import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import keras

train= pd.read\_csv('/content/Price\_Train.csv')

test= pd.read\_csv('/content/Price\_Test.csv')

train.head()

|  | **Date** | **Open** | **High** | **Low** | **Close** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 1/3/2012 | 325.25 | 332.83 | 324.97 | 663.59 | 7,380,500 |
| **1** | 1/4/2012 | 331.27 | 333.87 | 329.08 | 666.45 | 5,749,400 |
| **2** | 1/5/2012 | 329.83 | 330.75 | 326.89 | 657.21 | 6,590,300 |
| **3** | 1/6/2012 | 328.34 | 328.77 | 323.68 | 648.24 | 5,405,900 |
| **4** | 1/9/2012 | 322.04 | 322.29 | 309.46 | 620.76 | 11,688,800 |

train\_open= train.iloc[:, 1:2].values

from sklearn.preprocessing import MinMaxScaler

ss= MinMaxScaler(feature\_range=(0,1))

train\_open\_scaled= ss.fit\_transform(train\_open)

train\_open\_scaled[60]

array([0.08627874])

xtrain=[]

ytrain=[]

for i in range(60,len(train\_open\_scaled)):

    xtrain.append(train\_open\_scaled[i-60:i,0])

    ytrain.append(train\_open\_scaled[i,0])

xtrain, ytrain = np.array(xtrain), np.array(ytrain)

xtrain= np.reshape(xtrain,(xtrain.shape[0],xtrain.shape[1],1))

xtrain.shape

(1198, 60, 1)

from keras.models import Sequential

from keras.layers import LSTM

from keras.layers import Dense

from keras.layers import Dropout

#initialisizng the model

regression= Sequential()

#First Input layer and LSTM layer with 0.2% dropout

regression.add(LSTM(units=50,return\_sequences=True,kernel\_initializer='glorot\_uniform',input\_shape=(xtrain.shape[1],1)))

regression.add(Dropout(0.2))

# Where:

#     return\_sequences: Boolean. Whether to return the last output in the output sequence, or the full sequence.

# Second LSTM layer with 0.2% dropout

regression.add(LSTM(units=50,kernel\_initializer='glorot\_uniform',return\_sequences=True))

regression.add(Dropout(0.2))

#Third LSTM layer with 0.2% dropout

regression.add(LSTM(units=50,kernel\_initializer='glorot\_uniform',return\_sequences=True))

regression.add(Dropout(0.2))

#Fourth LSTM layer with 0.2% dropout, we wont use return sequence true in last layers as we dont want to previous output

regression.add(LSTM(units=50,kernel\_initializer='glorot\_uniform'))

regression.add(Dropout(0.2))

#Output layer , we wont pass any activation as its continous value model

regression.add(Dense(units=1))

#Compiling the network

regression.compile(optimizer='adam',loss='mean\_squared\_error')

#fitting the network

regression.fit(xtrain,ytrain,batch\_size=30,epochs=100)

Epoch 1/100

40/40 [==============================] - 16s 146ms/step - loss: 0.0419

Epoch 2/100

40/40 [==============================] - 6s 154ms/step - loss: 0.0062

Epoch 3/100

40/40 [==============================] - 5s 133ms/step - loss: 0.0056

Epoch 4/100

40/40 [==============================] - 7s 167ms/step - loss: 0.0052

Epoch 5/100

40/40 [==============================] - 5s 136ms/step - loss: 0.0049

Epoch 6/100

40/40 [==============================] - 7s 170ms/step - loss: 0.0049

Epoch 7/100

40/40 [==============================] - 6s 153ms/step - loss: 0.0047

Epoch 8/100

40/40 [==============================] - 7s 178ms/step - loss: 0.0044

Epoch 9/100

40/40 [==============================] - 6s 156ms/step - loss: 0.0045

Epoch 10/100

40/40 [==============================] - 9s 214ms/step - loss: 0.0044

Epoch 11/100

40/40 [==============================] - 9s 232ms/step - loss: 0.0045

Epoch 12/100

40/40 [==============================] - 7s 174ms/step - loss: 0.0042

Epoch 13/100

40/40 [==============================] - 8s 196ms/step - loss: 0.0044

Epoch 14/100

40/40 [==============================] - 6s 146ms/step - loss: 0.0044

Epoch 15/100

40/40 [==============================] - 7s 167ms/step - loss: 0.0046

Epoch 16/100

40/40 [==============================] - 6s 143ms/step - loss: 0.0035

Epoch 17/100

40/40 [==============================] - 7s 167ms/step - loss: 0.0037

Epoch 18/100

40/40 [==============================] - 5s 136ms/step - loss: 0.0036

Epoch 19/100

40/40 [==============================] - 6s 148ms/step - loss: 0.0038

Epoch 20/100

40/40 [==============================] - 6s 148ms/step - loss: 0.0034

Epoch 21/100

40/40 [==============================] - 5s 136ms/step - loss: 0.0035

Epoch 22/100

40/40 [==============================] - 7s 167ms/step - loss: 0.0037

Epoch 23/100

40/40 [==============================] - 5s 133ms/step - loss: 0.0031

Epoch 24/100

40/40 [==============================] - 7s 164ms/step - loss: 0.0035

Epoch 25/100

40/40 [==============================] - 5s 132ms/step - loss: 0.0031

Epoch 26/100

40/40 [==============================] - 7s 168ms/step - loss: 0.0031

Epoch 27/100

40/40 [==============================] - 5s 134ms/step - loss: 0.0029

Epoch 28/100

40/40 [==============================] - 6s 161ms/step - loss: 0.0030

Epoch 29/100

40/40 [==============================] - 5s 134ms/step - loss: 0.0028

Epoch 30/100

40/40 [==============================] - 6s 143ms/step - loss: 0.0031

Epoch 31/100

40/40 [==============================] - 6s 155ms/step - loss: 0.0033

Epoch 32/100

40/40 [==============================] - 5s 134ms/step - loss: 0.0027

Epoch 33/100

40/40 [==============================] - 7s 165ms/step - loss: 0.0026

Epoch 34/100

40/40 [==============================] - 5s 130ms/step - loss: 0.0029

Epoch 35/100

40/40 [==============================] - 6s 162ms/step - loss: 0.0027

Epoch 36/100

40/40 [==============================] - 5s 137ms/step - loss: 0.0026

Epoch 37/100

40/40 [==============================] - 8s 195ms/step - loss: 0.0028

Epoch 38/100

40/40 [==============================] - 6s 138ms/step - loss: 0.0026

Epoch 39/100

40/40 [==============================] - 6s 139ms/step - loss: 0.0027

Epoch 40/100

40/40 [==============================] - 6s 160ms/step - loss: 0.0026

Epoch 41/100

40/40 [==============================] - 5s 133ms/step - loss: 0.0024

Epoch 42/100

40/40 [==============================] - 7s 165ms/step - loss: 0.0023

Epoch 43/100

40/40 [==============================] - 5s 132ms/step - loss: 0.0023

Epoch 44/100

40/40 [==============================] - 7s 168ms/step - loss: 0.0024

Epoch 45/100

40/40 [==============================] - 5s 135ms/step - loss: 0.0026

Epoch 46/100

40/40 [==============================] - 7s 167ms/step - loss: 0.0022

Epoch 47/100

40/40 [==============================] - 5s 132ms/step - loss: 0.0024

Epoch 48/100

40/40 [==============================] - 6s 151ms/step - loss: 0.0021

Epoch 49/100

40/40 [==============================] - 6s 144ms/step - loss: 0.0023

Epoch 50/100

40/40 [==============================] - 5s 135ms/step - loss: 0.0020

Epoch 51/100

40/40 [==============================] - 7s 165ms/step - loss: 0.0024

Epoch 52/100

40/40 [==============================] - 5s 135ms/step - loss: 0.0020

Epoch 53/100

40/40 [==============================] - 7s 166ms/step - loss: 0.0021

Epoch 54/100

40/40 [==============================] - 5s 134ms/step - loss: 0.0019

Epoch 55/100

40/40 [==============================] - 7s 164ms/step - loss: 0.0021

Epoch 56/100

40/40 [==============================] - 5s 134ms/step - loss: 0.0020

Epoch 57/100

40/40 [==============================] - 6s 162ms/step - loss: 0.0019

Epoch 58/100

40/40 [==============================] - 5s 130ms/step - loss: 0.0019

Epoch 59/100

40/40 [==============================] - 6s 139ms/step - loss: 0.0021

Epoch 60/100

40/40 [==============================] - 6s 159ms/step - loss: 0.0018

Epoch 61/100

40/40 [==============================] - 5s 137ms/step - loss: 0.0021

Epoch 62/100

40/40 [==============================] - 7s 169ms/step - loss: 0.0021

Epoch 63/100

40/40 [==============================] - 5s 136ms/step - loss: 0.0019

Epoch 64/100

40/40 [==============================] - 7s 167ms/step - loss: 0.0018

Epoch 65/100

40/40 [==============================] - 5s 134ms/step - loss: 0.0019

Epoch 66/100

40/40 [==============================] - 6s 163ms/step - loss: 0.0019

Epoch 67/100

40/40 [==============================] - 5s 133ms/step - loss: 0.0018

Epoch 68/100

40/40 [==============================] - 6s 156ms/step - loss: 0.0018

Epoch 69/100

40/40 [==============================] - 6s 135ms/step - loss: 0.0018

Epoch 70/100

40/40 [==============================] - 5s 135ms/step - loss: 0.0019

Epoch 71/100

40/40 [==============================] - 7s 163ms/step - loss: 0.0019

Epoch 72/100

40/40 [==============================] - 5s 135ms/step - loss: 0.0017

Epoch 73/100

40/40 [==============================] - 7s 166ms/step - loss: 0.0017

Epoch 74/100

40/40 [==============================] - 6s 138ms/step - loss: 0.0017

Epoch 75/100

40/40 [==============================] - 7s 173ms/step - loss: 0.0018

Epoch 76/100

40/40 [==============================] - 6s 141ms/step - loss: 0.0018

Epoch 77/100

40/40 [==============================] - 8s 193ms/step - loss: 0.0015

Epoch 78/100

40/40 [==============================] - 6s 140ms/step - loss: 0.0020

Epoch 79/100

40/40 [==============================] - 7s 167ms/step - loss: 0.0019

Epoch 80/100

40/40 [==============================] - 5s 134ms/step - loss: 0.0016

Epoch 81/100

40/40 [==============================] - 6s 160ms/step - loss: 0.0016

Epoch 82/100

40/40 [==============================] - 6s 144ms/step - loss: 0.0015

Epoch 83/100

40/40 [==============================] - 6s 144ms/step - loss: 0.0015

Epoch 84/100

40/40 [==============================] - 6s 158ms/step - loss: 0.0017

Epoch 85/100

40/40 [==============================] - 5s 135ms/step - loss: 0.0015

Epoch 86/100

40/40 [==============================] - 7s 169ms/step - loss: 0.0016

Epoch 87/100

40/40 [==============================] - 5s 135ms/step - loss: 0.0018

Epoch 88/100

40/40 [==============================] - 7s 165ms/step - loss: 0.0015

Epoch 89/100

40/40 [==============================] - 5s 137ms/step - loss: 0.0015

Epoch 90/100

40/40 [==============================] - 7s 171ms/step - loss: 0.0015

Epoch 91/100

40/40 [==============================] - 6s 138ms/step - loss: 0.0015

Epoch 92/100

40/40 [==============================] - 7s 169ms/step - loss: 0.0016

Epoch 93/100

40/40 [==============================] - 5s 136ms/step - loss: 0.0015

Epoch 94/100

40/40 [==============================] - 6s 150ms/step - loss: 0.0014

Epoch 95/100

40/40 [==============================] - 6s 150ms/step - loss: 0.0013

Epoch 96/100

40/40 [==============================] - 6s 140ms/step - loss: 0.0015

Epoch 97/100

40/40 [==============================] - 7s 167ms/step - loss: 0.0014

Epoch 98/100

40/40 [==============================] - 5s 136ms/step - loss: 0.0014

Epoch 99/100

40/40 [==============================] - 7s 166ms/step - loss: 0.0014

Epoch 100/100

40/40 [==============================] - 5s 136ms/step - loss: 0.0012

<keras.callbacks.History at 0x78027eeb9720>

test\_open= test.iloc[:, 1:2].values #taking  open price

total= pd.concat([train['Open'],test['Open']],axis=0) # Concating train and test and then will take last 60 train point

test\_input = total[len(total)-len(test)-60:].values

test\_input= test\_input.reshape(-1,1) # reshaping it to get it transformed

test\_input= ss.transform(test\_input)

xtest= []

for i in range(60,80):

    xtest.append(test\_input[i-60:i,0]) #creating input for lstm prediction

xtest= np.array(xtest)

xtest= np.reshape(xtest,(xtest.shape[0],xtest.shape[1],1))

predicted\_value= regression.predict(xtest)

1/1 [==============================] - 2s 2s/step

predicted\_value= ss.inverse\_transform(predicted\_value)

plt.figure(figsize=(20,10))

plt.plot(test\_open,'red',label='Real Prices')

plt.plot(predicted\_value,'blue',label='Predicted Prices')

plt.xlabel('Time')

plt.ylabel('Prices')

plt.title('Real vs Predicted Prices')

plt.legend(loc='best', fontsize=20)

<matplotlib.legend.Legend at 0x78027d86ebc0>

from keras.wrappers.scikit\_learn import KerasRegressor

def reg(optimizer):

    #initialisizng the model

    regression= Sequential()

    #First Input layer and LSTM layer with 0.2% dropout

    regression.add(LSTM(units=50,return\_sequences=True,kernel\_initializer='glorot\_uniform',input\_shape=(xtrain.shape[1],1)))

    regression.add(Dropout(0.2))

    # Second LSTM layer with 0.2% dropout

    regression.add(LSTM(units=50,kernel\_initializer= 'glorot\_uniform',return\_sequences=True))

    regression.add(Dropout(0.2))

    #Third LSTM layer with 0.2% dropout

    regression.add(LSTM(units=50,kernel\_initializer='glorot\_uniform',return\_sequences=True))

    regression.add(Dropout(0.2))

    #Fourth LSTM layer with 0.2% dropout, we wont use return sequence true in last layers as we dont want to previous output

    regression.add(LSTM(units=50,kernel\_initializer='glorot\_uniform'))

    regression.add(Dropout(0.2))

    #Output layer , we wont pass any activation as its continous value model

    regression.add(Dense(units=1))

    #Compiling the network

    regression.compile(optimizer=optimizer,loss='mean\_squared\_error')

    return regression

model= KerasRegressor(build\_fn=reg)

<ipython-input-19-9e456592bc69>:28: DeprecationWarning: KerasRegressor is deprecated, use Sci-Keras (<https://github.com/adriangb/scikeras>) instead. See <https://www.adriangb.com/scikeras/stable/migration.html> for help migrating.

model= KerasRegressor(build\_fn=reg)

from sklearn.model\_selection import RandomizedSearchCV

parameters = {'batch\_size': [50, 32],

              'epochs': [50, 25],

              'optimizer': ['adam', 'rmsprop','sgd','adadelta']}

grid\_search = RandomizedSearchCV(estimator = model,param\_distributions=parameters,n\_iter=5)

# fitting the model and Calculating the best parameters.

grid\_search = grid\_search.fit(xtrain, ytrain)

best\_parameters = grid\_search.best\_params\_

model

<keras.callbacks.History at 0x78027d06a9b0>

predicted\_value= grid\_search.predict(xtest)

predicted\_value= ss.inverse\_transform(predicted\_value.reshape(-1,1))

plt.figure(figsize=(20,10))

plt.plot(test\_open,'red',label='Real Prices')

plt.plot(predicted\_value,'blue',label='Predicted Prices')

plt.xlabel('Time')

plt.ylabel('Prices')

plt.title('Real vs Predicted Prices')

plt.legend(loc='best', fontsize=20)

1/1 [==============================] - 2s 2s/step

<matplotlib.legend.Legend at 0x78026da61270>

