Task 1: Weather Forecasting Part 2 Report

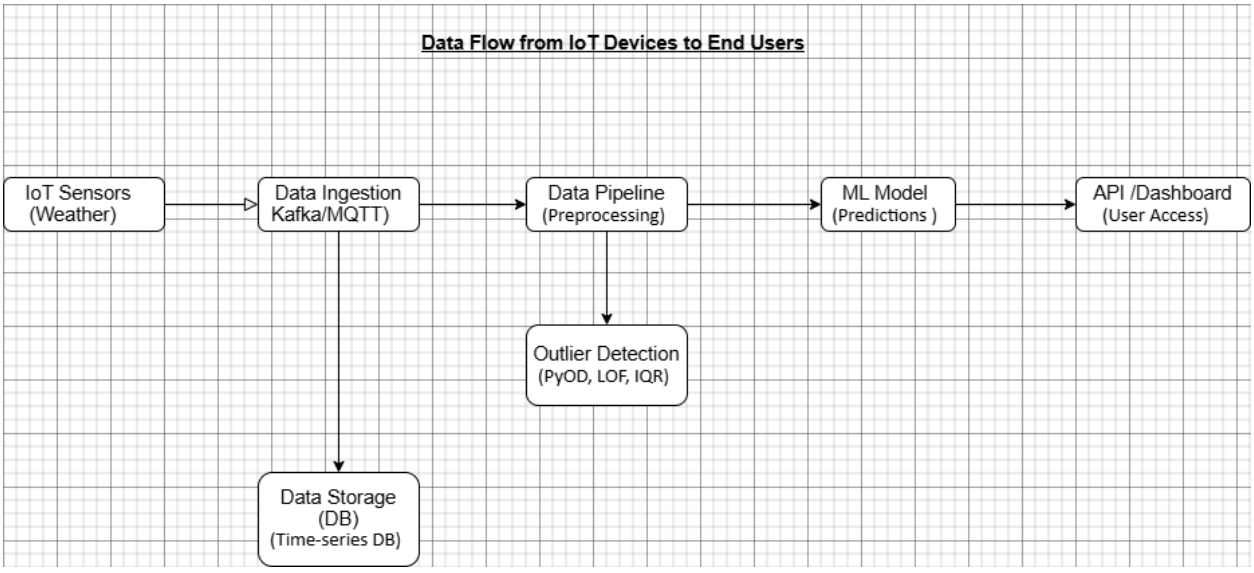
Team Name: Data Dominators

Date: 2025/03/09

System Overview

Farmers need accurate and timely rainfall predictions to plan their activities, but traditional weather forecasts may not be reliable for hyper-local conditions. The system is designed to collect, process, and analyze real-time weather data from IoT sensors to predict rainfall probability for the next 21 days.

Diagram of the System



Component Description

Component	Description
IoT Sensors	Collect real-time weather data (temperature, humidity, wind speed) every minute. Sensors may experience malfunctions, requiring error handling.

Data Ingestion Layer	Receives data from IoT sensors through APIs or message queues (e.g., MQTT, Kafka). Ensures smooth and continuous data flow.
Data Storage	Stores raw and processed weather data using cloud storage (e.g., AWS S3, PostgreSQL, MongoDB) for efficient access and retrieval.
Data Preprocessing	Cleans data by handling missing values, filtering out anomalies, and formatting it for model training. Ensures data quality.
Machine Learning Model	Uses algorithms like Random Forest, XGBoost, or LSTMs to analyze weather trends and predict rainfall probability for the next 21 days.
Model Monitoring & Retraining	Continuously evaluates prediction accuracy. If performance drops, retrains the model using new data to maintain accuracy.
Prediction Output	Generates rainfall probability predictions and makes them accessible through a web dashboard, API, or automated reports.
User Interface (Dashboard/API)	Provides farmers and stakeholders with real-time weather predictions via a user-friendly web dashboard, API, or scheduled reports.

Error Handling

To ensure accurate and reliable predictions, the system implements multiple error-handling mechanisms to manage sensor failures, malfunctions, and missing data.

To ensure accurate predictions, the system continuously monitors sensor data for errors. If a sensor shows strange values (like a temperature below -50°C or humidity over 100%), the system triggers an alert. It also performs regular "heartbeat" checks to ensure sensors are working. If a sensor stops sending data, it's marked as faulty.

When data is missing, the system fills in the gaps using different methods. For example, it can replace missing values with the average or median of previous data. If there's a gap in time, the system can estimate the missing data by looking at nearby values. In some cases, it might use machine learning to predict the missing data based on other sensors. Additionally, if local sensor data is unavailable, the system can fetch data from external weather sources to maintain accurate forecasts.

To prevent faulty data from affecting predictions, the system removes or corrects outliers—values that seem impossible based on past data. It also uses backup sensors to gather data in case one sensor fails. Finally, before feeding the data to the model, the system checks its quality, ensuring that only clean and reliable data is used for predictions.

Conclusion

The real-time weather prediction system provides farmers with **accurate rainfall forecasts for the next 21 days**, helping them make informed decisions about irrigation, planting, and harvesting. By integrating **IoT sensors, real-time data processing, machine learning models, and user-friendly dashboards**, the system ensures reliability and accessibility. Additionally, it incorporates **error-handling mechanisms** to manage sensor failures and missing data, maintaining prediction accuracy even in challenging conditions.