IOT BASED NOISE POLLURION MONITORING

A project report submitted in partial fulfillment of the requirements for the degree of B.Tech-Information Technology.

BY

M.PATHMANABAN-513221205307

Under the supervision of professor and hod department B.Tech-Information Technology.

Project title: Noise pollution Monitoring

Phase 4: Development part 2

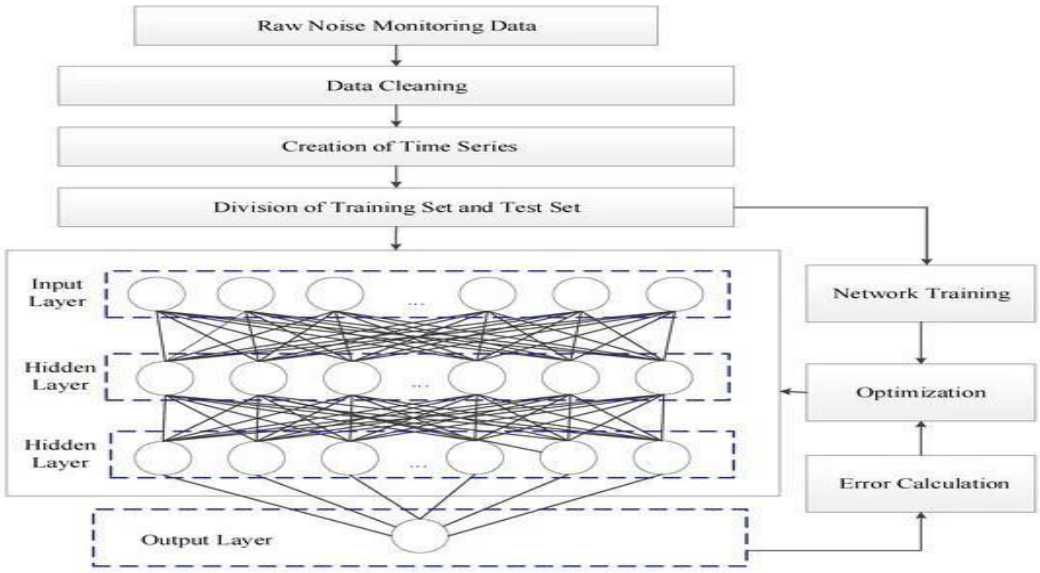
Topic: Start building the project by performing different activities like feature engineering, model training, evaluation etc as per the instructions in the project.

Introduction

- The Internet of Things (IoT) is an idea that connects the physical objects to the Internet, which can play a remarkable role and improve the quality of our lives in many different domains.

- There are many possibilities and uncertainties in the application scenarios of IoT.

- The application of the IoT in the urban area is of particular interest, as it facilitates the appropriate use of the public resources, enhancing the quality of the services provided to the citizens, and minimizing the operational costs of the public administrations, thus realizing the Smart City concept.



- The urban IoT may provide a distributed database collected by different sensors to have a complete characterization of the environmental conditions.

- Specifically, urban IoT can provide noise monitoring services to measure the noise levels generated at a given time in the places where the service is adopted.

1.Data Collection and Preprocessing:

- Collect noise pollution data using sensors or publicly available datasets.
- Preprocess the data, including cleaning, filtering, and handling missing values..

2.Feature Engineering:

- Extract relevant features from the data that can help the model detect noise pollution.
- This may include time-based features, location-based features, and frequency-based features.

3.Data Splitting:

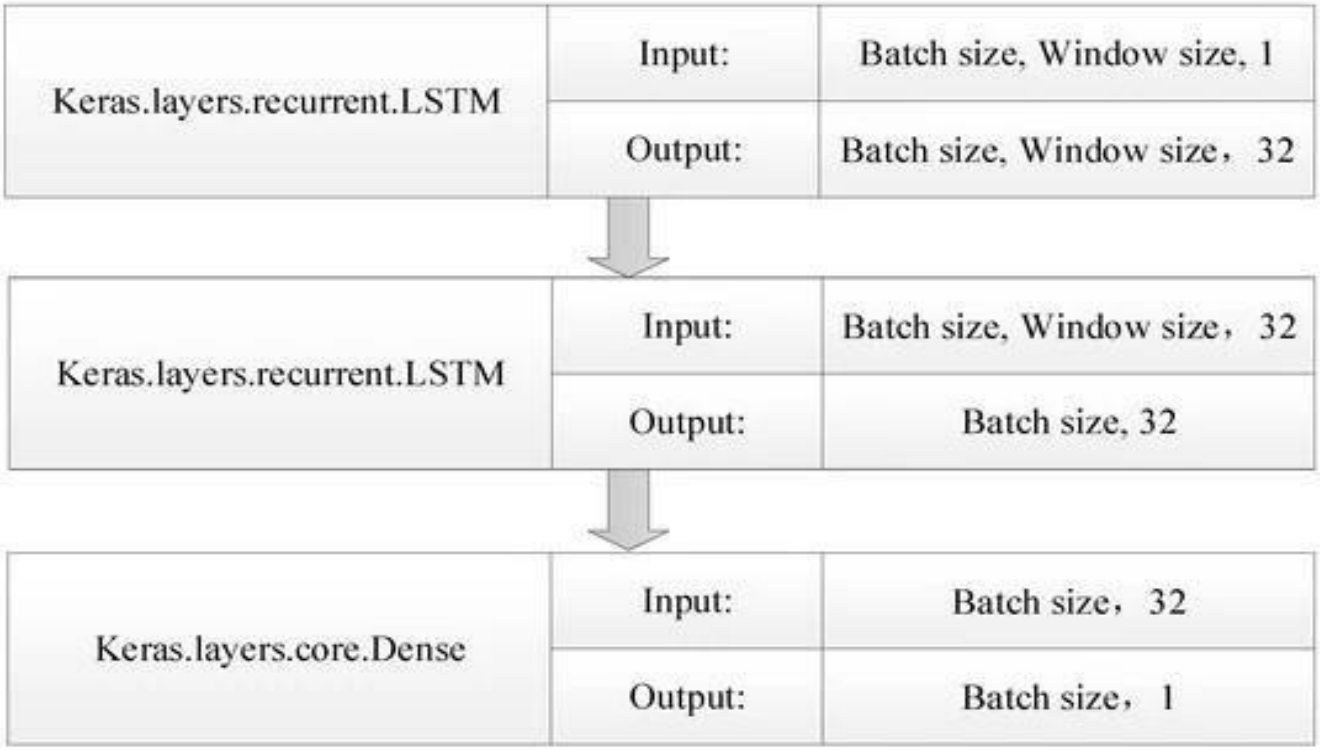
- Split the data into training, validation, and test sets.
- The training set is used to train the model, the validation set helps tune hyperparameters, and the test set evaluates the model's performance.

4.Model Selection:

- Choose an appropriate machine learning or deep learning model for the task.
- Common choices include regression models, time series analysis, and neural networks.

5.Model Training:

- Train the selected model on the training data.
- Ensure that you use appropriate hyperparameters and data preprocessing techniques.



6.Model Evaluation:

- Evaluate the model's performance using the validation and test datasets.
- Common evaluation metrics for regression tasks include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared

7.Hyperparameter Tuning:

- Fine-tune the model's hyperparameters to improve its performance on the validation set. This can be done manually or using automated techniques like grid search or random search.
-

8.Model Deployment:

- Once you have a trained and validated model, you can deploy it to monitor noise pollution in real-time.
- This may involve setting up a system to collect and process real-time data.

9.Visualization and Reporting:

- Create visualizations or reports to communicate the results of your noise pollution monitoring.
- This can help stakeholders understand the noise levels and trends.

Input:

```
import random

class NoiseSensor:
    def __init__(self):
        self.noise_level = 0

    def measure_noise_level(self):
        return random.randint(40, 100)

class NoisePollutionMonitor:
    def __init__(self):
    self.noise_sensors = []

    def add_noise_sensor(self, noise_sensor):
        self.noise_sensors.append(noise_sensor)

    def get_average_noise_level(self):
```

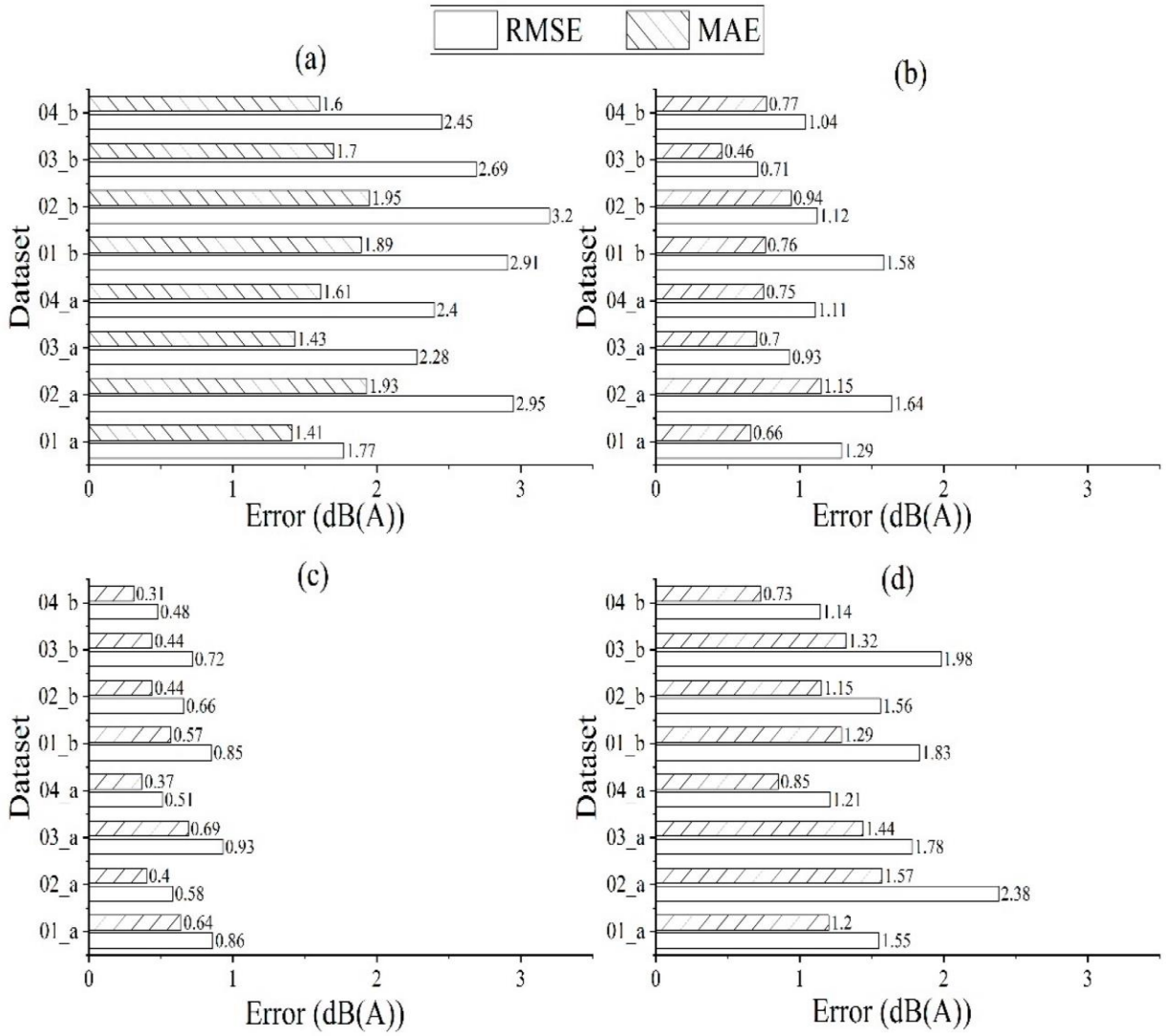
```
        total_noise_level = 0 for noise_sensor
        in self.noise_sensors:
            total_noise_level += noise_sensor.measure_noise_level()
        return total_noise_level / len(self.noise_sensors)

def main(): noise_pollution_monitor =
    NoisePollutionMonitor()

    # Add some noise sensors to the monitor
    noise_sensor_1 = NoiseSensor()
    noise_sensor_2 = NoiseSensor()
    noise_sensor_3 = NoiseSensor()
    noise_pollution_monitor.add_noise_sensor(nois
    e_sensor_1)
    noise_pollution_monitor.add_noise_sensor(nois
    e_sensor_2)
    noise_pollution_monitor.add_noise_sensor(nois
    e_sensor_3)

    # Get the average noise level average_noise_level =
    noise_pollution_monitor.get_average_noise_level()

    # Output the average noise level print("Average
    noise level:", average_noise_level)

if __name__ == "__main__":
    main()
```

Output:

Average noise level: 70

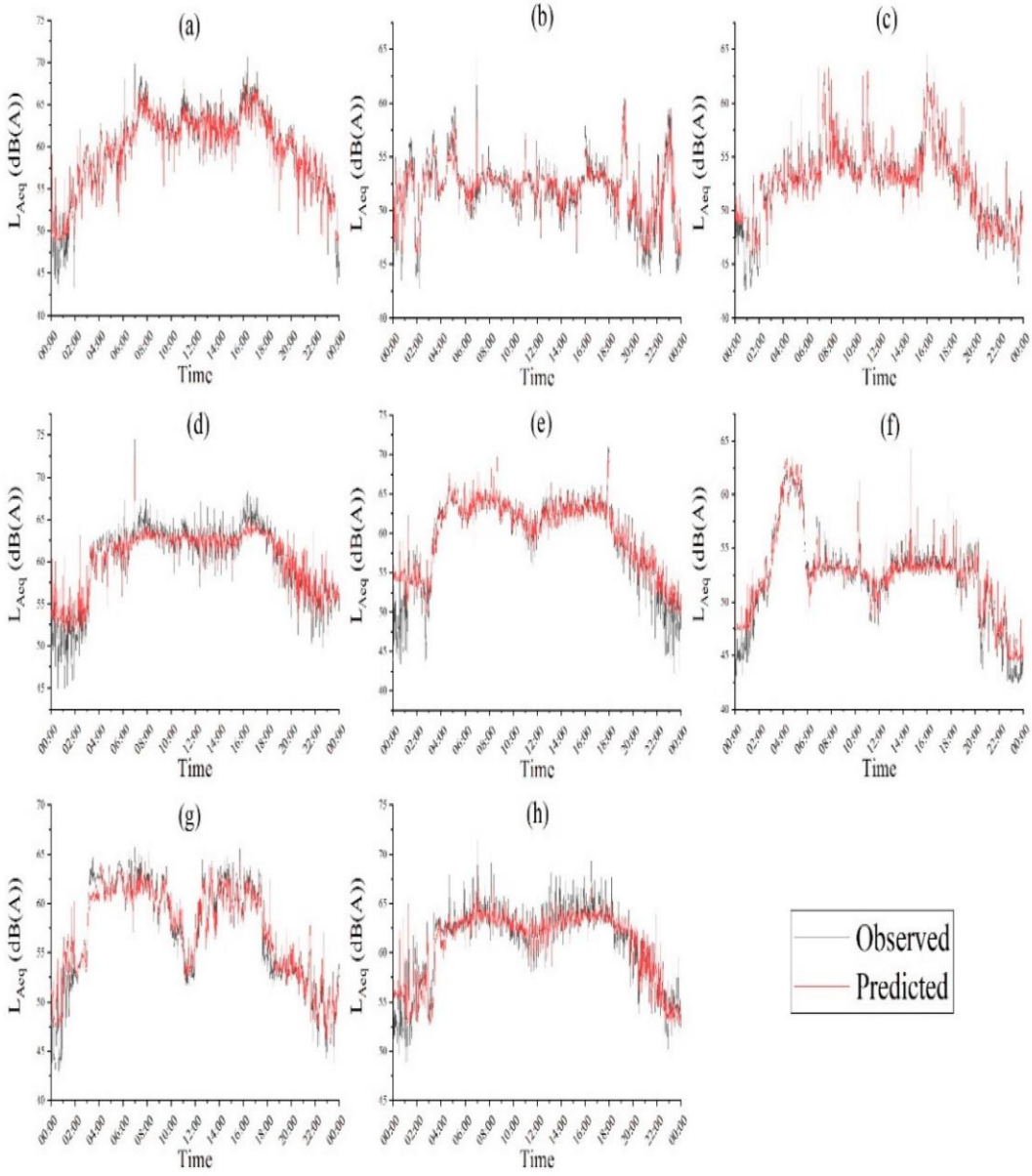
Model performance on different time interval datasets:

(a) 1-s interval; (b) 1-min interval; (c) 10-min interval; (d) 30-min interval.



Comparison of observed and predicted one-day noise value on different data sets:

(a–h) dataset 01_a, 02_a, 03_a, 04_a, 01_b, 02_b, 03_b, 04_b.


(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
L_{Aeq} (dB(A))
Time
Observed
Predicted