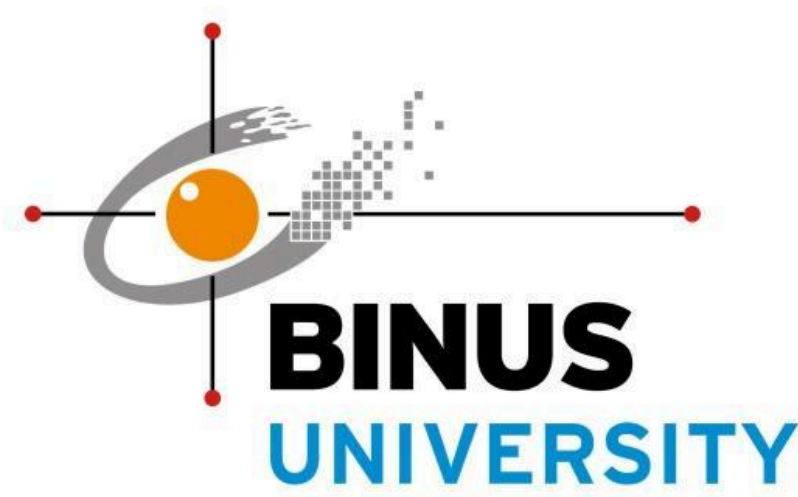


LAPORAN PROJECT MACHINE LEARNING



PREDIKSI TREN SAHAM: PENERAPAN DEEP LEARNING UNTUK ANALISIS PASAR SAHAM

Oleh

Louis	2602147276	Computer Science
Clarice Arlin Wijaya	2602135143	Computer Science
Janssen Mitchellano Hamaziah	2602117525	Computer Science

UNIVERSITAS BINA NUSANTARA JAKARTA

2024

1. LATAR BELAKANG

Pasar saham adalah salah satu aspek paling krusial dalam dunia investasi. Pergerakan harga saham dipengaruhi oleh banyak faktor termasuk kondisi ekonomi, perkembangan politik, dan sentimen pasar. Memprediksi harga saham menjadi topik yang sangat menarik dan penting karena keputusan investasi yang dibuat berdasarkan prediksi ini dapat berujung pada keuntungan atau kerugian finansial yang signifikan. Mengingat kompleksitas dan volatilitas pasar saham, metode tradisional seringkali kurang efektif dalam memprediksi pergerakan harga.

Perkembangan teknologi informasi dan data science dalam beberapa tahun terakhir telah membuka peluang baru dalam analisis pasar saham. Dengan menggunakan teknik machine learning, kita bisa menggali pola tersembunyi dari data historis dan memprediksi harga saham di masa depan dengan lebih akurat. Proyek ini bertujuan untuk mengembangkan model prediksi harga saham yang dapat digunakan untuk berbagai simbol saham, memberikan wawasan yang lebih mendalam, dan membantu investor dalam membuat keputusan yang lebih tepat.

Pasar saham yang dinamis menuntut pendekatan yang lebih adaptif dan canggih dalam menganalisa dan memprediksi pergerakan harga. Data yang tersedia sangat melimpah dan kompleks, sehingga diperlukan alat analisis yang mampu mengolah informasi tersebut secara efisien dan efektif. Machine learning menawarkan solusi inovatif dengan algoritma yang bisa belajar dari data historis dan mengidentifikasi pola yang mungkin tidak terlihat dengan metode konvensional.

Selain itu, dengan meningkatnya ketersediaan data real-time dan kemampuan komputasi yang lebih tinggi, prediksi harga saham kini dapat dilakukan dengan lebih cepat dan akurat. Ini memberikan keuntungan tidak hanya bagi investor individual tetapi juga institusi keuangan yang mengelola portofolio besar. Dengan prediksi yang lebih baik, risiko bisa dikelola dengan lebih efektif dan peluang keuntungan bisa dimaksimalkan.

Proyek ini juga mengutamakan aspek visualisasi data untuk menyampaikan informasi yang kompleks dengan cara yang lebih mudah dipahami. Melalui pengembangan

website interaktif, pengguna dapat melihat data historis dan hasil prediksi dalam format yang intuitif, sehingga memudahkan analisis dan pengambilan keputusan.

2. TUJUAN

- a. Mengembangkan model prediksi harga saham
- b. Menyediakan visualisasi data interaktif
- c. Meningkatkan akurasi prediksi
- d. Menyediakan alat analisis untuk investor


3. DATASET


Dataset yang digunakan dalam proyek ini diambil dari Yahoo Finance, sumber terpercaya untuk informasi pasar saham. Dataset awal mencakup informasi penting seperti harga penutupan harian, volume perdagangan, harga pembukaan, harga tertinggi, dan harga terendah untuk berbagai simbol saham.

XAU/USD Historical Data ⓘ

Time Frame

Daily ▾

 Download Data

02/05/2024 - 03/03/2024 

Date	Price	Open	High	Low	Vol.	Change %
03/01/2024	2,083.39	2,043.44	2,088.40	2,038.55		+1.96%
02/29/2024	2,043.24	2,034.92	2,050.79	2,027.75		+0.42%
02/28/2024	2,034.62	2,030.05	2,038.30	2,024.56		+0.25%
02/27/2024	2,029.64	2,030.99	2,039.99	2,028.78		-0.05%
02/26/2024	2,030.66	2,034.49	2,037.59	2,025.10		-0.25%
02/23/2024	2,035.72	2,024.49	2,041.43	2,015.55		+0.57%
02/22/2024	2,024.11	2,025.24	2,034.84	2,019.70		-0.04%
02/21/2024	2,024.99	2,023.80	2,032.22	2,020.19		+0.07%
02/20/2024	2,023.53	2,017.99	2,030.96	2,015.02		+0.29%
02/19/2024	2,017.63	2,013.16	2,023.34	2,011.57		+0.22%
02/16/2024	2,013.10	2,004.40	2,015.25	1,995.26		+0.45%
02/15/2024	2,004.09	1,992.69	2,008.49	1,990.25		+0.59%
02/14/2024	1,992.39	1,992.55	1,996.14	1,984.30		+0.01%

Data ini akan diolah dan dianalisis menggunakan teknik machine learning untuk mengembangkan model prediksi harga saham yang akurat. Pengambilan data dilakukan dengan menggunakan pustaka `yfinance`, yang memungkinkan pengambilan data secara otomatis dan pembaruan data secara *real-time*, memastikan model selalu menggunakan informasi terbaru untuk prediksi yang lebih akurat.

4. METODE

4.1. Data Preparation

4.1.1. Import Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_absolute_percentage_error
from math import sqrt
import tensorflow as tf
from keras import Model
from keras.layers import Input, Dense, Dropout
from keras.layers import LSTM
import plotly.graph_objects as go
```

Pertama, import library python yang dibutuhkan untuk analisis data seperti `numpy`, `pandas`, `matplotlib`, `plotly`, dan `yfinance`. `yfinance` sendiri merupakan *library* python yang menyediakan data dari pasar saham melalui Yahoo finance.

4.1.2. Download Data

```
import yfinance as yf
start = '2010-01-01'
end = '2024-01-01'
stock = 'XAUT-USD'
data = yf.download(stock, start, end)
data.reset_index(inplace = True)
```

Data diambil 1 Januari 2010 hingga 31 Desember 2024. Namun, data yang tercatat pada Yahoo! Finance berawal dari 7 Februari 2020 akan tetapi ini tidak akan menjadi masalah untuk *training LSTM* model kita.

- 'XAUT-USD' merupakan simbol yang mewakili harga atau nilai tukar antara emas (XAU) dan Dolar Amerika Serikat (USD) di pasar keuangan. Ini sebagai simbol *default*, yang nantinya dapat diubah sesuai inputan.
- 'yf.download' digunakan untuk mengunduh data historis harga saham, mata uang, ataupun emas.
- `reset_index` berfungsi untuk melakukan pengaturan ulang pada index ke *settingan default* (0 ke n)

4.1.3. Display Data

data							
	Date	Open	High	Low	Close	Adj Close	Volume
0	2020-02-07	1582.213135	1585.620361	1576.241089	1581.786865	1581.786865	1305750
1	2020-02-08	1581.844116	1587.325439	1567.624634	1578.869751	1578.869751	1609288
2	2020-02-09	1578.880005	1590.733154	1572.618164	1578.702271	1578.702271	575041
3	2020-02-10	1579.318970	1594.563721	1560.719360	1572.435547	1572.435547	123025
4	2020-02-11	1572.398926	1617.188721	1546.927002	1563.408325	1563.408325	340994
...
1419	2023-12-27	2067.487061	2083.567383	2063.218018	2074.964355	2074.964355	4770225
1420	2023-12-28	2075.123047	2087.295654	2066.581787	2072.599121	2072.599121	4990665
1421	2023-12-29	2072.603760	2078.412354	2060.760742	2068.144775	2068.144775	5308010
1422	2023-12-30	2068.159912	2069.790527	2060.790771	2067.288574	2067.288574	5317621

Data yfinance XAUT-USD memiliki beberapa variabel (Date, Open, High, Low, Close, Adj Close, Volume)

4.2. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) merujuk pada proses penting untuk memahami karakteristik dan struktur dari dataset. Pada tahap ini, dilakukan penyelidikan awal terhadap data untuk menemukan pola, tren, dan anomali dengan bantuan summary statistics dan visualisasi grafis.

Eksplorasi data dimulai dengan mengecek informasi dataset. Hal ini penting untuk memahami karakteristik dan struktur data sebelum melakukan analisis lebih lanjut. Didapatkan informasi dataset sebagai berikut:



Nilai tukar antara emas dan Dolar Amerika Serikat menunjukkan adanya tren yang fluktuatif tapi cenderung naik seiring waktu.

```
[9] data.drop(['Adj Close', 'Volume'], axis = 1, inplace = True)
    data.head()
```

	Date	Open	High	Low	Close
0	2020-02-07	1582.213135	1585.620361	1576.241089	1581.786865
1	2020-02-08	1581.844116	1587.325439	1567.624634	1578.869751
2	2020-02-09	1578.880005	1590.733154	1572.618164	1578.702271
3	2020-02-10	1579.318970	1594.563721	1560.719360	1572.435547
4	2020-02-11	1572.398926	1617.188721	1546.927002	1563.408325

Dalam kasus ini, ada kolom yang tidak diperlukan dalam analisis seperti kolom 'Adj Close' yang memiliki data yang sama dengan kolom 'Close' dan juga 'Volume', sehingga dapat dihapus untuk mengurangi kompleksitas dan fokus pada fitur yang relevan. Selain itu, penting untuk memeriksa apakah terdapat nilai yang hilang dalam dataset agar dapat ditangani nantinya.

Selanjutnya, eksplorasi dilanjutkan dalam beberapa tahapan seperti:

4.2.1. Analisis Statistik Deskriptif

Analisis ini dilakukan untuk membantu mendapatkan gambaran awal tentang sebaran dan karakteristik dasar dari data. Hal ini melibatkan peninjauan nilai-nilai utama seperti mean, median, dan deviasi standar.

```
data.describe()
```

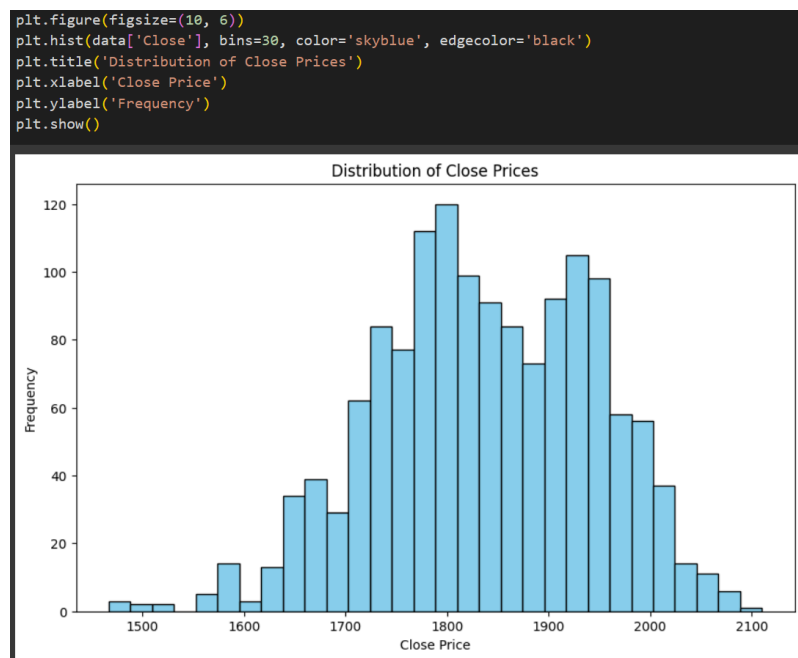
Ringkasan statistik deskriptif:

	Date	Open	High	Low	Close	Volume
count	1424	1424.000000	1424.000000	1424.000000	1424.000000	1.424000e+03
mean	2022-01-18 12:00:00	1836.495990	1848.578122	1825.145456	1836.914046	1.940930e+06
min	2020-02-07 00:00:00	1466.926514	1504.015747	1408.882324	1466.926514	1.099000e+03
25%	2021-01-27 18:00:00	1764.909424	1776.206360	1754.835693	1764.998566	1.765810e+05
50%	2022-01-18 12:00:00	1835.339355	1847.020996	1823.907593	1835.474243	5.428170e+05

Dari statistik deskriptif ini, kita mendapatkan gambaran umum tentang distribusi dan variasi harga aset serta volume perdagangan selama periode waktu 7 Februari 2020 sampai dengan 31 Desember 2023.

4.2.2. Visualisasi Distribusi Data

Berbagai jenis plot, seperti histogram atau box plot digunakan untuk mengeksplorasi distribusi data lebih lanjut. Visualisasi ini membantu dalam mengidentifikasi pola dan melihat sebaran nilai dalam dataset dengan lebih jelas.

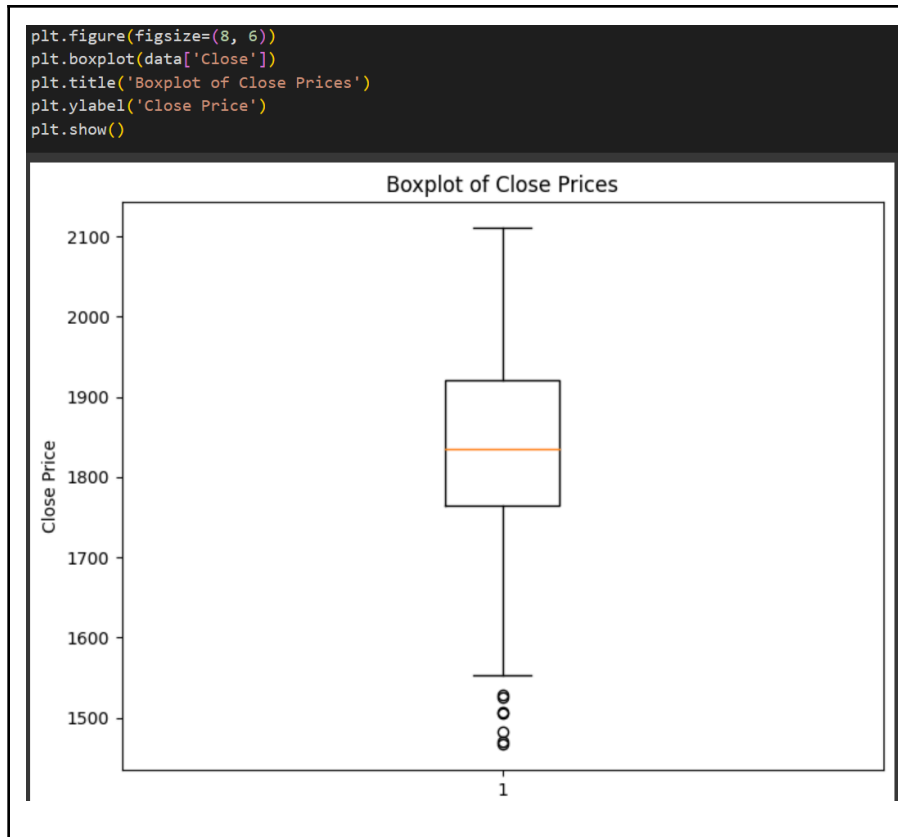


Histogram di atas membantu kita untuk memahami distribusi harga penutupan saham secara grafis. Sementara box plot di bawah membantu untuk melihat sebaran harga pada 'Open', 'High', 'Low', dan 'Close'.



4.2.3. Identifikasi Outliers

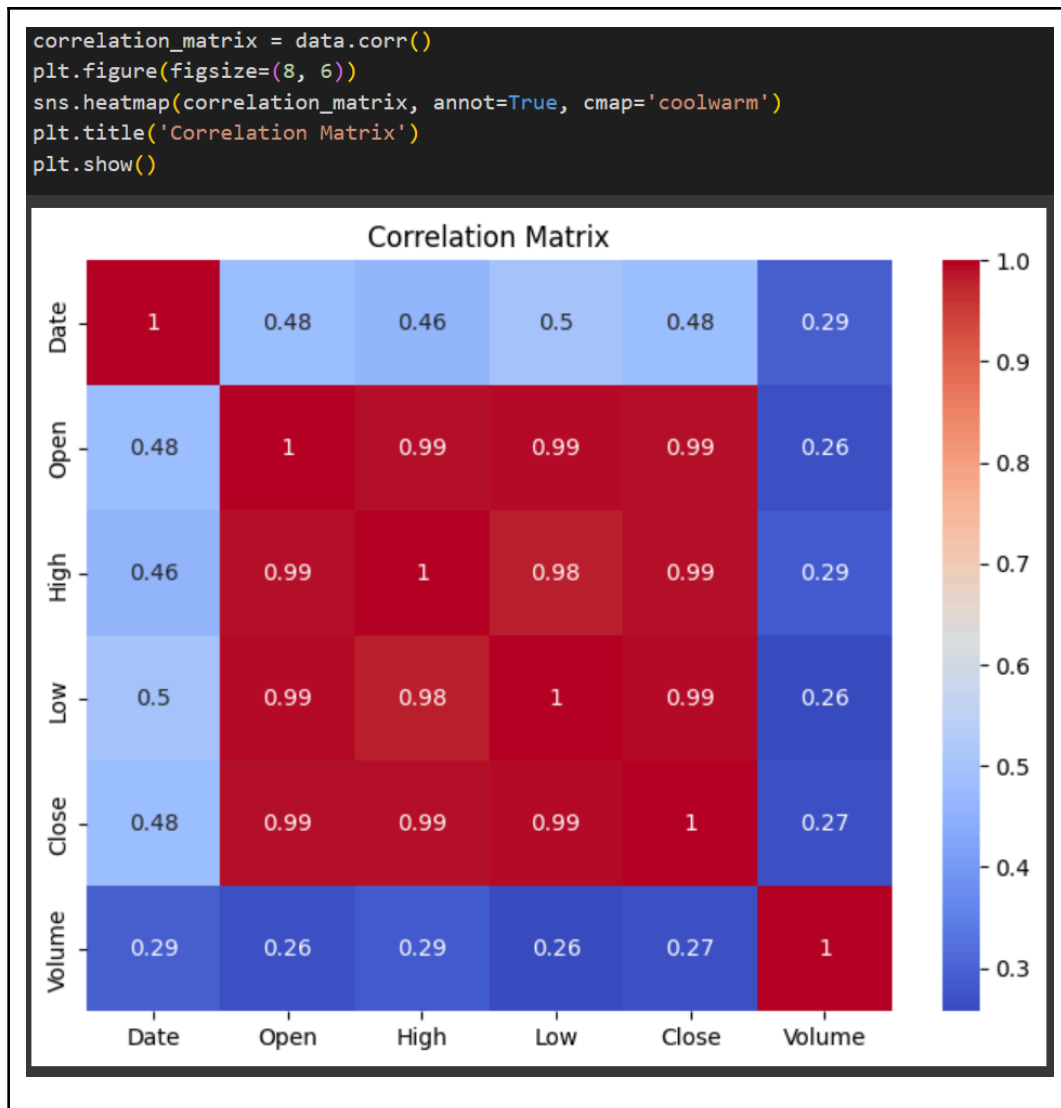
Keberadaann outliers dalam data diidentifikasi menggunakan metode box plot. Outlier adalah nilai yang jauh berbeda dari sebagian besar data dan dapat memiliki pengaruh yang signifikan terhadap hasil analisis.



Box plot di atas menunjukkan adanya beberapa outlier pada 'Close Price' di kisaran 1500.

4.2.4. Eksplorasi Korelasi

Dengan menganalisis korelasi antara atribut-atribut dalam dataset menggunakan heatmap, kami dapat memahami hubungan antar variabel. Ini membantu dalam menentukan atribut-atribut yang saling berkaitan dan relevan untuk analisis lebih lanjut.



Heatmap korelasi di atas menunjukkan adanya hubungan yang kuat antara fitur ‘Open’, ‘High’, ‘Low’, dan ‘Close’. Dengan informasi ini, keempat fitur tersebut akan menjadi fitur penting dalam penganalisisan dan pembangunan model lebih lanjut.

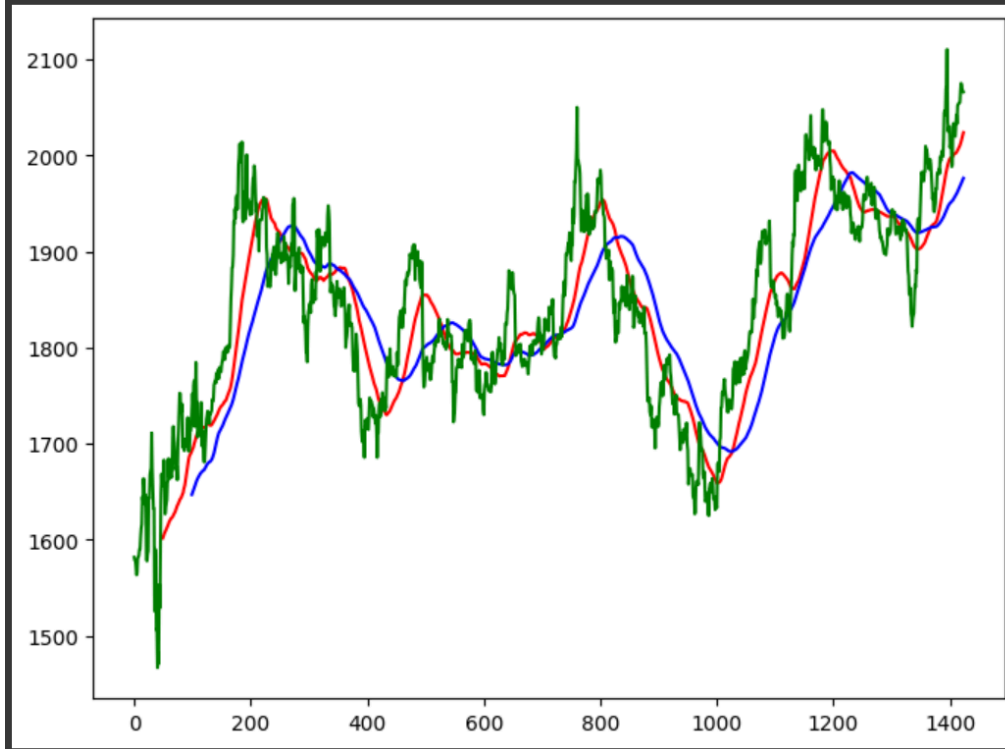
4.2.5. Rata-rata Pergerakan

Moving average adalah salah satu konsep yang penting dalam analisis data dan sering digunakan dalam prediksi tren. Ini adalah metode statistik untuk menghaluskan data dengan mengambil rata-rata dari sejumlah titik data berturut-turut dalam rentang waktu tertentu, sehingga memungkinkan kita untuk melihat tren atau pola yang mendasarinya.

```

movingAve_50 = data.Close.rolling(50).mean()
movingAve_100 = data.Close.rolling(100).mean()
plt.figure(figsize=(8, 6))
plt.plot(movingAve_50, 'r')
plt.plot(movingAve_100, 'b')
plt.plot(data.Close, 'g')
plt.show()

```



Moving average di atas mencari rata-rata 'Close' dari rentang waktu tertentu.

- Garis merah merepresentasikan moving average 50 hari terakhir
- Garis biru merepresentasikan moving average 100 hari terakhir
- Garis hijau merepresentasikan data 'Close'

Plot di atas menunjukkan bahwa moving average 50 hari lebih relevan dalam memperkirakan pergerakan data 'Close'. Hal ini menandakan bahwa moving average 50 hari akan dapat lebih diandalkan dalam memprediksi pergerakan harga 'Close'.

4.3. Pre-processing Data

Sebelum data dapat digunakan untuk melatih model, pre-processing data pun dilakukan untuk membersihkan, menormalkan, dan mengatur data agar sesuai dengan

kebutuhan analisis dan proses pemodelan lanjutan, termasuk dalam tahap pemilihan fitur, pelatihan model, dan evaluasi performa.

Berikut adalah langkah-langkah yang kami terapkan dalam pre-processing data untuk meningkatkan kemampuan model kami:

4.3.1. Feature Selection

Feature selection merupakan proses dari data preprocessing dimana memilih feature subset yang paling relevan dan berguna untuk melakukan sebuah prediksi. Dalam kasus kita kali ini kita tidak memerlukan “Adj Close”, “Volume”. Hal itu karena “Adj Close” memiliki nilai yang sama dengan “Close”, sedangkan kolom “Volume” tidak digunakan untuk melakukan prediksi.

```
data.drop(['Adj Close', 'Volume'], axis = 1, inplace = True)
data.head()
```

	Date	Open	High	Low	Close
0	2020-02-07	1582.213135	1585.620361	1576.241089	1581.786865
1	2020-02-08	1581.844116	1587.325439	1567.624634	1578.869751
2	2020-02-09	1578.880005	1590.733154	1572.618164	1578.702271
3	2020-02-10	1579.318970	1594.563721	1560.719360	1572.435547
4	2020-02-11	1572.398926	1617.188721	1546.927002	1563.408325

4.3.2. Transforming Data Types

Karena data variable “date” bersifat object maka, kita perlu merubah variable ini ke data type yang benar. Data types variable date yang bersifat object harus diubah menjadi datetime

```
data['Date'] = pd.to_datetime(data['Date'])
data.sort_values(by = 'Date', ascending = True, inplace = True)
```




4.3.4. Data Splitting

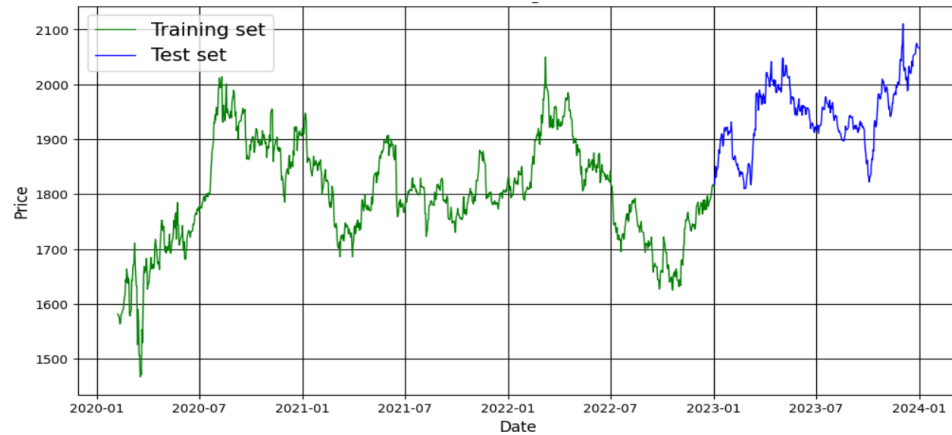
Pembagian data dilakukan dengan menghitung jumlah baris yang akan digunakan untuk data latih dan data uji, kemudian membagi dataset berdasarkan proporsi tersebut. Langkah ini memungkinkan kita untuk menguji kinerja model pada data yang belum pernah digunakan sebelumnya.

- a. Menentukan Training set dan Test set

```
test_size = data[data.Date.dt.year == 2022].shape[0]
test_size
```

- b. Plotting training set dan test set

```
plt.figure(figsize=(12, 6), dpi=100)
plt.rcParams['axes.facecolor'] = 'white'
plt.rc('axes', edgecolor='black')
plt.plot(data.Date[:-test_size], data.Close[:-test_size],
color='green', lw=1)
plt.plot(data.Date[-test_size:], data.Close[-test_size:],
color='blue', lw=1)
plt.title('Gold Price Training and Test Sets',
fontsize=15)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Price', fontsize=12)
plt.legend(['Training set', 'Test set'], loc='upper left',
prop={'size': 15})
plt.grid(color='black')
plt.show()
```



Di sini, kita bagi dataset ‘data’ menjadi dua bagian yaitu train dan test dataset. data dibawah tahun 2022 akan digunakan sebagai train dataset, sedangkan 2022 keatas sebagai test dataset.

4.3.5. Data Scaling

Skala nilai setiap atribut disesuaikan agar tidak ada atribut yang mendominasi karena rentang nilai yang besar. Salah satu teknik yang umum digunakan adalah Min-Max scaling, di mana nilai setiap fitur dinormalisasi ke rentang antara 0 dan 1, menjaga konsistensi dalam pemodelan.

- a. Scaling data pelatihan dan pengujian untuk model

```
scaler = MinMaxScaler()
scaler.fit(data.Close.values.reshape(-1, 1))
```

Membuat objek scaler dengan rentang fitur antara 0 dan 1. Juga, mengubah skala nilai pada data pelatihan, pengujian, dan pada fitur-fitur yang ada dalam ‘features’ menjadi rentang antara 0 dan 1.

- b. Restructure Data and Create Windowing

Sliding window = the use of prior time step to predict the next time step

Window width = the number of previous time steps

Dengan cara ini, kita dapat mengekspresikan data deret waktu sebagai supervised learning. Kita dapat melakukan ini dengan menggunakan langkah waktu sebelumnya sebagai variabel masukan dan langkah waktu berikutnya sebagai variabel keluaran. Lebar jendela yang digunakan adalah 60, yang berarti X_train dan X_test akan menjadi daftar bersarang yang berisi daftar 60 harga dalam waktu. y_train dan y_test juga merupakan daftar harga emas yang berisi harga emas hari berikutnya sesuai dengan masing-masing daftar di X_train dan X_test.

```
window_size = 60

train_data = data.Close[:-test_size]
train_data =
scaler.transform(train_data.values.reshape(-1, 1))

X_train = []
y_train = []

for i in range(window_size, len(train_data)):
    X_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])

test_data = data.Close[-test_size-60:]
test_data = scaler.transform(test_data.values.reshape(-1,
1))

X_test = []
y_test = []

for i in range(window_size, len(test_data)):
    X_test.append(test_data[i-60:i, 0])
    y_test.append(test_data[i, 0])
```

Membentuk sequences yang dibutuhkan oleh model LSTM. 'x_train' adalah sekumpulan data yang terdiri dari 60 baris sebelumnya, sedangkan 'y_train'

adalah data yang harus diprediksi setelah 60 baris. Sama seperti langkah sebelumnya untuk data pelatihan, kita membentuk sequences untuk data pengujian dan menyimpan nilai yang seharusnya diprediksi.

4.4. Modeling

4.4.1. Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) adalah model deep-learning yang dilakukan untuk memproses input yang berurutan ke output yang berurutan. Fitur RNN yang memiliki proses ‘mengingat’ yaitu output sebuah neuron yang kembali ke neuron tersebut membuat RNN berbeda dari algoritma Neural Network yang lain. Untuk *training* RNN pada model kami, kami membuat struktur RNN yang terdiri dari 3 layer SimpleRNN dengan activation function ReLU dengan Dense layer di akhir sebagai output. Dengan training epoch 200 dan batch_size 32 dan bantuan early_stopping, model kita berhenti *training* pada epoch 97 dan memiliki R^2 score dengan 0.884. RNN ini bisa dilakukan modifikasi untuk menciptakan algoritma yang lain, salah satunya yang juga lumayan terkenal adalah model LSTM (Long-Short Term Memory) yang merupakan model yang sangat baik untuk time-series forecasting.

```
from tensorflow.keras.callbacks import EarlyStopping
# Setting up an early stop
earlystop = EarlyStopping(monitor='val_loss',
min_delta=0.0001, patience=80, verbose=1, mode='min')
callbacks_list = [earlystop]

from tensorflow.keras.layers import Dense, SimpleRNN
from tensorflow.keras import optimizers

hl = [50, 45]

# Build and train the model
def fit_model(X_train, y_train, X_val, y_val, hl,):
    model_RNN = Sequential()
    model_RNN.add(SimpleRNN(hl[0], input_shape=(60, 1),
```

```

return_sequences=True, activation='relu'))
    for units in hl[1:]:
        model_RNN.add(SimpleRNN(units, activation='relu',
return_sequences=True))
    model_RNN.add(SimpleRNN(hl[-1], activation='relu'))
    model_RNN.add(Dense(1))

model_RNN.compile(optimizer=optimizers.Adam(learning_rate=1e-
3), loss='mean_squared_error')

    history = model_RNN.fit(X_train, y_train, epochs=200,
batch_size=32, validation_data=(X_val, y_val), verbose=1,
shuffle=False, callbacks=callbacks_list)
    model_RNN.reset_states()
    return model_RNN, history.history['loss'],
history.history['val_loss']

val_size = int(0.1 * len(X_train))
X_val, y_val = X_train[-val_size:], y_train[-val_size:]
X_train, y_train = X_train[:-val_size], y_train[:-val_size]
model_RNN, train_error, val_error = fit_model(X_train,
y_train, X_val, y_val, hl)

```

4.4.2. Long Short Term Memory (LSTM)

LSTM merupakan model improvisasi dari RNN. Yang dimana melakukan modifikasi terhadap RNN dengan menambahkan suatu memory cell yang dapat digunakan untuk menyimpan informasi untuk jangka waktu yang lama. LSTM memiliki algoritma yang mengingat dan melupakan data dengan efisien sehingga memiliki performa yang bagus pada memprediksikan harga pasar. Algoritma ini melakukan analisis pada harga pasar dari waktu ke waktu untuk mencari pola pada data tersebut. Pada dasarnya model LSTM sangat cocok untuk mengatasi data yang bersifat time-series atau sequential data. LSTM juga dapat mengatur

memori yang akan masuk pada setiap memory cells dan juga gates unit dan dari sana dapat menggabungkan informasi data sekarang dan juga history data, sehingga sangat efisien untuk menganalisis informasi jangka panjang. Maka secara kesimpulan LSTM sangat cocok untuk model prediksi pasar saham.

```
def define_model():
    input1 = Input(shape=(window_size,1))
    x = LSTM(units = 64, return_sequences=True)(input1)
    x = Dropout(0.2)(x)
    x = LSTM(units = 64, return_sequences=True)(x)
    x = Dropout(0.2)(x)
    x = LSTM(units = 64)(x)
    x = Dropout(0.2)(x)
    x = Dense(32, activation='relu')(x)
    dnn_output = Dense(1)(x)

    model = Model(inputs=input1, outputs=[dnn_output])
    model.compile(loss='mean_squared_error', optimizer=
'Nadam')
    model.summary()

    return model

model = define_model()
history = model.fit(X_train, y_train, epochs=150,
batch_size=32,
                    validation_split=0.1, verbose=1)
```

Model LSTM diinisialisasi sebagai objek dari kelas Sequential untuk membuat setiap layer dalam model terhubung secara berurutan, sehingga output dari layer sebelumnya menjadi input dari layer berikutnya.

Lalu, kita tambahkan layer LSTM yang pertama ke dalam model. Dalam layer ini, kita tentukan jumlah unit neuron, fungsi aktivasi ReLU untuk memperkenalkan

non-linearitas, dan juga apakah layer tersebut harus mengembalikan urutan sekuen karena kita akan memiliki beberapa layer LSTM lagi kedepannya.

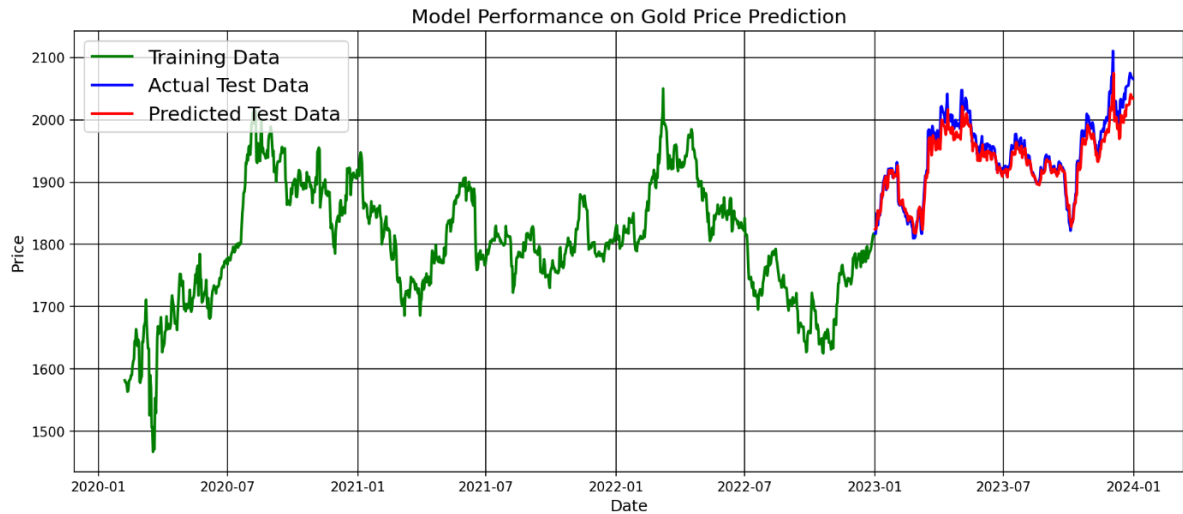
Setelah menambahkan layer LSTM pertama, kita menambahkan layer Dropout dengan nilai 0,2. Dropout digunakan untuk mengurangi overfitting dengan secara acak menonaktifkan 20% dari unit di layer LSTM sebelumnya selama proses pelatihan.

Model LSTM terus ditambahkan layer dan Dropout secara berurutan dengan jumlah unit (64, 64, 64, dan 32) untuk memungkinkan model untuk menangkap pola yang lebih kompleks dalam data.

Terakhir, kita tambahkan output layer dengan satu unit karena kita memprediksi nilai tunggal (harga emas).

```
model = define_model()  
history = model.fit(X_train, y_train, epochs=150,  
                    batch_size=32,  
                    validation_split=0.1, verbose=1)
```

Optimizer yang kami Gunakan adalah ‘Nadam’. optimizer ‘Nadam’ merupakan singkatan dari Nesterov-accelerated Adaptive Moment Estimation. Yang merupakan optimizer yang hampir sama dengan “adam”. Adam sendiri merupakan metode yang menghitung tingkat pembelajaran yang adaptif dari setiap parameter dari setiap gradient history. Tipe dari optimizer yang digunakan dapat berpengaruh kepada kecepatan algoritma. Pada dasarnya adam optimizer merupakan gabungan dari 2 optimizer, yaitu ADAGard dan RMSprop. Nadam pada dasarnya merupakan optimizer yang menggabungkan 2 optimizer yaitu NAG dan adam. Hal ini digunakan untuk menutupi kelemahan dari adam. Dan untuk mencari loss, kita menggunakan “mean_squared_error”. Metode mean squared error menghasilkan nilai loss yang positif dan pasti tidak negatif karena bersifat kuadratik. batch_size yang kami gunakan 32 dengan pertimbangan karena data yang kami gunakan tidak begitu banyak (hanya ribuan).



Hasil Prediksi dari Model LSTM (perbandingan data original dan prediksi)

Evaluation Accuracy

Test Loss: 0.0008073383942246437

Test MAPE: 0.02772972367833768

Test Accuracy: 0.9722702763216623

Root Mean Squared Error: 0.028413702523788147

R-Squared Error: 0.9112260501512194

4.4.3. Gated Recurrent Unit (GRU)

Model GRU (Gated Recurrent Unit) merupakan salah satu jenis deep learning yang termasuk kelompok dari RNN. Pada dasarnya GRU dirancang untuk mengatasi kelemahan dari model RNN tradisional. Keterbatasan yang dimiliki oleh RNN adalah long-term dependencies yang akhirnya diselesaikan atau dipecahkan oleh model GRU. Ada juga model yang termasuk kelompok dari RNN yaitu LSTM. Perbedaan yang paling mendasar dari kedua model ialah GRU memiliki struktur dan pola yang lebih sederhana dibandingkan LSTM. Berikut adalah beberapa alasan mengapa kelompok kami mencoba menggunakan model GRU pada konteks project kami yaitu stock market prediction :

a) Data Saham Bersifat Sequential : Karena Data saham bersifat time series dimana data sekarang memiliki keterkaitan dengan data historis maka, model GRU sangat cocok dimana GRU bersifat recurrent dan menangkap pola secara temporal yang ada dalam data.

b) Long-term Dependencies : Dimana model GRU dapat menangkap pola jangka panjang seperti harga saham, dan memiliki feature update dan reset gate yang membantu memisahkan antara data historis yang relevan dan tidak relevan.

c) Memiliki kecepatan komputasi : Dibanding LSTM, model GRU bisa dikatakan lebih efektif dari segi kecepatan karena GRU memiliki arsitektur yang lebih sederhana jika dibandingkan oleh model LSTM, namun pada project kami kali ini dapat disimpulkan bahwa LSTM menghasilkan akurasi yang lebih bagus walaupun dengan durasi training yang lebih lama.

Berikut adalah code dari model training GRU yang kelompok kami buat sebagai bahan perbandingan dari model RNN dan LSTM.

```
from keras.layers import Dense, Dropout, GRU, Bidirectional
from keras.optimizers import SGD
from keras.models import Sequential

# The GRU architecture
regressorGRU = Sequential()
# First GRU layer with Dropout regularisation
regressorGRU.add(GRU(units=50, return_sequences=True,
input_shape=(X_train.shape[1],1), activation='tanh'))
regressorGRU.add(Dropout(0.2))
# Second GRU layer
regressorGRU.add(GRU(units=50, return_sequences=True,
input_shape=(X_train.shape[1],1), activation='tanh'))
regressorGRU.add(Dropout(0.2))
# Third GRU layer
regressorGRU.add(GRU(units=50, return_sequences=True,
input_shape=(X_train.shape[1],1), activation='tanh'))
regressorGRU.add(Dropout(0.2))
# Fourth GRU layer
regressorGRU.add(GRU(units=50, activation='tanh'))
```

```
regressorGRU.add(Dropout(0.2))
# The output layer
regressorGRU.add(Dense(units=1))
# Compiling the RNN
regressorGRU.compile(optimizer=SGD(learning_rate=0.01,
momentum=0.9, nesterov=False), loss='mean_squared_error')
# Fitting to the training set
regressorGRU.fit(X_train, y_train, epochs=50, batch_size=150)

GRU_predicted_stock_price = regressorGRU.predict(X_test)
```

Model GRU ini kami buat dengan menginisialisasi model sebagai objek Sequential terlebih dahulu untuk menyusun layer secara berurutan.

Setelah itu, kami mulai dengan menambahkan layer GRU pertama dengan 50 unit neuron, aktivasi tanh, return_sequences=True, dan input_shape sesuai data pelatihan. Layer ini diikuti dengan Dropout 0,2 untuk mengurangi overfitting.

Langkah ini diulang untuk layer GRU kedua dan ketiga, masing-masing dengan 50 unit neuron dan return_sequences=True, serta Dropout 0,2 setelah setiap layer.

Layer GRU keempat juga memiliki 50 unit neuron dan aktivasi tanh, namun tanpa return_sequences, karena ini adalah layer terakhir sebelum layer output.

Selanjutnya, kita tambahkan layer Dense dengan satu neuron sebagai output layer yang menghasilkan prediksi akhir.

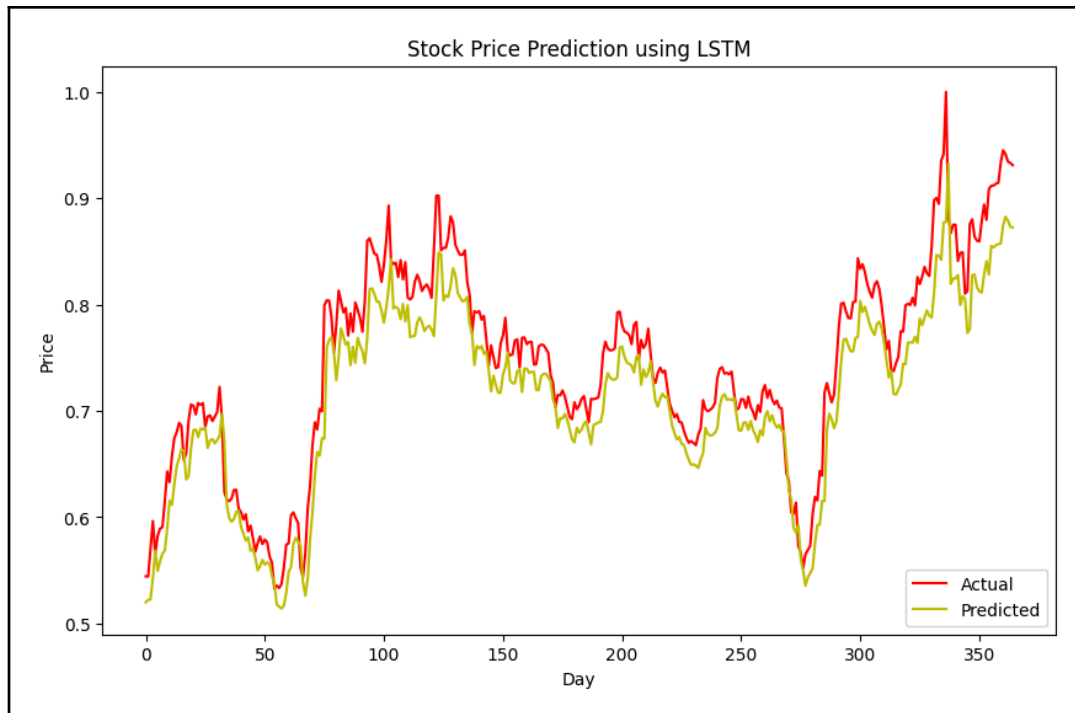
Model dikompilasi menggunakan optimizer SGD dengan learning rate 0.01, momentum 0.9, dan loss function mean_squared_error.

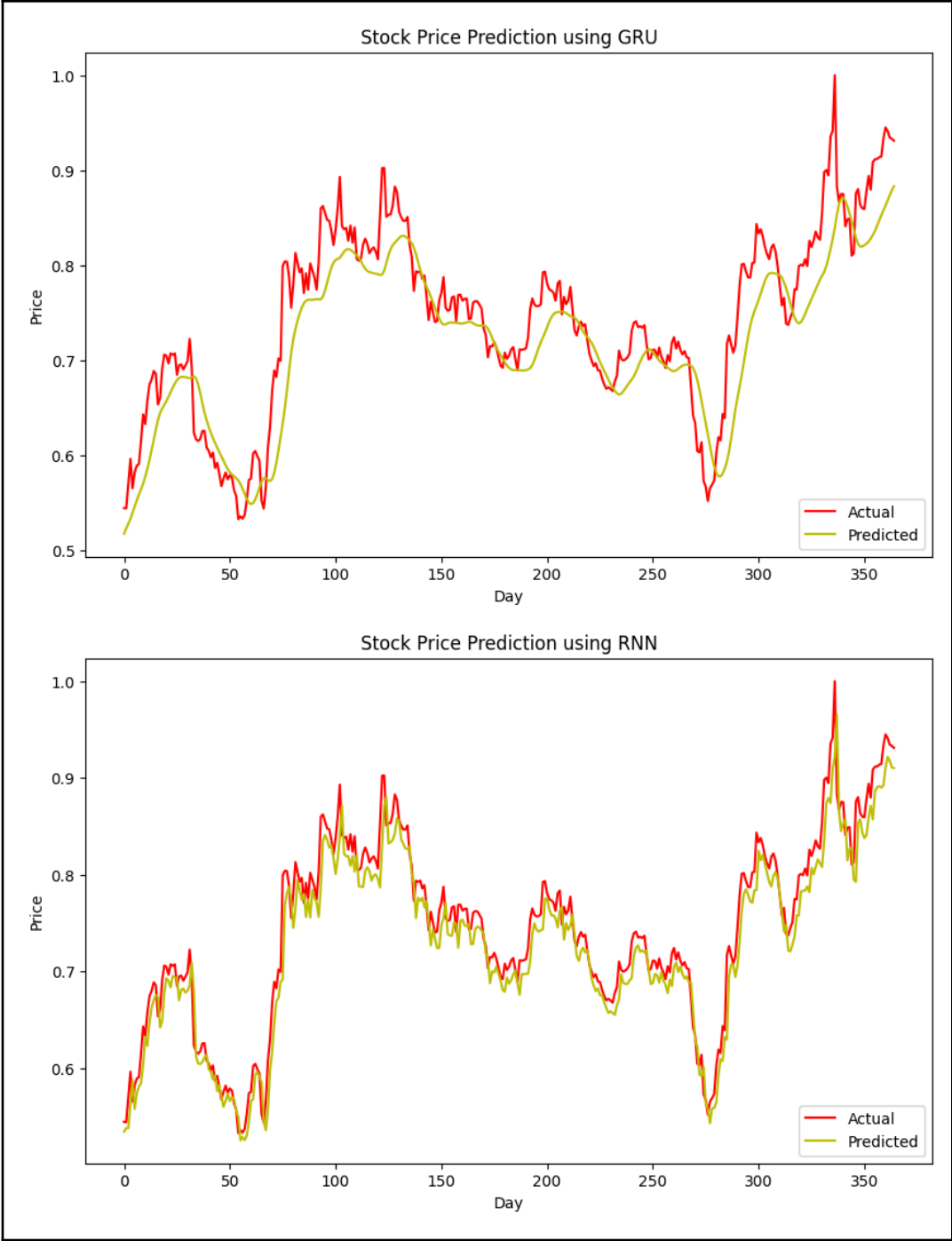
Akhirnya, model dilatih menggunakan data pelatihan (X_train, y_train) selama 50 epoch dengan batch size 150 untuk menghasilkan prediksi harga saham yang akurat.

4.5. Evaluation

```
metrics_dict = {'Model': [], 'MAE': [], 'MSE': [], 'MAPE':  
                [], 'RMSE': [], 'r2': []}  
models = [model, regressorGRU, model_RNN]  
model_names = ['LSTM', 'GRU', 'RNN']  
  
def evaluate_model(model, X_test, y_test):  
    y_pred = model.predict(X_test)  
  
    rmse = mean_squared_error(y_test, y_pred, squared=False)  
    mse = mean_squared_error(y_test, y_pred)  
    mae = mean_absolute_error(y_test, y_pred)  
    r_squared = r2_score(y_test, y_pred)  
    mape = mean_absolute_percentage_error(y_test, y_pred)  
  
    return mse, rmse, mae, r_squared, mape, y_test, y_pred  
  
def plot_data(Y_test, Y_hat, model_name):  
    plt.figure(figsize=(10, 6))  
    plt.plot(Y_test, color='r')  
    plt.plot(Y_hat, color='y')  
    plt.xlabel('Day')  
    plt.ylabel('Price')  
    plt.title(f"Stock Price Prediction using {model_name}")  
    plt.legend(['Actual', 'Predicted'], loc='lower right')  
    plt.show()  
  
for model, name in zip(models, model_names):  
    mse, rmse, mae, r_squared, mape, true, predicted =  
evaluate_model(model, X_test, y_test)  
  
    metrics_dict['Model'].append(name)  
    metrics_dict['MAE'].append(mae)  
    metrics_dict['MSE'].append(mse)  
    metrics_dict['MAPE'].append(mape)  
    metrics_dict['RMSE'].append(rmse)  
    metrics_dict['r2'].append(r_squared)  
  
    plot_data(true, predicted, name)
```

```
metrics_df = pd.DataFrame(metrics_dict)
metrics_df.style.format({'MAE': '{:.3f}', 'MSE': '{:.3f}',
'MAPE': '{:.3f}', 'RMSE': '{:.3f}', 'r2':
'{:.3f}'}).highlight_min(color='green', subset=['MAE', 'MSE',
'MAPE', 'RMSE']).highlight_max(color='green', subset=['r2'])
```





	Model	MAE	MSE	MAPE	RMSE	r2
0	LSTM	0.019	0.001	0.025	0.027	0.922
1	GRU	0.037	0.002	0.050	0.048	0.751
2	RNN	0.028	0.001	0.037	0.033	0.884

Berdasarkan hasil evaluasi diatas, dapat disimpulkan bahwa model LSTM menunjukkan performa yang paling baik di antara ketiga model dengan nilai MAE, MSE, MAPE, dan RMSE yang paling rendah, serta nilai R^2 Error yang paling tinggi. Hal ini menunjukkan bahwa model LSTM mampu menangkap pola dalam data dengan paling baik dan memberikan prediksi yang akurat dengan kesalahan minimal.

RNN juga menunjukkan performa yang baik namun tidak seakurat LSTM. Sementara itu, GRU memiliki performa yang paling rendah diantara ketiga model.

Link Colab:

<https://colab.research.google.com/drive/1vsj4qzHQMM6MZBQ0twvEtqy4YWGDJbOK#scrollTo=gRKEPrGby1Ig&uniqifier=1>

link drive untuk Video

<https://drive.google.com/drive/folders/186tme6r36T63wll4Zf7mDa2Fj9kiN60v>

5. Deploy Aplikasi berbasis Web (Framework = streamlit)

5.1 Import Libraries

```
import numpy as np
```

```
import pandas as pd
import yfinance as yf
from keras.models import load_model
import streamlit as st
from sklearn.preprocessing import MinMaxScaler
from datetime import datetime, timedelta
import plotly.graph_objects as go
import plotly.express as px
import matplotlib.pyplot as plt
```

5.2 Load Model Training

```
model = load_model('/Stock_Market_Prediction_ML
(4)/Gold-price-prediction.keras')
```

5.3 Input Stock Code and Download Data

```
st.set_page_config(layout="wide")
st.title('Stock Market Predictor')
# Section: Get stock data
st.sidebar.header('Select Stock Symbol')
stock_symbol = st.sidebar.text_input('Enter Stock
Symbol', 'XAUT-USD')

stock = stock_symbol
start = '2012-01-01'
end = datetime.today().strftime('%Y-%m-%d')
data = yf.download(stock_symbol, start, end)
data.reset_index(inplace=True)
```

5.4 Display Stock Data

```
st.header(f'{stock} ({stock_symbol}) Stock Price &
Analysis')
```

Stock Market Predictor

XAUT-USD (XAUT-USD) Stock Price & Analysis

5.5 Current Price Section

```
current_price = data['Close'].iloc[-1]
previous_close = data['Close'].iloc[-2]
change = current_price - previous_close
percentage_change = (change / previous_close) * 100

st.metric(label="Current Price",
value=f"${current_price:.2f}", delta=f"{change:.2f}
({percentage_change:.2f}%)")
```

Current Price

\$2295.20

↓ -0.44 (-0.02%)

5.6 Display plot

5.6.1 Advanced Chart

```
st.subheader("Advanced Chart")

tab_names = ["1d", "1w", "1m", "3m", "6m", "YTD",
"1y", "3y", "5y", "10y"]
tabs = st.tabs(tab_names)
```

```

for i, tab in enumerate(tabs):
    with tab:
        if tab_names[i] == "1d":
            # Use minute-level data for one day
            start_day = (datetime.today() -
timedelta(days=1)).strftime('%Y-%m-%d')
            end_day =
datetime.now().strftime('%Y-%m-%d')
            day_data = yf.download(stock_symbol,
start=start_day, end=end_day, interval='1m')
            day_data.reset_index(inplace=True)

            if not day_data.empty:
                fig = go.Figure()

                # Add price line with conditional
color
                fig.add_trace(go.Scatter(
                    x=day_data.index,
y=day_data['Close'], mode='lines',
                    line=dict(color='green' if
day_data['Close'].iloc[-1] >
day_data['Open'].iloc[-1] else 'red'),
                    name='Price',
                    hoverinfo='x+y'
                ))

                # Add volume bars
                colors = ['green' if row['Close'] >
row['Open'] else 'red' for index, row in
day_data.iterrows()]
                fig.add_trace(go.Bar(
                    x=day_data.index,
y=day_data['Volume'], name='Volume',
                    marker_color=colors,

```

```

yaxis='y2', opacity=0.3
    ))

    fig.update_layout(
        title='1 Day Price',
        yaxis=dict(title='Price'),
        yaxis2=dict(title='Volume',
overlying='y', side='right', showgrid=False),
        xaxis=dict(title='Time',
tickformat='%H:%M'),
        legend=dict(x=0, y=1.0),
        margin=dict(l=0, r=0, t=40,
b=0),

        hovermode='x',
        dragmode=False,
    )

    st.plotly_chart(fig,
use_container_width=True)
    else:
        st.write("No data available for the
selected period.")
    else:
        date_interval_map = {
            "1w": 7,
            "1m": 30,
            "3m": 90,
            "6m": 180,
            "YTD": 365,
            "1y": 365,
            "3y": 3 * 365,
            "5y": 5 * 365,
            "10y": 10 * 365,
        }

        interval =
date_interval_map.get(tab_names[i])

```



```

data_interval = data.tail(interval)

if not data_interval.empty:
    fig = go.Figure()

    # Add price line with conditional
color
fig.add_trace(go.Scatter(
    x=data_interval['Date'],
y=data_interval['Close'], mode='lines',
    line=dict(color='green' if
data_interval['Close'].iloc[-1] >
data_interval['Open'].iloc[-1] else 'red'),
    name='Price',
    hoverinfo='x+y'
))

    # Add volume bars
    colors = ['green' if row['Close'] >
row['Open'] else 'red' for index, row in
data_interval.iterrows()]
    fig.add_trace(go.Bar(
        x=data_interval['Date'],
y=data_interval['Volume'], name='Volume',
        marker_color=colors,
yaxis='y2', opacity=0.3
    ))

    fig.update_layout(
        title=f'{tab_names[i]} Price',
        yaxis=dict(title='Price'),
        yaxis2=dict(title='Volume',
overlying='y', side='right', showgrid=False),
        xaxis=dict(title='Date'),
        legend=dict(x=0, y=1.0),

```

```

margin=dict(l=0, r=0, t=40,
b=0),

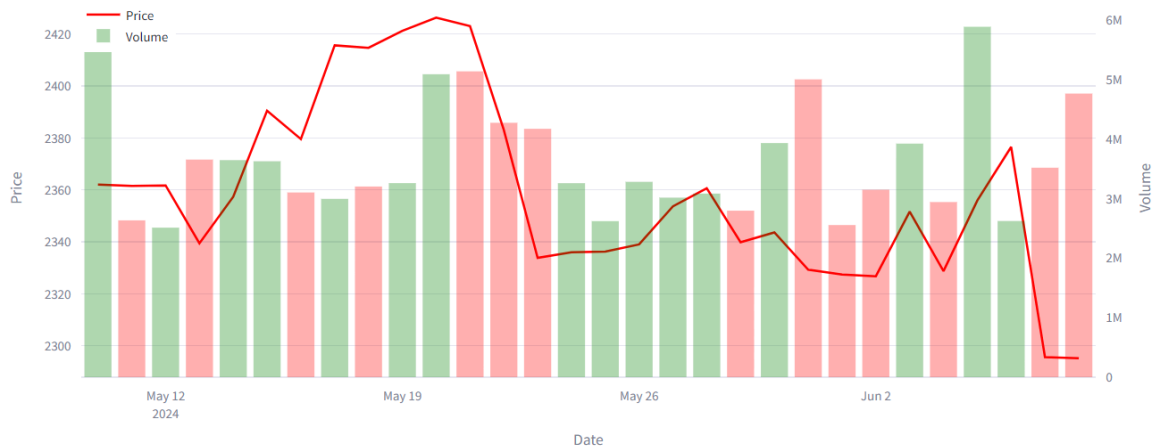
        hovermode='x',
        dragmode=False,
    )
    st.plotly_chart(fig,
use_container_width=True)
    else:
        st.write("No data available for the
selected period.")

```

Advanced Chart

1d 1w 1m 3m 6m YTD 1y 3y 5y 10y

1m Price



5.6.2 Plot Moving Average

Moving Averages

MA50 MA50 vs MA100 MA100 vs MA200

Price vs MA50



5.6.3 Plot the Predicted Data

Predicted XAUT-USD Prices for the Next 30 Days



5.6.4 Plot the Predicted Data (Zoom)



5.7 Display predicted prices in a scrollable table

```
days = [f'Day {i+1}' for i in
range(len(predicted_prices))]
predicted_data = {'Day': days, 'Predicted Price (USD)':
[f'${price:.2f}' for price in predicted_prices]}
predicted_df = pd.DataFrame(predicted_data)
st.write("### Predicted Prices:")
st.dataframe(predicted_df)
```

5.8 Overall Display App

×

Select Stock Symbol

Enter Stock Symbol

XAUT-USD

Deploy

Stock Market Predictor

XAUT-USD (XAUT-USD) Stock Price & Analysis

Current Price

\$2299.35

↑ 4.16 (0.18%)

	Date	Open	High	Low	Close	Adj Close	Volume
21	2020-02-28 00:00:00	1,641.4972	1,653.2557	1,571.7786	1,579.6929	1,579.6929	1,151,262
22	2020-02-29 00:00:00	1,579.1823	1,587.3325	1,577.5613	1,577.8896	1,577.8896	23,717
23	2020-03-01 00:00:00	1,577.424	1,599.1135	1,566.1229	1,585.6704	1,585.6704	283,093
24	2020-03-02 00:00:00	1,585.7954	1,612.4208	1,582.2263	1,588.1906	1,588.1906	131,475
25	2020-03-03 00:00:00	1,588.0465	1,645.89	1,588.0465	1,643.8767	1,643.8767	154,581
26	2020-03-04 00:00:00	1,646.1113	1,650.2825	1,627.3605	1,642.8939	1,642.8939	95,236
27	2020-03-05 00:00:00	1,643.0028	1,672.8112	1,478.6609	1,667.5936	1,667.5936	327,861
28	2020-03-06 00:00:00	1,667.6982	1,689.7241	1,651.9243	1,671.6957	1,671.6957	1,201,058

5.9 App Demo

Link:

https://drive.google.com/file/d/1FUmlWz8DrGh4JV_XoJeP934IPa38wiZs/view?usp=drive_link