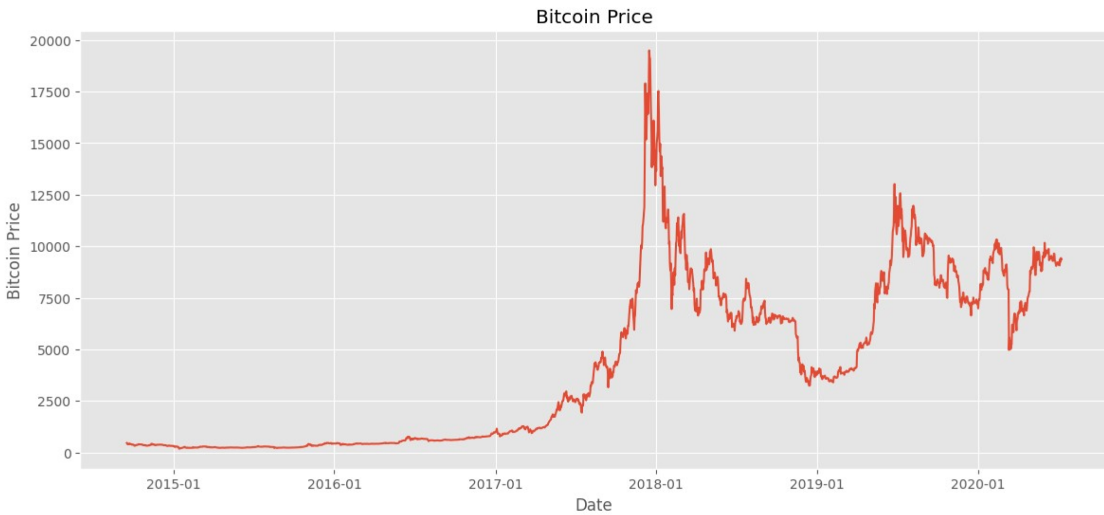
```python
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from sklearn.preprocessing import MinMaxScaler
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout,GRU
from keras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

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

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

data_raw = pd.read_csv("BTC-USD.csv", index_col="Date",
parse_dates=["Date"])
data_raw

data_raw.describe()
```

```
               Open          High           Low         Close     Adj
Close  \
count   2123.000000   2123.000000   2123.000000   2123.000000
2123.000000
mean    4185.932266   4294.749473   4069.363555   4190.018268
4190.018268
std     4029.791354   4152.358417   3886.728318   4030.549406
4030.549406
min      176.897003    211.731003    171.509995    178.102997
178.102997
25%      425.334488    432.388504    420.620514    424.749497
424.749497
50%     3341.840088   3453.449951   3247.669922   3378.939941
3378.939941
75%     7509.315185   7696.606445   7374.902588   7531.821777
7531.821777
max    19475.800780  20089.000000  18974.099610  19497.400390
19497.400390

             Volume
count  2.123000e+03
mean   7.431161e+09
std    1.144642e+10
min    5.914570e+06
25%    5.841410e+07
50%    1.537460e+09
```

```
75%    1.004265e+10
max    7.415677e+10
```

```
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('Bitcoin Price')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m"))
plt.title('Bitcoin Price')
plt.show()
```

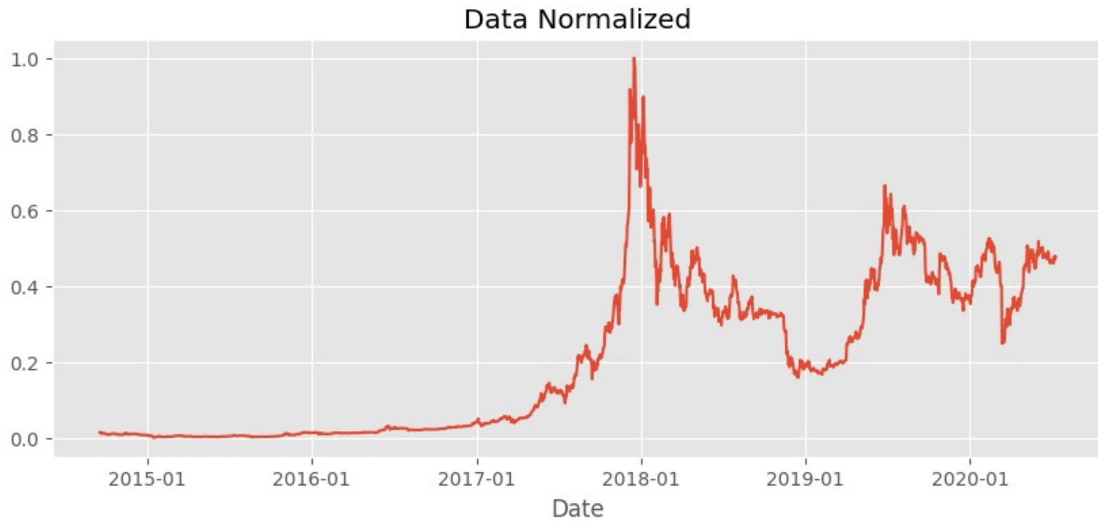
```
 Count row of data:  2123
```



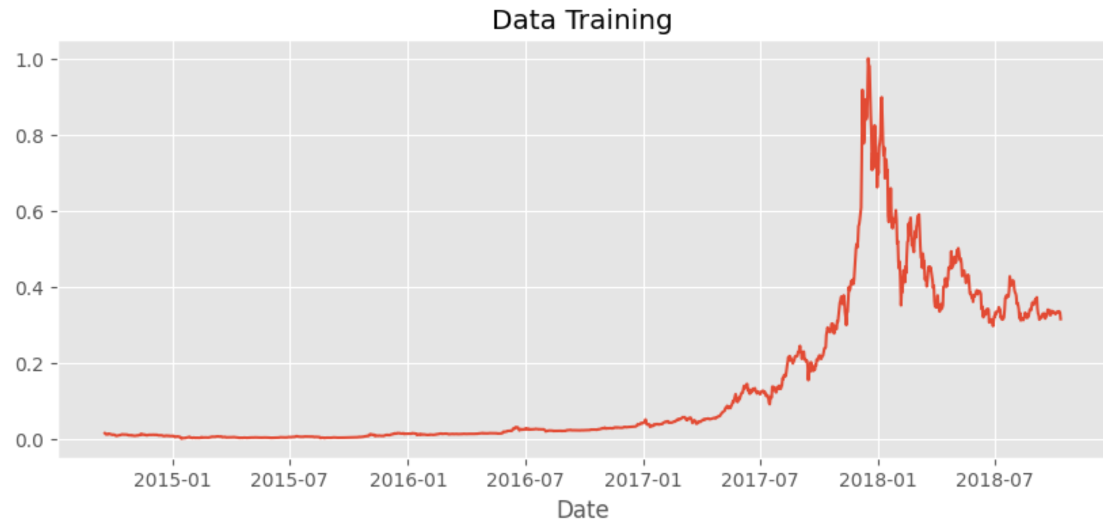

```
dataset_norm = dataset.copy()
dataset[['Close']]
scaler = MinMaxScaler()
dataset_norm['Close'] = scaler.fit_transform(dataset[['Close']])
dataset_norm
```

```
               Close
Date
2014-09-17  0.014453
2014-09-18  0.012751
2014-09-19  0.011216
2014-09-20  0.011947
2014-09-21  0.011425
...              ...
2020-07-05  0.460464
2020-07-06  0.476072
2020-07-07  0.469695
2020-07-08  0.478808
2020-07-09  0.475121
```

```
[2123 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()
```

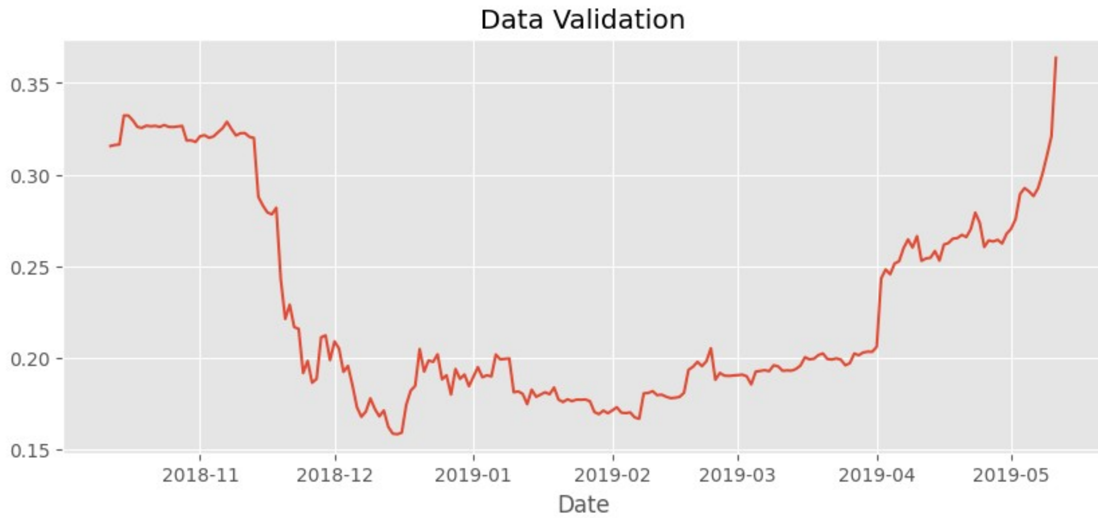



```
totaldata = dataset.values
totaldatatrain = int(len(totaldata)*0.7)
totaldataval = int(len(totaldata)*0.1)
totaldatatest = int(len(totaldata)*0.2)
training_set = dataset_norm[0:totaldatatrain]
val_set=dataset_norm[totaldatatrain:totaldatatrain+totaldataval]
test_set = dataset_norm[totaldatatrain+totaldataval:]

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()
```

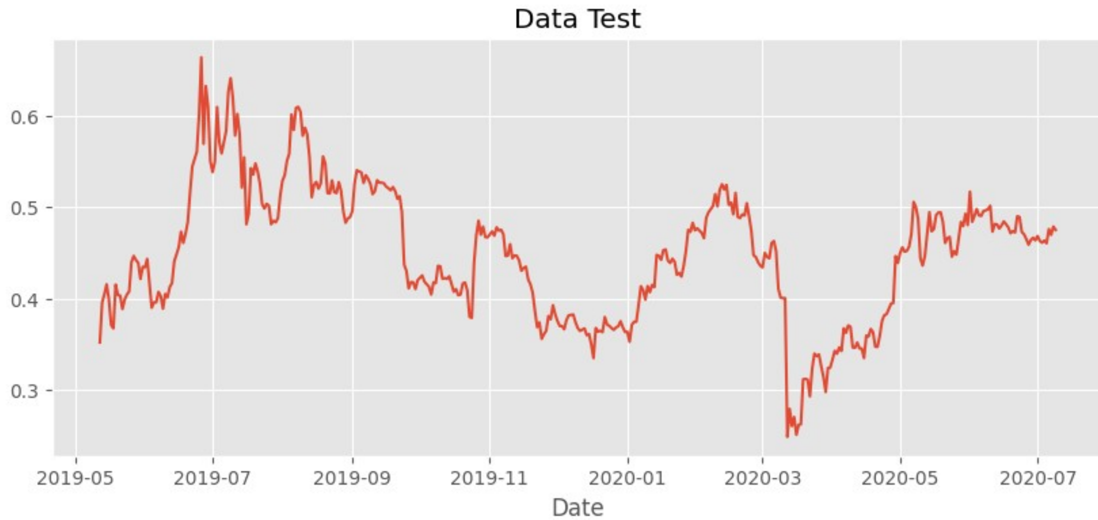



```
fig = plt.figure(figsize=(10, 4))
plt.plot(val_set)
plt.xlabel('Date')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m"))
plt.title('Data Validation')
val_set
```

```
               Close
Date
2018-10-12  0.315564
2018-10-13  0.316155
2018-10-14  0.316410
2018-10-15  0.332229
2018-10-16  0.332207
...              ...
2019-05-07  0.292526
2019-05-08  0.300443
2019-05-09  0.310385
2019-05-10  0.320961
2019-05-11  0.363712

[212 rows x 1 columns]
```




```
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
2019-05-12  0.351683
2019-05-13  0.395295
2019-05-14  0.404586
2019-05-15  0.415495
2019-05-16  0.398918
...              ...
2020-07-05  0.460464
```

```
2020-07-06  0.476072
2020-07-07  0.469695
2020-07-08  0.478808
2020-07-09  0.475121

[425 rows x 1 columns]
```

```
# 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=256
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 = 'tanh'))
regressorGRU.add(Dropout(0.2))
```

```python
# Second GRU layer with dropout
regressorGRU.add(GRU(units=hidden_unit, return_sequences=True,
activation = 'tanh'))
regressorGRU.add(Dropout(0.2))
# Third GRU layer with dropout
regressorGRU.add(GRU(units=hidden_unit, return_sequences=False,
activation = 'tanh'))
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
6/6 [==============================] - 7s 261ms/step - loss: 0.0454 -
val_loss: 0.0126
Epoch 2/100
6/6 [==============================] - 0s 23ms/step - loss: 0.0288 -
val_loss: 0.0017
Epoch 3/100
6/6 [==============================] - 0s 20ms/step - loss: 0.0209 -
val_loss: 0.0024
Epoch 4/100
6/6 [==============================] - 0s 18ms/step - loss: 0.0086 -
val_loss: 0.0025
Epoch 5/100
6/6 [==============================] - 0s 20ms/step - loss: 0.0019 -
val_loss: 0.0013
Epoch 6/100
6/6 [==============================] - 0s 20ms/step - loss: 0.0035 -
val_loss: 5.9475e-04
Epoch 7/100
6/6 [==============================] - 0s 24ms/step - loss: 0.0014 -
val_loss: 1.1086e-04
Epoch 8/100
6/6 [==============================] - 0s 20ms/step - loss: 0.0016 -
val_loss: 4.8574e-04
Epoch 9/100
6/6 [==============================] - 0s 20ms/step - loss: 0.0014 -
val_loss: 1.7171e-04
```

```
Epoch 10/100
6/6 [==============================] - 0s 19ms/step - loss: 0.0011 -
val_loss: 1.0061e-04
Epoch 11/100
6/6 [==============================] - 0s 20ms/step - loss: 0.0012 -
val_loss: 1.2893e-04
Epoch 12/100
6/6 [==============================] - 0s 20ms/step - loss: 0.0011 -
val_loss: 1.1376e-04
Epoch 13/100
6/6 [==============================] - 0s 19ms/step - loss: 0.0011 -
val_loss: 2.0370e-04
Epoch 14/100
6/6 [==============================] - 0s 23ms/step - loss: 9.9688e-04
- val_loss: 1.0401e-04
Epoch 15/100
6/6 [==============================] - 0s 20ms/step - loss: 0.0012 -
val_loss: 9.6851e-05
Epoch 16/100
6/6 [==============================] - 0s 19ms/step - loss: 0.0012 -
val_loss: 1.6423e-04
Epoch 17/100
6/6 [==============================] - 0s 20ms/step - loss: 9.4218e-04
- val_loss: 1.1865e-04
Epoch 18/100
6/6 [==============================] - 0s 20ms/step - loss: 8.9245e-04
- val_loss: 1.6757e-04
Epoch 19/100
6/6 [==============================] - 0s 19ms/step - loss: 9.5291e-04
- val_loss: 1.0826e-04
Epoch 20/100
6/6 [==============================] - 0s 19ms/step - loss: 8.8346e-04
- val_loss: 1.0803e-04
Epoch 21/100
6/6 [==============================] - 0s 17ms/step - loss: 8.8409e-04
- val_loss: 9.7854e-05
Epoch 22/100
6/6 [==============================] - 0s 22ms/step - loss: 9.3811e-04
- val_loss: 1.2498e-04
Epoch 23/100
6/6 [==============================] - 0s 21ms/step - loss: 9.0675e-04
- val_loss: 1.0133e-04
Epoch 24/100
6/6 [==============================] - 0s 20ms/step - loss: 9.1614e-04
- val_loss: 1.1963e-04
Epoch 25/100
6/6 [==============================] - 0s 19ms/step - loss: 9.0161e-04
- val_loss: 9.5969e-05
Epoch 26/100
6/6 [==============================] - 0s 19ms/step - loss: 9.1780e-04
```

```
- val_loss: 9.3839e-05
Epoch 27/100
6/6 [==============================] - 0s 19ms/step - loss: 8.4959e-04
- val_loss: 1.0242e-04
Epoch 28/100
6/6 [==============================] - 0s 19ms/step - loss: 9.3771e-04
- val_loss: 9.5759e-05
Epoch 29/100
6/6 [==============================] - 0s 19ms/step - loss: 6.8770e-04
- val_loss: 9.3451e-05
Epoch 30/100
6/6 [==============================] - 0s 19ms/step - loss: 9.0208e-04
- val_loss: 9.3456e-05
Epoch 31/100
6/6 [==============================] - 0s 21ms/step - loss: 8.8897e-04
- val_loss: 9.4025e-05
Epoch 32/100
6/6 [==============================] - 0s 23ms/step - loss: 8.1034e-04
- val_loss: 1.0013e-04
Epoch 33/100
6/6 [==============================] - 0s 19ms/step - loss: 9.2164e-04
- val_loss: 1.1481e-04
Epoch 34/100
6/6 [==============================] - 0s 23ms/step - loss: 9.0565e-04
- val_loss: 1.7241e-04
Epoch 35/100
6/6 [==============================] - 0s 21ms/step - loss: 8.2013e-04
- val_loss: 1.2244e-04
Epoch 36/100
6/6 [==============================] - 0s 19ms/step - loss: 8.4016e-04
- val_loss: 1.1055e-04
Epoch 37/100
6/6 [==============================] - 0s 18ms/step - loss: 8.9437e-04
- val_loss: 1.0730e-04
Epoch 38/100
6/6 [==============================] - 0s 26ms/step - loss: 8.8619e-04
- val_loss: 9.9385e-05
Epoch 39/100
6/6 [==============================] - 0s 21ms/step - loss: 8.9189e-04
- val_loss: 9.3496e-05
Epoch 40/100
6/6 [==============================] - 0s 20ms/step - loss: 9.1083e-04
- val_loss: 1.3472e-04
Epoch 41/100
6/6 [==============================] - 0s 19ms/step - loss: 8.7255e-04
- val_loss: 1.0316e-04
Epoch 42/100
6/6 [==============================] - 0s 18ms/step - loss: 8.3771e-04
- val_loss: 9.4199e-05
Epoch 43/100
```

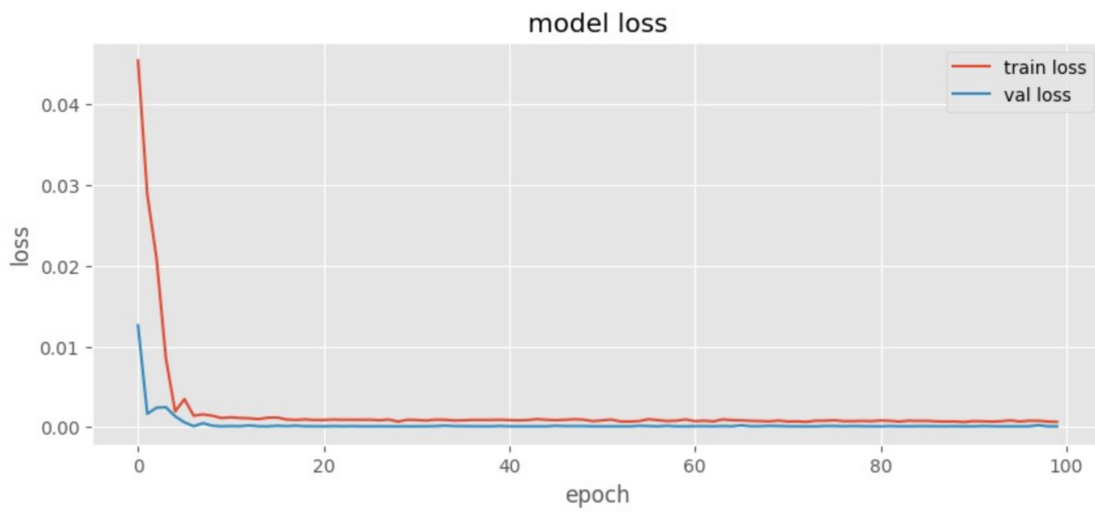
```
6/6 [==============================] - 0s 20ms/step - loss: 8.7517e-04
- val_loss: 1.0048e-04
Epoch 44/100
6/6 [==============================] - 0s 21ms/step - loss: 9.8757e-04
- val_loss: 9.7213e-05
Epoch 45/100
6/6 [==============================] - 0s 21ms/step - loss: 9.0589e-04
- val_loss: 9.3836e-05
Epoch 46/100
6/6 [==============================] - 0s 22ms/step - loss: 8.5902e-04
- val_loss: 1.6021e-04
Epoch 47/100
6/6 [==============================] - 0s 18ms/step - loss: 9.0748e-04
- val_loss: 1.2195e-04
Epoch 48/100
6/6 [==============================] - 0s 19ms/step - loss: 9.7000e-04
- val_loss: 1.2526e-04
Epoch 49/100
6/6 [==============================] - 0s 20ms/step - loss: 9.2114e-04
- val_loss: 1.2529e-04
Epoch 50/100
6/6 [==============================] - 0s 20ms/step - loss: 7.4015e-04
- val_loss: 9.4275e-05
Epoch 51/100
6/6 [==============================] - 0s 19ms/step - loss: 8.3786e-04
- val_loss: 9.7834e-05
Epoch 52/100
6/6 [==============================] - 0s 22ms/step - loss: 9.3313e-04
- val_loss: 9.8261e-05
Epoch 53/100
6/6 [==============================] - 0s 20ms/step - loss: 6.9593e-04
- val_loss: 9.8403e-05
Epoch 54/100
6/6 [==============================] - 0s 21ms/step - loss: 6.8713e-04
- val_loss: 9.4685e-05
Epoch 55/100
6/6 [==============================] - 0s 20ms/step - loss: 7.4687e-04
- val_loss: 1.5591e-04
Epoch 56/100
6/6 [==============================] - 0s 19ms/step - loss: 9.7563e-04
- val_loss: 1.2485e-04
Epoch 57/100
6/6 [==============================] - 0s 18ms/step - loss: 8.6208e-04
- val_loss: 9.4595e-05
Epoch 58/100
6/6 [==============================] - 0s 19ms/step - loss: 7.5339e-04
- val_loss: 1.5381e-04
Epoch 59/100
6/6 [==============================] - 0s 17ms/step - loss: 7.9709e-04
- val_loss: 9.4000e-05
```

```
Epoch 60/100
6/6 [==============================] - 0s 19ms/step - loss: 9.4381e-04
- val_loss: 9.4578e-05
Epoch 61/100
6/6 [==============================] - 0s 19ms/step - loss: 7.3834e-04
- val_loss: 1.1341e-04
Epoch 62/100
6/6 [==============================] - 0s 21ms/step - loss: 7.9540e-04
- val_loss: 1.1444e-04
Epoch 63/100
6/6 [==============================] - 0s 19ms/step - loss: 7.0601e-04
- val_loss: 9.7697e-05
Epoch 64/100
6/6 [==============================] - 0s 19ms/step - loss: 9.5211e-04
- val_loss: 1.3257e-04
Epoch 65/100
6/6 [==============================] - 0s 17ms/step - loss: 8.5920e-04
- val_loss: 1.0055e-04
Epoch 66/100
6/6 [==============================] - 0s 20ms/step - loss: 8.3267e-04
- val_loss: 2.1865e-04
Epoch 67/100
6/6 [==============================] - 0s 18ms/step - loss: 7.7634e-04
- val_loss: 1.0684e-04
Epoch 68/100
6/6 [==============================] - 0s 19ms/step - loss: 7.5220e-04
- val_loss: 1.0763e-04
Epoch 69/100
6/6 [==============================] - 0s 19ms/step - loss: 7.1219e-04
- val_loss: 1.5153e-04
Epoch 70/100
6/6 [==============================] - 0s 19ms/step - loss: 8.1112e-04
- val_loss: 1.3215e-04
Epoch 71/100
6/6 [==============================] - 0s 19ms/step - loss: 6.9552e-04
- val_loss: 1.0269e-04
Epoch 72/100
6/6 [==============================] - 0s 22ms/step - loss: 7.1780e-04
- val_loss: 9.8985e-05
Epoch 73/100
6/6 [==============================] - 0s 21ms/step - loss: 6.5172e-04
- val_loss: 9.2581e-05
Epoch 74/100
6/6 [==============================] - 0s 19ms/step - loss: 7.9582e-04
- val_loss: 9.3362e-05
Epoch 75/100
6/6 [==============================] - 0s 19ms/step - loss: 7.9329e-04
- val_loss: 1.3179e-04
Epoch 76/100
6/6 [==============================] - 0s 20ms/step - loss: 8.4698e-04
```

```
- val_loss: 1.3343e-04
Epoch 77/100
6/6 [==============================] - 0s 19ms/step - loss: 7.4268e-04
- val_loss: 9.9911e-05
Epoch 78/100
6/6 [==============================] - 0s 19ms/step - loss: 7.4162e-04
- val_loss: 1.3087e-04
Epoch 79/100
6/6 [==============================] - 0s 19ms/step - loss: 7.6898e-04
- val_loss: 1.1060e-04
Epoch 80/100
6/6 [==============================] - 0s 20ms/step - loss: 7.4081e-04
- val_loss: 1.0159e-04
Epoch 81/100
6/6 [==============================] - 0s 19ms/step - loss: 8.2323e-04
- val_loss: 9.5219e-05
Epoch 82/100
6/6 [==============================] - 0s 22ms/step - loss: 7.8298e-04
- val_loss: 1.2737e-04
Epoch 83/100
6/6 [==============================] - 0s 21ms/step - loss: 6.8356e-04
- val_loss: 1.0098e-04
Epoch 84/100
6/6 [==============================] - 0s 19ms/step - loss: 7.9674e-04
- val_loss: 9.9551e-05
Epoch 85/100
6/6 [==============================] - 0s 19ms/step - loss: 7.6973e-04
- val_loss: 1.1442e-04
Epoch 86/100
6/6 [==============================] - 0s 21ms/step - loss: 7.7723e-04
- val_loss: 1.1414e-04
Epoch 87/100
6/6 [==============================] - 0s 21ms/step - loss: 7.1047e-04
- val_loss: 1.0457e-04
Epoch 88/100
6/6 [==============================] - 0s 19ms/step - loss: 6.9472e-04
- val_loss: 9.7096e-05
Epoch 89/100
6/6 [==============================] - 0s 17ms/step - loss: 6.9782e-04
- val_loss: 1.1210e-04
Epoch 90/100
6/6 [==============================] - 0s 19ms/step - loss: 6.3124e-04
- val_loss: 9.4695e-05
Epoch 91/100
6/6 [==============================] - 0s 20ms/step - loss: 7.4426e-04
- val_loss: 9.4547e-05
Epoch 92/100
6/6 [==============================] - 0s 21ms/step - loss: 7.1840e-04
- val_loss: 1.3283e-04
Epoch 93/100
```

```
6/6 [==============================] - 0s 20ms/step - loss: 6.8155e-04
- val_loss: 1.0717e-04
Epoch 94/100
6/6 [==============================] - 0s 20ms/step - loss: 7.3159e-04
- val_loss: 9.2368e-05
Epoch 95/100
6/6 [==============================] - 0s 17ms/step - loss: 8.4040e-04
- val_loss: 1.0356e-04
Epoch 96/100
6/6 [==============================] - 0s 19ms/step - loss: 6.8983e-04
- val_loss: 9.2775e-05
Epoch 97/100
6/6 [==============================] - 0s 23ms/step - loss: 7.9131e-04
- val_loss: 1.0636e-04
Epoch 98/100
6/6 [==============================] - 0s 20ms/step - loss: 7.8247e-04
- val_loss: 2.3049e-04
Epoch 99/100
6/6 [==============================] - 0s 21ms/step - loss: 6.8680e-04
- val_loss: 1.0050e-04
Epoch 100/100
6/6 [==============================] - 0s 18ms/step - loss: 6.4989e-04
- val_loss: 9.5180e-05
```

```
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               0.00065         0.000095
```

```
# 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)
```

```
14/14 [==============================] - 1s 3ms/step
```

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

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

```
       Data Test  Prediction Results
0    7994.416016         7215.246582
1    8205.167969         7797.472168
2    7884.909180         7983.531738
3    7343.895508         7996.895020
4    7271.208008         7605.167969
..           ...                 ...
418  9073.942383         8983.612305
419  9375.474609         8990.070312
420  9252.277344         9062.615234
421  9428.333008         9201.458984
422  9357.105469         9189.769531

[423 rows x 2 columns]
```

```
def rmse(datatest, datapred):
    return np.round(np.sqrt(np.mean((datapred - datatest) ** 2)), 4)
print('Result Root Mean Square Error Prediction
Model :',rmse(datatest, datapred))

def mape(datatest, datapred):
    return np.round(np.mean(np.abs((datatest - datapred) / datatest) *
100), 4)
```

```
print('Result Mean Absolute Percentage Error Prediction Model : ',
mape(datatest, datapred), '%')
```

```
Result Root Mean Square Error Prediction Model : 1955.2445
Result Mean Absolute Percentage Error Prediction Model :  18.2465 %
```

```
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()
```