BAB 5
Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan eksperimen, analisis data, dan regresi yang telah dilakukan, maka kesimpulan yang dapat diambil dari penelitian ini adalah:

1. Telah berhasil mempersiapkan data indeks saham IHSG dan 5 indeks sektoral yakni agrikultur, finansial, infrastruktur, pertambangan, dan properti dengan menggunakan metode penambangan data temporal. Data tersebut disiapkan dengan melakukan normalisasi. Pengolahan data menggunakan RNN-LSTM dengan *Hidden Layer* yang dimulai dari 2 hingga 10 dan di optimasi oleh ADAM *optimizer*.
2. Telah berhasil menemukan model yang tepat dalam melakukan prediksi pada data yang diolah menggunakan metode penambangan data temporal. Prediksi regresi dilakukan dengan menggunakan algoritma *Long Short Term Memory* dengan *Hidden Layer* 6 *dan ADAM Optimizer*. Pada *Optimizer* sendiri, *jenis Optimizer* yang digunakan berjumlah 6 yaitu ADAM, ADAMAX, ADAGRAD, NADAM, ADADELTA, dan SGD. Hasil galat RMSE terendah didapatkan dengan menggunakan Optimizer ADAM. Dari hasil uji coba dan analisa masing-masing indeks saham maka dapat disimpulkan bahwa peramalan saham menggunakan RNN-LSTM ini memiliki persamaan dimana jumlah *Hidden Layer* pada *Neural Network* yang menghasilkan galat terendah adalah 6. Namun hasil uji akan memiliki galat yang berbeda – beda. Dimana untuk IHSG memiliki RMSE *Training* sebesar 29.4404 dengan MAE sebesar 23,2666. Untuk indeks saham agrikultur memiliki RMSE *Training* sebesar 30,1267 dengan MAE sebesar 17,3455. Untuk indeks saham finansial memiliki RMSE *Training* sebesar 4,1733 dengan MAE sebesar 3,7316. Untuk indeks infrastruktur memiliki RMSE *Training* sebesar 9.3361 dengan MAE sebesar 6,3399. Untuk indeks saham

pertambangan memiliki RMSE *Training* sebesar 40,8074 dengan MAE sebesar 22,6627. Untuk indeks saham properti memiliki RMSE *Training* sebesar 2,2995 dengan MAE sebesar 2,0085.

5.2 Saran

Pada penelitian selanjutnya, peneliti dapat mencoba menggunakan metode yang lain dalam memproses data temporal. Disarankan juga melakukan hybrid pada algoritma LSTM dengan algoritma lain seperti *rough-set, genetic programming,* dll, untuk menghasilkan model prediksi yang lebih baik. Selain itu, dapat juga mencoba untuk sumber data temporal yang lainnya seperti indeks perusahaan yang lebih spesifik seperti indeks saham BCA, Ciputra, dll.

Pelajari permasalahan yang ingin diselesaikan dan pahami dasar teori dengan baik terlebih dahulu. Hindari penggunaan resampling karena hal tersebut memanipulasi data sebenarnya. Kemudian ketika membentuk model regresi perhatikan juga aspek lain yang mempengaruhi hasil galat.

DAFTAR PUSTAKA

Andriani, R. S., 2017. *Bisnis Indonesia.* [Online]
Available at: http://www.market.bisnis.com

Anon., 2018. *Investing.com: Quote, Saham & Berita Keuangan.* [Online]
Available at: id.investing.com

Antunes, C. M. & Oliveir, A. L., 2001. Temporal Data Mining: an overview. *KDD workshop on temporal data mining,* pp. (Vol. 1, pp. 1-13).

Ardelia, I. & Dewi, F. R., 2016. Analisis Kinerja Portofolio Optimal Saham Sektor Pertambangan dan Saham Sektor Perdagangan. *Jurnal Manajemen dan Organisasi IPB Vol VII, No 3.*

Baohua, W., Heijao, H. & Xiaolong, W., 2012. A Novel Text Mining Approach to Financial Time Series Forecasting. *Neurocomputing,* pp. 83, 136-145.

Bengio, Y., Simard, P. & Frasconi, P., 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks,* pp. 157 - 166.

Cheng, C.-H., Chen, T.-L. & Wei, L.-Y., 2010. A Hybrid Model Based on Rough Sets Theory and Genetic Algorithms for Stock Price Forecasting. *Information Sciences,* pp. 1610-1629.

Chen, K., Zhou, Y. & Dai, F., 2015. *LSTM-based method for stock returns prediction : A case study of China stock market.* Santa Clara, IEEE, pp. 2823-2824.

Dubois, P. F., 2007. Guest Editor's Introduction: Python--Batteries Included.. *Computing in Science & Engineering,* pp. 9(3), 7-9.

Duchi, J., Hazan, E. & Singer, Y., 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research (JMLR),* pp. 2121-2159.

Edwards, R. D., Magee, J. & Bassetti, W., 2007. *Technical Analysis of Stock Trends.* s.l.:CRC press..

Ekonomi, B. R., 2009. *Krisis Ekonomi Global dan Dampaknya terhadap Perekonomian Indonesia,* Jakarta: Direktorat Riset Ekonomi dan Kebijakan Moneter..

Friedman, J., Hastie, T. & Tibshirani, R., 2001. *The Elements of Statistical Learning.* New York: s.l.: Springer.

García, S., Luengo, J. & Herrera, F., 2014. *Data Preprocessing in Data Mining.* Switzerland: s.l.:Springer.

Glienmourinsie, D., 2015. *Sindonews.* [Online]
Available at: www.Sindonews.com

Han, J., Pei, J. & Kamber, M., 2011. *Data Mining: Concepts and Techniques.* s.l.:Elsevier.

Hochreiter, S. & Schmidhuber, J., 1997. Long Short-Term Memory. *Journal Neural Computation,* pp. 1735-1780 .

Hsu, C.-W., Chang, C.-C. & Lin, C.-J., 2016. A Practical Guide to Support Vector Classification.

Keskar, N. S. & Socher, R., 2017. Improving Generalization Performance by Switching from Adam to SGD. *arXiv:1712.07628.*

Kingma, D. P. & Ba, J. L., 2014. Adam: A method for stochastic optimization.. *arXiv:1412.6980.*

Kuhn, M. & Johnson, K., 2013. *Applied Predictive Modeling.* New York: Springer.

Manurung, B., 2015. *PENGARUH PERISTIWA PEMILU INDONESIA 2014 TERHADAP INDEKS HARGA SAHAM GABUNGAN (IHSG),* Yogyakarta: Universitas Gadjah Mada.

May, E., 2017. *Detikcom.* [Online]
Available at: https://www.finance.detik.com/

Milmann, K. & Avaizis, M., 2011. Python for Scientists and Engineers. *Scientific Python. Vol. 11 of Computing in Science & Engineering.*

Mitsa, T., 2010. *Temporal data mining.* s.l.:CRC Press.

Nahal, S., Wilson, J., Renteria, A. & Moath, N., 2017. *The Relationship Between Deep Learning and Brain Function.* New York, Semantic Scholar.

Ozel, T. & Karpat, Y., 2005. Predictive Modeling of Surface Roughness and Tool Wear in Hard. *International Journal of Machine Tools & Manufacture* , p. 467–479.

Pappa, G. L. & Freitas, A. A., 2006. *Automatically Evolving Rule Induction Algorithms.* Berlin, Heidelberg, Springer, pp. 341-352.

Pascanu, R., Mikolov, T. & Bengio, Y., 2013. *On the Difficulty of Training Recurrent Neural Networks.* Atlanta, Journal of Machine Learning Research (JMLR), pp. 1310-1318.

Patro, S. G. K. & Sahu, K. K., 2015. Normalization: A Preprocessing Stage. *arXiv.*

Permana, C. D. & Asmara, A., 2010. ANALISIS PERANAN DAN DAMPAK INVESTASI INFRASTRUKTUR TERHADAP PEREKONOMIAN INDONESIA: ANALISIS INPUT-OUTPUT. *Journal IPB,* pp. Vol 7, No 1.

Pertanian, K. P. S. E. d. K., 2013. *KRISIS EKONOMI GLOBAL DAN DAMPAKNYA TERHADAP SEKTOR AGROINDUSTRI INDONESIA,* s.l.: Badan LITBANG Pertanian Kementerian Pertanian.

Roy, K., Das, N. R., Ambure, P. & Aher, R. B., 2016. Be aware of error measures. Further studies on validation of predictive QSAR Models. *Chemometrics and Intelligent Laboratory Systems,* Volume 152, pp. 18-33.

Schmidhuber, J., 2015. Deep learning in neural networks: An overview. *Neural Networks,* pp. 61: 85-117.

Schneider, G. & Wrede, P., 1998. Artificial Neural Networks for Computer-Based Molecular Design. *Progress in Biophysics and Molecular Biology,* 70(3), pp. 175-222.

Shafi, I., Ahmad, J., Shah, S. I. & Kashif, F. M., 2006. *Impact of Varying Neurons and Hidden Layers in Neural Network Architecture for a Time Frequency Application.* Islamabad, IEEE, pp. 188-193.

Shumway, R. H. & Stoffer, D. S., 2017. *Time Series Analysis and Its Applications.* s.l.:Springer.

Susanti, L. A. D., Fariza, A. & Setiawardhana, 2011. Peramalan Harga Saham Menggunakan Recurrent Neural Network dengan Algoritma Backpropagation Through Time (BPTT). *EEPIS Final Project.*

Sutskever, I., Vinyals, O. & Le, Q. V., 2014. *Sequence to Sequence Learning.* Montreal, Neural Information Processing Systems Foundation, Inc..

Tieleman, T. & Hinton, G., 2012. Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude. *COURSERA: Neural networks for machine learning,* pp. 26-31.

Tyas, A. A. W. P., 2012. Dampak Krisis Keuangan Global Terhadap Industri Properti di Indonesia. *Jurnal Universitas Esa Unggul.*

Wang, J.-J., Wang, J.-Z., Zhang, Z.-G. & Guo, S.-P., 2012. Stock Index Forecasting Based on a Hybrid Model. *Omega,* p. 758–766.

Wang, J., Yu, L.-C., Lai, K. R. & Zhang, X., 2016. Dimensional Sentiment Analysis Using a Regional CNN-LSTM Model. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* pp. Vol. 2, pp. 225-230.

Xiong, T., Bao, Y. & Hu, Z., 2014. Multiple-output support vector regression with a firefly algorithm for interval-valued stock price index forecasting. *Knowledge-Based Systems,* pp. 55, 87-100.

LAMPIRAN

Lampiran 1 – Pseudocode

```
data = pd.read_csv('IHSG.csv')

stringTanggal = data.loc[:,'Tanggal']

dateparse = lambda dates: pd.datetime.strptime(dates,'%d-%m-
%Y')

data = pd.read_csv('IHSG.csv', parse_dates=['Tanggal'],
index_col='Tanggal',date_parser=dateparse)

# Konversi data array kebentuk matriks

def create_dataset(dataset, look_back=1):
                    dataX, dataY = [], []
          for i in range(len(dataset)-look_back-1):
            a = dataset[i:(i+look_back), 0]
           dataX.append(a)
           dataY.append(dataset[i + look_back, 0])
        return numpy.array(dataX), numpy.array(dataY)

numpy.random.seed(12)

# Memanggil dataset

dataframe = read_csv('IHSG.csv', usecols=[1],
engine='python', skipfooter=3)

dataset = dataframe.values

dataset = dataset.astype('float32')

# Normalisasi

scaler = MinMaxScaler(feature_range=(0, 1))

dataset = scaler.fit_transform(dataset)

# Membagi data training dan test

train_size = int(len(dataset) * 0.70)

test_size = len(dataset) - train_size

train, test = dataset[0:train_size,:],
dataset[train_size:len(dataset),:]

# Membentuk time series X=t and Y=t+1

trainX, trainY = create_dataset(train, look_back)

testX, testY = create_dataset(test, look_back)
```

```
# Membentuk input menjadi [samples, time steps, features]
trainX = numpy.reshape(trainX, (trainX.shape[0], 1,
trainX.shape[1]))
testX = numpy.reshape(testX, (testX.shape[0], 1,
testX.shape[1]))
# Membuat Jaringan LSTM dengan 6 Hidden Layer
model = Sequential()
model.add(LSTM(6, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.compile(loss='mae', optimizer='adam')
model.fit(trainX, trainY, epochs=100, batch_size=1,
verbose=2)
# Membuat prediksi berdasarkan model LSTM
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)
# Melakukan invert
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])
# Mneghitung mean absolute error
trainScore = mean_absolute_error(trainY[0],
trainPredict[:,0])
print('Train Score: %.4f RMSE' % (trainScore))
from scipy.stats import linregress
testScore = mean_absolute_error(testY[0], testPredict[:,0])
print('Test Score: %.4f RMSE' % (testScore))
# Menghitung root mean squared error
trainScore = math.sqrt(mean_squared_error(trainY[0],
trainPredict[:,0]))
print('Train Score: %.4f RMSE' % (trainScore))
testScore = mean_absolute_error(testY[0], testPredict[:,0])
```

```
testScore = math.sqrt(mean_squared_error(testY[0],
testPredict[:,0]))

print('Test Score: %.4f RMSE' % (testScore))

# Menghitung Pearson Coeffisient Correlation

linregress(testY[0], testPredict[:,0])

# Menyiapkan data training untuk plot

trainPredictPlot = numpy.empty_like(dataset)

trainPredictPlot[:, :] = numpy.nan

trainPredictPlot[look_back:len(trainPredict)+look_back, :] =
trainPredict

# Menyiapkan data test untuk plot

testPredictPlot = numpy.empty_like(dataset)

testPredictPlot[:, :] = numpy.nan

testPredictPlot[len(trainPredict)+(look_back*2)+1:len(datase
t)-1, :] = testPredict

testPredict

# Membentuk plot dari data asli dan hasil prediksi

plt.plot(scaler.inverse_transform(dataset))

plt.plot(trainPredictPlot)

plt.plot(testPredictPlot)

plt.grid(True)

plt.show()

print(model.summary())

# Parsing tanggal

def parser(x):

        return datetime.strptime(x,'%d-%m-%Y')

# Konversi dari time series ke supervised learning

def series_to_supervised(data, n_in=1, n_out=1,
dropnan=True):

    n_vars = 1 if type(data) is list else
data.shape[1]

    df = DataFrame(data)

    cols, names = list(), list()
```

```
	# input sequence (t-n, ... t-1)
	for i in range(n_in, 0, -1):
		cols.append(df.shift(i))
		names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
	# Meramalkan sekuens kedepan (t, t+1, ... t+n)
	for i in range(0, n_out):
		cols.append(df.shift(-i))
		if i == 0:
			names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
		else:
			names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
	# put it all together
	agg = concat(cols, axis=1)
	agg.columns = names
	# drop rows with NaN values
	if dropnan:
		agg.dropna(inplace=True)
	return agg
# Membuat interval pada seri data
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)
# Transformasi menjadi train and test untuk 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)
    # Normalisasi untuk supervised learning -1, 1
    scaler = MinMaxScaler(feature_range=(-1, 1))
    scaled_values = scaler.fit_transform(diff_values)
    scaled_values = scaled_values.reshape(len(scaled_values), 1)
    # Transformasi ke 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
#Membuat jaringan LSTM untuk mengolah data training
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.shape[1], X.shape[2]), stateful=True))
    model.add(Dense(y.shape[1]))
    model.compile(loss='mean_squared_error', optimizer='adam')
    # fit network
    for i in range(nb_epoch):
        model.fit(X, y, epochs=1, batch_size=n_batch, verbose=0, shuffle=False)
```

```
            model.reset_states()
        return model
# Peramalan dengan LSTM,
def forecast_lstm(model, X, n_batch):
        # reshape input pattern to [samples, timesteps,
features]
        X = X.reshape(1, 1, len(X))
        # make forecast
        forecast = model.predict(X, batch_size=n_batch)
        # convert to array
        return [x for x in forecast[0, :]]
# Evaluasi model model
def make_forecasts(model, n_batch, train, test, n_lag,
n_seq):
        forecasts = list()
        for i in range(len(test)):
            X, y = test[i, 0:n_lag], test[i, n_lag:]
            # make forecast
            forecast = forecast_lstm(model, X, n_batch)
            # store the forecast
            forecasts.append(forecast)
        return forecasts
# Inversi
def inverse_difference(last_ob, forecast):
        # invert first forecast
        inverted = list()
        inverted.append(forecast[0] + last_ob)
        # propagate difference forecast using inverted
first value
        for i in range(1, len(forecast)):
            inverted.append(forecast[i] + inverted[i-1])
        return inverted
def inverse_transform(series, forecasts, scaler,
n_test):
```

```
	inverted = list()
	for i in range(len(forecasts)):
		# Membuat array dari hasil peramalan
		forecast = array(forecasts[i])
		forecast = forecast.reshape(1, len(forecast))
		# inversi dari proses scaling
		inv_scale = scaler.inverse_transform(forecast)
		inv_scale = inv_scale[0, :]
		# inversi dari selisih data
		index = len(series) - n_test + i - 1
		last_ob = series.values[index]
		inv_diff = inverse_difference(last_ob, inv_scale)
		# Penyimpanan hasil inversi
		inverted.append(inv_diff)
	return inverted
# Evaluasi dengan RMSE untuk tiap time-step
def evaluate_forecasts(test, forecasts, n_lag, n_seq):
	for i in range(n_seq):
		actual = [row[i] for row in test]
		predicted = [forecast[i] for forecast in forecasts]
		rmse = sqrt(mean_squared_error(actual, predicted))
		print('t+%d RMSE: %f' % ((i+1), rmse))
# plot peramalan dalam konteks data sebenarnya
def plot_forecasts(series, forecasts, n_test):
	# plot the entire dataset in blue
	pyplot.plot(series.values)
	# plot the forecasts in red
	for i in range(len(forecasts)):
		off_s = len(series) - n_test + i - 1
		off_e = off_s + len(forecasts[i]) + 1
```

```
            xaxis = [x for x in range(off_s, off_e)]
            yaxis = [series.values[off_s]] + forecasts[i]
            pyplot.plot(xaxis, yaxis, color='red')
    # show the plot
    pyplot.show()
# Memuat dataset
series = read_csv('reversed agri 2000.csv', header=0, parse_dates=[0], index_col=0, squeeze=True, date_parser=parser)
# konfigurasi proses LSTM
n_lag = 6
n_seq = 21
n_test = 1
n_epochs = 1000
n_batch = 1
n_neurons = 6
# Persiapan data
scaler, train, test = prepare_data(series, n_test, n_lag, n_seq)
# Model
model = fit_lstm(train, n_lag, n_seq, n_batch, n_epochs, n_neurons)
# Peramalan
forecasts = make_forecasts(model, n_batch, train, test, n_lag, n_seq)
# inverse transform dari peramalan dan test
forecasts = inverse_transform(series, forecasts, scaler, n_test+2)
actual = [row[n_lag:] for row in test]
actual = inverse_transform(series, actual, scaler, n_test+2)
# evaluasi peramalan
evaluate_forecasts(actual, forecasts, n_lag, n_seq)
# plot peramalan
plot_forecasts(series, forecasts, n_test+1)
```