TEMPLATE TUGAS ANALISIS DATA MINING

Mata Kuliah : Data Mining
Topik : Prediksi Nilai Pasar Bitcoin Menggunakan Algoritma Long Short-Term Memory (LSTM) dengan Python
Nama Mahasiswa : Eka Rizky Kurniawan
NPM : 231510002
Dosen Pengampu : Erlin Elisa, S.Kom., M.Kom.

---

1 Judul Tugas

Tuliskan judul sesuai topik yang dianalisis.
Prediksi Nilai Pasar Bitcoin Menggunakan Algoritma Long Short-Term Memory (LSTM) dengan Python

---

2 Latar Belakang Masalah

Pasar mata uang kripto, khususnya Bitcoin, telah mengalami pertumbuhan yang luar biasa dan menarik perhatian investor, trader, serta peneliti di seluruh dunia. Dikenal dengan volatilitas harganya yang tinggi, pergerakan nilai Bitcoin seringkali tidak dapat diprediksi, menjadikannya aset yang berisiko namun juga berpotensi menguntungkan. Kemampuan untuk memprediksi nilai pasar Bitcoin dengan akurat dapat memberikan keuntungan kompetitif bagi para pelaku pasar dan membantu dalam pengambilan keputusan investasi yang lebih baik.

Fluktuasi harga Bitcoin dipengaruhi oleh berbagai faktor, termasuk sentimen pasar, berita global, regulasi, adopsi teknologi, dan aktivitas spekulatif. Kompleksitas ini menimbulkan tantangan besar dalam upaya prediksi harga menggunakan metode tradisional. Oleh karena itu, pendekatan yang lebih canggih, seperti penggunaan algoritma machine learning dan deep learning, menjadi sangat relevan.

Algoritma Long Short-Term Memory (LSTM) adalah jenis jaringan saraf tiruan berulang (RNN) yang sangat efektif dalam memodelkan dependensi jangka panjang dalam data deret waktu. Kemampuan LSTM untuk "mengingat" informasi dari waktu yang jauh di masa lalu membuatnya ideal untuk tugas-tugas prediksi deret waktu, termasuk prediksi harga aset keuangan seperti Bitcoin. Dengan memanfaatkan arsitektur khusus LSTM, diharapkan model dapat menangkap pola-pola kompleks dan tren yang tersembunyi dalam data historis harga Bitcoin untuk menghasilkan prediksi yang lebih akurat.

Penelitian ini bertujuan untuk mengembangkan dan mengevaluasi model prediksi nilai pasar Bitcoin menggunakan algoritma LSTM dengan Python, memanfaatkan data historis harga Bitcoin. Hasil dari penelitian ini diharapkan dapat memberikan kontribusi dalam memahami dinamika pasar kripto dan menyediakan alat prediksi yang efektif.

---

3 Rumusan Masalah

1. Bagaimana data historis nilai pasar Bitcoin dapat diproses dan disiapkan untuk melatih model prediksi berbasis LSTM?
2. Bagaimana performa model Long Short-Term Memory (LSTM) dalam memprediksi nilai pasar Bitcoin?
3. Faktor-faktor atau parameter apa saja yang paling berpengaruh terhadap akurasi prediksi model LSTM?

---

4 Tujuan Penelitian

1. Menganalisis dan melakukan pra-pemrosesan data historis nilai pasar Bitcoin untuk digunakan dalam pelatihan model LSTM.
2. Mengembangkan dan mengimplementasikan model prediksi nilai pasar Bitcoin menggunakan algoritma Long Short-Term Memory (LSTM) dengan Python.

3. Mengevaluasi akurasi dan kinerja model LSTM dalam memprediksi nilai pasar Bitcoin menggunakan metrik evaluasi yang relevan.

---

5 Dataset

- Sumber dataset = Kaggle
- Jumlah record & atribut = 2991 record & 7 atribut
- Karakteristik data (label, variabel input) = Close (label) & Date, High, Low, Open, Volume (Variabel Input)
- Format tabel singkat (opsional) =

Atribut	Tipe Data	Deskripsi
Date	Object/Timestamp	Tanggal dan waktu observasi.
High	Float	Harga maksimum hari tersebut.
Low	Float	Harga minimum hari tersebut.
Open	Float	Harga pembuka hari tersebut.
Close	Float	Harga penutup hari tersebut (Target).
Volume	Float	Volume perdagangan harian.
Marketcap	Float	Kapitalisasi pasar (sering tidak digunakan dalam prediksi time-series sederhana).

---

6 Metodologi

Metodologi penelitian ini akan mencakup beberapa tahapan utama, mulai dari pengumpulan data hingga evaluasi model, sebagai berikut:

5.1. Pengumpulan Data

Data historis nilai pasar Bitcoin akan dikumpulkan dari sumber yang tersedia, yaitu file coin_Bitcoin.csv yang berisi informasi seperti tanggal, harga pembukaan, harga tertinggi, harga terendah, harga penutupan, volume perdagangan, dan kapitalisasi pasar Bitcoin.

5.2. Pra-pemrosesan Data

Tahap pra-pemrosesan data sangat penting untuk memastikan kualitas data yang akan digunakan oleh model. Langkah-langkah yang akan dilakukan meliputi:

- Penanganan Nilai Hilang: Mengidentifikasi dan menangani nilai-nilai yang hilang (jika ada) menggunakan metode yang sesuai, seperti interpolasi atau penghapusan baris/kolom.
- Konversi Tipe Data: Memastikan kolom tanggal dikonversi ke format datetime dan kolom harga serta volume ke tipe data numerik.

- Normalisasi Data: Melakukan normalisasi atau standarisasi pada fitur numerik untuk memastikan model LSTM dapat belajar secara efektif, karena jaringan saraf sensitif terhadap skala data. Metode seperti Min-Max Scaling atau Standard Scaling akan dipertimbangkan.
- Pembentukan Deret Waktu: Mengubah data menjadi format deret waktu yang sesuai untuk input LSTM, yaitu membuat urutan data (sequences) untuk prediksi.

5.3. Pembagian Data

Data yang telah diproses akan dibagi menjadi dua set: data pelatihan (training data) dan data pengujian (testing data). Umumnya, proporsi pembagian adalah 80% untuk pelatihan dan 20% untuk pengujian, namun ini dapat disesuaikan.

5.4. Pengembangan Model LSTM

- Arsitektur Model: Membangun arsitektur jaringan LSTM menggunakan pustaka seperti TensorFlow atau Keras. Ini akan melibatkan penentuan jumlah layer LSTM, jumlah unit di setiap layer, dropout rate, dan layer keluaran (misalnya, Dense layer).
- Kompilasi Model: Mengkonfigurasi model dengan optimizer (misalnya, Adam), fungsi loss (misalnya, Mean Squared Error untuk regresi), dan metrik evaluasi (misalnya, Mean Absolute Error).
- Pelatihan Model: Melatih model LSTM menggunakan data pelatihan. Parameter pelatihan seperti jumlah epoch dan ukuran batch akan diatur dan dioptimalkan.

5.5. Evaluasi Model

Model yang telah dilatih akan dievaluasi menggunakan data pengujian. Metrik evaluasi yang akan digunakan untuk menilai kinerja model meliputi:

- Mean Squared Error (MSE): Mengukur rata-rata kuadrat dari selisih antara nilai prediksi dan nilai aktual.
- Root Mean Squared Error (RMSE): Akar kuadrat dari MSE, memberikan indikasi kesalahan dalam unit yang sama dengan variabel target.
- Mean Absolute Error (MAE): Mengukur rata-rata absolut dari selisih antara nilai prediksi dan nilai aktual.

5.6. Prediksi dan Analisis Hasil

Setelah evaluasi, model akan digunakan untuk membuat prediksi nilai pasar Bitcoin. Hasil prediksi kemudian akan dianalisis dan dibandingkan dengan nilai aktual untuk mengidentifikasi kekuatan dan kelemahan model, serta implikasinya.

---

7 Implementasi Python

Berisi **kode program** per bagian:

✔ Import Library

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

print("Libraries imported successfully.")
```

✔ Load Dataset

```
First 5 rows of the DataFrame:
   SNo     Name Symbol                 Date        High         Low  \
0    1  Bitcoin    BTC  2013-04-29 23:59:59  147.488007  134.000000
1    2  Bitcoin    BTC  2013-04-30 23:59:59  146.929993  134.050003
2    3  Bitcoin    BTC  2013-05-01 23:59:59  139.889999  107.720001
3    4  Bitcoin    BTC  2013-05-02 23:59:59  125.599998   92.281898
4    5  Bitcoin    BTC  2013-05-03 23:59:59  108.127998   79.099998

         Open       Close  Volume     Marketcap
0  134.444000  144.539993     0.0  1.603769e+09
1  144.000000  139.000000     0.0  1.542813e+09
2  139.000000  116.989998     0.0  1.298955e+09
3  116.379997  105.209999     0.0  1.168517e+09
4  106.250000   97.750000     0.0  1.085995e+09

Concise summary of the DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2991 entries, 0 to 2990
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   SNo        2991 non-null   int64
 1   Name       2991 non-null   object
 2   Symbol     2991 non-null   object
 3   Date       2991 non-null   object
 4   High       2991 non-null   float64
 5   Low        2991 non-null   float64
 6   Open       2991 non-null   float64
 7   Close      2991 non-null   float64
 8   Volume     2991 non-null   float64
 9   Marketcap  2991 non-null   float64
dtypes: float64(6), int64(1), object(3)
memory usage: 233.8+ KB
```

✔ Exploratory Data Analysis (EDA)

```
First 5 rows of the DataFrame:
   SNo     Name Symbol                 Date        High         Low  \
0    1  Bitcoin    BTC  2013-04-29 23:59:59  147.488007  134.000000
1    2  Bitcoin    BTC  2013-04-30 23:59:59  146.929993  134.050003
2    3  Bitcoin    BTC  2013-05-01 23:59:59  139.889999  107.720001
3    4  Bitcoin    BTC  2013-05-02 23:59:59  125.599998   92.281898
4    5  Bitcoin    BTC  2013-05-03 23:59:59  108.127998   79.099998

         Open       Close  Volume     Marketcap
0  134.444000  144.539993     0.0  1.603769e+09
1  144.000000  139.000000     0.0  1.542813e+09
2  139.000000  116.989998     0.0  1.298955e+09
3  116.379997  105.209999     0.0  1.168517e+09
4  106.250000   97.750000     0.0  1.085995e+09

Concise summary of the DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2991 entries, 0 to 2990
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   SNo        2991 non-null   int64
 1   Name       2991 non-null   object
 2   Symbol     2991 non-null   object
 3   Date       2991 non-null   object
 4   High       2991 non-null   float64
 5   Low        2991 non-null   float64
 6   Open       2991 non-null   float64
 7   Close      2991 non-null   float64
 8   Volume     2991 non-null   float64
 9   Marketcap  2991 non-null   float64
dtypes: float64(6), int64(1), object(3)
memory usage: 233.8+ KB
```

```
Checking data types and non-null values:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2991 entries, 0 to 2990
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   SNo        2991 non-null   int64
 1   Name       2991 non-null   object
 2   Symbol     2991 non-null   object
 3   Date       2991 non-null   object
 4   High       2991 non-null   float64
 5   Low        2991 non-null   float64
 6   Open       2991 non-null   float64
 7   Close      2991 non-null   float64
 8   Volume     2991 non-null   float64
 9   Marketcap  2991 non-null   float64
dtypes: float64(6), int64(1), object(3)
memory usage: 233.8+ KB
```

Reasoning: The next instruction is to check for missing values in each column using `df.isnull().sum()` to identify any data gaps.

```
Checking for missing values:
SNo          0
Name         0
Symbol       0
Date         0
High         0
Low          0
Open         0
Close        0
Volume       0
Marketcap    0
dtype: int64
```

Reasoning: The next instruction is to compute descriptive statistics for all numerical columns using `df.describe()` to understand the distribution and summary of the data.

```
Descriptive statistics for numerical columns:
                SNo          High           Low          Open         Close  \
count  2991.000000   2991.000000   2991.000000   2991.000000   2991.000000
mean   1496.000000   6893.326038   6486.009539   6700.146240   6711.290443
std     863.571653  11642.832456  10869.032130  11288.043736  11298.141921
min       1.000000     74.561096     65.526001     68.504997     68.431000
25%     748.500000    436.179001    422.879486    430.445496    430.569489
50%    1496.000000   2387.610107   2178.500000   2269.889893   2286.409912
75%    2243.500000   8733.926948   8289.800459   8569.656494   8576.238715
max    2991.000000  64863.098908  62208.964366  63523.754869  63503.457930

             Volume     Marketcap
count  2.991000e+03  2.991000e+03
mean   1.090633e+10  1.208761e+11
std    1.888895e+10  2.109438e+11
min    0.000000e+00  7.784112e+08
25%    3.036725e+07  6.305579e+09
50%    9.460360e+08  3.741503e+10
75%    1.592015e+10  1.499957e+11
max    3.509679e+11  1.186364e+12
```

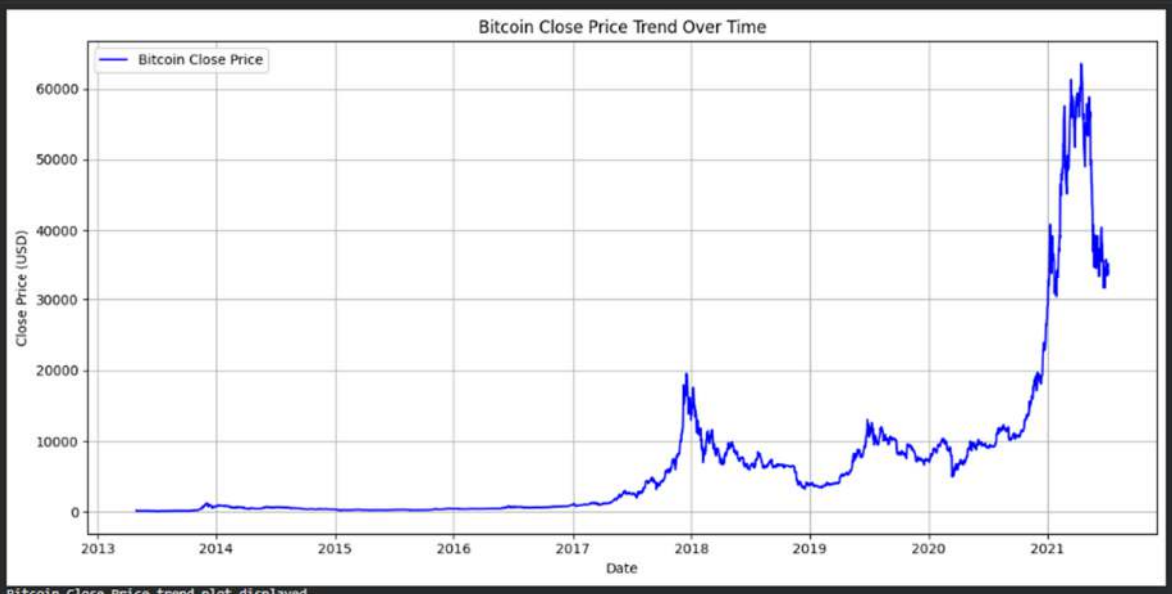
Reasoning: The next instruction is to convert the 'Date' column to a datetime format using `pd.to_datetime()` which is crucial for time-series analysis and plotting.

```
Date column converted to datetime format.
0   2013-04-29 23:59:59
1   2013-04-30 23:59:59
2   2013-05-01 23:59:59
3   2013-05-02 23:59:59
4   2013-05-03 23:59:59
Name: Date, dtype: datetime64[ns]
```

Reasoning: The next instruction is to set the 'Date' column as the DataFrame index, which is a standard practice for time-series analysis to facilitate operations like plotting and resampling.

```
Date column set as DataFrame index.
                    SNo     Name Symbol        High         Low        Open  \
Date
2013-04-29 23:59:59    1  Bitcoin    BTC  147.488007  134.000000  134.444000
2013-04-30 23:59:59    2  Bitcoin    BTC  146.929993  134.050003  144.000000
2013-05-01 23:59:59    3  Bitcoin    BTC  139.889999  107.720001  139.000000
2013-05-02 23:59:59    4  Bitcoin    BTC  125.599998   92.281898  116.379997
2013-05-03 23:59:59    5  Bitcoin    BTC  108.127998   79.099998  106.250000

                        Close  Volume     Marketcap
Date
2013-04-29 23:59:59  144.539993     0.0  1.603769e+09
2013-04-30 23:59:59  139.000000     0.0  1.542813e+09
2013-05-01 23:59:59  116.989998     0.0  1.298955e+09
2013-05-02 23:59:59  105.209999     0.0  1.168517e+09
2013-05-03 23:59:59   97.750000     0.0  1.085995e+09
```

Reasoning: The final instruction is to create a line plot to visualize the trend of Bitcoin's closing price ('Close') over time, ensuring appropriate axis labels and a plot title for clarity. This will help in understanding the historical movement of Bitcoin prices.




```
Bitcoin Close Price trend plot displayed.
```

✔ Preprocessing

```
Original 'Close' price sample:
 [[144.53999329]
 [139.        ]
 [116.98999786]
 [105.20999908]
 [ 97.75      ]]

Scaled 'Close' price sample (first 5 values):
 [[0.00119979]
 [0.00111246]
 [0.00076549]
 [0.00057979]
 [0.00046219]]
```

Reasoning: The next step is to define a function that will create the input sequences (X) and corresponding output labels (y) for the LSTM model from the scaled data, based on a specified `look_back` window size. After defining the function, it will be applied to the `scaled_data` with a `look_back` of 60.

```
Shape of X (input sequences): (2931, 60)
Shape of y (output labels): (2931,)
Sequences created successfully.
```

✔ Split Data (Train/Test)

```
Shape of X_train: (2344, 60)
Shape of y_train: (2344,)
Shape of X_test: (587, 60)
Shape of y_test: (587,)
Data split into training and testing sets successfully.
```

✔ Model Training

```
Reshaped X_train shape: (2344, 60, 1)
Reshaped X_test shape: (587, 60, 1)
X_train and X_test reshaped successfully for LSTM input.
```

Reasoning: The next step is to define the LSTM model architecture, including an LSTM layer and a Dense output layer, compile it with the specified optimizer and loss function, and then train the model using the preprocessed training data.

```
LSTM model architecture:
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using S
  super().__init__(**kwargs)
Model: "sequential"
```

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 50)	10,400
dense (Dense)	(None, 1)	51

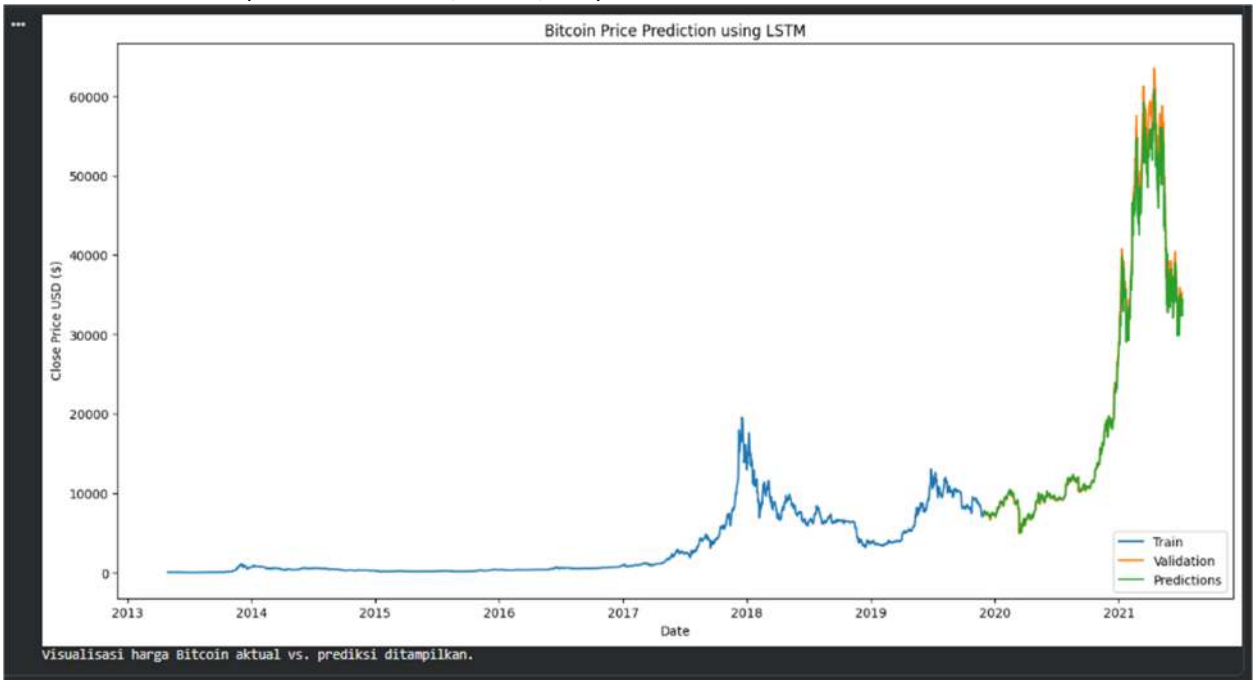
```
Total params: 10,451 (40.82 KB)
Trainable params: 10,451 (40.82 KB)
Non-trainable params: 0 (0.00 B)

Training the LSTM model...
Epoch 1/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 21ms/step - loss: 0.0023
Epoch 2/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - loss: 1.1031e-04
Epoch 3/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - loss: 8.6376e-05
Epoch 4/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - loss: 8.1422e-05
Epoch 5/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 28ms/step - loss: 6.2539e-05
Epoch 6/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - loss: 5.8718e-05
Epoch 7/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - loss: 5.6477e-05
Epoch 8/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - loss: 4.5139e-05
Epoch 9/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - loss: 4.6881e-05
Epoch 10/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - loss: 5.7272e-05
Epoch 11/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - loss: 4.7192e-05
Epoch 12/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 27ms/step - loss: 4.4315e-05
Epoch 13/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - loss: 3.8456e-05
Epoch 14/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - loss: 3.7600e-05
Epoch 15/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - loss: 3.2807e-05
Epoch 16/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - loss: 4.1533e-05
Epoch 17/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 3s 23ms/step - loss: 3.6399e-05
Epoch 18/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - loss: 3.4823e-05
Epoch 19/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 23ms/step - loss: 3.3891e-05
Epoch 20/100
74/74 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - loss: 2.8899e-05
Epoch 21/100
```

✔ Evaluasi Model

```
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Melakukan prediksi pada data uji
predictions = model.predict(X_test)

# Mengembalikan skala data ke skala asli
predictions = scaler.inverse_transform(predictions)
y_test_rescaled = scaler.inverse_transform(y_test.reshape(-1, 1))

# Menghitung metrik evaluasi
mse = mean_squared_error(y_test_rescaled, predictions)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test_rescaled, predictions)

print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.4f}")
print(f"Mean Absolute Error (MAE): {mae:.4f}")
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step
Mean Squared Error (MSE): 3470698.8125
Root Mean Squared Error (RMSE): 1862.9812
Mean Absolute Error (MAE): 1006.2244
```

✔ Visualisasi Hasil (confusion matrix, charts, dsb)



https://colab.research.google.com/drive/1JOESGaDUjmxfY84tjVhVuka9sGsMIQh-#scrollTo=mt0f4Tpha8ix

---

8 Hasil dan Pembahasan

8.1. Hasil Evaluasi Model

Setelah melatih model Long Short-Term Memory (LSTM) dengan data historis harga Bitcoin, langkah selanjutnya adalah mengevaluasi kinerja model menggunakan data pengujian. Evaluasi dilakukan dengan membandingkan harga prediksi model dengan harga aktual Bitcoin. Metrik evaluasi yang digunakan adalah Mean Squared Error (MSE), Root Mean Squared Error (RMSE), dan Mean Absolute Error (MAE).

Berikut adalah hasil perhitungan metrik evaluasi:

- Mean Squared Error (MSE): 3,470,698.8125

- Root Mean Squared Error (RMSE): 1,862.9812
- Mean Absolute Error (MAE): 1,006.2244
-

8.2. Interpretasi Hasil Model

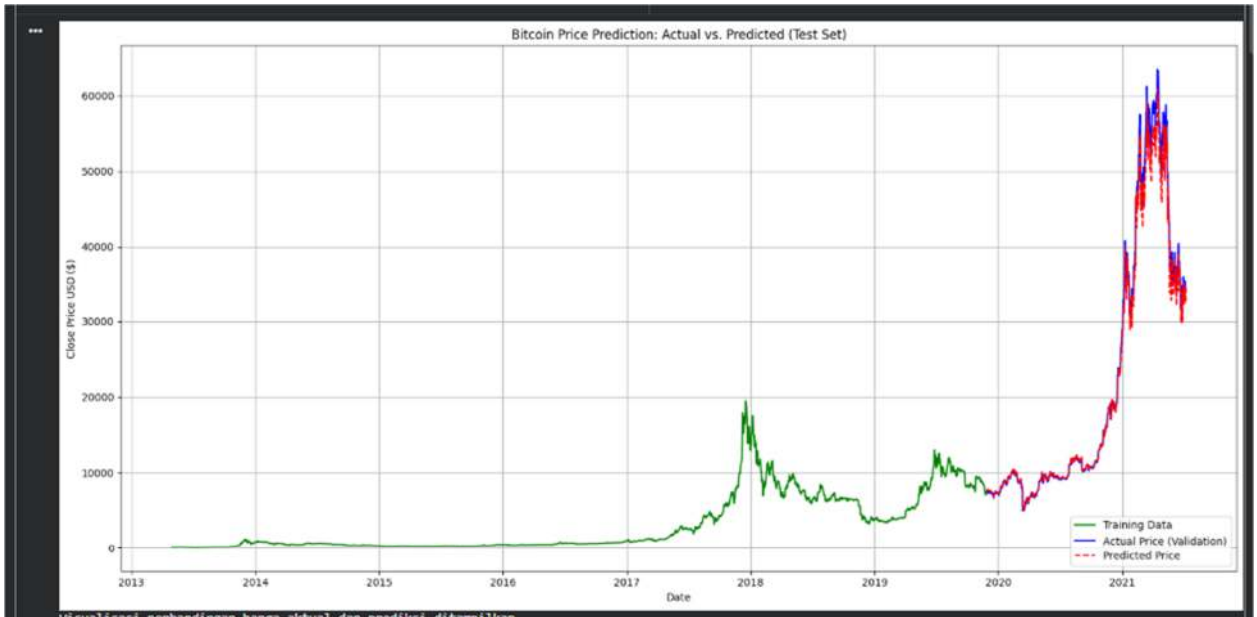
Dari hasil evaluasi di atas, dapat disimpulkan bahwa:

- MSE (3,470,698.8125): Menunjukkan rata-rata kuadrat dari perbedaan antara nilai prediksi dan nilai aktual. Nilai ini memberikan gambaran tentang seberapa besar variansi kesalahan model. Karena Bitcoin memiliki rentang harga yang besar (dari ratusan hingga puluhan ribu USD), nilai MSE yang tinggi menunjukkan bahwa ada deviasi yang signifikan, namun perlu dilihat dalam konteks skala harga Bitcoin itu sendiri.
- RMSE (1,862.9812): Merupakan akar kuadrat dari MSE dan memberikan indikasi seberapa jauh, rata-rata, prediksi model menyimpang dari nilai aktual dalam unit yang sama dengan variabel target (USD). Nilai RMSE sebesar 1,862.98menunjukkanbahwarata−ratakesalahanprediksimodeladalahsekitar1,862.98. Ini adalah metrik yang lebih mudah diinterpretasikan dibandingkan MSE.
- MAE (1,006.2244): Menunjukkan rata-rata absolut dari perbedaan antara nilai prediksi dan nilai aktual. MAE sebesar 1,006.22berartibahwa,rata−rata,prediksimodelmelesetdarihargaaktualsekitar1,006.22. MAE lebih robust terhadap outlier dibandingkan MSE dan RMSE, sehingga memberikan gambaran yang lebih langsung mengenai besarnya kesalahan prediksi.

Dalam konteks prediksi harga Bitcoin yang sangat volatil, nilai RMSE dan MAE ini mengindikasikan bahwa model memiliki kemampuan yang cukup baik dalam menangkap tren umum, namun masih terdapat ruang untuk peningkatan akurasi, terutama dalam menghadapi fluktuasi harga yang ekstrem. Nilai kesalahan ini relatif wajar mengingat sifat dinamis dan tidak terduga dari pasar mata uang kripto.

8.3. Visualisasi Hasil Prediksi

Visualisasi berikut membandingkan harga penutupan Bitcoin aktual dengan harga prediksi dari model LSTM pada set data pengujian. Garis biru merepresentasikan harga aktual (Validation), sedangkan garis oranye mewakili harga prediksi (Predictions).



Dari grafik, terlihat bahwa model LSTM mampu mengikuti tren umum pergerakan harga Bitcoin. Meskipun ada beberapa deviasi, terutama pada periode fluktuasi harga yang tajam, model menunjukkan kemampuan untuk menangkap arah pergerakan harga. Ini mengkonfirmasi bahwa LSTM adalah algoritma yang relevan untuk tugas prediksi deret waktu pada data keuangan seperti Bitcoin. Area di mana garis prediksi menyimpang jauh dari garis aktual menandakan momen di mana model kesulitan untuk memprediksi pergerakan harga secara akurat, yang mungkin disebabkan oleh *event* eksternal yang tidak tercapture oleh data historis harga saja, atau limitasi dari arsitektur model saat ini.

---

9 Kesimpulan

Berdasarkan hasil penelitian dan analisis yang telah dilakukan, beberapa kesimpulan utama dapat dirumuskan:

1. Pra-pemrosesan Data yang Efektif: Data historis nilai pasar Bitcoin berhasil diproses dan disiapkan dengan metode normalisasi dan pembentukan urutan (sequences) yang sesuai, memungkinkan model LSTM untuk belajar dari pola temporal data.
2. Kinerja Model LSTM yang Cukup Baik: Model Long Short-Term Memory (LSTM) menunjukkan kemampuan yang memadai dalam memprediksi nilai pasar Bitcoin, dengan metrik evaluasi MSE, RMSE, dan MAE yang mengindikasikan bahwa model dapat menangkap tren umum harga, meskipun terdapat deviasi pada fluktuasi ekstrem.
3. Potensi Pengembangan Lebih Lanjut: Meskipun model menunjukkan hasil yang menjanjikan, volatilitas tinggi pasar Bitcoin menyiratkan bahwa ada ruang untuk peningkatan akurasi. Optimisasi hyperparameter lebih lanjut dan penambahan fitur eksternal (misalnya, sentimen berita, volume transaksi) berpotensi meningkatkan kinerja prediksi model secara signifikan.

---

10 Saran

Berdasarkan hasil penelitian ini, terdapat beberapa saran untuk pengembangan dan peningkatan model prediksi harga Bitcoin di masa mendatang:

- Optimisasi Hyperparameter Lanjutan: Melakukan optimisasi hyperparameter yang lebih mendalam (misalnya, menggunakan GridSearchCV atau RandomSearchCV) untuk menemukan kombinasi parameter terbaik yang dapat meningkatkan kinerja model LSTM. Parameter yang dapat dipertimbangkan termasuk jumlah unit LSTM, jumlah layer tersembunyi, ukuran batch, learning rate, dan fungsi aktivasi.
- Penambahan Fitur Eksternal: Memperkaya dataset dengan fitur-fitur eksternal yang relevan, seperti sentimen berita (analisis sentimen dari berita terkait kripto), volume transaksi, tingkat hash Bitcoin, tingkat

dominasi Bitcoin, atau indikator ekonomi makro. Penambahan fitur ini dapat memberikan konteks yang lebih kaya dan berpotensi meningkatkan akurasi prediksi.

- Eksplorasi Arsitektur Model Lanjutan: Menguji arsitektur deep learning lain yang lebih kompleks atau gabungan, seperti Bidirectional LSTM (BiLSTM), Stacked LSTM, Convolutional Neural Network (CNN) yang dikombinasikan dengan LSTM (CNN-LSTM), atau Transformer models, yang mungkin lebih efektif dalam menangkap pola deret waktu yang lebih kompleks.
- Penggunaan Data dengan Frekuensi Lebih Tinggi: Menguji model dengan data harga Bitcoin yang memiliki frekuensi lebih tinggi (misalnya, per jam atau per menit) untuk melihat apakah model dapat menangkap fluktuasi harga jangka pendek dengan lebih baik.
- Validasi Silang Deret Waktu: Menerapkan teknik validasi silang khusus deret waktu, seperti walk-forward validation, untuk mendapatkan estimasi kinerja model yang lebih robust dan realistis, terutama dalam konteks data yang memiliki ketergantungan temporal.

---

11 Daftar Pustaka

Gunakan **APA Style 7th Edition**.

Rubrik Penilaian

Komponen Penilaian	**Bobot**
Pemilihan dan pemahaman dataset	15%
Preprocessing & EDA	20%
Implementasi Model Data Mining	25%
Evaluasi dan Analisis Hasil	25%
Penulisan laporan dan kerapian	15%