Penelitian Mandiri

Prediksi Indeks Harga Saham dengan Recurrent Neural Network

Penulis:
Achmad Maulana Gani
20917009

Dosen Pengampu:
Dr.rer.nat. Sparisoma Viridi S.Si.

Sains Komputasi
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Bandung

October 2019

Contents

List of Figures

List of Tables

Chapter 1

Pendahuluan

1.1 Latar Belakang

1.2 Tujuan

1.3 Rumusan Masalah

1.4 Ringkasan

Dalam modul ini

Chapter 2

Dasar Teori

Neural network adalah simulasi dari neuron-neuron biologis. Serupa dengan dendrit di dalam neuron biologis, kanal data memiliki peran penting untuk membawa informasi diantara neuron. Kekuatan kanal ini dapat dianggap sebagai berat (*weights*) pada jaringan neural network.

2.1 Pengenalan Saham

Saham merupakan salah satu instrumen keuangan yang cukup menarik bagi kalangan investor di sebuah pasar modal ataupun bagi perusahaan untuk mendapatkan dana bagi kepentingan perusahaan. Saham dapat didefinisikan sebagai tanda penyertaan atau pemilikan seorang atau badan dalam suatu perusahaan. Wujud saham adalah selembar kertas yang menerangkan bahwa pemiliknya merupakan pemegang saham yang sah. Dengan demikian kalau seorang investor membeli saham, maka ia pun menjadi pemilik perusahaan, dan memiliki andil pada aset perusahaan.

Saham memiliki jenis instrumen pasar modal yang bersifat kepemilikan dan dijelaskan sebagai berikut

1. Saham biasa : Merupakan saham yang menempatkan pemiliknya sebagai pemegang saham paling rendah terhadap hak atas harta kekayaan perusahaan apabila perusahaan tersebut dilikuidasi dan pembagian dividen
2. Saham preferen : Saham preferen merupakan saham yang memiliki karakteristik gabungan antara obligasi dan saham biasa, karena bisa menghasilkan pendapatan tetap, tetapi juga bisa tidak mendatangkan hasil seperti yang dikehendaki investor.

2.1.1 Keuntungan Saham

Pada dasarnya, ada dua keuntungan yang diperoleh pemodal dengan membeli atau memiliki saham. Yang pertama adalah dividen, yaitu pembagian keuntungan yang diberikan perusahaan penerbit saham tersebut atas keuntungan yang dihasilkan perusahaan. Keuntungan yang berbentuk dividen bisa didapatkan pada saat emiten membagikan sebagian laba bersihnya untuk pembayaran dividen kepada para pemegang saham. Periode pembagian bisa dilakukan pada saat setelah selesai penyelenggaraan RUPS (Rapat Umum Pemegang Saham). Besaran dividen biasanya dibayarkan maksimal 30% sampai 40% dari total laba bersih. Adanya rencana pembagian dividen akan memicu kenaikan harga saham tersebut di bursa efek. Manajer investasi yang profesional biasanya selalu memilih saham dari perusahaan yang konsisten membagikan dividennya. Dividen yang dibagikan perusahaan dapat berupa dividen tunai.

Kedua, *Capital Gain*, yang merupakan selisih antara harga beli dengan harga jual. *Capital Gain* terbentuk dengan adanya aktivitas perdagangan saham di pasar sekunder. Keuntungan ini dapat terealisasi apabila manajer investasi melakukan penjualan saham yang harganya mengalami kenaikan dari harga beli sebelumnya. Manajer Investasi yang profesional mampu melakukan analisis dan seleksi pembelian saham yang prospektif pada saat harganya murah atau membeli pada saat pasar mengalami kondisi saham menurun *bearish* dan melakukan penjualan pada saat harganya ada di posisi atas atau pada saat kondisi *bullish*. Praktik *Capital Gain* ini juga sering digunakan ketika melakukan pertukaran *forex*, jadi kedua hal ini memiliki konsep yang sama. *Capital Gain* merupakan sasaran target bagi manajer investasi dalam mendapatkan keuntungan atas investasi sahamnya.

2.1.2 Resiko Saham

Saham dikenal dengan karakteristik *high risk-high return* yang memiliki arti bahwa saham merupakan surat berharga yang memberikan peluang keuntungan tinggi namun juga beresiko tinggi. Saham memungkinkan pemodal untuk mendapatkan *return* atau keuntungan dalam jumlah besar dalam waktu singkat. Namun seiring dengan tingkat fluktuasi harga saham, maka saham juga dapat menyebabkan pemodal mengalami kerugian yang cukup besar dalam waktu singkat.

Resiko yang dihadapi pemodal dengan kepemiikan sahamnya adalah sebagai berikut:

1. *Tidak Mendapat Dividen.* Perusahaan akan membagikan dividen jika operasi perusahaan menghasilkan keuntungan. Dengan demikian perusahaan tidak dapat membagikan dividen jika perusahaan tersebut mengalami kerugian. Dengan

demikian potensi keuntungan pemodal untuk mendapatkan dividen ditentukan oleh kinerja perusahaan terebut.

2. *Capital loss.* Dalam aktivitas perdagangan saham, tidak selalu pemodal mendapatkan *capital gain* alias keuntungan atas saham yang dijualnya. Adakalanya pemodal harus menjual saham dengan harga jual lebih rendah dari harga beli

3. *Likuidasi Perusahaan.* Istilah ini memiliki arti bahwa perusahaan mengalami kebangkrutan, maka akan berdampak secara langsung kepada saham perusahaan tersebut. Sesuai dengan peraturan pencatatan saham di dalam bursa efek, maka jika suatu perusahaan bangkrut atau dilikuidasi, maka secara otomatis saham tersebut akan dikeluarkan dari bursa. Dalam kondisi likuidasi, pemegang saham akan menempati posisi lebih rendah dibanding kreditor atau pemegang obligasi, artinya setelah semua aset perusahaan tersebut dijual, terlebih dahulu dibagikan kepada para kreditor, dan jika masih terdapat sisa, akan dibagikan kepada para pemegang saham

4. *Delisting.* Resiko yang dapat dihadapi para pemodal adalah jika saham perusahaan dikeluarkan dari pencatatan bursa efek. Suatu perusahaan bisa dikeluarkan dari bursa pada umumnya karena kinerja yang buruk seperti mengalami kerugian beberapa tahun.

5. *Saham ditahan.* Saham ditahan atau *suspend*, kondisi ini merupakan penghentian perdagangan saham oleh otoritas bursa efek dalam waktu singkat. Hal tersebut dilakukan ototritas bursa jika suatu saham mengalami lonjakan harga yang luar biasa yang mengharuskan otoritas bursa menghentikan sementara perdagangan saham untuk dikonfirmasi kebenarannya kepada perusahaan tersebut

2.2 Struktur Neural Network

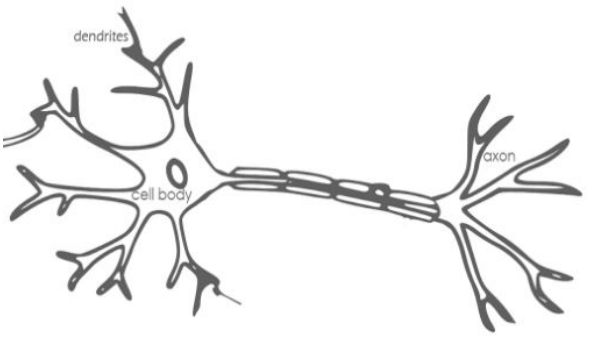

FIGURE 2.1: Struktur neuron biologis

Sebuah model sederhana yang dikembangkan dengan pendekatan dari otak manusia yang bertujuan untuk mengenal pola-pola. Interpretasi terhadap data sensor dilakukan melalui mesin persepsi, melabelkan input, atau mengelompokkan input. Pola-pola yang dikenal oleh algoritma ini bersifat numerikal, yang jika diartikan lebih luas ke dalam dunia nyata memiliki contoh gambar, suara, teks, atau fungsi waktu. Neuron dalam struktur neural network dikenal dengan sebuah neuron artifisial. Menggunakan fungsi batas (*threshold*) untuk menghasilkan sebuah keluaran berupa nilai 0 atau 1 dan berfungsi sebagai pengklasifikasi atau *classifier*. Model ini memiliki berat yang tetap dan tidak melakukan proses belajar. Ini merupakan neural network yang paling dasar dari sebuah perspektif biologis. Misalkan ada sebuah neuron batas dengan x^1 dan x^2 sebagai input dan t menjadi keluaran target awal. Neuron memiliki berat w^1 dan w^2 sebagai kanal dari data input. Untuk menghitung keluaran dari input u dilakukan

$$u = w^1 x^1 + w^2 x^2 \tag{2.1}$$

Input total dari keseluruhan input dihitung dan sebuah nilai sembarang dari nilai batas dipilih berdasarkan nilai di antara batas nilai terendah dan tertinggi dari input total. Misalkan nilai batas u^0 dipilih. Nilai keluaran akhir y akan dilakukan dengan pola sebagai berikut:

$$y = u < u^0 \quad \text{atau} \quad y = u \geq u^0 \tag{2.2}$$

Klasifikasi ditentukan dari nilai batas menjadi 2 kategori yang berbeda. Fungsi batas dari persamaan diatas yang mengklasifikasi input menggunakan sebuah neuron sederhana. Gambar dibawah adalah gambaran proses klasifikasi nilai keluaran dengan fungsi batas yang ditandai dengan garis putus-putus.

Inspirasi dari struktur neuron biologis di atas, sebuah neuron dapat dijelaskan dengan diagram pada 2.3. Sinyal-sinyal input yang dibawa oleh kanal-kanal input akan dijumlahkan atau diakumulasikan ($\sum$), kemudian memproses input tersebut menjadi sebuah output melalui $[f(\sum)]$. Ada 2 aspek-aspek penting dari model-model neural yang akan menjadi dasar-dasar dari neural network:

1. Jika 2 neuron aktif pada saat yang bersamaan, maka mereka akan mengeksitasi masing-masing individu yang akan membuat intensitas dari koneksi di antara mereka berkembang.

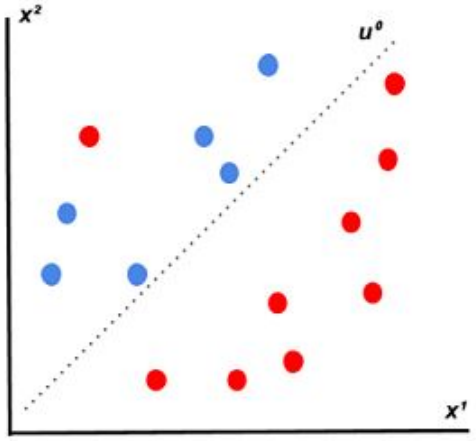

FIGURE 2.2: Klasifikasi kleuaran berdasarkan threshold

2. Jumlah dari sinyal-sinyal yang diterima sebuah neuron berdampak pada aktivitasnya, dimana kekuatan dari koneksi yang dilewati sinyal input akan proporsional dengan sinyal keluaran.

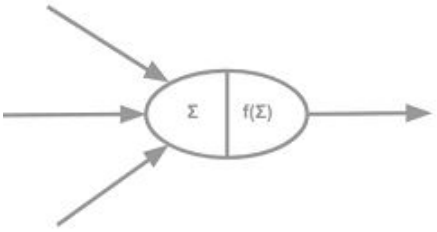

FIGURE 2.3: Struktur neuron sederhana

2.3 Recurrent Neural Network

Recurrent Neural Network (RNN) atau jaringan saraf berulang adalah jenis arsitektur jaringan saraf tiruan yang pemrosesannya dipanggil berulang-ulang untuk memproses masukan yang biasanya adalah data sekuensial. RNN masuk ke dalam kategori *deep learning* karena data diproses melalui banyak lapisan (*layer*). RNN telah mengalami kemajuan yang pesat dan telah merevolusi bidang-bidang seperti pemrosesan bahasa alami *NLP*, pengenalan suara, sintesis musik, pemrosesan data finansial waktu serial, analisis deret DNA, analisis video, dan sebagainya. Kelihaian metode ini dapat dikenal pada subbab dibawah ini.

2.3.1 Pemrosesan Data Sekuensial

Data sekuensial mempunyai karakteristik dimana sampel diproses dengan suatu urutan, misalnya waktu, dan suatu sampel dalam urutan mempunyai hubungan erat satu dengan yang lain. Contoh-contoh data sekuensial dan aplikasinya adalah:

1. Rangkaian kata-kata dalam penerjemahan bahasa
2. Sinyal audio dalam pengenalan suara
3. nada-nada dalam sintesis musik
4. Rangkaian kata-kata dalam klasifikasi sentimen
5. Deret DNA dalam pemrosesan rangkaian DNA
6. Rangkaian gambar-gambar pada pengenalan aktivitas video
7. Kata-kata dalam pengenalan nama entitas

Pemrosesan data sekuensial seperti di atas tidak cocok diakukan dengan metode-metode konvensional seperti jaringan *feedforward* yang sudah dikenal seperti model linear, *multi-layer perceptron* (MLP), dan CNN. Kendala utamanya adalah:

1. Model-model ini tidak memiliki ingatan memori, setiap sampel akan diproses secara serupa tanpa ada pertimbangan dari sampel-sampel sebelumnya.
2. Model-model tersebut mengasumsikan data yang IID *Independent and Identically Distributed*, dan hal ini tidak dapat dipenuhi oleh data sekuensial karena suatu sampel mempunyai ketergantungan yang erat dengan sampel-sampel lainnya.
3. Model-model ini memroses masukan dengan panjang input yang tetap, sedangkan data sekuensial mempunyai panjang yang tidak tentu, dan bisa jadi sangat panjang.
4. Model-model ini tidak bisa mengaplikasikan fitur yang dipelajari di satu posisi ke posisi lain

2.3.2 Definisi RNN

RNN memroses input secara sekuensial, sampel per sampel. Dalam tiap pemrosesan, keluaran yang dihasilkan tidak hanya merupakan fungsi dari sampel itu saja, tapi juga berdasarkan parameter internal yang merupakan hasil dari pemrosesan sampel-sampel

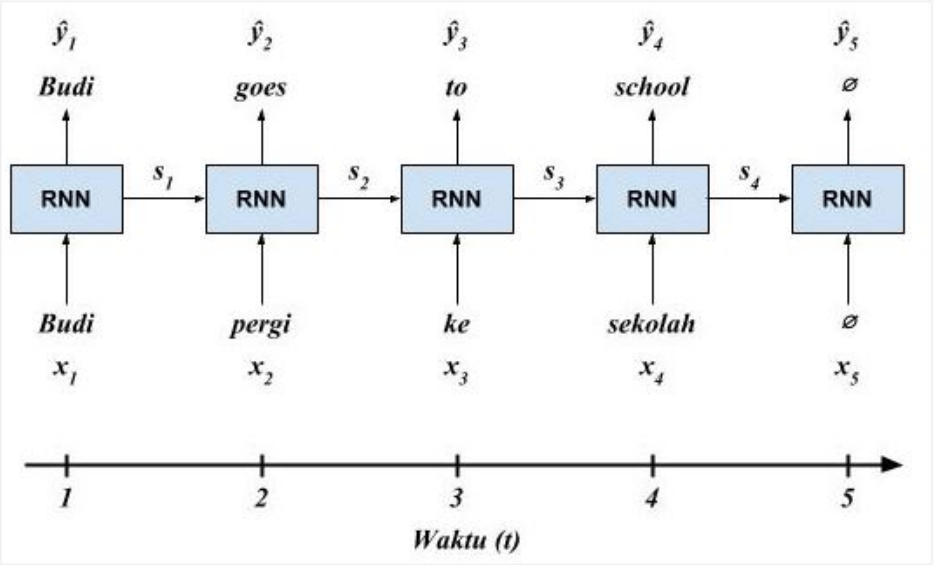

FIGURE 2.4: Struktur jaringan RNN

sebelumnya (*bidirectional RNN*). Ilustrasi dibawah merupakan ilustrasi bagaimana RNN bekerja 2.4.

1. Sumbu horizontal adalah waktu, direpresentasikan dengan simbol t. Pemrosesan RNN dilakukan dari kiri ke kanan. Selanjutnya t dapat dianggap sebagai langkah waktu *time step*.

2. Pemrosesan input RNN berupa "kata demi kata". Input ini disimbolkan dengan variabel $x_1, x_2, ..., x_4$, atau secara umum dianggap x_t.

3. Variabel keluaran juga serupa dengan input yang berarti "kata demi kata", dan disimbolkan dengan y1...y5.

4. Dalam tiap pemrosesan, RNN akan menyimpan parameter internal yaitu s_t, yang diberikan dari satu langkah waktu ke langkah waktu berikutnya. Inilah yang disebut memori dalam RNN.

Dengan demikian, RNN dapat digeneralisasikan seperti gambar dibawah ini 2.5.

2.3.3 Long Short-Term Memory (LSTM)

Aplikasi dalam RNN memungkingkan untuk mempelajari pola-pola temporal dari data sekuensial. Sifat-sifat memori dari RNN tidak dimunculkan pada metode-metode jaringan *feedforward* sebelumnya [1]. Namun, permasalahan pada gradien muncul ketika melatih

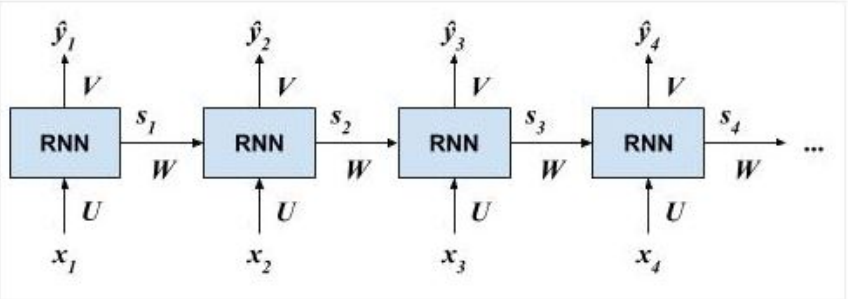

FIGURE 2.5: Formula RNN secara umum

jaringan-jaringan standar dari RNN yang memiliki arti bahwa jaringan-jaringan tersebut mengalami kesulitan dalam menjaga informasi dalam waktu yang lama. Unit LSTM mulai dikenalkan kepada umum sebagai sebuah metode alternatif untuk mempelajari pola-pola sekuensial. LSTM memiliki mekanisme untuk meregullasi arus informasi. Jumlah informasi yang datang yang ingin dijaga ditentukan secara sistematis pada setiap *time step* dan berakibat bahwa LSTM dapat mengingat pola-pola temporal dalam rentang waktu yang lebih luas.

Gambar dibawah ini merupakan struktur dari blok memori LSTM yang terdiri dari sel memori dan gerbang 2.6. x_t dan h_t merupakan input dan kondisi tersembunyi, pada waktu t dan f_t, i_t dan o_t adalah gerbang yang dikenal dengan gerbang *forget*, gerbang input, dan gerbang keluaran. n_t merupakan kandidat dari input yang disimpan, dan jumlah yang disimpan tersebut kemudian akan dikontrol oleh sebuah gerbang input. Kalkulasi dari setiap gerbang, kandidat input, kondisi sel, dan kondisi tersembunyi dilakukan oleh formula di bawah ini:

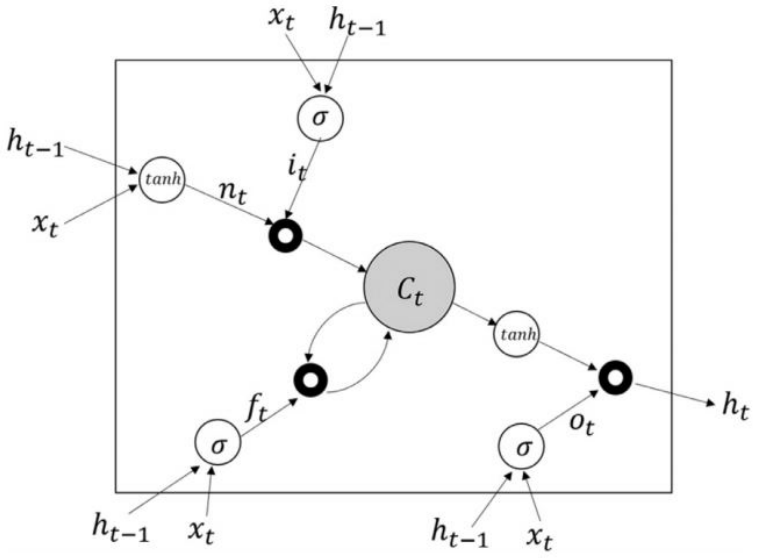

FIGURE 2.6: Struktur jaringan LSTM

$$f_t = \sigma(A_f x_t + B_f h_{t-1} + b_f), \tag{2.3}$$

$$i_t = \sigma(A_i x_t + B_i h_{t-1} + b_i), \tag{2.4}$$

$$o_t = \sigma(A_o x_t + B_o h_{t-1} + b_o), \tag{2.5}$$

$$n_t = tanh(A_n x_t + B_n h_{t-1} + b_n), \tag{2.6}$$

$$c_t = c_{t-1} * f_t + n_t * i_t, \tag{2.7}$$

$$h_t = tanh(c_t) * o_t \tag{2.8}$$

Dimana A dan B adalah matriks bobot dan b adalah vektor-vektor bias. Simbol * merupakan perkalian *element-wise* atau perkalian satu-satu setiap elemen. Fungsi aktivasi tanh digunakan untuk memperbaharui kondisi sembunyi. Fungsi aktivasi Sigmoid untuk gerbang ditandai oleh *sigma* pada 2.6 dan dihitung oleh fungsi sigmoid $\sigma(z) = (1 + e^{-z})^{-1}$, dimana keluaran dari fungsi tersebut berada diantara [0,1]. Jika keluaran bernilai 0, tidak ada informasi yang masuk ke dalam jaringan. Jika keluaran bernilai lebih dari 0, lebih banyak informasi yang dibolehkan, dan jika bernilai 1, semua informasi dapat digunakan. Berdasarkan pada refleksi selektif seperti ini, maka LSTM memiliki kapabilitas dalam proses belajar yang lebih lama untuk pola-pola temporal.

2.4 Perangkat Pendukung

Dalam melakukan penelitian mandiri ini, penulis menggunakan beberapa perangkat pendukung dalam membuat pemrograman. Bahasa pemrograman utama yang digunakan adalah Python dan *library* pendukung yang digunakan dalam analisis kasus-kasus Machine Learning. Bahasa pemrograman yang digunakan dan semua *library* yang digunakan merupakan perangkat lunak *open source* dan dapat langsung diunduh dengan gratis.

2.4.1 Python

Jika tujuan utama adalah menjadi pemrogram yang sukses, pemrogram harus memilikie pengetahuan mengenai banyak hal. Namun untuk mendalami *Machine Learning* dan *Data Science*, cukup dengan mempelajari setidaknya 1 bahasa pemrograman. Python menawarkan pilihan bagi para pemrogram untuk fokus menuju bidang *Machine Learning* dan *Data Science*. Python sendiri merupakan bahasa pemrograman yang mudah digunakan dan mulai mendapatkan popularitas sejak kemunculannya pada tahun 1990. Popularitas bahasa ini mendapat posisi keempat sebagai bahasa paling populer untuk pemrograman.

```
print('hello world')
```

FIGURE 2.7: Sintaks python Hello World

Python merupakan bahasa pemrograman yang mudah digunakan, kuat, dan serbaguna, yang memberikan sebuah kesan yang baik untuk pemrogram pemula maupun pemrogram yang sudah mahir. Kemampuan python dalam memberikan kemudahan untuk membuat program merupakan pilihan pertama bagi para pemula. Bahasa ini memiliki kelebihan dalam sintaks, para pemrogram tidak perlu membuang waktu dengan sintaks-sintaks yang membingungkan seperti pada bahasa pemrograman lainnya seperti Java atau C.

Perusahaan-perusahaan perangkat lunak cenderung memilih bahasa Python karena fitur-fitur yang serbaguna dan sintaks yang ringkas. Sekitar 14% dari para pemrogram menggunakan bahasa ini dengan sistem operasi UNIX, Linux, Windows, dan Mac OS. Bahasa ini memiliki karakteristik-karakteristik seperti:

1. Interaktif
2. Interpretasi
3. Modular
4. Dinamis
5. Berbasis Objek
6. Portabel
7. Level tinggi
8. Ekstensibel dalam C++ dan C

Bahasa python memiliki aplikasi yang beragam dalam perusahaan pengembangan perangkat lunak seperti *gaming*, kerangka website, pengembangan bahasa, aplikasi grafik desain, dan lain-lain. Beberapa keuntungan bahasa ini dapat dijelaskan pada poin-poin dibawah ini:

1. *Library* pendukung yang ekstensif : Python memiliki *library* standar yang cukup luas yang meliputi sejumlah bidang seperti operasi *string*, Internet, servis website, antarmuka sistem operasi dan protokol.

2. Fitur Integrasi : Python mengintegrasikan Integrasi Aplikasi Enterprise (EAI) yang membuat bahasa ini menjadi mudah unuk mengembangkan servis-servis website dengan menggunakan komponen-komponen COM atau COBRA.

3. Produktivitas : Python merupakan pilihan yang baik dalam membangun aplikasi jaringan multiprotokol. Python mempercepat pengembangan dalam banyak aplikasi dengan fitur integrasi yang kuat, kerangka testing dan kapabilitas kontrol yang kuat.

Pada subbab selanjutnya akan menjelaskan secara singkat mengenai *library* yang digunakan dalam analisis dalam *Machine Learning*.

2.4.2 Numpy

Numpy adalah *library* dari bahasa pemrograman Python, *Numpy* memiliki kegunaan dalam operasi vektor dan matriks. Fiturnya hampir sama dengan bahasa pemrograman MATLAB dalam mengelola *array* dan *array* multidimensi. *Numpy* merupakan salah satu *library* yang digunakan oleh *library* lain seperti *Scikit-Learn* untuk keperluan analisis data.

```
>>> x = np.array([[1,2,3],[4,5,6]], np.int32)
>>> type(x)
<class 'numpy.ndarray'>
>>> x.shape
(2, 3)
>>> x.dtype
dtype('int32')
```

FIGURE 2.8: Sintaks *Numpy*

2.4.3 Pandas

Dengan menggunakan sistem *dataframe*, pemrogram dapa memuat sebuah berkas ke dalam tabel virtual yang mirip dengan sistem *Excel* dengan menggunakan *Pandas*. Dengan menggunakan *Pandas*, pemrogram dapa mengolah suatu data dan mengolahnya seperti fungsi *join*, *distinct*, *group by*, agregasi dan teknik seperti SQL hanya saja dilakukan pada tabel yang dimuat dari berkasi ke dalam RAM. *Pandas* juga dapat membaca berkas dari berbagai macam format seperti .txt, .csv, .tsv, dan lainnya.

```
>>> import pandas as pd
... pd.set_option('display.mpl_style', 'default')
... figsize(15, 5)
... broken_df = pd.read_csv('../data/bikes.csv')
... # Look at the first 3 rows
... broken_df[:3]
```

FIGURE 2.9: Sintaks *Pandas*

2.4.4 Matplotlib

Data yang diolah tentu lebih baik jika ditampilkan dengan sejumlah grafik aga lebih menarik. Matplotlib dapat mengatasi permasalah seperti kasus tersebut dengan memvisualisasikan data dengan lebih indah dan rapi. *Matplotlib* dapat menampilkan data secara 2 dimensi ataupun secara 3 dimensi sesuai dengan kebutuhan. *Library* ini juga merupakan *library* yang paling banyak digunakan untuk pengolahan *Data Science* dengan menyajikan data dengan representasi visual.

```
>>> import matplotlib.pyplot as plt
... plt.plot([1,2,3,4])
... plt.ylabel('some numbers')
... plt.show()
```

FIGURE 2.10: Sintaks *Matplotlib*

2.4.5 Scikit-Learn

Analisis *Machine Learning* biasanya berbasis statistika dan ada juga yang non statistika. Contoh model yang berbasis statistika adalah *Support Vector Machine* (SVM) dan regresi linear. *Scikit-learn* dapat memberikan sejumlah fitur untuk keperluan analisis *Data Science* seperti algoritma regresi, Naive Bayes, Clustering, Decision Tree, *pipeline*, alat praproses data, dan lainnya. Scikit0Learn memiliki dokumentasi yang lengkap dan

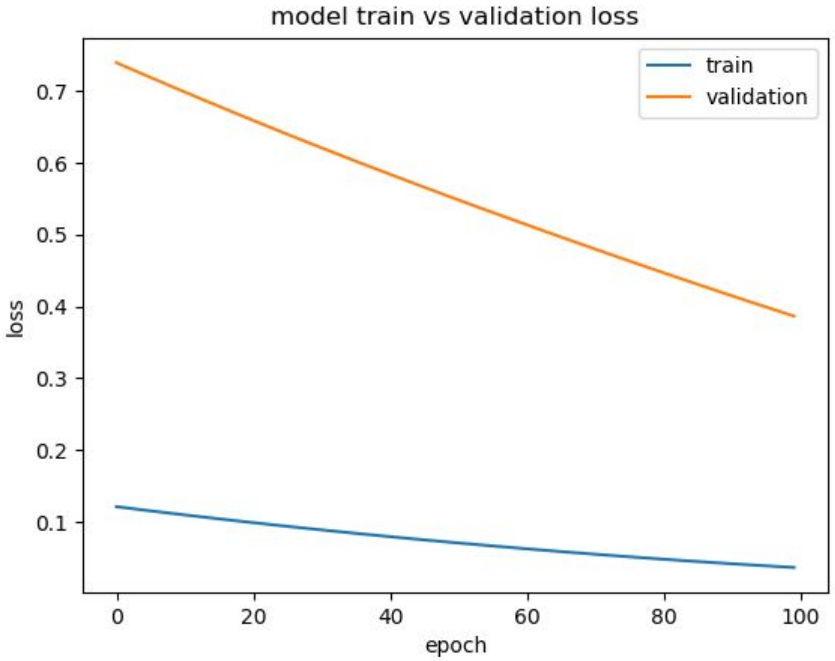

FIGURE 2.11: Visualisasi Data dengan Matplotlib

kontributor yang cukup banyak. *Scikit-Learn* menyediakan ekstensi juga untuk permasalahan *fuzzy logic* dan *computer vision*.

```
>>> from sklearn import svm
... clf = svm.SVC(gamma=0.001, C=100.)
... clf.fit(digits.data[:-1], digits.target[:-1])
... SVC(C=100.0, cache_size=200, class_weight=None, coef0=0.0,
...   decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
...   max_iter=-1, probability=False, random_state=None, shrinking=True,
...   tol=0.001, verbose=False)
```

FIGURE 2.12: Sintaks Scikit-Learn dengan model SVM

2.4.6 Keras

Keras adalah API neural network yang ditulis dengan menggunakan Python dan memiliki kapabilitas bekerja diatas TensorFlow, CNTK, atau Theano. Keras memiliki konsep untuk pengembangan dengan fokus kepada eksperimentasi yang cepat, dengan kata lain mengubah ide-ide menjadi sebuah hasil dengan jeda sekecil mungkin. Pengguna disarankan menggunakan Keras jika ingin menggunakan *library* deep learning yang mensupport jaringa konvolusional dan jaringan *recurrent*, atau kombinasi keduanya, dan juga purwarupa yang cepat dan mudah.

Struktur data utama Keras adalah model, sebuah cara untuk mengatur lapisan. Model yang paling sederhana dalam Keras adalah model Sequential. sebuah lapisan bertumpuk

linear. Untuk menggunakan arsitektur kompleks lainnya, pengguna bisa menggunakan API Keras fungsional yang memberikan keleluasaan untuk membangun lapisan sembarang. Sebuah model sekuensial lengkap dapat dilihat pada gambar dibawah ini 2.13.

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
# x_train and y_train are Numpy arrays --just like in the Scikit-Learn API.
model.fit(x_train, y_train, epochs=5, batch_size=32)
model.train_on_batch(x_batch, y_batch)
loss_and_metrics = model.evaluate(x_test, y_test, batch_size=128)
classes = model.predict(x_test, batch_size=128)
```

FIGURE 2.13: Model sekuensial sederhana dengan modul Keras

Chapter 3

Tutorial Prediksi Pasar Saham

Tutorial ini akan dibagi menjadi 2 subbagian, yang pertama adalah kasus data 1 variabel (univariat) dan banyak variabel (multivariat).

3.1 Akuisisi Data

Langkah pertama yang penting untuk dilakukan adalah memperoleh data. Dataset yang digunakan dalam penelitian ini berasal dari data nilai dari pasar modal yang dapat diunduh melalui website yahoo dengan mengunduh data historis. Perusahaan yang digunakan adalah data indeks saham harga gabungan (JKSE) dan bank BCA (BBCA.JK). Setiap data mengandung informasi seperti tanggal, harga buka, harga tutup, harga tinggi, harga rendah, dan volume, dengan format berkas menggunakan .csv. Dari dataset tersebut, data yang akan digunakan adalah data harga tutup untuk kasus 1 variabel karena investor lebih mengacu kepada harga tutup jika berkaitan dengan pengambilan keputusan. Kasus yang kedua adalah semua data digunakan untuk kasus multivariabel, untuk contoh yang kedua ini hasil yang diperoleh cenderung lebih akurat namun lebih sulit untuk diimplementasikan. Data yang dihimpun berasal dari tahun 2004 hingga tahun 2019 dan mengandung data sejumlah 3906 hari. Untuk pembagian data training, data validasi, dan data tes menggunakan rasio 8:1:1.

3.2 Normalisasi Data

Data yang telah dibagi menjadi data training, data validasi, dan data tes kemudian akan dinormalisasi menjadi rentang data antara 0 dan 1. Normalisasi data dilakukan untuk

mengubah data saham menjadi data dengan skala yang standar. Dalam Machine Learning, tidak semua data perlu untuk dinormalisasi. Data yang perlu dinormalisasi biasanya ketika variabel atau fitur-fitur memiliki rentang yang berbeda. Sebagai contoh, untuk kasus data saham, harga buka, harga tutup, harga tinggi, harga rendah memiliki rentang nilai yang serupa yaitu orde 10^3, namun berbeda untuk data volume yang memiliki data dengan orde 10^8. Proses normalisasi dapat dilakukan dengan menggunakan persamaan di bawah ini:

$$x_{norm} = (x - x_{min})/(x_{max} - x_{min}) \tag{3.1}$$

dimana x_{norm} adalah nilai normalisasi, x_{min} dan x_{max} adalah nilai minimum dan nilai maksimum dari data training. Data yang telah dinormalisasi akan digunakan sebagai input kedalam jaringan LSTM dengan ukuran 100. Keluaran dari jaringan tersebut akan melalui proses denormalisasi untuk mengestimasi nilai prediksi yang aktual.

Proses normalisasi dengan menggunakan Python dapat digunakan dengan bantuan *library* Scikit-learn dengan implementasi sebagai berikut:

```
#normalization
x = df_train.loc[:, train_cols].values
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x)
x_test = scaler.transform(df_test.loc[:, train_cols])
```

FIGURE 3.1: Implementasi normalisasi dengan scikit-learn

dimana fungsi MinMaxScaler() berfungsi sebagai model yang akan digunakan dalam normalisasi, jika tidak menggunakan parameter-parameter tertentu, secara otomatis model ini menentukan rentang nilai dari 0 hingga 1. Fungsi *model*.fit_transform() digunakan untuk mengaplikasikan model kedalam suatu data, dalam hal ini adalah dataset saham

3.3 Supervised Learning

Supervised learning adalah ketika ada sebuah data masukan dengan variabel X dan variabel keluaran y dan sebuah algoritma digunakan untuk mempelajari fungsi pemetaan dari masukan menuju keluaran

$$y = f(X) \tag{3.2}$$

Tujuan dari fungsi pemetaan ini adalah untuk mengaproksimasi pemetaan sedemikian rupa sehingga ketika ada masukan data baru (X), prediksi terhadap variabel keluaran

(y) dapat dilakukan dan akurat. Data saham yang disebut di awal bab merupakan data waktu serial, sejumlah angka sekuens yang diurutkan berdasarkan indeks waktu, dalam kasus pasar saham indeks waktu adalah data tanggal. Ini bisa dimodelkan sebagai sebuah *list* atau kolom dengan nilai yang diurutkan. Sebagai contoh data waktu serial adalah sebagai berikut

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \tag{3.3}$$

Permasalahan data supervised terdiri dari pola masukan (X) dan pola keluaran (y), pada dasarnya sebuah algoritma machine learning dapat mempelajari bagaimana cara memprediksi pola-pola keluaran dari pola-pola input. Sebagai contoh data supervised adalah sebagai berikut

X	y
0,1	2
1,2	3
2,3	4
3,4	5
4,5	6
5,6	7
6,7	8
7,8	9

FIGURE 3.2: Data supervised

Salah satu permasalahan yang muncul adalah bagaimana cara mengubah data waktu serial menjadi data supervised, salah satu algoritma yang dapat dilakukan adalah sebagai berikut

```
#turn into supervised learning
x_t, y_t = build_timeseries(x_train, 3)
x_t, y_t = trim_dataset(x_t, BATCH_SIZE), trim_dataset(y_t, BATCH_SIZE)
x_temp, y_temp = build_timeseries(x_test, 3)
x_val, x_test_t = np.split(trim_dataset(x_temp, BATCH_SIZE), 2)
y_val, y_test_t = np.split(trim_dataset(y_temp, BATCH_SIZE), 2)
```

FIGURE 3.3: algoritma mengubah data menjadi data supervised

3.4 Model LSTM

Pada dasarnya model LSTM memerlukan masukan dengan format batch_size, time_steps, features. sebuah array 3 dimensi. Setiap dimensi dapat dijelaskan sebagai berikut

1. **Batch Size** : batch size menunjukkan bahwa berapa banyak sampel dari masukan yang ingin dimasukkan sebelum memperbaharui bobot dari sebuah jaringan saraf

tiruan. Misalkan sebuah data memiliki 100 buah sampel dataset masukan dan model jaringan saraf ingin diperbaharui bobotnya tiap kali menerima sebuah masukan. Untuk kasus seperti maka nilai batch size adalah 1 dan jumlah batch size adalah 100. Sebaliknya, jika model jaringan saraf ingin diperbaharui bobotnya setelah menerima semua sampel, maka nilai batch size adalah 100 dan jumlah batch adalah 1. Perlu diketahui bahwa jika menggunakan batch size yang sangat kecil maka akan mengurangi kecepatan untuk mempelajari data. Di sisi lain jika menggunakan terlalu banyak batch size akan mengurangi kemampuan data untuk mengeneralisasi terhadap data lainnya dan juga menambah konsumsi memori.

2. **Time Steps** : mendefinisikan berapa banyak indeks waktu yang ingin dipelajari. Sebagai contoh untuk kasus ini jika data sampel yang digunakan ingin dimulai dari 10 hari yang lalu, maka time steps nya adalah 10.

3. **Features** : adalah jumlah atribut yang digunakan untuk merepresentasikan setiap time step.

Jadi dalam kasus prediksi saham ini, nilai batch size yang digunakan adalah 10, nilai time steps adalah 3, dan nilai features adalah 5 yang terdiri dari harga buka, tutup, tinggi, rendah, dan volume.

Kemudian dalam mendesain model yang akan digunakan, disini menggunakan model Sequential(), yaitu pemodelan yang berisikan lapisan-lapisan linear disusun bertumpuk. Dengan kata lain, setelah model pertama selesai dijalankan, akan dilanjutkan dengan model kedua, dan seterusnya. Dalam contoh kasus prediksi saham ini, ada 3 model yang disusun secara linear yang terdiri dari poin-poin dibawah ini:

1. **LSTM** : nilai unit dari LSTM diisikan 100 berarti keluaran dari model ini berupa keluaran dengan dimensi 100. Stateful juga bernilai *true* yang berarti kondisi terakhir untuk setiap sampel dari indeks i dalam sebuah batch akan digunakan sebagai kondisi awal dari sampel indeks di batch berikutnya.

2. **Dropout** : Tujuan dari Dropout adalah untuk mencegah model dari kondisi overfitting. (Srivastava, N dkk. 2014) Biasanya Dropout diletakkan setelah fungsi aktivasi untuk semua fungsi aktivasi selain /"relu/". Arti dari nilai Dropout dimasukkan angka 0.02 yang berarti 2% dari unit tersembunyi dalam jaringan saraf akan dimatikan sementara setiap fase training.

3. **Dense** : Secara matematis, lapisan Dense digunakan untuk mengubah dimensi dari vektor dan mengaplikasikan perhitungan vektor seperti rotasi, skala. translasi, dan transformasi. Dalam kasus ini Dense diisi dengan angka 1 yang berarti keluaran dari lapisan merupakan vektor dengan dimensi 1.

Source code tentang model LSTM bisa dilihat di bawah ini

```
#Build the model

lstm_model = Sequential()
lstm_model.add(LSTM(100, batch_input_shape=(BATCH_SIZE, 3, x_t.shape[2]), dropout=0.0, recurrent_dropout=0.0, stateful= True,
                    kernel_initializer='random_uniform'))
lstm_model.add(Dropout(0.02))
#lstm_model.add(Dense(20, activation='relu'))
lstm_model.add(Dense(1, activation='linear'))
optimizer = optimizers.RMSprop(lr=0.00008)
lstm_model.compile(loss='mean_squared_error', optimizer=optimizer)

history = lstm_model.fit(x_t, y_t, epochs=40, verbose=2, batch_size=BATCH_SIZE,
                    shuffle=False, validation_data=(trim_dataset(x_val, BATCH_SIZE),
                    trim_dataset(y_val, BATCH_SIZE)))
```

FIGURE 3.4: model LSTM

3.5 Hasil dan Analisis

Dua dataset dari pasar saham digunakan untuk menginvestigasi bahwa kerangka prediksi menunjukkan performa yang baik. Berikut adalah hasil dari pemelajaran yang dilakukan oleh model LSTM

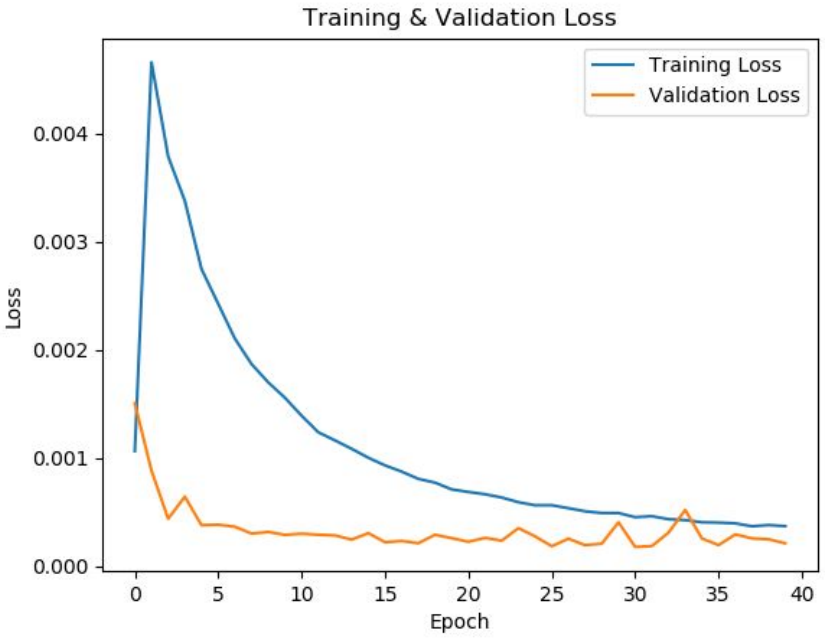

FIGURE 3.5: fungsi loss untuk data JKSE

Bisa dilihat bahwa grafik loss (MSE) untuk training dan validation loss akan menuju pada titik minimum seiring bertambahnya iterasi atau *epoch*. Hasil dari grafik training dan validation menunjukkan bahwa model pembelajaran tidak menunjukkan overfitting ataupun underfitting hingga mencapai epoch ke 40. Disini juga dapat dilihat untuk mengetahui performa model tidak menggunakan fungsi akurasi karena permasalahan kasus ini adalah regresi linear, fungsi akurasi digunakan untuk permasalahan klasifikasi.

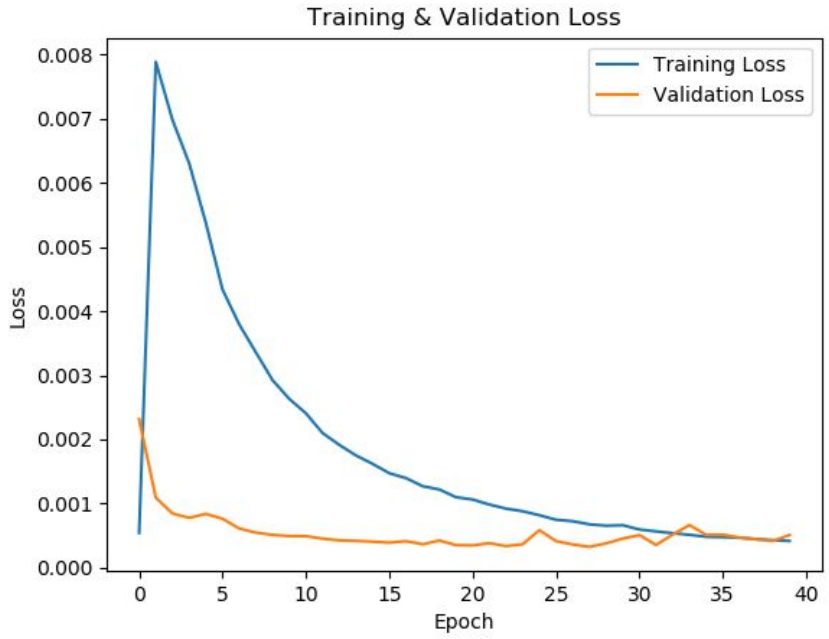

FIGURE 3.6: fungsi loss untuk data BBCA

Berikut adalah hasil dari data saham menggunakan data dari indeks saham harga gabungan

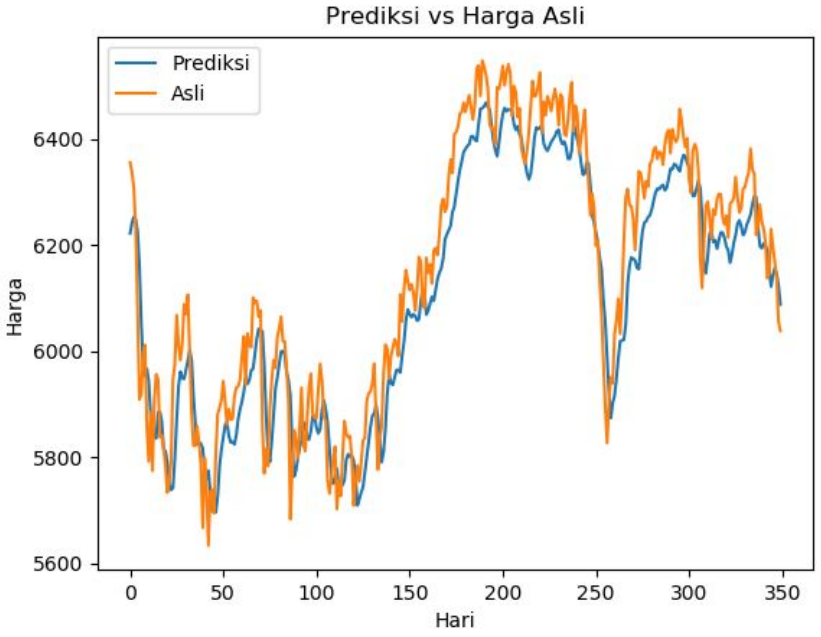

FIGURE 3.7: Indeks saham dari data JKSG

Meskipun tidak ada kriteria apakah model yang digunakan baik atau tidak, namun masih ada beberapa hal yang dapat dilakukan untuk meningkatkan akurasi atau mengurangi loss, berikut adalah beberapa opsi yang dapat dilakukan:

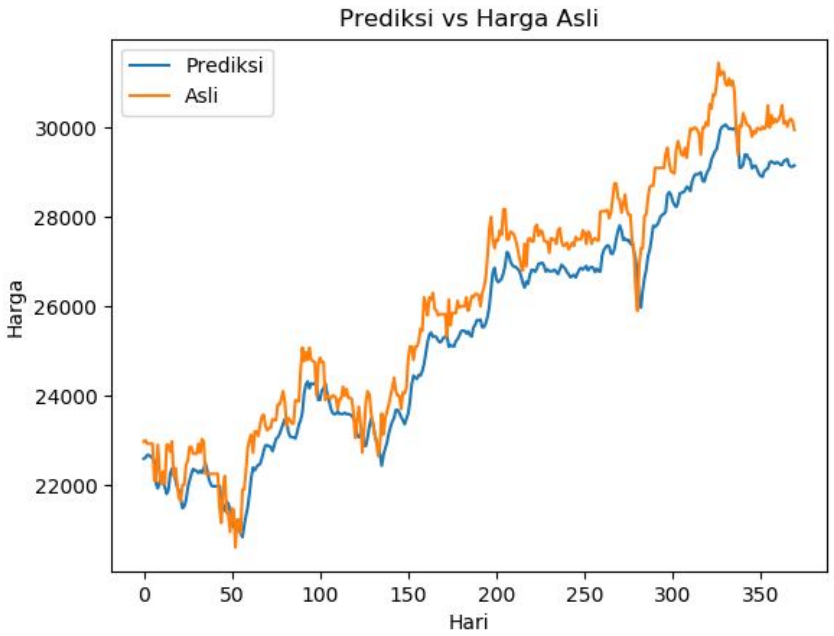

FIGURE 3.8: Indeks saham dari data JKSG

1. **Mengatur parameter secara manual** : ini merupakan cara yang cukup sulit untuk menemukan konfigurasi yang tepat karena dilakukan secara manual, biasanya cara ini digunakan jika sudah memiliki cukup pengalaman dalam melakukan analisis.

2. **Optimisasi Bayesian** : Metode ini merupakan metode matematis. Optimisasi bayesian menggunakan proses gaussian untuk menebak atau memodelkan fungsi objektif. Pertama sejumlah nilai acak ditentukan dalam rentang parameter tertentu. Kemudian proses Gaussian digunakan untuk menebak fungsi objektifnya, Proses ini diulang selama beberapa kali dalam limit tertentu.

3. **Implementasi Grid** : Ini merupakan salah satu cara yang bagus yang dapat memberikan beberapa parameter yang ingin di optimasi, kemudian gunakan model tersebut untuk melatih data dan temukan fungsi loss untuk setiap kombinasi, cara ini dapat digunakan dengan menggunakan *library* scikit-learn.

Chapter 4

Kesimpulan

Dalam beberapa tahun terakhir, metode-metode berbasis *deep learning* telah banyak digunakan untuk menyelesaikan masalah-masalah yang berkaitan dengan finansial. Ini disebabkan oleh beberapa keuntungan dari teknik-teknik *machine learning* dimana metode tersebut dapat menyelesaikan beberapa masalah kompleks yang berkaitan dengan data. Sebagaimana jaringan-jaringan neural network merupakan pendekatan dengan data, performa model akan Semakin baik seiring dengan bertambahnya data. Jika data semakin banyak, jaringan neural akan menaikkan akurasi.

Dalam membuat model deep learning dengan menggunakan LSTM ini terdiri dari beberapa langkah, yaitu pertama akuisisi data dari internet, dataset yang digunakan berasal dari *yahoo finance*, langkah kedua yaitu pembagian data training, data tes, dan data validasi dengan rasio tertentu. Kemudian langkah selanjutnya adalah normalisasi data menjadi rentang antara 0 dan 1, langkah ini dilakukan untuk mengubah data dengan skala yang standar. Langkah selanjutnya yaitu mengubah data waktu serial menjadi data supervised. Kemudian model LSTM didesain sedemikian rupa sehingga didapatkan model dengan performa yang baik. Tidak ada nilai yang pasti untuk menentukan bahwa model tersebut baik atau tidak, namun beberapa parameter dapat digunakan sebagai acuan, dalam penelitian ini parameter tersebut yang digunakan adalah parameter fungsi loss.

Beberapa hal dapat dilakukan untuk meningkatkan performa dari model jika ingin melanjutkan penelitian, yaitu mengatur parameter secara manual, optimisasi bayesian, dan implementasi grid. Metode ini dapat dilakuka untuk menemukan parameter yang lebih baik daripada parameter yang ada di modul ini.

Appendix A

Source Code

```
##########################################################################
# Title          : Prediksi Indeks Harga Saham dengan menggunakan Recurrent
Neural Network
# Author         : Achmad M. Gani
# Aims           : 1. Menerapkan metode neural network berbasis time series dengan program Python
#                  2. Meramal harga indeks saham di masa depan
# Input          : Matriks dengan dimensi m x n dan merupakan fungsi time series
# Output         : Grafik dan variabel dengan tipe data list berisi nilai prediksi terhadap waktu
# Outline Code   : 1. Header
#                  2. Definisi Fungsi
#                  3. Algorithma
#                  4. Prediksi / Forecasting
##########################################################################
#-------------------HEADER-----------------
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from keras import optimizers
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error

#------------------ PREDEFINE THE FUNCTIONS ---------------

def build_timeseries(mat, y_col_index):
    TIME_STEPS = 3
    dim_0 = mat.shape[0] - TIME_STEPS
    dim_1 = mat.shape[1]
    x = np.zeros((dim_0, TIME_STEPS, dim_1))
    y = np.zeros((dim_0,))
```

```
    for i in range(dim_0):
        x[i] = mat[i:TIME_STEPS+i]
        y[i] = mat[TIME_STEPS+i, y_col_index]
    print("length of time-series", x.shape, y.shape)
    return x, y

def trim_dataset(mat, batch_size):
    no_of_rows = mat.shape[0]% batch_size
    if(no_of_rows > 0) :
        return mat[:-no_of_rows]
    else:
        return mat
#---------------------------------------- ALGORITHMS ------------------------
#load the data and prepare the constants
BATCH_SIZE = 10
PATH = '../Dataset/JKSEMax.csv'
stock_time_series = pd.read_csv(PATH)

#------ Preprocessing -----
# drop null values in "Close"
stock_time_series = stock_time_series.dropna(subset=['Close'])

train_cols = ["Open", "High", "Low", "Close", "Volume"]
df_train, df_test = train_test_split(stock_time_series, train_size=0.8, test_size=0.2, shuffle=F
print("Train and Test size", len(df_train), len(df_test))

#normalization
x = df_train.loc[:, train_cols].values
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x)
x_test = scaler.transform(df_test.loc[:, train_cols])

#turn into supervised learning
x_t, y_t = build_timeseries(x_train, 3)
x_t, y_t = trim_dataset(x_t, BATCH_SIZE), trim_dataset(y_t, BATCH_SIZE)
x_temp, y_temp = build_timeseries(x_test, 3)
x_val, x_test_t = np.split(trim_dataset(x_temp, BATCH_SIZE), 2)
y_val, y_test_t = np.split(trim_dataset(y_temp, BATCH_SIZE), 2)

#Build the model

lstm_model = Sequential()
lstm_model.add(LSTM(100, batch_input_shape=(BATCH_SIZE, 3, x_t.shape[2]), dropout=0.0, recurrent
                    kernel_initializer='random_uniform'))
lstm_model.add(Dropout(0.02))
#lstm_model.add(Dense(20, activation='relu'))
lstm_model.add(Dense(1, activation='linear'))
optimizer = optimizers.RMSprop(lr=0.00008)
lstm_model.compile(loss='mean_squared_error', optimizer=optimizer)

history = lstm_model.fit(x_t, y_t, epochs=40, verbose=2, batch_size=BATCH_SIZE,
                    shuffle=False, validation_data=(trim_dataset(x_val, BATCH_SIZE),
                    trim_dataset(y_val, BATCH_SIZE)))

# predict the test data
```

```
y_pred = lstm_model.predict(trim_dataset(x_test_t, BATCH_SIZE), batch_size=BATCH_SIZE)
y_pred = y_pred.flatten()
y_test_t = trim_dataset(y_test_t, BATCH_SIZE)
error = mean_squared_error(y_test_t, y_pred)
print("Error is", error, y_pred.shape, y_test_t.shape)
print(y_pred[0:15])
print(y_test_t[0:15])

# convert the predicted value to range of real data
y_pred_org = (y_pred * scaler.data_range_[3]) + scaler.data_min_[3]
# min_max_scaler.inverse_transform(y_pred)
y_test_t_org = (y_test_t * scaler.data_range_[3]) + scaler.data_min_[3]
# min_max_scaler.inverse_transform(y_test_t)
print(y_pred_org[0:15])
print(y_test_t_org[0:15])

# Visualize the prediction
plt.figure()
plt.plot(y_pred_org)
plt.plot(y_test_t_org)
plt.title('Prediksi vs Harga Asli')
plt.ylabel('Harga')
plt.xlabel('Hari')
plt.legend(['Prediksi', 'Asli'], loc='upper left')
plt.show()

#Visualize the val_loss and loss
"""
plt.figure()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Training & Validation Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Training Loss', 'Validation Loss'], loc='upper right')
plt.show()
"""
#forecasting attempt
```

Bibliography

[1] Yujin Baek and Ha Young Kim. Modaugnet: A new forecasting framework for stock market index value with an overfitting prevention lstm module and a prediction lstm module. *Expert Systems with Applications*, 113:457 – 480, 2018. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2018.07.019. URL http://www.sciencedirect.com/science/article/pii/S0957417418304342.