NORTHEASTERN UNIVERSITY

EECE 7397

Advanced Machine Learning

Project Report

"Benchmarking ML Models for Boston's Weekly Weather"

Instructor: Dr. Jennifer Dy

Group:

Manivannan Senthil Kumar
Mohit Kakda
Muhammad Salman
Nikhil Anil Prakash

Introduction

Bayesian methods have emerged as powerful tools for optimization and prediction due to their ability to incorporate prior knowledge and quantify uncertainty in complex systems; forecasting precipitation—a highly dynamic and discontinuous component of Earth's hydrological cycle—is particularly challenging because small atmospheric changes can abruptly switch between rain and no-rain, and in Boston is further shaped by coastal influences, the North Atlantic Oscillation, seasonal storm tracks and urban heat islands. Leveraging NOAA's multi-sensor precipitation estimates—which merge radar, gauge and satellite data from six local monitoring sites—our collaborative team evaluated three Bayesian approaches (linear and polynomial regression, Gaussian Processes, and Neural Networks with Bayesian Dropout) alongside specialized Decision Trees as a non-Bayesian alternative to determine their comparative effectiveness in modeling volatile precipitation time series and quantifying predictive uncertainty amid inherent seasonality, unresolved sub-grid processes and complex non-linear interactions among temperature, humidity, pressure and wind.

Dataset

Data Source

The dataset used in this project was sourced from the Global Historical Climatology Network - Daily (GHCND), a public archive maintained by the National Centers for Environmental Information (NCEI), NOAA [4]. Raw daily observations were collected from two weather stations in Massachusetts: Boston Logan International Airport and Jamaica Plain. These station-level records include fields such as date, precipitation (PRCP), maximum temperature (TMAX), and minimum temperature (TMIN), along with metadata like station identifiers and geographic coordinates.

The dataset spans a period from January 2005 to December 2024, resulting in approximately 1,040 weekly observations after aggregation. All machine learning models were trained and evaluated using a 70%/10%/20% split for training, validation, and testing, respectively.

Preprocessing

To prepare the raw NOAA data for machine learning, a structured preprocessing pipeline was developed. First, missing temperature values were addressed by estimating the daily average temperature (TAVG) using the formula $\text{TAVG} = (\text{TMAX} + \text{TMIN})/2$ when direct values were unavailable. Data from both weather stations were merged on a daily basis, aligning temperature and precipitation fields side-by-side.

Once merged, the dataset was averaged across both stations to reduce noise and produce a representative daily series for the Boston region. While more sophisticated approaches such as graph-based models or geospatially-aware architectures could retain station-specific information, our focus was not on modeling inter-station relationships. Instead, we aimed to capture regional weather dynamics more robustly by simplifying the problem and creating a unified view of the area's climate behavior.

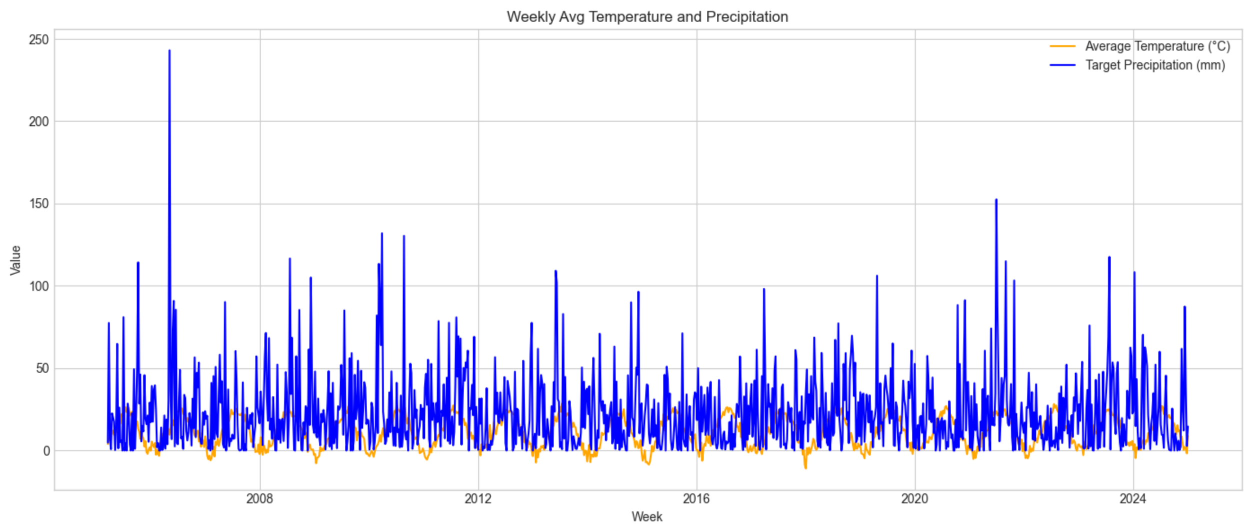


Figure 1: Weekly averaged temperature and precipitation trends across both stations. The data was smoothed by aggregating daily records into weekly summaries to reduce noise and better capture seasonal patterns.

This unified daily dataset was then aggregated to a weekly resolution, spanning Monday to Sunday, to better match the timescale of seasonal weather trends while reducing high-frequency fluctuations. The final weekly data included average temperature and precipitation values per week.

Feature Engineering

To mitigate the week-to-week volatility in precipitation, we augmented each observation at week t with lagged averages from the two preceding weeks—

$$\text{AvgTemp}_{t-2}, \quad \text{AvgTemp}_{t-1}, \quad \text{AvgPrecip}_{t-2}, \quad \text{AvgPrecip}_{t-1},$$

alongside the current week's AvgTemp_t, using Precip_t as the prediction target. We also added a numeric Season Code (0–3) and the calendar week number (1–52) to encode broader seasonal and cyclical effects. This compact feature set of short-term lags plus seasonal indicators helps the model learn structured temporal patterns from inherently noisy weather data.

Machine Learning Models

Bayesian and/or Polynomial Regression

Linear regression models the relationship between input features $\mathbf{x}$ and target variable y using a linear combination of the features. The model is typically represented as:

$$y = \mathbf{w}^\top \mathbf{x} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

where $\mathbf{w}$ is the weight vector and ϵ is Gaussian noise.

Though conceptually simple, linear regression serves as a foundational baseline and is widely used as a benchmark when evaluating more complex and expressive models. Its analytical tractability and interpretability make it a natural starting point for understanding more advanced methods, including the probabilistic techniques discussed later in this paper.

Bayesian linear regression extends this model by placing a prior distribution over the weights $\mathbf{w}$, thereby treating them as random variables.

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \alpha^{-1}\mathbf{I})$$

In our implementation, the dataset originally contained seven input features: *season*, *week*, *Avg_Temp_t-2*, *Avg_Temp_t-1*, *Avg_Temp*, *Avg_Precp_t-2*, and *Avg_Precp_t-1*, with *Target_Precp* as the output. To handle the periodic nature of the *week* feature, it was encoded using sine and cosine transformations. An additional numeric feature *week_num* was also added, resulting in ten input variables in total.

All input variables were treated as components of a single feature vector. A Bayesian regression model was constructed using PyMC3, with a shared Gaussian prior for the weights:

$$\text{intercept} \sim \mathcal{N}(0, 10), \quad \text{coefs} \sim \mathcal{N}(0, 1)$$

$$\sigma \sim \text{HalfNormal}(1)$$

$$y \sim \mathcal{N}(\mu, \sigma), \quad \mu = \text{intercept} + \mathbf{X} \cdot \text{coefs}$$

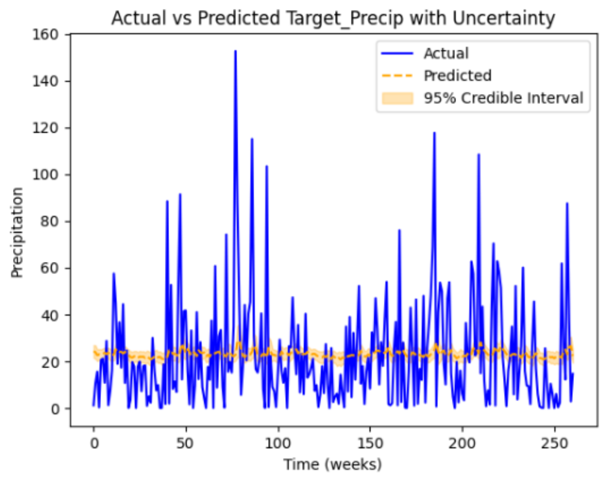


Figure 2: Bayesian Regression with One weight Variable from Gaussian Distribution

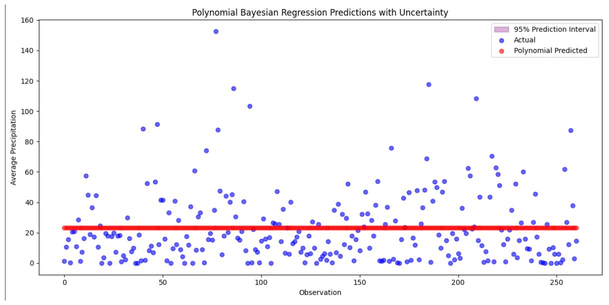


Figure 3: Output for Bayesian Regression with multiple Weight variables, the same output was also observed in Bayesian Polynomial Regression

The model achieved a Root Mean Squared Error (RMSE) of **23.631**.

To further explore the influence of prior distributions on model performance, an alternative Bayesian regression model was implemented where the weights and intercept were sampled from Gamma distributions instead of Gaussians. The aim was to evaluate whether a positively-skewed, strictly positive prior could yield better predictive performance.

The updated model specification was as follows:

$$\text{intercept} \sim \text{Gamma}(\alpha = 1, \beta = 0.02), \quad \text{coefs} \sim \text{Gamma}(\alpha = 2, \beta = 0.5)$$

$$\sigma \sim \text{HalfNormal}(1)$$

$$y \sim \mathcal{N}(\mu, \sigma), \quad \mu = \text{intercept} + \mathbf{X} \cdot \text{coefs}$$

This model yielded a Root Mean Squared Error (RMSE) of **27.52** and a Mean Absolute Error (MAE) of **18.20**.

Building upon the previous setup, the model was further refined by assigning distinct weight variables to each of the ten input features. This adjustment provides the regression model with greater flexibility to learn the varying impact of each feature on the target variable, rather than assuming a uniform influence across all predictors.

The posterior distributions for the weights were inferred using MCMC sampling with PyMC3, and the mean of each posterior was extracted to construct the prediction equation:

$$\mu = \text{intercept} + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_{10} x_{10}$$

The model achieved a Root Mean Squared Error (RMSE) of 23.79.

Although not the final experiment, this configuration demonstrates the value of modeling individual feature contributions and sets the stage for exploring more expressive priors or model structures in subsequent trials.

In the final experiment, a Bayesian polynomial regression model was developed that incorporated domain-informed priors based on observed correlations between the input features and precipitation. The goal was to align prior expectations with real-world relationships, thus improving model interpretability and potentially enhancing predictive accuracy.

Stronger priors were set for highly correlated features—particularly past precipitation values (`AvgPreT_2` and `AvgPereT_1`)—while weaker or negatively signed priors were assigned to features with moderate or weak correlations. This included temperature lags, seasonality, and week-number indicators. Additionally, a `HalfStudentT` prior was employed for the error term to better capture possible heavy-tailed behavior in precipitation patterns.

$$\mu = \text{intercept} + \sum_{i=1}^{9} \beta_i x_i, \quad y \sim \mathcal{N}(\mu, \sigma^2), \quad \sigma \sim \text{HalfStudentT}(3, 5)$$

Despite the tailored priors and improved formulation, the final model yielded a Root Mean Squared Error (RMSE) of **23.79**.

While regression initially showed promise by capturing nonlinear trends in its simpler forms, its performance deteriorated with increased complexity. The models became less reliable and failed to uncover meaningful relationships in the data as priors and formulations grew more elaborate. This suggests that Bayesian regression, both linear and polynomial, may not be well-suited for tasks involving this dataset or similar forecasting problems.

Gaussian Process Regression

Motivation and Background

Gaussian Process Regression (GPR) provides a non-parametric Bayesian approach to modeling complex functions, making it well-suited for weather prediction due to its ability to quantify uncertainty. A Gaussian process is defined as a collection of random variables, any finite subset of which follows a multivariate normal distribution. Formally, we write

$$f(x) \propto \mathcal{GP}(m(x), k(x, x'))$$

where $m(x)$ is the mean function (often set to zero) and $k(x, x')$ is the covariance (kernel) function. Given training inputs $X = \{x_i\}_{i=1}^{n}$ and the outputs y, the joint prior and liklihood yield the marginal log likelihood

$$log(p(y|X, \theta) = -\frac{1}{2}y^T K^{-1} y - \frac{1}{2}log|K| - \frac{n}{2}log 2\pi$$

where K is the nxn covariance matrix with entries $K_{ij} = k(x_i, x_j)$ and θ denotes hyperparameters of the kernel and noise level. For a test point $\mathbf{x}_*$, the predictive distribution is Gaussian with

$$\mathbb{E}\big[f(\mathbf{x}_*)\big] = \mathbf{k}_*^\top K^{-1}\mathbf{y}, \qquad \mathrm{Var}\big[f(\mathbf{x}_*)\big] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top K^{-1}\mathbf{k}_*,$$

where

$$\mathbf{k}_* = \big[k(\mathbf{x}_*, \mathbf{x}_1), \ldots, k(\mathbf{x}_*, \mathbf{x}_n)\big]^\top.$$

Kernel Selection

A critical choice in Gaussian Process Regression is the kernel k, which encodes assumptions about function smoothness and scale:

- **Constant Kernel × Squared Exponential Kernel**
  We use
  $$k(\mathbf{x}, \mathbf{x}') = C \times \exp\Big(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2}\Big),$$
  where C is a constant variance term and ℓ is the length-scale.

- **RBF (Squared Exponential) Kernel**
  The RBF kernel assumes the function is infinitely differentiable and enforces smooth variation. The parameter ℓ controls how quickly correlations decay with distance: small ℓ captures rapid changes, while large ℓ enforces broader, smoother trends. Its form is
  $$k_{\mathrm{RBF}}(r) = \exp\Big(-\frac{r^2}{2\ell^2}\Big), \quad r = \|\mathbf{x} - \mathbf{x}'\|.$$

This kernel is a natural first choice for weather variables, balancing flexibility and smoothness.

Hyperparameter Optimization

Two strategies were employed to tune the noise level α and kernel length-scale ℓ:

1. **Grid Search with Cross-Validation**
   A grid over $\alpha \in \{10^{-2}, 10^{-3}, 10^{-4}\}$ and $\ell \in \{0.1, 1.0, 10.0\}$ was evaluated using 3-fold cross-validation on the training set. The best configuration found was

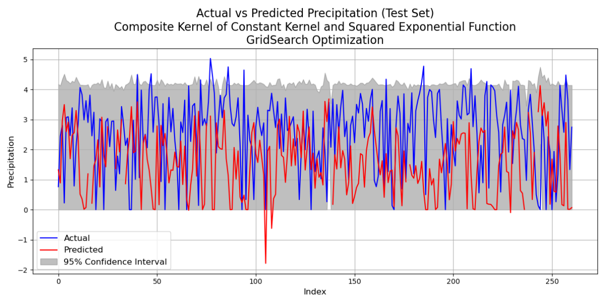
   $$\alpha = 10^{-2}, \quad \ell = 0.1,$$

   yielding a test root-mean-square error (rMSE) of 30.27.

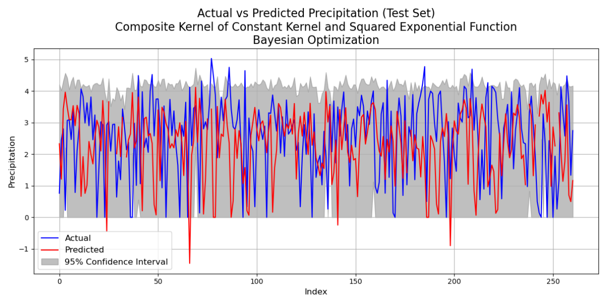
2. **Bayesian Optimization**
   A broader search over $\alpha \in [10^{-4}, 10^{2}]$ and $\ell \in [0.01, 100]$ used `skopt.gp_minimize` to minimize the negative CV MSE. After 25 calls, it suggested

   $$\alpha \approx 100.0, \quad \ell \approx 59.9,$$

   but this led to a higher test MSE (≈ 812), indicating that the wide search space and limited iterations may have trapped the optimizer in a suboptimal region.



(a) Grid Search Optimization



(b) Bayesian Optimization

Figure 4: Comparison of hyperparameter search methods

Analysis of Results

The results are shown in Figure 4

- **Grid Search Performance:** The grid-search model ($\alpha = 0.01$, $\ell = 0.1$) achieved rMSE = 30.27 and $R^2 = -0.62$, demonstrating its ability to track local precipitation swings at the expense of occasionally under-covering extreme spikes and exhibiting a convergence warning (constant kernel value hitting its upper bound).

- **Bayesian Optimization Outcome:** With $\alpha = 100$ and $\ell = 100$, Bayesian tuning yielded a slightly lower rMSE = 28.50 and $R^2 = -0.44$, indicating modest accuracy gains. However, predictions are overly smooth and the 95% confidence band is extremely wide, obscuring genuine temporal structure.

- **Uncertainty Quantification:** The grid-search model produces narrower, locally adaptive intervals that sometimes fail to cover high-variance events, whereas the Bayesian-optimized model's intervals, though broad enough to encompass extremes, lack the resolution to distinguish true volatility from noise—highlighting a trade-off between fidelity and uncertainty coverage.

- **Root Causes of Underperformance:** Both models' negative R^2 indicates they underperform a naïve mean predictor. Key limitations include: (1) a single Constant x RBF kernel cannot adapt to non-stationary, regime-dependent volatility; (2) relying solely on seasonal indicators and two-week lags omits crucial atmospheric drivers; (3) assuming constant Gaussian noise ignores heteroscedastic error during storms; (4) hyperparameter searches pushing $\updownarrow$ to extremes either overfit noise or oversmooth genuine signal; and (5) a small, weekly-aggregated dataset exacerbates sensitivity to outliers and hampers learning of complex covariance structures.

Comparing Dropout and Bayesian Neural Networks

Motivation and Background

Weekly precipitation data is highly volatile and exhibits complex, non-linear patterns that are difficult to model using traditional deterministic approaches. Standard regressors often provide a single point estimate without conveying how confident the model is, which can be misleading—especially in high-stakes decision-making scenarios such as weather forecasting or flood risk management.

In such cases, **probabilistic modeling** becomes essential. Rather than making a single best guess, these models aim to predict a distribution over possible outcomes, quantifying the uncertainty inherent in their predictions. Neural networks, while expressive and powerful, typically lack calibrated uncertainty by default.

A notable exception arises when **dropout**, a common regularization method, is interpreted through a Bayesian lens. Gal and Ghahramani [1] showed that applying dropout at inference time approximates Bayesian inference in deep learning. This results in **Monte Carlo Dropout (MC Dropout)**, a technique that enables uncertainty estimation by averaging multiple stochastic forward passes through the model.

To evaluate different strategies for incorporating uncertainty in neural models, we compared four architectures:

- MLP with MC Dropout (Bayesian approximation),

- RNN with MC Dropout (sequence modeling with uncertainty),

- Vanilla Bayesian MLP (with variational inference over weights),

- Laplace Bayesian MLP (learning both predictive mean and variance).

Architectures and Estimation Techniques

All models were trained on the same weekly-aggregated dataset using identical features. Each aimed to predict weekly precipitation while quantifying associated uncertainty, but differed in how they modeled it.

In both the MLP and RNN using MC Dropout, dropout layers remained active during inference. This allows the model to simulate different subnetworks on each pass, producing a distribution over predictions [1]. After $T = 100$ forward passes, the model computes the mean and variance as:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} y^{(t)}, \quad \mathrm{Var}(\hat{y}) = \frac{1}{T}\sum_{t=1}^{T} \left(y^{(t)} - \hat{y}\right)^2$$

This method captures **epistemic uncertainty**, representing uncertainty in model parameters due to limited data. RNNs are especially suited for time series due to their ability to maintain temporal dependencies [3].

The **Vanilla Bayesian MLP** adopts a more principled Bayesian framework by treating network weights as probability distributions. Using variational inference, the model approximates the posterior over weights and optimizes an evidence lower bound (ELBO), which combines a Huber likelihood loss and KL divergence [2]. This approach allows the model to propagate uncertainty from weights to outputs, unlike deterministic neural nets.

The **Laplace Bayesian MLP** extends this idea by explicitly modeling **aleatoric uncertainty**—irreducible noise inherent in the data itself. This is achieved by adding a second output head to predict the log of the scale parameter, minimizing a negative log-likelihood loss derived from a Laplace distribution [2, 6]. The full objective becomes:

$$\mathcal{L} = \log b + \frac{|y - \mu|}{b} + \lambda_{\mathrm{KL}} \cdot \mathrm{KL}(q(w) \parallel p(w)), \quad b = e^{\log_b}$$

This allows the model to jointly learn both the predictive mean and confidence level, adjusting its uncertainty based on input patterns—especially useful for capturing volatile precipitation trends.

Training and Evaluation

All models used a 70/10/20 split (train/val/test), ReLU activations, and Adam optimizer. Early stopping was based on validation loss. During inference, MC Dropout models sampled $T = 100$ passes. Metrics included:

- MAE and RMSE (point accuracy),
- NLL (log-likelihood of prediction),
- Empirical coverage (at 80% and 95%),
- Accuracy within $\pm$5mm.

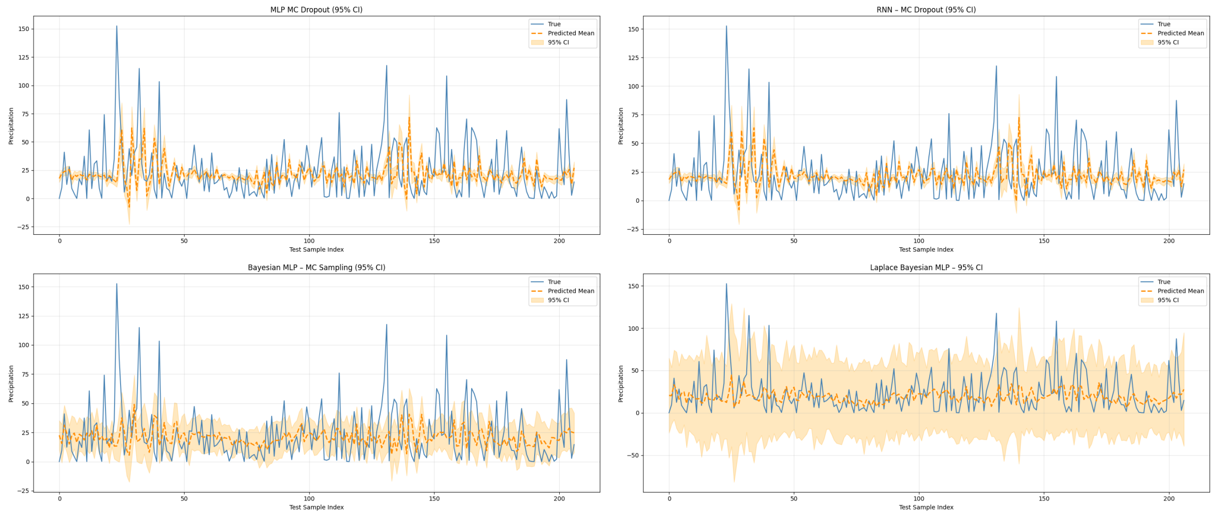


Figure 5: Test set predictions with 95% confidence intervals for each model. Top: MLP and RNN with MC Dropout. Bottom: Vanilla Bayesian MLP and Laplace Bayesian MLP.

Model	**RMSE** ↓	**MAE** ↓	**R^2** ↑	**Accuracy ±5mm** ↑	**NLL** ↓	**80% Cov.** ↑	**95% Cov.** ↑
MLP + MC Dropout	25.40	**17.92**	**-0.07**	15.46%	69.087	15.46%	21.74%
RNN Dropout	26.63	19.41	-0.18	13.53%	76.837	9.18%	13.04%
Bayesian MLP	**25.33**	**17.97**	**-0.07**	19.32%	8.998	33.33%	52.17%
Laplace BNN	26.66	**17.97**	-0.18	**21.74%**	**4.685**	**88.89%**	**92.75%**

Table 1: Evaluation of uncertainty-aware models. Bold = best performance per metric.

Results and Interpretation

Figure 5 illustrates predictions and confidence intervals. The Bayesian MLP yielded narrow intervals and good volatility response; RNNs slightly overestimated uncertainty. The Laplace model, though slightly less accurate, produced the most calibrated intervals.

Quantitatively (Table 1), the Bayesian MLP achieved the lowest RMSE, while the Laplace BNN offered the best NLL and coverage. MLP+MC Dropout had the lowest MAE but underperformed on calibration.

Conclusion: For tasks requiring reliable uncertainty quantification, the Laplace Bayesian MLP emerged as the most practical and robust model.

Decision Trees

A Decision Tree is a supervised machine learning algorithm that can be visualized as an inverted tree structure, starting from a root node and branching out with increasing depth. At each node, the algorithm splits the data into smaller subsets based on feature thresholds that best reduce the prediction error. This hierarchical tree structure makes sure that the model learns the complex, non-linear relationships between input features and the target variable.

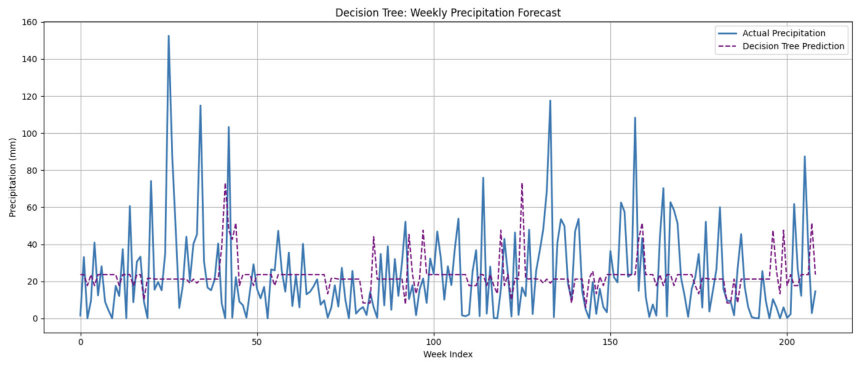


Figure 6: Prediction results of Decision Tree algorithm using standard features of the dataset. MAE: 18.98, RMSE: 26.11, R^2: -0.14

Node Splitting in Decision Tree Regression

At each node of the Decision Tree Regressor, the algorithm attempts to split the dataset into two subsets that minimize the prediction error. For every feature, the algorithm considers all possible threshold values that could be used to split the data. Each potential split divides the data into a left subtree (where the feature value is less than or equal to the threshold) and a right subtree (where the feature value is greater than the threshold). The quality of a split is measured by the weighted sum of the Mean Squared Errors (MSE) of the two subsets. The threshold value that minimizes the total error is selected.

Let R_1 and R_2 be the left and right subsets after a split. Let n_1 and n_2 be the number of samples in each subset, and $\bar{y}_1$, $\bar{y}_2$ be the mean target values in each subset. Then, the total split cost is given by:

$$\text{MSE}_{\text{split}} = \frac{n_1}{n} \sum_{i \in R_1} (y_i - \bar{y}_1)^2 + \frac{n_2}{n} \sum_{i \in R_2} (y_i - \bar{y}_2)^2$$

where $n = n_1 + n_2$ is the total number of samples in the current node. This process is recursively repeated for each child node until a stopping condition is met, such as a maximum depth or a minimum number of samples per leaf node.

For each week t, the dataset includes the average temperature and precipitation from the previous two weeks ($t-1$, $t-2$) as predictors. This yielded six lagged variables: `Avg_Temp_t-2`, `Avg_Temp_t-1`, `Avg_Temp`, `Avg_Precip_t-2`, `Avg_Precip_t-1`, and the target variable `Target_Precip_t`.

To capture short term changes in precipitation, **rolling mean** and **rolling standard deviation** features were newly added to the dataset. The rolling mean feature provides the average precipitation over a defined window of previous observations, offering a smoothed representation of recent rainfall patterns. The rolling standard deviation measures the variability in precipitation within the same window, highlighting periods of unusually stable or volatile weather. By including these rolling statistics as input features, the model gains contextual information about recent conditions and understands to predict the spikes in the precioitation pattern.

The rolling mean and standard deviation for precipitation at time t over a window size

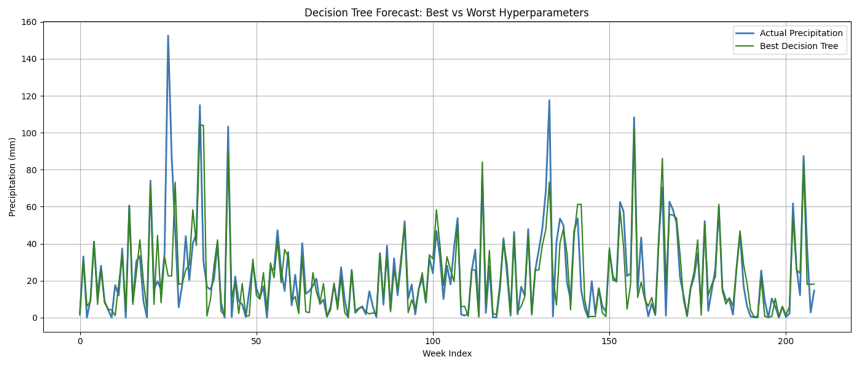


Figure 7: Prediction results of Decision Tree using rolling features and after performing hyperparameter tuning. MAE: 7.45, RMSE: 14.65, R²: 0.41

w are defined as:

$$\text{Rolling Mean}_t = \frac{1}{w} \sum_{i=t-w+1}^{t} P_i$$

$$\text{Rolling Std}_t = \sqrt{\frac{1}{w} \sum_{i=t-w+1}^{t} (P_i - \text{Rolling Mean}_t)^2}$$

where P_i denotes the precipitation at time i.

After adding rolling mean and rolling standard deviation features to the dataset, hyperparameter tuning was performed to optimize the Decision Tree Regressor. Initially the hyperparameters, (`max_depth`) the maximum depth of the tree and and the minimum number of samples required at each leaf node (`min_samples_leaf`) are chosen randomly.

Later, hyperparameter tuning is performed using Grid search algorithm to systematically explore different combinations of key parameters. Using cross- validation of the training data, the optimal set of hyperparameters was identified.

Conclusion

In summary, while probabilistic models such as Bayesian neural networks and Monte Carlo Dropout-based MLPs are capable of making prediction uncertainty and often provide more calibrated confidence intervals, the Decision Tree Regressor gave competitive accuracy in predicting weekly precipitation. The main strength of the decision trees is to capture non-linear transitions in weather data through hierarchical rule-based splits. However, probabilistic approaches offer additional insights by modeling uncertainty, which can be particularly valuable for risk assessment and decision-making in meteorological applications. Thereby, the choice between decision trees and probabilistic models depends on the specific requirements of the task: if model transparency and direct interpretability are priorities, decision trees are advantageous; if robust uncertainty estimation is critical, probabilistic models may be preferred.

Project Contribution

Muhammad Salman(002315661): Primarily responsible for dataset preparation, feature engineering, and implementing models focused on uncertainty estimation. He led the design and evaluation of multiple neural architectures, including MLPs, RNNs, and Bayesian models such as MC Dropout, Variational BNNs, and Laplace-approximated networks. He also developed the evaluation pipeline for these models.

Manivannan Senthil Kumar(002839102): Implemented and optimized Decision Tree models for weekly weather prediction. The dataset was enhanced with lagged variables and rolling statistical features, and key hyperparameters were tuned to improve model performance and prevent overfitting. The trained model performed very well on the spikes in the prediction data and was the best performing model in terms of rMSE .Analyzed the tradeoff between the tree-based models and the probabilistic models.

Nikhil Anil Prakash (002470046): Implemented a Gaussian Process Regression pipeline to forecast weekly precipitation by engineering seasonal indicators and two-week lag features, standardizing inputs, and splitting data around a 2020 pivot. Adopted a composite Constant $\times$ Squared Exponential Kernel kernel and conducted hyperparameter tuning via grid-search cross-validation (achieving $rMSE \approx 30.3$) as well as Bayesian optimization to explore the effects of local versus global smoothing. Analyzed predictive accuracy, uncertainty coverage, and convergence behavior, finding that grid-search tuning captured volatility more effectively while Bayesian optimization produced an overly smooth fit. Identified limitations in modeling extreme events and under-coverage of confidence intervals during sharp spikes.

Mohit Kakda (002087327): Spearheaded the development of Bayesian regression models for weekly precipitation forecasting, focusing on iterative refinement and performance optimization. Initiated with a baseline Bayesian linear regression using shared Gaussian priors, then progressively explored more complex configurations—including Gamma priors, feature-specific weights, and polynomial regression—to better capture the underlying non-linearities and temporal dependencies in the dataset. Introduced domain-informed priors derived from correlation analysis to align model assumptions with real-world relationships, and evaluated each variant using RMSE and MAE to guide model evolution. Conducted posterior inference using MCMC (NUTS) in PyMC3 and closely analyzed the tradeoffs between prior selection, model complexity, and predictive confidence. This systematic approach enabled continuous performance tuning and highlighted the limitations of Bayesian regression in more complex settings.

References

[1] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International conference on machine learning*, pages 1050–1059. PMLR, 2016.

[2] C. Leirbag. A first insight into bayesian neural networks (bnn). https://medium.com/@costaleirbag/a-first-insight-into-bayesian-neural-networks-bnn-c767551e9526. Accessed: 2025-04-19.

[3] MLPills. Theoretical introduction to recurrent neural networks. https://mlpills.dev/time-series/theoretical-introduction-to-recurrent-neural-networks/. Accessed: 2025-04-19.

[4] NOAA National Centers for Environmental Information. Global historical climatology network - daily (ghcnd). https://www.ncei.noaa.gov/data/daily-summaries/doc/GHCND_documentation.pdf. Accessed: 2025-04-21.

[5] Scikit-learn developers. sklearn.gaussian_process.GaussianProcessRegressor. https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.GaussianProcessRegressor.html. Accessed: 2025-04-19.

[6] Your Data Teacher. How many neurons for a neural network? https://www.yourdatateacher.com/2021/05/10/how-many-neurons-for-a-neural-network/, May 2021. Accessed: 2025-04-19.