Real Rain Prediction Model

This will be my proposed idea for the final project in software engineering.

Problem and Motivation

Accurate and localized rainfall prediction is crucial for various sectors, including agriculture, urban planning, and disaster preparedness, especially in areas with unique geographical characteristics like Nof-Hagalil and Nazareth. Current forecasting models often struggle with the granularity and real-time adaptability required for precise local predictions. This project aims to address this gap by leveraging real-world sensor data from the Israel Meteorological Service (IMS) to develop a robust and adaptive rainfall prediction model. The key challenges this project seeks to solve include:

- **Data Quality and Preprocessing:** Handling raw, noisy, and potentially incomplete sensor data from real-world sources.
- **Model Accuracy and Generalization:** Developing a model that can accurately predict rainfall in specific, localized areas and generalize well to new, unseen data.
- **Real-time Adaptability:** Implementing mechanisms for the model to continuously learn and adapt to changing environmental conditions and data patterns, mitigating issues like data drift and catastrophic forgetting.
- **Actionable Insights:** Providing an accessible and intuitive platform (mobile application) for users to compare model predictions with existing forecasts, facilitating informed decision-time decisions.

The planned high-level overview of the stages for this project:

1. Store a snapshot of the data and create a proof of concept jupyter notebook for data exploration and preprocessing.

2. After approval for the idea fetch and store a snapshot of the last 5 years of data entries, develop data preprocessing pipelines for dealing with missing data, sensor errors, noisy data and so on.

3. Perform model selection with cross-validation, after finding the best performing model perform parameter hyper-tuning to further improve the model's performance (The performance will be evaluated by the RMSLE between the model's predicted value and the IMS forecast value).

4. Build a backend API for the model with FastAPI or Flask, deploy the model on AWS and have it fetch the last 5 years of data entries to preprocess and train on with the pipelines built from stage 2.

5. Deploy the model with online mini-batch learning, its performance will be evaluated with RMSLE between the model's prediction and the IMS forecast each time it trains on a new batch of data, if the RMSLE exceeds a certain threshold the backend will trigger a retraining routine that includes the first 5 year snapshot and the new accumulated data entries to prevent catastrophic forgetting and adapt to data drift (Trigger-Based Retraining).

6. Develop a front-end React Native mobile weather app that interacts directly with the model's backend API and displays the model's predictions next to the IMS forecast predictions, the app will include an interactive map using leaflet with a grid over the Nof-Hagalil and Nazareth area with clickable blocks

such that each block when clicked would show a small pop-up that compares the model's prediction to the IMS forecast.

Documentation and Knowledge Sharing

The entire project process, from initial data exploration and preprocessing to model deployment and the front-end application development, will be thoroughly documented. This documentation aims to detail the challenges encountered and the solutions implemented throughout the project lifecycle, serving as a comprehensive record and a resource for knowledge sharing.