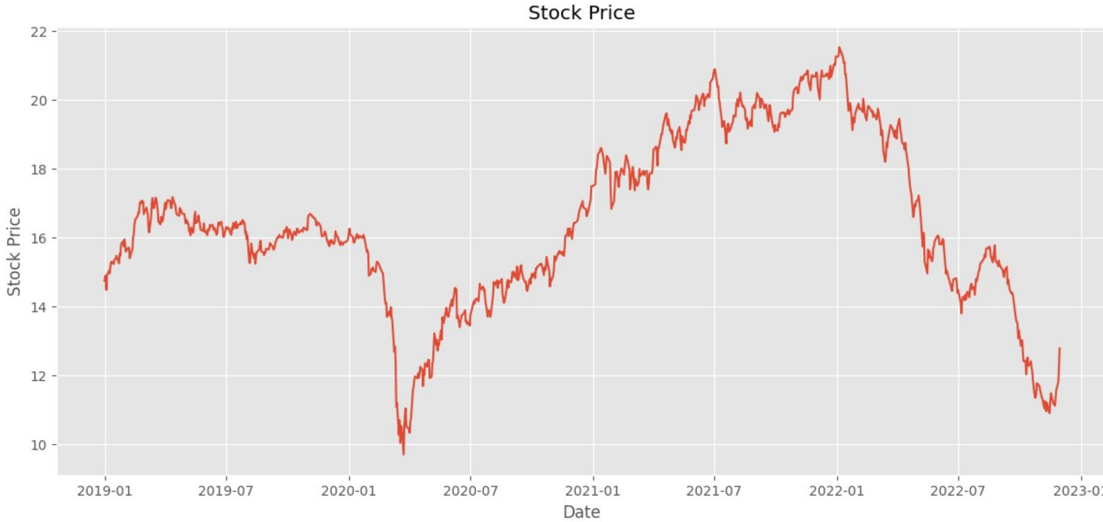
```python
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from sklearn.preprocessing import MinMaxScaler
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout,GRU
from keras import optimizers
from sklearn.metrics import mean_squared_error

seed = 1234
np.random.seed(seed)
plt.style.use('ggplot')

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

data_raw = pd.read_csv("VNM.csv", index_col="Date",
parse_dates=["Date"])
data_raw = data_raw.dropna()

# use feature 'Date' & 'Close'
dataset = pd.DataFrame(data_raw['Close'])
print(' Count row of data: ',len(dataset))

fig = plt.figure(figsize=(14, 6))
plt.plot(dataset)
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m"))
plt.title('Stock Price')
plt.show()
```

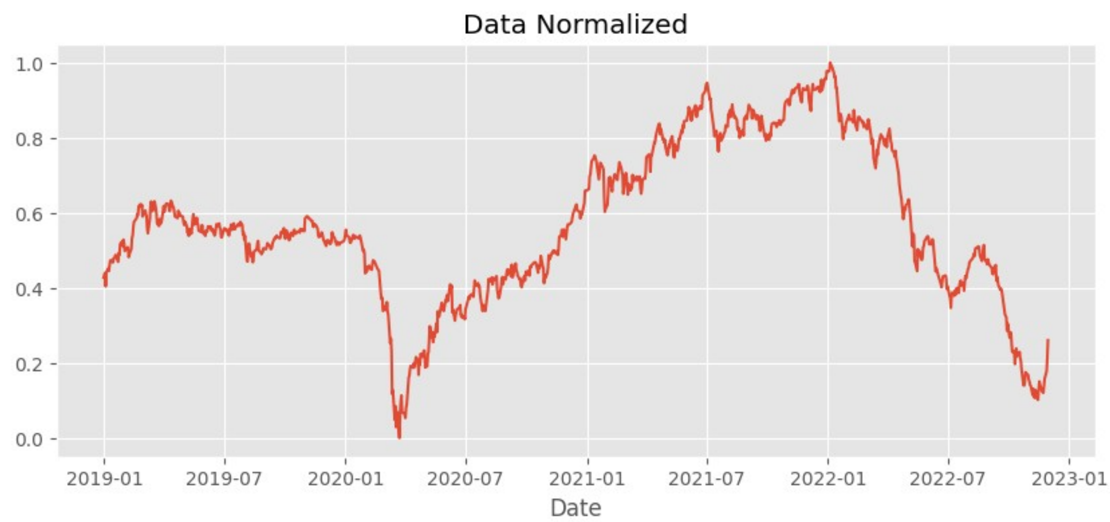
```
 Count row of data:  988
```




```
#Min-Max Normalization
dataset_norm = dataset.copy()
dataset[['Close']]
scaler = MinMaxScaler()
dataset_norm['Close'] = scaler.fit_transform(dataset[['Close']])
dataset_norm
```

```
               Close
Date
2018-12-31  0.426881
2019-01-02  0.439560
2019-01-03  0.404057
2019-01-04  0.440406
2019-01-07  0.451395
...              ...
2022-11-23  0.120034
2022-11-25  0.158073
2022-11-28  0.177515
2022-11-29  0.211327
2022-11-30  0.260355

[988 rows x 1 columns]
```

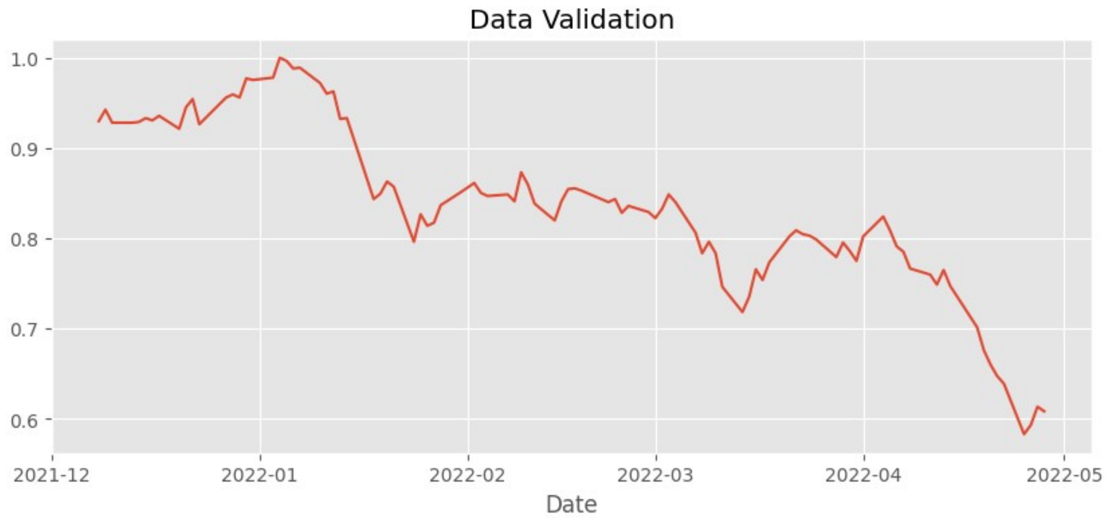
```
fig = plt.figure(figsize=(10, 4))
plt.plot(dataset_norm)
plt.xlabel('Date')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m"))
plt.title('Data Normalized')
plt.show()
```



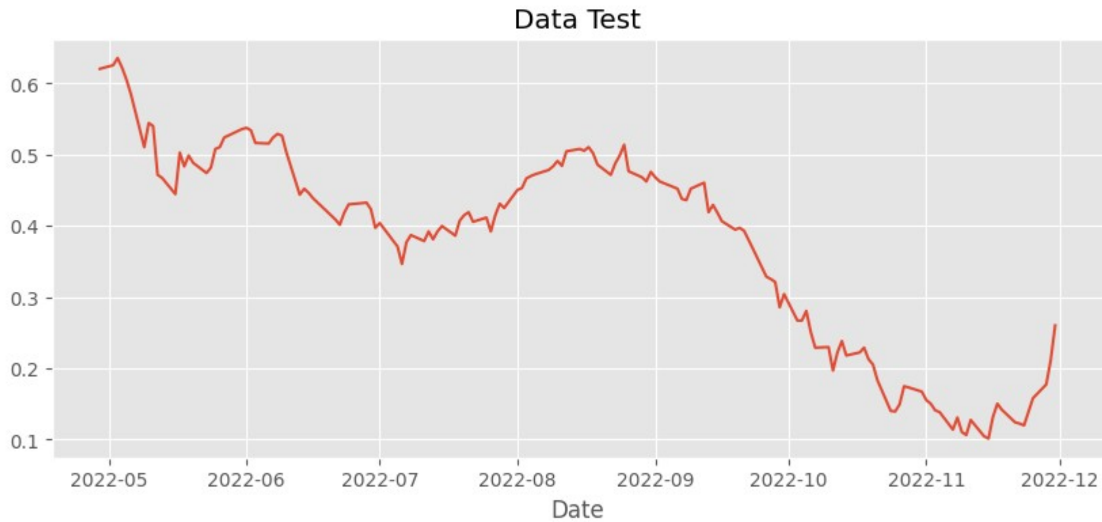

```
# Partition data into data train, val & test
totaldata = dataset.values
totaldatatrain = int(len(totaldata)*0.75)
totaldataval = int(len(totaldata)*0.1)
totaldatatest = int(len(totaldata)*0.15)

# Store data into each partition
training_set = dataset_norm[0:totaldatatrain]
val_set=dataset_norm[totaldatatrain:totaldatatrain+totaldataval]
test_set = dataset_norm[totaldatatrain+totaldataval:]

# graph of data training
fig = plt.figure(figsize=(10, 4))
plt.plot(training_set)
plt.xlabel('Date')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m"))
plt.title('Data Training')
plt.show()
```


```
# graph of data validation
fig = plt.figure(figsize=(10, 4))
plt.plot(val_set)
plt.xlabel('Date')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m"))
plt.title('Data Validation')
val_set
```

```
               Close
Date
2021-12-08  0.929839
2021-12-09  0.942519
2021-12-10  0.928149
2021-12-13  0.928149
2021-12-14  0.928994
...              ...
2022-04-22  0.639053
2022-04-25  0.583263
2022-04-26  0.593406
2022-04-27  0.613694
2022-04-28  0.608622

[98 rows x 1 columns]
```




```
# graph of data test
fig = plt.figure(figsize=(10, 4))
plt.plot(test_set)
plt.xlabel('Date')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m"))
plt.title('Data Test')
plt.show()
test_set
```




```
               Close
Date
2022-04-29  0.620456
2022-05-02  0.625528
2022-05-03  0.635672
2022-05-04  0.622147
2022-05-05  0.605241
...              ...
```

```
2022-11-23  0.120034
2022-11-25  0.158073
2022-11-28  0.177515
2022-11-29  0.211327
2022-11-30  0.260355

[149 rows x 1 columns]
```

```python
# Initiaton value of lag
lag = 2
# sliding windows function
def create_sliding_windows(data,len_data,lag):
    x=[]
    y=[]
    for i in range(lag,len_data):
        x.append(data[i-lag:i,0])
        y.append(data[i,0])
    return np.array(x),np.array(y)

# Formating data into array for create sliding windows
array_training_set = np.array(training_set)
array_val_set = np.array(val_set)
array_test_set = np.array(test_set)

# Create sliding windows into training data
x_train, y_train =
create_sliding_windows(array_training_set,len(array_training_set),
lag)
x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))
# Create sliding windows into validation data
x_val,y_val =
create_sliding_windows(array_val_set,len(array_val_set),lag)
x_val = np.reshape(x_val, (x_val.shape[0],x_val.shape[1],1))
# Create sliding windows into test data
x_test,y_test =
create_sliding_windows(array_test_set,len(array_test_set),lag)
x_test = np.reshape(x_test, (x_test.shape[0],x_test.shape[1],1))

# Hyperparameters
learning_rate = 0.0001
hidden_unit = 64
batch_size= 32
epoch = 100

# Architecture Gated Recurrent Unit
regressorGRU = Sequential()

# First GRU layer with dropout
regressorGRU.add(GRU(units=hidden_unit, return_sequences=True,
input_shape=(x_train.shape[1],1), activation = 'relu'))
```

```
regressorGRU.add(Dropout(0.2))
# Second GRU layer with dropout
regressorGRU.add(GRU(units=hidden_unit, return_sequences=True,
activation = 'relu'))
regressorGRU.add(Dropout(0.2))
# Third GRU layer with dropout
regressorGRU.add(GRU(units=hidden_unit, return_sequences=False,
activation = 'relu'))
regressorGRU.add(Dropout(0.2))

# Output layer
regressorGRU.add(Dense(units=1))

# Compiling the Gated Recurrent Unit
regressorGRU.compile(optimizer=tf.keras.optimizers.Adam(lr=learning_ra
te),loss='mean_squared_error')

# Fitting ke data training dan data validation
pred = regressorGRU.fit(x_train, y_train,
validation_data=(x_val,y_val), batch_size=batch_size, epochs=epoch)
```

```
WARNING:absl:`lr` is deprecated, please use `learning_rate` instead,
or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.

Epoch 1/100
24/24 [==============================] - 6s 34ms/step - loss: 0.2279 -
val_loss: 0.1093
Epoch 2/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0322 -
val_loss: 0.0748
Epoch 3/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0243 -
val_loss: 0.0213
Epoch 4/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0163 -
val_loss: 0.0096
Epoch 5/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0095 -
val_loss: 0.0061
Epoch 6/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0075 -
val_loss: 9.2571e-04
Epoch 7/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0077 -
val_loss: 0.0025
Epoch 8/100
24/24 [==============================] - 0s 12ms/step - loss: 0.0072 -
val_loss: 8.9729e-04
Epoch 9/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0065 -
```

```
val_loss: 0.0010
Epoch 10/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0059 -
val_loss: 8.8966e-04
Epoch 11/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0055 -
val_loss: 0.0020
Epoch 12/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0053 -
val_loss: 8.6860e-04
Epoch 13/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0055 -
val_loss: 0.0012
Epoch 14/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0055 -
val_loss: 0.0022
Epoch 15/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0053 -
val_loss: 0.0016
Epoch 16/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0051 -
val_loss: 0.0016
Epoch 17/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0050 -
val_loss: 8.0628e-04
Epoch 18/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0050 -
val_loss: 8.5653e-04
Epoch 19/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0049 -
val_loss: 0.0015
Epoch 20/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0051 -
val_loss: 0.0013
Epoch 21/100
24/24 [==============================] - 0s 9ms/step - loss: 0.0044 -
val_loss: 0.0015
Epoch 22/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0046 -
val_loss: 7.5642e-04
Epoch 23/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0046 -
val_loss: 9.4107e-04
Epoch 24/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0039 -
val_loss: 0.0011
Epoch 25/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0043 -
val_loss: 9.7396e-04
Epoch 26/100
```

```
24/24 [==============================] - 0s 6ms/step - loss: 0.0044 -
val_loss: 0.0016
Epoch 27/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0040 -
val_loss: 0.0014
Epoch 28/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0041 -
val_loss: 9.2762e-04
Epoch 29/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0043 -
val_loss: 0.0016
Epoch 30/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0042 -
val_loss: 0.0016
Epoch 31/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0047 -
val_loss: 0.0014
Epoch 32/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0035 -
val_loss: 9.1640e-04
Epoch 33/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0040 -
val_loss: 8.8191e-04
Epoch 34/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0037 -
val_loss: 0.0011
Epoch 35/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0038 -
val_loss: 0.0027
Epoch 36/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0052 -
val_loss: 0.0012
Epoch 37/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0035 -
val_loss: 0.0013
Epoch 38/100
24/24 [==============================] - 0s 10ms/step - loss: 0.0036 -
val_loss: 0.0011
Epoch 39/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0034 -
val_loss: 0.0013
Epoch 40/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0036 -
val_loss: 0.0013
Epoch 41/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0034 -
val_loss: 9.4422e-04
Epoch 42/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0037 -
val_loss: 0.0029
```

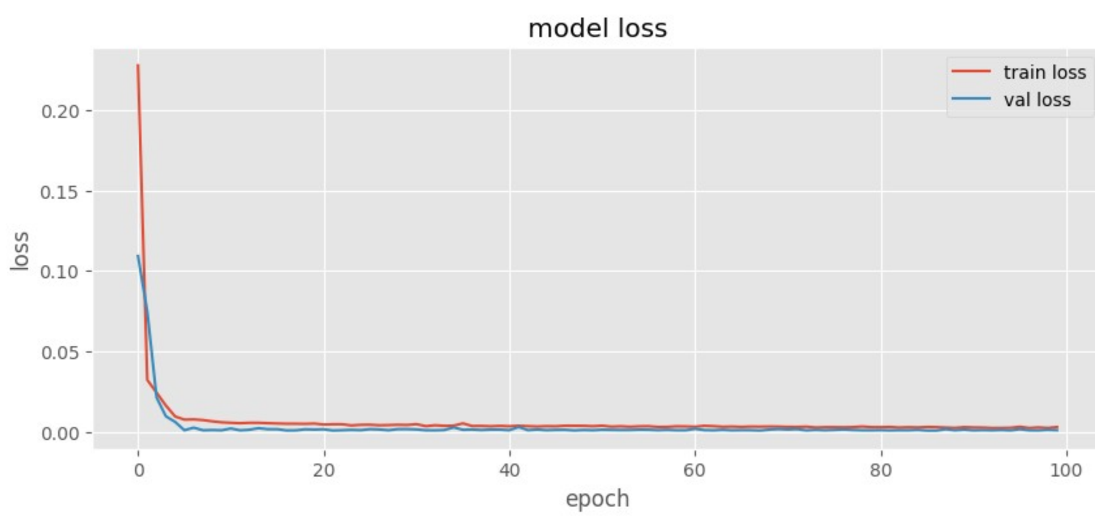
```
Epoch 43/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0035 -
val_loss: 9.7760e-04
Epoch 44/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0033 -
val_loss: 0.0014
Epoch 45/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0035 -
val_loss: 9.6170e-04
Epoch 46/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0034 -
val_loss: 0.0011
Epoch 47/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0037 -
val_loss: 0.0012
Epoch 48/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0037 -
val_loss: 8.1504e-04
Epoch 49/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0037 -
val_loss: 0.0011
Epoch 50/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0034 -
val_loss: 9.2272e-04
Epoch 51/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0038 -
val_loss: 0.0012
Epoch 52/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0032 -
val_loss: 0.0011
Epoch 53/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0034 -
val_loss: 0.0011
Epoch 54/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0031 -
val_loss: 0.0011
Epoch 55/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0034 -
val_loss: 0.0013
Epoch 56/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0034 -
val_loss: 0.0012
Epoch 57/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0029 -
val_loss: 9.8277e-04
Epoch 58/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0030 -
val_loss: 0.0012
Epoch 59/100
24/24 [==============================] - 0s 11ms/step - loss: 0.0034 -
```

```
val_loss: 9.3750e-04
Epoch 60/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0033 -
val_loss: 9.2434e-04
Epoch 61/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0031 -
val_loss: 0.0018
Epoch 62/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0037 -
val_loss: 0.0010
Epoch 63/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0035 -
val_loss: 8.9183e-04
Epoch 64/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0031 -
val_loss: 0.0012
Epoch 65/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0032 -
val_loss: 8.9978e-04
Epoch 66/100
24/24 [==============================] - 0s 10ms/step - loss: 0.0030 -
val_loss: 0.0010
Epoch 67/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0032 -
val_loss: 9.7271e-04
Epoch 68/100
24/24 [==============================] - 0s 9ms/step - loss: 0.0032 -
val_loss: 7.6197e-04
Epoch 69/100
24/24 [==============================] - 0s 9ms/step - loss: 0.0032 -
val_loss: 0.0013
Epoch 70/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0032 -
val_loss: 0.0016
Epoch 71/100
24/24 [==============================] - 0s 10ms/step - loss: 0.0030 -
val_loss: 0.0014
Epoch 72/100
24/24 [==============================] - 0s 12ms/step - loss: 0.0030 -
val_loss: 0.0016
Epoch 73/100
24/24 [==============================] - 0s 9ms/step - loss: 0.0031 -
val_loss: 8.6280e-04
Epoch 74/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0026 -
val_loss: 0.0012
Epoch 75/100
24/24 [==============================] - 0s 10ms/step - loss: 0.0028 -
val_loss: 9.4254e-04
Epoch 76/100
```

```
24/24 [==============================] - 0s 11ms/step - loss: 0.0028 -
val_loss: 0.0012
Epoch 77/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0027 -
val_loss: 0.0014
Epoch 78/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0029 -
val_loss: 0.0011
Epoch 79/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0032 -
val_loss: 9.2356e-04
Epoch 80/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0028 -
val_loss: 8.9974e-04
Epoch 81/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0028 -
val_loss: 9.9361e-04
Epoch 82/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0030 -
val_loss: 8.3149e-04
Epoch 83/100
24/24 [==============================] - 0s 8ms/step - loss: 0.0026 -
val_loss: 9.5829e-04
Epoch 84/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0028 -
val_loss: 9.3956e-04
Epoch 85/100
24/24 [==============================] - 0s 6ms/step - loss: 0.0027 -
val_loss: 0.0012
Epoch 86/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0030 -
val_loss: 7.2997e-04
Epoch 87/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0028 -
val_loss: 7.2829e-04
Epoch 88/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0026 -
val_loss: 0.0016
Epoch 89/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0024 -
val_loss: 9.3320e-04
Epoch 90/100
24/24 [==============================] - 0s 9ms/step - loss: 0.0028 -
val_loss: 0.0013
Epoch 91/100
24/24 [==============================] - 0s 9ms/step - loss: 0.0026 -
val_loss: 8.6551e-04
Epoch 92/100
24/24 [==============================] - 0s 9ms/step - loss: 0.0026 -
val_loss: 0.0010
```

```
Epoch 93/100
24/24 [==============================] - 0s 12ms/step - loss: 0.0023 -
val_loss: 8.8743e-04
Epoch 94/100
24/24 [==============================] - 0s 12ms/step - loss: 0.0024 -
val_loss: 0.0011
Epoch 95/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0025 -
val_loss: 8.4602e-04
Epoch 96/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0031 -
val_loss: 0.0016
Epoch 97/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0023 -
val_loss: 9.1249e-04
Epoch 98/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0027 -
val_loss: 8.5656e-04
Epoch 99/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0024 -
val_loss: 0.0012
Epoch 100/100
24/24 [==============================] - 0s 7ms/step - loss: 0.0029 -
val_loss: 9.8275e-04
```

```python
fig = plt.figure(figsize=(10, 4))
plt.plot(pred.history['loss'], label='train loss')
plt.plot(pred.history['val_loss'], label='val loss')
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(loc='upper right')
plt.show()
```



```
learningrate_parameter = learning_rate
train_loss=pred.history['loss'][-1]
validation_loss=pred.history['val_loss'][-1]
learningrate_parameter=pd.DataFrame(data=[[learningrate_parameter,
train_loss, validation_loss]],
                                    columns=['Learning Rate',
'Training Loss', 'Validation Loss'])
learningrate_parameter.set_index('Learning Rate')
```

```
               Training Loss  Validation Loss
Learning Rate
0.0001              0.002907         0.000983
```

```
# Implementation model into data test
y_pred_test = regressorGRU.predict(x_test)

# Invert normalization min-max
y_pred_invert_norm = scaler.inverse_transform(y_pred_test)
```

```
5/5 [==============================] - 1s 3ms/step
```

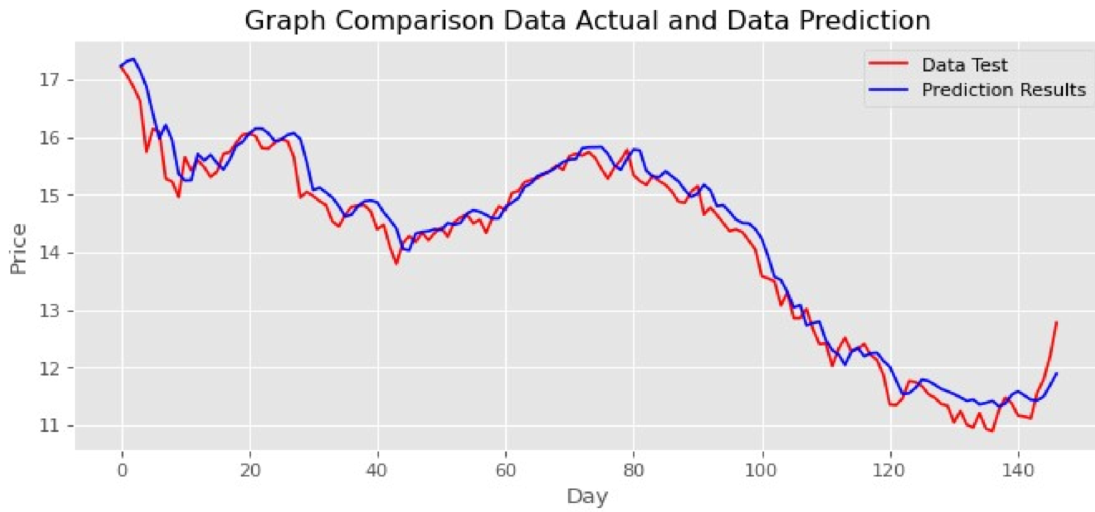
```
set_test = dataset["Close"]
```

```
# Comparison data test with data prediction
datacompare = pd.DataFrame()
datatest=np.array(set_test[totaldatatrain+totaldataval+lag:])
datapred= y_pred_invert_norm

datacompare['Data Test'] = datatest
datacompare['Prediction Results'] = datapred
datacompare
```

```
     Data Test  Prediction Results
0    17.219999           17.232634
1    17.059999           17.319471
2    16.860001           17.355923
3    16.620001           17.145063
4    15.740000           16.874239
..         ...                 ...
142  11.120000           11.443179
143  11.570000           11.432139
144  11.800000           11.501204
145  12.200000           11.699862
146  12.780000           11.895922

[147 rows x 2 columns]
```

```
plt.figure(num=None, figsize=(10, 4), dpi=80,facecolor='w',
edgecolor='k')
plt.title('Graph Comparison Data Actual and Data Prediction')
plt.plot(datacompare['Data Test'], color='red',label='Data Test')
```

```
plt.plot(datacompare['Prediction Results'],
color='blue',label='Prediction Results')
plt.xlabel('Day')
plt.ylabel('Price')
plt.legend()
plt.show()
```




```
def MAPE(Y_actual,Y_Predicted):
    mape = np.mean(np.abs((Y_actual - Y_Predicted)/Y_actual))*100
    return mape
MAPE(datatest, datapred)
```

```
13.672779563378977
```

```
from sklearn.metrics import mean_squared_error
import math
MSE = mean_squared_error(datatest, datapred)
RMSE = math.sqrt(MSE)
print(RMSE)
```

```
0.31862003365973074
```

```
from sklearn.metrics import mean_absolute_error

mean_absolute_error(
    y_true=datatest,
    y_pred=datapred
)
```

```
0.24245089270517772
```

```
n_ahead=input("How many values do you want to predict ?");
n_ahead=int(n_ahead)
# Making the prediction list
def predict_ahead(n_ahead, X_train):
```

```
    yhat = []
    for _ in range(n_ahead):
    # Making the prediction
        fc = regressorGRU.predict(X_train)
        yhat.append(fc)

    # Creating a new input matrix for forecasting
        X_train = np.append(X_train, fc)

    # Ommitting the first variable
        X_train = np.delete(X_train, 0)

    # Reshaping for the next iteration
        X_train = np.reshape(X_train, (1, len(X_train), 1))

    return yhat
y30 = predict_ahead(n_ahead, x_test[len(x_test)-30:])
```

```
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 431ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
```

```
y30
```

```
[array([[0.21542725],
        [0.21618377],
        [0.20416437],
        [0.1957688 ],
        [0.17579836],
        [0.15542054],
        [0.15689984],
        [0.16480839],
        [0.17705113],
        [0.17502397],
        [0.16971573],
        [0.16337366],
        [0.15968338],
        [0.15557352],
        [0.15084796],
        [0.14566335],
        [0.14770925],
        [0.14046457],
        [0.14265561],
        [0.1457422 ],
        [0.13772464],
        [0.14181575],
        [0.15452975],
        [0.15981373],
        [0.15357187],
        [0.14735237],
        [0.1464192 ],
        [0.15225738],
        [0.16905008],
        [0.18562308]], dtype=float32),
 array([[-85335616.]], dtype=float32),
 array([[-1.4407331e+08]], dtype=float32),
 array([[-2.8648858e+08]], dtype=float32),
 array([[-6.0431424e+08]], dtype=float32),
 array([[-1.2853347e+09]], dtype=float32),
 array([[-2.756881e+09]], dtype=float32),
 array([[-5.941977e+09]], dtype=float32),
 array([[-1.2817267e+10]], dtype=float32),
 array([[-2.7706239e+10]], dtype=float32),
 array([[-5.997319e+10]], dtype=float32),
 array([[-1.2992859e+11]], dtype=float32),
 array([[-2.8151533e+11]], dtype=float32),
 array([[-6.1000516e+11]], dtype=float32),
 array([[-1.3215216e+12]], dtype=float32),
 array([[-2.8638254e+12]], dtype=float32),
 array([[-6.2060165e+12]], dtype=float32),
 array([[-1.3448776e+13]], dtype=float32),
 array([[-2.9144272e+13]], dtype=float32),
 array([[-6.3157385e+13]], dtype=float32),
 array([[-1.3686595e+14]], dtype=float32),
```

```
 array([[-2.96597e+14]], dtype=float32),
 array([[-6.4274414e+14]], dtype=float32),
 array([[-1.3928614e+15]], dtype=float32),
 array([[-3.0184187e+15]], dtype=float32),
 array([[-6.541104e+15]], dtype=float32),
 array([[-1.4174983e+16]], dtype=float32),
 array([[-3.0718076e+16]], dtype=float32),
 array([[-6.6568003e+16]], dtype=float32),
 array([[-1.4425706e+17]], dtype=float32)]
```

```
y30 = [[0.21542725],
        [0.21618377],
        [0.20416437],
        [0.1957688 ],
        [0.17579836],
        [0.15542054],
        [0.15689984],
        [0.16480839],
        [0.17705113],
        [0.17502397],
        [0.16971573],
        [0.16337366],
        [0.15968338],
        [0.15557352],
        [0.15084796],
        [0.14566335],
        [0.14770925],
        [0.14046457],
        [0.14265561],
        [0.1457422 ],
        [0.13772464],
        [0.14181575],
        [0.15452975],
        [0.15981373],
        [0.15357187],
        [0.14735237],
        [0.1464192 ],
        [0.15225738],
        [0.16905008],
        [0.18562308]]

y30 = scaler.inverse_transform(y30)

from numpy import savetxt
savetxt('GRU.csv', y30, delimiter=',')
```