PERBANDINGAN PENERAPAN LONG SHORT TERM MEMORY DAN BIDIRECTIONAL LONG SHORT TERM MEMORY UNTUK PREDIKSI HARGA BITCOIN

TUGAS AKHIR

Markus Clarent Calvianto
1118005

PROGRAM STUDI INFORMATIKA
INSTITUT TEKNOLOGI HARAPAN BANGSA
BANDUNG
2022

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Cryptocurrency merupakan sebuah jaringan *peer-to-peer* yang terenkripsi untuk transaksi digital. *Bitcoin* merupakan salah satu *cryptocurrency* yang ditemukan sejak tahun 2008. *Bitcoin* muncul sebagai mata uang digital dalam kapitalisasi pasar dan terus menerus menarik perhatian investor. Dalam beberapa tahun terakhir, *bitcoin* memiliki volatilitas harga yang tinggi. *Bitcoin* mengalami kenaikan pada 2016, diikuti dengan penurunan yang signifikan pada tahun 2018 [1]. Salah satu kebutuhan yang muncul di masyarakat adalah adanya sebuah sistem yang mampu memberikan rekomendasi atau memprediksi dalam investasi *bitcoin*. Aplikasi yang mampu memprediksi harga *bitcoin* digunakan untuk memberikan rekomendasi dalam pengambilan keputusan berinvestasi.

Pada saat ini, pendekatan untuk prediksi data deret waktu banyak dikembangkan. Penelitian yang dilakukan oleh Azari menerapkan *autoregressive integrated moving average* (ARIMA) untuk memprediksi harga *bitcoin*. Model ARIMA diuji menggunakan *dataset bitcoin* dengan periode 3 tahun dimulai dari tahun 2015. *Dataset* tersebut terlebih dahulu dibuat lebih stasioner dikarenakan model ARIMA membutuhkan data yang stasioner. Hasil *mean squarred error* (MSE) dari model ARIMA yaitu 45 [2].

Penelitian yang dilakukan oleh Munim membandingkan ARIMA dan *Neural Network Autoregression* (NNAR) untuk memprediksi harga *bitcoin*. Penelitian tersebut menggunakan *dataset* harian *bitcoin* periode Januari 2014 hingga Oktober 2018. *Dataset* tersebut terlebih dahulu dibuat stasioner dengan dilakukan transformasi log. Hasil *root mean squarred error* (RMSE) dari model ARIMA dan NNAR adalah 0.037 dan 0.042 [3].

Selain metode tradisional, pendekatan *deep learning* digunakan untuk memprediksi data deret waktu. Penelitian yang dilakukan oleh Jay membandingkan beberapa metode *deep learning* seperti *multilayer perceptron* (MLP) dan *long short term memory* (LSTM) untuk memprediksi harga *bitcoin*. Hasil dari penelitian tersebut yaitu LSTM menghasilkan RMSE sebesar 0.06083 sedangkan MLP menghasilkan RMSE sebesar 0.06921. Nilai RMSE LSTM yang lebih kecil menunjukkan bahwa LSTM mampu memprediksi harga *bitcoin* dengan lebih baik dibandingkan MLP [4].

Penelitian lainnya menunjukkan bahwa metode *bidirectional long short term memory* (BiLSTM) mampu memprediksi harga *bitcoin* lebih baik dibandingkan LSTM. Dengan regularisasi *dropout* model BiLSTM mampu memprediksi harga *bitcoin* dengan RMSE sebesar 387.036 sedangkan model LSTM menghasilkan RMSE sebesar 392.752 [5].

Berdasarkan paparan sebelumnya penelitian ini membandingkan metode LSTM dengan BiLSTM untuk memprediksi harga penutupan *bitcoin*. Penelitian dilakukan menggunakan *dataset bitcoin* untuk memprediksi harga penutupan *bitcoin* dalam interval jam. Regularisasi *dropout* digunakan untuk mencegah terjadinya *overfitting* terhadap model. Pengujian kinerja model LSTM dan BiLSTM menggunakan nilai RMSE untuk mengukur *prediction error*.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka di identifikasi rumusan masalah sebagai berikut :

1. Berapa nilai akurasi dalam penerapan LSTM dan BiLSTM dalam memprediksi harga *bitcoin* ?
2. Berapa nilai unit, *dropout*, *epoch*, *batch size* yang optimal untuk mencapai nilai akurasi yang maksimal ?

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah :

1. Data yang digunakan yaitu *Open*, *Close*, *High*, *Low*, Volume(BTC), Volume(USD) yang diambil dari Bitcoin Historical Data dalam rentang waktu 1 jam.
2. *Dataset* yang digunakan periode Mei 2018 hingga Februari 2022.

1.4 Tujuan Penelitian

Penelitian ini memiliki tujuan sebagai berikut :

1. Menguji tingkat akurasi algoritme LSTM dan BiLSTM dalam memprediksi harga *bitcoin*
2. Mengidentifikasi nilai unit, *dropout*, *epoch*, *batch size* yang optimal untuk mencapai nilai akurasi yang maksimal

1.5 Kontribusi Penelitian

Kontribusi yang diberikan dari penelitian ini yaitu membuktikan algoritme manakah yang lebih baik untuk memprediksi harga *bitcoin*. Hal tersebut dilakukan dengan menguji dan menganalisis algoritme LSTM dan BiLSTM.

1.6 Metodologi Penelitian

Metode penelitian yang digunakan dalam penelitan sebagai berikut :

1. Studi Literatur
   Sebelum melakukan penelitian dilakukan studi literatur dengan cara mengulas bahan referensi dari jurnal, paper, dan buku yang berhubungan dengan prediksi harga *bitcoin*.

2. Pengumpulan Data
   Data sampel yang digunakan diambil dari Bitcoin Historical Dataset, data yang ada berupa data tabular.

3. Analisis Masalah
   Pada tahap ini dilakukan analisis masalah, batasan masalah, dan kebutuhan yang diperlukan.

4. Perancangan dan Implementasi Algoritme
   Pada tahap ini dilakukan implementasi LSTM dan BiLSTM untuk menyelesaikan masalah yang ada.

5. Pengujian
   Pada tahap ini dilakukan pengujian terhadap implementasi yang telah dilakukan.

6. Dokumentasi
   Pada tahap ini dilakukan dokumentasi terhadap hasil analisis dan implementasi yang telah dilakukan dalam bentuk laporan.

1.7 Sistematika Pembahasan

BAB 1 : PENDAHULUAN

Bab ini berisi uraian dari latar belakang , rumusan masalah, tujuan penelitian, batasan masalah, kontribusi penelitian, dan metode yang digunakan dalam penelitian ini yaitu LSTM dan BiLSTM.

BAB 2 : LANDASAN TEORI

Bab ini berisi kajian teori dan referensi yang mendukung dalam penelitian ini,

diantaranya adalah teori mengenai algoritme LSTM dan BiLSTM.

BAB 3 : ANALISIS DAN PERANCANGAN

Bab ini berisi analisis kerangka pemikiran, algoritme LSTM dan BiLSTM yang digunakan dalam penelitian dan analisis data sampel.

BAB 4 : IMPLEMENTASI DAN PENGUJIAN

Bab ini berisi cara implementasi dan pengujian yang dilakukan dalam penelitian ini. Bab ini meliputi lingkungan implementasi, implementasi aplikasi, hasil pengujian algoritme LSTM dan BiLSTM, dan analisis kesalahan.

BAB 5 : KESIMPULAN DAN SARAN

Bab ini berisi penutup berupa kesimpulan yang dihasilkan dari penelitian ini dan saran untuk perkembangan penelitian ini ke depan.

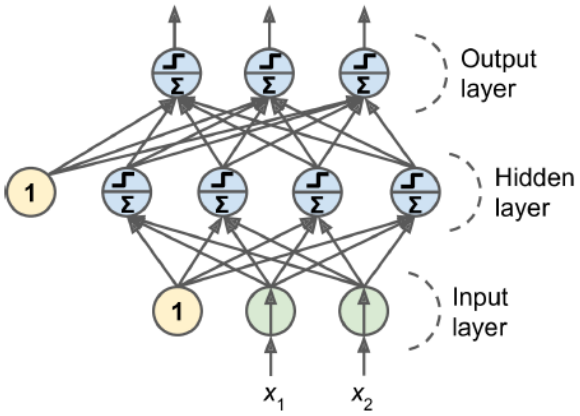
BAB 2 LANDASAN TEORI

2.1 Tinjauan Pustaka

Pada bagian ini dijelaskan teori yang terkait dalam penelitian. Teori yang dibahas meliputi *artificial neural network*, pendekatan *deep learning* untuk data deret waktu, dan beberapa teori spesifik berikut dengan formula yang digunakan dalam penelitian.

2.1.1 *Artificial Neural Network*

Artificial Neural Network (ANN) merupakan sebuah metode pembelajaran mesin yang terinspirasi dari saraf yang ada pada otak manusia. *Perceptron* merupakan arsitektur yang paling sederhana dari ANN. Perceptron memiliki *threshold logic unit* (TLU) yang bisa menerima masukan dan memberikan keluaran berupa angka, dan setiap masukan memiliki bobot tersendiri. Sebuah *perceptron* berisi satu buah lapisan TLU, dimana setiap TLU terhubung dengan seluruh masukan. Ketika semua *neuron* di dalam lapisan terhubung dengan seluruh *neuron* di lapisan sebelumnya, maka lapisan tersebut dinamakan *fully connected layer* atau *dense layer*. *Multi Layer Perceptron* (MLP) terdiri dari satu *input layer*, satu atau lebih TLU yang dinamakan *hidden layer*, dan satu *output layer* [6].



Gambar 2.1 *Multi Layer perceptron* [7]

Gambar 2.1 menunjukkan bahwa pada MLP semua lapisan kecuali *output layer* memiliki bias neuron dan saling terhubung dengan lapisan berikutnya. Alur sinyal dalam MLP hanya satu arah yaitu dari *input layer* menuju *output layer*, maka dari itu disebut juga sebagai *feedforward neural network*. Langkah selanjutnya ketika keluaran telah dihasilkan oleh MLP yaitu meminimalkan nilai *error* dengan melakukan *backpropagation*. *Backpropagation* meminimalkan nilai

error dengan mencari tahu nilai bobot mana yang harus dirubah dalam *neuron* untuk meminimalkan nilai *error*. Hal ini diulang hingga memberikan hasil yang optimal [7].

2.1.2 *Recurrent Neural Network*

Recurrent neural network (RNN) dapat dilihat sebagai *neural network* yang bergerak secara maju, tetapi ia juga memiliki sebuah koneksi yang menunjuk kembali ke unit tersebut. Keluaran dan masukan RNN dipengaruhi oleh perhitungan sebelumnya dikarenakan ada sebuah koneksi yang menunjuk kembali ke *unit* tersebut. Hal inilah yang membedakan RNN dengan ANN di mana keluaran ANN tidak dipengaruhi oleh perhitungan sebelumnya [6].

Gambar 2.2 *Recurrent Neural Network* [6]

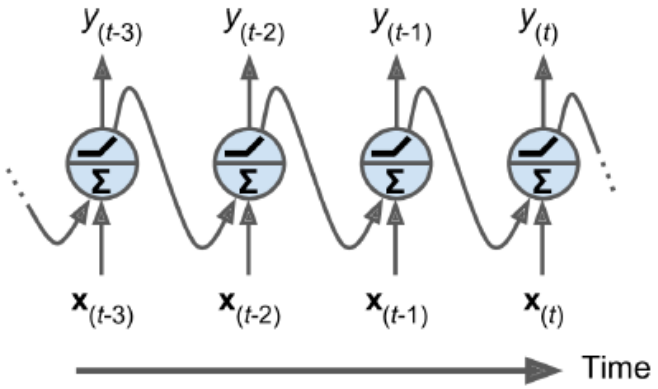
Keterangan :

x = masukan pada waktu t

y = keluaran pada waktu t

$\sum$ = *weighted sum*

Gambar 2.2 menunjukkan bahwa RNN menerima *input* x dan memberikan keluaran y pada waktu t. Berikut merupakan gambar ketika pengulangan RNN dilakukan :



Gambar 2.3 *Unrolled Recurrent Neural Network* [6]

Gambar 2.3 menunjukkan setiap *time step t*, RNN menerima masukan $x_{(t)}$ dan juga keluaran dari *time step* sebelumnya yaitu $y_{(t-1)}$. Setiap RNN memiliki dua buah kelompok bobot, satu untuk masukan (x_t) dan satu untuk *time step* sebelumnya ($y_{(t-1)}$). Berikut merupakan persamaan untuk keluaran dari RNN :

$$y_(t) = \phi({W_{(x)}}^T x_{(t)} + {W_{(y)}}^T y_{(t-1)} + b) \tag{2.1}$$

Keterangan :

$y_{(t)}$ = keluaran pada waktu *t*

ϕ = fungsi aktivasi

${W_x}^T$ = bobot untuk x_t

x_t = data masukan pada waktu *t*

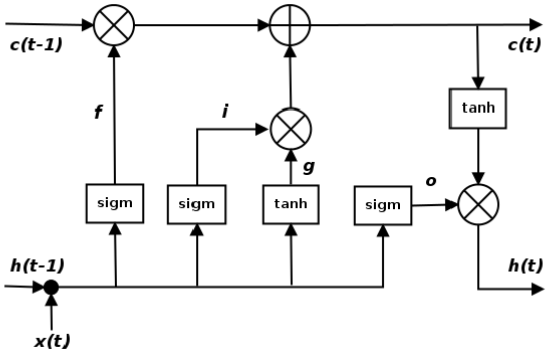
${W_y}^T$ = bobot untuk y_{t-1}

b = nilai bias

Keluaran dari setiap RNN pada *time step t* merupakan sebuah fungsi dari perhitungan dari *time step* sebelumnya dan memiliki bentuk *memory*. Bagian dari *neural network* yang menyimpan sebuah *state* lintas waktu disebut *memory cell*. Sebuah *recurrent neuron* merupakan sel sederhana yang mampu belajar untuk pola yang pendek. Sebuah RNN menerima deretan masukan dan menghasilkan deretan keluaran, hal inilah yang membuat RNN cocok untuk memprediksi kasus deret waktu seperti memprediksi harga saham [6].

2.1.3 *Long Short Term Memory*

LSTM merupakan sebuah varian dari RNN yang dapat belajar secara *long-term*. RNN dapat menggabungkan *hidden state* dari langkah waktu sebelumnya dengan *input* saat ini menggunakan lapisan *tanh* agar dapat melakukan pengulangan. LSTM juga mengimplementasikan pengulangan yang sama tetapi tidak dengan satu lapisan *tanh*, melainkan menggunakan empat buah lapisan yang saling berinteraksi [7].



Gambar 2.4 Struktur LSTM

Pada gambar 2.4 terdapat 2 buah vektor yaitu h_t dan c_t, h_t merupakan *short-term state* sedangkan c_t merupakan *long-term state*. LSTM ini dapat belajar untuk mengetahui apa yang harus di simpan dalam *long-term state*, informasi apa yang harus dibuang, dan informasi apa yang harus dibaca. Ketika *long-term state* c_{t-1} menelusuri jaringan dari kiri ke kanan melewati *forget gate* terlebih dahulu untuk membuang beberapa memori dan menambahkan memori baru melalui penambahan dengan hasil dari *input gate*. Hasil c_t diteruskan tanpa mengalami perubahan apa pun, maka dari itu setiap langkah waktu ada memori yang dibuang dan ditambahkan. Setelah operasi penambahan dilakukan, *long-term state* disalin dan melewati fungsi *tanh* lalu disaring oleh *output gate* dan menghasilkan *short-term* h_t.

Tahapan datangnya memori baru dan bagaimana cara *gate* tersebut bekerja adalah sebagai berikut. Pertama masukan vektor X_t sekarang dan *short-term state* sebelumnya h_{t-1} di masukan ke dalam empat lapisan yang berbeda dan saling terhubung yang memiliki fungsi masing - masing. Lapisan utama merupakan lapisan yang memberikan keluaran g_t. Lapisan ini memiliki fungsi untuk menganalisis masukan X_t dan *short-term state* sebelumnya yaitu h_{t-1}. Tiga lapisan lainnya merupakan *gate controller*. Karena ketiga lapisan tersebut menggunakan fungsi aktivasi *sigmoid* maka keluaran yang dihasilkan memiliki nilai antara 0 hingga 1. Dapat dilihat keluaran yang dihasilkan di masukan ke dalam operasi

perkalian, jika keluaran yang dihasilkan 0 maka *gate* ditutup sebaliknya jika 1 maka dibuka. *Forget gate* f_t mengontrol bagian mana dari *long-term state* yang harus dihapus. *Input gate* i_t mengontrol bagian mana dari g_t yang harus ditambahkan kedalam *long-term state*. *Output gate* o_t mengontrol bagian mana dari *long-term state* yang harus dibaca dan yang harus dikeluarkan. Secara singkat sebuah sel LSTM dapat belajar untuk mengetahui bagian masukan mana yang penting dengan bantuan *input gate*, lalu menyimpannya ke dalam *long-term state* dan disimpan selama informasi tersebut dibutuhkan dengan bantuan *forget gate*. Hal inilah yang membuat sel ini bisa dengan baik menangkap pola yang ada pada data *time series*, teks, audio, dan lain - lain. Berikut merupakan persamaan yang ada pada sel LSTM :

$$f_{(t)} = \sigma(W_{xf}{}^T x_{(t)} + W_{hf}{}^T h_{(t-1)} + b_f) \tag{2.2}$$

Keterangan :

$f_{(t)}$ = *forget gate* pada waktu *t*

σ = fungsi aktivasi *sigmoid*

$W_{xf}{}^T$ = bobot *forget gate* untuk x_t

x_t = data masukan pada waktu *t*

$W_{hf}{}^T$ = bobot *forget gate* untuk h_{t-1}

h_{t-1} = *hidden state* sebelumnya

b_f = nilai *bias forget gate*

$$i_{(t)} = \sigma(W_{xi}{}^T x_{(t)} + W_{hi}{}^T h_{(t-1)} + b_i) \tag{2.3}$$

$$g_{(t)} = tanh(W_{xg}{}^T x_{(t)} + W_{hg}{}^T h_{(t-1)} + b_g) \tag{2.4}$$

$$c_{(t)} = f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)} \tag{2.5}$$

Keterangan :

$i_{(t)}$ = *input gate* pada waktu *t*

σ = fungsi aktivasi *sigmoid*

$W_{xi}{}^T$ = bobot *input gate* untuk x_t

x_t = data masukan pada waktu *t*

$W_{hi}{}^T$ = bobot *input gate* untuk h_{t-1}

h_{t-1} = *hidden state* sebelumnya

b_i = nilai *bias input gate*

$g_{(t)}$ = *memory gate* pada waktu *t*

$tanh$ = fungsi aktivasi *hyperbolic tangent*

$W_{xg}{}^T$ = bobot *memory gate* untuk x_t

$W_{hg}{}^T$ = bobot *memory gate* untuk h_{t-1}

b_g = nilai bias *memory gate*

c_t = memori sel pada waktu *t*

f_t = nilai aktivasi *forget gate*

c_{t-1} = memori sel sebelumnya

i_t = nilai aktivasi *input gate*

$$o_{(t)} = \sigma(W_{xo}{}^T x_{(t)} + W_{ho}{}^T h_{(t-1)} + b_o) \tag{2.6}$$

$$y_{(t)} = h_{(t)} = o_{(t)} \otimes tanh(c_{(t)}) \tag{2.7}$$

Keterangan :

$o_{(t)}$ = *output gate* pada waktu *t*

σ = fungsi aktivasi *sigmoid*

$W_{xo}{}^T$ = bobot *output gate* untuk x_t

x_t = data masukan pada waktu *t*

$W_{ho}{}^T$ = bobot *output gate* untuk h_{t-1}

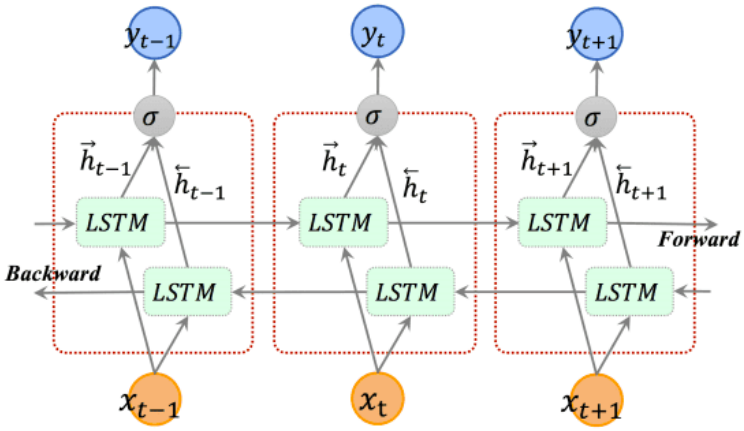
h_{t-1} = *hidden state* sebelumnya

b_o = nilai *bias output gate*

2.1.4 *Bidirectional Long Short Term Memory*

RNN dapat digunakan untuk memprediksi sebuah kejadian dengan menggunakan data masa lalu sebagai masukan. Tetapi tidak menutup kemungkinan bahwa keluaran yang dihasilkan oleh RNN memiliki hubungan dengan kejadian di masa depan. LSTM memiliki sifat yang sama dengan RNN yang hanya mempertimbangkan data dari masa lalu untuk memprediksi sebuah hal. Hal inilah yang menjadi kekurangan LSTM di mana LSTM tidak mempertimbangkan data masa depan. Untuk mengatasi hal ini digunakan pendekatan BiLSTM. BiLSTM dapat mempertimbangkan informasi yang ada pada masa lalu dan masa depan untuk memprediksi suatu kejadian [7].

BiLSTM berasal dari *bidirectional* RNN yang memproses data masukan dengan dua arah yaitu maju dan mundur. BiLSTM menelusuri data masukan dengan arah waktu maju dan mundur. Secara sederhana arsitektur BiLSTM melakukan duplikasi terhadap lapisan pertama yang ada pada jaringan. Dari duplikasi tersebut BiLSTM memiliki dua buah lapisan yang bersebelahan. Lapisan pertama menggunakan masukan dengan arah waktu positif sedangkan lapisan kedua menggunakan masukan dengan arah waktu terbalik [8].



Gambar 2.5 Struktur BiLSTM [9]

Dalam struktur BiLSTM *forward layer* $\vec{h}$ memproses data masukan dengan urutan waktu positif dari $T-n$ hingga $T-1$. *Backward layer* $\overleftarrow{h}$ memproses data masukan dengan urutan waktu terbalik dari $T-n$ hingga $T-1$ [9]. Keluaran pada waktu t (y_t) merupakan penggabungan dari keluaran *forward* dan *backward layer* dengan menggunakan persamaan 2.8.

$$y_t = \left[\vec{h}, \overleftarrow{h}\right] \tag{2.8}$$

Keterangan :

y_t = keluaran pada waktu t

$\overrightarrow{h}$ = keluaran lapisan maju

$\overleftarrow{h}$ = keluaran lapisan mundur

BiLSTM memiliki arsitektur yang berbeda dengan LSTM di mana dalam BiLSTM memiliki dua buah lapisan LSTM digunakan untuk memproses data masukan. Pertama data di masukan ke dalam lapisan LSTM pertama atau *forward layer* dengan arah waktu positif dan kedua di masukan ke dalam lapisan LSTM kedua atau *backward layer* dengan arah waktu negatif. Hal ini dilakukan agar model BiLSTM dapat belajar secara dua arah. Dengan dilakukan pembelajaran secara dua arah diharapkan model BiLSTM dapat memberikan hasil prediksi yang lebih baik dibandingkan LSTM. Hal yang membedakan BiLSTM dengan LSTM yaitu dalam arsitektur LSTM, LSTM hanya belajar secara satu arah dengan arah waktu positif. Dengan pembelajaran satu arah, LSTM tidak dapat menggunakan informasi yang ada di masa depan untuk memprediksi kejadian saat ini. Oleh sebab itu BiLSTM memiliki kelebihan dari LSTM di mana BiLSTM dapat belajar secara dua arah [10].

Algoritme BiLSTM dapat digunakan untuk memprediksi data deret waktu. Penelitian [10] membandingkan performa LSTM dan BiLSTM dalam memprediksi data deret waktu. Penelitian [10] menyatakan bahwa pembelajaran secara dua arah memiliki dampak positif untuk memprediksi data deret waktu. Dengan digunakan nya pembelajaran secara dua arah dapat meningkatkan akurasi sebesar 37.7% [10].

2.1.5 Fungsi Aktivasi

Beberapa fungsi aktivasi yang dipakai dalam penelitian sebagai berikut :

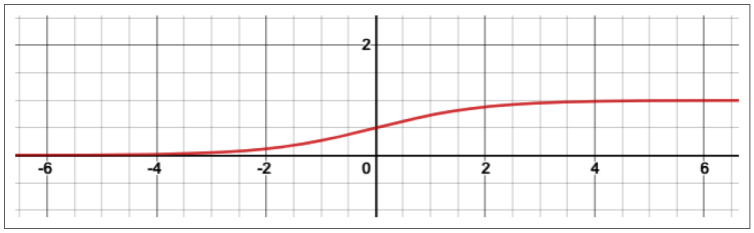
1. Fungsi aktivasi *sigmoid* merupakan fungsi aktivasi yang memiliki keluaran dengan skala 0 sampai 1 dan memiliki persamaan sebagai berikut :

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.9}$$

Keterangan :

x = masukan bilangan *real*

e = bilangan euler = 2.71828

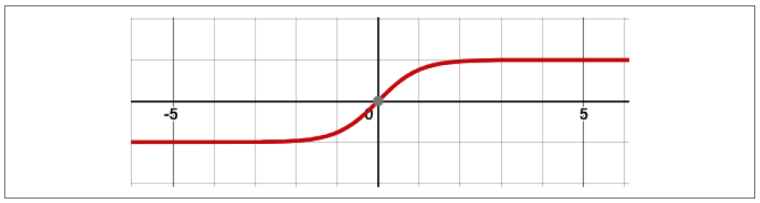


Gambar 2.6 Fungsi *sigmoid* [7]

2. Fungsi aktivasi *hyperbolic tangent* atau tanh merupakan fungsi aktivasi yang memiliki keluaran dengan skala -1 sampai 1 dan memiliki persamaan sebagai berikut :

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.10}$$

Keterangan :
x = masukan bilangan *real*
e = bilangan euler = 2.71828

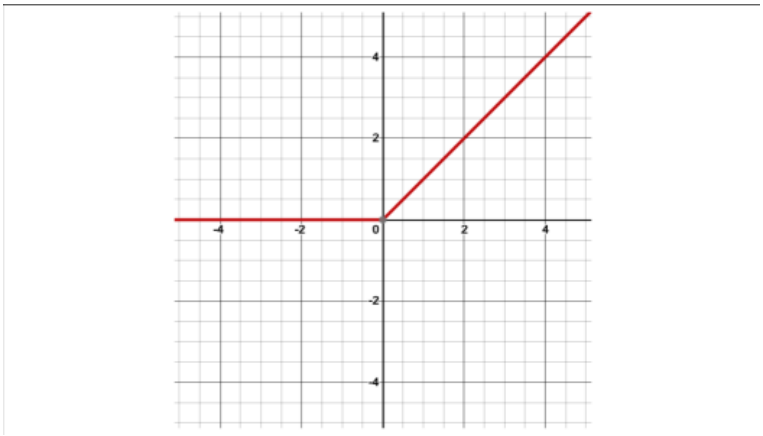


Gambar 2.7 Fungsi aktivasi *tanh* [7]

3. Fungsi aktivasi *rectified linear unit* atau ReLU memiliki keluaran dengan skala 0 sampai tak hingga dan memiliki persamaan sebagai berikut :

$$f(x) = max(0, x) \tag{2.11}$$

Keterangan :
x = masukan bilangan *real*
max = nilai maksimum



Gambar 2.8 Fungsi aktivasi ReLU [7]

2.1.6 *Min-Max Normalization*

Min-max normalization melakukan transformasi linear terhadap data asli. *Min-max normalization* menghasilkan keluaran dengan skala 0 sampai dengan 1. Dengan dilakukannya *Min-max normalization* standar deviasi mengecil dan membuat pengaruh dari *outliers* berkurang [11]. *Min-max normalization* memiliki persamaan sebagai berikut :

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.12}$$

Keterangan :

x_{scaled} = nilai baru

x = nilai data asli

x_{min} = nilai terkecil dari data asli

x_{max} = nilai terbesar dari data asli

2.1.7 *Root Mean Square Error*

Root mean square error (RMSE) merupakan sebuah alat ukur yang digunakan dalam penelitian. Nilai RMSE dapat memberi tahu seberapa besar *error* yang dihasilkan sistem ketika membuat sebuah prediksi [6]. Berikut merupakan persamaan yang digunakan untuk menghitung nilai RMSE :

$$RMSE = (X,h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2} \tag{2.13}$$

Keterangan :

X = matriks yang berisi seluruh fitur

h = prediksi yang dihasilkan sistem

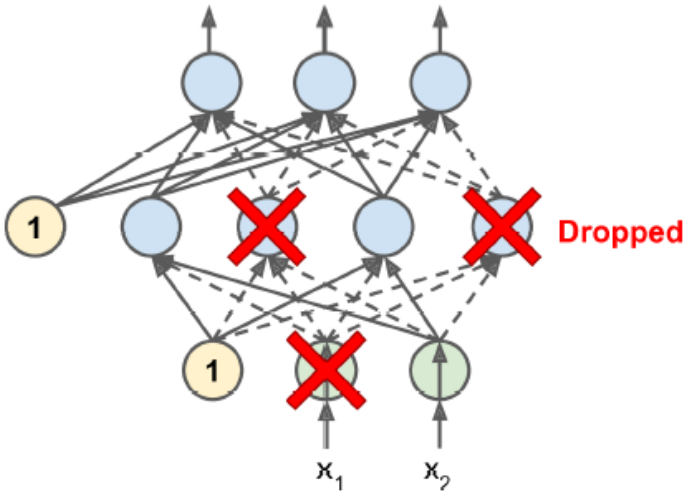
m = jumlah instansi pada *dataset*

$x^{(i)}$ = matriks yang berisi seluruh fitur kecuali *label* pada urutan ke i

$y^{(i)}$ = label pada urutan ke i

2.1.8 *Dropout*

Dropout merupakan sebuah teknik regularisasi yang populer dalam *neural network*. *Dropout* memiliki algoritme yang sederhana di mana setiap proses pelatihan, setiap neuron memiliki probabilitas untuk dibuang. Dengan regulasi *dropout* maka dalam setiap iterasi proses pelatihan ada *neuron* yang dipilih secara acak dan dibuang [6].



Gambar 2.9 *Dropout* [6]

2.1.9 *Dense Layer*

Dense layer atau yang biasa disebut *fully connected layer* merupakan sebuah lapisan yang menghubungkan seluruh *neuron* yang ada dalam sebuah lapisan dengan lapisan lainnnya [6]. *Dense layer* dapat digunakan untuk menggabungkan neuron dan dapat memberikan keluaran pada arsitektur LSTM [8]. Berikut merupakan persamaan untuk *dense layer* :

$$y = f(Wx + b) \tag{2.14}$$

Keterangan :

f = fungsi aktivasi

W = matriks bobot

x = matriks masukan

b = bias

2.1.10 *Library* yang digunakan

Pada bagian ini dijelaskan *library* yang digunakan dalam penelitian. *Library* digunakan untuk mempermudah proses implementasi sistem.

2.1.10.1 *Library* Pandas

Pandas merupakan *library* yang digunakan untuk *preprocessing* data. *Library* Pandas membantu dalam proses pengolahan data agar lebih mudah.

Tabel 2.1 Daftar metode yang digunakan dalam *library* pandas

No.	Metode	Masukan	Luaran	Keterangan
1.	read_csv	file path: string	DataFrame	Membaca data dalam bentuk .csv dan memberikan luaran dalam bentuk DataFrame
2.	drop	label: string, array, axis: int, inplace: boolean	DataFrame	Menghapus baris atau kolom dari DataFrame
3.	isnull	-	DataFrame	Memberi tahu apakah ada *missing value* dalam DataFrame
4.	max	axis: int, skipna: boolean, level:int, numeric_only: boolean	int	Menghitung jumlah data maksimal dalam sebuah kolom pada DataFrame

5.	min	axis: int, skipna: boolean, level:int, numeric_only: boolean	int	Menghitung jumlah data minimal dalam sebuah kolom pada DataFrame
6.	describe	-	Series atau DataFrame	Memberi informasi statistik dari data

2.1.10.2 *Library* Matplotlib

Matplotlib merupakan *library* yang digunakan untuk visualisasi data. *Library* Matplotlib mempermudah untuk visualisasi hasil dari model yang di implementasi.

Tabel 2.2 Daftar metode yang digunakan dalam *library* matplotlib

No.	Metode	Masukan	Luaran	Keterangan
1.	figure	-	Figure	Membuat sebuah figur
2.	plot	x: array atau scalar y: array atau scalar	list	Membuat plot data dengan sumbu x dan y
3.	xlabel	-	xlabel: str	Menentukan label untuk sumbu x
4.	ylabel	-	ylabel: str	Menentukan label untuk sumbu y
5.	title	title: str	-	Memberi judul untuk plot data
6.	legend	-	loc: str	Memberi legenda pada plot data
7.	show	-	Figure	Menampilkan *figure* yang telah dibuat sebelumnya

2.1.10.3 *Library* Keras

Keras merupakan *library* yang digunakan untuk pengembangan sistem berbasis *deep learning*. *Library* Keras digunakan untuk proses pelatihan,

pengujian model.

Tabel 2.3 Daftar metode yang digunakan dalam *library* keras

No.	Metode	Masukan	Luaran	Keterangan
1.	Sequential	keras.layer	-	Menggabungkan beberapa lapisan secara berurutan menjadi sebuah model
2.	LSTM	units: int	N-D Tensor	Lapisan LSTM untuk ditambahkan ke dalam sebuah model
3.	Bidirectional	layer: keras.layer, merge_mode: string	-	Wrapper untuk bidirectional RNN
3.	Dropout	rate: int	N-D Tensor	Lapisan dropout untuk ditambahkan ke dalam model
4.	model.add	keras.layer	-	Menambahkan lapisan ke dalam model
5.	model.compile	loss: string	-	Konfigurasi model untuk pelatihan
6.	model.fit	x: input data, y: target data, epoch:int, batch_size: int		Melakukan pelatihan terhadap model sesuai dengan konfigurasi yang telah ditentukan

2.1.10.4 *Long Short Term Memory* dalam *library* Keras

Dalam penelitian *library* Keras digunakan untuk mengimplementasi LSTM ke dalam sistem. LSTM dalam *library* Keras memiliki banyak parameter. Berikut merupakan penjelasan parameter yang ada:

1. units: Bilangan integer positif, dimensi keluaran. Unit digunakan untuk menentukan seberapa banyak sel LSTM yang ada dalam sebuah lapisan [12].
2. activation: Fungsi aktivasi yang digunakan. Secara *default* fungsi aktivasi yang digunakan yaitu *tanh*. Fungsi aktivasi ini digunakan pada *memory gate* dan *hidden state* [12].
3. recurrent_activation: Fungsi aktivasi yang digunakan untuk *recurrent step*. Secara default fungsi aktivasi yang digunakan yaitu *sigmoid*. Fungsi aktivasi ini digunakan pada *input gate* dan *output gate* [12].
4. use_bias: Boolean, secara *default* memiliki nilai *true*. Parameter ini digunakan apakah LSTM menggunakan bias atau tidak [12].
5. kernel_initializer: Digunakan untuk inisialisasi matriks bobot, memiliki nilai *default* glorot_uniform. Nilai bobot terdistribusi antara -*limit* hingga *limit*. Nilai *limit* dihitung menggunakan persamaan *limit* = $sqrt(6/(fan_in + fan_out)$. fan_in merupakan jumlah unit masukan sedangkan fan_out merupakan jumlah unit keluaran. [12].
6. recurrent_initializer: Inisialisasi untuk bobot matriks *recurrent state*, memiliki nilai *default* orthogonal. Matriks bobot ini digunakan pada *input gate*, *output gate*, *memory gate* dan *forget gate* [12].
7. bias_initializer: Inisialisasi nilai vektor bias, secara default memiliki nilai *zeros*. *Zeros* menandakan bahwa nilai bias yang digunakan adalah nol [12].
8. kernel_regularizer: Digunakan untuk regularisasi kernel bobot, secara default memiliki nilai *None*. Parameter ini dapat diisi dengan *weight regularizers* seperti L1, L2, L1L2, dan OrthogonalRegularizer [12].
9. recurrent_regularizer: Digunakan untuk regularisasi pada *reccurent kernel*, secara *default* memiliki nilai *None*. Parameter ini dapat diisi dengan *weight regularizers* seperti L1, L2, L1L2, dan OrthogonalRegularizer [12].
10. bias_regularizer: Digunakan untuk regularisasi pada kernel bias, secara default memiliki nilai *None*. Parameter ini dapat diisi dengan *weight regularizers* seperti L1, L2, L1L2, dan OrthogonalRegularizer [12].
11. activity_regularizer: Merupakan fungsi regularisasi yang digunakan pada hasil keluaran LSTM, secara default memiliki nilai *None* [12].
12. kernel_constraint: Digunakan untuk menerapkan *constraint function* pada

kernel bobot, secara *default* memiliki nilai *None*. Parameter ini digunakan untuk membatasi nilai bobot sesuai dengan fungsi yang digunakan [12].

13. recurrent_constraint: Digunakan untuk menerapkan *constraint function* pada *recurrent kernel*, secara *default* memiliki nilai *None*. Parameter ini digunakan untuk membatasi nilai bobot pada *recurrent kernel* sesuai dengan fungsi yang digunakan [12].

14. bias_constraint: Digunakan untuk menerapkan *constraint function* pada bias, secara *default* memiliki nilai *None*. Parameter ini digunakan untuk membatasi nilai bobot pada bias sesuai dengan fungsi yang digunakan [12].

15. dropout: Berisi *float* antara 0 hingga 1, secara *default* memiliki niali 0. Digunakan untuk membuang unit sesuai dengan bilangan masukan [12].

16. recurrent_dropout: Berisi *float* antara 0 hingga 1, secara *default* memiliki niali 0. Digunakan untuk membuang unit sesuai dengan bilangan masukan pada *recurrent state* [12].

17. return_sequences: Boolean, secara *default* memiliki nilai False. Parameter ini digunakan untuk mengembalikan seluruh *sequence* yang ada atau *sequence* yang ada pada *output sequence* [12].

18. go_backwards: Boolean, secara *default* memiliki nilai False. Jika memiliki nilai True maka LSTM akan memproses masukan dengan arah terbalik [12].

19. stateful: Boolean, secara *default* memiliki nilai False. Jika memiliki nilai True maka *state* terakhir dalam setiap sampel pada indeks i dalam sebuah *batch* akan digunakan sebagai *initial state* sebagai sampel untuk indeks i pada *batch* berikutnya [12].

20. time_major: Boolean, secara *default* memiliki nilai False. Parameter ini digunakan untuk menentukan bentuk tensor masukan dan keluaran. Jika bernilai True masukan dan keluaran akan ada dalam bentuk *timestep, batch, feature*, sedangkan jika False maka *batch, timesteps, feature* [12].

21. unroll: Boolean, secara *default* memiliki nilai False. Jika memiliki nilai True maka jaringan akan di *unroll*. Dengan dilakukannya *unroll* dapat mempercepat proses perhitungan, tetapi menggunakan memori yang lebih banyak. *Unroll* hanya cocok untuk sebuah *sequence* yang pendek [12].

2.1.10.5 *Bidirectional* dalam *Library* Keras

Dalam penelitian, *library* Keras digunakan untuk implementasi BiLSTM. Berikut merupakan penjelasan parameter yang ada:

1. layer: Berisi sebuah lapisan RNN seperti LSTM dan *gated recurrent unit* atau sebuah *sequence processing layer* [12].
2. merge_mode: Merupakan sebuah mode dimana keluaran dari *forward* dan *backward* RNN digabungkan, secara default memiliki nilai concat. Nilai merge_mode dapat diisi dengan sum, mul, concat, ave, dan None. Jika memiliki nilai None maka keluaran tidak akan digabungkan melainkan di kembalikan sebagai *list* [12].
3. backward_layer: Opsional, dapat berisi sebuah lapisan RNN untuk menghitung *bacward input processing*. Jika backward_layer tidak diisi maka lapisan yang ada pada argumen layer digunakan secara otomatis [12].

2.2 Tinjauan Studi

Pada bagian ini dijelaskan mengenai perbandingan penelitian sebelumnya. Penelitian yang digunakan merupakan penelitian yang terkait dengan prediksi harga *bitcoin*. Pada Tabel 2.4 diberikan penjelasan mengenai studi terkait dalam penelitian. Pemilihan metode LSTM dan BiLSTM didasarkan dari penelitian sebelumnya di mana kedua metode tersebut dapat memprediksi data deret waktu dengan nilai RMSE yang rendah.

Tabel 2.4 Tinjauan Studi

No.	Judul	Rumusan Masalah	Metode	Hasil
1.	Jiang X. Bitcoin price prediction based on deep learning methods [13]	Membandingkan metode MLP, LSTM, dan GRU untuk memprediksi harga *bitcoin*.	MLP, LSTM, GRU	Hasil penelitian terbaik yaitu metode LSTM dengan 2 *hidden layer* dapat memprediksi harga *bitcoin* dengan RMSE 125.387

2.	Sunny, M.A.I., Maswood, M.M.S. and Alharbi, A.G. Deep learning-based stock price prediction using LSTM and bi-directional LSTM model [14]	Membandingkan metode LSTM dan BiLSTM untuk memprediksi harga saham.	LSTM, BiLSTM	Hasil dari penelitian ini yaitu BiLSTM memiliki hasil terbaik dengan 2 *dense layer* dan 100 *epoch* dengan RMSE sebesar 0.0002421.
3.	Jia, M., Huang, J., Pang, L. and Zhao, Q. Analysis and Research on Stock Price of LSTM and Bidirectional LSTM Neural Network [15]	Membandingkan metode LSTM dan BiLSTM untuk memprediksi harga saham.	LSTM, BiLSTM	Hasil penelitian ini yaitu metode BiLSTM dapat memprediksi harga saham terbaik dengan RMSE sebesar 1.3382.
4.	S. McNally, J. Roche and S. Caton. Predicting the Price of Bitcoin Using Machine Learning [16]	Membandingkan metode ARIMA, RNN, dan LSTM untuk memprediksi harga *bitcoin*.	ARIMA, RNN, LSTM	Hasil dari penelitian ini yaitu Metode LSTM dapat memprediksi harga *bitcoin* terbaik dengan RMSE sebesar 6.58%.

5.	K. A. Althelaya, E. M. El-Alfy and S. Mohammed, Evaluation of bidirectional LSTM for short-and long-term stock market prediction [17]	Membandingkan metode LSTM dan BiLSTM untuk memprediksi harga saham.	LSTM, BiLSTM	BiLSTM dapat memprediksi harga saham jangka pendek dan panjang terbaik dengan RMSE 0.028 dan RMSE 0.0746.

Untuk memprediksi harga *bitcoin* dapat dilakukan dengan menggunakan beberapa metode. Pada penelitian [13] menerapkan metode MLP, LSTM dan GRU untuk memprediksi harga *bitcoin*. Penelitian tersebut menggunakan data 24 jam sebagai masukan dan memprediksi harga *bitcoin* pada jam berikutnya. Model LSTM menghasilkan RMSE terendah yaitu 125.387. Pada penelitian [16] menerapkan metode RNN, ARIMA,LSTM dan regularisasi *dropout* untuk memprediksi harga *bitcoin*. Penelitian [16] menggunakan data harian *bitcoin* untuk memprediksi harga penutupan di hari berikutnya. Dengan konfigurasi model 20 unit LSTM dapat memprediksi harga *bitcoin* dengan RMSE 6.58%. Selain metode LSTM terdapat juga metode terkait yang dapat digunakan untuk memprediksi harga *bitcoin* yaitu BiLSTM. BiLSTM memiliki perbedaan dengan LSTM, di mana BiLSTM dapat menggunakan informasi yang ada di masa depan dan masa lalu untuk memprediksi sebuah kejadian. Penelitian [14,15,17] menerapkan metode BiLSTM untuk memprediksi harga saham. Metode BiLSTM dapat memprediksi harga saham dengan nilai RMSE yang rendah. Oleh karena itu dalam penelitian dibandingkan metode LSTM dan BiLSTM untuk memprediksi harga *bitcoin*. Berdasarkan penelitian [15,16] regularisasi *dropout* digunakan untuk mencegah terjadinya *overfitting*.

2.3 Tinjauan Objek

Pada bagian ini akan dijelaskan objek yang terkait dengan prediksi harga *bitcoin*. Objek yang dibahas meliputi *cryptocurrency* dan *bitcoin*.

2.3.1 *Cryptocurreny*

Cryptocurrency merupakan sebuah jaringan *peer-to-peer* di mana teknik *cryptography* digunakan untuk membuat dan mendistribusikan mata uang kripto. Proses ini membutuhkan sebuah sistem untuk memverifikasi transaksi tanpa otoritas pusat. Proses verifikasi transaksi mengonfirmasi jumlah transaksi dan apakah sang pembayar memiliki mata uang yang ingin di bayarkan dan memastikan mata uang yang di bayarkan tidak terjadi dua kali transaksi. Proses ini disebut juga sebagai *mining* [18].

Tidak seperti mata uang digital lainnya yang dikeluarkan secara terpusat atau memiliki sirkulasi keuangan pada komunitas atau daerah tertentu atau memiliki keterkaitan dengan mata uang *fiat*, *cryptocurrency* memiliki karakteristik yang berbeda. Teknologi *blockchain* yang digunakan oleh *crptocurrency* salah satunya yaitu *bitcoin*, merupakan sebuah *ledger* yang tersebar secara terbuka yang menyimpan data transaksi. Hal ini memecahkan masalah *double-spending* dan tidak membutuhkan pihak ketiga. Sistem desentralisasi membuat teknologi *blockchain* untuk memiliki kapasitas yang tinggi, keamanan yang lebih baik, dan penyelesaian yang lebih cepat. [19]

2.3.1.1 Blockchain

Blockchain merupakan sebuah buku besar yang di distribusikan secara publik untuk transaksi *cryptocurrency*. Setiap transaksi yang telah di verifikasi disimpan dalam sebuah blok. Setiap blok berisi sejumlah transaksi yang telah di verifikasi. Ukuran blok pada sistem *cryptocurrency* telah ditetapkan sebelumnya, di mana setiap blok memiliki kapasitas maksimum untuk menyimpan data transaksi. Sebuah blok berisi *magic number* yang telah ditetapkan sebelumnya, *block size*, *block header* yang menyimpan *hash* dari blok sebelumnya, *transaction counter*, dan *transactions*. Blok pertama atau yang disebut juga sebagai *genesis block* menyimpan transaksi pertama untuk *cryptocurrency*. *Hash* dari blok pertama tersebut diteruskan kepada *miner*, yang selanjutnya digunakan untuk membuat *number used only once* atau yang biasa disebut juga sebagai *nonce*. *Nonce* digunakan untuk membuat *hash* untuk blok kedua. Begitu juga untuk *hash* blok ketiga berisi *hash* dari blok pertama dan kedua dan seterusnya. Sebuah hubungan yang tersusun secara kronologis dan ketat terbuat dari *genesis block* hingga blok sekarang dengan penyertaan *hash*. Dari blok pertama hingga blok sekarang terdapat sebuah jalur yang unik. Hubungan inilah yang membuat penyerang sangat sulit untuk menyerang atau merusak informasi yang ada dalam sebuah blok [18].

2.3.1.2 Mining

Sebuah transaksi terjadi ketika seorang pembayar mengirimkan sejumlah *cryptocurrency* kepada penerima pembayaran [18]. Transaksi tersebut disimpan dalam *blockchain* dan di verifikasi oleh *network nodes*, di mana *network nodes* dapat berupa sebuah individu yang menggunakan sistem komputer untuk memverifikasi transaksi tersebut. Setelah transaksi tersebut berhasil di verifikasi transaksi tersebut akan disimpan dalam *blockchain* dan transaksi tersebut telah selesai dilakukan. Proses verifikasi dan menyimpan transaksi ke dalam *blockchain* disebut juga sebagai *mining* [19].

Ketika sebuah transaksi baru terjadi, individu yang melakukan *mining* atau yang biasa disebut juga *miner* akan melakukan pengecekan apakah *cryptocurrency* yang digunakan merupakan milik pembayar atau pembayar sedang mencoba untuk melakukan *double spend*. Kepemilikan sebuah *cryptocurrency* ada dalam jaringan *blockchain*. Sebuah pengguna yang jahat dapat membuat *nodes* yang banyak dan mencoba untuk memvalidasi sebuah transaksi yang invalid. Untuk mengatasi hal ini *miners* dituntut untuk memecahkan masalah yang membutuhkan sumber daya yang besar. Sebuah masalah yang dapat dikerjakan oleh *miner* yaitu *proof of work* di mana sebuah hasil yang mudah di verifikasi dari pekerjaan yang telah dilakukan bahwa pekerjaan tersebut telah selesai dilakukan, *proof of stake* di mana *miner* harus menunjukkan seberapa banyak *currency* yang dimiliki olehnya, *proof of retrievability* di mana *miner* harus menunjukan data yang telah disimpan utuh dan dapat diambil kembali. Setelah masalah tersebut diselesaikan oleh *miner* maka *miner* akan mendapatkan imbalan berupa *cryptocurrency* [18].

Setiap blok yang di *mined* memiliki sebuah batasan untuk transaksi yang dapat di validasi. Batasan ini dibutuhkan karena setiap blok yang di *mined* menghasilkan unit *cryptocurrency* dan unit dari *cryptocurrency* tersebut memiliki jumlah terbatas. Maka dari itu proses produksi unit baru harus dibatasi untuk mencegah kehabisan persediaan. Sebagai contoh, untuk kasus *bitcoin*, setiap blok memperkenalkan 50 *bitcoin* baru ke dalam sistem. Jumlah *bitcoin* yang diperkenalkan dikurangi setengah setiap 210000 blok. Maka dari itu jumlah *bitcoin* maksimum yang ada hanya 21 juta *bitcoin*. Jika jumlah *bitcoin* yang dapat di *mined* per hari tidak dibatasi maka persediaan *bitcoin* akan cepat habis [18].

2.3.2 Bitcoin

Bitcoin merupakan sebuah protokol jaringan komunikasi *online* yang memfasilitasi mata uang *virtual* dan juga proses transaksi. Sejak munculnya

bitcoin pada tahun 2008 oleh sekelompok *developer*, *bitcoin* telah melayani setidaknya 62.5 juta transaksi dengan 109 juta akun yang terlibat. Per Maret 2015 volume transaksi harian *bitcoin* mencapai kurang lebih 200.000 *bitcoin* dengan total kurang lebih 50 juta *United State Dollar* (USD) dan total nilai pasar *bitcoin* mencapai 3.5 miliar USD. Peraturan *bitcoin* yang dibuat oleh *developer* tidak memiliki pengaruh terhadap pemerintah maupun regulator. Dibanding memilih untuk menyimpan data transaksi dalam sebuah *server* atau *data center*, *bitcoin* memilih untuk menyimpan data transaksi secara terdistribusi dalam sebuah jaringan. Sistem yang dirancang oleh *bitcoin* membuat transaksi menjadi tidak dapat dikembalikan, pembuatan uang dalam waktu ke waktu yang telah ditentukan, dan riwayat transaksi yang bersifat publik [20].

Bitcoin dapat digunakan sebagai alat pembayaran untuk barang maupun jasa atau dapat dibeli dari orang lain dengan sistem pertukaran. *Bitcoin* dapat di transaksi melalui *software*, aplikasi, atau berbagai macam platform *online* yang menyediakan dompet digital. Setiap transaksi memiliki masukan dan keluaran. Masukan memiliki referensi kepada keluaran dari transaksi sebelumnya, dan keluaran dari transaksi menyimpan alamat penerima dan jumlah transaksi. Secara general sejumlah *bitcoin* akan dikirimkan dari dompet *bitcoin* ke alamat yang spesifik, jika memiliki jumlah yang sesuai pada dompet di transaksi sebelumnya. Transaksi yang dilakukan tidak terenkripsi dan dapat dilihat di *blockchain* dengan alamat yang spesifik, tetapi identitas dari pengirim dan penerima dibuat anonim. Secara khusus, dompet *bitcoin* memiliki *private key* atau *seed* untuk menandatangani transaksi. *Private key* atau *seed* merupakan data yang memberikan bukti secara matematis bahwa koin yang ada pada transaksi datang dari pemilik dompet. Dengan *private key* dan tanda tangan, akun hanya dapat di akses oleh pemiliknya dan transaksi tidak dapat diganggu oleh orang lain [19].

Bitcoin memiliki fitur sebagai berikut, yang pertama yaitu desentralisasi. Sama dengan mata uang konvensional yang diperdagangkan secara digital, *bitcoin* dapat digunakan untuk membeli barang secara elektronik. Tidak seperti uang kertas, *bitcoin* memiliki sifat ter desentralisasi. Dengan kata lain tidak ada kelompok atau institusi mana pun yang dapat mengontrol jaringan *bitcoin*. Persediaan *bitcoin* diatur oleh sebuah algorime dan siapa pun dapat mengakses melalui internet. Kedua *bitcoin* memiliki sifat fleksibel. Dompet atau alamat *bitcoin* dapat dengan mudah di atur secara *online* tanpa biaya atau regulasi. Terlebih lagi transaksi yang dilakukan tidak memerlukan lokasi yang spesifik, dengan kata lain *bitcoin* dapat di transfer antar negara tanpa kendala. Ketiga

bitcoin memiliki sifat transparan. Setiap transaksi yang terjadi akan di masukan ke seluruh jaringan. *Mining nodes* atau *miner* akan memvalidasi, menyimpan transaksi tersebut dalam *blockchain*, dan menyiarkan blok yang telah selesai ke *node* lainnya. Seluruh rekaman transaksi disimpan dalam *blockchain*, terbuka dan terdistribusi, jadi seluruh *miner* memiliki salinan dan dapat memverifikasi. Transaksi *bitcoin* dilakukan dengan cepat. Transaksi terjadi dalam hitungan detik dan membutuhkan kurang lebih 10 menit untuk *miner* dapat memverifikasi nya. Terakhir *bitcoin* memiliki biaya transfer yang rendah. Untuk melakukan transaksi tidak dipungut biaya tetapi pemilik dapat membayar ekstra untuk memfasilitasi transaksi yang lebih cepat [19].

BAB 3 ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Masalah

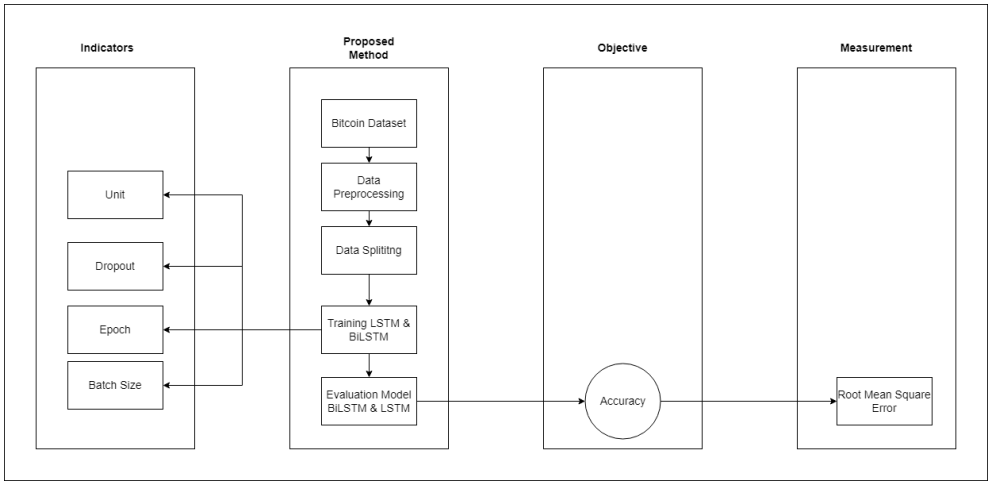
Seiring bertambah banyak masyarakat yang ingin melakukan investasi terhadap *bitcoin*, diperlukan sebuah sistem yang untuk memprediksi harga *bitcoin*. Data deret waktu pada umumnya memiliki pola tertentu yang dipelajari dan diprediksi untuk masa mendatang. Maka dari itu peneliti berasumsi bahwa metode *deep learning* LSTM dan BiLSTM mampu memprediksi harga *bitcoin* menggunakan data deret waktu *bitcoin*. Penelitian membangun sebuah sistem prediksi harga *bitcoin* yang menerima masukan data harga *bitcoin* dalam kurun waktu tertentu. Pendekatan yang digunakan dalam penelitian yaitu pendekatan *deep learning* dengan metode LSTM dan BiLSTM. Keluaran yang dihasilkan dari penelitian adalah prediksi harga penutupan *bitcoin*.

Tahap awal dalam penelitian yaitu melakukan *preprocessing* terhadap data yang digunakan sebagai data latih dan uji. Dengan dilakukannya *preprocessing* data disesuaikan sehingga hanya memiliki bagian penting saja yang diperlukan oleh sistem. *Preprocessing* yang dilakukan yaitu normalisasi data dan membuang kolom yang tidak diperlukan. Normalisasi data yang digunakan yaitu *min - max normalization* di mana nilai dari sebuah data dirubah dengan rentang 0 sampai 1. Tahap *preprocessing* menghasilkan data latih dan uji. Harga penutupan *bitcoin* tidak hanya dipengaruhi oleh harga penutupan sebelumnya, bisa saja faktor lainnya seperti harga pembuka, harga tertinggi, harga terendah, dan volume memiliki pengaruh terhadap harga penutupan. Maka dari itu seluruh fitur digunakan dalam penelitian, tidak hanya fitur *close* saja.

Setelah tahap *preprocessing* selanjutnya adalah tahap pelatihan. Di tahap ini data latih merupakan masukan dalam pemodelan LSTM dan BiLSTM. Proses pelatihan dilakukan dengan nilai unit, *dropout*, *epoch*, dan *batch size* yang telah ditetapkan. Hasil dari proses pelatihan ini adalah model LSTM dan BiLSTM yang digunakan dalam tahap pengujian. Pada tahap pengujian data uji digunakan untuk menguji model yang telah dihasilkan pada tahap sebelumnya. Hasil dari tahap pengujian ini adalah nilai RMSE yang digunakan untuk mengevaluasi model.

3.2 Kerangka Pemikiran

Pada bagian ini dijelaskan kerangka pemikiran dari metode yang diusulkan untuk memprediksi harga *bitcoin*.



Gambar 3.1 Kerangka Pemikiran

Gambar 3.1 merupakan kerangka pemikiran yang telah disusun dalam penelitian :

1. Data *preprocessing* : Membuang fitur/kolom yang tidak terpakai, dan melakukan *min-max normalization* pada data.
2. Data *splitting* : Membagi data menjadi data latih dan data uji.
3. Training : Melatih model LSTM dan BiLSTM menggunakan data latih.
4. Evaluasi : Mengevaluasi model hasil dari pelatihan dengan RMSE.
5. *Indicators* : Merupakan *hyperparameter* yang diuji pada model LSTM dan BiLSTM.

Pada bagian ini dijelaskan *hyperparameter* yang digunakan pada LSTM dan BiLSTM. Berikut ini merupakan penjelasan dari indikator yang digunakan pada LSTM dan BiLSTM :

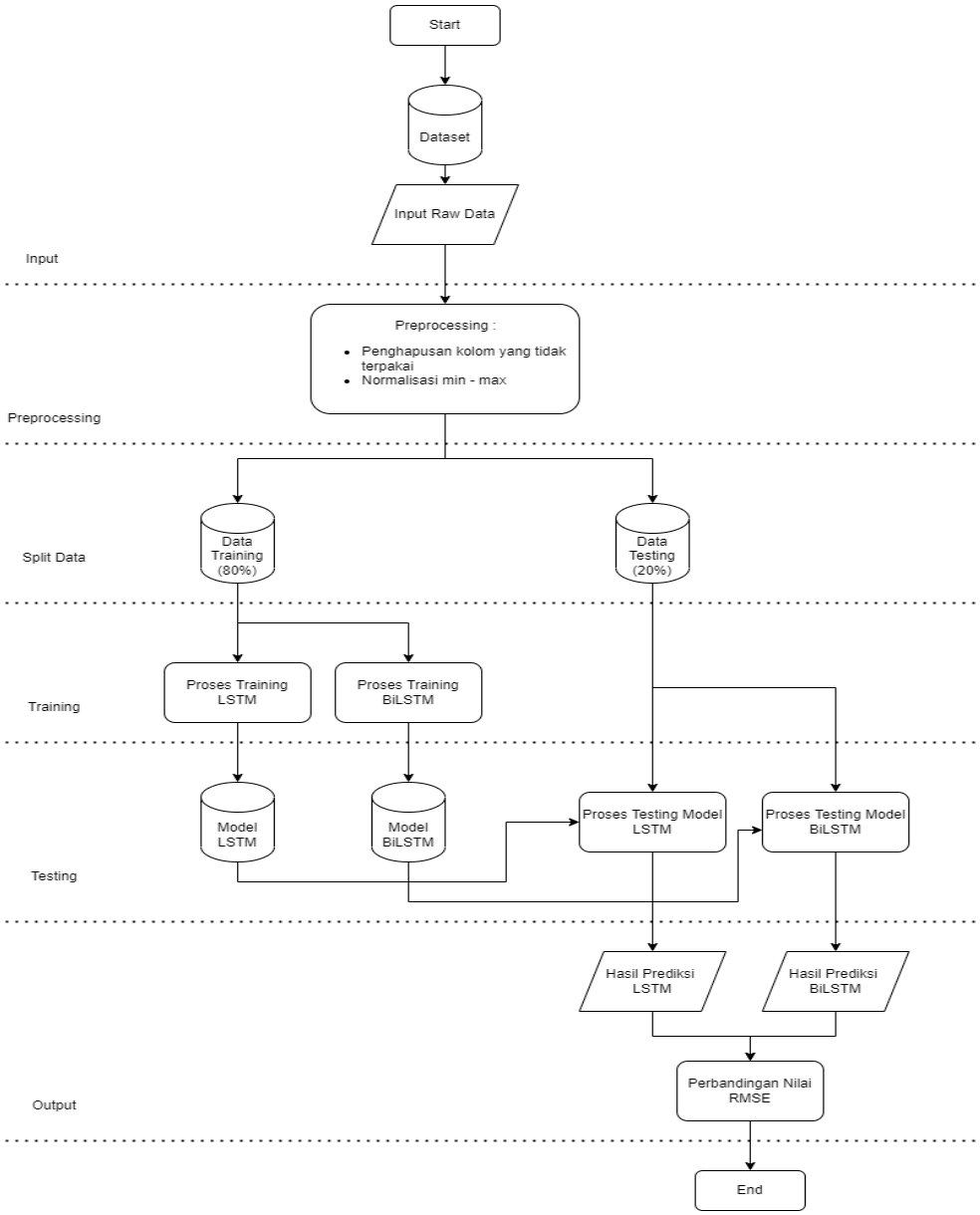
1. Unit : Merupakan jumlah unit yang ada pada *hidden layer*. Semakin besar nilai unit maka *hidden pattern* yang ada pada data semakin ditemukan, tetapi jika terlalu besar menyebabkan *overfitting*. Berdasarkan penelitian [14,17] nilai unit yang digunakan dalam penelitian yaitu 32, 64, 128.
2. *Dropout* : Merupakan penentuan *neuron* yang dibuang selama proses pelatihan secara acak. Berdasarkan penelitian [15] nilai *dropout* yang digunakan dalam penelitian yaitu 0.2, 0.5, dan tanpa *dropout*.
3. *Epoch* : Merupakan keadaan di mana seluruh *dataset* telah melalui proses pelatihan. Berdasarkan penelitian [14,15,16] nilai *epoch* yang digunakan

dalam penelitian yaitu 10, 50 dan 100.

4. *Batch Size* : Merupakan pembagian jumlah data sampel yang digunakan untuk pelatihan. Berdasarkan penelitian [16] nilai *batch size* yang digunakan dalam penelitian yaitu 20, 50, dan 100.

3.3 Urutan Proses Global

Pada penilitian ini metode LSTM digunakan untuk memprediksi harga *bitocin*. Berikut merupakan urutan proses global dari penelitian :



Gambar 3.2 Flowchart proses global

Berikut urutan proses global sesuai dengan *Flowchart* 3.2 :

1. Langkah pertama *raw data* melalui tahap *preprocessing* untuk mendapatkan data yang sudah siap digunakan sebagai masukan perhitungan LSTM dan BiLSTM. Tahap *preprocessing* menghasilkan data latih dan uji.
2. Selanjutnya data latih digunakan sebagai masukan dalam pelatihan LSTM dan BiLSTM. Hasil dari pelatihan ini adalah model LSTM dan BiLSTM yang diuji di tahap selanjutnya.
3. Kemudian data uji digunakan sebagai masukan untuk proses pengujian model LSTM dan BiLSTM. Hasil dari pengujian ini adalah nilai RMSE yang digunakan untuk evaluasi model.

3.4 Analisis Manual

Pada bagian ini dijelaskan analisis proses yang dilakukan dalam sistem. Perhitungan yang dilakukan dalam bagian ini dilakukan secara manual.

3.4.1 Dataset

Dataset yang digunakan dalam penelitian berasal dari *Bitcoin Historical Dataset*. *Dataset* diperoleh dari situs Kaggle [21] , yang berisi rekaman data harga *bitcoin* dari tahun 2018 hingga 2022. *Dataset* ini memiliki 33260 baris dan 9 buah fitur/kolom yaitu :

1. Unix: satuan waktu unix ketika data diambil
2. *Date*: satuan waktu UTC ketika data diambil
3. *Symbol*: simbol yang mengacu pada data
4. *Open*: harga pembukaan pada satuan waktu tertentu
5. *High*: harga tertinggi pada satuan waktu tertentu
6. *Low*: harga terendah pada satuan waktu tertentu
7. *Close*: harga penutupan pada satuan waktu tertentu
8. Volume (BTC): volume transaksi dalam BTC
9. Volume (USD): volume transaksi dalam USD

Tabel 3.1 Contoh data *Bitcoin*

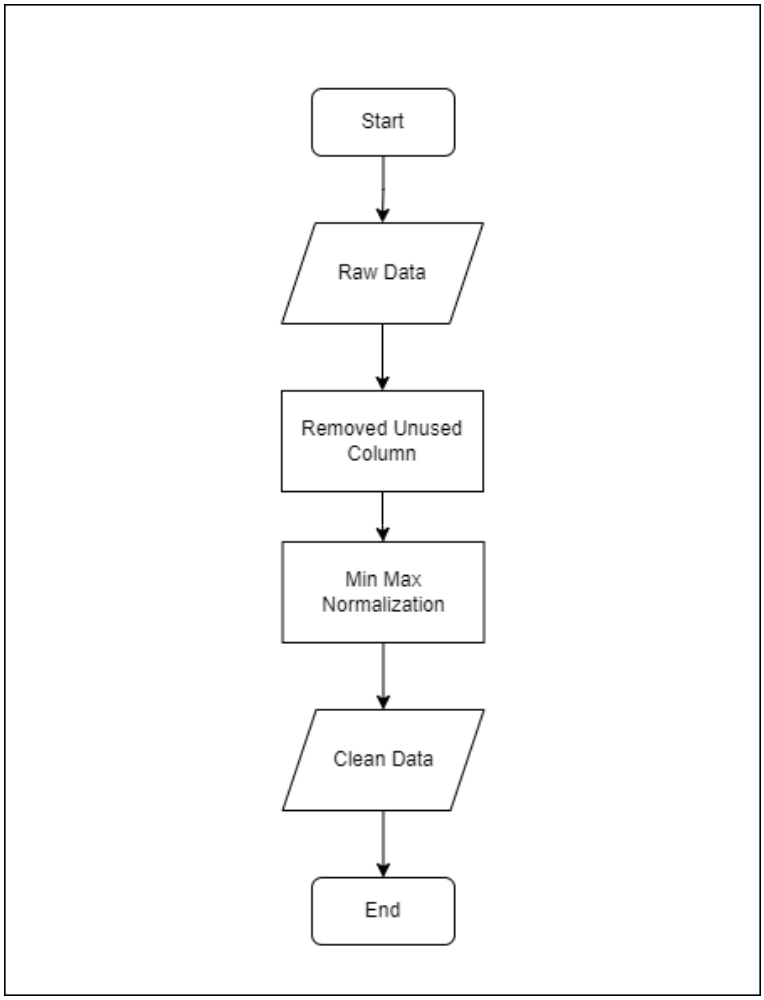
Unix	Date	Symbol	Open	High	Low	Close	Volume BTC	Volume USD
1526364000	15/5/2018 06:00:00	BTC/USD	8733	8796	8707	8740	4906603	559
1526367600	15/5/2018 07:00:00	BTC/USD	8740	8766	8721	8739	2390399	273
1526371200	15/5/2018 08:00:00	BTC/USD	8739	8750	8660	8728	7986063	917
1526374800	15/5/2018 09:00:00	BTC/USD	8728	8754	8701	8708	1593991	182
1526378400	15/5/2018 10:00:00	BTC/USD	8708	8865	8695	8795	1110127	1260

3.4.2 *Preprocessing* Data

Tahap *preprocessing* dilakukan agar data siap digunakan dalam pemodelan LSTM dan BiLSTM. Gambar 3.3 menunjukkan data diolah dengan dilakukan penghapusan kolom yang tidak terpakai dan *min - max normalization*. Kolom yang dihapus sebagai berikut :

1. Unix : Kolom Unix menunjukkan satuan waktu. Dalam *dataset* sudah ada kolom *date* maka dari itu kolom unix tidak diperlukan.

2. *Symbol* : kolom *symbol* hanya berisi satu nilai unik dan tidak memiliki keterkaitan dengan harga *bitcoin*, maka dari itu kolom *symbol* tidak dipakai.



Gambar 3.3 *Flowchart preprocessing* data

Setelah dilakukan penghapusan kolom *dataset* menjadi seperti berikut :

Tabel 3.2 Contoh data *Bitcoin* setelah penghapusan kolom

Date	Open	High	Low	Close	Volume BTC	Volume USD
15/5/2018 06:00:00	8733	8796	8707	8740	4906603	559
15/5/2018 07:00:00	8740	8766	8721	8739	2390399	273
15/5/2018 08:00:00	8739	8750	8660	8728	7986063	917
15/5/2018 09:00:00	8728	8754	8701	8708	1593991	182

15/5/2018 10:00:00	8708	8865	8695	8795	1110127	1260

Setelah menghapus kolom yang tidak terpakai, dilakukan *min - max normalization*. *Min - max normalization* merubah seluruh nilai dalam rentang 0 sampai 1. Normalisasi ini dilakukan terhadap seluruh data kecuali pada kolom *date*. *Min - max normalization* dilakukan dengan menggunakan persamaan 2.12. Berikut merupakan contoh perhitungan *min - max normalization*.

$x_i = 8733$
$x_{max} = 8740$
$x_{min} = 8708$
$x_{scaled} = \frac{8733-8708}{8740-8708}$
$x_{scaled} = 0.71$

Dapat dilihat pada perhitungan di atas bahwa untuk melakukan *min max normalization* dibutuhkan nilai maksimum dan minimum sebuah fitur. *Min max normalization* merubah nilai sebuah data dengan cara mengurangi dengan nilai minimum dan membaginya dengan nilai maksimum dikurangi nilai minimum. Setelah dilakukan *min - max normalization, dataset* menjadi seperti berikut :

Tabel 3.3 Contoh data *Bitcoin* setelah normalisasi

Date	**Open**	**High**	**Low**	**Close**	**Volume BTC**	**Volume USD**
15/5/2018 06:00:00	0.78	0.4	0.77	0.36	0.55	0.34
15/5/2018 07:00:00	1	0.13	1	0.35	0.34	0.08
15/5/2018 08:00:00	0.93	0	0	0.22	1	0.68
15/5/2018 09:00:00	0.62	0.03	0.67	0	0.07	0
15/5/2018 10:00:00	0	1	0.57	1	0	1

Setelah dilakukan *min max normalization* data dibagi menjadi data latih dan data uji. Data dibagi sebesar 80% untuk data latih dan sisanya sebagai data uji. Data latih digunakan untuk pemodelan sedangkan data uji digunakan untuk mengevaluasi model yang telah dibuat.

3.4.3 *Long Short Term Memory*

Perhitungan manual menggunakan data harga *bitcoin* untuk memprediksi harga penutupan *bitcoin* dengan menggunakan metode LSTM dan BiLSTM. Pada contoh perhitungan LSTM, dilakukan dengan dua *timestep* dan dengan dua unit LSTM. Masukan berupa data yang sudah di normalisasi, nilai bobot W_xf dan W_hf di inisialiasasi secara acak dan nilai bias b_f di inisialisasi nol. Untuk perhitungan *timestep* pertama dikarenakan *hidden state* sebelumnya belum ada maka h_{t-1} di inisialisasi nol.

3.4.3.1 *Timestep* Pertama

Pada *timestep* pertama dimulai dengan data pada tanggal 15-05-2018 06:00:00 yang memiliki vektor [0.0854122487, 0.0856348397, 0.0854959654, 0.0855316111, 0.07155548000, 0000021683]. Vektor tersebut menunjukkan atribut *open, high, low, close*, Volume BTC, Volume USD yang dijadikan sebagai masukan. Berikut merupakan perhitungan *forget gate* menggunakan persamaan 2.2

$$f_1 = \sigma(W_{xf}x_1 + W_{hf}h_0 + b_f)$$

$$f_1 = \sigma\left(\begin{bmatrix} 0.04190454 & 0.02481002 & -0.01951982 & 0.03418238 & 0.00209754 & 0.01316079 \\ -0.03049609 & -0.00972463 & -0.03368445 & -0.02692162 & 0.04277313 & -0.0051742 \end{bmatrix} \bullet \begin{bmatrix} 0.0854122487 \\ 0.0856348397 \\ 0.0854959654 \\ 0.0855316111 \\ 0.0715554800 \\ 0.0000021683 \end{bmatrix} + \begin{bmatrix} -0.025804 & -0.019003 \\ -0.026350 & -0.034445 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$$

Dari perhitungan di atas, maka matriks vektor *forget gate* sebagai berikut :

$$f_1 = \begin{bmatrix} 0.501777 \\ 0.49861 \end{bmatrix}$$

Setelah dilakukan perhitungan untuk *forget gate* maka selanjutnya dilakukan perhitungan untuk *input gate*. Nilai bobot W_{xi} dan W_{hi} di inisialisasi secara acak dan nilai bias b_i di inisialisasi nol. Perhitungan *forget gate* dilakukan menggunakan persamaan 2.3. Berikut merupakan perhitungan untuk *input gate*.

$$i_1 = \sigma(W_{xi}x_1 + W_{hi}h_0 + b_i)$$

$$i_1 = \sigma\left(\begin{bmatrix} -0.04663104 & 0.04068497 & 0.0404686 & -0.02446541 & -0.01712576 & 0.03562227 \\ 0.04126063 & -0.04613255 & -0.03730052 & 0.04292736 & -0.03635566 & 0.01664639 \end{bmatrix} \bullet \begin{bmatrix} 0.0854122487 \\ 0.0856348397 \\ 0.0854959654 \\ 0.0855316111 \\ 0.0715554800 \\ 0.0000021683 \end{bmatrix} + \begin{bmatrix} 0.012406 & -0.000243 \\ 0.035534 & -0.042836 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$$

Dari perhitungan di atas, maka matriks vektor *input gate* sebagai berikut :

$$i_1 = \begin{bmatrix} 0.500089 \\ 0.499364 \end{bmatrix}$$

Kemudian setelah perhitungan *input gate* selesai, dilakukan perhitungan untuk *memory gate* menggunakan persamaan 2.4. Nilai bobot W_{xg} dan W_{hg} di

inisialisasi secara acak dan nilai bias b_f di inisialisasi nol. Berikut merupakan perhitungan untuk *memory gate*.

$$g_1 = tanh(W_{xg}x_1 + W_{hg}h_0 + b_f)$$

$$g_1 = tanh\left(\begin{bmatrix} 0.04133164 & 0.03050084 & -0.00227099 & -0.044721 & 0.00982435 & 0.00515036 \\ 0.0137399 & -0.0018558 & -0.04297994 & 0.03585858 & -0.03557081 & -0.03730149 \end{bmatrix} \bullet \begin{bmatrix} 0.0854122487 \\ 0.0856348397 \\ 0.0854959654 \\ 0.0855316111 \\ 0.0715554800 \\ 0.0000021683 \end{bmatrix} + \begin{bmatrix} -0.036712 & 0.032300 \\ 0.047714 & -0.041611 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$$

Dari perhitungan di atas, maka matriks vektor *memory gate* sebagai berikut :

$$g_1 = \begin{bmatrix} -0.0021383 \\ 0.00282593 \end{bmatrix}$$

Selanjutnya dilakukan perhitungan untuk *output gate*. *Output gate* dihitung menggunakan persamaan 2.6. Nilai W_{xo} dan W_{ho} di inisialisasi secara acak dan nilai b_o di inisialiasai nol. Berikut merupakan perhitungan untuk *output gate*.

$$o_1 = \sigma(W_{xo}x_1 + W_{ho}h_0 + b_o)$$

$$o_1 = \sigma\left(\begin{bmatrix} 0.0335452 & 0.0229295 & 0.03630516 & -0.02009355 & -0.02319946 & 0.04819829 \\ -0.01947961 & -0.0322933 & -0.04229622 & -0.00412643 & 0.02155382 & 0.00907402 \end{bmatrix} \bullet \begin{bmatrix} 0.0854122487 \\ 0.0856348397 \\ 0.0854959654 \\ 0.0855316111 \\ 0.0715554800 \\ 0.0000021683 \end{bmatrix} + \begin{bmatrix} -0.019133 & -0.036155 \\ -0.041662 & -0.036990 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan di atas, maka matriks vektor *output gate* sebagai berikut :

$$o_1 = \begin{bmatrix} 0.501139 \\ 0.49829 \end{bmatrix}$$

Perhitungan selanjutnya yaitu memperbarui nilai *cell state*. Perhitungan *cell state* menggunakan persamaan 2.5. Nilai *cell state* sebelumnya pada *timestep* pertama di inisialisasi nol karena masih perhitungan awal. Perhitungan *cell state* menggunakan nilai f_1, i_1, dan g_1 yang telah dihitung sebelumnya. Berikut merupakan perhitungan *cell state*.

$$c_1 = f_1 \otimes c_0 + i_1 \otimes g_1$$

$$c_1 = \begin{bmatrix} 0.501777 \\ 0.49861 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.500089 \\ 0.499364 \end{bmatrix} \otimes \begin{bmatrix} -0.0021383 \\ 0.00282593 \end{bmatrix}$$

Dari perhitungan di atas, maka matriks vektor *cell state* sebagai berikut :

$$c_1 = \begin{bmatrix} -0.001069 \\ 0.0014111 \end{bmatrix}$$

Perhitungan terakhir yaitu *hidden state*. Nilai *hidden state* digunakan untuk perhitungan selanjutnya. Perhitungan *hidden state* menggunakan persamaan 2.7. Berikut merupakan perhitungan *hidden state*.

$$h_1 = o_1 \otimes tanh(c_1)$$

$$h_1 = \begin{bmatrix} 0.501139 \\ 0.49829 \end{bmatrix} \otimes tanh\left(\begin{bmatrix} -0.001069 \\ 0.0014111 \end{bmatrix} \right)$$

Dari perhitungan di atas maka nilai *hidden state* dari cell adalah sebagai berikut :

$$h_1 = \begin{bmatrix} -0.0005357 \\ 0.00070313 \end{bmatrix}$$

3.4.3.2 *Timestep* Kedua

Setelah didapatkan nilai *hidden state* dari perhitungan *timestep* sebelumnya, perhitungan dilanjutkan dengan *input* selanjutnya. Input kedua memiliki vektor [0.0855211115, 0.0851688733, 0.0857076770, 0.0855012235, 0.0348603984, 0.0000010594]. Perhitungan dilakukan dengan urutan yang sama dengan *timestep* sebelumnya. Nilai *hidden state* sebelumnya (h_1) digunakan dalam *timestep* kedua. Berikut merupakan perhitungan *forget gate* menggunakan persamaan 2.2 untuk *timestep* kedua.

$$f_2 = \sigma(W_{xf} X_2 + W_{hf} h_1 + b_f)$$

$$f_2 = \sigma\left(\begin{bmatrix} 0.04190454 & 0.02481002 & -0.01951982 & 0.03418238 & 0.00209754 & 0.01316079 \\ -0.03049609 & -0.00972463 & -0.03368445 & -0.02692162 & 0.04277313 & -0.0051742 \end{bmatrix} \bullet \begin{bmatrix} 0.0855211115 \\ 0.0851688733 \\ 0.0857076770 \\ 0.0855012235 \\ 0.0348603984 \\ 0.0000010594 \end{bmatrix} + \begin{bmatrix} -0.025804 & -0.019003 \\ -0.026350 & -0.034445 \end{bmatrix} \bullet \begin{bmatrix} -0.0005357 \\ 0.00070313 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$$

Dari perhitungan di atas, maka matriks vektor *forget gate timestep* kedua sebagai berikut :

$$f_2 = \begin{bmatrix} 0.501755 \\ 0.498214 \end{bmatrix}$$

Setelah perhitungan *forget gate* selesai, selanjutnya perhitungan untuk *input gate* dilakukan dengan persamaan 2.3. Berikut merupakan perhitungan *input gate* untuk *timestep* kedua.

$$i_2 = \sigma(W_{xi}x_2 + W_{hi}h_1 + b_i)$$

$$i_2 = \sigma\left(\begin{bmatrix} -0.04663104 & 0.04068497 & 0.0404686 & -0.02446541 & -0.01712576 & 0.03562227 \\ 0.04126063 & -0.04613255 & -0.03730052 & 0.04292736 & -0.03635566 & 0.01664639 \end{bmatrix} \right.$$
$$\bullet \begin{bmatrix} 0.0855211115 \\ 0.0851688733 \\ 0.0857076770 \\ 0.0855012235 \\ 0.0348603984 \\ 0.0000010594 \end{bmatrix} + \begin{bmatrix} 0.012406 & -0.000243 \\ 0.035534 & -0.042836 \end{bmatrix} \bullet \begin{bmatrix} -0.0005357 \\ 0.00070313 \end{bmatrix}$$
$$\left. + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan di atas, maka matriks vektor *input gate* untuk *timestep* kedua sebagai berikut :

$$i_2 = \begin{bmatrix} 0.500063 \\ 0.499689 \end{bmatrix}$$

Selanjutnya dilakukan perhitungan untuk *memory gate* dengan menggunakan persamaan 2.4. Berikut merupakan perhitungan *memory gate* untuk *timestep* kedua.

$$g_2 = tanh(W_{xg}x_2 + W_{hg}h_1 + b_f)$$

$$g_2 = tanh\left(\begin{bmatrix} 0.04133164 & 0.03050084 & -0.00227099 & -0.044721 & 0.00982435 & 0.00515036 \\ 0.0137399 & -0.0018558 & -0.04297994 & 0.03585858 & -0.03557081 & -0.03730149 \end{bmatrix} \bullet \begin{bmatrix} 0.0855211115 \\ 0.0851688733 \\ 0.0857076770 \\ 0.0855012235 \\ 0.0348603984 \\ 0.0000010594 \end{bmatrix} + \begin{bmatrix} -0.036712 & 0.032300 \\ 0.047714 & -0.041611 \end{bmatrix} \bullet \begin{bmatrix} -0.0005357 \\ 0.00070313 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$$

Dari perhitungan di atas, maka matriks vektor *memory gate* untuk *timestep* kedua sebagai berikut :

$$g_2 = \begin{bmatrix} 0.00249897 \\ -0.00089563 \end{bmatrix}$$

Selanjutnya dilakukan perhitungan untuk *output gate* dengan menggunakan persamaan 2.6. Berikut merupakan perhitungan *output gate* untuk *timestep* kedua.

$$o_2 = \sigma(W_{xo}x_2 + W_{ho}h_1 + b_o)$$

$$o_2 = \sigma\left(\begin{bmatrix} 0.0335452 & 0.0229295 & 0.03630516 & -0.02009355 & -0.02319946 & 0.04819829 \\ -0.01947961 & -0.0322933 & -0.04229622 & -0.00412643 & 0.02155382 & 0.00907402 \end{bmatrix} \bullet \begin{bmatrix} 0.0855211115 \\ 0.0851688733 \\ 0.0857076770 \\ 0.0855012235 \\ 0.0348603984 \\ 0.0000010594 \end{bmatrix} + \begin{bmatrix} -0.019133 & -0.036155 \\ -0.041662 & -0.036990 \end{bmatrix} \bullet \begin{bmatrix} -0.0005357 \\ 0.00070313 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan di atas, maka matriks vektor *output gate* untuk *timestep* kedua sebagai berikut :

$$o_2 = \begin{bmatrix} 0.501348 \\ 0.498088 \end{bmatrix}$$

Perhitungan selanjutnya yaitu memperbarui nilai *cell state*. Perhitungan *cell state* menggunakan persamaan 2.5. Nilai *cell state* sebelumnya (c_1), f_2, i_2, dan g_2 yang telah dihitung sebelumnya digunakan dalam perhitungan *cell state timestep* kedua. Berikut merupakan perhitungan *cell state* untuk *timestep* kedua.

$$c_2 = f_2 \otimes c_1 + i_2 \otimes g_2$$

$$c_2 = \begin{bmatrix} 0.501755 \\ 0.498214 \end{bmatrix} \otimes \begin{bmatrix} -0.001069 \\ 0.0014111 \end{bmatrix} + \begin{bmatrix} 0.500063 \\ 0.499689 \end{bmatrix} \otimes \begin{bmatrix} 0.00249897 \\ -0.00089563 \end{bmatrix}$$

Dari perhitungan di atas, maka matriks vektor *cell state* sebagai berikut :

$$c_2 = \begin{bmatrix} 0.00071326634 \\ 0.00025549332 \end{bmatrix}$$

Selanjutnya dilakukan perhitungan *hidden state* menggunakan persamaan 2.7. Nilai o_2 dan c_2 digunakan dalam perhitungan *hidden state timestep* kedua. Berikut merupakan perhitungan *hidden state*.

$$h_2 = o_2 \otimes tanh(c_2)$$

$$h_2 = \begin{bmatrix} 0.501348 \\ 0.498088 \end{bmatrix} \otimes tanh\left(\begin{bmatrix} 0.00071326634 \\ 0.00025549332 \end{bmatrix} \right)$$

Dari perhitungan di atas maka nilai *hidden state* dari cell adalah sebagai berikut :

$$h_2 = \begin{bmatrix} 0.00035759459 \\ 0.00012725815 \end{bmatrix}$$

Perhitungan untuk *timestep* ketiga dan seterusnya memiliki perhitungan yang sama. Setelah perhitungan LSTM selesai, dilanjutkan dengan perhitungan *dense layer* menggunakan persamaan 2.14. Berikut merupakan perhitungan *dense layer* dengan 1 unit dan fungsi aktivasi ReLU.

$$y = max\left(0, \begin{bmatrix} -0.01515524 & 0.0056293 \end{bmatrix} \cdot \begin{bmatrix} 0.00035759459 \\ 0.00012725815 \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} \right)$$

Berikut merupakan hasil perhitungan *dense layer* :

$$y = 0.0000061358$$

Hasil perhitungan *dense layer* merupakan nilai dari *min max normalization*, maka dari itu dilakukan *inverse* untuk mengembalikan ke nilai asal. Hasil dari *inverse* yaitu 3140.16.

3.4.4 *Bidirectional Long Short Term Memory*

Perhitungan manual untuk BiLSTM dilakukan dengan menggunakan data dan konfigurasi yang sama dengan perhitungan LSTM di atas. Perhitungan BiLSTM memiliki perbedaan dengan perhitungan LSTM dimana BiLSTM memiliki dua buah lapisan yaitu *forward* dan *backward layer*. *Forward layer* menghitung masukan data secara berurutan sedangkan *backward layer* menghitung masukan data dengan urutan terbalik. *Hidden state* yang dihasilkan oleh kedua lapisan digabungkan dan diteruskan ke dalam *dense layer* untuk menghasilkan sebuah keluaran. Berikut merupakan perhitungan *forward* dan *backward layer* untuk BiLSTM.

3.4.4.1 *Forward Layer*

Hasil perhitungan dari *forward layer* menggunakan hasil dari perhitungan LSTM *timestep* kedua dikarenakan proses perhitungan yang sama. Data masukan yang digunakan dalam *forward layer* yaitu pada tanggal 15 Mei 2018 pukul enam dan tujuh. Maka hasil *hidden state* dari *forward layer* sebagai berikut.

$$\overrightarrow{h_2} = \begin{bmatrix} 0.00035759459 \\ 0.00012725815 \end{bmatrix}$$

3.4.4.2 *Backward Layer*

Perhitungan *backward layer* menggunakan data masukan yang berbeda dengan *forward layer*. Pada contoh perhitungan manual digunakan data pada tanggal 15 Mei 2018 pukul delapan dan tujuh. Pada *timestep* pertama data masukan yang digunakan memiliki nilai vektor [0.0854907277, 0.0849299668, 0.0847803098, 0.0853407343, 0.1164648013, 0.0000035541]. Pertama dilakukan perhitungan *forget gate* menggunakan persamaan 2.2. Nilai bobot W_{xf}, W_{hf} di inisialisasi secara acak dan nilai bias b_f di inisialisasi nol. Nilai *hidden state*

sebelumnya (h_{t-1}) di inisialisasi nol karena ini merupakan perhitungan pertama. Berikut merupakan perhitungan *forget gate timestep* pertama untuk *backward layer*.

$$f_1 = \sigma(W_{xf}x_1 + W_{hf}h_0 + b_f)$$

$$f_1 = \sigma\left(\begin{bmatrix} 0.01307937 & -0.00214584 & 0.04244098 & -0.04639167 & -0.00390333 & 0.01422502 \\ 0.02722302 & -0.04189857 & -0.0005831 & 0.00846891 & 0.03577209 & -0.01530174 \end{bmatrix} \right.$$
$$\left. \bullet \begin{bmatrix} 0.0854907277 \\ 0.0849299668 \\ 0.0847803098 \\ 0.0853407343 \\ 0.1164648013 \\ 0.0000035541 \end{bmatrix} + \begin{bmatrix} 0.02227971 & 0.01624389 \\ -0.04032481 & 0.0496243 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan di atas, maka matriks vektor *forget gate* sebagai berikut :

$$f_1 = \begin{bmatrix} 0.500003 \\ 0.500902 \end{bmatrix}$$

Selanjutnya dilakukan perhitungan *input gate* menggunakan persamaan 2.3. Nilai bobot W_{xi}, W_{hi} di inisialisasi secara acak dan nilai bias b_i di inisialisasi nol. Berikut merupakan perhitungan *input gate*.

$$i_1 = \sigma(W_{xi}x_1 + W_{hi}h_0 + b_i)$$

$$i_1 = \sigma\left(\begin{bmatrix} -0.00663434 & 0.01118148 & -0.01595994 & 0.01829355 & 0.00083327 & 0.03976749 \\ 0.0083657 & -0.01311449 & 0.04566706 & 0.01977969 & -0.04014456 & -0.02572298 \end{bmatrix} \bullet \begin{bmatrix} 0.0854907277 \\ 0.0849299668 \\ 0.0847803098 \\ 0.0853407343 \\ 0.1164648013 \\ 0.0000035541 \end{bmatrix} + \begin{bmatrix} 0.04865564 & 0.02651871 \\ -0.04122456 & 0.04264008 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan di atas, maka matriks vektor *input gate* sebagai berikut :

$$i_1 = \begin{bmatrix} 0.50017 \\ 0.500121 \end{bmatrix}$$

Selanjutnya perhitungan dilanjutkan dengan menghitung *memory gate* menggunakan persamaan 2.4. Nilai bobot W_{xg}, W_{hg} di inisialisasi secara acak dan nilai bias b_f di inisialisasi nol. Berikut merupakan perhitungan untuk *memory gate*.

$$g_1 = tanh(W_{xg}x_1 + W_{hg}h_0 + b_f)$$

$$g_1 = tanh\left(\begin{bmatrix} 0.01997966 & -0.0122071 & 0.0209696 & -0.03410584 & -0.00693742 & -0.04688313 \\ -0.01143169 & 0.02658543 & -0.00890241 & 0.0239327 & -0.02926738 & -0.02594836 \end{bmatrix} \bullet \begin{bmatrix} 0.0854907277 \\ 0.0849299668 \\ 0.0847803098 \\ 0.0853407343 \\ 0.1164648013 \\ 0.0000035541 \end{bmatrix} + \begin{bmatrix} 0.02977636 & 0.03629604 \\ 0.00417225 & 0.02737285 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan di atas, maka matriks vektor *memory gate* sebagai berikut :

$$g_1 = \begin{bmatrix} 0.001269 \\ -0.0008404 \end{bmatrix}$$

Selanjutnya perhitungan dilanjutkan dengan menghitung *output gate* menggunakan persamaan 2.6. Nilai bobot W_{xo}, W_{ho} di inisialisasi secara acak dan nilai bias b_o di inisialisasi nol. Berikut merupakan perhitungan *output gate*.

$$o_1 = \sigma(W_{xo}x_1 + W_{ho}h_0 + b_o)$$

$$o_1 = \sigma\left(\begin{bmatrix} 0.02379688 & 0.01159606 & 0.00748036 & 0.02679605 & -0.01460398 & 0.03823848 \\ -0.00019294 & 0.03575045 & 0.00343465 & 0.02096329 & 0.00197579 & -0.02937965 \end{bmatrix} \right.$$
$$\left. \bullet \begin{bmatrix} 0.0854907277 \\ 0.0849299668 \\ 0.0847803098 \\ 0.0853407343 \\ 0.1164648013 \\ 0.0000035541 \end{bmatrix} + \begin{bmatrix} -0.02645702 & -0.03059443 \\ 0.0329853 & -0.03826525 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan di atas, maka matriks vektor *output gate* sebagai berikut :

$$o_1 = \begin{bmatrix} 0.50106 \\ 0.501333 \end{bmatrix}$$

Perhitungan dilanjutkan dengan menghitung *cell state* menggunakan persamaan 2.5. Nilai f_1, i_1, dan g_1 yang telah dihitung sebelumnya digunakan

dalam perhitungan *cell state*. Nilai *cell state* sebelumnya (c_0) di inisialisasi nol dikarenakan perhitungan pertama. Berikut merupakan perhitungan *cell state*.

$$c_1 = f_1 \otimes c_0 + i_1 \otimes g_1$$

$$c_1 = \begin{bmatrix} 0.500003 \\ 0.500902 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.50017 \\ 0.500121 \end{bmatrix} \otimes \begin{bmatrix} 0.001269 \\ -0.0008404 \end{bmatrix}$$

Dari perhitungan di atas, maka matriks vektor *cell state* sebagai berikut :

$$c_1 = \begin{bmatrix} 0.0006347 \\ -0.0004203 \end{bmatrix}$$

Perhitungan dilanjutkan dengan menghitung nilai *hidden state* yang digunakan untuk perhitungan selanjutnya menggunakan persamaan 2.7. Berikut merupakan perhitungan *hidden state*.

$$h_1 = o_1 \otimes tanh(c_1)$$

$$h_1 = \begin{bmatrix} 0.50106 \\ 0.501333 \end{bmatrix} \otimes tanh\left(\begin{bmatrix} 0.0006347 \\ -0.0004203 \end{bmatrix} \right)$$

Dari perhitungan di atas maka nilai *hidden state* dari cell adalah sebagai berikut :

$$\overleftarrow{h_1} = \begin{bmatrix} 0.00031802 \\ -0.00021071 \end{bmatrix}$$

Setelah didapat nilai *hidden state* dari *timestep* pertama, dilanjutkan perhitungan untuk *timestep* kedua. Data masukan yang digunakan memiliki nilai vektor [0.0855211115, 0.0851688733, 0.0857076770, 0.0855012235, 0.0348603984, 0.0000010594]. Perhitungan dilakukan dengan urutan yang sama dengan *timestep* sebelumnya. Berikut merupakan perhitungan *forget gate* menggunakan persamaan 2.2 untuk *timestep* kedua.

$$f_2 = \sigma(W_{xf}x_2 + W_{hf}h_1 + b_f)$$

$$f_2 = \sigma\left(\begin{bmatrix} 0.01307937 & -0.00214584 & 0.04244098 & -0.04639167 & -0.00390333 & 0.01422502 \\ 0.02722302 & -0.04189857 & -0.0005831 & 0.00846891 & 0.03577209 & -0.01530174 \end{bmatrix} \bullet \begin{bmatrix} 0.0855211115 \\ 0.0851688733 \\ 0.0857076770 \\ 0.0855012235 \\ 0.0348603984 \\ 0.0000010594 \end{bmatrix} + \begin{bmatrix} 0.02227971 & 0.01624389 \\ -0.04032481 & 0.0496243 \end{bmatrix} \bullet \begin{bmatrix} 0.00031802 \\ -0.00021071 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$$

Dari perhitungan di atas, maka matriks vektor *forget gate timestep* kedua sebagai berikut :

$$f_2 = \begin{bmatrix} 0.500119 \\ 0.500164 \end{bmatrix}$$

Selanjutnya perhitungan dilanjutkan untuk *input gate* menggunakan persamaan 2.3. Berikut merupakan perhitungan *input gate* untuk *timestep* kedua.

$$i_2 = \sigma(W_{xi}x_2 + W_{hi}h_1 + b_i)$$

$$i_2 = \sigma\left(\begin{bmatrix} -0.00663434 & 0.01118148 & -0.01595994 & 0.01829355 & 0.00083327 & 0.03976749 \\ 0.0083657 & -0.01311449 & 0.04566706 & 0.01977969 & -0.04014456 & -0.02572298 \end{bmatrix} \bullet \begin{bmatrix} 0.0855211115 \\ 0.0851688733 \\ 0.0857076770 \\ 0.0855012235 \\ 0.0348603984 \\ 0.0000010594 \end{bmatrix} + \begin{bmatrix} 0.04865564 & 0.02651871 \\ -0.04122456 & 0.04264008 \end{bmatrix} \bullet \begin{bmatrix} 0.00031802 \\ -0.00021071 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$$

Dari perhitungan di atas, maka matriks vektor *input gate* untuk *timestep* kedua sebagai berikut :

$$i_2 = \begin{bmatrix} 0.500155 \\ 0.500946 \end{bmatrix}$$

Selanjutnya dilakukan perhitungan untuk *memory gate* dengan menggunakan persamaan 2.4. Berikut merupakan perhitungan *memory gate* untuk *timestep* kedua.

$$g_2 = tanh(W_{xg}x_2 + W_{hg}h_1 + b_f)$$

$$g_2 = tanh\left(\begin{bmatrix} 0.01997966 & -0.0122071 & 0.0209696 & -0.03410584 & -0.00693742 & -0.04688313 \\ -0.01143169 & 0.02658543 & -0.00890241 & 0.0239327 & -0.02926738 & -0.02594836 \end{bmatrix} \bullet \begin{bmatrix} 0.0855211115 \\ 0.0851688733 \\ 0.0857076770 \\ 0.0855012235 \\ 0.0348603984 \\ 0.0000010594 \end{bmatrix} + \begin{bmatrix} 0.02977636 & 0.03629604 \\ 0.00417225 & 0.02737285 \end{bmatrix} \bullet \begin{bmatrix} 0.00031802 \\ -0.00021071 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$$

Dari perhitungan di atas, maka matriks vektor *memory gate* untuk *timestep* kedua sebagai berikut :

$$g_2 = \begin{bmatrix} -0.0006898 \\ 0.0015451 \end{bmatrix}$$

Selanjutnya dilakukan perhitungan untuk *output gate* dengan menggunakan persamaan 2.6. Berikut merupakan perhitungan *output gate* untuk *timestep* kedua.

$$o_2 = \sigma(W_{xo}x_2 + W_{ho}h_1 + b_o)$$

$$o_2 = \sigma\left(\begin{bmatrix} 0.0335452 & 0.0229295 & 0.03630516 & -0.02009355 & -0.02319946 & 0.04819829 \\ -0.01947961 & -0.0322933 & -0.04229622 & -0.00412643 & 0.02155382 & 0.00907402 \end{bmatrix} \bullet \begin{bmatrix} 0.0855211115 \\ 0.0851688733 \\ 0.0857076770 \\ 0.0855012235 \\ 0.0348603984 \\ 0.0000010594 \end{bmatrix} + \begin{bmatrix} -0.019133 & -0.036155 \\ -0.041662 & -0.036990 \end{bmatrix} \bullet \begin{bmatrix} 0.00031802 \\ -0.00021071 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan di atas, maka matriks vektor *output gate* untuk *timestep* kedua sebagai berikut :

$$o_2 = \begin{bmatrix} 0.501352 \\ 0.498088 \end{bmatrix}$$

Perhitungan selanjutnya yaitu memperbarui nilai *cell state*. Perhitungan *cell state* menggunakan persamaan 2.5. Berikut merupakan perhitungan *cell state* untuk *timestep* kedua.

$$c_2 = f_1 \otimes c_1 + i_1 \otimes g_1$$

$$c_2 = \begin{bmatrix} 0.500119 \\ 0.500164 \end{bmatrix} \otimes \begin{bmatrix} 0.0006347 \\ -0.0004203 \end{bmatrix} + \begin{bmatrix} 0.500155 \\ 0.500946 \end{bmatrix} \otimes \begin{bmatrix} -0.0006898 \\ 0.0015451 \end{bmatrix}$$

Dari perhitungan di atas, maka matriks vektor *cell state* sebagai berikut :

$$c_2 = \begin{bmatrix} -0.000027581 \\ 0.000563792 \end{bmatrix}$$

Perhitungan dilanjutkan dengan menghitung nilai *hidden state* yang digunakan untuk perhitungan selanjutnya menggunakan persamaan 2.7. Berikut merupakan perhitungan *hidden state*.

$$h_2 = o_2 \otimes tanh(c_2)$$

$$h_2 = \begin{bmatrix} 0.501352 \\ 0.498088 \end{bmatrix} \otimes tanh\left(\begin{bmatrix} -0.000027581 \\ 0.000563792 \end{bmatrix} \right)$$

Dari perhitungan di atas maka nilai *hidden state* dari cell adalah sebagai berikut :

$$\overleftarrow{h_2} = \begin{bmatrix} -0.00001382 \\ 0.00028081 \end{bmatrix}$$

Perhitungan untuk *timestep* ketiga dan seterusnya memiliki perhitungan yang sama. Selanjutnya perhitungan untuk keluaran BiLSTM dilakukan menggunakan persamaan 2.8. Berikut hasil perhitungan keluaran BiLSTM.

$$y_2 = \begin{bmatrix} 0.00031802 \\ -0.00021071 \\ -0.00001382 \\ 0.00028081 \end{bmatrix}$$

Setelah perhitungan keluaran BiLSTM selesai, dilanjutkan dengan perhitungan *dense layer* menggunakan persamaan 2.14. Berikut merupakan perhitungan *dense layer* dengan 1 unit dan fungsi aktivasi ReLU.

$$y = max\left(0, \begin{bmatrix} -0.03583039 & -0.03783366 & -0.02599822 & 0.02402738 \end{bmatrix} \cdot \begin{bmatrix} 0.00031802 \\ -0.00021071 \\ -0.00001382 \\ 0.00028081 \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix}\right)$$

Berikut merupakan hasil perhitungan *dense layer* :

$$y = 0.00000368357$$

Hasil perhitungan *dense layer* merupakan nilai dari *min max normalization*, maka dari itu dilakukan *inverse* untuk mengembalikan ke nilai asal. Hasil dari *inverse* yaitu 3140.0012

3.4.5 *Root Mean Square Error*

Nilai RMSE digunakan untuk mengukur perbedaan antara nilai prediksi dengan asli. RMSE juga mengukur presentase nilai *error* yang ada dalam model. Berikut merupakan contoh untuk perhitungan RMSE.

Time	*Actual*	*Predicted* LSTM	*Predicted* BiLSTM
2018-05-15 07:00:00	8645	8740	8700
2018-05-15 08:00:00	8670	8739	8720
2018-05-15 09:00:00	8722	8728	8650
2018-05-15 10:00:00	8700	8708	8695
2018-05-15 11:00:00	8788	8795	8790

$$RMSE_{LSTM} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})^2}$$

$$RMSE_{LSTM} = \sqrt{\frac{1}{5}((8740-8645)^2 + (8739-8670)^2 + (8728-8722)^2 + (8708-8700)^2 + (8795-8788)^2)}$$

$$RMSE_{LSTM} = \sqrt{\frac{1}{5}(9025+4761+36+64+49)}$$

$$RMSE_{LSTM} = \sqrt{2787}$$

$$RMSE_{LSTM} = 52.7920$$

$$RMSE_{BiLSTM} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})^2}$$

$$RMSE_{BiLSTM} = \sqrt{\frac{1}{5}((8700-8645)^2 + (8720-8670)^2 + (8650-8722)^2 + (8695-8700)^2 + (8790-8788)^2)}$$

$$RMSE_{BiLSTM} = \sqrt{\frac{1}{5}(3025+2500+5184+25+4)}$$

$$RMSE_{BiLSTM} = \sqrt{2147.6}$$

$$RMSE_{BiLSTM} = 46.34$$

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Bab ini menjelaskan proses implementasi dan pengujian terhadap sistem yang telah dibangun dan dirancang. Bab ini memberikan gambaran secara garis besar tentang hasil dari penelitian ini.

4.1 Lingkungan Implementasi

Bagian ini menjelaskan perangkat keras yang digunakan untuk proses implementasi dan pengujian. Proses implementasi dan pengujian menggunakan perangkat keras dan lunak.

4.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk implementasi dan pengujian adalah sebagai berikut :

1. *Laptop* LENOVO LEGION Y540.
2. *Processor* Intel Core i7-9750H CPU @ 2.6Ghz (6 CPUs).
3. *Solid State Drive* kapasitas 512 GB.
4. *RAM* dengan kapasitas 16 GB.

4.1.2 Spesifikasi Perangkat Lunak

Lingkungan perangkat lunak yang digunakan untuk implementasi dan pengujian adalah sebagai berikut :

1. Sistem Operasi Windows 10 Home.
2. IDE: Jupyter Notebook 6.1.4.
3. Development Tools Anaconda 3, 2020.07 (Python 3.8.3 64-bit)

4.2 Implementasi Perangkat Lunak

Bagian ini membahas proses implementasi perangkat lunak yang digunakan dalam penelitian ini. Hal yang dibahas meliputi daftar metode dan fungsi, penggunaan Jupyter Notebook dan *dataset*.

4.2.1 Implementasi *Class* dan Metode

Bagian ini membahas *class* dan metode yang digunakan dalam penelitian ini. *Class* dan metode yang digunakan dalam penelitian ini dibuat dari awal.

4.2.1.1 *Class Preprocessing*

Class Preprocessing dibuat untuk melakukan *data preprocessing*. *Class Preprocessing* mempunyai atribut dan metode sebagai berikut :

Tabel 4.1 Tabel atribut *Class Preprocessing*

No.	Nama Atribut	Tipe Data	Keterangan
1.	df	DataFrame	Menyimpan data dalam bentuk DataFrame.
2.	CLOSE_MIN	Float	Menyimpan nilai close minimum pada dataset.
3.	CLOSE_MAX	Float	Menyimpan nilai close maximum pada dataset.

Tabel 4.2 Tabel metode *Class Preprocessing*

No.	Metode	Masukan	Luaran	Keterangan
1.	__init__	df: DataFrame	-	Memuat dataset ke dalam sistem dan menyimpannya ke dalam atribut df.
2.	drop_column	column: Array	DataFrame	Membuang kolom yang tidak terpakai dalam dataset.
3.	min_max _normalization	df: DataFrame	DataFrame	Melakukan normalisasi min max terhadap dataset.
4.	inverse_min _max	df: DataFrame	DataFrame	Melakukan invers min max terhadap dataset.

4.2.1.2 *Class SplitData*

Class SplitData dibuat untuk melakukan pembagian data menjadi data latih dan data uji. Berikut merupakan daftar metode yang ada pada *Class SplitData* :

Tabel 4.3 Tabel metode *Class SplitData*

No.	Metode	Masukan	Luaran	Keterangan
1.	split_data	df: DataFrame, test_size: float	Array numPy	Membagi dataset menjadi data latih dan data uji.
2.	split_data_single	df: DataFrame, test_size: float	DataFrame	Membagi dataset menjadi data latih dan data uji untuk satu fitur.

4.2.1.3 *Class* Model

Class Model dibuat untuk membuat dan melatih model LSTM dan BiLSTM. Berikut merupakan daftar atribut dan metode yang ada pada *Class Model*.

Tabel 4.4 Tabel atribut *Class Model*

No.	Nama Atribut	Tipe Data	Keterangan
1.	UNIT	Array	Konstanta untuk menyimpan unit yang diuji.
2.	DROPOUT	Array	Konstanta untuk menyimpan dropout yang diuji.
3.	EPOCH	Array	Konstanta untuk menyimpan epoch yang diuji.
4.	BATCH_SIZE	Array	Konstanta untuk menyimpan batch size yang diuji.

Tabel 4.5 Tabel metode *Class Model*

No.	Metode	Masukan	Luaran	Keterangan
1.	build_model	method: String, file_path: String, X_train: Array, y_train: Array, unit: int, dropout: int, epoch: int, batch_size: int, model_name: String	-	Melakukan pelatihan sesuai dengan metode dan hyperparameter yang ditentukan.
2.	train_lstm	X_train: array, y_train: array	-	Melakukan proses pelatihan model dengan algoritme LSTM.
3.	train_bilstm	X_train: array, y_train: array	-	Melakukan proses pelatihan model dengan algoritme BiLSTM.

4.2.1.4 Class Visualization

Class Visualization dibuat untuk melakukan visualisasi terhadap pelatihan dan pengujian model. Berikut merupakan daftar metode yang ada pada *Class Visualization* :

Tabel 4.6 Tabel metode *Class Model*

No.	Metode	Masukan	Luaran	Keterangan
1.	plot_train_loss	model_name: String, file_path: String	-	Melakukan visualisasi terhadap hasil pelatihan model.
2.	plot_prediction	model_name: String, file_path: String, X_test: array, y_test: array	-	Melakukan visualisasi terhadap hasil pengujian model.

3.	visualize_train	file_path: String	-	Menyimpan dan menampilkan hasil visualisasi pelatihan model.
4.	visualize _prediction	file_path: String, X_test: array, y_test: array	-	Menyimpan dan menampilkan hasil visualisasi prediksi model.

4.2.2 Penggunaan Jupyter Notebook

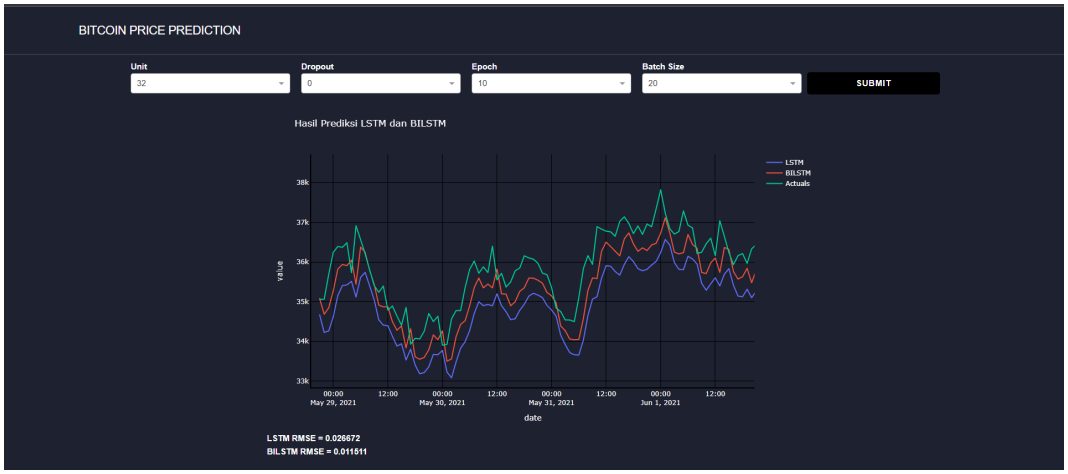
Dalam penelitian ini Jupyter Notebook digunakan sebagai *Integrated Development Environment*. Penggunaan Jupyter Notebook memudahkan untuk melakukan implementasi sistem menggunakan bahasa pemrograman Python.

4.2.3 Penggunaan Dataset

Dataset yang digunakan dalam penelitian ini yaitu *dataset* harga *bitcoin* periode Mei 2018 hingga Februari 2022. *Dataset* yang digunakan diambil dalam rentang satu jam. Semua data yang ada digunakan secara keseluruhan. Fitur *unix* dan *symbol* tidak digunakan dalam penelitian ini. Pembagian *dataset* dilakukan dengan cara membagi data menjadi data latih sebesar 80% atau sama dengan 22607 dan sisa 6628 sebagai data uji.

4.3 Implementasi Aplikasi

Bagian ini menjelaskan *user inteface* (UI) yang dibuat untuk sistem prediksi harga *bitcoin*. UI dibuat untuk menguji model yang telah dibuat pada saat proses pelatihan. Pengguna pertama harus memasukkan nilai unit, *dropout*, *epoch*, dan *batch size* yang telah disediakan di dalam *dropdown*. UI dibuat menggunakan *framework* Dash. Gambar 4.1 merupakan UI yang telah dibuat. Dalam UI tersebut terdapat beberapa *dropdown* yang dapat dipilih untuk memasukkan nilai unit, *dropout*, *epoch*, dan *batch size*. Terdapat tombol 'submit' yang dapat ditekan untuk menampilkan hasil prediksi model.



Gambar 4.1 *User Interface*

4.4 Pengujian

Bagian ini menjelaskan pengujian terhadap sistem prediksi harga *bitcoin* yang telah dibuat. Pengujian yang dilakukan yaitu menguji *hyperparameter* terhadap model. Pengujian model LSTM maupun BiLSTM menggunakan *dataset* yang sama. Pengujian dilakukan untuk menguji nilai akurasi yang dihasilkan oleh kedua model. Pengujian yang dilakukan dijelaskan sebagai berikut.

4.4.1 Skenario Pengujian *Long Short Term Memory*

Pengujian pertama dilakukan dengan algoritme LSTM. Pengujian LSTM menggunakan *hyperparameter* yang telah ditentukan sebelumnya. Seluruh kombinasi *hyperparameter* dilakukan dalam pengujian ini. Tabel 4.7 merupakan kombinasi *hyperparameter* yang dilakukan dalam pengujian ini.

Tabel 4.7 Tabel Skenario Pengujian *Long Short Term Memory*

Metode	**Hyperparameter**			
	Unit	**Dropout**	**Epoch**	**Batch Size**
LSTM	32	-	10	20
	64	0.2	50	50
	128	0.5	100	100
Total Pengujian	3 x 3 x 3 x 3			
	81			

4.4.2 Skenario Pengujian *Bidirectional Long Short Term Memory*

Selanjutnya pengujian dilakukan dengan algoritme BiLSTM. Pengujian BiLSTM menggunakan *hyperparameter* yang sama dengan algoritme LSTM.

Seluruh kombinasi parameter dilakukan dalam pengujian ini. Tabel 4.8 merupakan kombinasi *hyperparameter* yang dilakukan dalam pengujian ini.

Tabel 4.8 Tabel Skenario Pengujian *Bidirectional Long Short Term Memory*

Metode	**Hyperparameter**			
	Unit	**Dropout**	**Epoch**	**Batch Size**
BILSTM	32	-	10	20
	64	0.2	50	50
	128	0.5	100	100
Total Pengujian	3 x 3 x 3 x 3			
	81			

4.4.3 Pengujian Unit

Nilai unit yang digunakan dalam sebuah lapisan memengaruhi nilai akurasi. Semakin besar nilai unit maka pola tersembunyi baru yang ada dalam data bisa ditemukan. Pengujian nilai unit dilakukan untuk membuktikan bahwa nilai unit memengaruhi nilai akurasi. Nilai unit yang digunakan dalam penelitian ini yaitu 32, 64, dan 128. Pengujian ini mencari nilai unit yang memberikan akurasi terbaik.

4.4.4 Pengujian *Dropout*

Dropout merupakan sebuah *hyperparameter* yang digunakan untuk meningkatkan nilai akurasi. Pengujian nilai *dropout* digunakan untuk membuktikan apakah *dropout* memengaruhi nilai akurasi. Nilai *dropout* yang digunakan dalam penelitian ini yaitu tanpa *dropout*, 0.2, dan 0.5. Pengujian ini mencari nilai *dropout* yang memberikan akurasi terbaik.

4.4.5 Pengujian *Epoch*

Epoch merupakan sebuah *hyperparameter* yang digunakan untuk meningkatkan nilai akurasi. Pengujian nilai *epoch* dilakukan untuk membuktikan apakah *epoch* memengaruhi nilai akurasi. Nilai *epoch* yang digunakan dalam penelitian ini yaitu 10, 50, dan 100. Pengujian ini mencari nilai *epoch* yang memberikan nilai akurasi terbaik.

4.4.6 Pengujian *Batch Size*

Batch size merupakan sebuah *hyperparameter* yang digunakan untuk meningkatkan nilai akurasi. Pengujian nilai *batch size* dilakukan untuk

membuktikan apakah *epoch* memengaruhi nilai akurasi. Nilai *batch size* yang digunakan dalam penelitian ini yaitu 20, 50, dan 100. Pengujian ini mencari nilai *batch size* yang memberikan nilai akurasi terbaik.

4.4.7 Pengujian Kombinasi Parameter Terbaik

Pengujian ini dilakukan untuk mencari parameter terbaik untuk masing - masing metode. Pengujian ini dilakukan dengan mencoba seluruh kombinasi parameter yang ada dengan masing - masing metode. Nilai RMSE terkecil dijadikan sebagai penunjuk bahwa kombinasi parameter tersebut baik untuk masing - masing metode dalam memprediksi harga *bitcoin*.

4.4.8 Pengujian Kombinasi Parameter Optimal Untuk Masing - Masing Fitur

Pengujian ini dilakukan untuk menguji hasil akurasi masing - masing metode dengan kombinasi parameter optimal menggunakan satu fitur. Fitur yang digunakan diambil dari *dataset* secara satu per satu. Nilai RMSE terkecil dijadikan sebagai penunjuk bahwa kombinasi parameter untuk metode dan parameter tersebut baik dalam memprediksi harga *bitcoin*.

4.5 Hasil Pengujian

Dalam bagian ini dijelaskan hasil pengujian yang telah di paparkan pada bagian sebelumnya. Hasil pengujian merupakan pengujian *unit*, *dropout*, *epoch*, *batch size*, kombinasi parameter optimal, dan kombinasi parameter optimal untuk masing - masing fitur.

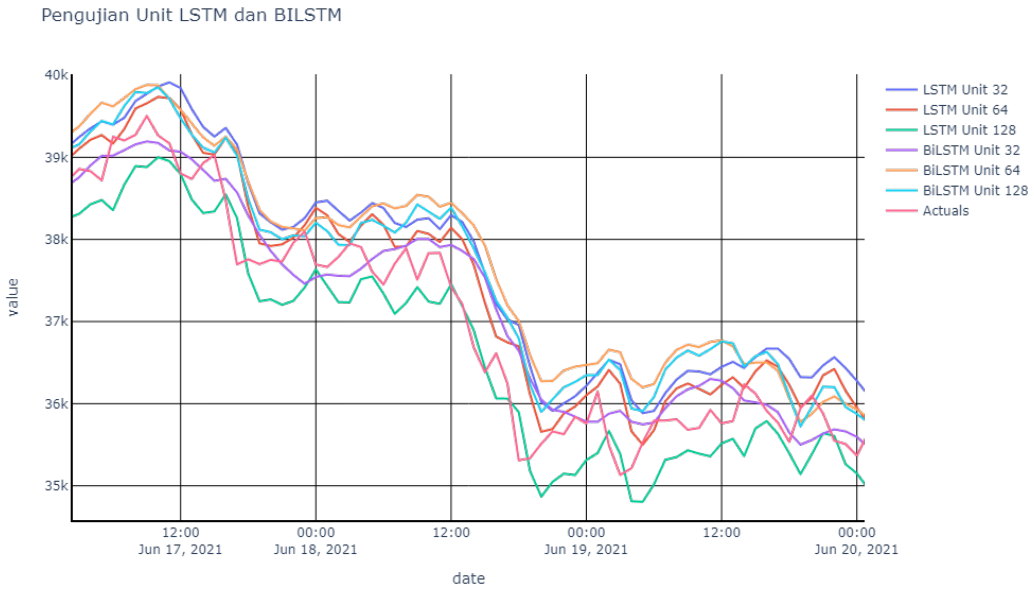
4.5.1 Hasil Pengujian *Unit*

Pengujian ini dilakukan dengan membandingkan hasil model dengan beragam unit sesuai dengan yang telah dijelaskan sebelumnya. Asumsi digunakan untuk *epoch*, *dropout*, dan *batch size* dalam pengujian ini. Asumsi diambil dari nilai *hyperparameter* yang menghasilkan akurasi terbaik pada penelitian [17]. Nilai *epoch*, *dropout*, *batch size* yang digunakan dalam pengujian ini yaitu 100, 0.5, 128. Hasil pengujian dapat dilihat pada tabel 4.9.

Tabel 4.9 Tabel Nilai RMSE Berdasarkan Unit

No	Metode	Parameter				RMSE
		Unit	Dropout	Epoch	Batch Size	
1	LSTM	32	0.5	100	100	0.009785
2		64				0.007285
3		128				0.011336
4	BILSTM	32				0.012869
5		64				0.011575
6		128				0.009227

Tabel 4.9 menunjukkan bahwa pengujian metode LSTM dengan nilai *unit* 64 menghasilkan nilai RMSE terkecil. Untuk pengujian metode BiLSTM nilai unit 128 menghasilkan nilai RMSE terkecil. Tabel di atas membuktikan bahwa nilai *unit* memengaruhi nilai RMSE yang dihasilkan baik untuk kedua metode. Dalam pengujian *unit* metode LSTM menghasilkan RMSE lebih kecil dibandingkan dengan BiLSTM. Gambar 4.2 merupakan hasil prediksi model pada pengujian unit.



Gambar 4.2 Pengujian Unit LSTM dan BiLSTM

4.5.2 Hasil Pengujian *Dropout*

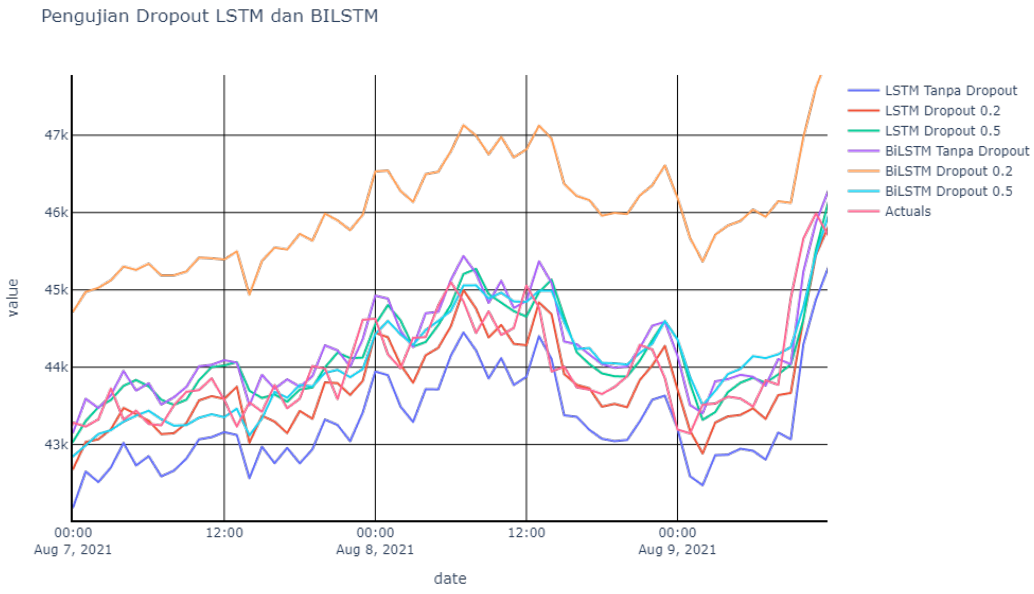
Pengujian ini dilakukan menggunakan nilai *unit* yang menghasilkan nilai RMSE terkecil berdasarkan pengujian sebelumnya. Asumsi digunakan untuk nilai

epoch, dan *batch size* dengan nilai masing - masing 100 dan 100 berdasarkan penelitian [17]. Hasil pengujian dapat dilihat pada tabel 4.10.

Tabel 4.10 Tabel Nilai RMSE Berdasarkan Dropout

No	Metode	Parameter				RMSE
		Unit	Dropout	Epoch	Batch Size	
1	LSTM	64	-	100	100	0.012111
2			0.2			0.006443
3			0.5			0.007285
4	BILSTM	128	-			0.006791
5			0.2			0.035820
6			0.5			0.009227

Tabel 4.10 menunjukkan bahwa pengujian metode LSTM dengan nilai *dropout* 0.2 menghasilkan nilai RMSE terkecil. Sedangkan pengujian metode BiLSTM tanpa *dropout* menghasilkan nilai RMSE terkecil. Hal tersebut membuktikan bahwa *dropout* memengaruhi akurasi baik untuk metode LSTM maupun BiLSTM. Dalam pengujian *dropout* metode LSTM menghasilkan RMSE lebih kecil dibandingkan dengan BiLSTM. Gambar 4.3 merupakan hasil prediksi model pada pengujian *dropout*.



Gambar 4.3 Pengujian Dropout LSTM dan BiLSTM

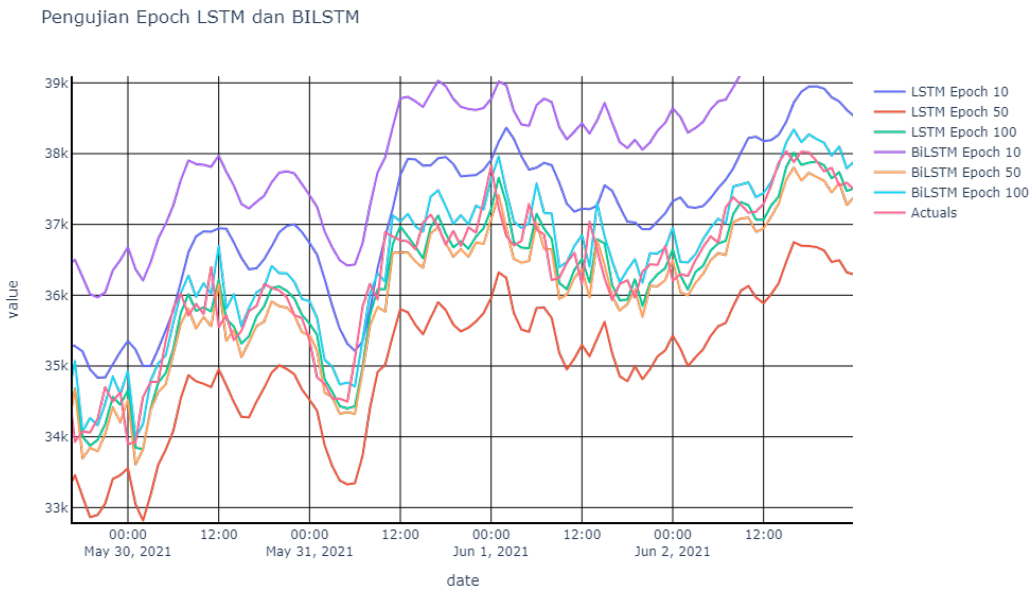
4.5.3 Hasil Pengujian *Epoch*

Pengujian ini dilakukan menggunakan nilai *unit* dan *dropout* yang menghasilkan RMSE terkecil berdasarkan pengujian sebelumnya. Asumsi digunakan untuk nilai *batch size* dengan nilai 100 berdasarkan penelitian [17]. Hasil pengujian dapat dilihat pada tabel 4.11.

Tabel 4.11 Tabel Nilai RMSE Berdasarkan Epoch

No	Metode	Parameter				RMSE
		Unit	Dropout	Epoch	Batch Size	
1	LSTM	64	0.2	10	100	0.012453
2				50		0.029882
3				100		0.006443
4	BILSTM	128	-	10		0.033056
5				50		0.008453
6				100		0.006791

Tabel 4.11 menunjukkan bahwa pengujian metode LSTM dengan nilai *epoch* 100 menghasilkan nilai RMSE terkecil. Begitu juga dengan pengujian metode BiLSTM dengan nilai *epoch* 100 menghasilkan nilai RMSE terkecil. Kedua metode menghasilkan nilai RMSE terkecil dengan nilai *epoch* yang sama. Dalam pengujian *epoch* metode LSTM menghasilkan RMSE lebih kecil dibandingkan dengan BiLSTM. Gambar 4.4 merupakan hasil prediksi model pada pengujian *epoch*.



Gambar 4.4 Pengujian Epoch LSTM dan BiLSTM

4.5.4 Hasil Pengujian *Batch Size*

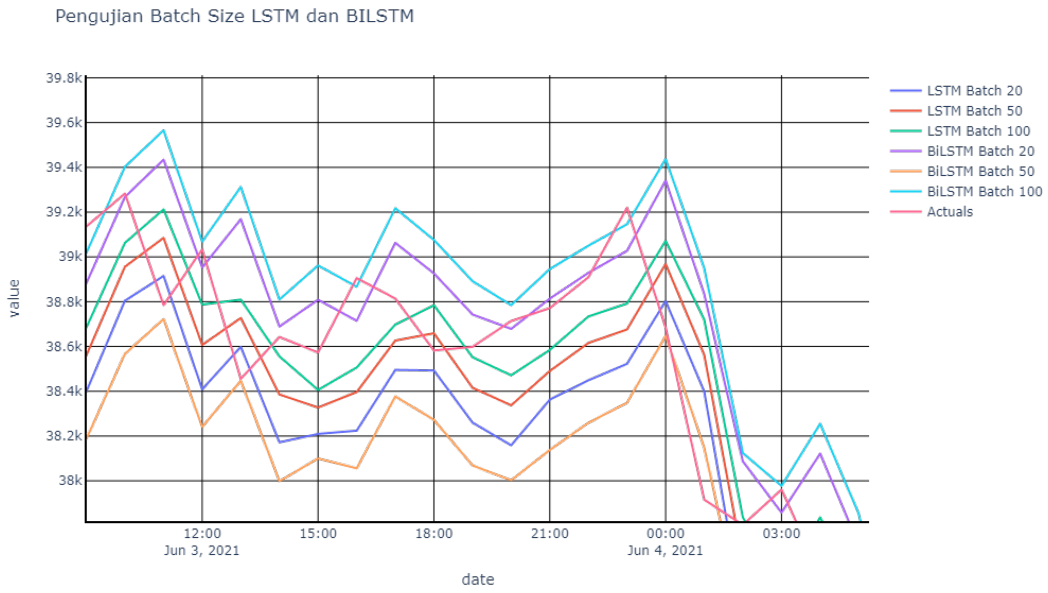
Pengujian ini dilakukan menggunakan nilai *unit*, *dropout*, *batch size* yang menghasilkan RMSE terkecil pada pengujian sebelumnya. Hasil pengujian dapat dilihat pada tabel 4.12.

Tabel 4.12 Tabel Nilai RMSE Berdasarkan Batch Size

No	Metode	Parameter				RMSE
		Unit	Dropout	Epoch	Batch Size	
1	LSTM	64	0.2	100	20	0.009427
2					50	0.007866
3					100	0.006443
4	BILSTM	128	-		20	0.006056
5					50	0.012030
6					100	0.006791

Tabel 4.12 menunjukkan bahwa pengujian metode LSTM dengan nilai *batch size* 100 menghasilkan nilai RMSE terkecil. Sedangkan untuk pengujian metode BiLSTM dengan nilai *batch size* 20 menghasilkan nilai RMSE terkecil. Dalam pengujian *batch size* metode BiLSTM menghasilkan RMSE lebih kecil dibandingkan LSTM. Kombinasi parameter yang digunakan pada pengujian ini belum tentu optimal, oleh karena itu pengujian dengan kombinasi seluruh

parameter dilakukan. Gambar 4.5 merupakan hasil prediksi model pada pengujian *batch size*.



Gambar 4.5 Pengujian Batch Size LSTM dan BiLSTM

4.5.5 Hasil Pengujian Kombinasi Parameter Optimal

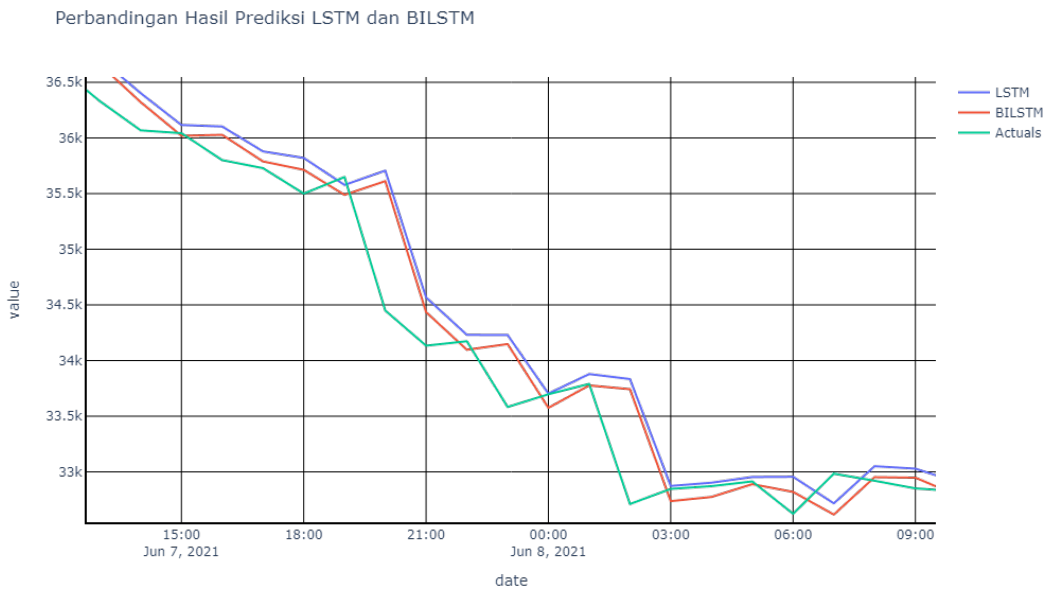
Pengujian ini dilakukan dengan kombinasi seluruh parameter *unit*, *dropout*, *epoch*, dan *batch size* yang telah dijelaskan pada bagian 4.4. Pengujian ini dilakukan untuk mencari nilai RMSE terkecil untuk masing - masing metode. Tabel 4.13 merupakan hasil pengujian yang menghasilkan RMSE paling kecil.

Tabel 4.13 Tabel Model dengan Nilai RMSE Terkecil

No	Metode	Parameter				RMSE
		Unit	Dropout	Epoch	Batch Size	
1	LSTM	128	0	100	100	0.005244
2	BILSTM	32	0	100	50	0.005156

Dapat dilihat pada tabel 4.13 membuktikan bahwa pengujian sebelumnya belum memiliki parameter yang optimal. Dalam pengujian ini metode BiLSTM menghasilkan nilai RMSE yang lebih kecil dibandingkan dengan LSTM. Hal ini membuktikan bahwa metode BiLSTM lebih baik dalam memprediksi harga *bitcoin*. Model LSTM memiliki parameter terbaik yaitu 128 *unit*, 0 *dropout*, 100 *epoch*, 100 *batch size*. Model BiLSTM memiliki parameter terbaik yaitu 32 *unit*, 0

dropout, 100 *epoch*, 50 *batch size*. Dalam pengujian ini metode LSTM dan BiLSTM memiliki nilai parameter *dropout* dan *epoch* yang sama yaitu 0 dan 100 untuk mencapai hasil yang optimal. Perbandingan hasil prediksi kedua model di atas dapat dilihat pada gambar 4.6.



Gambar 4.6 Perbandingan Hasil Prediksi LSTM dan BiLSTM

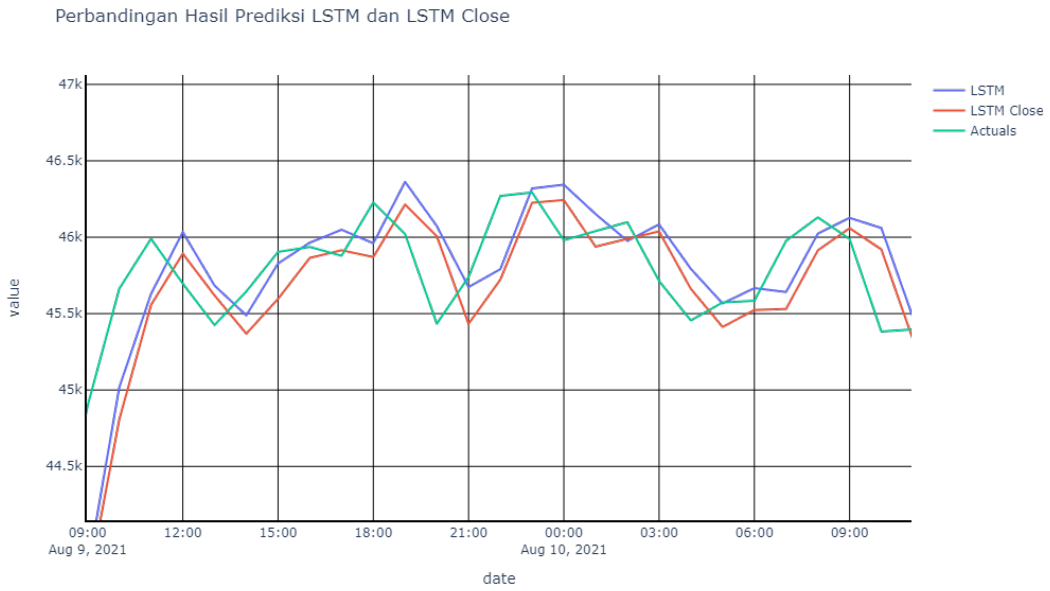
4.5.6 Hasil Pengujian Kombinasi Parameter Optimal Untuk Masing - Masing Fitur

Pengujian ini dilakukan menggunakan nilai parameter yang optimal yang dihasilkan dari pengujian sebelumnya dengan masing - masing fitur. Hasil pengujian dapat dilihat pada tabel 4.14.

Tabel 4.14 Tabel Pengujian Model LSTM dengan Parameter Optimal untuk Masing - masing Fitur

No	Metode	Fitur	Parameter				RMSE
			Unit	Dropout	Epoch	Batch Size	
1	LSTM	Close	128	0	100	100	0.005404
2		Open					0.014559
3		High					0.006664
4		Low					0.008172
5		Volume USD					0.663708
6		Volume BTC					0.410784

Dapat dilihat pada tabel 4.14 bahwa pengujian LSTM dengan fitur *close* menghasilkan RMSE terkecil dibandingkan dengan fitur lainnya. Akan tetapi pengujian LSTM dengan seluruh fitur menghasilkan RMSE yang lebih kecil dibandingkan hanya menggunakan salah satu fitur saja. Gambar 4.7 merupakan perbandingan hasil prediksi antara LSTM dengan seluruh fitur dengan fitur *close*.



Gambar 4.7 Perbandingan Hasil Prediksi LSTM dan LSTM Close

Meskipun pengujian LSTM dengan seluruh fitur menghasilkan RMSE yang lebih kecil dibandingkan dengan LSTM dengan fitur *close*, perbedaan hasil RMSE yang dihasilkan tidaklah signifikan. Hal tersebut membuktikan bahwa pengujian LSTM dengan seluruh fitur tidak memiliki dampak yang besar dalam memprediksi harga *bitcoin*. Hanya dengan fitur *close*, model LSTM bisa

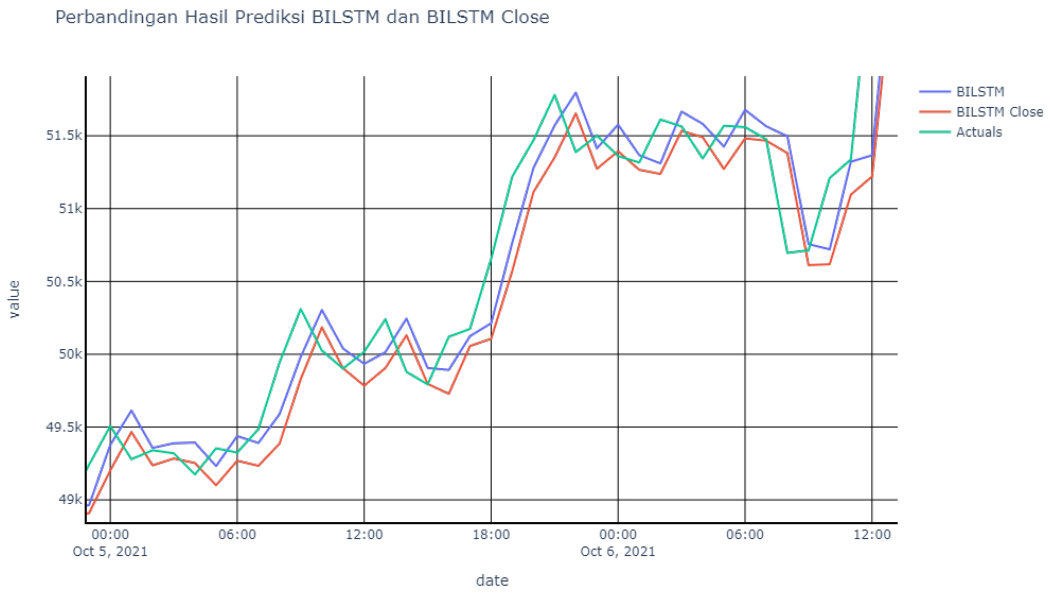
memprediksi harga *bitcoin* dengan RMSE yang rendah. Untuk LSTM penggunaan seluruh fitur tidak disarankan dibandingkan dengan fitur *close* saja, karena perbedaan nilai RMSE yang tidak signifikan dan proses pelatihan yang lebih lama.

Selanjutnya dilakukan pengujian dengan metode BiLSTM menggunakan parameter yang optimal dari pengujian sebelumnya dengan masing - masing fitur. Hasil pengujian dapat dilihat pada tabel 4.15.

Tabel 4.15 Tabel Pengujian Model BiLSTM dengan Parameter Optimal untuk Masing - masing Fitur

No	**Metode**	**Fitur**	**Parameter**				**RMSE**
			Unit	**Dropout**	**Epoch**	**Batch Size**	
1	BiLSTM	Close	32	0	100	50	0.005259
2		Open					0.007341
3		High					0.007146
4		Low					0.007056
5		Volume USD					0.663708
6		Volume BTC					0.442114

Dapat dilihat pada tabel di atas bahwa pengujian BiLSTM dengan fitur *close* menghasilkan RMSE terkecil dibandingkan dengan fitur lainnya. Pengujian BiLSTM dengan fitur *close* menghasilkan RMSE yang lebih kecil dibandingkan dengan LSTM dengan fitur *close*. Hal ini membuktikan bahwa BiLSTM masih mengungguli LSTM dalam memprediksi harga *bitcoin*. Akan tetapi BiLSTM dengan seluruh fitur memiliki nilai RMSE yang lebih kecil dari BiLSTM dengan fitur close. Gambar 4.8 merupakan perbandingan hasil prediksi antara BiLSTM dengan seluruh fitur dengan BiLSTM dengan fitur *close*.



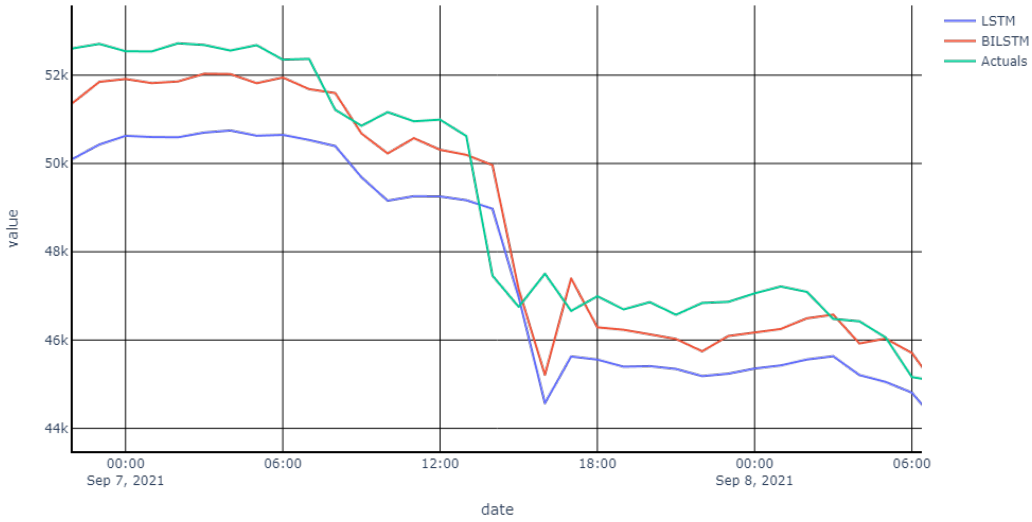
Gambar 4.8 Perbandingan Hasil Prediksi BILSTM dan BILSTM Close

Dilihat dari hasil pengujian BiLSTM dengan seluruh fitur dan fitur *close*, perbedaan hasil RMSE yang dihasilkan tidaklah signifikan. Sama dengan model LSTM, BiLSTM dengan fitur *close* saja bisa memprediksi harga *bitcoin* dengan nilai RMSE yang rendah. Maka dari itu penggunaan seluruh fitur tidak disarankan dibandingkan fitur *close* saja untuk BiLSTM. Dilihat dari hasil pengujian LSTM dan BiLSTM dengan seluruh fitur dan masing - masing fitur, BiLSTM mengungguli LSTM. Hal tersebut dapat dilihat dari nilai RMSE yang dihasilkan BiLSTM lebih kecil dibandingkan dengan LSTM untuk seluruh fitur maupun masing - masing fitur.

4.6 Analisis Kesalahan

Bagian ini menjelaskan analisis kesalahan yang terjadi pada saat pengujian model LSTM dan BiLSTM. Kesalahan terjadi karena adanya hal yang memengaruhi model pada saat pengujian. Berikut merupakan kesalahan yang dibahas.

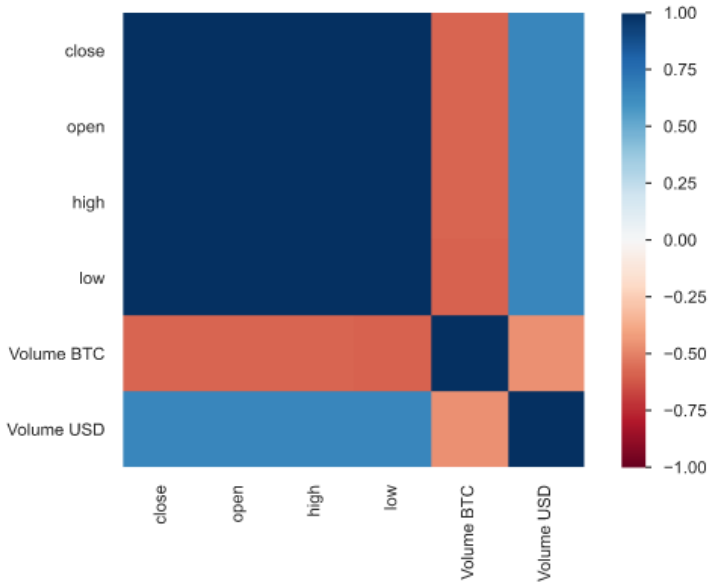
1. Dilihat dari karakteristik data, harga *bitcoin* seringkali mengalami perubahan yang cukup drastis. Hal ini dapat memengaruhi model dalam memprediksi harga *bitcoin*. Gambar 4.9 merupakan salah satu kesalahan yang terjadi pada saat proses prediksi.



Gambar 4.9 Kesalahan Prediksi

Dapat dilihat pada gambar 4.9 harga *bitcoin* mengalami penurunan yang cukup drastis. Pada saat terjadinya penurunan harga yang drastis, model LSTM dan BiLSTM mengalami kesalahan dalam memprediksi harga *bitcoin*. Kesalahan tersebut terjadi karena harga *bitcoin* mengalami perubahan yang drastis secara tiba - tiba, yang menyebabkan model tidak mengenali pola data. Meskipun harga mengalami penurunan yang cukup drastis, model BiLSTM memprediksi lebih baik dibandingkan dengan model LSTM. Hal ini dikarenakan BiLSTM memiliki *backward layer* yang menggunakan informasi yang ada pada masa depan untuk mengantisipasi kejadian ini. Perubahan yang drastis sering terjadi dalam kenaikan atau penurunan harga *bitcoin*, dan menyebabkan data menjadi tidak stasioner. Data yang tidak stasioner memengaruhi model dalam memprediksi harga *bitcoin*, maka dari itu algoritme lain seperti *gated recurrent unit* (GRU) dapat digunakan untuk mengatasi hal tersebut.

2. Pada saat proses pelatihan seluruh fitur digunakan untuk memprediksi harga *bitcoin*. Fitur yang digunakan yaitu *open*, *high*, *close*, *low*, Volume (BTC), dan Volume (USD) untuk memprediksi *close*. Akan tetapi fitur yang digunakan bisa saja tidak memiliki korelasi atau pengaruh terhadap *close*. Gambar 4.10 merupakan korelasi spearman.



Gambar 4.10 Korelasi Spearman

Dapat dilihat pada gambar 4.10 bahwa fitur Volume (BTC) tidak memiliki korelasi yang kuat dengan fitur *close*. Maka dari itu, salah satu cara untuk mengatasi hal tersebut yaitu dengan seleksi fitur. Seleksi fitur dilakukan agar pada saat proses pelatihan hanya fitur yang memiliki pengaruh saja yang digunakan.

BAB 5 KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil dari pengujian yang telah dilakukan dalam penelitian, menghasilkan kesimpulan untuk menjawab pertanyaan yang ada pada rumusan masalah. Kesimpulan yang dihasilkan yaitu :

1. Hasil penelitian menunjukkan algoritme LSTM memiliki nilai akurasi tertinggi dengan nilai RMSE sebesar 0.005244. Algoritme BiLSTM terbukti mampu memprediksi lebih baik dibandingkan dengan algoritme LSTM dengan nilai RMSE 0.005156.

2. Pada penelitian nilai *hyperparameter* optimal yang didapatkan adalah sebagai berikut: nilai unit sebesar 128, tanpa *dropout*, *epoch* sebesar 100, dan *batch size* sebesar 100 untuk algoritme LSTM. Sedangkan untuk algoritme BiLSTM nilai *hyperparameter* optimal yang didapatkan adalah sebagai berikut: nilai unit sebesar 32, tanpa *dropout*, *epoch* sebesar 100, dan *batch size* sebesar 50.

5.2 Saran

Saran untuk pengembangan penelitian selanjutnya dalam memprediksi harga *bitcoin* sebagai berikut :

1. Penelitian selanjutnya menggunakan algoritme GRU untuk mengatasi data yang tidak stasioner.

2. Dalam memprediksi harga *bitcoin* jangan hanya menggunakan data harga *bitcoin* saja tetapi gunakan juga faktor lain yang memengaruhi harga *bitcoin*. Beberapa data yang dapat digunakan yaitu data informasi *blockchain* atau *social sentiment* seperti *google trend* atau *tweet volume*.

3. Penelitian selanjutnya menggunakan *hyperparameter* yang lebih beragam untuk memaksimalkan nilai akurasi.

DAFTAR REFERENSI

[1] Khamis Hamed Al-Yahyaee, Mobeen Ur Rehman, Walid Mensi, Idries Mohammad Wanas Al-Jarrah. Can uncertainty indices predict Bitcoin prices? A revisited analysis using partial and multivariate wavelet approaches, The North American Journal of Economics and Finance,Volume 49, Pages 47-56, doi.org/10.1016/j.naref.2019.03.019 2019.

[2] Azari, A., Bitcoin price prediction: An ARIMA approach. arXiv preprint arXiv:1904.05315. 2019.

[3] Munim, Z.H., Shakil, M.H. and Alon, I. Next-day bitcoin price forecast. Journal of Risk and Financial Management, 12(2), p.103, 2019.

[4] Jay, P., Kalariya, V., Parmar, P., Tanwar, S., Kumar, N. and Alazab, M. Stochastic neural networks for cryptocurrency price prediction. Ieee access, 8, pp.82804-82818. 2020.

[5] Livieris, I.E., Stavroyiannis, S., Pintelas, E., Kotsilieris, T. and Pintelas, P. A dropout weight-constrained recurrent neural network model for forecasting the price of major cryptocurrencies and CCi30 index. Evolving Systems, 13(1), pp.85-100. 2021.

[6] Géron, A., Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.". 2019.

[7] Gulli, A., Kapoor, A. and Pal, S., Deep learning with TensorFlow 2 and Keras: regression, ConvNets, GANs, RNNs, NLP, and more with TensorFlow 2 and the Keras API. Packt Publishing Ltd, 2019.

[8] Brownlee, J., Long short-term memory networks with python: develop sequence prediction models with deep learning. Machine Learning Mastery. 2017.

[9] Cui, Z., Ke, R., Pu, Z. and Wang, Y., Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. arXiv preprint arXiv:1801.02143. 2018.

[10] Siami-Namini, S., Tavakoli, N. and Namin, A.S., The performance of LSTM and BiLSTM in forecasting time series. IEEE International Conference on Big Data (Big Data) (pp. 3285-3292). 2019.

[11] Ciaburro, G., Ayyadevara, V.K. and Perrier, A., Hands-on machine learning on google cloud platform: Implementing smart and efficient analytics using cloud ml engine. Packt Publishing Ltd. 2018.

[12] Keras team on Google, ”Keras API Reference,” Keras documentation. [Online]. Available: https://keras.io/api/.

[13] Jiang, X., Bitcoin price prediction based on deep learning methods. Journal of Mathematical Finance, 10(1), pp.132-139. 2020.

[14] Sunny, M.A.I., Maswood, M.M.S. and Alharbi, A.G., October. Deep learning-based stock price prediction using LSTM and bi-directional LSTM model. In 2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES) (pp. 87-92). IEEE, 2020.

[15] Jia, M., Huang, J., Pang, L. and Zhao, Q. Analysis and research on stock price of LSTM and bidirectional LSTM neural network. Proceedings of the 3rd International Conference on Computer Engineering, Information Science & Application Technology (ICCIA 2019), Chongqing, China (Vol. 10), 2019.

[16] S. McNally, J. Roche and S. Caton, "Predicting the Price of Bitcoin Using Machine Learning," 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), pp. 339-343, doi: 10.1109/PDP2018.2018.00060, 2018.

[17] K. A. Althelaya, E. M. El-Alfy and S. Mohammed, Evaluation of bidirectional LSTM for short-and long-term stock market prediction, 2018 9th International Conference on Information and Communication Systems (ICICS), pp. 151-156, doi: 10.1109/IACS.2018.8355458. 2018.

[18] Mukhopadhyay, U., Skjellum, A., Hambolu, O., Oakley, J., Yu, L. and Brooks, R., December. A brief survey of cryptocurrency systems. In 2016 14th annual conference on privacy, security and trust (PST) (pp. 745-752). IEEE. 2016.

[19] Chuen, D.L.K., Guo, L. and Wang, Y., Cryptocurrency: A new investment opportunity?. The journal of alternative investments, 20(3), pp.16-40. 2017.

[20] Böhme, R., Christin, N., Edelman, B. and Moore, T., Bitcoin: Economics, technology, and governance. Journal of economic Perspectives, 29(2), pp.213-38. 2015.

[21] Prasoon Kottarathil. (2022,March). Bitcoin Historical Dataset, Version 4.https://www.kaggle.com/datasets/prasoonkottarathil/btcinusd.