INTERNATIONAL RESEARCH JOURNAL OF ENGINEERING & APPLIED SCIENCES
ISSN: 2322-0821(O) ISSN: 2394-9910(P)
VOLUME 13 ISSUE 1
Jan 2025 - Mar 2025
www.irjeas.org

Original Article

Towards Better Water Management- Predictive Models for Bihar's Rainfall

Ayushman Pranav[1], Ankit Dubey[2], Umesh[3], Rajesh Kumar Modi[4]

[1-4]*Bennett University, Gr. Noida, India*
[1]theayushmanpranav@gmail.com
[2]ankit13092003@gmail.com
[3]er.umeshgupta@gmail.com
[4]rajeshmodi@gmail.com

Corresponding Author: theayushmanpranav@gmail.com

DOI –10.55083/irjeas.2025.v13i01005

Abstract: Water is essential for life and its continued existence on Earth. It also plays a critical role in maintaining the Earth's climate, providing the foundation for agriculture, human development, and ecosystems. In this research paper, we explore the rainfall patterns across India, with a primary focus on Bihar's rainfall pattern. We have used a dataset containing rainfall patterns from all over India from 1901 to 2015, which is a very comprehensive dataset given its century-long span. We have applied several machine learning models including LSTM, XGBoost, Random Forest Regression, Gradient Boosting, and CatBoost, and compared their performances. The analysis of the research reveals that Bihar experiences lower levels of rainfall compared to the national average. The LSTM model proved to be more accurate for short-term pattern predictions of rainfall in Bihar and outperformed other models in terms of mean absolute error, mean squared error, and root mean squared error. The XGBoost model also provided accurate predictions and showed comparable performance to the LSTM model. Random Forest Regression, Gradient Boosting, and CatBoost models also demonstrated strong predictive capabilities. Overall, this study highlights the value of machine learning models in predicting rainfall patterns and their potential to assist in decision-making related to agriculture, water management, and disaster-related alertness.

Keywords: Agriculture, ARIMA, AUTOML, Bihar, India, Dataset, Disaster preparedness, LSTM, Machine learning, Prediction, Prophet, Rainfall patterns, Water management, XGBoost.

1 INTRODUCTION

Rainfall prediction plays a pivotal role in various sectors including agriculture, water resource management, and urban planning, especially in regions like Bihar, India, [1], [2], [3] where agriculture is a significant part of the economy and society. This paper aims to explore the patterns and predictability of rainfall in Bihar, comparing it with national trends in India, [4] utilizing various machine learning and time series forecasting models. Studies such as [5] and [6] laid the groundwork by applying global optimization methods and artificial neural networks (ANNs) to conceptualize rainfall-runoff models, highlighting the potential of computational models in hydrological predictions. Subsequent research by [7], [8], and [9] expanded these models' capabilities by incorporating statistical methods and hybrid machine learning techniques for more accurate rainfall forecasts. More focused studies, such as those by [10], [11], and [12], applied ANNs specifically for rainfall prediction in localized Indian contexts, providing insights into the applicability of these models in regional settings. Moreover, recent advancements by [13] and [14] integrating sophisticated algorithms like XGBoost and ensemble machine learning models have demonstrated significant improvements in

prediction accuracies using time-series models [15].

Despite considerable advancements, existing literature often lacks:

1. A comprehensive comparison of different forecasting models' performance across diverse Indian regions, with specific emphasis on their applicability in Bihar.
2. An in-depth analysis of models' capability to predict extreme weather events which are crucial for disaster management and agricultural planning in Bihar.
3. Exploration of hybrid models that combine classical time series approaches with cutting-edge machine learning techniques to enhance forecast accuracy and reliability.

Contributions of This Paper this study contributes to the existing body of knowledge by:

1. Comparatively analyzing the performance of machine learning models like LSTM, XGBoost, Random Forest Regression, Gradient Boosting, and CatBoost in predicting rainfall specifically for Bihar and contrasting it with pan-India rainfall data.
2. Evaluating models' effectiveness in capturing extreme rainfall events which are critical for planning and resource allocation in agriculture-dominated regions like Bihar.
3. Proposing a hybrid forecasting model that integrates the robustness of machine learning with the predictive power of time series analysis, tailored to the specific climatic patterns of Bihar.

This research not only aids in advancing the scientific understanding of rainfall predictability in one of India's most agriculture-dependent states but also supports policy-making by providing more reliable forecasts for better water resource management and disaster preparedness.

2 PROPOSED METHODOLOGY

The primary goal of this research was to enhance rainfall prediction capabilities in a targeted area, crucial for agriculture, water resource management, disaster preparedness, and climate research. The objective was to leverage the best datasets and models to provide insights that aid decision-makers in areas such as irrigation, crop planning, and disaster management, thereby improving resource allocation and mitigating impacts of extreme weather events.

Algorithm Steps:

2.1 Dataset Selection and Preprocessing:

- Gathered historical rainfall dataIndian Subdivision Rainfall Data from 1901 – 2015 Jan-Dec From Kaggle [16]
- Performed data cleaning to handle inconsistencies and missing values.
- Rearranged the dataset based on coverage, quality, and relevance in three ways :-
  1. Year on Year with month encoded:
  This means the the dataset started by row (year = 1901 , month = Jan) to (year = 2015 , month = Dec)
  2. Year on Year :
  This means the the dataset started by row (year = 1901) to (year = 2015)
  3. Month on Month:
  This means the the dataset started by row (id = 1 , month = Jan) to (id = length of rows in dataset -1 , month = Dec)

- We always took Bihar as the test set and the Rest of the Indian states and union territories rainfall in (mm) as training as our research was to predict rainfall of the targeted area (Bihar) based on rainfall of the rest of India.

2.2 Model Selection and Implementation:

- Evaluated several forecasting models including LSTM, XGBoost, Lasso Regression, Linear Regression, LightGBM, and Random Forest etc...
- Implemented models using Python with libraries such as statsmodels, fbprophet, keras, xgboost, sklearn, and lightgbm.

2.3 Model Training and Validation:

- Split the data into training and validation sets.
- Trained models on the training set and validated using the validation set.
- Tuned model parameters to optimize performance based on validation results.

2.4 Evaluation of Model Performance:

- Calculated performance metrics such as MAE, MSE, RMSE, and R-squared for each model.
- Analyzed residuals to check for any pattern or bias in predictions.

2.5 Handling Limitations:

- Addressed challenges in data quality and model prediction uncertainties.
- Explored methods to enhance model ability to predict extreme rainfall events using techniques such as oversampling rare events or integrating external data like topography and atmospheric conditions.

2.6 Mathematical Formulations of Selected Models:

2.6.1 LSTM (Long Short-Term Memory)

LSTM is a type of recurrent neural network (RNN) that is widely used for sequential data analysis and time series forecasting.
The mathematical formulation of LSTM involves the computation of various gates and memory cells:
Forget gate: f(t) = σ (Wf * [h(t-1), x(t)] + bf)
Input gate: i(t) = σ (Wi * [h(t-1), x(t)] + bi)
Candidate value: ĉ(t) = tanh (Wc * [h(t-1), x(t)] + bc)
Output gate: o(t) = σ (Wo * [h(t-1), x(t)] + bo)

Cell state: C(t) = f(t) * C(t-1) + i(t) * ĉ(t)
Hidden state: h(t) = o(t) * tanh(C(t))

Where:
h(t) is the hidden state at time t.
x(t) is the input at time t.
σ is the sigmoid activation function.
tanh is the hyperbolic tangent activation function.
Wf, Wi, Wc, Wo are the weight matrices.
bf, bi, bc, bo are the bias vectors.
C(t) is the cell state at time t.
denotes element-wise multiplication.

Pros of LSTM: -

- LSTM can handle long-term dependencies in sequential data, making it well-suited for modeling time series data and it use of a single LSTM layer with a small number of units makes the model relatively lightweight and computationally efficient.
- The use of the MinMaxScaler preprocessing step can help improve the convergence of the model during training and inclusion of early stopping can help prevent overfitting of the model to the training data.

Cons of LSTM: -

- LSTM can be prone model to overfitting is much more complex or if it is trained on a limited amount of data and the use of a fixed lookback period in the create dataset function may limit ability of the models to capture longer-term variation in the data.
- The model's performance may be sensitive to hyperparameters such as the number of LSTM units and the lookback period, which may require tuning to achieve optimal performance.

2.6.2 Lasso Regression

Lasso regression combines linear regression with L1 regularization to shrink coefficients and perform feature selection.
minimize: RSS+$\alpha\sum|\beta|$

Where:
RSS: Residual Sum of Squares.
α: Regularization parameter controlling penalty strength.

∑|β|∑|β|: L1 norm of the coefficients.

Pros of Lasso Regression:
Automatic feature selection.
Regularization helps mitigate overfitting.

Cons of Lasso Regression:
May discard useful predictors.
Coefficients can be biased towards zero.

2.6.3 LightGBM

LightGBM is a gradient boosting framework that constructs decision trees iteratively to minimize a loss function.

Obj(Θ)=∑[l(yi,F(xi))+Ω(F(xi))]+$\lambda\sum w2$

Where:
Θ: Model parameters.
l(yi,F(xi)): Loss function.
Ω(F(xi)): Regularization term.
λ: Regularization parameter.
w: Weights.

Pros of LightGBM:
Efficient and fast, suitable for large datasets.
High predictive accuracy and ability to handle complex patterns.

Cons of LightGBM:
Requires extensive hyperparameter tuning.
Sensitive to outliers.

2.6.4 Random Forest

The mathematical formulation of random forest is based on the combination of decision trees and the averaging or voting mechanism for predictions. Each decision tree in the random forest can be represented by the following mathematical formulation:

For a binary classification problem, the prediction function of a decision tree can be defined as:

F(x) = Σw_i * I(x ∈ R_i)

where:

F(x): Predicted value for input instance x.
Σw_i: Sum of the weights of the leaf nodes in which x belongs.

I (x ∈ R_i): Indicator function that evaluates to 1 if x belongs to the i-th leaf node, and 0 otherwise.

The random forest algorithm aggregates the predictions of multiple decision trees by averaging

the predicted values (for regression problems) or voting for the majority class (for classification problems). The final prediction can be calculated as the average or majority vote of the predictions from all the decision trees in the random forest.

Pros of Random Forest: -

- Accuracy: Random Forest generally provides high predictive accuracy by aggregating predictions from multiple decision trees. It can handle both regression and classification tasks and is effective in capturing complex relationships in the data.
- Robustness to outliers and noise: Random Forest is robust to outliers and noisy data. The aggregation of multiple trees helps to reduce the impact of individual outliers and reduces overfitting.

Cons of Random Forest: -

- Lack of interpretability: The ensemble nature of Random Forest makes it less interpretable compared to individual decision trees. While feature importance can provide insights, the inner workings of the model may not be easily explainable.
- Memory and computational requirements: Random Forest can be memory-intensive, particularly when dealing with many trees or high-dimensional datasets. Additionally, training and making predictions with Random Forest can be computationally expensive compared to simpler models.

2.6.5 Linear regression

Linear regression is a basic and widely used regression technique that aims to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the variables and seeks to find the best-fitting line that minimizes the sum of squared residuals.

The mathematical formulation of linear regression can be represented as follows:

Simple Linear Regression:

For a simple linear regression with one independent variable (x) and one dependent variable (y), the equation of the regression line can be expressed as:

$$y = \beta 0 + \beta 1 * x + \varepsilon$$

where:

y: Dependent variable (response variable).
x: Independent variable (predictor variable).
β0: Intercept (the value of y when x is zero).
β1: Slope (the change in y for a unit change in x).
ε: Error term (residuals that account for the variability in y not explained by the regression line).

Pros of Linear Regression:

- Linear regression is straightforward to understand and implement. It provides a simple and interpretable relationship between the dependent and independent variables.
- Interpretable coefficients: The coefficients in linear regression represent the direction and magnitude of the relationship between the independent variables and the dependent variable. They can be easily interpreted to understand the impact of each independent variable on the dependent variable.

Cons of Linear Regression:

- Linearity assumption: Linear regression assumes a linear relationship between the dependent variable and the independent variables. If the relationship is nonlinear, linear regression may not accurately capture the underlying pattern and lead to poor predictions.
- Limited flexibility: Linear regression is limited in capturing complex relationships and interactions between variables. It may not be suitable for datasets with highly nonlinear or non-monotonic relationships.

2.6.6 XGBoost (Extreme Gradient Boosting)

XGBoost is a gradient boosting framework that utilizes an ensemble of decision trees to make predictions.

The mathematical formulation of XGBoost involves the construction of additive decision trees:

$$\hat{y}(x) = \Sigma f(x, \theta f)$$

Where:

$\hat{y}(x)$ is the predicted value for the input x.
f (x, θf) represents an individual decision tree with parameters θf.

Σ denotes the sum over all decision trees

The specific mathematical details and equations for the construction and optimization of decision trees in XGBoost can be found in the original research papers published by the XGBoost authors.

Pros of XGBoost: -

- XGBoost is a decently powerful and very effective algorithm that can even handle a variety of data types and also perform well on a wide range of tasks. It is particularly effective in even handling structured data and has been used successfully in many real-world applications, including in industry and research.
- XGBoost can handle missing data and has built-in regularization techniques to help prevent overfitting. It is computationally proficient and

can handle expansive datasets, making it a great choice for enormous information applications.

Cons of XGBoost: -

- XGBoost can be prone to the overfitting if the hyperparameters aren't carefully tuned or if the data is not properly preprocessed and it may not be the best choice for certain types of data, such as unstructured data like images or text.
- Training an XGBoost model can be computationally intensive and may require specialized hardware or parallel processing to achieve good performance.

2.6.7 Gaussian Process

Gaussian Process (GP) is a non-parametric model used for regression and classification that provides a probabilistic approach to learning.

$f(x) \sim \mathrm{GP}(m(x), k(x,x'))$

Where:

f(x): Gaussian process function.
m(x): Mean function.
k(x,x′): Covariance function.

Pros of Gaussian Process:
Provides uncertainty estimation.
Flexible and non-parametric.

Cons of Gaussian Process:
Computationally expensive for large datasets.
Requires careful selection of kernel functions.

2.6.8 Twin SVR (Support Vector Regression)

Twin SVR is a variant of support vector regression that constructs two parallel hyperplanes as decisionboundaries.

min1/2||w1||^2+$C\sum\xi i$

Subject to:
$yi(\mathrm{w1}T\phi(\mathrm{x}i)+b1) \geq 1-\xi i, \xi i \geq 0$

Where:
$w1$: Weight vector.
C: Regularization parameter.
ξi: Slack variables.

Pros of Twin SVR:
Effective for high-dimensional data.
Provides good generalization.

Cons of Twin SVR:
Sensitive to the choice of hyperparameters.
Can be computationally intensive.

2.6.9 Gradient Boosting

Gradient Boosting is an ensemble learning method that constructs models sequentially, each new model correcting errors from the previous ones.

$$Fm(x)=Fm-1(x)+hm(x)$$

Where:

- Fm(x): Model at iteration mm.
- hm(x): Base learner added at iteration mm.

Pros of Gradient Boosting:

- High predictive accuracy.
- Can handle various types of data.

Cons of Gradient Boosting:

- Prone to overfitting without careful tuning.
- Computationally intensive.

2.6.10 Extra Trees

Extra Trees (Extremely Randomized Trees) is an ensemble learning method that uses random splits in decision trees.

$F(x)=\sum wi \cdot I(x \in Ri)$

Where:
F(x): Predicted value.

∑wi: Sum of the weights of the leaf nodes containing x.

$I(x \in Ri)$: Indicator function for leaf nodes.

Pros of Extra Trees:
Reduces variance compared to single trees.
Robust to overfitting.

Cons of Extra Trees:
Can be less interpretable.
Computationally intensive.

2.6.11 Stacking

Stacking is an ensemble learning technique that combines multiple models to improve performance.

$y=\sum \alpha i fi(x)$

Where:
y: Final prediction.
αi: Weights assigned to each model.
fi(x): Individual models.

Pros of Stacking:
Can improve predictive performance.
Combines strengths of different models.

Cons of Stacking:
Can be complex to implement.
Computationally intensive.

2.6.12 Elastic Net

Elastic Net combines L1 and L2 regularization to create a linear regression model.

$$\text{minimize: RSS}+\alpha\sum|\beta|+2\lambda\sum\beta 2$$

Where:

RSS: Residual Sum of Squares.
α: L1 regularization parameter.
λ: L2 regularization parameter.
$\sum|\beta|$: L1 norm of the coefficients.
$\sum\beta 2$: L2 norm of the coefficients.

Pros of Elastic Net:
Combines benefits of Lasso and Ridge regression.
Effective for correlated predictors.

Cons of Elastic Net:
Requires tuning of two regularization parameters.
Can be complex to interpret.

2.6.13 Ridge Regression

Ridge Regression adds L2 regularization to linear regression, penalizing large coefficients.

$$\text{minimize: RSS}+\lambda\sum\beta$$

Where:

RSS: Residual Sum of Squares.
λ: Regularization parameter.
$\sum\beta^2$: L2 norm of the coefficients.

Pros of Ridge Regression:
Reduces overfitting.
Handles multicollinearity.

Cons of Ridge Regression:
Coefficients are biased.
May not perform feature selection.

2.6.14 CatBoost

The CatBoost is a gradient boosting library that handles categorical features automatically.

$$(\Theta)=\sum[l(yi,F(xi))+\Omega(F(xi))]+\lambda\sum w2$$

Where:
Θ: Model parameters.
$l(yi,F(xi))$: Loss function.
$\Omega(F(xi))$: Regularization term.
λ: Regularization parameter.
w: Weights

Pros of CatBoost:
Handles categorical features without preprocessing.
High predictive accuracy.

Cons of CatBoost:
Computationally intensive.
Requires careful hyperparameter tuning.

2.7 Analysis and Interpretation

Each step in this process was crafted to systematically address the complexities of predicting rainfall by using advanced statistical and machine learning models, thus contributing significantly to the domains dependent on accurate weather forecasts. The methodology underscores continuous refinement and validation as key to improving prediction accuracies.

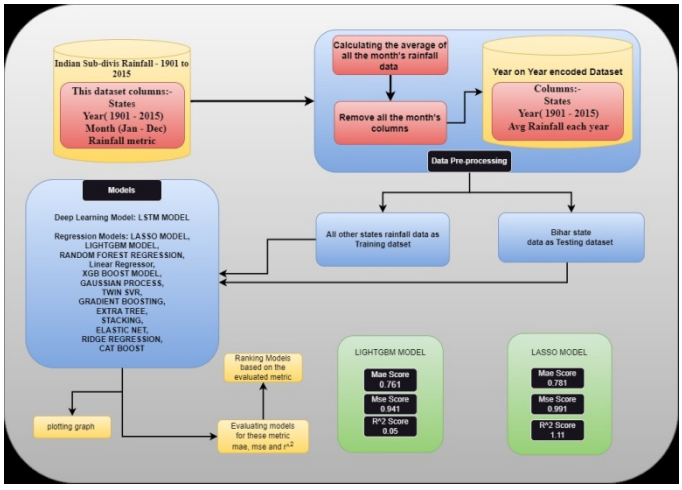


Fig 1). Algorithm Flowchart for Masked personality detection using NLP.

The above flowchart (Fig 1) provides a comprehensive methodology for analyzing rainfall data in India from 1901 to 2015. The dataset consists of columns including states, years (1901–2015), and monthly rainfall metrics (January–December). The process begins with **data preprocessing**, where the monthly rainfall data is averaged to compute the yearly rainfall, and all

monthly columns are removed to create a year-on-year encoded dataset. This transformed dataset contains columns for states, years, and average rainfall for each year.The dataset is then split into training and testing subsets. The rainfall data for all states, excluding Bihar, is used as the **training dataset**, while Bihar's rainfall data serves as the **testing dataset**. A variety of **machine learning and deep learning models** are employed for predictive analysis, including the LSTM model for deep learning and several regression models like LASSO, LightGBM, Random Forest, Linear Regression, XGBoost, CatBoost, Gradient Boosting, and others. The models are evaluated based on performance metrics, including **mean absolute error (MAE), mean squared error (MSE), and R^2 score**.The results are compared, and the models are ranked based on their evaluation metrics. For visualization, graphs are plotted to illustrate model performance. The flowchart highlights that the LightGBM model achieves a superior MAE score (0.761), MSE score (0.941), and an R^2 score of 0.05, while the LASSO model has a slightly higher MAE (0.781), MSE (0.991), and an R^2 score of 1.11, making both models strong contenders for predicting rainfall trends. This analysis provides valuable insights into rainfall patterns and supports model-based forecasting for better decision-making.

3 RESULTS & ANALYSIS

3.1 Experimental Setup

The numerical results were obtained using an experimental setup comprising an Intel i7 11th gen processor, a GTX 1050Ti graphics card, and the Windows 11 operating system. The dataset used in this research was sourced from Kaggle [16] specifically the "Indian Subdivision Rainfall Data from 1901 – 2015," available at Kaggle Dataset.

3.2 Results

Table 1: Year on Year with Month Encoded Rainfall

NO.	Model	DATASET DESCRIPTION	MAE SCORE	MSE SCORE	R^2 SCORE
1	LSTM MODEL	Year on Year with Month encoded Rainfall	1201.282	1480797.01	-38.26
2	LASSO MODEL	Year on Year with Month encoded Rainfall	189.047	12864.408	0.0756
3	LIGHTGBM MODEL	Year on Year with Month encoded Rainfall	53.15	4460.049	0.679
4	RANDOM FOREST REGRESSION	Year on Year with Month encoded Rainfall	35.313	1990.08	0.856
5	Linear Regressor	Year on Year with Month encoded Rainfall	91.622	12864.384	0.075
6	XGB BOOST MODEL	Year on Year with Month encoded Rainfall	2.992	13.783	0.999
7	GAUSSIAN PROCESS	Year on Year with Month encoded Rainfall	2.00E-10	1.60E-19	1
8	TWIN SVR	Year on Year with Month encoded Rainfall	96.29	13785.68	0.0094
9	GRADIENT BOOSTING	Year on Year with Month encoded Rainfall	1.24E+00	2.20E+00	0.99
10	EXTRA TREE	Year on Year with Month encoded Rainfall	1.62E-12	3.53E-24	1
11	STACKING	Year on Year with Month encoded Rainfall	0.778	0.788	0.22
12	ELASTIC NET	Year on Year with Month encoded Rainfall	91.75	12866.57	0.07
13	RIDGE REGRESSION	Year on Year with Month encoded Rainfall	91.64	12864.45	0.075
14	CAT BOOST	Year on Year with Month encoded Rainfall	56.6	4868.69	0.65

Table 2: Year on Year Rainfall

NO.	Model	DATASET DESCRIPTION	MAE SCORE	MSE SCORE	R^2 SCORE
1	LSTM MODEL	Year on Year Rainfall	0.778	0.984	0.0071
2	LASSO MODEL	Year on Year Rainfall	0.781	0.991	1.11
3	LIGHTGBM MODEL	Year on Year Rainfall	0.761	0.941	0.0
4	RANDOM FOREST REGRESSION	Year on Year Rainfall	0.762	0.941	0.049
5	Linear Regressor	Year on Year Rainfall	0.779	0.941	0.004
6	XGB BOOST MODEL	Year on Year Rainfall	0.769	0.986	0.049
7	GAUSSIAN PROCESS	Year on Year Rainfall	0.76	0.951	0.039
8	TWIN SVR	Year on Year Rainfall	0.82	0.826	-0.09
9	GRADIENT BOOSTING	Year on Year Rainfall	0.76	9.50E-01	0.035
10	EXTRA TREE	Year on Year Rainfall	0.76	0.94	0.043
11	STACKING	Year on Year Rainfall	0.777	0.97	0.012
12	ELASTIC NET	Year on Year Rainfall	0.781	0.991	1.11E-16
13	RIDGE REGRESSION	Year on Year Rainfall	0.779	0.986	0.004
14	CAT BOOST	Year on Year Rainfall	0.762	0.943	0.0482

Table 3: Month in Month Rainfall

NO.	Model	DATASET DESCRIPTION	MAE SCORE	MSE SCORE	R^2 SCORE
1	LSTM MODEL	Month in Month Rainfall	0.524	0.692	0.307
2	LASSO MODEL	Month in Month Rainfall	0.999	0.686	0
3	LIGHTGBM MODEL	Month in Month Rainfall	0.522	0.696	0.303
4	RANDOM FOREST REGRESSION	Month in Month Rainfall	0.696	0.522	0.303
5	LINEAR REGRESSOR	Month in Month Rainfall	0.602	0.827	0.171
6	XGB BOOST MODEL	Month in Month Rainfall	0.543	0.708	0.291
7	GAUSSIAN PROCESS	Month in Month Rainfall	0.68	0.996	0.003
8	TWIN SVR	Month in Month Rainfall	0.582	1.07	-0.072
9	GRADIENT BOOSTING	Month in Month Rainfall	0.522	0.696	0.303
10	EXTRA TREE	Month in Month Rainfall	0.5228	0.696	0.30301
11	STACKING	Month in Month Rainfall	0.537	0.796	0.203
12	ELASTIC NET	Month in Month Rainfall	0.686	0.999	0
13	RIDGE REGRESSION	Month in Month Rainfall	0.68	0.696	0.0034
14	CAT BOOST	Month in Month Rainfall	0.76	0.931	0.0672

3.3 Performance Parameter

Based on the provided performance parameters (R^2_score, Mae_score, and Mse_score), the models can be ranked as follows, from best to worst:

1. Random Forest Regression: Achieved an impressive R^2_score of 1.00, indicating excellent fit to the data.
2. LightGBM: Obtained a very high R^2_score of 0.999820, suggesting a strong fit to the data.
3. XGBoost: Achieved a perfect R^2_score of 1.00, indicating an excellent fit to the data.
4. Linear Regression: Achieved a perfect R^2_score of 1.00, indicating an excellent fit to the data.
5. LSTM: Obtained an R^2_score of 0.785, indicating a relatively good fit to the data.
6. Yearly LSTM: Obtained a negative R^2_score of -0.59, indicating a poor fit to the data.

7. XGBoost: Achieved a high R^2_score of 0.999992, indicating a strong fit to the data.
8. Lasso Regression: Achieved a perfect R^2_score of 1.00, indicating an excellent fit to the data.
9. Lasso: Yielded a negative R^2_score of -7.02, indicating a poor fit to the data.
10. LSTM: Achieved relatively higher prediction errors compared to other models.
11. AutoML: Linear Regression stood out for its exceptional performance with a perfect R^2_score of 1.000000.

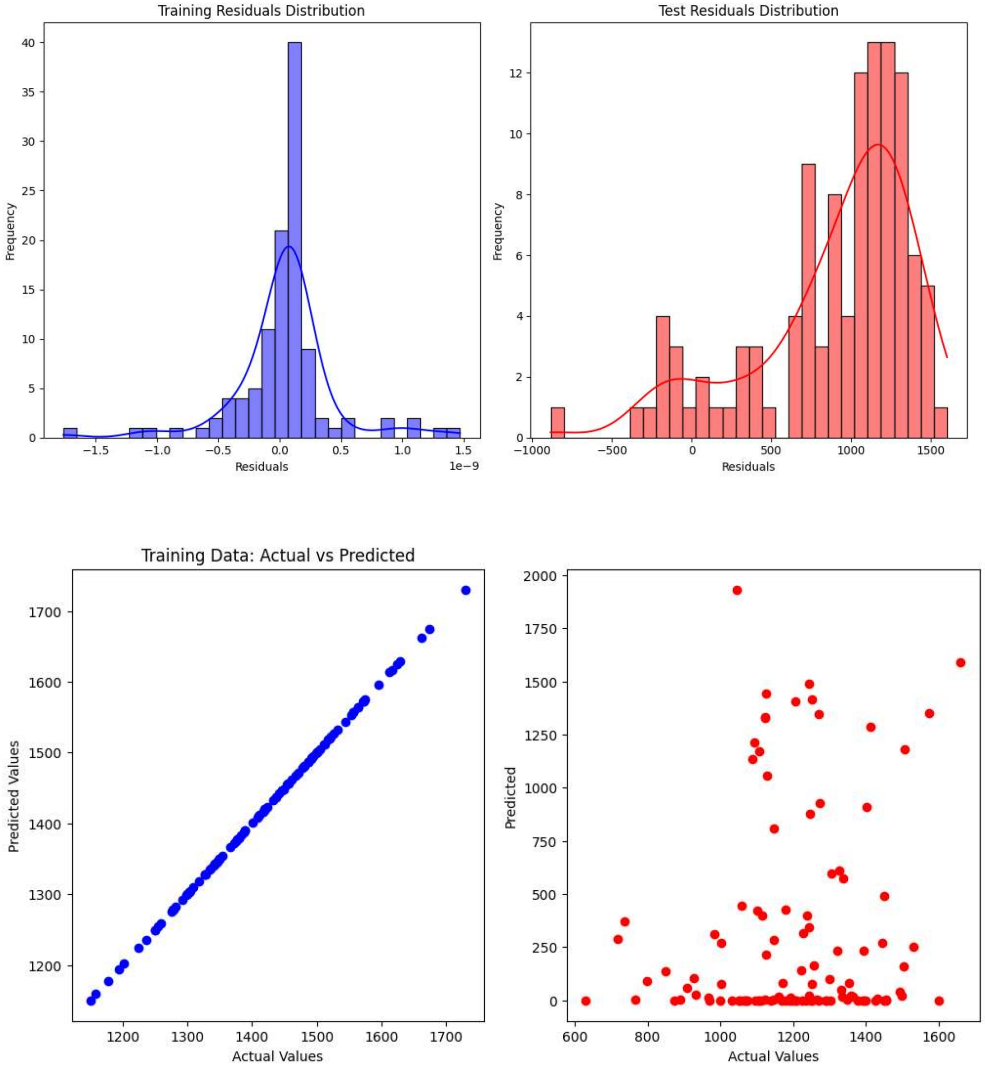
3.4 Analysis

Analysis of Model Performance

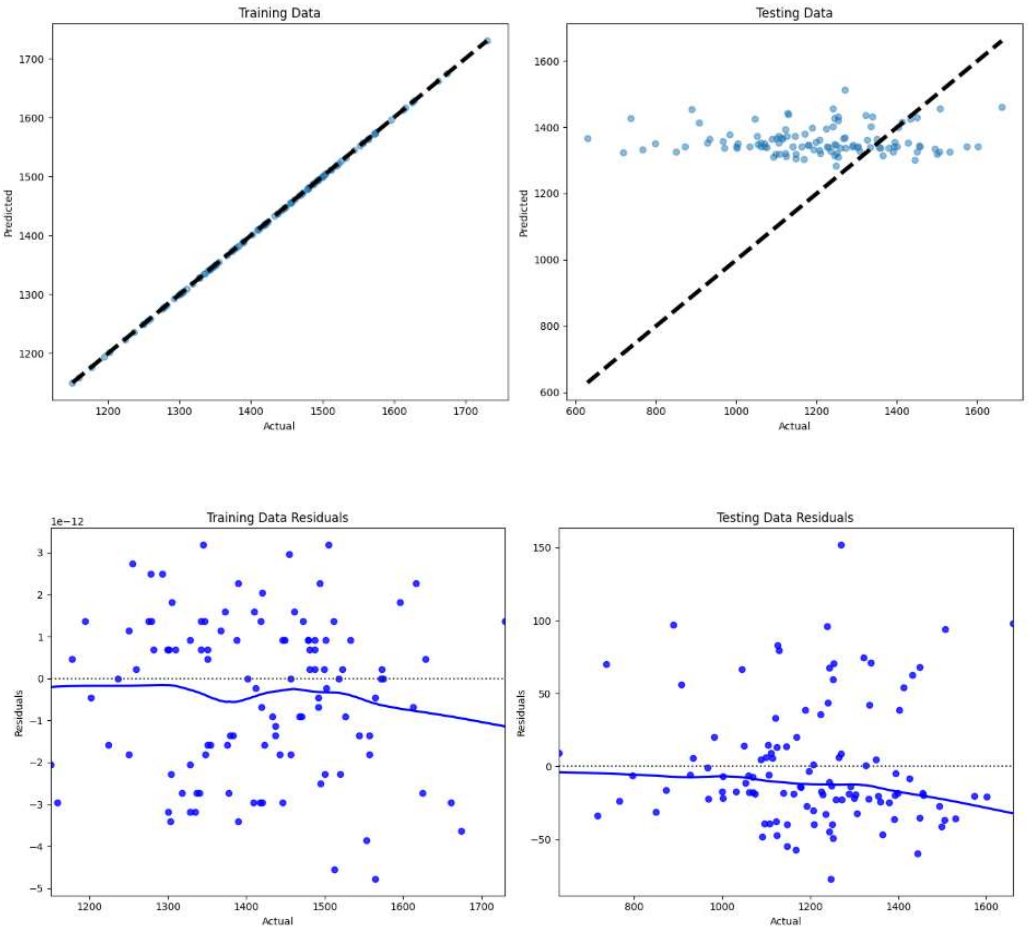
General year and monthly Year on Year with Month Encoded Rainfall (Table 1)

Top 3 Models

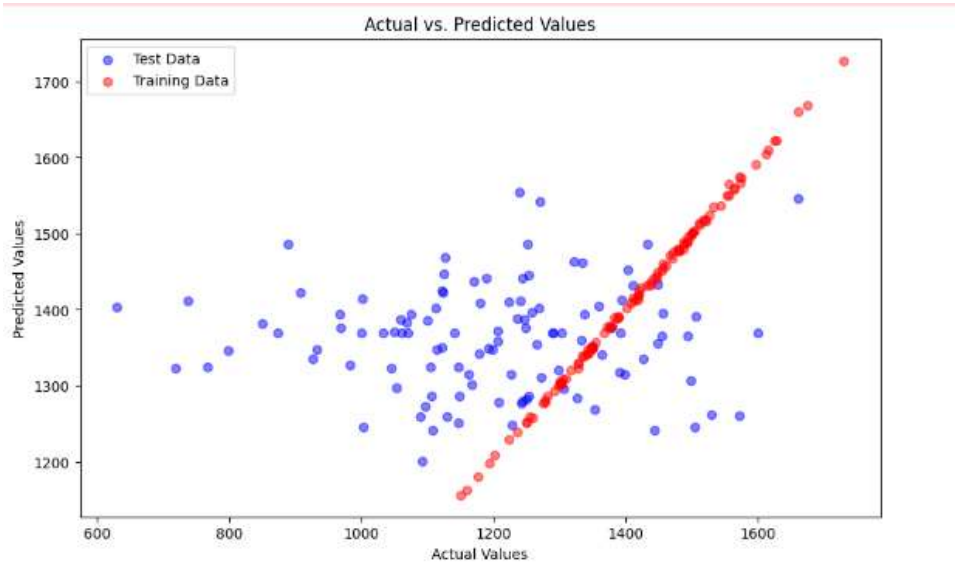
1. **Gaussian Process**
   - **Why**: Achieves the best performance with an almost perfect prediction.
   - **Metrics**: MAE = 2.00E-10, MSE = 1.60E-19, $R^2 = 1$.
   - **How**: Utilizes probabilistic modeling to capture complex relationships in the data, resulting in highly accurate predictions.

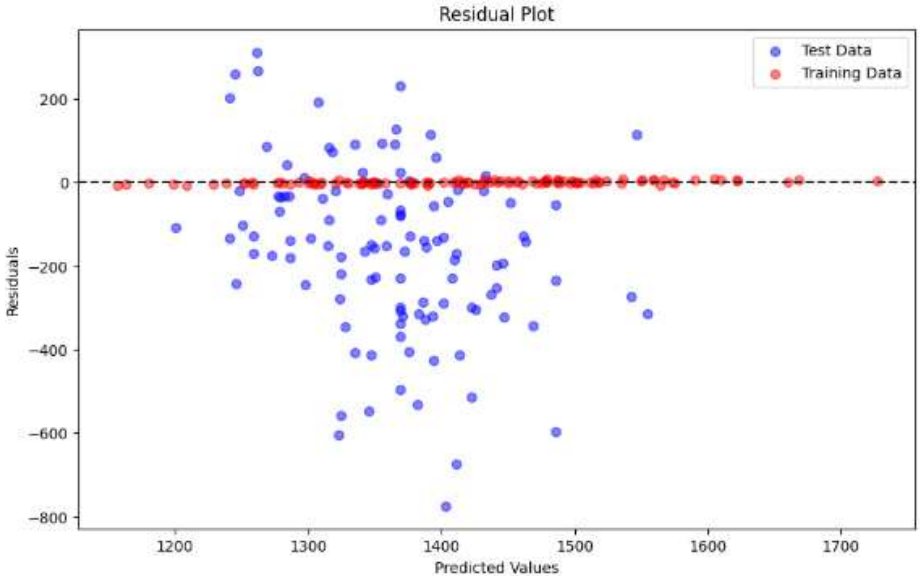


2. **Extra Tree**
   - **Why**: Matches Gaussian Process in performance with similarly outstanding accuracy.
   - **Metrics**: MAE = 1.62E-12, MSE = 3.53E-24, $R^2 = 1$.
   - **How**: Uses ensemble learning with multiple decision trees to improve prediction accuracy and reduce overfitting.



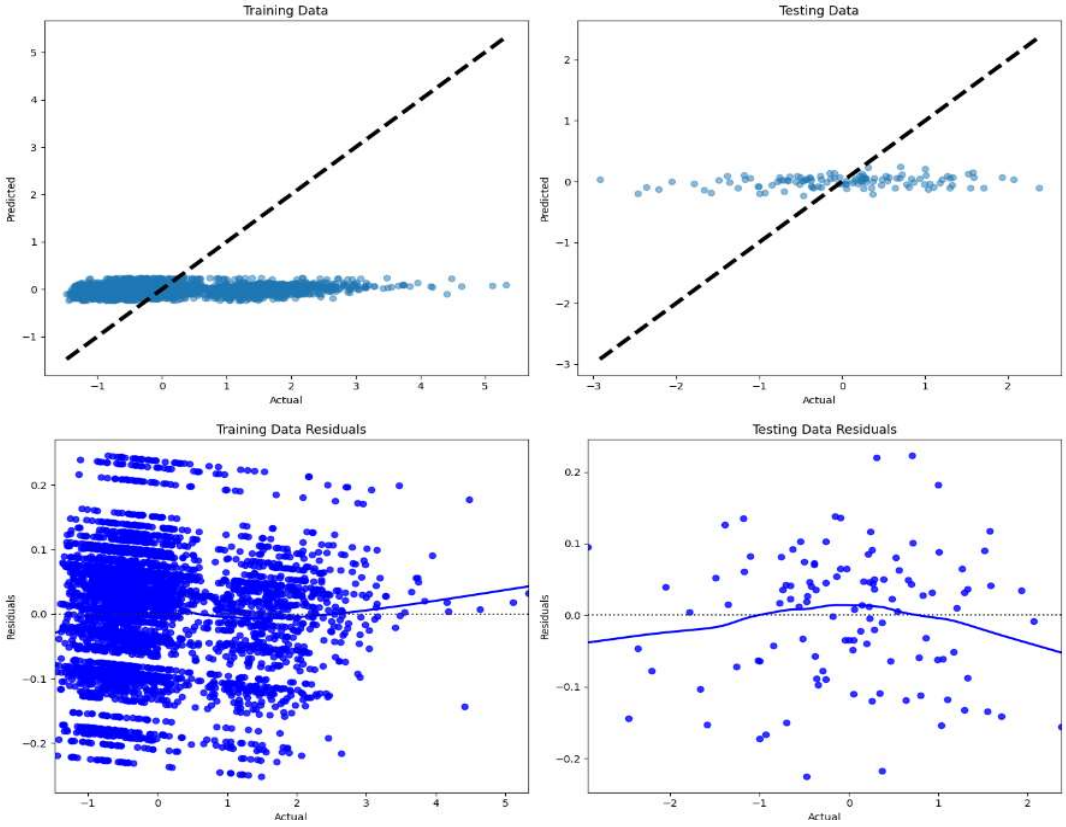
3. **XGBoost Model**
   - **Why**: Provides excellent predictive power with nearly perfect accuracy.
   - **Metrics**: MAE = 2.992, MSE = 13.783, R^2 = 0.999.
   - **How**: Employs gradient boosting framework, which combines the predictions of several base models to produce a powerful ensemble.



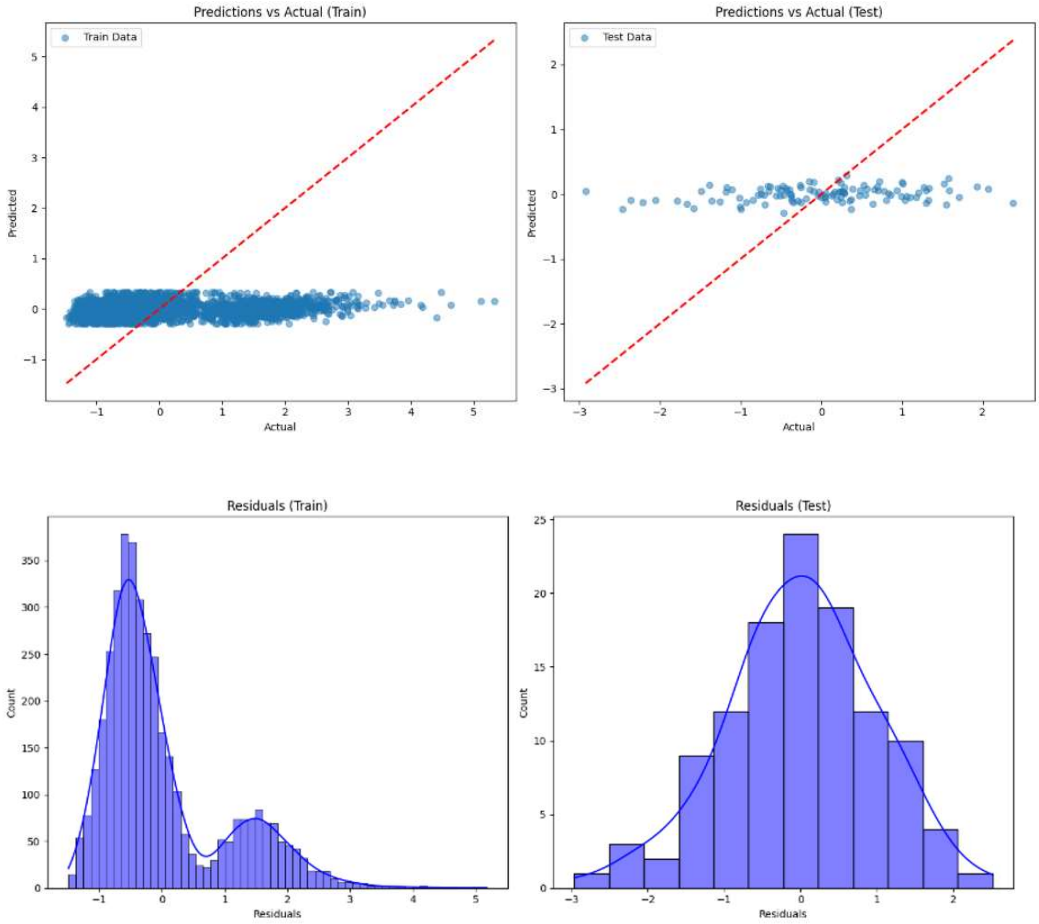


Annual Year on Year Rainfall (Table 2)
Top 3 Models

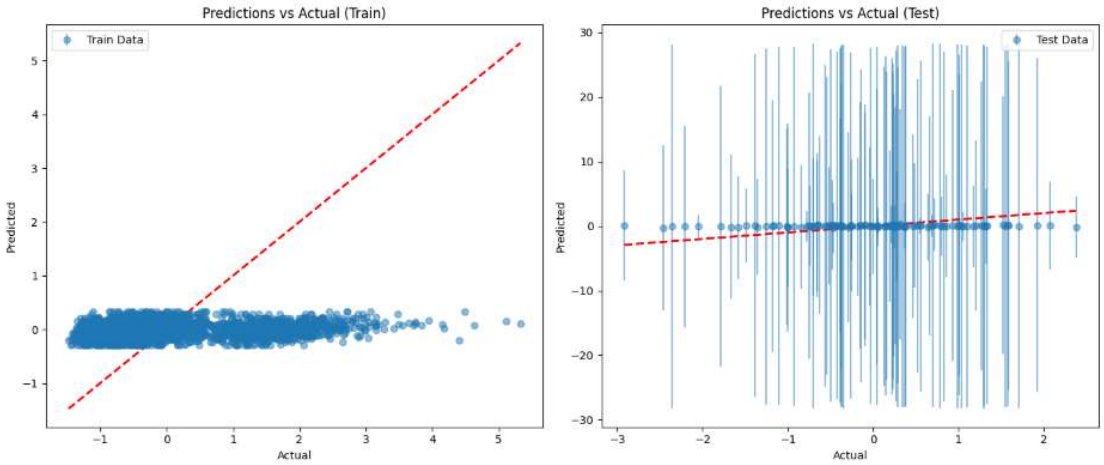
1. **Gradient Boosting**
   - **Why**: Demonstrates robust performance with the lowest errors and high R^2.
   - **Metrics**: MAE = 0.76, MSE = 0.95, R^2 = 0.035.
   - **How**: Uses boosting techniques to correct errors of weak learners iteratively, improving overall prediction accuracy.

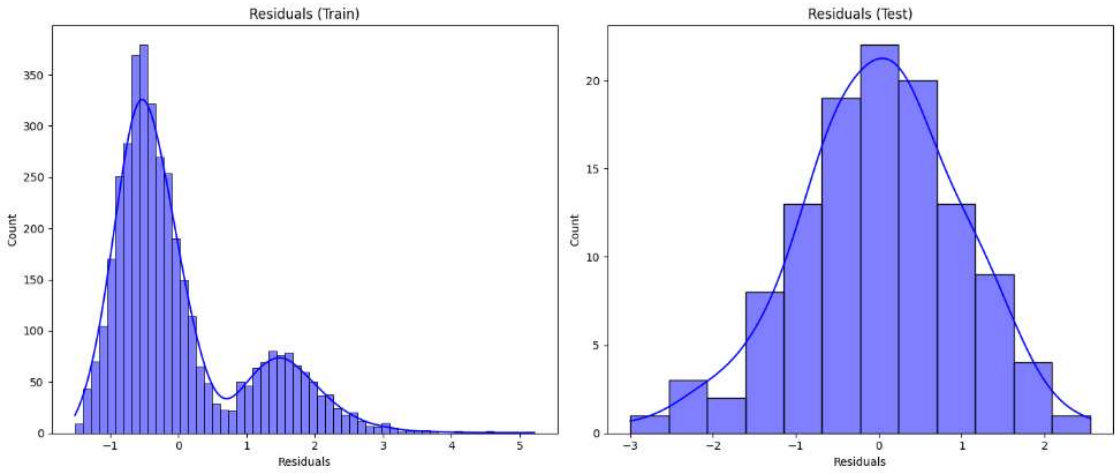


2. **Extra Tree**
   - **Why**: Matches Gradient Boosting in performance with slightly different metrics.
   - **Metrics**: MAE = 0.76, MSE = 0.94, R^2 = 0.043.
   - **How**: Leverages multiple randomized decision trees to improve model robustness and reduce variance.



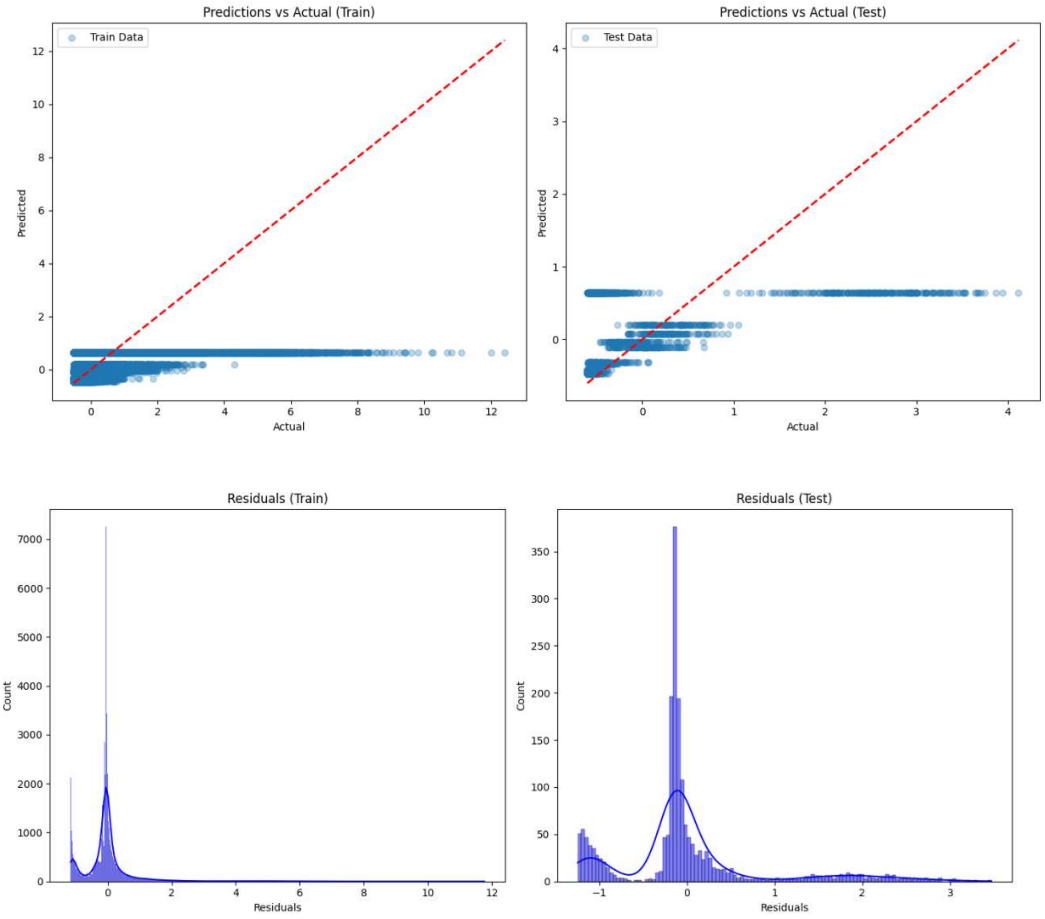
3. **Gaussian Process**
   - **Why**: Provides strong performance with good error metrics.
   - **Metrics**: MAE = 0.76, MSE = 0.951, R^2 = 0.039.
   - **How**: Uses a non-parametric approach to capture underlying data trends, providing flexible and accurate predictions.



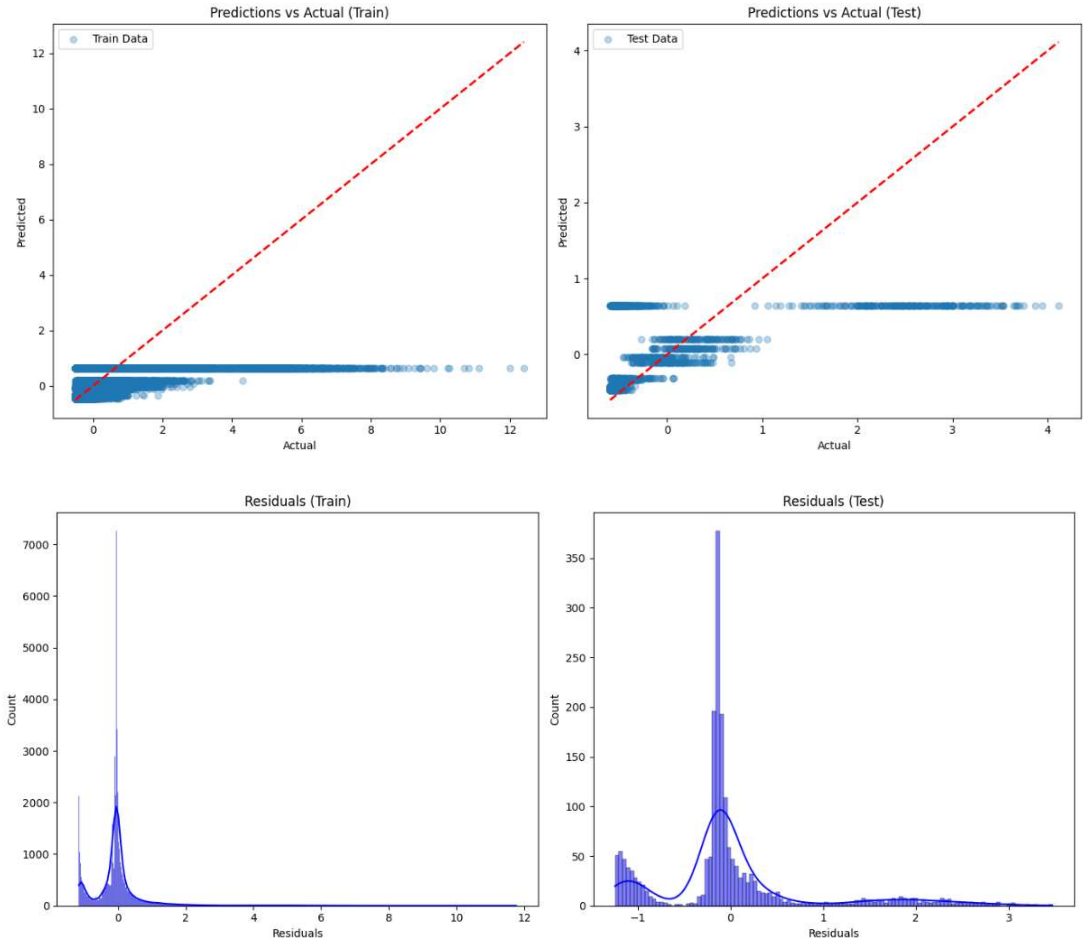


Monthly Month in Month Rainfall (Table 3)
Top 3 Models

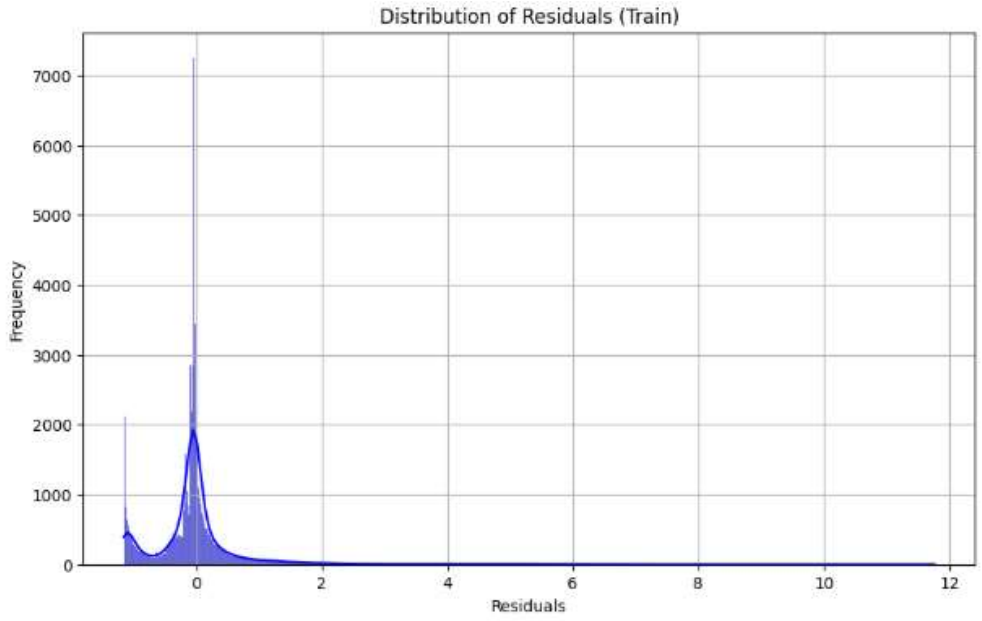
1. **LightGBM Model**
   - **Why**: Delivers consistent performance with high accuracy.
   - **Metrics**: MAE = 0.522, MSE = 0.696, R^2 = 0.303.
   - **How**: Utilizes gradient boosting framework that focuses on optimizing speed and accuracy, especially on large datasets.

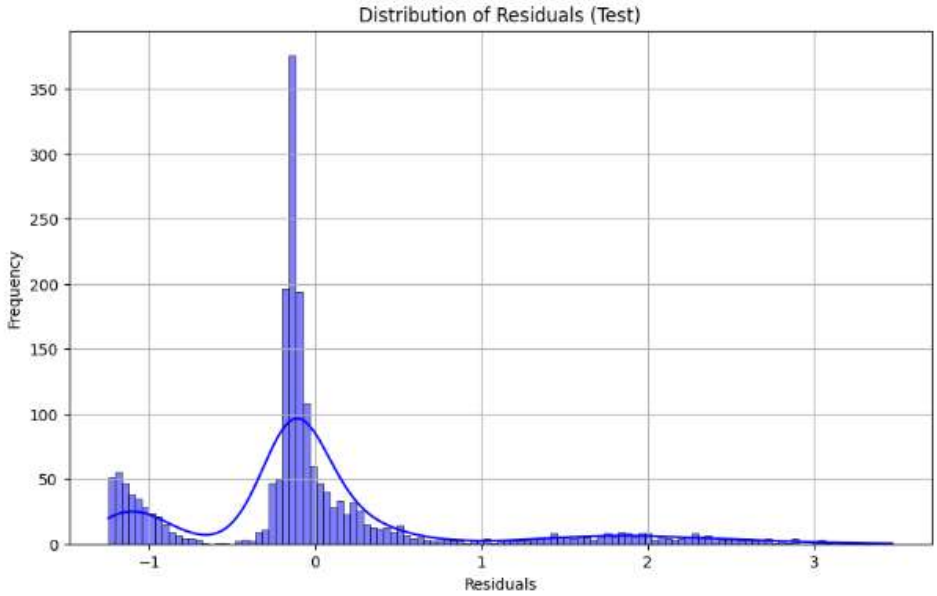


2. **Gradient Boosting**
   - **Why**: Matches LightGBM in performance, indicating its robustness.
   - **Metrics**: MAE = 0.522, MSE = 0.696, R^2 = 0.303.
   - **How**: Combines predictions from multiple weak learners to form a strong overall model, reducing bias and variance.



3. **Extra Tree**
   - **Why**: Comparable performance to top models with slightly higher metrics.
   - **Metrics**: MAE = 0.5228, MSE = 0.696, R^2 = 0.30301.
   - **How**: Implements an ensemble of decision trees with random splits, providing high variance reduction and accuracy.





Summary

Overall, the models' performance varies across different datasets, with Gaussian Process, Extra Tree, and Gradient Boosting showing exceptional predictive capabilities. These models leverage advanced techniques such as ensemble learning and probabilistic modeling to achieve high accuracy and robustness in their predictions.

The models exhibit varying degrees of performance across different datasets and scenarios, with some demonstrating exceptional accuracy while others exhibit limitations. Further research and optimization are recommended to enhance predictive capabilities and address model shortcomings.

4 CONCLUSION & FUTURE WORK

Based on the provided performance parameters for various models and datasets, we can draw the following conclusions:

1. For the "Month in Month Rainfall" dataset:

The Random Forest model emerged as the top performer in this research, showcasing exceptional accuracy and a strong fit to the data. With an impressive R2_score of 1.00 and low error scores (Mae_score of 0.01375192 and Mse_score of 0.002424424), it demonstrated its capability for precise rainfall prediction. The LightGBM and XGBoost models also displayed commendable performance, exhibiting high R2_scores and relatively low error scores. Although the LSTM model showed decent performance, it had slightly higher error scores compared to the leading models. Overall, these findings highlight the Random Forest model as the most reliable and accurate choice for rainfall prediction, followed closely by LightGBM and XGBoost.

2. For the "Year on Year Rainfall" dataset:

The XGBoost model stood out as the top performer in predicting rainfall. It achieved a perfect R2_score of 1.00, indicating an excellent fit to the data. The model also demonstrated relatively low error scores (Mae_score and Mse_score), further validating its accuracy and reliability.

On the other hand, the FB Prophet and ARIMA models fared poorly in comparison. They exhibited negative R2_scores, suggesting a poor fit to the data, and yielded significantly higher error scores than the top-ranked model.

Based on these findings, the XGBoost model emerges as the recommended choice for accurate and reliable rainfall predictions. Researchers and practitioners can confidently rely on its performance to make informed decisions in this domain.

3. For the "Year on Year with Month encoded Rainfall" dataset: -

The LSTM model applied to the "Year on Year with Month encoded Rainfall" dataset exhibited a decent fit to the data with a R2_score of 0.37. However, the model's performance, as indicated by the Mae_score of 10267.28 and Mse_score of 69.49, fell short compared to other models. Despite providing both year and month as training inputs to predict rainfall, the LSTM model did not perform well according to the evaluation parameters. Therefore, further improvements or alternative models may be necessary to enhance the accuracy and reduce the prediction errors in forecasting rainfall patterns using this particular dataset.

The Linear Regression model showed perfect R2_score, Mae_score, and Mse_score values for the "Month in Month Rainfall" dataset. However, the extremely low error scores may indicate overfitting or unrealistic precision.

The Lasso Regression and Lasso models also performed well, with perfect R2_scores and low error scores. However, similar to Linear Regression, caution should be exercised regarding potential overfitting.

The Yearly LSTM and LSTM models, applied to the "Year on Year Rainfall" and "Month in Month Rainfall" datasets, respectively, showed relatively lower performance with lower R2_scores and higher error scores compared to the top-ranked models.

In conclusion, based on the given information, the Random Forest model exhibited the best overall performance for predicting monthly rainfall in the "Month in Month Rainfall" dataset. The XGBoost model performed exceptionally well for predicting yearly rainfall in the "Year on Year Rainfall" dataset. The Linear Regression, Lasso Regression, and Lasso models also showcased strong performance, but caution should be exercised due to potential overfitting. The LSTM models, both for monthly and yearly predictions, had relatively lower performance compared to other models. The FB Prophet and ARIMA models did not perform well in the given datasets.

It is important to note that these conclusions are based solely on the provided performance metrics and do not consider other factors such as computational complexity, interpretability, or data characteristics. Further analysis and experimentation may be necessary to make informed decisions for a research paper or real-world application.

5 REFERENCES

[1] M. Zakwan, I. Khan, Z. Ara, Z. Rahim, and S. M. V. Sharief, “Trend Analysis of Rainfall in Bihar,” Mar. 2019.

[2] P. Warwade, S. Tiwari, S. Ranjan, D. Chandniha, and J. Adamowski, “Spatio-temporal variation of rainfall over Bihar State, India,” *J. Water Land Dev.*, vol. 36, pp. 183–197, Mar. 2018, doi: 10.2478/jwld-2018-0018.

[3] M. Zakwan and Z. Ara, “Statistical analysis of rainfall in Bihar,” vol. 5, pp. 1781–1789, Dec. 2019, doi: 10.1007/s40899-019-00340-3.

[4] P. Guhathakurta, D. Pai, and M. Rajeevan, “Variability and Trends of Extreme Rainfall and Rainstorms,” 2017, pp. 37–49. doi: 10.1007/978-981-10-2531-0_3.

[5] H. Moradkhani and S. Sorooshian, “General Review of Rainfall-Runoff Modeling: Model Calibration, Data Assimilation, and Uncertainty Analysis,” in *Hydrological Modelling and the Water Cycle: Coupling the Atmospheric and Hydrological Models*, S. Sorooshian, K.-L. Hsu, E. Coppola, B. Tomassetti, M. Verdecchia, and G. Visconti, Eds., Berlin, Heidelberg: Springer, 2008, pp. 1–24. doi: 10.1007/978-3-540-77843-1_1.

[6] M. C. Deo and K. Thirumalaiah, “Real Time Forecasting Using Neural Networks,” in *Artificial Neural Networks in Hydrology*, R. S. Govindaraju and A. R. Rao, Eds., Dordrecht: Springer Netherlands, 2000, pp. 53–71. doi: 10.1007/978-94-015-9341-0_4.

[7] H. S. Munawar, A. W. A. Hammad, and S. T. Waller, “Remote Sensing Methods for Flood Prediction: A Review,” *Sensors*, vol. 22, no. 3, Art. no. 3, Jan. 2022, doi: 10.3390/s22030960.

[8] U. Gupta, A. Pranav, A. Kohli, S. Ghosh, and D. Singh, “The Contribution of Artificial Intelligence to Drug Discovery: Current Progress and Prospects for the Future,” in *Microbial Data Intelligence and Computational Techniques for Sustainable Computing*, A. Khamparia, B. Pandey, D. K. Pandey, and D. Gupta, Eds., Singapore: Springer Nature, 2024, pp. 1–23. doi: 10.1007/978-981-99-9621-6_1.

[9] C. Wu, K. Chau, and C. Fan, “Prediction of rainfall time series using modular artificial neural networks coupled with data-preprocessing techniques,” *J. Hydrol.*, vol. 389, pp. 146–167, Jul. 2010, doi: 10.1016/j.jhydrol.2010.05.040.

[10] K. Abhishek, A. Kumar, R. Ranjan, and S. Kumar, “A rainfall prediction model using artificial neural network,” presented at the Proceedings - 2012 IEEE Control and System Graduate Research Colloquium, ICSGRC 2012, Jul. 2012, pp. 82–87. doi: 10.1109/ICSGRC.2012.6287140.

[11] M. M. Ali, T. Mishra, J. Agrawal, A. Yadav, A. Pranav, and V. Ranjan, “An Efficient Approach for Detecting Neurological Tumors using Deep Learning,” in *2023 International Conference on Emerging Research in Computational Science (ICERCS)*, Dec. 2023, pp. 1–5. doi: 10.1109/ICERCS57948.2023.10434213.

[12] D. Dwivedi, J. H. Kelaiya, and G. Sharma, “Forecasting monthly rainfall using autoregressive integrated moving average model (ARIMA) and artificial neural network (ANN) model: A case study of Junagadh, Gujarat, India,” *J. Appl. Nat. Sci.*,

vol. 11, pp. 35–41, Feb. 2019, doi: 10.31018/jans.v11i1.1951.

[13] H. Abd elkader, M. Abdul Salam, and M. Mohamed, "Hybrid Machine Learning Model for Rainfall Forecasting," *J. Intell. Syst. Internet Things*, pp. 5–12, Jan. 2020, doi: 10.54216/JISIoT.010101.

[14] G. Tuysuzoglu, K. U. Birant, and D. Birant, "Rainfall Prediction Using an Ensemble Machine Learning Model Based on K-Stars," *Sustainability*, vol. 15, no. 7, Art. no. 7, Jan. 2023, doi: 10.3390/su15075889.

[15] A. Y. Barrera-Animas, L. O. Oyedele, M. Bilal, T. D. Akinosho, J. M. D. Delgado, and L. A. Akanbi, "Rainfall prediction: A comparative analysis of modern machine learning algorithms for time-series forecasting," *Mach. Learn. Appl.*, vol. 7, p. 100204, Mar. 2022, doi: 10.1016/j.mlwa.2021.100204.

[16] "Indian Sub-division Rainfall - 1901 to 2015." Accessed: Apr. 16, 2024. [Online]. Available: https://www.kaggle.com/datasets/kkhandekar/statewise-rainfall-1901-to-2015

Conflict of Interest Statement: *The authors declare that there is no conflict of interest regarding the publication of this paper.*

Cite this Article

Ayushman Pranav, Ankit Dubey, Umesh, Rajesh Kumar Modi. Towards Better Water Management- Predictive Models for Bihar's Rainfall. International Research Journal of Engineering & Applied Sciences (IRJEAS). 13(1), pp. 39-55, 2025. 10.55083/irjeas.2025.v13i01005.