Daftar Pustaka

Adisaputro, A. (2000). *Anggaran Perusahaan* (3 ed.). Yogyakarta: BPFE Yogyakarta.

Bagas, dkk. (2019). Perbandingan Prediksi Harga Saham Dengan Model ARIMA dan Artificial Neural Network. *Ind. Journal on Computing, 4*(2). doi:10.21108/indojc.2019.4.2.344

Bottieau, dkk. (2018, Juni). Leveraging provision of frequency regulation services from wind generation by improving day-ahead predictions using LSTM neural networks. *2018 IEEE International Energy Conference (ENERGYCON)*. doi: 10.1109/ENERGYCON.2018.8398741

Brockwell, R. (2002). Introduction to time series and forecasting. Springer. doi:https://doi.org/10.1007/b97391

Brownlee. (2019, Jan 30). *How to Choose Loss Functions When Training Deep Learning Neural Networks*. Diambil kembali dari machine Learning Mastery: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

David, dkk. (2017). Stock Market's Price Movement Prediction WithLSTM Neural Networks. *2017 International Joint Conference on Neural Networks (IJCNN)*. doi:10.1109/IJCNN.2017.7966019

Entrepreneur. *Mengenal Forecasting, Manfaat, Fungsi, dan Jenisnya untuk Kesuksesan Bisnis Anda*. (Jurnal by Mekari) Dipetik Juni 1, 2021, dari Jurnal Entrepreneur: https://www.jurnal.id/id/blog/2018-forecasting-pengertian-manfaat-fungsi-dan-jenisnya-bagi-kesuksesan-bisnis/

Felix, dkk. (2002). Learning Precise Timing with LSTM Recurrent Networks. *Journal of Machine Learning Research 3 (2002)*, 115-143. doi:10.1162/153244303768966139

Lei (2020). Time Series-oriented Load Prediction Using Deep Peephole LSTM. *12th International Conference on Advanced Computational Intelligence (ICACI)*. doi:10.1109/ICACI49185.2020.9177688

Gardner. (1985). Exponential smoothing: the state of the art. *Journal of Forecasting*, (hal. 1-28).

Gemilang, F. (2017). Prediksi Harga Penutupan Saham Menggunakan Fuzzy Timeseries. Dipetik Mei 24, 2021, dari https://repository.usd.ac.id/11870/2/135314089_full.pdf

Guangyu, L. (2019). Study on the prediction of stock price based on the associated network model of LSTM. *International Journal of Machine Learning and Cybernetics (2020),* 11:1307-1317. doi:https://doi.org/10.1007/s13042-019-01041-1

Harvey. (1993). Structural time series models. *11*, 261-302. Dipetik 2021

Harvey, P. (1990). Estimation procedures for structural time series models. *Journal of Forecasting*, *9*, hal. 89-108.

Hastie, T. (1987). Generalized additive models: some applications. *Jour-nal of the American Statistical Association*, (hal. 371–386).

Jianjing, dkk. (2018). Long short-term memory for machine remaining life prediction. Dalam L. Wang (Penyunt.), *Journal of Manufacturing Systems.* ScienceDirect. doi:10.1016/j.jmsy.2018.05.011

Ludwig, S. (2019). Comparison of Time Series Approaches applied toGreenhouse Gas Analysis:ANFIS, RNN, and LSTM. *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. doi:10.1109/FUZZ-IEEE.2019.8859013

Luthfianto, dkk. (2011). eramalan Jumlah Penumpang Kereta Api dengan Jaringan Syaraf Tiruan Metode Perambatan Balik (Backpropogation). J. Universitas Diponegoro. Dipetik Juni 1, 2021

Madhuri Raga, dkk. (2020). Stock Market Prediction for Time-series Forecasting using Prophet upon ARIMA. *IEEE 7th International Conference on Smart Structures and Systems ICSSS 2020*. Dipetik Juni 6, 2021

Moghar, M. (2020). Stock Market Prediction Using LSTM Recurrent Neural Network. *International Workshop on Statistical Methods and Artificial Intelligence (IWSMAI 2020). 170*, hal. 1168-1173. Warsaw: Elsevier B.V. doi:https://doi.org/10.1016/j.procs.2020.03.049

Olah. (2015, Agustus 27). *Understanding LSTM Networks*. Diambil kembali dari colah's blog: https://colah.github.io/posts/2015-08-Understanding-LSTMs

Panthula. (2018). *Stock Price Trends Prediction Using LSTM Neural Network.* Dipetik Feb https://github.com/ani1cr7/Stock-Price-Prediction-using-LSTM-Neural-Networks21, 2021

Pascanu, dkk. (2013). On the difficulty of training Recurrent Neural Networks. arXiv. doi: arXiv:1211.5063

Ravikumar, S. (2020). Prediction of Stock Prices using Machine Learning (Regression, Classification) Algorithms. *2020 International Conference for*

Emerging Technology (INCET). Belgaum: IEEE. doi:10.1109/INCET49848.2020.9154061

Rizal, dkk. (2020). Analisis Kinerja Adaptive Neuro-Fuzzy Inference System (ANFIS) Dengan Subtractive Clustering Pada Proses Klasifikasi. Dipetik Mei 8, 2021, dari http://repositori.usu.ac.id/handle/123456789/27531

Roondiwala, dkk. (2017). PredictingStockPricesUsingLSTM. International Journal of Science and Research (IJSR). doi:10.21275/ART20172755

Salehinejad, V. (2019). Ising-Dropout: A Regularization Method For Training And Compression of Deep Neural Networks. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* Brigthon: IEEE. doi:10.1109/ICASSP.2019.8682914

Schmidhuber, H. (1997). Long short term memory. *Neural Computation.* doi:http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Siyuan, dkk. (2018). Stock transaction prediction modeling and analysis based on LSTM. *Konferensi IEEE ke-13 2018 tentang Elektronika dan Aplikasi Industri (ICIEA)*. doi:10.1109/ICIEA.2018.8398183

Spiegel, dkk. (2007). *Statistik Schaum's Outlines* (3 ed.). Jakarta, Indonesia: Erlangga. Dipetik Feb 24, 2021

Tawum, dkk. (2021). Using LSTM and ARIMA to Simulate and Predict Limestone Price Variations. Dalam M. &. Society for Mining (Penyunt.). Wuhan: Springer. Dipetik Feb 24, 2021, dari https://doi.org/10.1007/s42461-020-00362-y

Taylor, L. (2017). Forecasting at Scale. *PeerJ Preprints.* California: Facebook. doi:https://doi.org/10.7287/peerj.preprints.3190v2

Taylor, L. (2017, Feb 23). *Prophet: forecasting at scale*. Dipetik Mei 25, 2021, dari Facebook Research: https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

Ting, dkk. (2018, November). A Novel Method of Wind Speed Prediction by Peephole LSTM. *2018 International Conference on Power System Technology (POWERCON)*. doi:10.1109/POWERCON.2018.8601550

Upadhayula, dkk. (2019). Study Area Recommendation via Network Log Analytics. *2019 7th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud).* Newark: IEEE. doi:10.1109/MobileCloud.2019.00014

Vateekul, K. (2016). A Study of Sentiment Analysis Using Deep Learning Techniques on Thai Twitter Data. *13th Int. Jt. Conf. Comput. Sci. Softw. Eng. JCSSE 2016.* Bangkok: IEEE. doi:10.1109/JCSSE.2016.7748849

Wen-Xiang, dkk. (2019). Combine Facebook Prophet and LSTM with BPNN Forecasting financial markets : the Morgan Taiwan Index. *2019 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS.* Taipei: IEEE. doi:10.1109/ISPACS48206.2019.8986377

Yahoo! (2021, Mei 24). *What is the adjusted close?* Diambil kembali dari Yahoo!: https://help.yahoo.com/kb/SLN28256.html#:~:text=Adjusted%20close%20is%20the%20closing,Security%20Prices%20(CRSP)%20standards.

Yanhui Chen, dkk. (2017). Forecasting Crude Oil Prices: a Deep Learning based Model. *Information Technology and Quantitative Management (ITQM 2017)*, 300 - 307. Dipetik Mei 28, 2021

Yuniar. (2016). Sistem Prediksi saham Menggunakan Adaptive Neuro Fuzzy Inference System (Studi Kasus Saham Mingguan PT Astra Agro Lestari, Tbk). *02*. Surabaya: SYSTEMIC : Information System and Informatics Journal. doi:https://doi.org/10.29080/systemic.v2i2.113

Zhang, dkk. (2016). Sentiment classification using Comprehensive Attention Recurrent models. *2016 International Joint Conference on Neural Networks (IJCNN).* Vancouver: IEEE. doi:10.1109/IJCNN.2016.7727384

LAMPIRAN

Lampiran 1. Perbandingan *training prediction* model terhadap nilai aktual

1. *Training Prediction* Model *Vanilla* LSTM

2. *Training Prediction* Model *Peephole Connection* LSTM

(lanjutan)

3. *Training Prediction* Model *Facebook's Prophet*

Lampiran 2. Sintaks model LSTM dan *Prophet*

Mengimpor *library* yang digunakan

```
#Import Modules
import math
import pandas_datareader as web
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as plticker
import tensorflow as tf
from tensorflow_addons.rnn import PeepholeLSTMCell
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, LSTM, RNN
from tensorflow.keras import Input
from matplotlib.dates import DateFormatter
from sklearn.metrics import mean_absolute_error, mean_squared_error
from prophet import Prophet
import time
```

Mengambil dan membaca data dari *website* Yahoo! Finance

```
df = web.DataReader('TLKM.jk',
                    data_source='yahoo',
                    start='2005-09-30',
                    end='2021-06-11')
```

Menampilkan 5 data terakhir

```
df.tail()
```

Menampilkan grafik harga penutupan (*close price*) saham TLKM

```
fig, ax = plt.subplots(figsize=(16,8))
ax.set_title('Historikal Harga Penutupan TLKM')
ax.plot(df['Close'])
one = ax.get_xticks()
two = np.array(one) + np.array([1,0,1,0,1,0,1,0,1,0]) + 365
year_ticks = np.array([one,two]).T.flatten().tolist()[:-1]
ax.set_xticks(year_ticks)
ax.xaxis.set_major_formatter(DateFormatter("%Y"))
ax.set_xlabel('Tahun', fontsize=18)
ax.set_ylabel('Harga penutup (IDR)', fontsize=18)
ax.grid()
```

Mengambil data harga penutupan (*close price*)

```
new_df = pd.DataFrame()
new_df['y'] = df['Close']
new_df['ds'] = df.index
new_df.reset_index(inplace=True,drop=True)
new_df
```

(lanjutan)

Data preparation

```
NUM_TEST_DAYS = 30
INPUT_LENGTH = 60
def preprocess(stock):
    stock = stock.reshape(-1,1)
    #Normalization
    scaler = MinMaxScaler()
    transformed = scaler.fit_transform(stock)

    X=[]
    Y=[]

    for i in range(stock.shape[0] - INPUT_LENGTH):
        X.append(transformed[i : i+INPUT_LENGTH])
        Y.append(transformed[i+INPUT_LENGTH])
    X = np.array(X)
    Y = np.array(Y)
    return X, Y, scaler
```

Menormalisasi data

```
X, y, scaler = preprocess(np.array(new_df['y']))
pd.DataFrame({'actual':new_df['y'], 'normalized':MinMaxScaler().fit_transform(np.a
rray(new_df['y']).reshape(-1,1)).flatten().tolist()}).to_excel('normalized.xlsx')

X_train = X[:-NUM_TEST_DAYS]
X_test = X[-NUM_TEST_DAYS:]
y_train = y[:-NUM_TEST_DAYS]
y_test = y[-NUM_TEST_DAYS:]

df_train = new_df.iloc[:-NUM_TEST_DAYS]
df_test = new_df.iloc[-NUM_TEST_DAYS:]
```

Menampilkan data yang telah dinormalisasi

```
y
```

Mendefinisikan fungsi *Facebook's Prophet*

```
prophet = Prophet(changepoint_prior_scale=0.5)
prophet.add_country_holidays(country_name='ID')
time1=time.time()
prophet.fit(df_train)
time2 = time.time()
print(time2-time1)
```

Melatih model peramalan *Prophet*

```
# Train predictions
forecast_train = prophet.predict(df_train.drop('y',axis=1))

y_true_train = df_train.y.values
y_pred_train_prophet = forecast_train['yhat'].values

forecast_train[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

(lanjutan)

Menampilkan *changepoint* tren *Prophet*

```
fig1 = prophet.plot(forecast)
```

Menampilkan komponen *trend*, *weekly seasonality*, *yearly seasonality*, dan *holiday* pada data *train* (*Prophet*)

```
fig_train_component = prophet.plot_components(forecast_train)
```

Menampilkan komponen *trend*, *weekly seasonality*, *yearly seasonality*, dan *holiday* pada data *test* (*Prophet*)

```
fig2 = prophet.plot_components(forecast)
```

Menampilkan contoh perhitungan komponen *Prophet* pada t = 9 Feb 2021

```
forecast_train[forecast_train['ds']=='2021-02-
09'][['trend','weekly','yearly', 'holidays', 'yhat']]
```

Membuat hasil peramalan *Prophet* menggunakan data *test*

```
# Test predictions
forecast = prophet.predict(df_test.drop('y',axis=1))

y_true = df_test.y.values
y_pred_prophet = forecast['yhat'].values

forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

Membuat model *Vanilla* LSTM

```
# Vanilla
model = Sequential()
model.add(LSTM(50, return_sequences=True,
               input_shape=(INPUT_LENGTH, 1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mean_squared_error')
```

(lanjutan)

Membuat model *Peephole Connection* LSTM

```
# Peephole
units = 50
output_size = 1
def build_model():
    lstm_layer = RNN(
        PeepholeLSTMCell(units), input_shape=(INPUT_LENGTH, 1), return_sequences=T
rue
    )
    lstm_layer2 = RNN(
        PeepholeLSTMCell(units), return_sequences=False
    )
    model = Sequential(
        [lstm_layer,
         lstm_layer2,
         Dense(25),
         Dense(output_size),
        ]
    )
    return model
model2 = build_model()
model2.compile(optimizer='adam', loss='mean_squared_error')
```

Melatih model *Vanilla* LSTM

```
history1 = model.fit(X_train, y_train, batch_size=1, epochs=100)
```

Melatih model *Peephole Connection* LSTM

```
history2 = model2.fit(X_train, y_train, batch_size=1, epochs=100)
```

Meramalkan data *train* menggunakan model *Vanilla* LSTM

```
# Train predictions
y_pred_train_lstm = model.predict(X_train)
y_pred_train_lstm_denorm = scaler.inverse_transform(y_pred_train_lstm)
y_train_denorm = scaler.inverse_transform(y_train)
```

Meramalkan data *test* menggunakan model *Vanilla* LSTM

```
# Test predictions
y_pred_lstm = []

last_60 = X_train[-1:].flatten().tolist()
for i in range(NUM_TEST_DAYS):
    y_pred_lstm += model.predict(np.array(last_60[-
INPUT_LENGTH+i+1:]+[X_test[i][INPUT_LENGTH-
1][0]]+y_pred_lstm).reshape(1,INPUT_LENGTH,1)).flatten().tolist()
y_test_denorm = scaler.inverse_transform(y_test)
y_pred_lstm_denorm = scaler.inverse_transform(np.array([y_pred_lstm])).reshape(NUM
_TEST_DAYS,-1)
```

(lanjutan)

Meramalkan data *train* menggunakan model *Peephole Connection* LSTM

```
# Train predictions
y_pred_train_peephole = model2.predict(X_train)
y_pred_train_peephole_denorm = scaler.inverse_transform(y_pred_train_peephole)
y_train_denorm = scaler.inverse_transform(y_train)
```

Meramalkan data *test* menggunakan model *Peephole Connection* LSTM

```
# Test predictions
y_pred_peephole = []

last_60 = X_train[-1:].flatten().tolist()
for i in range(NUM_TEST_DAYS):
    y_pred_peephole += model2.predict(np.array(last_60[-
INPUT_LENGTH+i+1:]+[X_test[i][INPUT_LENGTH-
1][0]]+y_pred_peephole).reshape(1,INPUT_LENGTH,1)).flatten().tolist()
y_test_denorm = scaler.inverse_transform(y_test)
y_pred_peephole_denorm = scaler.inverse_transform(np.array([y_pred_peephole])).res
hape(NUM_TEST_DAYS,-1)
```

Menampilkan grafik peramalan data *train* 3 model

```
# Training plot
fig, ax = plt.subplots(figsize=(12,6))
ax.plot(df_train['ds'][60:], y_train_denorm, label='Actual', color='#000')
ax.plot(df_train['ds'][60:], y_pred_train_lstm_denorm, label='Vanilla LSTM Predict
ion', color='#FF7274')
ax.plot(df_train['ds'][60:], y_pred_train_peephole_denorm, label='Peephole LSTM Pr
ediction', color='#20948B')
ax.plot(df_train['ds'][60:], y_pred_train_prophet[60:], label='Prophet Prediction'
, color='#DE7A22')
ax.set_title('Training')
ax.grid()
ax.legend()
```

Menampilkan grafik peramalan data *test* 3 model

```
# Testing plot
fig, ax = plt.subplots(figsize=(12,6))
ax.plot(df_test['ds'], y_test_denorm, label='Actual', color='#000')
ax.plot(df_test['ds'], y_pred_lstm_denorm, label='Vanilla LSTM Prediction', color=
'#FF7274')
ax.plot(df_test['ds'], y_pred_peephole_denorm, label='Peephole LSTM Prediction', c
olor='#20948B')
ax.plot(df_test['ds'], y_pred_test_lstm_denorm2, '--
', label='Vanilla LSTM Prediction (with actual values)',  color='#FF7274')
ax.plot(df_test['ds'], y_pred_test_peephole_denorm2, '--
', label='Peephole LSTM Prediction (with actual values)', color='#20948B')
ax.plot(df_test['ds'], y_pred_prophet, label='Prophet Prediction', color='#DE7A22'
)
ax.set_title('Testing')
ax.grid()
ax.legend()
```

(lanjutan)

Menghitung nilai *Mean Absolute Error* 3 model

```
mae_train_prophet = mean_absolute_error(y_train_denorm, y_pred_train_prophet[60:])
mae_train_lstm = mean_absolute_error(y_train_denorm, y_pred_train_lstm_denorm)
mae_train_peephole = mean_absolute_error(y_train_denorm, y_pred_train_peephole_den
orm)

mae_test_prophet = mean_absolute_error(y_test_denorm, y_pred_prophet)
mae_test_lstm = mean_absolute_error(y_test_denorm, y_pred_lstm_denorm)
mae_test_peephole = mean_absolute_error(y_test_denorm, y_pred_peephole_denorm)
```

Menghitung nilai *Root Mean Squared Error* 3 model

```
rmse_train_prophet = np.sqrt(mean_squared_error(y_train_denorm, y_pred_train_proph
et[60:]))
rmse_train_lstm = np.sqrt(mean_squared_error(y_train_denorm, y_pred_train_lstm_den
orm))
rmse_train_peephole = np.sqrt(mean_squared_error(y_train_denorm, y_pred_train_peep
hole_denorm))

rmse_test_prophet = np.sqrt(mean_squared_error(y_test_denorm, y_pred_prophet))
rmse_test_lstm = np.sqrt(mean_squared_error(y_test_denorm, y_pred_lstm_denorm))
rmse_test_peephole = np.sqrt(mean_squared_error(y_test_denorm, y_pred_peephole_den
orm))
```

Menampilkan *barplot* hasil perhitungan MAE 3 model

```
def autolabel(rects):
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., height+0.015, f'{height:.3f}',
                ha='center', va='bottom')

labels = ['prophet', 'vanilla LSTM', 'peephole LSTM']
x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars
fig, ax = plt.subplots(figsize=(12,8))

rects1 = ax.bar(x - 0.02 - width/2, [mae_train_prophet, mae_train_lstm, mae_train_
peephole], width, label='training', color='#FBB1A8', edgecolor='#A77670')
rects2 = ax.bar(x + 0.02 + width/2, [mae_test_prophet, mae_test_lstm, mae_test_pee
phole], width, label='testing', color='#B0D5BF', edgecolor='#758E7F')
autolabel(rects1)
autolabel(rects2)

title = 'Mean Absolute Error by each model'
ax.set_ylabel('Mean Absolute Error')
ax.set_title(title)
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend(loc='lower right')
ax.grid()
```

(lanjutan)

Menampilkan *barplot* hasil perhitungan RMSE 3 model

```
labels = ['prophet', 'vanilla LSTM', 'peephole LSTM']
x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars
fig, ax = plt.subplots(figsize=(12,8))

rects1 = ax.bar(x - 0.02 - width/2, [rmse_train_prophet, rmse_train_lstm, rmse_tra
in_peephole], width, label='training', color='#FBB1A8', edgecolor='#A77670')
rects2 = ax.bar(x + 0.02 + width/2, [rmse_test_prophet, rmse_test_lstm, rmse_test_
peephole], width, label='testing', color='#B0D5BF', edgecolor='#758E7F')

autolabel(rects1)
autolabel(rects2)
title = 'Root Mean Squared Error by each model'

ax.set_ylabel('Root Mean Squared Error')
ax.set_title(title)
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend(loc='lower right')
ax.grid()
```