TUGAS AKHIR – IF184802

Penerapan Metode *Deep Learning Long Short-Term Memory Network* Pada Prediksi Data *Time-Series*

MUHAMMAD AUFA WIBOWO
05111640000184

Dosen Pembimbing I :
Rully Soelaiman,S.Kom., M.Kom.

Dosen Pembimbing II :
Ir. F.X. ARUNANTO, M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya
2020

TUGAS AKHIR – IF184802

Penerapan Metode *Deep Learning Long Short-Term Memory Network* Pada Prediksi Data *Time-Series*

MUHAMMAD AUFA WIBOWO
05111640000184

Dosen Pembimbing I :
Rully Soelaiman,S.Kom., M.Kom.

Dosen Pembimbing II :
Ir. F.X. ARUNANTO, M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya
2020

[*Halaman ini sengaja dikosongkan*]

FINAL PROJECT – IF184802

Application of Deep Learning Long Short-Term Memory Network Method in Time-Series Data Prediction

MUHAMMAD AUFA WIBOWO
05111640000184

Supervisor I
Rully Soelaiman, S.Kom., M.Kom.

Supervisor II
Ir. F.X. ARUNANTO, M.Sc.

DEPARTMENT OF INFORMATICS ENGINEERING
Faculty of Intelligent Electrical and Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya
2020

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

Penerapan Metode *Deep Learning Long Short-Term Memory Network* Pada Prediksi Data *Time-Series*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Algoritma dan Pemrograman
Program Studi S-1 Teknik Informatika
Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:
Muhammad Aufa Wibowo
NRP: 051116 40000 184

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Rully Soelaiman, S.Kom., M.Kom.
NIP. 197002131994021001

(Pembimbing 1)

Ir. F.X. ARUNANTO, M.Sc.
NIP. 195701011983031004

(Pembimbing 2)

SURABAYA
2020

[*Halaman ini sengaja dikosongkan*]

PENERAPAN METODE DEEP LEARNING LONG SHORT-TERM MEMORY NETWORK PADA PREDIKSI DATA TIME-SERIES

Nama Mahasiswa : Muhammad Aufa Wibowo
NRP : 051116 40000 184
Departemen : Teknik Informatika Fakultas Teknologi Elektro dan Informatika Cerdas - ITS
Dosen Pembimbing 1 : Rully Soelaiman, S.Kom., M.Kom.
Dosen Pembimbing 2 : Ir. F.X. ARUNANTO, M.Sc.

Abstrak

Banyak contoh kasus permasalahan yang membutuhkan komponen waktu. Permasalah ini begitu banyak dibiarkan begitu saja karena komponen waktu membuat permasalahan time-series semakin sulit. Dalam dunia finansial, indeks saham merupakan acuan penting untuk setiap investor dalam menentukan kemana arah harga berikutnya.

Didalam tugas akhir ini, penerapan metode LSTM digunakan untuk melakukan prediksi data indeks saham pada kondisi stabilitas yang berbeda. Sebagai perbandingan, metode ARIMA dikenalkan untuk dilakukan perbandingan terhadap model LSTM.

Hasil eksperimen menunjukkan bahwa metode LSTM memiliki kelebihan dibanding dengan metode ARIMA dan tidak sensitif terhadap perbedaan stabilitas.

Kata kunci: Prediksi time-series, prediksi saham, Stasioneritas, LSTM, ARIMA

APPLICATION OF DEEP LEARNING LONG SHORT-TERM MEMORY NETWORK METHOD IN TIME-SERIES DATA PREDICTION

Student Name	: MUHAMMAD AUFA WIBOWO
Registration Number	: 051116 40000 184
Department	: Informatics Engineering Department Faculty of Intelligent and Informatics Technology - ITS
First Supervisor	: Rully Soelaiman, S.Kom., M.Kom.
Second Supervisor	: Ir. F.X. ARUNANTO, M.Sc.

Abstract

Many examples of problems that require time components. This problem is left out so much because the time component makes the time-series problem more difficult. In the financial world, the stock index is an important reference for every investor in determining where the next price will go.

In this final project, the application of the LSTM method is used to predict stock index data under different stability conditions. As a comparison, the ARIMA method was introduced to compare the LSTM model.

The experimental results show that the LSTM method has advantages compared to the ARIMA method and is not sensitive to differences in stability..

Keywords: Time series prediction, stock prediction, Stationerity, LSTM, ARIMA

KATA PENGANTAR

Puji syukur penulis ucapkan kepada Tuhan Yang Maha Esa atas pimpinan, penyertaan, dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

PENERAPAN METODE DEEP LEARNING LONG SHORT-TERM MEMORY NETWORK PADA PREDIKSI DATA TIME-SERIES

Pengerjaan Tugas Akhir ini dilakukan untuk memenuhi salah satu syarat meraih gelar Sarjana di Departemen Informatika Fakultas Teknologi Informasi dan Komunikasi Institut Teknologi Sepuluh Nopember.

Dengan selesainya Tugas Akhir ini diharapkan apa yang telah dikerjakan penulis dapat memeberikan manfaat bagi perkembangan ilmu pengetahuan terutama di bidang teknologi informasi serta bagi diri penulis sendiri selaku peneliti.

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan baik secara langsung maupun tidak langsung selama penulis mengerjakan Tugas Akhir maupun selama menempuh masa studi antara lain:

1. Bapak Rully Soelaiman, S.Kom., M.Kom. selaku Dosen Pembimbing yang telah membimbing saya selama masa kuliah maupun selama penyelesaian Tugas Akhir ini, Dosen yang paling perhatian kepada saya dalam memberi ilmu, nasihat, dan motivasi selama berada menempuh kuliah di Departemen Informatika ITS.
2. Bapak Ir. F.X. ARUNANTO, M.Sc. selaku dosen pembimbing yang telah memberikan ilmu, dan masukan kepada penulis.
3. Bapak, Ibu, dan keluarga penulis yang selalu memberikan dukungan, perhatian, dan kasih sayang bagi penulis yang menjadi semangat selama perkuliahan maupun pengerjaan Tugas Akhir.
4. Ivanda Zevi Amalia yang telah membantu mengoreksi teknis dari tugas akhir ini.

5. Teman-teman angkatan 2016 Departemen Informatika ITS yang telah menemani perjuangan penulis selama 4 tahun masa perkuliahan.
6. Serta pihak-pihak lain yang tidak dapat disebutkan disini yang telah banyak membantu penulis dalam penyusunan Tugas Akhir ini.

Penulis mohon maaf apabila masih ada kekurangan pada Tugas Akhir ini. Penulis juga mengharapkan kritik dan saran yang membangun untuk pembelajaran dan perbaikan di kemudian hari. Semoga melalui Tugas Akhir ini penulis dapat memberikan kontribusi dan manfaat yang sebaik-baiknya.

Surabaya, Juni 2020

Muhammad Aufa Wibowo

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

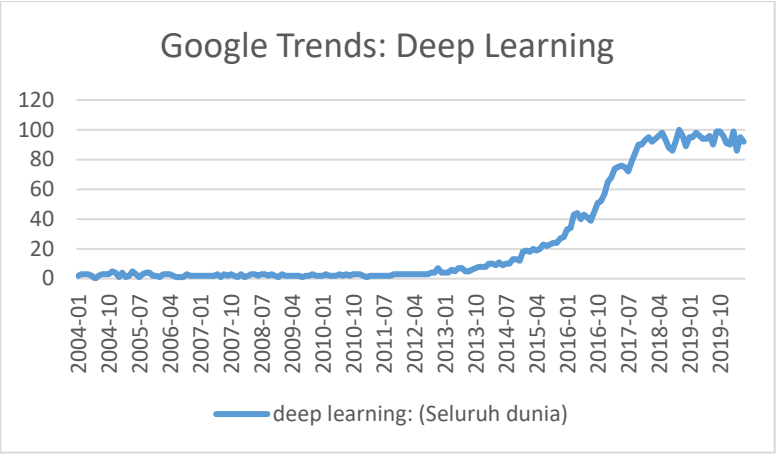
Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan masalah, batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan buku Tugas Akhir ini.

1.1 Latar Belakang

Time-series yang bersifat stasioner berarti data deret waktu (time-series) yang cenderung memiliki karakter statistik yang konstan. Karakter statistik yang konstan ini tidak akan berubah seiring dengan perubahan waktu (semisal terhadap rata-rata dan kovarian) [1]. Deret yang stabil memiliki korelasi jangka pendek, menunjukkan bahwa hanya deret baru yang memiliki efek signifikan pada nilai sekarang, dan semakin jauh intervalnya, semakin kecil pengaruh nilai masa lalu pada nilai saat ini.

Prediksi harga saham merupakan hal yang paling umum dalam kasus penerapan prediksi *time-series*. Banyak peneliti yang mencoba menyelesaikan kasus *time-series* menggunakan metode *deep learning* [2]. *Deep Learning* yang merupakan bagian dari *machine learning*, dalam bahasa Indonesia disebut kecerdasan buatan, adalah metode, bagian dari kecerdasan buatan, yang dikembangkan meniru jaringan saraf tiruan (atau yang lebih dikenal dengan *neural network*) [3].

Dalam beberapa dekade kebelakang ini topik *Deep Learning*, mengutip data Google Trend, sangat sering dibahas di dalam rumpun ilmu komputer maupun diluar rumpun ilmu komputer yang biasa dikonsumsi secara umum melalui media. Sebagai contoh penerapan *Deep Learning* meliputi *automatic speech recognition*, *image recognition*, *natural language processing*, dan banyak lainnya.



Gambar 1.6.1.1 Tren pencarian kata 'Deep Learning' oleh mesin pencarian Google

Jaringan *long short-term memory* atau yang selanjutnya disingkat LSTM adalah salah satu contoh metode *Deep Learning* yang berdasar pada *artificial neural networks* dengan kemampuan yang memungkinkan jaringan tersebut untuk belajar.

Sebagai contoh kasus, penerapan model LSTM berdasar pembobotan waktu untuk mendefinisikan ulang prediksi tren dalam saham yang diajukan di [4]. Kemudian terdapat kasus perbandingan model *bidirectional* LSTM dan *stacked LSTM* dengan LSTM sederhana untuk prediksi harga saham, dan mendapatkan hasil bahwa performa *bidirectional* LSTM adalah yang terbaik [5].

1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut.

1. Bagaimana model LSTM dapat digunakan untuk memprediksi data *time-series*?

2. Apakah prediksi kasus *time-series* dengan menggunakan model LSTM bersifat lebih baik dibanding dengan menggunakan pendekatan statistika klasik, ARIMA?
3. Bagaimana pengaruh kondisi perbedaan stabilitas pada karakter data *time-series* berpengaruh terhadap performa model prediksi pada kedua model, baik LSTM dan ARIMA?

1.3 Batasan Permasalahan

Permasalahan pada pembuatan Tugas Akhir ini memiliki beberapa Batasan. Batasan-batasan tersebut adalah sebagai berikut.

- Implementasi algoritma menggunakan bahasa pemrograman Python 3.
- Data diambil dari *investing.com* dan *Kaggle.com*.
- Data *time-series* yang digunakan adalah data indeks harga saham.
- Model ARIMA menggunakan perangkat lunak dari *library* stasmodel.
- Model LSTM menggunakan perangkat lunak dari library keras.
- Pemodelan latih, pemodelan uji, pemodelan evaluasi, dan kegiatan lainnya yang berkaitan dengan implementasi topik Tugas Akhir ini dilakukan didalam *platform Google Colab Pro* dengan ketentuan dan penggunaan produk yang ditetapkan oleh *Google Colab* per Mei 2020.

1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah sebagai berikut:

1. Mengetahui bagaimana LSTM dapat digunakan untuk memprediksi data time-series.
2. Mengevaluasi hasil kinerja model LSTM dan model ARIMA pada kasus prediksi data time-series.

3. Mengetahui pengaruh perbedaan stabilitas pada prediksi data time-series menggunakan model LSTM.

1.5 Manfaat

Tugas akhir ini diharapkan dapat menambah referensi dalam penerapan teknologi komputasi ke dalam dunia finansial dan dapat membantu memprediksi harga saham dan mengurangi potensi risiko yang dimiliki setiap investor.

1.6 Metodologi

Metodologi untuk pengerjaan Tugas Akhir dilakukan sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Proposal Tugas Akhir tersebut berisi pendahuluan, deskripsi dan gagasan metode-metode. Pendahuluan terdiri dari latar belakang Tugas Akhir, rumusan masalah dan Batasan masalah yang ditetapkan, serta manfaat dari hasil pembuatan Tugas Akhir. Selain itu, dijelaskan tinjauan Pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir.

1.6.2 Studi Literatur

Pada tahap ini dilakukan pencarian informasi dan studi literatur yang relevan untuk dijadikan referensi dalam melakukan pengerjaan Tugas Akhir. Literatur yang dipelajari pada pengerjaan tugas akhir berasal dari buku, artikel, dan jurnal ilmiah yang diolah oleh penulis dari berbagai sumber di internet.

1.6.3 Implementasi Perangkat Lunak

Tahapan ini merupakan tahapan untuk melakukan implementasi metode yang telah direncanakan. Implementasi ini

dilakukan dengan menggunakan bahasa pemrograman Python dengan perangkat lunak pendukung seperti *TensorFlow* dan *Keras*. Beberapa perangkat lunak lain yang digunakan akan dijelaskan secara detail di bagian tinjauan pustaka dalam buku ini.

1.6.4 Pengujian dan Evaluasi

Tahap pengujian dan evaluasi dilakukan menggunakan data time-series dengan ketentuan sebagai berikut:

1. Data SZSE 200, SZSE 300, dan CSI 300 diambil untuk periode tanggal mulai 5 Januari 2015 hingga tanggal12 Februari 2019 dengan pembagian data menggunakan metode *hold-out*, 70% data latih dan 30% data uji.
2. Data JKSE diambil untuk periode tanggal mulai 2 Januari 2015 hingga tanggal 15 Februari 2019 dengan pembagian data menggunakan metode *hold-out*, 70% data latih dan 30% data uji.
3. Seluruh data yang digunakan dievaluasi dengan menggunakan metode *Root Mean Square Error* dan *Mean Absolute Error*.

1.6.5 Penyusunan Buku

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan

Buku Tugas Akhir ini merupakan laporan secara lengkap mengenai Tugas Akhir yang telah dikerjakan baik dari sisi teori, rancangan, maupun implementasi sehingga memudahkan bagi pembaca dan juga pihak yang ingin mengembangkan lebih lanjut. Sistematika penulisan buku Tugas Akhir secara garis besar adalah sebagai berikut.

Bab I Pendahuluan

Bab ini berisi penjelasan latar belakang, rumusan masalah, batasan permasalahan, tujuan, dan manfaat Tugas Akhir.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir. Secara umum, bab ini berisi tentang *Deep Learning*, *Long Short-Term Memory*, stasioneritas, ARIMA, dan perangkat lunak yang digunakan.

Bab III Analisis dan Perancangan Sistem

Bab ini berisi penjelasan tentang rancangan dari *Long Short-Term Memory* sistem yang akan dibangun. Di bab ini juga dijelaskan mengenai rancangan model ARIMA sebagai pembanding.

Bab IV Implementasi

Bab ini berisi penjelasan implementasi dari rancangan yang telah dibuat pada bab III. Implementasi disajikan dalam bentuk cuplikan kode disertai dengan penjelasannya.

Bab V Pengujian dan Evaluasi

Bab ini berisi penjelasan mengenai data hasil percobaan dan pembahasan mengenai hasil percobaan yang telah dilakukan.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab terakhir yang menjelaskan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan tentang dasar teori yang digunakan sebagai teori pendukung tugas akhir ini. Teori-teori tersebut diantaranya adalah konsep kecerdasan buatan, *deep learning*, jaringan *long short-term memory*. Beberapa teori pendukung pembuatan tugas akhir seperti data deret waktu (*time series*), konsep stasioneritas, dan konsep saham dan pasar modal juga akan dibahas didalam bab ini. Penjelasan ini bertujuan untuk memberikan gambaran umum dan diharapkan dapat mendukung sistem yang dibangun

2.1 *Time-series* dan Kasus Peramalan

Banyak contoh kasus permasalahan yang membutuhkan komponen waktu. Permasalah ini begitu banyak dibiarkan begitu saja karena komponen waktu membuat permasalahan *time-series* semakin sulit.

Time-series secara struktur dapat dibentuk oleh kumpulan dari dua data yang terdiri dari data waktu dan data observasi seperti contoh sebagai berikut.

Tabel 2.1.1 Contoh data time-series

Tahun	Observasi
2010	123
2011	64
2012	92
2013	697

Berbeda dengan kasus prediksi lain, data *time-series* sangat erat kaitannya dengan dimensi waktu.

Terdapat perbedaan secara fundamental antara peramalan *time-series* dengan analisis *time-series*. Analisis *time-series* dibuat untuk menentukan karakter dari *time-series* tersebut seperti pola musiman, tren, atau hubungan dengan faktor yang lain. Sementara, berbeda dengan analisis *time-series*, kasus peramalan *time-series*

menggunakan informasi atau data yang ada (dengan tambahan informasi lain) untuk memprediksi nilai di masa yang akan datang [6] .

Analisis *time-series* menitikberatkan pada penggunaan statistik. Analisis ini berorientasi kepada "kenapa" dibalik sebuah data *time-series*. Kualitas dari analisis *time-series*, model deskriptif, ditentukan oleh bagaimana model tersebut mendeskripsikan data dan interpretasi yang disediakan untuk menceritakan dari domain masalah.

Lain halnya dengan analisis *time-series*, pada kasus peramalan data *time-series*, peramalan melibatkan pembuatan model yang sesuai dengan data masa lampau dan menggunakan model tersebut untuk memprediksi pengamatan di masa depan. Keahlian model peramalan *time-series* ditentukan oleh kinerjanya dalam memprediksi masa depan. Hal ini seringkali mengorbankan kemampuan untuk menjelaskan mengapa prediksi tertentu dibuat, *convidence interval* dan bahkan lebih memahami penyebab yang mendasari di balik masalah.

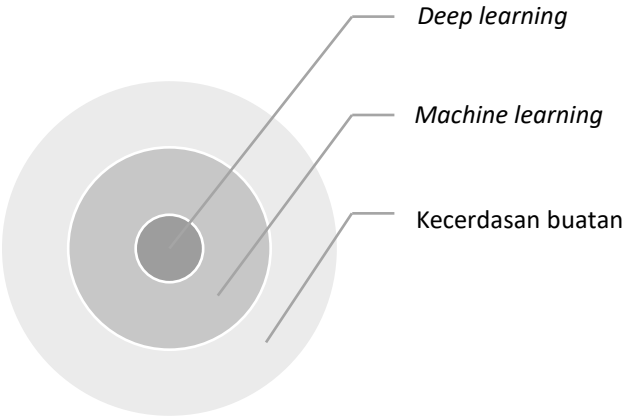
Terdapat beberapa komponen penting pembentuk *time-series*. Pertama adalah level, nilai datar apabila *time-series* tersebut berupa sebuah garis lurus. Kedua adalah *trend*, sebuah opsional dan paling sering berbentuk kurva *linear* yang meningkat atau menurun secara teratur berdasar waktu. Ketiga adalah musim (*seasonality*), sebuah pola tambahan yang berulang atau bersifat siklis dalam suatu periode waktu *time-series*. Keempat adalah *noise*, variabel tambahan dalam sebuah observasi yang tidak bisa diungkapkan oleh model.

Ada beberapa hal yang perlu diperhatikan dalam melakukan kasus prediksi. Pertama, jumlah data yang tersedia dan kemampuan untuk menggunakan data yang tersedia untuk melakukan pembuatan model. Semakin banyak data yang tersedia akan memberikan peluang yang semakin luas untuk melakukan *exploratory data analysis*, *model testing*, dan *tuning*. Kedua, periode waktu yang diambil dari data *time-series*. Periode waktu yang pendek mempermudah untuk melakukan prediksi dengan

tingkat kepercayaan yang tinggi, namun akan susah untuk melakukan keluar dari periode waktu yang diambil. Ketiga, apakah prediksi harus diperbarui secara berkala atau hanya butuh dibuat diawal dan statis.

2.2 *Deep Learning*

Deep Learning merupakan bagian dari kecerdasan buatan yang merupakan bagian dari *machine learning*. Salah satu metode *machine learning* yang berdasar pada arsitektur jaringan saraf tiruan otak manusia (atau yang lebih dikenal dan selanjutnya disebut dengan *artifical neural network*) untuk mengekstrak fitur dari suatu koleksi data yang massif disebut *Deep Learning* [7] .



Gambar 1.6.5.1 Hubungan antara kecerdasan buatan, machine learning, dan deep learning

Terdapat beberapa contoh arsitektur *Deep Learning* seperti *deep neural networks*, *deep belief networks*, *recurrent neural networks*, *convolutional neural networks*, dan *long-short term memory network* yang telah diterapkan untuk aplikasi praktis pada bidang-bidang seperti *computer vision*, pengenalan suara, *natural language processing*, pengenalan citra, analisis jejaring media

sosial, bioinformatika, pembuatan obat, analisis gambar medis, dan finansial dimana telah memberikan hasil yang dapat dibandingkan dan pada beberapa kasus contoh dapat mengungguli kemampuan manusia yang ahli dibidangnya [8] [9] .

Deep learning terkait erat dengan teori perkembangan otak yang diusulkan oleh para ilmuwan saraf kognitif pada awal 1990-an. Teori-teori perkembangan ini dipakai dalam model komputasi, menjadikannya teori ini pendahulu dari *deep learning*. Di satu sisi, beberapa varian dari algoritma *Backpropagation* telah diusulkan untuk meningkatkan realisme pemrosesan. Peneliti lain berpendapat bahwa bentuk pembelajaran mendalam yang tidak diawasi, seperti yang didasarkan pada model generatif hierarkis dan jaringan kepercayaan yang mendalam, mungkin lebih dekat dengan realitas biologis. Dalam hal ini, model jaringan saraf generatif telah dikaitkan dengan bukti neurobiologis tentang pemrosesan berbasis sampel di korteks serebral [10].

2.2.1 *Neural Network*

Deep learning tidak bisa terlepas dari *neural network* karena hampir semua model *deep learning* terbangun dari *neural network*. Sebelum *deep neural network* banyak digunakan seperti sekarang, *neural network* sederhana (*artificial neural network* atau yang selanjutnya disingkat dengan ANN) banyak digunakan untuk menerapkan proses pembelajaran [11]. *Neural network* sederhana ini meniru jaringan saraf biologis yang membentuk otak hewan dan melakukan proses pembelajaran layaknya jaringan saraf biologis hewan. Sayangnya proses ini sangat susah direpresentasikan kedalam bentuk algoritma konvensional karena belum ada pemahaman komprehensif tentang proses optimasi atau organisasi *internal neural network* (dan turunannya), dan hal tersebut sering dikritik karena dianggap sebagai "kotak hitam" (*black box*) yang misterius [12] [13].

ANN didasarkan pada kumpulan unit yang terhubung yang disebut *neuron* buatan. Setiap koneksi antara *neuron* dapat

mengirimkan sinyal ke *neuron* lain. *Neuron* penerima dapat memproses sinyal dan kemudian memberi sinyal *neuron* hilir yang terhubung dengannya. *Neuron* dan sinapsis mungkin juga memiliki bobot yang bervariasi ketika pembelajaran berlangsung, yang dapat meningkatkan atau mengurangi kekuatan sinyal yang dikirimnya ke hilir.

Biasanya, *neuron* diatur dalam lapisan. Tujuan awal dari pendekatan jaringan saraf adalah untuk memecahkan masalah dengan cara yang sama seperti otak manusia. Seiring waktu, perhatian difokuskan pada pencocokan kemampuan mental tertentu, yang mengarah ke penyimpangan dari biologi seperti *backpropagation*, atau meneruskan informasi ke arah sebaliknya dan menyesuaikan jaringan untuk mencerminkan informasi itu.

Turunan dari *neural network* sederhana adalah *deep neural network* dimana terdapat lebih banyak lapisan dibandingkan dengan *neural network* sederhana [3] [11] [14] .

2.2.2 *Deep Neural Network*

Deep Neural Network (atau yang selanjutnya disingkat dengan DNN) adalah jaringan saraf tiruan dengan lebih banyak lapisan antara lapisan *input* dan *output* dibandingkan dengan *neural network* sederhana. DNN melakukan manipulasi matematis, didalam *layer*, untuk mengubah *input* menjadi *output*, apakah itu merupakan hubungan *linear* atau hubungan *non-linear*. Jaringan tersebut bergerak melalui lapisan-lapisan yang menghitung probabilitas setiap keluaran. Sebagai contoh, DNN yang dilatih untuk mengenali ras anjing akan memeriksa gambar yang diberikan dan menghitung probabilitas bahwa anjing dalam gambar tersebut adalah jenis tertentu. DNN dapat memodelkan hubungan *non-linear* yang kompleks. Arsitektur DNN menghasilkan model komposisi di mana objek diekspresikan sebagai komposisi primitif. Lapisan tambahan memungkinkan komposisi fitur dari lapisan bawah, berpotensi memodelkan data

kompleks dengan unit lebih sedikit daripada jaringan dangkal yang melakukan hal yang sama.

Arsitektur yang dalam mencakup banyak varian dari beberapa pendekatan dasar. Setiap arsitektur telah menemukan kesuksesan di domain tertentu. Pada awalnya, DNN membuat peta *neuron* virtual dan menetapkan nilai numerik acak, atau bobot, untuk koneksi di antara mereka. Bobot dan *input* dikalikan dan mengembalikan *output* antara nol dan satu. Jika jaringan tidak secara akurat mengenali pola tertentu, suatu algoritma akan menyesuaikan bobot. Dengan cara itu algoritma dapat membuat parameter tertentu lebih berpengaruh, sampai menentukan manipulasi matematis yang benar untuk sepenuhnya memproses data.

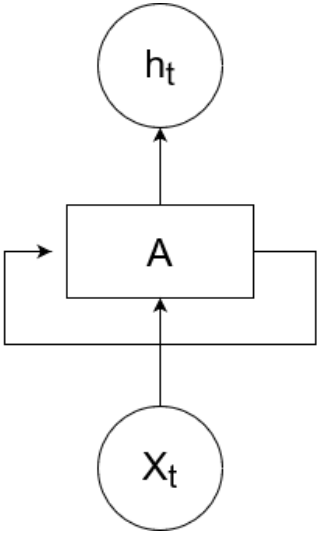
2.3 *Long Short-Term Memory*

Long short-term memory (LSTM) adalah arsitektur jaringan saraf tiruan berulang yang digunakan dalam bidang *deep learning*. Tidak seperti jaringan *neural network* standar, LSTM memiliki koneksi umpan balik. Misalnya, LSTM dapat diterapkan untuk tugas-tugas seperti pengenalan tulisan tangan yang tidak tersegmentasi dan terhubung, pengenalan ucapan manusia dan deteksi anomali dalam lalu lintas jaringan atau IDS (*intrusion detection systems*). Jaringan LSTM sangat cocok untuk mengklasifikasikan, memproses dan membuat prediksi berdasarkan data deret waktu, karena mungkin ada kelambatan durasi yang tidak diketahui antara peristiwa-peristiwa penting dalam deret waktu.

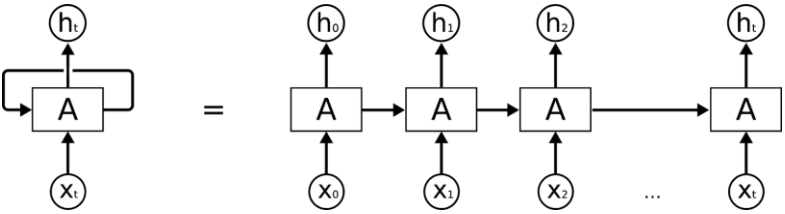
2.3.1 *Recurrent Neural Network* dan *Vanishing Gradient Problem*

Ide dibalik recurrent *neural network* atau yang selanjutnya disebut RNN adalah proses pembelajaran tidak selalu dimulai dengan kondisi hampa. Ada kemungkinan proses pembelajaran

tersebut mengambil hasil dari proses pembelajaran sebelumnya. Berbeda dengan neural network sederhana yang tidak bisa menyimpan proses pembelajaran tersebut, RNN memiliki memori untuk menyimpan proses pembelajran sebelumnya.



Gambar 2.3.1.1 Arsitektur Recurrent Neural Network



Gambar 2.3.1.2 Arsitektur RNN dalam telaah lebih lanjut

Arsitektur RNN yang serupa dengan rantai ini sangat cocok untuk menyelesaikan kasus permasalahan yang membutuhkan informasi yang berurutan. Menurut teori, RNN sangat cocok untuk

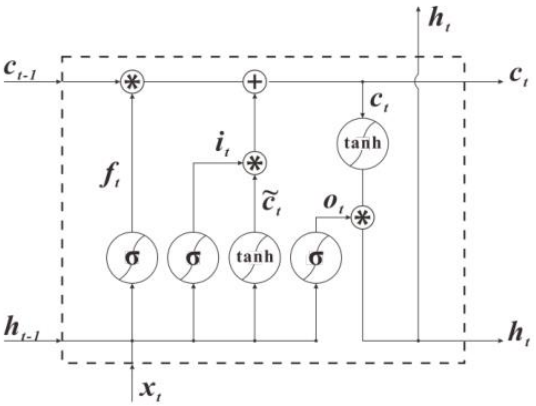
mengatasi permasalahan "*long-term dependencies*". Namun pada kenyataannya, RNN memiliki permasalahan ketika sebuah informasi dengan konteks yang lebih luas (pada kondisi tertentu dimana banyak data yang dihadapi oleh RNN tidak bisa dimanfaatkan lagi). [15]

Hal ini dikarenakan didalam proses pembelajaran kecerdasan buatan terdapat permasalahan yang disebut *vanishing gradient problem*. Permasalahan ini adalah sebuah hambatan yang ditemukan ketika melakukan proses pelatihan *artificial neural network* dengan proses pembelajaran dan *backpropagation* yang berdasar metode *gradient-based*. Didalam metode tersebut, bobot dari setiap *neural network* mendapatkan pembaharuan yang proporsional terhadap gradien dari fungsi kesalahan dengan mengacu kepada pembobotan yang sekarang dalam setiap proses iterasi pelatihan. Fungsi aktivasi sederhana seperti fungsi *hyperbolic tangent* memiliki rata-rata *gradien* pada rentang (-1,1) atau [0,1), dan proses *backpropagation* melakukan proses komputasi dengan menggunakan proses *chain rule*. Hal ini memiliki efek mengalikan sejumlah n dari angka-angka (yang bernilai kecil) untuk menghitung *gradien* dari lapisan luar sebuah jaringan *neural network* yang berarti *gradien* (*error signal*) berkurang secara eksponensial dengan n dan lapisan luar tersebut sebelumnya melatih dengan sangat lambat.

2.3.2 Arsitektur LSTM

Long Short Term Memory Network atau yang selanjutnya disingkat LSTM adalah versi pembaharuan dari RNN. Perbedaan dengan *neural network* tradisional adalah terdapat hubungan antara *hidden layer* RNN. Oleh karena itu, masukan dari *hidden layer* tidak hanya masukan dari *input layer* tetapi juga termasuk keluaran dari *hidden layer* dari saat sebelumnya. Perluasan dari struktur RNN ditunjukkan di Gambar 2.3.3. LSTM dan RNN memiliki struktur perluasan yang sama, tetapi struktur *memory cell* dari *hidden layer* keduanya berbeda. *Forget gate*, *input gate* dan *output gate* telah ditambahkan kedalam *hidden layer's memory cell* LSTM yang berdasar dari struktur RNN dan desain dari ketiga

gerbang ini memiliki struktur yang secara efektif menyelesaikan permasalahan *vanishing gradient*. Struktur *memory cell* LSTM yang terdapat di *hidden layer* ditunjukkan oleh Gambar 2.3.3.



Gambar 2.3.2.1 Arsitektur Unit LSTM

$$i_t = \sigma(W_i X_t + H_i h_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_f X_f + H_f h_{t-1} + b_f) \quad (2)$$

$$o_t = \sigma(W_o X_f + H_o h_{t-1} + b_o) \quad (3)$$

$$c_t = tanh(W_c X_f + H_c h_{t-1} + b_c) \quad (4)$$

$$c_t = f_t * c_{t-1} + i_t * c_t \quad (5)$$

$$h_t = o_t * \tanh(c_t) \quad (6)$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (7)$$

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (8)$$

Pada sisi kiri persamaan:

- i_t adalah *input gate* dan menentukan informasi mana yang harus diperbarui didalam *cell*.
- f_t adalah *forget gate* dan menentukan informasi mana yang harus diturunkan dari *cell.*
- o_t adalah *output gate* yang menentukan seberapa banyak informasi yang keluar.
- c_t adalah nilai kandidat untuk *state* dari *memory cell* pada waktu t.
- c_t adalah *state* dari *memory cell* saat ini pada waktu t, yang didapat dari kombinasi i_t dan c_t, f_t dan c_{t-1} dari perkalian *element-wise* (*).
- h_t adalah nilai keluaran yang telah difilter oleh output gate.
- σ melambangkan fungsi sigmoid antara rentan 0 hingga 1, fungsi tanh digunakan untuk menaruh nilai antara -1 dan 1.

Pada sisi kanan persamaan:

- x_t adalah masukan untuk *memory cell* pada waktu t.
- W_i, W_f, W_o, W_c, H_i, H_f, H_o, dan H_c adalah matriks pembobotan.
- b_i, b_f, b_o, dan b_c adalah vektor bias.

2.4 ARIMA

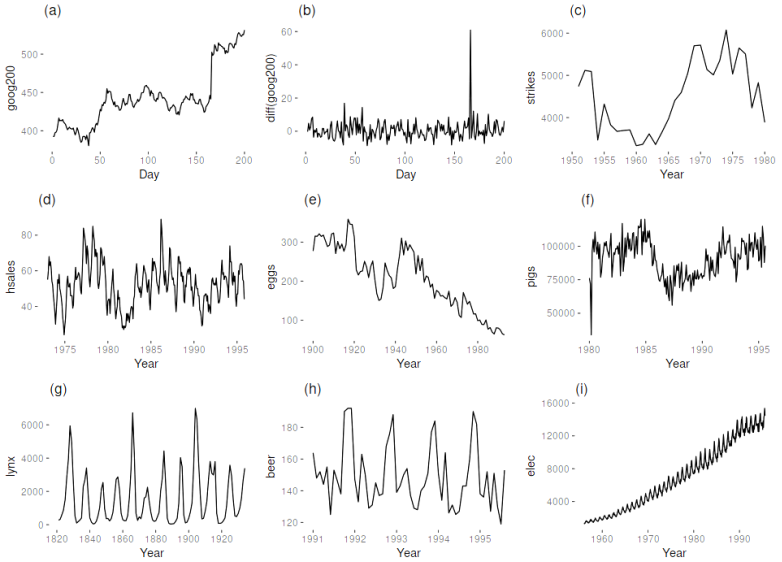
ARIMA merupakan singkatan dari *Auto Regressive Integrated Moving Average*. Singkatan tersebut pada dasarnya cukup menjelaskan tentang ARIMA itu sendiri. *Auto Regressive* merupakan sebuah model yang menggunakan hubungan yang tergantung pada sebuah observasi dan beberapa observasi yang lambat. *Integrated* sendiri berarti penggunaan hasil selisih dari

pengamatan kasar (misal, pengurangan sebuah pengamatan dari pengamatan sebelumnya). *Moving average* berarti sebuah model yang menggunakan ketergantungan pengamatan dari model rata-rata bergerak.

2.4.1 Stasioneritas dan Deteksi Stasioneritas

Stasioneritas *time-series* adalah satu karakteristik *time-series* yang tidak bergantung pada waktu ketika dilakukan observasi. Sehingga, *time-series* yang memiliki tren atau musim tidak termasuk stasioner (tren dan musim akan berpengaruh terhadap nilai dari *time-series* pada waktu yang berbeda, tetapi dilain hal *white noise series* adalah stationer).

Pada permasalahan prediksi *time-series*, stasioneritas adalah prasayarat untuk melakukan prediksi. Metode untuk mendeteksi stasioneritas yang paling umum adalah metode observasi dan metode *unit root testing*. Metode observasi secara umum bertujuan untuk memeriksa apabila diagram dari sebuah *sequence* memiliki kecenderungan atau periodisitas yang jelas. Metode observasi secara umum juga bertujuan untuk membandingkan dengan sifat-sifat fungsi autokorelasi dan fungsi parsial korelasi.



Gambar 2.4.1.1 Sembilan contoh time-series

(a) Harga saham Google selama 200 hari berturut-turut; (B) Perubahan harian dalam harga saham Google selama 200 hari berturut-turut; (c) Jumlah pemogokan tahunan di AS; (d) Penjualan bulanan rumah keluarga baru yang dijual di AS (dalam satuan dollar); (f) Total babi setiap bulan yang disembelih di Victoria, Australia; (g) Total lynx tahunan yang terperangkap di distrik sungai McKenzie di Kanada barat laut; (h) Produksi bir Australia bulanan; (i) Produksi listrik bulanan Australia [16]

2.4.2 *Differencing*

Pada Gambar 2.4.1, menunjukkan bahwa harga saham Google yang tertera merupakan termasuk non-stasioner dipanel (a), tetapi ketika diubah kedalam bentuk perubahan harian menjadi stasioner di panel (b). Hal ini menunjukkan bahwa salah satu cara untuk membuat sebuah data *time-series* yang awalnya bersifat non-stasioner menjadi stasioner adalah dengan cara menghitung selisih dari hasil pengamatan. Hal ini disebut dengan *differencing*.

2.4.3 *Unit Root Test* dan *Augmented Dickey–Fuller Test*

Tes statistik memperkuat sebuah asumsi tentang data, termasuk data *time-series*. Tes statistik hanya dapat digunakan untuk menginformasikan sejauh mana hipotesis nol dapat ditolak atau gagal ditolak. Hasil selanjutnya harus ditafsir agar masalah yang diberikan menjadi bermakna. Meskipun demikian, tes statistik dapat memberikan hasil pemeriksaan yang cepat dan bukti konfirmasi bahwa suatu data *time-series* bersifat stasioner atau tidak.

Tes *Augmented Dickey-Fuller* adalah jenis uji statistik yang disebut uji akar unit. Intuisi di balik *unit root test* adalah ia menentukan seberapa kuat deret waktu ditentukan oleh tren. Ada sejumlah *unit root test* dan *Augmented Dickey-Fuller* mungkin salah satu yang lebih banyak digunakan. Ini menggunakan model autoregresif dan mengoptimalkan kriteria informasi di beberapa nilai lag yang berbeda.

Hipotesis nol dari tes ini adalah bahwa deret waktu dapat diwakili oleh *root unit*, bahwa ia tidak stasioner (memiliki beberapa struktur yang tergantung waktu). Hipotesis alternatif (menolak hipotesis nol) adalah bahwa deret waktu stasioner.

Null Hypothesis (H_0) berarti jika gagal ditolak, berarti *time-series* tersebut memiliki *unit root* (non-stasioner). Alternatif *Hypothesis* (H_1) berarti jika *null hypothesis* ditolak, maka *time-series* tidak memiliki *unit root* (stasioner).

Hasil tersebut dapat diinterpretasi menggunakan nilai *p-value* dari hasil tes. Sebuah *p-value* yang dibawah batas ambang (seperti 5% atau 1%) berarti bahwa *null hypothesis* tersebut tertolak (bersifat stasioner), sebaliknya sebuah *p-value* diatas ambang batas berarti bahwa *null hypothesis* tersebut diterima (bersifat non-stasioner).

2.5 Metode Evaluasi Root Mean Square

Metode akar kuadrat rata-rata atau dengan nama lain *root mean square* atau yang selanjutnya disingkat dengan RMSE

adalah metode yang sering dipakai untuk mengukur perbedaan nilai (nilai sampel atau populasi) yang diprediksi oleh suatu model atau pemerkira. RMSE mencerminkan akar kuadrat dari perbedaan nilai tersebut.

RMSE selalu bersifat nonnegatif dan nilai 0 berarti menandakan bahwa sebuah hasil perkiraan dari model tidak terdapat kesalahan. Secara singkat, semakin rendah nilai RMSE menandakan model tersebut semakin baik dan semakin tinggi nilai RMSE menandakan model tersebut semakin buruk.

Karena RMSE merupakan hasil akar kuadrat dari perbedaan dua nilai, semakin besar nilai error maka akan sebanding dengan besarnya nilai RMSE. Oleh karena itu, RMSE sangat sensitive terhadap perbedaan nilai yang sangat jauh atau *outlier*. Secara matematis, RMSE direpresentasikan dalam ekspresi matematika berikut.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\left|y_i - y_i'\right|\right)^2} \qquad (9)$$

2.6 Metode Evaluasi Mean Absolute Error

Metode rata-rata kesalahan absolut atau dengan nama lain *mean absolute error* atau yang selanjutnya disingkat dengan MAE adalah metode yang sering dipakai untuk mengukur perbedaan nilai (nilai sampel atau populasi) yang diprediksi oleh suatu model atau pemerkira pada kejadian yang sama. MAE meng

Dengan demikian, MAE adalah rata-rata aritmatika dari kesalahan absolut yang direpresentasikan dalam bentuk matematika sebagai berikut.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - x_i\right| \qquad (10)$$

Dimana Y adalah prediksi dan X nilai sebenarnya. MAE menggunakan skala yang sama dengan data yang diukur. Hal ini dikenal sebagai ukuran akurasi yang bergantung pada skala dan oleh karena itu tidak dapat digunakan untuk membuat perbandingan yang menggunakan skala yang berbeda. MAE adalah ukuran umum kesalahan perkiraan dalam analisis deret waktu, kadang-kadang digunakan tertukar dengan *mean absolute deviation*.

2.7 Metode Evaluasi Mean Absolute Percentage Error

Metode rata-rata persentase kesalahan absolut atau dengan nama lain *mean absolute percentage error* (MAPE) adalah ukuran akurasi prediksi metode peramalan dalam statistik, misalnya dalam estimasi tren, juga digunakan sebagai fungsi kerugian untuk masalah regresi dalam *machine learning*. Biasanya mengungkapkan akurasi sebagai rasio yang ditentukan oleh rumus:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \quad (11)$$

Di mana A_t adalah nilai aktual dan F_t adalah nilai perkiraan. MAPE juga dapat diekspresikan sebagai persentase, yang merupakan persamaan di atas dikalikan dengan 100. Nilai absolut dalam perhitungan ini dijumlahkan untuk setiap titik perkiraan waktu dan dibagi dengan jumlah titik pas *n*. Mengalikan 100% menjadikannya persentase kesalahan dari sebuah akurasi prediksi.

2.8 Saham dan Pasar Modal

Saham adalah sebuah bukti kepemilikian nilai sebuah perusahaan. Dengan menerbitkan saham, memungkinkan perusahaan-perusahaan yang membutuhkan pendanaan jangka panjang untuk 'menjual' kepentingan dalam bisnis - saham (efek ekuitas) - dengan imbalan uang tunai. Ini adalah metode utama untuk meningkatkan modal bisnis selain menerbitkan obligasi. Saham dijual melalui pasar primer (*primary market*) atau pasar

sekunder (*secondary market*). Saham (*stock*) merupakan salah satu instrumen pasar keuangan yang paling popular. Menerbitkan saham merupakan salah satu pilihan perusahaan ketika memutuskan untuk pendanaan perusahaan. Pada sisi yang lain, saham merupakan instrumen investasi yang banyak dipilih para investor karena saham mampu memberikan tingkat keuntungan yang menarik.

2.9 Python

Python merupakan salah satu bahasa pemrograman yang banyak dikenal. Bahasa pemrgoraman ini memiliki karakter *high-level*, *interpreted*, dan umum. Python disebut sebagai bahasa pemrograman yang bersifat *high-level* karena Python memiliki abstraksi yang kuat dari karakter detail yang disediakan komputer. Selain itu Python juga memiliki sifat *interpreted* yang mengeksekusi kode secara bebas dan langsung, tanpa harus melakukan kompilasi program menjadi instruksi bahasa mesin. Python juga memiliki karakter bahasa pemrograman umum, artinya bahasa pemrgoraman ini dapat diimplementasikan untuk apapun seperti pengembangan web, perangkat lunak untuk pengguna, *system scripting, system backend*, ataupun penelitian. Didalam ranah *Deep Learning,* Python menjadi bahasa pemrograman yang paling banyak digunakan.

2.10 Tensorflow

TensorFlow adalah *software library* yang bersifat gratis dan *open-source* untuk pembuatan program yang membutuhkan komputasi numerik dengan sumber daya yang besar. TensorFlow dikembangkan oleh tim Google Brain.

2.11 Keras

Keras adalah *software library neural-network* yang bersifat *open-source* ditulis menggunakan bahasa pemrograman Python.

Keras mampu berjalan diatas TensorFlow, Theano, dan beberapa *software library* lain. Keras dikembangkan untuk memungkinkan eksperimentasi dengan cepat dengan *deep neural networks*, dengan focus kepada penggunaan yang ramah-pengguna, *modular*, dan mudah untuk dikembangkan. Keras dapat berjalan baik di CPU dan GPU. Keras berisi banyak implementasi *neural network* umum, fungsi aktivasi, *optimizer* dan tool lain yang memudahkan dalam pengolahan citra dan data teks.

2.12 Pandas

Pandas adalah *software library* yang ditulis untuk bahasa pemrograman Python untuk analisis dan manipulasi data. Secara khusus, Pandas menyediakan struktur data dan operasi untuk memanipulasi table angka dan data *time-series*.

2.13 Numpy

Numpy adalah *software library* yang ditulis untuk bahasa pemrograman Python menambahkan dukungan untuk *array* dan matriks multi-dimensi yang besar, bersama dengan banyak koleksi fungsi matematika tingkat tinggi untuk beroperasi pada *array*.

2.14 Scikit-learn

Scikit-learn adalah *software library* yang ditulis untuk bahasa pemrograman Python dengan memiliki beragam algoritma klasifikasi, regresi, dan pengelompokan termasuk dukungan kepada *vector machines*, *random forest*, *gradient boosting*, k-means dan DBSCAN, dan dirancang untuk beroperasi dengan NumPy dan SciPy.

2.15 Matplotlib

Matplotlib adalah *software library* yang ditulis untuk bahasa pemrograman Python yang populer digunakan untuk visualisasi

data. Matplotlib dapat membantu untuk membuat visualisasi yang berkualitas.

2.16 Seaborn

Seaborn adalah *software library* yang ditulis untuk bahasa pemrograman Python berdasarkan matplotlib. Seaborn menyediakan antarmuka untuk menggambar grafik statistik yang menarik dan informatif.

2.17 Statsmodel

Statsmodel adalah *software library* yang ditulis untuk bahasa pemrograman Python yang menyediakan fungsionalitas untuk berbagai model statistik, melakukan tes statistik, dan eksplorasi data statistik.

2.17.1 Tes ADFuller

Statsmodel menyediakan implementasi tes ADFuller atau yang biasa disingkat ADF. Fungsi ini mengembalikan keluaran sebagai berikut:

1. Nilai p-value
2. Nilai dari tes statistik
3. Jumlah lag dari sebuah tes
4. Batas nilai kritis

2.18 Google Colab

Google Colab merupakan produk serupa seperti Jupyter Notebook yang dikeluarkan oleh Google Research. Berbeda dengan Jupyter Notebook, Google Colab menawarkan integrasi antara layanan Google Colab dengan perangkat keras, GPU, yang disediakan oleh Google. Hal ini mempermudah pengembang atau siapapun yang ingin melakukan eksplorasi didalam domain data science tanpa perlu memikirkan tentang pemasangan infrastruktur perangkat sama sekali.

BAB III
PERANCANGAN SISTEM

Bab ini menjelaskan mengenai perancangan data dan model prediksi time-series menggunakan ARIMA dan *Long-Short Term Memory network*.

3.1 Perancangan dan Pemilihan Data

Data yang digunakan sebagai masukan model diambil dari situs *investing.com* dengan penjelasan ditabel berikut.

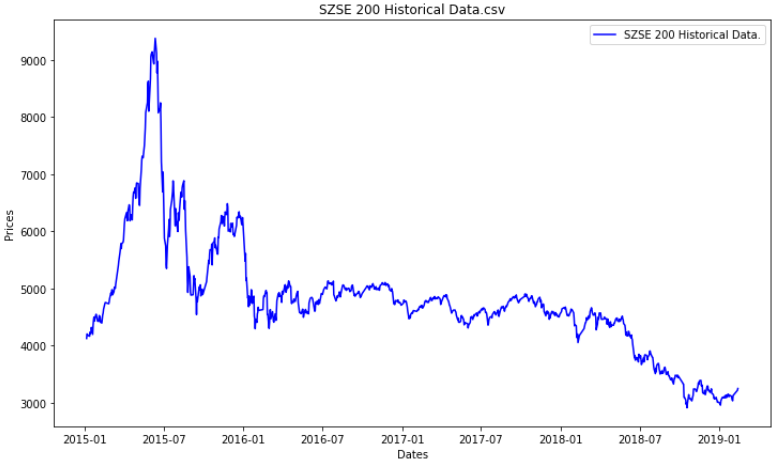
Tabel 3.1.1 Daftar rinci dataset yang akan digunakan

	Ticker	Nama ticker	P-value	Periode
1	399300.SZ	Shenzen CSI 300 Index	0.061947	05-01-2015 s.d. 12-02-2019
2	399007.SZ	Shenzen SZSE 300 Index	0.119512	05-01-2015 s.d. 12-02-2019
3	399009.SZ	Shenzen SZSE 200 Index	0.271582	05-01-2015 s.d. 12-02-2019
4	JKSE	Jakarta Composite Index	0.861231	02-01-2015 s.d. 15-02-2019

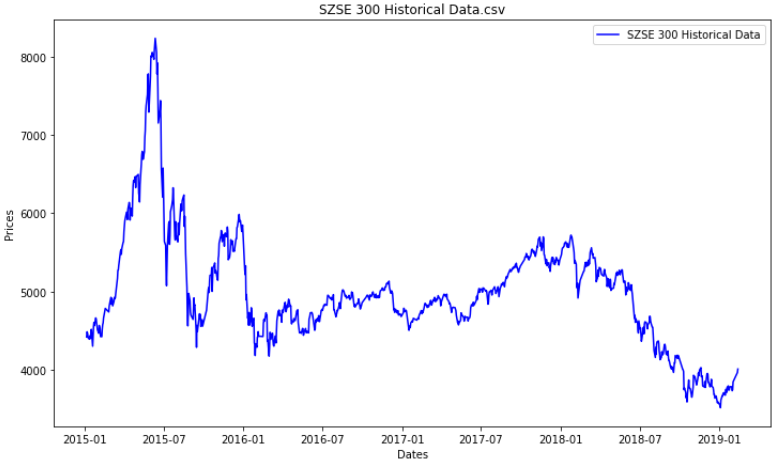
Untuk memastikan pengaruh stabilitas pada algoritma prediksi LSTM, data-data tersebut dipilih berdasar kondisi *stability* yang berbeda dengan menggunakan uji ADF di Python. Seiring meningkatnya nilai *p-value*, nilai *stability* akan berkurang juga secara signifikan. [17] Tujuan dari pemilihan data ini adalah untuk menguji kondisi *stability* yang beragam.

Masing-masing data *time-series* di Tabel 3.1 memiliki periode yang sama yaitu seribu hari. Kemudian data tersebut dibagi menjadi dua: sebanyak 700 data untuk data latih dan sebanyak 300

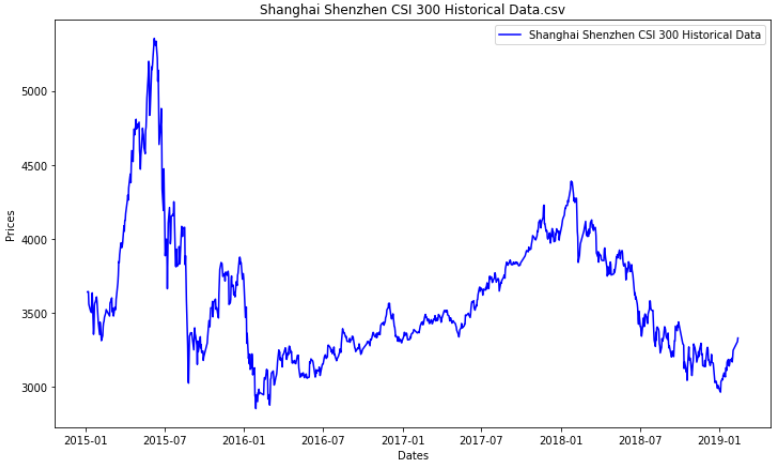
data untuk data uji. Berikut adalah tampilan dari keempat data indeks saham yang akan digunakan.



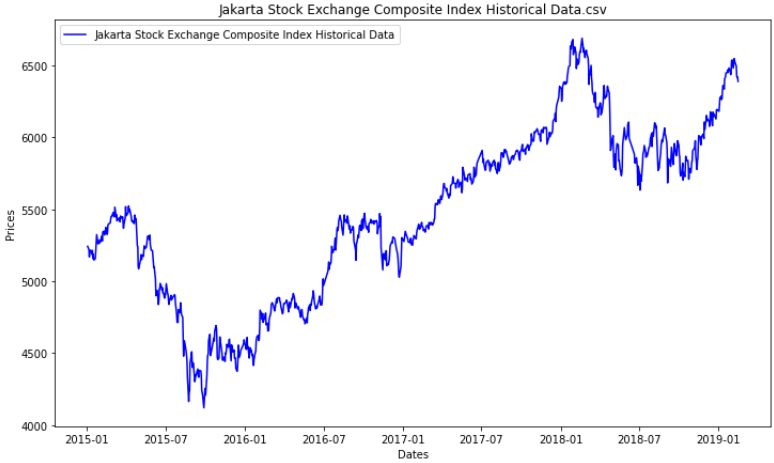
Gambar 2.17.1.1 Tampilan grafik data SZSE 200 yang akan digunakan



Gambar 2.17.1.2 Tampilan grafik data SZSE 300 yang akan digunakan



Gambar 2.17.1.3 Tampilan grafik data CSI 300 yang akan digunakan

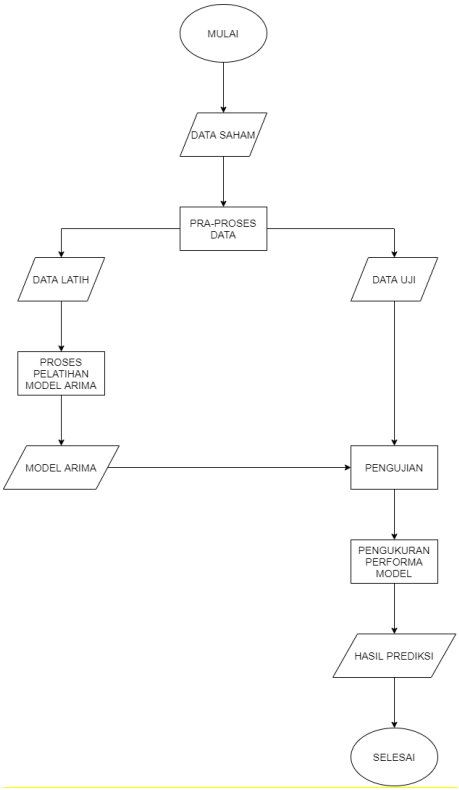


Gambar 2.17.1.4 Tampilan grafik data JKSE yang akan digunakan

3.2 Desain Umum Model ARIMA

Desain umum model ARIMA untuk kasus prediksi data time-series dibagi kedalam beberapa tahap utama diantaranya tahap praproses data, tahap pembangunan model ARIMA, dan tahap pelatihan dan pengujian. Pembagian ini dilakukan agar terdapat batasan yang jelas terhadap perubahan data time-series yang terjadi.

Diagram alir dari model ARIMA ditunjukan oleh gambar sebagai berikut.



Gambar 2.17.1.1 Diagram alur untuk desain umum model ARIMA

3.2.1 Tahap Praproses Data

Sebelum dataset dimasukkan kedalam model ARIMA, perlu dilakukan praposes data agar data yang dikonsumsi oleh model dapat diterima sesuai yang diharapkan. Praproses data ini dilakukan diawal dan dilakukan hanya sekali. Data yang telah dilakukan praproses, kemudian siap digunakan kedalam model ARIMA.

Model ARIMA hanya menerima data time-series dalam bentuk array dan tiga parameter lain yang membentuk ARIMA. Data yang akan dimasukkan kedalam model ARIMA selanjutnya hanya merupakan satu variat. Dalam konteks data indeks saham, data tersebut dipilih antara data 'Open', 'High', 'Low', dan 'Close'.

3.2.2 Tahap Pembangunan Model ARIMA

Pada tahap ini data telah dilakukan praproses ditahap sebelumnya dan siap digunakan oleh model ARIMA. Model ARIMA yang akan digunakan sebagai model prediksi akan memanfaatkan implementasi dari statsmodel. Hal ini dikarenakan fokus dari tugas akhir ini lebih kepada model LSTM.

Model ARIMA dibuat dengan memanggil objek ARIMA yang disediakan oleh statsmodel. Objek ARIMA yang disediakan tersebut memiliki dua parameter yang harus dipenuhi untuk membuat sebuah model, yaitu data yang akan dilatih dan parameter untuk membuat model statistik ARIMA (*lag*, *degree of differencing*, dan *moving average*). Nilai parameter tersebut secara berurutan dari *lag*, *degree of differencing*, dan *moving average* adalah 5, 1, dan 0.

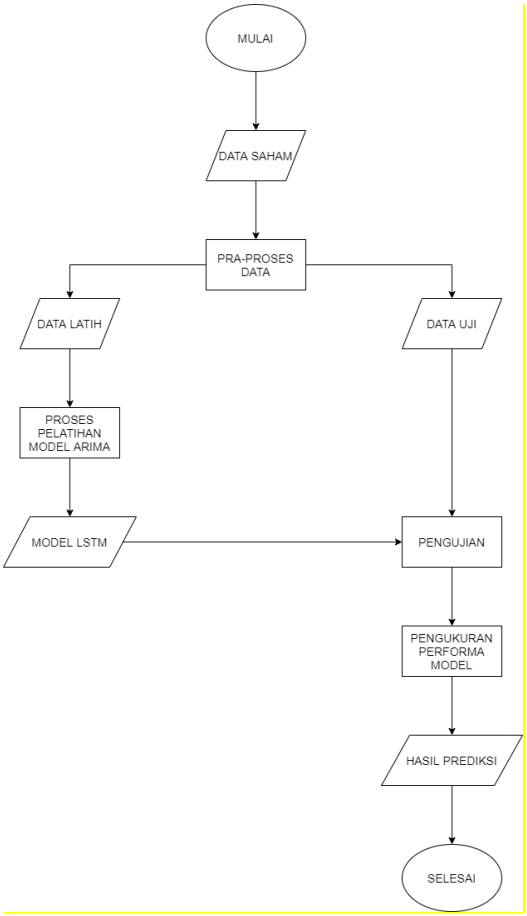
3.2.3 Tahap Pelatihan dan Pengujian Model ARIMA

Model ARIMA yang telah dibuat dapat digunakan untuk memperkirakan nilai waktu di masa depan. Fungsi predict pada objek hasil ARIMA dapat dipanggil untuk membuat prediksi. Fungsi tersebut menerima indeks langkah-langkah waktu untuk membuat prediksi sebagai argumen. Indeks ini relatif terhadap awal dataset pelatihan yang digunakan untuk membuat prediksi.

Jika 100 pengamatan digunakan dalam dataset pelatihan, maka indeks dari langkah waktu berikutnya untuk membuat prediksi akan ditentukan untuk fungsi prediksi sebagai (mulai = 101, akhir = 101). Hal ini akan mengembalikan *array* dengan satu elemen yang berisi prediksi. Didalam proses pelatihan dan pengujian ini nilai yang diperkirakan akan tetap dalam skala asli. Kemudian dilakukan tahap pengamatan secara iteratif didalam sebuah *array*. *Array* tersebut kemudian dapat dibandingkan dan dievaluasi menggunakan metode RMSE.

3.3 Desain Umum Model LSTM

Desain umum model LSTM memiliki beberapa proses utama seperti praproses data, pembangunan arsitektur, pelatihan dan pengujian, dan pascaproses data. Berikut merupakan diagram alir dari sistem yang dibangun.



Gambar 3.2.3.1 Diagram alur untuk desain umum model LSTM

Praproses merupakan tahap pertama yang akan dilakukan sebelum data tersebut digunakan ke dalam tahap selanjutnya. Praproses untuk model LSTM secara garis besar berupa proses skala ulang ke dalam rentan [0,1], perubahan menjadi data stasioner, dan memanipulasi data *time-series* menjadi beberapa *batch*.

3.3.1 Tahap Praproses Data

Sebelum *dataset* dimasukkan kedalam model LSTM, perlu dilakukan proposes data agar data yang dikonsumsi oleh model dapat diterima sesuai yang diharapkan. Praproses data ini dilakukan diawal dan dilakukan hanya sekali. Data yang telah dilakukan praproses kemudian siap digunakan kedalam model LSTM.

Data yang akan dimasukkan kedalam model LSTM selanjutnya hanya merupakan satu variat. Dalam konteks data indeks saham, data tersebut dipilih antara data 'Open', 'High', 'Low', dan 'Close'. Data *time-series* tersebut, secara jamak memiliki karakter yang beragam.

Data saham bersifat nonstasioner, ini berarti bahwa ada struktur dalam data yang tergantung pada waktu. Secara khusus, ada tren peningkatan dalam data. Data stasioner lebih mudah untuk dimodelkan dan kemungkinan besar akan menghasilkan perkiraan yang lebih baik. Tren dapat dihapus dari pengamatan, kemudian ditambahkan kembali ke perkiraan kemudian untuk mengembalikan prediksi ke skala asli.

Selanjutnya data yang sudah diubah menjadi data stasioner dilakukan skala ulang kedalam rentan [0,1]. Tahap terakhir dari praproses data adalah memanipulasi *dataset* tersebut menjadi beberapa *batch* yang menyerupai pendekatan untuk menyelesaikan permasalahan *supervised learning.* Ditahap ini data juga dikonversi menjadi bentuk *difference*, yaitu selisih antara data periode saat ini dengan periode masa lalu.

Untuk memastikan bahwa data hasil keluaran dari tahap praproses ini bersifat stasioner, maka data hasil tersebut dites ulang karakter stasioneritasnya. Pengujian ini menggunakan metode uji Augmented Dickey-Fuller. Metode ini merupakan metode yang paling sering digunakan untuk menguji *unit root test.*

3.3.2 Tahap Pembangunan Arsitektur LSTM

Desain arsitektur LSTM ini dibuat untuk memenuhi kebutuhan kasus prediksi *time-series*. Arsitektur (dalam konteks *neural network*) disini didefinisikan sebagai desain untuk

peletakan fungsi-fungsi *neural network* yang membentuk model. Arsitektur ini menetapkan *learning rate* sebesar 0.001, *timestep* sebesar lima dan jumlah *node hidden layer* sejumlah sepuluh.

Didalam desain arsitektur LSTM ini terdapat *input layer*, satu *hidden layer* dengan satu unit lstm dan *output layer*. *Input layer* menerima *dataset* yang siap pakai berbentuk 3D *array* dengan parameter *sample*, *timestep* dan *feature*. *Hidden layer* berisi sepuluh unit lstm. Arsitektur LSTM yang didesain akan bersifat *stateful*.

3.3.3 Tahap Pelatihan dan Pengujian Arsitektur LSTM

Setelah melakukan praproses data dan menyiapkan arsitektur, data latih dan data uji dengan masing-masing sejumlah 700 dan 300 data, akan digunakan langsung untuk tahap pelatihan dan pengujian arsitektur LSTM. Tujuh ratus data latih tersebut digunakan pertama kali terhadap arsitektur LSTM untuk melatih arsitektur tersebut agar nilai pembobotan didalam unit LSTM dapat diset relatif sesuai dataset yang diberikan.

Proses pelatihan memanfaatkan data latih untuk membangun model LSTM. Pelatihan menggunakan optimisasi Adam dengan *learning rate* 0.001 dan lima *timestep*. Proses pelatihan dijalankan kedalam beberapa *batch* berukuran lima dan seribu *epoch*. Hal ini berarti dengan total seribu *dataset* untuk setiap data saham akan dibagi menjadi kedalam 400 *batch* dengan masing-masing berisi lima sampel data. Pembobotan model akan diperbarui setelah setiap *batch* dengan lima sampel data. Ini juga berarti satu *epoch* akan melibatkan empat ratus *batch* atau empat ratus pembaharuan terhadap model. Model akan terbuka terhadap seluruh *dataset* sebanyak seribu kali, dengan *hyperparameter epoch* sebanyak seribu, yang berarti empat ratus ribu total *batch* selama proses pelatihan.

3.3.4 Tahap Pascaproses Data

Setelah data peramalan dibuat, data tersebut perlu dikembalikan kedalam rentan dataset semula sebelum dilakukan tahap praproses dan dikembalikan ke bentuk non-*difference*. Hal

ini dibutuhkan karena data tersebut masih dalam rentan angka [0,1]. Data yang telah dikembalikan kemudian dapat diinterpretasi sebagai hasil dari model prediksi LSTM.

BAB IV
IMPLEMENTASI

Pada bab ini akan dijelaskan tentang implementasi dari desain arsitektur rancangan sistem yang telah dibahas pada Bab 3 meliputi kode program dalam perangkat lunak. Selain itu, implementasi dari tiap proses, parameter masukan, keluaran dan beberapa keterangan yang berhubungan dengan program juga dijelaskan.

4.1 Lingkungan Implementasi

Pemodelan latih, pemodelan uji, pemodelan evaluasi, dan kegiatan lainnya yang berkaitan dengan implementasi topik ini dilakukan dengan menggunakan platform Google Colab Pro dengan ketentuan dan penggunaan produk yang ditetapkan oleh Google Colab per Mei 2020.

4.2 Implementasi Untuk Model ARIMA

Pada subbab ini akan dijabarkan mengenai implementasi untuk model ARIMA yang mengacu pada bab tiga mengenai perancangan sistem desain umum model ARIMA.

Implementasi umum model ARIMA memiliki beberapa tahap utama diantaranya tahap praproses data, tahap pembangunan model ARIMA, dan tahap pelatihan dan pengujian. Implementasi umum model ARIMA memanfaatkan *software library* stasmodel. Data yang telah dilakukan praproses kemudian digunakan untuk membuat model ARIMA. Model yang telah dibuat kemudian dilakukan evaluasi menggunakan metode RMSE yang memanfaatkan *software library* sklearn.

4.2.1 Implementasi Praproses Data Untuk Model ARIMA

Pada subbab ini akan dijabarkan mengenai implementasi praproses data untuk model ARIMA yang mengacu pada bab tiga mengenai perancangan sistem desain umum model ARIMA.

Implementasi praproses memanfaatkan fungsionalitas dari objek Pandas DataFrame, dimana dataset yang diambil, dimasukkan kedalam objek Pandas DataFrame. Manfaat dari penggunaan Pandas DataFrame sebagai perantara untuk melakukan praproses data adalah kemudahan seperti pada cuplikan kode dibawah.

Series merupakan objek Pandas DataFrame yang membawa data time-series (data saham yang dipilih). Sebagai objek yang memanfaatkan fungsionalitas dari Pandas DataFrame, series dapat memanggil langsung nilai 'Price' dengan perintah yang relatif singkat, mudah, dan eksplisit. Beberapa perintah selanjutnya untuk melakukan proses pembagian data uji dan data latih juga mendapatkan manfaat yang sama. Deklarasi perintah untuk membagi data uji dan data latih sebesar 70% dan 30% secara berturut-turut dapat diekspresikan secara singkat, mudah, dan eksplisit.

Terakhir, sebagai hasil dari proses implementasi praproses ini terdapat lima objek yang akan dipakai untuk proses selanjutnya, seperti size, train, test, history, dan predictions.

```
X = series['Price'].values
size = int(len(X) * 0.7)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
```

Kode Sumber 4.2.1.1 Implementasi tahap praproses data untuk model ARIMA

4.2.2 Implementasi Pembangunan Model ARIMA

Untuk pembangunan model ARIMA, implementasi dibawah memanfaatkan library dari stasmodel. Terdapat dua parameter yaitu history dimana sebagai parameter data masukan dan order.

Parameter order merupakan parameter yang dibutuhkan oleh objek ARIMA untuk membuat model ARIMA. Nilai parameter tersebut secara berurutan dari *lag*, *degree of differencing*, dan *moving average* adalah 5, 1, dan 0.

```
model = ARIMA(history, order=(5,1,0))
```

Kode Sumber 4.2.2.1 Implementasi pembangunan model ARIMA

4.2.3 Implementasi Pelatihan dan Pengujian Model ARIMA

Sebagai tahap terakhir, objek ARIMA yang memiliki fungsi fit kemudian dipanggil. Hal ini bertujuan untuk melakukan proses pelatihan terhadap data yang telah masuk. Keluaran dari fungsi fit tersebut kemudian disimpan kedalam variabel dan dilakukan proses prediksi oleh ARIMA.

Hasil dari prediksi disimpan dalam sebuah *array* yang nanti akan dilakukan evaluasi terhadap data uji. Didalam proses pelatihan dan pengujian ini nilai yang diperkirakan akan tetap dalam skala asli. Kemudian dilakukan tahap pengamatan secara iteratif didalam sebuah *array*. *Array* tersebut kemudian dapat dibandingkan dan dievaluasi menggunakan metode RMSE.

```
model_fit = model.fit(disp=0)
output = model_fit.forecast()
yhat = output[0]
predictions.append(yhat)
error = mean_squared_error(test, predictions)
```

Kode Sumber 4.2.3.1 Implementasi pelatihan dan pengujian model ARIMA

4.3 Implementasi Untuk Model LSTM

Pada subbab ini akan dijabarkan mengenai implementasi untuk model LSTM yang mengacu pada bab tiga mengenai perancangan sistem desain umum model LSTM.

Secara umum, model LSTM yang akan dibuat memanfaatkan *library* Keras. Keras menyediakan beberapa fungsi yang dapat digunakan sebagai pabrik pembuat model. Untuk membuat model LSTM, didalam desain ini memanfaatkan fungsi

sequence didalam Keras. Setelah model LSTM terbuat, model tersebut kemudian dikompilasi.

4.3.1 Implementasi Praproses Data untuk Model LSTM

Pada subbab ini akan dijabarkan mengenai implementasi praproses data untuk model LSTM yang mengacu pada bab tiga mengenai perancangan sistem desain umum model LSTM.

Implementasi praproses data untuk model LSTM dibungkus kedalam beberapa fungsi agar implementasi tetap mudah dibaca. Setiap implementasi praproses data untuk model LSTM

Implementasi praproses memanfaatkan fungsionalitas dari objek Pandas DataFrame, dimana dataset yang diambil, dimasukkan kedalam objek Pandas DataFrame. Series merupakan objek Pandas DataFrame yang membawa data time-series (data saham yang dipilih). Sebagai objek yang memanfaatkan fungsionalitas dari Pandas DataFrame, series dapat memanggil langsung semua nilai yang terkandung dengan perintah yang relatif singkat, mudah, dan eksplisit. Beberapa perintah selanjutnya juga mendapatkan manfaat yang sama.

Terakhir, sebagai hasil dari proses implementasi praproses ini terdapat lima objek yang akan dipakai untuk proses selanjutnya, seperti size, train, test, history, dan predictions. Implementasi praproses data diimplementasikan kedalam fungsi prepare_data seperti cuplikan kode dibawah berikut.

```
# transform series into train and test sets for supervised learning
def prepare_data(series, n_test, n_lag, n_seq):
# extract raw values
raw_values = series.values
# transform data to be stationary
diff_series = difference(raw_values, 1)
diff_values = diff_series.values
diff_values = diff_values.reshape(len(diff_values), 1)
# rescale values to -1, 1
scaler = MinMaxScaler(feature_range=(-1, 1))
scaled_values = scaler.fit_transform(diff_values)
scaled_values = scaled_values.reshape(len(scaled_values), 1)
# transform into supervised learning problem X, y
supervised = series_to_supervised(scaled_values, n_lag, n_seq)
supervised_values = supervised.values
# split into train and test sets
train, test = supervised_values[0:-n_test], supervised_values[-
n_test:]
return scaler, train, test
```

Kode Sumber 4.3.1.1 Implementasi tahap praproses data untuk model LSTM

4.3.1.1 Implementasi Pengambilan Dataset

Proses pengambilan variabel dari dataset diimplementasikan pada cuplikan kode berikut. Sumber data yang berasal dari github diimplementasikan kedalam sebuah kelas bernama SourceFile. Selain karena sifat *immutability* dari tiap nama file, juga karena memudahkan '*tag*' untuk setiap dataset. Sehingga, kelas SourceFile didesain hanya untuk mengembalikan nilai yang diminta.

Kelas SourceFile kemudian diinstansiasi untuk kemudian digunakan oleh pandas sebagai sumber file.

```
class SourceFile:

def __init__(self):
self.source_file = ["https://raw.githubusercontent.com/aufawibo
wo/lstm-under-different-
stability/master/dataset/SZSE%20200%20Historical%20Data.csv",
"https://raw.githubusercontent.com/aufawibowo/lstm-
under-different-
stability/master/dataset/SZSE%20300%20Historical%20Data.csv",
"https://raw.githubusercontent.com/aufawibowo/lstm-
under-different-
stability/master/dataset/Shanghai%20Shenzhen%20CSI%20300%20
Historical%20Data.csv",
"https://raw.githubusercontent.com/aufawibowo/lstm-
under-different-
stability/master/dataset/Jakarta%20Stock%20Exchange%20Composi
te%20Index%20Historical%20Data.csv"]

self.stock_name = ["SZSE 200",
"SZSE 300",
"CSI 300",
"JKSE"]

def get_one_file(self, i):
if isinstance(i, int):
return self.source_file[i]
else:
return "Parameter i to index must be an integer"

def get_one_stock_name(self, i):
return self.stock_name[i]

def get_all_file():
return self.source_file

def get_all_stock_name(self,i):
return self.stock_name
```

```
file_object = SourceFile()
stock_array_index = 0
file_name = file_object.get_one_file(stock_array_index)
stock_name = file_object.get_one_stock_name(stock_array_index)

# read data:
series = read_csv(file_name, header=0, parse_dates=[0], index_col
=0, squeeze=True)
series = series.head(1000)
```

Kode Sumber 4.3.1.2 Implementasi pengambilan dataset untuk model LSTM

4.3.1.2 Implementasi Mengubah Data Menjadi Stasioner

Untuk mengubah data menjadi stasioner dilakukan implementasi sebagai berikut. Data dimanipulasi ulang sehingga hanya menampilkan selisih antara data sekarang dibanding dengan data dimasa yang akan datang.

```
# transform data menjadi proses stasioner
diff_series = difference(raw_values, 1)
diff_values = diff_series.values
diff_values = diff_values.reshape(len(diff_values), 1)
```

Kode Sumber 4.3.1.3 Implementasi mengubah data kedalam proses stasioner

Implementasi ini tidak memanfaatkan framework apapun sehingga untuk membuat *difference* pada dataset sebuah fungsi *difference* dibuat. Fungsi ini menerima masukan dataset yang akan dikonversi menjadi selesih dataset. Dataset yang masuk kemudian dikonversi secara iteratif dari awal hingga akhir.

```
# membuat difference pada dataset
def difference(dataset, interval=1):
	diff = list()
 for i in range(interval, len(dataset)):
 value = dataset[i] - dataset[i - interval]
	diff.append(value)
 return Series(diff)
```

Kode Sumber 4.3.1.4 Implementasi untuk membuat difference pada dataset

4.3.1.3 Implementasi Skala Ulang Dataset

Implementasi proses skala ulang memanfaatkan fungsi MinMaxScaler yang disediakan oleh sklearn. Objek MinMaxScaler harus dibuat sebelum dapat digunakan untuk melakukan proses skala ulang dengan parameter nilai rentan yang menjadi target skala ulang. Setelah objek terbuat, objek MinMaxScaler kemudian dapat digunakan. Dataset kemudian diolah oleh objek MinMaxScaler yang kemudian mengeluarkan nilai dalam rentan [0,1].

```
# rescale values to 0, 1
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_values = scaler.fit_transform(diff_values)
scaled_values = scaled_values.reshape(len(scaled_values), 1)
```

Kode Sumber 4.3.1.5 Implementasi proses skala ulang pada dataset

4.3.1.4 Implementasi Pembagian Dataset Menjadi Data Uji dan Data Latih

Proses terakhir sebelum dataset siap dipakai oleh model, dataset dipisah menjadi data uji dan data latih. Disini dataset dipisah sebesar 70% dari awal dataset untuk menjadi data latih dan 30% dataset akhir menjadi data uji.

```
train, test = supervised_values[0:-n_test], supervised_values[-n_test:]
```

Kode Sumber 4.3.1.6 Implementasi pembagian dataset menjadi data uji dan data latih

4.3.1.5 Implementasi Cek Karakter Stasioner

Untuk mengecek karakter stasioner suatu data, sebuah fungsi dibuat yang memanfaatkan *library* statsmodel yang mengambil fungsionalitas ADF.

Fungsi ini membutuhkan dua parameter, series dan file_address. Series merupakan variabel yang digunakan untuk menampung data timeseries yang akan diuji. Variabel ini nantinya digunakan oleh fungsi adfuller yang memberikan kemudahan untuk mengetahui karakter stasioner dari suatu data.

Untuk memudahkan, sebuah alur kondisional dibuat untuk menentukan apakah sebuah data bersifat stasioner atau nonstasioner. Jika nilai ADF lebih besar dari ketiga nilai kritis 1%, 5%, dan 10%, maka data tersebut bersifat nonstasioner. Kebalikannya, jika nilai ADF lebih kecil dari ketiga nilai kritis 1%, 5%, dan 10%, maka data tersebut bersifat stasioner.

```
def stationarity_info(series, file_address):
    results = adfuller(series)
    print(file_address)   # google colab address only
    print('ADF Statistic: %f' % results[0])
    print('p-value: %f' % results[1])
    print('Critical Values:')
    values = []
    for key, value in results[4].items():
        print('\t%s: %.3f' % (key, value))
        values.append(value)
    if(results[0] > max(values)):
        print("Non-stationary and most likely random walk.")
    else:
        print("Stationary")
    print("")
```

Kode Sumber 4.3.1.7 Implementasi cek karakter stasioner

4.3.2 Implementasi Pembangunan Arsitektur LSTM

Pada subbab ini dijelaskan implementasi pembangunan arsitektur LSTM. Arsitektur LSTM ini diimplementasikan dengan memanfaatkan pustaka keras model. Model didalam keras

didefinisikan sebagai *layer sequence*. Menurut dokumentasi keras, *sequential* model cocok untuk *plain stack of layer* dimana setiap *layer* memiliki satu *input tensor* dan satu *output tensor*.

Objek *sequence* harus dibuat terlebih dahulu sebagai prasyarat dan kemudian menambahkan *layer* yang dibutuhkan yang sesuai dengan desain arsitektur yang telah dibuat. Setelah selesai dideklarasikan, model tersebut dikompilasi untuk memanfaatkan kerangka kerja (Keras) yang mendasari untuk mengoptimalkan perhitungan yang akan dilakukan oleh model.

Setelah objek *sequence* terbuat, selanjutnya objek tersebut dapat dispesifikkan lebih lanjut. Dalam implementasi arsitektur LSTM ini, objek *sequence* ditambahkan objek LSTM yang berisi detil berapa jumlah *neuron* yang akan digunakan, bentuk data *batch* yang akan masuk kedalam model, dan sifat *stateful* dari LSTM. Untuk membuat *layer* yang saling terhubung, implementasi arsitektur tugas akhir ini memanfaatkan kelas *Dense*. Terakhir, model dikompilasi dengan memanggil fungsi *Compile* dengan parameter *loss* menggunakan *mean square error* sebagai metode evaluasi (yang akan dijelaskan di bab berikutnya) dan metode optimisasi Adam.

```
# fit an LSTM network to training data
def fit_lstm(train, n_lag, n_seq, n_batch, nb_epoch, n_neurons):
# reshape training into [samples, timesteps, features]
X, y = train[:, 0:n_lag], train[:, n_lag:]
X = X.reshape(X.shape[0], 1, X.shape[1])
# design network
model = Sequential()
model.add(LSTM(n_neurons, batch_input_shape=(n_batch, X.sha
pe[1], X.shape[2]), stateful=True))
model.add(Dense(y.shape[1]))
model.compile(loss='mean_squared_error', optimizer='adam')
...
```

Kode Sumber 4.3.2.1 Implementasi pembangunan arsarsitektur LSTM

4.3.3 Implementasi Pelatihan dan Pengujian Model LSTM

Setelah dilakukan kompilasi, model yang terbuat dari implementasi desain arsitektur harus disesuaikan nilai pembobotannya dengan data. Hal ini dapat dilakukan dengan cara melakukan proses pelatihan dan pengujian terhadap model yang telah dibuat.

Ukuran *batch* yang sama harus digunakan didalam tahap pelatihan dan pengujian model. Proses prediksi akan dilakukan setiap akhir *timestep*.

```
# fit network
for i in range(nb_epoch):
model.fit(X, y, epochs=1, batch_size=n_batch, verbose=0, shuffl
e=False)
model.reset_states()
return model
```

Kode Sumber 4.3.3.1 Implementasi pelatihan dan pengujian model

4.3.4 Implementasi Pascaproses Data

Terakhir, sebelum dapat diinterpretasi, data harus dikembalikan kedalam bentuk semula sebelum praproses dilakukan. Implementasi ini kembali menggunakan fungsi MinMaxScaler.

```
def invert_scale(scaler, X, yhat):
new_row = [x for x in X] + [yhat]
array = np.array(new_row)
array = array.reshape(1, len(array))
inverted = scaler.inverse_transform(array)
return inverted[0, -1]
```

Kode Sumber 4.3.4.1 Implementasi fungsi invert_scale

Kemudian keluaran dari fungsi invers_scale dimasukkan kedalan fungsi inverse_difference agar data yang masih berbentuk *difference* dapat kembali kedalam data indeks saham biasa.

```
def inverse_difference(history, yhat, interval=1):
return yhat + history[-interval]
```

Kode Sumber 4.3.4.2 Implementasi fungsi inverse_difference

[*Halaman ini sengaja dikosongkan*]

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dijelaskan tentang uji coba dan evaluasi dari implementasi sistem yang telah dilakukan pada Bab 4.

5.1 Lingkungan Uji Coba

Pemodelan latih, pemodelan uji, pemodelan evaluasi, dan kegiatan lainnya yang berkaitan dengan implementasi topik ini dilakukan dengan menggunakan platform Google Colab Pro dengan ketentuan dan penggunaan produk yang ditetapkan oleh Google Colab per Mei 2020.

5.2 Dataset

Pada tugas akhir ini, data yang digunakan adalah empat data yang telah dipilih pada Bab 3.1.

5.3 Hasil Praproses

Subbab ini melaporkan hasil rinci tahapan praproses.

Praproses bertujuan untuk mengubah data nonstasioner menjadi stasioner. Data yang telah mengalami tahapan praproses kemudian diuji lagi menggunakan tes ADF. Statsmodel menyediakan implementasi tes ADFuller atau yang biasa disingkat ADF. Fungsi ini mengembalikan keluaran sebagai berikut:

1. Nilai p-value
2. Nilai dari tes statistik
3. Jumlah lag dari sebuah tes
4. Batas nilai kritis

Ketika nilai tes statistik bernilai dibawah nilai kritis, hipotesis null ditolak dan secara implisit memberitahu bahwa time-series tersebut bersifat stasioner.

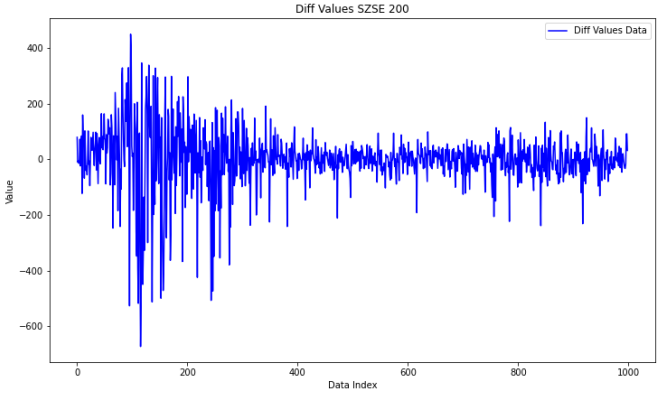
Secara keseluruhan berikut adalah hasil tes ADF keempat data.

Tabel 5.3.1 Hasil akhir tes ADFuller terhadap keempat dataset

Dataset	P-value	Tes Statistik ADF	Stasioner
399300.SZ (CSI 300)	.0	-6.040668	Ya
399007.SZ (SZSE 300)	.0	-7.686888	Ya
399009.SZ (SZSE 200)	.0	-7.682671	Ya
Jakarta Composite Index (JKSE)	.0	-12.229461	Ya

5.3.1 Pengubahan Data

Tahap pertama dalam praproses data adalah proses *differencing*. Berikut adalah hasil visualisasi proses *differencing* terhadap data SZSE 200.



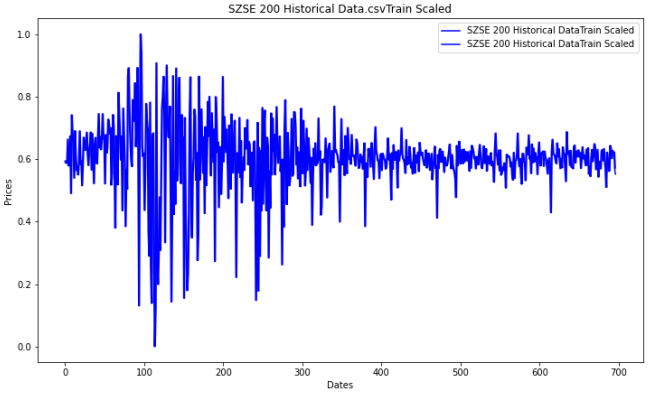
Gambar 5.3.1.1 Visualisasi proses differencing terhadap data SZSE 200

Untuk mengecek kondisi stasioneritas dari hasil praproses, setiap dataset dicek menggunakan fungsi tes ADFuller yang memanfaatkan *library* statsmodel.

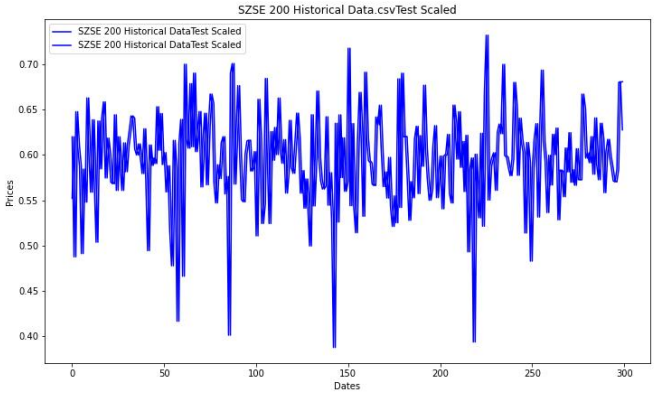
5.3.2 Skala Ulang Dataset

Hasil skala ulang dataset merupakan hasil final tahap praproses. Data ini kemudian siap digunakan oleh model.

Berikut merupakan visualisasi hasil praproses terhadap data SZSE 200. Hasil praproses terhadap data SZSE 300, CSI 300, dan JKSE dilampirkan didalam Lampiran G sampai dengan Lampiran I.



Gambar 5.3.2.1 Hasil praproses terhadap data SZSE 200 untuk data latih



Gambar 5.3.2.2 Hasil praproses terhadap data SZSE 200 untuk data uji

5.4 Skenario Uji Coba

Proses uji coba berguna untuk menemukan hasil prediksi yang terbaik. Skenario uji coba dilakukan pada dataset SZSE 200, SZSE 300, dan CSI 300 dengan ARIMA sebagai pembanding [17]. Dataset JKSE digunakan sebagai dataset tambahan dengan karakter diluar ketiga saham tersebut.

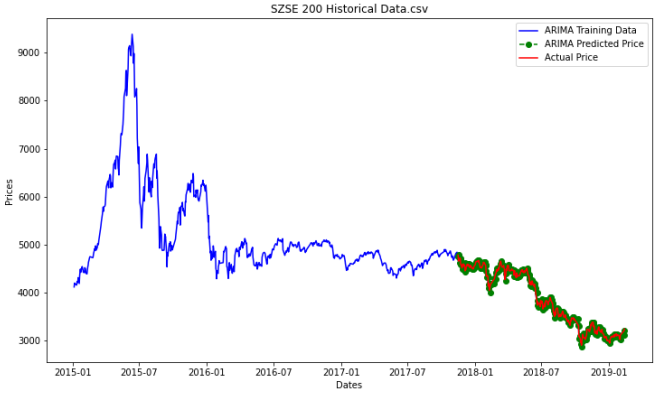
Hasil prediksi dari kedua model kemudian dievaluasi dengan menggunakan metode RMSE (*Root Mean Square* Error), MAE (*Mean Absolute Error*), dan MAPE (*Mean Absolute Percentage Error*). Skenario uji coba ini dilakukan sebanyak sepuluh kali dan diambil nilai rata-rata.

Hidden layer node disederhanakan menjadi satu layer. Hal ini dikarenakan batasan dari penggunaan layanan Google Colab.

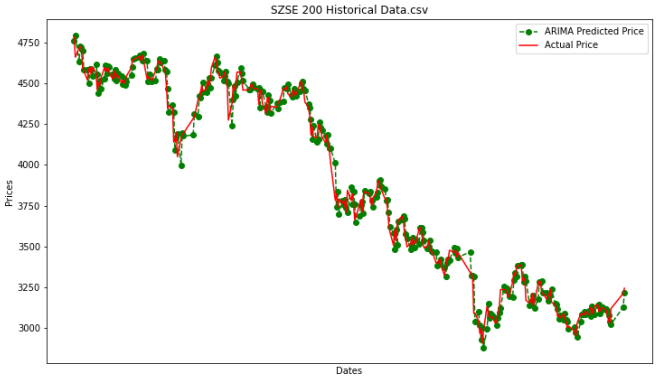
5.5 Hasil Uji Coba

Hasil uji coba berupa hasil prediksi dari keempat dataset. Secara visual dapat digambarkan sebagai berikut.

5.5.1 Data SZSE 200



Gambar 5.5.1.1 Visualisasi keseluruhan hasil prediksi SZSE 200 menggunakan ARIMA

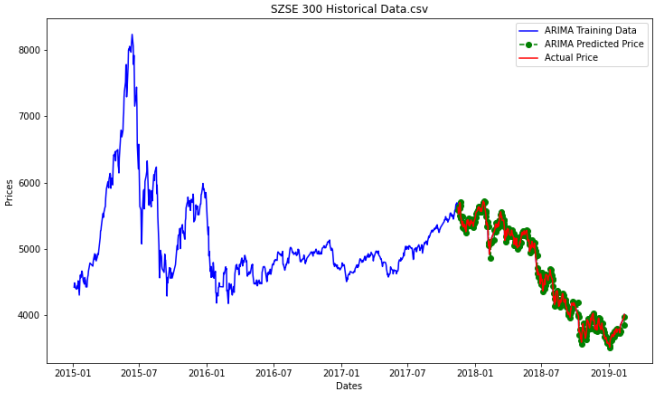


Gambar 5.5.1.2 Visualisasi fokus terhadap hasil prediksi SZSE 200 menggunakan ARIMA

Gambar 5.5.1.3 Visualisasi keseluruhan hasil prediksi SZSE 200 menggunakan LSTM

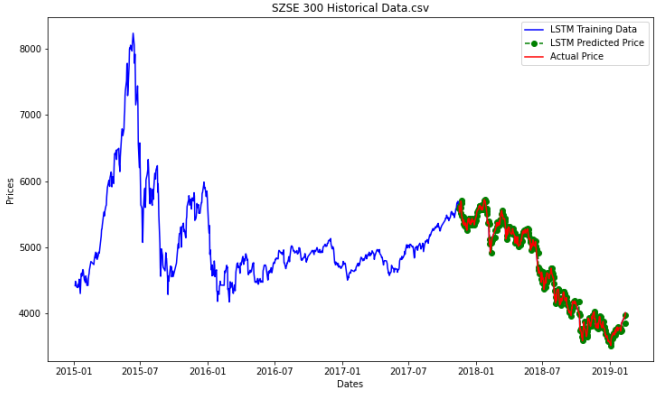
Gambar 5.5.1.4 Visualisasi fokus terhadap hasil prediksi SZSE 200 menggunakan LSTM

5.5.2 Data SZSE 300



Gambar 5.5.2.1 Visualisasi keseluruhan hasil prediksi SZSE 300 menggunakan ARIMA

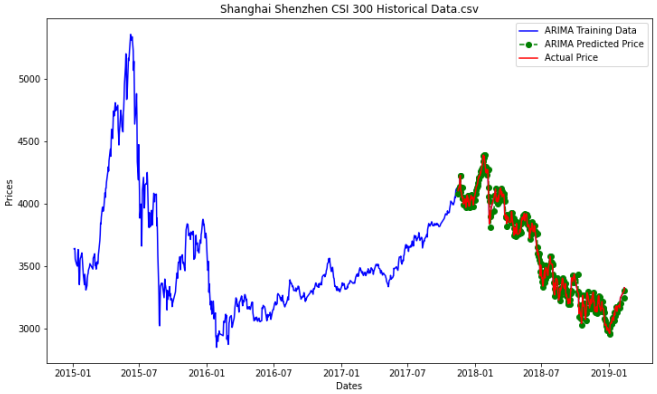
Gambar 5.5.2.2 Visualisasi fokus terhadap hasil prediksi SZSE 300 menggunakan ARIMA



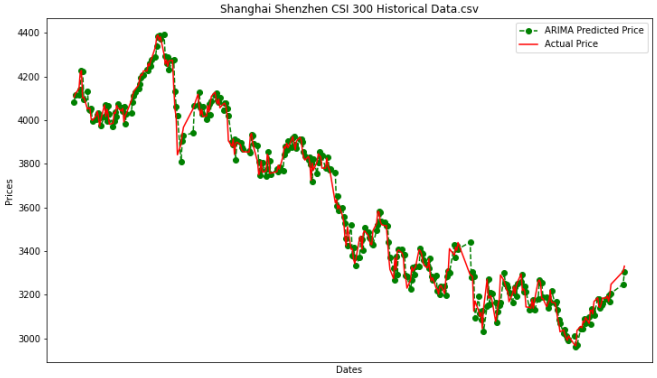
Gambar 5.5.2.3 Visualisasi keseluruhan hasil prediksi SZSE 300 menggunakan LSTM

Gambar 5.5.2.4 Visualisasi fokus terhadap hasil prediksi SZSE 300 menggunakan LSTM

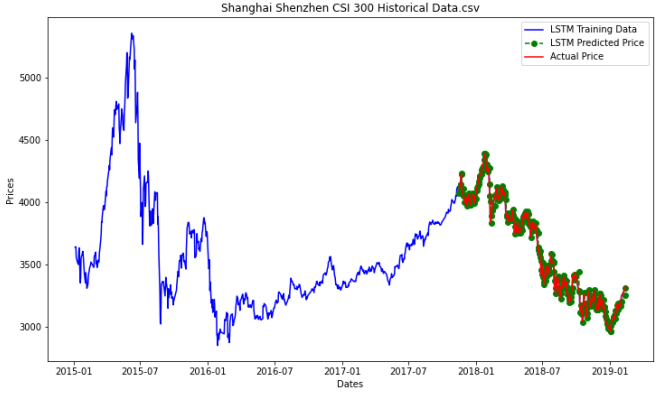
5.5.3 Data CSI 300



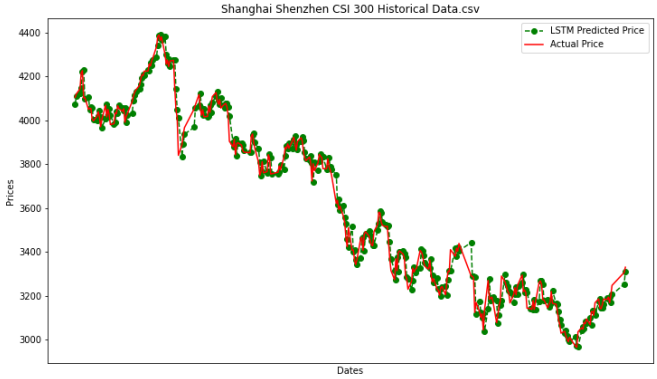
Gambar 5.5.3.1 Visualisasi keseluruhan hasil prediksi CSI 300 menggunakan ARIMA



Gambar 5.5.3.2 Visualisasi fokus terhadap hasil prediksi CSI 300 menggunakan ARIMA

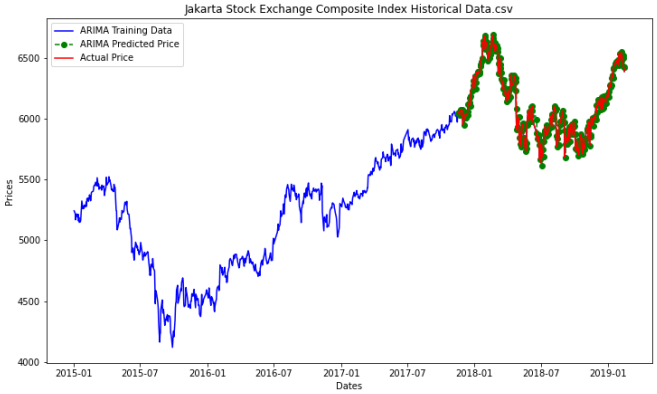


Gambar 5.5.3.3 Visualisasi keseluruhan hasil prediksi CSI 300 menggunakan LSTM

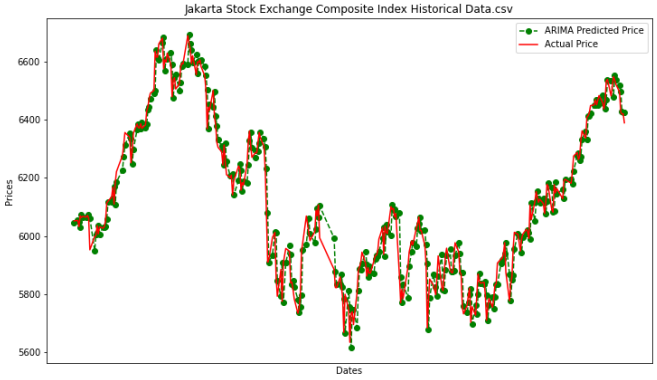


Gambar 5.5.3.4 Visualisasi fokus terhadap hasil prediksi CSI 300 menggunakan LSTM

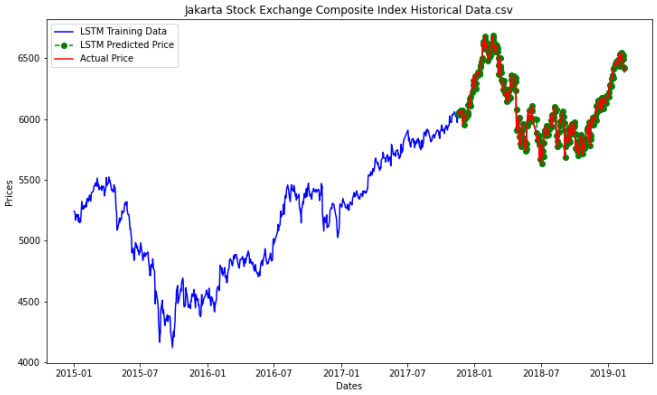
5.5.4 Data JKSE



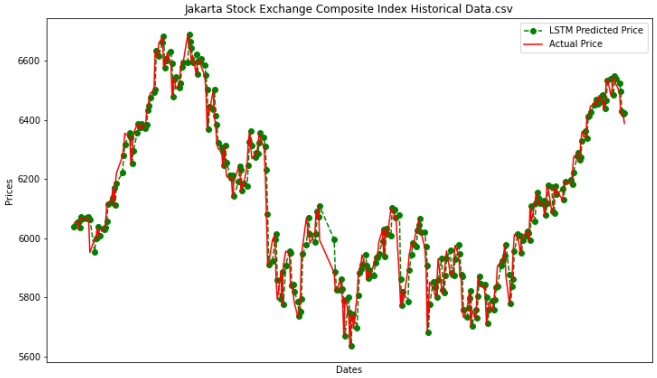
Gambar 5.5.4.1 Visualisasi keseluruhan hasil prediksi JKSE menggunakan ARIMA



Gambar 5.5.4.2 Visualisasi fokus terhadap hasil prediksi JKSE menggunakan ARIMA



Gambar 5.5.4.3 Visualisasi keseluruhan hasil prediksi JKSE menggunakan LSTM



Gambar 5.5.4.4 Visualisasi fokus terhadap hasil prediksi JKSE menggunakan LSTM

5.6 Evaluasi

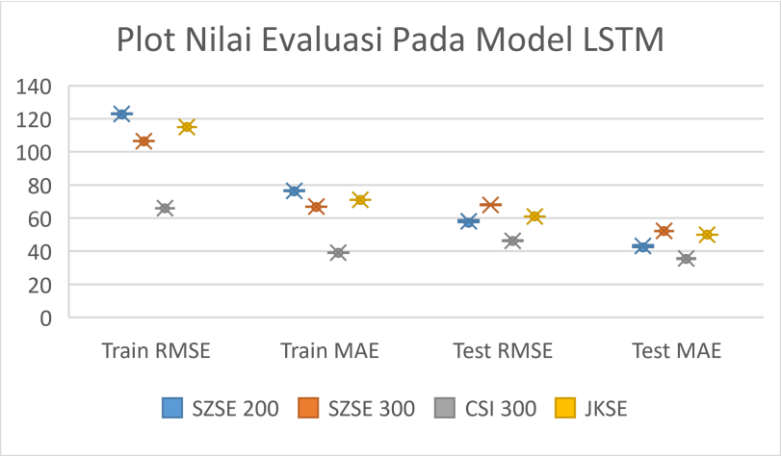
Dari skenario uji coba, didapat hasil evaluasi sebagai berikut.

Tabel 5.6.1 Daftar skor uji RMSE dan MAE terhadap empat dataset pada model LSTM

Dataset	P-value	ADF Test Value	Skor uji data latih RMSE	Skor uji data tes RMSE	Skor uji data latih MAE	Skor uji data tes MAE	Skor uji data latih MAPE	Skor uji data tes MAPE
399300.SZ (CSI 300)	0.061947	2.77	65.9	46.4	39.2	35.5	1.09	.01
399007.SZ (SZSE 300)	0.119512	2.48	106.5	68	66.8	52.3	1.27	.115
399009.SZ (SZSE 200)	0.271582	2.03	122.9	58.2	76.5	43.2	1.38	.113
Jakarta Composite Index (JKSE)	0.861231	0.64	43.3	56	31.3	43	.62	.7

Skor uji RMSE dan MAE pada Tabel 5.5.1 merupakan rata-rata hasil uji coba 10 kali percobaan. Sebagai perbandingan skor evaluasi RMSE dan MAE pada data uji (data test) terlihat lebih kecil dibanding dengan data latih (data train). Hal ini menunjukkan bahwa model telah melalui proses pembelajaran untuk mengenali pola yang terdapat disetiap dataset.

Namun, terlihat pada model data JKSE, model LSTM menunjukkan kemungkinan untuk *overvitting* atau *underfitting*.



Gambar 5.5.4.1 Plot nilai evaluasi pada model LSTM

Tabel 5.6.2 Daftar skor uji RMSE terhadap empat dataset pada model ARIMA

Dataset	P-value	ADF Test Value	Skor uji data latih RMSE	Skor uji data latih MAE	Skor uji data latih MAPE
399300.SZ (CSI 300)	0.061947	2.77	47.83	36.77	.01
399007.SZ (SZSE 300)	0.119512	2.48	69.45	54.08	.012
399009.SZ (SZSE 200)	0.271582	2.03	58.84	44.67	.012
Jakarta Composite Index (JKSE)	0.861231	0.64	57.17	43.17	.007

Dari sepuluh kali hasil uji coba, hasil skor uji RMSE dan MAE dari model ARIMA menunjukkan nilai yang sama.

Dari kedua tabel diatas menunjukkan bahwa model LSTM bersifat sedikit lebih baik dibanding dengan ARIMA pada kondisi sampel data time-series yang memiliki perbedaan stabilitas.

[*Halaman ini sengaja dikosongkan*]

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan yang didasari oleh hasil uji coba yang telah dilakukan pada bab sebelumnya. Kesimpulan nantinya sebagai jawaban dari ruusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan topik lebih lanjut.

6.1 Kesimpulan

Dalam pengerjaan tugas akhir ini, berdasarkan penjabaran di bab-bab sebelumnya, dapat disimpulkan beberapa poin sebagai berikut:

1. Model LSTM memiliki kelebihan dibanding dengan model ARIMA, meskipun secara hasil evaluasi memiliki nilai yang tidak beda jauh.
2. Hasil dari percobaan stabilitas memiliki pengaruh yang kecil pada hasil prediksi, namun memiliki pengaruh terhadap kecepatan untuk konvergen.

6.2 Saran

Saran yang diberikan terhadap hasil dari tugas akhir ini adalah:

1. Penerapan model LSTM untuk kasus prediksi indeks saham pada data *realtime*.
2. Penerapan model LSTM dapat dilakukan pada data saham.
3. Menambah fitur tambahan untuk menambah skor akurasi prediksi.
4. Penggunaan MAPE sebaiknya tidak digunakan untuk mengevaluasi hasil prediksi timeseries karena berimbas pada lemahnya kulitas hasil prediksi karena terdapat bias dalam penggunaanya [18]. Sebagai contoh [19], ketika sebuah nilai aktual adalah 150 dan nilai prediksi

100 maka nilai MAPE adalah $\left|\frac{150-100}{150}\right| = 33.3\%$, sementara jika nilai aktual bernilai 100 dan nilai prediksi adalah 150 maka nilai MAPE adalah $\left|\frac{100-150}{100}\right| = 50\%$. Meskipun kedua nilai hanya berselisih sebesar 50, tetapi dapat menghasilkan nilai MAPE yang berbeda.

5. Hasil dari prediksi saham baik dari model ARIMA ataupun LSTM terlihat, seakan, mengekor pada nilai berikutnya. Sehingga terkesan bahwa hasil prediksi untuk hari berikutnya merupakan nilai terlambat dari hari sebelumnya. Hal ini dapat diperbaiki dengan membuat model yang dapat memprediksi hingga *n* hari kedepan.

DAFTAR PUSTAKA

[1] J.-F. Zhao, "Modeling of Software Aging Based on Non-stationary Time Series," *2016 International Conference on Information System and Artificial Intelligence (ISAI),* pp. 176-180, 2016.

[2] D. E. Rumelhart, G. E. Hinton dan R. J. Williams, "Learning representations by back-propagating errors," *Nature,* vol. 323, pp. 533-536, 1986.

[3] I. Goodfellow, Y. Bengio dan A. Courvil, Deep Learning, MIT Press, 2016.

[4] Z. Zhao, R. Rao dan S. Tu, "Time-weighted LSTM Model with Redefined Labeling for Stock Trend Prediction," *International Conference on Tools with Artificial Intelligence,* pp. 1210-1217, 2017.

[5] K. A. Althelaya, E.-S. M. El-Alfy dan S. Mohammed, "Evaluation of Bidirectional LSTM for Short- and Long-Term Stock Market Prediction," *9th International Conference on Information and Communication Systems (ICICS),* pp. 151-156, 2018.

[6] G. Shmueli dan K. C. Lichtendahl Jr, Practical Time Series Forecasting with R: A Hands-On Guide [2nd Edition] (Practical Analytics), Axelrod Schnall Publishers, 2016.

[7] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," *Neural Networks,* vol. 61, pp. 85-117, 2015.

[8] J. Russell, "TechCrunch," 25 May 2017. [Online]. Available: https://techcrunch.com/2017/05/24/alphago-beats-planets-best-human-go-player-ke-jie/. [Diakses 8 June 2020].

[9] A. Krizhevsky, I. Sutskever dan G. E. Hinton, "ImageNet Classification with Deep Convolutional," *Neural Information Processing Systems,* 2012.

[10] G. Marcus, “Newyorker,” 25 11 2012. [Online]. Available: https://www.newyorker.com/news/news-desk/is-deep-learning-a-revolution-in-artificial-intelligence. [Diakses 8 6 2020].

[11] Y. Bengio, Learning Deep Architectures for AI, Now Foundations and Trends, 2009.

[12] G. Alain dan Y. Bengio, “Understanding intermediate layers using linear classifier probes,” 2016.

[13] Wikipedia contributors, “Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Deep_learning. [Diakses 9 6 2020].

[14] J. Schmidhuber, “Deep Learning in Neural Networks: An Overview,” *Neural Networks,* vol. 61, pp. 85-117, 2014.

[15] Hochreiter, Untersuchungen zu dynamischen neuronalen Netzen, 1991.

[16] R. J. Hyndman dan G. Athanasopoulos, Forecasting: Principle and Practice, Melbourne: OTexts, 2018.

[17] F. Qian dan X. Chen, “Stock Prediction based on LSTM under Different Stability,” *2019 IEEE 4th International Conference on Cloud Computing and Big Data Analytics*, pp. 483-486, 2019.

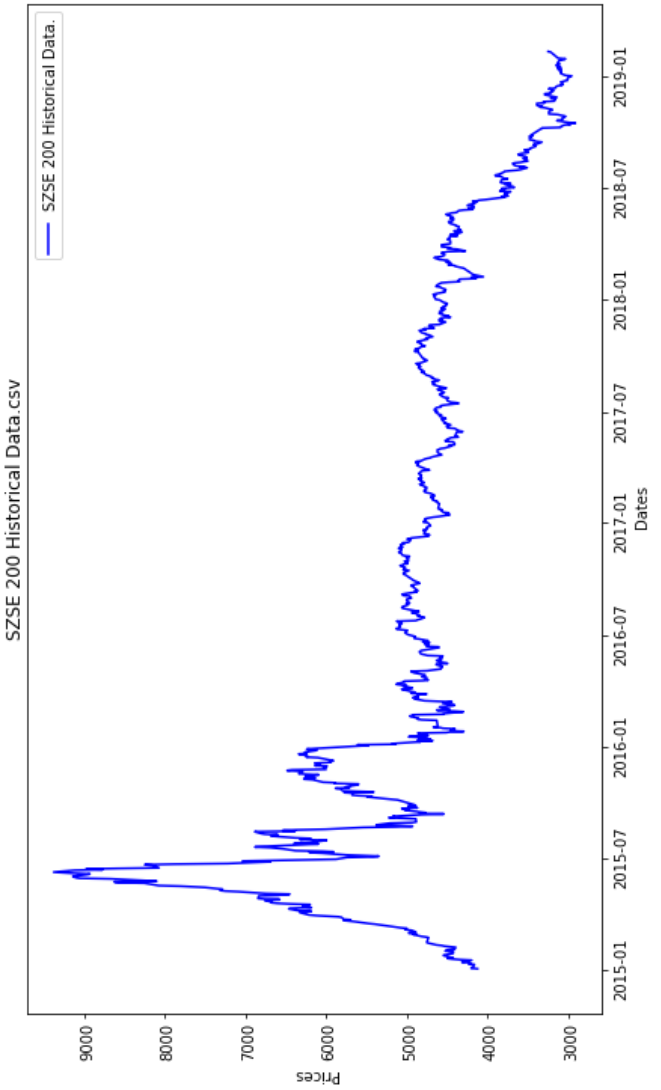
[18] F. Muklis, “Perkembangan Dan Tantangan Pasar Modal Indonesia,” *Al Masraf (Jurnal Lembaga Keuangan dan Perbankan)-Volume 1, No.1, Januari-Juni 2016*, pp. 67-68, 2016.

[19] G. Heshan, Z. Shulian dan L. Mengya, “Clustering Univariate Time Series into Stationary and Non-stationary,” *Seventh International Conference on Fuzzy Systems and Knowledge Discovery,* pp. 2782-2785, 2010.

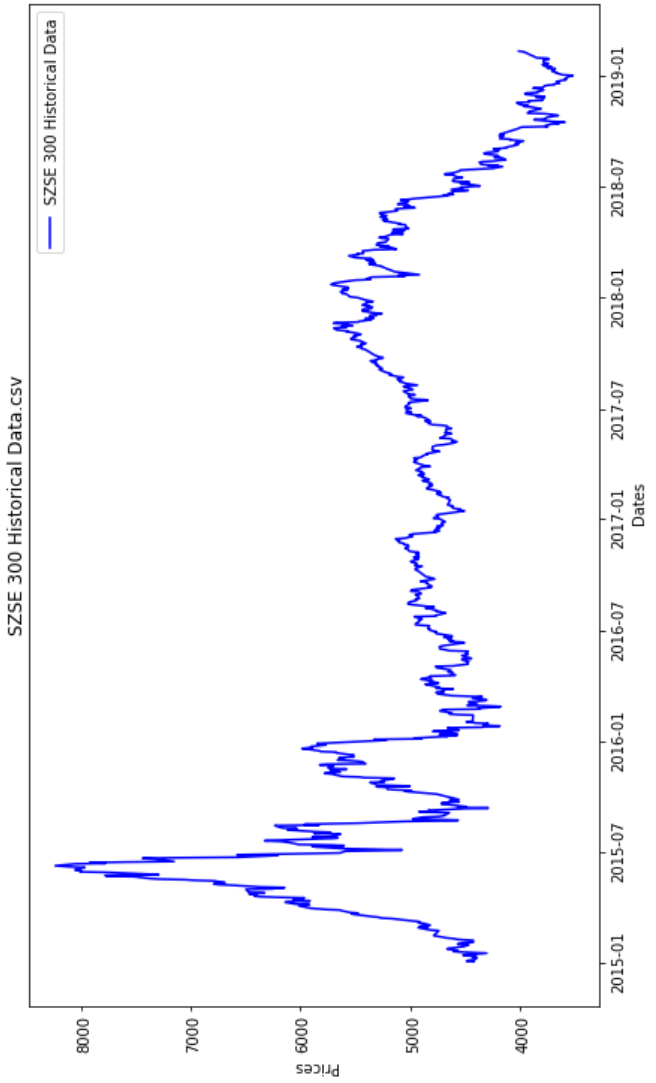
[20] D. A. DIckey dan W. A. Fuller, “Distribution of the estimators for autore gressive time series with aunit

root," *Journal of the American Statistic al Ass ociation,* vol. 74, pp. 427-431, 1979.

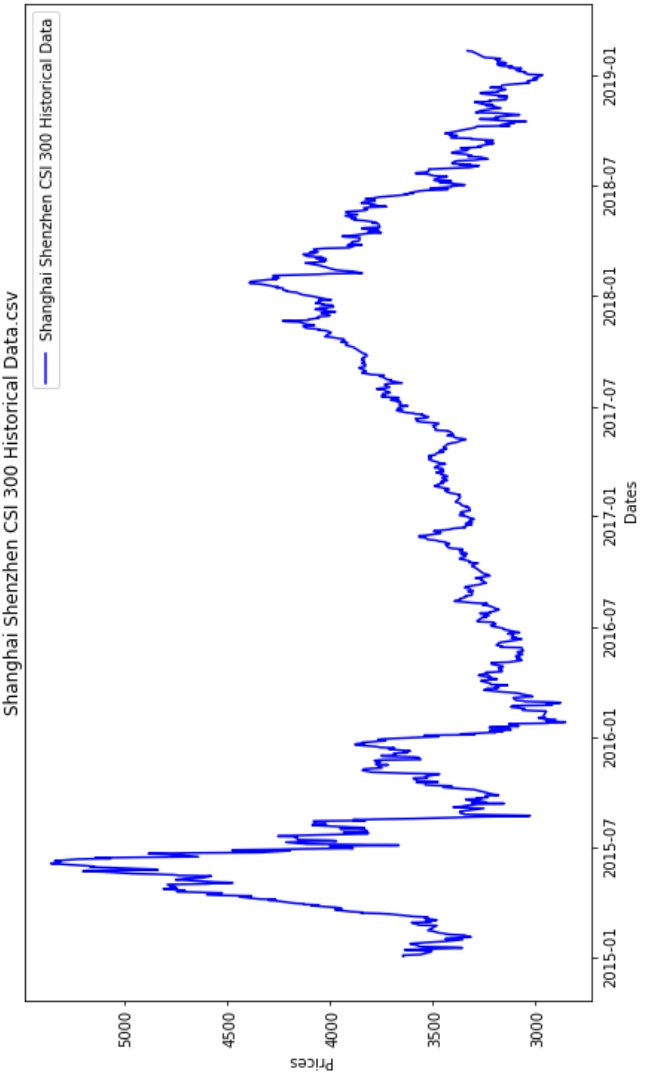
LAMPIRAN A: GAMBAR GRAFIK KEEMPAT INDEKS SAHAM



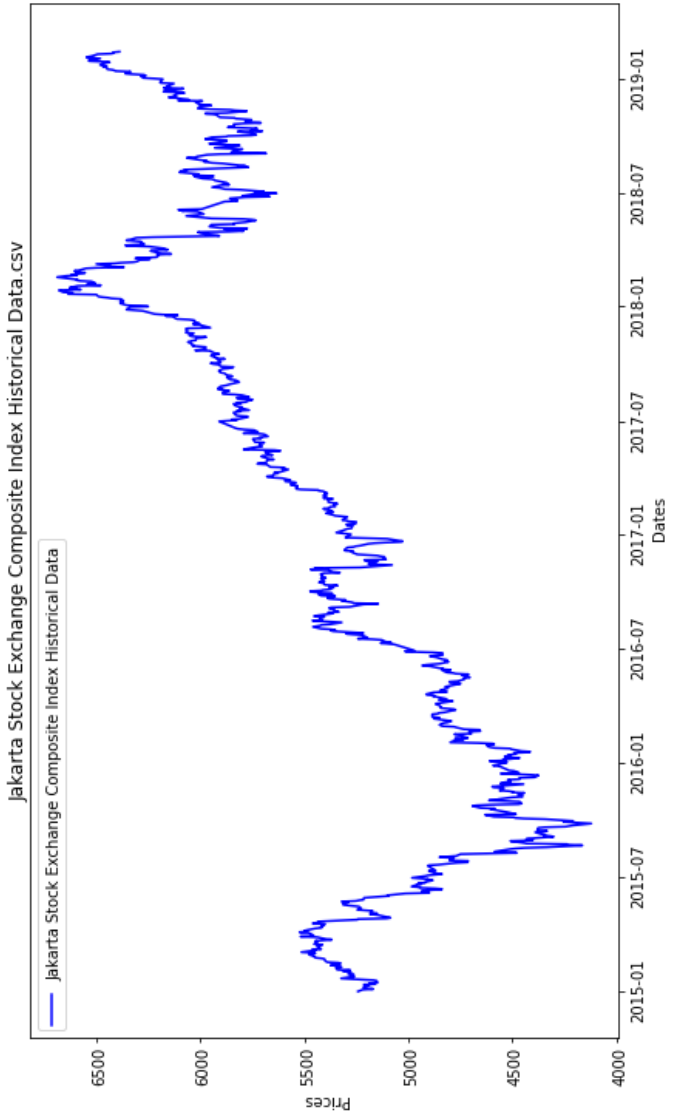
LAMPIRAN B: GRAFIK INDEKS SAHAM SZSE 300



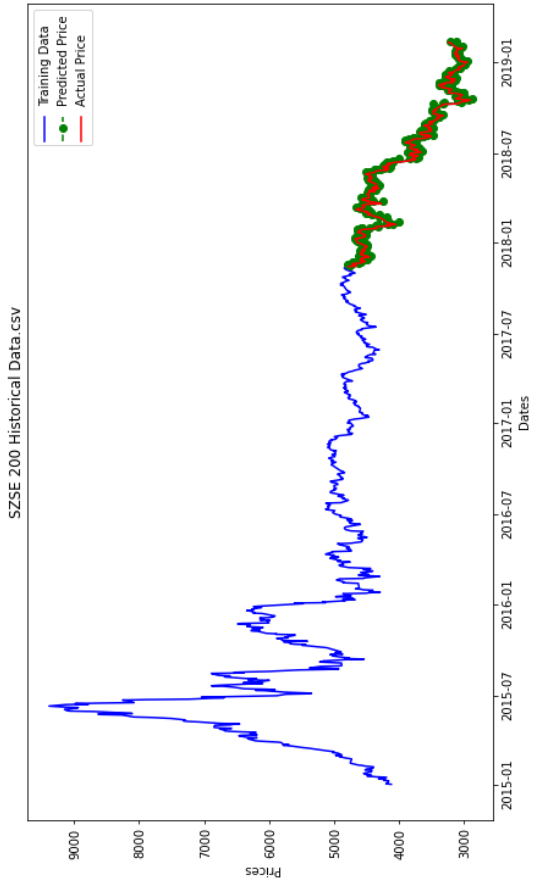
LAMPIRAN C: GRAFIK INDEKS SAHAM CSI 300

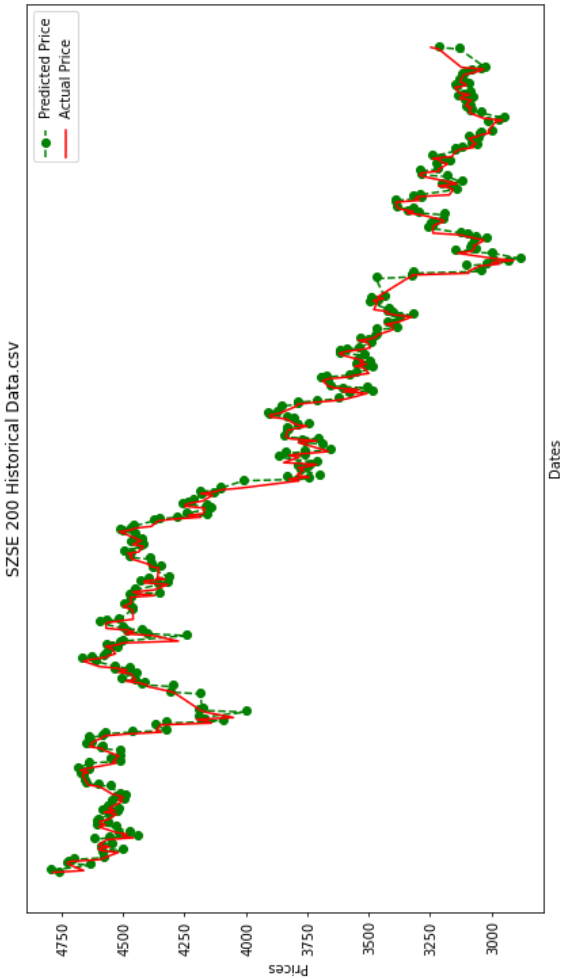


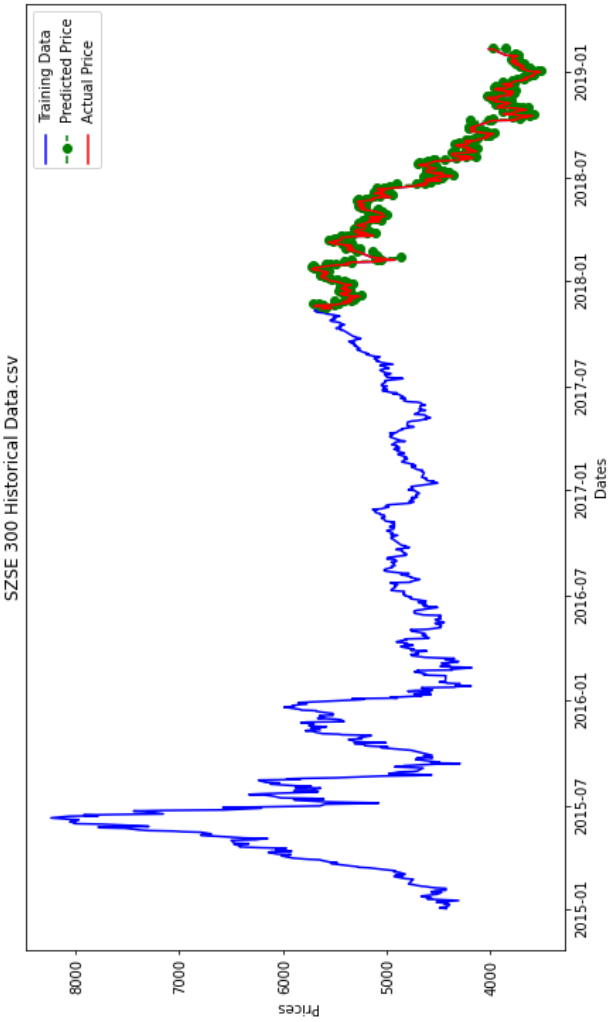
LAMPIRAN D: GRAFIK INDEKS SAHAM JKSE

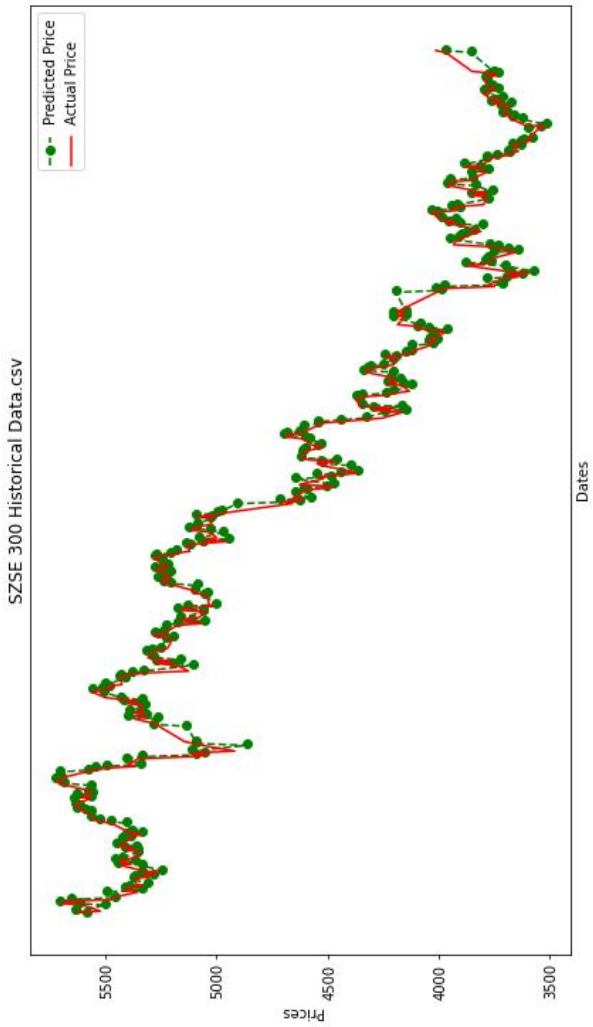


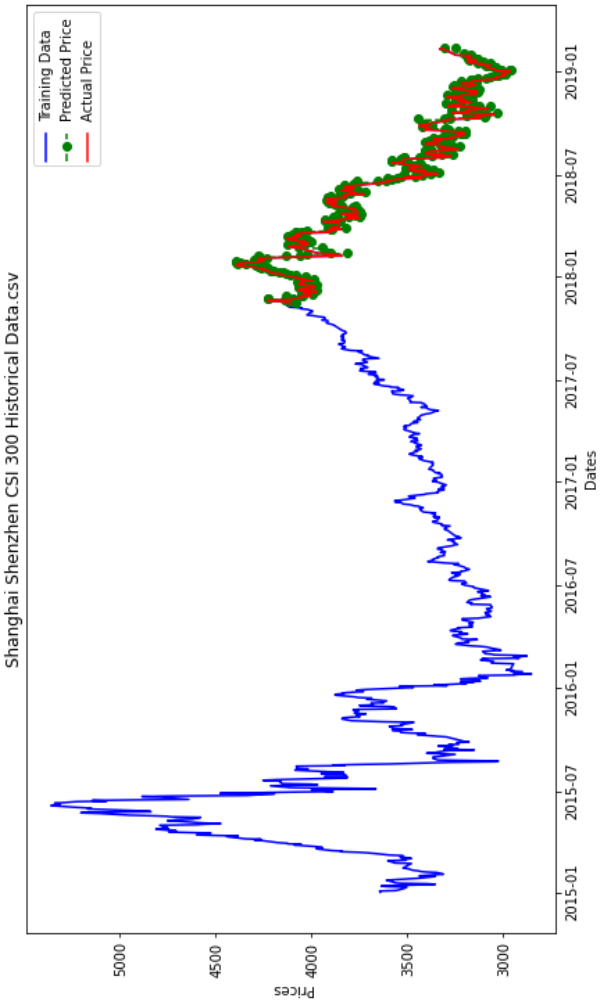
LAMPIRAN E: GAMBAR HASIL PREDIKSI ARIMA

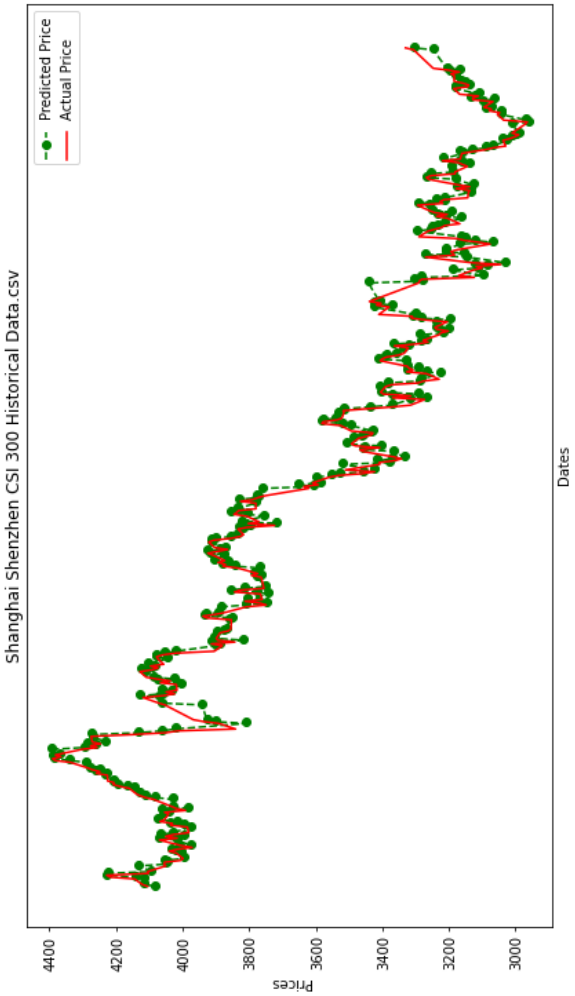


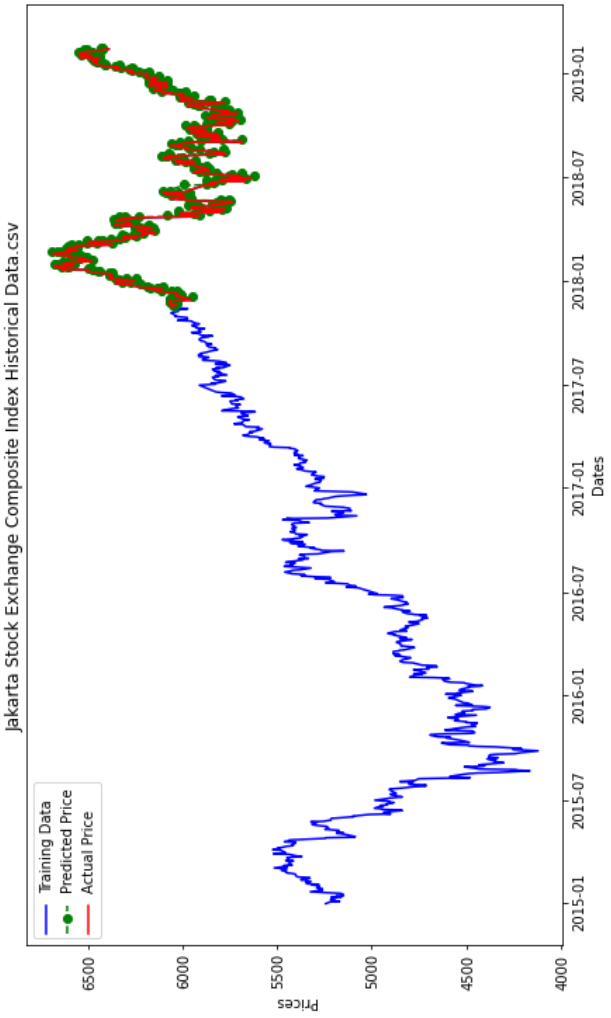

SZSE 200 Historical Data.csv
Predicted Price
Actual Price
Prices
4750
4500
4250
4000
3750
3500
3250
3000
Dates

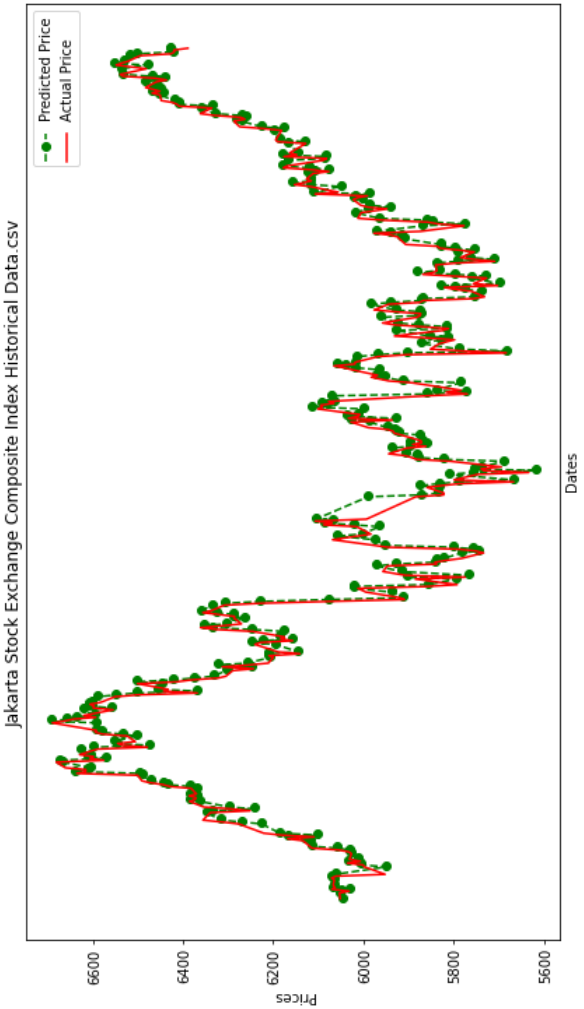

SZSE 300 Historical Data.csv
Training Data
Predicted Price
Actual Price
Prices
8000
7000
6000
5000
4000
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
Dates


SZSE 300 Historical Data.csv
Predicted Price
Actual Price
Prices
Dates
5500
5000
4500
4000
3500

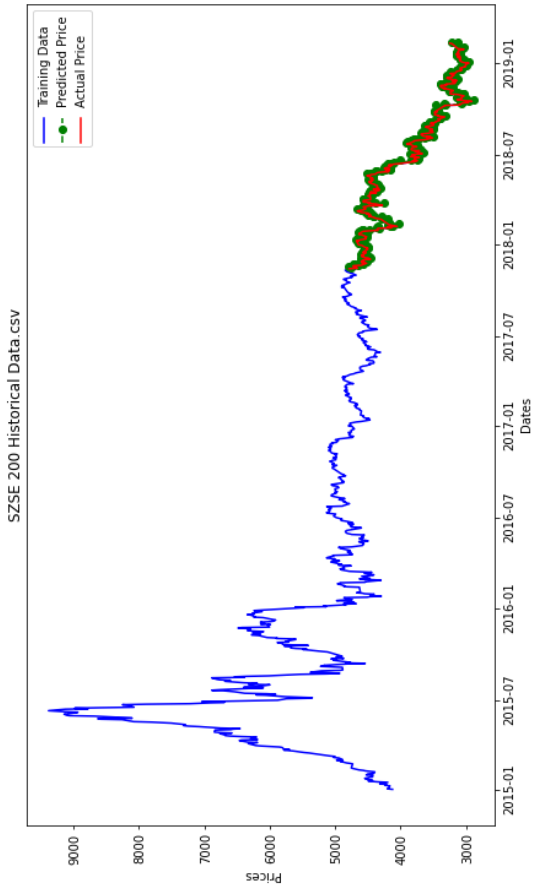

Shanghai Shenzhen CSI 300 Historical Data.csv
Training Data
Predicted Price
Actual Price
Prices
5000
4500
4000
3500
3000
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
Dates

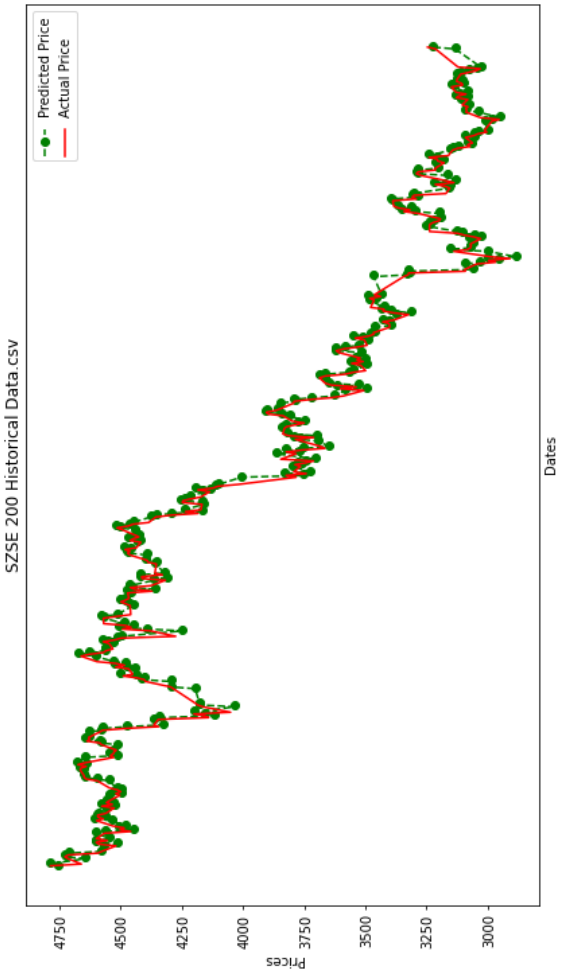

Shanghai Shenzhen CSI 300 Historical Data.csv
Predicted Price
Actual Price
Prices
Dates
4400
4200
4000
3800
3600
3400
3200
3000

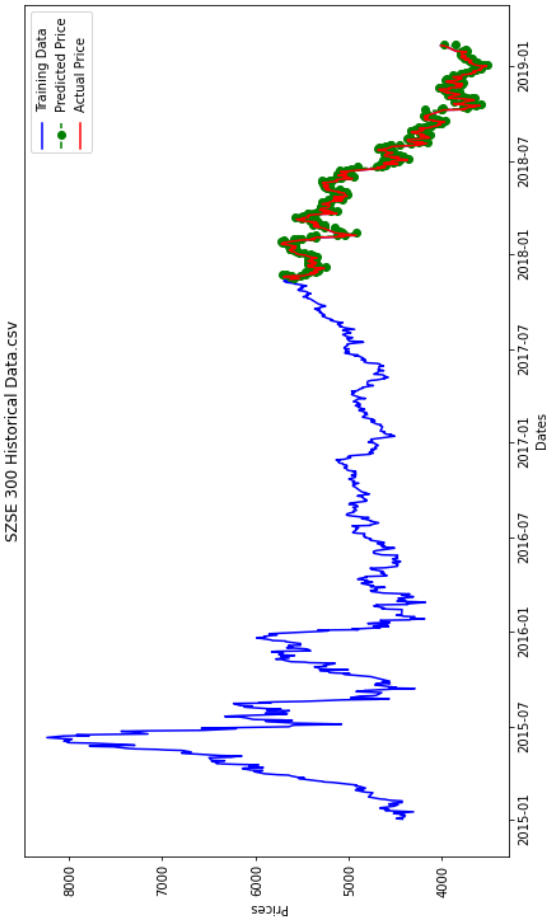

Jakarta Stock Exchange Composite Index Historical Data.csv
Training Data
Predicted Price
Actual Price
Prices
6500
6000
5500
5000
4500
4000
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
Dates

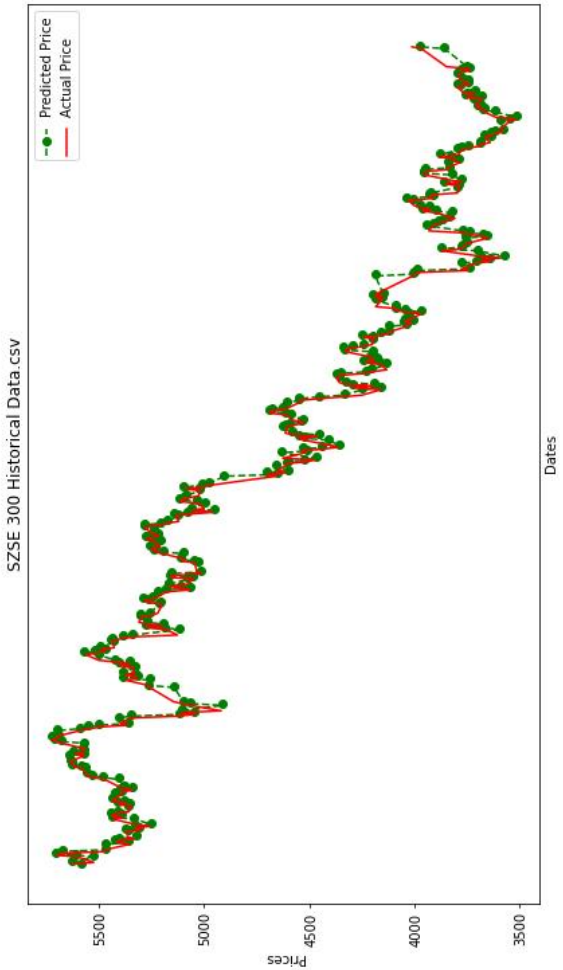

Jakarta Stock Exchange Composite Index Historical Data.csv
Predicted Price
Actual Price
Prices
6600
6400
6200
6000
5800
5600
Dates

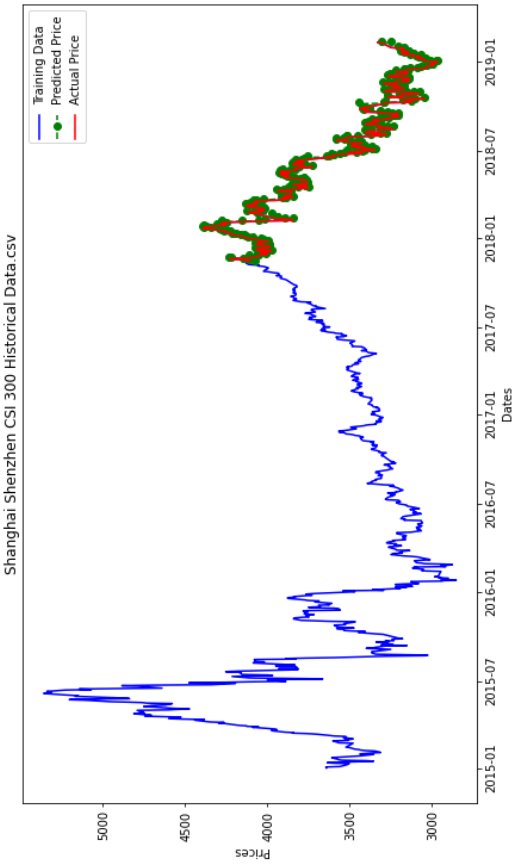
LAMPIRAN F: GAMBAR HASIL PREDIKSI LSTM

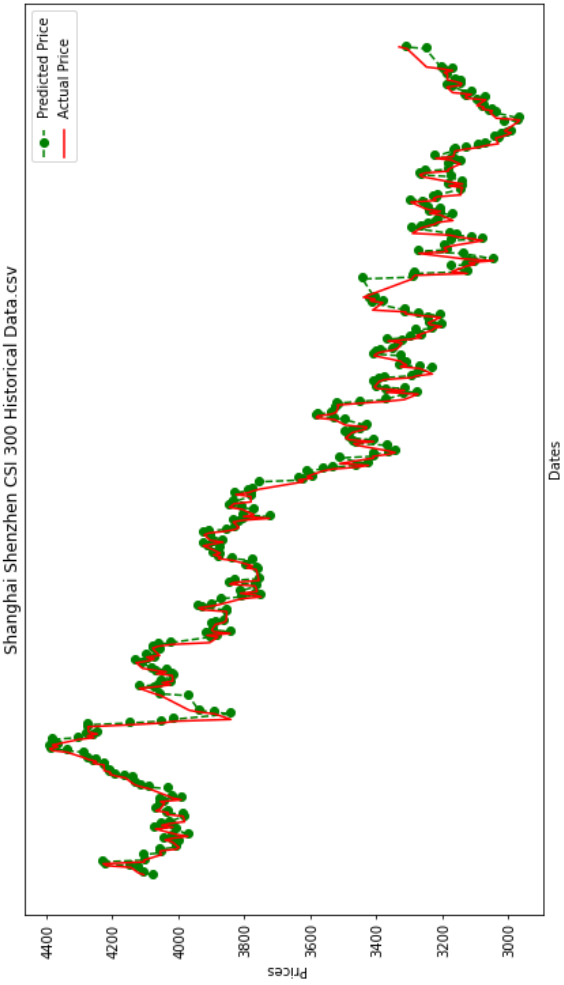


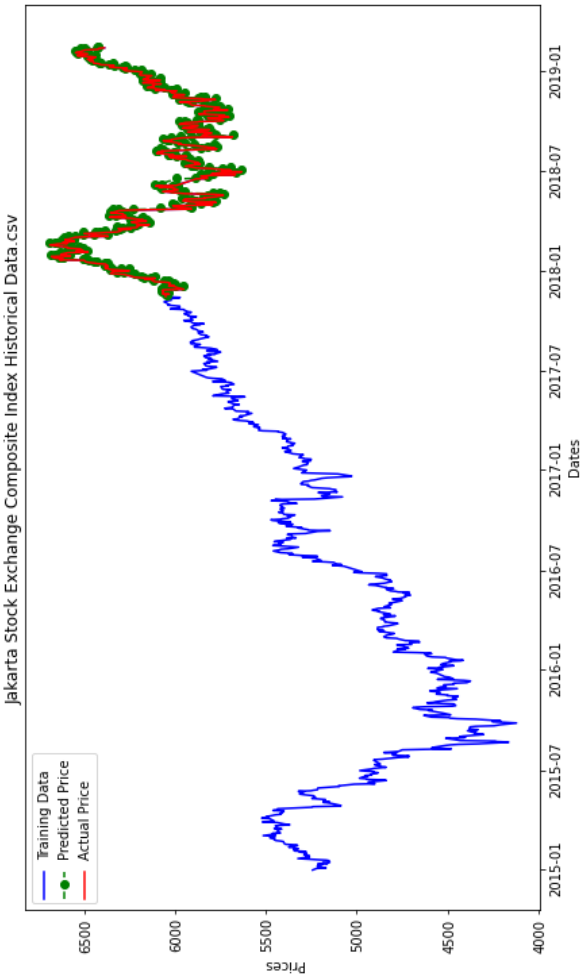

Predicted Price
Actual Price
SZSE 200 Historical Data.csv
Prices
Dates
4750
4500
4250
4000
3750
3500
3250
3000

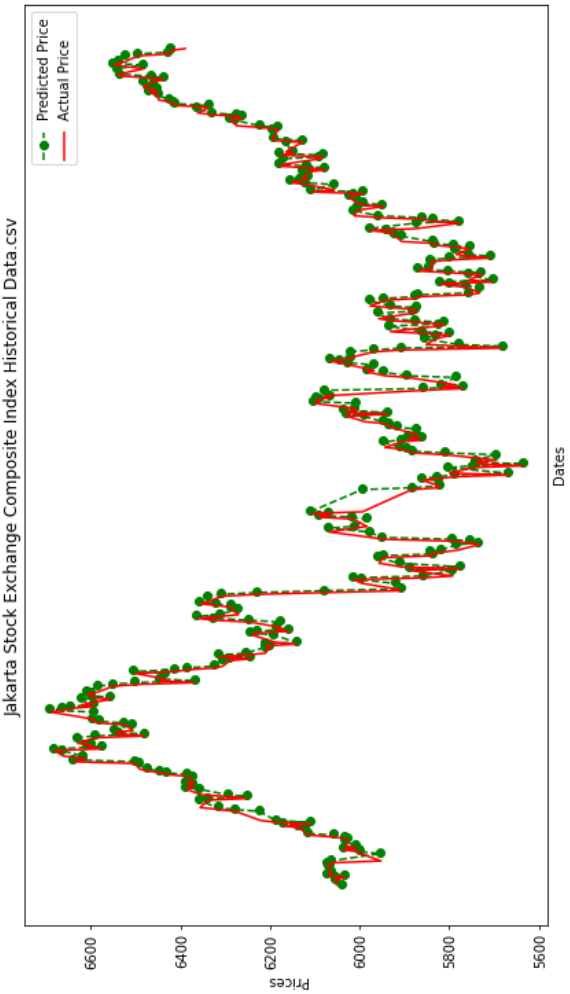

SZSE 300 Historical Data.csv
Training Data
Predicted Price
Actual Price
Prices
8000
7000
6000
5000
4000
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
Dates


SZSE 300 Historical Data.csv
Predicted Price
Actual Price
Prices
5500
5000
4500
4000
3500
Dates

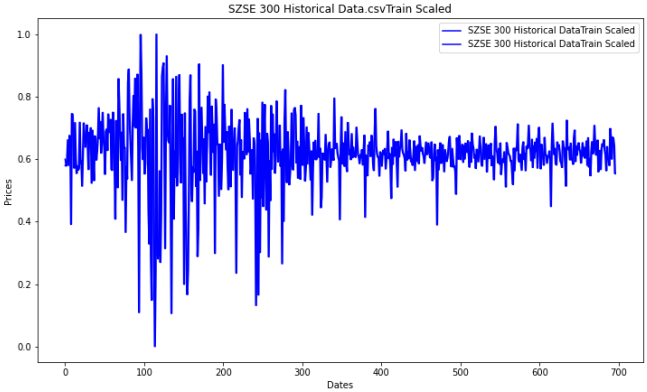

Shanghai Shenzhen CSI 300 Historical Data.csv
Training Data
Predicted Price
Actual Price
Prices
5000
4500
4000
3500
3000
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
Dates

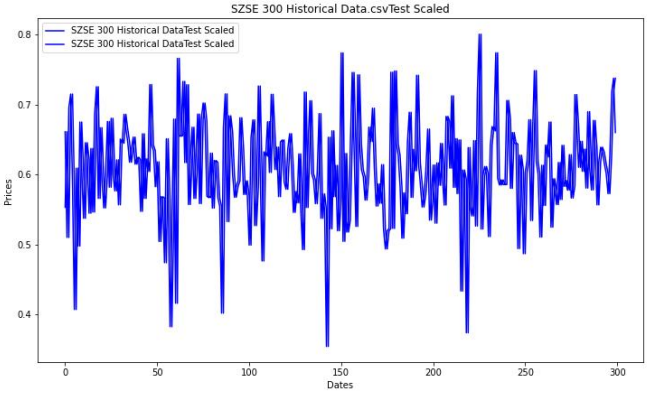

Predicted Price
Actual Price
Shanghai Shenzhen CSI 300 Historical Data.csv
Prices
Dates
4400
4200
4000
3800
3600
3400
3200
3000


Jakarta Stock Exchange Composite Index Historical Data.csv
Training Data
Predicted Price
Actual Price
Prices
6500
6000
5500
5000
4500
4000
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
Dates

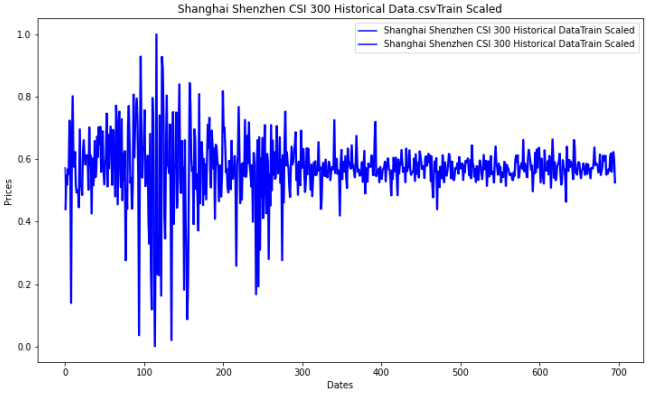

Jakarta Stock Exchange Composite Index Historical Data.csv
Predicted Price
Actual Price
Prices
6600
6400
6200
6000
5800
5600
Dates

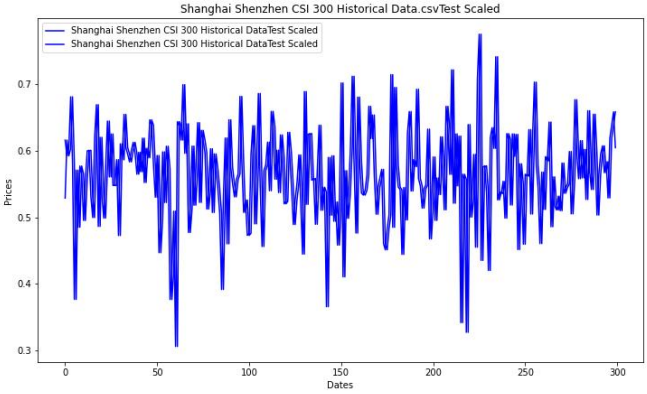
LAMPIRAN G: HASIL PRAPROSES TERHADAP DATA SZSE 300



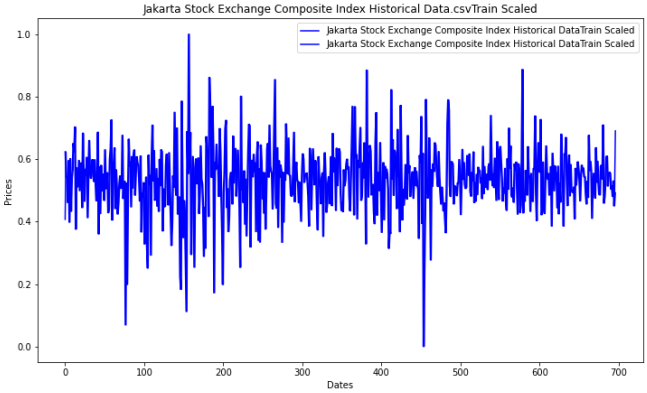


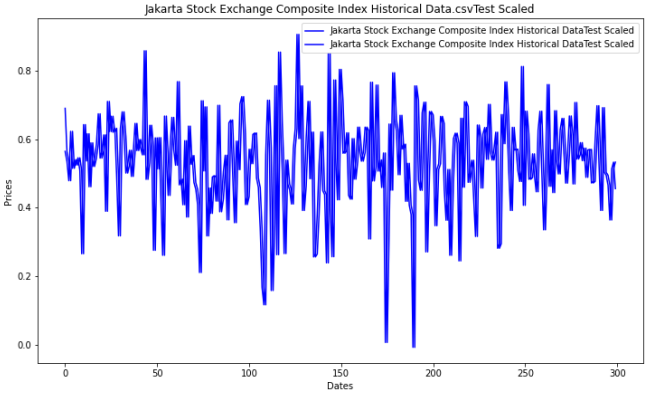
LAMPIRAN H: HASIL PRAPROSES TERHADAP DATA CSI 300





LAMPIRAN I: HASIL PRAPROSES TERHADAP DATA JKSE





BIODATA PENULIS

Muhammad Aufa Wibowo, lahir pada 16 Mei 1998 di Malang. Penulis merupakan mahasiswa yang sedang menempuh studi di Departemen Informatika Institut Teknologi Sepuluh Nopember (ITS). Beberapa bidang minat yang digeluti oleh penulis seperti pengembangan web, analisis data, kecerdasan buatan, dan penerapan ilmu komputer dalam investasi saham. Penulis pernah melakukan magang di Laboratorium CITI Departemen Industrial Management, National Taiwan University of Science and Technology, Taiwan yang saat ini penulis jadikan topik kerja praktik. Penulis juga aktif berorganisasi di dalam kampus dengan menjadi staff Departemen Hubungan Luar pada Himpunan Mahasiswa Teknik Computer-Informatika (HMTC).