

**PREDIKSI HARGA ETHEREUM BERDASARKAN  
INFORMASI *BLOCKCHAIN* MENGGUNAKAN METODE  
*LONG SHORT TERM MEMORY***

**TUGAS AKHIR**

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana  
Program Studi Statistika



Disusun Oleh  
Nilda Aulia  
16611123

**PROGRAM STUDI STATISTIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS ISLAM INDONESIA  
YOGYAKARTA  
2020**

**HALAMAN PERSETUJUAN PEMBIMBING**

**TUGAS AKHIR**

Judul : Prediksi Harga Ethereum Berdasarkan Informasi  
*Blockchain* Menggunakan Metode *Long Short Term  
Memory*

Nama Mahasiswa : Nilda Aulia

NIM : 16611123

**TUGAS AKHIR INI TELAH DIPERIKSA DAN DISETUJUI UNTUK  
DIUJIKAN**

Yogyakarta, ( 9 Maret 2020 )

**Pembimbing**



**(Arum Handini Primandari, S.Pd.Si., M.Sc.)**

**HALAMAN PENGESAHAN**

**TUGAS AKHIR**

**PREDIKSI HARGA ETHEREUM BERDASARKAN INFORMASI  
BLOCKCHAIN MENGGUNAKAN METODE *LONG SHORT TERM  
MEMORY***

**Nama Mahasiswa : Nilda Aulia**

**NIM : 16611123**

**TUGAS AKHIR INI TELAH DIUJIKAN  
PADA TANGGAL : 2 April 2020**

**Nama Penguji:**

**Tanda Tangan**

1. (Mujiati Dwi Kartikasari, S.Si., M.Sc.)

2. (Dr. RB. Fajriya Hakim, M.Si)

3. (Arum Handini Primandari, S.Pd.Si., M.Sc.)

Mengetahui,

Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam



(Prof. Riyanto, S.Pd., M.Si., Ph.D.)

## KATA PENGANTAR

*Assalamu'alaikum Wr.Wb*

Alhamdulillahirabbil'aalamin, puji syukur kehadiran Allah SWT yang mana beliau telah melimpahkan rahmat, hidayah dan inayah nya sehingga penulis dapat menyelesaikan Tugas Akhir ini, Sholawat serta salam semoga terlimpahkan kepada junjungan nabi besar Muhammad SAW yang telah membawa umatnya dari zaman jahiliah menuju zaman terang benderang dan penuh ilmu pengetahuan.

Tugas Akhir dengan judul "**Prediksi Harga Ethereum berdasarkan Informasi Blockchain Menggunakan Metode Long Short Term Memory**" disusun sebagai syarat untuk memperoleh gelar sarjana Statistika pada program studi Statistika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia.

Selama proses penyusunan Tugas Akhir ini, tentunya tak lepas dari bantuan dan dukungan dari berbagai pihak, baik secara langsung maupun tidak langsung. Maka, penulis bermaksud menyampaikan terima kasih kepada :

1. Bapak Prof. Riyanto, S.Pd., M.Si., Ph.D. selaku dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia.
2. Bapak Dr.Edy Widodo,S.Si., M.Si. selaku ketua Program Studi Statistika Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia.
3. Ibu Arum Handini Primandari, S.Pd.Si., M.Si. selaku dosen pembimbing yang telah membantu dan memberikan masukan serta motivasi dari awal sampai selesai Tugas Akhir ini.
4. Seluruh staff pengajar Program Studi Statistika Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia yang telah memberikan ilmu ilmu yang bermanfaat bagi penulis.
5. Kedua Orang tua penulis, Bapak Syafril dan Ibu Mawarni, Dua saudara laki laki penulis, David Rilmadona dan Fio Saputra, Satu Saudara Perempuan

Penulis, Nitri Asriani serta seluruh keluarga besar yang selalu memberikan kasih sayang, dukungan, motivasi, dan doa untuk kelancaran segala urusan penulis.

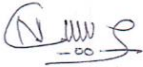
6. Sahabat seperjuangan : Barlinda Titania, Yesan Tiara, Diana Nabilla dan Firli Windika atas rasa kekeluargaan serta semua dukungan, bantuan, dan memberikan semangat bagi penulis
7. Sahabat penulis : Kinanti Dhea Larasati sebagai *partner* Tugas Akhir yang selalu menemani, membantu dan berperan besar dalam penyelesaian Tugas Akhir, serta Ria Amelia sahabat kami yang selalu ada.
8. Teman satu bimbingan Tugas Akhir yang saling menguatkan dan memberikan hiburan selama proses pengerjaan Tugas Akhir.
9. Teman Statistika Angkatan 2016, Teman IKS FMIPA UII, dan saudara di Jarvokasi IKS yang telah menemani proses perjalanan penulis selama mengembangkan diri dan memberikan pelajaran pelajaran berharga bagi penulis.
10. Semua pihak yang telah membantu yang tidak bisa penulis sebutkan satu per satu.

Semoga ALLAH SWT senantiasa selalu memberikan rahmat, Karunia dan Anugerahnya kepada mereka semua.

Penulis menyadari bahwa masih terdapat banyak kekurangan dalam penulisan Tugas Akhir ini dikarenakan keterbatasan pengetahuan dan kemampuan yang dimiliki oleh Penulis. Oleh karena itu, penulis mengharapkan kritik dan saran untuk menyempurnakan penulisan Tugas Akhir ini. Semoga Tugas Akhir ini dapat bermanfaat bagi banyak pihak.

*Wassalamualaikum Wr.Wb*

Yogyakarta, ( 9 Maret 2020 )

  
(Nilda Aulia)

## DAFTAR ISI

HALAMAN SAMPUL .....	i
HALAMAN PERSETUJUAN PEMBIMBING .....	ii
HALAMAN PENGESAHAN.....	iii
KATA PENGANTAR .....	iv
DAFTAR ISI.....	vi
DAFTAR TABEL.....	viii
DAFTAR GAMBAR .....	ix
DAFTAR LAMPIRAN.....	xi
PERNYATAAN.....	xii
ABSTRAK .....	xiii
ABSTRACT.....	xiv
BAB 1 PENDAHULUAN .....	1
1.1 Latar Belakang Masalah .....	1
1.2 Rumusan Masalah .....	3
1.3 Batasan Masalah.....	3
1.4 Tujuan Penelitian.....	3
1.5 Manfaat Penelitian.....	4
BAB 2 TINJAUAN PUSTAKA.....	5
BAB 3 LANDASAN TEORI .....	11
3.1 <i>Cryptocurrency</i> .....	11
3.2 Ethereum.....	13
3.3 <i>Blockchain</i> .....	15
3.4 Statistika Deskriptif .....	18
3.5 <i>Time Series</i> .....	21
3.4.2 Jenis Data Time Series Berdasarkan Plot Data .....	22
3.6 <i>Artificial Neural Network</i> .....	24
3.7 <i>Recurrent neural network</i> .....	27
3.8 <i>Long Short Term Memory</i> .....	29
3.9 <i>Adaptive Moment Estimation (Adam)</i> .....	35

3.10	Parameter Evaluasi .....	36
	3.7.1 Mean Square Error (MSE).....	36
	3.7.2 Root Mean Square Error (RMSE) .....	37
	3.7.3 Mean Absolute Percentage Error (MAPE) .....	37
	3.8.3 Training LSTM Network.....	38
	3.8.4 Testing .....	39
	3.8.5 Denormalisasi Data.....	39
BAB 4	METODOLOGI PENELITIAN .....	40
4.1	Populasi dan Sampel.....	40
4.2	Sumber Data .....	40
4.3	Variabel Penelitian .....	40
4.4	Metode Analisis Data .....	41
BAB 5	HASIL DAN PEMBAHASAN .....	43
5.1	Analisis Deskriptif data Ethereum.....	43
5.2	Analisis Long Short Term Memory.....	45
	5.2.1 Normalisasi Data .....	45
	5.2.2 Pembagian Data Training dan Testing .....	47
	5.2.3 Penentuan jumlah Neuron dan Epoch .....	48
	5.2.4 Prediksi harga Ethereum.....	52
BAB 6	PENUTUP .....	55
6.1	Kesimpulan.....	55
6.2	Saran .....	56
	DAFTAR PUSTAKA .....	57
	LAMPIRAN.....	62

## DAFTAR TABEL

<b>Tabel 2.1.</b> Daftar Penelitian Terdahulu .....	5
<b>Tabel 4.1.</b> Variabel Penelitian .....	40
<b>Tabel 5.1.</b> Rata-rata harga Ethereum .....	44
<b>Tabel 5.2.</b> Hasil Normalisasi Data $X_1 - X_5$ .....	46
<b>Tabel 5.3.</b> Hasil Normalisasi Data $X_6 - X_{10}$ & $Y$ .....	47
<b>Tabel 5.4.</b> Hasil Normalisasi Data $X_6 - X_{10}$ & $Y$ .....	47
<b>Tabel 5.5.</b> Hasil Percobaan <i>neuron</i> dan <i>Epoch</i> .....	48
<b>Tabel 5.6.</b> Hasil Prediksi .....	53



## DAFTAR GAMBAR

<b>Gambar 3. 1</b> ilustrasi proses transaksi keuangan tersentralisasi.....	12
<b>Gambar 3. 2</b> ilustrasi proses keuangan terdesentralisasi .....	13
<b>Gambar 3. 3</b> Logo Ethereum .....	14
<b>Gambar 3. 4</b> Transaksi pada <i>Blockchain</i> .....	16
<b>Gambar 3. 5</b> contoh Histogram.....	19
<b>Gambar 3. 6</b> Contoh Pie Chart.....	19
<b>Gambar 3. 7</b> Contoh Poligon .....	20
<b>Gambar 3. 8</b> Contoh Grafik Garis .....	20
<b>Gambar 3. 9</b> Pola Trand.....	23
<b>Gambar 3. 10</b> Pola Siklus .....	23
<b>Gambar 3. 11</b> Pola Musiman .....	24
<b>Gambar 3. 12</b> Pola <i>Irregular</i> .....	24
<b>Gambar 3. 13</b> Jaringan ANN .....	26
<b>Gambar 3. 14</b> Struktur ANN dasar .....	26
<b>Gambar 3. 15</b> Struktur umum RNN .....	27
<b>Gambar 3. 16</b> Proses RNN .....	28
<b>Gambar 3. 17</b> Jaringan LSTM .....	29
<b>Gambar 3. 18</b> Struktur LSTM .....	30
<b>Gambar 3. 19</b> <i>memory cell</i> .....	30
<b>Gambar 3. 20</b> Lapisan Sigmoid .....	31
<b>Gambar 3. 21</b> Alur Informasi pada <i>forget gate</i> .....	31
<b>Gambar 3. 22</b> Alur informasi yang melewati <i>input gate</i> .....	32
<b>Gambar 3. 23</b> memperbarui status <i>cell</i> .....	33
<b>Gambar 3. 24</b> Alur informasi melewati <i>output gate</i> .....	34
<b>Gambar 4. 1</b> <i>Flowchart</i> Analisis Data .....	42

<b>Gambar 5. 1</b> Grafik harga harian Ethereum dari 7 Januari 2018 s.d 7 Januari 2020 .....	43
<b>Gambar 5. 2</b> Grafik Rata rata harga Ethereum .....	45
<b>Gambar 5. 3</b> Hasil Bobot dari setiap Neuron untuk Variabel <i>Price</i> .....	50
<b>Gambar 5. 4</b> Bias untuk setiap Neuron.....	51
<b>Gambar 5. 5</b> Grafik data aktual dan Prediksi .....	52

## DAFTAR LAMPIRAN

<b>Lampiran 1</b> Data Ethereum dan informasi <i>Blockchain</i> .....	62
<b>Lampiran 2</b> <i>Output error</i> hasil percobaan.....	67
<b>Lampiran 3</b> <i>Script LSTM</i> dan <i>Output</i> .....	69

## PERNYATAAN

Dengan ini saya menyatakan bahwa dalam Tugas Akhir ini tidak terdapat karya yang sebelumnya pernah diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan orang lain, kecuali yang diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, (2 April 2020)



(Nilda Aulia)

## **ABSTRAK**

### **PREDIKSI HARGA ETHEREUM BERDASARKAN INFORMASI BLOCKCHAIN MENGGUNAKAN METODE *LONG SHORT TERM MEMORY***

Nilda Aulia

Program Studi Statistika, Fakultas MIPA

Universitas Islam Indonesia

*Cryptocurrency* menjadi salah satu topik yang hangat di perbincangkan karena meningkatnya berbagai jenis uang virtual seperti Bitcoin, Ethereum, dan lainnya, *cryptocurrency* dijadikan sebagai salah satu alternatif investasi bagi masyarakat Indonesia. *Cryptocurrency* terbaik selain Bitcoin adalah Ethereum. Ethereum merupakan jaringan *peer to peer* publik dengan mata uang digitalnya yang disebut Ether, tujuan Ethereum adalah untuk menjadi platform dimana *smart contract* dapat di ciptakan dan dijalankan. Prediksi harga Ethereum dengan mempertimbangkan informasi lain yaitu informasi *blockchain* diharapkan sebagai penyedia informasi yang tepat untuk melakukan perencanaan bagi setiap perorangan maupun pelaku bisnis untuk mengurangi resiko. Penelitian ini menggunakan metode *Long Short Term Memory* (LSTM) untuk melakukan prediksi dengan informasi *blockchain*. Metode LSTM merupakan pengembangan dari *Recurrent Neural Network* (RNN) dan RNN merupakan salah satu jenis dari *Artificial Neural Network* (ANN). Metode LSTM memerlukan beberapa parameter yang tepat guna menghasilkan prediksi yang akurat. Penelitian menganalisis beberapa parameter seperti jumlah neuron pada *hidden layer* dan *max epoch* yang paling tepat digunakan. Hasil analisis menunjukkan bahwa dengan menggunakan neuron 50 dan *max epoch* 500 mampu melakukan prediksi harga Ethereum menggunakan informasi *blockchain* dengan baik, dilihat dari *error* yang sangat kecil yaitu MAPE sebesar 1.69 %.

**Kata Kunci:** *Cryptocurrency, Ethereum, Blockchain, Prediksi, LSTM, MAPE*

## **ABSTRACT**

### **PREDICTION OF ETHEREUM PRICE BASED ON BLOCKCHAIN INFORMATION USING LONG SHORT TERM MEMORY METHOD**

Nilda Aulia

Program Studi Statistika, Fakultas MIPA

Universitas Islam Indonesia

Cryptocurrency is one of the hot topics discussed because it offers various kinds of money such as Bitcoin, Ethereum, and others, cryptocurrency is used as an alternative inversion for the people of Indonesia. The best cryptocurrency besides bitcoin is ethereum. Ethereum is a public peer to peer network with a digital currency called Ether, Ethereum's goal is to become a platform where smart contracts can be created and implemented. Price prediction by considering other information that is blockchain information is expected to be the right information provider for planning for each individual or business to increase risk. This research uses the Long Short Term Memory (LSTM) method to make predictions with blockchain information. LSTM method is the development of the Repeated Neural Network (RNN) and RNN is one type of Artificial Neural Network (ANN). The LSTM method buys several appropriate parameters to produce accurate predictions. The study analyzes several parameters such as the number of neurons in the hidden layer and the max epoch that is most appropriate. The results of the analysis show that by using neurons 50 and the age of 500 can predict ethereum prices using information blockchain well, seen from a very small error that is MAPE of 1.69%.

**Keywords:** *Cryptocurrency, Ethereum, Blockchain, Prediction , LSTM, MAPE*

# BAB 1 PENDAHULUAN

## 1.1 Latar Belakang Masalah

Dewasa ini, istilah *cryptocurrency* menjadi salah satu topik yang selalu hangat diperbincangkan karena meningkatnya berbagai jenis uang virtual seperti Bitcoin, Ethereum, dan lainnya. *Cryptocurrency* (mata uang *crypto*) adalah sebuah mata uang digital atau virtual yang di rancang sebagai alat tukar. Menurut wartaekonomi.co.id salah satu lembaga riset terkemuka dunia TNS merilis data bahwa 63% penduduk Indonesia sudah mengenal *cryptocurrency*, popularitasnya di Indonesia mengalahkan Malaysia, Perancis, Italia dan Romania. Bahkan menurut laporan IDACB menyebutkan Jakarta sebagai salah satu dari 10 *crypto-capital* tertinggi di dunia. Hasil riset mengatakan *cryptocurrency* memiliki potensi besar untuk di adopsi di Indonesia. *Cryptocurrency* di jadikan sebagai salah satu alternatif Investasi, Investasi merupakan salah satu dampak positif dari kemajuan teknologi saat ini, dengan berinvestasi, manusia dapat mempersiapkan hal finansial untuk masa mendatang, dari lembaga riset pemasaran inside ID, angka kepemilikan *cryptocurrency* mengalahkan kepemilikan instrumen investasi lain yaitu properti, reksadana dan saham (Rosmayanti, 2018)

*Cryptocurrency* terbaik selain bitcoin adalah ethereum. Ethereum adalah jaringan *peer to peer* publik yang terfokus untuk menjalankan kode program yang terdesentralisasi dengan mata uang digitalnya yang disebut Ether. Ethereum diciptakan oleh Vitalik Buterin pada tahun 2014 dan tujuan Ethereum adalah untuk menjadi platform dimana *smart contract* dapat di ciptakan dan dijalankan (LUNO, 2020). Kelebihan ethereum adalah yang pertama membuat *smart contract* dan memiliki aplikasi terdesentralisasi, atau dapps. Ethereum tentu saja dapat diperdagangkan setiap hari karena tidak memiliki periode tutup. Dengan ini para pemilik Ether dapat bertransaksi kapanpun dan dimanapun, semua transaksi di ethereum tercatat di *blockchain*.

Dunia kripto yang menyajikan investasi dengan tingkat yang sangat tinggi, tentu saja diikuti dengan tingkat resiko yang tinggi pula, maka dari itu, penyediaan

informasi yang tepat sangat berguna dalam melakukan perencanaan atau merancang strategi yang matang untuk dapat mengambil keputusan bagi setiap perorangan maupun pelaku bisnis untuk mengurangi resiko dan menangkap keuntungan keuntungan di depan. Salah satu cara yang dapat digunakan adalah prediksi harga Ethereum secara akurat. Sejauh ini, masih sedikit penelitian yang dilakukan untuk prediksi harga Ethereum, dan dalam melakukan prediksi diperlukan metode yang tepat agar hasil yang didapatkan bisa seakurat mungkin.

*Long Short Term Memory* (LSTM) merupakan pengembangan dari *Recurrent Neural Network* (RNN) dan RNN merupakan salah satu jenis dari *Artificial Neural Network* (ANN). Dari beberapa cara dalam melakukan prediksi, Algoritma ANN banyak di gunakan untuk memprediksi harga di kemudian hari. ANN merupakan suatu sistem pemrosesan informasi yang meniru kerja otak manusia yang kemudian di implementasikan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran berlangsung. RNN merupakan salah satu bentuk arsitek ANN yang di rancang khusus untuk memproses data yang bersambung atau berurutan (*sequential data*). RNN tidak membuang informasi pada masa lalu dalam proses pembelajarannya, hal inilah yang membedakan RNN dengan ANN biasa. RNN di rancang khusus untuk memproses data yang bersambung atau berurutan. RNN biasanya digunakan untuk menyelesaikan masalah dengan data *time series*. Arsitek RNN dapat mengingat data *input* sebelumnya untuk setiap *output*, pada intinya RNN tidak melupakan data sebelumnya, namun arsitek dasar dari RNN dapat menimbulkan masalah *vanishing gradient* atau *exploding gradient* sehingga dikatakan kurang cocok dalam mempelajari pola data berurutan yang terlalu panjang. Maka, Pada tahun 1997 Hochreiter dan Schmidhuber menemukan arsitektur baru untuk masalah data yang terlalu lama, yaitu dengan arsitektur *Long Short Term Memory* (LSTM). LSTM menyaring informasi melalui struktur gerbang untuk mempertahankan dan memperbarui keadaan sel memori. Struktur pintunya mencakup gerbang *input*, *forget gate*, dan gerbang *output*. Setiap sel memori memiliki tiga lapisan sigmoid dan satu lapisan tanh. Salah satu penelitian terdahulu yang dilakukan oleh (Zheng & dkk, 2017) model LSTM mampu mengungguli metode lainnya dengan



menghasilkan nilai RMSE dan MAPE yang lebih rendah. Nilai RMSE yang dihasilkan sebesar 0,0702, sementara nilai MAPE yang dihasilkan sebesar 0,0535%

Pada penelitian ini, peneliti tertarik untuk melakukan prediksi harga Ethereum dengan menggunakan informasi *blockchain* menggunakan LSTM, karena belum ada penelitian yang melakukan prediksi terhadap harga Ethereum menggunakan variabel lain yaitu informasi dari *blockchain*.

## **1.2 Rumusan Masalah**

Berdasarkan latar belakang masalah di atas, maka rumusan masalah yang akan di bahas pada penelitian ini adalah :

1. Bagaimana deskripsi atau gambaran umum harga Ethereum selama 2 tahun terakhir?
2. Bagaimana penerapan LSTM dalam melakukan prediksi harga Ethereum berdasarkan informasi *blockchain*?

## **1.3 Batasan Masalah**

Batasan masalah dalam penelitian ini adalah sebagai berikut :

1. Data yang digunakan adalah data Historis Ethereum dalam *United States Dollar (US\$)*, informasi *blockchain* yaitu *Block Size, Average Block Usage, Block Time, Hash Rate, Difficulty, Fee Mining Reward, Mining Revenue, total account, dan transaction*.
2. Data yang digunakan selama 2 tahun untuk masing masing variabel yaitu dari 8 Januari 2018 sampai dengan 7 Januari 2020.
3. *Software* yang di gunakan adalah *Python*.

## **1.4 Tujuan Penelitian**

Tujuan dari penelitian ini yaitu :

1. Mengetahui deskripsi atau gambaran umum harga Ethereum selama 2 tahun terakhir.
2. Mengetahui penerapan LSTM dalam melakukan prediksi Ethereum berdasarkan informasi *blockchain*.

## **1.5 Manfaat Penelitian**

Manfaat dari penelitian ini adalah sebagai berikut :

1. Memberikan pengetahuan terkait pemodelan dan prediksi harga ethereum berdasarkan informasi *blockchain*.
2. Penelitian ini dapat digunakan sebagai sarana belajar dan mendalami ilmu terkait perdagangan atau investasi sebagai pertimbangan dalam mengambil keputusan.
3. Dapat menjadi referensi bagi peneliti selanjutnya tentang metode LSTM maupun tentang prediksi harga Ethereum informasi *blockchain*.

## BAB 2 TINJAUAN PUSTAKA

Dalam melakukan sebuah penelitian, maka diperlukan acuan penelitian sebelumnya untuk memperkuat penelitian yang dilakukan penulis dan menghindari terjadinya duplikasi penelitian. Penelitian dengan tema prediksi sudah banyak dilakukan oleh peneliti sebelumnya, namun sangat sedikit yang melakukan prediksi terhadap Ethereum menggunakan informasi *blockchain* dengan metode LSTM. Berikut beberapa penelitian yang dijadikan acuan oleh penulis :

**Tabel 2.1. Daftar Penelitian Terdahulu**

No	Penulis/Tahun	Judul Penelitian	Metode	Hasil Penelitian
1.	(Zheng & dkk, 2017)	<i>Electric Load Forecasting in Smart Grid Using Long-Short-Term-Memory based Recurrent neural network</i>	LSTM	RNN berbasis LSTM untuk mengatasi masalah peramalan pada beban listrik. Dalam penelitian ini model LSTM dibandingkan dengan beberapa model peramalan lain yaitu dengan SARIMA, NARX, SVR, dan NNETAR. Berdasarkan hasil penelitian, model LSTM mampu mengungguli metode peramalan lain dengan menghasilkan nilai RMSE dan MAPE yang lebih rendah. Nilai RMSE yang dihasilkan yaitu sebesar 0,0702, sementara nilai MAPE yang dihasilkan sebesar 0,0535%.
2.	(Jang & Lee, 2017)	<i>An Empirical Study on Modeling and Prediction</i>	BNN	Pemilihan fitur yang paling relevan dari informasi <i>Blockchain</i> yang terlibat dalam

No	Penulis/Tahun	Judul Penelitian	Metode	Hasil Penelitian
		<i>of Bitcoin Prices With Bayesian Neural Networks Based on Blockchain Information</i>		<p>Penawaran dan permintaan Bitcoin dan menggunakannya untuk melatih model guna meningkatkan kinerja prediksi.</p> <p>Pada penelitian dilakukan studi empiris yang membandingkan jaringan saraf Bayesian dengan model <i>benchmark</i> linear dan non-linear lainnya pada pemodelan dan memprediksi proses Bitcoin. studi empiris menunjukkan bahwa BNN memiliki kinerja baik dalam memprediksi seri waktu harga Bitcoin dan menjelaskan volatilitas tinggi dari harga Bitcoin baru-baru ini.</p>
3.	(Karim & dkk, 2019)	<i>Multivariate LSTM-FCNs for time series classification</i>	MLSTM-FCN	<p>Dalam beberapa waktu terakhir, klasifikasi deret waktu multivarian telah menerima banyak perhatian. Penelitian mengusulkan mengubah model klasifikasi deret waktu univariat yang ada, LSTM-FCN dan Perhatian LSTM-FCN (ALSTM-FCN), menjadi multivarian model klasifikasi deret waktu dengan menambah blok konvolusional penuh dengan pemerasan dan eksitasi blokir untuk lebih meningkatkan</p>

No	Penulis/Tahun	Judul Penelitian	Metode	Hasil Penelitian
				<p>akurasi. Model yang di usulkan mengungguli sebagian besar model sementara membutuhkan preprocessing minimum. Model yang diusulkan bekerja secara efisien di berbagai bidang, tugas klasifikasi deret waktu multivariat yang kompleks seperti pengenalan aktivitas atau pengenalan tindakan. Selain itu, model yang diusulkan sangat efisien pada waktu pengujian.</p>
4.	(C.Lipton & dkk, 2016)	<i>LEARNING TO DIAGNOSE WITH LSTM RECURRENT NEURAL NETWORKS – REVISED</i>	LSTM	<p>Penelitian ini dari data medis klinis, terutama di unit perawatan intensif (ICU), terdiri dari multivariat serangkaian pengamatan. Meskipun berpotensi berisi banyak wawasan, data sulit ditambang efektif, karena panjang bervariasi, pengambilan sampel tidak teratur dan data yang hilang. RNN terutama LSTM adalah model pembelajaran yang kuat dan semakin populer dari data urutan. Mereka secara efektif memodelkan berbagai urutan panjang dan tangkapan ketergantungan jarak jauh. Dalam penelitian disajikan studi pertama untuk mengevaluasi secara empiris</p>

No	Penulis/Tahun	Judul Penelitian	Metode	Hasil Penelitian
				<p>kemampuan LSTM untuk mengenali pola dalam serangkaian waktu pengukuran klinis multivariat. Secara khusus, penelitian mempertimbangkan klasifikasi multilabel diagnosis, pelatihan sebuah model untuk mengklasifikasikan 128 diagnosis yang diberikan 13 sampel yang sering tetapi tidak teratur pengukuran klinis.</p>
5.	(R & Das, 2018)	<i>Cryptocurrency Price Prediction Using Long-Short Term Memory Model</i>	LSTM	<p><i>Cryptocurrency</i> adalah digital atau virtual yang terdesentralisasi mata uang. Penggunaan kriptografi untuk keamanan membuatnya sulit palsu. Bitcoin, Ethereum, Ripple, uang Bitcoin, Bit connect, Dash, Ethereum Klasik, Iota, Litecoin, Monero, Nem, Neo, Numeraire, Stratis, Gelombang dll adalah beberapa <i>Cryptocurrency</i> populer. <i>Cryptocurrency</i> mulai mendapat perhatian pada 2013 dan sejak itu menyaksikan sejumlah besar transaksi dan karenanya harga fluktuasi. Pasar <i>cryptocurrency</i> mirip dengan saham pasar. Ini telah mendapatkan perhatian publik dan</p>

No	Penulis/Tahun	Judul Penelitian	Metode	Hasil Penelitian
				<p>prediksi yang sangat efektif pergerakan harga <i>cryptocurrency</i> akan membantu masyarakat untuk berinvestasi menguntungkan dalam sistem. Makalah ini mencoba memprediksi harga <i>Cryptocurrency</i>. Teknik pembelajaran mesin adalah diimplementasikan dan penggunaan pengoptimal Adam dan Long Short Term Memori (LSTM) terbukti sangat efisien di memprediksi harga mata uang digital.</p>
6.	(Aldi & dkk, 2018)	Analisis dan Implementasi <i>Long Short Term Memory Neural Network</i> untuk Prediksi Harga Bitcoin	LSTM	<p>Sistem yang dibangun dalam penelitian ini yaitu menggunakan metode jaringan syaraf tiruan yaitu arsitektur <i>Long Short Term Memory Neural Networks</i>. Teknik ini memerlukan parameter yang tepat untuk mendapatkan hasil prediksi yang akurat. Dilakukan analisis beberapa parameter seperti jumlah pola time series, jumlah neuron hidden, max <i>epoch</i>, dan komposisi data latih dan uji terhadap akurasi prediksi yang didapatkan. Hasil analisis menunjukkan bahwa sistem yang dibangun mampu</p>

No	Penulis/Tahun	Judul Penelitian	Metode	Hasil Penelitian
				memprediksi harga Bitcoin dengan baik, dengan rata-rata tingkat akurasi sebesar 93.5% terhadap data <i>testing</i> .

Meskipun pada **Tabel 2.1.** telah disebutkan beberapa penelitian dengan tema yang serupa, namun pada penelitian ini terdapat perbedaan yaitu Variabel Y yang akan digunakan adalah harga Ethereum dan variabel lainnya sebagai  $X_1$  sampai  $X_{10}$  merupakan variabel harga Ethereum periode sebelumnya dan informasi *Blockchain*. Pada penelitian menggunakan metode LSTM Multivariat.



## BAB 3 LANDASAN TEORI

### 3.1 *Cryptocurrency*

Secara sederhana, *cryptocurrency* dapat diartikan sebagai mata uang digital. *Cryptocurrency* adalah suatu metode membuat “koin” virtual dan menyediakannya serta mengamankan kepemilikan dan transaksi menggunakan masalah kriptografi, hal ini dirancang supaya mudah di verifikasi namun secara komputasi sulit untuk menemukan solusi (Harwick, 2016). Kriptografi secara sederhana merupakan teknik melindungi informasi dengan mentransformasikannya (misal mengenskripsi informasi) ke dalam format yang tidak dapat di baca dan hanya dapat di uraikan (dideskripsikan) oleh seseorang yang memiliki kunci rahasia (Houben & Snyers, 2018). *Cryptocurrency* menggunakan fungsi yang berbeda, untuk tujuan ini yang paling umum yaitu target *hash* yang disesuaikan setiap kali berdasarkan daya komputasi total pada jaringan, yang memiliki keuntungan menjaga waktu antara solusi lebih atau kurang konstan. Bukti kerja intensif komputasi adalah metode dimana transaksi diverifikasi sebagai unik dan dapat dipercaya. Untuk mendorong partisipasi, *transactor* dapat menyertakan biaya transaksi yang masuk ke pengguna pertama yang berhasil memverifikasinya. Selain itu, jaringan menghadiahkan verifier dengan beberapa jumlah koin setelah mereka berhasil memverifikasi satu blok transaksi. Proses ini yang di sebut “penambangan”. Penambangan adalah cara dimana pasokan koin pada jaringan di perluas, dan kesulitan yang dapat di sesuaikan memastikan bahwa kemajuan komputasi tidak akan mempengaruhi tingkat ekspansi (Harwick, 2016)

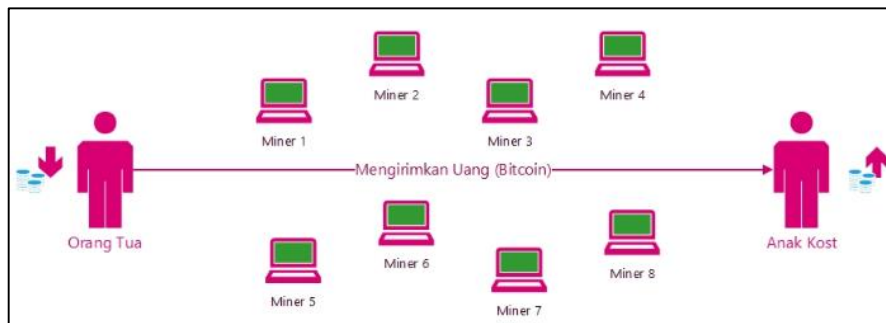
*Cryptocurrency* memiliki sifat terdesentralisasi yang artinya transaksi dilakukan secara *peer-to-peer* dari pengirim ke penerima tanpa adanya perantara. Beberapa *cryptocurrency* yang terkenal di Indonesia seperti Bitcoin, Ethereum, Litecoin, Dash, Ripple, Bitcoin Hash, Bitcoin Gold, Zcash, Monero, Maker, Byteball dan lainnya. *Cryptocurrrency* berbeda dengan mata uang pada umumnya, karena model transaksi yang biasa di gunakan oleh orang umum adalah tersentralisasi. Berikut penjelasan tentang perbedaan dua sifat dalam studi kasus (Eka, 2018)

Sifat tersentralisasi dicontohkan pada model transaksi yang selama ini pada umumnya digunakan oleh masyarakat. Misalnya orang tua mengirimkan uang kepada anaknya, maka yang ia lakukan ialah menggunakan layanan perbankan (ATM, *Mobile Banking*, atau datang langsung ke *bank* terkait) lalu mentransfer sejumlah uang ke nomor rekening anaknya tersebut. Transaksi tersebut pada dasarnya dilakukan melalui perantara bank dan layanan yang dipercaya. Jadi prosesnya uang yang ditransfer masuk ke bank terlebih dulu, lalu diteruskan ke penerima. Prosesnya *real time* sehingga perpindahan tersebut tidak terasa. Namun yang dirasakan karena prosesnya melalui perantara, maka ada imbalan yang harus dibayarkan, yakni biaya administrasi.



**Gambar 3. 1** Ilustrasi Proses Transaksi Keuangan Tersentralisasi  
Sumber : (Eka, 2018)

Sedangkan sifat terdesentralisasi artinya tidak ada yang menjadi penengah atau pihak ketiga yang menjadi perantara. Transaksi dilakukan secara *peer-to-peer* dari pengirim ke penerima. Seluruh transaksi dicatat dalam komputer yang berada di jaringan tersebut, di seluruh dunia, disebut dengan *miner* (penambang yang ikut membantu mengamankan dan mencatat transaksi di jaringan). *Miner* akan mendapatkan komisi dengan uang virtual yang digunakan, namun tidak semua orang bisa menjadi *miner*, karena dibutuhkan keahlian khusus dengan pemrosesan komputasi yang rumit untuk memecahkan kriptografi yang digunakan. Hal ini menjadi salah satu alasan para penambang *cryptocurrency* umumnya menggunakan komputer berspesifikasi tinggi dan khusus.



**Gambar 3. 2** Ilustrasi Proses Keuangan Terdesentralisasi  
 Sumber : (Eka, 2018)

Sifat desentralisasi ini yang menjadi DNA sistem *Blockchain*. Pada dasarnya *Blockchain* menjadi platform yang memungkinkan mata uang digital *Cryptocurrency* dapat digunakan untuk bertransaksi. *Blockchain* secara rinci akan di bahas pada bagian 3.3

### 3.2 Ethereum

Ethereum adalah jaringan rantai blok publik terdistribusi yang terfokus untuk menjalankan kode program yang terdesentralisasi, Ethereum memiliki mata uangnya sendiri yang disebut Ether, Ether juga dikenal sebagai ETH. Ethereum dikatakan sebagai platform terdesentralisasi yang bisa menjalankan *smart contract* yaitu suatu aplikasi yang berjalan sendiri sesuai pemrograman tanpa adanya *down time*, penipuan atau gangguan dari pihak lain, transaksi Ethereum dapat berjalan selama 24 jam tanpa adanya sistem tutup. Ethereum diciptakan oleh Vitalik Buterin pada tahun 2014.

Ethereum adalah platform *blockchain* terbuka untuk membangun aplikasi yang terdesentralisasi (Dapps). Dapps, juga disebut sebagai kontrak pintar atau kode dari kontrak pintar, yaitu program yang menentukan bagaimana nilai bergerak. Segala sesuatu yang memiliki nilai seperti uang, logam, barang dan jasa dapat diuntungkan dari app. Beberapa keunggulan Dapps yaitu penghapusan server dan skalabilitas besar besaran, Dapps sebagai internet tanpa server atau internet pasca server, yang berarti penggunaan aplikasi di internet tidak lagi berdasarkan pada satu sever atau satu perusahaan, Aplikasi didasarkan pada internet itu sendiri (Macedo, 2018).

Dalam akun ethereum memiliki alamat 20-byte dan transisi status menjadi transfer nilai dan informasi langsung antara perhitungan. Akun ethereum memiliki empat bidang di antaranya (Jani, 2018) :

1. *The nonce*, penghitung yang digunakan untuk memastikan bahwa masing masing transaksi yang terjadi hanya dapat di proses satu kali.
2. Saldo ether yang ada pada akun
3. Kode kontrak akun jika ada
4. Penyimpanan akun

Pada dasarnya kelebihan Ethereum dibandingkan dengan *cryptocurrency* lainnya yaitu pada ethereum di desain khusus menjadi *smart contract* yang terbuka. Transaksi ethereum yang dilakukan di *blockchain* dapat mengakses semua *smart contract* dengan berbagai cara, misalnya seperti mengirimkan mata uang digital yang disebut ether atau eth maupun mengirim data ke alamat kontrak. Jika berhasil di eksekusi, maka *smart contract* tersebut dapat memproses transaksi yang lebih banyak atau mengeksekusi *smart contract* lainnya. Contoh dari penggunaan ethereum pada satu kasus yaitu “ Jikan Jono ingin membayar Kipli 10 ETH untuk mengecat rumahnya, dia dapat menggunakan *smart contract* Ethereum. Transaksinya akan terlihat seperti ini ; JIKA Kipli mengecat rumah Jono MAKA 10 ETH akan dikirim ke Kipli. Jadi ETH dapat digunakan sebagai alat pembayaran dengan teknologi *smart contract* dimana berarti Jono tidak akan membayar Kipli sampai Kipli mengecat rumah Jono dan Kipli tidak akan dibayar jika dia tidak mengecat rumah Jono, jadi dalam proses ini tidak bisa menipu”.



**Gambar 3. 3** Logo Ethereum  
Sumber : (ethereum.org, 2020)

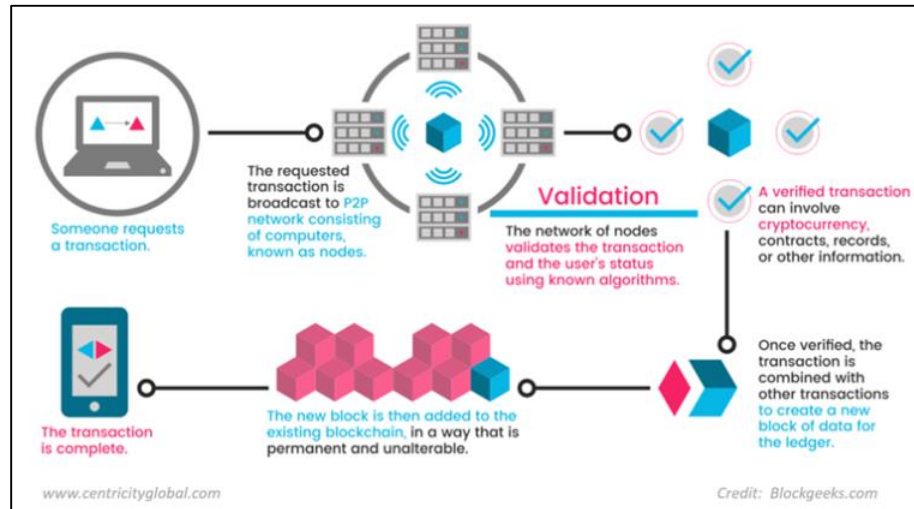
### 3.3 *Blockchain*

*Blockchain* adalah catatan dari suatu transaksi digital berdasarkan strukturnya, dimana catatan individu disebut blok dan di hubungkan dalam satu rantai yang disebut rantai. *Blockchain* digunakan untuk mencatat transaksi yang dilakukan dengan *cryptocurrency* seperti ethereum, bitcoin dan *cryptocurrency* yang lainnya. *Blockchain* merupakan penemuan cerdas gagasan dari seseorang atau sekelompok yang dikenal dengan nama samaran Satoshi Nakamoto dan terus berkembang menjadi sesuatu yang semakin besar.

*Blockchain* adalah teknologi *peer to peer* yang merekam dan memverifikasi transaksi dengan cara desentralisasi. Aplikasi *blockchain* dapat secara global mengelola catatan dalam bentuk apapun yang menjanjikan untuk beberapa bidang, termasuk perdagangan Internasional. (Macedo, 2018)

Aplikasi *blockchain* di kenal sebagai buku besar publik transaksi untuk *cryptocurrency* seperti Bitcoin dan ether. Seperti dalam kasus buku besar lainnya, buku besar rantai kas memberikan catatan asal dan pengalihan kepemilikan aset. Struktur transaksional protokol *blockchain* memfasilitasi tidak hanya transfer mata uang digital, tetapi aset digital lainnya. Aset dapat berwujud seperti kekayaan intelektual, misalnya paten, hak cipta, atau merek. Karena *blockchain* dirancang untuk merekam dan menjaga transaksi, semua *blockchain* secara tradisional memiliki mata uang digital dari beberapa jenis yang terkait dengan mereka sebagai aset paling dasar yang di transaksikan di seluruh jaringan. *Blockchain* sebagai buku besar yang merekam kelompok transaksi atau dikenal sebagai blok, yang di hubungkan bersama secara kriptografis dalam uraian temporal linier. Properti tombol lainnya yang terkait dengan *blockchain*-keamanan, imutabilitas, programabilitas tergantung pada arsitek *blockchain* dan karakter protokol konsensus yang dijalankan oleh *blockchain* itu sendiri. Untuk dapat mengidentifikasi siapa yang memiliki aset tertentu, suatu pihak hanya perlu berkonsultasi dengan buku besar untuk memeriksa siapa pemiliknya yang paling baru (Grech & Camillera, 2017). *Blockchain* ethereum dapat dilihat di etherchain. Etherchain yaitu *explorer* untuk *blockchain* ethereum, disini dapat melihat saldo akun, mencari transaksi dan menjelajah kontrak pintar.

Dalam transaksi *blockchain*, untuk lebih jelasnya akan dipaparkan sebagai berikut :



**Gambar 3. 4** Transaksi Pada *Blockchain*

Sumber : (Zabrocki, 2017)

Pada **Gambar 3.2** menggambarkan transaksi *blockchain* sampai transaksi selesai, pertama, seseorang meminta transaksi, kemudian transaksi yang diminta akan disiarkan ke jaringan P2P dari komputer, dikenal sebagai *node*. Kemudian masuk pada validasi, Transaksi atau *node* divalidasi oleh komunitas, apakah benar terjadi atau tidak (Zainudin, 2018), jaringan *node* melakukan validasi transaksi dan status pengguna menggunakan algoritma yang di kenal, transaksi yang diverifikasi dapat melibatkan mata uang kripto, kontrak, catatan, atau informasi lainnya, setelah diverifikasi, transaksi tersebut digabungkan dengan transaksi lain untuk membuat blok data baru untuk buku besar, blok baru kemudian ditambahkan ke *blockchain* yang ada dengan cara yang permanen dan tidak dapat diubah maka transaksi selesai (Zabrocki, 2017)

Ketika akun para pengguna diaktifkan, pengguna memiliki folder masing-masing pada *dated based system*. Folder ini nantinya sebagai tempat untuk menyimpan data yang berupa file-file dari semua aktivitas pengguna. Jadi folder itu bukan hanya menyimpan rekaman data pemilik akun saja, melainkan juga menyimpan data aktivitas yang dilakukan oleh orang lain dalam jaringan *blockchain*.

Penulis ilustrasikan sebagai berikut agar lebih mudah dipahami, Ketika salah seorang dalam jaringan *blockchain* hendak bertransaksi, maka akan ada pemberitahuan kepada akun lainnya. Masing-masing akun akan memeriksa terkait informasi jumlah nomina yang akan ditransfer, lalu memeriksa kecukupan saldo yang dimiliki oleh pihak yang akan melakukan transfer. Jika saldo pemilik akun mencukupi, maka masing-masing folder pengguna akan mencatat kegiatan transaksi tersebut. Transaksi dikatakan selesai jika data-data transaksi telah selesai dicatat.

Hal ini hanya contoh dari satu kegiatan transaksi. Padahal kita sama-sama tahu jika kebutuhan untuk transfer bukan hanya sekali. Bisa jadi dalam beberapa minggu akan terjadi transaksi berkali-kali. Apalagi jika pemilik akun adalah orang yang aktif di dunia online marketing.

Jadi, setiap kali ada kegiatan transaksi, maka system *blockchain* berjalan. Satu akun yang hendak bertransaksi memberitahukan kepada akun lain, sedangkan akun lain dalam jaringan *blockchain* akan mencatat kegiatan transaksi tersebut.

Hingga pada batasan tertentu, folder-folder itu akan penuh lalu akan dibuat folder baru lagi. sebelum data transaksi tersebut disimpan dalam folder, data tersebut dikunci atau disegel menggunakan kode unik yang akan membuat siapa pun tidak bisa mengubah datanya. Baru kemudian data tersebut disimpan ke folder-folder pengguna dalam jaringan *blockchain*. System penyegelan inilah yang menjadi inti dari metode *blockchain* (NS, 2019).

Contoh penerapan teknologi *Blockchain* di Indonesia adalah menggunakan teknologi *blockchain* Indodax, atau sebelumnya dikenal dengan Bitcoin Indonesia. *Startup* ini mulai dirintis tahun 2013 dan sudah memiliki pengguna aktif sekitar 750.000 orang setiap harinya. *Startup* indodax menyediakan fasilitas yang memungkinkan penggunanya untuk bisa menjual dan membeli uang digital. Produk uang digital yang disediakan antara lain seperti Bitcoin, Ethereum, dan Ripple. Uniknya tidak harus membelinya menggunakan mata uang dollar, melainkan bisa menggunakan Rupiah.

Beberapa variabel yang terdapat pada etherchain diantaranya (Etherchain,2020)

1. *Block Size* : Evolusi ukuran rata-rata blok Ethereum.
2. *Average Block Usage* : Evolusi pemanfaatan rata-rata blok Ethereum.
3. *Block Time* : Rata-rata waktu blokir dari Jaringan Ethereum.
4. *Hash Rate* : Rata-rata hashrate dari Jaringan Ethereum.
5. *Difficulty* : Kesulitan rata-rata harian dari Jaringan Ethereum.
6. *Fee Mining Reward* : Evolusi imbalan penambangan biaya harian.
7. *Mining Revenue* : Evolusi pendapatan harian saat menambang di jaringan Ethereum.
8. *Total Account* : Evolusi jumlah total akun Ethereum.
9. *Transaction* : Jumlah total transaksi per hari.

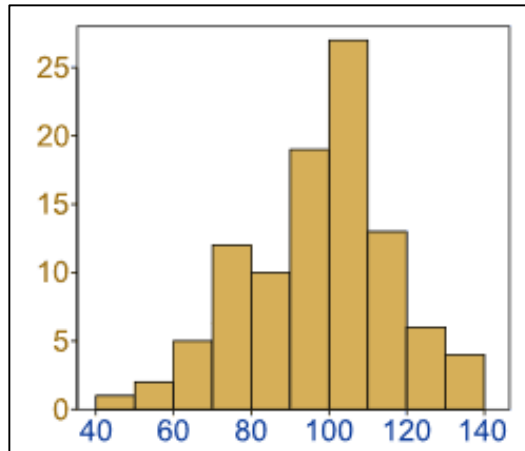
### **3.4 Statistika Deskriptif**

Statistika Deskriptif merupakan suatu metode yang berhubungan dengan mengumpulkan dan menyajikan data sehingga data tersebut mudah dipahami dan dapat memberikan informasi yang berguna. Statistika deskriptif mampu memberikan informasi yang ringkas dari banyaknya data awal sehingga mudah dipahami dan di cerna oleh orang umum. Beberapa penyajian Statistika deskriptif adalah sebagai berikut (Tiyas, 2019)

#### **1. Histogram**

Histogram adalah suatu grafik dari distribusi frekuensi dari sebuah variabel, tampilan histogram umumnya berwujud balok. Penyajian data histogram terdiri atas dua sumbu utama dengan sudut 90 di mana sebagai absis sumbu X serta sebagai ordinat Y. Lebar balok akan menunjukkan suatu jarak dari batas kelas interval, sementara untuk tinggi balok akan menunjukkan besarnya frekuensi dari suatu data.

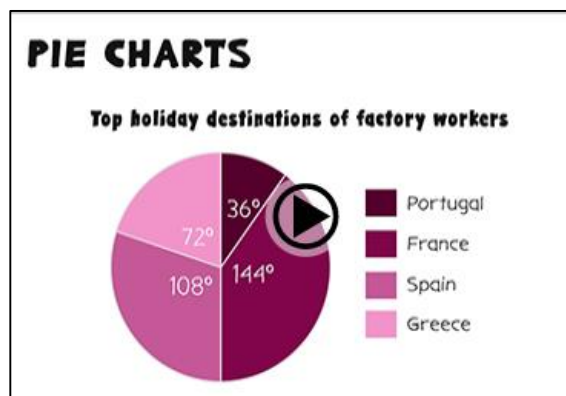




**Gambar 3. 5** contoh Histogram  
 Sumber : (Mathsisfun, 2019)

## 2. Pie Chart

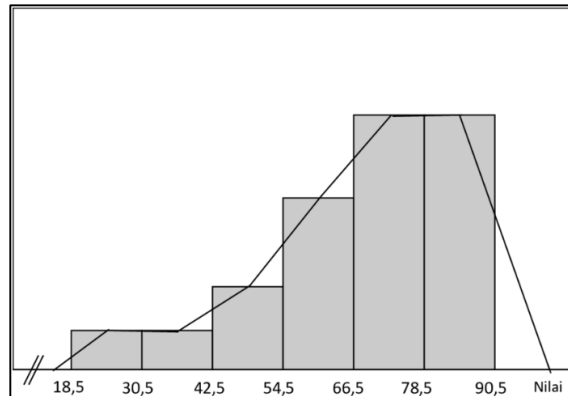
*Pie Chart* atau dalam bahasa Indonesia disebut Diagram kue merupakan suatu lingkaran yang dibagi menjadi beberapa sektor. Pada masing-masing sektor bisa menyatakan besarnya presentase atau bagian untuk tiap-tiap kelompok data.



**Gambar 3. 6** Contoh Pie Chart  
 Sumber : (OpenLearn, 2020)

## 3. Poligon

Poligon adalah suatu grafik dari distribusi frekuensi yang tergolong pada suatu variabel. Tampilan dari poligon pada umumnya berupa garis – garis patah yang didapatkan dengan cara menghubungkan puncak pada masing – masing nilai tengah kelas. Poligon sangat baik dimanfaatkan dalam hal membandingkan bentuk dari dua distribusi.

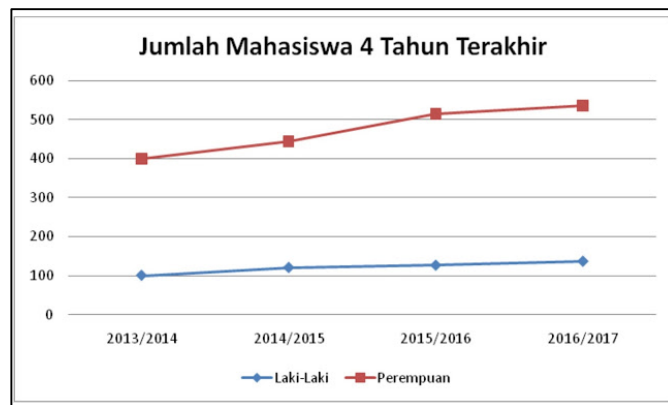


**Gambar 3. 7** Contoh Poligon  
Sumber : (Raswan, 2018)

Grafik lain yang yang paling mudah dalam penyajian data statistika menurut (StatistikaPendidikan, 2017)

#### 4. Grafik Garis

Grafik garis biasanya dibuat untuk menunjukkan perkembangan dari suatu keadaan. Perkembangan tersebut bisa naik, bisa turun, atau mendatar. Hal ini dapat dilihat secara visual melalui garis dalam grafik. Dalam grafik terdapat garis vertikal yang menunjukkan jumlah (frekuensi) dan yang mendatar menunjukkan variabel waktu



**Gambar 3. 8** Contoh Grafik Garis  
Sumber : (StatistikaPendidikan, 2017)

Statistika Deskriptif lain yang mudah dipahami dan biasa digunakan dalam menggambarkan kondisi suata data yaitu menggunakan ukuran pemusatan data, Adapun tiga jenis ukuran pemusatan data (tendensi *central*) yang sering dimanfaatkan, diantaranya yaitu (Tiyas, 2019) :

### 1. *Mean*

*Mean* atau Rata-rata hitung adalah suatu metode yang paling banyak dipakai dalam menunjukkan ukuran tendensi sentral. *Mean* ini dihitung dengan menjumlahkan seluruh nilai data pengamatan lalu dibagi dengan banyaknya data.

### 2. Median

Median adalah nilai yang membagi himpunan pengamatan menjadi dua bagian yang sama besar atau 50% dari pengamatan yang berada di bawah median serta 50% lagi berada di atas median. Median dari  $n$  pengukuran/ pengamatan  $x_1, x_2, \dots, x_n$  merupakan suatu nilai pengamatan yang berada di tengah gugus data sesudah data tersebut diurutkan. Jika banyaknya pengamatan ( $n$ ) ganjil, median berada tepat ditengah gugus data, sementara jika  $n$  genap, median didapatkan dengan cara interpolasi. Yakni cara di mana rata-rata dari dua data yang berada di tengah gugus data.

### 3. Modus

Modus adalah suatu data yang paling sering muncul atau terjadi. Untuk menentukan adanya modus, pertama kali dengan menyusun data dalam urutan meningkat atau sebaliknya. Lalu diikuti dengan menghitung frekuensinya. Nilai yang frekuensinya paling besar atau sering muncul itulah yang dinamakan sebagai modus. Modus dipakai baik untuk tipe data numerik maupun data kategoris.

## 3.5 *Time Series*

*Time Series* (Deret Waktu) adalah serangkaian nilai-nilai variabel yang disusun berdasarkan waktu, Analisis *time series* mempelajari pola gerakan nilai-nilai variabel pada satu interval waktu seperti mingguan, bulanan, tahunan yang teratur. Metode *time series* didasarkan pada asumsi bahwa pola lama akan terulang. Manfaat dari analisis *time series* dapat diperoleh ukuran-ukuran yang dapat digunakan untuk membuat keputusan pada saat ini, untuk peramalan dan untuk merencanakan masa depan (Jayanti, 2013)

### 3.5.1 **Jenis Time Serie berdasarkan pengamatan**

Analisis data *time series* dapat digolongkan berdasarkan banyaknya peubah yang menjadi pengamatan bagi peneliti. Data *time series* yang diambil hanya dari

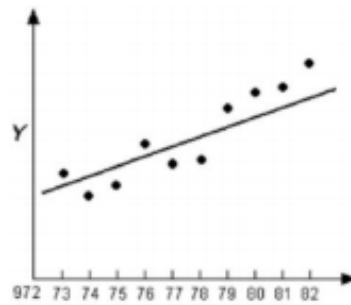
satu peubah pengamatan disebut dengan *time series univariat*. Salah satu contoh data *time series univariat* adalah data nilai tukar dolar terhadap rupiah, selain satu peubah, pengamatan data *time series* juga dapat dilihat dari dua peubah yaitu *time series bivariat*. Analisis *time series bivariat* yaitu suatu analisis *time series* pada suatu peubah tertentu yang hasilnya akan menjadi lebih baik jika melibatkan peubah lainnya, dimana peubah lainnya ini dapat menjelaskan keragaman dari peubah yang menjadi target analisis. Sebagai contoh, pengukuran panjang seorang bayi perperiode waktu yang diduga dipengaruhi oleh berat bayi. Sehingga pada setiap pengamatan akan dicatat panjang serta berat bayi. Kasus yang seperti ini tergolong pada analisis *time series bivariat* yang dapat dianalisis menggunakan model fungsi transfer. Selanjutnya, analisis *time series* dengan *multivariat* dilakukan pada saat menganalisis suatu peubah dalam pengamatan berkaitan dengan peubah lainnya yang tersusun dalam suatu sistem yang saling terkait. Sehingga pengamatan suatu peubah tidak hanya dipengaruhi oleh satu variabel yang akan diteliti tapi juga variabel lain yang berkaitan dengan yang akan diteliti. (Wizza, 2018)

### **3.4.2 Jenis Data *Time Series* Berdasarkan Plot Data**

Data yang masuk dalam jenis *time series* dapat di lihat plot berdasarkan waktu. Hal ini dilakukan untuk mengamati pola dari data untuk selanjutnya menentukan langkah analisis yang akan dilakukan. Berdasarkan bentuk pola yang dibentuk, *data time series* dapat dibagi menjadi empat, pada umumnya, suatu data dapat terdiri atas satu atau beberapa komponen dari empat komponen utama yaitu (Hansun, 2016):

1. *Trend* (T)

*Trend* adalah komponen jangka panjang yang menunjukkan kenaikan atau penurunan dalam data *time series* untuk suatu periode waktu tertentu. Dengan lebih sederhana, dapat dikatakan bahwa *trend* adalah suatu garis atau kurva yang menunjukkan suatu kecenderungan umum dari suatu data runtun waktu. Misalnya kenaikan produksi, inflasi, dan perubahan populasi.

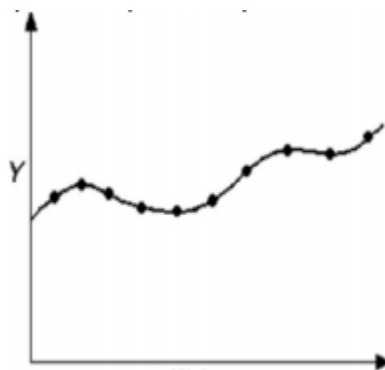


**Gambar 3. 9** Pola Trand

Sumber : (Andini & Auristandi, 2016)

## 2. Siklus (*Cycles/ C*)

Komponen siklus adalah deret yang tidak beraturan berupa fluktuasi gelombang atau siklus dengan durasi waktu yang panjang. Komponen ini biasanya berhubungan dengan siklus bisnis (*business cycle*), suatu gerakan dianggap sebagai siklus apabila timbul kembali setelah jangka waktu lebih dari satu tahun. Contoh data runtun waktu dengan komponen siklus adalah data kondisi perekonomian yang perubahannya lebih dari satu tahun.

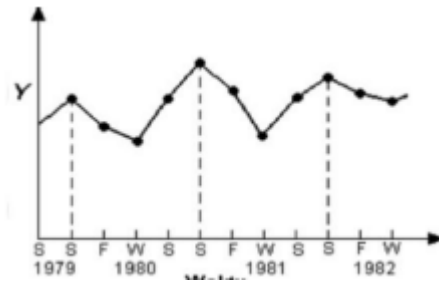


**Gambar 3. 10** Pola Siklus

Sumber : (Andini & Auristandi, 2016)

## 3. Musiman (*Seasonality/ S*)

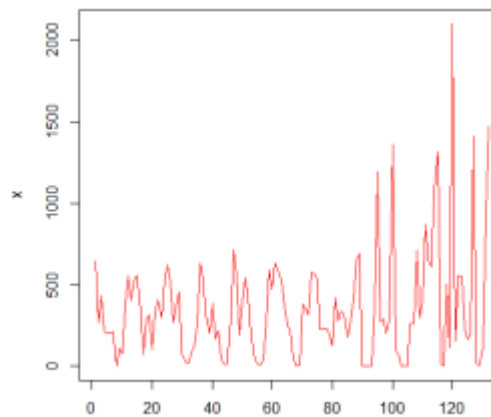
Komponen musiman adalah suatu pola fluktuasi permintaan (*demand*) di atas atau di bawah garis *trend* yang terjadi tiap tahunnya. Fluktuasi musiman yang dimaksud dapat diklasifikasikan secara kuartal, bulanan, mingguan, atau harian, dan mengarah pada pola yang berubah secara regular dalam suatu waktu, misalnya kenaikan harga bahan-bahan pokok yang terjadi menjelang Hari Raya Idul Fitri tiap tahun, pola jumlah penumpang yang padat di waktu tertentu dan sebagainya.



**Gambar 3. 11** Pola Musiman  
 Sumber : (Andini & Auristandi, 2016)

#### 4. *Irregular (I)*

Komponen irregular adalah gerakan fluktuasi yang diakibatkan oleh kejadian yang tidak dapat diprediksi atau kejadian non-periodik, seperti terjadinya perang, bencana alam, dan hal lainnya.



**Gambar 3. 12** Pola *Irregular*  
 Sumber : (Darmawan & dkk, 2016)

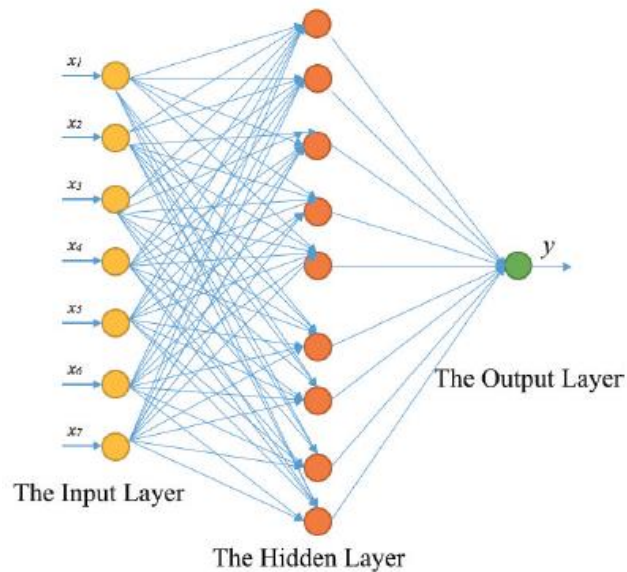
### 3.6 *Artificial Neural Network*

*Artificial Neural Networks* (ANN) atau Jaringan Saraf Tiruan (JST) merupakan suatu model kecerdasan yang diilhami dari struktur otak manusia yang kemudian di implementasikan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran berlangsung. ANN yaitu suatu model penalaran yang didasarkan pada otak manusia. ANN terdiri dari sejumlah prosesor sangat sederhana dan saling berhubungan yang disebut neuron (Kurniawansyah, 2018). Neuron memiliki karakteristik yang sama dalam ANN, terdiri dalam kelompok kelompok yang di sebut layer. Neuron neuron yang ada dalam satu layer terhubung dalam layer layer

lainnya yang berdekatan. Kekuatan hubungan antara neuron yang berdekatan dipresentasikan dalam kekuatan hubungan atau bobot. (Dharma, Putera, & Ardana, 2011).

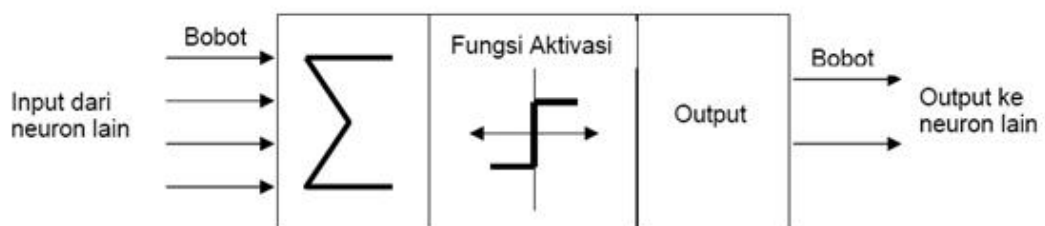
Langkah penting dalam pengembangan sebuah model ANN yaitu penentuan bobot matrik melalui pelatihan atau *training*. Ada dua tipe mekanisme *training* yaitu *supervised training* dan *unsupervised training*, *Supervised training* memerlukan supervisi dari luar untuk memandu proses *training*. Algoritma ini menggunakan sejumlah pasangan data *input-output* yang dipergunakan sebagai contoh, dimana data yang dipergunakan sebagai contoh sebaiknya menggunakan data yang sudah diketahui kebenarannya. *Output* dari jaringan lalu dibandingkan dengan data *output* yang diharapkan (*output* contoh) untuk mendapatkan selisih dari *output* perkiraan dengan *output* sebenarnya. Selisih inilah yang dipergunakan untuk mengubah bobot jaringan sehingga diperoleh *output* yang sama atau *output* yang mendekati target. Mekanisme sebuah *supervised training* disebut algoritma *back-propagation training* pada umumnya digunakan dalam aplikasi-aplikasi *engineering*. Karena pada ANN tidak mempertimbangkan fisik dari permasalahan, ANN merupakan sebuah model *blackbox* namun dapat mendeteksi proses fisik dalam model ANN yang sudah *training*. (Dharma, Putera, & Ardana, 2011).

Struktur ANN terdiri dari *input layer*, *hidden layer* dan *output layer*. Suatu informasi ( $\alpha$ ) diterima *input layer* menggunakan bobot kedatangan ( $w$ ) tertentu. Selanjutnya akan dilakukan penjumlahan bobot pada *hidden layer*. Kemudian hasil dari penjumlahan dibandingkan dengan nilai ambang (*threshold*). Jika nilai melewati ambang batas, maka akan diteruskan ke *output layer*, sedangkan jika nilai tidak melewati ambang batas, maka tidak akan diteruskan ke *output layer* (Habibi & Riksakomara, 2017). Gambar 3.5 menunjukkan struktur jaringan dari ANN



**Gambar 3. 13** Jaringan ANN  
 Sumber : (Prathama & dkk, 2017)

Layer *input* terdiri dari neuron-neuron yang menerima sebuah *input* dari lingkungan luar, *Input* disini dapat berupa penggambaran dari suatu masalah. Layer tersembunyi (*hidden layer*) terdiri dari neuron-neuron yang menerima masukan dari *input* layer dan kemudian membawa *output* ke layer berikutnya. Lapisan *output* disebut unit-unit *output*, terdiri dari neuron-neuron yang menerima *output* dari *hidden layer* dan mengirimkannya kepada pemakai (Dharma, Putera, & Ardana, 2011)



**Gambar 3. 14** Struktur ANN dasar  
 Sumber : (Suhartono, 2012)

Gambar 3.6 menunjukkan struktur ANN dasar dimana setelah masuk ke dalam neuron, nilai *input* yang ada akan dijumlahkan oleh suatu fungsi perambatan (*summing function*), yang bisa dilihat seperti pada dengan lambang sigma ( $\Sigma$ ). Hasil penjumlahan akan diproses oleh fungsi aktivasi setiap neuron, disini akan dibandingkan hasil penjumlahan dengan *threshold* (nilai ambang) tertentu. Jika

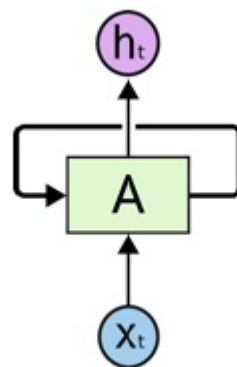


nilai melebihi *threshold*, maka aktivasi neuron akan dibatalkan, sebaliknya, jika masih dibawah nilai *threshold*, neuron akan diaktifkan. Setelah aktif, neuron akan mengirimkan nilai *output* melalui bobot-bobot outputnya ke semua neuron yang berhubungan dengannya. Proses ini akan terus berulang pada *input-input* selanjutnya. (Suhartono, 2012)

### 3.7 *Recurrent neural network*

Pada umumnya, dalam kehidupan sehari hari manusia tidak membuat keputusan secara tunggal setiap saat. Manusia akan memperhitungkan masa lalu dalam membuat sebuah keputusan. Cara berpikir seperti ini yang menjadi dasar dari pengembangan *Recurrent neural network*. Sama seperti analogi tersebut, RNN tidak membuang begitu saja informasi dari masa lalu dalam proses pembelajarannya. Hal inilah yang membedakan RNN dari *Artificial Neural Network* biasa (Lapian, Osmond, & Saputra, 2018)

*Recurrent Neural Network* (RNN) pertama dikembangkan oleh Jeff Elman pada tahun 1990. RNN merupakan variasi dari *Artificial Neural Network* (ANN) yang di rancang khusus untuk memproses data yang bersambung atau berurutan. RNN biasa digunakan untuk menyelesaikan masalah dengan data *time series*.

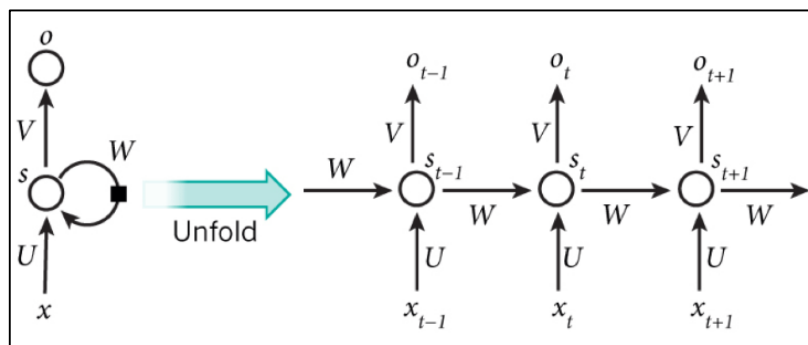


**Gambar 3. 15** Struktur umum RNN

Sumber : (colah, 2015)

Jika struktur pada gambar 3.1 di uraikan, maka struktur dari RNN akan lebar tengahnya selebar panjang pola data yang ingin di pelajari oleh RNN karena inilah dikatakan RNN di desain khusus untuk *handle* data berurutan. *Input*  $x_t$  menghasilkan *output*  $h_t$ .

Pada model RNN sinyal dapat mengalir secara *forward* dan *backward* secara berulang. Untuk bisa melakukan hal tersebut, maka ditambahkan sebuah layer baru yang disebut dengan *context layer*. Selain melewati *input* antar layer, *output* dari setiap layer juga menuju ke *context layer* untuk digunakan sebagai *input* pada *timestep* berikutnya. RNN menyimpan informasi di *contextlayer*, yang membuatnya dapat mempelajari urutan data dan menghasilkan *output* atau urutan lain. Jika ditarik kesimpulan maka dapat dikatakan bahwa RNN memiliki memori yang berisikan hasil rekaman informasi yang dihasilkan sebelumnya (Juanda, Jondri, & Rohmawati, 2018).



**Gambar 3. 16** Proses RNN  
Sumber : (R A. Y., 2018)

Diagram pada gambar 3.16 yaitu membuka gulungan RNN maka simpelnya kita menuliskan seluruh jaringan dengan urutan (*sequence*) secara lengkap. Berikut ialah keterangan simbol yang ada (Juanda, Jondri, & Rohmawati, 2018) :

1.  $X_t$  ialah *input* pada setiap *time step*.
2.  $S_t$  adalah *hidden state* pada setiap *time step*  $t$
3.  $O_t$  adalah *output* untuk setiap *step*  $t$

Pemetaan satu simpul  $S_t$  dan *output*  $O_t$  dapat ditulis sebagai (Tian & dkk, 2018):

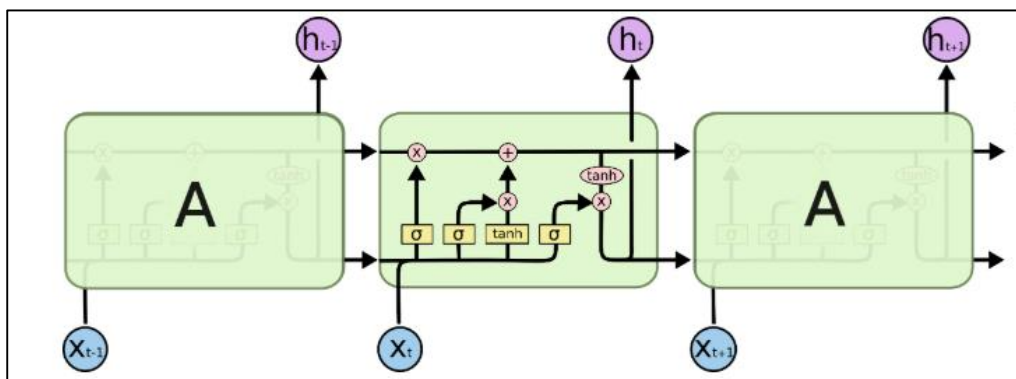
$$S_t = f((U * X_t) + (W * S_{t-1})) \quad (3.1)$$

$$O_t = g(V * S_t) \quad (3.2)$$

di mana  $S_t$  dikatakan memori jaringan pada waktu  $t$ ,  $U$ ,  $W$ , dan  $V$  adalah matriks bobot berbagi di setiap lapisan;  $X_t$  dan  $O_t$  mewakili *input* dan *output* pada waktu  $t$ ; dan  $f(.)$  dan  $g(.)$  mewakili fungsi nonlinear.

### 3.8 Long Short Term Memory

Long Short Term Memory (LSTM) merupakan pengembangan dari *Recurrent Neural Network* (RNN) dengan mengatasi salah satu kekurangan RNN yaitu kemampuan pengelolaan informasi dalam periode yang lama. Diusulkan oleh Sepp Hochreiter dan Jurgen Schmidhuber pada tahun 1997, LSTM banyak dipilih untuk prediksi berbasis waktu atau *time-series* karena dikenal lebih unggul dan handal dalam melakukan prediksi dalam waktu lama dibanding algoritma lain (Zahara, Sugianto, & Ilmiddafiq, 2019). Lstm memiliki struktur jaringan seperti gambar 3.17 berikut ini :



**Gambar 3. 17** Jaringan LSTM  
Sumber : (colah, 2015)

Model LSTM menyaring informasi melalui struktur gerbang untuk mempertahankan dan memperbarui keadaan sel memori. Struktur pintunya mencakup *input gate*, *forget gate*, dan *output gate*. Setiap sel memori memiliki tiga lapisan sigmoid dan satu lapisan tanh (Qiu, Wang, & Zhou, 2020)

Fungsi sigmoid ditunjukkan pada persamaan (3.3) (Ma, 2015):

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (3.3)$$

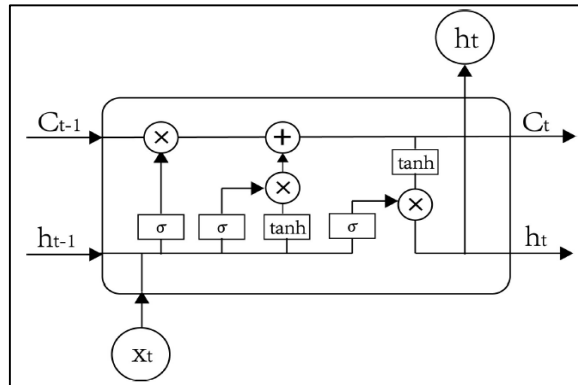
Fungsi tanh adalah sebagai berikut :

$$\tanh(x) = 2\sigma(2x) - 1 \quad (3.4)$$

Dimana:

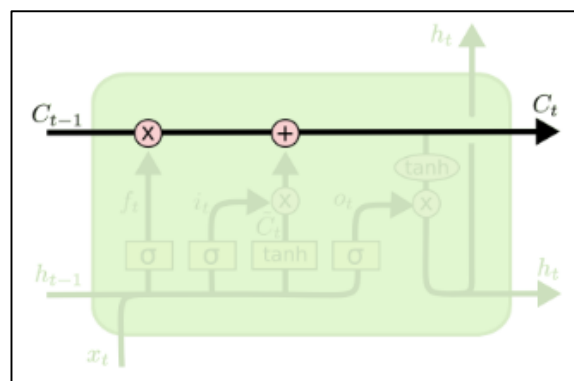
$\sigma$  = Fungsi aktivasi sigmoid

$x$  = Data *input*



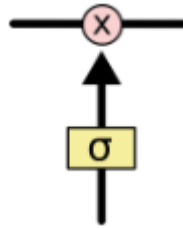
**Gambar 3. 18** Struktur LSTM  
 Sumber : (Qiu, Wang, & Zhou, 2020)

Gambar 3.18 memperlihatkan struktur dalam satu sel LSTM, kunci dari LSTM yaitu adanya jalur yang menghubungkan *memory cell* lama ( $C_{t-1}$ ) ke *memory cell* baru ( $C_t$ ). *Memory cell* adalah garis horizontal yang menghubungkan semua *output layer* pada LSTM. Dengan adanya jalur tersebut, suatu nilai *memory cell* yang lama akan dengan mudah diteruskan ke *memory cell* yang baru dengan sedikit sekali modifikasi.



**Gambar 3. 19** *memory cell*  
 Sumber : (colah, 2015)

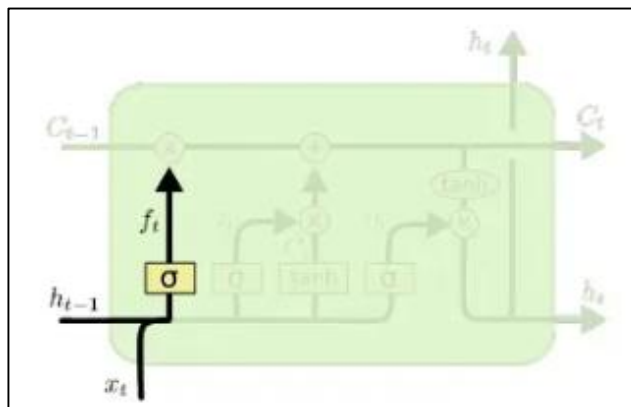
LSTM memiliki kemampuan untuk menambahkan atau menghapus informasi sebelumnya yang masuk ke sel saat ini, Lapisan sigmoid menampilkan angka antara nol dan satu, menggambarkan berapa banyak masing-masing komponen harus dibiarkan masuk. Nilai nol diartikan sebagai "jangan biarkan apa pun lewat," sedangkan nilai satu berarti "biarkan semuanya lewat!" (colah, 2015)



**Gambar 3. 20** Lapisan Sigmoid  
 Sumber : (colah, 2015)

Dari (colah, 2015) dijelaskan secara lebih rinci langkah kerja dari LSTM adalah sebagai berikut :

Langkah pertama dalam LSTM yaitu memutuskan informasi apa yang akan di buang dari kondisi sel. Keputusan ini dibuat oleh lapisan sigmoid yang disebut "*forget gate*" Ini terlihat pada  $h_{t-1}$  dan  $x_t$ , dan menghasilkan angka antara 0 dan 1. Jika *output* yang dihasilkan sigmoid lebih dari 0,5, maka dapat mengklasifikasikan hasil sebagai 1 dan jika kurang dari 0,5, dapat mengklasifikasikannya sebagai 0.



**Gambar 3. 21** Alur Informasi pada *forget gate*  
 Sumber : (colah, 2015)

*Forget gate* dalam unit LSTM menentukan informasi status sel mana yang dibuang dari model. Seperti yang ditunjukkan pada **Gambar 3.21**, sel memori menerima *output*  $h_{t-1}$  dari momen sebelumnya dan informasi *eksternal*  $x_t$  dari momen saat ini sebagai *input* dan menggabungkannya dalam vektor panjang  $[h_{t-1}, x_t]$  melalui transformasi  $\sigma$  menjadi:

$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3.5)$$

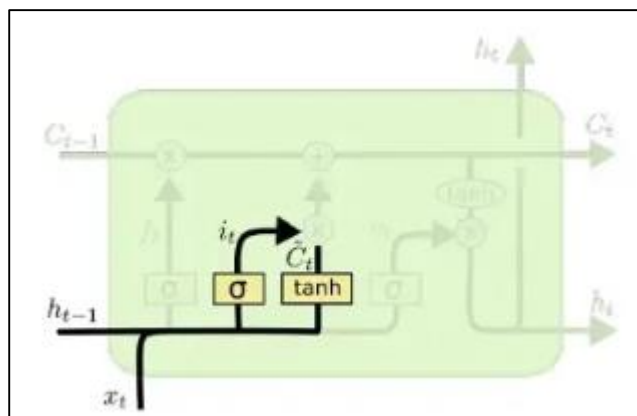
dimana :

$f_t$  : *Forget gate*

$\sigma$  : Fungsi sigmoid

- $W_f$  : Nilai *weight* untuk *forget gate*
- $h_{t-1}$  : Nilai *output* sebelum orde ke t
- $x_t$  : Nilai *input* pada orde ke t
- $b_f$  : Nilai bias pada *forget gate*

$W_f$  dan  $b_f$  adalah bobot dan bias dari *forget gate* sedangkan  $\sigma$  adalah fungsi sigmoid. Fungsi utama dari *forget gate* adalah untuk merekam seberapa banyak status sel  $C_{t-1}$  dari waktu sebelumnya dicadangkan ke status sel  $C_t$  dari waktu saat ini. Gerbang akan menampilkan nilai antara 0 dan 1 berdasarkan pada  $h_{t-1}$  dan  $x_t$ , nilai 1 menunjukkan reservasi lengkap dan 0 menunjukkan pembuangan lengkap. Notasi  $[h_{t-1}, X_t]$  merupakan operasi konkatenasi artinya menambahkan baris dari  $X_t$  dengan baris dari  $h_{t-1}$ .



**Gambar 3. 22** Alur informasi yang melewati *input gate*  
 Sumber : (colah, 2015)

Langkah selanjutnya adalah memutuskan informasi baru apa yang akan di simpan di keadaan sel. *Input gate* memiliki dua fungsi, pertama adalah menemukan keadaan sel yang harus diperbarui nilai yang akan diperbarui dipilih oleh lapisan sigmoid, Sedangkan fungsi lainnya adalah vektor kandidat baru  $\hat{C}_t$  dibuat melalui lapisan *tanh* untuk mengontrol berapa banyak informasi baru ditambahkan, seperti dalam Persamaan (3.6). Dan Persamaan (3.7) digunakan untuk memperbarui keadaan sel dari sel memori:

$$i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.6}$$

$$\hat{C}_t = \tanh (W_i \cdot [h_{t-1}, x_t] + b_c) \tag{3.7}$$

dimana :

$i_t$  : *Input gate*

$\sigma$  : Fungsi sigmoid

$W_i$  : Nilai *weight* untuk *input gate*

$h_{t-1}$  : Nilai *output* sebelum orde ke t

$x_t$  : Nilai *input* pada orde ke t

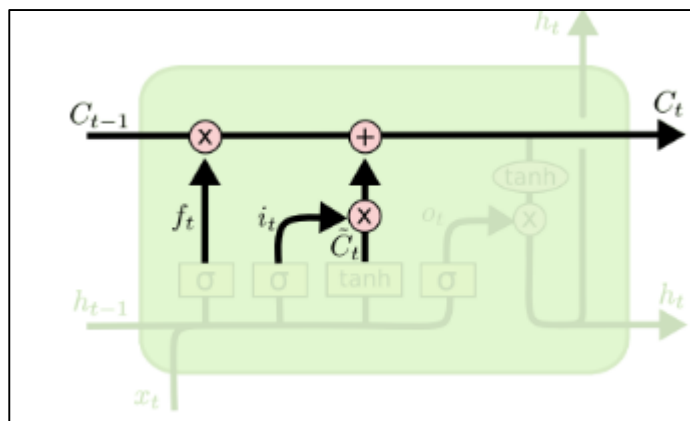
$b_i$  : Nilai bias pada *input*

$\hat{C}_t$  : Nilai baru yang dapat ditambahkan ke *cell state*

$b_c$  : Nilai bias pada *cell state*

$C_{t-1}$  : *Cell state* sebelum orde ke t

$f_t$  : *Forget gate*



**Gambar 3. 23** memperbarui status *cell*

Sumber : (colah, 2015)

Gambar 2.23 yaitu proses memperbarui status sel lama  $C_{t-1}$ , ke dalam status sel baru  $C_t$ . Prosesnya yaitu dengan mengalikan state lama dengan  $f_t$ , menghapus hal yang di hapus sebelumnya, kemudian tambhkan operasi  $i_t * \hat{C}_t$ , ini merupakan nilai kandidat baru yang diskalakan dengan seberapa banyak ingin memperbarui setiap nilai *state*, dapat dirumuskan menggunakan persamaan 3.8 berikut :

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \quad (3.8)$$

dimana :

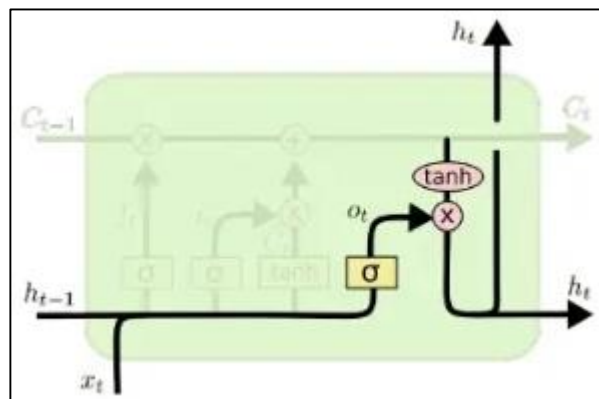
$C_t$  : *Cell state* orde ke t

$f_t$  : *Forget gate*

$C_{t-1}$  : *Cell state* sebelum orde ke t

$i_t$  : *Input gate*

$C_{t-1}$  : *Cell state* sebelum orde ke t



**Gambar 3. 24** Alur informasi melewati *output gate*

Sumber : (colah, 2015)

*Output gate* mengontrol seberapa banyak keadaan sel saat ini dibuang. Informasi keluaran pertama-tama ditentukan oleh lapisan sigmoid, dan kemudian keadaan sel diproses oleh tanh dan dikalikan dengan keluaran lapisan sigmoid untuk mendapatkan bagian keluaran akhir:

$$O_t = \sigma (W_0 \cdot [h_{t-1}, x_t] + b_o) \quad (3.9)$$

dimana :

$O_t$  : *Output gate*

$\sigma$  : Fungsi sigmoid

$W_0$  : Nilai *weight* untuk *output gate*

$h_{t-1}$  : Nilai *output* sebelum orde ke t

$x_t$  : Nilai *input* pada orde ke t

$b_o$  : Nilai bias pada *output gate*

Nilai *output* akhir sel didefinisikan sebagai:

$$h_t = O_t * \tanh(C_t) \quad (3.10)$$



Dimana :

$h_t$  : Nilai *output* orde ke t

$O_t$  : *Output gate*

$\tanh$  : Fungsi tanh

$C_t$  : *Cell state*

### 3.9 Adaptive Moment Estimation (Adam)

Algoritma optimisasi bertujuan untuk menemukan bobot optimal, meminimalkan kesalahan atau *error* dan memaksimalkan akurasi. Selama proses pelatihan, parameter (bobot) model diubah untuk mencoba dan meminimalkan fungsi *error*, agar dapat memprediksi seakurat mungkin. Namun bagaimana cara yang tepat, kapan, dan berapa banyaknya perubahan itu masih belum bisa dipastikan, maka disinilah pengoptimal masuk, Mereka menyatukan fungsi *error* dan parameter model dengan memperbarui model dalam menanggapi *output* dari fungsi kerugian. Dalam istilah yang lebih sederhana, pengoptimalisasi membentuk model yang kita punya ke dalam bentuk yang paling akurat dengan memanfaatkan bobotnya (Donges, 2018). Adapun optimasi yang di pakai dalam penelitian ini adalah optimasi Adam.

*Adaptive Moment Estimation* (Adam) adalah algoritma yang populer di bidang *deep learning* karena mencapai hasil yang baik dengan cepat. Optimasi Adam merupakan algoritma yang dikembangkan dengan memanfaatkan kelebihan dari algoritma *Adaptive Gradient* (AdaGrad) dan *Root Mean Square Propagation* (RMSProp). Adam tidak hanya mengadaptasi tingkat pembelajaran parameter berdasarkan rata rata pertama (*mean*) seperti pada RMSProp, tetapi juga menggunakan rata rata kedua dari gradien (*varians uncentered*). Algoritma digunakan untuk menghitung rata-rata pergerakan eksponensial dari gradien dan gradien kuadratnya, dan pada parameter  $\beta_1$ ,  $\beta_2$  untuk mengontrol tingkat peluruhan rata-rata pergerakan.

Nilai parameter yang direkomendasikan adalah  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ , dan  $\epsilon = 10^{-8}$  dengan  $\beta_1 = \beta_2 =$  tingkat penurunan eksponensial dan  $\epsilon =$  nilai epsilon untuk *update* parameter (Kingma & Ba, 2015).  $\beta_1$  merupakan *Exponential decay rate*

untuk estimasi momen pertama,  $\beta_2$  merupakan *the exponential decay rate* untuk estimasi *second-moment*, Nilai ini harus diset mendekati 1.0 pada masalah dengan *sparse gradien*,  $\epsilon$  Merupakan angka yang sangat kecil untuk mencegah pembagian dengan nol saat implementasi (onnocenter, 2019).

### 3.10 Parameter Evaluasi

Menurut Carlo Vercilis (Varcellis, 2009) terdapat dua alasan utama untuk dapat melihat tingkat akurasi pada suatu prediksi dengan model *time series*. Pertama, pada tahap pengembangan dan identifikasi model, ukuran akurasi digunakan dalam membandingkan antar model alternatif yang satu dengan lainnya dan untuk menentukan nilai parameter yang muncul dalam fungsi prediksi. Dalam identifikasi model prediksi yang paling akurat, masing masing model di anggap diterapkan pada data masa lalu, dan nilai model dengan tingkat *error* terendah atau minimum yang dipilih.

Kedua, setelah dilakukan pengembangan model prediksi dan digunakan untuk menghasilkan prediksi untuk masa mendatang, perlu secara berkala untuk menilai keakuratan, untuk mendeteksi kelainan dan kekurangan dalam model yang mungkin timbul di lain waktu. Evaluasi keakuratan prediksi pada tahap ini membuat mungkin dalam menentukan apakah model masih akurat atau membutuhkan suatu revisi. Dalam mengevaluasi akurasi dan peramalan kinerja model berbeda, maka penelitian ini mengadopsi dua indeks evaluasi yaitu *Mean Square Error* (MSE) atau *Root Mean Square Error* (RMSE), dan Mean Absolute Percentage Error (MAPE). Formula untuk menghitung indeks ini adalah sebagai berikut (Budiman, 2016) :

#### 3.7.1 Mean Square Error (MSE)

*Mean Square Error* (MSE ) adalah penjumlahan kuadrat eror atau selisih antara nilai sebenarnya (aktual) dan nilai prediksi, kemudian membagi jumlah tersebut dengan banyaknya waktu data peramalan, dapat dirumuskan sebagai berikut:

$$MSE = \frac{\sum_{i=1}^n (Y_i - \hat{Y}_i)^2}{n} \quad (3.11)$$

$n$  : Jumlah data

$\hat{Y}_i$  : Nilai data prediksi

$Y_i$  : Nilai data sebenarnya

### 3.7.2 Root Mean Square Error (RMSE)

*Root Mean Square Error* (RMSE) adalah penjumlahan dari kuadrat error atau selisih antara nilai sebenarnya dan nilai prediksi dan membagi hasil penjumlahan yang diperoleh dengan banyaknya waktu peramalan dan menarik akarnya. Dapat dirumuskan sebagai berikut :

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (Y_i - \hat{Y}_i)^2}{n}} \quad (3.12)$$

dimana :

$n$  : Jumlah data

$\hat{Y}_i$  : Nilai data prediksi

$Y_i$  : Nilai data sebenarnya

### 1.7.3 Mean Absolute Percentage Error (MAPE)

*Mean Absolute Percentage Error* (MAPE) adalah nilai absolute dari persentase error data terhadap mean. Persamaannya di tulis sebagai berikut :

$$MAPE = \frac{\sum_{i=1}^n |\hat{Y}_i - Y_i|}{n} \times 100 \% \quad (3.13)$$

dimana :

$n$  : Jumlah data

$\hat{Y}_i$  : Nilai data prediksi

$Y_i$  : Nilai data sebenarnya

## 1.8 Sistem Pengerjaan LSTM

Dalam membangun sistem pada metode LSTM dilakukan dengan beberapa langkah, diantaranya *pre-processing* data, inisialisasi parameter, *training* LSTM network, dan juga melakukan uji terhadap data *testing*. Dalam sistem yang dibangun, dataset yang didapatkan terlebih dahulu dioalah dengan menggunakan teknik normalisasi *min max scaling*. Dilakukan inisialisasi pada setiap parameter, setelah itu dilakukan *training* pada jaringan yang dibuat sesuai dengan parameter

yang telah ditentukan. Selanjutnya dilakukan uji pada model yang telah didapatkan dari proses *training* terhadap data *testing*. Proses tersebut terus diulang hingga mendapatkan model dengan akurasi yang dapat diterima. Langkah lebih jelasnya adalah sebagai berikut : (Aldi, Jondri, & Aditsania, 2018)

### 3.8.1 Preprocessing Data

Dalam meminimalkan *error*, dilakukan normalisasi pada dataset dengan mengubah data aktual menjadi nilai dengan *range* interval [0,1]. Teknik yang digunakan pada proses normalisasi yaitu *min-max scaling*. Adapun untuk rumus normalisasi *min-max scaling* adalah :

$$X_{sc} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (3.14)$$

dimana :

X : Data yang akan dinormalisasikan

$X_{sc}$  : Data setelah dinormalisasikan

$X_{min}$  : Nilai minimum dari keseluruhan data

$X_{max}$  : Nilai maksimum dari keseluruhan data

### 3.8.2 Inisialisasi Parameter

Setelah dilakukan *preprocessing* pada dataset, selanjutnya menentukan inisialisasi parameter-parameter dasar yang dibutuhkan, antara lain :

- a) Jumlah *Hidden layer*
- b) Jumlah Neuron pada *Hidden Layer*
- c) Target *error* yang berupa *Mean Square Error* (MSE)
- d) *Epoch* Maksimum

### 3.8.3 Training LSTM Network

Berikutnya adalah penjelasan dari proses *training* pada jaringan LSTM Network :

1. Hitung semua fungsi *gates* unit pada setiap neuron. Dengan berurut fungsi *gates* yang akan dihitung adalah *forget gates* dengan persamaan (3.5), fungsi *input gates* dengan persamaan (3.6) dan (3.7), fungsi *cell gates* dengan persamaan (3.8), dan yang terakhir fungsi *output gates* dengan persamaan (3.9) dan (3.10)

2. Jika telah melakukan perulangan sebanyak *epoch* yang telah ditentukan, maka berhenti. Jika belum, akan dilakukan optimasi dengan optimasi Adam dan memperbarui bobot dan bias pada sistem, kemudian kembali ke langkah dua.

### 3.8.4 *Testing*

Model yang telah didapatkan pada proses *training* akan diuji dengan menggunakan data *testing* yang telah didapat dari *preprocessing* data, dengan metode akurasi yang digunakan menggunakan MSE, RMSE dan MAPE dengan persamaan(3.11), (3.12) dan (3.13)

### 3.8.5 Denormalisasi Data

Data yang pada awalnya sudah dilakukan normalisasi, maka akan di kembalikan ke data sebenarnya untuk mengetahui hasil prediksi sebenarnya, Rumus untuk denormalisasi dalam range [0, 1] adalah sebagai berikut :

$$X_t = y (X_{\max} - X_{\min}) + X_{\min} \quad (3.15)$$

dimana :

$X_t$  = Nilai data normal

$y$  = Hasil *Output* Jaringan

$X_{\max}$  = Nilai maksimum data aktual keseluruhan.

$X_{\min}$  = Nilai minimum data aktual keseluruhan

## BAB 4 METODOLOGI PENELITIAN

### 4.1 Populasi dan Sampel

Populasi dari penelitian ini adalah seluruh data harga Ethereum dan informasi *blockchain* Ethereum yang dilihat dari Etherchain. Sedangkan yang menjadi sampel dalam penelitian adalah data harga Ethereum /USD, *Block Size*, *Average Block*, *Block Time*, *Hash Rate*, *Difficulty*, *Fee Mining Reward*, *Mining Revenue*, *Total Account* dan *Transaction*. Masing masing variabel di ambil data dari 8 Januari 2018 sampai 7 Januari 2020 setiap hari selama 2 tahun.

### 4.2 Sumber Data

Jenis data yang di gunakan dalam penelitian ini adalah data Sekunder. Data tersebut di peroleh dari beberapa *website* yaitu “etherchain.org” untuk mendapatkan informasi data dari *blockchain* Ethereum serta pengambilan data dari website coindesk.com dan investing.com untuk mendapatkan data harga Ethereum selama 2 tahun setiap harinya.

### 4.3 Variabel Penelitian

Variabel yang penulis gunakan dalam penelitian ini terdiri dari 10 variabel yang dapat dilihat pada tabel 4.1 berikut

**Tabel 4.1. Variabel Penelitian**

No	Variabel	Definisi
1	Ethereum_USD (Y)	Harga Ethereum dalam <i>United States Dollar</i> (USD)
2	Ethereum_USD (X <sub>1</sub> )	Harga Ethereum hari sebelumnya dalam <i>United States Dollar</i> (USD)
3	<i>Block Size</i> (X <sub>2</sub> )	Evolusi ukuran rata-rata blok Ethereum.
4	<i>Average Block Usage</i> (X <sub>3</sub> )	Evolusi pemanfaatan rata-rata blok Ethereum.
5	<i>Block Time</i> (X <sub>4</sub> )	Rata-rata waktu blok yang diperlukan Jaringan Ethereum.

No	Variabel	Definisi
6	<i>Hash Rate</i> ( $X_5$ )	Rata-rata <i>hashrate</i> dari Jaringan Ethereum.
7	<i>Difficulty</i> ( $X_6$ )	Kesulitan rata-rata harian dari Jaringan Ethereum.
8	<i>Fee Mining Reward</i> ( $X_7$ )	Evolusi imbalan penambangan biaya harian.
9	<i>Mining Revenue</i> ( $X_8$ )	Evolusi pendapatan harian saat menambang di jaringan Ethereum.
10	<i>Total Account</i> ( $X_9$ )	Evolusi jumlah total akun Ethereum.
11	<i>Transaction</i> ( $X_{10}$ )	Jumlah total transaksi per hari.

#### 4.4 Metode Analisis Data

Berikut merupakan uraian langkah dari penelitian ini :

1. *Preprocessing*

Dilakukan normalisasi pada dataset dengan mengubah data aktual menjadi nilai dengan *range* interval [0,1]. Teknik yang digunakan pada proses normalisasi yaitu *min-max scaling* dengan rumus pada persamaan 3.14.

2. Analisis Deskriptif data harian Ethereum

3. Melakukan analisis Multivariat LSTM dengan tahapan berikut ini :

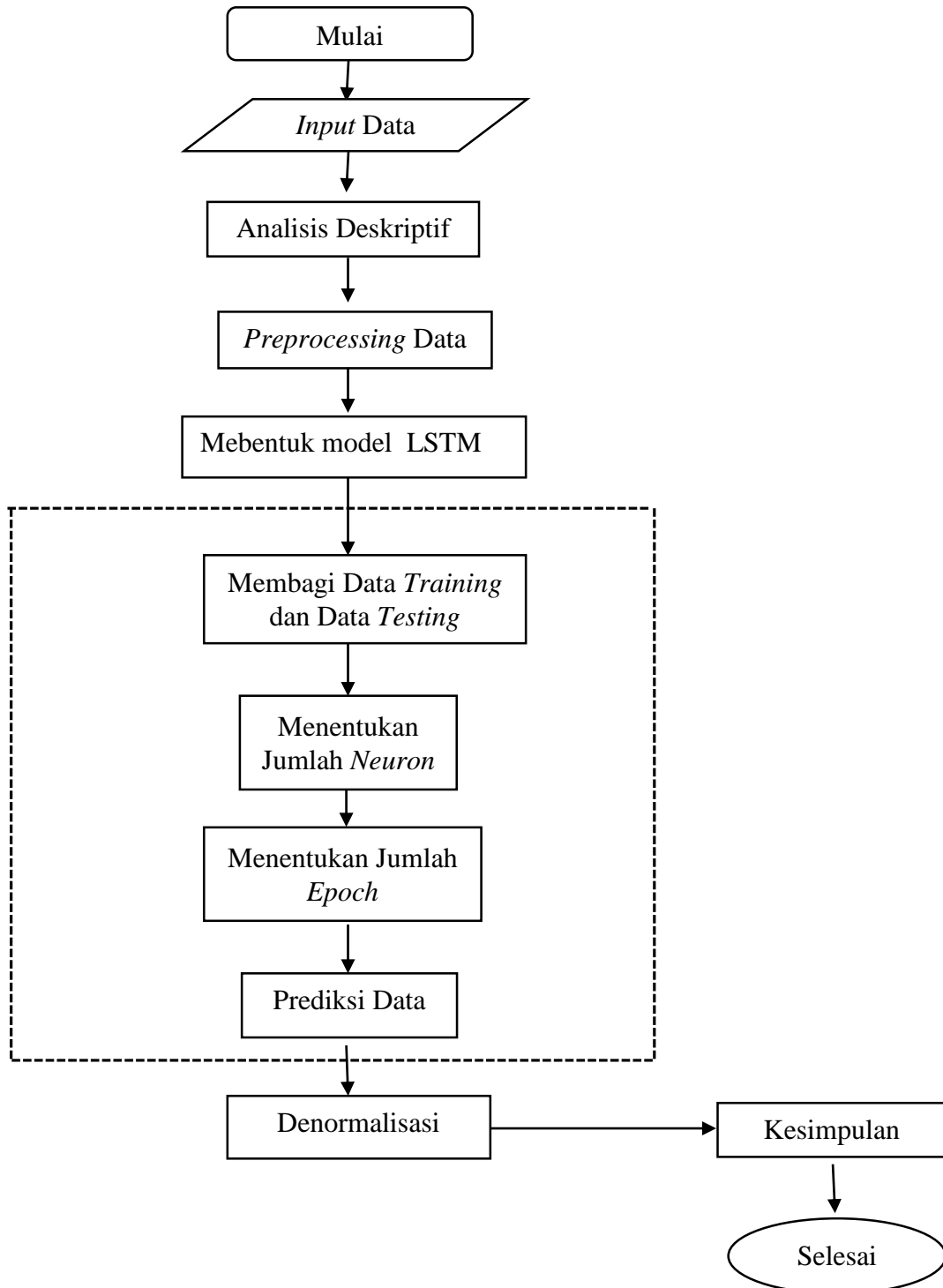
- Membagi data menjadi data *training* dan *testing*
- Menentukan jumlah neuron pada *hidden layer*
- Menentukan jumlah *epoch*
- Melakukan Prediksi

4. Denormalisasi data prediksi

Pada tahap ini data yang awalnya di normalisasi akan dikembalikan pada data *real* untuk mengetahui nilai prediksi sebenarnya.

5. Membandingkan nilai *error* yaitu MSE, RMSE dan MAPE.

Berikut merupakan *flowchart* dari langkah penelitian :



**Gambar 4. 1** *Flowchart* Analisis Data



## BAB 5 HASIL DAN PEMBAHASAN

### 5.1 Analisis Deskriptif data Ethereum

Sebelum melakukan analisis data menggunakan metode *Long Short Term Memory*, peneliti melakukan analisis Deskriptif terlebih dahulu pada data historis Ethereum untuk mengetahui gambaran umum harga Ethereum selama dua tahun terakhir yaitu dari 8 Januari 2018 sampai 7 Januari 2020.



**Gambar 5. 1** Grafik harga harian Ethereum dari 7 Januari 2018 s.d 7 Januari 2020

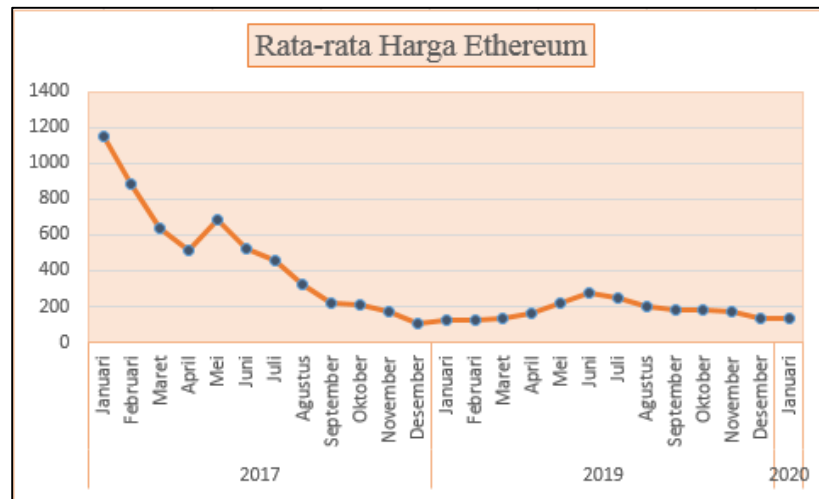
Gambar 5.1 menunjukkan grafik harga Ethereum selama 2 tahun terakhir, terlihat dari grafik bahwa data memiliki *trend* turun, dimana pada awal Januari 2018 harga ethereum tinggi dan mengalami penurunan selama dua tahun berikutnya. Grafik menunjukkan harga tertinggi terdapat pada tanggal 10 Januari 2018 dengan harga 1405.21 US\$, terlihat bahwa harga Ethereum tinggi di awal tahun 2018 namun secara umum juga mengalami penurunan selama tahun 2018 dan mulai stabil di tahun 2019 dengan sedikit kenaikan di pertengahan 2019. Harga Ethereum yang sangat tinggi pada awal tahun 2018 disebabkan oleh demam *Initial Coin Offering* (ICO) yaitu konsep penggalangan dana dalam mata uang kripto, namun melemah di pertengahan tahun 2018 dan secara umum mulai stabil sampai hari ini. pada tabel 5.1 di deskripsikan rata rata harga ethereum dalam bulan dari data harga ethereum yang penulis teliti.

**Tabel 5.1.** Rata-rata harga Ethereum

Tahun	Bulan	Rata-Rata	Tahun	Bulan	Rata-Rata
2018	Januari	1148.166	2019	Januari	127.3568
	Februari	879.4732		Februari	124.1537
	Maret	635.7553		Maret	135.237
	April	513.9751		April	164.5632
	Mei	680.4127		Mei	216.7866
	Juni	520.1225		Juni	272.2108
	Juli	458.6427		Juli	250.7951
	Agustus	326.8763		Agustus	200.8032
	September	218.6123		September	185.5804
	Oktober	210.4634		Oktober	178.0947
	November	169.9677		November	173.979
	Desember	106.6684		Desember	137.9575
Tahun	Bulan	Rata-Rata			
2020	Januari	132.9176			

Tabel 5.1 merupakan nilai rata rata harga ethereum dalam US\$ selama dua tahun dari tanggal 8 Januari 2018 sampai 7 Januari 2020. Dari tabel dapat diketahui gambaran umum nilai rata rata harga ethereum per bulannya guna mewakili gambaran data dari keseluruhan data. Dari nilai rata – rata harga ethereum tahun 2018, nilai terendah yaitu pada bulan Desember dengan harga Ethereum 106,6684 US\$, sedangkan nilai tertinggi berada pada bulan Januari dengan harga Ethereum 1148,4731 US\$, pada tahun 2019 rata – rata harga ethereum diketahui terendah yaitu pada bulan Februari dengan harga Ethereum 124.1537 US\$, sedangkan nilai tertinggi berada pada bulan Juni dengan harga Ethereum 272.2108 US\$. Pada tahun 2020 dengan data 8 hari pada bulan januari rata – rata harga ethereum yaitu 132,9176 US\$. Selama 2 tahun terakhir, jika di dibandingkan dengan satu bulan sebelumnya, rata – rata harga Ethereum selalu turun pada bulan Februari dan naik pada bulan Mei, dan secara keseluruhan rata – rata harga ethereum cenderung

mengalami penurunan dari pertengahan tahun bulan Juni sampai akhir tahun bulan Desember.



**Gambar 5. 2** Grafik Rata rata harga Ethereum

Gambar 5.2 menggambarkan grafik rata rata harga Ethereum selama dua tahun terakhir dari 8 Januari 2018 sampai 7 Januari 2020. Grafik menunjukkan *trend* menurun pada rata rata harga ethereum selama dua tahun terakhir. Penurunan yang terjadi dari tahun 2018 nampak membaik kembali pada tahun 2019, rata - rata tertinggi selama tahun 2019 dan 2020 nampak pada bulan juni 2019.

## 5.2 Analisis Long Short Term Memory

### 5.2.1 Normalisasi Data

Dalam melakukan analisis LSTM dilakukan normalisasi data set terlebih dahulu untuk meminimalisir *error*, data aktual di ubah menjadi nilai dengan *range* interval 0 – 1 menggunakan rumus *min-max scaling* pada persamaan 3.10. berikut contoh perhitungan *min-max scaler* menggunakan data pertama pada variabel *price*:

1. Normalisasi  $X_1$  Variabel pertama

Diketahui :

$$x = 1261.41$$

$$\min_x = 83.15$$

$$\max_x = 1405.21$$

Ditanya :

$$x' \dots ?$$

Jawab :

$$\begin{aligned} x' &= \frac{x - \min_x}{\max_x - \min_x} = \frac{1261.41 - 83.15}{1405.21 - 83.15} \\ &= \frac{1178.26}{1322.06} \\ &= 0.891230 \end{aligned}$$

2. Normalisasi  $X_4$  Variabel pertama

Diketahui :

$$x = 15.98$$

$$\min_x = 13.11$$

$$\max_x = 20.85$$

Ditanya :

$$x' \dots ?$$

Jawab :

$$\begin{aligned} x' &= \frac{x - \min_x}{\max_x - \min_x} = \frac{15.98 - 13.11}{20.85 - 13.11} \\ &= \frac{2.87}{7.74} \\ &= 0.370903 \end{aligned}$$

Berikut adalah hasil normalisasi seluruh variabel data :

**Tabel 5.2.** Hasil Normalisasi Data  $X_1 - X_5$

No	$X_1$	$X_2$	$X_3$	$X_4$	$X_5$
1	0.891230	0.827616	0.977680	0.370903	0.413538
2	0.859704	0.850116	0.947660	0.380347	0.423363
3	1.000000	0.745258	0.952820	0.366673	0.439746
4	0.806249	0.774107	0.926320	0.360955	0.464499
5	0.849198	0.746293	0.957487	0.368379	0.468368
⋮	⋮	⋮	⋮	⋮	⋮

No	X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	X <sub>4</sub>	X <sub>5</sub>
725	0.035352	0.289276	0.208547	0.278135	0.327924
726	0.033211	0.253242	0.173493	0.019444	0.348573
727	0.037685	0.235090	0.014694	0.005541	0.387380
728	0.038328	0.208030	0.021381	0.026704	0.362830
729	0.039094	0.290556	0.206651	0.000000	0.375784

**Tabel 5.3.** Hasil Normalisasi Data X<sub>6</sub> - X<sub>10</sub> & Y

No	X <sub>6</sub>	X <sub>7</sub>	X <sub>8</sub>	X <sub>9</sub>	X <sub>10</sub>	Y
1	0.152949	0.494888	0.843330	0.000000	0.934044	0.859704
2	0.155272	0.520897	0.902408	0.004694	0.977572	1.000000
3	0.153052	0.629776	1.000000	0.009723	0.891840	0.806249
4	0.190087	0.527354	0.875089	0.014628	0.885350	0.849198
5	0.202915	0.382780	0.838627	0.019035	0.826865	0.920382
⋮	⋮	⋮	⋮	⋮	⋮	⋮
725	0.269397	0.015964	0.008393	0.994802	0.148399	0.033211
726	0.140855	0.025112	0.016457	0.996063	0.230988	0.037685
727	0.190913	0.013453	0.014501	0.997235	0.206305	0.038328
728	0.172244	0.011839	0.016801	0.998291	0.168457	0.039094
729	0.166797	0.028341	0.019523	0.999398	0.279390	0.045370

### 5.2.2 Pembagian Data *Training* dan *Testing*

Data yang di teliti di bagi menjadi data *training* dan *testing*, pada penelitian ini peneliti menggunakan perbandingan 80 % untuk data *training* dan 20 % untuk data *testing*, dimana ini berarti sebanyak 583 data untuk *training* dan 146 data untuk *testing*.

**Tabel 5.4.** Hasil Normalisasi Data X<sub>6</sub> - X<sub>10</sub> & Y

Satuan	Data <i>Training</i>	Data <i>Testing</i>
Persen	80 %	20 %
Jumlah data	583	146

Pembuatan data *training* dilakukan dalam rangka meningkatkan kinerja LSTM terhadap data *testing*, *training* lebih besar dibandingkan data *testing* dengan maksud agar mesin pembelajaran atau algoritma pembelajaran lebih terlatih dengan pola data dari data *training*. Hal ini berguna ketika algoritma atau mesin menghasilkan suatu model dan model tersebut diberikan kepada data *testing* akan memberikan prediksi data *testing* yang akurat. Data *training* yang sudah didapatkan, akan digunakan untuk proses pelatihan. Proses pelatihan dilakukan dengan menggunakan metode LSTM sehingga terbentuk suatu model yang nanti akan diuji performasinya terhadap data *testing*. Proses tersebut terus diulang hingga mendapatkan model dengan akurasi yang paling bagus. Kemudian setelah didapatkan model yang terbaik, model tersebut akan digunakan untuk proses prediksi.

### 5.2.3 Penentuan jumlah *Neuron* dan *Epoch*

Dalam menentukan jumlah *neuron*, tidak ada aturan yang mengatur banyaknya *neuron* yang digunakan, sehingga nilai *neuron* dapat di pakai peneliti dengan terlebih dahulu melakukan percobaan sampai mendapatkan hasil yang optimal dalam prediksi yang bisa di lihat dari nilai *error*. Begitu juga dengan *Epoch*, *Epoch* adalah langkah yang dilakukan pada proses pembelajaran neural network, dimana besarnya *epoch* yang telah ditetapkan akan mempengaruhi besaran proses pembelajaran dan berhenti tepat pada nilai *epoch* yang telah ditentukan tersebut.

Pada penelitian ini jaringan yang dibentuk dengan 10 variabel *input* dan 1 *output layer* dengan jumlah neuron pada hidden layer yang akan digunakan untuk percobaan yaitu 10, 20, 30 dan 40, kemudian percobaan pada *epoch* menggunakan *epoch* 100, 500, 1000, dan 1500 setelah dilakukan percobaan maka akan diketahui jumlah neuron dan *epoch* yang lebih tepat, dinyatakan dengan nilai *error* paling kecil yang dihasilkan, selain itu, peneliti juga melihat grafik hasil prediksi dan data aktual apakah saling mengikuti atau tidak, dengan menggunakan optimasi ADAM, berikut adalah hasil beberapa percobaan yang di lakukan.

**Tabel 5.5.** Hasil Percobaan *neuron* dan *Epoch*

No	Jumlah <i>Neuron</i>	<i>Epoch</i>	Error (MSE)
1	10	100	1625.887
2		500	71.694
3		1000	180.757
4	20	100	4959.099
5		500	62.183
6		1000	49.014
7	30	100	1302.105
8		500	62.909
9		1000	73.327
10	40	100	2865.294
11		500	81.461
12		1000	60.642
13	50	100	2347.884
14		500	45.449
15		1000	48.350

Tabel 5.5 menunjukkan akurasi terbaik pada jumlah *neuron* dan *epoch* adalah pada no 14 yaitu dengan menggunakan *neuron* 50 dan *epoch* 500 di peroleh nilai *error* terendah yaitu 45,449. Jumlah *epochs* (iterasi) merepresentasikan lamanya proses pembelajaran yang dilakukan terhadap jaringan yang sedang diobservasi. Jumlah *epochs* yang terlalu sedikit mengakibatkan jaringan yang terbentuk bersifat terlalu general, berarti kemampuan jaringan dalam mengenali pola terlalu sedikit atau bahkan tidak ada sama sekali. Sedangkan jumlah *epochs* yang terlalu banyak akan mengakibatkan jaringan mengalami kondisi overfit (jaringan bersifat terlalu spesifik terhadap data pelatihan), itu tampak pada tabel 5.4 sehingga hasil terbaik tidak berada pada nilai *epoch* terbesar, melainkan *neuron* dan *epoch* 500, arsitektur inilah yang akan dipakai dalam prediksi harga Ethereum.

Nilai bobot dan bias yang didapatkan dari *neuron* 50 dan *epoch* 500 dapat dilihat pada Gambar 5.3 dan Gambar 5.4

```

W_i[0]
array([ 0.04137211, -0.3269089 , -0.0932183 , -0.04260813,  0.09186599,
        -0.01484952,  0.00335789, -0.05930417, -0.03215478,  0.14579527,
        -0.15690951,  0.10897666,  0.16906671,  0.05078574,  0.21529323,
        -0.01014763, -0.01565879,  0.01974018,  0.16075662,  0.12433653,
        -0.02449567,  0.04801587, -0.22152877, -0.00338336, -0.19439912,
         0.01109931,  0.03926298,  0.18273611, -0.27153018, -0.12701054,
        -0.24727367, -0.596266 , -0.05463586, -0.0967299 , -0.04921798,
        -0.06819604,  0.21947128,  0.27735054,  0.09617663,  0.05920425,
        -0.27996507,  0.05379887,  0.19067082, -0.08303878,  0.06120738,
         0.04669481,  0.20326303, -0.03799409, -0.1799614 ,  0.09892186],
      dtype=float32)

W_f[0]
array([ 0.01314129,  0.10163048,  0.1223585 ,  0.04357025,  0.00389226,
         0.16125572, -0.02808368, -0.069257 ,  0.11343402,  0.1467818 ,
         0.08740884, -0.10587753,  0.12328762,  0.0387512 ,  0.01009312,
        -0.13373719,  0.14790389, -0.10399911,  0.01423633, -0.10066371,
        -0.10919246,  0.1615569 , -0.11019754,  0.04213645,  0.13865238,
         0.15088421, -0.09544336, -0.1654825 , -0.01250237,  0.05355339,
        -0.02542089, -0.07596826, -0.12203111, -0.10759126,  0.10467523,
         0.02555542, -0.06051321, -0.00302392, -0.01208712, -0.06301242,
         0.15452841, -0.02877028, -0.09334952, -0.04117413,  0.09503075,
        -0.14011827, -0.14858934, -0.02185668,  0.02162004,  0.02042618],
      dtype=float32)

: W_c[0]
: array([ 0.16894172, -0.26389325,  0.09045851,  0.15400624, -0.18527617,
         0.16784856,  0.1968077 , -0.09361622, -0.15257116,  0.16573514,
        -0.23481572,  0.2159641 ,  0.02246537,  0.18755306, -0.03905859,
        -0.00260331, -0.25798953,  0.19815311,  0.2560286 ,  0.09068008,
        -0.02446297, -0.06482977, -0.36104625, -0.2837686 , -0.23980175,
        -0.02904546, -0.10336313,  0.2436763 ,  0.20970415, -0.08752364,
        -0.3204812 , -0.28539947, -0.0151619 , -0.02522712,  0.01256865,
         0.19163364, -0.23023376, -0.09350353, -0.03624837,  0.52276903,
        -0.2627832 ,  0.00856243,  0.05155641, -0.05910116, -0.0768264 ,
        -0.1540255 ,  0.17212872,  0.17155644, -0.2321383 , -0.16890171],
      dtype=float32)

: W_o[0]
: array([-0.13565929,  0.02533614,  0.03979781,  0.03755527, -0.20824978,
        -0.03602203, -0.14087728,  0.13040479,  0.10668406,  0.08961078,
        -0.01848932,  0.07352116, -0.06958553,  0.10305837,  0.02653852,
        -0.02596452,  0.19271825, -0.25303814,  0.04781611,  0.1519498 ,
        -0.01870206, -0.16226214, -0.22602814, -0.01734742, -0.3656995 ,
         0.10826843,  0.15363248,  0.07809943, -0.1753169 , -0.09955008,
         0.04054255, -0.24131292,  0.00878605, -0.0592967 ,  0.05203234,
        -0.06687432,  0.13607286, -0.05322987,  0.12544833, -0.42939028,
        -0.11093324, -0.04227012, -0.04344539,  0.01900772,  0.07632752,
        -0.04383947, -0.05674011, -0.16383064, -0.18063852, -0.10211511],
      dtype=float32)

```

**Gambar 5. 3** Hasil Bobot dari setiap Neuron untuk Variabel *Price*

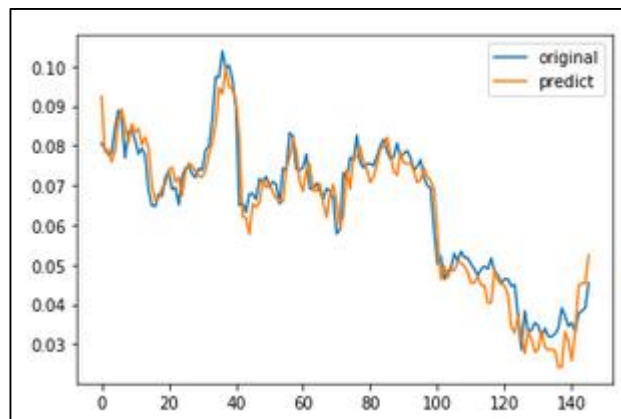




setiap data masukan akan diolah dan dipilih data mana saja yang akan disimpan atau dibuang pada *memory cells*, pada *input gate* yaitu masukan nilai dan *output gate* yaitu keluaran, dan semua aktifitas dalam sel LSTM memiliki bias dan bobot.

#### 5.2.4 Prediksi harga Ethereum

Setelah diperoleh arsitektur terbaik untuk melakukan prediksi, dengan menggunakan data *training* 80 % data *testing* 20 % serta dengan *neuron* 50 dan *Epoch* 500, maka di dapatkan grafik perbandingan data aktual harga Ethereum dengan data prediksi harga ethereum seperti pada gambar 5.1



**Gambar 5. 5** Grafik data aktual dan Prediksi

Gambar 5.1 menunjukkan bahwa dengan menggunakan model yang sudah di bentuk, menghasilkan output yang sesuai, terlihat dari pola data garis *testing* dan pola data garis prediksi tidak jauh berbeda. Selanjutnya dilakukan denormalisasi data untuk dapat menampilkan hasil prediksi dengan nilai prediksi sebenarnya, denormalisasi dilakukan dengan cara berikut berdasarkan persamaan 3.15 :

1. Denormalisasi prediksi harga ethereum tanggal 15 Agustus 2019

Diketahui :

$$y = 0.09233867$$

$$X_{min} = 83.14924$$

$$X_{max} = 1405.21$$

Ditanya :

$$X_n \dots ?$$

Jawab :

$$X_t = y (X_{max} - X_{min}) + X_{min}$$

$$\begin{aligned}
&= 0.09233867(1405.21 - 83.14924) + 83.14924 \\
&= 0.09233867 (1322.06076) + 83.14924 \\
&= 122.0773322 + 83.14924 \\
&= 205.22658
\end{aligned}$$

2. Denormalisasi prediksi harga tanggal 7 januari 2020

Diketahui :

$$y = 0.05241586$$

$$X_{min} = 83.14924$$

$$X_{max} = 1405.21$$

Ditanya :

$$X_n \dots ?$$

Jawab :

$$\begin{aligned}
X_t &= y (X_{max} - X_{min}) + X_{min} \\
&= 0.05241586 (1405.21 - 83.14924) + 83.14924 \\
&= 0.05241586 (1322.06076) + 83.14924 \\
&= 69.29695171 + 83.14924 \\
&= 152.446
\end{aligned}$$

Berikut merupakan hasil prediksi harga Ethereum :

**Tabel 5.6.** Hasil Prediksi

No	Tanggal	Prediksi harga Ethereum	Data Aktual
1	15 Agustus 2019	205.22658	189.68762
2	16 Agustus 2019	186.93863	187.92783
3	17 Agustus 2019	187.59734	185.93251
4	18 Agustus 2019	183.47719	186.37363
5	19 Agustus 2019	187.10217	194.77446
6	20 Agustus 2019	196.9037	200.65839
7	21 Agustus 2019	201.11385	198.13786
8	22 Agustus 2019	195.56195	184.89603
9	23 Agustus 2019	189.94884	193.73198

No	Tanggal	Prediksi harga Ethereum	Data Aktual
10	24 Agustus 2019	196.11545	193.9669
	:	:	:
	:	:	:
137	29 Desember 2019	115.012856	128.01114
138	30 Desember 2019	115.16473	134.76804
139	31 Desember 2019	126.90002	132.07307
140	1 Januari 2020	123.758095	128.72261
141	2 Januari 2020	117.24311	129.88736
142	3 Januari 2020	126.677795	127.0556
143	4 Januari 2020	142.20247	132.97084
144	5 Januari 2020	143.16585	133.82062
145	6 Januari 2020	143.0002	134.83426
146	7 Januari 2020	152.44618	143.1316

Tabel 5.6 merupakan hasil prediksi yang didapatkan untuk harga ethereum dengan akurasi 98.34 % berdasarkan nilai *Mean Absolute Percentage Error* (MAPE). Dengan nilai MAPE 1.69 % menunjukkan bahwa hasil prediksi sangat baik.

## BAB 6 PENUTUP

### 6.1 Kesimpulan

Tuliskan kesimpulan yang menjawab rumusan masalah:

1. Harga Ethereum selama 2 tahun terakhir menunjukkan harga tertinggi terdapat pada tanggal 10 Januari 2018 dengan harga 1405.21 US\$, terlihat bahwa harga Ethereum tinggi di awal tahun 2018 namun secara umum juga mengalami penurunan yang mana dilihat dari pergerakan grafik harga mengalami tren turun selama tahun 2018 dan mulai stabil di tahun 2019 dengan sedikit kenaikan di pertengahan 2019. Harga Ethereum yang sangat tinggi pada awal tahun 2018 disebabkan oleh demam *Initial Coin Offering* (ICO) yaitu konsep penggalangan dana dalam mata uang kripto, namun melemah di pertengahan tahun 2018 dan secara umum mulai stabil sampai hari ini.
2. LSTM berhasil di terapkan dalam prediksi harga Ethereum dengan menggunakan informasi *blockchain*, menggunakan data *training* sebesar 80 % yaitu sebanyak 583 data, data *testing* sebesar 20 % yaitu sebanyak 146 data, kemudian dengan menggunakan *neuron* 50 dan *epoch* 500 diperoleh prediksi dengan nilai error terkecil. Prediksi harga ethereum selama 10 hari awal data prediksi dimulai tanggal 15 Agustus 2019 sampai 24 Agustus 2019 masing masing adalah 205.22658 ; 186.93863 ; 187.59734 ; 183.47719 ; 187.10217 ; 196.9037 ; 201.11385 ; 195.56195 ; 189.94884 ; 196.11545 dan 10 hari terakhir dari data harga ethereum diperoleh nilai prediksi tanggal 29 Desember 2019 sampai 7 Januari 2020 masing masing adalah 115.012856 ; 115.16473 ; 126.90002 ; 123.758095 ; 117.24311 ; 126.677795 ; 142.20247 ; 143.16585 ; 143.0002 ; 152.44618 dengan nilai error yang sangat kecil yaitu MAPE 1.69% menunjukkan hasil prediksi sangat baik.

## 6.2 Saran

Berdasarkan hasil penelitian, saran yang dapat di berikan penulis :

1. Penelitian selanjutnya dapat menggunakan dan membandingkan optimasi lain seperti optimasi Adamax dan optimasi RMSprop sehingga dapat memperoleh hasil prediksi yang lebih akurat.
2. Penelitian selannjutnya dapat menambahkan variabel lain seperti makro ekonomi untuk melihat hubungan antara *cryptocurrency* dengan kondisi ekonomi.

## DAFTAR PUSTAKA

- Aldi, M. W., & dkk. (2018). Analisis dan Implementasi Long Short Term Memory Neural Network untuk Prediksi. *ISSN*, 1-8.
- Aldi, M. W., & dkk. (2018). Analisis dan Implementasi Long Short Term Memory Neural Network untuk Prediksi. *ISSN*, 3551.
- Aldi, M. W., Jondri, & Aditsania, A. (2018). Analisis dan Implementasi Long Short Term Memory Neural Network untuk Prediksi. *e-Proceeding of Engineering*, 3551-3552.
- Andini, T. D., & Auristandi, P. (2016). Peramalan Jumlah Stok Alat Tulis Kantor Di UD ACHMAD. *JITIKA*, 2-3.
- Ausop, A. Z., & Aulia, E. S. (2018). TEKNOLOGI CRYPTOCURRENCY BITCOIN DALAM TRANSAKSI BISNIS MENURUT SYARIAT ISLAM. *Sosioteknologi*, 19.
- BBC. (2014, December 30). *AirAsia QZ8501: Does bad weather cause plane crashes?* Retrieved from BBC: <http://www.bbc.com/news/world-30631968>
- Budiman, H. (2016). Analisis dan Perbandingan Akurasi Model Prediksi Rentet Waktu SUPPORT VECTOR MACHINES dengan Support Vektor Machine Particle Swarm Optimization untuk Arus Lalu Lintas Jangka Pendek. *SYSTEMIC*, 21-22.
- C.Lipton, Z., & dkk. (2016). LEARNING TO DIAGNOSE WITH LSTM RECURRENT. *ICLR*, 1-18.
- colah. (2015, Agustus 27). Retrieved from Colahsblog: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- Darmawan, G., & dkk. (2016). PERAMALAN PADA DATA IRREGULAR SINUSOIDAL DENGAN MENGGUNAKAN MODEL HOLT-WINTERS . *Prosiding Seminar Nasional Matematika dan Terapannya*, 183.
- Dharma, S., Putera, A., & Ardana, P. D. (2011). ARTIFICIAL NEURAL NETWORKS UNTUK PEMODELAN. *Bumi Lestari*, 10-11.
- Donges, N. (2018). Gradient Descent in a Nutshell.

- Eka, R. (2018, 03 15). Retrieved from DailySocialid: <https://dailysocial.id/post/mengenal-cryptocurrency-dan-mekanisme-transaksinya>
- ethereum.org. (2020). Retrieved from Ethereum.org: <https://ethereum.org/>
- Grech, A., & Camillera, A. F. (2017). Blockchain in Education. 11.
- Habibi, M. Y., & Riksakomara, E. (2017). Peramalan Harga Garam Konsumsi Menggunakan Artificial Neural Network Feedforward-Backpropagation. *TEKNIK ITS*, 307.
- Hansun, S. (2016). Peramalan Data IHSG Menggunakan. *IJCCS*, 80-81.
- Harwick, C. (2016). Cryptocurrency and the problem of intermediation. *Independent Review*, 570.
- Houben, R., & Snyers, A. (2018). Cryptocurrencies and Bitcoin. *IPOL*, 20.
- Isna, T. D. (2019). *Potensi Mata Uang Virtual Indonesia dan Meluncurnya Teknologi LEN oleh Liqnet*. 30 Januari 2019.
- Jang, H., & Lee, J. (2017). An Empirical Study on Modeling and Prediction. *IEEE ACCESS*, Vol.6, No. 2779181, 1-11.
- Jani, S. (2018). An Overview of Ethereum & Its Comparison with Bitcoin. *International Journal of Scientific & Engineering Research*, 2.
- Jayanti, Y. D. (2013, 10 23). Retrieved from SlideShare: <https://www.slideshare.net/yunitadwijayanti/materi-8-analisis-time-series>
- Johnson, R. A., & Bhattacharyya, G. K. (2010). *Statistics Principles & Methods*. USA: John Wiley & Sons.
- Juanda, R. A., Jondri, & Rohmawati, A. A. (2018). Prediksi Harga Bitcoin Dengan Menggunakan Recurrent Neural Network. *eproc*, 3684.
- Karim, F., & dkk. (2019). Multivariate LSTM-FCNs for time series classification. *ELSEVIER*, 1-9.
- Kingma, D. P., & Ba, J. L. (2015). ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. *ICLR*, 2.
- Kurniawansyah, A. S. (2018). IMPLEMENTASI METODE ARTIFICIAL NEURAL NETWORK DALAM MEMPREDIKSI HASIL UJIAN KOMPETENSI KEBIDANAN. *Pseudocode*, 38.



- Lapian, E., Osmond, A. B., & Saputra, R. E. (2018). RECURRENT NEURAL NETWORK UNTUK PENGENALAN UCAPAN PADA DIALEK MANADO. *e-Proceeding of Engineering*, 6438.
- Lapian, E., Osmond, A. B., & Saputra, R. E. (2018). RECURRENT NEURAL NETWORK UNTUK PENGENALAN UCAPAN PADA DIALEK MANADO. *e-Proceeding of Engineering*, 6438.
- Li, X. (2013). Comparison and Analysis between Holt Exponential Smoothing and Brown Exponential Smoothing Used for Freight Turnover Forecast. *Third International Conference on Intelligent System Design and Engineering Applications* (pp. 453-456). IEEE.
- LUNO. (2020). Retrieved from Apa itu Ethereum?: <https://www.luno.com/learn/id/article/what-is-ethereum>
- Ma, X. (2015). Long Short-Term Memory Neural Network for Traffic Speed Prediction Using Remote Microwave Sensor Data. *Transportation Research Part C*, 191. <https://www.sciencedirect.com/science/article/pii/S0968090X15000935>.
- Macedo, L. (2018). Blockchain for trade facilitation: Ethereum,eWTP, COs and regulatory issues. *World Custom Journal*, 87.
- Mathsisfun. (2019). *Histogram*. Retrieved from <https://www.mathsisfun.com/data/histograms.html>
- NS, S. (2019, 01 14). Retrieved from bixbux: <https://bixbux.com/blockchain/>
- onnocenter. (2019, September 8). Retrieved from onnocenter: [https://lms.onnocenter.or.id/wiki/index.php/Keras:\\_Introduction\\_to\\_the\\_Adam\\_Optimization\\_Algorithm](https://lms.onnocenter.or.id/wiki/index.php/Keras:_Introduction_to_the_Adam_Optimization_Algorithm)
- OpenLearn. (2020). Retrieved from OpenLearn: <https://www.open.edu/openlearn/ocw/mod/oucontent/view.php?id=81977&section=4>
- Prathama, A. Y., & dkk. (2017). PENDEKATAN ANN (ARTIFICIAL NEURAL NETWORK). *Tekno Sains*, 17.
- Qiu, J., Wang, B., & Zhou, C. (2020). Forecasting stock prices with long-short term. *Plos one*, 2.

- R, A. Y. (2018, July 1). Retrieved from Universitas Gajah Mada Menara Ilmu  
Mechine Learning:  
<http://machinelearning.mipa.ugm.ac.id/2018/07/01/recurrent-neural-network-rnn/>
- R, P. J., & Das, V. S. (2018). Cryptocurrency Price Prediction Using Long-Short. *International Journal of Research and Scientific Innovation*, 1-3.
- Raswan. (2018). PENGARUH METODE PEMBELAJARAN EKLEKTIK TERHADAP HASIL BELAJAR BAHASA ARAB SISWA. *Arabiyat*, 134.
- Rosmayanti. (2018). *49% Orang Indonesia Ingin Jadikan Cryptocurrency sebagai Investasi*. *WartaEkonomi.co.id*.
- StatistikaPendidikan. (2017, 01 07). Retrieved from Statistika Pendidikan:  
<http://www.statistikaonline.com/2017/01/grafik.html>
- Suhartono, D. (2012, July 28). Retrieved from Binus Uversity School of Computer Science: <https://socs.binus.ac.id/2012/07/26/konsep-neural-network/>
- Tian, C., & dkk. (2018). A Deep Neural Network Model for Short-Term Load Forecast Based on Long Short-Term Memory Network and Convolutional Neural Network. *Energies*, 3-4. <https://www.mdpi.com/1996-1073/11/12/3493/pdf>.
- Tiyas. (2019, 08 24). *Statistika Deskriptif*. Retrieved from yuksinau:  
<https://www.yuksinau.id/statistika-deskriptif/>
- Varcellis, C. (2009). *Business Intelligence: Data Mining and Optimization for Decision Making*. Milano, Italy.
- Walpole, R. E., Myers, R. H., Myers, S. L., & Ye, K. (2011). *Probability & Statistics for Engineers & Scientists 9th Ed*. USA: Pearson.
- Wizsa, U. A. (2018, 11 3). Retrieved from Statistics Consultant:  
<https://swanstatistics.com/data-time-series-deret-waktu/>
- Zabrocki, D. (2017, 8 25). Retrieved from PMA:  
<https://www.pma.com/Content/Articles/2017/08/Blockchain-Disruption-of-Produce-and-Floral>

- Zahara, S., Sugianto, & Ilmiddafiq, M. B. (2019). Prediksi Indeks Harga Konsumen Menggunakan Metode Long Short Term Memory (LSTM) Berbasis Cloud Computing. *Resti*, 357.
- Zainudin, A. (2018, 04 26). Retrieved from tirtoid: <https://tirtoid/membayangkan-blockchain-dari-cara-yang-paling-sederhana-cJIT>
- Zheng, J., & dkk. (2017). Electric Load Forecasting in Smart Grid Using Long-Short-Term-Memory based. *51st Annual Conference on Information Sciences and Systems (CISS)*, doi: 10.1109/CISS.2017.7926112, 1-6.

## LAMPIRAN

**Lampiran 1** Data Ethereum dan informasi *Blockchain*

<i>No</i>	<i>Date</i>	<i>Harga</i>	<i>Block Size</i>	<i>Average Block Usage</i>	<i>Block Time</i>	<i>Hashrate</i>
1	1/8/2018	1261.41	32508.05	96.07	15.98	169244991382652.00
2	1/9/2018	1219.73	33018.51	94.94	16.06	171313311047650.00
3	1/10/2018	1405.21	30639.61	95.13	15.95	174761946185360.00
4	1/11/2018	1149.06	31294.12	94.13	15.91	179972415156598.00
5	1/12/2018	1205.84	30663.09	95.31	15.96	180786758961002.00
6	1/13/2018	1299.95	31270.73	95.95	16.01	181494941324746.00
7	1/14/2018	1358.78	29318.47	95.02	15.36	185843023517461.00
8	1/15/2018	1345.07	31849.79	96.91	15.93	181940893609997.00
9	1/16/2018	1244.83	32776.93	95.95	15.64	187440005412479.00
10	1/17/2018	987.103	32423.17	94.41	15.51	191230836873224.00
11	1/18/2018	1004.16	31697.84	94.26	15.52	189752929074887.00
12	1/19/2018	1025.69	28845.75	89.41	15.21	189092097892190.00
13	1/20/2018	1076.94	27413.71	89.88	14.72	196850251225637.00
14	1/21/2018	1126.83	26812.40	88.50	14.87	192869908852699.00
15	1/22/2018	1069.17	25874.35	90.09	14.69	197240411165367.00
16	1/23/2018	996.166	25080.40	90.28	14.66	196701279913322.00
17	1/24/2018	970.868	24560.16	86.25	14.49	201852061186898.00
18	1/25/2018	1088.43	23675.50	81.86	14.62	203633450816362.00
19	1/26/2018	1066.74	26058.77	85.31	14.62	203670332503258.00
20	1/27/2018	1057.04	22449.08	76.85	14.48	202235278058218.00
21	1/28/2018	1147.66	24199.05	81.72	14.48	204792377634458.00
22	1/29/2018	1222.23	25243.28	84.95	14.59	212962921171344.00
23	1/30/2018	1160.35	25234.14	87.10	14.73	210634344369913.00
24	1/31/2018	1066.72	24903.05	87.71	14.68	216063873158441.00
25	2/1/2018	1143.47	26336.15	89.90	14.97	221345110424755.00
26	2/2/2018	956.065	26186.40	87.10	14.73	224647210507972.00
27	2/3/2018	881.151	22592.41	83.07	14.47	223994180118119.00
28	2/4/2018	952.92	22742.30	81.60	14.50	223706972770319.00
29	2/5/2018	834.563	24170.76	83.52	14.34	229478421604306.00
30	2/6/2018	597.362	24937.34	83.10	14.58	226773860135618.00
31	2/7/2018	748.825	22980.04	82.43	14.63	232354077314709.00
32	2/8/2018	819.749	20101.96	78.16	14.40	229680233009038.00
33	2/9/2018	806.536	22107.10	81.63	14.56	229630691293985.00
34	2/10/2018	899.807	22205.93	83.15	14.34	229649902635456.00
35	2/11/2018	818.451	20918.57	80.50	14.56	226660097391199.00
36	2/12/2018	847.771	22146.37	81.44	14.53	231713545056539.00

<i>No</i>	<i>Date</i>	<i>Harga</i>	<i>Block Size</i>	<i>Average Block Usage</i>	<i>Block Time</i>	<i>Hashrate</i>
37	2/13/2018	852.951	20531.71	77.94	14.51	235830502167110.00
38	2/14/2018	852.05	21818.24	78.29	14.77	230128891433938.00
39	2/15/2018	937.988	21833.22	80.23	14.49	235883562181430.00
40	2/16/2018	931.282	20716.17	79.65	14.50	240514756599544.00
41	2/17/2018	966.12	21848.77	81.05	14.36	238082227967925.00
42	2/18/2018	958.569	20740.00	82.37	14.42	243014670864273.00
43	2/19/2018	933.128	21824.90	83.99	14.64	239628368224037.00
44	2/20/2018	953.288	24506.91	90.34	14.87	229849344601025.00
45	2/21/2018	884.42	23338.20	87.42	14.79	239933869683796.00
46	2/22/2018	860.26	21841.67	82.86	14.71	238282819357045.00
47	2/23/2018	846.845	20812.46	80.23	14.55	244375551683154.00
48	2/24/2018	875.513	21326.57	81.78	14.78	241261546449377.00
49	2/25/2018	844.86	20615.06	80.86	14.76	247758704582663.00
50	2/26/2018	856.866	21753.97	80.62	14.89	240655628331217.00
:	:	:	:	:	:	:
:	:	:	:	:	:	:
681	11/19/2019	178.4222	25556.80	81.12	14.64	174983158769387.00
682	11/20/2019	175.5937	23007.71	82.65	14.44	172230399039944.00
683	11/21/2019	174.7797	29001.13	84.98	14.21	180911115058217.00
684	11/22/2019	160.518	31879.99	88.83	14.51	175592628242277.00
685	11/23/2019	149.5245	26196.02	76.63	14.35	103959120019678.00
686	11/24/2019	152.1362	23445.04	72.48	14.36	178707222347234.00
687	11/25/2019	144.3153	30500.22	89.91	14.62	183253886207914.00
688	11/26/2019	146.9818	30737.46	89.89	15.27	167125156225239.00
689	11/27/2019	148.0069	28471.82	86.43	15.30	172666159152170.00
690	11/28/2019	152.9573	27630.72	87.97	15.37	172237731575273.00
691	11/29/2019	150.412	27712.84	82.94	15.36	172213907796870.00
692	11/30/2019	153.6738	24388.17	79.25	15.12	173893157073057.00
693	12/1/2019	151.6554	25977.61	80.66	15.42	171131408271819.00
694	12/2/2019	151.5529	26553.09	79.37	15.41	133604585802497.00
695	12/3/2019	149.4547	24375.48	77.27	15.33	177178650477123.00
696	12/4/2019	147.8113	29407.95	83.53	15.50	171735247982346.00
697	12/5/2019	145.6033	24651.95	78.99	15.37	169007997387650.00
698	12/6/2019	147.827	23053.19	76.90	15.42	170342303854662.00
699	12/7/2019	148.6504	19511.92	72.92	15.26	171851312637692.00
700	12/8/2019	147.8713	20686.03	73.43	15.41	163881034594765.00
701	12/9/2019	151.4107	25137.51	77.93	15.32	166869045468602.00
702	12/10/2019	147.1595	25316.46	82.94	15.16	173696037315900.00
703	12/11/2019	145.7973	24694.58	85.00	15.50	171095723605735.00

<i>No</i>	<i>Date</i>	<i>Harga</i>	<i>Block Size</i>	<i>Average Block Usage</i>	<i>Block Time</i>	<i>Hashrate</i>
704	12/12/2019	143.0108	24858.57	85.29	15.14	173963329800487.00
705	12/13/2019	144.5101	24709.88	84.29	15.81	168890716525569.00
706	12/14/2019	144.4856	23201.99	80.85	16.76	171936352881389.00
707	12/15/2019	142.0555	23013.14	80.48	16.87	169573941718717.00
708	12/16/2019	142.6819	27038.74	85.90	16.95	168106147382969.00
709	12/17/2019	132.3436	27211.34	90.23	17.10	163476584619579.00
710	12/18/2019	120.8798	28260.89	89.65	17.40	154658663994811.00
711	12/19/2019	133.8778	27192.05	88.04	17.24	154841511036878.00
712	12/20/2019	127.2737	27328.13	88.13	17.38	147664855465749.00
713	12/21/2019	127.3753	24681.26	81.65	17.01	151641842311546.00
714	12/22/2019	129.8708	24543.96	80.50	17.35	148605845882650.00
715	12/23/2019	128.9106	27378.90	87.32	17.21	150168401071212.00
716	12/24/2019	125.0138	25054.72	83.13	17.08	148874665797153.00
717	12/25/2019	128.0057	24134.76	81.47	17.22	145579227862651.00
718	12/26/2019	125.3335	24829.21	81.91	17.28	143877457016215.00
719	12/27/2019	125.2583	24642.25	81.35	17.18	151090445613425.00
720	12/28/2019	126.1492	21572.41	72.61	17.04	154156317512679.00
721	12/29/2019	128.0111	21074.22	73.97	17.16	153889701787443.00
722	12/30/2019	134.768	24644.70	77.47	17.23	153401125771453.00
723	12/31/2019	132.0731	24268.71	77.97	17.33	151244880523948.00
724	1/1/2020	128.7226	20542.13	75.16	17.34	145848801705056.00
725	1/2/2020	129.8874	20294.85	67.01	15.27	151223324831301.00
726	1/3/2020	127.0556	19477.35	65.69	13.26	155569786332788.00
727	1/4/2020	132.9708	19065.55	59.69	13.16	163738715349880.00
728	1/5/2020	133.8206	18451.63	59.94	13.32	158570896108521.00
729	1/6/2020	134.8343	20323.88	66.94	13.11	161297836006202.00
730	1/7/2020	143.1316	23160.00	72.65	13.25	82194882422458.90

<i>No</i>	<i>Date</i>	<i>Difficulty</i>	<i>Fee Mining Reward</i>	<i>Mining Revenue</i>	<i>Total Accounts</i>	<i>Transaction</i>
1	1/8/2018	2005041444599080.00	2958.00	158.27	20898221.00	1220815.00
2	1/9/2018	2009431180358000.00	3103.00	168.74	21186211.00	1260681.00
3	1/10/2018	2005236673361900.00	3710.00	186.05	21494797.00	1182162.00
4	1/11/2018	2075219093758190.00	3139.00	163.90	21795749.00	1176218.00
5	1/12/2018	2099459136321310.00	2333.00	157.43	22066179.00	1122654.00
6	1/13/2018	2140480880488410.00	2420.00	171.93	22317576.00	1183718.00
7	1/14/2018	2168670614790460.00	2248.00	167.82	22546848.00	1085165.00
8	1/15/2018	2189067429035480.00	2232.00	164.02	22790245.00	1209525.00

No	Date	Difficulty	Fee Mining Reward	Mining Revenue	Total Accounts	Transaction
9	1/16/2018	2201579182658810.00	2235.00	138.50	23070305.00	1281222.00
10	1/17/2018	2256842549630730.00	2073.00	112.69	23343310.00	1279357.00
11	1/18/2018	2275454971834870.00	2157.00	123.96	23596820.00	1207441.00
12	1/19/2018	2268475042749960.00	1870.00	123.98	23793860.00	1045361.00
13	1/20/2018	2362531144033200.00	1779.00	127.88	23972857.00	985664.00
14	1/21/2018	2372769489516330.00	1328.00	120.93	24147373.00	1009720.00
15	1/22/2018	2394824553778490.00	1198.00	112.31	24364992.00	974765.00
16	1/23/2018	2399066993860960.00	1440.00	108.91	24592186.00	929309.00
17	1/24/2018	2459805089781100.00	1293.00	109.36	24822208.00	914140.00
18	1/25/2018	2508044212478070.00	1060.00	111.53	24963761.00	789412.00
19	1/26/2018	2501011473604530.00	960.00	109.25	25141354.00	954711.00
20	1/27/2018	2485155457862330.00	817.00	116.57	25288646.00	832819.00
21	1/28/2018	2500573972246640.00	888.00	120.79	25459152.00	889367.00
22	1/29/2018	2590818979048730.00	1275.00	120.80	25617918.00	937027.00
23	1/30/2018	2595380398397110.00	1064.00	114.15	25794792.00	908128.00
24	1/31/2018	2626694616700340.00	1070.00	108.87	25962960.00	878195.00
25	2/1/2018	2666139347650760.00	1555.00	108.62	26126214.00	1011550.00
26	2/2/2018	2710826674286140.00	1204.00	89.45	26305990.00	982533.00
27	2/3/2018	2765913557554020.00	888.00	89.13	26458450.00	815547.00
28	2/4/2018	2758818505189200.00	846.00	85.06	26613069.00	801428.00
29	2/5/2018	2786247277489040.00	874.00	72.43	26766442.00	876254.00
30	2/6/2018	2786696275880730.00	927.00	63.08	26917454.00	889046.00
31	2/7/2018	2837110271338450.00	853.00	73.06	27051537.00	820431.00
32	2/8/2018	2821608436226480.00	656.00	74.28	27157971.00	682343.00
33	2/9/2018	2878713251276890.00	693.00	75.34	27285641.00	778851.00
34	2/10/2018	2830896217392410.00	750.00	80.80	27433183.00	836314.00
35	2/11/2018	2953315097964050.00	624.00	72.07	27557012.00	740191.00
36	2/12/2018	2942394222940180.00	621.00	74.72	27706846.00	797475.00
37	2/13/2018	2970659773616900.00	618.00	73.23	27831154.00	730671.00
38	2/14/2018	2955721156517350.00	627.00	76.96	27949262.00	725534.00
39	2/15/2018	2891648662383020.00	628.00	82.74	28081062.00	770024.00
40	2/16/2018	2992149023786980.00	595.00	80.38	28198536.00	696587.00
41	2/17/2018	2973016981706700.00	662.00	83.95	28313680.00	753773.00
42	2/18/2018	3003786489652820.00	765.00	81.57	28420316.00	754772.00
43	2/19/2018	2976407563909830.00	647.00	81.56	28544066.00	801205.00
44	2/20/2018	2933380114258280.00	720.00	82.83	28669806.00	852627.00
45	2/21/2018	2909922282521060.00	739.00	77.15	28778845.00	846967.00
46	2/22/2018	2981123022364420.00	602.00	71.55	28873402.00	749784.00
47	2/23/2018	2966706149420340.00	623.00	73.15	28968107.00	729775.00

No	Date	Difficulty	Fee Mining Reward	Mining Revenue	Total Accounts	Transaction
48	2/24/2018	2972336747929020.00	621.00	72.65	29049281.00	726595.00
49	2/25/2018	3096535813097360.00	546.00	69.39	29126101.00	686085.00
50	2/26/2018	3029231685288050.00	536.00	72.36	29226162.00	720568.00
:	:	:	:	:	:	:
:	:	:	:	:	:	:
681	11/19/2019	2410482289170120.00	514.00	13.01	78016425.00	734294.00
682	11/20/2019	2356406409667640.00	506.00	13.29	78082360.00	686435.00
683	11/21/2019	2454958432704570.00	757.00	12.41	78161865.00	755556.00
684	11/22/2019	2434403575117580.00	1244.00	11.93	78236629.00	788944.00
685	11/23/2019	2425619219470760.00	290.00	11.08	78277501.00	427747.00
686	11/24/2019	2450064108346710.00	419.00	10.74	78379922.00	591819.00
687	11/25/2019	2532086321503880.00	982.00	10.44	78466586.00	762225.00
688	11/26/2019	2452796585333900.00	681.00	10.85	78530983.00	674676.00
689	11/27/2019	2521696441978950.00	545.00	10.60	78587355.00	669997.00
690	11/28/2019	2538178408341060.00	541.00	10.80	78653688.00	618938.00
691	11/29/2019	2538693334524760.00	464.00	10.82	78719966.00	666146.00
692	11/30/2019	2531059429095710.00	402.00	10.74	78797900.00	578402.00
693	12/1/2019	2530074208536140.00	490.00	10.57	78869390.00	612420.00
694	12/2/2019	2539809947928650.00	354.00	10.47	78925678.00	482378.00
695	12/3/2019	2584977271398760.00	433.00	10.21	79005287.00	606538.00
696	12/4/2019	2559145468381280.00	481.00	10.26	79058025.00	650001.00
697	12/5/2019	2511142042010840.00	412.00	10.38	79133950.00	608282.00
698	12/6/2019	2491970039458650.00	371.00	10.50	79192183.00	571641.00
699	12/7/2019	2557786978793560.00	304.00	10.24	79237489.00	480277.00
700	12/8/2019	2485835918010480.00	258.00	10.54	79282330.00	429938.00
701	12/9/2019	2498697665249090.00	375.00	10.59	79349046.00	624550.00
702	12/10/2019	2567112149177860.00	411.00	10.15	79403528.00	618272.00
703	12/11/2019	2524363135166580.00	428.00	10.21	79463503.00	623616.00
704	12/12/2019	2506743111201150.00	419.00	10.17	79599874.00	635373.00
705	12/13/2019	2615550799033730.00	391.00	9.83	79751712.00	608990.00
706	12/14/2019	2850211221978520.00	334.00	8.93	79912440.00	540825.00
707	12/15/2019	2805666136441430.00	345.00	9.01	80069036.00	516242.00
708	12/16/2019	2801769123049480.00	493.00	8.96	80179657.00	623225.00
709	12/17/2019	2691382795566250.00	664.00	8.77	80263194.00	642713.00
710	12/18/2019	2600721792361160.00	565.00	8.70	80349491.00	655647.00
711	12/19/2019	2594203326272300.00	427.00	8.81	80434557.00	613095.00
712	12/20/2019	2529390069833600.00	432.00	9.04	80530800.00	602024.00
713	12/21/2019	2551978024097710.00	336.00	8.87	80721464.00	534158.00
714	12/22/2019	2554624967023670.00	315.00	9.00	80857314.00	517971.00



No	Date	Difficulty	Fee Mining Reward	Mining Revenue	Total Accounts	Transaction
715	12/23/2019	2566168879065020.00	424.00	9.22	80966506.00	620211.00
716	12/24/2019	2534536182241190.00	367.00	9.00	81099253.00	590671.00
717	12/25/2019	2505586710624110.00	349.00	8.90	81244625.00	551530.00
718	12/26/2019	2488193011649510.00	356.00	9.01	81374237.00	570669.00
719	12/27/2019	2492213535891540.00	355.00	8.94	81460483.00	579424.00
720	12/28/2019	2529264305563140.00	305.00	8.93	81538413.00	515482.00
721	12/29/2019	2559397542720900.00	292.00	9.03	81655518.00	478118.00
722	12/30/2019	2547838767138330.00	351.00	9.28	81728106.00	581511.00
723	12/31/2019	2525619960817370.00	372.00	9.20	81791592.00	551055.00
724	1/1/2020	2455443582875460.00	298.00	9.42	81877841.00	462307.00
725	1/2/2020	2225084343566820.00	288.00	10.19	81936540.00	501273.00
726	1/3/2020	1982189874524830.00	339.00	11.62	82013975.00	576913.00
727	1/4/2020	2076779948066590.00	274.00	11.27	82085829.00	554307.00
728	1/5/2020	2041502819814660.00	265.00	11.68	82150681.00	519643.00
729	1/6/2020	2031210177953050.00	357.00	12.17	82218551.00	621243.00
730	1/7/2020	2109815163785040.00	199.00	12.01	82255512.00	369130.00

**Lampiran 2** Output error hasil percobaan

No	Jumlah Neuron	Epoch	Output MSE
1	10	100	<pre>In [21]: # calculate MSE mse = mean_squared_error(inv_y, inv_yhat) print('Test MSE: %.3f' % mse)  Test MSE: 1625.887</pre>
2		500	<pre>In [21]: # calculate MSE mse = mean_squared_error(inv_y, inv_yhat) print('Test MSE: %.3f' % mse)  Test MSE: 71.694</pre>
3		1000	<pre>In [21]: # calculate MSE mse = mean_squared_error(inv_y, inv_yhat) print('Test MSE: %.3f' % mse)  Test MSE: 180.757</pre>
4	20	100	<pre>In [21]: # calculate MSE mse = mean_squared_error(inv_y, inv_yhat) print('Test MSE: %.3f' % mse)  Test MSE: 4959.099</pre>

No	Jumlah Neuron	Epoch	Output MSE
5		500	<pre>In [21]: # calculate MSE mse = mean_squared_error(inv_y, inv_yhat) print('Test MSE: %.3f' % mse)  Test MSE: 62.183</pre>
6		1000	<pre>In [21]: # calculate MSE mse = mean_squared_error(inv_y, inv_yhat) print('Test MSE: %.3f' % mse)  Test MSE: 49.014</pre>
7	30	100	<pre>In [21]: # calculate MSE mse = mean_squared_error(inv_y, inv_yhat) print('Test MSE: %.3f' % mse)  Test MSE: 1302.105</pre>
8		500	<pre>In [21]: # calculate MSE mse = mean_squared_error(inv_y, inv_yhat) print('Test MSE: %.3f' % mse)  Test MSE: 62.909</pre>
9		1000	<pre>In [21]: # calculate MSE mse = mean_squared_error(inv_y, inv_yhat) print('Test MSE: %.3f' % mse)  Test MSE: 73.327</pre>
10	40	100	<pre>In [21]: # calculate MSE mse = mean_squared_error(inv_y, inv_yhat) print('Test MSE: %.3f' % mse)  Test MSE: 2865.294</pre>
11		500	<pre>In [21]: # calculate MSE mse = mean_squared_error(inv_y, inv_yhat) print('Test MSE: %.3f' % mse)  Test MSE: 81.461</pre>
12		1000	<pre>In [21]: # calculate MSE mse = mean_squared_error(inv_y, inv_yhat) print('Test MSE: %.3f' % mse)  Test MSE: 60.642</pre>
13	50	100	<pre>In [21]: # calculate MSE mse = mean_squared_error(inv_y, inv_yhat) print('Test MSE: %.3f' % mse)  Test MSE: 2347.884</pre>
14		500	<pre>In [21]: # calculate MSE mse = mean_squared_error(inv_y, inv_yhat) print('Test MSE: %.3f' % mse)  Test MSE: 45.449</pre>

No	Jumlah Neuron	Epoch	Output MSE
15		1000	<pre>In [21]: # calculate MSE mse = mean_squared_error(inv_y, inv_yhat) print('Test MSE: %.3f' % mse)  Test MSE: 48.350</pre>

### Lampiran 3 Script LSTM dan Output

```
In [1]: import pandas as pd
import numpy as np
from datetime import datetime
from math import sqrt
from numpy import concatenate
from matplotlib import pyplot
from pandas import read_excel
from pandas import DataFrame
from pandas import concat
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
from keras.models import Sequential, load_model
from keras.layers import Dense
from keras.layers import LSTM
```

Using TensorFlow backend.

```
In [2]: # Load dataset
dataset = read_excel('FiksDataETH.xlsx', header=0, index_col=0)
dataset
```

Out[2]:

	Harga	Block Size	Average Block Usage	Block Time	Hashrate	Difficulty	Fe Minin Rewar
<b>Date</b>							
2018-01-08	1261.410000	32508.053959	96.069777	15.983948	1.692450e+14	2005041444599080	295
2018-01-09	1219.730000	33018.509630	94.935654	16.057048	1.713133e+14	2009431180358000	310
2018-01-10	1405.210000	30639.611479	95.130602	15.951206	1.747619e+14	2005236673361900	371
2018-01-11	1149.060000	31294.118706	94.129407	15.906951	1.799724e+14	2075219093758190	313
2018-01-12	1205.840000	30663.092514	95.306888	15.964411	1.807868e+14	2099459136321310	233
2018-01-13	1299.950000	31270.733740	95.950728	16.012752	1.814949e+14	2140480880488410	242
2018-01-14	1358.780000	29318.474309	95.016757	15.361800	1.858430e+14	2168670614790460	224
2018-01-15	1345.070000	31849.789483	96.913026	15.927085	1.819409e+14	2189067429035480	223
2018-01-16	1244.830000	32776.925732	95.950859	15.640224	1.874400e+14	2201579182658810	223
2018-01-17	987.103000	32423.171444	94.407280	15.505644	1.912308e+14	2256842549630730	207
2018-01-18	1004.160000	31697.840351	94.255242	15.523946	1.897529e+14	2275454971834870	215
2018-01-19	1025.690000	28845.749298	89.412498	15.207799	1.890921e+14	2268475042749960	187
2018-01-20	1076.940000	27413.709864	89.882551	14.723426	1.968503e+14	2362531144033200	177
2018-01-21	1126.830000	26812.395089	88.495848	14.873818	1.928699e+14	2372769489516330	132
2018-01-22	1069.170000	25874.352402	90.087897	14.690205	1.972404e+14	2394824553778490	119
2018-01-23	996.166000	25080.400813	90.275345	14.664292	1.967013e+14	2399066993860960	144
2018-01-24	970.868000	24560.163929	86.245099	14.491284	2.018521e+14	2459805089781100	129
...	...	...	...	...	...	...	...
2019-12-27	125.258299	24642.250897	81.354952	17.183340	1.510904e+14	2492213535891540	35
2019-12-28	126.149214	21572.407510	72.611728	17.035771	1.541563e+14	2529264305563140	30

2019-12-29	128.011145	21074.223666	73.965369	17.157705	1.538897e+14	2559397542720900	29
2019-12-30	134.768035	24644.704061	77.466685	17.230398	1.534011e+14	2547838767138330	35
2019-12-31	132.073074	24268.710239	77.968397	17.333266	1.512449e+14	2525619960817370	37
2020-01-01	128.722604	20542.127232	75.161371	17.343344	1.458488e+14	2455443582875460	29
2020-01-02	129.887363	20294.849418	67.012202	15.265891	1.512233e+14	2225084343566820	28
2020-01-03	127.055601	19477.352456	65.687873	13.263526	1.555698e+14	1982189874524830	33
2020-01-04	132.970833	19065.550787	59.688509	13.155912	1.637387e+14	2076779948066590	27
2020-01-05	133.820621	18451.629282	59.941169	13.319725	1.585709e+14	2041502819814660	26
2020-01-06	134.834252	20323.881132	66.940578	13.113025	1.612978e+14	2031210177953050	35
2020-01-07	143.131601	23159.997188	72.654106	13.252109	8.219488e+13	2109815163785040	19

730 rows × 10 columns

```
In [3]: values = dataset.values
        values
```

```
Out[3]: array([[1.26141000e+03, 3.25080540e+04, 9.60697767e+01, ...,
                1.58266275e+02, 2.08982210e+07, 1.22081500e+06],
               [1.21973000e+03, 3.30185096e+04, 9.49356539e+01, ...,
                1.68743743e+02, 2.11862110e+07, 1.26068100e+06],
               [1.40521000e+03, 3.06396115e+04, 9.51306024e+01, ...,
                1.86051596e+02, 2.14947970e+07, 1.18216200e+06],
               ...,
               [1.33820621e+02, 1.84516293e+04, 5.99411688e+01, ...,
                1.16823438e+01, 8.21506810e+07, 5.19643000e+05],
               [1.34834252e+02, 2.03238811e+04, 6.69405779e+01, ...,
                1.21651202e+01, 8.22185510e+07, 6.21243000e+05],
               [1.43131601e+02, 2.31599972e+04, 7.26541064e+01, ...,
                1.20085935e+01, 8.22555120e+07, 3.69130000e+05]])
```

```
In [4]: # convert series to supervised learning
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    df = DataFrame(data)
    cols, names = list(), list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
        names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
        if i == 0:
            names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
        else:
            names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
    # put it all together
    agg = concat(cols, axis=1)
    agg.columns = names
    # drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)
    return agg
```

```
In [5]: # ensure all data is float
values = values.astype('float32')
```

```
In [6]: # normalize features
scaler = MinMaxScaler(feature_range=(0, 1))
scaled = scaler.fit_transform(values)
```

```
In [7]: # frame as supervised learning
reframed = series_to_supervised(scaled, 1, 1)
reframed
```

Out[7]:

	var1(t-1)	var2(t-1)	var3(t-1)	var4(t-1)	var5(t-1)	var6(t-1)	var7(t-1)	var8(t-1)	var9(t-1)
1	0.891230	0.827616	0.977680	0.370903	0.413538	0.152949	0.494888	0.843330	0.000000
2	0.859704	0.850116	0.947660	0.380347	0.423363	0.155272	0.520897	0.902408	0.004694
3	1.000000	0.745258	0.952820	0.366673	0.439746	0.153052	0.629776	1.000000	0.009723
4	0.806249	0.774107	0.926320	0.360955	0.464499	0.190087	0.527354	0.875089	0.014628
5	0.849198	0.746293	0.957487	0.368379	0.468368	0.202915	0.382780	0.838627	0.019035
6	0.920382	0.773076	0.974529	0.374624	0.471732	0.224624	0.398386	0.920379	0.023133
7	0.964881	0.687024	0.949807	0.290526	0.492388	0.239542	0.367534	0.897193	0.026869
8	0.954510	0.798600	1.000000	0.363557	0.473850	0.250336	0.364664	0.875758	0.030836
9	0.878689	0.839467	0.974532	0.326496	0.499974	0.256958	0.365202	0.731866	0.035401
10	0.683746	0.823874	0.933675	0.309109	0.517983	0.286204	0.336143	0.586336	0.039850

```

718 0.031908 0.489144 0.602979 0.537983 0.293027 0.408636 0.028161 0.001749 0.985637
719 0.031851 0.480903 0.588189 0.525856 0.327293 0.410764 0.027982 0.001322 0.987043
720 0.032525 0.345589 0.356762 0.506791 0.341858 0.430371 0.019013 0.001307 0.988313
721 0.033933 0.323630 0.392592 0.522544 0.340591 0.446318 0.016882 0.001819 0.990221
722 0.039044 0.481011 0.485269 0.531935 0.338270 0.440201 0.027265 0.003229 0.991404
723 0.037006 0.464438 0.498549 0.545225 0.328027 0.428442 0.031031 0.002827 0.992439
724 0.034471 0.300176 0.424249 0.546527 0.302392 0.391305 0.017758 0.004018 0.993845
725 0.035352 0.289276 0.208547 0.278135 0.327924 0.289397 0.015964 0.008393 0.994802
726 0.033211 0.253242 0.173493 0.019444 0.348573 0.140855 0.025112 0.016457 0.998063
727 0.037685 0.235090 0.014694 0.005541 0.387380 0.190913 0.013453 0.014501 0.997235
728 0.038328 0.208030 0.021381 0.026704 0.362830 0.172244 0.011839 0.016801 0.998291
729 0.039094 0.290556 0.206651 0.000000 0.375784 0.166797 0.028341 0.019523 0.999398

```

729 rows × 20 columns

```

In [8]: # drop columns we don't want to predict
reframed.drop(reframed.columns[[11,12,13,14,15,16,17,18,19]], axis=1, inplace=
True)
print(reframed.head())

```

```

   var1(t-1)  var2(t-1)  var3(t-1)  var4(t-1)  var5(t-1)  var6(t-1)  \
1  0.891230  0.827616  0.977680  0.370903  0.413538  0.152949
2  0.859704  0.850116  0.947660  0.380347  0.423363  0.155272
3  1.000000  0.745258  0.952820  0.366673  0.439746  0.153052
4  0.806249  0.774107  0.926320  0.360955  0.464499  0.190087
5  0.849198  0.746293  0.957487  0.368379  0.468368  0.202915

   var7(t-1)  var8(t-1)  var9(t-1)  var10(t-1)  var1(t)
1  0.494888  0.843330  0.000000  0.934044  0.859704
2  0.520897  0.902408  0.004694  0.977572  1.000000
3  0.629776  1.000000  0.009723  0.891840  0.806249
4  0.527354  0.875089  0.014628  0.885350  0.849198
5  0.382780  0.838627  0.019035  0.826865  0.920382

```



```
In [9]: # split into train and test sets
values = reframed.values
n_train = 583
train = values[:n_train, :]
test = values[n_train:, :]
```

```
In [10]: # split into input and outputs
train_X, train_y = train[:, :-1], train[:, -1]
test_X, test_y = test[:, :-1], test[:, -1]
```

```
In [11]: # reshape input to be 3D [samples, timesteps, features]
train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))
print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)
```

```
In [12]: # design network
model = Sequential()
model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])
```

```
In [13]: # fit network
history = model.fit(train_X, train_y, epochs=500, batch_size=72, validation_data=(test_X, test_y), verbose=2, shuffle=False)
model.save('./savedModel')
```

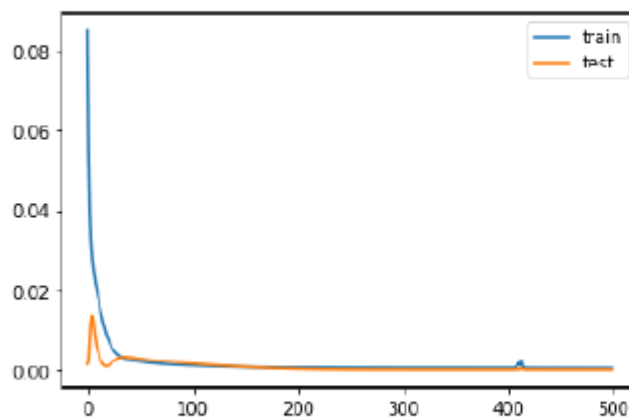
```
Train on 583 samples, validate on 146 samples
Epoch 1/500
- 1s - loss: 0.0856 - accuracy: 0.0017 - val_loss: 0.0013 - val_accuracy: 0.0000e+00
Epoch 2/500
- 0s - loss: 0.0555 - accuracy: 0.0017 - val_loss: 0.0018 - val_accuracy: 0.0000e+00
Epoch 3/500
- 0s - loss: 0.0399 - accuracy: 0.0017 - val_loss: 0.0072 - val_accuracy: 0.0000e+00
Epoch 4/500
- 0s - loss: 0.0322 - accuracy: 0.0017 - val_loss: 0.0120 - val_accuracy: 0.0000e+00
Epoch 5/500
- 0s - loss: 0.0282 - accuracy: 0.0017 - val_loss: 0.0134 - val_accuracy: 0.0000e+00
Epoch 6/500
- 0s - loss: 0.0256 - accuracy: 0.0017 - val_loss: 0.0121 - val_accuracy: 0.0000e+00
Epoch 7/500
- 0s - loss: 0.0234 - accuracy: 0.0017 - val_loss: 0.0098 - val_accuracy: 0.0000e+00
```

.....  
.....



```
Epoch 495/500
- 0s - loss: 3.6598e-04 - accuracy: 0.0017 - val_loss: 2.7950e-05 - val_accuracy: 0.0000e+00
Epoch 496/500
- 0s - loss: 3.6567e-04 - accuracy: 0.0017 - val_loss: 2.7827e-05 - val_accuracy: 0.0000e+00
Epoch 497/500
- 0s - loss: 3.6536e-04 - accuracy: 0.0017 - val_loss: 2.7732e-05 - val_accuracy: 0.0000e+00
Epoch 498/500
- 0s - loss: 3.6511e-04 - accuracy: 0.0017 - val_loss: 2.8004e-05 - val_accuracy: 0.0000e+00
Epoch 499/500
- 0s - loss: 3.6489e-04 - accuracy: 0.0017 - val_loss: 2.8132e-05 - val_accuracy: 0.0000e+00
Epoch 500/500
- 0s - loss: 3.6448e-04 - accuracy: 0.0017 - val_loss: 2.7567e-05 - val_accuracy: 0.0000e+00
```

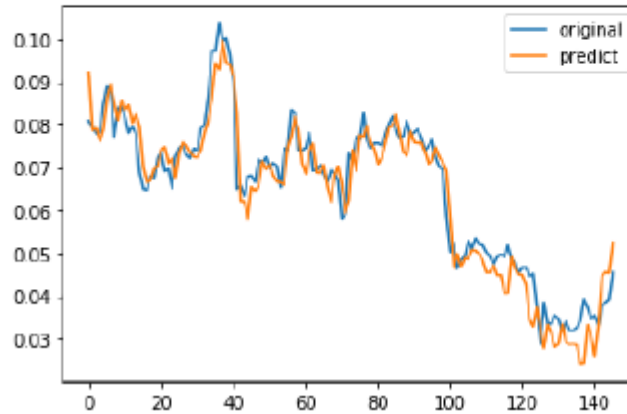
```
In [14]: # plot history
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()
```



```
In [15]: # make a prediction
yhat = model.predict(test_X)
test_X = test_X.reshape((test_X.shape[0], test_X.shape[2]))
```

```
In [16]: pyplot.plot(test_y, label="original")
pyplot.plot(yhat, label="predict")
pyplot.legend(loc="best")
```

Out[16]: <matplotlib.legend.Legend at 0xb6a814c908>



```
In [17]: print(yhat)
```

```
[[0.09233867]
 [0.07850576]
 [0.079004 ]
 [0.07588755]
 [0.07862947]
 [0.08604328]
 [0.08922783]
 [0.08502841]
 [0.08078267]
 [0.08544707]
 [0.08331907]
 [0.0843089 ]
 [0.08018018]
 [0.08220502]
 [0.07935438]
 [0.06982663]
 [0.06638048]
 [0.06675933]
 [0.06938022]
 [0.07021905]
 :
```

```

-----
[0.02864398]
[0.02837124]
[0.02410148]
[0.02421635]
[0.03309286]
[0.03071633]
[0.02578843]
[0.03292478]
[0.04466757]
[0.04539625]
[0.04527096]
[0.05241586]

```

```

In [18]: # invert scaling for predict
inv_yhat = concatenate((yhat, test_X[:, 1:]), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat)
inv_yhat = inv_yhat[:,0]

```

```

In [19]: print("Price predict: ")
print(inv_yhat)

```

```

Price predict:
[205.22658 186.93863 187.59734 183.47719 187.10217 196.9037
 201.11385 195.56195 189.94884 196.11545 193.3021 194.61072
 189.15231 191.82927 188.06053 175.46428 170.90826 171.40915
 174.8741 175.98308 180.78978 181.70953 176.89558 178.29951
 172.1795 180.1092 183.10934 182.21031 180.24301 178.73022
 178.55075 180.79823 186.21489 189.83734 196.22551 208.20767
 206.17801 214.8762 208.45903 207.78871 203.90546 192.44722
 165.21 165.01514 159.41656 169.51794 168.38066 169.81271
 177.36354 174.93242 176.66008 173.28667 171.17323 170.87224
 170.1271 181.20233 184.97835 190.96269 187.21954 176.79024
 173.64354 180.70287 182.70671 173.94315 173.83156 176.32433

 169.77766 165.00131 172.82716 175.91376 170.15883 161.53111
 165.53813 180.88666 174.41869 185.33623 184.92728 188.45193
 183.23895 181.0743 176.66136 178.32893 182.71255 187.78433
 188.56166 191.67644 186.05371 180.2334 179.05345 186.44862
 183.11954 182.91692 183.0874 180.39037 176.6929 177.42946
 181.57927 178.54494 177.25916 174.35641 159.76015 144.37791
 148.92424 144.8426 147.46458 147.26205 150.03001 149.88182
 148.77132 146.905 143.19107 143.20244 145.39111 142.36263
 142.30995 136.52843 136.5655 147.79533 143.9935 142.64001
 142.30524 139.12843 128.51682 126.51062 132.18594 125.59606
 119.75134 126.9642 124.35819 120.10519 120.95187 127.05367
 121.69799 120.878845 121.01831 120.65774 115.012856 115.16473
 126.90002 123.758095 117.24311 126.677795 142.20247 143.16585
 143.0002 152.44618 ]

```

```

In [20]: # invert scaling for actual
test_y = test_y.reshape((len(test_y), 1))
inv_y = concatenate((test_y, test_X[:, 1:]), axis=1)
inv_y = scaler.inverse_transform(inv_y)
inv_y = inv_y[:,0]

```

```
In [21]: print("Price data aktual: ")
print(inv_y)
```

```
Price data aktual:
[189.68762 187.92783 185.93251 186.37363 194.77446 200.65839
198.13786 184.89603 193.73198 193.9669 190.79851 186.11194
188.07095 185.75099 174.05902 168.96144 168.6899 172.84558
172.01442 178.40596 180.39723 174.36972 174.98471 169.17091
178.64523 181.44545 182.27177 179.40338 178.34366 181.28731
180.79585 187.49625 188.72102 197.7619 211.84465 211.76866
220.45088 214.54721 215.69041 210.6311 201.73451 168.99927
169.94382 166.6987 172.73892 172.99756 171.00732 177.91689
176.93129 178.61226 175.1318 176.8659 176.06046 169.46971
181.40286 180.78917 193.15123 192.1982 181.00853 180.79134
181.53667 186.12383 174.53 174.75409 176.3901 173.75008
171.15852 174.66939 173.76582 171.30185 159.5431 161.06876
179.98045 177.41394 184.9333 183.9328 192.3885 183.36522
181.5668 182.87398 182.8533 182.24382 185.3687 188.8572
191.05861 187.07494 184.46948 185.1916 189.72275 185.35951
186.32516 187.25111 184.80746 180.72498 182.10675 184.19214
178.42216 175.59369 174.77966 160.51796 149.52454 152.13625
144.31529 146.98181 148.00688 152.95728 150.41197 153.67381
151.65543 151.5529 149.45465 147.81133 145.60333 147.82701
148.6504 147.87125 151.41068 147.15953 145.79732 143.0108
144.5101 144.48558 142.05554 142.68193 132.34358 120.879814
133.87784 127.27368 127.375275 129.87083 128.91061 125.013824
128.00565 125.33354 125.2583 126.14921 128.01114 134.76804
132.07307 128.72261 129.88736 127.0556 132.97084 133.82062
134.83426 143.1316 ]
```

```
In [22]: mape=np.mean(np.abs((inv_y-inv_yhat)/inv_y))**100
mape
```

```
Out[22]: 1.6868886949611747e-149
```

```
In [23]: units = int(int(model.layers[0].trainable_weights[0].shape[1])/4)
```

```
In [24]: W = model.layers[0].get_weights()[0]
b = model.layers[0].get_weights()[2]
```

```
In [25]: W_i = W[:, :units]
W_f = W[:, units: units * 2]
W_c = W[:, units * 2: units * 3]
W_o = W[:, units * 3:]
```

In [26]: `W_i[0]`

```
Out[26]: array([ 0.04137211, -0.3269089 , -0.0932183 , -0.04260813,  0.09186599,
                -0.01484952,  0.00335789, -0.05930417, -0.03215478,  0.14579527,
                -0.15690951,  0.10897666,  0.16906671,  0.05078574,  0.21529323,
                -0.01014763, -0.01565879,  0.01974018,  0.16075662,  0.12433653,
                -0.02449567,  0.04801587, -0.22152877, -0.00338336, -0.19439912,
                 0.01109931,  0.03926298,  0.18273611, -0.27153018, -0.12701054,
                -0.24727367, -0.596266 , -0.05463586, -0.0967299 , -0.04921798,
                -0.06819604,  0.21947128,  0.27735054,  0.09617663,  0.05920425,
                -0.27996507,  0.05379887,  0.19067082, -0.08303878,  0.06120738,
                 0.04669481,  0.20326303, -0.03799409, -0.1799614 ,  0.09892186],
                dtype=float32)
```

In [27]: `W_f[0]`

```
Out[27]: array([ 0.01314129,  0.10163048,  0.1223585 ,  0.04357025,  0.00389226,
                 0.16125572, -0.02808368, -0.069257 ,  0.11343402,  0.1467818 ,
                 0.08740884, -0.10587753,  0.12328762,  0.0387512 ,  0.01009312,
                -0.13373719,  0.14790389, -0.10399911,  0.01423633, -0.10066371,
                -0.10919246,  0.1615569 , -0.11019754,  0.04213645,  0.13865238,
                 0.15088421, -0.09544336, -0.1654825 , -0.01250237,  0.05355339,
                -0.02542089, -0.07596826, -0.12203111, -0.10759126,  0.10467523,
                 0.02555542, -0.06051321, -0.00302392, -0.01208712, -0.06301242,
                 0.15452841, -0.02877028, -0.09334952, -0.04117413,  0.09503075,
                -0.14011827, -0.14858934, -0.02185668,  0.02162004,  0.02042618],
                dtype=float32)
```

In [28]: `W_c[0]`

```
Out[28]: array([ 0.16894172, -0.26389325,  0.09045851,  0.15400624, -0.18527617,
                 0.16784856,  0.1968077 , -0.09361622, -0.15257116,  0.16573514,
                -0.23481572,  0.2159641 ,  0.02246537,  0.18755306, -0.03905859,
                -0.00260331, -0.25798953,  0.19815311,  0.2560286 ,  0.09068008,
                -0.02446297, -0.06482977, -0.36104625, -0.2837686 , -0.23980175,
                -0.02904546, -0.10336313,  0.2436763 ,  0.20970415, -0.08752364,
                -0.3204812 , -0.28539947, -0.0151619 , -0.02522712,  0.01256865,
                 0.19163364, -0.23023376, -0.09350353, -0.03624837,  0.52276903,
                -0.2627832 ,  0.00856243,  0.05155641, -0.05910116, -0.0768264 ,
                -0.1540255 ,  0.17212872,  0.17155644, -0.2321383 , -0.16890171],
                dtype=float32)
```

In [29]: `W_o[0]`

```
Out[29]: array([-0.13565929,  0.02533614,  0.03979781,  0.03755527, -0.20824978,
                -0.03602203, -0.14087728,  0.13040479,  0.10668406,  0.08961078,
                -0.01848932,  0.07352116, -0.06958553,  0.10305837,  0.02653852,
                -0.02596452,  0.19271825, -0.25303814,  0.04781611,  0.1519498 ,
                -0.01870206, -0.16226214, -0.22602814, -0.01734742, -0.3656995 ,
                 0.10826843,  0.15363248,  0.07809943, -0.1753169 , -0.09955008,
                 0.04054255, -0.24131292,  0.00878605, -0.0592967 ,  0.05203234,
                -0.06687432,  0.13607286, -0.05322987,  0.12544833, -0.42939028,
                -0.11093324, -0.04227012, -0.04344539,  0.01900772,  0.07632752,
                -0.04383947, -0.05674011, -0.16383064, -0.18063852, -0.10211511],
                dtype=float32)
```



```
In [30]: b_i = b[:units]
b_f = b[units: units * 2]
b_c = b[units * 2: units * 3]
b_o = b[units * 3:]
```

```
In [31]: b_i
```

```
Out[31]: array([[ 0.05499813,  0.11477534, -0.01445123,  0.05280429,  0.15998179,
  0.00970906,  0.1752284 ,  0.02354815,  0.07711723,  0.07164802,
  0.17312759,  0.03491374,  0.06991372,  0.01507865,  0.00145572,
 -0.04909939,  0.06123362,  0.06332818,  0.0829168 ,  0.01016528,
  0.02355498,  0.05519648,  0.27413538,  0.15450196,  0.08322424,
 -0.00591292,  0.08689663,  0.05685337,  0.09586519,  0.00458194,
  0.13517548,  0.37165222,  0.02682636,  0.00154733, -0.02973573,
  0.06730992,  0.0969284 ,  0.09792693,  0.06316678,  0.38903704,
  0.01967352,  0.0344969 ,  0.03642342, -0.02628577,  0.01327802,
  0.33697507,  0.08523975, -0.00205321,  0.13330324,  0.02004351],
 dtype=float32)
```

```
In [32]: b_f
```

```
Out[32]: array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
  1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
  1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
  1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
 dtype=float32)
```

```
In [33]: b_c
```

```
Out[33]: array([-0.00662116, -0.0018823 , -0.00038865,  0.00125452, -0.00971203,
 -0.00154688, -0.00665545,  0.00109587, -0.00023018,  0.00870121,
  0.01759261,  0.00401628,  0.00221441,  0.01050818,  0.01180183,
 -0.00222267, -0.00275685,  0.00829769,  0.00057293,  0.01630913,
  0.01544386,  0.00537859,  0.0338867 ,  0.01240178,  0.00378604,
  0.00750727,  0.0032934 , -0.00655983, -0.014582 ,  0.00599307,
  0.04922644,  0.00962272, -0.00249771, -0.00888808,  0.01911578,
 -0.00537935, -0.00195946,  0.00049039, -0.00088185, -0.02784163,
  0.00258407,  0.00144062, -0.00640112, -0.01131438,  0.00732941,
  0.08368523,  0.00771657, -0.00889259,  0.01046761,  0.00419811],
 dtype=float32)
```

```
In [34]: b_o
```

```
Out[34]: array([[ 0.0458709 ,  0.13608906, -0.02394633,  0.05580169,  0.1283779 ,
  0.00923728,  0.17136471,  0.01747411,  0.07890861,  0.07603615,
  0.19307187,  0.0346565 ,  0.06893215,  0.00818658, -0.00457056,
 -0.04148333,  0.06671921,  0.04171071,  0.09753298,  0.00412206,
  0.0267511 ,  0.04196123,  0.24482429,  0.15579219,  0.04296616,
 -0.01415157,  0.0914549 ,  0.06019245,  0.08127557,  0.00129738,
  0.17454587,  0.43173498,  0.02116197, -0.0013472 , -0.03156053,
  0.05840117,  0.08599425,  0.09643751,  0.05622401,  0.26779225,
  0.01771018,  0.02558538,  0.02528858, -0.02681425,  0.01527463,
  0.32267815,  0.06916632, -0.00106142,  0.13086592,  0.03455272],
 dtype=float32)
```