Sparse Data-Based Rain Prediction

The Problem and Motivation:

Accurate and localized rain prediction is crucial for various sectors, including agriculture, urban planning, and disaster preparedness. Many developing countries and hard-to-reach areas worldwide lack the resources to implement radar systems for granular and precise weather forecasts. A basic Doppler radar system can cost between $300,000 and $500,000, with more advanced models costing $1 million or more (excluding maintenance and operational costs). In contrast, a basic commercial weather system costs between $2,000 and $5,000+ for advanced models. Therefore, this project aims to develop and explore a much cheaper and more accessible solution by leveraging artificial intelligence, spatial interpolation, and data from commercial weather systems, addressing a critical need for many countries and data-sparse regions around the world.

Overview of Planned Project Stages:

1. **Data Collection and Storage:** Collect and store a snapshot of the last five years of data from the Israeli Meteorological Service (IMS), using the data repository in 5-minute windows. An API key allowing access to this data has already been obtained, store the data in a local database.
2. **Data Quality Assessment/Exploratory Data Analaysis:** Write a comprehensive data report that describes everything from missing data to sensor errors, etc.
3. **Data Cleaning and Feature Engineering:** Implement data cleaning pipelines to prepare the dataset for feature engineering and model training, afterwards implement feature engineering pipelines (with a strong emphasis on no data leakage) for strong features from meteorological papers or relevant books.
4. **Statistical Testing:** Perform feature importance analysis and statistical tests to understand the correlation and gauge importance of each feature.
5. **Cross Validation:** Cross validate base machine learning models like XGBoost, Random Forsts and RNN LSTM on the dataset without engineered features and with in order to measure the impact of the engineered features on model performance and generalization.
6. **Fine-Tuning Models:** Finetune each model's hyper-parameters to maximize performance, afterwards display the key predictors for each model (most important features) and display the models performance before and after finetuning.

Optional Extensions (subject to project time constraints):

7. **Backend API:** Build a backend API for the model using FastAPI or Flask, and deploy the trained model artifact to an AWS environment. The API will serve predictions from this model.
8. **Online Learning:** Deploy the model with online mini-batch learning. Its performance will be evaluated using RMSLE between the model's forecast and the IMS forecast each time it is trained on a new batch of data. If the RMSLE exceeds a certain threshold, the backend will trigger a retraining routine that includes the initial five-year snapshot and the new accumulated data, to prevent catastrophic forgetting and adapt to data drift (Trigger-Based Retraining).

9. **Frontend Application:** Develop a mobile front-end weather application in React Native that will communicate directly with the model's backend API and display the model's forecasts alongside the IMS forecasts. The application will include an interactive map using Leaflet with a grid over the Galilee and Nazareth landscape area, with clickable blocks, so that clicking on each block will display a small window comparing the model's forecast to the IMS forecast.