Jurnal Teknik Informatika C.I.T Medicom, 16 (6) (2025) 405-414

Published by: Institute of Computer Science (IOCS)

Jurnal Teknik Informatika C.I.T Medicom

Journal homepage: www.medikom.iocspublisher.org

Comparing optimization hyperparameter long short term memory for rainfall prediction model

Ilham Nur Hermawan[1], Martanto[2], Arif Rinaldi Dikananda[3], Mulyawan[4]

[1] Teknik Informatika, STMIK IKMI Cirebon, Jawa Barat, Indonesia
[2] Manajemen Informatika, STMIK IKMI Cirebon, Jawa Barat, Indonesia
[3] Rekayasa Perangkat Lunak, STMIK IKMI Cirebon, Jawa Barat, Indonesia
[4]Sistem Informasi, STMIK IKMI Cirebon, Jawa Barat, Indonesia

Article Info

Article history

Received : 20 Dec, 2024
Revised : Jan 17, 2025
Accepted : Jan 28, 2025

Keywords:

Deep learning;
Hyperparameter optimization;
LSTM algorithm;
Rainfall prediction.

Abstract

Improving the accuracy of weather prediction, especially rainfall, is very important in various sectors such as agriculture, water resource management, and disaster mitigation. This research aims to optimize the Long Short-Term Memory (LSTM) model in rainfall prediction through the application of hyperparameter optimization using two main techniques: Grid Search and Bayesian Optimization (Optuna). This hyperparameter optimization includes finding the best configuration of important parameters, such as the number of LSTM units, batch size, learning rate, and number of epochs. A historical rainfall dataset from BMKG is used, which is then divided into training and test data to build and test the prediction model. Grid Search performs a thorough exploration of all possible parameter combinations, while Optuna uses a probabilistic Bayesian approach to speed up the optimization process. The results show that hyperparameter optimization significantly improves the performance of LSTM models. The model optimized with Optuna produces a Mean Squared Error (MSE) value of 0.179578 with an execution time of 105.26 seconds, while Grid Search has an MSE of 0.286778 with an execution time of 457.69 seconds. The lower MSE value indicates that the Optuna model has a smaller prediction error, making it more accurate in predicting rainfall. The faster execution time of Optuna also confirms its efficiency in finding the optimal hyperparameter configuration compared to Grid Search. The conclusion of this study confirms that hyperparameter optimization plays an important role in improving the prediction accuracy of LSTM for rainfall. The developed method is expected to be the basis for the development of other weather prediction models as well as support decision-making in various sectors that rely on weather prediction. In addition, this research opens up opportunities for further studies in the optimization of deep learning models in handling complex climate data.

Corresponding Author:

Ilham Nur Hermawan,
Teknik Informatika,
STMIK IKMI Cirebon,
Jl. Perjuangan No.10B, Karyamulya, Kec. Kesambi, Kota Cirebon, Jawa Barat 45135,Indonesia
E-mail : ilhamnurhermawan@gmail.com

1. Introduction

The rapid advancement of artificial intelligence (AI) and machine learning (ML) has revolutionized various industries, particularly in the prediction of natural phenomena such as rainfall[1], [2]. Among

the wide range of ML algorithms, Long Short-Term Memory (LSTM), a specialized type of recurrent neural network, has gained recognition for its exceptional ability to manage time-series data. Its architecture, designed to capture both short and long term dependencies, makes it particularly effective in addressing complex and dynamic datasets. In applications such as stock price forecasting, sentiment analysis, and weather prediction, LSTM consistently delivers superior performance compared to traditional methods. The accurate prediction of rainfall is essential for decision-making processes in disaster mitigation, agricultural planning, and urban resource management[2], [3]. It enables stakeholders to anticipate potential risks and make informed decisions that can minimize damage and optimize resource use[4],[5].

Despite its advantages, rainfall prediction remains a challenging task due to the inherent complexity of weather data and the dynamic nature of meteorological factors[6], [7]. Rainfall patterns are influenced by numerous interrelated factors such as temperature, humidity, and atmospheric pressure, which vary across regions and timeframes. Although LSTM has been widely recognized for its effectiveness in handling sequential data, its performance is highly dependent on the proper configuration of hyperparameters[4], [8], [9]. Parameters such as the number of LSTM units, learning rate, and epochs significantly influence the model's accuracy and stability. Without proper optimization, LSTM models are prone to overfitting or underfitting, which can result in reduced predictive performance. Addressing these challenges requires a systematic approach to hyperparameter optimization, which has been underexplored in previous studies[4], [8], [9].

Previous research has demonstrated the potential of LSTM in improving rainfall prediction accuracy. Badriyah et al. [4] implemented LSTM for rainfall forecasting and reported that it outperformed traditional statistical methods in handling volatile weather data. However, their study lacked an in-depth focus on hyperparameter optimization, a critical factor in maximizing model performance. Similarly, Musfiroh et al. [10] integrated Principal Component Analysis (PCA) with LSTM to simplify complex datasets, achieving moderate success in prediction accuracy. Nevertheless, their work did not fully leverage advanced optimization techniques such as grid search or Bayesian optimization. These techniques have proven effective in other domains, as shown by Fajriyani et al.[5], who optimized neural networks for cyberbullying detection on social media, and Isnain et al. [11], who compared LSTM and Naive Bayes for sentiment analysis. While these studies highlight the importance of hyperparameter tuning, they do not address its application in rainfall prediction, leaving a gap in the literature[12], [13].

This study aims to fill this gap by systematically optimizing hyperparameters in LSTM models to enhance rainfall prediction accuracy[14]. The optimization process employs both grid search and Bayesian optimization techniques, each offering distinct advantages in exploring the parameter space[5], [15]. By identifying optimal configurations, this research seeks to address the challenges posed by complex and dynamic rainfall patterns[16]. The anticipated outcomes include improved predictive accuracy and model stability, which can significantly benefit practical applications[17]. These include better agricultural planning, more effective disaster mitigation strategies, and informed decision-making in urban development[4]. Ultimately, the findings of this study contribute to the growing body of knowledge in weather prediction and machine learning, offering insights for future research and broader applications in related fields[18].

2. Research Methodology

This study adopts the Deep Learning Workflow methodology for developing a Long Short-Term Memory model to predict rainfall[17]. The workflow, as described in the "Deep Learning Bible-2"[19], comprises four primary stages: Project Setup, Data Engineering, Model Engineering, and Code Engineering. These stages work in a cohesive sequence to ensure the development of an optimal and accurate predictive model tailored to address the complexity of rainfall data[1], [17], [20].

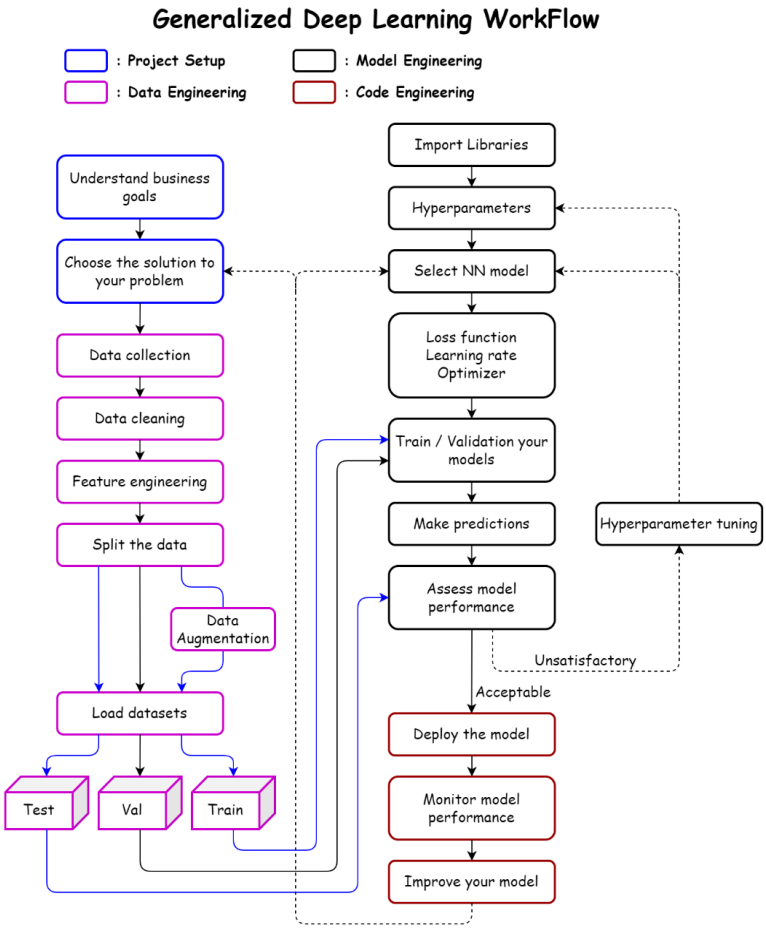

Figure 1. Deep Learning Workflow

a. Project Setup

The first stage is focused on defining the research objectives and selecting appropriate solutions to address the identified problem[1], [17]. The objective of this study is to enhance rainfall prediction accuracy through hyperparameter optimization in the LSTM model[1], [17]. LSTM is chosen due to its established effectiveness in managing sequential data, particularly in meteorological applications[18]. The selection of this methodology is supported by prior studies in deep learning and weather forecasting[21], [22]. The Deep Learning Workflow framework guides this study by providing structured and systematic steps for model development.

b. Data Engineering

Data engineering is critical for preparing a high-quality dataset, serving as the foundation for building and training the model[23]. This stage includes:

1) Data Collection

Historical rainfall data were obtained from the BMKG (Meteorological, Climatological, and Geophysical Agency) database https://dataonline.bmkg.go.id, specifically from the Penggung Meteorological Post in Cirebon[16]. The dataset spans from 2017 to 2023 and includes variables such as minimum temperature (Tn), maximum temperature (Tx), average temperature (Tavg), relative humidity (RH_avg), and rainfall intensity (RR). This comprehensive dataset is critical for analyzing patterns over an extended temporal scope.

2) Data Cleaning

The collected data were preprocessed to ensure quality by addressing missing values, removing duplicate records, and verifying data consistency. Proper cleaning ensures the dataset is free of anomalies, thereby reducing potential biases during model training.

3) Feature Engineering
Normalization techniques were applied to standardize feature scales, enhancing the compatibility of data for the LSTM model. Additionally, relevant features were extracted to improve model accuracy, adhering to the principles outlined in the Deep Learning Workflow[19].
4) Data Splitting
The dataset was divided into training, validation, and testing subsets. This process ensures the model is evaluated on unseen data, offering a realistic measure of its generalization capabilities.

c. Model Engineering

The third stage involves building and optimizing the LSTM model:

1) Library Import and Model Selection
Libraries such as TensorFlow, Keras, and Optuna were utilized to facilitate model development and hyperparameter optimization. LSTM architecture was configured based on established best practices for handling time-series data[24].
2) Initial Hyperparameter Selection
Fundamental hyperparameters like epochs, batch size, and learning rate were assigned initial values. These configurations served as the basis for subsequent optimization processes.
3) Loss Function and Optimizer Selection
The Mean Squared Error (MSE) was chosen as the loss function, while the Adam optimizer was employed for iterative weight updates. These choices align with the Deep Learning Workflow's emphasis on tailored optimization for specific tasks[19].
4) Training and Validation
The model was trained using the training dataset and validated with the validation subset. During training, the model's performance was monitored across epochs to ensure effective learning.
5) Hyperparameter Optimization
Advanced optimization techniques were implemented, including Grid Search for exhaustive parameter exploration and Optuna for efficient Bayesian-based optimization. This dual approach ensured the discovery of optimal configurations for the LSTM model[1].

d. Code Engineering

The final stage encompasses the evaluation and deployment of the model:

1) Model Evaluation
The trained model was evaluated using the test dataset to measure accuracy and reliability. Metrics such as MSE provided quantitative insights into the model's performance.
2) Implementation and Monitoring
Once optimized, the model was deployed for real-time predictions. Its performance was continuously monitored to identify potential degradations, enabling timely adjustments to maintain accuracy.
3) Model Enhancement
Based on monitoring results, the model underwent iterative improvements. These included adding new data, refining hyperparameters, and modifying the architecture to further enhance predictive capabilities.

3. Results and Discussion

Data Collection

The data used in this study is historical weather data that includes several meteorological

variables: minimum temperature (Tn), maximum temperature (Tx), average temperature (Tavg), average humidity (RH_avg), and rainfall (RR) as the target variable. This data is obtained from reliable sources and covers a long enough period to be adequate in building a prediction model. The following is a table of data that has been collected as much as 2556 data:

Table 1. Dataset BMKG

No	Date	Tn	Tx	Tavg	RH_avg	RR	Rain Rate
0	2017-01-01	24.0	33.2	27.3	89	100.2	Very heavy
1	2017-01-02	24.0	31.8	26.9	89	58.8	Heavy
2	2017-01-03	25.0	31.0	26.4	93	14.2	Light
3	2017-01-04	0.0	31.2	26.5	87	90.8	Heavy
4	2017-01-05	24.0	31.4	27.8	85	0.0	Moderate
...	...	...	...	...	...	...	...
2555	2023-12-31	26.0	34.2	29.4	81	12.7	Light

Data Cleaning

A cleaning process was performed to handle missing values, duplication, and anomalies (outliers). In addition, normalization of the input data is performed so that all features are on a uniform scale using MinMaxScaler, so that the model can learn more effectively and stably.

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0, 1))
features_scaled = scaler.fit_transform(features)
```

Figure 2. Data Cleaning

Feature Engineering

The features used in the model include minimum temperature (Tn), maximum temperature (Tx), average temperature (Tavg), and average humidity (RH_avg) variables, with rainfall (RR) as the prediction target. This feature engineering process is carried out to ensure the relevance and efficiency of the features in improving the performance of the model in predicting rainfall.

```
features = df[['Tn', 'Tx', 'Tavg', 'RH_avg','RR']].values
target = df['RR'].values
```

Figure 3. Feature Engineering

Data Splitting

The dataset is divided into two main parts, namely training data (80%) and test data (20%). This division aims to ensure the model can be trained using sufficient data and tested on data not involved in training to measure the generalization ability of the model to data that has never been seen before.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(features_scaled,
target, test_size=0.2, random_state=42)
```

Figure 4. Data Splitting

The dataset is divided into training data (80%) and test data (20%) using the train_test_split function. In addition, the Reshape function is also used to recreate or reorganize the rows and columns.

```
import numpy as np

X_train = np.reshape(X_train, (X_train.shape[0], 1,
X_train.shape[1]))
X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))
```

Figure 5. Reshape Data

Library Import and Model Selection

The LSTM model is built using the Keras library of tensorflow, with an architecture consisting of one LSTM layer and one Dense layer to predict the target variable (RR). The LSTM layer is used to handle sequential data, which is suitable for time series prediction such as rainfall. The model is trained using the Adam optimizer, which is known for its fast convergence capability in deep learning[25].

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Input
from tensorflow.keras.optimizers import Adam
```

Figure 6. Library

Initial Hyperparameter Selection

At the initial stage, several hyperparameters were selected for the LSTM model including the number of neurons in the hidden layer (units), batch size, number of epochs, and learning rate. These values were determined based on related literature as well as preliminary experimental results that showed promising configurations.

```
def create_model(units=100, learning_rate=0.001):
    model = Sequential()
    model.add(Input(shape=(X_train.shape[1], X_train.shape[2])))
    model.add(LSTM(units))
    model.add(Dense(1))
    optimizer = Adam(learning_rate=learning_rate)
    model.compile(loss='mean_squared_error', optimizer=optimizer)
return model
```

Figure 7. Initial Hyperparameter

The LSTM model is built using Keras with a Sequential approach. This model consists of one LSTM layer and one Dense layer with one neuron to generate rainfall prediction. The model uses the Adam optimizer, which is a gradient-based optimization algorithm commonly used to accelerate the convergence of neural network model training.

Loss Function and Optimizer Selection

The loss function used is the Mean Squared Error (MSE), which is a common metric in regression problems. This function is used to measure how much difference there is between the predicted value of the model and the actual value of the rainfall data of interest[26].

```
model.compile(loss='mean_squared_error', optimizer=optimizer)
```

Figure 8. Loss Function

Training, Validation and Hyperparameter Optimization

a. Hyperparameter Optimization with Grid Search

Hyperparameter optimization is performed to find the best combination of parameters such as number of neurons (units), batch size, number of epochs, and learning rate. The Grid Search method is used to brute-force try various hyperparameter combinations. Grid Search tests all predefined hyperparameter combinations in the parameter space and selects the combination that gives the best prediction results (lowest MSE). This process is extensive and time-consuming, but provides a complete picture of how each parameter combination affects the model performance.

```
from scikeras.wrappers import KerasRegressor
from sklearn.model_selection import GridSearchCV
```

```
    model = KerasRegressor(
        model=create_model,
        units=100,
        batch_size=16,
        epochs=10,
        learning_rate=0.001,
        verbose=1
    )

    param_grid = {
        'units': [50, 100, 150],
        'batch_size': [16, 32, 64],
        'epochs': [10, 20, 30],
        'learning_rate': [0.001, 0.01]
    }

    grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=1,
cv=3, verbose=3)
    grid_result = grid.fit(X_train, y_train)

print(f"Best Hyperparameters from Grid Search: {grid_result.best_params_}")
```

Figure 9. Hyperparameter Optimization Grid Search

This code uses GridSearchCV to find the optimal combination of hyperparameters. The code tries all combinations of the number of neurons, batch size, epochs, and learning rate. After the search is complete, the code displays the best hyperparameter combination selected based on the lowest MSE.

b. Hyperparameter Optimization with Bayesian Optimization (Optuna)

Besides Grid Search, hyperparameter optimization is also performed using Optuna, which is a Bayesian Optimization technique. Optuna works more efficiently by narrowing down the hyperparameter search based on probability. In this way, Optuna can save search time while still finding the best hyperparameter combination.

```
    import optuna
    from sklearn.metrics import mean_squared_error

    def objective(trial):
        units = trial.suggest_int('units', 50, 200)
        learning_rate = trial.suggest_float('learning_rate', 1e-4, 1e-2, log=True)
        epochs = trial.suggest_int('epochs', 10, 50)

        model = create_model(units, learning_rate)
        model.fit(X_train, y_train, epochs=epochs, batch_size=trial.suggest_int('batch_size',
16, 128), verbose=0)

        pred = model.predict(X_test)
        mse = mean_squared_error(y_test, pred)
        return mse

    study = optuna.create_study(direction='minimize')
    study.optimize(objective, n_trials=20)
```

```
    best_params_optuna = study.best_params
print(f"Best Hyperparameters from Optuna: {best_params_optuna}")
```

Figure 10. Hyperparameter Optimization Optuna

This code uses Optuna, which is a Bayesian Optimization technique to find the optimal hyperparameter combination. Compared to Grid Search, Optuna is more efficient as it uses a probabilistic approach. The code displays the best hyperparameters found by Optuna based on the lowest MSE.

Discussion

After hyperparameter optimization is performed using Grid Search and Optuna, the results are analyzed to evaluate the comparative effectiveness of each method in improving the accuracy of the LSTM model for rainfall prediction. Model performance is measured using Mean Squared Error (MSE) and execution time to find out how fast each method finds the optimal hyperparameter. The following is a plot image of the output results on the comparison of grid search and Bayesian optimization hyperparameter optimization methods with Optuna:

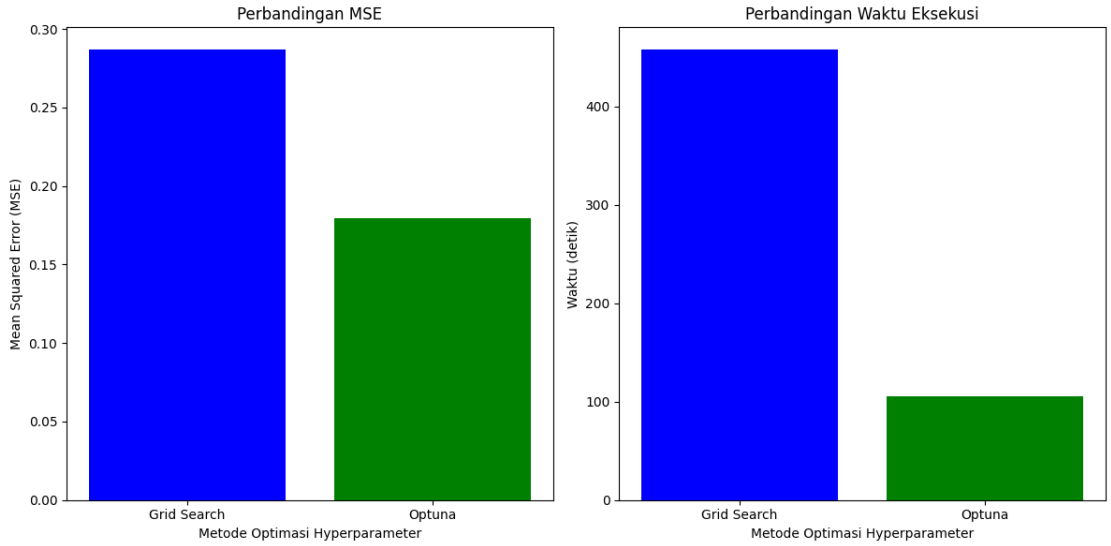

Figure 11. Result Evaluation

The following table summarizes the performance results of the two hyperparameter optimization methods:

Table 2. Performance MSE

Method	MSE	Time
Grid Search	0.286778	457.689601
Bayesian Optimization (Optuna)	0.179578	105.256263

This research shows that hyperparameter optimization significantly improves the performance of the LSTM model in predicting rainfall. The comparison results between Grid Search and Optuna show that Optuna is superior both in terms of accuracy and time efficiency. Grid Search produces an MSE of 0.286778 with an execution time of 457.69 seconds, while Optuna produces an MSE of 0.179578 with an execution time of 105.26 seconds. This result shows that Optuna can find a better hyperparameter configuration in a much shorter time than Grid Search. Overall, the results of this study indicate that hyperparameter optimization via Optuna not only improves model accuracy, but also

provides significant advantages in terms of execution time. Thus, Optuna proved to be more efficient and effective than Grid Search for the task of rainfall prediction using LSTM models.

The experimental results confirm that hyperparameter optimization significantly enhances the predictive accuracy and efficiency of LSTM models. The model optimized with Optuna achieved a lower Mean Squared Error (MSE) of 0.179578 with a significantly reduced execution time of 105.26 seconds, compared to Grid Search, which yielded an MSE of 0.286778 and required 457.69 seconds for optimization. These findings align with prior studies, such as those conducted by Fajriyani et al. [5], which emphasize the effectiveness of Bayesian optimization in improving deep learning model performance. Similarly, Maisat et al. [15] demonstrated that Grid Search, while exhaustive, tends to require more computational time without necessarily achieving the best results. The improved accuracy and computational efficiency of Optuna suggest its suitability for real-time rainfall prediction applications, where rapid and accurate forecasts are crucial for decision-making in agriculture and disaster mitigation.

4. Conclusion

This study confirms that hyperparameter optimization is essential for improving the accuracy and efficiency of rainfall prediction using LSTM models. The results demonstrate that Bayesian Optimization (Optuna) is more effective than Grid Search in finding the optimal hyperparameter configuration while reducing computational cost. The findings contribute to the ongoing research in deep learning-based weather forecasting by providing a comparative evaluation of hyperparameter tuning techniques. Future research could explore the integration of additional meteorological variables such as air pressure and wind direction or investigate alternative optimization techniques, including heuristic-based approaches, to further enhance model performance. The implication of these results is an improved rainfall prediction capability that can be applied to weather prediction systems, with great impact on disaster mitigation and the agricultural sector. Despite the promising results, this study is limited to the use of only two optimization methods and does not address other variations in the LSTM structure. For future research, further exploration on other optimization methods and integration with other variables that affect rainfall is expected. In addition, the development of more complex LSTM models can improve prediction accuracy for real-world applications.

References

[1] Afis Julianto, Andi Sunyoto, and Ferry Wahyu Wibowo, "Optimasi Hyperparameter Convolutional Neural Network Untuk Klasifikasi Penyakit Tanaman Padi," *Tek. Teknol. Inf. dan Multimed.*, vol. 3, no. 2, pp. 98–105, 2022, doi: 10.46764/teknimedia.v3i2.77.

[2] A. S. Agung, A. A. Fauzi, A. A. Nur Risal, and F. Adiba, "Implementasi Teknik Data Mining terhadap Klasifikasi Data Prediksi Curah Hujan BMKG Di Sulawesi Selatan," *J. Tekno Insentif*, vol. 17, no. 1, pp. 22–23, 2023, doi: 10.36787/jti.v17i1.955.

[3] M. H. Al-Areef and K. Saputra S, "Analisis Sentimen Pengguna Twitter Mengenai Calon Presiden Indonesia Tahun 2024 Menggunakan Algoritma LSTM," *J. SAINTIKOM (Jurnal Sains Manaj. Inform. dan Komputer)*, vol. 22, no. 2, p. 270, 2023, doi: 10.53513/jis.v22i2.8680.

[4] J. Badriyah, A. Fariza, and T. Harsono, *Prediksi Curah Hujan Menggunakan Long Short Term Memory*, vol. 6, no. 3. digilib.unila.ac.id, 2022. doi: 10.30865/mib.v6i3.4008.

[5] N. Fajriyani, E. E. Pratama, and R. Septiriana, "Optimasi Hyperparameter pada Neural Network (Studi Kasus: Identifikasi Komentar Cyberbullying Instagram)," *J. Edukasi dan Penelit. Inform.*, vol. 9, no. 2, p. 339, 2023, doi: 10.26418/jp.v9i2.68319.

[6] R. Aprianto, P. A. D. Puspitasari, S. Fitriyanto, and ..., "Analisis Potensi Bencana Banjir Berdasarkan Hasil Prediksi Curah Hujan di Kabupaten Sumbawa," *Titian Ilmu J. ...*, 2024, [Online]. Available: http://journal.unuha.ac.id/index.php/JTI/article/view/3436%0Ahttp://journal.unuha.ac.id/index.php/JTI/article/download/3436/967

[7] M. B. Arya Darmawan, F. Dewanta, and S. Astuti, "Analisis Perbandingan Algoritma Decision Tree, Random Forest, dan Naïve Bayes untuk Prediksi Banjir di Desa Dayeuhkolot," *TELKA - Telekomun. Elektron. Komputasi dan Kontrol*, vol. 9, no. 1, pp. 52–61, 2023, doi: 10.15575/telka.v9n1.52-61.

[8] R. A. Asmara, Arief Prasetyo, Siska Stevani, and R. I. Hapsari, "Prediksi Banjir Lahar Dingin pada Lereng

Merapi menggunakan Data Curah Hujan dari Satelit," *J. Inform. Polinema*, vol. 7, no. 2, pp. 35–42, 2021, doi: 10.33795/jip.v7i2.494.

[9] N. M. M. Candra Devi, I. P. A. Bayupati, and N. K. A. Wirdiani, "Prediksi Curah Hujan Dasarian dengan Metode Vanilla RNN dan LSTM untuk Menentukan Awal Musim Hujan dan Kemarau," *J. Edukasi dan Penelit. Inform.*, vol. 8, no. 3, p. 405, 2022, doi: 10.26418/jp.v8i3.56606.

[10] M. Musfiroh, D. C. R. Novitasari, P. K. Intan, and G. G. Wisnawa, "Penerapan Metode Principal Component Analysis (PCA) dan Long Short-Term Memory (LSTM) dalam Memprediksi Prediksi Curah Hujan Harian," *Build. Informatics, Technol. Sci.*, vol. 5, no. 1, 2023, doi: 10.47065/bits.v5i1.3114.

[11] A. R. Isnain, H. Sulistiani, B. M. Hurohman, A. Nurkholis, and S. Styawati, "Analisis Perbandingan Algoritma LSTM dan Naive Bayes untuk Analisis Sentimen," *J. Edukasi dan Penelit. Inform.*, vol. 8, no. 2, p. 299, 2022, doi: 10.26418/jp.v8i2.54704.

[12] A. Toha, P. Purwono, and W. Gata, "Model Prediksi Kualitas Udara dengan Support Vector Machines dengan Optimasi Hyperparameter GridSearch CV," *Bul. Ilm. Sarj. Tek. Elektro*, vol. 4, no. 1, pp. 12–21, 2022, doi: 10.12928/biste.v4i1.6079.

[13] M. Yusuf, A. Setyanto, and K. Aryasa, "Analisis Prediksi Curah Hujan Bulanan Wilayah Kota Sorong Menggunakan Metode Multiple Regression," *J. Sains Komput. Inform.*, vol. 6, no. 1, pp. 405–417, 2022.

[14] R. Farikhul Firdaus and I. V. Paputungan, *Prediksi Curah Hujan di Kota Bandung Menggunakan Metode Long Short Term Memory*, vol. 2, no. 3. dspace.uii.ac.id, 2022. doi: 10.54082/jupin.99.

[15] Z. Maisat, E. Darmawan, and A. Fauzan, "Implementasi Optimasi Hyperparameter GridSearchCV Pada Sistem Prediksi Serangan Jantung Menggunakan SVM Implementation of GridSearchCV Hyperparameter Optimization in Heart Attack Prediction System Using SVM," *Unipdu*, vol. 13, no. 1, pp. 8–15, 2023.

[16] Y. Hendra, H. Mukhtar, B. Baidarus, and R. Hafsari, "Prediksi Curah hujan di Kota Pekanbaru Menggunakan lSTM (Long Short Term Memory)," 2021, *ejurnal.umri.ac.id*. doi: 10.37859/seis.v3i2.5606.

[17] Cristianto Sihombing, Agung Hari Saputra, Fitria Puspita Sari, and Aditya Mulya, "Prediksi Curah Hujan di Wilayah DKI Jakarta dengan Model NeuralProphet," *J. Apl. Meteorol.*, vol. 1, no. 2, pp. 9–19, 2023, doi: 10.36754/jam.v1i2.317.

[18] B. Dharma Saputra, L. Hiryanto, and T. Handhayani, "Prediksi Curah Hujan Di Kabupaten Badung, Bali Menggunakan Metode Long Short-Term Memory," *J. Ilmu Komput. dan Sist. Inf.*, vol. 11, no. 2, 2023, doi: 10.24912/jiksi.v11i2.26002.

[19] G. Min-soo, *Deep Learning Bible - 2*. Wikidocs, 2024. [Online]. Available: https://wikidocs.net/179772

[20] M. A. Hasanah, S. Soim, and A. S. Handayani, "Implementasi CRISP-DM Model Menggunakan Metode Decision Tree dengan Algoritma CART untuk Prediksi Curah Hujan Berpotensi Banjir," *J. Appl. Informatics Comput.*, vol. 5, no. 2, pp. 103–108, 2021, doi: 10.30871/jaic.v5i2.3200.

[21] M. Rizki, S. Basuki, and Y. Azhar, "Implementasi Deep Learning Menggunakan Arsitektur Long Short Term Memory(LSTM) Untuk Prediksi Curah Hujan Kota Malang," *J. Repos.*, vol. 2, no. 3, pp. 331–338, 2020, doi: 10.22219/repositor.v2i3.470.

[22] P. Sugiartawan and S. G. Santoso, "Multivariate Forecasting Curah Hujan Menggunakan Algoritma LSTM Di Kota Denpasar," *Semin. Nas. Corisindo*, pp. 580–585, 2022, [Online]. Available: https://corisindo.stikom-bali.ac.id/penelitian/index.php/semnas/article/view/129

[23] S. Cumel, David Zamri, Rahmaddeni, "Perbandingan Metode Data Mining untuk Prediksi Banjir dengan Algoritma Naïve Bayes dan KNN," *SENTIMAS Semin. Nas. Penelit. dan …*, pp. 40–48, 2022, [Online]. Available: https://journal.irpi.or.id/index.php/sentimas/article/view/353%0Ahttps://journal.irpi.or.id/index.php/sentimas/article/download/353/132

[24] S. Mujilahwati, M. Sholihin, and R. Wardhani, "Optimasi Hyperparameter TensorFlow dengan Menggunakan Optuna di Python: Study Kasus Klasifikasi Dokumen Abstrak Skripsi," *J. Media Inform. Budidarma*, vol. 5, no. 3, p. 1084, 2021, doi: 10.30865/mib.v5i3.3090.

[25] L. Wulandari, "Optimisasi Algoritma Xgboost Untuk Prediksi Hasil Pemilu," *J. Dunia Data*, vol. 1, no. 5, pp. 1–16, 2024, [Online]. Available: http://www.portaldata.org/index.php/duniadata/article/view/100

[26] S. Setiyaris, M. A. Hariyadi, and C. Crysdian, "Prediksi Curah Hujan Bulanan Berdasrkan Parameter Cuaca Menggunakan Jaringan Saraf Tiruan Levenberg Marquardt," *J. Media Inform. Budidarma*, vol. 7, no. 3, p. 1125, 2023, doi: 10.30865/mib.v7i3.6328.