Stock Market Price Prediction using LSTM

October 3, 2023

Susmit Neogi

B.Tech
Mechanical Engineering

IIT Bombay

1 Abstract

Stock market prediction is a challenging task with significant implications for investors and financial markets. This research paper explores the application of Long Short-Term Memory (LSTM) neural networks to predict the open and closing values of Amazon stocks over a ten-year period. We investigate the effectiveness of LSTM networks in capturing temporal dependencies and patterns in historical stock price data.

2 Introduction

The stock market is a complex system influenced by various factors, making accurate predictions challenging. Machine learning techniques have gained popularity in predicting stock prices due to their ability to process vast amounts of data and capture intricate patterns. This study focuses on predicting Amazon stock prices using LSTM neural networks, a type of recurrent neural network (RNN) known for its effectiveness in handling sequential data.

3 Methodology

3.1 Data Collection

We collected historical daily stock price data of 12 years for Amazon till 2020. The dataset includes open and closing values, volume, high and low price. This data has been taken from Kaggle under DJIA Stock 30 Time Series Dataset.

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import torch
import math
import torch.nn as nn
import sklearn
from sklearn.metrics import mean_squared_error
```

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
```

3.2 Data Loading

[2]:
```python
stock=pd.read_csv("amzn.csv")
stock
```

[2]:

	Date	Open	High	Low	Close	Volume
0	1	47.47	47.85	46.25	47.58	7582127
1	2	47.48	47.73	46.69	47.25	7440914
2	3	47.16	48.20	47.11	47.65	5417258
3	4	47.97	48.58	47.32	47.87	6154285
4	5	46.55	47.10	46.40	47.08	8945056
...	...	...	...	...	...	...
3014	3015	1172.08	1174.62	1167.83	1168.36	1585054
3015	3016	1168.36	1178.32	1160.55	1176.76	2005187
3016	3017	1179.91	1187.29	1175.61	1182.26	1867208
3017	3018	1189.00	1190.10	1184.38	1186.10	1841676
3018	3019	1182.35	1184.00	1167.50	1169.47	2688391

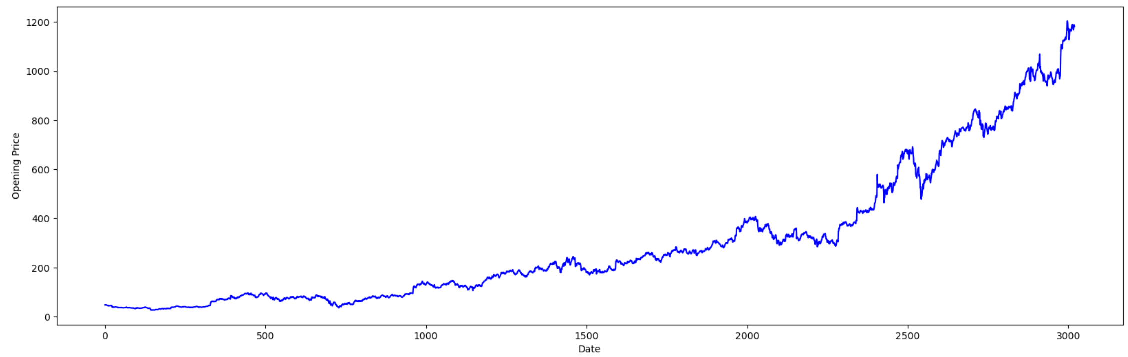
[3019 rows x 6 columns]

[3]:
```python
x=stock['Date']
y1=stock['Open']
y2=stock['Close']
```

4 Next Day Opening Price Prediction

4.1 Training

[4]:
```python
plt.figure(figsize=(20,6))
plt.plot(x,y1, c='b')
plt.xlabel('Date')
plt.ylabel('Opening Price')
f = plt.figure()
f.set_figwidth(10)
plt.show()
```




```
<Figure size 1000x480 with 0 Axes>
```

4.2 Data Normalisation

[5]:
```python
open_price = stock[['Open']]
open_scaler = MinMaxScaler(feature_range=(-1, 1))
open_price['Open'] = open_scaler.fit_transform(open_price['Open'].values.
 ↪reshape(-1,1))
```

```
C:\Users\SusmitPC_3\AppData\Local\Temp\ipykernel_20724\3140400763.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  open_price['Open'] =
open_scaler.fit_transform(open_price['Open'].values.reshape(-1,1))
```

4.3 Data Splitting

As we are using Long Short Term Memory (LSTM) architecture, which comes under Recurrent Neural Networks (RNN), so we need to pass a sequential data as X_input for every Y_output to be predicted. For this, we use the concept of LOOKBACK. It means the number of previous days that we take into account for predicting the value for a particular day. So we take every possible sequences of length LOOKBACK and append it to an array. Now we split this sequence data into training dataset and testing dataset in ratio 80:20.

[6]:
```python
def split_data(stock, lookback):
    data_raw = stock.to_numpy() # convert to numpy array
    data = []

    # create all possible sequences of length seq_len
    for index in range(len(data_raw) - lookback):
        data.append(data_raw[index: index + lookback])
```

```
    data = np.array(data);
    test_set_size = int(np.round(0.2*data.shape[0]));
    train_set_size = data.shape[0] - (test_set_size);

    x_train = data[:train_set_size,:-1,:]
    y_train = data[:train_set_size,-1,:]

    x_test = data[train_set_size:,:-1]
    y_test = data[train_set_size:,-1,:]

    return [x_train, y_train, x_test, y_test]
lookback = 20 # choose sequence length
x_train, open_train, x_test, open_test = split_data(open_price, lookback)
```

Now, we begin our model. But first of all, we need to convert our dataset in the form of numpy arrays into tensors using Pytorch.

[7]:
```
x_train = torch.from_numpy(np.array(x_train)).type(torch.Tensor)
x_test = torch.from_numpy(np.array(x_test)).type(torch.Tensor)
open_train_lstm = torch.from_numpy(np.array(open_train)).type(torch.Tensor)
open_test_lstm = torch.from_numpy(np.array(open_test)).type(torch.Tensor)
```

Now, we define some parameters for our model. The dimension of input and output data is 1. We take 2 hidden layers with dimension 32. For training and testing dataset, we take number of epochs to be 150.

[8]:
```
input_dim = 1
hidden_dim = 32
num_layers = 2
output_dim = 1
num_epochs = 150
```

Finally, we create our LSTM Model.

[9]:
```
class LSTM(nn.Module):
    def __init__(self, input_dim, hidden_dim, num_layers, output_dim):
        super(LSTM, self).__init__()
        self.hidden_dim = hidden_dim
        self.num_layers = num_layers

        self.lstm = nn.LSTM(input_dim, hidden_dim, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_dim, output_dim)
    def forward(self, x):
        h0 = torch.zeros(self.num_layers, x.size(0), self.hidden_dim).
↪requires_grad_()
        c0 = torch.zeros(self.num_layers, x.size(0), self.hidden_dim).
↪requires_grad_()
```

```
        out, (hn, cn) = self.lstm(x, (h0.detach(), c0.detach()))
        out = self.fc(out[:, -1, :])
        return out
```

[10]:
```
model = LSTM(input_dim=input_dim, hidden_dim=hidden_dim, output_dim=output_dim,␣
 ↪num_layers=num_layers)
criterion = torch.nn.MSELoss(reduction='mean')
optimiser = torch.optim.Adam(model.parameters(), lr=0.01)
```

Once we have created our model, we train our model using our training dataset and print the Mean Squred Error (MSE) for each epoch. As expected, the MSE decreases with number of epochs. This implies, our optimiser in the model is working perfectly.

[11]:
```
import time
hist = np.zeros(num_epochs)
start_time = time.time()
lstm = []
loss_val=[]
for t in range(num_epochs):
    y_train_pred = model(x_train)
    loss = criterion(y_train_pred, open_train_lstm)
    print("Epoch ", t, "MSE: ", loss.item())
    hist[t] = loss.item()
    optimiser.zero_grad()
    loss.backward()
    optimiser.step()
    loss_val.append(loss.item())
training_time = time.time()-start_time
print("Training time: {}".format(training_time))
```

```
Epoch  0 MSE:  0.7647774815559387
Epoch  1 MSE:  0.5510662794113159
Epoch  2 MSE:  0.320406436920166
Epoch  3 MSE:  0.06965189427137375
Epoch  4 MSE:  0.19082149863243103
Epoch  5 MSE:  0.14434009790420532
Epoch  6 MSE:  0.055115483701229095
Epoch  7 MSE:  0.03570643812417984
Epoch  8 MSE:  0.05492817237973213
Epoch  9 MSE:  0.07279935479164124
Epoch  10 MSE:  0.0776025578379631
Epoch  11 MSE:  0.07190708816051483
Epoch  12 MSE:  0.06105731800198555
Epoch  13 MSE:  0.04975711926817894
Epoch  14 MSE:  0.041243474930524826
Epoch  15 MSE:  0.037110112607479095
Epoch  16 MSE:  0.03732771798968315
Epoch  17 MSE:  0.04045694321393967
```

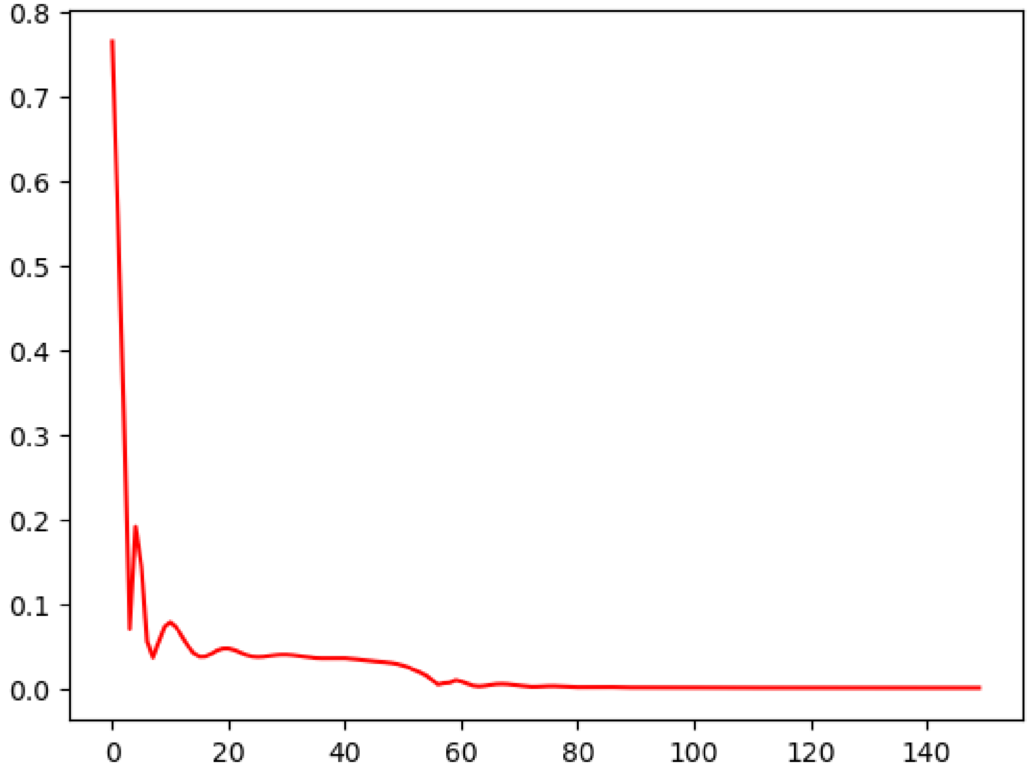
```
Epoch  18 MSE:  0.04425065591931343
Epoch  19 MSE:  0.04667714610695839
Epoch  20 MSE:  0.04678264632821083
Epoch  21 MSE:  0.04485180974006653
Epoch  22 MSE:  0.0419386625289917
Epoch  23 MSE:  0.039198990911245346
Epoch  24 MSE:  0.037408776581287384
Epoch  25 MSE:  0.03680329769849777
Epoch  26 MSE:  0.0371687859296798 7
Epoch  27 MSE:  0.038048919290304184
Epoch  28 MSE:  0.03895215690135956
Epoch  29 MSE:  0.039499539881944656
Epoch  30 MSE:  0.039498310536146164
Epoch  31 MSE:  0.038951024413108826
Epoch  32 MSE:  0.03801781311631203
Epoch  33 MSE:  0.036949120461940765
Epoch  34 MSE:  0.0360044501721859
Epoch  35 MSE:  0.035373881459236145
Epoch  36 MSE:  0.035121217370033264
Epoch  37 MSE:  0.035167671740055084
Epoch  38 MSE:  0.03532629832625389
Epoch  39 MSE:  0.035378843545913696
Epoch  40 MSE:  0.035164713859558105
Epoch  41 MSE:  0.03464119881391525
Epoch  42 MSE:  0.03388669341802597
Epoch  43 MSE:  0.033048342913389206
Epoch  44 MSE:  0.032264262437820435
Epoch  45 MSE:  0.0315987691283226
Epoch  46 MSE:  0.031015614047646523
Epoch  47 MSE:  0.03039151430130005
Epoch  48 MSE:  0.02955467440187931
Epoch  49 MSE:  0.028327718377113342
Epoch  50 MSE:  0.02656213380396366
Epoch  51 MSE:  0.024174420163035393
Epoch  52 MSE:  0.02123979479074478
Epoch  53 MSE:  0.01813780516386032
Epoch  54 MSE:  0.014852311462163925
Epoch  55 MSE:  0.009231407195329666
Epoch  56 MSE:  0.0042311400175094604
Epoch  57 MSE:  0.005809180438518524
Epoch  58 MSE:  0.006343638524413109
Epoch  59 MSE:  0.00931602530181408
Epoch  60 MSE:  0.00794310960918665
Epoch  61 MSE:  0.005166448652744293
Epoch  62 MSE:  0.0028894057031720877
Epoch  63 MSE:  0.0019669309258461
Epoch  64 MSE:  0.0025547111872583628
Epoch  65 MSE:  0.003694621380418539
```

```
Epoch  66 MSE:  0.004565088078379631
Epoch  67 MSE:  0.004833885934203863
Epoch  68 MSE:  0.004479583352804184
Epoch  69 MSE:  0.0036694398149847984
Epoch  70 MSE:  0.0026886994019150734
Epoch  71 MSE:  0.0018576446454972029
Epoch  72 MSE:  0.0014368111733347178
Epoch  73 MSE:  0.0015011123614385724
Epoch  74 MSE:  0.0018591894768178463
Epoch  75 MSE:  0.002187095582485199
Epoch  76 MSE:  0.002222997834905982
Epoch  77 MSE:  0.0019285011803731322
Epoch  78 MSE:  0.0015158462338149548
Epoch  79 MSE:  0.001191324437968433
Epoch  80 MSE:  0.0010478560579940677
Epoch  81 MSE:  0.001096248859539628
Epoch  82 MSE:  0.0012292491737753153
Epoch  83 MSE:  0.0013430275721475482
Epoch  84 MSE:  0.001379068358801305З
Epoch  85 MSE:  0.0013050022535026073
Epoch  86 MSE:  0.00115004344843328
Epoch  87 MSE:  0.0009648025734350085
Epoch  88 MSE:  0.0008050304022617638
Epoch  89 MSE:  0.0007278620032593608
Epoch  90 MSE:  0.000735454901587218
Epoch  91 MSE:  0.0007937896880321205
Epoch  92 MSE:  0.0008412112947553396
Epoch  93 MSE:  0.0008217683061957359
Epoch  94 MSE:  0.0007409361423924565
Epoch  95 MSE:  0.0006365572917275131
Epoch  96 MSE:  0.0005612552631646395
Epoch  97 MSE:  0.0005386951379477978
Epoch  98 MSE:  0.0005551757058128715
Epoch  99 MSE:  0.0005828730063512921
Epoch  100 MSE:  0.0005911172484047711
Epoch  101 MSE:  0.0005697126034647226
Epoch  102 MSE:  0.0005258837481960654
Epoch  103 MSE:  0.00047875355812720954
Epoch  104 MSE:  0.0004505914985202253
Epoch  105 MSE:  0.00044738606084138155
Epoch  106 MSE:  0.00046020300942473114
Epoch  107 MSE:  0.00046825449680909514
Epoch  108 MSE:  0.0004566299030557275
Epoch  109 MSE:  0.0004290929064154625
Epoch  110 MSE:  0.00039976637344807386
Epoch  111 MSE:  0.0003831679350696504
Epoch  112 MSE:  0.0003806270251516253
Epoch  113 MSE:  0.0003843160520773381
```

```
Epoch  114 MSE:  0.0003839567070826888
Epoch  115 MSE:  0.00037406900082714856
Epoch  116 MSE:  0.0003575040900614126
Epoch  117 MSE:  0.00034085867810063064
Epoch  118 MSE:  0.00033093904494307935
Epoch  119 MSE:  0.0003283970872871578
Epoch  120 MSE:  0.0003288240113761276
Epoch  121 MSE:  0.0003262359241489321
Epoch  122 MSE:  0.0003182794607710093
Epoch  123 MSE:  0.0003079261223319918
Epoch  124 MSE:  0.00029954855563119054
Epoch  125 MSE:  0.00029566738521680236
Epoch  126 MSE:  0.0002946220338344574
Epoch  127 MSE:  0.00029336486477404833
Epoch  128 MSE:  0.00028949195984750986
Epoch  129 MSE:  0.0002834081242326647
Epoch  130 MSE:  0.00027723106904886663
Epoch  131 MSE:  0.0002730549604166299
Epoch  132 MSE:  0.0002711516572162509
Epoch  133 MSE:  0.0002699307515285909
Epoch  134 MSE:  0.0002676528238225728 3
Epoch  135 MSE:  0.0002638345758896321
Epoch  136 MSE:  0.00025967712281271815
Epoch  137 MSE:  0.0002564993337728083
Epoch  138 MSE:  0.0002547317126300186
Epoch  139 MSE:  0.00025350681971758604
Epoch  140 MSE:  0.0002518333785701543
Epoch  141 MSE:  0.00024931394727900624
Epoch  142 MSE:  0.0002465304278302938
Epoch  143 MSE:  0.0002442452241666615
Epoch  144 MSE:  0.0002427723811706528
Epoch  145 MSE:  0.00024166033836081624
Epoch  146 MSE:  0.00024029030464589596
Epoch  147 MSE:  0.00023844106181059033
Epoch  148 MSE:  0.00023647386115044355
Epoch  149 MSE:  0.0002348512934986502
Training time: 18.633631229400635
```

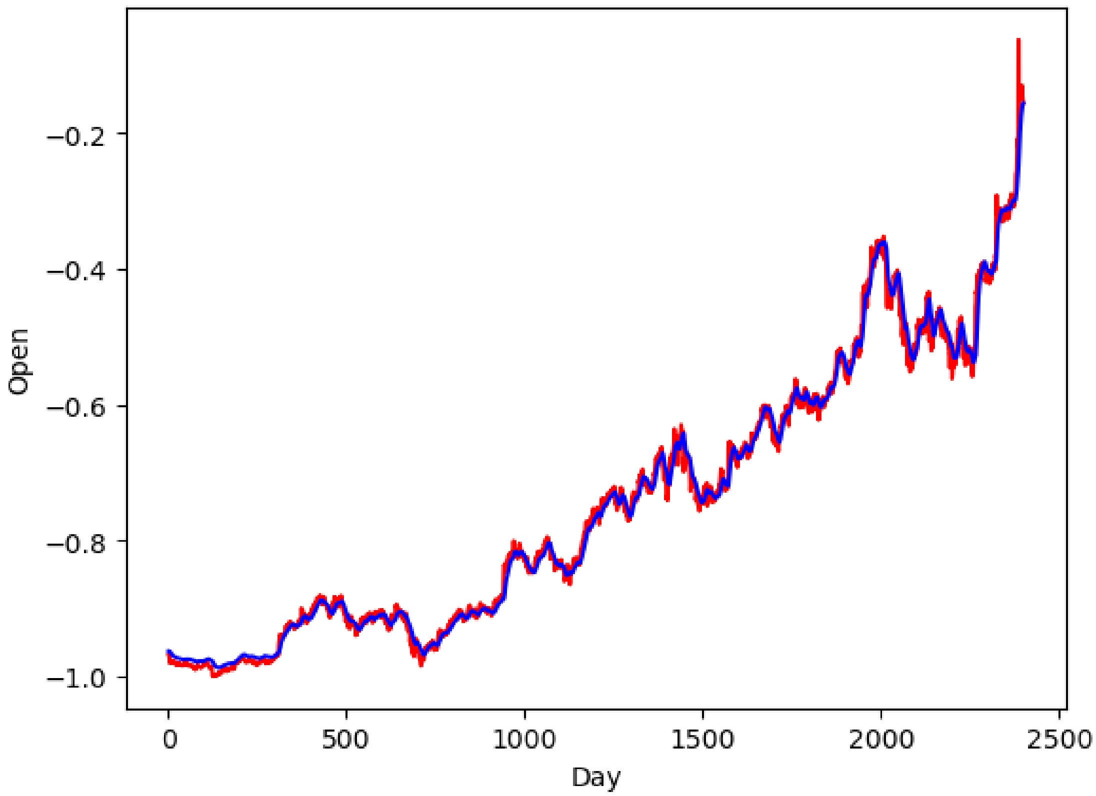
Below is the visualisation of the relation between MSE value with number of Epochs.

[12]:
```python
epoch=[i for i in range(num_epochs)]
plt.plot(epoch,loss_val,c='r')
plt.show()
```



Now, we visualise and calculate, how well is our prediction on the training dataset.

[13]:
```
y_train_pred = model(x_train)
plt.plot(stock['Date'][:2399],(open_train_lstm), c='r')
plt.plot(stock['Date'][:2399], (y_train_pred).detach().numpy(), c='b')
plt.xlabel('Day')
plt.ylabel('Open')
plt.show()
```



[14]:
```
trainScore = math.sqrt(mean_squared_error(open_train[:,0], (y_train_pred).
 ↪detach().numpy()[:,0]))
print(f'Train Score: {trainScore*100} % RMSE ')
```

```
Train Score: 1.5286849700606244 % RMSE
```

Thus, the Root Mean Square Error percent of our model in training dataset is 1.53%

4.4 Testing

Now, we perform the most important part- Testing of our model on test dataset.

[15]:
```
import time
hist = np.zeros(num_epochs)
start_time = time.time()
lstm = []
loss_val_test=[]
for t in range(num_epochs):
    y_test_pred = model(x_test)
    loss = criterion(y_test_pred, open_test_lstm)
    print("Epoch ", t, "MSE: ", loss.item())
    hist[t] = loss.item()
```

```
    optimiser.zero_grad()
    loss.backward()
    optimiser.step()
    loss_val_test.append(loss.item())
testing_time = time.time()-start_time
print("Testing time: {}".format(testing_time))
```

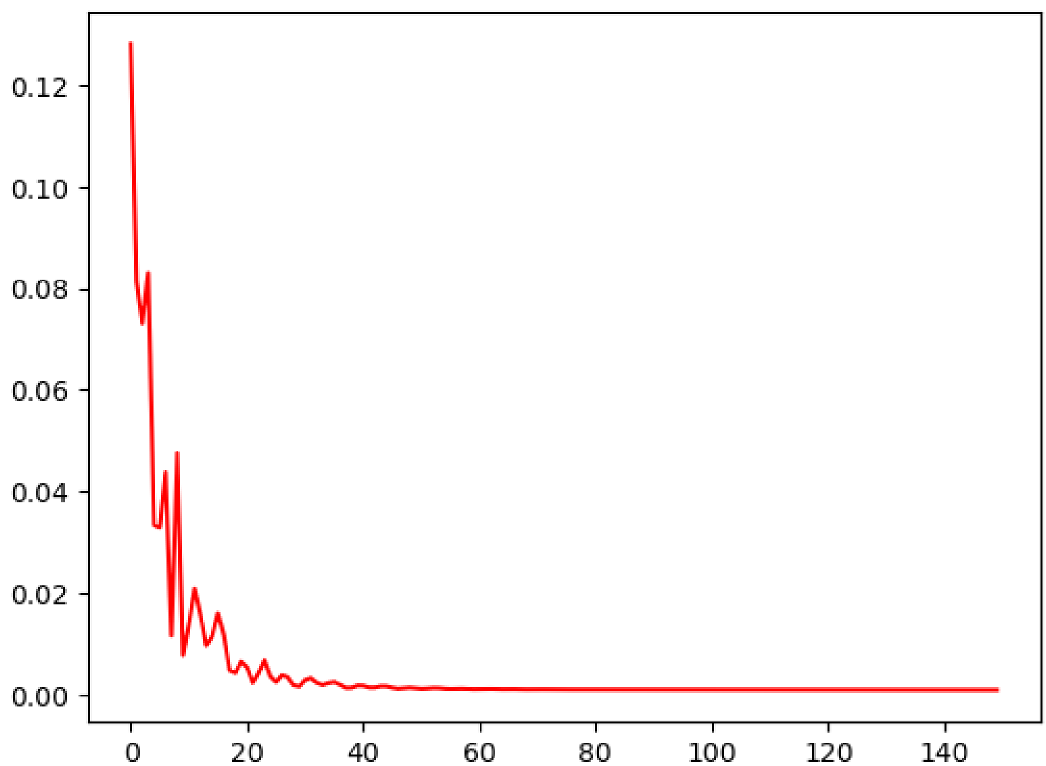
```
Epoch  0 MSE:  0.1281680315732956
Epoch  1 MSE:  0.08129425346851349
Epoch  2 MSE:  0.07314121723175049
Epoch  3 MSE:  0.08306019753217697
Epoch  4 MSE:  0.03330494090914726
Epoch  5 MSE:  0.03284766525030136
Epoch  6 MSE:  0.04388337954878807
Epoch  7 MSE:  0.01161788310855627
Epoch  8 MSE:  0.04761037975549698
Epoch  9 MSE:  0.007757868617773056
Epoch  10 MSE:  0.01373734325170517
Epoch  11 MSE:  0.020871244370937347
Epoch  12 MSE:  0.015603607520461082
Epoch  13 MSE:  0.009603855200111866
Epoch  14 MSE:  0.01148057822138071
Epoch  15 MSE:  0.016018839552998543
Epoch  16 MSE:  0.011762741021811962
Epoch  17 MSE:  0.004750784020870924
Epoch  18 MSE:  0.004258454777300358
Epoch  19 MSE:  0.00651119789108634
Epoch  20 MSE:  0.00536726787686348
Epoch  21 MSE:  0.002323766937479377
Epoch  22 MSE:  0.004242870025336742
Epoch  23 MSE:  0.006815090775489807
Epoch  24 MSE:  0.003556556301191449
Epoch  25 MSE:  0.0024231288116425276
Epoch  26 MSE:  0.003772566793486476
Epoch  27 MSE:  0.003444052068516612
Epoch  28 MSE:  0.0018365908181294799
Epoch  29 MSE:  0.0015779139939695597
Epoch  30 MSE:  0.0028555195312947035
Epoch  31 MSE:  0.0032343126367777586
Epoch  32 MSE:  0.0023222833406180143
Epoch  33 MSE:  0.001877278438769281
Epoch  34 MSE:  0.0022696959786117077
Epoch  35 MSE:  0.0024745415430516005
Epoch  36 MSE:  0.00195457530207932
Epoch  37 MSE:  0.0013039844343438745
Epoch  38 MSE:  0.0013295664684846997
Epoch  39 MSE:  0.00176485616248101
Epoch  40 MSE:  0.0017351069254800677
```

```
Epoch  41 MSE:  0.001377010834403336
Epoch  42 MSE:  0.0013789883814752102
Epoch  43 MSE:  0.0016384285409003496
Epoch  44 MSE:  0.0016320180147886276
Epoch  45 MSE:  0.0013319619465619326
Epoch  46 MSE:  0.0011526976013556123
Epoch  47 MSE:  0.0012562520569190383
Epoch  48 MSE:  0.001348210615105927
Epoch  49 MSE:  0.0012408194597810507
Epoch  50 MSE:  0.0011308725224807858
Epoch  51 MSE:  0.0011844323016703129
Epoch  52 MSE:  0.001284475321881473
Epoch  53 MSE:  0.001266045612283051
Epoch  54 MSE:  0.0011552668875083327
Epoch  55 MSE:  0.0010977057972922921
Epoch  56 MSE:  0.0011320988414809108
Epoch  57 MSE:  0.0011499724350869656
Epoch  58 MSE:  0.0010941469809040427
Epoch  59 MSE:  0.0010476488387212157
Epoch  60 MSE:  0.0010731805814430118
Epoch  61 MSE:  0.001112717785872519
Epoch  62 MSE:  0.0010974552715197206
Epoch  63 MSE:  0.001053099986165762
Epoch  64 MSE:  0.0010418164310976863
Epoch  65 MSE:  0.0010578356450423598
Epoch  66 MSE:  0.0010505676036700606
Epoch  67 MSE:  0.0010194919304922223
Epoch  68 MSE:  0.0010076954495161772
Epoch  69 MSE:  0.0010230409679934382
Epoch  70 MSE:  0.0010319628054276109
Epoch  71 MSE:  0.001018858514726162
Epoch  72 MSE:  0.0010050894925370812
Epoch  73 MSE:  0.0010074687888845801
Epoch  74 MSE:  0.0010117373894900084
Epoch  75 MSE:  0.001001875032670796
Epoch  76 MSE:  0.0009884671308100224
Epoch  77 MSE:  0.0009875273099169135
Epoch  78 MSE:  0.0009933756664395332
Epoch  79 MSE:  0.0009918475989252329
Epoch  80 MSE:  0.0009844813030213118
Epoch  81 MSE:  0.0009823006112128496
Epoch  82 MSE:  0.0009849824709817767
Epoch  83 MSE:  0.0009830945637077093
Epoch  84 MSE:  0.000976142764557153
Epoch  85 MSE:  0.0009723452967591584
Epoch  86 MSE:  0.000973476970102638
Epoch  87 MSE:  0.0009734280756674707
Epoch  88 MSE:  0.0009699970250949264
```

```
Epoch  89 MSE:  0.0009675654000602663
Epoch  90 MSE:  0.0009681074880063534
Epoch  91 MSE:  0.0009679458453319967
Epoch  92 MSE:  0.000964952225331217
Epoch  93 MSE:  0.0009620285127311945
Epoch  94 MSE:  0.000961389159783721
Epoch  95 MSE:  0.0009610916022211313
Epoch  96 MSE:  0.0009592861752025783
Epoch  97 MSE:  0.0009573949500918388
Epoch  98 MSE:  0.000956953561399132
Epoch  99 MSE:  0.0009567475062794983
Epoch  100 MSE:  0.0009552950505167246
Epoch  101 MSE:  0.0009534551645629108
Epoch  102 MSE:  0.0009525117347948253
Epoch  103 MSE:  0.0009519168525002897
Epoch  104 MSE:  0.0009506905335001647
Epoch  105 MSE:  0.0009492810349911451
Epoch  106 MSE:  0.000948512926697731
Epoch  107 MSE:  0.0009480127482675016
Epoch  108 MSE:  0.0009470251970924437
Epoch  109 MSE:  0.0009457762935198843
Epoch  110 MSE:  0.0009448688360862434
Epoch  111 MSE:  0.0009441495058126748
Epoch  112 MSE:  0.0009431582875549793
Epoch  113 MSE:  0.0009420463466085494
Epoch  114 MSE:  0.0009412160725332797
Epoch  115 MSE:  0.0009405480814166367
Epoch  116 MSE:  0.0009396907407790422
Epoch  117 MSE:  0.0009387138416059315
Epoch  118 MSE:  0.0009378837421536446
Epoch  119 MSE:  0.0009371343185193837
Epoch  120 MSE:  0.0009362560231238604
Epoch  121 MSE:  0.0009353206842206419
Epoch  122 MSE:  0.000934512703679502
Epoch  123 MSE:  0.000933778181206435
Epoch  124 MSE:  0.0009329644963145256
Epoch  125 MSE:  0.0009321098332293332
Epoch  126 MSE:  0.0009313260670751333
Epoch  127 MSE:  0.000930570182390511
Epoch  128 MSE:  0.0009297510841861367
Epoch  129 MSE:  0.0009289112640544772
Epoch  130 MSE:  0.0009281307575292885
Epoch  131 MSE:  0.0009273765608668327
Epoch  132 MSE:  0.0009265872067771852
Epoch  133 MSE:  0.0009257907513529062
Epoch  134 MSE:  0.0009250310831703246
Epoch  135 MSE:  0.0009242775849997997
Epoch  136 MSE:  0.0009234938770532608
```

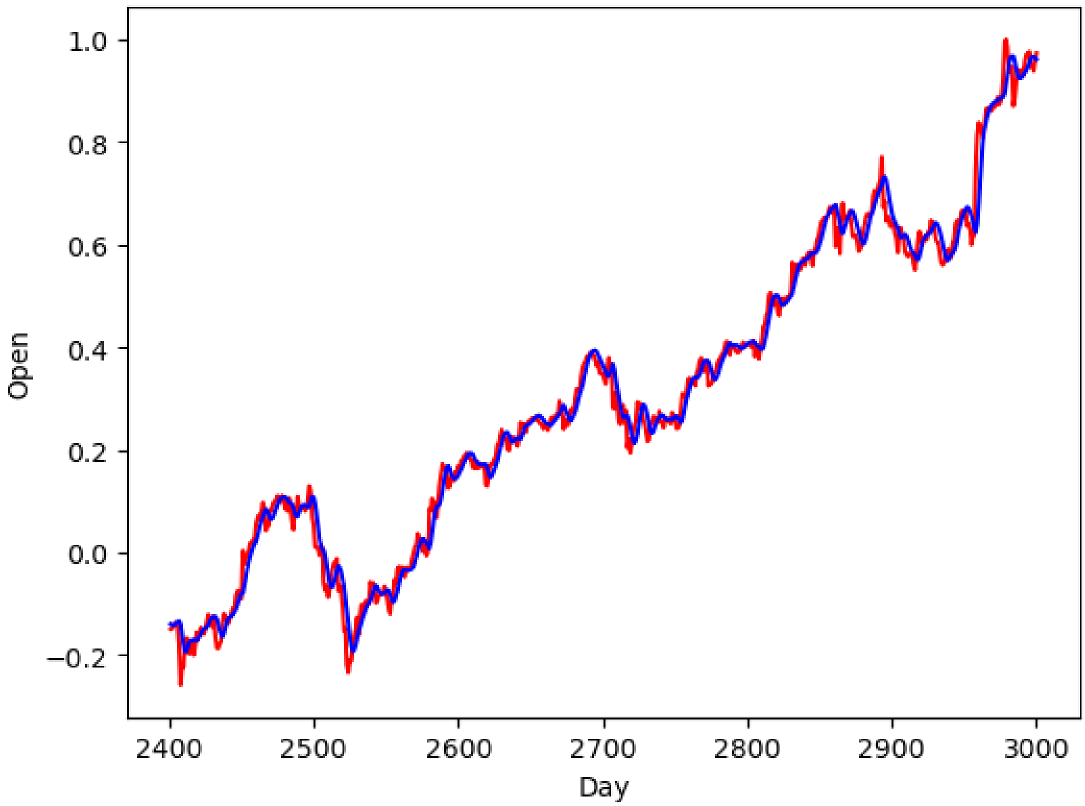
```
Epoch  137 MSE:  0.0009227076079696417
Epoch  138 MSE:  0.0009219490457326174
Epoch  139 MSE:  0.0009211962460540235
Epoch  140 MSE:  0.0009204283123835921
Epoch  141 MSE:  0.0009196646278724074
Epoch  142 MSE:  0.0009189192205667496
Epoch  143 MSE:  0.0009181727073155344
Epoch  144 MSE:  0.00091741350479424
Epoch  145 MSE:  0.0009166580275632441
Epoch  146 MSE:  0.0009159144829027355
Epoch  147 MSE:  0.0009151702979579568
Epoch  148 MSE:  0.0009144206997007132
Epoch  149 MSE:  0.0009136770968325436
Testing time: 5.911567687988281
```

[16]:
```
epoch=[i for i in range(num_epochs)]
plt.plot(epoch,loss_val_test,c='r')
plt.show()
```



[17]:
```
y_test_pred = model(x_test)
plt.plot(stock['Date'][2400:3000],(open_test_lstm), c='r')
plt.plot(stock['Date'][2400:3000], (y_test_pred).detach().numpy(), c='b')
```

```
plt.xlabel('Day')
plt.ylabel('Open')
plt.show()
```



[18]:
```
testScore = math.sqrt(mean_squared_error(open_test[:,0], (y_test_pred).detach().
↪numpy()[:,0]))
print(f'Test Score: {testScore*100} % RMSE ')
```

```
Test Score: 3.0214925850566177 % RMSE
```

Thus, the Root Mean Square Error of our model on Testing dataset is 3.02%. Hence, the performance of our model is quite

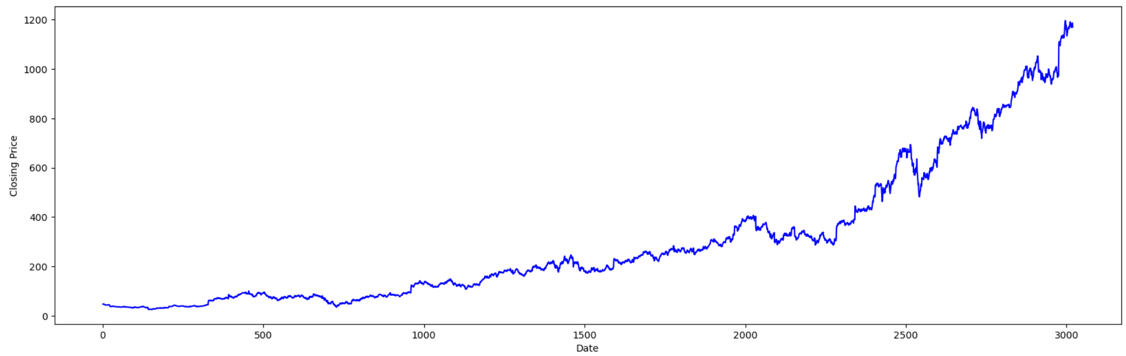
5 Next Day Closing Price Prediction

5.1 Training

We will do a similar set of steps in case of predicting closing price of AMZN stocks for a period of 12 years.

[19]:
```
plt.figure(figsize=(20,6))
plt.plot(x,y2, c='b')
```

```
plt.xlabel('Date')
plt.ylabel('Closing Price')
f = plt.figure()
f.set_figwidth(10)
plt.show()
```




```
<Figure size 1000x480 with 0 Axes>
```

[20]:
```
close_price = stock[['Close']]
close_scaler = MinMaxScaler(feature_range=(-1, 1))
close_price['Close'] = close_scaler.fit_transform(close_price['Close'].values.
 ↪reshape(-1,1))
```

```
C:\Users\SusmitPC_3\AppData\Local\Temp\ipykernel_20724\3375544693.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  close_price['Close'] =
close_scaler.fit_transform(close_price['Close'].values.reshape(-1,1))
```

[21]:
```
lookback=30
x2_train, close_train, x2_test, close_test = split_data(open_price, lookback)
```

[22]:
```
x2_train = torch.from_numpy(np.array(x2_train)).type(torch.Tensor)
x2_test = torch.from_numpy(np.array(x2_test)).type(torch.Tensor)
close_train_lstm = torch.from_numpy(np.array(close_train)).type(torch.Tensor)
close_test_lstm = torch.from_numpy(np.array(close_test)).type(torch.Tensor)
```

[23]:
```
import time
hist = np.zeros(num_epochs)
start_time = time.time()
lstm = []
```

```python
loss_val_close=[]
for t in range(num_epochs):
    y_train_pred = model(x2_train)
    loss = criterion(y_train_pred, close_train_lstm)
    print("Epoch ", t, "MSE: ", loss.item())
    hist[t] = loss.item()
    optimiser.zero_grad()
    loss.backward()
    optimiser.step()
    loss_val_close.append(loss.item())
training_time = time.time()-start_time
print("Training time: {}".format(training_time))
```

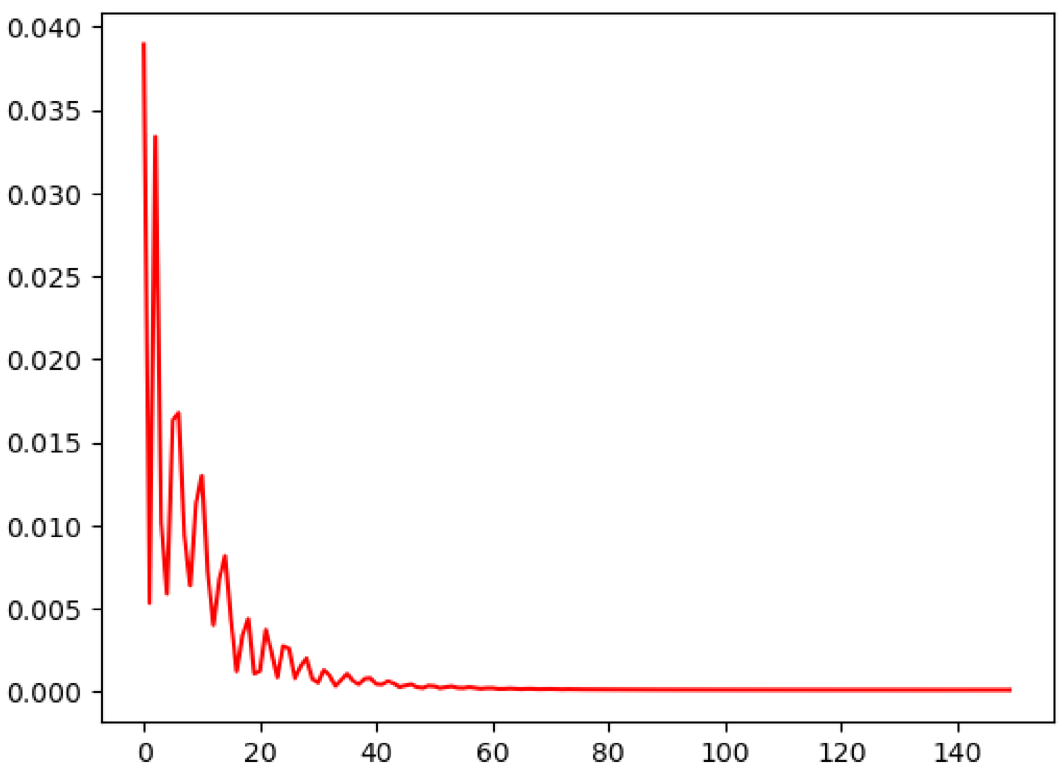
```
Epoch  0 MSE:  0.03895028308033943
Epoch  1 MSE:  0.005333165638148785
Epoch  2 MSE:  0.03339124098420143
Epoch  3 MSE:  0.010186548344790936
Epoch  4 MSE:  0.005904796067625284
Epoch  5 MSE:  0.016338054090738297
Epoch  6 MSE:  0.01678246259689331
Epoch  7 MSE:  0.009387219324707985
Epoch  8 MSE:  0.006384963169693947
Epoch  9 MSE:  0.011342295445501804
Epoch  10 MSE:  0.01299329474568367
Epoch  11 MSE:  0.007179200649261475
Epoch  12 MSE:  0.004012695979326963
Epoch  13 MSE:  0.0067667486146092415
Epoch  14 MSE:  0.008166331797838211
Epoch  15 MSE:  0.004403274040669203
Epoch  16 MSE:  0.0012254492612555623
Epoch  17 MSE:  0.0034114893060177565
Epoch  18 MSE:  0.00438624108210206
Epoch  19 MSE:  0.0010889222612604499
Epoch  20 MSE:  0.0012338345404714346
Epoch  21 MSE:  0.003727769711986184
Epoch  22 MSE:  0.0023866777773946524
Epoch  23 MSE:  0.0008693902054801583
Epoch  24 MSE:  0.002756645204499364
Epoch  25 MSE:  0.002583250170573592
Epoch  26 MSE:  0.0008074995712377131
Epoch  27 MSE:  0.001537327654659748
Epoch  28 MSE:  0.002029613358899951
Epoch  29 MSE:  0.0007836160366423428
Epoch  30 MSE:  0.0005228013615123928
Epoch  31 MSE:  0.0013074390590190887
Epoch  32 MSE:  0.0009680011426098645
Epoch  33 MSE:  0.0003460960288066417
Epoch  34 MSE:  0.0007282800506800413
```

```
Epoch  35 MSE:  0.0010639813262969255
Epoch  36 MSE:  0.0006613549776375294
Epoch  37 MSE:  0.00044586212607100606
Epoch  38 MSE:  0.0007837837911210954
Epoch  39 MSE:  0.0008206644561141729
Epoch  40 MSE:  0.0004613923083525151
Epoch  41 MSE:  0.00043683298281393945
Epoch  42 MSE:  0.0006369652110151947
Epoch  43 MSE:  0.0005077103851363063
Epoch  44 MSE:  0.0002766670077107847
Epoch  45 MSE:  0.00036191928666085005
Epoch  46 MSE:  0.00044757293653674424
Epoch  47 MSE:  0.00028009506058879197
Epoch  48 MSE:  0.00022785950568504632
Epoch  49 MSE:  0.0003526503569446504
Epoch  50 MSE:  0.0003223384846933186
Epoch  51 MSE:  0.00021608249517157674
Epoch  52 MSE:  0.00027528463397175074
Epoch  53 MSE:  0.0003214000607840717
Epoch  54 MSE:  0.0002337345649721101
Epoch  55 MSE:  0.00021973792172502726
Epoch  56 MSE:  0.00027382589178159833
Epoch  57 MSE:  0.00023335799050983042
Epoch  58 MSE:  0.0001818302844185382
Epoch  59 MSE:  0.00021333956101443619
Epoch  60 MSE:  0.00021543152979575098
Epoch  61 MSE:  0.0001679991983110085
Epoch  62 MSE:  0.00017235575069207698
Epoch  63 MSE:  0.00019500193593557924
Epoch  64 MSE:  0.00017069617751985788
Epoch  65 MSE:  0.00015569324023090303
Epoch  66 MSE:  0.00017610911163501441
Epoch  67 MSE:  0.0001728575152810663
Epoch  68 MSE:  0.00015231258294079453
Epoch  69 MSE:  0.00015900490689091384
Epoch  70 MSE:  0.00016604062693659216
Epoch  71 MSE:  0.00015022621664684266
Epoch  72 MSE:  0.00014477618969976902
Epoch  73 MSE:  0.0001528484863229096
Epoch  74 MSE:  0.00014567792823072523
Epoch  75 MSE:  0.00013582585961557925
Epoch  76 MSE:  0.00014053989434614778
Epoch  77 MSE:  0.00014051383186597377
Epoch  78 MSE:  0.00013195934297982603
Epoch  79 MSE:  0.00013272663636598736
Epoch  80 MSE:  0.0001359611051157117
Epoch  81 MSE:  0.00013061673962511122
Epoch  82 MSE:  0.0001285231701331213
```

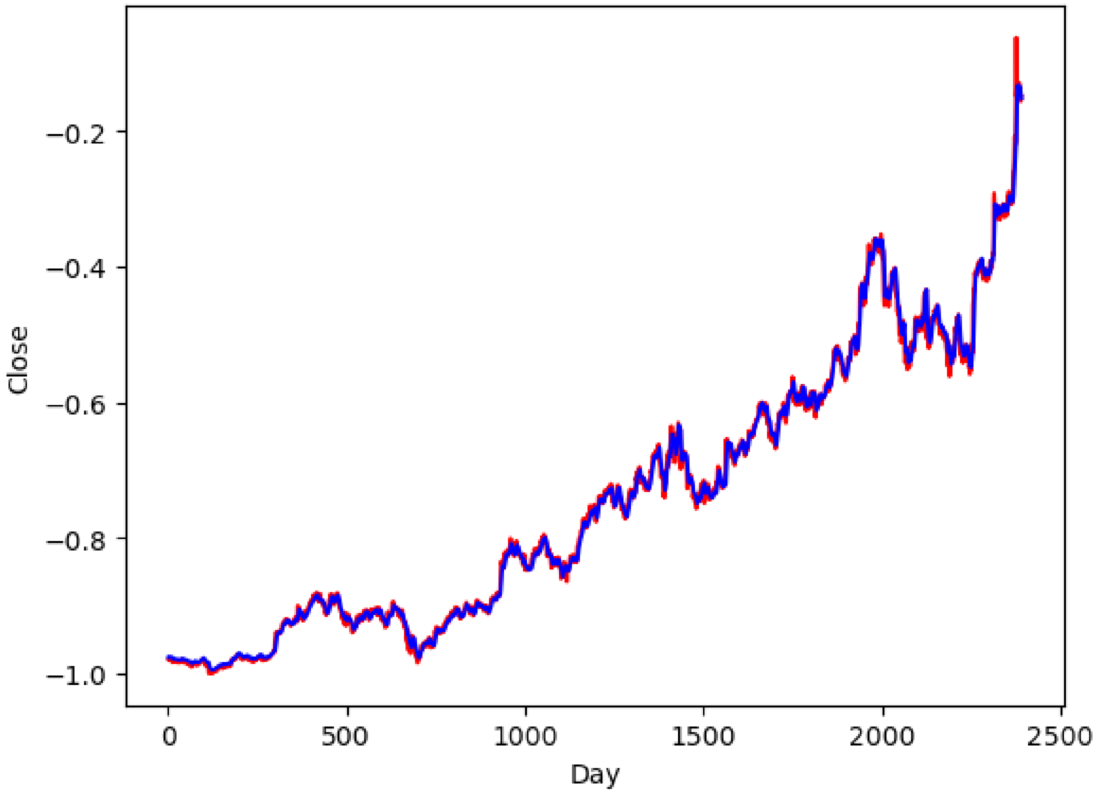
```
Epoch  83 MSE:  0.00013167186989448965
Epoch  84 MSE:  0.00012912831152789295
Epoch  85 MSE:  0.00012560770846903324
Epoch  86 MSE:  0.00012724389671348035
Epoch  87 MSE:  0.00012652724399231374
Epoch  88 MSE:  0.00012304481060709804
Epoch  89 MSE:  0.00012323590635787696
Epoch  90 MSE:  0.00012355744547676295
Epoch  91 MSE:  0.00012099832383682951
Epoch  92 MSE:  0.00012030879588564858
Epoch  93 MSE:  0.00012102836626581848
Epoch  94 MSE:  0.00011955642548855394
Epoch  95 MSE:  0.0001184892826131545
Epoch  96 MSE:  0.00011908759915968403
Epoch  97 MSE:  0.00011841044761240482
Epoch  98 MSE:  0.00011724464275175706
Epoch  99 MSE:  0.00011746126983780414
Epoch  100 MSE:  0.00011721077316906303
Epoch  101 MSE:  0.00011614632967393845
Epoch  102 MSE:  0.00011600364814512432
Epoch  103 MSE:  0.00011594512034207582
Epoch  104 MSE:  0.00011511521734064445
Epoch  105 MSE:  0.00011477059160824865
Epoch  106 MSE:  0.00011478953092591837
Epoch  107 MSE:  0.00011423225078033283
Epoch  108 MSE:  0.00011383226956240833
Epoch  109 MSE:  0.00011385617835912853
Epoch  110 MSE:  0.00011351741704856977
Epoch  111 MSE:  0.00011312790593365207
Epoch  112 MSE:  0.00011310523404972628
Epoch  113 MSE:  0.00011289668327663094
Epoch  114 MSE:  0.00011254075798206031
Epoch  115 MSE:  0.0001124563641496934
Epoch  116 MSE:  0.00011231406824663281
Epoch  117 MSE:  0.00011200597509741783
Epoch  118 MSE:  0.00011187823110958561
Epoch  119 MSE:  0.00011177380656590685
Epoch  120 MSE:  0.00011152567458339036
Epoch  121 MSE:  0.00011138420086354017
Epoch  122 MSE:  0.00011130443454021588
Epoch  123 MSE:  0.00011111525964224711
Epoch  124 MSE:  0.0001109767472371459
Epoch  125 MSE:  0.00011090919724665582
Epoch  126 MSE:  0.00011076500231865793
Epoch  127 MSE:  0.00011063445708714426
Epoch  128 MSE:  0.00011056838411604986
Epoch  129 MSE:  0.00011045355495298281
Epoch  130 MSE:  0.00011033199552912265
```

```
Epoch  131 MSE:  0.00011026357969967648
Epoch  132 MSE:  0.00011016876669600606
Epoch  133 MSE:  0.00011005916894646361
Epoch  134 MSE:  0.00010999094956787303
Epoch  135 MSE:  0.00010991191084031016
Epoch  136 MSE:  0.00010981653758790344
Epoch  137 MSE:  0.00010975139593938366
Epoch  138 MSE:  0.00010968481365125626
Epoch  139 MSE:  0.00010960299550788477
Epoch  140 MSE:  0.00010954190656775609
Epoch  141 MSE:  0.00010948388808174059
Epoch  142 MSE:  0.00010941275104414672
Epoch  143 MSE:  0.00010935511818388477
Epoch  144 MSE:  0.00010930250573437661
Epoch  145 MSE:  0.00010923964873654768
Epoch  146 MSE:  0.00010918534826487303
Epoch  147 MSE:  0.0001091367521439679
Epoch  148 MSE:  0.00010908090189332142
Epoch  149 MSE:  0.00010903033398790285
Training time: 27.7250018119812
```

[24]:
```
epoch=[i for i in range(num_epochs)]
plt.plot(epoch,loss_val_close,c='r')
plt.show()
```



[25]:
```
y_train_pred = model(x2_train)
plt.plot(stock['Date'][:2391],(close_train_lstm), c='r')
plt.plot(stock['Date'][:2391], (y_train_pred).detach().numpy(), c='b')
plt.xlabel('Day')
plt.ylabel('Close')
plt.show()
```



[26]:
```
trainScore = math.sqrt(mean_squared_error(close_train[:,0], (y_train_pred).
 ↪detach().numpy()[:,0]))
print(f'Train Score: {trainScore*100} % RMSE ')
```

```
Train Score: 1.0439612639814269 % RMSE
```

Thus, we get that Root Mean Square Error of our model on training dataset is 1.04%

5.2 Testing

[27]:
```
import time
hist = np.zeros(num_epochs)
start_time = time.time()
lstm = []
loss_val_close_test=[]
for t in range(num_epochs):
    y_test_pred = model(x2_test)
    loss = criterion(y_test_pred, close_test_lstm)
    print("Epoch ", t, "MSE: ", loss.item())
    hist[t] = loss.item()
    optimiser.zero_grad()
    loss.backward()
    optimiser.step()
    loss_val_close_test.append(loss.item())
training_time = time.time()-start_time
print(f"Testing time: {testing_time}")
```

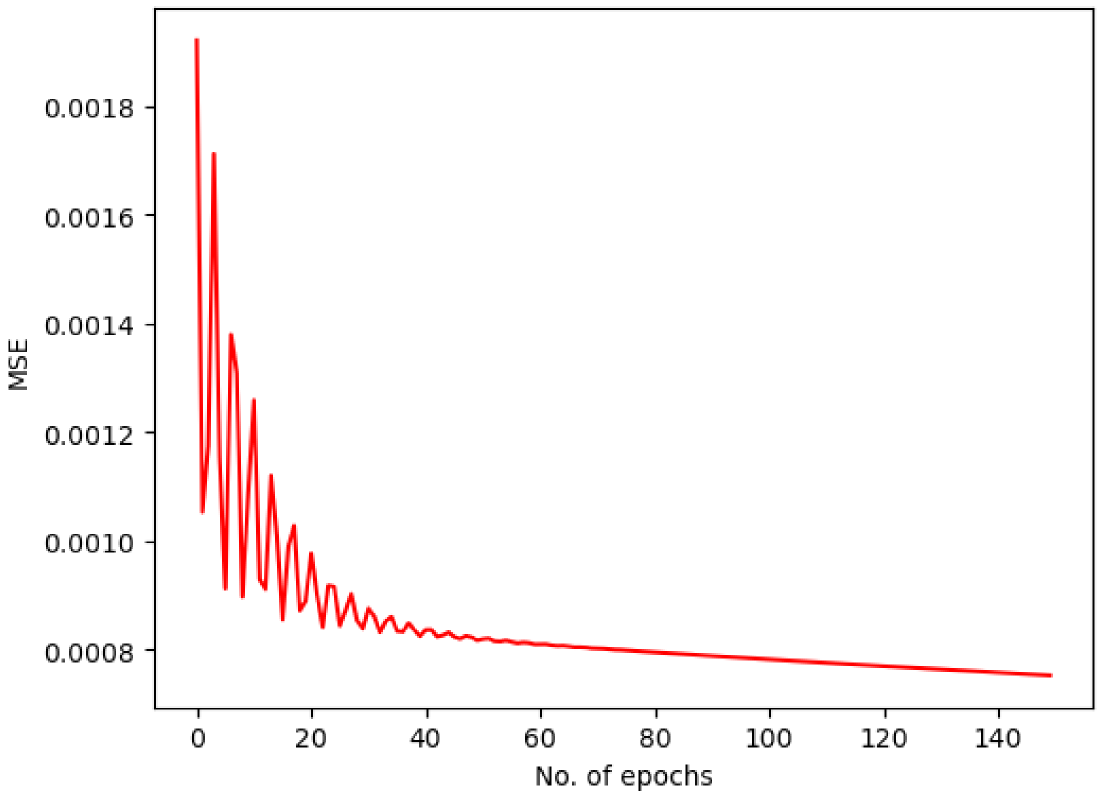
```
Epoch  0 MSE:  0.0019208743469789624
Epoch  1 MSE:  0.0010527464328333735
Epoch  2 MSE:  0.0011748813558369875
Epoch  3 MSE:  0.001711831078864634
Epoch  4 MSE:  0.0011533446377143264
Epoch  5 MSE:  0.0009118386078625917
Epoch  6 MSE:  0.0013792356476187706
Epoch  7 MSE:  0.00130767363589257
Epoch  8 MSE:  0.0008966927998699248
Epoch  9 MSE:  0.0010860870825126767
Epoch  10 MSE:  0.0012584548676386476
Epoch  11 MSE:  0.0009287974680773914
Epoch  12 MSE:  0.0009105970966629684
Epoch  13 MSE:  0.0011201307643204927
Epoch  14 MSE:  0.0010089483112096786
Epoch  15 MSE:  0.0008547245524823666
Epoch  16 MSE:  0.00098942394834018
Epoch  17 MSE:  0.001027778722345829
Epoch  18 MSE:  0.0008712115813978016
Epoch  19 MSE:  0.0008884635171853006
Epoch  20 MSE:  0.0009764222777448595
Epoch  21 MSE:  0.0009004850289784372
Epoch  22 MSE:  0.0008408999419771135
Epoch  23 MSE:  0.0009175813174806535
Epoch  24 MSE:  0.0009154776926152408
Epoch  25 MSE:  0.000843545189127326
Epoch  26 MSE:  0.0008711410337127745
Epoch  27 MSE:  0.0009023778256960213
Epoch  28 MSE:  0.0008528961916454136
```

```
Epoch  29 MSE:  0.0008381372899748385
Epoch  30 MSE:  0.0008754604496061802
Epoch  31 MSE:  0.0008605413604527712
Epoch  32 MSE:  0.0008315399172715843
Epoch  33 MSE:  0.0008518572431057692
Epoch  34 MSE:  0.0008600602159276605
Epoch  35 MSE:  0.0008338571060448885
Epoch  36 MSE:  0.000832590099889785
Epoch  37 MSE:  0.0008482451667077839
Epoch  38 MSE:  0.000835641345474869
Epoch  39 MSE:  0.0008243927150033414
Epoch  40 MSE:  0.0008361288928426802
Epoch  41 MSE:  0.0008361694635823369
Epoch  42 MSE:  0.0008233677945099771
Epoch  43 MSE:  0.0008261394104920328
Epoch  44 MSE:  0.0008315862505696714
Epoch  45 MSE:  0.0008228856022469699
Epoch  46 MSE:  0.0008194153197109699
Epoch  47 MSE:  0.0008250162936747074
Epoch  48 MSE:  0.0008223609183914959
Epoch  49 MSE:  0.0008166623883880675
Epoch  50 MSE:  0.0008192929672077298
Epoch  51 MSE:  0.0008202515891753137
Epoch  52 MSE:  0.0008151721558533609
Epoch  53 MSE:  0.0008144747698679566
Epoch  54 MSE:  0.0008164670434780419
Epoch  55 MSE:  0.0008136820397339761
Epoch  56 MSE:  0.0008112696232274175
Epoch  57 MSE:  0.0008127213804982603
Epoch  58 MSE:  0.0008119516423903406
Epoch  59 MSE:  0.0008092263014987111
Epoch  60 MSE:  0.0008093189098872244
Epoch  61 MSE:  0.000809483986813575
Epoch  62 MSE:  0.0008073312928900123
Epoch  63 MSE:  0.000806373602245003
Epoch  64 MSE:  0.0008067410090006888
Epoch  65 MSE:  0.0008054990321397781
Epoch  66 MSE:  0.0008041036198846996
Epoch  67 MSE:  0.0008041203836910427
Epoch  68 MSE:  0.000803501985501498
Epoch  69 MSE:  0.0008020683890208602
Epoch  70 MSE:  0.0008015873609110713
Epoch  71 MSE:  0.0008012639591470361
Epoch  72 MSE:  0.0008001051610335708
Epoch  73 MSE:  0.0007993166800588369
Epoch  74 MSE:  0.0007990291342139244
Epoch  75 MSE:  0.0007981795934028924
Epoch  76 MSE:  0.0007972405874170363
```

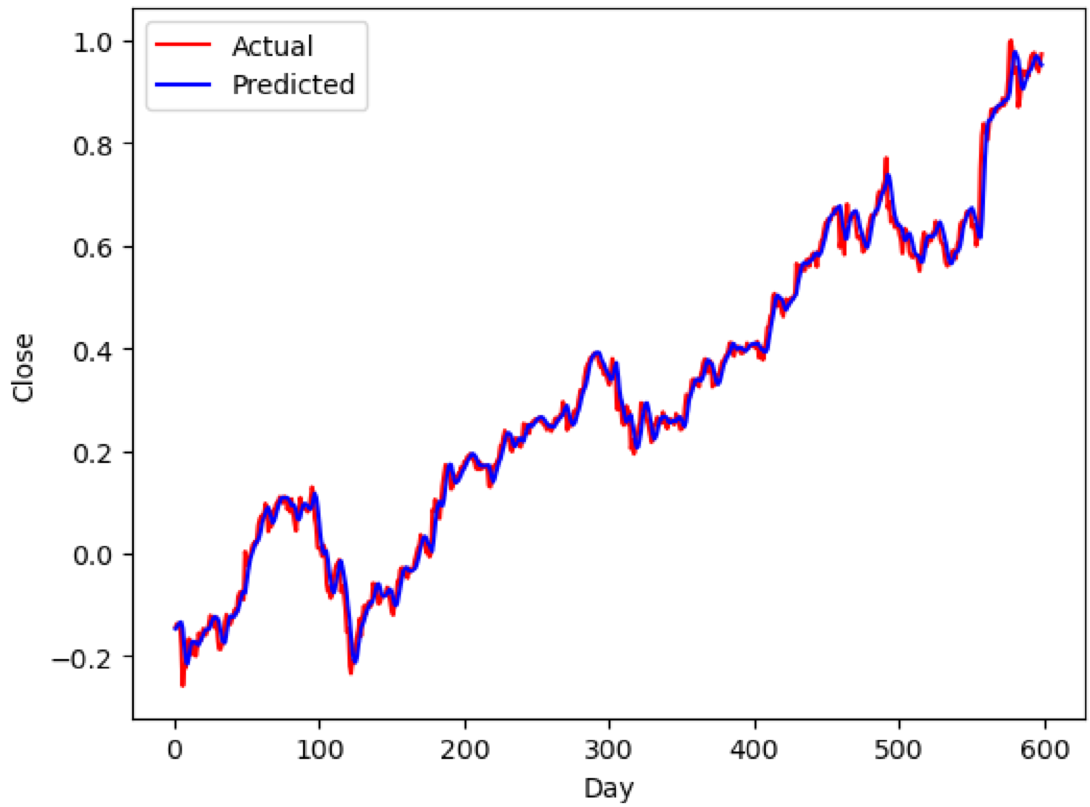
```
Epoch  77 MSE:  0.0007968071149662137
Epoch  78 MSE:  0.0007961598457768559
Epoch  79 MSE:  0.0007952286978252232
Epoch  80 MSE:  0.0007946458063088357
Epoch  81 MSE:  0.0007941103540360928
Epoch  82 MSE:  0.0007932804874144495
Epoch  83 MSE:  0.0007925918907858431
Epoch  84 MSE:  0.00079206726513803
Epoch  85 MSE:  0.0007913355948403478
Epoch  86 MSE:  0.0007905934471637011
Epoch  87 MSE:  0.0007900293567217886
Epoch  88 MSE:  0.0007893758011050522
Epoch  89 MSE:  0.0007886384264566004
Epoch  90 MSE:  0.0007880321354605258
Epoch  91 MSE:  0.0007874235743656754
Epoch  92 MSE:  0.0007867159438319504
Epoch  93 MSE:  0.0007860730402171612
Epoch  94 MSE:  0.0007854749565012753
Epoch  95 MSE:  0.00078479980584234
Epoch  96 MSE:  0.0007841388578526676
Epoch  97 MSE:  0.0007835371652618051
Epoch  98 MSE:  0.0007828935049474239
Epoch  99 MSE:  0.0007822331390343606
Epoch  100 MSE:  0.0007816215511411428
Epoch  101 MSE:  0.0007809969247318804
Epoch  102 MSE:  0.0007803434855304658
Epoch  103 MSE:  0.0007797214784659445
Epoch  104 MSE:  0.0007791065727360547
Epoch  105 MSE:  0.0007784656481817365
Epoch  106 MSE:  0.0007778393919579685
Epoch  107 MSE:  0.0007772292010486126
Epoch  108 MSE:  0.000776600674726069
Epoch  109 MSE:  0.0007759739528410137
Epoch  110 MSE:  0.0007753637037239969
Epoch  111 MSE:  0.0007747444906271994
Epoch  112 MSE:  0.0007741207955405116
Epoch  113 MSE:  0.0007735104882158339
Epoch  114 MSE:  0.0007728984928689897
Epoch  115 MSE:  0.0007722798618488014
Epoch  116 MSE:  0.0007716703112237155
Epoch  117 MSE:  0.0007710629142820835
Epoch  118 MSE:  0.000770449754782021
Epoch  119 MSE:  0.0007698411936871707
Epoch  120 MSE:  0.0007692372892051935
Epoch  121 MSE:  0.0007686293101869524
Epoch  122 MSE:  0.0007680230773985386
Epoch  123 MSE:  0.0007674218504689634
Epoch  124 MSE:  0.0007668186444789171
```

```
Epoch  125 MSE:  0.0007662154966965318
Epoch  126 MSE:  0.0007656165398657322
Epoch  127 MSE:  0.0007650171755813062
Epoch  128 MSE:  0.0007644173456355929
Epoch  129 MSE:  0.0007638204842805862
Epoch  130 MSE:  0.0007632247870787978
Epoch  131 MSE:  0.0007626281585544348
Epoch  132 MSE:  0.0007620343822054565
Epoch  133 MSE:  0.0007614415371790528
Epoch  134 MSE:  0.0007608483429066837
Epoch  135 MSE:  0.0007602566620334983
Epoch  136 MSE:  0.000759666902013123
Epoch  137 MSE:  0.0007590768509544432
Epoch  138 MSE:  0.0007584878476336598
Epoch  139 MSE:  0.0007579006487503648
Epoch  140 MSE:  0.0007573136826977134
Epoch  141 MSE:  0.000756727356929332
Epoch  142 MSE:  0.0007561427773907781
Epoch  143 MSE:  0.0007555586635135114
Epoch  144 MSE:  0.0007549751317128539
Epoch  145 MSE:  0.0007543928804807365
Epoch  146 MSE:  0.000753811385948211
Epoch  147 MSE:  0.0007532306481152773
Epoch  148 MSE:  0.0007526509580202401
Epoch  149 MSE:  0.0007520719664171338
Testing time: 5.911567687988281
```

[28]:
```
epoch=[i for i in range(num_epochs)]
plt.plot(epoch,loss_val_close_test,c='r')
plt.xlabel("No. of epochs")
plt.ylabel("MSE")
plt.show()
```



[29]:
```
y_test_pred = model(x2_test)
plt.plot(stock['Date'][:598],(close_test_lstm), c='r')
plt.plot(stock['Date'][:598], (y_test_pred).detach().numpy(), c='b')
plt.xlabel('Day')
plt.ylabel('Close')
plt.legend(["Actual","Predicted"],loc="upper left")
plt.show()
```



[30]:
```
testScore = math.sqrt(mean_squared_error(close_test[:,0], (y_test_pred).detach().
↪numpy()[:,0]))
print(f'Test Score: {testScore*100} % RMSE ')
```

```
Test Score: 2.7413387093343946 % RMSE
```

The Root Mean Square Error of our model in testing dataset is 2.74%

6 Result

Our LSTM model achieved promising results in predicting Amazon stock prices. The RMSE values for Open price prediction is 1.53% in training dataset and 3.02% in testing dataset. The RMSE values for Close price prediction is 1.04% in training dataset and 2.74% in testing dataset.

7 Conclusion

This research paper explored the application of LSTM neural networks for predicting Amazon stock prices over a ten-year period. The results suggest that LSTM networks can be a valuable tool in stock market prediction. However, it is essential to consider the limitations of the model and the unpredictable nature of financial markets.