Faktor Exacta

Vol. 15, No. 4, Dec 2022, pp. 290~300
eISSN: 2502-339X, pISSN: 1979-276X, DOI: 10.30998/faktorexacta.v15i4.15286 

# Konfigurasi Hyperparameter Long Short Term Memory untuk Optimalisasi Prediksi Penjualan

**Nuke L. Chusna[1], Dhistianti Mei Rahmawan Tari[2], Ali Khumaidi[3]**
[1,3] Teknik Informatika, Fakultas Teknik, Universitas Krisnadwipayana, Indonesia
[2] Manajemen, Fakultas Ekonomi, Universitas Krisnadwipayana, Indonesia

**Article Info**

***Article history:***

Received Dec 01, 2022
Revised Dec 09, 2022
Accepted Jan 24, 2023

***Keywords:***

CRISP-DM
Hyperparameter
LSTM
Penjualan
Prediksi

**ABSTRACT**

PT. Sumber Prima Inti Motor experienced problems in selling spare parts, namely the availability of customer orders which resulted in customers waiting long and even moving messages to other places. Therefore it is necessary to forecast sales and procurement of spare parts that are accurate. Long Short Term Memory (LSTM) is a fairly good algorithm for forecasting. The Cross Industry Standard Process for Data Mining (CRISP-DM) method is used and hyperparameter configuration is performed. Hyperparameter configuration provides optimal sales prediction model performance. The configurations used are number of hidden layers, data partition, epoch, batch size, and dropout scenario. This research is useful for companies because it can predict sales of spare parts for the next 60 days. The best results from the LSTM model hyperparameter configuration are 3 hidden layers, 3 dropouts, epoch 150, and batch size 30. The performance of the training and testing models with RMSE is 0.0855 and 0.0846.



***Corresponding Author:***

Ali Khumaidi,
Teknik Informatika,
Universitas Krisnadwipayana,
Jl. Kampus Unkris, Jatiwaringin, Pondokgede, Jakarta Timur.
Email: alikhumaidi@unkris.ac.id

## 1. PENDAHULUAN (10 PT)

Pelaku usaha saat ini dihadapkan pada tantangan persaingan usaha yang semakin kompetitif, yang menuntut setiap pelaku usaha untuk mendesain ulang strategi usahanya agar dapat memenuhi permintaan pasar. Revolusi industri 4.0 dan perkembangan *machine learning* dapat dimanfaatkan oleh para pelaku usaha untuk melakukan peramalan dengan akurasi terbaik. Beberapa pelaku usaha telah menerapkan metode peramalan untuk memperkirakan penjualan [1]. Namun, prediksi yang digunakan seringkali tidak akurat dan karenanya kurang efektif. Hal ini berdampak pada akumulasi produk ketika permintaan konsumen dan frekuensi penjualan rendah sehingga menyebabkan biaya penyimpanan meningkat. Selain itu, ketika permintaan produk meningkat tetapi terjadi stock out yang cukup lama, menyebabkan toko kehilangan penjualan (*lost sales*). Untuk meminimalisir kerugian akibat kesalahan dalam memprediksi penjualan maka diperlukan prediksi dengan memanfaatkan data historis transaksi penjualan dengan menggunakan suatu metode untuk mendapatkan akurasi hasil prediksi yang optimal [2].

Penelitian sebelumnya terkait peramalan penjualan telah dilakukan, *Long Short Term Memory* (LSTM) mampu memprediksi permintaan yang berfluktuasi dengan baik dan mengungguli *autoregressive integrated moving average* (ARIMA), *support vector machine* (SVM), *K-nearest neighbor* (KNN), dan *artificial neural network* (ANN) [3][4]. Peramalan harga komoditas pertanian dengan model LSTM lebih unggul dari metode *Holt-Winter's Seasonal* dan model *seasonal* ARIMA (SARIMA) dengan evaluasi berdasarkan nilai root mean square error (RMSE) [5]. Peramalan ketergantungan kebutuhan listrik dengan

menggunakan LSTM menunjukkan hasil yang lebih baik dibandingkan dengan metode tradisional, SARIMA, *auto regressive moving average* (ARMA) dan ARMA *exogenous* (ARMAX) dengan menggunakan *mean absolut percentage error* (MAPE) dan RMSE [6]. LSTM memiliki kinerja yang sangat baik dalam memodelkan perilaku pelanggan di lingkungan yang cukup kompleks [7]. Prediksi penjualan di 1.150 toko di Jerman, menggunakan perbandingan algoritma *extreme gradient boosting* (XGB) dan *random forest* (RF) dengan LSTM. Hasil penelitian menunjukkan bahwa LSTM memiliki akurasi 12-14% lebih baik daripada XGB dan RF [8]. Penelitian yang telah dilakukan menunjukkan bahwa algoritma LSTM memiliki akurasi yang cukup baik untuk melakukan prediksi dan pada penelitian ini akan lebih meningkatkan performa LSTM dengan melakukan konfigurasi *hyperparameter*.

Berdasarkan penelitian-penelitian sebelumnya di bidang peramalan penjualan, cukup banyak yang telah dilakukan dengan tujuan untuk menentukan produk yang paling laris dengan menggunakan algoritma tertentu atau membandingkan algoritma tertentu namun hasil dari penelitian hanya mampu memprediksi dengan baik untuk jangka waktu 1 bulan. Penelitian ini akan fokus untuk memprediksi jumlah produk yang terjual di masa mendatang dengan mengoptimalkan algoritme hyperparameter LSTM untuk meningkatkan kinerja model yang terbentuk selama 60 hari ke depan atau 2 bulan. Penelitian ini untuk menghasilkan prediksi penjualan menggunakan algoritme LSTM RNN dengan mencari konfigurasi model terbaik melalui penentuan parameter terbaik sehingga dapat meningkatkan performa model yang dibentuk sehingga dapat membantu perencanaan strategi penjualan dan penjadwalan persediaan produk.

## 2. METODE

### 2.1. Dataset

Sumber data yang digunakan dalam penelitian ini adalah laporan data pengeluaran spare part pada bengkel PT. Sumber Prima Inti Motor periode Mei 2020 hingga Mei 2022 sebanyak 48.808 data. Sampel data dapat dilihat pada Gambar 1.

| | no | no_polisi | no_transaksi | tanggal | kode_transaksi | kode_spare_part | nama_spare_part | qty | satuan | Harga Satuan | total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | B9317KXS | 1 | 2020-05-02 | 186 | 1543 | OLI MESIN 15W-40 API CI-4 | 4.0 | Ltr | 32576.0 | 114018.0 |
| **1** | 2 | B9317KXS | 2 | 2020-05-02 | 186 | 23 | AIR ACCU + DRIGEN ( 20LITER ) | 4.0 | Ltr | 2750.0 | 11000.0 |
| **2** | 3 | B9926TCI | 1 | 2020-05-02 | 609 | 1543 | OLI MESIN 15W-40 API CI-4 | 4.0 | Ltr | 32576.0 | 114018.0 |
| **3** | 4 | B9519KCE | 1 | 2020-05-02 | 392 | 1543 | OLI MESIN 15W-40 API CI-4 | 2.0 | Ltr | 32576.0 | 48865.0 |
| **4** | 5 | B9960TCC | 1 | 2020-05-02 | 641 | 1698 | RECO-COOL MULTIROAD WATERBASED COLANT | 5.0 | Ltr | 20100.0 | 100500.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **48803** | 48804 | B9355TCE | 1 | 2022-05-31 | 223 | 576 | CONNECTOR PR GEPENG | 4.0 | PC | 567.0 | 2267.0 |
| **48804** | 48805 | B9817PJ | 1 | 2022-05-31 | 584 | 1984 | SKRING MINI 15A | 2.0 | PC | 2045.0 | 4091.0 |
| **48805** | 48806 | B9355TCE | 1 | 2022-05-31 | 223 | 347 | BOHLAM K2B 12V | 2.0 | PC | 6591.0 | 13182.0 |
| **48806** | 48807 | B9355TCE | 1 | 2022-05-31 | 223 | 345 | BOHLAM K1B 12V21W | 2.0 | PC | 3409.0 | 6818.0 |
| **48807** | 48808 | B9355TCE | 1 | 2022-05-31 | 223 | 14 | ACCU NS70 / 12V/65AH | 2.0 | PC | 840000.0 | 1680000.0 |

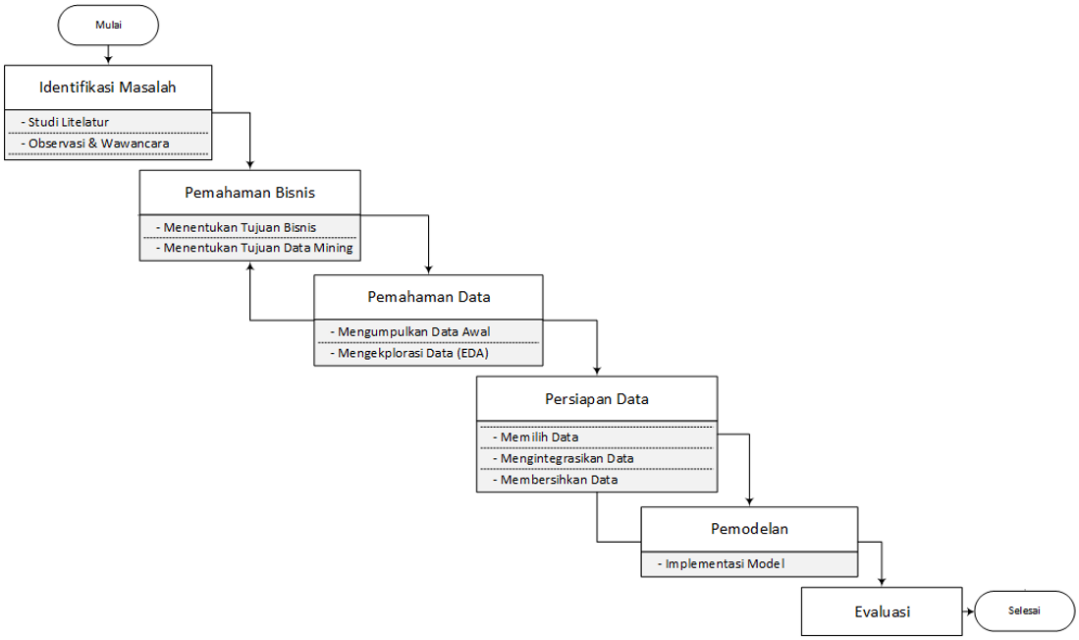
48808 rows × 11 columns

Gambar 1. Sampel data pengeluaran barang

### 2.2. Tahapan penelitian

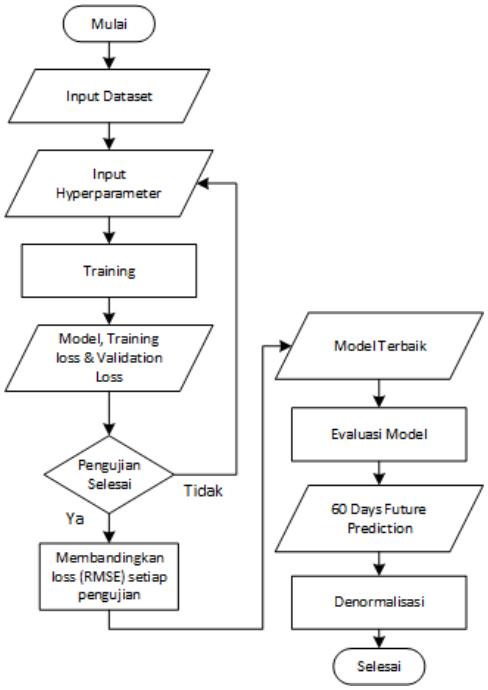
Metode yang digunakan dalam penelitian ini adalah *Cross Industry Standard Process for Data Mining* (CRISP-DM) yang merupakan standar proses untuk pemecahan masalah pada unit riset bidang data mining [9]. Gambar 2 adalah tahapan-tahapan yang dikerjakan pada penelitian ini. Pada tahap identifikasi masalah bertujuan supaya analisis masalah lebih terarah. Pada tahap pemahaman bisnis berfokus pada pemahaman berdasarkan sudut pandang bisnis dan penentuan strateginya. Pada tahap pemahaman data memberikan fondasi analitik untuk sebuah penelitian dengan membuat ringkasan (*summary*) dan mengidentifikasi potensi masalah yang terdapat dalam data. Proses pemahaman data dilakukan eksplorasi data dengan menggunakan metode *exploratory data analysis* (EDA). Teknik tersebut menerapkan dengan menerapkan teknik grafis dan aritmatika untuk meringkas dan memudahkan pemahaman data [10]. Tahap persiapan data atau data *preprocessing* mencangkup seluruh kegiatan untuk membangun dataset akhir dari data mentah awal untuk set up data mining [11]. Tahap ini dapat dilakukan secara berulang untuk mendapatkan data yang sesuai, mencakup langkah-langkah berikut :

a. Membersihkan Data (*Data Cleaning*)
   Menyesuaikan format data yang ada sesuai dengan kebutuhan, menghilangkan missing value dan data-data yang tidak dibutuhkan.
b. Transformasi Data (*Data Transformation*)
   Tahap ini merupaka cara menormalisasi data untuk menyamakan format dalam bentuk skala umum dengan rentang 0 sampai 1. Dalam penelitian ini akan menggunakan metode *Min-Max Scaler.*




Gambar 2. Diagram alir penelitian

Proses pemodelan dapat dilihat pada Gambar 3. Terdiri dari 7 proses sebagai berikut:




Gambar 3. Diagram alir pemodelan

a. Input Dataset
Tahap input data merupakan tahap memasukan data hasil *preprocessing* ke dalam skenario model yang akan dilakukan uji coba.

b. Input *Hyperparameter*
Tahap ini merupakan proses mengatur skenario parameter agar menghasilkan parameter yang tepat agar algoritme LSTM dapat mempelajari pola-pola yang terdapat dalam data dengan baik [12]. Adapun tahapan dalam melakukan input hyperparameter dapat dilihat pada Tabel 1.
Pada tahap awal perlu dilakukan uji coba terhadap komposisi *training* dan *testing* yang menghasilkan *loss* terkecil. Setelah didapatkan komposisi yang telah sesuai maka hasil dari percobaan tersebut

digunakan untuk mencari *hidden layer* yang terbaik. Apabila jumlah *hidden layer* telah menghasilkan *loss* terkecil maka tahap selanjutnya adalah membuat skenario *dropout*. *Dropout* berfungsi untuk mencegah data mengalami *overfitting.* Apabila jumlah *dropout* telah menghasilkan *error* yang kecil maka akan didapatkan partisi data dengan layer yang telah sesuai. Selanjutnya adalah penentuan *batch size* dan *epoch* yang terbaik berdasarkan komposisi data dan *layer* yang telah ditentukan.

c. *Training*
Pada tahap ini, penerapan uji coba prediksi berbasis machine learning yaitu algotitme LSTM dapat diimplementasi pada data training. Proses training bertujuan untuk memberi petunjuk pada data melalui algoritme agar dapat belajar dari pola data yang diberikan dan mencari korelasinya.

d. Model, *Training Loss* dan *Validation Loss*
Pada tahap ini hasil dari konfigurasi hyperparameter tuning akan ditampilkan dengan hasil berupa nilai *training loss* dan *validation loss*. Hasil dari training dan validation loss merupakan representasi seberapa baik model yang dihasilkan dengan parameter uji berupa RMSE.

e. Evaluasi Model
Setelah melalui serangkaian proses, selanjutnya perlu dilakukan evaluasi kinerja model dengan menggunakan perhitungan *Root Mean Square Error* (RMSE) [13]. RMSE merupakan metode alternatif yang digunakan untuk mengukur tingkat akurasi teknik peramalan yang digunakan.

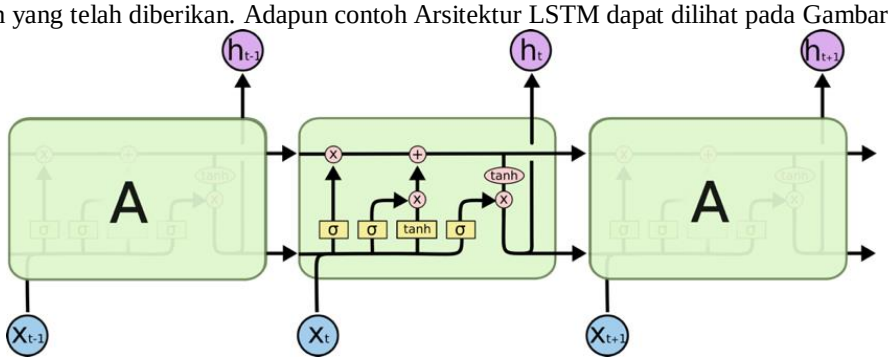
f. Prediksi 60 hari ke Depan
Tahap prediksi merupakan proses untuk meramalkan suat nilai untuk masa yang akan datang berdasarkan pola tertentu pada suatu data. Pada penelitian ini prediksi dilakukan untuk memperkirakan penjualan dalam satu tahun kedepan. Hasil dari proses prediksi selanjutnya akan direkomendasikan untuk digunakan sebagai alat bantu pengambilan keputusan.

g. Denormalisasi
Setelah mendapatkan hasil prediksi dari proses prediksi kemudian data dilakukan denormalisasi yaitu pengubahan data kembali ke nilai *real* [14]. Karena data masih berbentuk *range interval* yang sebelumnya telah dilakukan normalisasi data. Tujuan dilakukannya denormalisasi adalah untuk mempermudah saat membaca nilai output yang dihasilkan

Tabel 1. Tahapan uji coba parameter

| Tahap Ke | Parameter |
|---|---|
| Satu | Jumlah data |
| Dua | Hidden layer |
| Tiga | Dropout |
| Empat | Batch size |
| Lima | Epoch |

*Long Short Term Memory* (LSTM) merupakan salah satu jenis dari *Recurrent Neural Network* (RNN). RNN mampu memproses data sekuensial yang panjang dan kompleks. LSTM dikembangkan untuk mengatasi *vanishing gradient* pada RNN. LSTM telah dimodifikasi dengan menambahkan *memory cell* sehingga dapat menyimpan informasi dalam jangka waktu yang lebih lama. LSTM merupakan hasil pengembangan dari ANN yang dapat digunakan sebagai pemodelan data *time series*. LSTM memiliki keunggulan karena memiliki *memory block* yang akan menentukan nilai mana yang akan dipilih sebagai keluaran yang relevan terhadap masukan yang telah diberikan. Adapun contoh Arsitektur LSTM dapat dilihat pada Gambar 4.




Gambar 4. Arsitektur Long Short Term Memory (LSTM)

Arsitektur LSTM terdiri dari lapisan masukan *(input layer)*, lapisan keluaran *(output layer)* dan lapisan tersembunyi *(Hidden Layer)*. Lapisan tersembunyi terdiri dari *memory cell*, setiap memori terdiri dari gerbang masukan *(input gate)*, gerbang lupa *(forget gate)* dan gerbang keluaran *(output gate)*. Penjelasan untuk setiap gerbang *(gate)* yang terdapat pada LSTM adalah sebagai berikut :

a. *Forget Gate* ($f_t$)
*Forget gate* berfungsi mengendalikan sejauh mana nilai tetap berada di dalam *memory cell*. *Forget gate* merupakan lapisan sigmoid yang mengambil nilai *output* pada waktu t-1 dan nilai *input* pada waktu t, kemudian menggabungkan dan menerapkan pada fungsi aktivasi sigmoid. Dimana hasil *output* sigmoid bernilai 0 dan 1. Jika nilai $f_t$ = 1 maka *state* sebelumnya tidak berubah dan data akan disimpan, sedangkan jika $f_t$ = 0 maka *state* sebelumnya akan dilupakan. . Persamaan dari $f_t$ yaitu: $f_t = \sigma\,(W_f S_{t-1} + W_f X_t)$

b. *Input Gate* ($I_t$)
*Input gate* berfungsi untuk mengendalikan sejauh mana nilai baru mengalir di dalam *cell*. Hal tersebut bertujuan untuk menghindari penyimpanan data yang tidak perlu. *Input gate* mengambil nilai *output* sebelumnya dan nilai *input* baru serta melewatkan lapisan sigmoid. Gate ini mengembalikan nilai menjadi 0 atau 1. Persamaan dari $I_t$ yaitu: $I_t = \sigma\,(W_i S_{t-1} + W_i X_t)$
Nilai dari gerbang input *(input gate)* kemudian dikalikan dengan nilai *output* dari lapisan kandidat $\hat{C}$. Persamaan dari $\hat{C}$ yaitu : $\widehat{C} = \tanh\,(W_c S_{t-1} + W_c X_t)$

c. *Output Gate* ($O_t$)
*Output Gate* berfungsi untuk mengendalikan banyaknya nilai yang beada di dalam *memory cell* yang digunakan untuk menghitung nilai *output*. Rumus dari $O_t$ yaitu : $O_t = \sigma\,(W_t S_{t-1} + W_t X_t)$

Untuk memenuhi kebutuhan perangkat lunak dalam penelitian digunakan Python 3.7, Jupiter Notebook atau Google Colab, Tensorflow, Pandas, Matplotlib, Numpy, Math, Sklearn.metrics.

## 3. HASIL DAN PEMBAHASAN

### 3.1. Preprocessing data

Pada tahap pre-processing data ini terdapat beberapa tahapan yang dilakukan sebagai berikut :

a. *Data cleaning* merupakan salah satu metode untuk mengisi data yang hilang (missing value) atau mengapus data yang tidak dibutuhkan. Pada penelitian ini, data yang digunakan terlebih dahulu diindentifikasi kolom dalam tabel yang hilang (*missing*), dapat dilihat pada Gambar 5. Setelah ditinjau kembali pada dataset yang tersedia perlu dilakukan *forward fill* pada baris yang Nan (*Not a Number*) dan menghapus harga satuan yang kosong.

b. *Data transformation* merupakan teknik menormalisasi data dalam skala 0-1 [15]. Dalam penelitian ini digunakan metode *Min-MaxScaler* untuk data modelling, dapat dillihat pada Gambar 6.

```
[ ]  1 #Identifikasi Missing Value
     2
     3 # memeriksa persentase nilai nan yang ada di setiap fitur
     4 # membuat daftar fitur yang memiliki missing value
     5 features_with_na=[features for features in df.columns if df[features].isnull().sum()>1]
     6
     7 ## cetak nama fitur dan persentase missing value
     8 for feature in features_with_na:
     9     print(feature, np.round(df[feature].isnull().mean(), 4), ' % missing values')
```

```
qty = 0.0001% missing values
Harga Satuan = 0.0015% missing values
total = 0.0001% missing values
```

Gambar 5. Identifikasi missing value

```
1 scaler = MinMaxScaler(feature_range=(0,1))
2 qty_df = scaler.fit_transform(np.array(qty_df).reshape(-1,1))
3 qty_df
```

```
array([[2.42797022e-02],
       [1.70929103e-01],
       [1.24959534e-01],
       [1.19779864e-01],
       [4.59695694e-02],
       [4.14373584e-02],
       [6.47458725e-02],
```

Gambar 6. Transformasi menggunakan minmax scaler

### 3.2. Pemodelan

Dalam pembuatan sebuah model LSTM digunakan beberapa parameter diantaranya adalah partisi data, jumlah *hidden layer*, skenario *dropout* untuk mencegah *overfitting*, jumlah neuron, *epoch* menggambarkan jumlah iterasi training, *batch size* adalah jumlah data training yang harus dipertimbangkan setiap proses memperbaharui bobot. Untuk mendapatkan hasil yang terbaik dari model LSTM, proses *training* akan menggunakan parameter yang berbeda-beda agar mendapatkan hasil yang terbaik. Dari sejumlah model yang dihasilkan akan dibandingkan dan dianalisis dari setiap nilai loss dan RMSE yang dihasilkan setiap kali eksperimen. Parameter yang digunakan untuk pembuatan model LSTM dapat dilihat pada Tabel 2.

Tabel 2. Parameter Pengujian LSTM

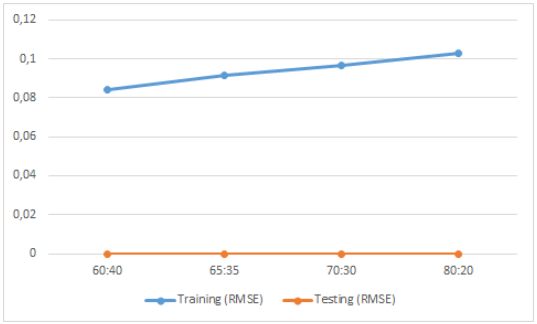
| Parameter | Jumlah | Keterangan |
|---|---|---|
| *Input* | 3 | Data total penjualan |
| *Output* | 1 | Data penjualan harian |
| *Optimizer* | 1 | Adam |
| *Hidden Layer (neuron)* | Percobaan | 30 – 120 |
| *Dropout (layer)* | Percobaan | 1 - 3 |
| *Batch Size* | Percobaan | 10 – 400 |
| *Epoch* | Percobaan | 50 - 200 |

Pengujian akan dilakukan pada setiap parameter, dimana hasil pengujian parameter yang telah menghasilkan loss yang cukup baik akan digunakan untuk pengujian berikutnya sehingga diharapkan akan menghasilkan model terbaik. Tahapan pengujian berdasarkan Tabel 1.

Pada eksperimen pertama, dilakukan eksperimen untul mendapatkan komposisi data yang optimal. *Hyperparameter* yang digunakan untuk mencari jumlah komposisi data yaitu 30 *Batch Size*, 100 *Epoch* dan menggunakan *optimizer Adam*. Berdasarkan hasil eksperimen pada Tabel 3, *loss* yang didapat pada pengujian 1 dan 2 dikategorikan overfitting karena training loss lebih kecil dari validation loss. Sedangkan pada percobaan 3 sampai 5 dikategorikan sebagai underfitting karena *training loss* lebih besar dari *validation loss*. Untuk mendapat model yang optimal dari hasil eksperimen pertama ini diambil dari selisih terkecil antara nilai RMSE *Training* dan RMSE *Testing*, yang dapat dilihat pada Gambar 7. Berdasarkan pengujian pertama jumlah komposisi data yang optimal dari dataset yang di*training* terletak pada percobaan kedua dengan RMSE pelatihan sebesar 0.0913 dan RMSE testing 0.0858. Sedangkan untuk jumlah komposisi data yang kurang optimal terletak pada percobaan ke tiga.

Tabel 3. Hasil pengujian partisi data

| *Training* | *Testing* | *Loss* (MSE) | *Validation Loss* (MSE) | *Training* (RMSE) | *Testing* (RMSE) |
|---|---|---|---|---|---|
| 60 | 40 | 0.0077 | 0.0115 | 0.0844 | 0.1070 |
| 65 | 35 | 0.0092 | 0.0074 | 0.0913 | 0.0858 |
| 70 | 30 | 0.0093 | 0.0051 | 0.9638 | 0.0714 |
| 80 | 20 | 0.0106 | 0.0058 | 0.1028 | 0.0760 |



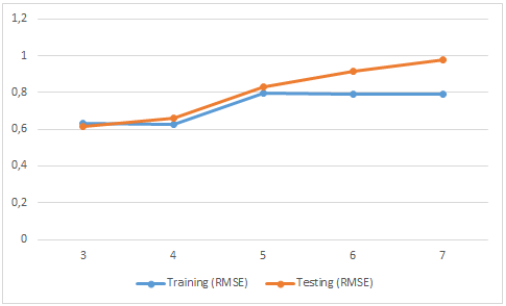

Gambar 7. Grafik hasil pengujian partisi data

Setelah mendapatkan hasil pengujian pertama, dilakukan pengujian tahap 2 untuk mendapatkan *Hidden layer* yang optimal. Pengujian tahap 2 dilakukan dengan *Batch size* 30, *Epoch* 100 dan komposisi data 65:35 yang merupakan hasil pengujian pertama. Hasil pengujian untuk mencari hidden layer dapat dilihat pada Tabel 4. Berdasarkan hasil eksperimen, nilai loss yang di dapat pada pengujian 2 sampai dengan 5 cenderung *overfitting* karena nilai tersebut lebih kecil daripada nilai *validation loss*. Untuk mendapatkan model yang optimal dari hasil eksperimen tersebut maka dapat dilihat dari nilai loss dan validation loss yang memiliki selisih kecil, didukung oleh nilai RMSE *Training* dan RMSE *Testing* yang juga memiliki selisih yang paling kecil, yang dapat dilihat pada Gambar 8. Berdasarkan hasil pengujian, dalam penelitian ini penambahan jumlah hidden layer tidak terlalu memberikan pengaruh yang signifikan terhadap berkurangnya suatu nilai parameter uji. Hal tersebut dibuktikan dengan bertambahnya nilai loss yang dihasilkan pada percobaab ke-2 sampai ke-

5 pada tabel diatas. Dari hasil pengujian ini jumlah hidden layer yang optimal dari dataset yang telah di training yaitu pada neuron hidden 3 dengan nilai RMSE Training sebesar 0,0910 dan RMSE Testing sebesar 0,0889. Sedangkan untuk jumlah *hidden layer* yang kurang optimal terletak pada percobaan ke-5 dengan jumlah hidden layer 7. Pengujian tahap berikutnya untuk mendapatkan skenario *dropout* terbaik. *Dropout* berfungsi untuk mencegah terjadinya *overfitting* yang terlalu besar. Pengujian tahap ketiga dilakukan dengan menggunakan *batch size* 50, *epoch* 100, komposisi data 60:35 berdasarkan hasil pengujian pertama dan jumlah hidden layer 3 berdasarkan hasil pengujian kedua.

Tabel 4. Hasil pengujian *hidden layer*

| *Hidden Layer* | *Loss* (MSE) | *Validation Loss* (MSE) | *Training* (RMSE) | *Testing* (RMSE) |
|---|---|---|---|---|
| 3 | 0.0089 | 0.0079 | 0.0910 | 0.0889 |
| 4 | 0.0083 | 0.0090 | 0.0900 | 0.0949 |
| 5 | 0.0134 | 0.0142 | 0.1145 | 0.1193 |
| 6 | 0.0132 | 0.0173 | 0.1140 | 0.1314 |
| 7 | 0.0133 | 0.0199 | 0.1141 | 0.1409 |



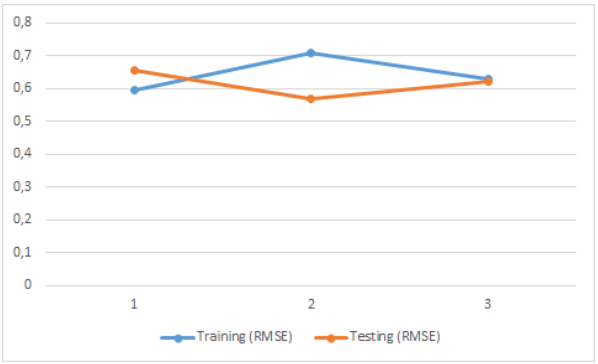

Gambar 8. Grafik hasil pengujian *hidden layer*

Hasil eksperimen ketiga dapat dilihat pada Tabel 5. Berdasarkan hasil pengujian menunjukan bahwa semakin bertambahnya jumlah *dropout* maka nilai yang dihasilkan cenderung naik dan turun, sehingga dapat disimpulkan bahwa penambahan nilai dropout tidak terlalu memberikan pengaruh yang signifikan apabila melebihi jumlah *hidden layer* yang telah ditentukan. Pada percobaan kedua nilai *loss training* cenderung meningkat, namun nilai validation cenderung menurun. Sedangkan pada pengujian ketiga nilai *loss training* cenderung menurun dan nilai *validation loss* cenderung meningkat. Pada tahap ini untuk menentukan nilai *dropout* yang optimal dengan cara membandingkan nilai hasil RMSE *Training* dan RMSE *Testing* yang memiliki selisih terkecil pada setiap pengujian, yang dapat dilihat pada Gambar 9. Berdasarkan hasil pada tahapan ini, nilai dropout yang optimal terletak pada pengujian ketiga dengan nilai RMSE *Training* sebesar 0,0906 dan RMSE *Testing* sebesar 0,0895. Pengujian tahap berikutnya untuk mendapatkan nilai batch size terbaik. Pengujian tahap keempat dilakukan dengan menggunakan *batch size* 50, *epoch* 100, komposisi data 60:35 berdasarkan hasil pengujian pertama dan jumlah hidden layer 3 berdasarkan hasil pengujian kedua dan 3 jumlah dropout berdasarkan pengujian ketiga.

Tabel 5. Hasil pengujian skenario *dropout*

| *Dropout* | *Loss* (MSE) | *Validation Loss* (MSE) | *Training* (RMSE) | *Testing* (RMSE) |
|---|---|---|---|---|
| 3 | 0.0091 | 0.0080 | 0.0906 | 0.0895 |
| 2 | 0.0102 | 0.0067 | 0.1021 | 0.0821 |
| 1 | 0.0097 | 0.0073 | 0.0857 | 0.0941 |

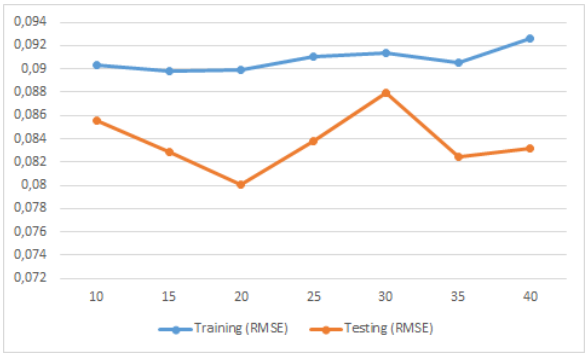
Hasil eksperimen keempat dapat dilihat pada Tabel 6. Berdasarkan hasil pengujian penambahan nilai *bacth size* cenderung menghasilkan nilai *loss* yang cenderung naik dan turun. Sehingga dapat disimpulkan bahwa tidak terdapat ketentuan khusus untuk nilai *batch size* terbaik. Hal tersebut karena setiap data memiliki *treatment* tersendiri dalam menentukan jumlah *batch size* yang tepat. Pada tahap ini untuk menentukan nilai *batch size* yang optimal dengan cara membandingkan nilai hasil RMSE *Training* dan RMSE *Testing* yang memiliki selisih terkecil pada setiap pengujian, yang dapat dilihat pada Gambar 10. Berdasarkan hasil pada tahapan ini, nilai *batch size* yang optimal terletak pada pengujian kelima dengan nilai RMSE Training sebesar 0,0914 dan RMSE Testing sebesar 0,0879. Pengujian tahap berikutnya untuk mendapatkan nilai *epoch* terbaik. Pengujian tahap kelima dilakukan dengan menggunakan, komposisi data 60:35 berdasarkan hasil pengujian pertama dan jumlah hidden layer 3 berdasarkan hasil pengujian kedua dan 3 jumlah *dropout* berdasarkan pengujian ketiga dan *batch size* sebesar 30 berdasarkan hasil pengujian tahap keempat.




Gambar 9. Grafik hasil pengujian *dropout*

Tabel 6. Hasil pengujian *batch size*

| *Batch Size* | *Loss* (MSE) | *Validation Loss* (MSE) | *Training* (RMSE) | *Testing* (RMSE) |
|---|---|---|---|---|
| 10 | 0.0088 | 0.0073 | 0.0903 | 0.0856 |
| 15 | 0.0086 | 0.0069 | 0.0898 | 0.0829 |
| 20 | 0.0079 | 0.0064 | 0.0899 | 0.0801 |
| 25 | 0.0092 | 0.0070 | 0.0911 | 0.0838 |
| 30 | 0.0097 | 0.0077 | 0.0914 | 0.0879 |
| 35 | 0.0087 | 0.0068 | 0.0905 | 0.0824 |
| 40 | 0.0094 | 0.0069 | 0.0926 | 0.0832 |



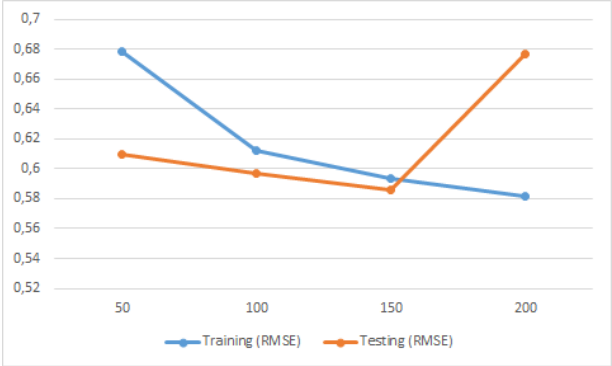

Gambar 10. Grafik hasil pengujian *batch size*

Hasil eksperimen kelima dapat dilihat pada Tabel 7. Berdasarkan hasil pengujian menunjukan bahwa semakin bertambahnya jumlah *epoch* maka nilai yang dihasilkan cenderung fluktuasi, sehingga dapat disimpulkan bahwa penentuan banyaknya *epoch* terbaik untuk menurunkan jumlah loss berdasarkan uji coba. Hal tersebut karena setiap data memiliki pola dan kompleksitas yang berbeda. Pada tahap ini untuk menentukan nilai epoch yang optimal dengan cara membandingkan nilai hasil RMSE *Training* dan RMSE *Testing* yang memiliki selisih terkecil pada setiap pengujian, yang dapat dilihat pada Gambar 11. Berdasarkan hasil pada tahapan ini, nilai *batch size* yang optimal terletak pada pengujian ketiga dengan nilai RMSE *Training* sebesar 0,0855 dan RMSE *Testing* sebesar 0,0844.

Tabel 7. Hasil pengujian *epoch*

| *Epoch* | *Loss* (MSE) | *Validation Loss* (MSE) | *Training* (RMSE) | *Testing* (RMSE) |
|---|---|---|---|---|
| 50 | 0.0111 | 0.0077 | 0.0977 | 0.0878 |
| 100 | 0.0087 | 0.0074 | 0.0881 | 0.0860 |
| 150 | 0.0076 | 0.0071 | 0.0855 | 0.0844 |
| 200 | 0.0076 | 0.0095 | 0.0837 | 0.0974 |

 DOI: 10.30998/faktorexacta.v15i4.15286



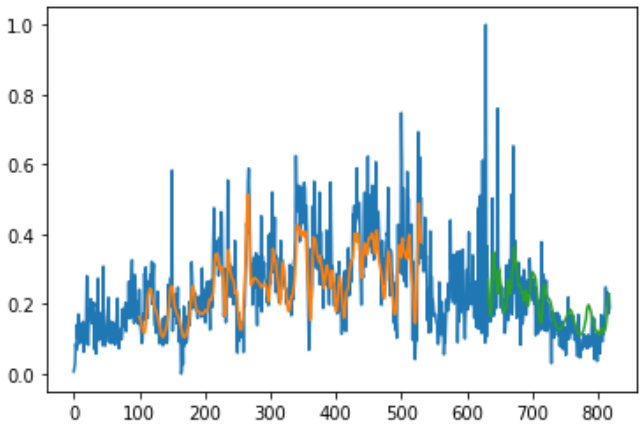

Gambar 11. Grafik hasil pengujian *epoch*

Prediksi dengan penentuan hyperparameter untuk model terbaik yang telah didapatkan pada proses pengujian yaitu dengan menggunakan algoritme LSTM dengan arsitektur pemodelan pada Gambar 12.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 100, 100)          40800
_________________________________________________________________
dropout (Dropout)            (None, 100, 100)          0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100, 100)          80400
_________________________________________________________________
dropout_1 (Dropout)          (None, 100, 100)          0
_________________________________________________________________
lstm_2 (LSTM)                (None, 100)               80400
_________________________________________________________________
dropout_2 (Dropout)          (None, 100)               0
_________________________________________________________________
dense (Dense)                (None, 1)                 101
=================================================================
Total params: 201,701
Trainable params: 201,701
Non-trainable params: 0
```

Gambar 12. Arsitektur layer LSTM

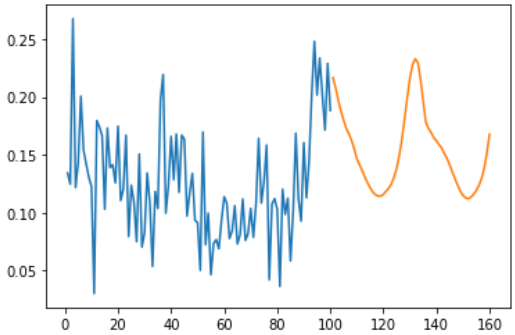
Pada Gambar 6 merupakan arsitektur layer LSTM model *sequential* dengan nilai lstm, *dropout*, dan *dense* yang dapat dilihat pada *output shape* dan *param*. Gambar 13 menunjukkan hasil prediksi penjualan dengan algoritme LSTM. Hasil berupa perbandingan antara data *actual* dengan data hasil prediksi *training* dan *testing*. Warna biru menjukkan data *actual*, warna oren adalah prediksi untuk *training* dan warna hijau merupakan prediksi untuk *testing*.




Gambar 13. Grafik hasil prediksi

### 3.3. Prediksi 60 hari ke depan

Dengan menggunakan komposisi *hyperparameter* yang telah *fit* peneliti menggunakan *lookback* 100 hari dari data *testing* untuk memprediksi kemungkinan penjualan 60 hari mendatang. Pada Gambar 14 menunjukan grafik mulai hari ke 781 (berwarna biru) sampai dengan hasil prediksi yang ditunjukan dengan garis grafik berwarna orange. Garis berwarna oranye menunjukkan hasil prediksi.




Gambar 14. Hasil prediksi penjualan 60 hari mendatang

## 4. PENUTUP

Berdasarkan penelitian yang telah dilakukan dapat disimpulkan bahwa model terbaik yang diperoleh dari LSTM dengan *hyperparameter batch size* 30, 3 *hidden layer*, *epoch* 150 dan 3 *dropout* menghasilkan RMSE *training* dan *testing* sebesar 0.0855 dan 0.0846. Selama proses pelatihan dikategorikan *overfitting* yaitu nilai validasi loss lebih besar dari pada *training loss*. *Overfitting* terjadi karena data *training* lebih mudah dipelajari daripada data *testing*. Selain itu, besar kecilnya nilai loss sangat dipengaruhi oleh konfigurasi *tuning hyperparameter* seperti data *partition, hidden layer, batch* size dan *epoch*.

## UCAPAN TERIMAKASIH

Penulis mengucapkan terima kasih kepada Kementerian Pendidikan, Kebudayaan, Riset dan Teknologi (Kemdikbudristek) Republik Indonesia yang telah mendanai penelitian pada tahun 2022.

## DAFTAR PUSTAKA

[1] R. M. van Steenbergen and M. R. K. Mes, "Forecasting demand profiles of new products," Decis. Support Syst., vol. 139, p. 113401, Dec. 2020, doi: 10.1016/j.dss.2020.113401.
[2] L. R. Berry, P. Helman, and M. West, "Probabilistic forecasting of heterogeneous consumer transaction–sales time series," Int. J. Forecast., vol. 36, no. 2, pp. 552–569, Apr. 2020, doi: 10.1016/j.ijforecast.2019.07.007.
[3] H. Abbasimehr, M. Shabani, and M. Yousefi, "An optimized model using LSTM network for demand forecasting," Comput. Ind. Eng., vol. 143, p. 106435, May 2020, doi: 10.1016/j.cie.2020.106435.
[4] L. Liang and X. Cai, "Forecasting peer-to-peer platform default rate with LSTM neural network," Electron. Commer. Res. Appl., vol. 43, p. 100997, Sep. 2020, doi: 10.1016/j.elerap.2020.100997.
[5] K. M. Sabu and T. K. M. Kumar, "Predictive analytics in Agriculture: Forecasting prices of Arecanuts in Kerala," Procedia Comput. Sci., vol. 171, pp. 699–708, 2020, doi: 10.1016/j.procs.2020.04.076.
[6] S. Muzaffar and A. Afshari, "Short-Term Load Forecasts Using LSTM Networks," Energy Procedia, vol. 158, pp. 2922–2927, Feb. 2019, doi: 10.1016/j.egypro.2019.01.952.
[7] M. Sarkar and A. De Bruyn, "LSTM Response Models for Direct Marketing Analytics: Replacing Feature Engineering with Deep Learning," J. Interact. Mark., vol. 53, pp. 80–95, Feb. 2021, doi: 10.1016/j.intmar.2020.07.002.
[8] S. Helmini, N. Jihan, M. Jayasinghe, and S. Perera, "Sales forecasting using multivariate long short term memorynetwork models," PeerJ Prepr., vol. 7, pp. 1–16, 2019, doi: https://doi.org/10.7287/peerj.preprints.27712v1.
[9] A. Khumaidi, "Data Mining For Predicting The Amount Of Coffee Production Using CRISP-DM Method," J. Techno Nusa Mandiri, vol. 17, no. 1, pp. 1–8, Feb. 2020, doi: 10.33480/techno.v17i1.1240.
[10] R. Nisbet, G. Miner, and K. Yale, "A Data Preparation Cookbook," in Handbook of Statistical Analysis and Data Mining Applications, Elsevier, 2018, pp. 727–740.
[11] R. Indrakumari, T. Poongodi, and S. R. Jena, "Heart Disease Prediction using Exploratory Data Analysis," Procedia Comput. Sci., vol. 173, pp. 130–139, 2020, doi: 10.1016/j.procs.2020.06.017.

[12] W. Li, B. Wang, J. Liu, G. Zhang, and J. Wang, "IGBT aging monitoring and remaining lifetime prediction based on long short-term memory (LSTM) networks," Microelectron. Reliab., vol. 114, p. 113902, Nov. 2020, doi: 10.1016/j.microrel.2020.113902.
[13] C. Panem, V. R. Gad, and R. S. Gad, "Sensor's data transmission with BPSK using LDPC (Min-Sum) error corrections over MIMO channel: Analysis over RMSE and BER," Mater. Today Proc., vol. 27, pp. 571–575, 2020, doi: 10.1016/j.matpr.2019.12.039.
[14] S. Lightstone, T. Teorey, and T. Nadeau, "Denormalization," in Physical Database Design, Elsevier, 2007, pp. 337–355.
[15] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," Appl. Soft Comput., vol. 97, p. 105524, Dec. 2020, doi: 10.1016/j.asoc.2019.105524.