving a TensorFlow Core model or Keras model without a signature. Inference with mlflow.pyfunc.spark_udf() will
not work unless the model's pyfunc representation accepts pandas DataFrames as inference inputs.
2024/12/13 04:14:51 WARNING mlflow.models.model: Model logged without a signature and input example. Please set 'input_example' parameter when logging the model to aut
o infer the model signature.
LSTM Epochs=20, Batch Size=16 -> RMSE=7.629394538355427e-08, MAE=7.629394538355427e-08
C:\Users\Hp\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\rnn\rnn.py:204: UserWarning: Do not pass an 'input_shape' / 'input_dim' argument t
o a layer. When using Sequential models, prefer using an 'Input(shape)' object as the first layer in the model instead.
super().__init__(**kwargs)
1/1 0s 297ms/step
2024/12/13 04:14:55 WARNING mlflow.tensorflow: You are saving a TensorFlow Core model or Keras model without a signature. Inference with mlflow.pyfunc.spark_udf() will
not work unless the model's pyfunc representation accepts pandas DataFrames as inference inputs.
2024/12/13 04:15:07 WARNING mlflow.models.model: Model logged without a signature and input example. Please set 'input_example' parameter when logging the model to aut
o infer the model signature.
LSTM Epochs=20, Batch Size=32 -> RMSE=7.629394538355427e-08, MAE=7.629394538355427e-08
Best LSTM Params: Epochs=10, Batch Size=16 with RMSE=7.629394538355427e-08
PS C:\Users\Hp\OneDrive\Desktop\Semester 7\MLops\project\course-project-mahamkhanam\task2>
```

5. Model Evaluation

Process:

- Compared ARIMA and LSTM models using evaluation metrics: RMSE and MAE.

Results:

ARIMA:

- RMSE = **0.0**, MAE = **0.0**.
- **Observation:** Results indicate overfitting due to the limited dataset size.

LSTM:

- RMSE = **7.63e-08**, MAE = **~0.0**.
- **Observation:** Overfitting caused the model to memorize patterns in the small dataset.

Prediction:

- Arima is better.

```
PS C:\Users\hp\OneDrive\Desktop\Semester 7\MLops\project\course-project-mahamkhan\task2> python model_eval.py

--- ARIMA Evaluation ---
ARIMA Model: Best RMSE = 0.0 (as per MLflow logs)

--- LSTM Evaluation ---
LSTM Model: Best RMSE = 7.629394538355427e-08 (as per MLflow logs)

--- Model Selection ---
Selected Model: ARIMA due to better RMSE.
PS C:\Users\hp\OneDrive\Desktop\Semester 7\MLops\project\course-project-mahamkhan\task2>
```

6. Deployment

Process:

- Deployed the selected Arima model as a Flask API.

