LSTM Without Google Trend and Financial Indicators

May 16, 2021

[1]:
```
import pandas as pd
import numpy as np
import datetime
import pytz  #function of time region
import statsmodels.api as sm  # Unit root test
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.dates as mdate

from keras.layers import Dropout
from statsmodels.tsa.arima.model import ARIMA  #ARIMA  model
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from datetime import datetime
from pmdarima.arima import auto_arima
from math import sqrt
from sklearn.metrics import mean_squared_error
```

[3]:
```
bitcoin = pd.read_csv(r"E:\PhD study\ELEG5491 Introduction to Deep␣
↪Learning\bitcoin\datasets\bitcoin1dimtrain.csv")
training_set1=bitcoin.values            #converting to 2d array
training_set1
```

[3]:
```
array([[ 382.845],
       [ 386.475],
       [ 383.158],
       …,
       [7152.302],
       [6932.48 ],
       [6640.515]])
```

[4]:
```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()                         #scaling using normalisation
training_set1 = sc.fit_transform(training_set1)
xtrain=training_set1[0:1884]                #input values of rows␣
↪[0-2694]
ytrain=training_set1[1:1885]
```

[5]:
```
today=pd.DataFrame(xtrain)                  #assigning the values of xtrain to␣
 ↪today
tomorrow=pd.DataFrame(ytrain)               #assigning the values of xtrain to␣
 ↪tomorrow
ex= pd.concat([today,tomorrow],axis=1)         #concat two columns
ex.columns=(['today','tomorrow'])
xtrain = np.reshape(xtrain, (1884, 1, 1))
```

[6]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
```

[7]:
```
regressor=Sequential()                                                    ␣
 ↪#initialize the RNN
regressor.add(LSTM(units=4,activation='sigmoid',input_shape=(None,1)))
regressor.add(Dropout(0.01))      #adding input layerand the LSTM layer
regressor.add(Dense(units=1))                                             ␣
 ↪#adding output layers
regressor.compile(optimizer='adam',loss='mean_squared_error')             ␣
 ↪#compiling the RNN
regressor.fit(xtrain,ytrain,batch_size=25,epochs=30)
```

```
Epoch 1/30
76/76 [==============================] - 0s 682us/step - loss: 0.0843
Epoch 2/30
76/76 [==============================] - 0s 656us/step - loss: 0.0427
Epoch 3/30
76/76 [==============================] - 0s 682us/step - loss: 0.0358
Epoch 4/30
76/76 [==============================] - 0s 696us/step - loss: 0.0327
Epoch 5/30
76/76 [==============================] - 0s 682us/step - loss: 0.0313
Epoch 6/30
76/76 [==============================] - 0s 656us/step - loss: 0.0298
Epoch 7/30
76/76 [==============================] - 0s 630us/step - loss: 0.0285
Epoch 8/30
76/76 [==============================] - 0s 654us/step - loss: 0.0269
Epoch 9/30
76/76 [==============================] - 0s 656us/step - loss: 0.0254
Epoch 10/30
76/76 [==============================] - 0s 669us/step - loss: 0.0244
Epoch 11/30
76/76 [==============================] - 0s 706us/step - loss: 0.0218
Epoch 12/30
76/76 [==============================] - 0s 669us/step - loss: 0.0212
Epoch 13/30
```

```
76/76 [==============================] - 0s 720us/step - loss: 0.0195
Epoch 14/30
76/76 [==============================] - 0s 709us/step - loss: 0.0188
Epoch 15/30
76/76 [==============================] - 0s 643us/step - loss: 0.0171
Epoch 16/30
76/76 [==============================] - 0s 643us/step - loss: 0.0153
Epoch 17/30
76/76 [==============================] - 0s 653us/step - loss: 0.0145
Epoch 18/30
76/76 [==============================] - 0s 722us/step - loss: 0.0132
Epoch 19/30
76/76 [==============================] - 0s 782us/step - loss: 0.0115
Epoch 20/30
76/76 [==============================] - 0s 718us/step - loss: 0.0107
Epoch 21/30
76/76 [==============================] - 0s 774us/step - loss: 0.0098
Epoch 22/30
76/76 [==============================] - 0s 722us/step - loss: 0.0088
Epoch 23/30
76/76 [==============================] - 0s 709us/step - loss: 0.0078
Epoch 24/30
76/76 [==============================] - 0s 643us/step - loss: 0.0068
Epoch 25/30
76/76 [==============================] - 0s 682us/step - loss: 0.0063
Epoch 26/30
76/76 [==============================] - 0s 577us/step - loss: 0.0075
Epoch 27/30
76/76 [==============================] - 0s 630us/step - loss: 0.0068
Epoch 28/30
76/76 [==============================] - 0s 617us/step - loss: 0.0041
Epoch 29/30
76/76 [==============================] - 0s 643us/step - loss: 0.0048
Epoch 30/30
76/76 [==============================] - 0s 643us/step - loss: 0.0059
```

[7]:
```
<tensorflow.python.keras.callbacks.History at 0x1e42fd10fd0>
```

[9]:
```
#Get the test set
test_set = pd.read_csv(r"E:\PhD study\ELEG5491 Introduction to Deep␣
↪Learning\bitcoin\datasets\bitcoin1dimtest.csv")
test_set.head()
```

[9]:

	Close
0	7276.803
1	7202.844
2	7218.816

```
3  7191.159
4  7511.589
```

[13]:
```
#Get the prediction result
inputs = test_set.values      #converting to 2D array
inputs = sc.fit_transform(inputs)
inputs = np.reshape(inputs, (471, 1, 1))
predicted_price = regressor.predict(inputs)
predicted_price = sc.inverse_transform(predicted_price)
testScore = sqrt(mean_squared_error(predicted_price, test_set))
print('Test Score: %.2f RMSE' % (testScore))
```

```
Test Score: 3094.80 RMSE
```

[ ]: