Discover Internet of Things

Research

A pragmatic ensemble learning approach for rainfall prediction

Soumili Ghosh[1] · Mahendra Kumar Gourisaria[1] · Biswajit Sahoo[1] · Himansu Das[1]

Received: 15 June 2023 / Accepted: 23 September 2023
Published online: 09 October 2023

Abstract
Heavy rainfall and precipitation play a massive role in shaping the socio-agricultural landscape of a country. Being one of the key indicators of climate change, natural disasters, and of the general topology of a region, rainfall prediction is a gift of estimation that can be used for multiple beneficial causes. Machine learning has an impressive repertoire in aiding prediction and estimation of rainfall. This paper aims to find the effect of ensemble learning, a subset of machine learning, on a rainfall prediction dataset, to increase the predictability of the models used. The classification models used in this paper were tested once individually, and then with applied ensemble techniques like bagging and boosting, on a rainfall dataset based in Australia. The objective of this paper is to demonstrate a reduction in bias and variance via ensemble learning techniques while also analyzing the increase or decrease in the aforementioned metrics. The study shows an overall reduction in bias by an average of 6% using boosting, and an average reduction in variance by 13.6%. Model performance was observed to become more generalized by lowering the false negative rate by an average of more than 20%. The techniques explored in this paper can be further utilized to improve model performance even further via hyper-parameter tuning.

Keywords Rainfall prediction · Ensemble learning · Bagging · Boosting · Bias-variance

1 Introduction

Rainfall, and its levels of intensity, and frequency, are important indicators of climate change, pertaining to a specific region. It is also the key determinant of spatial and temporal availability of water in a region. On superficial levels, irregular rainfall, and its variations can bring about risks of floods [1], increasing temperatures, and cyclones [2]. Studies show that rainfall, as a component of climate change even affects seemingly unrelated industries like the fishing industry. Results of regional studies even suggest that the affected yield can vary so much as 80%, in accordance to the rainfall variation or its lack thereof [3]. Extreme daily rainfall has also been studied as an indicator of slower global economy [4, 5], health properties such as height, and even cognitive development consequences [6]. A study conducted by S. K. Pariyar et al. suggested that tropical cyclones, cause the highest increase in the probability of extreme rainfall, followed by the Madden-Julein Oscillation (MJO) [7, 8]. The pressure, relative humidity, sunshine, wind speeds, evaporation, and sunshine rates, are also determining factors of rainfall intensity [9]. Rainfall prediction, is, therefore, a crucial activity for boosting agricultural productivity, socioeconomic status of a region, and environmental welfare, in general.

✉ Himansu Das, das.himansu2007@gmail.com; Soumili Ghosh, soumilighosh737@gmail.com; Mahendra Kumar Gourisaria, mkgourisaria2010@gmail.com; Biswajit Sahoo, bsahoofcs@kiit.ac.in | [1]School of Computer Engineering, KIIT Deemed to Be University, Bhubaneswar, Odisha 751024, India.

Methods of rainfall prediction, earlier included Big Data Analytics [10], and data mining techniques [11]. With steady, but hefty advances, Machine Learning has found applications in this sector as well. With the exploitation of both regression and classification models, in various fields such as healthcare [12, 13], agriculture [14], infrastructure [15] etc., it has proved to be a powerful ally for predictive tasks. ML has also become very prevalent in both prediction and estimation of rainfall variations. Classical models like Logistic Regression, Decision Trees, and K-NearestNeighbors, have been explored as classifiers, to predict the occurrences of rainfall. Deep learning, and its subsets, have also been implemented for detection and approximation purposes.

However, all these models have one common drawback when trained on highly imbalanced datasets like the one used in this experiment. Rainfall prediction datasets tend to be imbalanced in general, due to the general tendency of the weather pattern repeating itself according to prior conditions. When models are trained on the same imbalanced data without any consideration given to the imbalance, the prediction rate, while accurate, might turn out to be inefficient in times of occurrences of positive instances. High bias and variance are commonly observed problems with imbalanced datasets. Inconsideration of bias and variance during the prediction making process may cause the model to overfit or underfit the data, since the model tends to favor the majority class. Ensemble learning techniques are a potential solution to this problem. Through multiple estimations from multiple models of the same type, which have been trained on different samples of the training set, variation is introduced into both the training process, and the prediction making. Error correction and parameter updation is also one more technique used in boosting, which helps remove the bias that a model tends to have towards the majority class.

With this experiment, we address the issue of high bias and variance in rainfall prediction with classical ML models, by introducing bagging and boosting into the prediction-making process. The results obtained by the classical models and the ensemble learning models were compared to demonstrate the increase/decrease in bias and variance. In this experiment, the models used have been passed through two Ensemble learning techniques, Bagging and Boosting. The final results suggest an overall improvement in performance using the ensemble techniques.

This paper has been divided into four subsequent sections, following the introduction section. This section is followed by a report on the related works associated with the same objective. This section is followed by the results and analysis sections which is further divided into parts, for reader convenience. The first subpart of the results section details the dataset details, acquisition and preprocessing techniques. The subpart following that details the results obtained, and allows the readers to take a look at the comparative analysis between the metrics with and without the ensemble learning application. Finally, the paper is wrapped up with the conclusion, acknowledgment, and references sections.

2 Related work

D. Pirone et al. proposed a probabilisticc machine learning model for predicting rainfall based on cumulative data for intervals of 10 min. The short leads can range from 30 min to 6 h. Cumulative rainfall data is fed as input to feed forward neural networks for predictive purposes. In line with the team's expectations, the accuracy of predictions decreased with longer time leads according to physical models. Observed results suggested that use of both temporal and spatial information aids the model to make predictions rapidly, and replicably [16]. C. Z. Basha et al. explored the working of two deep learning techniques, on rainfall prediction. The two used techniques were Multilayer Perceptrons and AutoEncoders. The model received best results, in the 26th iteration, in epoch 20, with the least Mean Square Error (MSE) loss [17]. S. Fahad et al. presented a deep weather forecasting model based on Gated Recurrent Unit (GRU) which was optimized, for prediction of rainfall in Pakistan, from 1991 to 2020. The climate associated features were intially extracted, and then optimized by outlier elimination. Normalization techniques were used to bring the input numbers between a specific range, without losing any information. The presented model was shown to acquire high accuracies whilst minimizing Normalized Mean Absolute Error(NMAE) and Root Mean Squred Error(RMSE), as compared to state-of-the-art models [18]. A machine learning fusion technique was developed by A. U. Rahman et al. in 2022. This proposed framework fused four widely used classification techniques, namely Decision Trees, Naïve Bayes classifier, K-Nearest Neighbors classifier, and Support Vector machines. Integration of Fuzzy Logic, for effective prediction of rainfall, led to the proposed framework outperforming other relevant models. The fuzzy logic worked on the basis of majority voting. If any of the three models used in the fusion, predicted the possibility of rainfall as positive/negative, then the model predicted the possibility as positive/negative.The proposed model obtained excellent results with a high accuracy of 99% [19]. A study comparing machine learning and deep learning based time-series analysis models was presented by A. Y. Barerra-Animas et al. in 2022. The team used LSTM, stacked LSTM, Bidirectional LSTM networks, XGBoost, an ensemble of Gradient Boosters, Linear Support Vector Regression model, and an Extra-trees regressor for their study of forecasting hourly volumes of rainfall. Their stacked LSTM models with two hidden layers, and Bidirection LSTM models, performed best, based on the metrics used for comparison [20]. In 2023, T. Manna et al. combined Rough Spaces in fuzzy sets with Deep learning methods for time-series forecasting of rainfall levels in Southern India. The rough spaces of the model aided with handling of uncertainty in prediction, and the deep learning methods aided classification. The presented model was evaluated and compared to existing DL and ML models, and proved to acquire high accuracy with minimization of error rates [21].

A comparative analysis by A. Y. Barrera-Animas et al. showed that their stacked LSTM(Long Short Term Memory) model consisting of two hidden layers, and bidirectional LSTM model worked commendable well for predicting rainfall, as compared to LSTM, XGBoost, and an ensemble of Gradient Boosting Regressor and Support Vector regressor. The input data used conatined climatic conditions of five major cities in the UK from 2000 to 2020. Evaluation metrics of loss, RMSE, Mean Absolute Error(MAE), and Root Mean Squared Logarithmic Squared Error(RMLSE), were used to rank the models on their predictability. Their study suggested promising accuracy from LSTM models with fewer hidden layers [20]. In 2020, W. Suparta et al. explored the application of Adaptive NeuroFuzzy Infer-ence Systems (ANFIS), on rainfall prediction. The presented approach incorporated the neural networks with linguistic representations of fuzzy techniques. The time series analysis of the model garnered an accuracy of 80%, on a dataset containing data about the rain patterns in South Tangern City, Banten, for 6 years, on a month-ly basis [22]. LSTM and T-LSTM(Transitional Long Short Term Memory) models were also observed to display high predictive accuracies in a proposed model by K. Venkatachalam et al. in 2023. The mentioned model was evaluated and compared via RMSE and MAE, on HHWD and Jena climatic datasets. The study revealed that the T-LSTM model displayed higher accuracy of 98.2%, as compared to other existing deep learning models and provided a strong solution to use of hydrological features [23]. In 2017 T. Kashiwao et al. developed and tested an ANN based local rainfall prediction system. They used radial basis function networks (RBFN) with a least squares method (LSM) and multi-layer perceptrons (MLP) with a hybrid algorithm composed of back-propagation (BP) and random optimisation (RO) methods as neural network (NN) models for the system, before comparing the prediction perfor-mance of the two models. Their study suggested the MLP model outperformed the RBFN model [24].

Most of the work done on the task so far, have explored applications of deep learning techniques, especially CNNs [25], and ANNs [26], alongside MLPs. A relative few utilize the classical classification models, and a large chunk of the work focuses on forecasting the rainfall estimate, using time series analysis. Even fewer studies combine the usage of predefined techniques, along with deep learning techniques [27], and other hybrid models [28] for the prediction of rainfall. A lot of the research done on rainfall prediction so far has addressed the issue with complex as well as simpler architectures. However, none of the mentioned works look at the problem if rainfall prediction with consideration of bias and variance in the dataset. The inherent imbalance in the datasets seem to have been fixed mostly using SMOTE, or sampling methods. While this works in the favour of the classifier, it does quite less to generalize the prediction, since SMOTE provides synthetic samples closer to the minority samples, rather than distribute the synthetic samples evenly. This makes little difference to prediction powers of a weak or lazy classifier. Hence, in this paper, we address the issue of bias and variance reduction specifically, using ensemble learning in predicting rainfall, to provide a fairer prediction without favoring either of the classes. We also analyze the bias-variance trade-off for the techniques, and observe how they affect the model's overall performance.

3 Methods and materials

Our experiment was aimed at comparing the performance of several machine learning techniques on a rainfall prediction dataset, before and after integrating them with ensemble learning. Ensemble learning is a field of study, where predictions from several weak learning models are combined to get the prediction from a collectively strong model. Two ensemble learning techniques were explored here, Bagging and Boosting. Bagging was used with the objective of reducing variance, while boosting was utilized to reduce bias.

3.1 Bagging

This is an abbreviation for the term 'Bootstrap Aggregation', and involves taking multiple, suppose n copies of the same model, and making predictions separately for the same data point, based on a randomized subset of features. All these predictions are then used to find the predicted label with a higher frequency, or majority voting [29]. That label is then assigned to that specific data point. The training size of the dataset is the same as the original dataset, since the training set is randomly generated with replacement, and hence, the same data point may be drawn multiple times. All the base classifiers are trained parallelly. The voting method, reduces overfitting of data, but usually increases the bias, which is again countered by a decrease in variance. An overall representation of the bagging mechanism is shown below in Fig. 1, where T_i represents each randomly generated training set, CL_i represents each base classifier, P_i represents each prediction made, and n is the number of base classifiers specified.

3.2 Boosting

Boosting is another ensemble learning technique which takes multiple weak learners, and forms a strong learner, by correcting the consecutive residual errors of the models. An initial model is built based on a training set, and the second model tries to correct the errors made by the first model, and so on. This error correction is done by changing the weights assigned to each data point for prediction. For this experiment, the Adaboost classifier was used by replacing the base learners with the models considered. It assumes a sequence of weak learners and combines their results to form a string learner. The updation of weights for the model is made according to the error of the previous weak learner, via formula 1, where w_i^t represents the weight of sample point i at iteration t, α^t represents the weight of the current classifier, and $h^t\left(x_i\right)$ is the output of the current learner.

$$w_i^t = w_i^{t-1} * (\exp(-\alpha^t * y_i * h^t\left(x_i\right)) \tag{1}$$

The weight α^t is calculated via the formula shown in Eq. 2, where ε^t is the weighted error of the weak learner.

$$\alpha^t = 0.5 * \ln(\frac{1-\varepsilon^t}{\varepsilon^t}) \tag{2}$$

The mathematical depiction of the final learner is shown in Eq. 3, with weights being proportional to the alpha weights. The sign function being used return + 1 if it receives a positive argument, − 1 if the received argument is negative, and 0 otherwise.

$$H(a) = sign(\sum_t^M \alpha^t * h_t(a)) \tag{3}$$

All the models and their evaluation followed the pattern shown in Fig. 2.

The methodology followed for this experiment required initial preprocessing of the dataset to fill in null values with estimates. The next step involved dividing the preprocessed dataset into training and testing datasets. Due to the observed imbalance in the training dataset, SMOTE was applied to a proportion of 50:50 for both positive and negative instances. After this, the models were trained individually on the training set and the results were recorded. Bagging

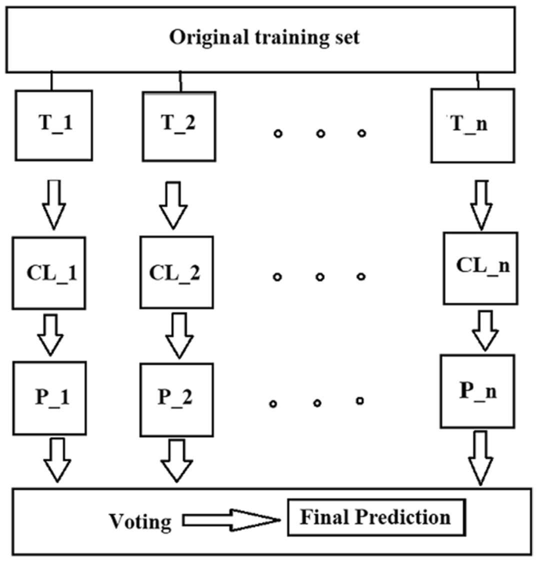

Fig. 1 Bagging Mechanism

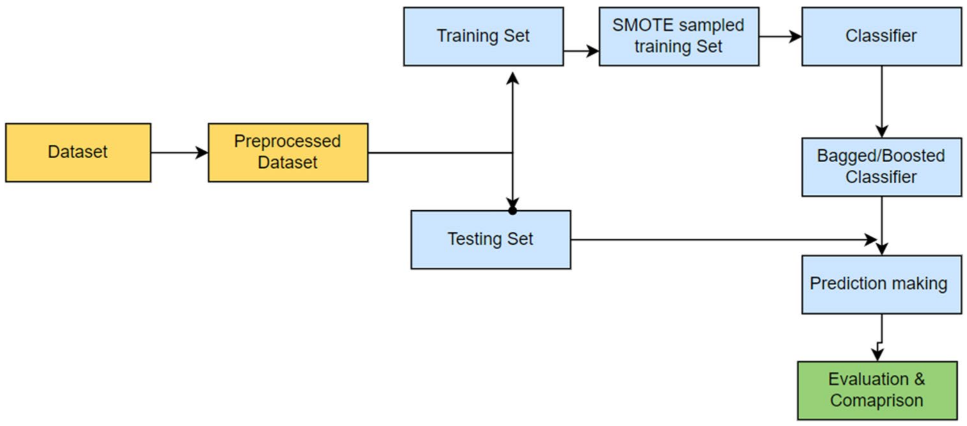

Fig. 2 Methodology of experiment

and boosting were then carried out separately for each of the models and the results were recorded for comparison with the initial scores. The bias and variance, and the deviation in both were given special attention during analysis. 7 models were tested and compared in this experiment. Namely, Logistic Regression, K-Nearest neighbors, DecisionTree Classifier, RandomForest Classifier, XGBoost Classifier, CatBoost Classifier, and LightBGM classifier. Brief explanations of the models used follow from Sects. "3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9".

3.3 Logistic regression

Principal Logistic regression is a supervised model, used for both classification and regression purposes, and is especially prevalent for binary classification tasks [30]. It works on the mathematical basis of the sigmoid function, whose definition is shown in Eq. 4.

$$f(a) = \frac{1}{1 + e^{-a}} \tag{4}$$

This model classifies binary labels, such as in this experiment, with calculation of probability, taking some independent variables into account. The formula in Eq. 1 outputs the final label as 1, if the probability is greater than 0.5, and 0 if the probability is lesser than 0.5. This probability is calculated by fitting the function into likelihood. The loss function/cost function used in this model conventionally is the log loss, which is again derived from maximum likelihood estimation. The mathematical representation of log loss is given below in Eq. 5.

$$Loss = \frac{1}{n} \sum -(y_i * \log\left(Y_i\right) + \left(1 - y_i\right) * \log\left(1 - Y_i\right)) \tag{5}$$

The reason behind Logistic regression models not using the Mean Squared Error loss, used in Linear regression is that, the function used in the former is not linear. That is why, the gradient descent curve for the Logistic Regression model will produce multiple local minima, if MSE is used as its cost function. Hence, the use of log loss.

3.4 K-nearest neighbors

The K-Nearest neighbors is another supervised statistical model, used for both classification and regressional tasks. It is simple but effective [31]. This type of model primarily relies on the distance metric to make predictions, and assumes that inputs have similar outputs, for classification purpose. In this paper, the model has been used for binary classification, and hence it looks around for the k-nearest data points, and assigns the label, which has greater frequency amongst those neighbors. The three most commonly used distance metrics are Euclidean distance, Minkowski distance, and Manhattan distance. The mathematical equations for Euclidean metric is shown below, in Eq. 6, as it is the metric being used for the KNN model used in this experiment. The summation of the terms under square root, are done from $i = 1$ to $i = n$, where n is the number of data points in the dataset. This model does not have any specific training error.

$$distance(x,y) = \sqrt{\sum (x_i - y_i)^2} \tag{6}$$

3.5 DecisionTree classifier

The decision tree classifier is yet another supervised model, fit for both regression and classification [32] tasks. It implements a tree structure where each internal node, is a question about a determining feature, and every leaf node is an outcome. The nodes are therefore divided into decision making nodes, and resulting nodes. A general decision tree structure is shown in Fig. 3.

The first step to classification using decision tree is to find the entropy of the target, which is the measurement of disorder of the data point via formula shown in Eq. 7, where q_i is the fraction of that class, in the dataset.

$$Entropy(Sample) = \sum -q_i \log_2 q_i \tag{7}$$

If all classes in the dataset are equally distributed, then the entropy amounts to 1. After entropy calculation of each class in the dataset, the difference between entropy before and after the dataset is split on basis of classes. This difference is called the information gain, and thereafter, the feature with the highest information gain is made the root node, on basis of which further splitting is done.

3.6 RandomForest classifier

RandomForest classifier is another ensemble learning model, which integrates the technique of bagging. It is a powerful model fit for regression and classification [33]. Several decision trees are built upon randomized subsets of the training dataset, and are assigned to make predictions parallel, but separately. After the initial prediction process is completed, the predictions are voted upon, and the prediction with the higher frequency of occurrence is determined as the final prediction. This model is particularly useful in cases of larger datasets, where it maintains higher values of accuracy even with larger proportions of missing data.

3.7 XGBoost classifier

Extreme Gradient Boosting or XGBoost is an extremely prevalent machine learning algorithm [34], which creates an ensemble of models with weaker learning abilities, namely decision (CART) trees, and uses the boosting technique to compile a strong learner. In use cases with several features, and sample data points, it comes especially handy, due to its

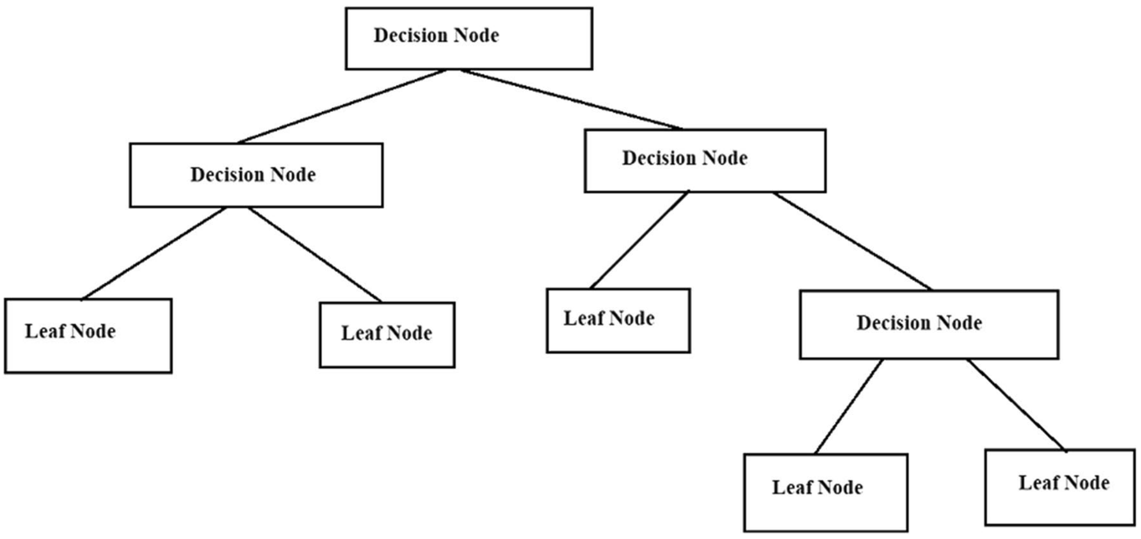

Fig. 3 Typical Decision Tree Structure

ability to handle high-dimensionality and non-linearity with respect to correlations. This approach constructs decision trees on the model iteratively, with each new tree given the responsibility of fixing the errors made by the preceding tree. For our experiment, the model was used for Binary Classification purposes, and hence the log loss objective function was used as the loss function to be minimised, as shown previously in Eq. 5. This model calculates the gradient of the loss function with respect to the previous trees' predicted values, in each iteration. It then fits a completely new tree to the negative gradient of the loss function. This iterative process is continued till a cutoff point is reached, where the model ceases to improve its performance metrics.

3.8 CatBoost classifier

CatBoost classifier is a composite form for Categorical Boosting classifier. As suggested by its name, this type of model excels at working with categorical data. It provides state-of-art results without requiring extensive training, as with other models, and can be used for powerful classification and regression tasks [35] such as fraud detection [36]. This model also works on the foundation of the Gradient Boosting algorithm, which is a powerful boosting algorithm meant for optimization by minimization of loss functions. Furthermore, the CatBoost classifiercan be used without extensive preprocessing of categorical data, due to its ability to handle and process categorical data internally while training. By default, the model uses one-hot encoding for this task. It also inhibits several distinct properties like integrated cross validation implementation. CatBoost classifiers perform exceptionally well in classification tasks owing to their automated missing information management, regularization utilization, and implementation of feature selection.

3.9 LightGBM classifier

A framework implementing gradient boosting which uses decision trees to efficiently handle massive datasets, via lesser computation. It introduces two unique techniques, namely Gradient-Based One Side Sampling (GOSS) and Exclusive feature bundling (EFB). The usage of these two algorithms helps overcome the shortcomings of histogram-based decision tree frameworks, as seen in other Gradient-Boosted Decision Tree frameworks. It uses GOSS to select points to split on, during tree construction, which supports quicker and efficient model training. To add to its efficiency, added properties like categorical data handling, data parallelism, and GPU support. Overall, LightBGM has proved to be a strong, and adaptive technique for handling larger datasets with higher dimensionalities.

4 Results and analysis

This section has been divided into two smaller parts for detailing the dataset features, its analysis and preprocessing.

4.1 *Dataset*

This section is further divided into two parts, (a) Dataset details, and (b) Dataset preprocessing. The dataset used in this experiment is a comprehensive collection of 10 years of daily weather observations from across various locations in Australia. The dataset comprises of 23 columns, and 145,460 rows. This data was recorded from various meteorological weather stations. The target variable of this dataset is the 'RainTomorrow' variable, which determines whether it is going to rain the next day or not, based on the other features. The determining values alternate between 'yes' and 'no', with 'yes' representing that it did rain tomorrow, and 'no' representing that it did not rain tomorrow. Table 1 given below shows details of the dataset. The table records the feature names, their data type (categorical/numeric), their description, and the count of null values in said features.

The null value distribution of features in percentages is depicted in Fig. 4. Although two features 'Sunshine', and 'Evaporation' have large chunks of missing data, the missing percentage does not stand to 50% or above it, and hence were included in the experiment.

The response variable, and the 'RainToday' variable was encoded as 0 for 'no', and 1 for 'yes', at the beginning of preprocessing. The value distributions of '0' and '1' in the response variable and the 'RainToday' variable, are shown below in figs. 5 and 6 respectively.

The next step of formatting involved data imputation for the categorical variables. Since the 'Date' and 'Location' variables did not have any null values, the rest of the categorical variables were imputed with the mode values, in replacement

Table 1 Dataset analysis

Feature name	Type	Description	Not null Count	Null Count
Date	Categorical	Date of the recorded day	145,460	0
Location	Categorical	Common location name, for the weather station	145,460	0
MinTemp	Numerical	Minimum recorded temperature for that day, in Celsius	143,975	1485
MaxTemp	Numerical	Maximum temperature for that day, in Celsius	144,199	1261
Rainfall	Numerical	Rainfall estimated amount, for that particular day	142,199	3261
Evaporation	Numerical	Class A pan evaporation in mm	83,670	62,790
Sunshine	Numerical	Count of hours in a day with significant sunshine	75,625	69,835
WindGustDir	Categorical	Strongest wind gust direction from 24 h to midnight	135,134	10,326
WindGustSpeed	Numerical	Strongest wind gust speed from 24 h to midnight	135,197	10,263
WindDir9am	Categorical	Wind propagation direction at 9am	134,894	10,566
WinDir3pm	Categorical	Direction of propagating wind at 3 pm	141,232	4228
WindSpeed9am	Numerical	Average wind speed, approximately 10 min before 9am	143,693	1767
WindSpeed3pm	Numerical	Average wind speed, approximately 10 min before 3 pm	142,398	3062
Humidity9am	Numerical	Humidity percentage at 9am	142,806	2654
Humidity3pm	Numerical	Humidity percentage at 3 pm	140,953	4507
Pressure9am	Numerical	Atmospheric pressure at 9am, reduced to mean sea level	130,395	15,065
Pressure3pm	Numerical	Atmospheric pressure at 3 pm, reduced to mean sea level	130,432	15,028
Cloud9am	Numerical	Fraction of sky covered by clouds at 9am	89,572	55,888
Cloud3pm	Numerical	Fraction of sky covered by clouds at 3 pm	86,102	59,358
Temp9am	Numerical	Temperature in degrees (Celsius) at 9am	143,693	1767
Temp3pm	Numerical	Temperature in degrees (celsius) at 3 pm	141,851	3609
RainToday	Categorical (Binary)	Depicts whether it rained that particular day or not. 'yes' if precipitation levels in 24 h is above 1 mm, otherwise 'no'	142,199	3261
RainTomorrow	Categorical (Binary)	Response Variable, depicting whether it rains the next day or not	142,193	3267

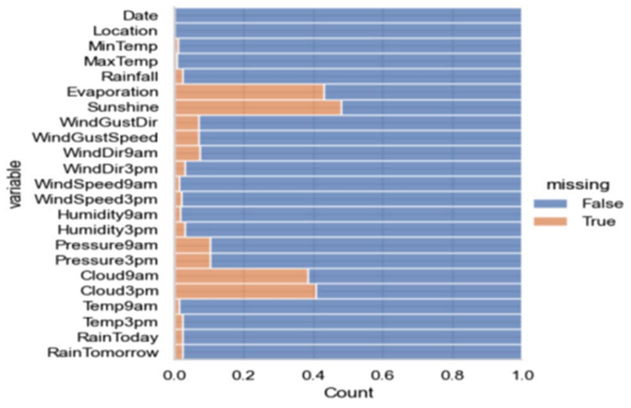

Fig. 4 Percentage distribution of null values in each feature of the dataset

for the missing data points. The data points in which the response variable itself was missing/null, were dropped, since imputing those points, could result in manipulation of the model during training. These categorical features were then encoded using Label Encoder. The label encoder assigned numeric values, usually on basis of the frequency of the labels. The rest of the features with continuous values, and missing points, were imputed using the MICE (multi-variate Imputation by Chained Equations) imputation. This imputation technique initially replaces the missing values of the features required with some value (suppose mean of each feature), then replaces the imputed value back to missing for each feature, and predicts the value of that data point, using linear/logistic regression with the other features being used to make the prediction. This process is repeated for each feature, and the resultant dataset is subtracted from the very first dataset, with mean imputed values. This process is repeated till the absolute difference between the imputed data points becomes close to zero. Before dividing the dataset into training and testing sets, it was divided into 'x' and 'y'. With

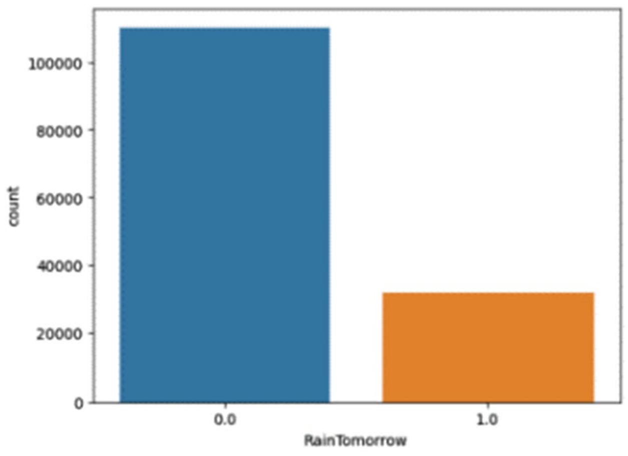

Fig. 5 Value distribution 'RainTomorrow

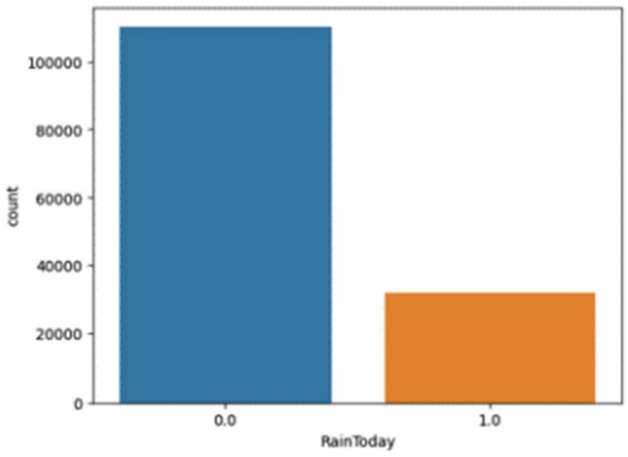

Fig. 6 Value distribution of 'RainToday'

Table 2 Count of data points after dividing into training and testing sets

Datasets	Features	Data points
Training	x: 21, y: 1	112,629
Testing	x: 21, y: 1	28,158

y consisting of just the response variable, 'RainTomorrow', and x consisting of all features but the response feature, and the 'Date' feature, this was divided into training and testing sets with the train_test_split function from sklearn. Table 2 below shows the shapes of training and testing datasets.

Before processing, the SMOTE was applied to the training set to handle the skew ness of the majority: minority ratio. After applying SMOTE, synthetic minority data points were inculcated into the training set for better training of the model. The final training dataset amounted to a shape of (175378 21) as x, and (175378,) as y.

4.2 Experimental conditions

This experiment was carried out in Jupyter Notebook 6.5.2, in Python. The system specifications include the inherent properties of Lenovo IdeaPad 5, with AMD Ryzen 75,000 processor, 1.80 GHz clock speed. This experiment required usage of multiple libraries. For the base predictors, parameter optimization was not performed, so as to get a proper estimate of how well the models perform on a clean slate. Bagging classifier was run with specifications as shown in Table 3. All other hyperparamteres were used at their default value.

For booting, AdaBoost Classifier was used with the base estimator set to the different models used in the experiment, and 50 estimators were used for XGBoost, CatBoost, LightGBM, and RandomForest. A count of 200 estimators were used for the rest of the models.

Table 3 Parameter specifications for bagging classifier

Base estimator used	Base estimator count	Max samples
Logistic regression	500	0.8
KNN	200	0.8
DecisionTree	500	0.8
RandomForest	100	0.8
XGBoost	30	0.8
CatBoost	20	0.8
LightGBM	700	0.8

5 Results

Ensemble learning is especially useful when trying to reduce errors of bias and variance. High bias in model performance is likely hinting that the model is not learning adequately from the given data and is running into underfitting problems. We use boosting in this experiment to take care of the same. Higher variance shows that the model has tried to fit their function too close to the provided sample data points, and is running into problems with overfitting. We have sought to improve the variance by usage of bagging. In every model, therefore, there exists a bias-variance trade-off. Ideally, it is preferable to have low bias and low variance. But practically, the decline in one of the two errors comes at the cost of an increase in the other. In this experiment, classification models like Logistic Regression, K-Nearest Neighbors, DecisionTree Classifier, RandomForest Classifier,Catboost Classifier, LightGBM Classifier, and XGBoost Classifier, were tested and compared on the dataset.With these models, there was effort to demonstrate the changes in the bias-variance error trade-off by using bagging and boosting. As shown in Tables 5 and 6, the decrease in one has inadvertently resulted in an increase for the other one. Even in case of boosting, which is theoretically supposed to reduce the biasing error, we saw an increase in the biasing error while observing declining variance errors. This might have been caused by the use of the AdaBoost algorithm, in which the base estimators were changed by placing the chosen models in place of the defaulted decision tree. For the bagging process, the BaggingClassifier model from sklearn.ensemble was used, with its hyperparameters, especially the number of estimators used, were adjusted accordingly. The metrics used for the comparison of the models were accuracy, precision, recall, and FNR(False Negative Rate). Their significance is as follows:-

- Accuracy is the measure of the percentage of data classified correctly (both positive and negative), among all of the predictions made.
- The precision metric, represents the percentage of data (i.e. data marked '1'), classified correctly, among all the predictions made by the model to be positive.
- Recall signifies the ability of the model to correctly identify the datapoints belonging to the positive class (i.e. class marked '1'), from all predictions made.
- The False Negative Rate (FNR)represents the model's misclassification rate, i.e. it measures the data points, which have been misclassified as negative (marked '0'), when their actual labelling is positive (marked '1').

The mathematical formulae for these metrics are given in Eqs. (8), (9), (10), and (11) respectively, where TP, TN, FP and FN are classified as follows:

- TP (True Positives): Data points that are really labeled positive (1), and are predicted as positive (1).
- TN (True Negatives): Data points that are really labeled negative (0), and are predicted as negative (0).
- FP (False Positives): Data points that are really labeled negative (0), but are predicted as positive (1).
- FN (False Negatives): Data points that are really labeled positive (1), but are predicted as negative (0).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

Table 4 Results for base estimators without any ensemble technique

Models	Accuracy	Precision	Recall	FNR	Bias	Variance
Logistic regression	0.7976	0.7944	0.5299	0.4701	0.1879	0.0111
KNN	0.7819	0.8014	0.5061	0.4938	0.1820	0.0442
DecisionTree	0.8229	0.6939	0.5861	0.4114	0.1030	0.0845
RandomForest	0.8737	0.75131	0.7019	0.3414	0.1118	0.0202
XGBoost	0.9031	0.7221	0.8205	0.1794	0.0839	0.0202
CatBoost	0.8718	0.6312	0.7523	0.2477	0.0935	0.0153
LightGBM	0.8665	0.6585	0.7521	0.2479	0.1053	0.0217

Table 5 Results for base estimators after application of boosting using AdaBoost

Models	Accuracy	Precision	Recall	FNR	Bias	Variance
logistic regression	0.7975	0.7827	0.5302	0.4697	0.2010	0.0017
DecisionTree	0.8267	0.6991	0.5926	0.4073	0.1004	0.0817
RandomForest	0.8711	0.7522	0.7051	0.2949	0.1188	0.0099
XGBoost	0.7754	0.0	0.0	0.0	0.0733	0.0214
CatBoost	0.9026	0.7264	0.8156	0.1844	0.0808	0.0376
LightGBM	0.8909	0.6992	0.7864	0.2135	0.0811	0.0316

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

$$FNR = \frac{FN}{TP + FN} \tag{11}$$

We also looked at the bias and variance percentage in each model, before and after taking them through ensemble learning processes, to check if the applications truly do affect underfitting and overfitting, in a model. The results obtained via testing, without bagging/boosting are recorded in Table 4. These results show a general increase in overall performance, except with the models, which are inbuilt applications of ensemble learning.

The accuracy was lowest for the logistic regression and KNN models, due to their linearity which makes them less effective when it comes to picking up non-linear relationships between the input features, and the class variable. Results obtained via testing, with boosting are recorded in Table 5.

The KNN model was not utilized for boosting purposes due to its lack of the sample-weights parameter which makes it unfit for the AdaBoost Classifier, or any existing boosting models in scikit-learn since the core concept of boosting stands on decreasing the errors in iterative base learners, to improve performance by updation of weights. It can however be boosted via techniques like LPBoost, or local wrapping of the distance metric [37]. One more notable observation is that the XGBoost classifier provides 0.0% precision and recall score, and hence is deemed unfit for further boosting, as it is already an implementation of the boosting technique. The other models Catboost and LightGBM classifiers provide expected results due to their nature of being meta-ensemble learning models rather than being direct applications of the method. Results obtained via testing, with bagging are recorded in Table 6.

From Table 6 and Figs. 7, 8 we can see a decrease in predictive metrics (accuracy, precision, and recall), in the cases of RandomForest Classifier and XGBoost Classifier in our experiment. The notable observation, in this case, was that both of these models are already existing applications of the bagging and boosting techniques. Hence, redundancy in operations, and increase in complexity might have contributed to the decrease in predictive accuracy. The bias-variance trade-off however performs as expected for all models, with a sharp decrease in variance error, whilst alternatively increasing the biasing error. This signifies that applying bagging to models does prevent overfitting.

For boosting, some decrease in variance with an increase in biasing was also observed, even though theoretically, boosting only reduces biasing with a probable increment in variance. We can see the results of bagging and boosting, with context to bias and variance in Figs. 9 and 10. For bagging, the bias variance trade-off is abundantly clear. For boosting, we observe a similar trade-off with decrease in variance where biasing increase and vice-versa. Notable

Table 6 Results for base estimators after bagging using BaggingClassifier

Models	Accuracy	Precision	Recall	FNR	Bias	Variance
Logistic regression	0.8022	0.7936	0.5374	0.4625	0.2036	0.0016
KNN	0.8012	0.8106	0.5149	0.4851	0.1708	0.0316
DecisionTree	0.8597	0.6531	0.6969	0.3030	0.1283	0.0141
RandomForest	0.8687	0.7619	0.6837	0.3163	0.1336	0.0117
XGBoost	0.8889	0.6740	0.7915	0.1959	0.1048	0.0085
CatBoost	0.8780	0.6436	0.7703	0.2297	0.1144	0.0075
LightGBM	0.8754	0.6572	0.7509	0.2490	0.1180	0.0062

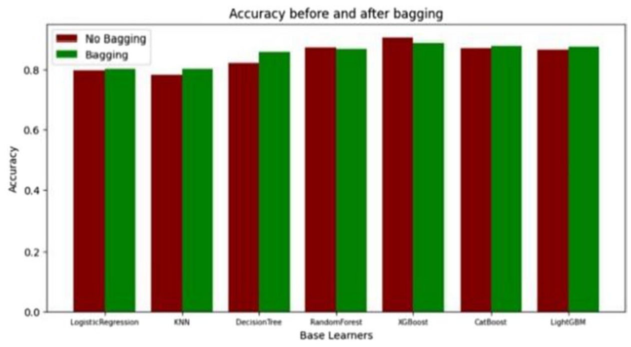

Fig. 7 Data plot of accuracy before and after bagging

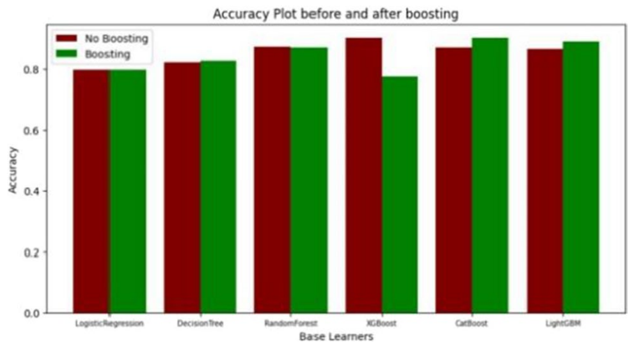

Fig. 8 Data plot of accuracy before and after boosting

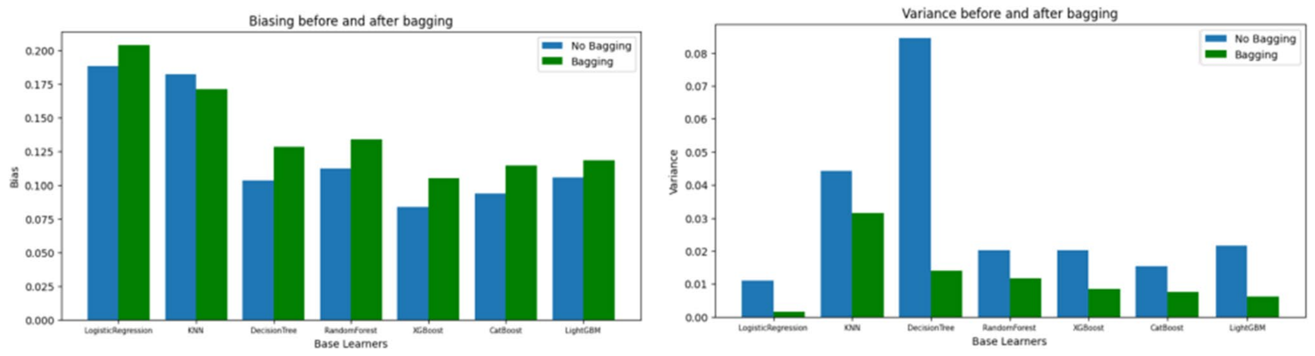

Fig. 9 (Left)Bias plot before and after bagging, (Right)Variance plot before and after bagging

observation in this case was the increase in biasing error and decrease in variance in case of logistic regression, and RandomForest, in Fig. 10. This might have occurred due linearity in logistic regression, and increase in complexity for the RandomForest classifier.

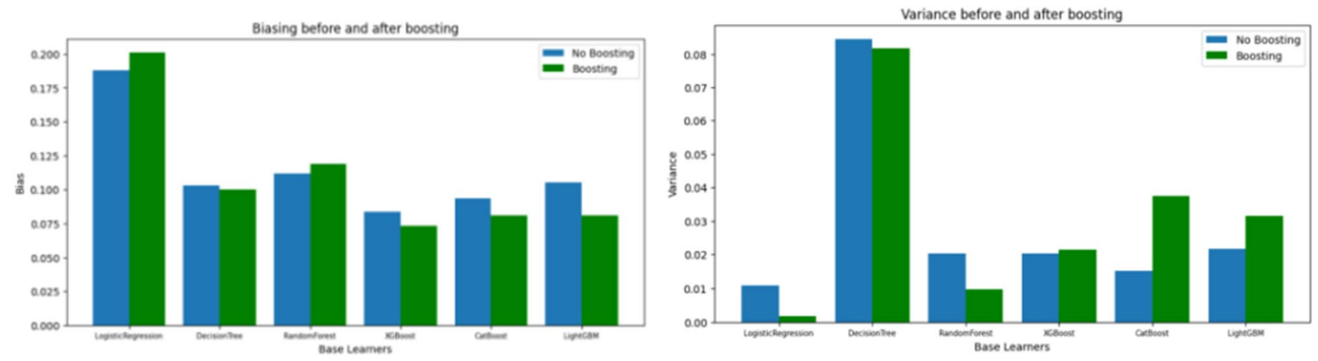

Fig. 10 (Left)Bias plot before and after boosting, (Right)Variance plot before and after boosting

With bagging, the DecisionTree Classifier improved the most with an increase of 3.68% in accuracy, and 0.0435 drastic decrease in variance error. To counter the variance drop, the bias error faced an increase of 0.0253. For the boosted models, the greatest decrease in bias occurred for the LightGBM classifier, depicting a drop of 0.0242 with an increase in predictive accuracy by 2.44%. The increase in bias for this model was by 0.0099.

6 Conclusion and future work

From the experiment, it was concluded that application of ensemble techniques does indeed improve overall performance of pre-existing models, whilst maintaining prevention of overfitting and underfitting. This is especially seen via the decrease in biasing and variance errors after application of the used techniques. The use of multiple similiar models to predict the final outcome seems to work quite well, for producing even the littlest bit of improvement in performance, evening out the predictive property of the classifiers. We looked at models which were direct applications of the techniques used to begin with and observed that in some cases, like XGBoost, its inherent properties make it unfit for further boosting, and we also concluded that AdaBoost could not be applied to linear models without the usage of sample weights. With boosting, we see a decrease in the biasing error in most models and a decline in variance with the others, whereas with application of bagging, we can observe a decline in the variance. However, it is obvious, that with increase in one parameter, the other one decreases, and vice versa. We can also conclude, that while ensemble learning can affect, and improve the model performance, it is not extremely practically for datasets as large as the one being used in this experiment. The sheer size of the dataset may lead to decrease in aggregate performance regardless of the number or type of estimators we use for bagging/boosting. The training time for each model was quite high, especially with application of ensemble learning, and with systems running the experiment on less powerful processors, the system might even risk crashing altogether. Even with further advancement of ML and AI, it is imperative that more work needs to be done, especially in application of actions being taken against climate change. Further applications of ensemble techniques, such as stacking or application of techniques like bagging and boosting consecutively on the same model, may better the model performance even more. Working collaboratively, with machine learning and its various applications, as demonstrated in this paper, can help bring about at least minuscule improvements in the global climate crisis. It will, however be a large step towards the fight against climate change, as more fields continue to be explored, with help of which, we can hopefully find an optimal solution for this dilemma. The unpredictability of rainfall forecasting, in general, can also be taken care of using integration of fuzzy sets and ensemble learning to improve predictions as well.

Author contribution The authors confirm contribution to the paper as follows: study conception and design: HD; data collection: BS; analysis and interpretation of results: HD, SG, MKG; draft manuscript preparation: SG. HD. MKG, BS. All authors reviewed the results and approved the final version of the manuscript.

Funding No funding was obtained for this study.

Data availability The data used for this experiment are available in public repository (Kaggle). The detailed information is given in the result analysis section.

Declarations

Competing interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Yilmaz AG. The effects of climate change on historical and future extreme rainfall in Antalya Turkey. Hydrol Sci J. 2015;60(12):2148–62.
2. Loo YY, Billa L, Singh A. Effect of climate change on seasonal monsoon in Asia and its impact on the variability of monsoon rainfall in Southeast Asia. Geosci Front. 2015;6(6):817–23.
3. Meynecke JO, Lee SY, Duke NC, Warnken J. Effect of rainfall as a component of climate change on estuarine fish production in Queensland, Australia. Estuar Coast Shelf Sci. 2006;69(3–4):491–504.
4. Kotz M, Levermann A, Wenz L. The effect of rainfall changes on economic production. Nature. 2022;601(7892):223–7.
5. Theis L, Oord AVD, Bethge M. 2015. A note on the evaluation of generative models. arXiv preprint arXiv:1511.01844. 2015
6. Pazos N, Favara M, Sánchez A, Scott D, Behrman J. Long-term effects of rainfall shocks on foundational cognitive skills: evidence from Peru. SSRN Electron J. 2023. https://doi.org/10.2139/ssrn.4360823.
7. Pariyar SK, Keenlyside N, Sorteberg A, Spengler T, Bhatt BC, Ogawa F. Factors affecting extreme rainfall events in the South Pacific. Weather Clim Extremes. 2020;29:100262.
8. Yue W, Wang Z, Chen H, Payne A, Liu X. Machine learning with applications in breast cancer diagnosis and prognosis. Designs. 2018;2(2):13.
9. Liyew CM, Melese HA. Machine learning techniques to predict daily rainfall amount. J Big Data. 2021;8:1–11.
10. Manandhar S, Dev S, Lee YH, Meng YS, Winkler S. A data-driven approach for accurate rainfall prediction. IEEE Trans Geosci Remote Sens. 2019;5(11):9323–31.
11. Zainudin S, Jasim DS, Bakar AA. Comparative analysis of data mining techniques for Malaysian rainfall prediction. Int J AdvSciEng Inform Technol. 2016;6(6):1148–53.
12. Chandra S, Gourisaria MK, Gm H, Konar D, Gao X, Wang T, Xu M. Prolificacy assessment of spermatozoan via state-of-the-art deep learning frameworks. IEEE Access. 2022;10:13715–27.
13. Jee G, Harshvardhan GM, Gourisaria MK. Juxtaposing inference capabilities of deep neural models over posteroanterior chest radiographs facilitating COVID-19 detection. J Interdiscip Math. 2021;24(2):299–325.
14. Agrawal R, Singh V, Gourisaria MK, Sharma A, Das H. Comparative analysis of CNN Architectures for maize crop disease. In: 2022 10th International conference on emerging trends in engineering and technology-signal and information processing (ICETET-SIP-22). IEEE. 2022. pp. 1–7
15. Khare S, Gourisaria MK, Harshvardhan GM, Joardar S, Singh V. Real estate cost estimation through data mining techniques. IOP Conf series Mater Sci Eng. 2021;1099(1):012053.
16. Pirone D, Cimorelli L, Del Giudice G, Pianese D. Short-term rainfall forecasting using cumulative precipitation fields from station data: a probabilistic machine learning approach. J Hydrol. 2023;617:128949.
17. Basha CZ, Bhavana N, Bhavya P, Sowmya V. Rainfall prediction using machine learning & deep learning techniques. In: 2020 international conference on electronics and sustainable communication systems (ICESC). IEEE. 2020. pp. 92–97
18. Fahad S, Su F, Khan SU, Naeem MR, Wei K. Implementing a novel deep learning technique for rainfall forecasting via climatic variables: an approach via hierarchical clustering analysis. Sci Total Environ. 2023;854:158760.
19. Rahman AU, Abbas S, Gollapalli M, Ahmed R, Aftab S, Ahmad M, Khan MA, Mosavi A. Rainfall prediction system using machine learning fusion for smart cities. Sensors. 2022;22(9):3504.
20. Barrera-Animas AY, Oyedele LO, Bilal M, Akinosho TD, Delgado JMD, Akanbi LA. Rainfall prediction: a comparative analysis of modern machine learning algorithms for time-series forecasting. Mach Learn Appl. 2022;7:100204.
21. Manna T, Anitha A. Precipitation prediction by integrating rough set on Fuzzy approximation space with deep learning techniques. Appl Soft Comput. 2023;139:110253.
22. Suparta W, Samah AA. Rainfall prediction by using ANFIS times series technique in South Tangerang Indonesia. Geod Geodyn. 2020;11(6):411–7.
23. Venkatachalam K, Trojovský P, Pamucar D, Bacanin N, Simic V. DWFH: an improved data-driven deep weather forecasting hybrid model using transductive long short term memory (T-LSTM). Expert Syst Appl. 2023;213:119270.
24. Kashiwao T, Nakayama K, Ando S, Ikeda K, Lee M, Bahadori A. A neural network-based local rainfall prediction system using meteorological data on the Internet: a case study using data from the Japan meteorological agency. Appl Soft Comput. 2017;56:317–30.
25. Van SP, Le HM, Thanh DV, Dang TD, Loc HH, Anh DT. Deep learning convolutional neural network in rainfall–runoff modelling. J Hydroinf. 2020;22(3):541–61.
26. Hudnurkar S, Rayavarapu N. On the performance analysis of rainfall prediction using mutual information with artificial neural network. Intl J Electr Computer Eng. 2023;13(2):2101.

27. Tran Anh D, Duc Dang T, Van Pham S. Improved rainfall prediction using combined pre-processing methods and feed-forward neural networks. J. 2019;2(1):65–83.
28. Khan MI, Maity R. Hybrid deep learning approach for multi-step-ahead daily rainfall prediction using GCM simulations. IEEE Access. 2020;8:52774–84.
29. Kaur H, Kumar M, Gupta A, Sachdeva M, Mittal A, Kumar K. Bagging: an ensemble approach for recognition of handwritten place-names in gurumukhi script. ACM Trans Asian Low-Resour Lang Inf Process. 2023. https://doi.org/10.1145/3593024.
30. Sarah S, Gourisaria MK, Khare S, Das H. Heart disease prediction using core machine learning techniques—a comparative study in advances in data and Information sciences proceedings of ICDIS 2021. Singapore: Springer Singapore; 2022. p. 247–60.
31. Ukey N, Yang Z, Li B, Zhang G, Hu Y, Zhang W. Survey on exact knn queries over high-dimensional data space. Sensors. 2023;23(2):629.
32. Azam Z, Islam MM, Huda MN. Comparative analysis of intrusion detection systems and machine learning based model analysis through decision tree. IEEE Access. 2023. https://doi.org/10.1109/ACCESS.2023.3296444.
33. Jain N, Jana PK. LRF: a logically randomized forest algorithm for classification and regression problems. Expert Syst Appl. 2023;213:119225.
34. Singh V, Gourisaria MK, Das H. Performance analysis of machine learning algorithms for prediction of liver disease. In: 2021 IEEE 4th international conference on computing, power and communication technologies (GUCON). IEEE. 2021. pp. 1–7
35. Jhaveri S, Khedkar I, Kantharia Y, Jaswal S. Success prediction using random forest, catboost, xgboost and adaboost for Kickstarter campaigns. In: 2019 3rd International conference on computing methodologies and communication (ICCMC). IEEE. 2019. pp. 1170–1173
36. Hancock J, Khoshgoftaar TM. Medicare fraud detection using catboost. In: 2020 IEEE 21st international conference on information reuse and Integration for data science (IRI). IEEE. 2020. pp. 97–103
37. Neo TKC, Ventura D. A direct boosting algorithm for the k-nearest neighbor classifier via local warping of the distance metric. Pattern Recogn Lett. 2012;33(1):92–102.