Documentation

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
```

pandas, numpy — manipulasi data
matplotlib — visualisasi grafik
scikit-learn — normalisasi data
tensorflow.keras — membangun dan melatih model LSTM

```
df = pd.read_csv('bbca_2000_2024.csv', skiprows=2)
df.columns = ['Date', 'Close', 'High', 'Low', 'Open', 'Volume']
df['Date'] = pd.to_datetime(df['Date'])
df = df.sort_values('Date')
df.dropna(inplace=True)
df.set_index('Date', inplace=True)
print(df.head())
```

Menghapus baris keterangan di atas header

Menganti nama kolom.

Konversi tanggal menjadi datetime.

Menghapus data yang kosong dan mengatur Date sebagai index.

Menampilakan 5 kolom pertama

```
[ ]  train_df = df['2004':'2019']
     test_df = df['2020':'2024']
```

Dikarenakan datanya banyak dari tahiun 2004 hingga 2024 kita akan membagi datanya menjadi 2 set 1 untuk testing dan 1 untuk training, untuk data training kitta gunakan 15 tahun dari 2004 hingga 2019, dan data testing kita gunakan 5 tahun daro 2020 – 2024/

```
[ ]  scaler = MinMaxScaler()
     scaled_train = scaler.fit_transform(train_df)
     scaled_test = scaler.transform(test_df)
```

Data dinormalisasi ke rentang [0, 1] menggunakan MinMaxScaler.

```python
def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(seq_length, len(data)):
        X.append(data[i-seq_length:i])
        y.append(data[i, 0])
    return np.array(X), np.array(y)

sequence_length = 60
X_train, y_train = create_sequences(scaled_train, sequence_length)
X_test, y_test = create_sequences(scaled_test, sequence_length)
```

Mengubah data deret waktu menjadi potongan-potongan berisi 60 hari (sebagai input model).

```python
model = Sequential()
model.add(LSTM(64, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))
model.add(LSTM(64))
model.add(Dropout(0.2))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, y_train, epochs=25, batch_size=32)
```

Terdiri dari 2 layer LSTM dengan dropout untuk menghindari overfitting, Layer terakhir Dense(1) menghasilkan prediksi harga penutupan.

1. model.compile(...)
Fungsi ini digunakan untuk meng-konfigurasi model sebelum dilatih.

optimizer='adam':
Optimizer adalah algoritma yang digunakan untuk memperbarui bobot (weights) selama proses training.
Adam adalah optimizer yang paling umum digunakan karena menggabungkan kelebihan dari optimizers lain seperti SGD, RMSProp, dan Momentum.

loss='mean_squared_error':
Fungsi kerugian (loss function) digunakan untuk mengukur seberapa jauh prediksi model dari nilai aktual.

Mean Squared Error (MSE) menghitung rata-rata kuadrat dari selisih antara nilai aktual dan prediksi. Cocok digunakan untuk regresi seperti prediksi harga.

2. model.fit(...)
Digunakan untuk melatih model menggunakan data yang sudah disiapkan (X_train, y_train).

epochs=25:
Jumlah epoch berarti berapa kali seluruh data latih akan diproses oleh model.
Dalam hal ini, model akan melihat data sebanyak 25 kali secara penuh.

batch_size=32:

Menentukan berapa banyak data yang diproses sekaligus sebelum model memperbarui bobotnya.
sebelum memperbarui parameter.

```
predicted_scaled = model.predict(X_test)

dummy = np.zeros((len(predicted_scaled), scaled_test.shape[1]))
dummy[:, 0] = predicted_scaled[:, 0]
predicted_close = scaler.inverse_transform(dummy)[:, 0]
```

Menghasilkan prediksi harga penutupan (Close) dari data uji (X_test).
Mengembalikan hasil prediksi dari bentuk ternormalisasi ke bentuk asli (harga sebenarnya dalam rupiah).

```
actual_close = test_df['Close'].values[sequence_length:]
```

Tujuan dari baris ini adalah untuk mengambil nilai harga penutupan (Close) aktual dari data uji (test_df) — dimulai dari indeks ke-60 dan seterusnya.

```
[ ] import matplotlib.dates as mdates

    plt.figure(figsize=(12,6))
    plt.plot(test_df.index[sequence_length:], actual_close, color='blue', label='Actual Close Price')
    plt.plot(test_df.index[sequence_length:], predicted_close, color='red', label='Predicted Close Price')
    plt.title('BBCA Stock Price Prediction (2020-2024)')
    plt.xlabel('Year')
    plt.ylabel('Close Price (Rp)')
    plt.legend()


    plt.gca().xaxis.set_major_locator(mdates.YearLocator())
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
    plt.xticks(rotation=45)
    plt.tight_layout()

    plt.show()
```

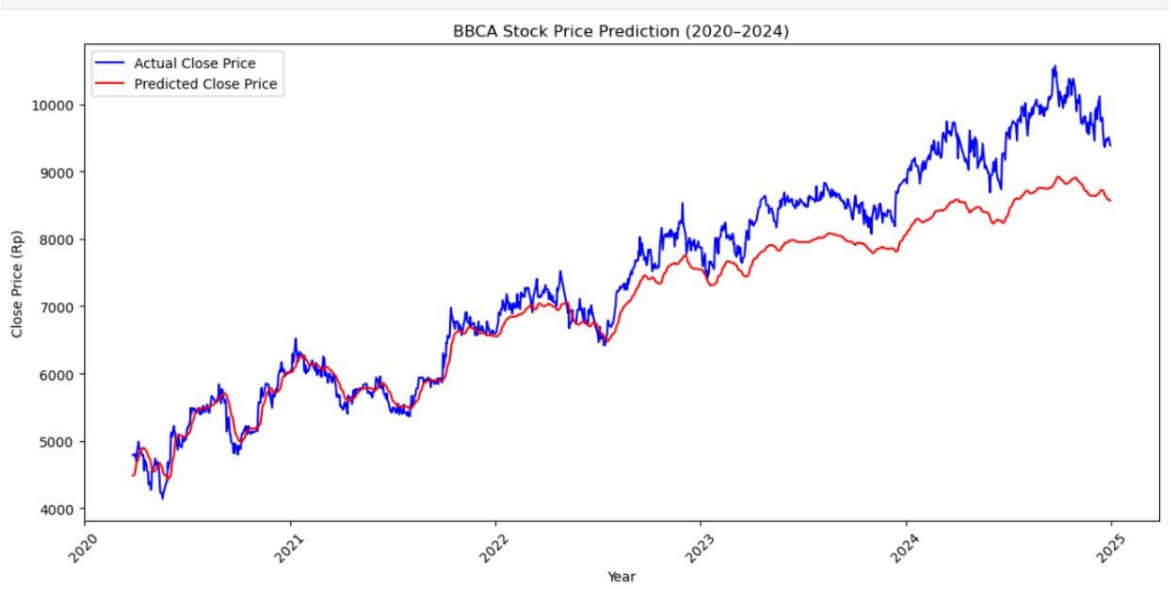
Mengvisualisasikan hasil prediksi modelnya modelnya degan hasil aktualnya dalam bentuk line graph mengunakan matplotlib

```python
def mean_absolute_percentage_error(y_true, y_pred):
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mape = mean_absolute_percentage_error(actual_close, predicted_close)
accuracy = 100 - mape

print(f"MAPE: {mape:.2f}%")
print(f"Akurasi Model (approx): {accuracy:.2f}%")
```

```
MAPE: 5.14%
Akurasi Model (approx): 94.86%
```

Menghitung akurasi model mengunakan MAPE(Mean Absolute Percentage Eror)

HASIL MODEL




```
MAPE: 5.14%
Akurasi Model (approx): 94.86%
```