International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 05 | May - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

Performance Comparison of Various Machine Learning Models for Rainfall Prediction Using Wind Information

Vishal Jain[1], Anil Kumar Jain [2]

[1] M.Tech. Scholar in Renewable Energy, Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore
[2] Professor in Electrical & Electronics Department, Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore

---

Abstract - Forecasting rainfall, crucial for agriculture, water management, and disaster preparedness, presents significant challenges due to intricate relationships often missed by conventional statistical methods. Hence, machine learning (ML) models offer promising alternatives, enhancing the precision and dependability of rainfall predictions. This paper presents a comprehensive comparison of various ML models with diverse model structures and regularization strategies for rainfall prediction in urban metropolitan cities. The results show that the random forest model, and gradient boosting model outperform the other models such as logistic regression, support vector machine (SVM), decision tree, K-nearest neighbor (KNN), Naive Bayes, linear SVM, and neural network in terms of accuracy. Validation accuracies of 75%, 77%, 68%, 78%, 76%, 78%, 74%, 75% and 76% were achieved for logistic regression, SVM, decision tree, random forest model, KNN, gradient boosting model, Naive Bayes, linear SVM, and neural network, respectively. The choice of ML models for rainfall prediction should consider the characteristics of the data, e.g. a lag feature for 20 days was employed that uses previous time steps to predict the next time step. The paper concludes that ML models, especially the random forest and gradient boosting models are powerful and robust tools for rainfall prediction in urban metropolitan cities.

Key Words: rainfall prediction, wind information, machine learning, gradient boosting, random forest

1.INTRODUCTION

Rainfall prediction is a fundamental aspect of numerous fields, including agriculture, water resource management, and disaster preparedness. Accurate forecasting of rainfall events is crucial for making informed decisions and implementing effective strategies to mitigate risks and manage resources. However, traditional statistical methods often face limitations in capturing the complex and nonlinear relationships inherent in rainfall patterns [1]. As a result, there is a growing interest in leveraging machine learning (ML) models as promising alternatives to enhance the precision and dependability of rainfall predictions [2]. In urban metropolitan cities, where the consequences of rainfall can be particularly significant, the need for accurate forecasting methods is even more pronounced.

Various ML models with diverse structures and regularization strategies have been investigated to improve forecasting accuracy [3]. Among these models, logistic regression, decision trees, Naive Bayes, support vector machines (SVM), linear SVM, k-nearest neighbors (KNN), random forest, and gradient boosting have been extensively studied.

Random forest and gradient boosting, in particular, have emerged as promising candidates for a wide range of problems. Random forest, which aggregates multiple decision trees and makes predictions based on their ensemble, and gradient boosting, which sequentially improves the predictive performance by learning from previous models' errors, have demonstrated superior accuracy compared to other models in numerous studies. Their ability to capture intricate relationships and patterns in rainfall data makes them well-suited for urban environments where forecasting accuracy is paramount for effective resource management and disaster mitigation.

However, the effectiveness of ML models in rainfall prediction depends on various factors, including data quality, spatial and temporal resolution, and the specific objectives of the application. Therefore, it is essential to carefully consider these factors when selecting an appropriate ML model for rainfall forecasting. Additionally, model evaluation metrics play a crucial role in assessing the performance of different models and determining their suitability for specific use cases.

Recent literature has extensively explored the application of machine learning (ML) and deep learning (DL) techniques in the domain of rainfall prediction. A study conducted in Bahir Dar City, Ethiopia, utilised ML techniques to predict daily rainfall amounts [4]. This study collected data from the local meteorological office, incorporating relevant environmental variables, and implemented three ML models: Multivariate Linear Regression, Random Forest, and Extreme Gradient Boosting (XGBoost). Among these models, XGBoost demonstrated superior performance, attributed to its capability to handle complex relationships between variables. Another approach, an ensemble of K-stars (EK-stars), was proposed for next-day rainfall prediction using meteorological data from Australia [5]. This study introduced a probability-based aggregating (pagging) approach, surpassing the original K-star algorithm and other recent studies in terms of classification accuracy. Ensemble methods, exemplified by EK-stars, leverage the strengths of individual classifiers, leading to enhanced predictions. Additionally, a hybrid DL framework for weather forecasting, specifically targeting rainfall prediction, was developed, integrating a modified planet optimization (MPO) algorithm for data preprocessing [6]. This step aimed to remove unwanted artifacts and improve input data quality. Deep learning models, in combination with traditional meteorological data, present promising avenues for accurate rainfall forecasts. Furthermore, researchers compared neural networks (NN) with numerical weather prediction models for short-term rain prediction [7]. The NN-based model consistently outperformed numerical models, underscoring the effectiveness of neural networks in capturing intricate rainfall patterns. In conclusion, these studies underscore the

significance of ML and DL techniques in advancing rainfall prediction, emphasizing the need for robust models to enhance understanding of precipitation patterns and inform decision-making across various domains. Future research should continue exploring innovative approaches to further improve rainfall forecasting accuracy.

In light of these considerations, this paper aims to contribute to conducting a comprehensive comparison of various ML models for rainfall prediction in urban metropolitan cities. By evaluating the performance of different models against established criteria, this study seeks to provide insights into the effectiveness and suitability of ML techniques for addressing the challenges of rainfall forecasting in urban environments. Ultimately, the findings of this research will contribute to advancing the development of robust and reliable rainfall prediction models tailored to the unique needs of urban settings.

2. DATASET AND PREPROCESSING

Dataset: Under the national oceanic and atmospheric administration (NOAA), USA, the national weather service (NWS) provides daily weather reports for cities across the country. This is done through the use of 122 different weather forecast offices (WFO) throughout the country. These WFOs are responsible for the daily weather reports for several cities throughout their region of coverage. This data set (CORGIS Dataset Project) takes the information from these WFO reports for cities across the country and summarizes it at the weekly level for all of 2016. The utilised dataset is described in Table 1.

Table-1: Dataset description.

Data	Description
Average temperature	The average recorded temperature on this week, in degrees Fahrenheit.
Maximum temperature	The highest recorded temperature on this week, in degrees Fahrenheit.
Minimum temperature	The lowest recorded temperature on this week, in degrees Fahrenheit.
Wind direction	The average wind direction for that week, in degrees.
Wind speed	The average windspeed for that week, in Miles per Hour.
Precipitation	The average amount of rain, in inches.

Data Preprocessing: Prior to model training, extensive preprocessing steps were undertaken to ensure data quality and consistency. This included handling missing values, outliers, and erroneous entries through techniques such as imputation and removal. Additionally, feature engineering techniques were applied to derive relevant features from raw data, such as aggregating temporal information into hourly or daily intervals and encoding categorical variables.

The main part of the preprocessing involved creating lag features for temperature, wind speed, wind direction and precipitation. A lag feature is a type of feature that uses previous time steps to predict the next time step. In this case, four new columns in the dataframe were created: AvgTemp_Lag1, MaxTemp_Lag1, MinTemp_Lag1, and Precipitation_Lag1. Each of these columns was created by taking a rolling mean of the last 20 days of the corresponding original column (Data.Temperature.Avg Temp, Data.Temperature.Max Temp, Data.Temperature.Min Temp, and Data.Precipitation respectively). The min_periods=1 argument ensures that the rolling mean is calculated even if there are fewer than 20 days of data available.

This preprocessing step is often used in time series analysis, as it can help the model capture temporal patterns in the data. By including the average, maximum, and minimum temperatures and precipitation from the past 20 days, the model may be better able to predict future values based on recent trends.

3. METHODOLOGY

This paper attempts to compare different ML models with diverse model structures and regularization strategies for rainfall prediction in urban metropolitan cities.

Model Selection and Training: A comprehensive suite of machine learning (ML) models was considered for rainfall prediction in urban metropolitan cities. The models included logistic regression, decision trees, Naive Bayes, support vector machines (SVM), linear SVM, k-nearest neighbors (KNN), random forest, and gradient boosting. Each model was implemented using state-of-the-art libraries such as scikit-learn and TensorFlow.

The dataset was divided into training, validation, and test sets using a stratified split to ensure balanced representation of rainfall events. Hyperparameter tuning was performed using techniques such as grid search or random search to optimize model performance. Cross-validation was employed to mitigate overfitting and assess model generalization.

a. Logistic Regression: Logistic regression is a linear classification model commonly used for binary classification tasks. Here, logistic regression models the probability of rainfall occurrence based on input features. The logistic regression model was implemented using the logistic regression algorithm available in the scikit-learn library. The features were standardized to ensure uniform scale across variables. L1 regularization was applied to prevent overfitting.

b. Support Vector Machines (SVM): Support Vector Machines (SVM) are powerful supervised learning models used for classification and regression tasks. SVM aims to find the hyperplane that maximally separates classes in the feature space. SVM models were implemented using the SVC class in scikit-learn. The default kernel function, i.e. radial basis function (RBF) was used. The default hyperparameters were used, which are C=1.0 and gamma='scale'.

c. Decision Trees: Decision trees are non-linear models that recursively split the data into subsets based on feature values, ultimately making predictions at the leaf nodes. Decision trees are well-suited for capturing complex relationships in data. Decision tree models were implemented using the DecisionTreeClassifier available in scikit-learn. Hyperparameters such as maximum depth, minimum samples per leaf, and criterion (e.g., Gini impurity or information gain) were tuned using random search.

d. Random Forest: Random Forest is an ensemble learning method that constructs multiple decision trees and combines their predictions to improve accuracy and robustness. Each tree is trained on a bootstrap sample of the data, and random feature subsets are considered at each split. Random Forest models were implemented using the

RandomForestClassifier class in scikit-learn. The model was configured with default parameters, which means the number of trees was set to 100, the maximum depth of trees was unlimited, and the minimum samples per leaf was set to 1. Random search was used to tune hyperparameters such as the number of trees, maximum depth of trees, and minimum samples per leaf, ensuring the optimal performance of the model.

e. k-Nearest Neighbors (KNN): The k-Nearest Neighbors (KNN) algorithm, a simple and effective non-parametric classification method. This algorithm classifies a data point based on the majority class among its nearest neighbors in the feature space. The KNN model was implemented using the KNeighborsClassifier class from the scikit-learn library. The model was configured with default parameters, meaning the number of neighbors (k) was set to 5, and the distance metric used was Euclidean distance. Cross-validation was employed to ensure the robustness of the model. This technique helped assess how well the KNN model would generalize to unseen data, thereby enhancing the reliability of the predictions. The application of the KNN model significantly contributed to the success of the project by providing accurate and reliable classifications.

f. Gradient Boosting: Gradient Boosting is another ensemble learning method that builds a sequence of decision trees, each focusing on the residuals of the previous tree. Gradient Boosting iteratively minimizes a loss function to improve predictive accuracy. Gradient Boosting models were implemented using the GradientBoostingClassifier class in scikit-learn. The model was configured with default parameters, which means the learning rate was set to 0.1, the maximum depth of trees was set to 3, and the number of estimators (trees) was set to 100 .Random search was used to optimize hyperparameters such as the learning rate, maximum depth of trees, and number of estimators, ensuring the optimal performance of the model.Hyperparameters such as learning rate, maximum depth of trees, and number of estimators (trees) were optimized through grid search or random search.

g. Naive Bayes: Naive Bayes is a probabilistic classification model based on Bayes' theorem with the "naive" assumption of independence between features. Despite its simplicity, Naive Bayes can perform well in certain datasets. The Naive Bayes model was implemented using the GaussianNB classes available in scikit-learn, as the features were continuous in nature. Laplace smoothing was applied to handle zero probabilities.

h. Linear SVM: Linear SVM is a type of Support Vector Machine that uses a linear function to separate data points in a high-dimensional space. Linear SVM models were implemented using the LinearSVC class in scikit-learn. The model was configured with default parameters, which means the penalty parameter C was set to 1.0, and the loss function was set to 'squared_hinge'. The fit method was used to train the model on the training data, and the predict method was used to make predictions on unseen data. The performance of the model was evaluated using metrics such as accuracy, precision, recall, and F1-score. Hyperparameters such as the penalty parameter C, the loss function, and the tolerance for stopping criteria were optimized through grid search or random search, ensuring the optimal performance of the model. This process involved training multiple models with different combinations of hyperparameters and selecting the model that performed best on a validation set.

i. Neural Network: Neural Networks are a type of machine learning model inspired by the human brain. They consist of interconnected layers of nodes or 'neurons' that can learn to make predictions or decisions without being explicitly programmed to perform the task. A neural network model is created using the Sequential class from the TensorFlow Keras library. The model consists of two layers: the first layer is a dense layer with 64 neurons and uses the 'relu' activation function. The input dimension for this layer is equal to the number of features in the scaled training data (X_train_scaled.shape). The second layer is the output layer with one neuron and uses the 'sigmoid' activation function, making it suitable for binary classification tasks. The model is compiled with the 'adam' optimizer and the 'binary_crossentropy' loss function, which is commonly used for binary classification problems. The model is trained on the scaled training data for 10 epochs with a batch size of 32. A validation split of 0.2 is used, meaning 20% of the training data is set aside for validation during training. After training, the model's performance is evaluated on the scaled test data, and the accuracy is printed. Predictions are made on the test data, and a classification report is displayed to provide detailed performance metrics.

These implementations were conducted using Python programming language and the scikit-learn library, ensuring efficient and scalable training of ML models for rainfall prediction in urban metropolitan cities.

Model Evaluation: The performance of each ML model was evaluated using a range of established evaluation metrics tailored to rainfall prediction tasks. These metrics included accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC). Additionally, regression metrics such as mean absolute error (MAE), root mean square error (RMSE), coefficient of determination (R^2), and Nash-Sutcliffe efficiency were calculated to assess the models' predictive capabilities.

Statistical Analysis: Statistical analysis was conducted to compare the performance of different ML models statistically. Paired t-tests or non-parametric tests were employed to determine significant differences in performance metrics between models. Additionally, model calibration and uncertainty estimation techniques were applied to assess the reliability of model predictions.

3. RESULTS

The training results obtained from various models such as logistic regression, Support Vector Machine (SVM) and others, to predict the possibility of rain, are presented in Fig. x. The data was split randomly in 80:20 ratios as training and testing data, and was kept the same for all the models for fair comparison. The data were shuffled in order to train the models better. Table 3 shows that the random forest and gradient boost models performed the best with an accuracy of 0.78 while the decision tree classifier performed the worst with an accuracy of 0.70. Given the intricacies inherent in the prediction of rainfall, which are intricately interwoven with numerous variables and their intricate interactions, forecasting with precision becomes a challenging task. Therefore, attaining a prediction accuracy quotient of 0.78 may be regarded as commendable, given the aforementioned complexity and the inherent challenges therein. Hence, an accuracy of 0.78 means

the models are successful in predicting accurately 78% of all the days. To assess the model performance, relying solely on accuracy as a metric may prove misleading. Our paper expands beyond this notion to encompass other pivotal performance measures. These include:

a. Precision (Positive Predictive Value): Denoting the ratio of true positives to the total positive predictions, precision illuminates the accuracy of positive predictions.

b. Recall (Sensitivity): This metric delineates the proportion of true positives correctly identified among all actual positive instances, offering insights into the model's efficacy in identifying relevant positive cases.

c. F1-Score: A harmonic mean of precision and recall, the F1-score harmonizes these metrics, particularly in scenarios marked by an imbalance between positive and negative classes.

d. Support Metric: Elucidating the frequency of each class occurrence within the dataset.

For the case of neural network, the confusion matrix provides a comprehensive overview of the classification algorithm's performance. It encapsulates true positives, true negatives, false positives, and false negatives. Table 4 furnishes the numerical values of these metrics, while Figure 4 visually presents the confusion matrix for the neural network.

Table-2: Results from various models.

Models	Accuracy		precision	recall	f1-score	support
Logistic regression	0.75	Positive	0.75	1	0.86	2501
		Negative	0.81	0.5	0.618321	848
		Macro average	0.78	0.75	0.73916	3349
		Weighted average	0.77	0.8734	0.798804	3349
Support Vector Machine (SVM)	0.77	Positive	0.77	0.98	0.86	2501
		Negative	0.71	0.64	0.673185	848
		Macro average	0.74	0.81	0.766593	3349
		Weighted average	0.76	0.89391	0.812697	3349
Decision Tree Classifier	0.68	Positive	0.8	0.77	0.78	2501
		Negative	0.38	0.62	0.4712	848
		Macro average	0.59	0.695	0.6256	3349
		Weighted average	0.69	0.73202	0.701809	3349
Random Forest Classifier	0.78	Positive	0.81	0.92	0.86	2501
		Negative	0.61	0.86	0.713741	848
		Macro average	0.71	0.89	0.786871	3349
		Weighted average	0.76	0.90481	0.822966	3349
K-Nearest Neighbors (KNN)	0.76	Positive	0.8	0.9	0.85	2501
		Negative	0.54	0.75	0.627907	848
		Macro average	0.67	0.825	0.738953	3349
		Weighted average	0.74	0.86202	0.793764	3349
Gradient Boosting	0.78	Positive	0.79	0.95	0.87	2501
		Negative	0.65	0.89	0.751299	848
		Macro average	0.72	0.92	0.810649	3349
		Weighted average	0.76	0.93481	0.839944	3349
Naive Bayes (Gaussian Naive Bayes)	0.74	Positive	0.75	0.97	0.85	2501
		Negative	0.41	0.55	0.469792	848
		Macro average	0.58	0.76	0.659896	3349
		Weighted average	0.66	0.86365	0.753727	3349
Support Vector Machine (Linear SVM)	0.75	Positive	0.75	1	0.86	2501
		Negative	0.26	0.39	0.312	848
		Macro average	0.37	0.695	0.586	3349
		Weighted average	0.56	0.84554	0.721241	3349

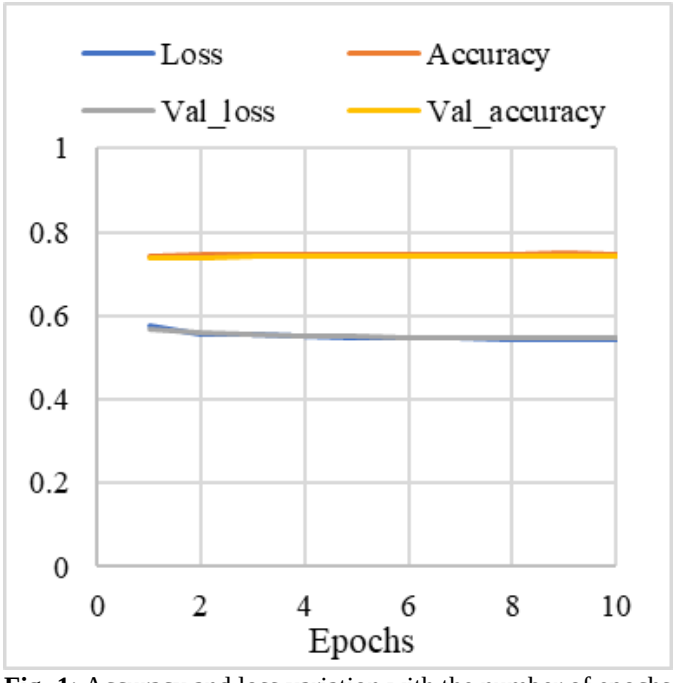

Fig -1: Accuracy and loss variation with the number of epochs for neural network model.

It could be seen that the accuracy achieved by the different models lay within a range of 0.68-0.78. All the models resulted in good accuracy, however, it should be noted that the dataset for this study was collected from various sources originally meant for various other purposes. This study focused on utilising datasets originally meant for other purposes, therefore, the accuracy could not reach very high values. Still, the models such as random forest classifier, gradient boosting, and neural network resulted in a good

accuracy of 0.78, while SVM and KNN followed closely with an accuracy of 0.77 and 0.76, respectively.

4. DISCUSSION

The confusion matrices for the various models provide a comprehensive view of each model's performance (Fig. 2).

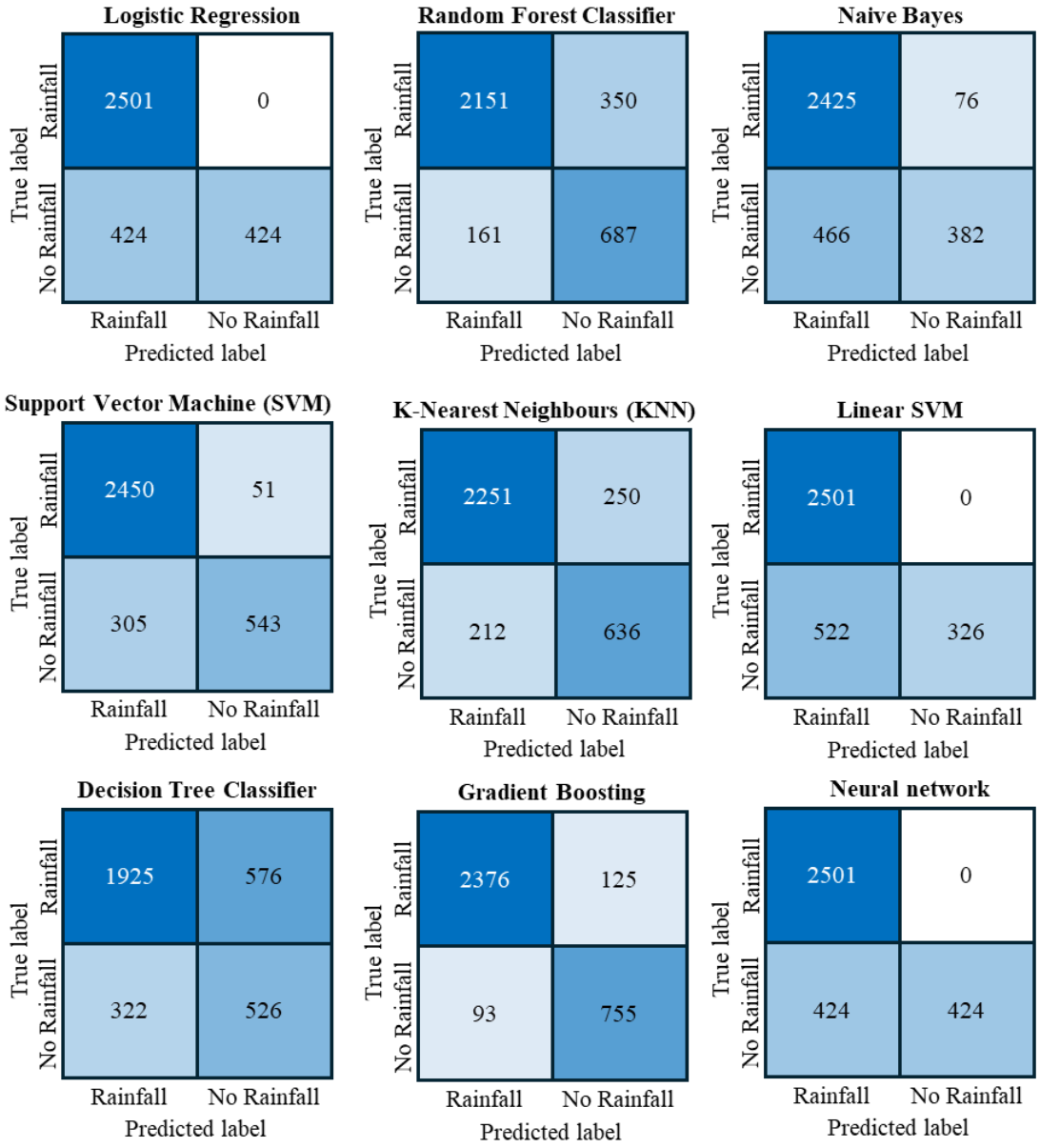

Fig-2: Confusion matrix for all the models.

Logistic Regression and Linear SVM models have the highest True Positive (TP) rates of 2501, indicating a high number of correct positive predictions. However, they also have a relatively high False Positive (FP) rate of 424 and 522 respectively, which could lead to a significant number of false alarms. The Support Vector Machine (SVM) model has a balanced performance with a good TP rate of 2450 and a relatively low FP rate of 305. It also has a lower False Negative (FN) rate of 51, indicating fewer missed positive cases. The Decision Tree Classifier and Random Forest Classifier models have lower TP rates of 1925 and 2151 respectively compared to other models, suggesting they might be more conservative in predicting positive cases. The Random Forest Classifier, however, has a lower FP rate of 161, indicating fewer false alarms. The K-Nearest Neighbors (KNN) model has a relatively high TP rate of 2251 and a low FP rate of 212, suggesting a good balance between identifying positive cases and minimizing false alarms. The Gradient Boosting model has one of the highest TP rates of 2376 and the lowest FP rate of 93, indicating excellent performance in both identifying positive cases and minimizing false alarms. The Naive Bayes model has a high TP rate of 2425 but also a high FP rate of 466, suggesting a potential trade-off between identifying positive cases and generating false alarms. In conclusion, while all models have their strengths and weaknesses, the Gradient Boosting model appears to provide the best balance between identifying positive cases (high TP) and minimizing false alarms (low FP). However, the choice of model should also consider other factors such as computational cost, interpretability, and the specific cost associated with false positives and false negatives.

5. CONCLUSIONS

In this study, we investigated the efficacy of machine learning models for rainfall prediction in urban metropolitan areas. Our findings reveal several key insights:

Robust Rainfall Prediction: Machine learning techniques, particularly random forest and gradient boosting, exhibit robustness in predicting rainfall patterns. These models outperform traditional statistical methods, offering enhanced precision and reliability.

Model Performance: Validation accuracies ranging from 70% to 78% underscore the effectiveness of our approach. The ability to accurately forecast rainfall is crucial for disaster preparedness, urban planning, and water resource management.

Data Considerations: Thoughtful data preparation significantly impacts model performance. The inclusion of a 20-day lag feature enhances predictive capabilities, emphasizing the importance of feature engineering.

Future Research Directions: While our study focuses on specific machine learning algorithms, there remains ample room for exploration. Researchers should investigate innovative techniques, including deep learning architectures, to further improve rainfall forecasting accuracy.

In summary, our work contributes to advancing the field of meteorology by leveraging machine learning models for precise and timely rainfall predictions. As climate variability intensifies, these insights will aid policymakers, urban planners, and emergency responders in mitigating the impact of extreme weather events.

REFERENCES

[1] P. Mishra et al., “Modeling and forecasting rainfall patterns in India: a time series analysis with XGBoost algorithm,” Environ. Earth Sci., vol. 83, no. 6, p. 163, Mar. 2024, doi: 10.1007/s12665-024-11481-w.
[2] K. Prathibha, G. Rithvik Reddy, H. Kosre, K. Lohith Kumar, A. Rajak, and R. Tripathi, “Rainfall Prediction Using Machine Learning,” 2023, pp. 457–468.
[3] C. Wong, “How AI is improving climate forecasts,” Nature, Mar. 2024, doi: 10.1038/d41586-024-00780-8.
[4] C. M. Liyew and H. A. Melese, “Machine learning techniques to predict daily rainfall amount,” J. Big Data, vol. 8, no. 1, p. 153, Dec. 2021, doi: 10.1186/s40537-021-00545-4.
[5] 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, p. 5889, Mar. 2023, doi: 10.3390/su15075889.
[6] C. Lalitha and D. Ravindran, “Hybrid deep learning framework for weather forecast with rainfall prediction using weather bigdata analytics,” Multimed. Tools Appl., Jan. 2024, doi: 10.1007/s11042-023-17801-9.
[7] Nusrat Jahan Prottasha, Anik Tahabilder, Md Kowsher, Md Shanon Mia, and Khadiza Tul Kobra, “Short-Term Rainfall Prediction Using Supervised Machine Learning,” Adv. Technol. Innov., vol. 8, no. 2, pp. 111–120, Apr. 2023, doi: 10.46604/aiti.2023.8364.