IOT BASED ENVIRONMENTAL MONITORING SYSTEM

A Project report submitted in partial fullfilment of the requirements for the degree of B.TECH in InformationTechnology Engineering

By

U.Dhanushkumar(513221104304)

Under the supervision of professor & HOD department of Information technology.

PHASE 4: Development of environmentalmonitoring system:

Data Collection:

- Gather relevant environmental data,
- which might include temperature,
- humidity, air quality, and other factors.
- Ensure data quality, clean the data, and handle missing values if necessary.

Feature Engineering:

- Extract meaningful features from the collected data.
- This may involve time-series analysis.
- statistical calculations, or domain-specific knowledge.
- Normalize or scale features to ensure they have a consistent range.

Model Selection:

- Choose an appropriate machine learning model for your specific task.
- For environmental monitoring.
- time-series models like LSTM or traditional regression models may be suitable.

Data Splitting:

- Split your dataset into training, validation, and test sets. ● This helps you assess your model's performance.

Model Training:

- Train your selected model on the training data using Python libraries like scikit-learn or TensorFlow/Keras for deep learning.

- Tune hyperparameters to optimize model performance, e.g., learning rates, batch sizes, or network architectures.

Model Evaluation:

- Evaluate the model on the validation set using appropriate metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE). ● Visualize the results to gain insights.

Fine-Tuning and Validation:

- Make necessary adjustments to the model based on the validation results.
- Repeat the training and evaluation steps if needed.

Testing:

- Assess the model's performance on the test set to ensure it generalizes well to unseen data. Deployment:

- If the model meets your requirements, deploy it in a production environment for real-time monitoring.
- You can use libraries like Flask or Django to create a web application.

Monitoring:

- Continuously monitor your deployed model to ensure it remains accurate and up-to-date. ● Reporting and Visualization:

- Create reports and dashboards to communicate results to stakeholders.
- Python libraries like Matplotlib, Seaborn, or Plotly can help with data visualization.

Documentation:

- Maintain documentation for your project, including code comments, README files, and model documentation.

For example:

1. Collect Data:

- You'll need to interface with environmental sensors to gather data.
- For this example, let's assume you have a temperature and humidity sensorconnected to your Raspberry Pi.

Input:

```
pip ins import time import board import adafruit_dht

dht_sensor = adafruit_dht.DHT22(board.D4) # GPIO pin

where the sensor is connected while True:

try:

 temperature_c = dht_sensor.temperature humidity =

dht_sensor.humidity print(f"Temperature:

{temperature_c}°C, Humidity: {humidity}%") except

RuntimeError as e:

 print(f"Error: {e}") time.sleep(60) # Collect data every 60

secondstall adafruit-circuitpythondht
```

Create a Python script to collect sensor data:

Output:

```
Temperature: 25.0°C, Humidity: 50.0%

Temperature: 25.1°C, Humidity: 49.9%

Temperature: 25.2°C, Humidity: 50.2%
```

2. Data Processing and Analysis:

- You can perform data analysis on the collected data to identify
- trends or anomalies. For this example, let's calculate the average

Input:

```
import

time

data = []
while True:
try:
temperature_c =
dht_sensor.temperature humidity
= dht_sensor.humidity
data.append((temperature_c,
humidity)) time.sleep(60) except
RuntimeError as e:
print(f"Error: {e}")

if len(data) >= 10:
avg_temp = sum([temp for temp, _ in data]) / len(data)
avg_humidity = sum([hum for _, hum in data]) /
len(data) print(f"Average Temperature: {avg_temp}°C,
Average Humidity:
{avg_humidity}%")
 data = [] # Reset
 data
```

output:

```
Average Temperature: 25.0°C, Average Humidity: 50.0%
```