Penerapan Model LSTM dan GRU pada Prediksi Harga Saham TLKM

Nopandi Ariyanto
PREDICT

Contents

Problem Scooping

- Pada permasalahan ini dilakukan penerapan kinerja model dan membandingkan evaluasi model sistem yang dibuat dalam memprediksi pergerakan saham TLKM, model yang digunakan yaitu: Long Short Term Memory (LSTM) dan Gated Recurrent Units (GRU).

- Sebagai tambahannya, menggunakan variable harga penutupan (Close) selama satu tahun terakhir yaitu di tahun 2021-2022. Selanjutnya model dilakukan training dan testing pada dataset yang digunakan, sebagai evaluasi model dihitung dengan metode MSE

Rumusan Masalah

Berdasarkan permasalahan yang didapat maka ditentukan rumusan masalah sebagai berikut:

1. Bagaimana penerapan kinerja model dalam melakukan prediksi harga saham TLKM?

2. Bagaimana hasil perbandingan pada prediksi harga saham TLKM dengan menggunakan model LSTM dan GRU dari evaluasi model yang dilakukan?

Data & Variable

- Data yang digunakan yaitu sebuah dataset TLKM.CSV didapat dari sumber: https://finance.yahoo.com/quote/TLKM.JK/history?p=TLKM.JK

- Dalam dataset tersebut, terdapat beberapa variable diantaranya:
  a) Date
  b) Open
  c) High
  d) Low
  e) Close
  f) Adj Close
  g) Volume

```
#Menyiapkan dataset TLKM
data = pd.read_csv('TLKM.JK.csv')

data.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2017-01-02	3980.0	3980.0	3980.0	3980.0	3310.131592	0.0
1	2017-01-03	3950.0	3990.0	3920.0	3950.0	3285.180664	71660600.0
2	2017-01-04	3880.0	3980.0	3880.0	3950.0	3285.180664	68494500.0
3	2017-01-05	3960.0	4030.0	3940.0	3950.0	3285.180664	74018400.0
4	2017-01-06	3970.0	4010.0	3960.0	4000.0	3326.765381	44136900.0

Data Preprocessing

Data Cleaning (Missing Value)

Proses yang dilakukan yaitu mengecek missing value pada data dengan menampilkan fungsi .isnull() dan fugsi .dropna() pada parameter how='any' untuk meghilangkan banyaknya data yang tidak dikenali nilainya.

```
# Data masih terdapat Missing Value
data.isnull().sum()
```

```
Date         0
Open         1
High         1
Low          1
Close        1
Adj Close    1
Volume       1
dtype: int64
```

```
# Data sesudah dilakukan pembersihan
data = data.dropna(how='any',axis=0)
data.isnull().sum()
```

```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

```
# Menampilkan informasi data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1421 entries, 0 to 1421
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       1421 non-null   object 
 1   Open       1421 non-null   float64
 2   High       1421 non-null   float64
 3   Low        1421 non-null   float64
 4   Close      1421 non-null   float64
 5   Adj Close  1421 non-null   float64
 6   Volume     1421 non-null   float64
dtypes: float64(6), object(1)
memory usage: 88.8+ KB
```

Data Preprocessing

Data Category

Proses melakukan pengelompokkan pada variable sesuai dengan informasi data yang diperlukan

```
# Mengubah tipe data Date menjadi datetime
data["Date"] = pd.to_datetime(data["Date"])
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1421 entries, 0 to 1421
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       1421 non-null   datetime64[ns]
 1   Open       1421 non-null   float64
 2   High       1421 non-null   float64
 3   Low        1421 non-null   float64
 4   Close      1421 non-null   float64
 5   Adj Close  1421 non-null   float64
 6   Volume     1421 non-null   float64
dtypes: datetime64[ns](1), float64(6)
memory usage: 88.8 KB
```

```
#Mengurutkan data berdasarkan data dimulainya dan ditutupnya harga saham
print("Starting date: ", data.iloc[0][0])
print("Ending date: ", data.iloc[-1][0])
print("Duration: ", data.iloc[-1][0]-data.iloc[0][0])
```

```
Starting date:  2017-01-02 00:00:00
Ending date:  2022-09-01 00:00:00
Duration:  2068 days 00:00:00
```

```
# Seleksi variable data dan close untuk prediksi
tlkm_log = pd.DataFrame(columns=["Date","Close"])
tlkm_log["Date"] = data["Date"]
tlkm_log["Close"] = data["Close"]

# Specify datetime frequency
tlkm_log.dropna(inplace=True)

# ts_log.dropna(inplace=True)
tlkm_log
```

	Date	Close
0	2017-01-02	3980.0
1	2017-01-03	3950.0
2	2017-01-04	3950.0

```
# Mengubah varible Date menjadi Index
tlkm_log.set_index('Date', inplace=True)
#check datatype of index
tlkm_log
```

Date	Close
2021-03-12	3450.0
2021-03-15	3380.0
2021-03-16	3360.0
2021-03-17	3390.0
2021-03-18	3450.0

Data Preprocessing

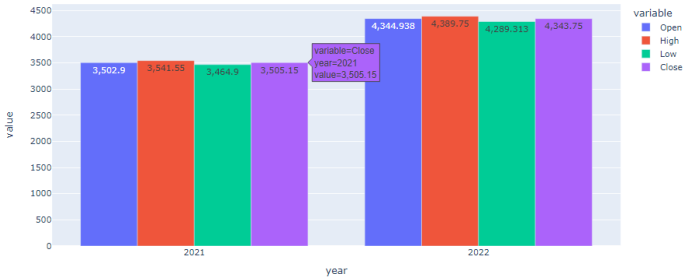
Data Category

Proses melakukan pengelompokkan pada variable sesuai dengan informasi data yang diperlukan

```
#Membuat variable df untuk mengkategorikan bulan dan tahun
df = data.copy()
df['year'] = pd.Index(df['Date']).year
df['month'] = pd.Index(df['Date']).month
df['month_year'] = df['Date'].dt.to_period('M')
```

```
#Mengkategorikan distribusi Open, High, Low dan Close pertahun
stocks_year = df.groupby(by='year').mean()
fig = px.bar(stocks_year, x=stocks_year.index,
             y=stocks_year.iloc[:,:4].columns,
             title="Distrubitions of Opening, High, Low and Closing Prices by Years",
             barmode = 'group',text_auto=True)
fig.show()
```

Distrubitions of Opening, High, Low and Closing Prices by Years

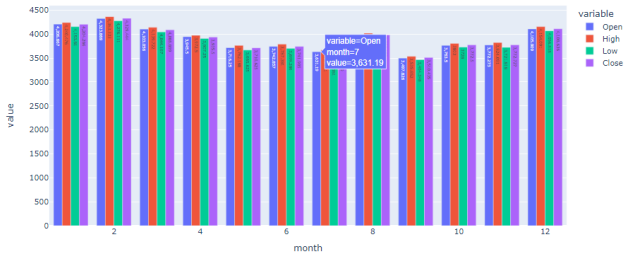



```
#Membuat data satu tahun terakhir dengan nilai index
data = data[-12*30:].reset_index(drop=True)
data
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2021-03-12	3440.0	3480.0	3430.0	3450.0	3168.497070	100195200.0
1	2021-03-15	3440.0	3450.0	3380.0	3380.0	3104.208984	50077300.0
2	2021-03-16	3400.0	3410.0	3360.0	3360.0	3085.840820	60111200.0
3	2021-03-17	3410.0	3430.0	3370.0	3390.0	3113.392822	61701400.0
4	2021-03-18	3450.0	3480.0	3430.0	3450.0	3168.497070	101896400.0
...	...	...	...	...	...	...	...
355	2022-08-26	4510.0	4540.0	4480.0	4490.0	4490.000000	184317100.0
356	2022-08-29	4370.0	4550.0	4360.0	4520.0	4520.000000	101529800.0

```
#Visualisasi rata-rata harga saham pertahunnya
stocks_year = df.groupby(by='month').mean()
fig = px.bar(stocks_year, x=stocks_year.index,
             y=stocks_year.iloc[:,:4].columns,
             title="Average Stock Prices by Month",
             barmode = 'group',text_auto=True)
fig.show()
```

Average Stock Prices by Month



Data Preprocessing

Train Test Split

Pemisahan/pembagian data digunakan untuk membedakan antara data yang digunakan untuk proses pelatihan dan data yang digunakan untuk proses pengujian.

```
# Menentukan banyaknya data train yaitu sebesar 80% data
train_size = int(len(tlkm_log) * 0.8)
train = tlkm_log[:train_size]
test = tlkm_log[train_size:].reset_index(drop=True)
```

```
#Melakukan proses data training dan testing menggunakan MinMaxScaller
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(train[['Close']])

training_scaled = scaler.transform(train[['Close']])
testing_scaled = scaler.transform(test[['Close']])
train
```

Date	Close
2021-03-12	3450.0
2021-03-15	3380.0
2021-03-16	3360.0
2021-03-17	3390.0
2021-03-18	3450.0
...	...
2022-05-12	4300.0
2022-05-13	4260.0
2022-05-17	4180.0

Data Preprocessing

Data Training, Testing Prediksi

Pemisahan/pembagian data digunakan untuk membedakan antara data yang digunakan untuk proses pelatihan sebelum hypertunning pada model, data yang digunakan untuk proses pengujian dan prediksi.

Data Training Model

```
#Menyiapkan data train sebelum dilakukan hypertunning pada Model
X_train = []
y_train = []
for i in range(30, training_scaled.shape[0]):
    X_train.append(training_scaled[i-30:i,0])
    y_train.append(training_scaled[i,0])

X_train, y_train = np.array(X_train), np.array(y_train)
X_train = np.reshape(X_train, (X_train.shape[0],X_train.shape[1],1))
```

Data Testing LSTM

```
#Menyiapkan data test untuk prediksi
X_test = []
y_test= []
for i in range(30,input.shape[0]):
    X_test.append(input[i-30:i,0])
    y_test.append(input[i,0])
X_test,y_test = np.array(X_test),np.array(y_test)
y_pred = lstm_reg.predict(X_test)
y_pred[-1,0]
```

```
#Menyiapkan data test untuk prediksi
X_test = []
y_test= []
for i in range(30,input.shape[0]):
    X_test.append(input[i-30:i,0])
    y_test.append(input[i,0])
X_test,y_test = np.array(X_test),np.array(y_test)
y_pred = gru_reg.predict(X_test)
y_pred[-1,0]
```

Data Testing GRU

Model & Parameter

LSTM

Para meter yang digunakan
- Optimizer = adam
- Loss = MSE
- Metrics = MAE

GRU

Para meter yang digunakan
- Optimizer = Sigmoid
- Loss = MSE
- Metrics = MAE

Kinerja Model

Model LSTM

Melakukan hypertunning parameter dengan LSTM pada unit: 50 dan dropout: 0.2, serta pembagian model berdasarkan 5 layer dan output layer dengan activation: ReLu, lalu di compile menggunakan optimizer: adam, loss: mean_squared_error dan melakukan fitting pada model berdasarkan data train, epoch: 50, dan batch size: 32 yang digunakan.

Hypertunning

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 30, 50)            10400

 dropout (Dropout)           (None, 30, 50)            0

 lstm_1 (LSTM)               (None, 30, 50)            20200

 dropout_1 (Dropout)         (None, 30, 50)            0

 lstm_2 (LSTM)               (None, 30, 50)            20200

 dropout_2 (Dropout)         (None, 30, 50)            0

 lstm_3 (LSTM)               (None, 30, 50)            20200

 dropout_3 (Dropout)         (None, 30, 50)            0

 lstm_4 (LSTM)               (None, 50)                20200

 dropout_4 (Dropout)         (None, 50)                0

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 91,251
Trainable params: 91,251
Non-trainable params: 0
```

Fitting Model

```
Epoch 40/50
9/9 [==============================] - 0s 52ms/step - loss: 0.0113 - mae: 0.0876
Epoch 41/50
9/9 [==============================] - 0s 50ms/step - loss: 0.0092 - mae: 0.0766
Epoch 42/50
9/9 [==============================] - 1s 55ms/step - loss: 0.0101 - mae: 0.0792
Epoch 43/50
9/9 [==============================] - 1s 55ms/step - loss: 0.0144 - mae: 0.0981
Epoch 44/50
9/9 [==============================] - 0s 50ms/step - loss: 0.0157 - mae: 0.1014
Epoch 45/50
9/9 [==============================] - 1s 57ms/step - loss: 0.0136 - mae: 0.0910
Epoch 46/50
9/9 [==============================] - 1s 75ms/step - loss: 0.0114 - mae: 0.0843
Epoch 47/50
9/9 [==============================] - 1s 62ms/step - loss: 0.0097 - mae: 0.0803
Epoch 48/50
9/9 [==============================] - 1s 56ms/step - loss: 0.0099 - mae: 0.0813
Epoch 49/50
9/9 [==============================] - 0s 50ms/step - loss: 0.0109 - mae: 0.0835
Epoch 50/50
9/9 [==============================] - 0s 52ms/step - loss: 0.0105 - mae: 0.0820
<keras.callbacks.History at 0x7f142487a0d0>
```

Kinerja Model

Model GRU

Melakukan hypertunning parameter dengan GRU pada unit: 50 dan dropout: 0.2, serta pembagian model berdasarkan 5 layer dan output layer dengan activation: Sigmoid, lalu di compile menggunakan optimizer: SGD, loss: mean_squared_error dan melakukan fitting pada model berdasarkan data train, epoch: 50, dan batch size: 32 yang digunakan.

Hypertunning

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 30, 50)            7950

 dropout_5 (Dropout)         (None, 30, 50)            0

 gru_1 (GRU)                 (None, 30, 50)            15300

 dropout_6 (Dropout)         (None, 30, 50)            0

 gru_2 (GRU)                 (None, 30, 50)            15300

 dropout_7 (Dropout)         (None, 30, 50)            0

 gru_3 (GRU)                 (None, 30, 50)            15300

 dropout_8 (Dropout)         (None, 30, 50)            0

 gru_4 (GRU)                 (None, 50)                15300

 dropout_9 (Dropout)         (None, 50)                0

 dense_1 (Dense)             (None, 1)                 51

=================================================================
Total params: 69,201
Trainable params: 69,201
Non-trainable params: 0
```

Fitting Model

```
Epoch 40/50
9/9 [==============================] - 0s 50ms/step - loss: 0.0061 - mae: 0.0629
Epoch 41/50
9/9 [==============================] - 0s 49ms/step - loss: 0.0058 - mae: 0.0602
Epoch 42/50
9/9 [==============================] - 0s 51ms/step - loss: 0.0060 - mae: 0.0627
Epoch 43/50
9/9 [==============================] - 0s 51ms/step - loss: 0.0059 - mae: 0.0620
Epoch 44/50
9/9 [==============================] - 0s 49ms/step - loss: 0.0067 - mae: 0.0663
Epoch 45/50
9/9 [==============================] - 0s 51ms/step - loss: 0.0060 - mae: 0.0621
Epoch 46/50
9/9 [==============================] - 1s 59ms/step - loss: 0.0061 - mae: 0.0625
Epoch 47/50
9/9 [==============================] - 1s 59ms/step - loss: 0.0064 - mae: 0.0643
Epoch 48/50
9/9 [==============================] - 0s 50ms/step - loss: 0.0063 - mae: 0.0650
Epoch 49/50
9/9 [==============================] - 0s 51ms/step - loss: 0.0060 - mae: 0.0614
Epoch 50/50
9/9 [==============================] - 0s 53ms/step - loss: 0.0062 - mae: 0.0642
<keras.callbacks.History at 0x7f14240a7b10>
```

Evaluasi Model

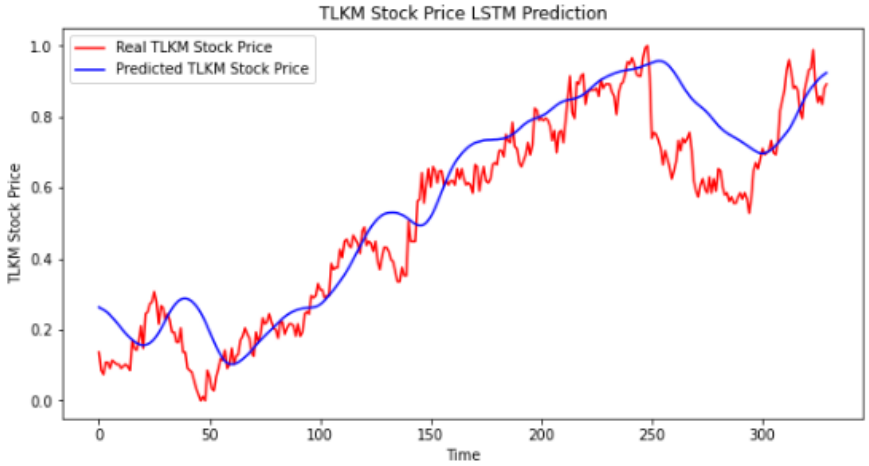
Hasil Evaluasi LSTM

Berdasarkan hasil training dan testing model menggunakan perhitungan Mean Squared Error (MSE) memperoleh hasil terbaiknya adalah: 0,0118

```
#Membuat variable dan fungsi MSE
def return_mse(test,predicted):
    mse = mean_squared_error(test, predicted)
    print("The Mean Squared Error = {}.".format(mse))
```

```
# Menampilkan hasil perhitungan MSE Evaluasi Model LSTM
return_mse(y_test,y_pred)
```

```
The Mean Squared Error = 0.011825426247456701.
```



Evaluasi Model

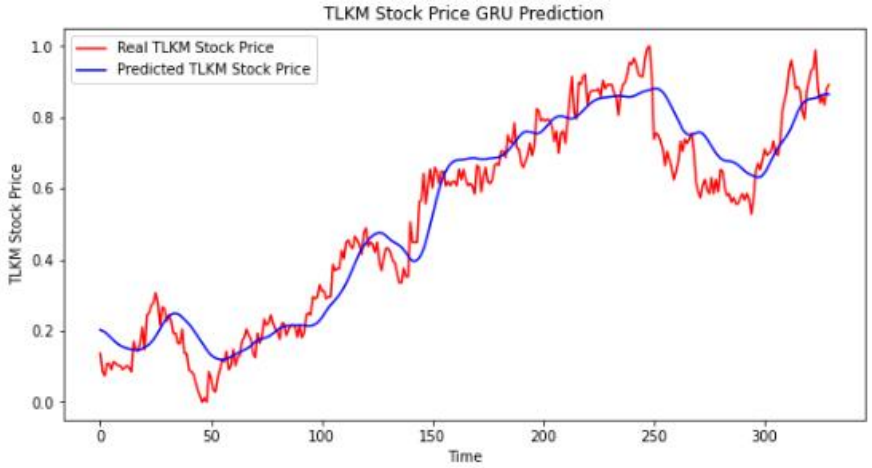
Hasil Evaluasi GRU

Berdasarkan hasil training dan testing model menggunakan perhitungan Mean Squared Error (MSE) memperoleh hasil terbaiknya adalah: 0,0058

```
#Membuat variable dan fungsi MSE
def return_mse(test,predicted):
    mse = mean_squared_error(test, predicted)
    print("The Mean Squared Error = {}.".format(mse))
```

```
# Menampilkan hasil perhitungan MSE Evaluasi Model GRU
return_mse(y_test,y_pred)
```

```
The Mean Squared Error = 0.005888035060882174.
```



Conclusion

- Prediksi harga saham memakai data Close sebagai fiture sesuai di data 2021-2022 dengan menerapkan kinerja model, yaitu melakukan hypertunning pada model LSTM dan GRU pada unit: 50 dan dropout: 0.2 berdasarkan 5 layer dan output layer dengan activation: ReLU(LSTM) dan Sigmoid(GRU) serta compiling menggnakan optimizer: adam(LSTM), SGD(GRU), loss: mean_squared_errror dan melakukan fitting pada model dari data train, epoch: 50, dan batch size: 32 yang digunakan.

- Hasil yang terbaik prediksi harga saham dari Model LSTM didapat dengan perhitungan Mean Squered Error memperoleh hasil MSE: 0.0118. Sedangkan perhitungan Model GRU dengan Mean Squered Error memperoleh hasil MSE: 0.0058. Perhitungan nilai MSE yang rendah atau MSE mendekati nol menunjukkan bahwa hasil prediksi sesuai dengan data aktual.

- Dengan demikian dapat disimpulkan bahwa penerapan model GRU lebih baik dibandingkan dengan modelLSTM dalam memprediksi harga saham TLKM, mampu menanggulangi ketergantungan jangka panjang dan mampu memprediksi harga saham dengan hasil yang akurat.

Thank You!