Nama : Andrew

NIM : 2540119601

Kelas : LA09

Mata Kuliah : Deep Learning

Jurusan : Data Science

Link Video : https://www.youtube.com/watch?v=c6NuhT20-so

Import Dataset

```
# libary
import pandas as pd
```

Data diambil melalui drive

```
# connect to drive to easily get data
from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

Parse colomn date sehingga sesuai dengan format date dan juga membuat kolom date menjadi index, karena time series data

```
# specify to make date as a index
df1 =
pd.read_csv('/content/drive/MyDrive/UAS_deepLearning_data/Dataset
C/GOOGL.csv',parse_dates=["Date"],index_col=["Date"])
df2 =
pd.read_csv('/content/drive/MyDrive/UAS_deepLearning_data/Dataset
C/INTC.csv',parse_dates=["Date"],index_col=["Date"])

print(df1)
print(df2)

                 Open       High        Low      Close    Adj
Close  \
Date

2004-08-19  50.050049  52.082081  48.028027  50.220219
50.220219
2004-08-20  50.555557  54.594593  50.300301  54.209209
54.209209
2004-08-23  55.430431  56.796795  54.579578  54.754753
54.754753
2004-08-24  55.675674  55.855854  51.836838  52.487488
```

```
52.487488
2004-08-25    52.532532    54.054054    51.991993    53.053055
53.053055
...                 ...          ...          ...          ...
...
2020-03-26  1114.719971  1171.479980  1092.030029  1162.920044
1162.920044
2020-03-27  1127.469971  1151.050049  1104.000000  1110.260010
1110.260010
2020-03-30  1132.640015  1151.000000  1098.489990  1146.310059
1146.310059
2020-03-31  1148.729980  1173.400024  1136.719971  1161.949951
1161.949951
2020-04-01  1124.000000  1129.420044  1093.489990  1102.099976
1102.099976

              Volume
Date
2004-08-19  44659000
2004-08-20  22834300
2004-08-23  18256100
2004-08-24  15247300
2004-08-25   9188600
...              ...
2020-03-26   3828100
2020-03-27   3139700
2020-03-30   2936800
2020-03-31   3261400
2020-04-01   2597100

[3932 rows x 6 columns]
                 Open       High        Low      Close  Adj Close
Volume
Date

1980-03-17   0.325521   0.330729   0.325521   0.325521   0.204750
10924800
1980-03-18   0.325521   0.328125   0.322917   0.322917   0.203112
17068800
1980-03-19   0.330729   0.335938   0.330729   0.330729   0.208026
18508800
1980-03-20   0.330729   0.334635   0.329427   0.329427   0.207207
11174400
1980-03-21   0.322917   0.322917   0.317708   0.317708   0.199836
12172800
...               ...        ...        ...        ...        ...
...
2020-03-26  51.740002  55.950001  51.660000  55.540001  55.540001
41459800
2020-03-27  53.419998  54.639999  52.070000  52.369999  52.369999
```

```
31633500
2020-03-30  52.990002  56.099998  52.830002  55.490002  55.490002
31628600
2020-03-31  55.060001  55.799999  53.220001  54.119999  54.119999
48074700
2020-04-01  52.500000  54.689999  51.430000  51.880001  51.880001
29582100

[10098 rows x 6 columns]
```

Pada soal yang dipakai hanya kolom close untuk kedua data, sehingga dapat membuat dataframe baru yang berisikan colomn close.

```
# takes only index and close on each day
google = pd.DataFrame(df1["Close"])
intc = pd.DataFrame(df2["Close"])
print(google.head())
print(intc.head())

                Close
Date
2004-08-19  50.220219
2004-08-20  54.209209
2004-08-23  54.754753
2004-08-24  52.487488
2004-08-25  53.053055
               Close
Date
1980-03-17  0.325521
1980-03-18  0.322917
1980-03-19  0.330729
1980-03-20  0.329427
1980-03-21  0.317708
```

Database telah terbentuk dengan berisikan kolom close dan index date

> With this we can proceeed to the next step, which is LSTM preprocessing

[LO 3, LO 4, 10 poin] Lakukan eksplorasi data terlebih dahulu untuk memahami permasalahan yang dihadapi terlebih dahulu. Dataset yang diberikan adalah data time series, lakukan praproses data untuk menyelesaikan problem dari data tersebut. Pisahkan data time seriestersebut menjadi dua bagian input dan output dengan window size = 5 [dari hari senin s.d jumat] dan horizon = 1 [hari senin saja]. Selanjutnya pisahkan dataset menjadi train, test dan validation set dengan ketentuan (80 train, 10 val, 10 test)

```
# Library
from matplotlib import pyplot as plt
import numpy as np
import math
from sklearn.preprocessing import MinMaxScaler
```

```python
print(google.head(20))
print()
print(intc.head(20))
```

```
                Close
Date
2004-08-19  50.220219
2004-08-20  54.209209
2004-08-23  54.754753
2004-08-24  52.487488
2004-08-25  53.053055
2004-08-26  54.009010
2004-08-27  53.128128
2004-08-30  51.056057
2004-08-31  51.236237
2004-09-01  50.175175
2004-09-02  50.805805
2004-09-03  50.055054
2004-09-07  50.840839
2004-09-08  51.201202
2004-09-09  51.206207
2004-09-10  52.717716
2004-09-13  53.803802
2004-09-14  55.800800
2004-09-15  56.056057
2004-09-16  57.042042

               Close
Date
1980-03-17  0.325521
1980-03-18  0.322917
1980-03-19  0.330729
1980-03-20  0.329427
1980-03-21  0.317708
1980-03-24  0.311198
1980-03-25  0.312500
1980-03-26  0.309896
1980-03-27  0.299479
1980-03-28  0.311198
1980-03-31  0.321615
1980-04-01  0.322917
1980-04-02  0.325521
1980-04-03  0.319010
1980-04-07  0.311198
1980-04-08  0.312500
1980-04-09  0.305990
1980-04-10  0.304688
1980-04-11  0.304688
1980-04-14  0.307292
```

Disini dapat dilihat terdapat beberapa hari libur dimana pada hari tersebut bursa saham tutup, selain itu juga setiap hari sabtu dan minggu bursa sama tutup.

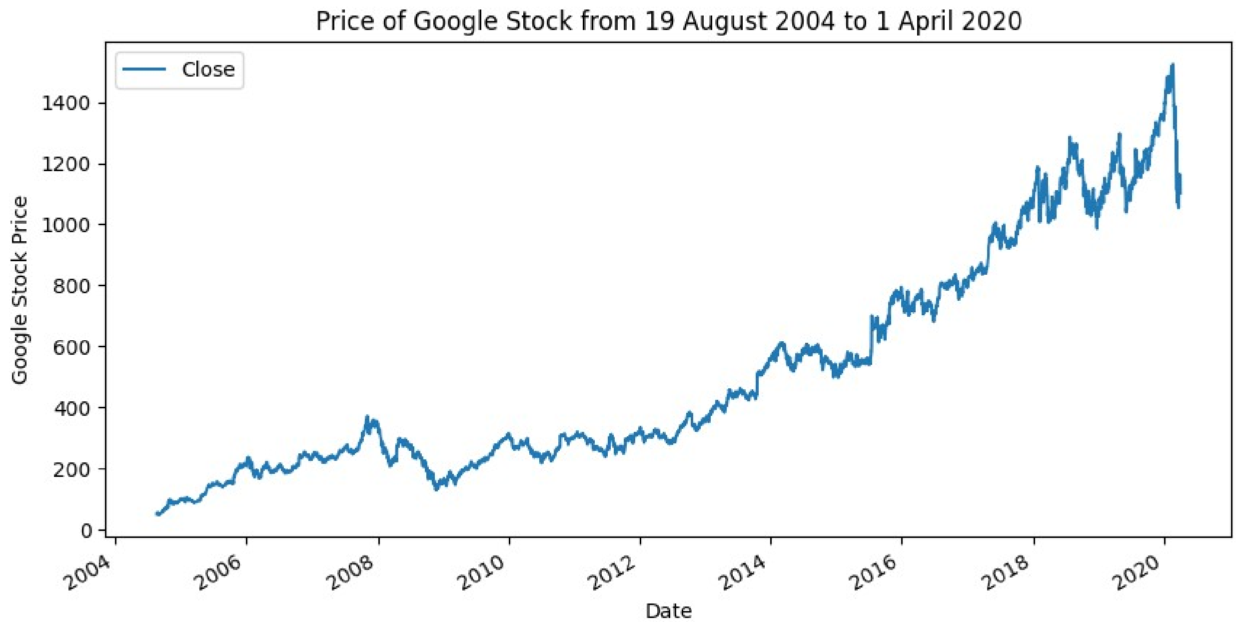
```
# Exploration data on each dataset
# Checking null value
google.info()
print()
intc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3932 entries, 2004-08-19 to 2020-04-01
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Close   3932 non-null   float64
dtypes: float64(1)
memory usage: 61.4 KB

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 10098 entries, 1980-03-17 to 2020-04-01
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Close   10098 non-null  float64
dtypes: float64(1)
memory usage: 157.8 KB
```

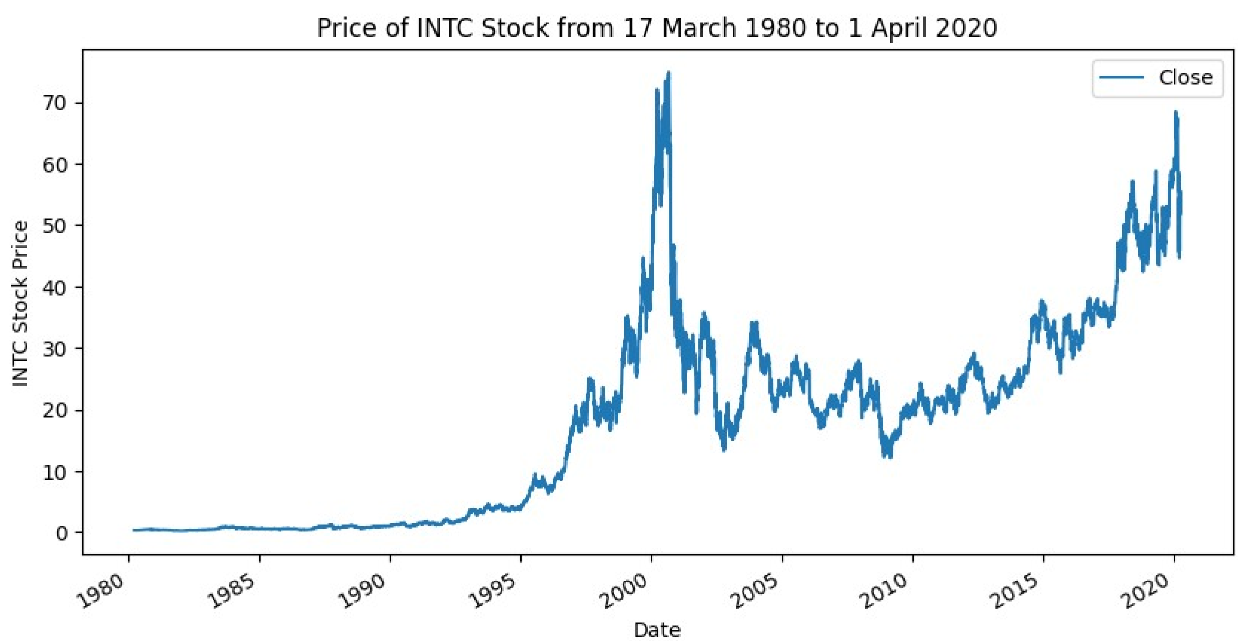
Dari sini diketahui tidak ada null value yang dihasilkan dan terdapat 3932 data pada saham google dan 10098 data pada saham intc

disini data duplicated tidak cek, karena ada kemungkinan saham memiliki value yang sama pada saat close di hari yang berbeda dan juga untuk outlier tidak cek, karena ada kemungkinan di dalam data saham memiliki data yang melonjak tinggi pada hari-hari tertentu atau suatu event.

```
# see the data in plot
google.plot(figsize=(10, 5))
plt.ylabel("Google Stock Price")
plt.title("Price of Google Stock from 19 August 2004 to 1 April 2020",
fontsize=12)
plt.legend(fontsize=10);
```




```
# see the data in plot
intc.plot(figsize=(10, 5))
plt.ylabel("INTC Stock Price")
plt.title("Price of INTC Stock from 17 March 1980 to 1 April 2020",
fontsize=12)
plt.legend(fontsize=10);
```



Dari kedau plot yang telah terbentuk kita mengetahui intel memiliki kenaikan harga close lebih kecil dibandingkan dengan google.

Selain itu intc memiliki data dari tahun 1980, sedangkan google memiliki data tahun 2004.

Untuk trend dari data sendiri dapat dilihat google meningkat setiap tahun, tetapi untuk saham intel megalami kenaikan tinggi disekitar tahun 200.

Before windowing changing make new variable of array to hold a value of the clossing price

```
# to hold value of price
close_goo = google["Close"].to_numpy()
close_intc = intc["Close"].to_numpy()

print(close_goo)
print(close_intc)

[  50.22021866   54.20920944   54.75475311 ... 1146.31005859
1161.94995117
 1102.09997559]
[ 0.32552084  0.32291666  0.33072916 ... 55.49000168 54.11999893
 51.88000107]
```

Membuat function untuk pembuatan window dan horizon yang sekaligus melakukan splitting.

Window yang diminta adalah 5 dari senin hingga jumat dan horizon adalah 1 hari senin saja.

Define scaler untuk melakukan scalling, karena pada modeling akan dikalukan scaling.

```
scaler = MinMaxScaler(feature_range=(0,1))

def window_data(df_close,scaling=False,train_size=0.8,
check_value=False):

    training_data_len = math.ceil(len(df_close)* train_size)

    if scaling is True :
      data = scaler.fit_transform(df_close.values.reshape(-1, 1))
    else:
      data = df_close.values

    train_df = df_close.iloc[: training_data_len]
    train_data = data[:training_data_len]


    #Train set data
    # Define variable for train
    train_window = []
    train_horizon = []

    # using for loop with validate only accept closing when there is
start from monday to friday(window)
    # Also have next following data of monday(horizon)
    for i in range(len(train_df)):
```

```
      if train_df.index[i].weekday() == 0 and i+5 < len(train_df) and
train_df.index[i + 5].weekday() == 0:
        train_window.append(train_data[i:i+5])
        train_horizon.append(train_data[i+5])

    #Determine where the start value of validation and test
    val_test_df = df_close.iloc[training_data_len: ]
    val_test_data = data[training_data_len: ]
    val_test_len = len(val_test_data)
    val_len = int(val_test_len * 0.5)
    test_len = val_test_len - val_len

    # Validation set data
    # Define variable for validation
    val_window = []
    val_horizon = []
    val_df =val_test_df.iloc[:val_len]
    val_data = val_test_data[:val_len]
    # using for loop with validate only accept closing when there is
start from monday to friday(window)
    # Also have next following data of monday(horizon)
    for i in range(len(val_data)):
      if val_df.index[i].weekday() == 0 and i+5 < len(val_df) and
val_df.index[i + 5].weekday() == 0:
        val_window.append(val_data[i:i+5])
        val_horizon.append(val_data[i+5])

    # Test set
    # Define variable for validation
    test_window = []
    test_horizon = []
    test_df = val_test_df.iloc[test_len:]
    test_data = val_test_data[test_len:]
    # using for loop with validate only accept closing when there is
start from monday to friday(window)
    # Also have next following data of monday(horizon)
    for i in range(len(test_df)):
      if test_df.index[i].weekday() == 0 and i+5 < len(test_df) and
test_df.index[i + 5].weekday() == 0:
        test_window.append(test_data[i:i+5])
        test_horizon.append(test_data[i+5])

    # change the window data to array
    train_window = np.array(train_window)
    train_horizon = np.array(train_horizon)
    val_window = np.array(val_window)
    val_horizon = np.array(val_horizon)
    test_window = np.array(test_window)
    test_horizon = np.array(test_horizon)
```

```
    # Reshape the data so it can use in training data
    train_window = np.reshape(train_window, (train_window.shape[0],
train_window.shape[1], 1))
    val_window = np.reshape(val_window, (val_window.shape[0],
val_window.shape[1], 1))
    test_window = np.reshape(test_window, (test_window.shape[0],
test_window.shape[1], 1))
    if check_value is True :
      print("Sample Window :")
      for i in range(5):
        print("train window :",train_window[i].flatten(),"->
Horizon :",train_horizon[i].flatten())
    else:
      return train_window, train_horizon, val_window, val_horizon,
test_window, test_horizon
```

Contoh dari window dan horizon yang akan terbentuk pada train set

```
print("google")
window_data(google,check_value=True)
print()
print("intel")
window_data(intc,check_value=True)
```

```
google
Sample Window :
train window : [54.75475311 52.48748779 53.05305481 54.00901031
53.12812805] -> Horizon : [51.05605698]
train window : [53.80380249 55.80080032 56.05605698 57.04204178
58.80380249] -> Horizon : [59.73973846]
train window : [59.73973846 58.9789772  59.2492485  60.47047043
59.97497559] -> Horizon : [59.18918991]
train window : [59.18918991 63.49349213 65.60560608 64.86486816
66.35635376] -> Horizon : [67.59759521]
train window : [67.59759521 69.2542572  68.60861206 69.49449158
68.93393707] -> Horizon : [67.6977005]

intel
Sample Window :
train window : [0.32552084 0.32291666 0.33072916 0.32942709
0.31770834] -> Horizon : [0.31119791]
train window : [0.31119791 0.3125     0.30989584 0.29947916
0.31119791] -> Horizon : [0.32161459]
train window : [0.31119791 0.3125     0.30598959 0.3046875
0.3046875 ] -> Horizon : [0.30729166]
train window : [0.30729166 0.30338541 0.29166666 0.28645834
0.29036459] -> Horizon : [0.28776041]
train window : [0.28776041 0.30078125 0.31901041 0.3203125
0.31510416] -> Horizon : [0.3125]
```

```
train_win_goo, train_lab_goo, val_win_goo, val_lab_goo, test_win_goo,
test_lab_goo = window_data(google, scaling = True)
len(train_win_goo), len(train_lab_goo), len(val_win_goo),
len(val_lab_goo), len(test_win_goo), len(test_lab_goo)
```

```
(480, 480, 60, 60, 60, 60)
```

```
train_win_int, train_lab_int, val_win_int, val_lab_int, test_win_int,
test_lab_int = window_data(intc, scaling = True)
len(train_win_int), len(train_lab_int), len(val_win_int),
len(val_lab_int), len(test_win_int), len(test_lab_int)
```

```
(1269, 1269, 154, 154, 153, 153)
```

```
test_win_int_check =
scaler.inverse_transform(test_win_int[:,0,0].reshape(-1, 1))
test_win_goo_check =
scaler.inverse_transform(test_win_goo[:,0,0].reshape(-1, 1))
```

Degan begitu kedua dataset telah siap untuk dimodelkan

[LO 3, LO 4, 5 poin] Buatlah arsitektur baseline dengan LSTM (units=50) dan layer akhir berupa node Perceptron dengan units=1. Activation function untuk LSTM menggunakan ReLU

Sebelum modeling membuat beberapa function untuk mengevaluasi model

```
# library
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

def make_preds(model, input_data):
  forecast = model.predict(input_data)
  return tf.squeeze(forecast)

# function for evaluate model
def evaluate_preds(y_true, y_pred):
  # Make sure float32 (for metric calculations)
  y_true = tf.cast(y_true, dtype=tf.float32)
  y_pred = tf.cast(y_pred, dtype=tf.float32)

  # Calculate various metrics
  mae = tf.keras.metrics.mean_absolute_error(y_true, y_pred)
  mse = tf.keras.metrics.mean_squared_error(y_true, y_pred)
  rmse = tf.sqrt(mse)
  mape = tf.keras.metrics.mean_absolute_percentage_error(y_true,
y_pred)

  return mae.numpy(),rmse.numpy(),mape.numpy()
```

Membuat dataframe yang akan menampung result dari model

```
google_model = pd.DataFrame(columns=['Model', 'MAE', 'RMSE', 'MAPE'])

intel_model = pd.DataFrame(columns=['Model', 'MAE', 'RMSE', 'MAPE'])
```

Basemodel google

```
basemodel_goo =  keras.Sequential()

# imput layer next with lstm
basemodel_goo.add(layers.LSTM(units=50, input_shape=(5, 1),
activation="relu"))
# output layer
basemodel_goo.add(layers.Dense(1))

basemodel_goo.compile(loss="mae")

basemodel_goo.fit(train_win_goo,
            train_lab_goo,
            epochs=40,
            verbose=1,
            batch_size = 32,
            validation_data=(val_win_goo, val_lab_goo))

Epoch 1/40
15/15 [==============================] - 3s 23ms/step - loss: 0.1601 -
val_loss: 0.5111
Epoch 2/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0862 -
val_loss: 0.3806
Epoch 3/40
15/15 [==============================] - 0s 8ms/step - loss: 0.0592 -
val_loss: 0.2820
Epoch 4/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0387 -
val_loss: 0.1713
Epoch 5/40
15/15 [==============================] - 0s 9ms/step - loss: 0.0176 -
val_loss: 0.0544
Epoch 6/40
15/15 [==============================] - 0s 7ms/step - loss: 0.0108 -
val_loss: 0.0563
Epoch 7/40
15/15 [==============================] - 0s 7ms/step - loss: 0.0098 -
val_loss: 0.0347
Epoch 8/40
15/15 [==============================] - 0s 7ms/step - loss: 0.0097 -
val_loss: 0.0565
Epoch 9/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0094 -
val_loss: 0.0444
```

```
Epoch 10/40
15/15 [==============================] - 0s 8ms/step - loss: 0.0091 -
val_loss: 0.0592
Epoch 11/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0096 -
val_loss: 0.0444
Epoch 12/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0093 -
val_loss: 0.0586
Epoch 13/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0094 -
val_loss: 0.0321
Epoch 14/40
15/15 [==============================] - 0s 7ms/step - loss: 0.0090 -
val_loss: 0.0522
Epoch 15/40
15/15 [==============================] - 0s 5ms/step - loss: 0.0093 -
val_loss: 0.0325
Epoch 16/40
15/15 [==============================] - 0s 5ms/step - loss: 0.0093 -
val_loss: 0.0552
Epoch 17/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0098 -
val_loss: 0.0337
Epoch 18/40
15/15 [==============================] - 0s 5ms/step - loss: 0.0092 -
val_loss: 0.0503
Epoch 19/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0093 -
val_loss: 0.0381
Epoch 20/40
15/15 [==============================] - 0s 5ms/step - loss: 0.0089 -
val_loss: 0.0572
Epoch 21/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0090 -
val_loss: 0.0359
Epoch 22/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0094 -
val_loss: 0.0473
Epoch 23/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0084 -
val_loss: 0.0224
Epoch 24/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0097 -
val_loss: 0.0449
Epoch 25/40
15/15 [==============================] - 0s 8ms/step - loss: 0.0086 -
val_loss: 0.0199
Epoch 26/40
```

```
15/15 [==============================] - 0s 8ms/step - loss: 0.0098 -
val_loss: 0.0454
Epoch 27/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0091 -
val_loss: 0.0261
Epoch 28/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0084 -
val_loss: 0.0223
Epoch 29/40
15/15 [==============================] - 0s 7ms/step - loss: 0.0092 -
val_loss: 0.0464
Epoch 30/40
15/15 [==============================] - 0s 7ms/step - loss: 0.0082 -
val_loss: 0.0253
Epoch 31/40
15/15 [==============================] - 0s 7ms/step - loss: 0.0091 -
val_loss: 0.0452
Epoch 32/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0089 -
val_loss: 0.0231
Epoch 33/40
15/15 [==============================] - 0s 7ms/step - loss: 0.0089 -
val_loss: 0.0393
Epoch 34/40
15/15 [==============================] - 0s 7ms/step - loss: 0.0088 -
val_loss: 0.0270
Epoch 35/40
15/15 [==============================] - 0s 7ms/step - loss: 0.0090 -
val_loss: 0.0452
Epoch 36/40
15/15 [==============================] - 0s 7ms/step - loss: 0.0086 -
val_loss: 0.0211
Epoch 37/40
15/15 [==============================] - 0s 7ms/step - loss: 0.0090 -
val_loss: 0.0413
Epoch 38/40
15/15 [==============================] - 0s 7ms/step - loss: 0.0090 -
val_loss: 0.0225
Epoch 39/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0085 -
val_loss: 0.0239
Epoch 40/40
15/15 [==============================] - 0s 6ms/step - loss: 0.0092 -
val_loss: 0.0395

<keras.callbacks.History at 0x7fe1d4ce5c00>
```

```
basemodel_goo.evaluate(test_win_goo, test_lab_goo)
```

```
2/2 [==============================] - 0s 7ms/step - loss: 0.0584
```

```
0.05842805653810501
```

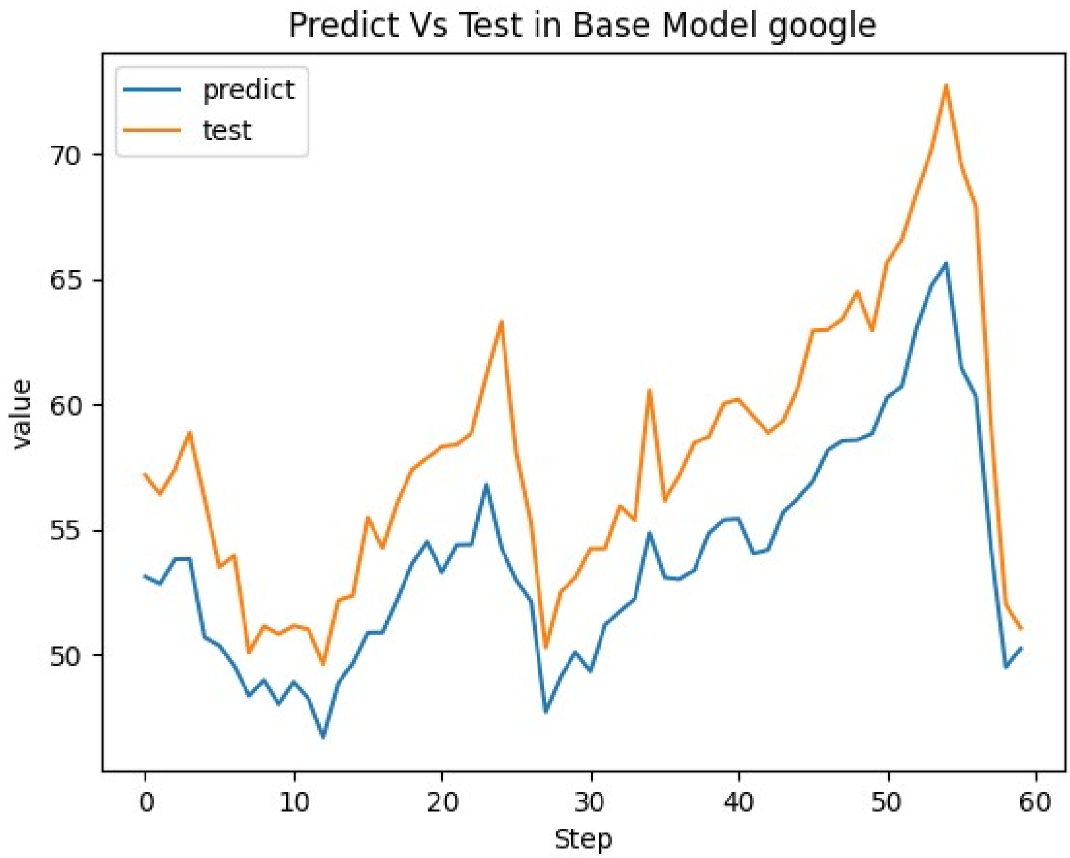
```
basemodel_goo_preds = basemodel_goo.predict(test_win_goo)
basemodel_goo_preds_check =
scaler.inverse_transform(basemodel_goo_preds)
```

```
2/2 [==============================] - 0s 17ms/step
```

```
basemodel_goo_results =
evaluate_preds(y_true=tf.squeeze(test_lab_goo),
y_pred=basemodel_goo_preds.flatten())
google_model.loc[0] =
["Base",basemodel_goo_results[0],basemodel_goo_results[1],basemodel_go
o_results[2]]
print("MAE :",basemodel_goo_results[0])
print("RMSE :",basemodel_goo_results[1])
print("MAPE :",basemodel_goo_results[2])
```

```
MAE : 0.058428064
RMSE : 0.06350657
MAPE : 7.3717737
```

```
plt.plot(basemodel_goo_preds_check, label='predict')
plt.plot(test_win_goo_check,label='test')
plt.title("Predict Vs Test in Base Model google")
plt.xlabel("Step")
plt.ylabel("value")
plt.legend()
plt.show()
```




```
basemodel_int = keras.Sequential()

# imput layer next with lstm
basemodel_int.add(layers.LSTM(units=50, input_shape=(5,1),
activation="relu"))
# output layer
basemodel_int.add(layers.Dense(1))

basemodel_int.compile(loss="mae")

history = basemodel_int.fit(train_win_int,
            train_lab_int,
            epochs=40,
            verbose=1,
            validation_data=(val_win_int, val_lab_int))

Epoch 1/40
40/40 [==============================] - 2s 11ms/step - loss: 0.1021 -
val_loss: 0.0985
Epoch 2/40
40/40 [==============================] - 0s 5ms/step - loss: 0.0164 -
val_loss: 0.0075
Epoch 3/40
40/40 [==============================] - 0s 5ms/step - loss: 0.0097 -
```

```
val_loss: 0.0068
Epoch 4/40
40/40 [==============================] - 0s 5ms/step - loss: 0.0095 -
val_loss: 0.0058
Epoch 5/40
40/40 [==============================] - 0s 5ms/step - loss: 0.0089 -
val_loss: 0.0068
Epoch 6/40
40/40 [==============================] - 0s 5ms/step - loss: 0.0091 -
val_loss: 0.0071
Epoch 7/40
40/40 [==============================] - 0s 5ms/step - loss: 0.0089 -
val_loss: 0.0089
Epoch 8/40
40/40 [==============================] - 0s 5ms/step - loss: 0.0089 -
val_loss: 0.0085
Epoch 9/40
40/40 [==============================] - 0s 9ms/step - loss: 0.0088 -
val_loss: 0.0098
Epoch 10/40
40/40 [==============================] - 0s 8ms/step - loss: 0.0089 -
val_loss: 0.0075
Epoch 11/40
40/40 [==============================] - 0s 7ms/step - loss: 0.0084 -
val_loss: 0.0063
Epoch 12/40
40/40 [==============================] - 0s 9ms/step - loss: 0.0087 -
val_loss: 0.0065
Epoch 13/40
40/40 [==============================] - 0s 9ms/step - loss: 0.0084 -
val_loss: 0.0113
Epoch 14/40
40/40 [==============================] - 0s 7ms/step - loss: 0.0081 -
val_loss: 0.0133
Epoch 15/40
40/40 [==============================] - 0s 9ms/step - loss: 0.0084 -
val_loss: 0.0073
Epoch 16/40
40/40 [==============================] - 0s 9ms/step - loss: 0.0083 -
val_loss: 0.0077
Epoch 17/40
40/40 [==============================] - 0s 9ms/step - loss: 0.0085 -
val_loss: 0.0065
Epoch 18/40
40/40 [==============================] - 0s 9ms/step - loss: 0.0081 -
val_loss: 0.0108
Epoch 19/40
40/40 [==============================] - 0s 9ms/step - loss: 0.0084 -
val_loss: 0.0108
```

```
Epoch 20/40
40/40 [==============================] - 0s 9ms/step - loss: 0.0081 -
val_loss: 0.0076
Epoch 21/40
40/40 [==============================] - 0s 9ms/step - loss: 0.0084 -
val_loss: 0.0091
Epoch 22/40
40/40 [==============================] - 0s 6ms/step - loss: 0.0082 -
val_loss: 0.0088
Epoch 23/40
40/40 [==============================] - 0s 5ms/step - loss: 0.0081 -
val_loss: 0.0064
Epoch 24/40
40/40 [==============================] - 0s 5ms/step - loss: 0.0083 -
val_loss: 0.0130
Epoch 25/40
40/40 [==============================] - 0s 5ms/step - loss: 0.0077 -
val_loss: 0.0093
Epoch 26/40
40/40 [==============================] - 0s 6ms/step - loss: 0.0082 -
val_loss: 0.0096
Epoch 27/40
40/40 [==============================] - 0s 6ms/step - loss: 0.0078 -
val_loss: 0.0061
Epoch 28/40
40/40 [==============================] - 0s 6ms/step - loss: 0.0081 -
val_loss: 0.0116
Epoch 29/40
40/40 [==============================] - 0s 6ms/step - loss: 0.0078 -
val_loss: 0.0073
Epoch 30/40
40/40 [==============================] - 0s 6ms/step - loss: 0.0078 -
val_loss: 0.0115
Epoch 31/40
40/40 [==============================] - 0s 6ms/step - loss: 0.0078 -
val_loss: 0.0081
Epoch 32/40
40/40 [==============================] - 0s 6ms/step - loss: 0.0078 -
val_loss: 0.0060
Epoch 33/40
40/40 [==============================] - 0s 5ms/step - loss: 0.0081 -
val_loss: 0.0164
Epoch 34/40
40/40 [==============================] - 0s 5ms/step - loss: 0.0077 -
val_loss: 0.0059
Epoch 35/40
40/40 [==============================] - 0s 5ms/step - loss: 0.0077 -
val_loss: 0.0088
Epoch 36/40
```

```
40/40 [==============================] - 0s 6ms/step - loss: 0.0078 -
val_loss: 0.0074
Epoch 37/40
40/40 [==============================] - 0s 5ms/step - loss: 0.0078 -
val_loss: 0.0060
Epoch 38/40
40/40 [==============================] - 0s 5ms/step - loss: 0.0078 -
val_loss: 0.0117
Epoch 39/40
40/40 [==============================] - 0s 6ms/step - loss: 0.0077 -
val_loss: 0.0062
Epoch 40/40
40/40 [==============================] - 0s 5ms/step - loss: 0.0075 -
val_loss: 0.0057
```

```
basemodel_int.evaluate(test_win_int, test_lab_int)
```

```
5/5 [==============================] - 0s 4ms/step - loss: 0.0125
```

```
0.012461621314287186
```

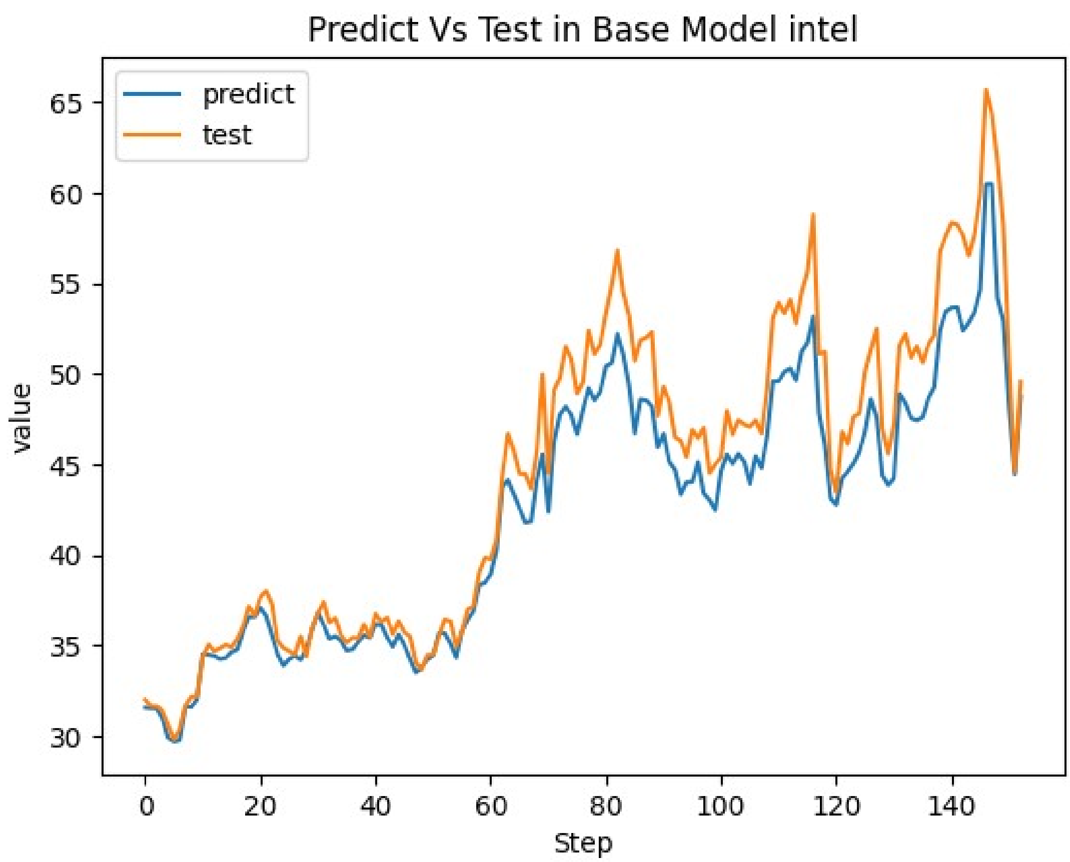
```
basemodel_int_preds = basemodel_goo.predict(test_win_int)
basemodel_int_preds_check =
scaler.inverse_transform(basemodel_int_preds)
```

```
5/5 [==============================] - 0s 3ms/step
```

```
basemodel_int_results =
evaluate_preds(y_true=tf.squeeze(test_lab_int),
y_pred=basemodel_int_preds.flatten())
intel_model.loc[0] =
["Base",basemodel_int_results[0],basemodel_int_results[1],basemodel_in
t_results[2]]
print("MAE :",basemodel_int_results[0])
print("RMSE :",basemodel_int_results[1])
print("MAPE :",basemodel_int_results[2])
```

```
MAE : 0.029387984
RMSE : 0.037500776
MAPE : 4.477377
```

```
plt.plot(basemodel_int_preds_check, label='predict')
plt.plot(test_win_int_check,label='test')
plt.title("Predict Vs Test in Base Model intel")
plt.xlabel("Step")
plt.ylabel("value")
plt.legend()
plt.show()
```



[LO 1, LO 2, LO 3, LO 4, 15 poin] Setelah mengetahui hasil dari nomor (1c), modifikasi arsitektur pada nomor 1c untuk mendapatkan unjuk kerja yang optimal (kalian dapat menambahkan atau mengurangi arsitektur tersebut, atau mengganti hyperparameter, atau menggunakan tuning pada hyperparameter). Jelaskan alasan kalian untuk menggunakan pendekatan yang kalian pilih

Lanjut ke modelling data LSTM dengan menggunakan arsitektur yang sama untuk google dengan menabhkan unit 75. Dan terdapat tambahan pada google dimana google akan diganti lossnya menjadi MSE , sendangkan untuk intel tetap dengan MAE.

```
model1_goo =  keras.Sequential()

# imput layer next with lstm
model1_goo.add(layers.LSTM(units=75, input_shape=(5, 1),
activation="relu"))
# output layer
model1_goo.add(layers.Dense(1,activation="linear"))

model1_goo.compile(loss="mse",optimizer=tf.optimizers.Adam())

model1_goo.fit(train_win_goo,
            train_lab_goo,
            epochs=50,
            verbose=1,
```

```
            batch_size=32,
            validation_data=(val_win_goo, val_lab_goo))
```

```
Epoch 1/50
15/15 [==============================] - 1s 23ms/step - loss: 0.0323 -
val_loss: 0.1576
Epoch 2/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0075 -
val_loss: 0.0313
Epoch 3/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0032 -
val_loss: 0.0171
Epoch 4/50
15/15 [==============================] - 0s 8ms/step - loss: 9.9217e-
04 - val_loss: 5.6652e-04
Epoch 5/50
15/15 [==============================] - 0s 8ms/step - loss: 1.3470e-
04 - val_loss: 9.4508e-04
Epoch 6/50
15/15 [==============================] - 0s 7ms/step - loss: 1.2114e-
04 - val_loss: 2.9611e-04
Epoch 7/50
15/15 [==============================] - 0s 7ms/step - loss: 7.7449e-
05 - val_loss: 2.7126e-04
Epoch 8/50
15/15 [==============================] - 0s 7ms/step - loss: 6.4564e-
05 - val_loss: 3.1775e-04
Epoch 9/50
15/15 [==============================] - 0s 6ms/step - loss: 6.3733e-
05 - val_loss: 3.9596e-04
Epoch 10/50
15/15 [==============================] - 0s 8ms/step - loss: 5.9902e-
05 - val_loss: 3.4570e-04
Epoch 11/50
15/15 [==============================] - 0s 8ms/step - loss: 5.9867e-
05 - val_loss: 3.2772e-04
Epoch 12/50
15/15 [==============================] - 0s 8ms/step - loss: 5.9743e-
05 - val_loss: 3.4698e-04
Epoch 13/50
15/15 [==============================] - 0s 7ms/step - loss: 5.9524e-
05 - val_loss: 3.6300e-04
Epoch 14/50
15/15 [==============================] - 0s 9ms/step - loss: 6.0247e-
05 - val_loss: 3.2966e-04
Epoch 15/50
15/15 [==============================] - 0s 8ms/step - loss: 5.8573e-
05 - val_loss: 3.4464e-04
Epoch 16/50
15/15 [==============================] - 0s 7ms/step - loss: 5.8987e-
```

```
05 - val_loss: 3.6849e-04
Epoch 17/50
15/15 [==============================] - 0s 8ms/step - loss: 6.4723e-
05 - val_loss: 3.5814e-04
Epoch 18/50
15/15 [==============================] - 0s 8ms/step - loss: 6.1569e-
05 - val_loss: 3.2144e-04
Epoch 19/50
15/15 [==============================] - 0s 8ms/step - loss: 6.2490e-
05 - val_loss: 2.9850e-04
Epoch 20/50
15/15 [==============================] - 0s 9ms/step - loss: 6.3607e-
05 - val_loss: 3.3969e-04
Epoch 21/50
15/15 [==============================] - 0s 8ms/step - loss: 5.9884e-
05 - val_loss: 3.2060e-04
Epoch 22/50
15/15 [==============================] - 0s 8ms/step - loss: 5.8298e-
05 - val_loss: 3.1672e-04
Epoch 23/50
15/15 [==============================] - 0s 8ms/step - loss: 6.0392e-
05 - val_loss: 3.6694e-04
Epoch 24/50
15/15 [==============================] - 0s 7ms/step - loss: 6.1647e-
05 - val_loss: 3.3074e-04
Epoch 25/50
15/15 [==============================] - 0s 7ms/step - loss: 5.8920e-
05 - val_loss: 3.4089e-04
Epoch 26/50
15/15 [==============================] - 0s 8ms/step - loss: 6.0422e-
05 - val_loss: 3.6006e-04
Epoch 27/50
15/15 [==============================] - 0s 7ms/step - loss: 6.0028e-
05 - val_loss: 3.4334e-04
Epoch 28/50
15/15 [==============================] - 0s 6ms/step - loss: 6.0156e-
05 - val_loss: 3.3836e-04
Epoch 29/50
15/15 [==============================] - 0s 7ms/step - loss: 5.9080e-
05 - val_loss: 4.0785e-04
Epoch 30/50
15/15 [==============================] - 0s 9ms/step - loss: 6.5897e-
05 - val_loss: 3.1646e-04
Epoch 31/50
15/15 [==============================] - 0s 10ms/step - loss: 6.1212e-
05 - val_loss: 3.4801e-04
Epoch 32/50
15/15 [==============================] - 0s 11ms/step - loss: 6.0250e-
05 - val_loss: 3.5725e-04
Epoch 33/50
```

```
15/15 [==============================] - 0s 10ms/step - loss: 5.8982e-
05 - val_loss: 3.2544e-04
Epoch 34/50
15/15 [==============================] - 0s 10ms/step - loss: 6.2455e-
05 - val_loss: 4.0550e-04
Epoch 35/50
15/15 [==============================] - 0s 10ms/step - loss: 6.1966e-
05 - val_loss: 3.0038e-04
Epoch 36/50
15/15 [==============================] - 0s 11ms/step - loss: 5.8421e-
05 - val_loss: 3.0402e-04
Epoch 37/50
15/15 [==============================] - 0s 12ms/step - loss: 5.7907e-
05 - val_loss: 3.4071e-04
Epoch 38/50
15/15 [==============================] - 0s 11ms/step - loss: 5.8245e-
05 - val_loss: 3.0729e-04
Epoch 39/50
15/15 [==============================] - 0s 13ms/step - loss: 5.8615e-
05 - val_loss: 3.8724e-04
Epoch 40/50
15/15 [==============================] - 0s 12ms/step - loss: 5.9259e-
05 - val_loss: 3.2495e-04
Epoch 41/50
15/15 [==============================] - 0s 12ms/step - loss: 5.7831e-
05 - val_loss: 3.1554e-04
Epoch 42/50
15/15 [==============================] - 0s 10ms/step - loss: 5.8790e-
05 - val_loss: 2.6115e-04
Epoch 43/50
15/15 [==============================] - 0s 12ms/step - loss: 7.4141e-
05 - val_loss: 4.1237e-04
Epoch 44/50
15/15 [==============================] - 0s 10ms/step - loss: 6.0204e-
05 - val_loss: 2.8191e-04
Epoch 45/50
15/15 [==============================] - 0s 11ms/step - loss: 6.0677e-
05 - val_loss: 3.4238e-04
Epoch 46/50
15/15 [==============================] - 0s 10ms/step - loss: 6.4818e-
05 - val_loss: 2.6467e-04
Epoch 47/50
15/15 [==============================] - 0s 11ms/step - loss: 6.6981e-
05 - val_loss: 4.4021e-04
Epoch 48/50
15/15 [==============================] - 0s 13ms/step - loss: 5.9340e-
05 - val_loss: 3.2720e-04
Epoch 49/50
15/15 [==============================] - 0s 12ms/step - loss: 6.1489e-
05 - val_loss: 3.1222e-04
```

```
Epoch 50/50
15/15 [==============================] - 0s 12ms/step - loss: 5.8303e-05 - val_loss: 2.8065e-04

<keras.callbacks.History at 0x7fe1e4047ee0>
```

```
model1_goo.evaluate(test_win_goo, test_lab_goo)
```

```
2/2 [==============================] - 0s 9ms/step - loss: 5.8986e-04

0.0005898595554754138
```

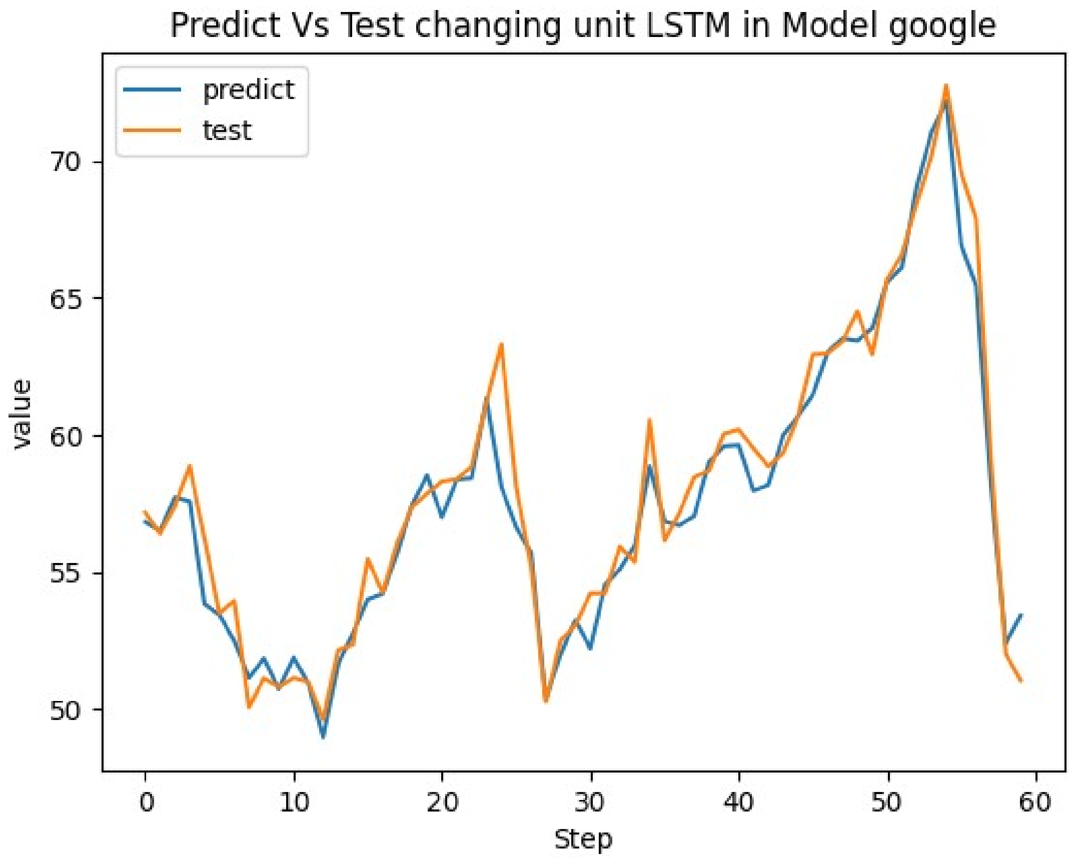
```
model1_goo_preds = model1_goo.predict(test_win_goo)
model1_goo_preds_check = scaler.inverse_transform(model1_goo_preds)
```

```
2/2 [==============================] - 0s 9ms/step
```

```
model1_goo_results = evaluate_preds(y_true=tf.squeeze(test_lab_goo), y_pred=model1_goo_preds.flatten())
google_model.loc[1] = ["Unit 75 with MSE",model1_goo_results[0],model1_goo_results[1],model1_goo_results[2]]
print("MAE :",model1_goo_results[0])
print("RMSE :",model1_goo_results[1])
print("MAPE :",model1_goo_results[2])
```

```
MAE : 0.017915709
RMSE : 0.02428702
MAPE : 2.3419776
```

```
plt.plot(model1_goo_preds_check, label='predict')
plt.plot(test_win_goo_check,label='test')
plt.title("Predict Vs Test changing unit LSTM in Model google")
plt.xlabel("Step")
plt.ylabel("value")
plt.legend()
plt.show()
```




```
model1_int =  keras.Sequential()

# imput layer next with lstm
model1_int.add(layers.LSTM(units=75, input_shape=(5, 1),
activation="relu"))
# output layer
model1_int.add(layers.Dense(1,activation="linear"))

model1_int.compile(loss="mae",optimizer=tf.optimizers.Adam())

model1_int.fit(train_win_goo,
            train_lab_goo,
            epochs=50,
            verbose=1,
            batch_size=32,
            validation_data=(val_win_goo, val_lab_goo))

Epoch 1/50
15/15 [==============================] - 1s 22ms/step - loss: 0.1716 -
val_loss: 0.5270
Epoch 2/50
15/15 [==============================] - 0s 7ms/step - loss: 0.1018 -
val_loss: 0.3642
Epoch 3/50
```

```
15/15 [==============================] - 0s 9ms/step - loss: 0.0624 -
val_loss: 0.2341
Epoch 4/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0436 -
val_loss: 0.0999
Epoch 5/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0147 -
val_loss: 0.0677
Epoch 6/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0078 -
val_loss: 0.0148
Epoch 7/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0060 -
val_loss: 0.0141
Epoch 8/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0061 -
val_loss: 0.0141
Epoch 9/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0057 -
val_loss: 0.0138
Epoch 10/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0055 -
val_loss: 0.0140
Epoch 11/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0053 -
val_loss: 0.0122
Epoch 12/50
15/15 [==============================] - 0s 21ms/step - loss: 0.0056 -
val_loss: 0.0150
Epoch 13/50
15/15 [==============================] - 0s 23ms/step - loss: 0.0053 -
val_loss: 0.0133
Epoch 14/50
15/15 [==============================] - 0s 20ms/step - loss: 0.0058 -
val_loss: 0.0124
Epoch 15/50
15/15 [==============================] - 0s 11ms/step - loss: 0.0053 -
val_loss: 0.0124
Epoch 16/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0052 -
val_loss: 0.0123
Epoch 17/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0052 -
val_loss: 0.0124
Epoch 18/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0056 -
val_loss: 0.0131
Epoch 19/50
15/15 [==============================] - 0s 17ms/step - loss: 0.0053 -
```

```
val_loss: 0.0123
Epoch 20/50
15/15 [==============================] - 0s 20ms/step - loss: 0.0059 -
val_loss: 0.0141
Epoch 21/50
15/15 [==============================] - 0s 20ms/step - loss: 0.0057 -
val_loss: 0.0123
Epoch 22/50
15/15 [==============================] - 0s 10ms/step - loss: 0.0056 -
val_loss: 0.0169
Epoch 23/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0062 -
val_loss: 0.0128
Epoch 24/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0058 -
val_loss: 0.0140
Epoch 25/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0052 -
val_loss: 0.0152
Epoch 26/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0055 -
val_loss: 0.0126
Epoch 27/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0056 -
val_loss: 0.0151
Epoch 28/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0051 -
val_loss: 0.0132
Epoch 29/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0053 -
val_loss: 0.0135
Epoch 30/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0054 -
val_loss: 0.0124
Epoch 31/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0057 -
val_loss: 0.0131
Epoch 32/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0052 -
val_loss: 0.0127
Epoch 33/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0053 -
val_loss: 0.0124
Epoch 34/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0052 -
val_loss: 0.0134
Epoch 35/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0054 -
val_loss: 0.0141
```

```
Epoch 36/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0056 -
val_loss: 0.0123
Epoch 37/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0060 -
val_loss: 0.0125
Epoch 38/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0056 -
val_loss: 0.0122
Epoch 39/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0058 -
val_loss: 0.0156
Epoch 40/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0054 -
val_loss: 0.0122
Epoch 41/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0055 -
val_loss: 0.0136
Epoch 42/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0057 -
val_loss: 0.0132
Epoch 43/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0054 -
val_loss: 0.0139
Epoch 44/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0061 -
val_loss: 0.0145
Epoch 45/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0060 -
val_loss: 0.0137
Epoch 46/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0052 -
val_loss: 0.0133
Epoch 47/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0053 -
val_loss: 0.0131
Epoch 48/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0060 -
val_loss: 0.0123
Epoch 49/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0051 -
val_loss: 0.0123
Epoch 50/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0055 -
val_loss: 0.0127

<keras.callbacks.History at 0x7fe1e61e1420>

model1_int.evaluate(test_win_int, test_lab_int)
```

```
5/5 [==============================] - 0s 4ms/step - loss: 0.0128

0.012817795388400555
```

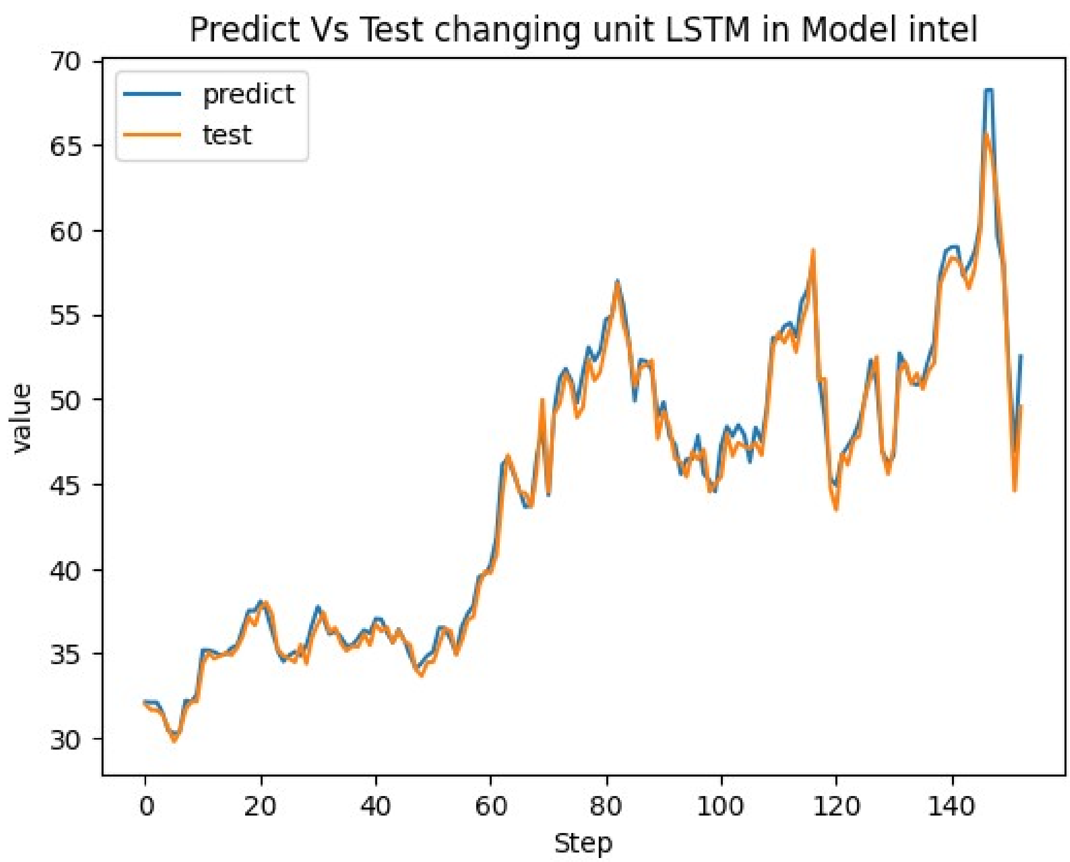
```
model1_int_preds = model1_int.predict(test_win_int)
model1_int_preds_check = scaler.inverse_transform(model1_int_preds)
```

```
5/5 [==============================] - 0s 5ms/step
```

```
model1_int_results =
evaluate_preds(y_true=tf.squeeze(test_lab_int),y_pred=model1_int_preds
.flatten())
intel_model.loc[1] = ["Unit 75 with
MAE",model1_int_results[0],model1_int_results[1],model1_int_results[2]
]
print("MAE :",model1_int_results[0])
print("RMSE :",model1_int_results[1])
print("MAPE :",model1_int_results[2])
```

```
MAE : 0.012817802
RMSE : 0.021113459
MAPE : 2.083454
```

```
plt.plot(model1_int_preds_check, label='predict')
plt.plot(test_win_int_check,label='test')
plt.title("Predict Vs Test changing unit LSTM in Model intel")
plt.xlabel("Step")
plt.ylabel("value")
plt.legend()
plt.show()
```



Setelah mendapatkan hasil dari model pertama, loss yang didapatkan cukup baik.

Untuk melakukan uji coba selenajutnya akan mecoba algoritma dari GRU untuk melihat performa mana yang lebih baik.

Pada model kedua akan menggunakan GRU sebagai pengganti dari LSTM

```
model2_goo =  keras.Sequential()

# imput layer next with lstm
model2_goo.add(layers.GRU(units=50, input_shape=(5, 1),
activation="relu"))
# output layer
model2_goo.add(layers.Dense(1,activation="linear"))

model2_goo.compile(loss="mse",optimizer=tf.optimizers.Adam())

model2_goo.fit(train_win_goo,
            train_lab_goo,
            epochs=50,
            verbose=1,
            batch_size=32,
            validation_data=(val_win_goo, val_lab_goo))
```

```
Epoch 1/50
15/15 [==============================] - 2s 24ms/step - loss: 0.0269 -
val_loss: 0.1247
Epoch 2/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0050 -
val_loss: 0.0333
Epoch 3/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0023 -
val_loss: 0.0181
Epoch 4/50
15/15 [==============================] - 0s 5ms/step - loss: 6.6459e-
04 - val_loss: 0.0054
Epoch 5/50
15/15 [==============================] - 0s 5ms/step - loss: 6.9551e-
05 - val_loss: 0.0020
Epoch 6/50
15/15 [==============================] - 0s 6ms/step - loss: 6.5356e-
05 - val_loss: 0.0025
Epoch 7/50
15/15 [==============================] - 0s 7ms/step - loss: 4.1987e-
05 - val_loss: 0.0033
Epoch 8/50
15/15 [==============================] - 0s 5ms/step - loss: 4.2685e-
05 - val_loss: 0.0028
Epoch 9/50
15/15 [==============================] - 0s 7ms/step - loss: 4.3131e-
05 - val_loss: 0.0027
Epoch 10/50
15/15 [==============================] - 0s 5ms/step - loss: 4.0675e-
05 - val_loss: 0.0026
Epoch 11/50
15/15 [==============================] - 0s 5ms/step - loss: 4.3831e-
05 - val_loss: 0.0026
Epoch 12/50
15/15 [==============================] - 0s 6ms/step - loss: 4.3114e-
05 - val_loss: 0.0028
Epoch 13/50
15/15 [==============================] - 0s 6ms/step - loss: 4.0302e-
05 - val_loss: 0.0026
Epoch 14/50
15/15 [==============================] - 0s 6ms/step - loss: 4.1054e-
05 - val_loss: 0.0025
Epoch 15/50
15/15 [==============================] - 0s 7ms/step - loss: 4.1749e-
05 - val_loss: 0.0024
Epoch 16/50
15/15 [==============================] - 0s 7ms/step - loss: 4.1266e-
05 - val_loss: 0.0024
Epoch 17/50
15/15 [==============================] - 0s 8ms/step - loss: 3.9834e-
```

```
05 - val_loss: 0.0024
Epoch 18/50
15/15 [==============================] - 0s 7ms/step - loss: 4.0451e-
05 - val_loss: 0.0024
Epoch 19/50
15/15 [==============================] - 0s 6ms/step - loss: 4.2439e-
05 - val_loss: 0.0025
Epoch 20/50
15/15 [==============================] - 0s 6ms/step - loss: 4.1639e-
05 - val_loss: 0.0024
Epoch 21/50
15/15 [==============================] - 0s 7ms/step - loss: 4.2440e-
05 - val_loss: 0.0024
Epoch 22/50
15/15 [==============================] - 0s 5ms/step - loss: 4.0218e-
05 - val_loss: 0.0025
Epoch 23/50
15/15 [==============================] - 0s 6ms/step - loss: 4.0123e-
05 - val_loss: 0.0023
Epoch 24/50
15/15 [==============================] - 0s 6ms/step - loss: 4.0245e-
05 - val_loss: 0.0023
Epoch 25/50
15/15 [==============================] - 0s 6ms/step - loss: 3.9630e-
05 - val_loss: 0.0026
Epoch 26/50
15/15 [==============================] - 0s 5ms/step - loss: 4.1495e-
05 - val_loss: 0.0024
Epoch 27/50
15/15 [==============================] - 0s 9ms/step - loss: 4.0457e-
05 - val_loss: 0.0025
Epoch 28/50
15/15 [==============================] - 0s 6ms/step - loss: 3.9867e-
05 - val_loss: 0.0023
Epoch 29/50
15/15 [==============================] - 0s 6ms/step - loss: 4.0358e-
05 - val_loss: 0.0024
Epoch 30/50
15/15 [==============================] - 0s 6ms/step - loss: 4.0691e-
05 - val_loss: 0.0022
Epoch 31/50
15/15 [==============================] - 0s 5ms/step - loss: 4.0590e-
05 - val_loss: 0.0023
Epoch 32/50
15/15 [==============================] - 0s 5ms/step - loss: 3.8860e-
05 - val_loss: 0.0023
Epoch 33/50
15/15 [==============================] - 0s 5ms/step - loss: 3.9360e-
05 - val_loss: 0.0024
Epoch 34/50
```

```
15/15 [==============================] - 0s 6ms/step - loss: 4.0221e-
05 - val_loss: 0.0023
Epoch 35/50
15/15 [==============================] - 0s 5ms/step - loss: 4.0014e-
05 - val_loss: 0.0025
Epoch 36/50
15/15 [==============================] - 0s 6ms/step - loss: 4.2085e-
05 - val_loss: 0.0022
Epoch 37/50
15/15 [==============================] - 0s 6ms/step - loss: 3.9952e-
05 - val_loss: 0.0024
Epoch 38/50
15/15 [==============================] - 0s 6ms/step - loss: 3.9420e-
05 - val_loss: 0.0022
Epoch 39/50
15/15 [==============================] - 0s 5ms/step - loss: 3.9722e-
05 - val_loss: 0.0024
Epoch 40/50
15/15 [==============================] - 0s 5ms/step - loss: 3.9283e-
05 - val_loss: 0.0023
Epoch 41/50
15/15 [==============================] - 0s 6ms/step - loss: 4.0183e-
05 - val_loss: 0.0024
Epoch 42/50
15/15 [==============================] - 0s 5ms/step - loss: 3.9661e-
05 - val_loss: 0.0023
Epoch 43/50
15/15 [==============================] - 0s 5ms/step - loss: 3.9982e-
05 - val_loss: 0.0022
Epoch 44/50
15/15 [==============================] - 0s 5ms/step - loss: 4.2989e-
05 - val_loss: 0.0023
Epoch 45/50
15/15 [==============================] - 0s 7ms/step - loss: 4.1282e-
05 - val_loss: 0.0023
Epoch 46/50
15/15 [==============================] - 0s 5ms/step - loss: 4.3003e-
05 - val_loss: 0.0024
Epoch 47/50
15/15 [==============================] - 0s 5ms/step - loss: 4.0453e-
05 - val_loss: 0.0022
Epoch 48/50
15/15 [==============================] - 0s 5ms/step - loss: 4.3270e-
05 - val_loss: 0.0023
Epoch 49/50
15/15 [==============================] - 0s 7ms/step - loss: 4.0093e-
05 - val_loss: 0.0023
Epoch 50/50
```

```
15/15 [==============================] - 0s 5ms/step - loss: 3.7625e-05 - val_loss: 0.0023
```

```
<keras.callbacks.History at 0x7fe1d3dd31c0>
```

```
model2_goo.evaluate(test_win_goo, test_lab_goo)
```

```
2/2 [==============================] - 0s 6ms/step - loss: 0.0065
```

```
0.006457001436501741
```

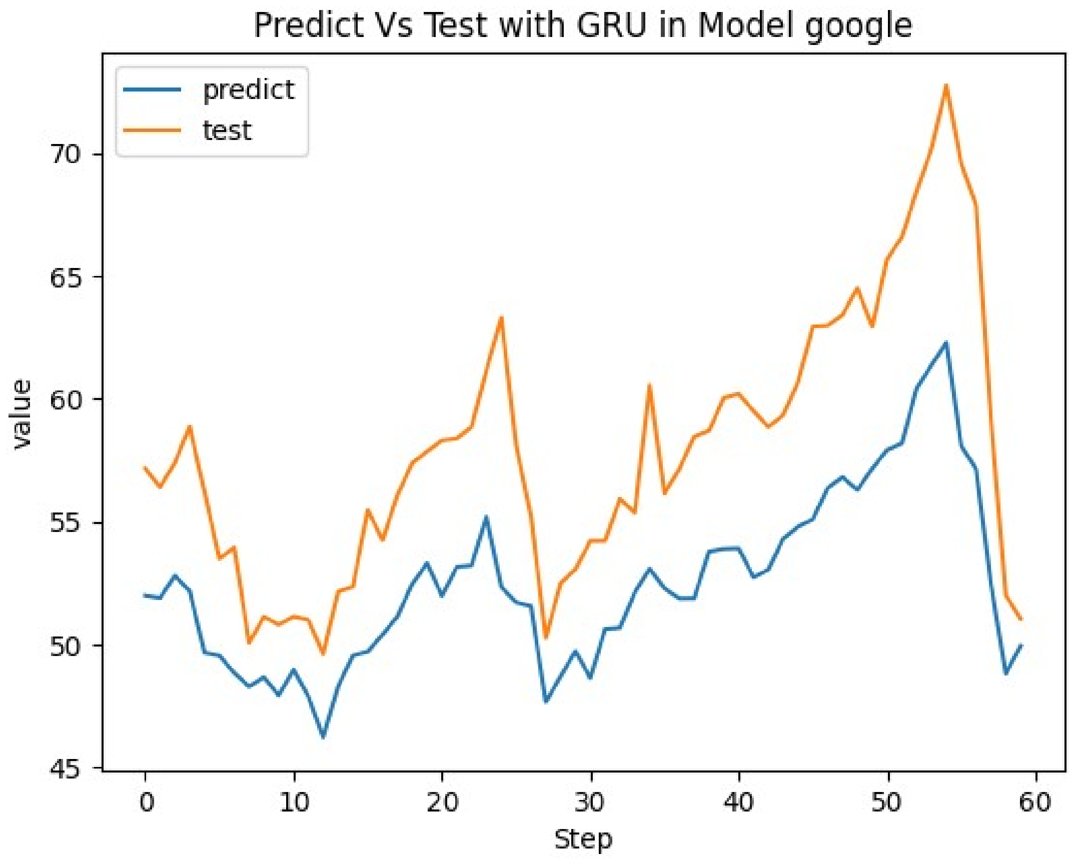
```
model2_goo_preds = model2_goo.predict(test_win_goo)
model2_goo_preds_check = scaler.inverse_transform(model2_goo_preds)
```

```
2/2 [==============================] - 0s 6ms/step
```

```
model2_goo_results = evaluate_preds(y_true=tf.squeeze(test_lab_goo), y_pred=model2_goo_preds.flatten())
google_model.loc[2] = ["GRU",model2_goo_results[0],model2_goo_results[1],model2_goo_results[2]]
print("MAE :",model2_goo_results[0])
print("RMSE :",model2_goo_results[1])
print("MAPE :",model2_goo_results[2])
```

```
MAE : 0.07354324
RMSE : 0.08035548
MAPE : 9.226133
```

```
plt.plot(model2_goo_preds_check, label='predict')
plt.plot(test_win_goo_check,label='test')
plt.title("Predict Vs Test with GRU in Model google")
plt.xlabel("Step")
plt.ylabel("value")
plt.legend()
plt.show()
```




```
model2_int =  keras.Sequential()

# imput layer next with lstm
model2_int.add(layers.GRU(units=50, input_shape=(5, 1),
activation="relu"))

# output layer
model2_int.add(layers.Dense(1,activation="linear"))

model2_int.compile(loss="mae",optimizer=tf.optimizers.Adam(learning_ra
te=0.001))

model2_int.fit(train_win_goo,
            train_lab_goo,
            epochs=50,
            verbose=1,
            batch_size=32,
            validation_data=(val_win_goo, val_lab_goo))
```

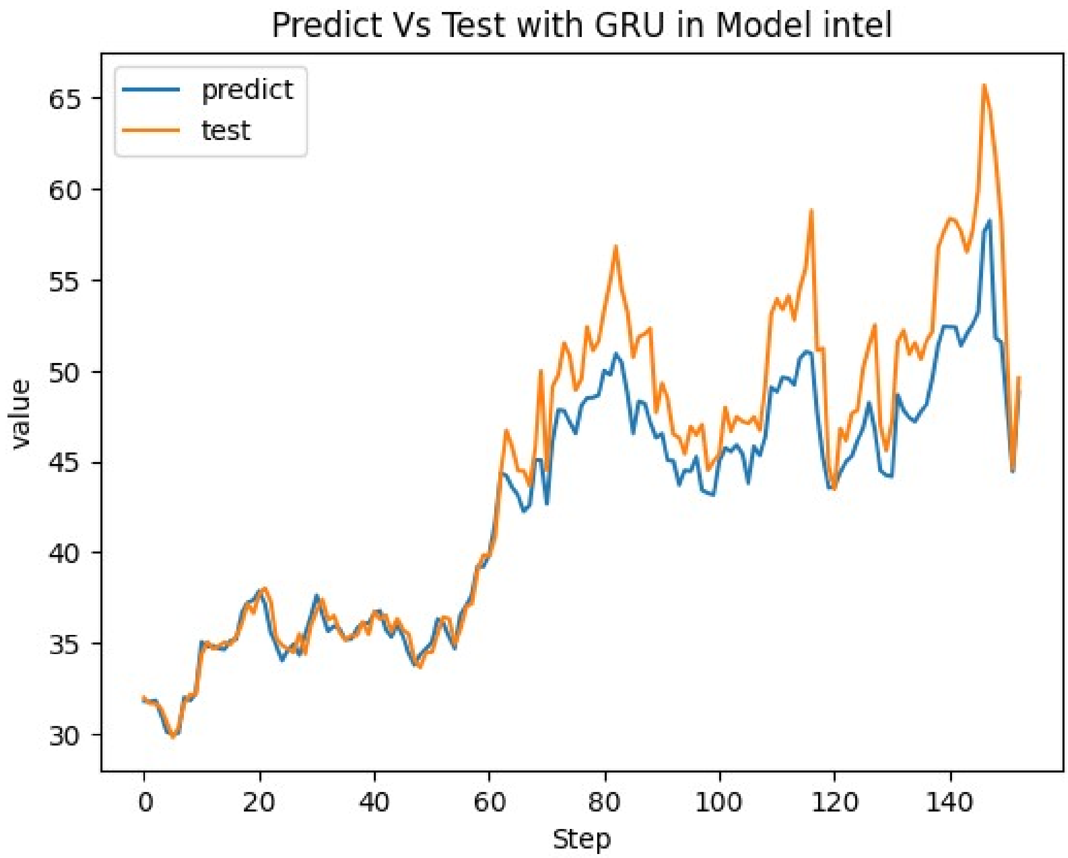
```
Epoch 1/50
15/15 [==============================] - 1s 22ms/step - loss: 0.1637 -
val_loss: 0.5182
Epoch 2/50
15/15 [==============================] - 0s 5ms/step - loss: 0.1010 -
```

```
val_loss: 0.4012
Epoch 3/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0696 -
val_loss: 0.3102
Epoch 4/50
15/15 [==============================] - 0s 5ms/step - loss: 0.0587 -
val_loss: 0.2607
Epoch 5/50
15/15 [==============================] - 0s 5ms/step - loss: 0.0443 -
val_loss: 0.1787
Epoch 6/50
15/15 [==============================] - 0s 5ms/step - loss: 0.0170 -
val_loss: 0.0252
Epoch 7/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0074 -
val_loss: 0.0516
Epoch 8/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0059 -
val_loss: 0.0370
Epoch 9/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0051 -
val_loss: 0.0397
Epoch 10/50
15/15 [==============================] - 0s 5ms/step - loss: 0.0056 -
val_loss: 0.0326
Epoch 11/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0056 -
val_loss: 0.0355
Epoch 12/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0053 -
val_loss: 0.0349
Epoch 13/50
15/15 [==============================] - 0s 5ms/step - loss: 0.0047 -
val_loss: 0.0366
Epoch 14/50
15/15 [==============================] - 0s 5ms/step - loss: 0.0050 -
val_loss: 0.0358
Epoch 15/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0049 -
val_loss: 0.0344
Epoch 16/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0049 -
val_loss: 0.0372
Epoch 17/50
15/15 [==============================] - 0s 5ms/step - loss: 0.0048 -
val_loss: 0.0288
Epoch 18/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0047 -
val_loss: 0.0362
```

```
Epoch 19/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0047 -
val_loss: 0.0302
Epoch 20/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0046 -
val_loss: 0.0324
Epoch 21/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0050 -
val_loss: 0.0302
Epoch 22/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0050 -
val_loss: 0.0300
Epoch 23/50
15/15 [==============================] - 0s 10ms/step - loss: 0.0046 -
val_loss: 0.0330
Epoch 24/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0048 -
val_loss: 0.0315
Epoch 25/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0046 -
val_loss: 0.0360
Epoch 26/50
15/15 [==============================] - 0s 11ms/step - loss: 0.0046 -
val_loss: 0.0329
Epoch 27/50
15/15 [==============================] - 0s 10ms/step - loss: 0.0050 -
val_loss: 0.0383
Epoch 28/50
15/15 [==============================] - 0s 10ms/step - loss: 0.0049 -
val_loss: 0.0336
Epoch 29/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0046 -
val_loss: 0.0343
Epoch 30/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0049 -
val_loss: 0.0326
Epoch 31/50
15/15 [==============================] - 0s 11ms/step - loss: 0.0046 -
val_loss: 0.0354
Epoch 32/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0043 -
val_loss: 0.0334
Epoch 33/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0048 -
val_loss: 0.0398
Epoch 34/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0047 -
val_loss: 0.0366
Epoch 35/50
```

```
15/15 [==============================] - 0s 9ms/step - loss: 0.0046 -
val_loss: 0.0372
Epoch 36/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0044 -
val_loss: 0.0313
Epoch 37/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0047 -
val_loss: 0.0322
Epoch 38/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0048 -
val_loss: 0.0306
Epoch 39/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0048 -
val_loss: 0.0371
Epoch 40/50
15/15 [==============================] - 0s 10ms/step - loss: 0.0047 -
val_loss: 0.0329
Epoch 41/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0048 -
val_loss: 0.0287
Epoch 42/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0049 -
val_loss: 0.0276
Epoch 43/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0054 -
val_loss: 0.0332
Epoch 44/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0053 -
val_loss: 0.0270
Epoch 45/50
15/15 [==============================] - 0s 10ms/step - loss: 0.0047 -
val_loss: 0.0354
Epoch 46/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0045 -
val_loss: 0.0316
Epoch 47/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0044 -
val_loss: 0.0372
Epoch 48/50
15/15 [==============================] - 0s 11ms/step - loss: 0.0049 -
val_loss: 0.0392
Epoch 49/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0053 -
val_loss: 0.0356
Epoch 50/50
15/15 [==============================] - 0s 10ms/step - loss: 0.0045 -
val_loss: 0.0381

<keras.callbacks.History at 0x7fe1d3bd6fb0>
```

```
model2_int.evaluate(test_win_int, test_lab_int)

5/5 [==============================] - 0s 5ms/step - loss: 0.0272

0.027195706963539124

model2_int_preds = model2_goo.predict(test_win_int)
model2_int_preds_check = scaler.inverse_transform(model2_int_preds)

5/5 [==============================] - 0s 3ms/step

model2_int_results = evaluate_preds(y_true=tf.squeeze(test_lab_int),
y_pred=model2_int_preds.flatten())
intel_model.loc[2] =
["GRU",model2_int_results[0],model2_int_results[1],model2_int_results[
2]]
print("MAE :",model2_int_results[0])
print("RMSE :",model2_int_results[1])
print("MAPE :",model2_int_results[2])

MAE : 0.02885242
RMSE : 0.039915234
MAPE : 4.256309

plt.plot(model2_int_preds_check, label='predict')
plt.plot(test_win_int_check,label='test')
plt.title("Predict Vs Test with GRU in Model intel")
plt.xlabel("Step")
plt.ylabel("value")
plt.legend()
plt.show()
```



Setelah melihat performa dari gru yang cukup kurang maka dari itu GRU tidak akan dilanjutkan.

Pada pemodelah selanjutnya akan menambahkan hidden layer di dalamm LSTM. Dimana secara teori menambahkan hidden layer akan meningkatkan akurasi model.

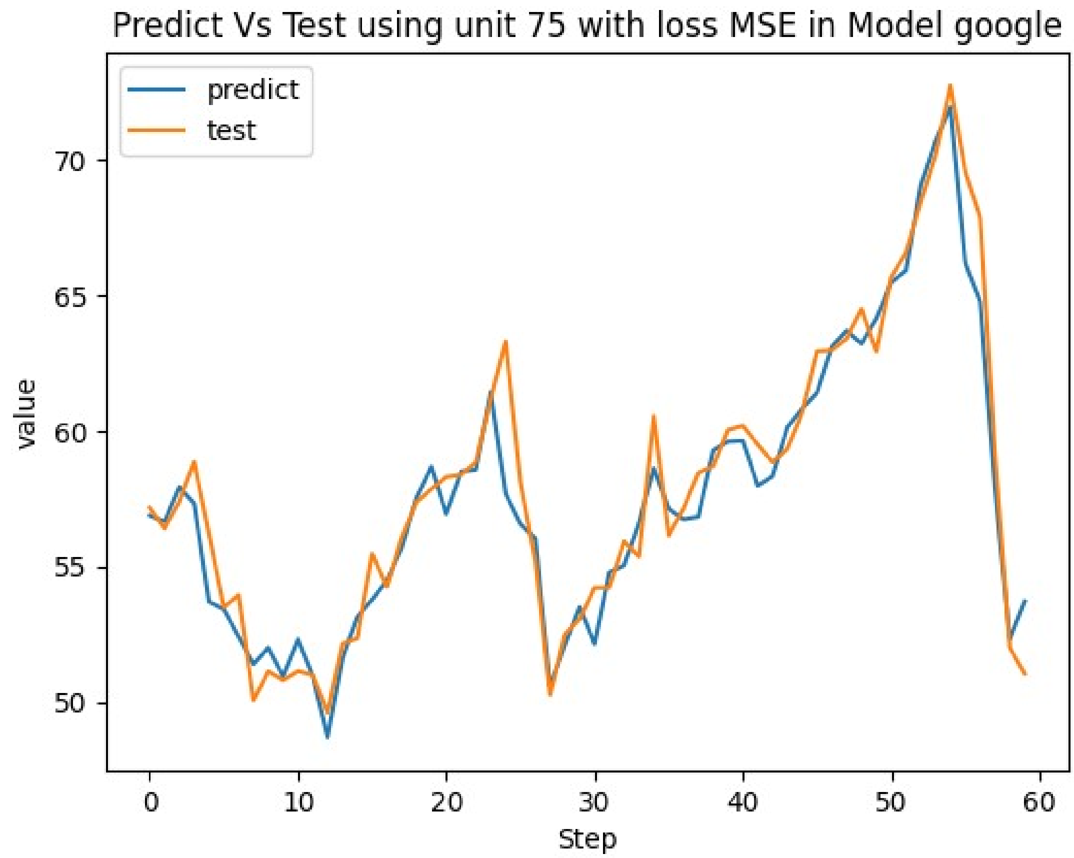
```
model3_goo =  keras.Sequential()

# imput layer next with lstm
model3_goo.add(layers.LSTM(units=75, input_shape=(5, 1),
activation="relu"))
model3_goo.add(layers.Dense(60))
# output layer
model3_goo.add(layers.Dense(1,activation="linear"))

model3_goo.compile(loss="mse",optimizer=tf.optimizers.Adam())

model3_goo.fit(train_win_goo,
            train_lab_goo,
            epochs=50,
            verbose=1,
            batch_size=32,
            validation_data=(val_win_goo, val_lab_goo))
```

```
Epoch 1/50
15/15 [==============================] - 2s 25ms/step - loss: 0.0220 -
val_loss: 0.0365
Epoch 2/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0064 -
val_loss: 0.0444
Epoch 3/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0014 -
val_loss: 8.3394e-04
Epoch 4/50
15/15 [==============================] - 0s 8ms/step - loss: 1.2744e-
04 - val_loss: 6.2626e-04
Epoch 5/50
15/15 [==============================] - 0s 7ms/step - loss: 6.3494e-
05 - val_loss: 5.2276e-04
Epoch 6/50
15/15 [==============================] - 0s 8ms/step - loss: 5.4546e-
05 - val_loss: 2.8245e-04
Epoch 7/50
15/15 [==============================] - 0s 9ms/step - loss: 5.2308e-
05 - val_loss: 3.3025e-04
Epoch 8/50
15/15 [==============================] - 0s 10ms/step - loss: 5.2101e-
05 - val_loss: 3.2280e-04
Epoch 9/50
15/15 [==============================] - 0s 7ms/step - loss: 5.0842e-
05 - val_loss: 3.0669e-04
Epoch 10/50
15/15 [==============================] - 0s 7ms/step - loss: 5.0737e-
05 - val_loss: 3.3434e-04
Epoch 11/50
15/15 [==============================] - 0s 7ms/step - loss: 4.8618e-
05 - val_loss: 2.5580e-04
Epoch 12/50
15/15 [==============================] - 0s 7ms/step - loss: 4.9020e-
05 - val_loss: 2.6838e-04
Epoch 13/50
15/15 [==============================] - 0s 7ms/step - loss: 4.6982e-
05 - val_loss: 2.6014e-04
Epoch 14/50
15/15 [==============================] - 0s 9ms/step - loss: 4.7741e-
05 - val_loss: 2.5631e-04
Epoch 15/50
15/15 [==============================] - 0s 8ms/step - loss: 4.9360e-
05 - val_loss: 2.6030e-04
Epoch 16/50
15/15 [==============================] - 0s 8ms/step - loss: 4.8687e-
05 - val_loss: 3.0392e-04
Epoch 17/50
15/15 [==============================] - 0s 8ms/step - loss: 5.0966e-
```

```
05 - val_loss: 2.2642e-04
Epoch 18/50
15/15 [==============================] - 0s 6ms/step - loss: 4.6669e-
05 - val_loss: 2.2433e-04
Epoch 19/50
15/15 [==============================] - 0s 8ms/step - loss: 4.9844e-
05 - val_loss: 2.8408e-04
Epoch 20/50
15/15 [==============================] - 0s 9ms/step - loss: 6.0507e-
05 - val_loss: 2.1042e-04
Epoch 21/50
15/15 [==============================] - 0s 8ms/step - loss: 6.1862e-
05 - val_loss: 2.7618e-04
Epoch 22/50
15/15 [==============================] - 0s 7ms/step - loss: 5.3292e-
05 - val_loss: 2.3950e-04
Epoch 23/50
15/15 [==============================] - 0s 6ms/step - loss: 4.8884e-
05 - val_loss: 2.0980e-04
Epoch 24/50
15/15 [==============================] - 0s 7ms/step - loss: 5.9577e-
05 - val_loss: 3.4007e-04
Epoch 25/50
15/15 [==============================] - 0s 8ms/step - loss: 6.6629e-
05 - val_loss: 2.1460e-04
Epoch 26/50
15/15 [==============================] - 0s 6ms/step - loss: 5.9682e-
05 - val_loss: 2.1101e-04
Epoch 27/50
15/15 [==============================] - 0s 8ms/step - loss: 5.5705e-
05 - val_loss: 2.5394e-04
Epoch 28/50
15/15 [==============================] - 0s 7ms/step - loss: 4.5635e-
05 - val_loss: 2.0982e-04
Epoch 29/50
15/15 [==============================] - 0s 7ms/step - loss: 4.8849e-
05 - val_loss: 2.9674e-04
Epoch 30/50
15/15 [==============================] - 0s 7ms/step - loss: 5.2881e-
05 - val_loss: 2.3751e-04
Epoch 31/50
15/15 [==============================] - 0s 8ms/step - loss: 5.5712e-
05 - val_loss: 2.0867e-04
Epoch 32/50
15/15 [==============================] - 0s 6ms/step - loss: 4.9545e-
05 - val_loss: 2.9540e-04
Epoch 33/50
15/15 [==============================] - 0s 8ms/step - loss: 4.8951e-
05 - val_loss: 2.0835e-04
Epoch 34/50
```

```
15/15 [==============================] - 0s 7ms/step - loss: 4.8041e-
05 - val_loss: 2.0837e-04
Epoch 35/50
15/15 [==============================] - 0s 6ms/step - loss: 4.6961e-
05 - val_loss: 2.1455e-04
Epoch 36/50
15/15 [==============================] - 0s 8ms/step - loss: 4.8445e-
05 - val_loss: 2.0814e-04
Epoch 37/50
15/15 [==============================] - 0s 7ms/step - loss: 5.8690e-
05 - val_loss: 3.0390e-04
Epoch 38/50
15/15 [==============================] - 0s 8ms/step - loss: 5.5004e-
05 - val_loss: 2.0792e-04
Epoch 39/50
15/15 [==============================] - 0s 7ms/step - loss: 5.2100e-
05 - val_loss: 2.0704e-04
Epoch 40/50
15/15 [==============================] - 0s 7ms/step - loss: 4.8899e-
05 - val_loss: 2.1867e-04
Epoch 41/50
15/15 [==============================] - 0s 6ms/step - loss: 5.3002e-
05 - val_loss: 2.8738e-04
Epoch 42/50
15/15 [==============================] - 0s 8ms/step - loss: 4.6420e-
05 - val_loss: 2.0666e-04
Epoch 43/50
15/15 [==============================] - 0s 8ms/step - loss: 4.4724e-
05 - val_loss: 2.2201e-04
Epoch 44/50
15/15 [==============================] - 0s 7ms/step - loss: 4.4446e-
05 - val_loss: 2.1541e-04
Epoch 45/50
15/15 [==============================] - 0s 7ms/step - loss: 4.8689e-
05 - val_loss: 2.2671e-04
Epoch 46/50
15/15 [==============================] - 0s 8ms/step - loss: 4.5709e-
05 - val_loss: 2.0490e-04
Epoch 47/50
15/15 [==============================] - 0s 8ms/step - loss: 4.6699e-
05 - val_loss: 2.0457e-04
Epoch 48/50
15/15 [==============================] - 0s 7ms/step - loss: 5.1089e-
05 - val_loss: 2.6670e-04
Epoch 49/50
15/15 [==============================] - 0s 6ms/step - loss: 4.8388e-
05 - val_loss: 2.2396e-04
Epoch 50/50
```

```
15/15 [==============================] - 0s 8ms/step - loss: 4.4985e-05 - val_loss: 2.3585e-04

<keras.callbacks.History at 0x7fe1c9f73190>

model3_goo.evaluate(test_win_goo, test_lab_goo)

2/2 [==============================] - 0s 9ms/step - loss: 5.1074e-04

0.0005107354372739792

model3_goo_preds = model3_goo.predict(test_win_goo)
model3_goo_preds_check = scaler.inverse_transform(model3_goo_preds)

2/2 [==============================] - 0s 4ms/step

model3_goo_results = 
evaluate_preds(y_true=tf.squeeze(test_lab_goo),y_pred=model3_goo_preds
.flatten())
google_model.loc[3] = ["Dense 
60",model3_goo_results[0],model3_goo_results[1],model3_goo_results[2]]
print("MAE :",model3_goo_results[0])
print("RMSE :",model3_goo_results[1])
print("MAPE :",model3_goo_results[2])

MAE : 0.017043483
RMSE : 0.02259945
MAPE : 2.2203598

plt.plot(model3_goo_preds_check, label='predict')
plt.plot(test_win_goo_check,label='test')
plt.title("Predict Vs Test using unit 75 with loss MSE in Model 
google")
plt.xlabel("Step")
plt.ylabel("value")
plt.legend()
plt.show()
```



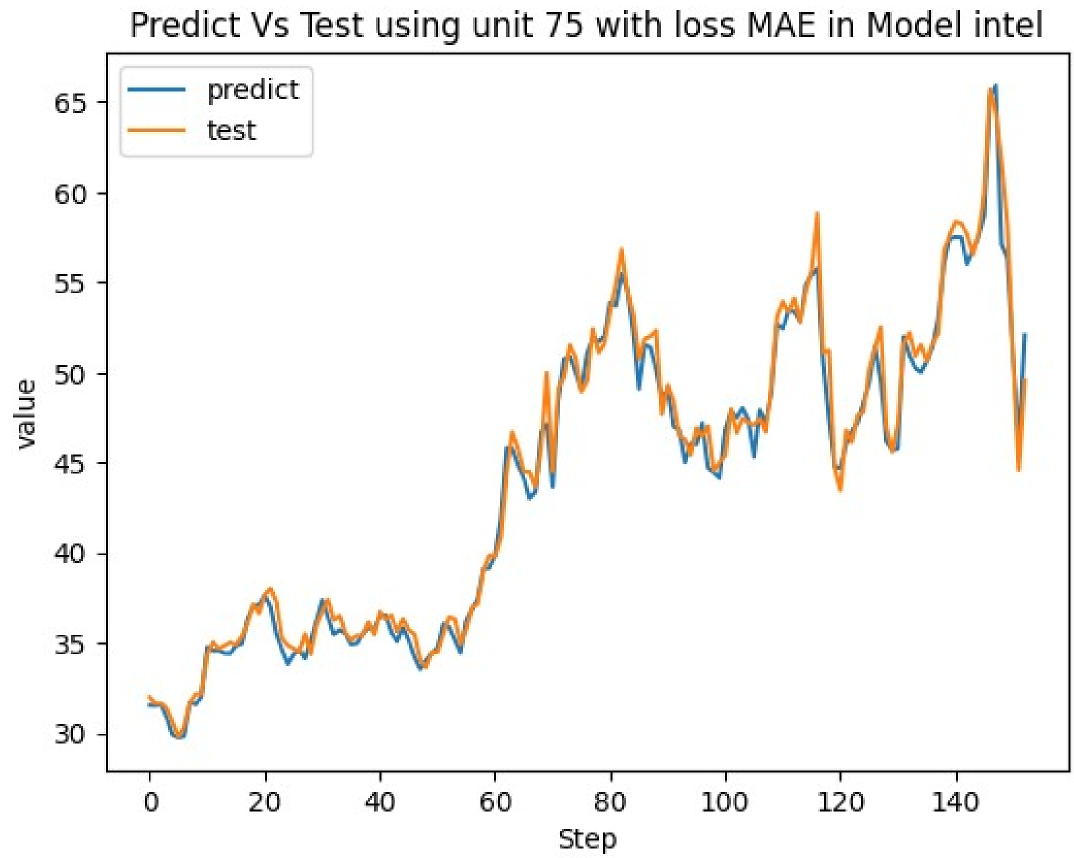

```
model3_int =  keras.Sequential()

# imput layer next with lstm
model3_int.add(layers.LSTM(units=75, input_shape=(5, 1),
activation="relu"))
model3_int.add(layers.Dense(60))
# output layer
model3_int.add(layers.Dense(1,activation="linear"))

model3_int.compile(loss="mae",optimizer=tf.optimizers.Adam())

model3_int.fit(train_win_goo,
            train_lab_goo,
            epochs=50,
            verbose=1,
            batch_size=32,
            validation_data=(val_win_goo, val_lab_goo))

Epoch 1/50
15/15 [==============================] - 2s 41ms/step - loss: 0.1103 -
val_loss: 0.1967
Epoch 2/50
15/15 [==============================] - 0s 12ms/step - loss: 0.0551 -
val_loss: 0.1272
```

```
Epoch 3/50
15/15 [==============================] - 0s 10ms/step - loss: 0.0171 -
val_loss: 0.0308
Epoch 4/50
15/15 [==============================] - 0s 10ms/step - loss: 0.0077 -
val_loss: 0.0110
Epoch 5/50
15/15 [==============================] - 0s 10ms/step - loss: 0.0058 -
val_loss: 0.0116
Epoch 6/50
15/15 [==============================] - 0s 10ms/step - loss: 0.0055 -
val_loss: 0.0122
Epoch 7/50
15/15 [==============================] - 0s 11ms/step - loss: 0.0061 -
val_loss: 0.0149
Epoch 8/50
15/15 [==============================] - 0s 10ms/step - loss: 0.0059 -
val_loss: 0.0181
Epoch 9/50
15/15 [==============================] - 0s 12ms/step - loss: 0.0053 -
val_loss: 0.0108
Epoch 10/50
15/15 [==============================] - 0s 11ms/step - loss: 0.0060 -
val_loss: 0.0111
Epoch 11/50
15/15 [==============================] - 0s 11ms/step - loss: 0.0065 -
val_loss: 0.0121
Epoch 12/50
15/15 [==============================] - 0s 11ms/step - loss: 0.0053 -
val_loss: 0.0137
Epoch 13/50
15/15 [==============================] - 0s 14ms/step - loss: 0.0057 -
val_loss: 0.0122
Epoch 14/50
15/15 [==============================] - 0s 12ms/step - loss: 0.0048 -
val_loss: 0.0122
Epoch 15/50
15/15 [==============================] - 0s 12ms/step - loss: 0.0061 -
val_loss: 0.0150
Epoch 16/50
15/15 [==============================] - 0s 11ms/step - loss: 0.0059 -
val_loss: 0.0122
Epoch 17/50
15/15 [==============================] - 0s 12ms/step - loss: 0.0051 -
val_loss: 0.0129
Epoch 18/50
15/15 [==============================] - 0s 11ms/step - loss: 0.0053 -
val_loss: 0.0133
Epoch 19/50
```

```
15/15 [==============================] - 0s 11ms/step - loss: 0.0051 -
val_loss: 0.0118
Epoch 20/50
15/15 [==============================] - 0s 11ms/step - loss: 0.0051 -
val_loss: 0.0109
Epoch 21/50
15/15 [==============================] - 0s 10ms/step - loss: 0.0049 -
val_loss: 0.0107
Epoch 22/50
15/15 [==============================] - 0s 11ms/step - loss: 0.0050 -
val_loss: 0.0108
Epoch 23/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0052 -
val_loss: 0.0111
Epoch 24/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0055 -
val_loss: 0.0121
Epoch 25/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0050 -
val_loss: 0.0107
Epoch 26/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0048 -
val_loss: 0.0143
Epoch 27/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0051 -
val_loss: 0.0106
Epoch 28/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0052 -
val_loss: 0.0153
Epoch 29/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0051 -
val_loss: 0.0105
Epoch 30/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0055 -
val_loss: 0.0125
Epoch 31/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0049 -
val_loss: 0.0137
Epoch 32/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0051 -
val_loss: 0.0125
Epoch 33/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0050 -
val_loss: 0.0115
Epoch 34/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0048 -
val_loss: 0.0106
Epoch 35/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0049 -
```

```
val_loss: 0.0107
Epoch 36/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0060 -
val_loss: 0.0131
Epoch 37/50
15/15 [==============================] - 0s 6ms/step - loss: 0.0047 -
val_loss: 0.0106
Epoch 38/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0050 -
val_loss: 0.0112
Epoch 39/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0049 -
val_loss: 0.0119
Epoch 40/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0056 -
val_loss: 0.0110
Epoch 41/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0052 -
val_loss: 0.0110
Epoch 42/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0053 -
val_loss: 0.0114
Epoch 43/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0052 -
val_loss: 0.0139
Epoch 44/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0054 -
val_loss: 0.0205
Epoch 45/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0075 -
val_loss: 0.0107
Epoch 46/50
15/15 [==============================] - 0s 7ms/step - loss: 0.0049 -
val_loss: 0.0133
Epoch 47/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0049 -
val_loss: 0.0121
Epoch 48/50
15/15 [==============================] - 0s 9ms/step - loss: 0.0054 -
val_loss: 0.0112
Epoch 49/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0049 -
val_loss: 0.0104
Epoch 50/50
15/15 [==============================] - 0s 8ms/step - loss: 0.0047 -
val_loss: 0.0118

<keras.callbacks.History at 0x7fe1cacb8580>

model3_int.evaluate(test_win_int, test_lab_int)
```

```
5/5 [==============================] - 0s 4ms/step - loss: 0.0126

0.012599618174135685

model3_int_preds = model3_goo.predict(test_win_int)
model3_int_preds_check = scaler.inverse_transform(model3_int_preds)

5/5 [==============================] - 0s 3ms/step

model3_int_results = evaluate_preds(y_true=tf.squeeze(test_lab_int),
y_pred=model3_int_preds.flatten())
intel_model.loc[3] = ["Dense
60",model3_int_results[0],model3_int_results[1],model3_int_results[2]]
print("MAE :",model3_int_results[0])
print("RMSE :",model3_int_results[1])
print("MAPE :",model3_int_results[2])

MAE : 0.012469493
RMSE : 0.0182536
MAPE : 2.0062723

plt.plot(model3_int_preds_check, label='predict')
plt.plot(test_win_int_check,label='test')
plt.title("Predict Vs Test using unit 75 with loss MAE in Model
intel")
plt.xlabel("Step")
plt.ylabel("value")
plt.legend()
plt.show()
```



[LO 3, LO 4, 5 poin] Lakukan evaluasi unjuk kerja kedua arsitektur di atas pada test set dengan mencari nilai RMSE, MAE dan MAPE. Dan berikan penjelasan mengenai hasilnya dengan rinci.

```
google_model
```

	Model	MAE	RMSE	MAPE
0	Base	0.058428	0.063507	7.371774
1	Unit 75 with MSE	0.017916	0.024287	2.341978
2	GRU	0.073543	0.080355	9.226133
3	Dense 60	0.017043	0.022599	2.220360

Penjelasan dari data diatas :

- Pada model base yang terbentuk oleh data google dapat dilihat dengan hanya base saja tidak ada optimizer dan menghasil hasil yang cukup baik, yaitu error 7.3 persen.
- Dari sini lah model dilakukan revisi kembali agar menghasilkan output yang lebih baik dengan mengati unit dari LSTM menjadi 75, selain itu menggati loss function mejadi MSE. Dengan mengganti loss dan unit error dari model menuru menjadi 2.3 persen.
- Pada model 3 akan mencoba merubah algoritma dari LSTM menjadi GRU, tetapi sayangnya cara ini tidak berhasil dimana error model naik menjadi 10 persen.

- Pada modified terkahir yang mana model terbaik. Pada model ini di tambahkan dense yang mana menjadi hidden layer dengan 60 unit. Disini error dari model menurun hingga 2 persen. Jika dilihat dengan penambahan dense akan membuat model lebih mengenal data, sehingga error dari model menurun.

💡 Kesimpulan yang didapatkan base model yang sudah cukup baik tinggal penambahan fitur-fitur. fitur yang ditambahakan pada model adalah optimizer adam, menaikan unit LSTM dari 50 kek 75, dan penambahan hidden layer pada model dengan unit 60, dan mengganti loss function menjadi MSE. Dengan cara tersebut menurunkan error model hingga 1 persen dari base.

```
intel_model
```

	Model	MAE	RMSE	MAPE
0	Base	0.029388	0.037501	4.477377
1	Unit 75 with MAE	0.012818	0.021113	2.083454
2	GRU	0.028852	0.039915	4.256309
3	Dense 60	0.012469	0.018254	2.006272

Penjelasan dari data diatas :

- Pada awalnya saat menggunaknan RNN dengan metode LSTM didapatkan hasil untuk data intel cukup memuaskan pada bagian MAE, RMSE, MAPE. hal ini dapat dilihat dari error model 3 persen.
- Untuk mningkatkan model lebih dalam dengan menggunakan menaikan unit pada LSTM, tetapi disini loss function tidak diubah seperti data google. Pada kasus ini loss function dengan MAE menghasilkan hasil yang lebih baik daripada MSE. Hasil yang didapatkan hampir menurunkan error model 1 persen.
- Selanjutnya pada pengetesan ini sama seperti google menggunkan GRU, tetapi cara ini tidaklah berhasil, karena GRU tidak memberikan error yang lebih baik dari pada model diatasnya.
- Pada percobaan terakhir dimana merupakan best model pada tahapan ini sama seperti model google akan ditambahan hidden layer dengan 60. Dengan menabahkan hidden layer membuat model lebih mengenal data dan menurunkan error model.

💡 Kesimpulan yang didapatkan pada model ini adalah model dari data intel pada saat baseline sudah cukup baik. Sehingga pada saat modifikasi tidak terlalu membutuhkan banyak hal. Modifikasi yang digunakan menaikan unit dari LSTM, penggunaan optimizer, dan penggunaan hidden layer.