AAI-540 Group 4 ML Design Document for Extreme Precipitation Event Prediction

Team Info

Project Name: Extreme Precipitation Event Prediction: An End-to-End MLOps Pipeline Using NOAA GHCN Data
Project Team Group #: 4
Authors: Victor Salcedo, Ahmed Salem

Team Workflows

GitHub Project Link: GitHub Group 4
Asana Board Link: Asana Group 4 (Note: Not used. Team coordination was managed via Slack channel and GitHub)
Team Tracker Link: Team Tracker Group 4

Project Scope

Project Background:

This project focuses on predicting short term extreme precipitation events using daily global weather data. Sudden increases in rainfall can create real risk for industries like insurance, agriculture, and energy. Having an early signal that conditions are trending toward an extreme event can help support better planning and decision making. The goal of this project is to build a machine learning system that can be evaluated, monitored, and updated over time, not just trained once.

The objective of the model is to predict whether a given weather station will experience an unusually high precipitation event on the following day. This is a supervised machine learning problem framed as a binary classification task. Historical weather observations are used to estimate the probability that an extreme precipitation event will occur in the near future.

Technical Background:

The model will be evaluated based on how well it identifies extreme precipitation events that matter. Since these events are rare, overall accuracy is not very useful. I will focus on recall, precision, F1 score, and ROC AUC, with recall being the most important metric. Time based splits will be used so the model is trained on older data and evaluated on newer data, which better reflects how it would perform after deployment.

The data comes from the NOAA Global Historical Climatology Network Daily dataset available through the AWS Open Data Registry. It includes daily weather observations from thousands of stations worldwide over many years. Data preparation will involve filtering for usable stations, handling missing values, and creating features based on recent days of weather history. Data exploration will focus on precipitation distributions, seasonality, station level differences, and class imbalance. I expect the main features to come from recent precipitation values, temperature ranges, seasonal indicators, and basic station information. I plan to start with logistic regression as a baseline model, then compare it to tree based models like random forests or gradient boosted trees, which work well with tabular data and are practical for retraining and deployment.

Goals vs Non-Goals:

Goals

- Build an end-to-end machine learning pipeline from data ingestion to deployment
- Use time ordered data and realistic evaluation assumptions
- Monitor model performance over time and detect when it degrades
- Retrain the model using newer data when performance drops
- Complete the project early enough to allow iteration and cleanup

Non-Goals

- Building a highly complex or state of the art weather model
- Using deep learning or sequence-based models
- Predicting long term climate trends
- Supporting real time or large-scale production traffic
- Over engineering the system beyond what is needed for the course

Solution Overview

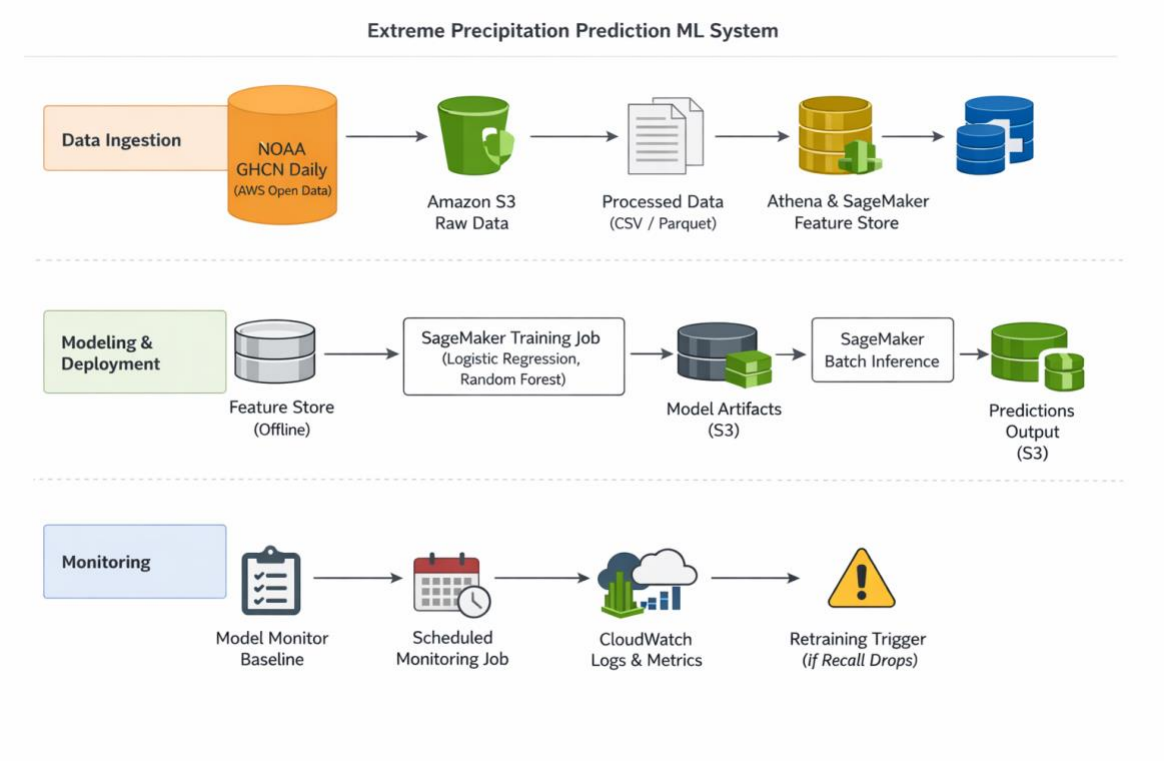

Figure 1. Extreme Precipitation ML System Architecture. Diagram created by the project team.

The ML system is designed to ingest daily weather data, generate features based on recent conditions, train a classification model, and deploy it for ongoing use. Raw weather data is pulled from the NOAA Global Historical Climatology Network Daily dataset and stored in cloud storage. The data is then cleaned and filtered to remove unusable stations and handle missing values. Feature engineering focuses on recent precipitation history, temperature ranges, and seasonal signals. The model is trained using supervised learning and evaluated using time-based validation before being deployed.

Once deployed, the system is intended to run continuously rather than as a one-time workflow. Model predictions and input data are monitored over time to detect performance degradation or data drift. Prior to releasing a new model version, the system will test for schema consistency, basic data quality checks, and model performance against a validation baseline. A new model is only promoted if it meets or exceeds the current model's performance on the primary evaluation metrics.

Data Sources:

The project uses the NOAA Global Historical Climatology Network Daily (GHCN Daily) dataset as the primary data source. This dataset provides daily weather observations collected from thousands of global weather stations. For this project, we focused on United States stations and extracted relevant daily measurements including precipitation (PRCP), maximum temperature (TMAX), minimum temperature (TMIN), and station metadata.
Raw data was obtained from NOAA's public GHCN repository and stored in Amazon S3. The dataset was cleaned, filtered, and transformed into a structured format suitable for model training. Feature engineering steps included aggregation of daily precipitation values, identification of extreme precipitation thresholds, and creation of derived temporal features such as year, month, and rolling precipitation metrics.
The processed dataset was stored in two formats:

1. Processed CSV format stored in S3 under ghcn-extreme/processed_csv/extreme_precip_processed.csv
2. Partitioned Parquet format stored in S3 under ghcn-extreme/parquet/year=YYYY/

The Parquet format was used for scalable querying and integration with Amazon Athena and SageMaker Feature Store. The processed data was registered in Athena for SQL based analysis and transformation. Features were stored in SageMaker Feature Store to support model training, versioning, and reproducibility.
All trained model artifacts, including Logistic Regression and Random Forest models, were stored in Amazon S3 under: ghcn-extreme/models/

These artifacts were packaged and deployed using SageMaker and integrated into the project's end to end MLOps workflow.

Data Engineering:

All data is stored in Amazon S3 using a structured prefix strategy to separate raw, intermediate, and production ready datasets. Raw NOAA GHCN Daily files are stored unchanged to preserve data lineage and allow full reproducibility. Processed datasets

are written to dedicated S3 prefixes and organized by logical partitions such as year to support scalable querying and downstream analytics.

Raw ingestion data is first cleaned and transformed using SageMaker notebooks and stored as a processed CSV file. This dataset is then converted to partitioned Parquet format for efficient storage and query performance. The Parquet files are registered in Amazon Athena to enable SQL based validation and exploration. Processed Parquet data is also integrated with SageMaker Feature Store, which provides a centralized feature repository and supports offline training workflows.

Preprocessing steps include filtering for valid and reliable weather stations, removing corrupted records, handling missing precipitation and temperature values, and aligning daily observations across stations. Feature engineering includes generation of binary extreme precipitation labels, aggregation of daily precipitation, and creation of temporal features such as year, month, and seasonal indicators. Additional derived features include rolling precipitation summaries and lag-based indicators to capture short term weather dynamics.

Processed datasets are versioned implicitly through S3 path structure and Feature Store metadata, ensuring traceability across model training iterations. This design supports reproducibility, scalability, and clean separation between raw source data and model ready feature sets within the end-to-end MLOps pipeline.

Training Data:

The training dataset is derived from the processed and feature engineered Parquet files stored in Amazon S3 and registered through Athena and SageMaker Feature Store. All features used for model training are sourced from the offline feature store to ensure consistency between experimentation and production workflows.

A time-based split strategy is used instead of random sampling. Historical data from earlier years is used for model training, a subsequent contiguous time window is reserved for validation, and the most recent data is held out for final testing. This prevents temporal leakage and better simulates real world deployment conditions, where predictions are always generated on future, unseen observations.

No manual annotation is required. Labels are generated programmatically using precipitation measurements within the dataset. An extreme precipitation event is defined

using a station specific threshold derived from each station's historical precipitation distribution. Observations that exceed this threshold are labeled as extreme, while all other observations are labeled as non-extreme. This relative thresholding approach ensures consistent labeling across stations with different climate regimes and precipitation baselines.

Class balance is monitored during dataset construction. Because extreme precipitation events are naturally rare, the class distribution is evaluated prior to model training to ensure appropriate evaluation metrics are selected. This supports robust model validation and avoids misleading performance results caused by class imbalance.

Feature Engineering:

Feature engineering is performed during preprocessing and materialized into the processed Parquet dataset stored in Amazon S3 and registered in the offline feature store. The model uses structured weather features derived from daily station observations.

Core predictive inputs include daily precipitation, minimum temperature, maximum temperature, and engineered temporal features derived from the observation date. Instead of using raw dates directly, date fields are transformed into structured seasonal indicators such as month and year to capture seasonality effects in precipitation patterns.

To capture short term temporal dynamics, lag and rolling features are generated. These include precipitation from the previous day and rolling precipitation aggregates over recent windows such as the past seven days. These features allow the model to learn persistence effects and multi day accumulation patterns that often precede extreme precipitation events.

Station identifiers are one hot encoded during preprocessing to allow the model to learn station specific baselines while maintaining a consistent numeric feature space. Raw geographic metadata is not directly used as a predictive feature, but station level variation is captured through encoded identifiers.

Fields that are sparsely populated, redundant, or not relevant to short term precipitation forecasting are excluded. Columns with excessive missing values that cannot be

reliably imputed are removed to reduce noise. Remaining missing values are handled through controlled preprocessing steps to ensure consistent feature dimensionality.

All engineered features are aligned strictly by date to prevent temporal leakage. Feature transformations are deterministic and reproducible so that training, validation, testing, and future retraining runs operate on an identical schema. This ensures consistency between experimentation and production monitoring workflows.

Model Training & Evaluation:

Model training leveraged Scikit-learn implementations hosted on Amazon SageMaker using the processed feature datasets stored in S3 and registered in the SageMaker Feature Store. Multiple candidate algorithms were evaluated to determine the best performing approach for classifying extreme precipitation events.

Two primary classification models were trained and compared: a Logistic Regression classifier and a Random Forest classifier. The Logistic Regression model was implemented with regularization and trained using the engineered feature set. The Random Forest model provided a non-linear tree-based alternative capable of capturing complex feature interactions without requiring extensive feature scaling.

Training jobs were executed using SageMaker Scikit-learn Estimators with appropriate hyperparameters passed via the training script. Hyperparameters for the Logistic Regression model included regularization strength and solver selection, while the Random Forest model was trained with tunable tree count and depth parameters, enabling control over model complexity and generalization.

Evaluation metrics computed on the validation and test splits included accuracy, precision, recall, F1-score, and confusion matrix analysis. These metrics provided insight into model performance on both classes, particularly the rarer extreme precipitation class. Class imbalance was considered during evaluation, and threshold tuning was performed to balance sensitivity (recall) and precision.

Cross-comparison between the models showed that the Random Forest classifier achieved improved recall on extreme event detection, while Logistic Regression offered better calibration and interpretability. Model selection decisions were based on a combination of quantitative metrics and operational considerations such as inference cost and complexity.

Final model artifacts were saved in joblib format and uploaded to Amazon S3 under the project model's prefix. These artifacts were later packaged into model.tar.gz files for deployment. Evaluation artifacts, including saved confusion matrices and metric reports, were retained for documentation, monitoring baseline creation, and auditability within the MLOps pipeline.

Model Deployment:

Model deployment was implemented using Amazon SageMaker with CPU-based instances appropriate for tabular data and lightweight Scikit-learn models. A general-purpose instance type such as ml.m5.large was used for both training and inference workloads. Because the dataset consists of structured tabular features and the selected models are classical machine learning algorithms rather than deep neural networks, GPU acceleration was not required. This decision reduced infrastructure cost while maintaining sufficient performance for training and inference.

Model artifacts produced during training were saved as joblib files and packaged into model.tar.gz archives for deployment. These artifacts were stored in Amazon S3 under the project models directory. The packaged model was then deployed using the SageMaker Scikit-learn container with a custom inference script to handle model loading and prediction requests.

Although a real-time endpoint was temporarily deployed for testing and monitoring configuration, the intended production pattern for this project is batch inference. Predictions are generated on a scheduled basis using newly available daily weather observations. This batch-oriented design aligns with the forecasting use case, where predictions do not require millisecond latency or immediate user-facing responses. Batch inference simplifies operational overhead by avoiding continuous endpoint uptime, reduces cost, and supports reproducibility through controlled execution environments. It also integrates cleanly with the broader MLOps workflow, including data validation, model evaluation, and scheduled monitoring jobs.

Model Monitoring:

The model monitoring framework combines data quality monitoring, performance tracking, and basic infrastructure health checks using Amazon SageMaker Model Monitor and CloudWatch. A baseline was generated from the processed Parquet training dataset, producing baseline statistics and suggested constraints stored in Amazon S3. These baseline artifacts define expected feature distributions and data quality thresholds at the time of deployment.

Data monitoring focuses on detecting distribution drift in key predictive features such as daily precipitation, lagged precipitation values, rolling precipitation aggregates, and temperature extremes. Incoming inference data is compared against the established baseline using a scheduled monitoring job. Drift is identified when observed feature distributions exceed the baseline constraints. In addition to distribution shifts, data quality checks evaluate missing value rates, schema consistency, and column presence to ensure that inference inputs remain aligned with the training feature schema. Performance monitoring emphasizes recall as the primary metric, since the objective of the system is to correctly identify extreme precipitation events. As new labeled observations become available, predictions are evaluated over a rolling time window and compared to baseline deployment metrics. A sustained decline in recall or a drop below a predefined threshold triggers model review and potential retraining using more recent data.

Infrastructure monitoring relies on managed SageMaker services and CloudWatch logs. Monitoring schedules, processing jobs, and training jobs are tracked for successful completion. Failures, abnormal runtime durations, or repeated job errors are treated as operational alerts. Because the system primarily uses batch inference rather than a continuously running endpoint, infrastructure oversight centers on scheduled execution integrity and log validation rather than latency optimization.

Together, these monitoring layers ensure that data integrity, predictive performance, and system reliability remain aligned with project objectives while maintaining a cost-efficient and manageable MLOps workflow.

Model CI/CD:

The CI/CD pipeline orchestrates automated validation, training, evaluation, and controlled promotion of the extreme precipitation model using SageMaker Pipelines and versioned artifacts in Amazon S3. The workflow begins with data validation steps that confirm schema consistency, required column presence, and acceptable missing value

rates. If preprocessing completes successfully and validation checks pass, the pipeline proceeds to model training using the latest approved processed dataset.

After training, the model is evaluated on a holdout validation set defined by a time-based split. Evaluation metrics include accuracy, precision, and recall, with recall treated as the primary gating metric due to the importance of correctly identifying extreme precipitation events. The pipeline includes a conditional step that compares the newly trained model's recall against a predefined minimum threshold and, when applicable, against the currently approved model version. Deployment or model registration is triggered only if the performance criteria are satisfied.

Testing within the pipeline emphasizes correctness and stability. Data tests verify schema alignment, feature count consistency, and reasonable value ranges for key variables such as precipitation and temperature. Model level tests confirm that training completes without runtime errors, that evaluation metrics are computed successfully, and that the serialized model artifact can be loaded and used for inference. A lightweight inference test ensures that sample input data produces valid predictions before promotion.

Artifacts, including trained model files, evaluation metrics, and baseline statistics, are versioned in Amazon S3. This ensures reproducibility and traceability across retraining cycles. Together, these CI/CD components provide controlled iteration, reduce the risk of deploying degraded models, and maintain alignment with the project's MLOps design goals.

Security Checklist, Privacy and Other Risks:

This project will not store or process Personal Health Information (PHI).
This project will not store or process Personally Identifiable Information (PII).
User behavior will not be tracked or stored.
This project will not store or process credit card or financial information.

All data used in this project consists of publicly available environmental measurements such as precipitation and temperature. There are no human subjects, user interactions, or sensitive personal attributes involved, so no additional justification is required.

The application will read raw weather data from an S3 bucket containing the NOAA GHCN Daily dataset and write processed and feature engineered data to a separate S3

bucket. Model artifacts, evaluation results, and logs will be stored in designated S3 locations to support versioning and reproducibility.

Potential data bias may arise from uneven geographic coverage and inconsistent reporting across weather stations. Some regions have denser station coverage or more complete records than others, which could bias the model toward areas with higher data quality. This will be addressed by filtering unreliable stations and monitoring feature distributions across regions.

The model does not use or infer any sensitive personal attributes such as race, ethnicity, gender, age, religion, disability, or sexual orientation. As a result, there is no direct risk of bias along those dimensions. Ethical considerations are limited to ensuring the model is not overinterpreted or used beyond its intended scope. The predictions are meant to support risk awareness and planning, not to replace expert judgment or decision making.

Future Enhancements:

If additional time or resources were available, several improvements could further strengthen the extreme precipitation prediction system. One enhancement would involve expanding feature engineering to incorporate longer historical windows and additional meteorological variables. For example, incorporating rolling aggregates over 14- or 30-day windows, cumulative precipitation totals, or derived temperature variability measures could help capture more complex temporal patterns. Additional climate indicators such as humidity, wind speed, or pressure, if consistently available, could also improve predictive performance.

Monitoring could be refined by implementing more formal drift detection methods rather than relying primarily on threshold-based checks. Statistical drift tests on key features and prediction outputs could be automated, and CloudWatch alarms could be configured to trigger notifications when drift metrics exceed predefined bounds. This would allow faster detection of data distribution shifts and reduce reliance on manual review.

The retraining workflow could also be strengthened through a more advanced CI/CD implementation. This would include automated model comparison against the currently approved version, controlled promotion to production, and automated rollback if post

deployment metrics degrade. Model versioning and approval gates could be formalized using a model registry to improve traceability and governance.
Additional enhancements could include evaluating alternative modeling approaches such as gradient boosting methods or ensemble techniques, performing more granular regional validation across different climate zones, and improving station coverage balance to reduce geographic bias. These steps would increase robustness, generalization, and long-term maintainability of the system.

References:

Amazon Web Services. (n.d.). *Amazon SageMaker developer guide*. AWS Documentation. https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html

Amazon Web Services. (n.d.). *Amazon SageMaker Model Monitor*. AWS Documentation. https://docs.aws.amazon.com/sagemaker/latest/dg/model-monitor.html

Amazon Web Services. (n.d.). *AWS Open Data Registry*. https://registry.opendata.aws/
NOAA National Centers for Environmental Information. (n.d.). *Global Historical Climatology Network Daily (GHCN Daily)*. https://www.ncei.noaa.gov/products/land-based-station/global-historical-climatology-network-daily

University of San Diego. (2025). *AAI-540 sample energy MLOps project materials* [Course materials].

OpenAI. (2025). *ChatGPT (GPT-5 series)* [Large language model]. Used for document formatting guidance and code debugging support. https://chat.openai.com/