**import** warnings

warnings**.**filterwarnings('ignore')

**import** math

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** keras.models **import** Sequential

**from** keras.layers **import** Dense, LSTM, Dropout, Dense, Activation

**import** nltk

**from** nltk.classify **import** NaiveBayesClassifier

**from** nltk.corpus **import** subjectivity

**from** nltk.sentiment **import** SentimentAnalyzer

**from** nltk.sentiment.util **import** **\***

**from** sklearn **import** preprocessing, metrics

**from** sklearn.preprocessing **import** MinMaxScaler

*# reading the datasets into pandas*

stock\_price **=** pd**.**read\_csv('AAPL.csv')

stock\_headlines **=** pd**.**read\_csv('india-news-headlines.csv')

*# displaying stock price dataset*

stock\_price**.**head()

|  | **Date** | **Open** | **High** | **Low** | **Close** | **Adj Close** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2001-01-02 | 0.265625 | 0.272321 | 0.260045 | 0.265625 | 0.229537 | 452312000 |
| **1** | 2001-01-03 | 0.258929 | 0.297991 | 0.257813 | 0.292411 | 0.252684 | 817073600 |
| **2** | 2001-01-04 | 0.323940 | 0.330357 | 0.300223 | 0.304688 | 0.263292 | 739396000 |
| **3** | 2001-01-05 | 0.302455 | 0.310268 | 0.286830 | 0.292411 | 0.252684 | 412356000 |
| **4** | 2001-01-08 | 0.302455 | 0.303292 | 0.284598 | 0.295759 | 0.255577 | 373699200 |

*# displaying stock news headlines dataset*

stock\_headlines**.**head()

|  | **publish\_date** | **headline\_category** | **headline\_text** |
| --- | --- | --- | --- |
| **0** | 20010101 | sports.wwe | win over cena satisfying but defeating underta... |
| **1** | 20010102 | unknown | Status quo will not be disturbed at Ayodhya; s... |
| **2** | 20010102 | unknown | Fissures in Hurriyat over Pak visit |
| **3** | 20010102 | unknown | America's unwanted heading for India? |
| **4** | 20010102 | unknown | For bigwigs; it is destination Goa |

*# displaying number of records in both stock\_price and stock\_headlines datasets*

len(stock\_price), len(stock\_headlines)

(4904, 3297172)

*# checking for null values in both the datasets*

stock\_price**.**isna()**.**any(), stock\_headlines**.**isna()**.**any()

(Date False

Open False

High False

Low False

Close False

Adj Close False

Volume False

dtype: bool,

publish\_date False

headline\_category False

headline\_text False

dtype: bool)

*# dropping duplicates*

stock\_price **=** stock\_price**.**drop\_duplicates()

*# coverting the datatype of column 'Date' from type object to type 'datetime'*

stock\_price['Date'] **=** pd**.**to\_datetime(stock\_price['Date'])**.**dt**.**normalize()

*# filtering the important columns required*

stock\_price **=** stock\_price**.**filter(['Date', 'Close', 'Open', 'High', 'Low', 'Volume'])

*# setting column 'Date' as the index column*

stock\_price**.**set\_index('Date', inplace**=** **True**)

*# sorting the data according to the index i.e 'Date'*

stock\_price **=** stock\_price**.**sort\_index(ascending**=True**, axis**=**0)

stock\_price

|  | **Close** | **Open** | **High** | **Low** | **Volume** |
| --- | --- | --- | --- | --- | --- |
| **Date** |  |  |  |  |  |
| **2001-01-02** | 0.265625 | 0.265625 | 0.272321 | 0.260045 | 452312000 |
| **2001-01-03** | 0.292411 | 0.258929 | 0.297991 | 0.257813 | 817073600 |
| **2001-01-04** | 0.304688 | 0.323940 | 0.330357 | 0.300223 | 739396000 |
| **2001-01-05** | 0.292411 | 0.302455 | 0.310268 | 0.286830 | 412356000 |
| **2001-01-08** | 0.295759 | 0.302455 | 0.303292 | 0.284598 | 373699200 |
| **...** | ... | ... | ... | ... | ... |
| **2020-06-24** | 90.014999 | 91.250000 | 92.197502 | 89.629997 | 192623200 |
| **2020-06-25** | 91.209999 | 90.175003 | 91.250000 | 89.392502 | 137522400 |
| **2020-06-26** | 88.407501 | 91.102501 | 91.330002 | 88.254997 | 205256800 |
| **2020-06-29** | 90.445000 | 88.312500 | 90.542503 | 87.820000 | 130646000 |
| **2020-06-30** | 91.199997 | 90.019997 | 91.495003 | 90.000000 | 140223200 |

*# dropping duplicates*

stock\_headlines **=** stock\_headlines**.**drop\_duplicates()

*# coverting the datatype of column 'Date' from type string to type 'datetime'*

stock\_headlines['publish\_date'] **=** stock\_headlines['publish\_date']**.**astype(str)

stock\_headlines['publish\_date'] **=** stock\_headlines['publish\_date']**.**apply(**lambda** x: x[0:4]**+**'-'**+**x[4:6]**+**'-'**+**x[6:8])

stock\_headlines['publish\_date'] **=** pd**.**to\_datetime(stock\_headlines['publish\_date'])**.**dt**.**normalize()

*# filtering the important columns required*

stock\_headlines **=** stock\_headlines**.**filter(['publish\_date', 'headline\_text'])

*# grouping the news headlines according to 'Date'*

stock\_headlines **=** stock\_headlines**.**groupby(['publish\_date'])['headline\_text']**.**apply(**lambda** x: ','**.**join(x))**.**reset\_index()

*# setting column 'Date' as the index column*

stock\_headlines**.**set\_index('publish\_date', inplace**=** **True**)

*# sorting the data according to the index i.e 'Date'*

stock\_headlines **=** stock\_headlines**.**sort\_index(ascending**=True**, axis**=**0)

stock\_headlines

|  | **headline\_text** |
| --- | --- |
| **publish\_date** |  |
| **2001-01-01** | win over cena satisfying but defeating underta... |
| **2001-01-02** | Status quo will not be disturbed at Ayodhya; s... |
| **2001-01-03** | Powerless north India gropes in the dark,Think... |
| **2001-01-04** | The string that pulled Stephen Hawking to Indi... |
| **2001-01-05** | Light combat craft takes India into club class... |
| **...** | ... |
| **2020-06-26** | Containment zone residents slam high prices ch... |
| **2020-06-27** | like me i wont let you have a toxic relationsh... |
| **2020-06-28** | Atanu Ghosh plans to rewrite old scripts to ma... |
| **2020-06-29** | 6 hot and stylish bikini looks of Katrina Kaif... |
| **2020-06-30** | Detective Byomkesh Bakshy!,Edge of Tomorrow,Fi... |

7080 rows × 1 columns

*# concatenating the datasets stock\_price and stock\_headlines*

stock\_data **=** pd**.**concat([stock\_price, stock\_headlines], axis**=**1)

*# dropping the null values if any*

stock\_data**.**dropna(axis**=**0, inplace**=True**)

*# displaying the combined stock\_data*

stock\_data

|  | **Close** | **Open** | **High** | **Low** | **Volume** | **headline\_text** |
| --- | --- | --- | --- | --- | --- | --- |
| **2001-01-02** | 0.265625 | 0.265625 | 0.272321 | 0.260045 | 452312000.0 | Status quo will not be disturbed at Ayodhya; s... |
| **2001-01-03** | 0.292411 | 0.258929 | 0.297991 | 0.257813 | 817073600.0 | Powerless north India gropes in the dark,Think... |
| **2001-01-04** | 0.304688 | 0.323940 | 0.330357 | 0.300223 | 739396000.0 | The string that pulled Stephen Hawking to Indi... |
| **2001-01-05** | 0.292411 | 0.302455 | 0.310268 | 0.286830 | 412356000.0 | Light combat craft takes India into club class... |
| **2001-01-08** | 0.295759 | 0.302455 | 0.303292 | 0.284598 | 373699200.0 | Sangh Parivar; Babri panel up the ante,Frontru... |
| **...** | ... | ... | ... | ... | ... | ... |
| **2020-06-24** | 90.014999 | 91.250000 | 92.197502 | 89.629997 | 192623200.0 | I never thought I had a voice until today: Vid... |
| **2020-06-25** | 91.209999 | 90.175003 | 91.250000 | 89.392502 | 137522400.0 | Truck firms look for new export markets to sel... |
| **2020-06-26** | 88.407501 | 91.102501 | 91.330002 | 88.254997 | 205256800.0 | Containment zone residents slam high prices ch... |
| **2020-06-29** | 90.445000 | 88.312500 | 90.542503 | 87.820000 | 130646000.0 | 6 hot and stylish bikini looks of Katrina Kaif... |
| **2020-06-30** | 91.199997 | 90.019997 | 91.495003 | 90.000000 | 140223200.0 | Detective Byomkesh Bakshy!,Edge of Tomorrow,Fi... |

4877 rows × 6 columns

*# adding empty sentiment columns to stock\_data for later calculation*

stock\_data['compound'] **=** ''

stock\_data['negative'] **=** ''

stock\_data['neutral'] **=** ''

stock\_data['positive'] **=** ''

stock\_data**.**head()

|  | **Close** | **Open** | **High** | **Low** | **Volume** | **headline\_text** | **compound** | **negative** | **neutral** | **positive** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **2001-01-02** | 0.265625 | 0.265625 | 0.272321 | 0.260045 | 452312000.0 | Status quo will not be disturbed at Ayodhya; s... |  |  |  |  |
| **2001-01-03** | 0.292411 | 0.258929 | 0.297991 | 0.257813 | 817073600.0 | Powerless north India gropes in the dark,Think... |  |  |  |  |
| **2001-01-04** | 0.304688 | 0.323940 | 0.330357 | 0.300223 | 739396000.0 | The string that pulled Stephen Hawking to Indi... |  |  |  |  |
| **2001-01-05** | 0.292411 | 0.302455 | 0.310268 | 0.286830 | 412356000.0 | Light combat craft takes India into club class... |  |  |  |  |
| **2001-01-08** | 0.295759 | 0.302455 | 0.303292 | 0.284598 | 373699200.0 | Sangh Parivar; Babri panel up the ante,Frontru... |  |  |  |  |

*# adding empty sentiment columns to stock\_data for later calculation*

stock\_data['compound'] **=** ''

stock\_data['negative'] **=** ''

stock\_data['neutral'] **=** ''

stock\_data['positive'] **=** ''

stock\_data**.**head()

|  | **Close** | **Open** | **High** | **Low** | **Volume** | **headline\_text** | **compound** | **negative** | **neutral** | **positive** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **2001-01-02** | 0.265625 | 0.265625 | 0.272321 | 0.260045 | 452312000.0 | Status quo will not be disturbed at Ayodhya; s... | -0.9621 | 0.119 | 0.817 | 0.064 |
| **2001-01-03** | 0.292411 | 0.258929 | 0.297991 | 0.257813 | 817073600.0 | Powerless north India gropes in the dark,Think... | 0.6322 | 0.084 | 0.817 | 0.098 |
| **2001-01-04** | 0.304688 | 0.323940 | 0.330357 | 0.300223 | 739396000.0 | The string that pulled Stephen Hawking to Indi... | 0.6648 | 0.077 | 0.843 | 0.080 |
| **2001-01-05** | 0.292411 | 0.302455 | 0.310268 | 0.286830 | 412356000.0 | Light combat craft takes India into club class... | 0.9032 | 0.105 | 0.746 | 0.149 |
| **2001-01-08** | 0.295759 | 0.302455 | 0.303292 | 0.284598 | 373699200.0 | Sangh Parivar; Babri panel up the ante,Frontru... | -0.9638 | 0.119 | 0.855 | 0.026 |

*# importing requires libraries to analyze the sentiments*

**from** nltk.sentiment.vader **import** SentimentIntensityAnalyzer

**import** unicodedata

*# instantiating the Sentiment Analyzer*

sid **=** SentimentIntensityAnalyzer()

*# calculating sentiment scores*

stock\_data['compound'] **=** stock\_data['headline\_text']**.**apply(**lambda** x: sid**.**polarity\_scores(x)['compound'])

stock\_data['negative'] **=** stock\_data['headline\_text']**.**apply(**lambda** x: sid**.**polarity\_scores(x)['neg'])

stock\_data['neutral'] **=** stock\_data['headline\_text']**.**apply(**lambda** x: sid**.**polarity\_scores(x)['neu'])

stock\_data['positive'] **=** stock\_data['headline\_text']**.**apply(**lambda** x: sid**.**polarity\_scores(x)['pos'])

*# displaying the stock data*

stock\_data**.**head()

|  | **Close** | **Open** | **High** | **Low** | **Volume** | **headline\_text** | **compound** | **negative** | **neutral** | **positive** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **2001-01-02** | 0.265625 | 0.265625 | 0.272321 | 0.260045 | 452312000.0 | Status quo will not be disturbed at Ayodhya; s... | -0.9621 | 0.119 | 0.817 | 0.064 |
| **2001-01-03** | 0.292411 | 0.258929 | 0.297991 | 0.257813 | 817073600.0 | Powerless north India gropes in the dark,Think... | 0.6322 | 0.084 | 0.817 | 0.098 |
| **2001-01-04** | 0.304688 | 0.323940 | 0.330357 | 0.300223 | 739396000.0 | The string that pulled Stephen Hawking to Indi... | 0.6648 | 0.077 | 0.843 | 0.080 |
| **2001-01-05** | 0.292411 | 0.302455 | 0.310268 | 0.286830 | 412356000.0 | Light combat craft takes India into club class... | 0.9032 | 0.105 | 0.746 | 0.149 |
| **2001-01-08** | 0.295759 | 0.302455 | 0.303292 | 0.284598 | 373699200.0 | Sangh Parivar; Babri panel up the ante,Frontru... | -0.9638 | 0.119 | 0.855 | 0.026 |

*# dropping the 'headline\_text' which is unwanted now*

stock\_data**.**drop(['headline\_text'], inplace**=True**, axis**=**1)

*# rearranging the columns of the whole stock\_data*

stock\_data **=** stock\_data[['Close', 'compound', 'negative', 'neutral', 'positive', 'Open', 'High', 'Low', 'Volume']]

*# displaying the final stock\_data*

stock\_data**.**head()

|  | **Close** | **compound** | **negative** | **neutral** | **positive** | **Open** | **High** | **Low** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **2001-01-02** | 0.265625 | -0.9621 | 0.119 | 0.817 | 0.064 | 0.265625 | 0.272321 | 0.260045 | 452312000.0 |
| **2001-01-03** | 0.292411 | 0.6322 | 0.084 | 0.817 | 0.098 | 0.258929 | 0.297991 | 0.257813 | 817073600.0 |
| **2001-01-04** | 0.304688 | 0.6648 | 0.077 | 0.843 | 0.080 | 0.323940 | 0.330357 | 0.300223 | 739396000.0 |
| **2001-01-05** | 0.292411 | 0.9032 | 0.105 | 0.746 | 0.149 | 0.302455 | 0.310268 | 0.286830 | 412356000.0 |
| **2001-01-08** | 0.295759 | -0.9638 | 0.119 | 0.855 | 0.026 | 0.302455 | 0.303292 | 0.284598 | 373699200.0 |

*# writing the prepared stock\_data to disk*

stock\_data**.**to\_csv('stock\_data.csv')

*# re-reading the stock\_data into pandas dataframe*

stock\_data **=** pd**.**read\_csv('stock\_data.csv', index\_col **=** **False**)

*# renaming the column*

stock\_data**.**rename(columns**=**{'Unnamed: 0':'Date'}, inplace **=** **True**)

*# setting the column 'Date' as the index column*

stock\_data**.**set\_index('Date', inplace**=True**)

*# displaying the stock\_data*

stock\_data**.**head()

|  | **Close** | **compound** | **negative** | **neutral** | **positive** | **Open** | **High** | **Low** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** |  |  |  |  |  |  |  |  |  |
| **2001-01-02** | 0.265625 | -0.9621 | 0.119 | 0.817 | 0.064 | 0.265625 | 0.272321 | 0.260045 | 452312000.0 |
| **2001-01-03** | 0.292411 | 0.6322 | 0.084 | 0.817 | 0.098 | 0.258929 | 0.297991 | 0.257813 | 817073600.0 |
| **2001-01-04** | 0.304688 | 0.6648 | 0.077 | 0.843 | 0.080 | 0.323940 | 0.330357 | 0.300223 | 739396000.0 |
| **2001-01-05** | 0.292411 | 0.9032 | 0.105 | 0.746 | 0.149 | 0.302455 | 0.310268 | 0.286830 | 412356000.0 |
| **2001-01-08** | 0.295759 | -0.9638 | 0.119 | 0.855 | 0.026 | 0.302455 | 0.303292 | 0.284598 | 373699200.0 |

*# displaying the shape i.e. number of rows and columns of stock\_data*

stock\_data**.**shape

(4877, 9)

*# checking for null values*

stock\_data**.**isna()**.**any()

Close False

compound False

negative False

neutral False

positive False

Open False

High False

Low False

Volume False

dtype: bool

*# displaying stock\_data statistics*

stock\_data**.**describe(include**=**'all')

|  | **Close** | **compound** | **negative** | **neutral** | **positive** | **Open** | **High** | **Low** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 4877.000000 | 4877.000000 | 4877.000000 | 4877.000000 | 4877.000000 | 4877.000000 | 4877.000000 | 4877.000000 | 4.877000e+03 |
| **mean** | 17.237386 | -0.870185 | 0.125720 | 0.788116 | 0.086175 | 17.228087 | 17.402949 | 17.056717 | 4.513386e+08 |
| **std** | 18.701168 | 0.453365 | 0.024431 | 0.033153 | 0.020881 | 18.670832 | 18.873534 | 18.499968 | 3.855476e+08 |
| **min** | 0.234286 | -1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.231964 | 0.235536 | 0.227143 | 3.247000e+07 |
| **25%** | 2.132857 | -0.999800 | 0.112000 | 0.768000 | 0.075000 | 2.136429 | 2.165357 | 2.096071 | 1.774640e+08 |
| **50%** | 10.901428 | -0.999000 | 0.127000 | 0.785000 | 0.086000 | 10.900000 | 11.000000 | 10.746428 | 3.374672e+08 |
| **75%** | 27.350000 | -0.994100 | 0.141000 | 0.806000 | 0.096000 | 27.334999 | 27.605000 | 27.100000 | 6.051080e+08 |
| **max** | 91.632500 | 1.000000 | 0.444000 | 1.000000 | 0.608000 | 91.250000 | 93.095001 | 90.567497 | 3.372970e+09 |

*# displaying stock\_data information*

stock\_data**.**info()

<class 'pandas.core.frame.DataFrame'>

Index: 4877 entries, 2001-01-02 to 2020-06-30

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Close 4877 non-null float64

1 compound 4877 non-null float64

2 negative 4877 non-null float64

3 neutral 4877 non-null float64

4 positive 4877 non-null float64

5 Open 4877 non-null float64

6 High 4877 non-null float64

7 Low 4877 non-null float64

8 Volume 4877 non-null float64

dtypes: float64(9)

*# setting figure size*

plt**.**figure(figsize**=**(16,10))

*# plotting close price*

stock\_data['Close']**.**plot()

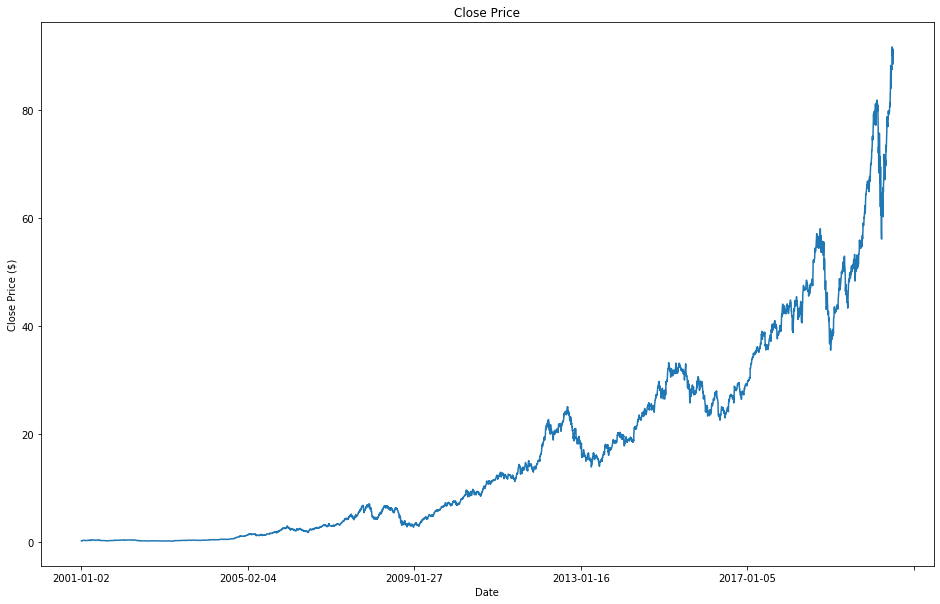
*# setting plot title, x and y labels*

plt**.**title("Close Price")

plt**.**xlabel('Date')

plt**.**ylabel('Close Price ($)')

Text(0, 0.5, 'Close Price ($)')



*# calculating 7 day rolling mean*

stock\_data**.**rolling(7)**.**mean()**.**head(20)

|  | **Close** | **compound** | **negative** | **neutral** | **positive** | **Open** | **High** | **Low** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** |  |  |  |  |  |  |  |  |  |
| **2001-01-02** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **2001-01-03** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **2001-01-04** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **2001-01-05** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **2001-01-08** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **2001-01-09** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **2001-01-10** | 0.293368 | -0.179071 | 0.121714 | 0.810429 | 0.067571 | 0.293088 | 0.304687 | 0.281728 | 566364000.0 |
| **2001-01-11** | 0.301340 | -0.041629 | 0.117857 | 0.810286 | 0.071571 | 0.296596 | 0.312978 | 0.286033 | 616567200.0 |
| **2001-01-23** | 0.311862 | 0.000843 | 0.111000 | 0.818429 | 0.070429 | 0.308873 | 0.323820 | 0.297832 | 625489600.0 |
| **2001-01-24** | 0.320631 | -0.223257 | 0.113714 | 0.821286 | 0.064714 | 0.315210 | 0.329400 | 0.304847 | 622303200.0 |
| **2001-01-25** | 0.329719 | -0.493057 | 0.118286 | 0.827857 | 0.053571 | 0.324458 | 0.337532 | 0.314254 | 633353600.0 |
| **2001-01-26** | 0.337372 | -0.227857 | 0.110714 | 0.822571 | 0.066429 | 0.330995 | 0.344747 | 0.322226 | 648942400.0 |
| **2001-01-29** | 0.348852 | 0.047229 | 0.099286 | 0.823571 | 0.077000 | 0.338010 | 0.355230 | 0.329879 | 687028000.0 |
| **2001-01-30** | 0.362085 | 0.006357 | 0.083571 | 0.828571 | 0.087714 | 0.350447 | 0.367985 | 0.342156 | 702976800.0 |
| **2001-01-31** | 0.371333 | -0.073514 | 0.084286 | 0.832286 | 0.083286 | 0.363839 | 0.378189 | 0.355389 | 692544000.0 |
| **2001-02-01** | 0.372927 | -0.324900 | 0.094286 | 0.825714 | 0.079857 | 0.367347 | 0.379624 | 0.359056 | 619710400.0 |
| **2001-02-02** | 0.373246 | -0.057300 | 0.090286 | 0.804429 | 0.105286 | 0.368623 | 0.382813 | 0.361448 | 578317600.0 |
| **2001-02-05** | 0.373884 | 0.158714 | 0.087571 | 0.797000 | 0.115429 | 0.368463 | 0.382813 | 0.361448 | 549232800.0 |
| **2001-02-06** | 0.377870 | 0.140486 | 0.087286 | 0.799000 | 0.113714 | 0.370137 | 0.386838 | 0.363839 | 546360000.0 |
| **2001-02-07** | 0.375478 | 0.121786 | 0.090000 | 0.796714 | 0.113286 | 0.372927 | 0.384606 | 0.364477 | 480411200.0 |

*# setting figure size*

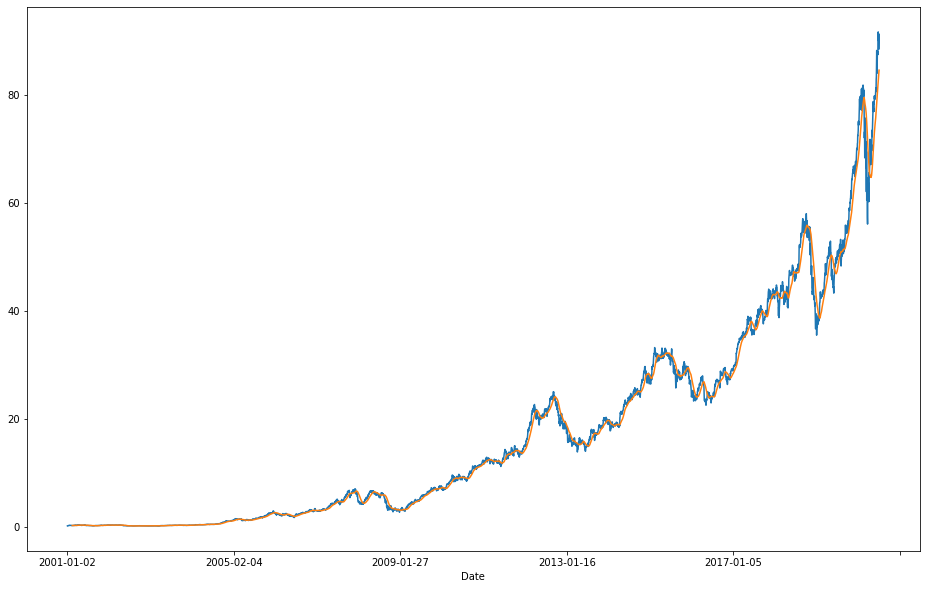
plt**.**figure(figsize**=**(16,10))

*# plotting the close price and a 30-day rolling mean of close price*

stock\_data['Close']**.**plot()

stock\_data**.**rolling(window**=**30)**.**mean()['Close']**.**plot()

<matplotlib.axes.\_subplots.AxesSubplot at 0x259bf683708>



*# displaying stock\_data*

stock\_data

|  | **Close** | **compound** | **negative** | **neutral** | **positive** | **Open** | **High** | **Low** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** |  |  |  |  |  |  |  |  |  |
| **2001-01-02** | 0.265625 | -0.9621 | 0.119 | 0.817 | 0.064 | 0.265625 | 0.272321 | 0.260045 | 452312000.0 |
| **2001-01-03** | 0.292411 | 0.6322 | 0.084 | 0.817 | 0.098 | 0.258929 | 0.297991 | 0.257813 | 817073600.0 |
| **2001-01-04** | 0.304688 | 0.6648 | 0.077 | 0.843 | 0.080 | 0.323940 | 0.330357 | 0.300223 | 739396000.0 |
| **2001-01-05** | 0.292411 | 0.9032 | 0.105 | 0.746 | 0.149 | 0.302455 | 0.310268 | 0.286830 | 412356000.0 |
| **2001-01-08** | 0.295759 | -0.9638 | 0.119 | 0.855 | 0.026 | 0.302455 | 0.303292 | 0.284598 | 373699200.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **2020-06-24** | 90.014999 | -0.9997 | 0.115 | 0.811 | 0.074 | 91.250000 | 92.197502 | 89.629997 | 192623200.0 |
| **2020-06-25** | 91.209999 | -0.9999 | 0.126 | 0.820 | 0.054 | 90.175003 | 91.250000 | 89.392502 | 137522400.0 |
| **2020-06-26** | 88.407501 | -0.9999 | 0.149 | 0.766 | 0.085 | 91.102501 | 91.330002 | 88.254997 | 205256800.0 |
| **2020-06-29** | 90.445000 | -0.9997 | 0.119 | 0.804 | 0.078 | 88.312500 | 90.542503 | 87.820000 | 130646000.0 |
| **2020-06-30** | 91.199997 | -0.9998 | 0.115 | 0.821 | 0.064 | 90.019997 | 91.495003 | 90.000000 | 140223200.0 |

*# calculating data\_to\_use*

percentage\_of\_data **=** 1.0

data\_to\_use **=** int(percentage\_of\_data**\***(len(stock\_data)**-**1))

*# using 80% of data for training*

train\_end **=** int(data\_to\_use**\***0.8)

total\_data **=** len(stock\_data)

start **=** total\_data **-** data\_to\_use

*# printing number of records in the training and test datasets*

print("Number of records in Training Data:", train\_end)

print("Number of records in Test Data:", total\_data **-** train\_end)

Number of records in Training Data: 3900

Number of records in Test Data: 977

*# predicting one step ahead*

steps\_to\_predict **=** 1

*# capturing data to be used for each column*

close\_price **=** stock\_data**.**iloc[start:total\_data,0] *#close*

compound **=** stock\_data**.**iloc[start:total\_data,1] *#compound*

negative **=** stock\_data**.**iloc[start:total\_data,2] *#neg*

neutral **=** stock\_data**.**iloc[start:total\_data,3] *#neu*

positive **=** stock\_data**.**iloc[start:total\_data,4] *#pos*

open\_price **=** stock\_data**.**iloc[start:total\_data,5] *#open*

high **=** stock\_data**.**iloc[start:total\_data,6] *#high*

low **=** stock\_data**.**iloc[start:total\_data,7] *#low*

volume **=** stock\_data**.**iloc[start:total\_data,8] *#volume*

*# printing close price*

print("Close Price:")

close\_price

Close Price:

Out[25]:

Date

2001-01-03 0.292411

2001-01-04 0.304688

2001-01-05 0.292411

2001-01-08 0.295759

2001-01-09 0.306920

...

2020-06-24 90.014999

2020-06-25 91.209999

2020-06-26 88.407501

2020-06-29 90.445000

2020-06-30 91.199997

Name: Close, Length: 4876, dtype: float64

*# shifting next day close*

close\_price\_shifted **=** close\_price**.**shift(**-**1)

*# shifting next day compound*

compound\_shifted **=** compound**.**shift(**-**1)

*# concatenating the captured training data into a dataframe*

data **=** pd**.**concat([close\_price, close\_price\_shifted, compound, compound\_shifted, volume, open\_price, high, low], axis**=**1)

*# setting column names of the revised stock data*

data**.**columns **=** ['close\_price', 'close\_price\_shifted', 'compound', 'compound\_shifted','volume', 'open\_price', 'high', 'low']

*# dropping nulls*

data **=** data**.**dropna()

data**.**head(10)

|  | **close\_price** | **close\_price\_shifted** | **compound** | **compound\_shifted** | **volume** | **open\_price** | **high** | **low** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** |  |  |  |  |  |  |  |  |
| **2001-01-03** | 0.292411 | 0.304688 | 0.6322 | 0.6648 | 817073600.0 | 0.258929 | 0.297991 | 0.257813 |
| **2001-01-04** | 0.304688 | 0.292411 | 0.6648 | 0.9032 | 739396000.0 | 0.323940 | 0.330357 | 0.300223 |
| **2001-01-05** | 0.292411 | 0.295759 | 0.9032 | -0.9638 | 412356000.0 | 0.302455 | 0.310268 | 0.286830 |
| **2001-01-08** | 0.295759 | 0.306920 | -0.9638 | -0.9559 | 373699200.0 | 0.302455 | 0.303292 | 0.284598 |
| **2001-01-09** | 0.306920 | 0.295759 | -0.9559 | -0.5719 | 588929600.0 | 0.300223 | 0.315011 | 0.295759 |
| **2001-01-10** | 0.295759 | 0.321429 | -0.5719 | 0.0000 | 580781600.0 | 0.297991 | 0.303571 | 0.286830 |
| **2001-01-11** | 0.321429 | 0.366071 | 0.0000 | 0.9295 | 803734400.0 | 0.290179 | 0.330357 | 0.290179 |
| **2001-01-23** | 0.366071 | 0.366071 | 0.9295 | -0.9039 | 879530400.0 | 0.344866 | 0.373884 | 0.340402 |
| **2001-01-24** | 0.366071 | 0.356027 | -0.9039 | -0.9854 | 717091200.0 | 0.368304 | 0.369420 | 0.349330 |
| **2001-01-25** | 0.356027 | 0.349330 | -0.9854 | 0.8926 | 489708800.0 | 0.367188 | 0.367188 | 0.352679 |

*# setting the target variable as the shifted close\_price*

y **=** data['close\_price\_shifted']

y

Date

2001-01-03 0.304688

2001-01-04 0.292411

2001-01-05 0.295759

2001-01-08 0.306920

2001-01-09 0.295759

...

2020-06-23 90.014999

2020-06-24 91.209999

2020-06-25 88.407501

2020-06-26 90.445000

2020-06-29 91.199997

Name: close\_price\_shifted, Length: 4875, dtype: float64

*# setting the features dataset for prediction*

cols **=** ['close\_price', 'compound', 'compound\_shifted', 'volume', 'open\_price', 'high', 'low']

x **=** data[cols]

x

| **close\_price** | **compound** | **compound\_shifted** | **volume** | **open\_price** | **high** | **low** |
| --- | --- | --- | --- | --- | --- | --- |
| **Date** |  |  |  |  |  |  |  |
| **2001-01-03** | 0.292411 | 0.6322 | 0.6648 | 817073600.0 | 0.258929 | 0.297991 | 0.257813 |
| **2001-01-04** | 0.304688 | 0.6648 | 0.9032 | 739396000.0 | 0.323940 | 0.330357 | 0.300223 |
| **2001-01-05** | 0.292411 | 0.9032 | -0.9638 | 412356000.0 | 0.302455 | 0.310268 | 0.286830 |
| **2001-01-08** | 0.295759 | -0.9638 | -0.9559 | 373699200.0 | 0.302455 | 0.303292 | 0.284598 |
| **2001-01-09** | 0.306920 | -0.9559 | -0.5719 | 588929600.0 | 0.300223 | 0.315011 | 0.295759 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **2020-06-23** | 91.632500 | -0.9999 | -0.9997 | 212155600.0 | 91.000000 | 93.095001 | 90.567497 |
| **2020-06-24** | 90.014999 | -0.9997 | -0.9999 | 192623200.0 | 91.250000 | 92.197502 | 89.629997 |
| **2020-06-25** | 91.209999 | -0.9999 | -0.9999 | 137522400.0 | 90.175003 | 91.250000 | 89.392502 |
| **2020-06-26** | 88.407501 | -0.9999 | -0.9997 | 205256800.0 | 91.102501 | 91.330002 | 88.254997 |
| **2020-06-29** | 90.445000 | -0.9997 | -0.9998 | 130646000.0 | 88.312500 | 90.542503 | 87.820000 |

4875 rows × 7 columns

*# scaling the feature dataset*

scaler\_x **=** preprocessing**.**MinMaxScaler (feature\_range**=**(**-**1, 1))

x **=** np**.**array(x)**.**reshape((len(x) ,len(cols)))

x **=** scaler\_x**.**fit\_transform(x)

*# scaling the target variable*

scaler\_y **=** preprocessing**.**MinMaxScaler (feature\_range**=**(**-**1, 1))

y **=** np**.**array (y)**.**reshape ((len( y), 1))

y **=** scaler\_y**.**fit\_transform (y)

*# displaying the scaled feature dataset and the target variable*

x, y

(array([[-0.99872809, 0.6322 , 0.6648 , ..., -0.99940748,

-0.99865485, -0.99932101],

[-0.99845944, 0.6648 , 0.9032 , ..., -0.99797895,

-0.99795775, -0.99838212],

[-0.99872809, 0.9032 , -0.9638 , ..., -0.99845105,

-0.99839043, -0.99867862],

...,

[ 0.99075472, -0.9999 , -0.9999 , ..., 0.97637837,

0.96026251, 0.97398737],

[ 0.92942972, -0.9999 , -0.9997 , ..., 0.99675891,

0.96198559, 0.94880472],

[ 0.97401481, -0.9997 , -0.9998 , ..., 0.93545235,

0.94502449, 0.93917454]]),

array([[-0.99845944],

[-0.99872809],

[-0.99865483],

...,

[ 0.92942972],

[ 0.97401481],

[ 0.99053585]]))

*# preparing training and test dataset*

X\_train **=** x[0 : train\_end,]

X\_test **=** x[train\_end**+**1 : len(x),]

y\_train **=** y[0 : train\_end]

y\_test **=** y[train\_end**+**1 : len(y)]

*# printing the shape of the training and the test datasets*

print('Number of rows and columns in the Training set X:', X\_train**.**shape, 'and y:', y\_train**.**shape)

print('Number of rows and columns in the Test set X:', X\_test**.**shape, 'and y:', y\_test**.**shape)

Number of rows and columns in the Training set X: (3900, 7) and y: (3900, 1)

Number of rows and columns in the Test set X: (974, 7) and y: (974, 1)

*# reshaping the feature dataset for feeding into the model*

X\_train **=** X\_train**.**reshape (X\_train**.**shape **+** (1,))

X\_test **=** X\_test**.**reshape(X\_test**.**shape **+** (1,))

*# printing the re-shaped feature dataset*

print('Shape of Training set X:', X\_train**.**shape)

print('Shape of Test set X:', X\_test**.**shape)

Shape of Training set X: (3900, 7, 1)

Shape of Test set X: (974, 7, 1)

*# setting the seed to achieve consistent and less random predictions at each execution*

np**.**random**.**seed(2016)

*# setting the model architecture*

model**=**Sequential()

model**.**add(LSTM(100,return\_sequences**=True**,activation**=**'tanh',input\_shape**=**(len(cols),1)))

model**.**add(Dropout(0.1))

model**.**add(LSTM(100,return\_sequences**=True**,activation**=**'tanh'))

model**.**add(Dropout(0.1))

model**.**add(LSTM(100,activation**=**'tanh'))

model**.**add(Dropout(0.1))

model**.**add(Dense(1))

*# printing the model summary*

model**.**summary()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

lstm\_1 (LSTM) (None, 7, 100) 40800

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 7, 100) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lstm\_2 (LSTM) (None, 7, 100) 80400

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_2 (Dropout) (None, 7, 100) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lstm\_3 (LSTM) (None, 100) 80400

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_3 (Dropout) (None, 100) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 1) 101

=================================================================

Total params: 201,701

Trainable params: 201,701

Non-trainable params: 0

*# compiling the model*

model**.**compile(loss**=**'mse' , optimizer**=**'adam')

*# fitting the model using the training dataset*

model**.**fit(X\_train, y\_train, validation\_split**=**0.2, epochs**=**10, batch\_size**=**8, verbose**=**1)

Train on 3120 samples, validate on 780 samples

Epoch 1/10

3120/3120 [==============================] - 14s 4ms/step - loss: 0.0335 - val\_loss: 0.1489

Epoch 2/10

3120/3120 [==============================] - 13s 4ms/step - loss: 0.0090 - val\_loss: 0.0014

Epoch 3/10

3120/3120 [==============================] - 12s 4ms/step - loss: 0.0025 - val\_loss: 3.4762e-04

Epoch 4/10

3120/3120 [==============================] - 13s 4ms/step - loss: 0.0021 - val\_loss: 8.7089e-04

Epoch 5/10

3120/3120 [==============================] - 14s 4ms/step - loss: 0.0020 - val\_loss: 8.4524e-04

Epoch 6/10

3120/3120 [==============================] - 14s 4ms/step - loss: 0.0015 - val\_loss: 0.0013

Epoch 7/10

3120/3120 [==============================] - 14s 5ms/step - loss: 0.0015 - val\_loss: 2.2926e-04

Epoch 8/10

3120/3120 [==============================] - 14s 5ms/step - loss: 0.0012 - val\_loss: 3.0356e-04

Epoch 9/10

3120/3120 [==============================] - 15s 5ms/step - loss: 0.0014 - val\_loss: 6.0174e-04

Epoch 10/10

3120/3120 [==============================] - 15s 5ms/step - loss: 0.0013 - val\_loss: 1.4272e-04

Out[33]:

<keras.callbacks.callbacks.History at 0x259c59d5248>

*# saving the model as a json file*

model\_json **=** model**.**to\_json()

**with** open('model.json', 'w') **as** json\_file:

json\_file**.**write(model\_json)

*# serialize weights to HDF5*

model**.**save\_weights('model.h5')

print('Model is saved to the disk')

Model is saved to the disk

*# performing predictions*

predictions **=** model**.**predict(X\_test)

*# unscaling the predictions*

predictions **=** scaler\_y**.**inverse\_transform(np**.**array(predictions)**.**reshape((len(predictions), 1)))

*# printing the predictions*

print('Predictions:')

predictions[0:5]

Predictions:

Out[35]:

array([[27.534704],

[27.351366],

[27.448437],

[27.37178 ],

[27.2595 ]], dtype=float32)

*# calculating the training mean-squared-error*

train\_loss **=** model**.**evaluate(X\_train, y\_train, batch\_size **=** 1)

*# calculating the test mean-squared-error*

test\_loss **=** model**.**evaluate(X\_test, y\_test, batch\_size **=** 1)

*# printing the training and the test mean-squared-errors*

print('Train Loss =', round(train\_loss,4))

print('Test Loss =', round(test\_loss,4))

3900/3900 [==============================] - 69s 18ms/step

974/974 [==============================] - 18s 19ms/step

Train Loss = 0.0002

Test Loss = 0.001

*# calculating root mean squared error*

root\_mean\_square\_error **=** np**.**sqrt(np**.**mean(np**.**power((y\_test **-** predictions),2)))

print('Root Mean Square Error =', round(root\_mean\_square\_error,4))

Root Mean Square Error = 50.3044

*# calculating root mean squared error using sklearn.metrics package*

rmse **=** metrics**.**mean\_squared\_error(y\_test, predictions)

print('Root Mean Square Error (sklearn.metrics) =', round(np**.**sqrt(rmse),4))

Root Mean Square Error (sklearn.metrics) = 50.3044

*# unscaling the test feature dataset, x\_test*

X\_test **=** scaler\_x**.**inverse\_transform(np**.**array(X\_test)**.**reshape((len(X\_test), len(cols))))

*# unscaling the test y dataset, y\_test*

y\_train **=** scaler\_y**.**inverse\_transform(np**.**array(y\_train)**.**reshape((len(y\_train), 1)))

y\_test **=** scaler\_y**.**inverse\_transform(np**.**array(y\_test)**.**reshape((len(y\_test), 1)))

*# plotting*

plt**.**figure(figsize**=**(16,10))

*# plt.plot([row[0] for row in y\_train], label="Training Close Price")*

plt**.**plot(predictions, label**=**"Predicted Close Price")

plt**.**plot([row[0] **for** row **in** y\_test], label**=**"Testing Close Price")

plt**.**legend(loc**=**'upper center', bbox\_to\_anchor**=**(0.5, **-**0.05), fancybox**=True**, shadow**=True**, ncol**=**2)

plt**.**show()

