Vol. 6 No. 1, 2025, pp. 57-70 **Jurnal Pepadun**

DOI: 10.23960/pepadun.v6i1.256

Rainfall Prediction Using Long Short-Term Memory Method (Case Study: Jambi City)

[*1]Ayu Indryani, [2]Ulfa Khaira and [3]Mutia Fadhila Putri

[1,2,3] Department of Information Systems, Faculty of Science and Technology, Universitas Jambi, Jambi, Indonesia.
e-mail:[*1]**ayuindriyani213@gmail.com**, [2]ulfakhaira@unja.ac.id, [3]mutia.fadhila@unja.ac.id

Abstract *- Rainfall prediction plays a crucial role in various fields, including meteorology, agriculture, and disaster mitigation. However, the accuracy of such predictions is often affected by missing values in historical rainfall data. To address this issue, this study employs three interpolation methods—linear, quadratic, and quadratic spline—to fill in the missing values. These values are then used for rainfall prediction in Jambi City using the LSTM model. The rainfall data used in this study were obtained from the BMKG Online Data portal, covering the period from 2016 to 2024. The preprocessing stages included data cleaning, applying interpolation methods to address missing values, normalizing the data, and splitting the dataset into training and testing sets. The LSTM model was developed with various hyperparameter configurations to achieve optimal performance. The model's performance was evaluated using the Root Mean Square Error (RMSE) metric to assess the accuracy of the predictions. The experimental results show that linear interpolation yields the best performance, with RMSE Train and Test values of 13.0661 and 13.1388, respectively. The model, configured with 50 neurons, 75 epochs, a batch size of 32, and using the RMSprop optimizer, demonstrated improved prediction accuracy compared to other interpolation methods. This research confirms the importance of proper data preprocessing in improving the accuracy of time series prediction models. With optimal handling of missing values, LSTM-based prediction models can produce more accurate rainfall forecasts, benefiting various sectors that depend on meteorological data. Finally, to facilitate forecasting for non-programmer users, a web-based Graphical User Interface (GUI) was developed using the Streamlit library.*

Keywords**:** *Interpolation; LSTM; Rainfall Prediction; RMSE; Time-Series Forecasting.*

1. INTRODUCTION

Indonesia is a tropical country because it lies on the equator [1]. This tropical climate makes Indonesia vulnerable to changes in weather patterns, particularly rainfall [2]. Weather refers to atmospheric conditions observed over a short period or within a limited area [3]. Rainfall, one of the key weather elements, is defined as the height of rainwater collected in a rain gauge placed on a flat, non-absorbent surface. A greater measured rainwater height indicates higher rainfall frequency or intensity in the area. [4]. Rainfall plays a crucial role in daily life, influencing sectors such as agriculture, infrastructure, and water resource management [5]. However, due to its fluctuating nature, rainfall remains difficult to predict accurately [6]. In Indonesia, the Meteorology, Climatology, and Geophysics Agency (BMKG) is responsible for providing weather forecasts and climate data, including for Jambi City. However, rainfall data often contains missing values, which can reduce the accuracy of prediction models if not handled properly [7].

Interpolation is one of the techniques commonly used to handle missing values [8]. This technique estimates missing values by filling in the data based on patterns between known points [9]. In this study, three interpolation methods are applied to address missing values in the daily rainfall data of Jambi City: linear interpolation, quadratic interpolation, and quadratic spline interpolation. Linear interpolation uses a first-degree polynomial to estimate the value of a point along a straight line between two known points [10]. Quadratic interpolation, on the other hand, uses a second-degree polynomial to estimate values between three points. Quadratic spline interpolation is an approximation method that estimates values between two points, x_n and x_{n+1}, by assuming a quadratic polynomial function between them [11]. The goal of using these interpolation techniques is to produce a clean and high-quality dataset that can improve the accuracy of rainfall prediction using the Long Short-Term Memory (LSTM) method.

Vol. 6 No. 1, 2025, pp. 57-70 **Jurnal Pepadun**

DOI: 10.23960/pepadun.v6i1.256

Recent studies show that predictions using deep learning have achieved accuracy levels exceeding 85% [12]. Among deep learning models frequently used for time series forecasting are Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and LSTM [13]. CNN is known for its strength in feature extraction and classification tasks [14], and RNN performs better with sequential data but often suffers from the vanishing gradient problem [15]. LSTM addresses this limitation through its gating mechanisms, namely the forget gate, input gate, and output gate, making it more effective in capturing long-term dependencies in data [16]. Research by [17] indicates that LSTM achieved an RMSE of 4.21%, which is slightly lower than the RMSE of the RNN, which is 4.26%. This result confirms LSTM's superiority in weather prediction. Therefore, LSTM was chosen in this study to enhance the accuracy of rainfall prediction in Jambi City.

Although LSTM has demonstrated high effectiveness in various forecasting applications, many previous studies have paid limited attention to handling missing values in datasets. For example, studies by [18] and [19] focused primarily on developing model architectures without addressing missing value processing techniques. This limitation indicates the need to integrate optimal missing value handling methods before applying prediction models such as LSTM to improve prediction accuracy. The present study aims to fill this gap by evaluating three interpolation techniques and identifying the most suitable one to support rainfall prediction. The findings are expected to contribute significantly to time series data processing and improve rainfall prediction accuracy in Jambi City.

2. RESEARCH METHODOLOGY

A clear research framework is essential to guide the implementation of this study. It outlines the structured steps taken to address the research problem, as illustrated in Figure 1.

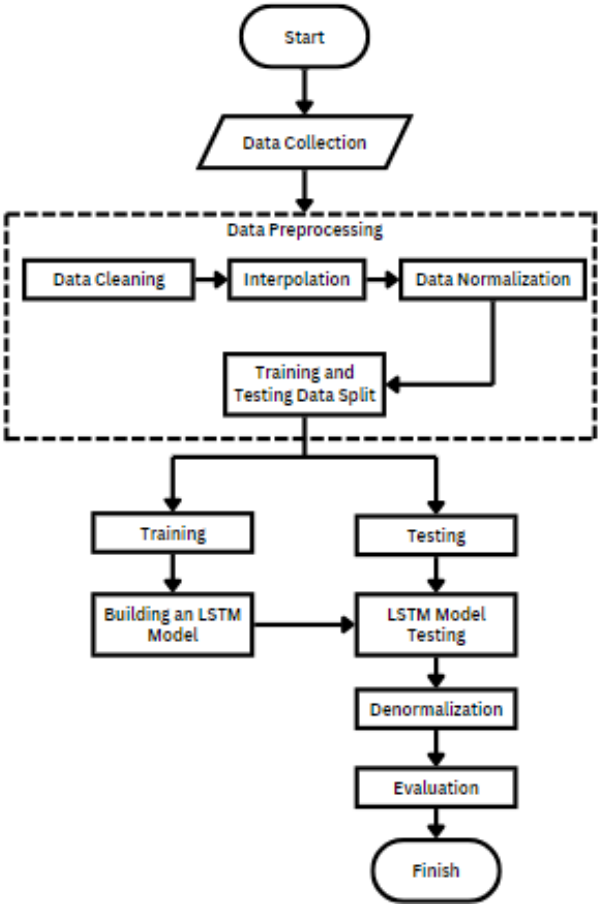

Figure 1. Research stages.

Vol. 6 No. 1, 2025, pp. 57-70 **Jurnal Pepadun**

DOI: 10.23960/pepadun.v6i1.256

2.1. Data Collection

The rainfall data used in this study was obtained from the BMKG Online Data Portal (https://dataonline.bmkg.go.id/home). The dataset consists of daily rainfall records from 2016-2024 in Jambi City, comprising a total of 3288 data points. The data is measured in millimeters (mm). Accurate and comprehensive data collection is essential as it influences the quality of the predictions generated by the model.

2.2. Data Preprocessing

Data preprocessing is an essential stage in preparing the dataset for the rainfall prediction model. Preprocessing aims to enhance data quality by addressing issues that can impact model accuracy, including missing data, invalid data, and disparate value scales. Some of the main steps in data preprocessing applied in this study include data cleaning, interpolation, data normalization, as well as training and testing data split.

2.2.1. Data Cleaning

During the data cleaning process, missing values in the rainfall dataset were identified. In this case, the values 8888 (indicating unmeasured rainfall or values <0.5) and 9999 (representing missing or faulty data) must be addressed. These values were then replaced with NaN (Not a Number) to facilitate further analysis. This transformation allows interpolation techniques to be applied more effectively.

2.2.2 Interpolation

Interpolation was used to handle the missing values identified in the dataset. After converting the invalid values (8888 and 9999) into NaN, three interpolation techniques were applied: linear interpolation, quadratic interpolation, and quadratic spline interpolation. These techniques were applied to ensure the dataset was complete and suitable for model training and evaluation.

a. Linear interpolation estimates missing values using two adjacent known data points. This method offers a straightforward and efficient approach.
b. Quadratic interpolation involves three points to provide a more refined estimate compared to the linear method.
c. Quadratic spline interpolation applies a second-degree polynomial function between each pair of data points, resulting in a smoother and more accurate approximation.

2.2.3 Data Normalization

The data was normalized using the Min-Max Scaling method to transform the original range of values into a uniform scale between 0 and 1. This process ensures consistency across all data values, both before and after the transformation. The Min-Max normalization formula is expressed in Formula 1. This normalization technique helps the model to converge faster and perform better during training by eliminating scale disparities among input features

$$X' = \frac{X - Xmin}{Xmax - Xmin} \quad (1)$$

Description: X' = Normalized value; X = Actual data value; Xmax = Maximum value in the dataset; Xmin = Minimum value in the dataset

2.2.4 Training and Testing Data Split

The dataset was divided into two categories: training data and testing data. The training data was used to build the model, while the testing data was used to evaluate the model's performance. In this study, the data split was set at a ratio of 80:20. That is, 80% of the dataset was allocated for training the model, and the remaining 20% was used for testing. This ratio was selected based on trial results and further consideration after conducting experiments with different proportions of training and testing data. This data division ensures that

Vol. 6 No. 1, 2025, pp. 57-70 **Jurnal Pepadun**

DOI: 10.23960/pepadun.v6i1.256

the model learns effectively from a substantial portion of the data while being evaluated on previously unseen data to assess its generalization ability.

2.3 Building an LSTM Model

The model used to predict rainfall in this study was a LSTM network, with all missing values previously filled using interpolation techniques. The model construction involved setting the initial weights for the forget gate, input gate, cell state, and output gate, represented as (Wf, Wi, Wct, Wo), along with their corresponding bias values (bf, bi, bct, bo). LSTM consists of several gates, each built using sigmoid layers and pointwise operations. These gates function to control the flow of information, determining which data is retained or discarded. The sigmoid function produces outputs in the range [0, 1], where a value of 0 indicates that information is blocked, and a value of 1 means that information is allowed to pass through.

2.4 LSTM Model Testing

Model testing was conducted to evaluate the predictive performance of the trained LSTM model. At this stage, the model was applied to the training and testing dataset to generate rainfall predictions. The predicted values were then compared with the actual observed values to measure model accuracy and performance.

2.5 Denormalization

Denormalization is the process of converting the predicted values back to their original scale. This step is essential because the input data used during model training was normalized in advance. By denormalizing the predicted output, the results can be interpreted in their actual units (millimeters), making them meaningful for practical evaluation and comparison with actual rainfall data. The denormalization calculation is shown Formula 2.

$$d = d'(\max - \min) + min \quad (2)$$

Description: d = Denormalized value (actual rainfall prediction); d' = Normalized predicted value; max = Maximum actual data value; min = Minimum actual data value

2.6 Evaluation

Model evaluation was conducted to assess the accuracy and performance of the LSTM model in predicting rainfall in Jambi City. In this study, the evaluation metric used was Root Mean Square Error (RMSE), which is commonly applied in time series prediction, particularly for rainfall forecasting. The RMSE measures the average magnitude of the error between predicted and actual values by taking the square root of the average squared differences. A lower RMSE value indicates higher model accuracy. The RMSE formula is shown Formula 3.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n}} \quad (3)$$

Description: Y_i = Actual value; $\hat{Y}_i$ = Forecasted/predicted value; n = Number of data points

2.7 Implementation of Graphical User Interface (GUI)

A Graphical User Interface (GUI) is a user-friendly platform that enables interaction between users and software applications [20]. After completing the data preprocessing and evaluating the LSTM model's predictions, a GUI was developed using Streamlit to simplify the forecasting process. The GUI was built based on the experimental results obtained in previous stages. It produces a web-based system that allows users to input time series rainfall data, which is then automatically processed and predicted by the system. This interface makes the forecasting process more accessible and interactive for end users.

Vol. 6 No. 1, 2025, pp. 57-70 **Jurnal Pepadun**

DOI: 10.23960/pepadun.v6i1.256

3. RESULTS AND DISCUSSION

3.1. Data Collection

The dataset used in this study was obtained from the BMKG through its official online data portal (https://dataonline.bmkg.go.id/home). This dataset contains daily rainfall records for Jambi City from 2016 to 2024, consisting of a total of 3288 data entries. All rainfall values are measured in millimeters (mm), which is the standard unit for rainfall measurement. The collected dataset serves as the primary input for the rainfall prediction model developed in this study using the LSTM method. Figure 2 shows the Python source code used to import the required libraries for data analysis and modeling.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import tensorflow as tf
from tensorflow.keras import Input
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
```

Figure 2. Source code import library.

3.1.1 Reading Rainfall Dataset

At this stage, the daily rainfall dataset for Jambi City is read using the Pandas library in Python. The dataset is stored in a CSV (Comma-Separated Values) file format, which is commonly used for storing tabular data. Figure 3 displays the source code used to read and load the dataset. The result of the source code shown in Figure 3 is presented in Figure 4, which displays the daily rainfall dataset for Jambi City from 2016 to 2024. This dataset serves as the primary data used in this study.

```
# read the datasets
df = pd.read_csv ('dataset_ch_fix.csv')
```

Figure 3. Source code for reading the rainfall dataset.

	Tanggal	RR
0	01-01-2016	1.0
1	02-01-2016	NaN
2	03-01-2016	3.7
3	04-01-2016	8888.0
4	05-01-2016	0.5
...	...	...
3283	27-12-2024	0.0
3284	28-12-2024	6.1
3285	29-12-2024	0.0
3286	30-12-2024	34.2
3287	31-12-2024	7.8

3288 rows × 2 columns

Figure 4. Rainfall dataset 2016-2024.

Vol. 6 No. 1, 2025, pp. 57-70 **Jurnal Pepadun**

DOI: 10.23960/pepadun.v6i1.256

3.2. Data Preprocessing

3.2.1 Data Cleaning

In the data cleaning stage, the first step is identifying missing values in the rainfall dataset. Specifically, the value 8888 indicates unmeasured rainfall or rainfall less than 0.5 mm, while the value 9999 signifies that the data has not been recorded or that the measuring instrument was malfunctioning. These values cannot be directly used in analysis because they may introduce bias or errors in statistical calculations. Therefore, these values are transformed into NaN (Not a Number) to allow further processing using interpolation methods. The source code used for this process is shown in Figure 5.

```
# Replace values 8888 and 9999 with NaN
df['RR'] = df['RR'].replace([8888, 9999], np.nan)
```

Figure 5. Source code data cleaning process.

3.2.2 Interpolation

a. Linear Interpolation

In Python, linear interpolation can be performed using the interpolate() function from the Pandas library by specifying the parameter method='linear'.

```
# Fill NaN values with linear interpolation
df['RR'] = df['RR'].interpolate(method='linear')
```

Figure 6. Source code linear interpolation.

b. Quadratic Interpolation

In Python, quadratic interpolation can be applied using the interpolate() function from the Pandas library by specifying method='quadratic'.

```
# Fill NaN value with Quadratic interpolation
df['RR'] = df['RR'].interpolate(method='quadratic')
```

Figure 7. Source code quadratic interpolation.

c. Quadratic Spline Interpolation

In Python, quadratic spline interpolation can be implemented using the interpolate() function from the Pandas library with parameters method='spline' and order=2. The order=2 parameter specifies that the spline should be a second-degree (quadratic) polynomial.

```
# Fill NaN value with Quadratic Spline interpolation
df['RR'] = df['RR'].interpolate(method='spline', order = 2)
```

Figure 8. Source code quadratic spline interpolation.

These three interpolation methods were subsequently used to build a prediction model using the LSTM method. The results of each method were then compared using error metrics such as Root Mean Square Error (RMSE) to determine the most effective approach for handling missing values.

Vol. 6 No. 1, 2025, pp. 57-70 **Jurnal Pepadun**

DOI: 10.23960/pepadun.v6i1.256

3.2.3 Data Normalization

In this study, normalization was performed using the Min-Max Scaling method, which transforms the range of rainfall values into a scale of 0 to 1. The normalization process was implemented using the MinMaxScaler function from the scikit-learn library. Figure 9 shows the source code used.

```
# normalize the dataset
scaler = MinMaxScaler()
df["RR_scaled"] = scaler.fit_transform(df[["RR"]])
```

Figure 9. Source code data normalization.

3.2.4 Training and Testing Data Split

In this study, the dataset was divided into two sets in an 80:20 ratio, where 80% of the data was used to train the model (training set), and the remaining 20% was used to test the model (testing set). Figure 10 shows the source code used:

```
# split into train and test sets
split = int(len(df) * 0.8)
train_data = df.loc[:split, ["Tanggal", "RR_scaled"]]
test_data = df.loc[split:, ["Tanggal", "RR_scaled"]]
```

Figure 10. Source code training and testing data split.

3.3. Building an LSTM Model

In the LSTM model, the data needed to be processed into time series sequences before being used for model training. This step aimed to enable the model to recognize historical patterns in the data and produce more accurate predictions. Therefore, the data were divided into input (X) and target (y) based on a specific time step (n_steps), which determined how much historical data was used to predict future values.

```
#  Create Sequence Data
def create_sequences(data, n_steps):
    X, y = [], []
    for i in range(len(data) - n_steps):
        X.append(data.iloc[i:i + n_steps, 1].values)
        y.append(data.iloc[i + n_steps, 1])
    return np.array(X), np.array(y)
n_steps = 30
X_train, y_train = create_sequences(train_data, n_steps)
X_test, y_test = create_sequences(test_data, n_steps)
# Save the appropriate date for prediction
train_dates = train_data["Tanggal"].iloc[n_steps:].values
test_dates = test_data["Tanggal"].iloc[n_steps:].values
```

Figure 11. Source code LSTM data sequences.

The next step was reshaping the data for both the training and testing datasets. This step was necessary because the LSTM model operated on a three-dimensional (3D) input structure, whereas the available data were still in a two-dimensional (2D) format. Therefore, the data were reshaped from 2D format (x, y) into 3D format (x, y, z) by adding an additional dimension (z). This reshaping process was performed using the code shown in Figure 12.

Vol. 6 No. 1, 2025, pp. 57-70 **Jurnal Pepadun**

DOI: 10.23960/pepadun.v6i1.256

```
# Reshape the data to fit the LSTM input (samples, time steps, features)
X_train = X_train.reshape(-1, n_steps, 1)
X_test = X_test.reshape(-1, n_steps, 1)
```

Figure 12. Source code for reshaping data.

The LSTM model used in this study was designed to predict rainfall based on historical data. The model was built using Sequential from Keras, which allows for the sequential addition of layers. The architecture consists of an input layer, an LSTM layer, a dropout layer, and a dense layer. The selection of hyperparameters, including the number of neurons per layer, the number of epochs, and batch size, was determined through extensive experimentation to achieve an optimal balance between accuracy and computational efficiency. The optimal configuration consisted of 50 neurons per layer, 75 epochs, and a batch size of 32. Increasing the number of neurons improved model accuracy but also increased computational costs. Smaller batch sizes resulted in unstable convergence, whereas larger batch sizes negatively impacted the model's generalization capability. The selected combination offered the most effective trade-off between performance and computational efficiency.

```
# build LSTM Model
model = Sequential()
# Use Input layer as the first layer
model.add(Input(shape=(n_steps, 1)))
model.add(LSTM(50, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1))
# Model compilation
from tensorflow.keras.optimizers import RMSprop
model.compile(optimizer=RMSprop(), loss='mse')
```

Figure 13. Source code for building an LSTM model.

The subsequent process involved training the model using the fit method with specific parameters, such as the number of epochs and batch size. The model was trained on the training dataset for 75 epochs, meaning the entire dataset was processed 75 times to update the model's weights and enable it to learn patterns in the rainfall data. Figure 14 presents the source code used in this process, and Figure 15 present the epoch of the training.

```
# Train the model
history = model.fit(X_train, y_train, epochs=75, batch_size=32)
```

Figure 14. Source code for training the LSTM model.

```
Epoch 72/75
82/82 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2s 27ms/step - loss: 0.0180
Epoch 73/75
82/82 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2s 26ms/step - loss: 0.0177
Epoch 74/75
82/82 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2s 27ms/step - loss: 0.0163
Epoch 75/75
82/82 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2s 26ms/step - loss: 0.0161
```

Figure 15. Epoch training.

Vol. 6 No. 1, 2025, pp. 57-70 **Jurnal Pepadun**

DOI: 10.23960/pepadun.v6i1.256

3.4. LSTM Model Testing

Model testing was conducted by making predictions on both the training and testing datasets to evaluate the model's performance in predicting rainfall based on the historical patterns it had learned. Figure 16 presents the source code used in this testing process.

```
# Predictions for train and test data
y_train_pred = model.predict(X_train.reshape(-1, n_steps, 1))
y_test_pred = model.predict(X_test.reshape(-1, n_steps, 1))
```

Figure 16. Source code for LSTM model testing.

The source code shown in Figure 16 generates a display of the predictions for both the training and testing datasets. The resulting visualization is presented in Figure 17.

```
82/82 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 32ms/step
20/20 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
```

Figure 17. Prediction of training and testing data.

3.5. Denormalization

The prediction results for the training data (y_train_pred) and the test data (y_test_pred) were converted back to their original scale using 'scaler.inverse_transform'. This step was performed to ensure that the scale of the data used in the evaluation remained consistent with the original scale prior to the normalization process.

```
# Denormalization of prediction results
y_train_pred_original = scaler.inverse_transform(y_train_pred)
y_test_pred_original = scaler.inverse_transform(y_test_pred)
```

Figure 18. Source code data denormalization.

3.6. Evaluation

The next stage was model evaluation, which aimed to measure the performance of the developed model. The evaluation metric used was RMSE. In this study, RMSE was calculated on data that had been returned to its original scale, referred to as denormalized RMSE. Calculating RMSE on the original data allowed the prediction error to be interpreted in actual units. The source code used is shown in Figure 19. The result generated from the source code in Figure 19 was the error percentage calculated using the RMSE after the denormalization process.

```
# Calculate RMSE
rmse_train =
np.sqrt(mean_squared_error(scaler.inverse_transform(y_train.reshape(-1,
1)), y_train_pred_original))
rmse_test =
np.sqrt(mean_squared_error(scaler.inverse_transform(y_test.reshape(-1, 1)),
y_test_pred_original))

print(f"RMSE Train: {rmse_train:.4f}")
print(f"RMSE Test: {rmse_test:.4f}")
```

Figure 19. Source code evaluation.

Vol. 6 No. 1, 2025, pp. 57-70
Jurnal Pepadun

DOI: 10.23960/pepadun.v6i1.256

```
RMSE Train: 13.0661
RMSE Test: 13.1388
```

Figure 20. RMSE results.

Based on the results above, the most optimal RMSE scores were 13.0661 for the training data and 13.1388 for the test data. These RMSE values indicate the degree of deviation between the model's predictions and the actual values in their original scale. A lower RMSE value reflects better model performance in generating accurate predictions. To further evaluate the model's predictive performance, a visualization was carried out using Matplotlib to compare the actual data with the predicted values for both the training and testing datasets.

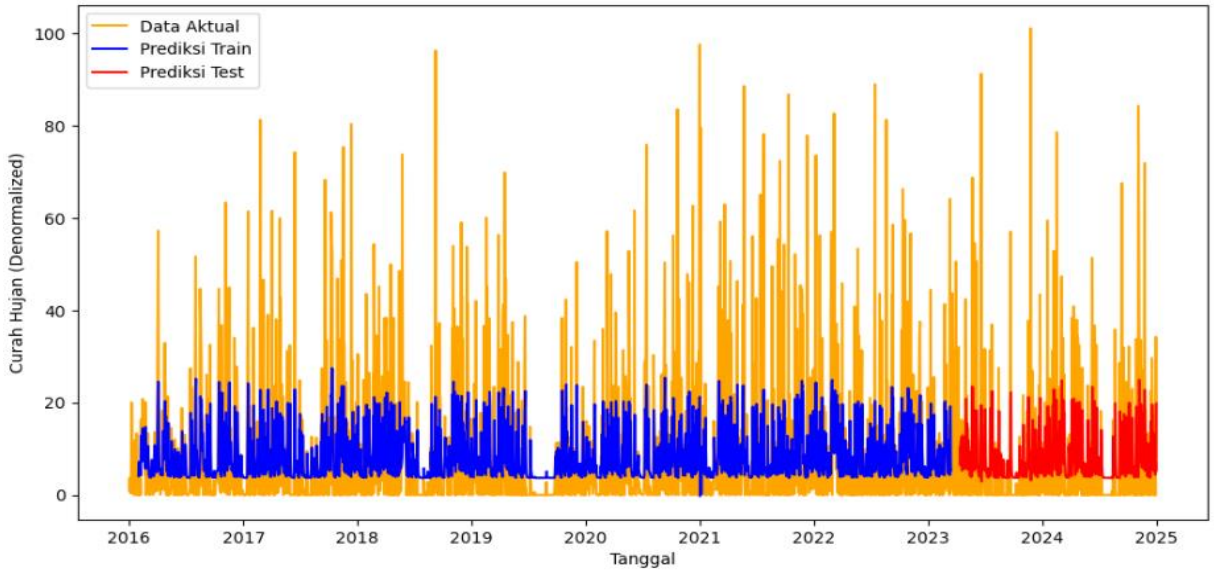

Figure 21. Actual predicted visualization.

The visualization above compares actual rainfall data with the model's predictions after denormalization. This graph illustrates the model's ability to accurately follow the rainfall data pattern. The orange line represents the actual data, the blue line shows the model's predictions using the training data, and the red line shows the model's predictions using the test data. Despite the promising performance of LSTM in rainfall forecasting, the model exhibited limitations in capturing extreme rainfall values. This issue is commonly observed in deep learning models, as they primarily rely on historical patterns, which makes it challenging to predict rare and extreme events accurately. The results showed that LSTM often failed to predict heavy rainfall, indicating that incorporating additional methods, such as integrating external weather data or combining different modeling techniques, could enhance the model's ability to forecast extreme weather.

Table 1. Performance test results.

Interpolation	Epoch	RMSE Train	RMSE Test
Linear	25	13.2667	13.2502
	50	13.1583	13.1775
	75	13.0661	13.1388
	100	13.0542	13.1463
	200	12.9978	13.3208
Quadratic	25	15.6443	14.6474
	50	15.3203	14.4355
	75	14.7959	14.3105
	100	14.7000	14.4892
	200	14.3861	14.1892
Quadratic Spline	25	15.4166	14.4863

Vol. 6 No. 1, 2025, pp. 57-70 **Jurnal Pepadun**

DOI: 10.23960/pepadun.v6i1.256

Interpolation	Epoch	RMSE Train	RMSE Test
	50	15.0268	14.3947
	75	14.7347	14.1844
	100	14.4463	14.0745
	200	14.2627	14.3252

From the performance test results shown in the table above, the lowest performance was achieved with an epoch value of 25 using the quadratic interpolation method, which resulted in a Train RMSE value of 15.6443 and a Test RMSE of 14.6474. Meanwhile, the highest performance was achieved with an epoch value of 75 using the linear interpolation method, which obtained a Train RMSE value of 13.0661 and a Test RMSE value of 13.1388. Linear interpolation outperformed quadratic and quadratic spline interpolation in this study. The results indicate that linear interpolation yields a more stable and consistent dataset for training, resulting in improved generalization. In contrast, quadratic and quadratic spline interpolation introduced additional variance that affected the stability of model predictions. The simplicity of linear interpolation also contributed to its superior performance, as it avoids unnecessary complexity that may lead to overfitting in some instances.

3.7. Implementation of Graphical User Interface (GUI)

In the development of a Graphical User Interface (GUI) for rainfall prediction, researchers utilize the Streamlit framework. Streamlit is a Python-based open-source framework designed to facilitate the creation of interactive web applications in the fields of data science and machine learning. The GUI developed by the researchers for rainfall prediction allows users to input rainfall data obtained from the BMKG Online Data Website. The prediction process is carried out using the LSTM method, and the output includes both the predicted values and the prediction result graphs, as shown in Figures 22 to 27.

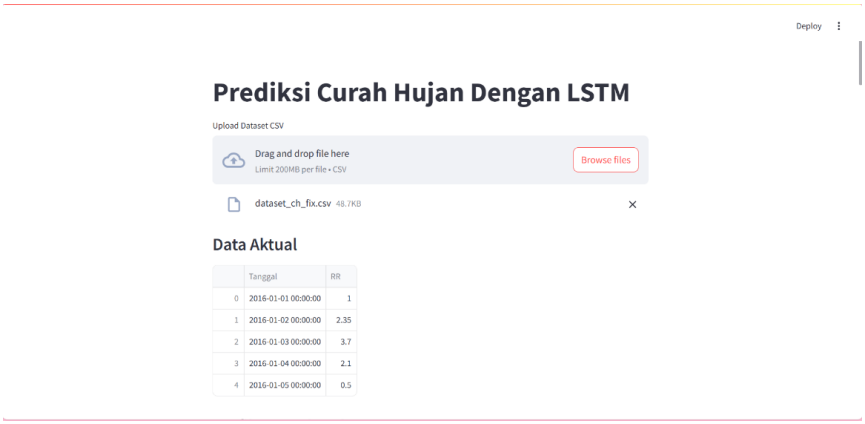

Figure 22. Main GUI display rainfall prediction.

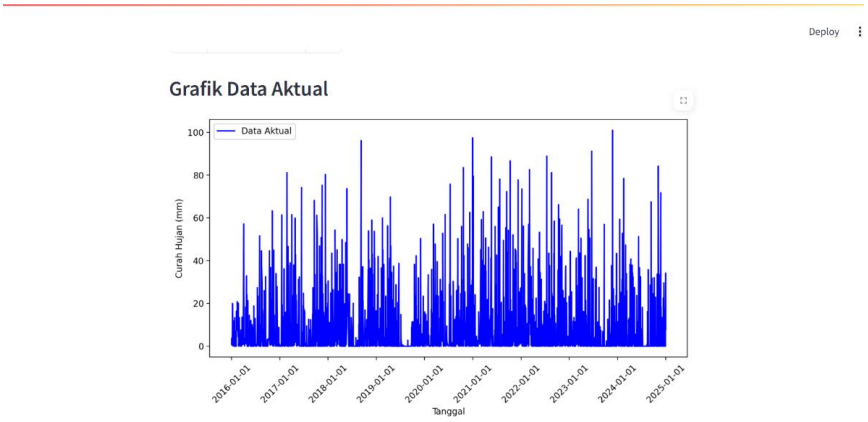

Figure 23. Actual data graph.

Vol. 6 No. 1, 2025, pp. 57-70

DOI: 10.23960/pepadun.v6i1.256

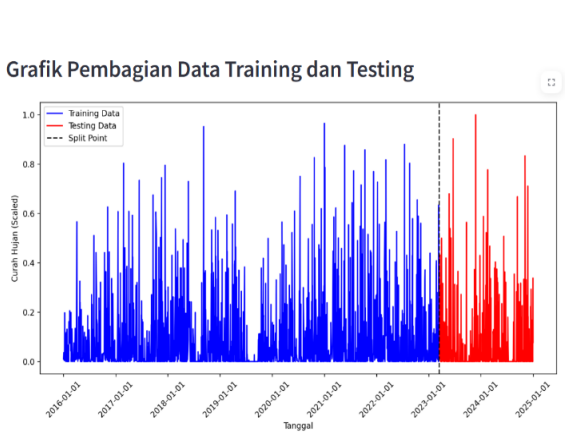

Figure 24. Graph of training and testing data distribution.

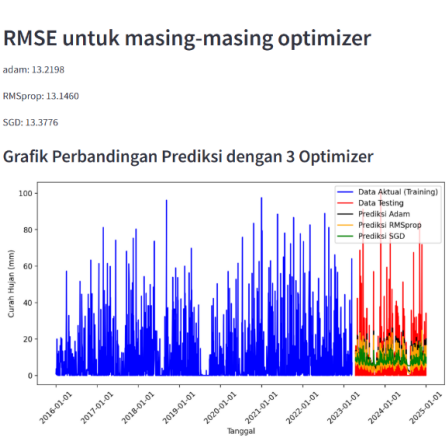

Figure 25. Model evaluation comparison chart.

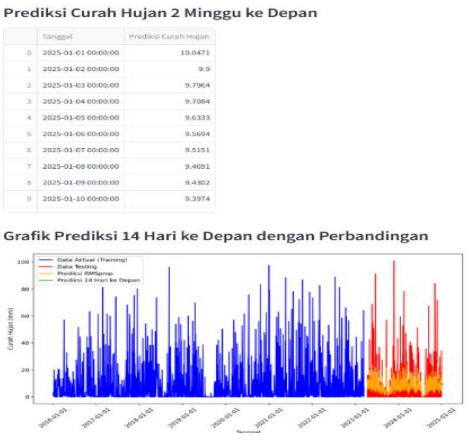

Figure 26. 14-Day forecast comparison chart.

Vol. 6 No. 1, 2025, pp. 57-70 **Jurnal Pepadun**

DOI: 10.23960/pepadun.v6i1.256

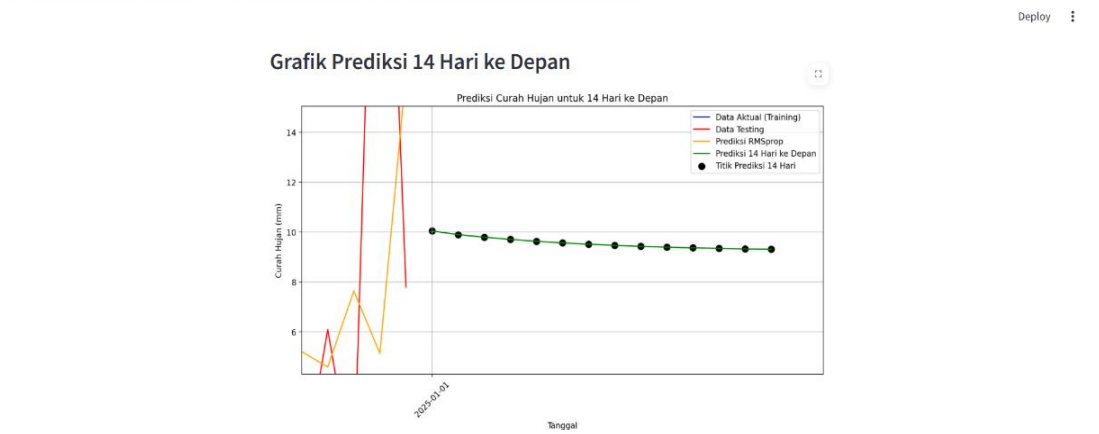

Figure 27. 14-Day forecast graph.

4. CONCLUSIONS

The results of this study indicate that applying interpolation before modeling with LSTM can enhance the accuracy of rainfall prediction. Of the three interpolation methods tested, the linear interpolation method produced the best RMSE value of 13.1388, demonstrating its effectiveness in handling missing values. The developed LSTM model is capable of predicting rainfall based on historical data with a composition of 50 neurons, 75 epochs, and a batch size of 32, using the RMSprop optimizer. The model evaluation shows an RMSE of 13.0661 for the training data and 13.1388 for the test data, indicating a fairly good performance in predicting rainfall in Jambi City. In addition, the web interface developed with Streamlit makes it easier for non-technical users to make real-time predictions. Further research is recommended to integrate additional meteorological variables to improve prediction accuracy further.

LITERATURE

[1] B. Susilo. (2021). *Mengenal Iklim dan Cuaca di Indonesia*. [Online]. Available: https://books.google.co.id/books?id=Cl5zEAAAQBAJ

[2] M. Toyib, T. Decky, K. Pratama, and I. Aqil, "Prediksi Kondisi Cuaca di Kabupaten Bayuwangi Menggunakan Metode LSTM," *Scientica: Jurnal Ilmiah Sains Dan Teknologi*, 2(7), 78–84, 2024.

[3] BMKG. (2024). Buku Saku CLIMATOLOGY. [Online]. Available: https://iklim.bmkg.go.id/bmkgadmin/storage/brosur/Buku%20Saku_KLIMATOLOGI_bnew%20.pdf

[4] R. P. Dhenanta and I. B. Kholifah, "Prediksi Curah Hujan Bulanan Kabupaten Trenggalek Tahun 2022 dan 2023 Menggunakan Metode ARIMA," *Seminar Nasional Official Statistics*, No. 1, pp. 1135-1144, 2022.

[5] Agungnoe. (2024). Pentingnya Memahami Cuaca dan Pengaruhnya ke Pertanian. [Online]. Available: https://ugm.ac.id/id/berita/pentingnya-memahami-cuaca-dan-pengaruhnya-ke-pertanian/

[6] A. Wicaksono, "Pengaruh Fenomena La Nina Terhadap Anomali Curah Hujan Bulanan di Sulawesi Selatan," *Buletin Meteorologi, Klimatologi Dan Geofisika*, 2(3), 35-49, 2022.

[7] T. Ericko, M. D. Lauro, and T. Handhayani, "Prediksi Harga Pangan Di Pasar Tradisional Kota Surabaya Dengan Metode LSTM," *Journal Ilmu Komputer dan Sistem Informasi*, 2023.

[8] A. Widianti and I. Pratama, "Penanganan Missing Values Dan Prediksi Data Timbunan Sampah Berbasis Machine Learning," *Rabit: Journal of Information Technology and Systems Univrab*, vol. 9, no. 2, pp. 242–251, Jul. 2024, doi: 10.36341/rabit.v9i2.4789.

DOI: 10.23960/pepadun.v6i1.256

[9] S. Febrianti, "Pemodelan Autoregressive Integrated Moving Average (Arima) Dengan Missing Data Melalui Metode Interpolasi," Thesis, Universitas Lampung, Bandar Lampung, 2019.

[10] M. R. Ismail, A. Zain, F. Dewantoro, and D. Pratiwi, "Perhitungan Data Curah Hujan yang Hilang dengan Menggunakan Metode Interpolasi Linier," *Jurnal Teknik Sipil*, Vol.4, No.2, 2023.

[11] S. Sofiyani and Y. Permanasari, "Penerapan Metode Cubic Spline Interpolation untuk Menentukan Peluang Kematian pada Tabel Mortalita," *Journal of Mathematical Research*, pp. 29–36, Jul. 2023, doi: 10.29313/jrm.v3i1.1735.

[12] R. S. Budi, R. Patmasari, and S. Saidah, "Klasifikasi Cuaca Menggunakan Metode Convolutional Neural Network," *e-Proceeding Eng*, Vol. 8, No. 5, 2021.

[13] C. Tian, J. Ma, C. Zhang, and P. Zhan, "A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network," *Energies (Basel)*, vol. 11, no. 12, Dec. 2018, doi: 10.3390/en11123493.

[14] E. N. Arrofiqoh and H. Harintaka, "Implementasi Metode Convolutional Neural Network Untuk Klasifikasi Tanaman Pada Citra Resolusi Tinggi," *GEOMATICS*, vol. 24, no. 2, p. 61, Nov. 2018, doi: 10.24895/jig.2018.24-2.810.

[15] A. Arwansyah, S. Suryani, H. Sy, U. Usman, A. Ahyuna, and S. Alam, "Time Series Forecasting Menggunakan Deep Gated Recurrent Units," *Digital Transformation Technology*, vol. 4, no. 1, pp. 410–416, Jun. 2022, doi: 10.47709/digitech.v4i1.4141.

[16] D. Tarkus, S. R. U. A. Sompie, and A. Jacobus, "Implementasi Metode Recurrent Neural Network pada Pengklasifikasian Kualitas Telur Puyuh," *Jurnal Teknik Informatika*, 2020.

[17] F. Irawan, "Prediksi Curah Hujan Menggunakan Recurrent Neural Network (RNN) Dan Long Short-Term Memory (LSTM)," Thesis, Universitas Pakuan, Bogor, 2024.

[18] M. Rizki, S. Basuki, and Y. Azhar, "Implementasi Deep Learning Menggunakan Arsitektur Long Short-Term Memory (LSTM) Untuk Prediksi Curah Hujan Kota Malang," *REPOSITOR*, vol. 2, no. 3, pp. 331–338, 2020.

[19] R. F. Firdaus and I. V. Paputungan, "Prediksi Curah Hujan di Kota Bandung Menggunakan Metode Long Short Term Memory," *Journal of Innovative Research*, vol. 2, no. 3, pp. 453–460, Nov. 2022, doi: 10.54082/jupin.99.

[20] M. M. Muhtadi, M. D. Friyadi, and A. Rahmani, "Analisis Pengujian Graphical User Interface E-Commerce Dengan Menggunakan Katalon Studio," *Sentik*, Vol. 5, No. 1, 2021.