International Journal on Advanced Science Engineering Information Technology

Vol.13 (2023) No. 3
ISSN: 2088-5334

GRU and XGBoost Performance with Hyperparameter Tuning Using GridSearchCV and Bayesian Optimization on an IoT-Based Weather Prediction System

Hendri Darmawan [a], Mike Yuliana [a,*], Moch. Zen Samsono Hadi [a]

[a] *Department of Postgraduate, Politeknik Elektronika Negeri Surabaya (PENS), Surabaya, 60111, Indonesia*
Corresponding author: [*]mieke@pens.ac.id

***Abstract*— Weather is essential to human life, but it is difficult to forecast due to its diverse nature. We evaluated and compared the accuracy of two machine learning algorithms, GRU and XGBoost, in predicting weather patterns. We used GridSearchCV to tune the hyperparameters for the GRU algorithm and Bayesian optimization for the XGBoost algorithm. We used regression to predict weather sensor data and classification to predict rainfall in the following four days. We then deployed the best-performing model to the cloud server and connected it to the local IoT device with weather sensors in Sedati, Sidoarjo Regency, Indonesia. We conducted tests using data from the BMKG Juanda Sidoarjo and data from the local IoT device. The findings indicated that the XGBoost regression model outperformed the GRU model in the first stage, with an average RMSE of 1.2728125. In comparison, the average RMSE for GRU regression was 1.551666667. In the second stage, however, GRU regression performed better, with an average RMSE of 2.23, while the XGBoost regression had 2.28. In the classification tests, the GRU model had a higher F1 score of 0.88 in the first stage, while the XGBoost classification was 0.86. Both models had the same accuracy of 0.75 when tested with IoT data. However, the GRU classification model was better since it considered the context of the prediction, resulting in a lower likelihood of rain when it was not raining.**

***Keywords*— Gated recurrent unit; XGBoost; multivariate weather prediction; internet of things.**

Manuscript received 27 Oct. 2022; revised 27 Dec. 2022; accepted 24 Jan. 2023. Date of publication 30 Jun. 2023.

I. INTRODUCTION

Accurate weather prediction is essential for many industries and activities, including agriculture, tourism, aviation, and transportation. Weather data is collected using observations, sensors, radar, and satellites from meteorological stations. The Meteorology, Climatology, and Geophysics Agency's current method for developing weather forecasts is called Numerical Weather Prediction (NWP) [1]. This method requires several factors, mathematical assumptions, and complex equations, and it requires a thorough understanding of atmospheric dynamics and calculations with many variables and data sets. The development of modern computer hardware has made advances in numerical weather forecasting. The potential for improving weather modeling techniques exists thanks to the availability of large amounts of data and technological advances. We can gain new insights and potentially enhance the existing weather modeling methods by utilizing a deep learning approach using decades of weather observation records. However, a large amount of weather and climate data complicates its analysis.

Researchers have developed a new modeling approach using machine learning that has many benefits over traditional methods. Unlike physical models or numerical weather prediction, machine learning models can provide results in seconds and offer accurate forecasts at a lower cost [2]. The purpose of the machine learning approach can be divided into two categories: description and prediction. A descriptive function examines the dynamics of data collection to extract meaningful characteristics. On the other hand, the predictive function looks for patterns in the data that can be used to estimate future outcomes using the variables in the data. These patterns are then used to forecast variables that have not yet been observed. Training and testing data are required for modeling using a machine-learning algorithm.

This study used 20.5 years of data from the BMKG Meteorology Station (Class I, Juanda) in Indonesia. There were issues with the data, including missing values and outliers. These issues may have resulted from data processing

issues, data entry mistakes, climatic anomalies, or sensor errors. Therefore, before training the model, it is essential to perform various data preparation steps, such as filling in missing values (interpolation), data cleansing, transformation, and standardization or normalization.

Munandar [3] has used multivariate time series input data using the ARIMA and MLP methods for weather forecasting with solar irradiance targets. MLP regression model single-day output prediction using a multilayer perceptron window method model with data from 3 and 7 days before. In order to predict future data, the ARIMA model considers parameters such as moving average, autoregressive, and data set features. The researchers tested two models and found that the MLP model using deep learning was more effective than the ARIMA model.

Chen et al. [4] have studied the weather prediction of average wind speed, average atmospheric pressure, daily minimum and maximum temperature, relative humidity, and temperature in Shenzhen, China. The researchers employ a fusion model based on LSTM. The method of filtering the correlation coefficients of the components of each variable decomposed by EMD and then recombining the data into an LSTM network maximizes the benefits of EMD in decomposing non-stationary data with seasonal trends. It minimizes the impact of data noise and seasonal fluctuations. The researchers employ a grid search approach for tuning the hyperparameters.

Our proposed research builds on previous work in which we developed a model using Internet of Things technology. This study used a server-side API endpoint to integrate the models deployed on the cloud server with the ESP32 microcontroller. The ESP32 microcontroller was equipped with several sensors to measure meteorological conditions. The prediction model used the data collected from these sensors as input. We also created a web-based surveillance system for this project to allow users to monitor the weather in near real-time and view the weather forecast.

II. Materials and Methods

The model development technique used for weather prediction is discussed in this section. Observations of weather at the surface were used to generate forecasts. We proposed two methods: the first used the GRU algorithm, and the second used the XGBoost algorithm. Each algorithm consists of a regression model and a classification model. In this study, we proposed four models consisting of two regression models and two classification models. The regression models were used to predict meteorological element data, including Maximum Temperature (MAX), Minimum Temperature (MIN), Maximum Wind Speed (MXWS), Daily Average Temperature (TEMP), Wind Current Speed (WS), Humidity (RH), Sea Level Atmospheric Pressure (SLP), and Dew Point Temperature (DP).

The classification models were used to classify rainfall prediction (PRCP) for the following four days. We evaluated these models to determine the most reliable models for regression and classification. A locally deployed Internet of Things (IoT) device was used to collect data as inputs for the most reliable models to provide a comprehensive weather forecasting system. An ESP32 microcontroller was connected to BME280, an anemometer, and rain gauge sensors to collect weather data. The server received this data, which was used as input for the prediction model. Since we were not using multilabel classification in this research, the regression model was used to predict the values input into the classification model for the next few days. The models were trained using historical weather data to provide future forecasts. The hyperparameters of the GRU algorithm were tuned using the GridSearchCV method, while the XGBoost algorithm was tuned using Bayesian optimization.

A. Research Methodology

The proposed study comprises three main stages, as shown in Fig. 1. The first stage is data preparation, which involves collecting data sets and pre-processing the data. Data pre-processing includes interpolation, treating outlier values, filtering data using Fast Fourier Transform (FFT), transforming data, and normalizing data. The second stage is modeling with two algorithms, GRU and XGBoost. The hyperparameters of these models need to be tuned to achieve the best performance and accuracy using GridSearchCV and Bayesian optimization. The results are then analyzed and evaluated. The third stage is implementing and integrating the best model with a local IoT device.

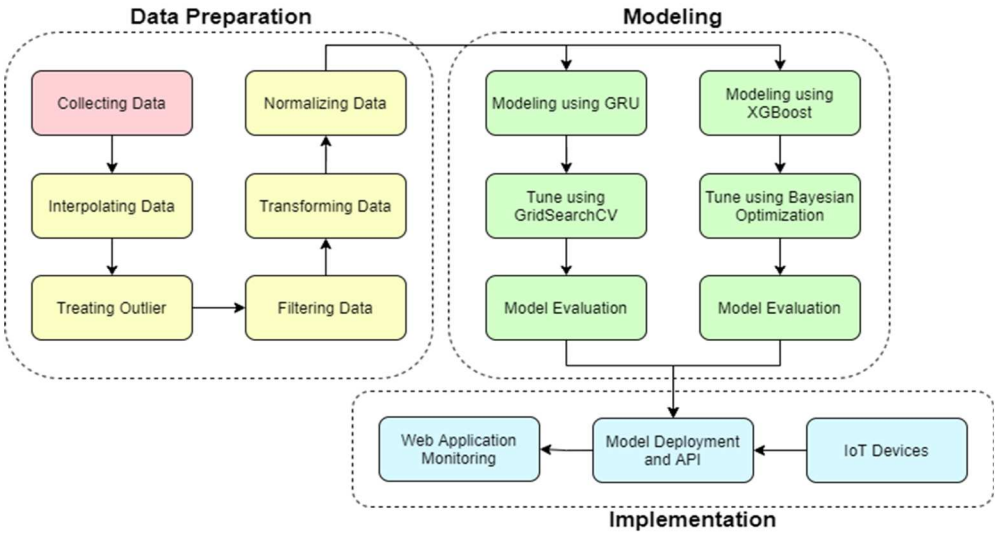


Fig. 1 Research methodology

B. Collecting Data

The models were trained using data from the BMKG Meteorology Station (Class I, Juanda) in Sidoarjo Regency, Indonesia. The National Climatic Data Center (NCDC) is an institution that gathers and maintains weather data from airport weather stations worldwide, which is made available for download. We used relevant variables from the raw data as input variables for the model, considering the availability of sensors. The data collected between January 2000 and June 2021 was divided into sets for testing and training.

Fig. 2 shows the descriptive statistics of the raw data. The first quartile (Q1) includes data points that are less than 25% of the total, the second quartile (Q2) includes data points that are less than 50% of the total, and the third quartile (Q3) includes data points that are less than 75% of the total, arranged in ascending order. Fig. 2 shows that the minimum and maximum values are significantly different from the values in the first quartile (Q1), the middle quartile (Q2), and the third quartile (Q3). A significant difference between quartiles and minimum-maximum values may indicate skewed data. There may also be outliers in the data. Histograms and boxplots were also used to further check for suspected outliers. Another issue with the dataset is missing values, which were addressed in the data interpolation stage by adding new data points.

	count	mean	std	min	25%	50%	75%	max	data lost
PRCP	7471.0	4.55	13.14	0.00	0.00	0.00	1.02	199.90	324
DP	7794.0	23.02	1.57	15.61	22.06	23.50	24.17	26.00	1
MAX	7795.0	32.23	1.27	25.28	31.39	32.22	33.00	38.00	0
MIN	7795.0	24.07	1.41	10.72	23.28	24.28	25.00	29.61	0
MXWS	7785.0	11.00	3.57	1.90	8.90	9.90	13.00	49.90	10
SLP	7665.0	1009.95	1.59	1003.50	1008.90	1010.10	1011.10	1015.20	130
TEMP	7795.0	28.04	1.18	23.72	27.17	28.00	28.89	32.11	0
WS	7793.0	5.59	2.07	0.00	4.20	5.40	6.70	17.70	2
RH	7794.0	74.12	7.75	47.23	68.31	73.89	80.11	96.33	1

Fig. 2 Descriptive statistics of raw data

C. Pre-Processing Data

The quality of the output of a system, such as a machine learning model, is directly related to the quality of the input data. The "garbage in, garbage out" principle states that if the input data is of poor quality, the resulting output will also be of poor quality. It is crucial to ensure that the input data is high quality to build a reliable and accurate model. The raw data provided by the NCDC had some anomalies and missing values, so we performed pre-processing to clean the data, as shown in Fig. 1. By performing pre-processing, we can improve the quality of the input data and increase the accuracy of the resulting model.

1) Interpolating Data: Data interpolation is a method for filling in missing data points by estimating values based on existing samples. This research employed means imputation, an interpolation using statistical method [5]. Missing data points are replaced with the average value from the same day in other years [6]. For example, if the sea level pressure record for October 1, 2021, is missing, it would be replaced with the average sea level pressure from prior years.

2) Treating Outlier: A value much smaller or larger than the rest of the data is called an extreme point or outlier. A boxplot is a helpful plot for visualizing data distribution based on five essential calculations: the lowest value, Q1, Q2, Q3, and the highest value. We used boxplots for each variable to identify outliers. The upper and lower bounds of the data set were used to define the cutoff point, as shown by the boxplot.

3) Filtering Data: FFT is a technique that converts data from the spatial or time domain into the frequency domain [7], [8]. It uses a complex exponential function to break the data into component frequencies. In contrast, Inverse Fast Fourier Transform (IFFT) transforms data from the frequency domain back into the spatial or time domain. The FFT equation for X(f) of x(t) in continuous time is shown in (1).

$$X(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi ft} dt \tag{1}$$

The IFFT equation is written in (2):

$$X(t) = \int_{-\infty}^{\infty} x(f) \cdot e^{j2\pi ft} df \tag{2}$$

The FFT filter technique converts the data into the frequency domain, reduces or amplifies high frequencies (acting as a low pass filter), and then inverts the filtered result using the IFFT method [9]. This filtering step helps to reduce high-point fluctuations, to improve the model's performance. By removing noise from the data, the model can more easily identify underlying patterns and trends, leading to better results [10]. This research used a window size (N) of eight to filter the variables, and a two-point sample was taken from the FFT (M) retained value. Fig. 3 shows the distribution of plots after the pre-processing data phase.

4) Transforming Data: In this step, continuous rainfall data was converted into two categories: no rain (class 0) and rain (class 1). If a rainfall data point was more significant than 0.5 mm, it was labeled as part of the rain class [11]. However, the data used in this research was unbalanced, meaning that some classes occurred less often than others. It can cause the model to be biased and perform better for frequent classes than for unusual ones [12]. Several solutions to the imbalanced data issue include SMOTE [13]. In this research, we did not use SMOTE to handle imbalanced classes because it produces unrealistic sequences for time series data, which does not improve model performance. Instead, we used a weight penalty to address the imbalanced data. It assigns a lower weight to the class with more labels and a higher weight with fewer labels. The classification model naturally gives more weight to the class with more labels, so we needed to weight the loss function to counteract this bias. The formula for estimating penalty weight is shown (3):

$$\frac{n\_total\_samples}{n\_class\_samples * frequency\_each\_element} \tag{3}$$

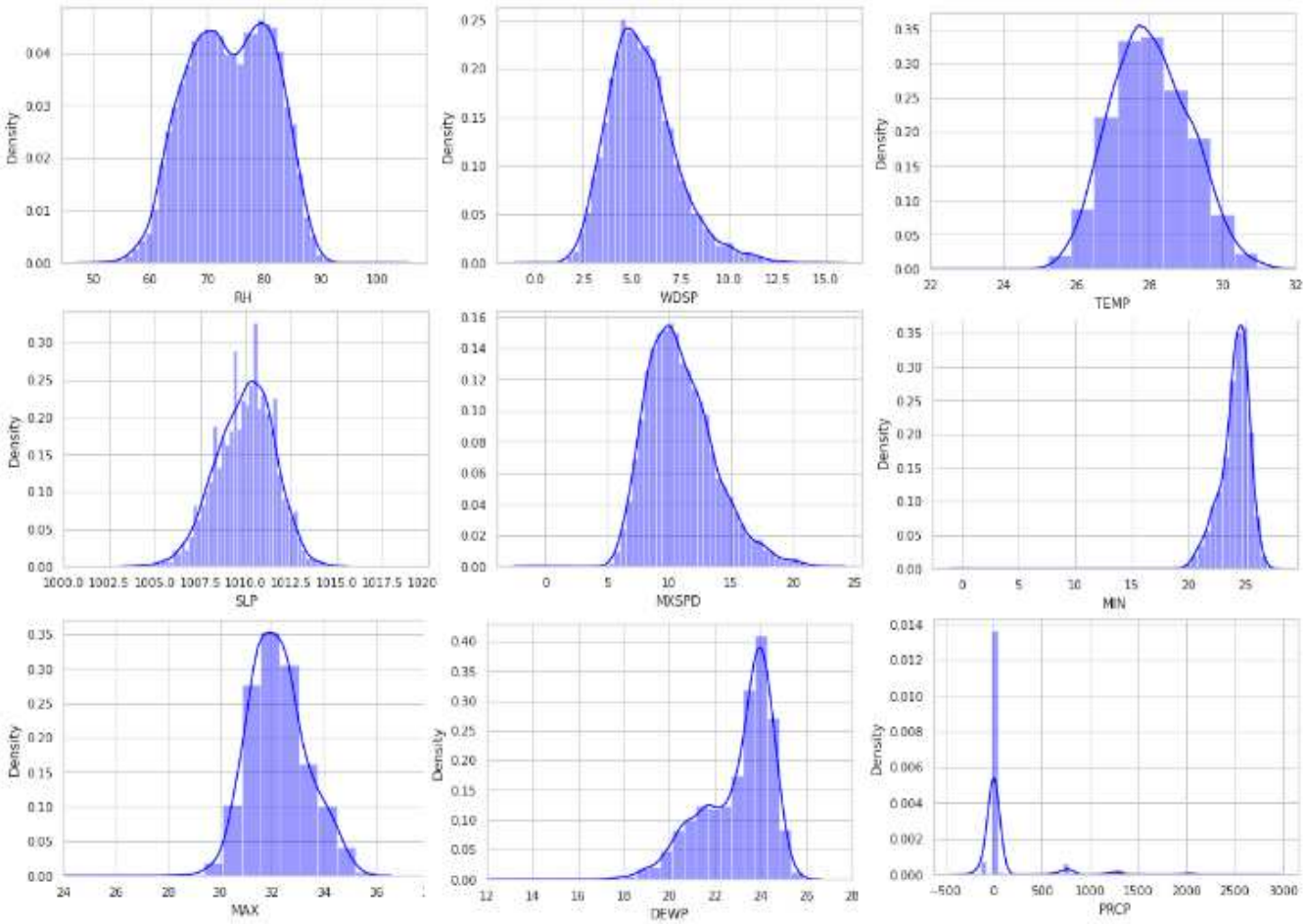


Fig. 3 Distribution plot after pre-processing data

Fig. 4 shows the calculated weights for each class, where the weight is inversely related to the frequency of the data [14]. These weight penalties were only applied to the GRU model classification because the XGBoost model can handle imbalanced data without customization. However, using a sampling technique may improve the performance of the XGBoost algorithm [15].

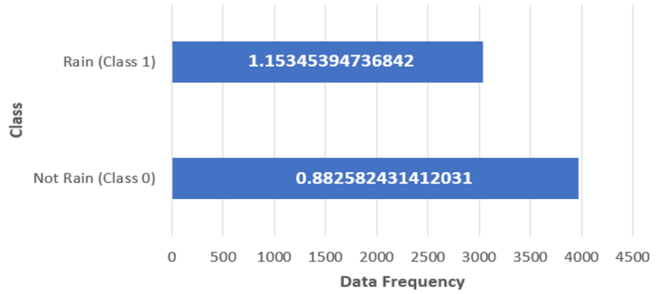


Fig. 4 Weighting value estimation for each class

5) Normalizing Data: Normalizing the data ensured that all the features were on the same scale, which helped the model learn more effectively. Data normalization helps the model to converge faster and produce better results [16]. In this study, a minimum-maximum scaler was used for data rescaling. The normalization procedure yields data ranging from 0 to +1 by scaling each feature [17]. The minimum-maximum equation is given by (4):

$$X_{i\_new} = \frac{X_i - min(X_i)}{max(X_i) - min(X_i)} \quad (4)$$

D. Modeling

This study used the GRU and XGBoost models for weather prediction. The GRU model is based on recurrent neural networks, while XGBoost is a boosting algorithm that uses decision trees as its base learners. We evaluated the performance of both models by tuning their hyperparameters and comparing their results.

1) Gated Recurrent Unit (GRU) Algorithm: GRU is a type of recurrent neural network that can be used to improve the performance of vanilla Recurrent Neural Networks (RNN) in predictive modeling tasks. Unlike vanilla RNN, which often suffers from vanishing gradient issues, GRU uses the update and reset gates to prevent vanishing gradients. These gates give GRU a more stable architecture with many hidden layers, improving model performance. The equations are shown in (5)-(8).

$$u_t = \sigma(W_u x_t + U_u h_{t-1} + b_u) \quad (5)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \quad (6)$$

$$c_t = \tanh(W x_t + U(r_t \circ h_{t-1}) + b) \quad (7)$$

$$h_t = u_t \circ h_{t-1} + (1 - u_t) \circ c_t \quad (8)$$

Where σ is the sigmoid activation function, W_u, B_u, W_r, and B_r are weight matrices and bias vectors for the update and reset gates, x_t is the input at time step t, and $h_{(t-1)}$ is the hidden state at the previous time step. The update (u_t) and reset gate (r_t) control the flow of information in the GRU, allowing it to retain or forget information from previous time steps as needed. The hidden state candidate (c_t) is used to generate the hidden state (h_t), with the result of the update gate calculation being used to control the effect of the previous hidden state on the hidden state candidate. This method helps the GRU model learn and make accurate predictions [18]. To prepare the data for training with a GRU model, we first rearranged it into three-dimensional forms compatible with the GRU compliance layer. The input layer's three-dimensional forms consist of data samples, the number of time steps, and dimensions. Each pattern represents a single sample of data, one measurement point within the sample represents one historical window (time step), and each feature represents a measurement point within the time step. It allowed us to train the GRU model on the pre-processed data [6].

Hyperparameters in machine learning algorithms affect model performance. Tuning hyperparameters can improve prediction accuracy for specific datasets. This study examined the effects of different time-step periods, including two, seven, fourteen, and twenty-one days. GridSearchCV was used to find the optimal hyperparameters for the regression model. The GridSearchCV method for tuning hyperparameters generates and assesses the model for each pair of the provided hyperparameters [19]. Table 1 and Table 2 present the hyperparameter values of the regression and classification models using GRU, along with the results that show which hyperparameters were the most reliable.

TABLE I
HYPERPARAMETERS FOR GRU REGRESSION MODEL

Hyperparameter	Range of Values	Best Value
Time step	2; 7; 14; 21	14
Bidirectional GRU units on layer 1	16; 32; 64; 128; 256; 512	16
Dense units on layer 2	16; 32; 64; 128; 256; 512	512
Batch size	16; 32; 64; 128; 256; 512	128

TABLE II
HYPERPARAMETERS FOR GRU CLASSIFICATION MODEL

Hyperparameter	Range of Values	Best Value
Time step	2; 7; 14; 21	2
GRU units on layer 1	16; 32; 64; 128; 256	128
Dense units on layer 2	16; 32; 64; 128; 256	64
Batch size	16; 32; 64; 128; 256	128

In order to optimize the hyperparameters for the rain category prediction, we manually varied the time step, GRU, Dense units on the hidden layer, and the batch size number. We evaluated the impact on the model performance. The adjustments were made one at a time, and the results from the most successful iteration were used to tune the hyperparameters for the next iteration. This process was repeated until the optimal hyperparameters were found.

We first defined the model architecture to build the GRU models and then used optimization algorithms and loss functions. The regression model was compiled using mean squared loss, while sequence classification model used categorical cross-entropy loss. Loss functions measure how much actual results differ from predicted results. Backpropagation through time is used to adjust the weights and biases in the GRU model to reduce the cost incurred during training.

The optimization algorithm repeatedly adjusts the network weights based on the training data. In this study, we used the adaptive optimization Adam because the default settings are usually practical. We also used a reduced learning rate on plateau function with a starting learning rate 0.001 [20]. A large learning rate is desirable at the beginning of training because it can lead to a higher generalization effect. If the metrics are not improving during training, slowing the learning rate can help the algorithm find an optimal solution and avoid oscillations around that solution [21]. If the metrics show no improvement after a predetermined number of epochs, we use the ReduceLROnPlateau callback to reduce the learning rate [22]. We also used early stopping to prevent overfitting by stopping the training process if the validation loss increases significantly. We can determine the optimal number of epochs by early stopping because the training process will automatically stop at a certain epoch [23]. Finally, we saved the best weights during training using the ModelCheckPoint callback, which was then used for deployment.

2) Extreme Gradient Boosting (XGBoost) Algorithm: XGBoost is a popular machine-learning algorithm often used for regression and classification tasks [24]. It is an ensemble learning method that combines the predictions of multiple weak models to create a more accurate final model. XGBoost uses decision trees as its base learners and trains them in an iterative process to improve the model's overall performance. Combining multiple weak models to create a more robust model is known as gradient boosting. This method allows XGBoost to produce highly accurate predictions, making it a popular choice for many machine-learning applications. During each iteration, the error residuals from the preceding model are used to fit the subsequent model. The final prediction is derived by a weighted summation of all the individual tree predictions. The XGBoost algorithm may be thought of as an additive model that is made up of K CART trees, $f_t(x_i)$ is a representation of the predicted value that may be produced by feeding the *i*-th sample x_i into the *t*-th tree, $\hat{y}_i$ is a representation of the prediction outcome of x_i, and F is the set space containing all the regression trees [15]. The final prediction result formula is given by (9):

$$\hat{y}_i = \sum_{t=1}^{K} f_t(x_i), f_t \in F \quad (9)$$

The objective function (loss function and regularization) at iteration t that has to be minimized is as follows (10):

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathrm{x}_i)\right) + \Omega(f_t) \quad (10)$$

The XGBoost model's hyperparameters were optimized using the Bayesian optimization method. This approach is more efficient than traditional search methods, such as random and grid searches, because it uses a probabilistic model to optimize the process [25]. Bayesian optimization uses Bayes' theorem to search global optimization problems

efficiently [26]. Bayesian optimization involves iteratively searching for the hyperparameters that minimize the objective loss function, using a surrogate function to represent the objective and an acquisition function to guide the search. This method can also reduce computational costs compared to grid search. The optimum hyperparameter values for multilabel regression using XGBoost are shown in Table 3. Determining the range of samples hyperparameters for the classification model on Bayesian optimization processes follows the same pattern as determining the range of samples for the regression model. Table 4 presents the optimal hyperparameter values for the XGBoost classifier, as determined by the Bayesian optimization method.

TABLE III
HYPERPARAMETERS FOR XGBOOST REGRESSION MODEL

Hyperparameter	Range of Values	Best Value
History window	2; 7; 14; 21	7
subsample	Uniform(0.6, 1)	0.72823302704416
colsample_bytree	Uniform(0.3, 1)	0.42000701842358
max_depth	Randint(4, 11)	7
min_child_weight	Uniform(0, 10)	5.76455502635815
learning_rate	Uniform(0.01, 0.3)	0.03816683337691
n_estimators	Randint(100, 500)	438
gamma	Uniform(0, 2.5)	0.00277475809094

TABLE IV
HYPERPARAMETERS FOR XGBOOST CLASSIFICATION MODEL

Hyperparameter	Range of Values	Best Value
History window	2; 7; 14; 21	14
subsample	Uniform(0.6, 1)	0.76743448256266
colsample_bytree	Uniform(0.3, 1)	0.50282874177672
max_depth	Randint(4, 11)	10
min_child_weight	Uniform(0, 10)	2.26328971561941
learning_rate	Uniform(0.01, 0.3)	0.09731752061633
n_estimators	Randint(100, 500)	102
gamma	Uniform(0, 2.5)	0.81687404301826

The hyperparameters of the XGBoost model were optimized using Bayesian optimization. It involved searching for the optimal values of the hyperparameters using one hundred assessments of different models for each historical window value (the number of past observations used as features). The Parzen estimators search technique was used to minimize the objective function for the regression and classification models. For the regression model, the objective function was the mean squared loss, while for the classification model, it was the negative accuracy. By minimizing these objective functions, we can improve the performance of the models and achieve better results. The hyperparameters obtained from Bayesian optimization show that the XGBoost model for classification is more complex than the regression model. The large value of depth of each tree, referred to as max_depth, makes the XGBoost model more complex. To prevent overfitting due to the increased complexity of the XGBoost model, we can adjust the values of two hyperparameters: min child weight and gamma. Min child weight is the minimum sum of instance weight in each leaf node, while gamma is the minimum loss reduction to produce a split. By increasing the values of these hyperparameters, we can lower the complexity of the model and prevent it from overfitting to the training data. It can improve the model's performance and help it generalize better to unseen data. Setting the ratio of features used, referred to as colsample_bytree, and the ratio of training instances, referred to as subsample, to a modest number also can reduce model complexity. Tables 3 and 4 also show that the regression model has more boosted trees, or n estimators, than the classification model. The learning rate of the XGBoost model was set to a constant value; when this value decreases, the computation becomes slower but sometimes yields the best optimum solution. The model was trained using the optimal hyperparameter values after the tuning process. Eighty percent of the tabular weather history data were used for training the model and twenty percent for testing. The gradient boosting tree technique is used, where XGBoost uses predictors sequentially and models them based on their predecessors' errors to assign greater weight to better-performing predictors. The XGBoost model is trained in three stages: raw data, residuals from the previous model, and the sum of the previous models.

E. Evaluation Metrics

To evaluate the performance of the models, we used several metrics. One of these metrics is the Root Mean Square Error (RMSE), a commonly used measure for regression models [27]. A smaller RMSE value indicates that the model's predictions are closer to the actual values and therefore have better performance. It calculates the average squared difference between the predicted values and the true values and takes the square root of the result as shown in (11):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}} \tag{11}$$

We compare the predicted values with the original labels to evaluate the performance of the classification model, which allows us to determine the extent to which the model can accurately predict the correct class for each data point [14]. We used a variety of metrics to assess the model's performance, including a confusion matrix, accuracy, recall, precision, and F1. A confusion matrix is a tabular representation of possible pairs of predicted and observed values [28]. The matrix consists of four possible outcomes: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). The formulae for classification model evaluations can be found given in (12)-(15) [29].

$$Accuracy = \frac{TN+TP}{TN+TP+FP+FN} \tag{12}$$

$$Precision = \frac{TP}{FP+TP} \tag{13}$$

$$Recall = \frac{TP}{FN+TP} \tag{14}$$

$$F1 = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{15}$$

F. Microcontroller and Model Deployment

After selecting the most accurate model, the next step was integrating it with a local IoT device and other components into a complete system. The integrated system allowed for the automatic processing of weather predictions and providing them to users through the application shown in Fig. 5. The sensors connected to an ESP32 microcontroller include a rain gauge, an anemometer, and a BME280. These sensors provide data on rain, wind, sea level atmospheric pressure, temperature, dew point, and relative humidity. The schematic

diagram of the microcontroller and sensors used in this research can be seen in Fig. 6. The ESP32 chip includes various communication interfaces such as Wi-Fi, Bluetooth, SPI, and I2C/UART. The availability of two separate processing cores is a significant advantage of the ESP32 chip [30]. In this project, ESP32 was used in dual-core mode, with the first core running the rain gauge sensor function and the second core running the anemometer and BME280 sensor functions. This separation of cores was implemented to eliminate delays in reading the rain gauge sensor and improve the overall performance.

Fig. 5 Display of the website-based application

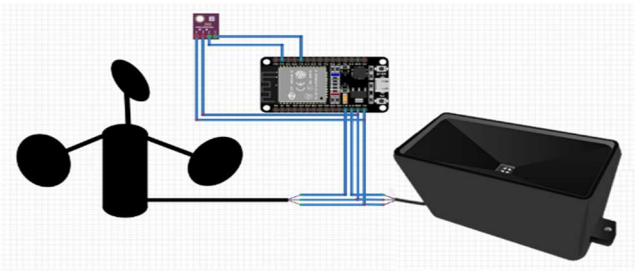


Fig. 6 Microcontroller and sensor schematic diagram

The weather data JSON was posted to the API endpoint via HTTP every minute using Wi-Fi [31]. The API, developed using Python Flask, then connected to the database and performed the necessary prediction processes. After receiving the data, the data was stored in MySQL (a relational database) and Google's Firebase Real-time Database [32]. The Firebase Real-time Database (a NoSQL database) was used to store the information provided by the website-based application, as it allows for immediate updates to the data in the application when it is changed [33]. Meanwhile, MySQL maintained a record of all sensor readings necessary for weather forecasting, and we used a query program to retrieve the data. The server then fetched the data from MySQL for a certain number of time steps, which were used as predictors for the prediction model. The average value of a day's sensor readings was calculated and used as input for the model at 23:59 WIB. These values were then used to generate the output predictions.

III. Result and Discussion

The model's accuracy was evaluated by comparing its predicted outcomes to the labeled data. The RMSE value was calculated to determine the regression model's effectiveness. A lower RMSE value indicates that the model's predictions are closer to the actual data, with a zero-value representing perfect accuracy. RMSE is commonly used in weather modeling, air quality studies, and climate studies to measure the accuracy of regression models [34]. The evaluation of the classification model is performed separately due to the categorical nature of its predictions. We used a variety of metrics, including the confusion matrix, accuracy, recall, precision, and F1, to assess the model's performance. We can determine the model's ability to accurately predict the different classes by analyzing these metrics. Overall, the evaluation of both the regression and classification models allows us to determine the effectiveness of the proposed model in predicting weather data.

A. Multi-Step Regression Model Testing With a 4-Day Lead Time

This section will present the results of evaluating the tuned model using the Root Mean Squared Error (RMSE) test. The GRU and XGBoost models could forecast the values of various weather parameters, including dew point, maximum temperature, minimum temperature, maximum wind flow speed, sea level atmospheric pressure, temperature, wind flow speed, and relative humidity, for four days in advance. The evaluation was conducted using the weather sensor data collected by the BMKG Meteorology Station (Class I, Juanda) from March 25, 2017, to June 30, 2021.

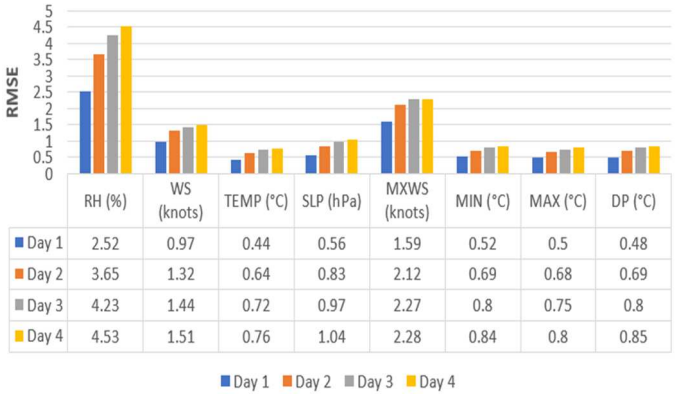
1) Gated Recurrent Unit (GRU): We tested the regression model using the results from GridSearchCV's tuning approach of the model hyperparameters by varying the time steps. The optimal hyperparameters for the GRU regression model were as follows: a Bidirectional GRU with 16 units in the first layer, a Dense layer with 512 units in the second layer, a Dense layer with 32 units in the output layer, a time step (historical) of fourteen days, and a batch size of 128. Fourteen-time steps mean that four days of predictions were made using fourteen days of historical data. The neurons in the recurrent neural network receive their input (the predictors) from the previous data based on the number of time steps. Fig. 7 shows the RMSE values for the GRU model's best-fitting model. The chart shows that the RMSE value increases as the prediction time increases, likely due to external factors that were not accounted for during the training process.

The location of an area plays a crucial role in the accuracy of weather predictions. In middle latitudes, weather forecasts can be made up to two weeks in advance [35], but in tropical areas, they can only be made up to four days in advance [36]. Long-term forecasting is not reliable as it tends to generate large errors. A high value of RMSE indicates a significant error in the prediction. The weather variables of temperature, maximum temperature, minimum temperature, sea level atmospheric pressure, and average daily dew point all had low RMSE values, indicating that they were accurately predicted.

It could be due to these variables' lack of significant fluctuations, particularly over the four days. On the other hand, the RMSE values for wind flow speed, relative humidity, and maximum wind speed were higher due to their greater fluctuation and volatility.

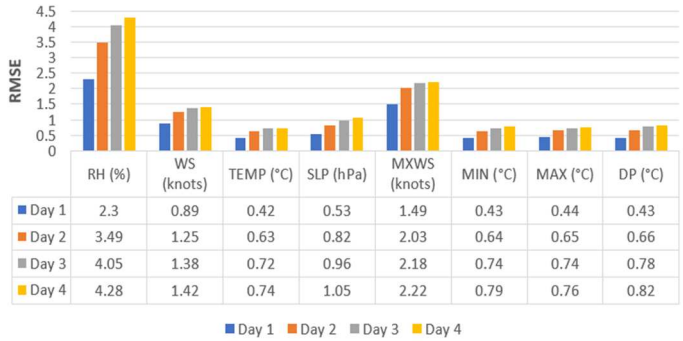
2) Extreme Gradient Boosting (XGBoost): The regression model using XGBoost with a seven-day history window, 0.73 subsamples, 0.42 colsample_bytree, seven max_depth, 5.76 min_child_weight, 0.038 learning rate, 438 n_estimators, and 0.0028 gammas yielded the optimal hyperparameters found using Bayesian optimization. Four days of predictions were made using seven-day of historical data. In contrast to the GRU method, which can analyze sequential input models, the XGBoost algorithm predicts using past data as all model features at once. Large dimensions are susceptible to dimensionality issues, which refers to the explosive nature of rising data dimensions and the exponential increase in computational work required for processing. The RMSE values for the optimal XGBoost model on the test data are shown in Fig. 8.

The comparison of the RMSE values between the XGBoost regressor and the GRU regressor model shows that the XGBoost model has a smaller average RMSE value of 1.2728125 for all variables compared to the GRU regression model's value of 1.551666667. It indicates that it performs better in predicting the test data. Fig. 8 also reveals that the RMSE value increases as the duration of the forecasted day increases, and the pattern is similar to the GRU regression model's RMSE pattern. This could be due to external factors not being considered during training. Overall, the XGBoost regressor model is a more effective for predicting weather data because it only uses seven-day historical data.



	RH (%)	WS (knots)	TEMP (°C)	SLP (hPa)	MXWS (knots)	MIN (°C)	MAX (°C)	DP (°C)
Day 1	2.52	0.97	0.44	0.56	1.59	0.52	0.5	0.48
Day 2	3.65	1.32	0.64	0.83	2.12	0.69	0.68	0.69
Day 3	4.23	1.44	0.72	0.97	2.27	0.8	0.75	0.8
Day 4	4.53	1.51	0.76	1.04	2.28	0.84	0.8	0.85

Fig. 7 RMSE of GRU regressor over four days



	RH (%)	WS (knots)	TEMP (°C)	SLP (hPa)	MXWS (knots)	MIN (°C)	MAX (°C)	DP (°C)
Day 1	2.3	0.89	0.42	0.53	1.49	0.43	0.44	0.43
Day 2	3.49	1.25	0.63	0.82	2.03	0.64	0.65	0.66
Day 3	4.05	1.38	0.72	0.96	2.18	0.74	0.74	0.78
Day 4	4.28	1.42	0.74	1.05	2.22	0.79	0.76	0.82

Fig. 8 RMSE of XGBoost regressor over four days

B. Classification Model Testing

This section presents the results of evaluating the classification models using the GRU sequence classifier and XGboost classifier algorithms. We compared the performance of the models by evaluating their scores from May 16, 2019, to June 30, 2021.

1) Gated Recurrent Unit (GRU): In this case, the sequence classification model was trained on weather data and was used to make predictions about the likelihood of rain. It was optimized by adjusting various parameters, including the time step, GRU units, number of dense units, and batch size. After testing different combinations of these parameters, the optimal model was found to have a batch size of 128, a time step of two days, and 128 GRU units in the first layer and 64 dense units in the second layer. Using categorical cross entropy as the loss function, the model could make binary predictions about the likelihood of rain. By considering the past two days of data, the model was able to predict the weather for the following day accurately. The results of testing this model are shown in Table 5, where it can be seen that the model achieved F1 score and an accuracy of 0.88.

Additionally, the model was more successful at predicting the likelihood of rain (class 1) than the likelihood of no rain (class 0), as indicated by the higher recall value for class 1. The false positive and false negative rates were 7.97% and 5.14%, respectively, indicating that the model was more likely to predict rain when it was not going to rain. Overall, the optimized sequence classification model performed well in predicting the likelihood of rain.

2) Extreme Gradient Boosting (XGBoost): The XGBoost classifier model achieved optimal hyperparameters using Bayesian optimization based on the tuning results. These hyperparameters include a maximum depth of 10, a minimum child weight of 2.26, a learning rate of 0.097, a number of estimators of 102, a gamma of 0.85, a subsample of 0.77, a colsample bytree of 0.50, and a fourteen-day history window. Table 5 displays the results of the performance metrics test, which demonstrate that the proposed GRU model outperforms the XGBoost when applied to test data. The XGBoost model's accuracy was 0.86, with a weighted average F1-Score of 0.86. Furthermore, the recall in class 0 (no rain) is lower than in class 1 (rain). The confusion matrix in Fig. 9b has the same spectrum form as the confusion matrix of the GRU model, which has a higher false positive rate of 9.54% than a false negative rate of 4.84%.

TABLE V
THE PERFORMANCE OF GRU AND XGBOOST CLASSIFICATION MODEL

Model	Class	Recall	Precision	F1	Acc
GRU	Not rain	0.87	0.92	0.88	0.88
	Rain	0.89	0.83		
XGBoost	Not rain	0.83	0.91	0.86	0.86
	Rain	0.89	0.80		

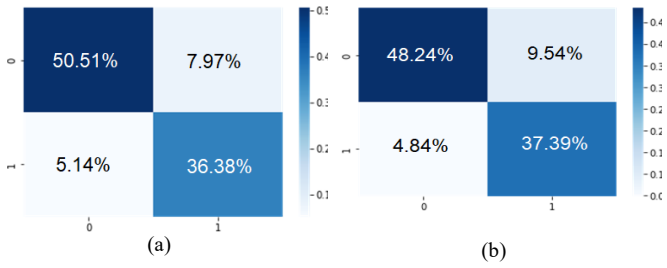


Fig. 9 Confusion matrix on test data: (a) GRU; (b) XGBoost

C. *Evaluation of Model Performance Using Microcontroller Observation Data*

Before testing the best model with local IoT data, the sensor values from our device were compared to the data from the sensors at the BMKG Meteorology Station (Class I, Juanda). This approach aims to reduce the error margin by standardizing the sensors' characteristics. The following differential values were obtained for dew point, sea level pressure, temperature, and wind flow speed: 0.94; -1.70; -1.12; -1.48; 1.01. For the next four days, sensor regressions were predicted using two weeks of data (GRU) and one week of data (XGBoost). The classification algorithm was then applied to the predicted sensor data to provide four-day rain category predictions.

1) Gated Recurrent Unit (GRU): Fig. 10 shows the results of sensor prediction based on the most reliable regressor model. The actual data for comparison was collected from the BMKG Meteorology Station (Class I, Juanda). The actual data is in orange, while the forecasted data is in yellow. The data from Fig. 10 can be used to calculate the RMSE and determine how well the model performs in the implementation phase. The average RMSE for the predictions of dew point, maximum temperature, minimum temperature, maximum wind speed, sea level pressure, temperature, wind flow speed, and humidity for the next four days is 0.55, 1.78, 3.04, 2.47, 0.57, 1.56, 2.3, and 5.6, respectively. Since the sea level pressure and dew point variables varied very little over the four-day test, the RMSE values for these variables were quite acceptable.

The RMSE for the humidity variable was the highest among all other variables. The forecasted values for humidity fell between 77.4 and 83.6, while the actual values were between 82 and 88. The model's forecast values had a smaller range than the observed data for humidity. The RMSE values were also relatively high for temperature, minimum, and maximum temperature, indicating that the forecasted values span a wider range than the observed data for these variables. Despite this, the model's forecasts accurately represent the natural environment's features. In their natural states, temperature and humidity are inversely related; when temperatures are high, humidity is often low. The differences in ambient sensor conditions between the BMKG Meteorology Station (Class I, Juanda) sensors and the sensors used in this study may be the reason for the large RMSE results for humidity and other variables. The meteorological station's temperature and relative humidity sensors are protected by radiation shields, which shield them from radiant heat and other environmental influences.

In this research, the sensors were not protected by radiation shields, so they were susceptible to interference from wind, sunshine, and other external factors. Next, we will discuss the results of the classification model. As shown in Table 6, the test using local IoT data yielded three accurate predictions and one incorrect one. The incorrect forecast occurred on June 2, when the model predicted rain but did not rain. However, there was a 32.81% increase in the likelihood that it would not rain on June 2 compared to the previous day.

Additionally, the likelihood of rain on June 3 increased by 9.14% compared to the previous day. The model predicted rain, but there were only 0.55 millimeters of precipitation. On June 4, the probability of precipitation decreased by 8.37%, but the model accurately predicted rainfall of 4 millimeters. Overall, the accuracy of the prediction system using microcontroller input data was 0.75 for predicting the next four days. However, to guarantee the performance of the sequence model, it is necessary to collect more observational data, mainly when it is evaluated with local IoT data as model predictors. The threshold for defining a rain forecast category can also be calculated using the ROC curve metric by examining more data results.

2) Extreme Gradient *Boosting (XGBoost)*: This section discusses the results of testing the XGBoost model using local IoT data. We will begin by examining the performance of the multilabel regression model on the original sensor observation data. Fig. 10 shows a chart of the regression model with seven-day input leading to four-day output. The orange line in the graph represents the observed data, while the green line represents the data predicted by the model. Based on the evaluation, the RMSE for four-day forecasts of dew point, maximum temperature, minimum temperature, maximum wind speed, sea level pressure, temperature, wind flow speed, and relative humidity was 0.31, 1.81, 1.08, 3.95, 0.37, 1.01, 3.62, and 6.16, respectively. These values were obtained by using local IoT data as model predictors.

According to this, the RMSE of sea level atmospheric pressure and the dew point had the lowest RMSE values compared to other variables, which were less than 1. The RMSE for all variable values was smaller than the RMSE obtained by the GRU model, except for humidity, wind flow speed, maximum wind flow speed, and maximum temperature. However, when the average RMSE was calculated for all variables, the RMSE produced by the XGBoost regressor was 2.28, while the RMSE produced by the GRU regressor was 2.23. Although the difference in performance was insignificant, the XGBoost regression model was more feasible to implement because it only needed seven days of history window. In comparison, the GRU model required fourteen days of history window (time step). More history windows make the computation slower because the model must process more predictors as model inputs.

Like the GRU sequence classification model, the XGBoost classifier method predicted three outcomes correctly and one incorrectly. The same incorrect forecast occurred on June 2. Even though there was no actual precipitation on June 2, the likelihood of rain increased by 20.57% compared to the previous day. This is worse than the GRU sequence classification model, which predicted a lower probability of rain than the previous day. Despite this, the accuracy of the XGBoost classification model using local IoT data can still be calculated at 0.75. However, further observation data is desirable to confirm the accuracy when applied to locally collected IoT data.

TABLE VI
RAIN PROBABILITY AND PRESENT WEATHER USING GRU AND XGBOOST

Date	GRU Probability		XGBoost Probability		Actual
	Not Rain	*Rain*	*Not Rain*	*Rain*	*Rainfall*
June 1	0.18315	0.81684	0.25595	0.74404	13.5 mm
June 2	0.24325	0.75674	0.10286	0.89713	0 mm
June 3	0.17405	0.82594	0.10438	0.89561	0.55 mm
June 4	0.21837	0.78162	0.07185	0.92814	4 mm

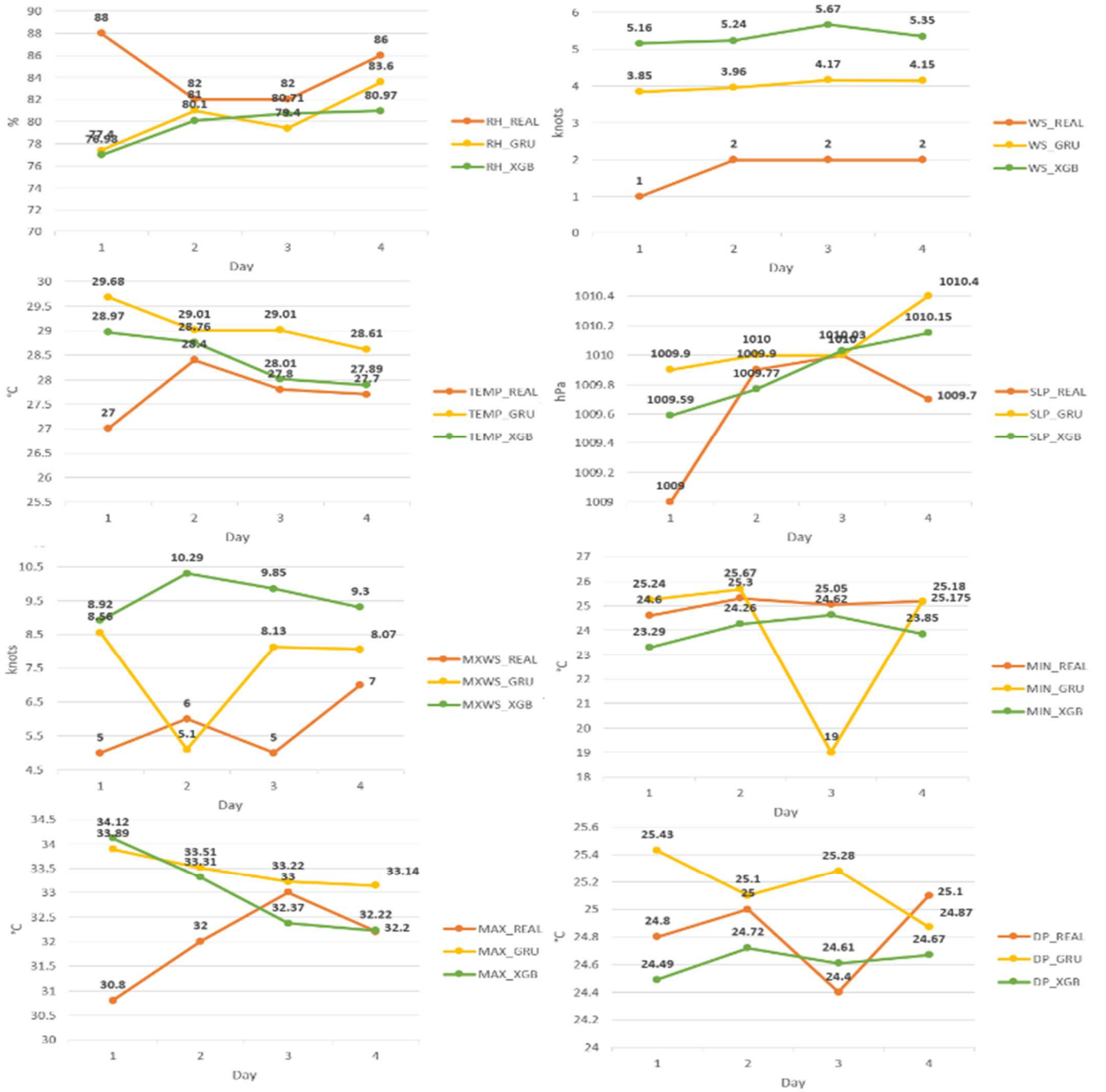


Fig. 10 Forecasting value from June 1, 2022, through June 4, 2022, using GRU and XGBoost

IV. CONCLUSION

The research compares the performance of two methods, GRU and XGBoost, for forecasting weather data over the next four days. We used GridSearchCV and Bayesian optimization to tune the models. The multilabel XGBoost sensor regression model outperformed the GRU model overall, with an average RMSE of 1.2728125 compared to the GRU model's 1.551666667. However, when evaluated with local IoT data, the GRU regression model performed better, with an RMSE of 2.23 compared to XGBoost's 2.28.

We also proposed a rainfall classification model, where the GRU model had a weighted F1 score of 0.88 and an accuracy of 0.88. The XGBoost model had a weighted F1 score of 0.86

and an accuracy of 0.86. When tested with microcontroller data from local IoT devices, the GRU model performed better because it could use context to make more accurate predictions about the likelihood of rain. However, we plan to improve future research by using more local IoT data and data from other weather sensors, such as UV and wind direction sensors.

ACKNOWLEDGMENT

We thank the BMKG Class I Juanda Meteorological Station for permitting us to calibrate our weather sensors using their station's equipment. We also appreciate the support of the Directorate General of DIKTI RISTEK through their PKM 2022 program, which provided funding for this research.

REFERENCES

[1] M. G. Schultz *et al.*, "Can deep learning beat numerical weather prediction?," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, no. 2194. Royal Society Publishing, Apr. 05, 2021. doi: 10.1098/rsta.2020.0097.

[2] S. F. Tekin, O. Karaahmetoglu, F. Ilhan, I. Balaban, and S. S. Kozat, "Spatio-temporal weather forecasting and attention mechanism on convolutional LSTMs," *ArXiv*, Feb. 2021, doi: 10.48550/ARXIV.2102.00696.

[3] D. Munandar, "Multilayer perceptron (MLP) and autoregressive integrated moving average (ARIMA) models in multivariate input time series data: solar irradiance forecasting," *International Journal on Advanced Science Engineering Information Technology*, vol. 9, no. 1, 2019, doi: 10.18517/ijaseit.9.1.6426.

[4] G. Chen, S. Liu, and F. Jiang, "Daily weather forecasting based on deep learning model: A case study of Shenzhen city, China," *Atmosphere (Basel)*, vol. 13, no. 8, Aug. 2022, doi: 10.3390/atmos13081208.

[5] X. Chen, Y. Liu, Y. Shen, K. Zhang, and H. Wei, "A data interpolation method for missing irradiance data of photovoltaic power station," in *2020 Chinese Automation Congress (CAC)*, Nov. 2020, pp. 4735–4740. doi: 10.1109/CAC51589.2020.9326730.

[6] M. Chhetri, S. Kumar, P. P. Roy, and B. G. Kim, "Deep BLSTM-GRU model for monthly rainfall prediction: A case study of Simtokha, Bhutan," *Remote Sens (Basel)*, vol. 12, no. 19, pp. 1–13, Oct. 2020, doi: 10.3390/rs12193174.

[7] T. E. Putra, Husaini, D. Asrina, and M. Dirhamsyah, "The ability of the fast fourier transform to de-noise a strain signal," in *IOP Conference Series: Materials Science and Engineering*, Oct. 2020, vol. 931, no. 1. doi: 10.1088/1757-899X/931/1/012011.

[8] A. González-Díez, J. A. Barreda-Argüeso, L. Rodríguez-Rodríguez, and J. Fernández-Lozano, "The use of filters based on the Fast Fourier Transform applied to DEMs for the objective mapping of karstic features," *Geomorphology*, vol. 385, Jul. 2021, doi: 10.1016/j.geomorph.2021.107724.

[9] S. U. Khan, M. H. Siddiqi, and Y. Alhwaiti, "Signal-to-noise ratio comparison of several filters against Phantom image," *J Healthc Eng*, vol. 2022, p. 4724342, 2022, doi: 10.1155/2022/4724342.

[10] P. Bellavista, A. Corradi, and C. Giannelli, "Evaluating filtering strategies for decentralized handover prediction in the wireless internet," in *11th IEEE Symposium on Computers and Communications (ISCC'06)*, 2006, pp. 167–174. doi: 10.1109/ISCC.2006.70.

[11] H. Darmawan, M. Yuliana, and Moch. Z. S. Hadi, "Real-time weather prediction system using GRU with daily surface observation data from IoT," in *2022 International Electronics Symposium (IES)*, 2022, pp. 221–226. doi: 10.1109/IES55876.2022.9888468.

[12] M. Steininger, K. Kobs, P. Davidson, A. Krause, and A. Hotho, "Density-based weighting for imbalanced regression," *Mach Learn*, vol. 110, no. 8, pp. 2187–2211, Aug. 2021, doi: 10.1007/s10994-021-06023-5.

[13] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *J Big Data*, vol. 6, no. 1, Dec. 2019, doi: 10.1186/s40537-019-0192-5.

[14] H. Patel, D. Singh Rajput, G. Thippa Reddy, C. Iwendi, A. Kashif Bashir, and O. Jo, "A review on classification of imbalanced data for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 16, no. 4. SAGE Publications Ltd, Apr. 01, 2020. doi: 10.1177/1550147720916404.

[15] P. Zhang, Y. Jia, and Y. Shang, "Research and application of XGBoost in imbalanced data," *Int J Distrib Sens Netw*, vol. 18, no. 6, Jun. 2022, doi: 10.1177/15501329221106935.

[16] L. Huang, J. Qin, Y. Zhou, F. Zhu, L. Liu, and L. Shao, "Normalization techniques in training DNNs: Methodology, analysis and application," Sep. 2020, doi: 10.48550/arXiv.2009.12836.

[17] G. Aksu, C. O. Güzeller, and M. T. Eser, "The effect of the normalization method used in different sample sizes on the success of artificial neural network model," *International Journal of Assessment Tools in Education*, pp. 170–192, Apr. 2019, doi: 10.21449/ijate.479404.

[18] X. Zhou, J. Xu, P. Zeng, and X. Meng, "Air pollutant concentration prediction based on GRU method," in *Journal of Physics: Conference Series*, Mar. 2019, vol. 1168, no. 3. doi: 10.1088/1742-6596/1168/3/032058.

[19] R. G. S. K., A. Kumar Verma, and S. Radhika, "K-nearest neighbors and grid search cv based real time fault monitoring system for industries," in *2019 5th International Conference for Convergence in Technology (I2CT)*, 2019, pp. 1–5.

[20] I. S. Isa, M. S. A. Rosli, U. K. Yusof, M. I. F. Maruzuki, and S. N. Sulaiman, "Optimizing the hyperparameter tuning of YOLOv5 for underwater detection," *IEEE Access*, vol. 10, pp. 52818–52831, 2022, doi: 10.1109/ACCESS.2022.3174583.

[21] K. Nakamura, B. Derbel, K. J. Won, and B. W. Hong, "Learning-rate annealing methods for deep neural networks," *Electronics (Switzerland)*, vol. 10, no. 16, Aug. 2021, doi: 10.3390/electronics10162029.

[22] K. Mukherjee, A. Khare, and A. Verma, "A simple dynamic learning rate tuning algorithm for automated training of DNNs," *ArXiv*, Oct. 2019, doi: 10.48550/ARXIV.1910.11605.

[23] P. Cu Thi, J. E. Ball, and N. H. Dao, "Early stopping technique using a genetic algorithm for calibration of an urban runoff model," *International Journal of River Basin Management*, 2021, doi: 10.1080/15715124.2021.1910517.

[24] A. Ibrahem Ahmed Osman, A. Najah Ahmed, M. F. Chow, Y. Feng Huang, and A. El-Shafie, "Extreme gradient boosting (Xgboost) model to predict the groundwater levels in Selangor Malaysia," *Ain Shams Engineering Journal*, vol. 12, no. 2, pp. 1545–1556, Jun. 2021, doi: 10.1016/j.asej.2020.11.011.

[25] M. Miranda, K. Valeriano, and J. Sulla-Torres, "A detailed study on the choice of hyperparameters for transfer learning in covid-19 image datasets using Bayesian optimization," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 4, pp. 327–335, 2021, doi: 10.14569/IJACSA.2021.0120441.

[26] Q. Liang *et al.*, "Benchmarking the performance of Bayesian optimization across multiple experimental materials science domains," *NPJ Comput Mater*, vol. 7, no. 1, Dec. 2021, doi: 10.1038/s41524-021-00656-9.

[27] M. Alizamir *et al.*, "Advanced machine learning model for better prediction accuracy of soil temperature at different depths," *PLoS One*, vol. 15, no. 4, Apr. 2020, doi: 10.1371/journal.pone.0231055.

[28] C. Esposito, G. A. Landrum, N. Schneider, N. Stiefl, and S. Riniker, "GHOST: Adjusting the decision threshold to handle imbalanced data in machine learning," *J Chem Inf Model*, vol. 61, no. 6, pp. 2623–2640, Jun. 2021, doi: 10.1021/acs.jcim.1c00160.

[29] I. M. de Diego, A. R. Redondo, R. R. Fernández, J. Navarro, and J. M. Moguerza, "General performance score for classification problems," *Applied Intelligence*, vol. 52, no. 10, pp. 12049–12063, Aug. 2022, doi: 10.1007/s10489-021-03041-7.

[30] P. Foltýnek, M. Babiuch, and P. Šuránek, "Measurement and data processing from Internet of Things modules by dual-core application using ESP32 board," *Measurement and Control (United Kingdom)*, vol. 52, no. 7–8, pp. 970–984, Sep. 2019, doi: 10.1177/0020294019857748.

[31] Y. S. Mandza and A. Raji, "IoTivity cloud-enabled platform for energy management applications," *IoT*, vol. 3, no. 1, pp. 73–90, Dec. 2021, doi: 10.3390/iot3010004.

[32] D. S. Anindya, M. Yuliana, and Moch. Z. S. Hadi, "IoT based climate prediction system using long short-term memory (LSTM) algorithm as part of smart farming 4.0," in *2022 International Electronics Symposium (IES)*, 2022, pp. 255–260. doi: 10.1109/IES55876.2022.9888486.

[33] M. Ohyver, J. v. Moniaga, I. Sungkawa, B. E. Subagyo, and I. A. Chandra, "The comparison firebase real-time database and MySQL database performance using wilcoxon signed-rank test," in *Procedia Computer Science*, 2019, vol. 157, pp. 396–405. doi: 10.1016/j.procs.2019.08.231.

[34] T. O. Hodson, "Root mean square error (RMSE) or mean absolute error (MAE): when to use them or not," *Geosci Model Dev*, vol. 15, no. 14, pp. 5481–5487, 2022, doi: 10.5194/gmd-2022-64.

[35] F. Zhang *et al.*, "What is the predictability limit of midlatitude weather?" *J Atmos Sci*, vol. 76, no. 4, pp. 1077–1091, 2019, doi: 10.1175/JAS-D-18-0269.1.

[36] H. Zhu, M. C. Wheeler, A. H. Sobel, and D. Hudson, "Seamless *precipitation* prediction skill in the tropics and extratropics from a global model," *Mon Weather Rev*, vol. 142, no. 4, pp. 1556–1569, 2014, doi: 10.1175/MWR-D-13-00222.1.