SisInfo

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 5 No. 2 Agustus 2023

Peramalan Data Univariat Menggunakan Metode *Long Short Term Memory*

Helma Syifa Izzadiana, Herlina Napitupulu, Firdaniza Firdaniza
Departemen Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Padjadjaran
Email: helma19001@mail.unpad.ac.id, herlina@unpad.ac.id, firdaniza@unpad.ac.id

Abstrak

Peramalan data univariat mengacu pada kegiatan meramalkan nilai pada data dengan satu variabel independen yang mungkin muncul di masa depan berdasarkan nilai-nilai yang ada di masa lalu. Penelitian ini bertujuan untuk memperoleh model yang dibangun menggunakan pendekatan *deep learning* jenis *supervised learning* yaitu metode *Long Short Term Memory* (LSTM) yang diterapkan pada data univariat. Metode LSTM merupakan pengembangan dari metode *Recurrent Neural Network* (RNN) dengan menambahkan 3 *gate* yang mampu memilih informasi yang dibutuhkan untuk pelatihan sel sehingga mampu mengurangi kemungkinan *exploding gradients* dan *vanishing gradients*. Model dibangun dengan *input layer* LSTM dengan unit sel dan *output dense layer* dengan tambahan *hyperparameter tuning* yang diset menggunakan *optimizer*, fungsi aktivasi sigmoid dan tanh, dan nilai *epoch*. Performa model peramalan diuji menggunakan *mean absolute percentage error* (MAPE).

Kata kunci: Peramalan, *machine learning*, *long short term memory.*

Abstract

Univariate data forecasting refers to the activity of forecasting data with one independent variable that may appear in the future based on values that existed in the past. This study aims to obtain a model built using a supervised learning type deep learning approach, namely the Long Short Term Memory (LSTM) method applied to univariate data. The LSTM method is a development of the Recurrent Neural Network (RNN) method by adding three gates that can select the information needed for cell training to reduce the possibility of exploding gradients and vanishing gradients. The model is built using LSTM layer with cell units and dense layer with an additional hyperparameter tuning set using optimizer, sigmoid *and* tanh *activation function, and number of epochs. The forecasting performance tested using mean absolute percentage error (MAPE).*

Keywords: *Forecasting, machine learning, long short term memory.*

1 PENDAHULUAN

Peramalan data univariat mengacu pada kegiatan meramalkan nilai pada suatu data yang mungkin muncul di masa depan berdasarkan nilai-nilai yang ada di masa lalu. Pada peramalan data univariat hanya terdapat satu variabel independen atau variabel yang tidak bergantung pada variabel lain (Hewamalage *et al.*, 2021).

Peramalan data univariat klasik bekerja sangat baik untuk data yang sedikit (Bandara *et al.*, 2019). Kekurangan peramalan data univariat klasik adalah metode ini tidak memiliki kemampuan untuk dapat menyelesaikan masalah peramalan yang kompleks dengan data yang memiliki jangka

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 5 No. 2 Agustus 2023

waktu panjang. Alasan kekurangan ini adalah karena peramalan data univariat klasik hanya memperhitungkan fitur dan pola yang ada pada data dalam rentang waktu yang terbatas (Hewamalage *et al.*, 2021). Secara umum, teori peramalan univariat klasik telah membantu mengembangkan banyak metode *machine learning* dalam bidang peramalan salah satunya adalah metode *neural network*. (Crone *et al.*, 2011).

Pendekatan *machine learning* atau pembelajaran mesin dapat digunakan untuk meramalkan data univariat. *Machine learning* merupakan alat yang digunakan untuk membuat mesin mempelajari cara menangani data dengan lebih efisien (Batta, 2018). Metode dalam *machine learning* dapat dibagi menjadi *shallow model* dan *deep learning*. *Shallow model* merupakan metode *machine learning* sederhana yang menggunakan *feature engineering* untuk mengubah data mentah menjadi *feature vector* atau sekumpulan data representatif dari data mentah yang dibuat dalam bentuk vektor. Metode *deep learning* dapat belajar melalui fitur yang ada pada data mentah sehingga tidak memerlukan *feature engineering* (Liu & Lang, 2019).

Struktur *deep learning* sangat kompleks dan mengandung sejumlah besar parameter sehingga metode *deep learning* memiliki kemampuan yang lebih kuat daripada *shallow model* dalam mempelajari sifat data. Pada *machine learning* terdapat 2 tipe pembelajaran yaitu *supervised learning* dan *unsupervised learning*. *Supervised learning* merupakan tipe pembelajaran yang menghasilkan fungsi pemetaan *input* ke *output* yang diinginkan, sedangkan *unsupervised learning* merupakan tipe pembelajaran yang mengekstrak fitur dari informasi yang ada pada data tidak berlabel (Nasteski, 2017).

Salah satu metode pada *shallow model* tipe *supervised learning* yang menggunakan serangkaian algoritme dalam mengenali hubungan sekumpulan data dengan meniru cara otak manusia beroperasi adalah *Artificial Neural Network* (ANN) (Batta, 2018). Keuntungan penggunaan ANN adalah metode ini dapat meminimalkan galat menggunakan berbagai algoritme dan memberikan nilai peramalan yang mendekati nilai nyata (Abhishek *et al.*, 2012). Akan tetapi, penggunaan ANN cenderung memakan waktu yang cukup lama dalam pelatihan model, metode ini juga kurang baik digunakan karena cenderung terjebak dalam optimalisasi lokal. Metode ANN menggunakan algoritme *backpropagation through time* (BPTT) dimana metode ini memungkinkan terjadinya *vanishing gradients* dan *exploding gradients*.

Menurut Liu dan Lang (2019), *deep learning* bekerja lebih baik untuk mengolah data dengan jumlah besar. Pada *deep learning*, penekanan utama yang dilakukan adalah arsitektur jaringan, pemilihan *hyperparameter*, dan strategi optimasi. Selain itu, metode *deep learning* dapat secara otomatis mempelajari representasi fitur data mentah dan kemudian hasil keluarannya beroperasi dengan cara *end-to-end* dan praktis. Salah satu metode adaptasi dari ANN yang dikategorikan sebagai *deep learning* tipe *supervised learning* adalah metode *Recurrent Neural Network* (RNN) dan salah satu pengembangan metode RNN yang mengganti algoritme BPTT menjadi mekanisme *gate* adalah *Long Short Term Memory* (LSTM). Hewamalage *et al.* (2021) membandingkan metode RNN, *Gated Recurrent Unit* (GRU) dan metode *Long Short Term Memory* (LSTM). Metode LSTM merupakan metode pengembangan dari metode RNN dengan menambahkan 3 *gate* untuk memodifikasi struktur sel yang ada pada metode RNN. Mengacu pada Hewamalage *et al.* (2021), tambahan 3 *gate* pada metode LSTM mampu menyaring informasi yang dibutuhkan untuk pelatihan sel, sehingga metode LSTM mampu mengurangi kemungkinan terjadinya *vanishing gradients* dan *exploding gradients*. Hewamalage *et al.* (2021) menggunakan metode RNN dan LSTM dengan menambahkan *hyperparameter tuning* yang sesuai untuk mendapatkan performa model yang optimal. Keempat metode ini digunakan

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 5 No. 2 Agustus 2023

untuk meramalkan data *CIF 2016 forecasting competition dataset, NN5 forecasting competition dataset, M3 forecasting competition dataset, M4 forecasting competition dataset, Wikipedia web traffic time series forecasting competition dataset*, dan *tourism forecasting competition dataset*. Pada penelitian Hewamalage *et al.* (2021) ini dihasilkan metode LSTM memberikan akurasi yang lebih baik dibanding metode lainnya yaitu metode RNN dan GRU. Perhitungan akurasi dilakukan menggunakan *The Symmetric Mean Absolute Percentage Error* (SMAPE).

2 KAJIAN PUSTAKA

2.1 *Data Preprocessing*

Mengacu pada Fan *et al.* (2021), *data preprocessing* atau pra-pemrosesan data merupakan kegiatan untuk menyiapkan data sebelum dapat digunakan. *Data preprocessing* mengacu pada teknik-teknik yang dilakukan dengan tujuan untuk meningkatkan kualitas data mentah atau *raw data*. Proses yang dilakukan pada *data preprocessing* adalah *cleaning data* atau pembersihan data, *scaling data* atau menskalakan data, dan *partitioning data* atau mempartisi data.

1. *Cleaning data* atau pembersihan data merupakan kegiatan yang bertujuan untuk menghapus data *outliers* dan mengubah data NaN atau data kosong. Penghapusan *outliers* menggunakan metode perhitungan untuk mendapatkan nilai *outliers* dari keseluruhan dataset, lalu nilai yang memenuhi syarat perhitungan *outliers* akan dihapus (Lou *et al.*, 2022). Data yang hilang atau kosong diisi menggunakan data pembobotan yang dihitung dengan mempertimbangkan perhitungan data yang tidak hilang, imputer KNN (*K-Nearest Neighbour*) membantu dalam imputasi nilai yang hilang (Juna *et al.*, 2022).
2. *Partitioning data* atau partisi data merupakan kegiatan yang bertujuan untuk membagi keseluruhan data menjadi beberapa kelompok data dengan fungsi dan karakteristik yang berbeda. Terdapat beberapa jenis data yang telah dipartisi yaitu data *training*, data *validation*, dan data *testing*. Data *training* digunakan untuk *training* model, data *validation* digunakan untuk validasi performa model, sedangkan data *testing* digunakan untuk meramalkan data pada tahap waktu selanjutnya. Beberapa penelitian hanya menggunakan 2 data partisi, contohnya (Pontoh *et al.*, 2022) yang mempartisi data menjadi data *training* dan *testing* dimana data *testing* memiliki 2 fungsi yaitu sebagai data *validation* dan sebagai data *testing*.
3. *Scaling data* adalah kegiatan untuk mengubah nilai data yang digunakan menjadi nilai interval untuk model peramalan. Terdapat beberapa jenis kegiatan *scaling data*, salah satunya dilakukan dengan bantuan modul MinMaxScaler dari *library* sklearn.preprocessing sehingga data memiliki nilai interval [0,1]. *Scaling data* digunakan agar nilai *input* memiliki interval nilai yang sama dengan nilai *output* karena sifat fungsi sigmoid yang mengeluarkan nilai dalam interval (0,1). Secara matematis, *scaling data* menjadi interval [0,1] dapat dinyatakan dalam Persamaan (1) (Scikit-learn Developers, 2023):

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}, \quad (1)$$

dengan x adalah data aktual yang digunakan, x' adalah data hasil *scaling*, x_{max} adalah data maksimum dari keseluruhan data yang digunakan, dan x_{min} adalah data minimun dari keseluruhan data yang digunakan. Adapun persamaan invers untuk mengembalikan nilai hasil *scaling* setelah peramalan selesai dilakukan baik untuk data aktual yang digunakan maupun data yang dihasilkan dari peramalan dinyatakan dalam Persamaan (2),

$$x = x'(x_{max} - x_{min}) + x_{min} \quad (2)$$

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 5 No. 2 Agustus 2023

2.2 Metode *Long Short Term Memory*

Metode *Long Short Term Memory* (LSTM) diperkenalkan oleh Hochreiter dan Schmidhuber pada tahun 1997 (Falah & Rachmaniah, 2022). Sebagai perbaikan dari metode RNN, LSTM dapat menggambarkan ketergantungan atau *dependencies* dalam data sekuensial sembari mengurangi masalah *vanishing gradients* dan *exploding gradients* (Hewamalage *et al.*, 2021).

Mengacu pada Hewamalage *et al.* (2021), sel pada LSTM memiliki 2 komponen yaitu *the hidden state* dan *internal cell state*. *The hidden state* menyatakan komponen *short term memory* sedangkan *internal cell state* menyatakan *long term memory*. Terdapat 3 mekanisme *gate* dalam LSTM yang berfungsi untuk mengatur konten dalam memori, yaitu *input gate* dan *forget gate* yang mengatur *internal cell state* dan *output gate* yang menghasilkan *output vector* atau *the hidden state*. Mekanisme *gate* ini membantu unit sel memori untuk menyimpan informasi dalam jangka waktu panjang yang membuat gradien kesalahan memiliki nilai yang independen dari perhitungan untuk setiap tahap waktu (Lillicrap dan Santoro, 2019).

Vektor yang dinyatakan sebagai *the hidden state* dari sel dinyatakan sebagai $\boldsymbol{h_t} \in \mathbb{R}^d$ dengan d adalah ukuran atau jumlah unit sel. Simbol $\widetilde{\boldsymbol{C_t}} \in \mathbb{R}^d$ menyatakan vektor *candidate cell state* pada setiap langkah waktu t yang menggambarkan informasi penting yang bertahan untuk meramalkan *cell state* pada interval waktu selanjutnya dan $\boldsymbol{C_t} \in \mathbb{R}^{\boldsymbol{d}}$ menyatakan *cell state* pada setiap langkah waktu t. Vektor *input gate* dinyatakan sebagai $\boldsymbol{i_t} \in \mathbb{R}^d$, vektor *output gate* dinyatakan sebagai $\boldsymbol{o_t} \in \mathbb{R}^d$, dan vektor *forget gate* dinyatakan sebagai $\mathbf{f_t} \in \mathbb{R}^d$, sedangkan simbol $x_t \in \mathbb{R}^m$ dan $\mathbf{z_t} \in \mathbb{R}^d$ adalah *input* dan *output* dari sel untuk setiap langkah waktu *t*. Secara matematis vektor *forget gate*, vektor *input gate*, vektor *candidate cell state*, vektor *output gate*, vektor *cell state*, *the hidden state*, dan *output* sel dinyatakan sebagai berikut:

$$\mathbf{f_t} = \sigma(\mathbf{W_f h_{t-1}} + \mathbf{V_f} x_t + \mathbf{b_f}), \quad (3)$$

dengan $\mathbf{W_f} \in \mathbb{R}^{d \times d}$ dan $\mathbf{V_f} \in \mathbb{R}^{d \times m}$ menyatakan matriks bobot *forget gate* dan $\mathbf{b_f} \in \mathbb{R}^d$ menyatakan *bias vector forget gate*.

$$\mathbf{i_t} = \sigma(\mathbf{W_i h_{t-1}} + \mathbf{V_i} x_t + \mathbf{b_i}), \quad (4)$$

dengan $\mathbf{W_i} \in \mathbb{R}^{d \times d}$ dan $\mathbf{V_i} \in \mathbb{R}^{d \times m}$ menyatakan matriks bobot *input gate* dan $\mathbf{b_i} \in \mathbb{R}^d$ menyatakan *bias vector input gate*.

$$\widetilde{\mathbf{C_t}} = \tanh(\mathbf{W_c h_{t-1}} + \mathbf{V_c} x_t + \mathbf{b_c}), \quad (5)$$

dengan $\mathbf{W_c} \in \mathbb{R}^{d \times d}$ dan $\mathbf{V_c} \in \mathbb{R}^{d \times m}$ menyatakan matriks bobot *internal cell states* dan $\mathbf{b_c} \in \mathbb{R}^d$ menyatakan *bias vector internal cell states*.

$$\mathbf{o_t} = \sigma(\mathbf{W_o h_{t-1}} + \mathbf{V_o} x_t + \mathbf{b_o}), \quad (6)$$

dengan $\mathbf{W_o} \in \mathbb{R}^{d \times d}$ dan $\mathbf{V_o} \in \mathbb{R}^{d \times m}$ menyatakan matriks bobot *output gate* dan $\mathbf{b_o} \in \mathbb{R}^d$ menyatakan *bias vector output gate*.

Fungsi aktivasi yang digunakan dalam metode LSTM adalah fungsi sigmoid yang dinyatakan dengan $\sigma(x)$ dan $\tanh(x)$ digunakan sebagai fungsi aktivasi *output*. Fungsi aktivasi $\tanh(x)$ digunakan untuk mengatur nilai yang melalui jaringan selalu bernilai (-1,1). Fungsi aktivasi tanh dinyatakan dalam Persamaan (7) (Yang *et al.*, 2020).

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \quad (7)$$

Fungsi aktivasi sigmoid digunakan untuk menskalakan nilai *output* $\tanh(x)$ menjadi nilai dengan interval $(0,1)$ dengan nilai *output* tunggal sebagai nilai hasil peramalan. Fungsi aktivasi sigmoid dinyatakan dalam Persamaan (8) (Juna *et al.*, 2022).

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \quad (8)$$

Pada Persamaan (3) dan (4), *forget gate* ($\mathbf{f_t}$) dan *input gate* ($\mathbf{i_t}$) menentukan berapa banyak informasi masa lalu yang dapat digunakan dalam *cell state* ($\mathbf{C_t}$) saat ini dan

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 5 No. 2 Agustus 2023

berapa banyak informasi saat ini yang dapat disebarkan untuk peramalan dalam interval waktu selanjutnya.

Nilai 0 pada *forget gate* ($\mathbf{f_t}$) menunjukkan bahwa tidak ada informasi yang dibawa ke sel saat ini dari sel sebelumnya atau keadaan sel sebelumnya harus benar-benar dilupakan untuk keadaan sel saat ini, sedangkan nilai 1 berarti status sel sebelumnya harus dipertahankan sepenuhnya. Konsep ini diterapkan juga untuk *gate* lainnya yaitu *input gate* ($\mathbf{i_t}$) dan *output gate* ($\mathbf{o_t}$). Nilai diantara 0 dan 1 menyatakan *input gate* ($\mathbf{i_t}$), *forget gate* ($\mathbf{f_t}$), dan *output gate* ($\mathbf{o_t}$) dapat mengontrol nilai status sel saat ini atau dapat menggunakan informasi penting dari status sel sebelumnya untuk meramalkan *candidate cell state* saat ini.

$$\mathbf{C_t} = \mathbf{i_t} \circ \widetilde{\mathbf{C_t}} + \mathbf{f_t} \circ \mathbf{C_{t-1}}, \quad (9)$$

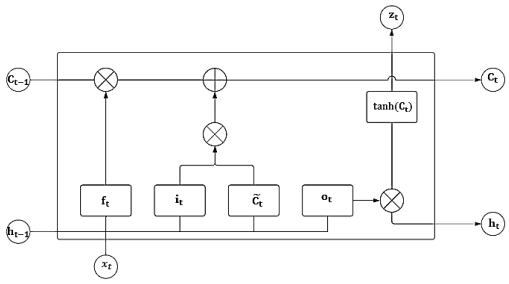
$$\mathbf{h_t} = \mathbf{o_t} \circ \tanh(\mathbf{C_t}), \quad (10)$$

$$\mathbf{z_t} = \mathbf{h_t}, \quad (11)$$

dengan symbol $\circ$ menyatakan Hadamart Product matriks **A** dan **B** dengan ordo m × n untuk seluruh $1 \leq i \leq m$ dan $1 \leq j \leq n$ yang secara matematika dinyatakan dalam Persamaan (12).

$$[\mathbf{A} \circ \mathbf{B}]_{ij} = [\mathbf{A}]_{ij}[\mathbf{B}]_{ij}. \quad (12)$$

Cell state ($\mathbf{C_t}$) dihitung menggunakan *gate* yang telah didapatkan dan *cell state* tahap waktu sebelumnya ($\mathbf{C_{t-1}}$). Selanjutnya, untuk menghitung *the hidden state* ($\mathbf{h_t}$) tahap waktu selanjutnya, fungsi aktivasi yang digunakan adalah fungsi $\tanh(x)$ yang didefinisikan dalam Persamaan (7) dan menghasilkan nilai dalam interval (-1,1). Perbedaan penting antara sel LSTM dan sel RNN sederhana adalah dalam LSTM *output* $\mathbf{z_t}$ sama dengan nilai pada *the hidden state* $\mathbf{h_t}$ dengan simbol $\circ$ menyatakan perkalian elemen untuk *Hadamart product* dan struktur sel LSTM diperlihatkan dalam Gambar 1.



Gambar 1. Struktur sel LSTM.

2.3 *Dense Layer*

Hasil kalkulasi pada sel LSTM menghasilkan matriks dengan ordo $d \times 1$, sehingga untuk membuat *output* memiliki satu nilai hasil peramalan perlu menyatukan semua nilai pada matriks hasil kalkulasi sel LSTM menggunakan *dense layer*. *Dense layer* merupakan perkalian matriks bobot dan *input* dengan tambahan vektor bias yang nantinya dapat dilatih ke *output* (Karim *et al.*, 2019). Secara matematis *output* setiap lapisan *dense layer* didefinisikan dalam Persamaan (13),

$$y = \sigma(\mathbf{W}\mathbf{z_t} + \mathbf{b}) \quad (13)$$

dengan **W** adalah matriks bobot, **b** adalah vektor bias, $\mathbf{z_t}$ merupakan nilai *output* dari proses kalkulasi dalam sel LSTM, dan $\sigma(x)$ merupakan fungsi aktivasi yang didefinisikan dalam Persamaan (8).

2.4 *Hyperparameter Tuning*

Performa model *machine learning* bergantung pada beberapa parameter yang disebut *hyperparameter*. *Hyperparameter* ditetapkan secara manual sebelum *training* model dilakukan guna mendapatkan performa model yang optimal (Hewamalage *et al.*, 2021). Pada metode LSTM, *hyperparameter* yang penting adalah fungsi aktivasi yaitu $\sigma(x)$ dan $\tanh(x)$, jumlah *epoch*, jumlah *hidden layer*, dan *optimizer*. *Optimizer* yang umum digunakan adalah *adaptive moment estimation* (Adam) karena efisien secara komputasi, hanya membutuhkan gradien orde pertama, membutuhkan memori yang kecil, mengubah skala diagonal gradien, dan cocok untuk

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 5 No. 2 Agustus 2023

masalah data atau parameter yang besar (Chang *et al.*, 2019).

2.5 *Mean Absolute Percentage Error*

Mean Absolute Percentage Error (MAPE) merupakan galat yang dihitung berdasarkan nilai absolut kesalahan atau selisih nilai ramalan dan nilai aktual yang dibagi dengan nilai aktual pada periode tersebut dan nilai yang didapat dibagi jumlah data (Krisma *et al.*, 2019). Secara matematis, MAPE didefinisikan dalam Persamaan (14).

$$\text{MAPE} = \frac{100\%}{N}\sum_{t=1}^{N}\frac{|y_t - x_t|}{x_t}, \qquad (14)$$

dengan N menyatakan banyaknya data, y_t menyatakan nilai yang didapat dari hasil peramalan, dan x_t adalah nilai aktual dengan waktu t.

MAPE memiliki interval nilai yang dapat dijadikan dasar pengukuran atau pengambilan keputusan mengenai kemampuan suatu model peramalan, interval nilai tersebut tersaji pada Tabel 1.

Tabel 1. Penerimaan nilai MAPE.

Interval nilai MAPE	**Keterangan**
$\leq 10\%$	Model peramalan sangat baik
$10\% < \text{MAPE} \leq 20\%$	Model peramalan baik
$20\% < \text{MAPE} \leq 50\%$	Model peramalan layak
$> 50\%$	Model peramalan buruk

Sumber: (Lewis, 1982)

MAPE memberikan interval galat yang dihasilkan dari peramalan. Semakin rendah nilai MAPE, kemampuan model peramalan yang digunakan semakin baik dan begitupun sebaliknya.

3 METODE PENELITIAN

Metode yang digunakan dalam penelitian ini adalah metode LSTM dengan diagram alir penelitian ditunjukkan oleh Gambar 2.

Gambar 2. Diagram alir penelitian.

Sebelum diteliti, data yang digunakan melewati 3 tahap data *preprocessing* yaitu *cleaning data*, partisi data, dan *scaling data*. Setelah proses data *preprocessing* selesai, data *training* dan data *testing* akan diramalkan menggunakan model yang telah dibuat dengan metode LSTM dan hasil peramalan tersebut akan diubah nilainya menggunakan invers *scaling* agar data peramalan yang didapatkan memiliki rentang nilai data yang digunakan. Jika nilai MAPE yang dihasilkan sudah memiliki nilai kurang dari 20% atau model peramalan baik, maka proses dilanjutkan dengan meramalkan data *testing* dan

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 5 No. 2 Agustus 2023

dilakukan invers *scaling* agar didapatkan data hasil peramalan untuk waktu yang akan datang.

4 HASIL DAN PEMBAHASAN

4.1 *Data Preprocessing*

Sebelum dilakukan peramalan meggunakan metode LSTM, perlu dilakukan langkah *data preprocessing* terlebih dahulu. Pada penelitian ini digunakan 3 metode *data preprocessing* yaitu *cleaning data*, *partitioning data*, dan *scaling data*.

1. Data *outliers* dihilangkan menggunakan metode perhitungan untuk mendapatkan nilai *outliers* dari keseluruhan dataset. *Code program* penghapusan data *outliers* adalah sebagai berikut:

```
def Remove_Outlier_Indices(df):
    Q1 = df.quantile(0.25)
    Q3 = df.quantile(0.75)
    IQR = Q3 - Q1
    trueList = ~((df < (Q1 - 1.5 * IQR
)) |(df > (Q3 + 1.5 * IQR)))
    return trueList
df1 = pd.DataFrame(df1)
nonOutlierList = Remove_Outlier_Indice
s(df1)
df1 = df1[nonOutlierList]
```

   Selanjutnya, data yang hilang atau kosong diubah nilainya menggunakan data pembobotan yang dihitung dengan mempertimbangkan perhitungan data yang tidak hilang dibantu modul KNNImputer. *Code program* pengisian data kosong adalah sebagai berikut:

```
imputer = KNNImputer(n_neighbors=1000,
weights='distance')
df = imputer.fit_transform(df1)
```

2. Data yang digunakan dipartisi menjadi 2 atau 3 dataset yang berbeda. Partisi data dilakukan sesuai kebutuhan penelitian. *Code program* partisi data adalah sebagai berikut:

```
train_size = int(0.80*len(df))
train_data = df.iloc[0:train_size]
test_data = df.iloc[train_size:]
```

3. *Scaling* data bertujuan untuk mengubah data menjadi bernilai [0,1]. *Scaling data* digunakan agar nilai *input* dan *output* bernilai [0,1] mengikuti sifat fungsi sigmoid yang mengeluarkan nilai *output* (0,1). *Code program scaling* data adalah sebagai berikut:

```
scaler = MinMaxScaler(feature_range = (
0, 1)).fit(train_data)
train_scaled = scaler.transform(train_d
ata)
test_scaled = scaler.transform(test_dat
a)
def create_dataset (X, look_back = 1):
    Xs, ys = [], []
    for i in range(len(X)-look_back):
        v = X[i:i+look_back]
        Xs.append(v)
        ys.append(X[i+look_back])
    return np.array(Xs), np.array(ys)
X_train, y_train = create_dataset(train
_scaled,14)
X_test, y_test = create_dataset(test_sc
aled,14)
```

4.2 Pembuatan Model

Pembuatan model dilakukan dengan bantuan *library* TensorFlow dengan modul keras yang ditambahkan dengan *layer* LSTM layer dan *Dense* layer. Model dibuat menggunakan kombinasi *hyperparameter tuning* yang diset menggunakan *optimizer*, epoch dan unit sel. Setelah melalui pelatihan model, selanjutnya dilakukan kegiatan peramalan data menggunakan model yang telah diperoleh. Untuk mendapatkan hasil peramalan data maka perlu dilakukan rangkaian kalkulasi yang terjadi dalam satu struktur sel LSTM, lalu *output* sel LSTM dihitung menggunakan fungsi sigmoid dalam *dense layer*. Langkah selanjutnya adalah menghitung hasil yang didapat dari *dense layer* menggunakan invers MinMaxScaler agar didapatkan hasil akhir peramalan. *Code program* model peramalan menggunakan metode LSTM dengan tambahan *dense layer* adalah sebagai berikut:

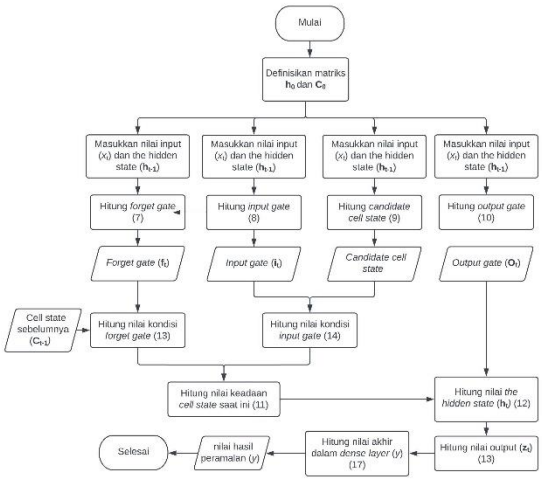
```
hunits = 1000
model = tf.keras.models.Sequential()
model.add(LSTM(hunits, input_shape = [X_tr
ain.shape[1], X_train.shape[2]]))
model.add(Dense(1, activation='sigmoid'))
```

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 5 No. 2 Agustus 2023

```
model.compile(optimizer=tf.keras.optimizer
s.Adam(), loss=tf.keras.losses.Huber(), me
trics="mae")
history = model.fit(X_train, y_train, vali
dation_data=(X_test, y_test), shuffle = Fa
lse, verbose = 2, epochs = 700, batch_size
=500)
```

4.3 Komputasi Matematika dalam Sel *Long Short Term Memory*

Proses kalkulasi yang terjadi dalam satu struktur sel LSTM diperlihatkan dalam Gambar 3.



Gambar 3. Proses perhitungan peramalan menggunakan metode LSTM.

1. Langkah pertama adalah mendefinisikan vektor *the hidden state* ($\mathbf{h_0}$) dan *cell state* ($\mathbf{C_0}$). Pada pendefinisian awal, seluruh elemen dari kedua vektor ini bernilai nol.
2. Data x_t masuk sebagai *input* data dan $\mathbf{h_{t-1}}$ masuk sebagai *the hidden state*, data tersebut diolah dalam *forget gate* menggunakan fungsi sigmoid dengan vektor $\mathbf{W_f}, \mathbf{V_f}$, dan $\mathbf{b_f}$. Matriks-matriks tersebut memiliki ordo yang sesuai dengan jumlah unit (d) yang digunakan selama pelatihan model. Secara matematis keadaan dalam *forget gate* diperlihatkan dalam Persamaan (3), didapat

$$\mathbf{f_t} = \sigma(\mathbf{W_f h_{t-1}} + \mathbf{V_f} x_t + \mathbf{b_f})$$

dengan

$$\mathbf{V_f} = \begin{bmatrix} v_{f_{11}} \\ v_{f_{21}} \\ \vdots \\ v_{f_{d1}} \end{bmatrix}$$

$$\mathbf{W_f} = \begin{bmatrix} w_{f_{11}} & w_{f_{12}} & \dots & w_{f_{1d}} \\ w_{f_{21}} & w_{f_{22}} & \dots & w_{f_{2d}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{f_{d1}} & w_{f_{d2}} & \dots & w_{f_{dd}} \end{bmatrix}$$

$$\mathbf{b_f} = \begin{bmatrix} b_{f_1} \\ b_{f_2} \\ \vdots \\ b_{f_d} \end{bmatrix}$$

3. Setelah didapatkan hasil *forget gate*, nilai *forget gate* dikalikan dengan kondisi sel sebelumnya ($\mathbf{C_{t-1}}$), sehingga didapatkan

$$\mathbf{C_t^f} = \mathbf{f_t} \circ \mathbf{C_{t-1}}. \qquad (15)$$

4. Data x_t masuk kembali sebagai *input* data dan $\mathbf{h_{t-1}}$ masuk sebagai *the hidden state*, data tersebut diolah dalam *input gate* menggunakan fungsi sigmoid dengan vektor $\mathbf{W_i}, \mathbf{V_i}$, dan $\mathbf{b_i}$. Matriks-matriks tersebut memiliki ordo yang sesuai dengan jumlah unit (d) yang digunakan selama pelatihan model. Secara matematis keadaan dalam *input gate* diperlihatkan dalam Persamaan (4), didapat

$$\mathbf{i_t} = \sigma(\mathbf{W_i h_{t-1}} + \mathbf{V_i} x_t + \mathbf{b_i})$$

dengan

$$\mathbf{V_i} = \begin{bmatrix} v_{i_{11}} \\ v_{i_{21}} \\ \vdots \\ v_{i_{d1}} \end{bmatrix}$$

$$\mathbf{W_i} = \begin{bmatrix} w_{i_{11}} & w_{i_{12}} & \dots & w_{i_{1d}} \\ V_{i_{21}} & w_{i_{22}} & \dots & w_{i_{2d}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{i_{d1}} & w_{i_{d2}} & \dots & w_{i_{dd}} \end{bmatrix}$$

$$\mathbf{b_i} = \begin{bmatrix} b_{i_1} \\ b_{i_2} \\ \vdots \\ b_{i_d} \end{bmatrix}$$

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 5 No. 2 Agustus 2023

5. Data x_t masuk kembali sebagai *input* data dan $\mathbf{h_{t-1}}$ masuk sebagai *the hidden state*, data tersebut diolah dalam *candidate cell* menggunakan fungsi tanh dan vektor $\mathbf{W_c}, \mathbf{V_c}$, dan $\mathbf{b_c}$. Matriks-matriks tersebut memiliki ordo yang sesuai dengan jumlah unit (d) yang digunakan selama pelatihan model. Secara matematis keadaan dalam *candidate cell* diperlihatkan dalam Persamaan (5), didapat

$$\widetilde{\mathbf{C_t}} = \tanh(\mathbf{W_c h_{t-1}} + \mathbf{V_c} x_t + \mathbf{b_c})$$

dengan

$$\mathbf{V_c} = \begin{bmatrix} v_{c_{11}} \\ v_{c_{21}} \\ \vdots \\ v_{c_{d1}} \end{bmatrix}$$

$$\mathbf{W_c} = \begin{bmatrix} w_{c_{11}} & w_{c_{12}} & \dots & w_{c_{1d}} \\ w_{c_{21}} & w_{c_{22}} & \dots & w_{c_{2d}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{c_{d1}} & w_{c_{d2}} & \dots & w_{c_{dd}} \end{bmatrix}$$

$$\mathbf{b_c} = \begin{bmatrix} b_{c_1} \\ b_{c_2} \\ \vdots \\ b_{c_d} \end{bmatrix}$$

6. Kalikan nilai *input gate* (i_t) dan *candidate cell state* $(\widetilde{\mathbf{C_t}})$ sehingga didapatkan

$$\mathbf{C_t^i} = \mathbf{i_t} \circ \widetilde{\mathbf{C_t}}. \quad (16)$$

7. Jumlahkan Persamaan (15) dan (16) yang sudah didapatkan untuk mendapatkan *cell state* atau dapat menggunakan Persamaan (9) sebagai berikut:

$$\mathbf{C_t} = \mathbf{C_t^f} + \mathbf{C_t^i}$$
$$\mathbf{C_t} = (\mathbf{f_t} \circ \mathbf{C_{t-1}}) + (\mathbf{i_t} \circ \widetilde{\mathbf{C_t}})$$

8. Data x_t masuk kembali sebagai *input* data dan $\boldsymbol{h_{t-1}}$ masuk sebagai *the hidden state*, data tersebut diolah dalam *output gate* menggunakan fungsi sigmoid dan vektor $\mathbf{W_o}, \mathbf{V_o}$, dan $\mathbf{b_o}$. Matriks-matriks tersebut memiliki ordo yang sesuai dengan jumlah unit (d) yang digunakan selama pelatihan model. Secara matematis keadaan dalam *output gate* diperlihatkan dalam Persamaan (6), didapat

$$\mathbf{o_t} = \sigma(\mathbf{W_o h_{t-1}} + \mathbf{V_o} x_t + \mathbf{b_o}),$$

dengan

$$\mathbf{V_o} = \begin{bmatrix} v_{o_{11}} \\ v_{o_{21}} \\ \vdots \\ v_{o_{d1}} \end{bmatrix}$$

$$\mathbf{W_o} = \begin{bmatrix} w_{o_{11}} & w_{o_{12}} & \dots & w_{o_{1d}} \\ w_{o_{21}} & w_{o_{22}} & \dots & w_{o_{2d}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{o_{d1}} & w_{o_{d2}} & \dots & w_{o_{dd}} \end{bmatrix}$$

$$\mathbf{b_o} = \begin{bmatrix} b_{o_1} \\ b_{o_2} \\ \vdots \\ b_{o_d} \end{bmatrix}$$

9. Nilai *output* dari Persamaan (6) dan (9) diolah dalam Persamaan (10) untuk akhirnya didapat nilai *output* yang digunakan untuk perhitungan pada sel selanjutnya dan perhitungan pada *dense layer*, didapat:

$$\mathbf{h_t} = \mathbf{o_t} \circ \tanh(\mathbf{C_t})$$
$$\mathbf{z_t} = \mathbf{h_t}$$

10. Hasil peramalan akhir didapatkan dari menyatukan semua nilai *output* pada unit LSTM menggunakan *dense layer* pada Persamaan (13):

$$y = \sigma(\mathbf{W z_t} + \mathbf{b}).$$

4.4 Peramalan Data Menggunakan Model yang Telah Dibangun

Setelah dilakukan *training* model menggunakan metode LSTM, maka data akan diramalkan. Selanjutnya untuk mendapatkan interval nilai pada data yang digunakan perlu dilakukan invers fungsi MinMaxScaler

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 5 No. 2 Agustus 2023

menggunakan Persamaan (2). *Code program invers scaling* data setelah data selesai diramalkan adalah sebagai berikut:

```
y_train=scaler.inverse_transform(y_train)
y_test=scaler.inverse_transform(y_test)
train_predictions=model.predict(X_train)
train_predictions=scaler.inverse_transform
(train_predictions)

test_predictions=model.predict(X_test)
test_predictions=scaler.inverse_transform(
test_predictions)
```

Data aktual dan data peramalan yang telah didapatkan akan digunakan untuk menghitung nilai MAPE menggunakan Persamaan (14) untuk selanjutnya akan dianalisis performa model yang telah dibuat. *Code program* perhitungan nilai MAPE adalah sebagai berikut:

```
def evaluate_prediction(predictions, actua
l, model_name):
    print(model_name + ':')
    print('Mean Absolute Percentage Error:
 {:.4f}'.format(np.mean(np.abs((actual - p
redictions) / actual)) * 100, 2))
    return('')

print(evaluate_prediction(train_prediction
s, y_train, 'LSTM for training data'))
print(evaluate_prediction(test_predictions
, y_test, 'LSTM for testing data'))
```

Selanjutnya dilakukan peramalan untuk data pada tahap selanjutnya. *Code program* persamalan data pada tahap selanjutnya adalah sebagai berikut:

```
def insert_end(Xin,new_input):
    for i in range(timestep-1):
        Xin[:,i,:] = Xin[:,i+1,:]
    Xin[:,timestep-1,:] = new_input
    return Xin
from datetime import timedelta
timestep = 14
future = 31
forecast = []
Xin = X_test[-1:,:,:]
for i in range(future):
    out = model.predict(Xin, batch_size=1)
    forecast.append(out[0,0])
    Xin2 = insert_end(X_test,out[0,0])
forecasted_output=np.asanyarray(forecast)

forecasted_output=forecasted_output.reshap
e(-1,1)
forecasted_output = scaler.inverse_transfo
rm(forecasted_output)
forecasted_output = pd.DataFrame(forecaste
d_output)
```

5 SIMPULAN

Berdasarkan tahapan penelitian yang telah dilakukan, maka disimpulkan bahwa peramalan data univariat dapat dilakukan menggunakan pendekatan *deep learning* jenis *supervised learning* yaitu metode LSTM dengan bantuan bahasa pemrograman Python. Dalam penelitian yang telah dilakukan penulis menggunakan data univariat. Untuk penelitian selanjutnya diharapkan dapat menggunakan data multivariat dengan tambahan variabel lain yang dapat memengaruhi hasil peramalan. Peramalan menggunakan metode LSTM pada data multivariat sudah dilakukan oleh Hewamalage *et al.* (2021).

DAFTAR PUSTAKA

Abhishek, K., Singh, M. P., Ghosh, S., & Anand, A. (2012). Weather Forecasting Model using Artificial Neural Network. *Procedia Technology*, *4*, 311–318. https://doi.org/10.1016/j.protcy.2012.05.047

Bandara, Kasun, Shi, P., Bergmeir, C., Hewamalage, H., Quoc Tran, & Seaman, B. (2019). Sales demand forecast in e-commerce using a long short-term memory neural network methodology. Dalam *Neural Information Processing: 26th International Conference, ICONIP 2019*, *III*(26), 462–474.

Batta, M. (2018). Machine Learning Algorithms - A Review. *International Journal of Science and Research (IJSR)*, *18*(8), 381–386. https://doi.org/10.21275/ART20203995

Chang, Z., Zhang, Y., & Chen, W. (2019). Electricity price prediction based on hybrid model of adam optimized LSTM neural network and wavelet transform. *Energy*, *187*, 115804. https://doi.org/10.1016/j.energy.2019.07.134

Crone, S. F., Hibon, M., & Nikolopoulos, K.

(2011). Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. *International Journal of Forecasting*, *27*(3), 635–660. https://doi.org/10.1016/j.ijforecast.2011.04.001

Falah, R. A., & Rachmaniah, M. (2022). Price Prediction Model for Red and Curly Red Chilies using Long Short Term Memory Method. *Indonesian Journal of Statistics and Its Applications*, *6*(1), 143–160. https://doi.org/10.29244/ijsa.v6i1p143-160

Fan, C., Chen, M., Wang, X., Wang, J., & Huang, B. (2021). A Review on Data Preprocessing Techniques Toward Efficient and Reliable Knowledge Discovery From Building Operational Data. *Frontiers in Energy Research*, *9*(March), 1–17. https://doi.org/10.3389/fenrg.2021.652801

Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). Recurrent Neural Networks for Time Series Forecasting: Current status and future directions. *International Journal of Forecasting*, *37*(1), 388–427. https://doi.org/10.1016/j.ijforecast.2020.06.008

Juna, A., Umer, M., Sadiq, S., Karamti, H., Eshmawi, A. A., Mohamed, A., & Ashraf, I. (2022). Water Quality Prediction Using KNN Imputer and Multilayer Perceptron. *Water (Switzerland)*, *14*(17), 1–19. https://doi.org/10.3390/w14172592

Karim, F., Majumdar, S., & Darabi, H. (2019). Insights into lstm fully convolutional networks for time series classification. *IEEE Access*, *7*, 67718–67725. https://doi.org/10.1109/ACCESS.2019.2916828

Krisma, A., Azhari, M., & Widagdo, P. P. (2019). Perbandingan Metode Double Exponential Smoothing Dan Triple Exponential Smoothing Dalam Parameter Tingkat Error Mean Absolute Percentage Error (MAPE) dan Means Absolute Deviation (MAD) Alviani Krisma Putut Pamilih Widagdo Kata kunci-forecasting, Double Ex. *Prosiding Seminar Nasional Ilmu Komputer Dan Teknologi Informasi*, *4*(2).

Lewis, C. D. (1982). Industrial and Business Forecasting Method. In *Butter-worth-Heinemann*.

Lillicrap, T. P., & Santoro, A. (2019). ScienceDirect Backpropagation through time and the brain. *Current Opinion in Neurobiology*, *55*, 82–89. https://doi.org/10.1016/j.conb.2019.01.011

Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences (Switzerland)*, *9*(20). https://doi.org/10.3390/app9204396

Lou, H. H., Fang, J., Gai, H., Xu, R., & Lin, S. (2022). A novel zone-based machine learning approach for the prediction of the performance of industrial flares. *Computers and Chemical Engineering*, *162*, 107795. https://doi.org/10.1016/j.compchemeng.2022.107795

Nasteski, V. (2017). An overview of the supervised machine learning methods. *Horizons.B*, *4*(December), 51–62. https://doi.org/10.20544/horizons.b.04.1.17.p05

Pontoh, R. S., Toharudin, T., Ruchjana, B. N., Gumelar, F., Putri, F. A., Agisya, M. N., & Caraka, R. E. (2022). Jakarta Pandemic to Endemic Transition: Forecasting COVID-19 Using NNAR and LSTM. *Applied Sciences (Switzerland)*, *12*(12). https://doi.org/10.3390/app12125771

Scikit-learn Developers. (2022). *sklearn.preprocessing.MinMaxScaler*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn-preprocessing-minmaxscaler

Yang, Y., Wang, J., & Wang, B. (2020). Prediction model of energy market by long short term memory with random system and complexity evaluation.

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 5 No. 2 Agustus 2023

Applied Soft Computing, *95*(106579).