```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
```

```
data = pd.read_csv('/content/2022-04-25-copy.csv', date_parser = True)
data.tail()
```

	date	estimated_transaction_volume	close_price	open_price	high_price	low_
4801	2/24/2022	172501.0	38376.88	37253.26	39720.00	34
4802	2/25/2022	136990.0	39231.64	38360.93	39727.97	38
4803	2/26/2022	61083.0	39146.66	39242.64	40330.99	38
4804	2/27/2022	47577.0	37712.68	39146.66	39886.92	37
4805	2/28/2022	47577.0	43178.98	37717.10	44256.08	37

```
data_training = data[data['date']< '2022-31-01'].copy()
data_training
```

	date	estimated_transaction_volume	close_price	open_price	high_price	low_
0	1/2/2009	0.0	0.00	0.00	0.00	
1	1/3/2009	0.0	0.00	0.00	0.00	
2	1/4/2009	0.0	0.00	0.00	0.00	
3	1/5/2009	0.0	0.00	0.00	0.00	
4	1/6/2009	0.0	0.00	0.00	0.00	
...	...	...	...	...	...	
4801	2/24/2022	172501.0	38376.88	37253.26	39720.00	34
4802	2/25/2022	136990.0	39231.64	38360.93	39727.97	38
4803	2/26/2022	61083.0	39146.66	39242.64	40330.99	38
4804	2/27/2022	47577.0	37712.68	39146.66	39886.92	37
4805	2/28/2022	47577.0	43178.98	37717.10	44256.08	37

2024 rows × 6 columns

```
data_test = data[data['date']< '2022-31-01'].copy()
data_test
```

	date	estimated_transaction_volume	close_price	open_price	high_price	low_
0	1/2/2009	0.0	0.00	0.00	0.00	
1	1/3/2009	0.0	0.00	0.00	0.00	
2	1/4/2009	0.0	0.00	0.00	0.00	
3	1/5/2009	0.0	0.00	0.00	0.00	
4	1/6/2009	0.0	0.00	0.00	0.00	
...	...	...	...	...	...	
4801	2/24/2022	172501.0	38376.88	37253.26	39720.00	34
4802	2/25/2022	136990.0	39231.64	38360.93	39727.97	38
4803	2/26/2022	61083.0	39146.66	39242.64	40330.99	38
4804	2/27/2022	47577.0	37712.68	39146.66	39886.92	37
4805	2/28/2022	47577.0	43178.98	37717.10	44256.08	37

2024 rows × 6 columns

```
training_data = data_training.drop(['date'], axis = 1)
training_data.head()
```

	estimated_transaction_volume	close_price	open_price	high_price	low_price
0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0

```
scaler = MinMaxScaler()
training_data = scaler.fit_transform(training_data)
training_data
```

```
array([[0.        , 0.        , 0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ],
       ...,
       [0.01149655, 0.57944404, 0.5809637 , 0.5845071 , 0.58264151],
       [0.00895456, 0.55821845, 0.57954278, 0.5780713 , 0.55872815],
       [0.00895456, 0.63912995, 0.558379  , 0.64139246, 0.56556966]])
```

```
X_train = []
Y_train = []
training_data.shape[0]
training_data.shape[0]
for i in range(60, training_data.shape[0]):
  X_train.append(training_data[i-60:i])
  Y_train.append(training_data[i,0])
X_train, Y_train = np.array(X_train), np.array(Y_train)
Y_train.shape
```

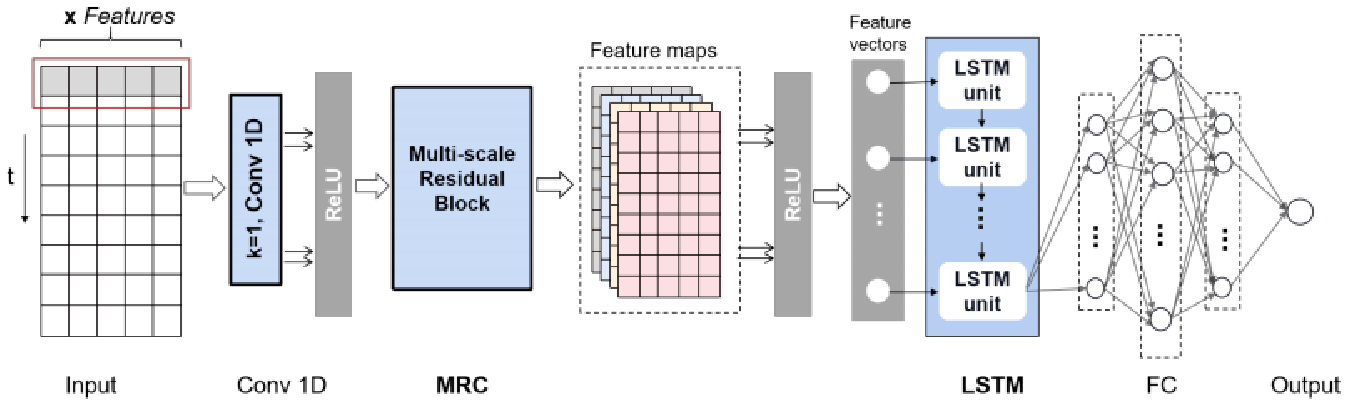
```
(1964,)
```

```
# from tensorflow.keras import Sequential
# from tensorflow.keras.layers import Dense, LSTM, Dropout
# #Initialize the RNN
# model = Sequential()
# model.add(LSTM(units = 50, activation = 'relu', return_sequences = True, input_shape =
# model.add(Dropout(0.2))
# model.add(LSTM(units = 60, activation = 'relu', return_sequences = True))
# model.add(Dropout(0.3))
# model.add(LSTM(units = 80, activation = 'relu', return_sequences = True))
# model.add(Dropout(0.4))
# model.add(LSTM(units = 120, activation = 'relu'))
# model.add(Dropout(0.5))
# model.add(Dense(units =1))
# model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 50)            11200

 dropout (Dropout)           (None, 60, 50)            0

 lstm_1 (LSTM)               (None, 60, 60)            26640

 dropout_1 (Dropout)         (None, 60, 60)            0

 lstm_2 (LSTM)               (None, 60, 80)            45120

 dropout_2 (Dropout)         (None, 60, 80)            0

 lstm_3 (LSTM)               (None, 120)               96480

 dropout_3 (Dropout)         (None, 120)               0

 dense (Dense)               (None, 1)                 121

=================================================================
Total params: 179,561
Trainable params: 179,561
Non-trainable params: 0
_________________________________________________________________
```

MRCLSTM neural network

- The network contains an input layer
- a 1D convolutional layer
- the multi-scale residual module
- an LSTM layer
- a fully connected layer
- an output layer



```
#1D kernal
from keras.layers import Input, Dense, LSTM, MaxPooling1D, Conv1D
from keras.models import Model
import tensorflow as tf
#input
input_layer = Input(shape=(X_train.shape[1], 5))
#Conv 1D Layer
conv1 = Conv1D(filters=16,
               kernel_size=1,
               strides=1,
               activation='relu',
               padding='same')(input_layer)
#MRC
l1 = Conv1D(filters=16,
               kernel_size=1,
               strides=1,
               activation='relu',
               padding='same')(conv1)
l2 = Conv1D(filters=16,
               kernel_size=2,
               strides=1,
               activation='relu',
               padding='same')(conv1)
l3 = Conv1D(filters=16,
               kernel_size=3,
               strides=1,
               activation='relu',
               padding='same')(conv1)
Multi_scale_Residual_Block = tf.keras.layers.Concatenate()([l1, l2, l3])
#LSTM layer
lstm1 = LSTM(50, return_sequences=True)(Multi_scale_Residual_Block)
#Fully Connected Layer
#output layer
output_layer = Dense(1, activation='sigmoid')(lstm1)
model = Model(inputs=input_layer, outputs=output_layer)

model.summary()
```

```
Model: "model_2"
_______________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
=======================================================================================
 input_6 (InputLayer)           [(None, 60, 5)]      0           []

 conv1d_20 (Conv1D)             (None, 60, 16)       96          ['input_6[0][0]']

 conv1d_21 (Conv1D)             (None, 60, 16)       272         ['conv1d_20[0][0]']

 conv1d_22 (Conv1D)             (None, 60, 16)       528         ['conv1d_20[0][0]']

 conv1d_23 (Conv1D)             (None, 60, 16)       784         ['conv1d_20[0][0]']

 concatenate_2 (Concatenate)    (None, 60, 48)       0           ['conv1d_21[0][0]',
                                                                  'conv1d_22[0][0]',
                                                                  'conv1d_23[0][0]']

 lstm_6 (LSTM)                  (None, 60, 50)       19800       ['concatenate_2[0][0]'

 dense_3 (Dense)                (None, 60, 1)        51          ['lstm_6[0][0]']

=======================================================================================
Total params: 21,531
Trainable params: 21,531
Non-trainable params: 0
_______________________________________________________________________________________
```

```
opt = tf.keras.optimizers.Adam(learning_rate=0.001)
```

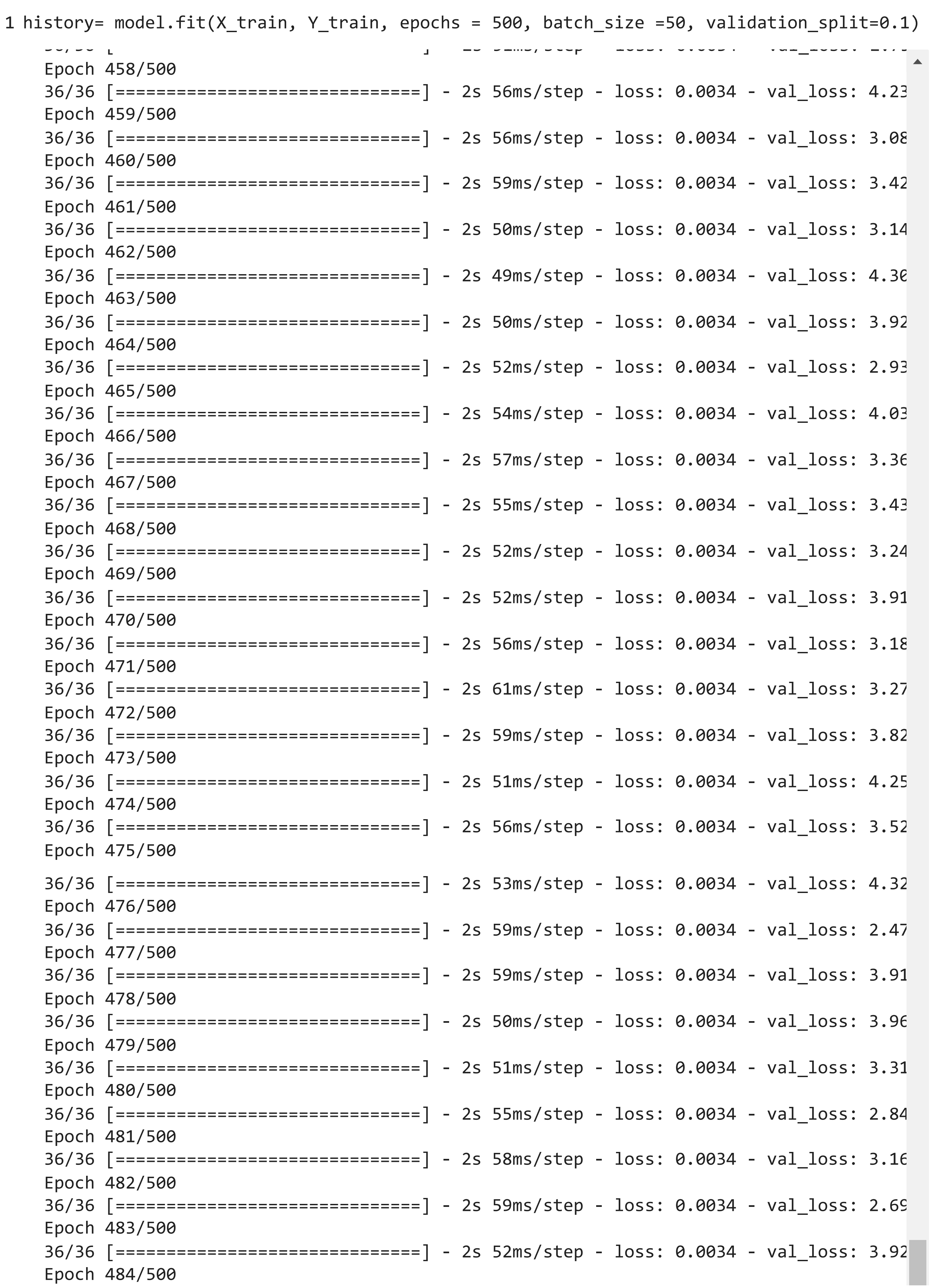
```
model.compile(optimizer = opt, loss = 'mean_squared_error')
```

```
X_train.shape
```

```
(1964, 60, 5)
```

```
Y_train.shape
```

```
(1964,)
```

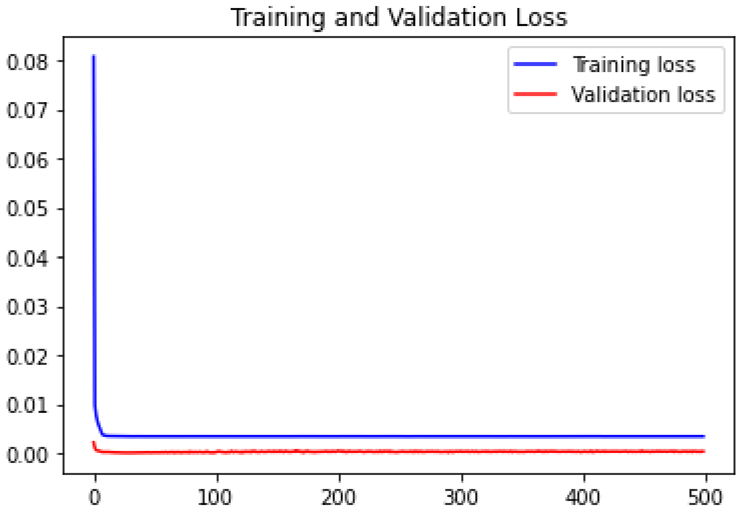


```
history= model.fit(X_train, Y_train, epochs = 500, batch_size =50, validation_split=0.1)
```

```
Epoch 458/500
36/36 [==============================] - 2s 56ms/step - loss: 0.0034 - val_loss: 4.23
Epoch 459/500
36/36 [==============================] - 2s 56ms/step - loss: 0.0034 - val_loss: 3.08
Epoch 460/500
36/36 [==============================] - 2s 59ms/step - loss: 0.0034 - val_loss: 3.42
Epoch 461/500
36/36 [==============================] - 2s 50ms/step - loss: 0.0034 - val_loss: 3.14
Epoch 462/500
36/36 [==============================] - 2s 49ms/step - loss: 0.0034 - val_loss: 4.30
Epoch 463/500
36/36 [==============================] - 2s 50ms/step - loss: 0.0034 - val_loss: 3.92
Epoch 464/500
36/36 [==============================] - 2s 52ms/step - loss: 0.0034 - val_loss: 2.93
Epoch 465/500
36/36 [==============================] - 2s 54ms/step - loss: 0.0034 - val_loss: 4.03
Epoch 466/500
36/36 [==============================] - 2s 57ms/step - loss: 0.0034 - val_loss: 3.36
Epoch 467/500
36/36 [==============================] - 2s 55ms/step - loss: 0.0034 - val_loss: 3.43
Epoch 468/500
36/36 [==============================] - 2s 52ms/step - loss: 0.0034 - val_loss: 3.24
Epoch 469/500
36/36 [==============================] - 2s 52ms/step - loss: 0.0034 - val_loss: 3.91
Epoch 470/500
36/36 [==============================] - 2s 56ms/step - loss: 0.0034 - val_loss: 3.18
Epoch 471/500
36/36 [==============================] - 2s 61ms/step - loss: 0.0034 - val_loss: 3.27
Epoch 472/500
36/36 [==============================] - 2s 59ms/step - loss: 0.0034 - val_loss: 3.82
Epoch 473/500
36/36 [==============================] - 2s 51ms/step - loss: 0.0034 - val_loss: 4.25
Epoch 474/500
36/36 [==============================] - 2s 56ms/step - loss: 0.0034 - val_loss: 3.52
Epoch 475/500

36/36 [==============================] - 2s 53ms/step - loss: 0.0034 - val_loss: 4.32
Epoch 476/500
36/36 [==============================] - 2s 59ms/step - loss: 0.0034 - val_loss: 2.47
Epoch 477/500
36/36 [==============================] - 2s 59ms/step - loss: 0.0034 - val_loss: 3.91
Epoch 478/500
36/36 [==============================] - 2s 50ms/step - loss: 0.0034 - val_loss: 3.96
Epoch 479/500
36/36 [==============================] - 2s 51ms/step - loss: 0.0034 - val_loss: 3.31
Epoch 480/500
36/36 [==============================] - 2s 55ms/step - loss: 0.0034 - val_loss: 2.84
Epoch 481/500
36/36 [==============================] - 2s 58ms/step - loss: 0.0034 - val_loss: 3.16
Epoch 482/500
36/36 [==============================] - 2s 59ms/step - loss: 0.0034 - val_loss: 2.69
Epoch 483/500
36/36 [==============================] - 2s 52ms/step - loss: 0.0034 - val_loss: 3.92
Epoch 484/500
```

```
36/36 [==============================] - 2s 58ms/step - loss: 0.0034 - val_loss: 3.69
Epoch 485/500
36/36 [==============================] - 2s 62ms/step - loss: 0.0034 - val_loss: 3.92
Epoch 486/500
```

```
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(loss))
plt.figure()
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title("Training and Validation Loss")
plt.legend()
plt.show()
```



```
part_60_days = data_training.tail(60)
df= part_60_days.append(data_test, ignore_index = True)
df = df.drop(['date'], axis = 1)
df.head()
```

	estimated_transaction_volume	close_price	open_price	high_price	low_price
0	172549.44990	46214.37	47110.30	48589.47	45655.31
1	48673.23487	47777.42	46230.00	47960.98	46199.90
2	30402.67709	47350.22	47745.25	47989.00	46660.00
3	60782.52690	46439.89	47290.55	47586.58	45692.13
4	136044.53810	45820.00	46464.01	47526.00	45539.05

```
inputs = scaler.transform(df)
```

```
X_test = []
```

```
Y_test = []
for i in range (60, inputs.shape[0]):
    X_test.append(inputs[i-60:i])
    Y_test.append(inputs[i, 0])
X_test, Y_test = np.array(X_test), np.array(Y_test)
X_test.shape, Y_test.shape
Y_pred = regressor.predict(X_test)
Y_pred, Y_test
scaler.scale_
```

```
scale = 1/5.18164146e-05
Y_test = Y_test*scale Y_pred = Y_pred*scale
Y_pred
```

```
Y_test
```

```
plt.figure(figsize=(14,5))
plt.plot(Y_test, color = 'red', label = 'Real Bitcoin Price')
plt.plot(Y_pred, color = 'green', label = 'Predicted Bitcoin Price')
plt.title('Bitcoin Price Prediction using RNN-LSTM')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt.show()
```