Mahesh Kumar Ghatti
Instructor's Name : Jorge Novillo
Course Title : CS 598
May 11, 2017

Sentiment Analysis of Twitter Data for Predicting Stock Market Movements

Abstract

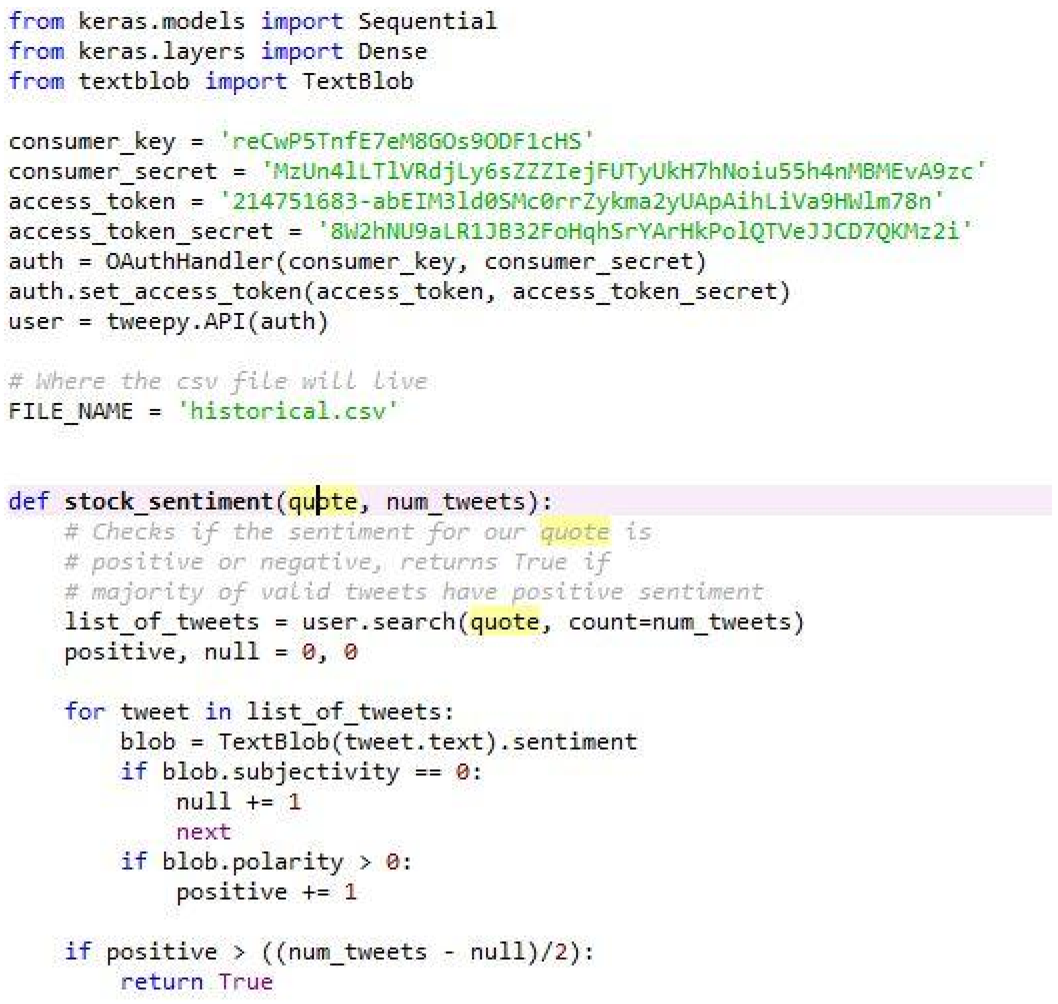
Predicting stock market movements is a well-known problem of interest. The thesis of this work is to observe how well the changes in stock prices of a company, the rises and falls, are correlated with the public opinions being expressed in tweets about that company. In this paper, I have applied sentiment analysis and supervised machine learning principles to the tweets extracted from twitter and analyze the correlation between stock market movements of a company and sentiments in tweets. In an elaborate way, positive news and tweets in social media about a company would definitely encourage people to invest in the stocks of that company and as a result the stock price of that company would increase. At the end of the paper, it is shown that a strong correlation exists between the rise and falls in stock prices with the public sentiments in tweets.

Sentiment Analysis

Tweets are classified as positive, negative and neutral based on the sentiment present. 3,216 tweets out of the total tweets are examined by humans and annotated as 1 for Positive, 0 for Neutral and 2 for Negative emotions. For classification of nonhuman annotated tweets a machine learning model is trained whose features are extracted from the human annotated tweets.

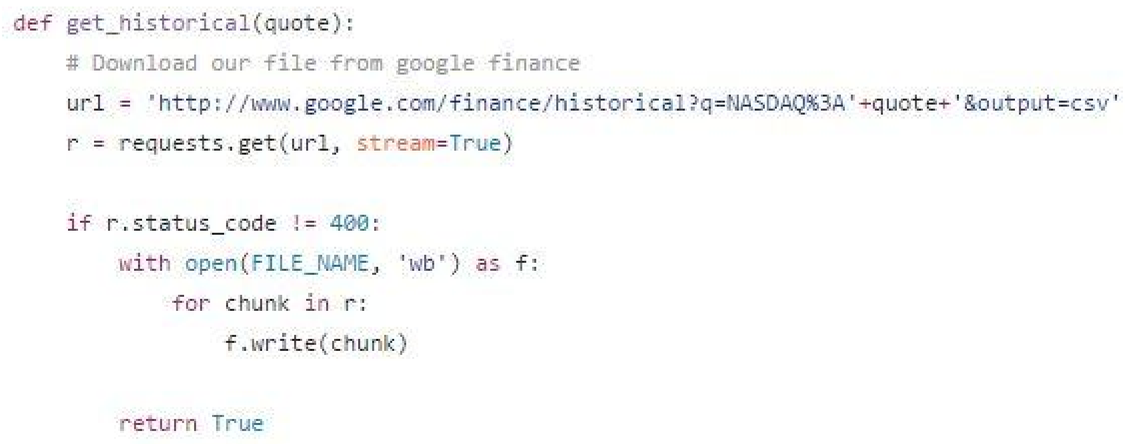
I have used Keras library and oauth keys for pulling the tweets from twitter tweepy library and classifying them by using keras library. If majority of the tweets are positive then keras library with neural network is used for stock prediction from the tweets. The core data structure of Keras is a **model**, a way to organize layers. The simplest type of model is the Sequential model, a linear stack of layers. The devised classifier is used to predict the emotions of non-human annotated tweets. The code for sentiment analysis is given below by using keras library.

Code:



```python
from keras.models import Sequential
from keras.layers import Dense
from textblob import TextBlob

consumer_key = 'reCwP5TnfE7eM8GOs9ODF1cHS'
consumer_secret = 'MzUn4lLTlVRdjLy6sZZZIejFUTyUkH7hNoiu55h4nMBMEvA9zc'
access_token = '214751683-abEIM3ld0SMc0rrZykma2yUApAihLiVa9HWlm78n'
access_token_secret = '8W2hNU9aLR1JB32FoHqhSrYArHkPolQTVeJJCD7QKMz2i'
auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
user = tweepy.API(auth)

# Where the csv file will live
FILE_NAME = 'historical.csv'


def stock_sentiment(quote, num_tweets):
    # Checks if the sentiment for our quote is
    # positive or negative, returns True if
    # majority of valid tweets have positive sentiment
    list_of_tweets = user.search(quote, count=num_tweets)
    positive, null = 0, 0

    for tweet in list_of_tweets:
        blob = TextBlob(tweet.text).sentiment
        if blob.subjectivity == 0:
            null += 1
            next
        if blob.polarity > 0:
            positive += 1

    if positive > ((num_tweets - null)/2):
        return True
```

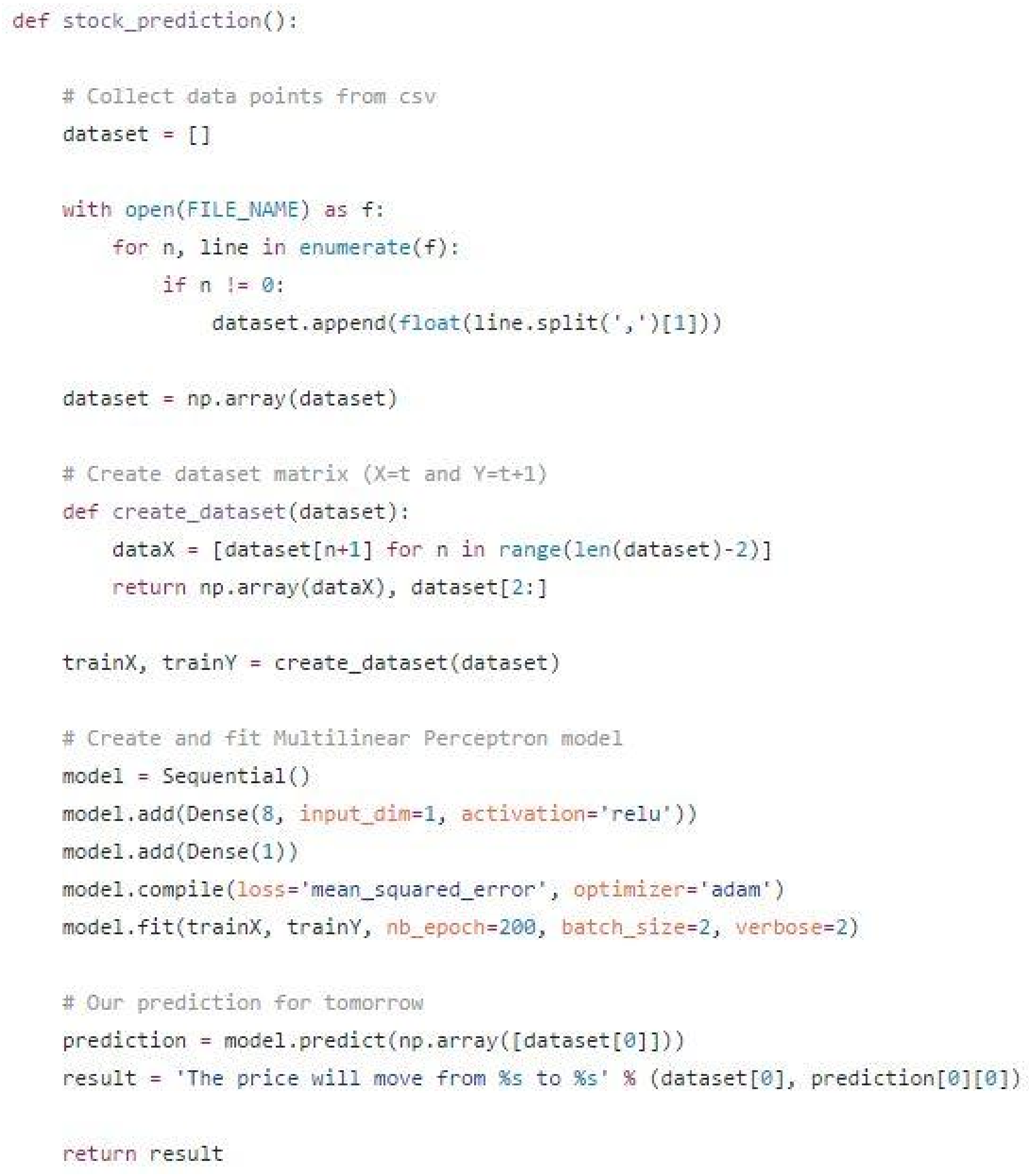
Getting the data from google finance for predicting the stocks about the company.



```python
def get_historical(quote):
    # Download our file from google finance
    url = 'http://www.google.com/finance/historical?q=NASDAQ%3A'+quote+'&output=csv'
    r = requests.get(url, stream=True)

    if r.status_code != 400:
        with open(FILE_NAME, 'wb') as f:
            for chunk in r:
                f.write(chunk)

        return True
```

Predicting Stocks

The features extracted using the above functions for the human annotated tweets are fed to the classifier and trained using random forest algorithm.It involves preprocessing the dataset and training the dataset by using keras model. A total of 355 instances, each with 3 attributes are fed to the classifier with a split proportions of 80% train dataset and the remaining dataset for testing.



```python
def stock_prediction():

    # Collect data points from csv
    dataset = []

    with open(FILE_NAME) as f:
        for n, line in enumerate(f):
            if n != 0:
                dataset.append(float(line.split(',')[1]))

    dataset = np.array(dataset)

    # Create dataset matrix (X=t and Y=t+1)
    def create_dataset(dataset):
        dataX = [dataset[n+1] for n in range(len(dataset)-2)]
        return np.array(dataX), dataset[2:]

    trainX, trainY = create_dataset(dataset)

    # Create and fit Multilinear Perceptron model
    model = Sequential()
    model.add(Dense(8, input_dim=1, activation='relu'))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    model.fit(trainX, trainY, nb_epoch=200, batch_size=2, verbose=2)

    # Our prediction for tomorrow
    prediction = model.predict(np.array([dataset[0]]))
    result = 'The price will move from %s to %s' % (dataset[0], prediction[0][0])

    return result
```

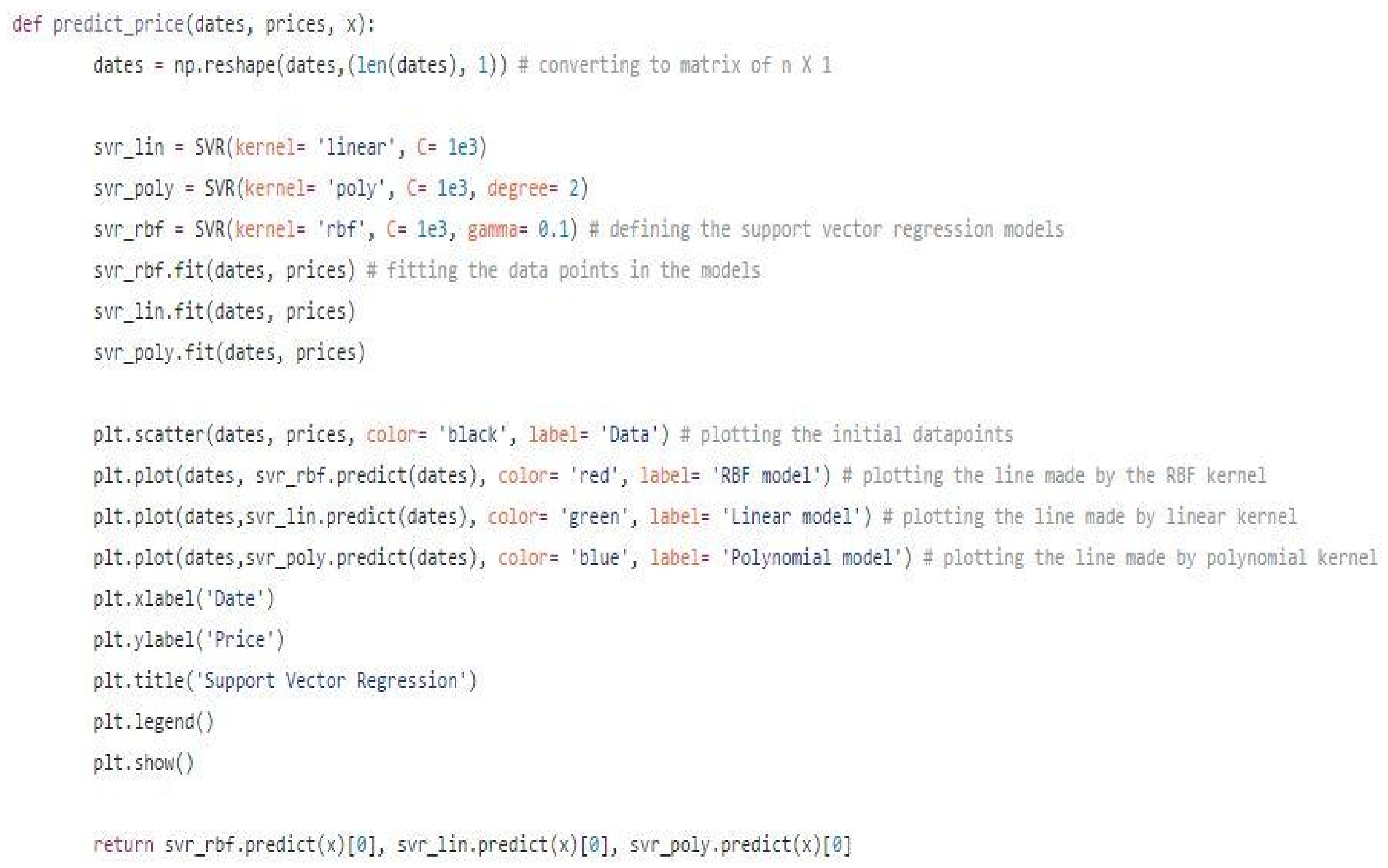
Predicting Stocks through SVR model

<u>SVR</u>

Support vector machines (SVMs) are a set of supervised learning methods used
for classification, regression and outliers detection.

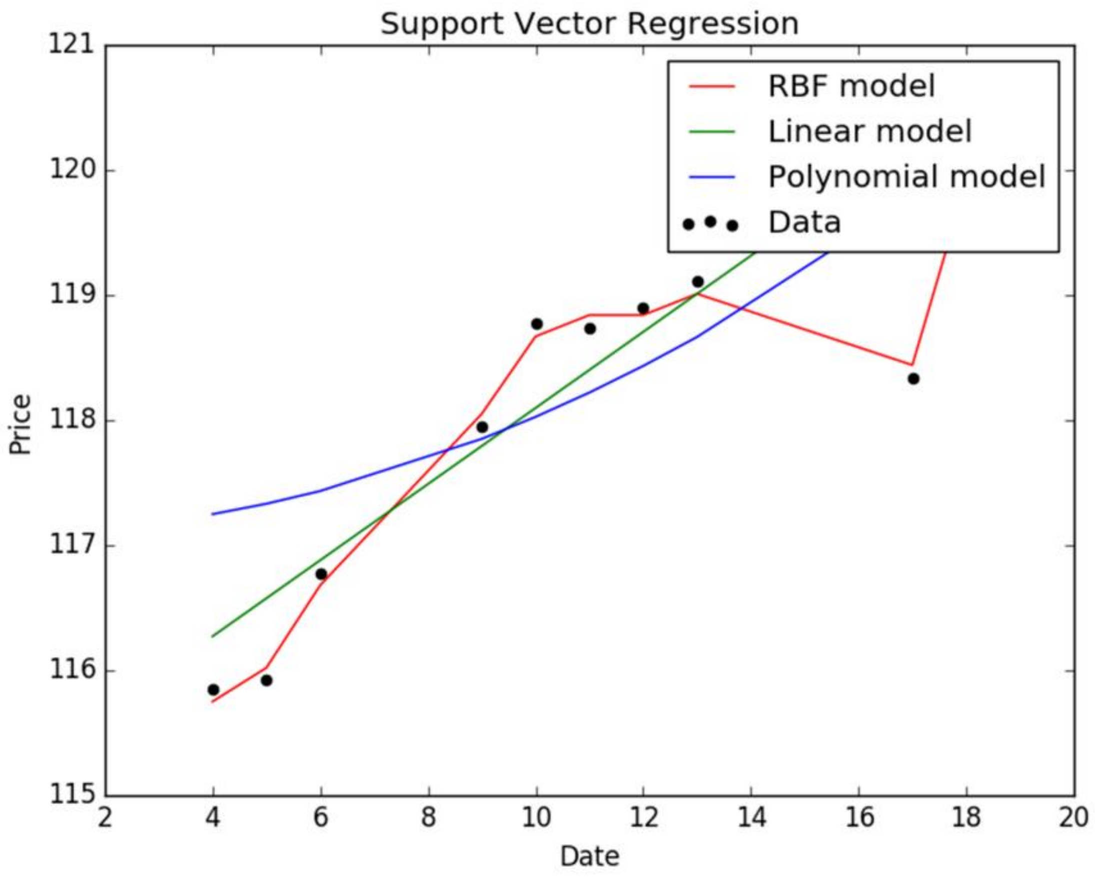
The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.



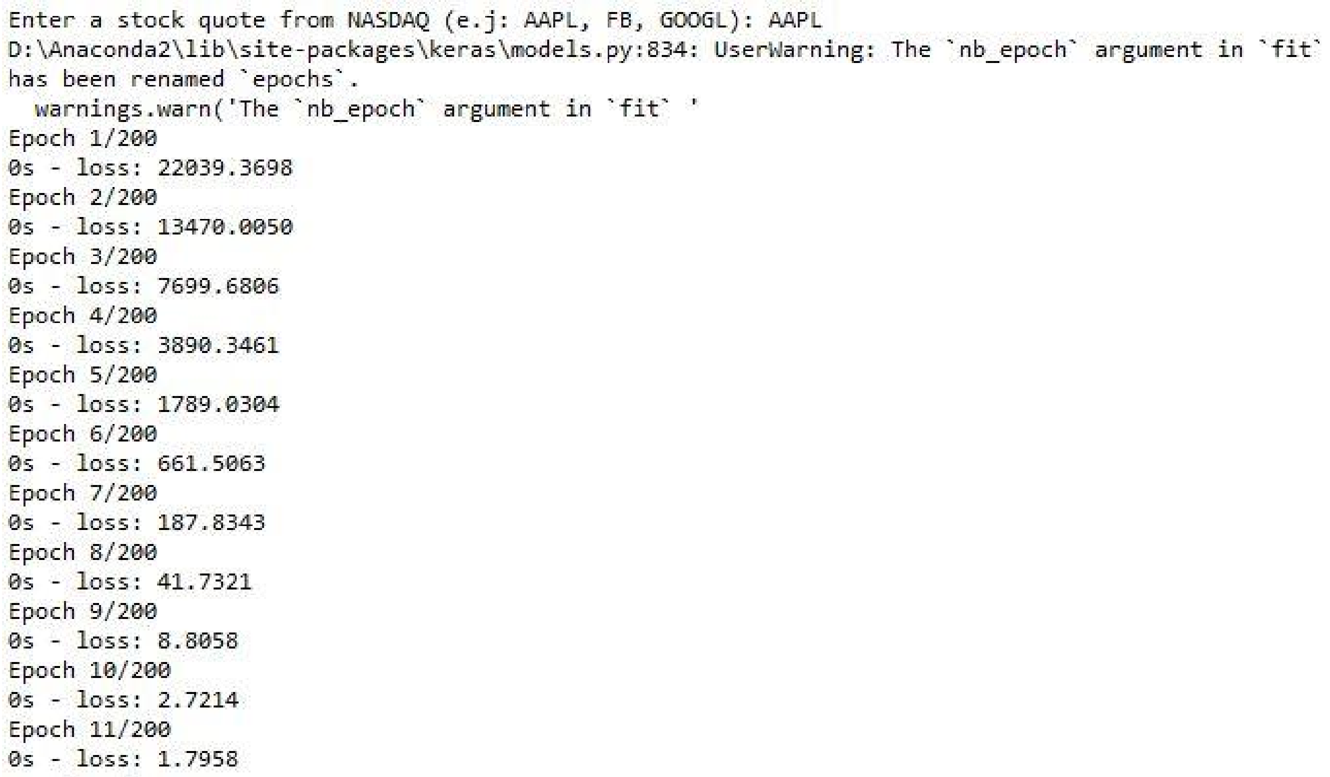
```python
def predict_price(dates, prices, x):
        dates = np.reshape(dates,(len(dates), 1)) # converting to matrix of n X 1

        svr_lin = SVR(kernel= 'linear', C= 1e3)
        svr_poly = SVR(kernel= 'poly', C= 1e3, degree= 2)
        svr_rbf = SVR(kernel= 'rbf', C= 1e3, gamma= 0.1) # defining the support vector regression models
        svr_rbf.fit(dates, prices) # fitting the data points in the models
        svr_lin.fit(dates, prices)
        svr_poly.fit(dates, prices)

        plt.scatter(dates, prices, color= 'black', label= 'Data') # plotting the initial datapoints
        plt.plot(dates, svr_rbf.predict(dates), color= 'red', label= 'RBF model') # plotting the line made by the RBF kernel
        plt.plot(dates,svr_lin.predict(dates), color= 'green', label= 'Linear model') # plotting the line made by linear kernel
        plt.plot(dates,svr_poly.predict(dates), color= 'blue', label= 'Polynomial model') # plotting the line made by polynomial kernel
        plt.xlabel('Date')
        plt.ylabel('Price')
        plt.title('Support Vector Regression')
        plt.legend()
        plt.show()

        return svr_rbf.predict(x)[0], svr_lin.predict(x)[0], svr_poly.predict(x)[0]
```

In the above code stock prices are predicted by using SVR library in python.The plot is shown below:

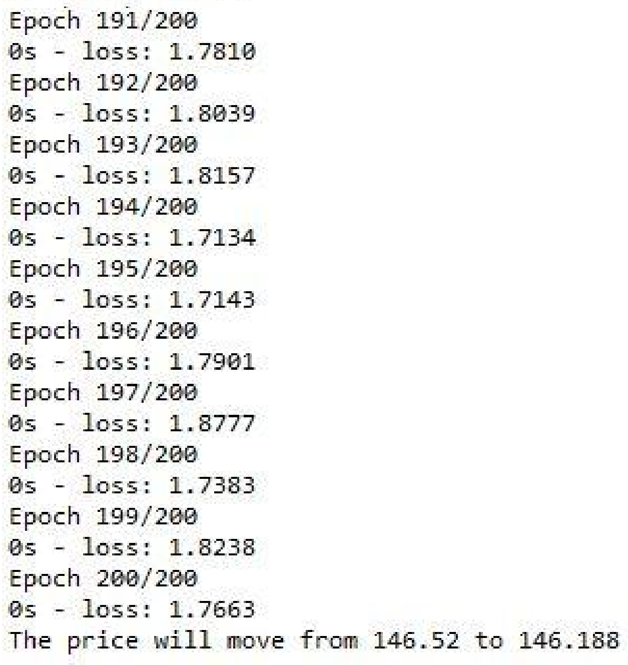


From the above visualization model, it is depicted that RBF model predicts the accurate data than linear and polynomial model.So in SVR, RBF model can be used for predicting the stock prices according to given date. But, using recurrent neural networks yields better results.

Output:



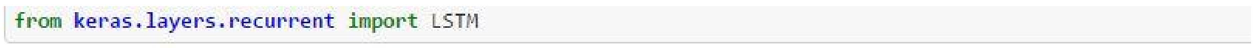
```
Enter a stock quote from NASDAQ (e.j: AAPL, FB, GOOGL): AAPL
D:\Anaconda2\lib\site-packages\keras\models.py:834: UserWarning: The `nb_epoch` argument in `fit`
has been renamed `epochs`.
  warnings.warn('The `nb_epoch` argument in `fit` '
Epoch 1/200
0s - loss: 22039.3698
Epoch 2/200
0s - loss: 13470.0050
Epoch 3/200
0s - loss: 7699.6806
Epoch 4/200
0s - loss: 3890.3461
Epoch 5/200
0s - loss: 1789.0304
Epoch 6/200
0s - loss: 661.5063
Epoch 7/200
0s - loss: 187.8343
Epoch 8/200
0s - loss: 41.7321
Epoch 9/200
0s - loss: 8.8058
Epoch 10/200
0s - loss: 2.7214
Epoch 11/200
0s - loss: 1.7958
```



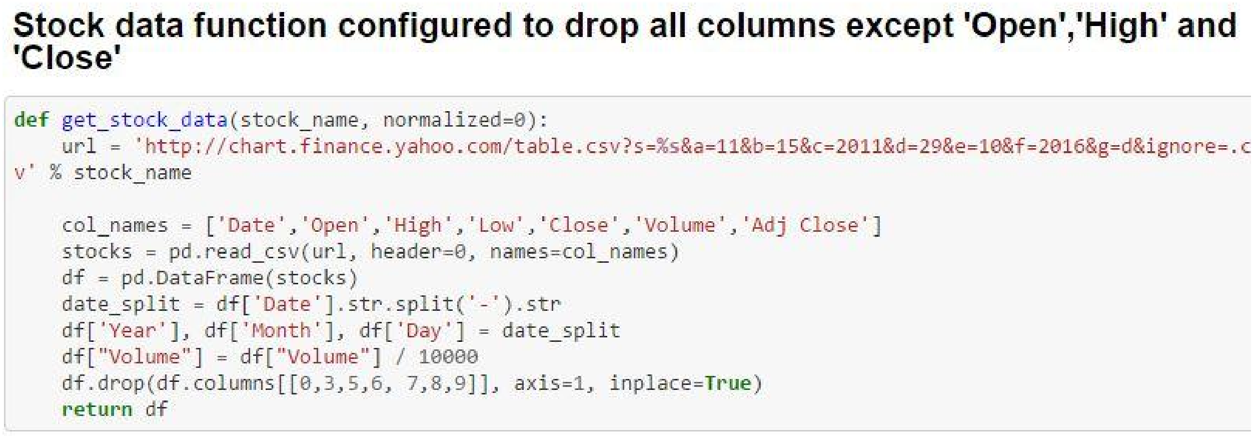
```
Epoch 191/200
0s - loss: 1.7810
Epoch 192/200
0s - loss: 1.8039
Epoch 193/200
0s - loss: 1.8157
Epoch 194/200
0s - loss: 1.7134
Epoch 195/200
0s - loss: 1.7143
Epoch 196/200
0s - loss: 1.7901
Epoch 197/200
0s - loss: 1.8777
Epoch 198/200
0s - loss: 1.7383
Epoch 199/200
0s - loss: 1.8238
Epoch 200/200
0s - loss: 1.7663
The price will move from 146.52 to 146.188
```

Stock prediction by the use of LSTM

LSTMs are explicitly designed to avoid the long-term dependency problem. Like most RNNs, a LSTM network is universal in the sense that given enough network units it can compute anything a conventional computer can compute, provided it has the proper weight matrix, which may be viewed as its program. Remembering information for long periods of time is practically their default behavior.I have taken google stocks for prediction by using LSTM in ipython notebook.

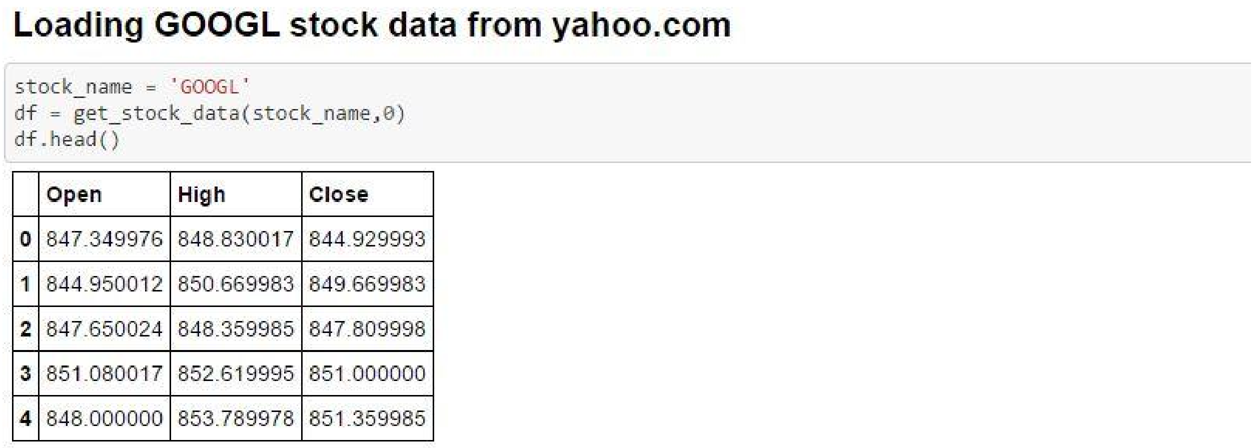


```
from keras.layers.recurrent import LSTM
```



Stock data function configured to drop all columns except 'Open','High' and 'Close'

```
def get_stock_data(stock_name, normalized=0):
    url = 'http://chart.finance.yahoo.com/table.csv?s=%s&a=11&b=15&c=2011&d=29&e=10&f=2016&g=d&ignore=.c
v' % stock_name

    col_names = ['Date','Open','High','Low','Close','Volume','Adj Close']
    stocks = pd.read_csv(url, header=0, names=col_names)
    df = pd.DataFrame(stocks)
    date_split = df['Date'].str.split('-').str
    df['Year'], df['Month'], df['Day'] = date_split
    df["Volume"] = df["Volume"] / 10000
    df.drop(df.columns[[0,3,5,6, 7,8,9]], axis=1, inplace=True)
    return df
```

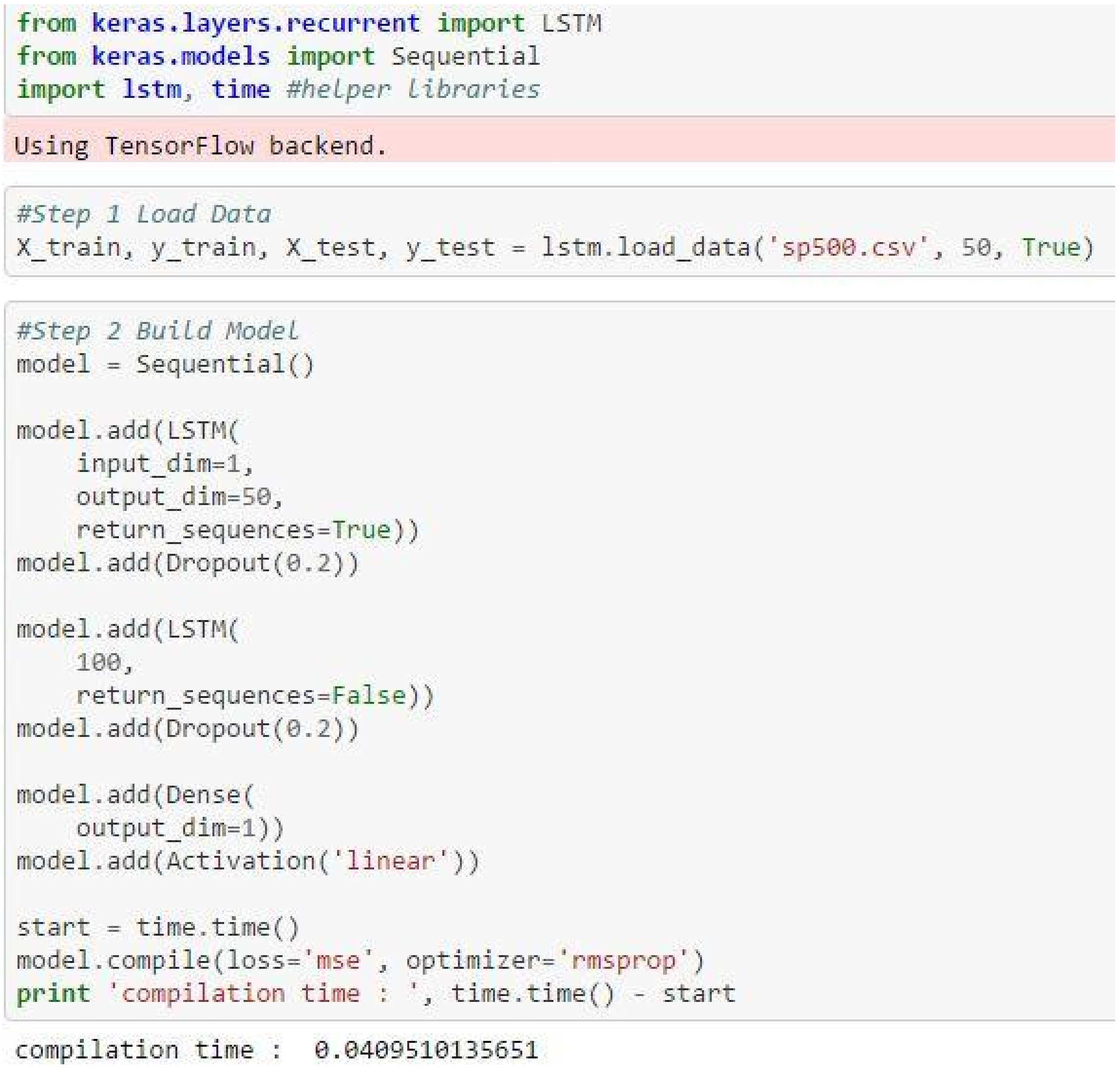


Loading GOOGL stock data from yahoo.com

```
stock_name = 'GOOGL'
df = get_stock_data(stock_name,0)
df.head()
```

	Open	High	Close
0	847.349976	848.830017	844.929993
1	844.950012	850.669983	849.669983
2	847.650024	848.359985	847.809998
3	851.080017	852.619995	851.000000
4	848.000000	853.789978	851.359985

The data is used for training and testing after the model is defined.



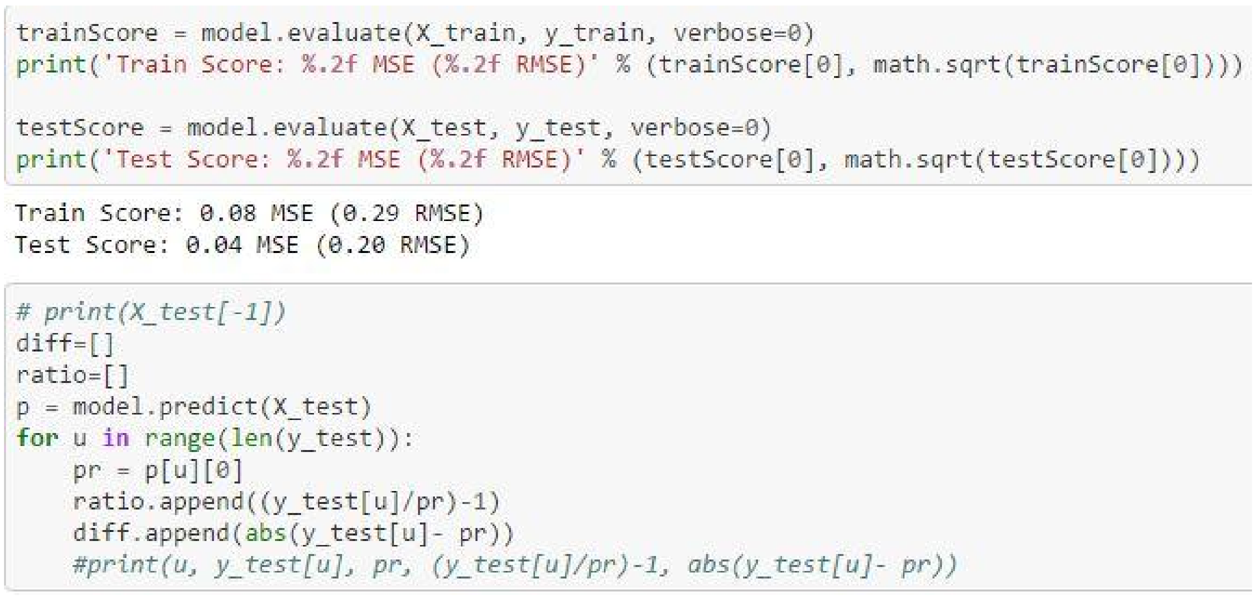
```
from keras.layers.recurrent import LSTM
from keras.models import Sequential
import lstm, time #helper libraries
```

```
Using TensorFlow backend.
```

```
#Step 1 Load Data
X_train, y_train, X_test, y_test = lstm.load_data('sp500.csv', 50, True)
```

```
#Step 2 Build Model
model = Sequential()

model.add(LSTM(
    input_dim=1,
    output_dim=50,
    return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(
    100,
    return_sequences=False))
model.add(Dropout(0.2))

model.add(Dense(
    output_dim=1))
model.add(Activation('linear'))

start = time.time()
model.compile(loss='mse', optimizer='rmsprop')
print 'compilation time : ', time.time() - start
```

```
compilation time :  0.0409510135651
```



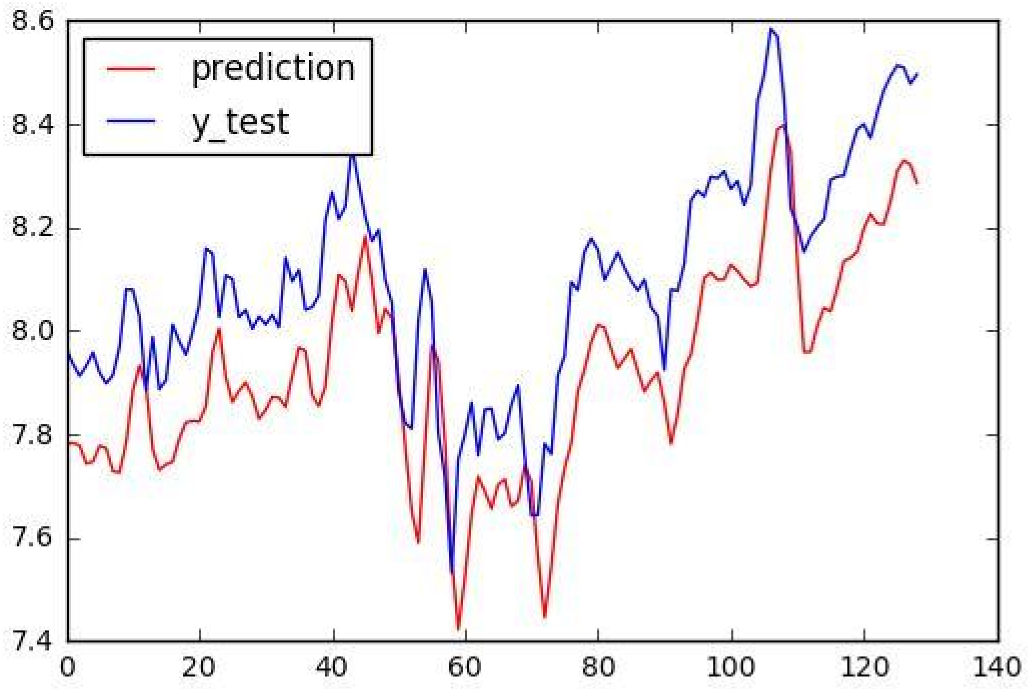
```
trainScore = model.evaluate(X_train, y_train, verbose=0)
print('Train Score: %.2f MSE (%.2f RMSE)' % (trainScore[0], math.sqrt(trainScore[0])))

testScore = model.evaluate(X_test, y_test, verbose=0)
print('Test Score: %.2f MSE (%.2f RMSE)' % (testScore[0], math.sqrt(testScore[0])))
```

```
Train Score: 0.08 MSE (0.29 RMSE)
Test Score: 0.04 MSE (0.20 RMSE)
```

```
# print(X_test[-1])
diff=[]
ratio=[]
p = model.predict(X_test)
for u in range(len(y_test)):
    pr = p[u][0]
    ratio.append((y_test[u]/pr)-1)
    diff.append(abs(y_test[u]- pr))
    #print(u, y_test[u], pr, (y_test[u]/pr)-1, abs(y_test[u]- pr))
```

Predictions vs Real results

```
import matplotlib.pyplot as plt2

plt2.plot(p,color='red', label='prediction')
plt2.plot(y_test,color='blue', label='y_test')
plt2.legend(loc='upper left')
plt2.show()
```



Total data is split into two parts, 80 percent to train the model and remaining for testing operations. The classifier results show an accuracy value of 69.01% when trained using Support vector regression algorithm and the accuracy rate varied with the training set. When the model with LSTM is trained with 90 percent of data, it gave a result of 71.82%.

Conclusion

In this paper, I have shown that a strong correlation exists between rise/fall in stock prices of a company to the public opinions or emotions about that company expressed on twitter through tweets.The main contribution of our work is the development of a stock price predictor by judging the type of sentiment present in the tweet.The tweets are classified into three categories: positive, negative and neutral. At the beginning, I have claimed that positive emotions or sentiment of public in twitter about a company would reflect in its stock price.Using different types of libraries in neural networks and estimating the accuracy of the stock price predicted from the positive tweets.

References

- [1] J. Bollen, H. Mao, X. Zeng. Twitter Mood Predicts the Stock Marker. arXiv: http://arxiv.org/abs/1010.3003
- [2] Alex Davies.A Word List for Sentiment Analysis of Twitter. http://alexdavies.net/2011/10/word-lists-for-sentiment-analysis-of-twitter/
- [3] SentiWordNet. An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. http://sentiwordnet.isti.cnr.it/
- [4] Jure Lescovic, Twitter Data. http://snap.stanford.edu/data/twitter7.html
- [5] Qian, Bo, Rasheed, Khaled, Stock market prediction with multiple classifiers, Applied Intelligence 26 (February (1)) (2007) 2533, http://dx.doi.org/10.1007/s10489-006-0001-7.
- E.F. Fama, The behavior of stock-market prices, The Journal of Business 38 1) (1965) 34105, http://dx.doi.org/10.2307/2350752
- J. Leskovec, L. Adamic and B. Huberman. The dynamics of viral marketing. In Proceedings of the 7th ACM Conference on Electronic Commerce. 2006