Skill Academy Project

— Final Project : Stock Price Forecasting

A Comprehensive Analysis for UNTR Stock Predictions with LSTM Neural Networks

— Membangun model yang dapat melakukan *forecasting* harga saham untuk beberapa hari ke depan,

Disusun oleh:
Andika Mandala Putra

1 Deskripsi Proyek

Swing trader membutuhkan tools tambahan

Saham UNTR relative stabil

UNTR berfundamental baik, minim resiko

Kesempatan volatilitas hingga 4%/hari

UNITED TRACTORS

2 Tujuan

- Membangun model yang dapat memprediksi harga Open, High, Low, dan Close,
- Membangun model dengan skor MAPE* di bawah tingkat volatilitas.

3 Alur Proyek

Alur Proyek Stock Forecasting

Ikhtisar Data
- Pemahaman data

Mempersiapkan Data
- Reduksi data
- Pengayaan data

Analisa Data Eksploratif
- Visualisasi data
- Statistika analisis

Membangun Model ML
- *Feature engineering*
- *Model training*
- *Model testing*

Membuat *User Interface*

*MAPE (*Mean Absolute Percentage Error*)

1 Ikhtisar Data

Date	Open	High	Low	Close	Adj Close	Volume
2016-01-19	16000	16100	15800	16075	9445.11	2482800
2016-01-20	16000	16100	15875	16025	9415.73	4175400
2016-01-21	15925	16125	15625	15825	9298.22	4104900
2016-01-22	16050	16325	15950	16325	9592	2270800
2016-01-25	16650	17050	16525	16925	9944.54	3907800
2024-01-12	23350	23925	23325	23900	23900	4858500
2024-01-15	23900	24750	23900	24725	24725	6691700
2024-01-16	24750	24800	24375	24600	24600	4671400
2024-01-17	25000	25000	24200	24225	24225	5386800
2024-01-18	24225	24550	24050	24100	24100	3894900

sumber: Yahoo Finance

Keterangan Kolom

- ‘Open’ : Harga pembukaan.
- ‘High’ : Harga tertinggi.
- ‘Low’ : Harga terendah.
- ‘Close’ : Harga penutupan.
- ‘Adj Close’ : Harga penutupan yang disesuaikan.
- ‘Volume’ : Volume transaksi

Informasi Data

- **Ukuran** : (1995, 6)
- **Interval** : 1D
- **Periode** : 19/Jan/2016 - 18/Jan/2024

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1995 entries, 2016-01-19 to 2024-01-18
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Open       1994 non-null   float64
 1   High       1994 non-null   float64
 2   Low        1994 non-null   float64
 3   Close      1994 non-null   float64
 4   Adj Close  1994 non-null   float64
 5   Volume     1994 non-null   float64
dtypes: float64(6)
memory usage: 109.1 KB
```

Temuan

- Tidak ada *missing value*.
- Tidak ada kesalahan tipe data.
- Tidak terdapat duplikat.
- Ada beberapa data hari yang hilang karena hari libur bursa. Akan tetapi, ini tidak menjadi masalah,

2 Pengayaan Data

```python
def make_features(data:pd.DataFrame, lag:list[int]=None):
    df = data.copy()

    # lag features
    if lag: # will be executed if the variable has a value
        for i in lag:
            df[f"Open(t-{i})"] = df['Open'].shift(i)
            df[f"High(t-{i})"] = df['High'].shift(i)
            df[f"Low(t-{i})"] = df['Low'].shift(i)
            df[f"Close(t-{i})"] = df['Close'].shift(i)

    return df.dropna()
```

Membuat *lag-features* untuk (t-1) hingga (t-7) untuk harga Open, High, Low, dan Close.

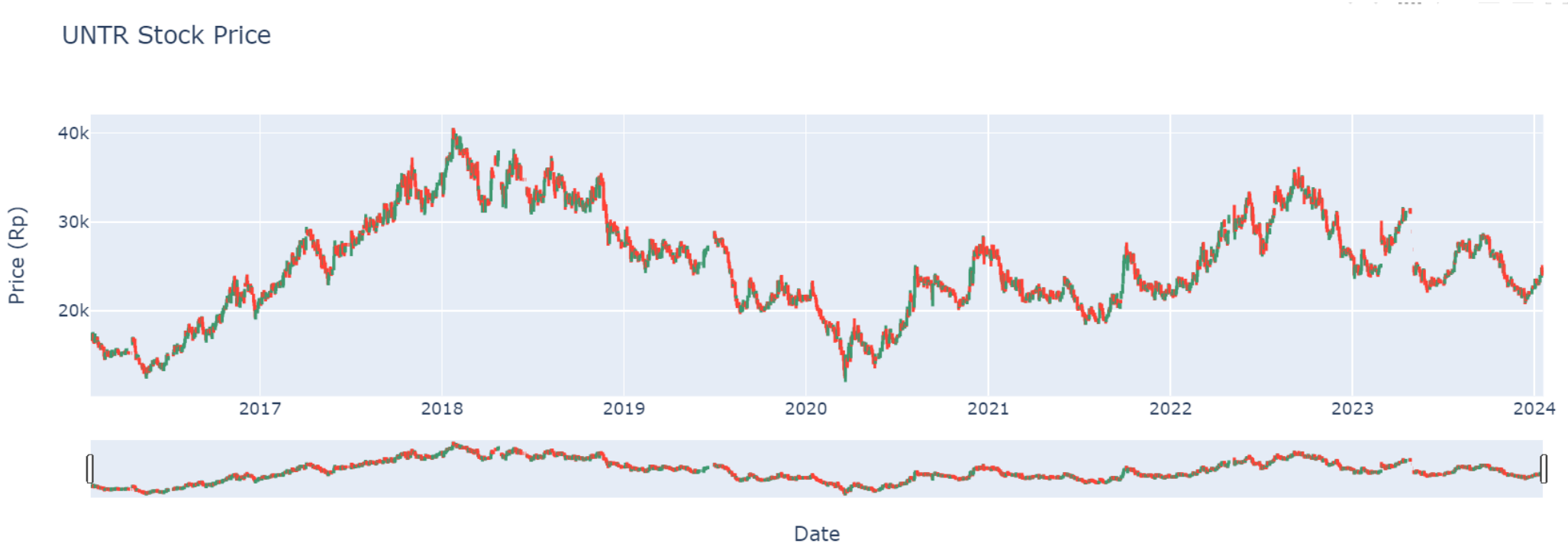
3 Reduksi Data

```python
def remove_unimportant_features(data:pd.DataFrame):

    return data.drop(columns=["Volume", "Adj Close"])
```

Membuang fitur volume transaksi dan *adjusted close* karena tidak dibutuhkan pada proyek ini.

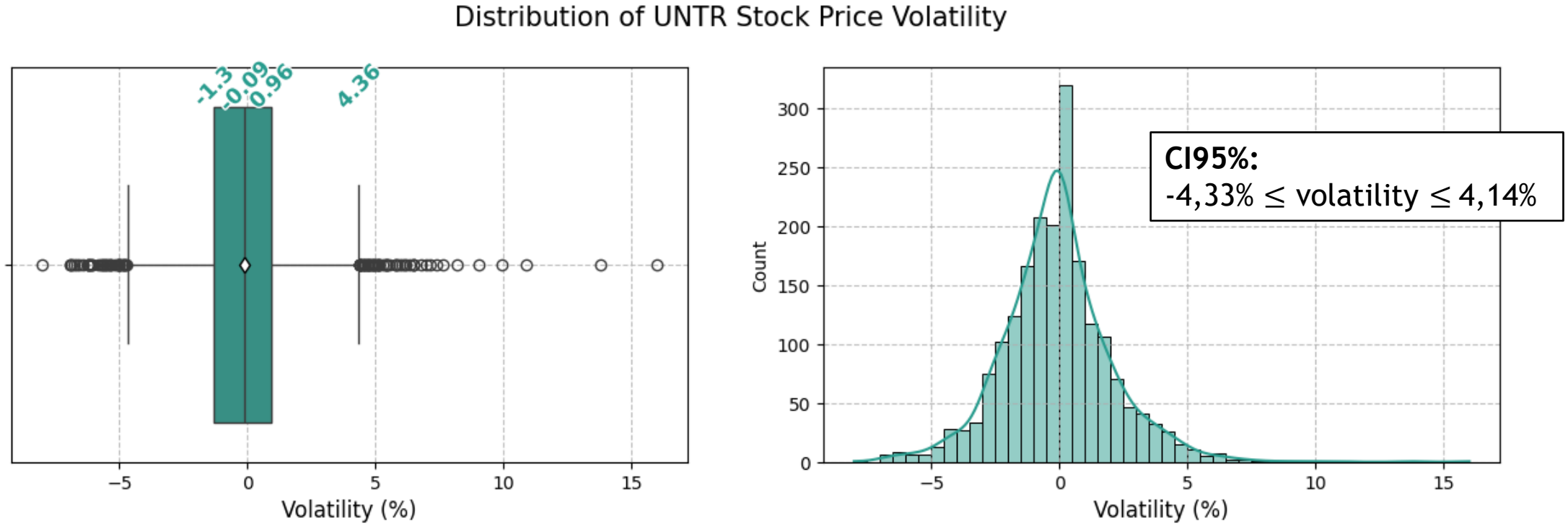
1 Bagaimana tren saham UNTR ?



Insights

- Saham UNTR tergolong saham siklikal. Terlihat dengan jelas siklus 5 tahunan pada saham ini.
- Secara tren, harga saham ini tergolong stagnan. Akan tetapi, pada momentum yang tepat, kesempatan saham siklikal ini sangat bagus untuk ditunggu.

2 Bagaimana Tingkat volatilitas saham UNTR?



Insights

- Rata-rata volatilitas harian berada pada angka -0,09% dengan standard deviasi 2,16%.
- Secara statistik, volatilitas harga saham UNTR berkisar pada -4% hingga 4% dengan *confidence interval* sebesar 95%. Nilai volatilitas ini yang akan menjadi acuan skor MAPE untuk model yang akan dilatih.

1 Features Engineering

```
# memisahkan targets dan features
targets_cols = ['Open', 'High', 'Low', 'Close']
features_cols = new_df.columns.drop(targets_cols).to_list()

x = new_df[features_cols]
y = new_df[targets_cols]
```

Memisahkan *targets* dan *features*.

```
# melakukan data split
x_train, x_val, y_train, y_val = train_test_split(x, y,
test_size=0.30, shuffle=False)
x_val, x_test, y_val, y_test = train_test_split(x_val,
y_val, test_size=0.50, shuffle=False)
```

Membagi data menjadi *training*, *validation*, dan *testing set* dengan proporsi 70:15:15.

```
# melakukan scaling fitur
scaler = StandardScaler()
scaler.fit(x_train)

x_train = scaler.transform(x_train)
x_val = scaler.transform(x_val)
x_test = scaler.transform(x_test)
```

Melakukan *features scaling*.

```
# melakukan transformasi dimensi matriks
x_train = x_train.reshape((-1, x_train.shape[1], 1))
x_val = x_val.reshape((-1, x_val.shape[1], 1))
x_test = x_test.reshape((-1, x_test.shape[1], 1))

y_train = y_train.values.reshape((-1, y_train.shape[1]))
y_val = y_val.values.reshape((-1, y_val.shape[1]))
y_test = y_test.values.reshape((-1, y_test.shape[1]))
```

Melakukan penyesuaian dimensi fitur.

```
# membuat dataset menjadi objek Dataset
class TimeSeriesDataset(Dataset):
    def __init__(self, X, y):
        self.X = torch.tensor(X, dtype=torch.float)
        self.y = torch.tensor(y, dtype=torch.float)

    def __len__(self):
        return len(self.X)

    def __getitem__(self, i):
        return self.X[i], self.y[i]

train_dataset = TimeSeriesDataset(x_train, y_train)
val_dataset = TimeSeriesDataset(x_val, y_val)
test_dataset = TimeSeriesDataset(x_test, y_test)
```

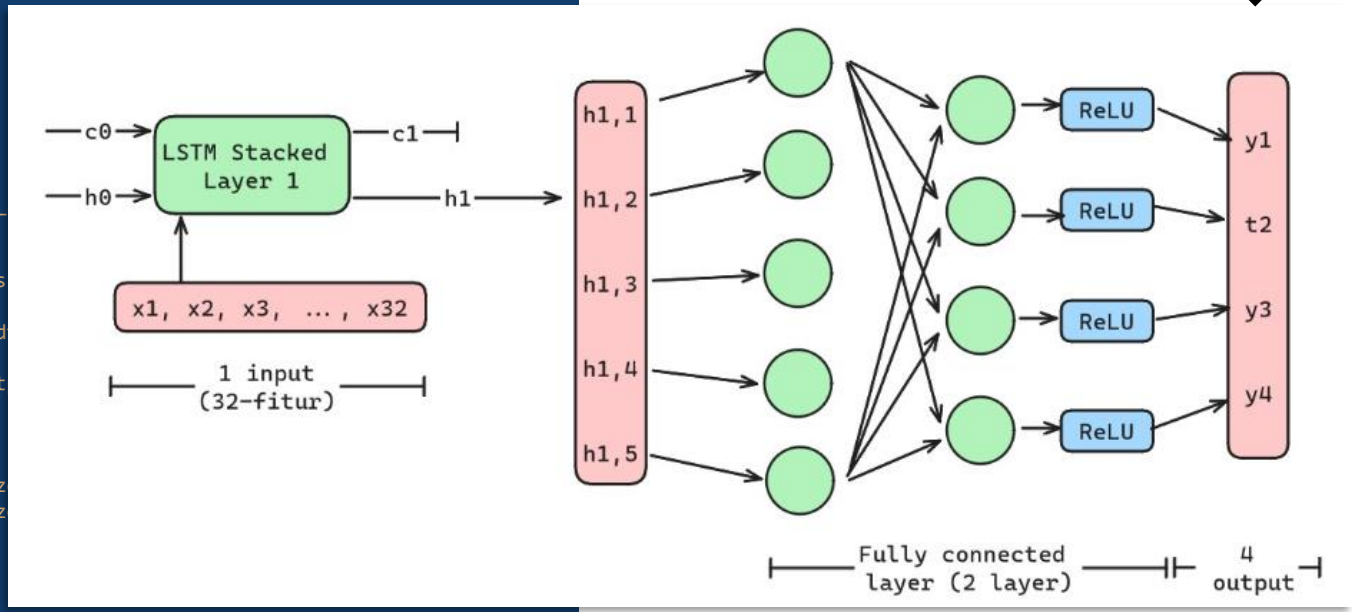
Menyimpan dataset dalam objek *Dataset*.

2 Membangun Model

```python
# membuat arsitektur model
class LSTMmodel(nn.Module):
    def __init__(self, n_input, num_stacked_layers, hidden_size, n_output,hidd_layers:tuple[int]=None, dtype=torch.float):
        super().__init__()
        self.num_stacked_layers = num_stacked_layers
        self.hidden_size = hidden_size
        # membuat LSTM cell
        self.lstm = nn.LSTM(input_size=n_input,
                            hidden_size=hidden_size,
                            num_layers=num_stacked_layers,
                            dtype=dtype,
                            batch_first=True)
        # membuat fully connected layers
        if hidd_layers:
            # input layer
            self.fc = torch.nn.ModuleList([torch.nn.Linear(hidden_size, hidd_
            # hidden layers
            for i in range(1, len(hidd_layers)):
                self.fc.extend([torch.nn.Linear(hidd_layers[i-1], hidd_layers
            # output layer
            self.fc.append(torch.nn.Linear(hidd_layers[-1], n_output, dtype=d
        else:
            self.fc = torch.nn.ModuleList([torch.nn.Linear(hidden_size, n_out

    def forward(self, x):
        batch_size = x.shape[0]
        h0 = torch.zeros(self.num_stacked_layers, batch_size, self.hidden_siz
        c0 = torch.zeros(self.num_stacked_layers, batch_size, self.hidden_siz
        h1, c1 = self.lstm(x, (h0, c0))
        x = h1[:,-1,:]
        for layer in self.fc:
            x = layer(x)
        return x
```

Ilustrasi arsitektur model.



3 Melatih Model

```
# random seed
torch.manual_seed(101010)


# melatih model
lstm = LSTMmodel(n_input=1,
                 num_stacked_layers=1,
                 hidden_size=15,
                 hidd_layers=(),
                 n_output=4)

learning_rate = 0.001
num_epochs = 1000
loss_function = nn.MSELoss()
```

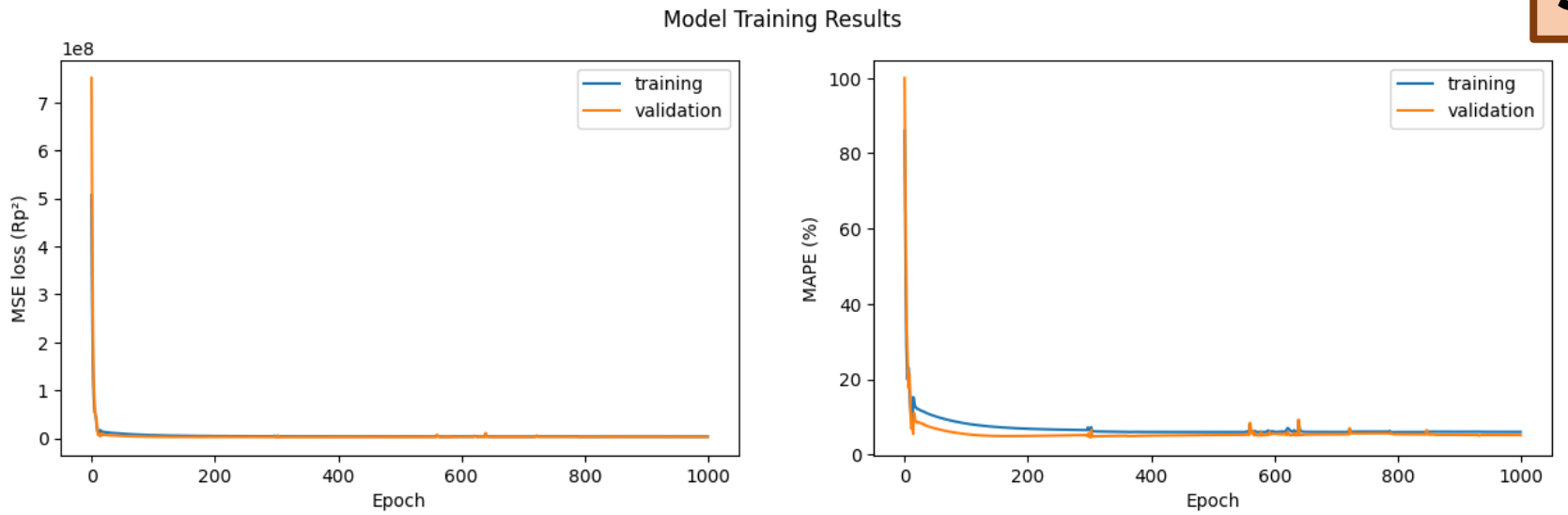
```
========== Epoch: 1 ==========
{'avg. loss (validation)': 751400643.3684, 'avg. mape (validation)': 100.00%}
{'avg. loss (training)': 506917462.6207, 'avg. mape (training)': 85.98%}

========== Epoch: 2 ==========
{'avg. loss (validation)': 443259159.5789, 'avg. mape (validation)': 75.36%}
{'avg. loss (training)': 298387169.5632, 'avg. mape (training)': 63.49%}
```

```
========== Epoch: 1000 ==========
{'avg. loss (validation)': 2818646.7270, 'avg. mape (validation)': 5.15%}
{'avg. loss (training)': 3401117.5639, 'avg. mape (training)': 5.98%}
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

(LSTMmodel(
   (lstm): LSTM(1, 15, batch_first=True)
   (fc): ModuleList(
     (0): Linear(in_features=15, out_features=4, bias=True)
   )
 ),
 'lr:0.001')
```

MAPE: 5,15%



4 Model Testing

```
with torch.inference_mode():
    dtype = torch.float
    x_tensor = torch.tensor(x_test,
dtype=dtype)
    y_tensor = torch.tensor(y_test,
dtype=dtype)
    pred = lstm(x_tensor)
    mape = mape_score(pred, y_tensor)
    loss = loss_function(pred,
y_tensor)

print(f"MAPE score : {mape:.2f}%")
print(f"MSE loss   : {loss}")
```

MAPE : 5,15%

Validation set

```
MAPE score : 7.14%
MSE loss   : 4504931.0
```

Testing set

1 Query Historical Data dari Yahoo Finance

```python
# function untuk query historical data harga saham UNTR

def query_historical_data(n_days):
```

2 Data Preprocessing

```python
# membuat function untuk data preprocessing
def data_preprocessing(data, n=7):
    def make_features(data:pd.DataFrame, lag:list[int]=None):
    def remove_unimportant_features(data:pd.DataFrame):
    def features_scaling(data):
    def reshape_data(data):

    data_prep = make_features(data, lag=list(range(1,n+1)))
    data_prep = remove_unimportant_features(data_prep)[-1:]
    data_prep = features_scaling(data_prep)
    data_prep = reshape_data(data_prep)

    return data_prep
```

3 Membuat Interface

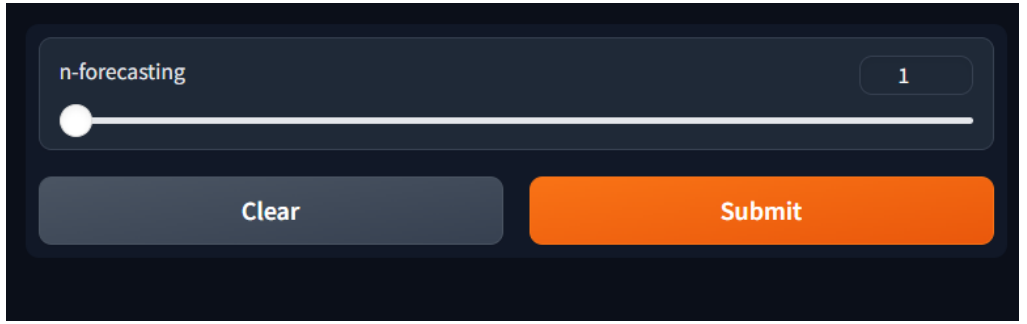
```python
def deploy(n_forecast):
```

Python

```python
# membuat interface
interface = gr.Interface(fn=deploy,
                         inputs=gr.Slider(minimum=1,maximum=5,step=1, label="n-forecasting"),
                         outputs=gr.Json())

interface.launch()
```

Python



output


```
[
  0:  {
    Open:  24396.87890625,
    High:  24708.9765625,
    Low:  23993.013671875,
    Close:  24331.658203125
  }
]
```

1 Query Historical Data dari Yahoo Finance

```python
# function untuk query historical data harga saham UNTR

def query_historical_data(n_days):
```

2 Data Preprocessing

```python
# membuat function untuk data preprocessing
def data_preprocessing(data, n=7):
    def make_features(data:pd.DataFrame, lag:list[int]=None):
    def remove_unimportant_features(data:pd.DataFrame):
    def features_scaling(data):
    def reshape_data(data):

    data_prep = make_features(data, lag=list(range(1,n+1)))
    data_prep = remove_unimportant_features(data_prep)[-1:]
    data_prep = features_scaling(data_prep)
    data_prep = reshape_data(data_prep)

    return data_prep
```

3 Membuat Interface

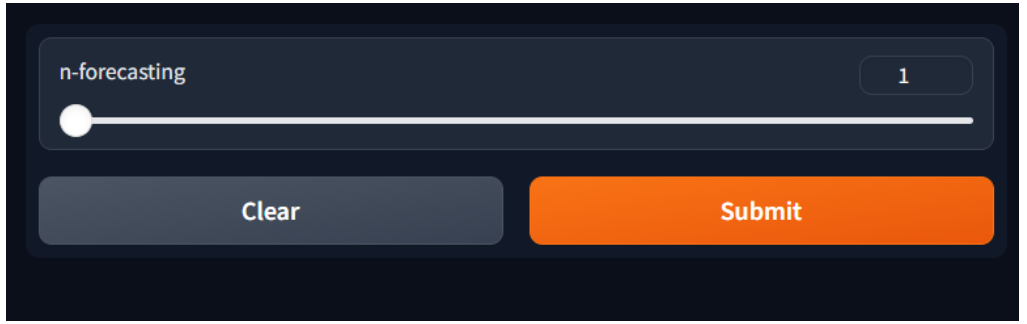
```python
def deploy(n_forecast):
```

Python

```python
# membuat interface
interface = gr.Interface(fn=deploy,
                         inputs=gr.Slider(minimum=1,maximum=5,step=1, label="n-forecasting"),
                         outputs=gr.Json())

interface.launch()
```

Python



output


```
[
  0:  {
    Open:  24396.87890625,
    High:  24708.9765625,
    Low:  23993.013671875,
    Close:  24331.658203125
  }
]
```

TERIMA KASIH

Project ini dapat dilihat pada:
https://github.com/andikaaa18/SkillAcademy_Project/tree/main/Final_project