STOCK MARKET PREDICTION AND FORECOSTING USING STACKED LMST

Introduction

Predicting stock prices is an uncertain task using machine learning. There are a lot of tools used for stock market prediction. The stock market is considered to be dynamic and complex. An accurate forecast of future prices may lead to a higher profit yield for investors through stock investments. Investors will pick stocks that may give a higher return as per the predictions.

LSTM

Long short-term memory is an artificial recurrent neural network (RNN) architecture used in deep learning. Unlike standard feedforward neural networks, Long short-term Memory has feedback connections. It can process single data points (e.g., images) and entire data sequences (such as speech or video inputs).

Implementation

Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [28]:
```
df_train=pd.read_csv('https://raw.githubusercontent.com/mwitiderrick/stockprice/master/N
```

Basic Chacks

In [3]:
```
df.head()
```

Out[3]:

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2018-09-28	234.05	235.95	230.20	233.50	233.75	3069914	7162.35
1	2018-09-27	234.55	236.80	231.10	233.80	233.25	5082859	11859.95
2	2018-09-26	240.00	240.00	232.50	235.00	234.25	2240909	5248.60
3	2018-09-25	233.30	236.75	232.00	236.25	236.10	2349368	5503.90
4	2018-09-24	233.55	239.20	230.75	234.00	233.30	3423509	7999.55

In [4]:
```
df.tail()
```

Out[4]:

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
2030	2010-07-27	117.6	119.50	112.00	118.80	118.65	586100	694.98
2031	2010-07-26	120.1	121.00	117.10	117.10	117.60	658440	780.01
2032	2010-07-23	121.8	121.95	120.25	120.35	120.65	281312	340.31
2033	2010-07-22	120.3	122.00	120.25	120.75	120.90	293312	355.17
2034	2010-07-21	122.1	123.00	121.05	121.10	121.55	658666	803.56

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2035 entries, 0 to 2034
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Date                  2035 non-null   object 
 1   Open                  2035 non-null   float64
 2   High                  2035 non-null   float64
 3   Low                   2035 non-null   float64
 4   Last                  2035 non-null   float64
 5   Close                 2035 non-null   float64
 6   Total Trade Quantity  2035 non-null   int64  
 7   Turnover (Lacs)       2035 non-null   float64
dtypes: float64(6), int64(1), object(1)
memory usage: 127.3+ KB
```

In [6]:

```
df.describe()
```

Out[6]:

	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
count	2035.000000	2035.000000	2035.000000	2035.000000	2035.00000	2.035000e+03	2035.000000
mean	149.713735	151.992826	147.293931	149.474251	149.45027	2.335681e+06	3899.980565
std	48.664509	49.413109	47.931958	48.732570	48.71204	2.091778e+06	4570.767877
min	81.100000	82.800000	80.000000	81.000000	80.95000	3.961000e+04	37.040000
25%	120.025000	122.100000	118.300000	120.075000	120.05000	1.146444e+06	1427.460000
50%	141.500000	143.400000	139.600000	141.100000	141.25000	1.783456e+06	2512.030000
75%	157.175000	159.400000	155.150000	156.925000	156.90000	2.813594e+06	4539.015000
max	327.700000	328.750000	321.650000	325.950000	325.75000	2.919102e+07	55755.080000

In [7]:

```
df.dtypes
```

Out[7]:

```
Date                     object
Open                    float64
High                    float64
Low                     float64
Last                    float64
Close                   float64
Total Trade Quantity      int64
Turnover (Lacs)         float64
dtype: object
```

Data Processing

In [8]:

```
df.isnull().sum()
```

Out[8]:
```
Date                      0
Open                      0
High                      0
Low                       0
Last                      0
Close                     0
Total Trade Quantity      0
Turnover (Lacs)           0
dtype: int64
```

In [9]:
```
close_prices = df['Close'].values.reshape(-1, 1)
```

Use the Open Stock Price Column to Train Your Model.

In [10]:
```
training_set=df.iloc[:,1:2].values
print(training_set)
print(training_set.shape)
```

```
[[234.05]
 [234.55]
 [240.  ]
 ...
 [121.8 ]
 [120.3 ]
 [122.1 ]]
(2035, 1)
```

Normalizing the dataset

In [11]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
scaled_training_set=scaler.fit_transform(training_set)
scaled_training_set
```

Out[11]:
```
array([[0.6202352 ],
       [0.62226277],
       [0.64436334],
       ...,
       [0.16504461],
       [0.15896188],
       [0.16626115]])
```

Creating X-train and y-train Data structures

In [15]:
```
X_train=[]
Y_train=[]
for i in range(60,1258):
    X_train.append(scaled_training_set[i-60:i,0])
    Y_train.append(scaled_training_set[i,0])
X_train=np.array(X_train)
Y_train=np.array(Y_train)
print(X_train.shape)
print(Y_train.shape)
```

```
(1198, 60)
(1198,)
```

Reshape the data

In [16]:
```
X_train=np.reshape(X_train,(X_train.shape[0],X_train.shape[1],1))
X_train.shape
```

Out[16]:
```
(1198, 60, 1)
```

Building the Model by Importing the Crucial Libraries and Adding Different Layers to LSTM.

In [17]:
```
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
from keras.layers import Dropout
```

In [21]:
```
regressor=Sequential()
regressor.add(LSTM(units=50,return_sequences=True,input_shape=(X_train.shape[1],1)))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units=50,return_sequences=True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units=50,return_sequences=True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units=50,return_sequences=True))
regressor.add(Dropout(0.2))
regressor.add(Dense(units=1))
```

Fitting the model

In [22]:
```
regressor.compile(optimizer='adam',loss='mean_squared_error')
regressor.fit(X_train,Y_train,epochs=100,batch_size=32)
```

```
Epoch 1/100
38/38 [==============================] - 19s 125ms/step - loss: 0.0591
Epoch 2/100
38/38 [==============================] - 5s 122ms/step - loss: 0.0446
Epoch 3/100
38/38 [==============================] - 5s 124ms/step - loss: 0.0442
Epoch 4/100
38/38 [==============================] - 5s 122ms/step - loss: 0.0430
Epoch 5/100
38/38 [==============================] - 5s 122ms/step - loss: 0.0421
Epoch 6/100
38/38 [==============================] - 5s 125ms/step - loss: 0.0422
Epoch 7/100
38/38 [==============================] - 5s 135ms/step - loss: 0.0421
Epoch 8/100
38/38 [==============================] - 5s 132ms/step - loss: 0.0417
Epoch 9/100
38/38 [==============================] - 5s 130ms/step - loss: 0.0413
Epoch 10/100
38/38 [==============================] - 5s 126ms/step - loss: 0.0412
Epoch 11/100
38/38 [==============================] - 5s 124ms/step - loss: 0.0412
Epoch 12/100
38/38 [==============================] - 5s 136ms/step - loss: 0.0411
Epoch 13/100
38/38 [==============================] - 5s 124ms/step - loss: 0.0412
Epoch 14/100
38/38 [==============================] - 5s 124ms/step - loss: 0.0409
Epoch 15/100
38/38 [==============================] - 5s 134ms/step - loss: 0.0410
Epoch 16/100
38/38 [==============================] - 5s 127ms/step - loss: 0.0411
Epoch 17/100
38/38 [==============================] - 5s 136ms/step - loss: 0.0407
Epoch 18/100
38/38 [==============================] - 5s 128ms/step - loss: 0.0413
Epoch 19/100
38/38 [==============================] - 5s 126ms/step - loss: 0.0407
Epoch 20/100
38/38 [==============================] - 5s 140ms/step - loss: 0.0408
Epoch 21/100
38/38 [==============================] - 5s 136ms/step - loss: 0.0406
Epoch 22/100
38/38 [==============================] - 5s 130ms/step - loss: 0.0406
Epoch 23/100
38/38 [==============================] - 5s 136ms/step - loss: 0.0406
Epoch 24/100
38/38 [==============================] - 5s 132ms/step - loss: 0.0407
Epoch 25/100
38/38 [==============================] - 5s 127ms/step - loss: 0.0406
Epoch 26/100
38/38 [==============================] - 6s 161ms/step - loss: 0.0405
Epoch 27/100
38/38 [==============================] - 7s 177ms/step - loss: 0.0411
Epoch 28/100
38/38 [==============================] - 10795s 292s/step - loss: 0.0405
Epoch 29/100
38/38 [==============================] - 7s 184ms/step - loss: 0.0408
Epoch 30/100
38/38 [==============================] - 6s 152ms/step - loss: 0.0404
Epoch 31/100
38/38 [==============================] - 6s 146ms/step - loss: 0.0405
Epoch 32/100
38/38 [==============================] - 28769s 778s/step - loss: 0.0404
```

```
Epoch 33/100
38/38 [==============================] - 6s 150ms/step - loss: 0.0404
Epoch 34/100
38/38 [==============================] - 5s 142ms/step - loss: 0.0407
Epoch 35/100
38/38 [==============================] - 5s 143ms/step - loss: 0.0407
Epoch 36/100
38/38 [==============================] - 5s 134ms/step - loss: 0.0405
Epoch 37/100
38/38 [==============================] - 5s 140ms/step - loss: 0.0404
Epoch 38/100
38/38 [==============================] - 5s 138ms/step - loss: 0.0406
Epoch 39/100
38/38 [==============================] - 6s 152ms/step - loss: 0.0404
Epoch 40/100
38/38 [==============================] - 5s 139ms/step - loss: 0.0405
Epoch 41/100
38/38 [==============================] - 5s 139ms/step - loss: 0.0404
Epoch 42/100
38/38 [==============================] - 5s 142ms/step - loss: 0.0403
Epoch 43/100
38/38 [==============================] - 6s 147ms/step - loss: 0.0405
Epoch 44/100
38/38 [==============================] - 5s 127ms/step - loss: 0.0404
Epoch 45/100
38/38 [==============================] - 5s 126ms/step - loss: 0.0406
Epoch 46/100
38/38 [==============================] - 5s 135ms/step - loss: 0.0403
Epoch 47/100
38/38 [==============================] - 5s 133ms/step - loss: 0.0403
Epoch 48/100
38/38 [==============================] - 5s 135ms/step - loss: 0.0406
Epoch 49/100
38/38 [==============================] - 5s 132ms/step - loss: 0.0404
Epoch 50/100
38/38 [==============================] - 5s 138ms/step - loss: 0.0403
Epoch 51/100
38/38 [==============================] - 5s 130ms/step - loss: 0.0404
Epoch 52/100
38/38 [==============================] - 5s 135ms/step - loss: 0.0403
Epoch 53/100
38/38 [==============================] - 5s 126ms/step - loss: 0.0403
Epoch 54/100
38/38 [==============================] - 5s 143ms/step - loss: 0.0404
Epoch 55/100
38/38 [==============================] - 5s 139ms/step - loss: 0.0403
Epoch 56/100
38/38 [==============================] - 6s 150ms/step - loss: 0.0405
Epoch 57/100
38/38 [==============================] - 6s 154ms/step - loss: 0.0403
Epoch 58/100
38/38 [==============================] - 6s 147ms/step - loss: 0.0403
Epoch 59/100
38/38 [==============================] - 5s 136ms/step - loss: 0.0404
Epoch 60/100
38/38 [==============================] - 6s 146ms/step - loss: 0.0403
Epoch 61/100
38/38 [==============================] - 6s 152ms/step - loss: 0.0403
Epoch 62/100
38/38 [==============================] - 6s 148ms/step - loss: 0.0403
Epoch 63/100
38/38 [==============================] - 5s 126ms/step - loss: 0.0403
Epoch 64/100
38/38 [==============================] - 5s 126ms/step - loss: 0.0403
```

```
Epoch 65/100
38/38 [==============================] - 5s 126ms/step - loss: 0.0402
Epoch 66/100
38/38 [==============================] - 5s 132ms/step - loss: 0.0403
Epoch 67/100
38/38 [==============================] - 6s 148ms/step - loss: 0.0403
Epoch 68/100
38/38 [==============================] - 6s 153ms/step - loss: 0.0402
Epoch 69/100
38/38 [==============================] - 6s 170ms/step - loss: 0.0404
Epoch 70/100
38/38 [==============================] - 6s 157ms/step - loss: 0.0403
Epoch 71/100
38/38 [==============================] - 6s 145ms/step - loss: 0.0405
Epoch 72/100
38/38 [==============================] - 5s 142ms/step - loss: 0.0403
Epoch 73/100
38/38 [==============================] - 5s 134ms/step - loss: 0.0403
Epoch 74/100
38/38 [==============================] - 5s 138ms/step - loss: 0.0403
Epoch 75/100
38/38 [==============================] - 5s 135ms/step - loss: 0.0403
Epoch 76/100
38/38 [==============================] - 5s 141ms/step - loss: 0.0403
Epoch 77/100
38/38 [==============================] - 5s 136ms/step - loss: 0.0403
Epoch 78/100
38/38 [==============================] - 5s 132ms/step - loss: 0.0403
Epoch 79/100
38/38 [==============================] - 5s 138ms/step - loss: 0.0403
Epoch 80/100
38/38 [==============================] - 6s 145ms/step - loss: 0.0403
Epoch 81/100
38/38 [==============================] - 5s 141ms/step - loss: 0.0403
Epoch 82/100
38/38 [==============================] - 5s 141ms/step - loss: 0.0403
Epoch 83/100
38/38 [==============================] - 6s 146ms/step - loss: 0.0403
Epoch 84/100
38/38 [==============================] - 5s 144ms/step - loss: 0.0403
Epoch 85/100
38/38 [==============================] - 5s 142ms/step - loss: 0.0402
Epoch 86/100
38/38 [==============================] - 5s 134ms/step - loss: 0.0403
Epoch 87/100
38/38 [==============================] - 5s 134ms/step - loss: 0.0402
Epoch 88/100
38/38 [==============================] - 5s 132ms/step - loss: 0.0402
Epoch 89/100
38/38 [==============================] - 5s 140ms/step - loss: 0.0403
Epoch 90/100
38/38 [==============================] - 5s 142ms/step - loss: 0.0403
Epoch 91/100
38/38 [==============================] - 5s 138ms/step - loss: 0.0402
Epoch 92/100
38/38 [==============================] - 5s 141ms/step - loss: 0.0403
Epoch 93/100
38/38 [==============================] - 6s 145ms/step - loss: 0.0403
Epoch 94/100
38/38 [==============================] - 5s 141ms/step - loss: 0.0403
Epoch 95/100
38/38 [==============================] - 5s 138ms/step - loss: 0.0402
Epoch 96/100
38/38 [==============================] - 6s 148ms/step - loss: 0.0403
```

```
Epoch 97/100
38/38 [==============================] - 6s 164ms/step - loss: 0.0403
Epoch 98/100
38/38 [==============================] - 6s 155ms/step - loss: 0.0404
Epoch 99/100
38/38 [==============================] - 6s 171ms/step - loss: 0.0402
Epoch 100/100
38/38 [==============================] - 6s 167ms/step - loss: 0.0403
```

Out[22]:

```
<keras.src.callbacks.History at 0x1e6e5bbf820>
```

Extracting the Actual Stock Prices

In [29]:

```
df_test=pd.read_csv('https://raw.githubusercontent.com/mwitiderrick/stockprice/master/NS
actual_stock_price=df.iloc[:,1:2].values
```

Preparing the Input for the Model.

In [50]:

```
dataset_total = pd.concat((df['Open'], df_test['Open']), axis = 0)
inputs = dataset_total[len(dataset_total) - len(df_test) - 60:].values
inputs = inputs.reshape(-1,1)
inputs = scaler.transform(inputs)
X_test = []
for i in range(60, 80):
    X_test.append(inputs[i-60:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
```