ISSN 2085-4552

Komparasi Metode *Multilayer Perceptron* (MLP) dan *Long Short Term Memory* (LSTM) dalam Peramalan Harga Beras

Steven Sen[1], Dedy Sugiarto[2], Abdul Rochman[3]
[1, 3] Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Trisakti, Jakarta, Indonesia
stev.strife@gmail.com
[2] Program Studi Sistem Informasi, Fakultas Teknologi Industri, Universitas Trisakti, Jakarta, Indonesia

Diterima 21 April 2020
Disetujui 17 Juni 2020

***Abstract*—Rice is one of the main commodities in Indonesian society. The main problem with rice nationally is inflation of rice prices. Therefore, this research predicts the price of rice using Multilayer Perceptron (MLP) artificial neural network architecture and deep learning: Long Short Term Memory (LSTM) to anticipate these problems. The data used in this study are real data on rice prices during 2016 - 2019 obtained from PT. Food Station. The total dataset is 1307 with the distribution of 1123 as data train and 184 as test data. The final results obtained in this study are LSTM superior to MLP, with the value of Root Mean Square Error (RMSE) training data: 0.49 RMSE loss value of test data is 0.27. The most optimal LSTM model from 3 tests conducted, is a model with a number of hidden layers = 16 and epochs = 150 times.**

***Index Terms*—Artificial Neural Network, Deep Learning, Forecasting Time Series, Long Short Term Memory (LSTM), Multilayer Perceptron (MLP)**

I. PENDAHULUAN

Beras merupakan komoditi pokok bagi masyarakat Indonesia sampai saat ini. Prediksi terhadap harga beras sangat diperlukan untuk mewujudkan stabilisasi harga bahan pokok guna tercapainya ketahanan pangan nasional, terutama pada masa pandemi Covid-19 seperti sekarang. Tentunya pada masa sulit ini, diharapkan harga-harga pangan tetap stabil untuk menunjang kehidupan masyarakat, terutama beras. Data harga beras yang dihimpun pemerintah tiap tahunnya, termasuk dalam tipe data *time series*.

Banyak metode *machine learning* tersedia untuk memprediksi data *time series*, baik yang memiliki arsitektur *shallow* seperti : *Principal Component Analysis* (PCA), Isomap, dan *Support Vector Machine* (SVM), maupun arsitektur *deep* seperti: *Convolutional Neural Network* (CNN), *Multilayer Perceptron* (MLP), dan *Long Short Term Memory* (LSTM).

Multilayer Perceptrons (MLP) adalah arsitektur *perceptron* yang paling banyak digunakan untuk jaringan saraf. Lapisan *perceptron* digabungkan dan membentuk arsitektur *multilayer*, hal ini memberikan kompleksitas yang diperlukan dari pemrosesan jaringan saraf. Dengan menambahkan lebih banyak lapisan dan lebih banyak neuron per lapisan, akan meningkatkan spesialisasi model untuk melatih data [1].

LSTM merupakan sebuah arsitektur RNN [2] yang memiliki *memory cell*. Dengan *memory cell*, arsitektur LSTM dapat bekerja lebih baik dibanding jaringan saraf rekuren biasa, karena memiliki kemampuan untuk mengingat informasi untuk periode waktu yang lebih lama, sehingga menjadikannya algoritma yang lebih baik untuk prediksi data berjenis *time series* [3].

Penelitian ini bertujuan untuk melakukan prediksi dan komparasi model terhadap data *time series* harga beras jenis IR-64 kualitas III selama 24 bulan pada tingkat grosir yang didapatkan dari *website* PT.Food Station. Metode yang akan digunakan adalah komparasi antara jaringan saraf tiruan yaitu *Multilayer Perceptron* (MLP) dan *deep learning* dengan arsitektur *Long Short Term Memory* (LSTM) yang akan dibangun dengan environment berbasis bahasa *python*. Hal ini termasuk sebagai inovasi baru dalam perbandingan kedua model jaringan syaraf tiruan dalam bidang deep learning peramalan data time series terhadap permasalahan yang dibahas, mengingat semakin besarnya data dan kompleksnya kebutuhan akan pengolahan data.

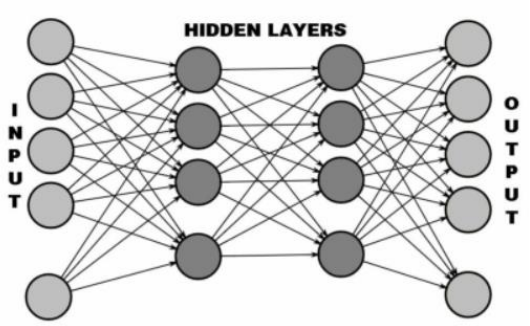
Proses dalam penelitian ini adalah mencari nilai prediksi dengan akurasi terbaik antara model MLP dan LSTM, serta membuat plot perbandingan data test dengan data hasil prediksi. Tingkat keakuratan dari tiap model, akan diukur menggunakan algoritma *Root Mean Squared Error* (RMSE) sesuai skenario yang penulis buat.

II. TINJAUAN PUSTAKA

A. *Multilayer Perceptron (MLP)*

Multilayer merupakan bentuk lapisan perceptron yang digabungkan, dengan menambahkan lebih banyak layer dan neuron tiap layer. *Multilayer*

Perceptron (MLP) sendiri merupakan arsitektur yang paling banyak digunakan untuk jaringan saraf.

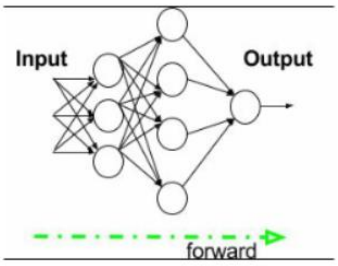


Gambar 1. *Deep neural network* MLP [1]

Pemrosesan dari *layer input* ke *hidden layer* dan kemudian ke lapisan *output* adalah disebut *forward propagation*. Jumlah (*input* * bobot) + bias diterapkan pada setiap *layer* dan kemudian nilai fungsi aktivasi disebarkan ke *layer* berikutnya. Lapisan selanjutnya bisa berupa *hidden layer* lain atau *output layer* [1].

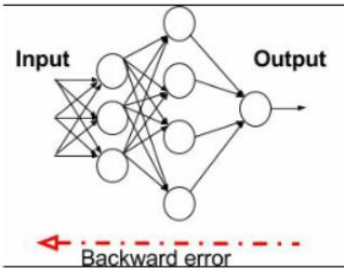
MLP dilatih dari data pelatihan melalui proses yang disebut backpropagation. Proses ini dapat digambarkan sebagai cara untuk memperbaiki kesalahan secara progresif segera setelah terdeteksi.

Pada awalnya, semua bobot ditetapkan secara acak. Kemudian jaringan diaktifkan untuk setiap input dalam set pelatihan: nilai disebarkan ke depan (*forward propagation*) dari tahap input melalui tahap tersembunyi ke tahap *output* di mana prediksi dibuat.



Gambar 2. *Feed forward*

Karena nilai *real* yang diamati dalam set pelatihan diketahui, maka memungkinkan untuk menghitung kesalahan yang dibuat dalam prediksi. Proses kerja utama dalam *backtracking* adalah melakukan alur kembali dari *output* menuju *input* dengan menggunakan algoritma pengoptimal yang tepat, seperti *gradient descent*, untuk menyesuaikan bobot (*weight*) jaringan saraf dengan tujuan mengurangi kesalahan.



Gambar 3. *Backpropagation*

Proses *feed propagation* dari *input* ke *output* dan *backpropagation* diulang beberapa kali sampai kesalahan mencapai nilai di bawah ambang batas yang telah ditentukan [4].

B. Recurrent Neural Network (RNN)

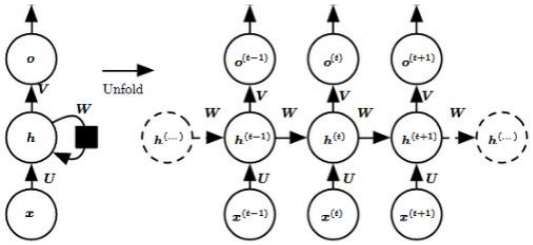
Recurrent Neural Network (RNN) adalah jenis arsitektur *Neural Network* (jaringan saraf tiruan) dimana *output* dari *hidden layer* akan menjadi input bagi pemrosesan berikutnya. Ciri RNN adalah melakukan prediksi terhadap data tidak hanya menggunakan input pada satu waktu saja, akan tetapi juga membutuhkan masukan dari input sebelumnya, sehingga antar *input* saling berhubungan dan menggunakan hubungan tersebut untuk memberikan informasi ke seluruh *hidden layer* yang ada dalam RNN. RNN memiliki memori yang berisikan hasil rekaman informasi yang dihasilkan sebelumnya. Persamaan yang terbentuk adalah forward propagation dalam RNN (1), *hidden layer* (2), *output gate* (3), dan target (4).

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)} \quad (1)$$

$$h^{(t)} = \tanh(a^{(t)}) \quad (2)$$

$$o^{(t)} = c + Vh^{(t)} \quad (3)$$

$$\hat{y}^{(t)} = softmax(o^{(t)}) \quad (4)$$



Gambar 4. *Recurrent Neural Network* dan formulasinya [5]

Satu masalah pada arsitektur RNN adalah masalah menghilangnya gradien (*vanishing problem*) pada proses *backward pass* [4]. Cara yang cukup populer

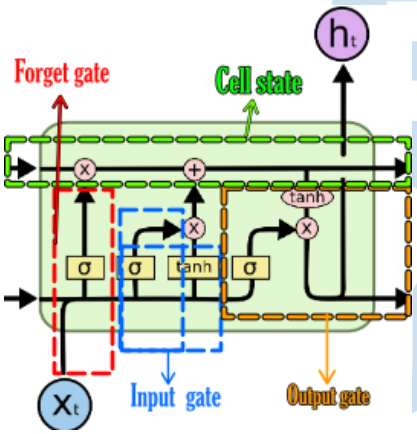
untuk mengatasi *vanishing problem* adalah dengan menggunakan unit RNN yaitu *Long Short Term Memory* (LSTM) [6].

C. Long Short Term Memory (LSTM)

LSTM adalah jenis RNN khusus yang bekerja lebih baik dalam praktiknya, dikarenakan adanya pembaruan dari persamaan dan adanya dinamika *backpropagation* didalamnya [7]. Arsitektur *Long Short Term Memory network* (LSTM), seperti terlihat pada gambar 3, memiliki gate yang berfungsi untuk menghapus maupun menambah informasi yaitu *forget gate, output gate,* dan *input gate*. [8].

Persamaan yang terbentuk dari *forget gate* (1), *input gate* (2), *output gate* (3), lapisan tanh menciptakan vektor nilai kandidat baru (4), *cell state* (5), dan *hidden layer* (6) adalah:

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \quad (1)$$
$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \quad (2)$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (3)$$
$$\tilde{C}_t = \tanh(W_C.[h_{t-1}, x_t] + b_C) \quad (4)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (5)$$
$$h_t = o_t * \tanh(C_t) \quad (6)$$



Gambar 5. Unit dari LSTM [6]

- *Forget Gate*: secara kondisional memutuskan apakah suatu informasi harus dibuang atau tidak dari pemrosesan.
- *Input Gate*: secara kondisional untuk menentukan sebuah masukan akan ditambahkan ke dalam memori *cell state* saat itu atau tidak [9].
- *Cell state* ini berfungsi sebagai memori atau ingatan untuk sebuah layer. Nilai suatu *cell state* dimanipulasi menggunakan sebuah sistem gerbang.
- *Output Gate*: secara kondisional memutuskan apa yang akan dihasilkan berdasarkan input dan memori blok [6].

D. Deep Learning with Keras

Keras adalah *framework deep learning* untuk *Python* yang menyediakan cara mudah untuk mendefinisikan dan melatih hampir semua jenis model *deep-learning*. Keras awalnya dikembangkan untuk para peneliti, dengan tujuan memungkinkan eksperimen berjalan lebih cepat. Sekarang, Keras adalah *framework* terpopuler di Kaggle dan berbagai kompetisi *deep learning* dunia [10]. *Framework* ini mendukung 2 bahasa *data science* yang paling populer yaitu: R and *Python* [11]. Kelebihan yang ditawarkan adalah dengan munculnya fitur-fitur : membuat *code* berjalan mulus pada CPU atau GPU, *user-friendly* API yang membuatnya mudah untuk membuat prototipe sebuah *deep-learning*, dan mendukung model RNN [10].

III. METODE PENELITIAN

Dalam membangun model LSTM ini, membutuhkan beberapa tahapan proses. Proses-proses tersebut terdiri dari proses *preprocessing* data, inisialisasi parameter, *training* jaringan LSTM, dan melakukan uji terhadap data *testing*. Dalam pembangunan sistem, dataset yang didapatkan diolah terlebih dahulu dengan menggunakan teknik normalisasi *min max scaling*. Dilakukan inisialisasi pada setiap parameter, setelah itu dilakukan *training* pada arsitektur jaringan saraf yang dibuat sesuai dengan parameter yang telah ditetapkan. Selanjutnya dilakukan uji pada model LSTM, hasil dari proses training terhadap data testing. Proses tersebut terus diulang hingga mendapatkan model dengan akurasi yang baik.

A. Tahap Preprocessing Data

Dataset yang digunakan adalah data *real* harga beras dari tanggal 1 Januari 2016 s.d 31 Januari 2020. Untuk meminimalkan *error*, dilakukan normalisasi pada dataset dengan mengubah data aktual menjadi nilai dengan *range* interval [0,1]. Teknik normalisasi yang digunakan adalah dengan *min-max scaling*.

Dataset kemudian dibagi dalam data *train* dan tes. Data *train* adalah data dari tanggal 1 Januari 2016 sampai 31 Juli 2019. Sedangkan data tes adalah sisanya, yaitu data dari tanggal 1 Agustus 2019 s.d 31 Januari 2020.

Tabel 1. Pembagian data

Data Latih	Data Tes
1245 Data	184 Data
01-01-2016 s.d 31-07-2019	01-08-2019 s.d 31-01-2020
Total data : 1429	

Data *train* dan tes di-*reshape* menjadi 3 dimensi.

B. Inisialisasi Parameter

B.1 Model MLP

Setelah *dataset* dilakukan *preprocessing*, dan selanjutnya menentukan inisialisasi parameter dasar. Pada MLP yang dibutuhkan antara lain:

Tabel 2. Parameter model MLP

Paramater	Jumlah
Total *layer*	5
Input layer	1 *layer* (64 neuron)
Hidden layer	3 *layer* (128,128,8 neuron)
Output layer	1 neuron
Jumlah *epochs*	100

B.2 Model LSTM

Inisialisasi parameter pada LSTM yang akan dibangun yaitu:

Tabel 3. Parameter model LSTM

Paramater	Jumlah
Total *layer*	4
Hidden layer	1 *layer*, 64 neuron
Jumlah LSTM *cell* (*input layer*)	128 neuron
Output layer	1 neuron
Jumlah *epochs*	100

C. Tahap Analisis dan Perancangan Model

C.1 Model MLP

Tahap analisis dan perancangan model dilakukan setelah data melewati tahapan *preproccesing*. Model yang digunakan di penelitian ini, yaitu MLP.

- Digunakan *dataset train* untuk melatih model MLP. Pada tahap awal, model dibangun dengan memanggil fungsi *sequential*.
- Kemudian ditambahkan 1 buah *layer input* MLP dengan *dense* 64 neuron, *input dimension* = 1, dan activation = 'relu'.
- Ditambahkan 3 *hidden layer* dengan jumlah neuron masing-masing 128, 128, dan 8.
- Pada *layer* terakhir, yaitu *output layer* dengan 1 buah neuron.
- Digunakan *loss function* yaitu RMSE dan *optimizer* Adam.
- Terakhir model dilatih dengan menggunakan fungsi fit, yang menggunakan parameter data *train*.

C.2 Model LSTM

Tahap analisis dan perancangan model dilakukan setelah data melewati tahapan *preproccesing*. Model yang digunakan kali ini yaitu LSTM dengan detail sebagai berikut:

- Digunakan *dataset train* untuk melatih model LSTM. Pada tahap awal, model dibangun dengan menggunakan fungsi *sequential*.
- Kemudian dibuat model dengan 1 buah *layer* LSTM sebagai *input layer*, dengan dimensi (1,1), 1 *hidden layer* dengan 64 neuron, dan *output* 1 neuron.
- Digunakan *loss function* yaitu RMSE dan *optimizer* Adam.
- Terakhir model dilatih dengan menggunakan fungsi fit, yang menggunakan parameter data *train*.

Tahap analisis dan perancangan model dilakukan setelah data melewati tahapan *preproccesing*. Model yang digunakan di penelitian ini, yaitu MLP.

D. Testing

Model yang telah dilatih pada proses *training*, akan diuji dengan menggunakan data *test* dari proses *preprocessing* data. Model akan menghasilkan nilai prediksi harga beras. Dilakukan juga proses perhitungan nilai RMSE pada hasil data *train* dan data *test.*

D.1 Skenario Pengujian MLP

Skenario pengujian dalam penelitian ini adalah menganalisa dampak tiap parameter terhadap akurasi yang didapatkan. Parameter yang diuji berupa jumlah neuron pada *input layer* dan besarnya *epoch* maksimum untuk menghasilkan bobot MLP yang optimal. Berikut adalah nilai parameter yang akan diuji:

- Jumlah neuron *input* : 64,128, 256
- Jumlah *Epoch* : 100, 150, 200

Dilakukan observasi sebanyak 3 kali. Hal ini dilakukan karena tidak cukup dilakukan observasi hanya sekali saja, karena hasil yang didapatkan bisa saja kebetulan baik ataupun buruk.

D.2 Skenario Pengujian LSTM

Skenario pengujian dalam penelitian ini adalah menganalisa dampak tiap parameter terhadap akurasi yang didapatkan. Parameter yang diuji berupa jumlah neuron pada *hidden layer* dan besarnya *epoch* maksimum untuk menghasilkan bobot LSTM yang optimal. Berikut adalah nilai parameter yang akan diuji:

- Jumlah neuron *hidden* : 64,128, 256
- Jumlah *Epoch* : 100, 150, 200

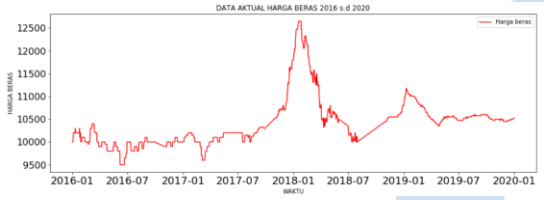
Dilakukan observasi sebanyak 3 kali. Hal ini dilakukan karena tidak cukup dilakukan observasi hanya sekali saja, karena hasil yang didapatkan bisa saja kebetulan baik ataupun buruk.

E. Membandingkan Data Hasil Prediksi dan Plotting Data

Tahap akhir yaitu membandingkan data hasil prediksi model LSTM dengan data aktual dalam sebuah tabel. Nilai train dan prediksi juga divisualisasikan dalam bentuk grafik plot.

IV. HASIL DAN PEMBAHASAN

Berdasarkan data *real* harga beras yang terdapat pada FoodStation, dapat dilihat bahwa terdapat kenaikan harga setiap awal tahun dibanding bulan lainnya terutama pada tahun 2018. Lonjakan tertinggi terdapat pada awal tahun 2018 yang mencapai harga lebih dari 12.500, diikuti lonjakan awal tahun 2019 yang juga cukup tinggi.



Gambar 6. Visualisasi data beras tanggal 1 Januari 2016 s.d 31 Januari 2020

Sedangkan harga beras cenderung menurun pada pertengahan tahun. Harga beras pada pertengahan tahun 2019 sampai awal tahun 2020 cenderung stabil. Dapat ditarik informasi bahwa pada awal tahun dan akhir tahun harga beras cenderung meningkat.

A. Model MLP

Tabel 4. Kombinasi uji coba MLP

Uji Coba ke -	Jumlah neuron *hidden layer*	*epochs*	Nilai RMSE *Train*	Nilai RMSE Tes
1	64	100	52.52	21.14
2	128	150	95.38	39.32
3	256	200	88.98	50.86

Berdasarkan lampiran Tabel 4, menunjukkan hasil bahwa model MLP dengan jumlah neuron *input layer*= 64 dan *epochs*= 100, menghasilkan nilai RMSE yang lebih kecil dibandingkan dengan percobaan lainnya. Bisa dikatakan bahwa percobaan pertama lebih optimal dibanding dengan kedua percobaan lainnya. Jumlah *input layer* mengolah data *input* dan menghubungkannya dengan neuron *output*. Tidak ada aturan pasti mengenai jumlah *hidden* neuron yang paling optimal dalam memprediksi data *time series* contohnya harga beras ini, maka dilakukanlah percobaan untuk mengetahui model paling optimal.

B. Model LSTM

Tabel 5. Kombinasi uji coba LSTM

Uji Coba ke -	Jumlah neuron *hidden layer*	*epochs*	Nilai RMSE *Train*	Nilai RMSE Tes
1	64	100	0.52	0,35
2	128	150	0.49	0.27
3	256	200	0.59	0.37

Berdasarkan lampiran Tabel 5, menunjukkan hasil bahwa model LSTM dengan jumlah neuron *hidden layer*= 128 dan *epochs*= 150, menghasilkan nilai RMSE yang lebih kecil dibandingkan dengan percobaan lainnya. Bisa dikatakan bahwa percobaan kedua lebih optimal dibanding dengan kedua percobaan lainnya. *Hidden layer* mengolah neuron *input* dan menghubungkannya dengan neuron *output*, sehingga jumlah *hidden neuron* akan menentukan nilai *output* yang dihasilkan oleh unit LSTM.

Tabel 6. Perbandingan data aktual dan prediksi MLP

Tanggal	Harga Aktual	Nilai Prediksi	Nilai akurasi RMSE
		MLP	MLP
01 Jan 2016	10000	9968,594	0,937
02 Jan 2016	10000	9968,594	0,937
03 Jan 2016	10000	9968,594	0,937
04 Jan 2016	10100	10034,286	4,945
05 Jan 2016	10200	10253,814	4,362
======	=====	=======	=======
27 Jan 2020	11225	11194,379	0,914
28 Jan 2020	11225	11194,379	0,914
29 Jan 2020	11250	11212,434	1,121
30 Jan 2020	11250	11212,434	1,121
31 Jan 2020	11250	11212,434	1,121

Dari tabel perbandingan data aktual dan prediksi MLP, menunjukkan bahwa MLP memprediksi nilai harga dengan rentang yang cukup besar terhadap harga aktual. Nilai *error* RMSE cukup besar menandakan bahwa model ini kurang efektif dalam mengolah data *time series* harga beras.

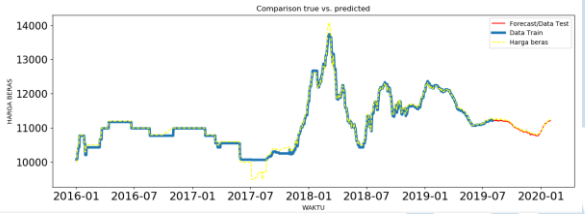
Hasil *training* model LSTM, maka akan dihasilkan nilai prediksi harga dan *loss* pada bulan selanjutnya yaitu sebagai berikut:

Tabel 7. Perbandingan data aktual dan prediksi LSTM

Tanggal	Harga Aktual	Nilai Prediksi	Nilai akurasi RMSE
		LSTM	LSTM
01 Jan 2016	10000	9999.547	0,014
02 Jan 2016	10000	9999.547	0,014
03 Jan 2016	10000	9999.547	0,014
04 Jan 2016	10100	10099.568	0,013
05 Jan 2016	10200	10199.624	0,011
======	======	======	======
27 Jan 2020	11225	11227,099	0,063
28 Jan 2020	11225	11227,099	0,063
29 Jan 2020	11250	11252,168	0,065
30 Jan 2020	11250	11252,168	0,065
31 Jan 2020	11250	11252,168	0,065

Tabel perbandingan data aktual dan prediksi LSTM, menunjukkan bahwa LSTM berhasil memprediksi nilai harga dengan optimal dan mendekati harga aktual. Nilai *error* RMSE yang sangat kecil menandakan bahwa model ini cukup efektif dalam mengolah data *time series* harga beras. Terkait nilai MSE yang optimal, adalah tergantung dari data dan model yang dibangun.

Hasil *plotting data train* dan tes pada model MLP yang dibangun.



Gambar 7. Visualisasi data *train* dan tes MLP

Visualisasi data *train (biru)*, *tes (merah)*, dan data harga aktual (kuning) menunjukkan bahwa harga beras cenderung berubah secara signifikan terutama pada tahun 2018. Menurut *training* dan *tes*ting MLP, harga 6 bulan kedepan cenderung fluktuatif yaitu dari 1 Agustus 2019 s.d 31 Januari 2020 akan berkisar antara 10.600 – 11.100. Garis harga prediksi dan aktual tidak nampak ada perbedaan, karena perbedaan nilai yang sangat kecil.

Hasil plotting data *train* dan *tes* pada model LSTM yang dibangun.

Gambar 8. Visualisasi data *train* dan tes LSTM

Visualisasi data *train* (biru), tes (merah), dan data harga aktual (kuning) menunjukkan bahwa harga beras cenderung tidak berubah secara signifikan seperti pada tahun 2018. Menurut prediksi LSTM, harga 6 bulan kedepan yaitu dari 1 Agustus 2019 s.d 31 Januari 2020 akan berkisar antara 10.300 – 10.700. Garis harga prediksi dan aktual tidak nampak ada perbedaan, karena perbedaan nilai yang sangat kecil.

V. SIMPULAN

Prediksi terhadap data *time series* harga beras pada PT. FoodStation dirancang dengan menggunakan perbandingan antara dua model *Multilayer Perceptron* (MLP) dan arsitektur *Long Short Term Memory* (LSTM). Sebagai data percobaan, digunakan data harga beras tanggal 1 Januari 2019 s.d 31 Juli 2019. Model MLP dan LSTM dibangun dengan *library* Keras dan menggunakan *loss function* yaitu RMSE.

Pada tahap *forecasting*, digunakan data 6 bulan terakhir harga beras sebagai data *testing* yang menghasilkan output prediksi harga dari kedua model. Perbandingan hasil *output*, menghasilkan kesimpulan bahwa model LSTM lebih akurat jika dilihat dari rentang dengan harga aktual dan nilai RMSE-nya yang lebih kecil dibandingkan MLP. Pada akhirnya, didapatkan nilai *loss* RMSE pada LSTM, yaitu *training data score*: 0.49 RMSE, *test data score*: 0.27 RMSE, dan hasil grafik plot yang menunjukkan bahwa tidak ada perbedaan signifikan antara hasil *train* dan *testing* dengan data aktual.

Penelitian ini, diharapkan dapat menjadi referensi pilihan metode untuk pengembangan penelitian lainnya dan dikembangkan lagi agar menghasilkan metode yang lebih akurat terutama dalam bidang *forecasting deep learning*.

DAFTAR PUSTAKA

[1] G. Ciaburro and B. Venkateswaran, Neural network with R, Packt Publishing, 2017.

[2] W. Ahmed and M. B. , "The Accuracy of the LSTM Model for Predicting the S&P 500 index and the Difference Between Prediction and Backtesting," Degree Project in Technology, 2018.

[3] H. Prasetyanwar and J. , "Peramalan Nilai Tukar IDR-USD Menggunakan Long Short Term Memory," e-Proceeding of Engineering : Vol.5, No.2 Agustus 2018 , vol. 5, p. 3820, 2018.

[4] A. GUlli and S. P. , Deep Learning with Keras, Birmingham: Packt Publishing Ltd, 2017.

[5] J. Nabi, "Recurrent Neural Networks (RNNs)," [Online]. Available: https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85.

[6] C. Olah, "Understanding LSTM Networks," 27 August 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[7] A. Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks," 21 May 2015.

[8] J. Brownlee, "Time Series Prediction with LSTM Recurrent Neural Network in Python with Keras," 2016.

[9] L. Zaman, S. S. and M. H. , "Analisis Kinerja LSTM dan GRU sebagai Model Generatif," vol. 8, p. 143, 2019.

[10] F. Chollet, Deep Learning with Python, New York: Manning, 2018.

[11] F. Chollet and J. J. Allaire, Deep Learning with R, vol. 1, Manning Publications, 2017.