Advanced Big Data Analytics
Assignment 2

Rohan Kulkarni (rk2845)

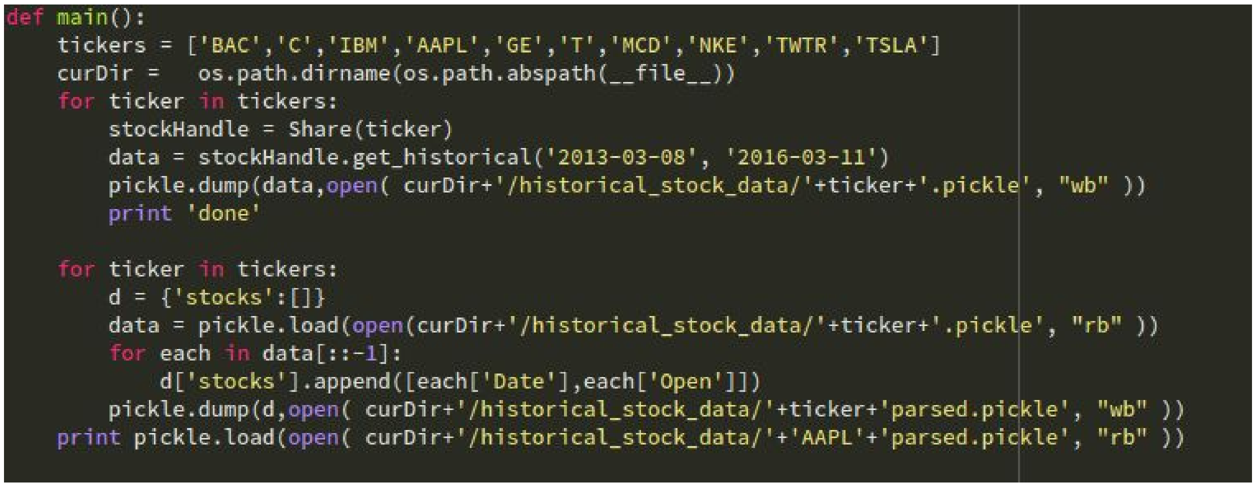
This assignment involves collection of historical stock data pertaining to various companies using their ticker values and using time series analysis techniques to predict the stock prices in the future. Following are the companies whose stocks have been analyzed using Auto Regressive Integrated Moving Average (ARIMA) time series analysis technique to predict their price in the future :

BAC = Bank of America
C = Citibank
IBM = IBM
AAPL = Apple
GE = General Electrics
T = AT&T
MCD = McDonald's corporation
NKE = Nike
TWTR = Twitter
TSLA = Tesla

Following are the various steps taken to accomplish stock prediction

1) Stock Data Collection using yahoo finance API

Using the yahoo finance API, historical stock price containing opening price, high , low and closing price for each day for the past 3 years for all stocks was downloaded :



```
def main():
    tickers = ['BAC','C','IBM','AAPL','GE','T','MCD','NKE','TWTR','TSLA']
    curDir =   os.path.dirname(os.path.abspath(__file__))
    for ticker in tickers:
        stockHandle = Share(ticker)
        data = stockHandle.get_historical('2013-03-08', '2016-03-11')
        pickle.dump(data,open( curDir+'/historical_stock_data/'+ticker+'.pickle', "wb" ))
        print 'done'

    for ticker in tickers:
        d = {'stocks':[]}
        data = pickle.load(open(curDir+'/historical_stock_data/'+ticker+'.pickle', "rb" ))
        for each in data[::-1]:
            d['stocks'].append([each['Date'],each['Open']])
        pickle.dump(d,open( curDir+'/historical_stock_data/'+ticker+'parsed.pickle', "wb" ))
    print pickle.load(open( curDir+'/historical_stock_data/'+'AAPL'+'parsed.pickle', "rb" ))
```

The stock prices received in the JSON format were parsed to make use of only “opening prices” for every day and the corresponding “date” and stored as pickle files to enable easy access in the future.

2. **Using ARIMA model to predict stock prices :**

How is stock price data a time series ?

Stock data that was collected consisted of 1 reading (stock price) per day. This makes stock data qualify as a time series.

About ARIMA :

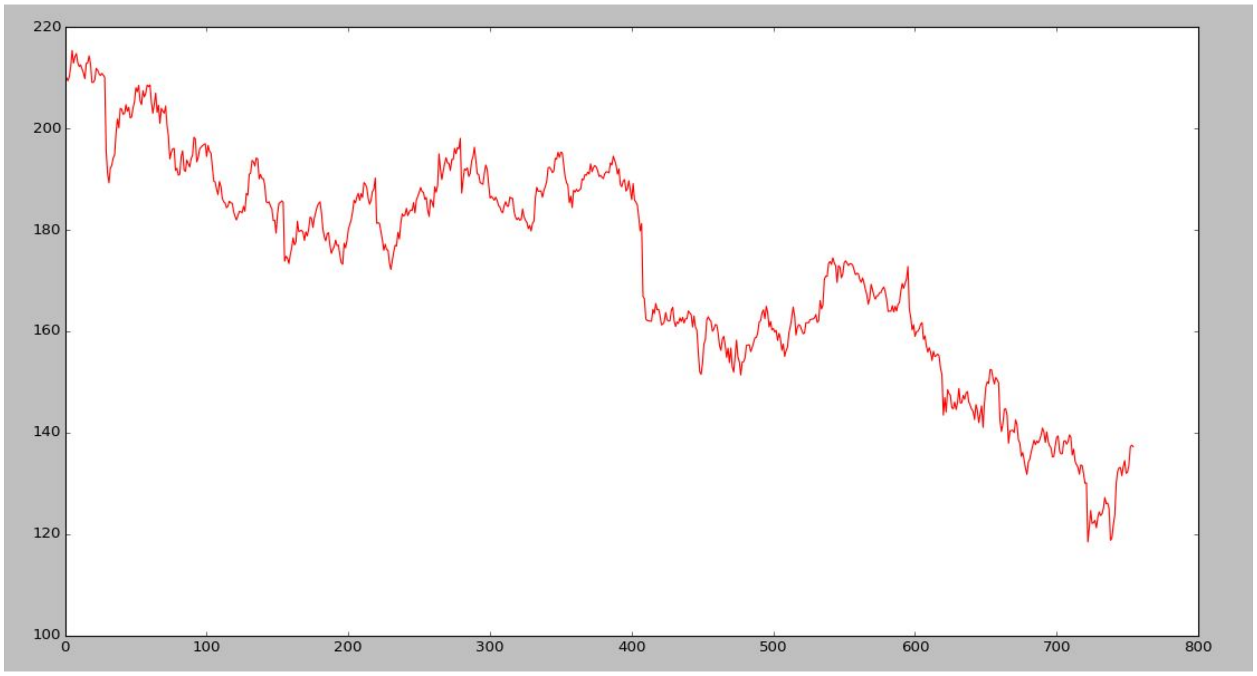
The wikipedia definition of ARIMA is as follows:

“In statistics and econometrics, and in particular in time series analysis, an autoregressive integrated moving average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model. These models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). They are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied to reduce the non-stationarity.”

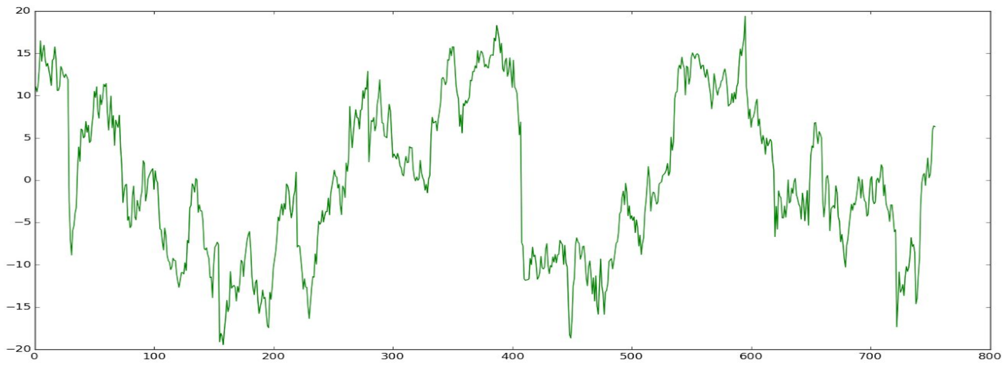
The ARIMA implementation in the package “statsmodels” in python was used for this assignment.

In ARIMA, we provide three parameters namely **(p,d,q)** corresponding to the lags for the autoregressive (AR), Integrated (I) and moving average (MA) parts of the model. These lags essentially determine how many of the past data points have an effect on the future value of the time series. One important note here is that for ARIMA to have good prediction results, the data should be stationary : ie, all the important characteristics defining the data like its mean and variance should remain almost constant over time.

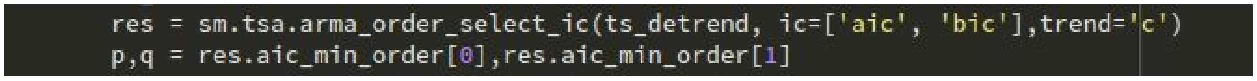
One can perform “de-trending” in order to impose stationarity in the data. For example , given the non stationary time series:



We can use the de_trend function in statsmodels to convert the above time series into a stationary one as follows :

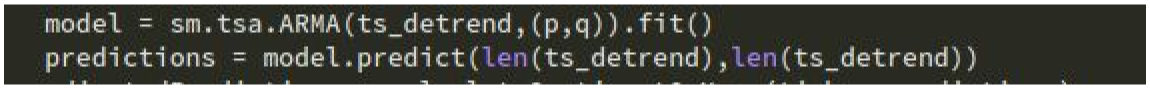


Following is the code for selecting the best possible (p,d,q) values for a given time series:



```
res = sm.tsa.arma_order_select_ic(ts_detrend, ic=['aic', 'bic'],trend='c')
p,q = res.aic_min_order[0],res.aic_min_order[1]
```

For the p,q coefficients selected above, we fit an ARIMA model and predict the opening file for the next day as follows :



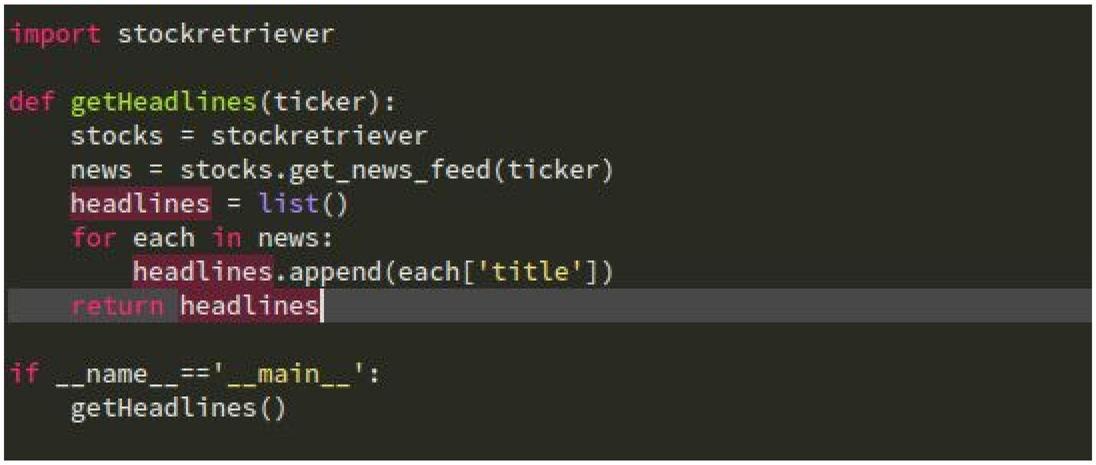
```
model = sm.tsa.ARMA(ts_detrend,(p,q)).fit()
predictions = model.predict(len(ts_detrend),len(ts_detrend))
```

3. Using news headlines to predict possible changes in the stock prices

I extracted the news headlines from the previous days using yahoo_finance api and performed sentiment analysis on them using alchemy.api to gauge whether the stock price is likely to go up or down.

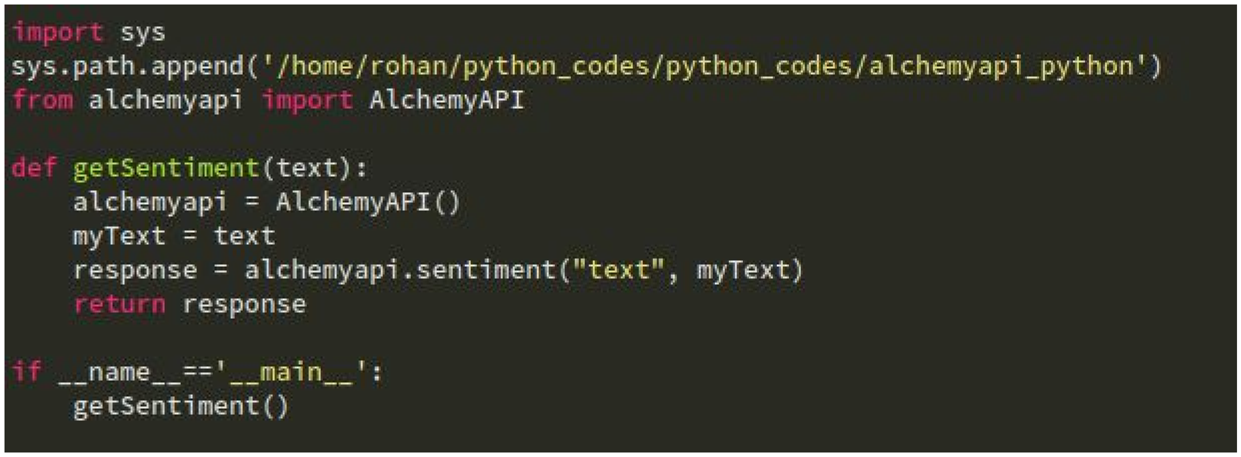
For example, an overall positive score returned by alchemy.api on the news for a particular company most likely means that the stock prices are likely to go up which a negative score is likely to affect the prices adversely.

I used a small open source wrapper over yahoo_finance api to extract news feeds. The following small snippet retrieves latest headlines for a company and forms a list of all news headlines for the company.



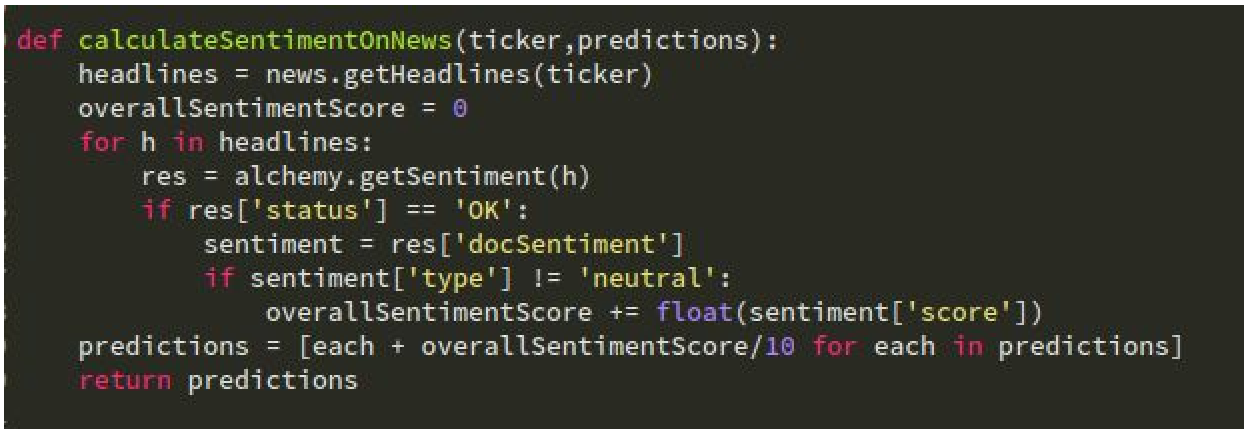
```
import stockretriever

def getHeadlines(ticker):
    stocks = stockretriever
    news = stocks.get_news_feed(ticker)
    headlines = list()
    for each in news:
        headlines.append(each['title'])
    return headlines

if __name__=='__main__':
    getHeadlines()
```

The list is then sent to a code that calls the alchemy.api library methods to get real time sentiments on the headlines .



```python
import sys
sys.path.append('/home/rohan/python_codes/python_codes/alchemyapi_python')
from alchemyapi import AlchemyAPI

def getSentiment(text):
    alchemyapi = AlchemyAPI()
    myText = text
    response = alchemyapi.sentiment("text", myText)
    return response

if __name__=='__main__':
    getSentiment()
```

The following function then calculates an overall sentiment score on the headlines and adjusts the predictions of ARIMA accordingly.

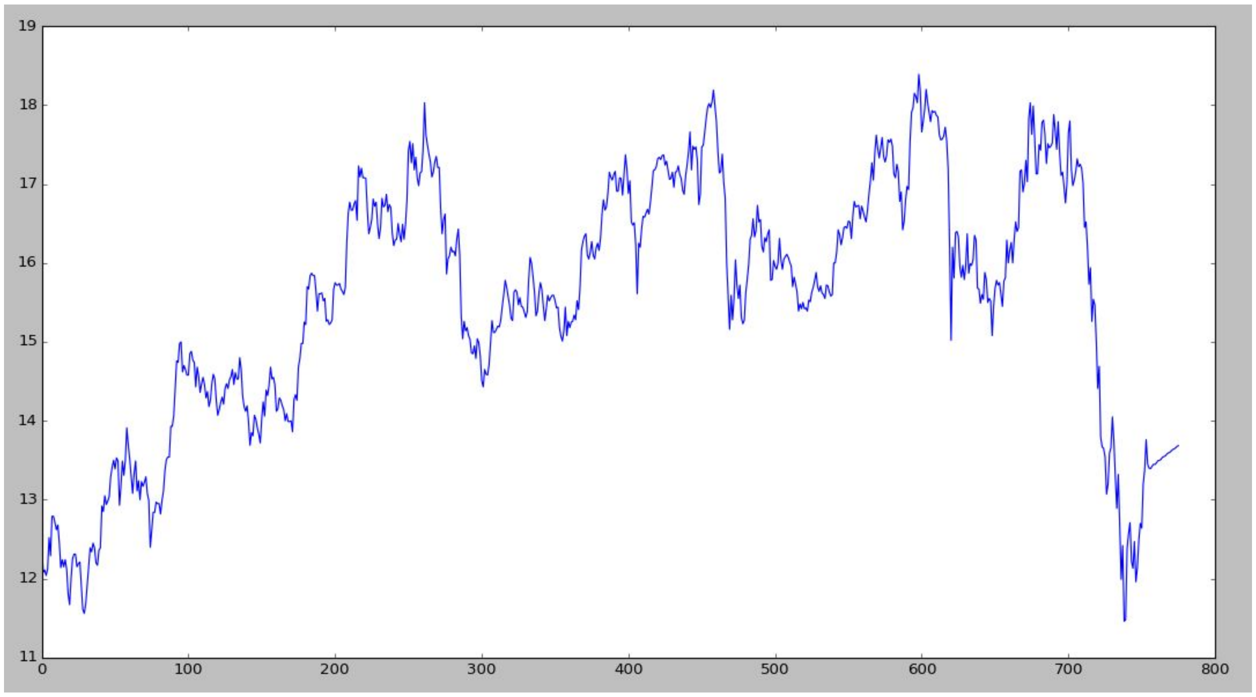


```python
def calculateSentimentOnNews(ticker,predictions):
    headlines = news.getHeadlines(ticker)
    overallSentimentScore = 0
    for h in headlines:
        res = alchemy.getSentiment(h)
        if res['status'] == 'OK':
            sentiment = res['docSentiment']
            if sentiment['type'] != 'neutral':
                overallSentimentScore += float(sentiment['score'])
    predictions = [each + overallSentimentScore/10 for each in predictions]
    return predictions
```

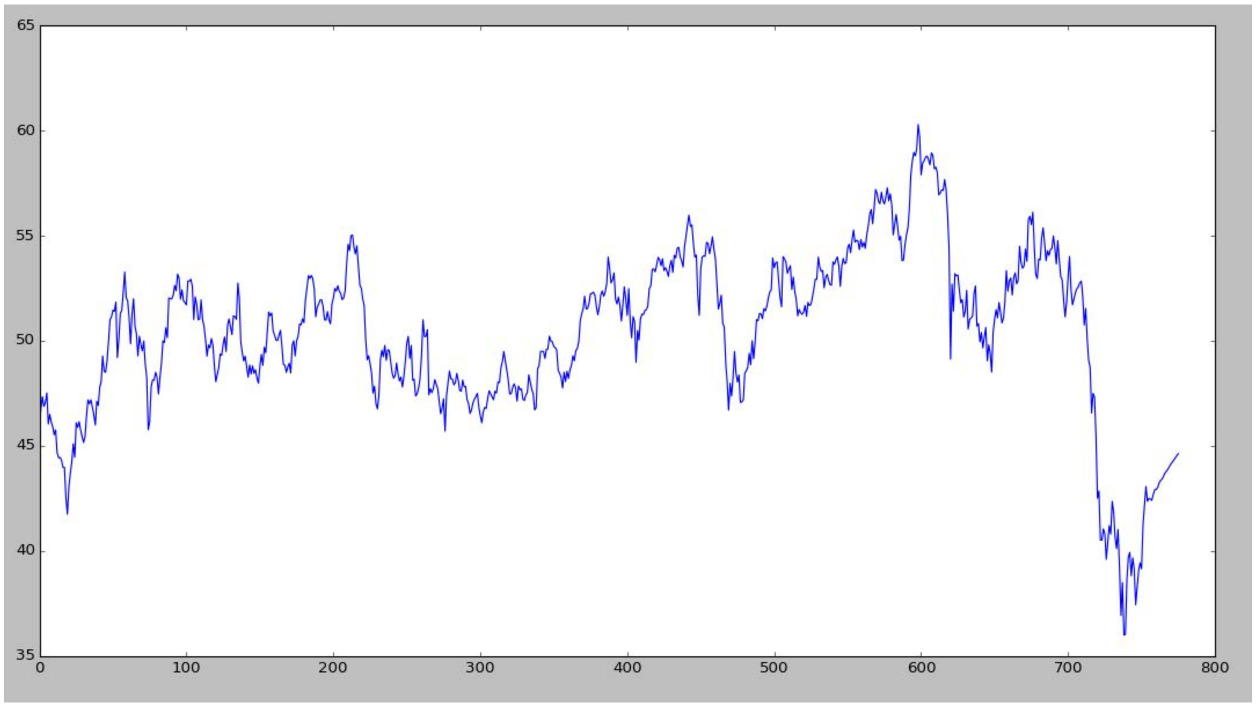
The effect of news and headlines on the stocks is a matter of serious debate in economics. After running the prediction algorithm on various stocks along with sentiment from headlines on various stocks for the last few days, I have come to the conclusion (possibly an assumption!) that the overall sentiment score affects the prediction value positively or negatively by not more than 10% of the score.

Some of the prediction plots can be seen as follows:

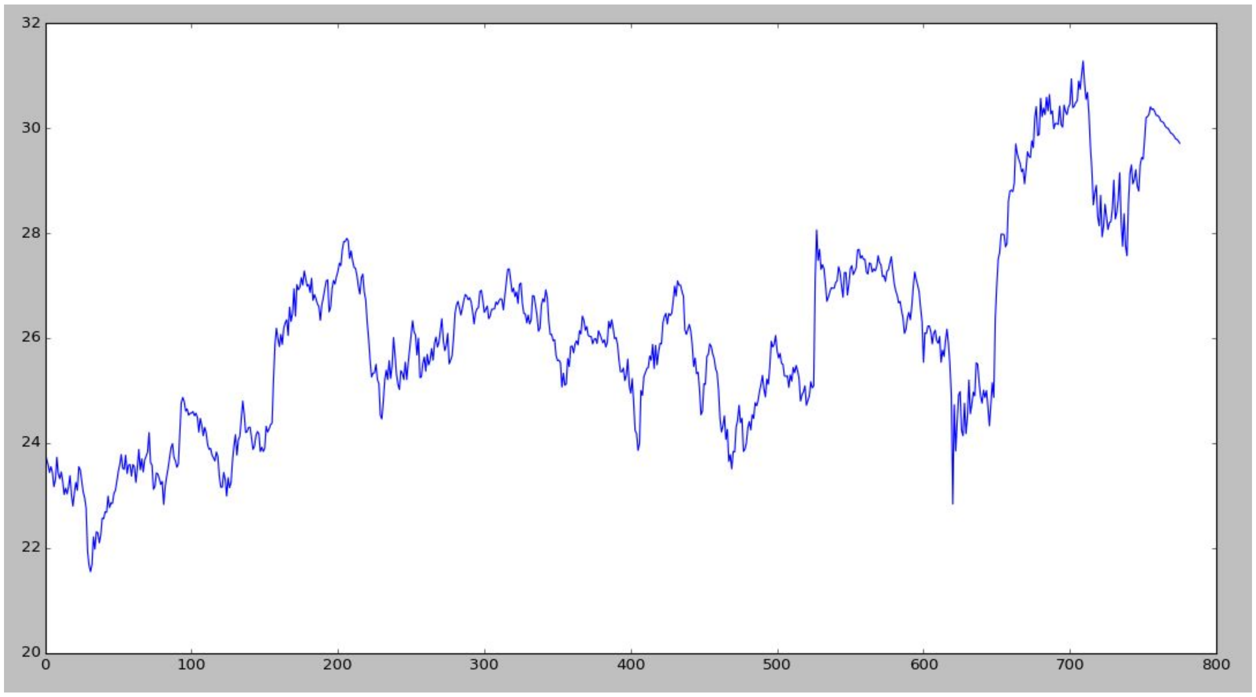
Bank of America : (BAC)



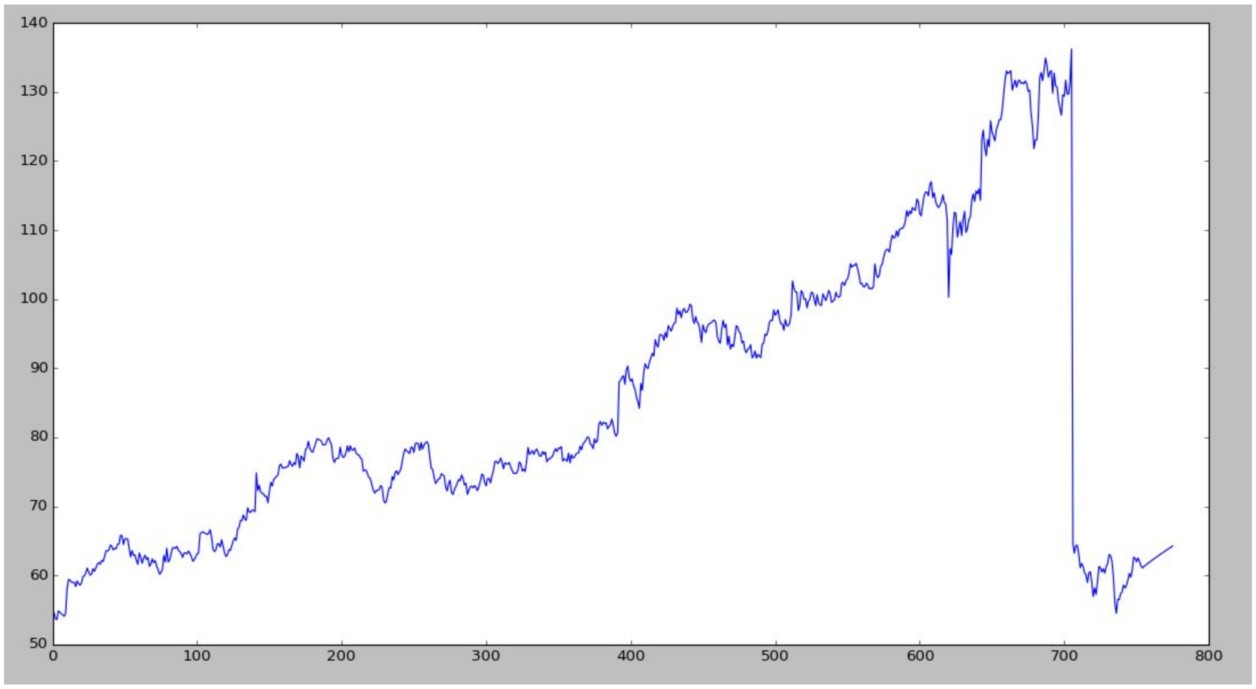
CitiBank : (C)



General Electrics (GE) :



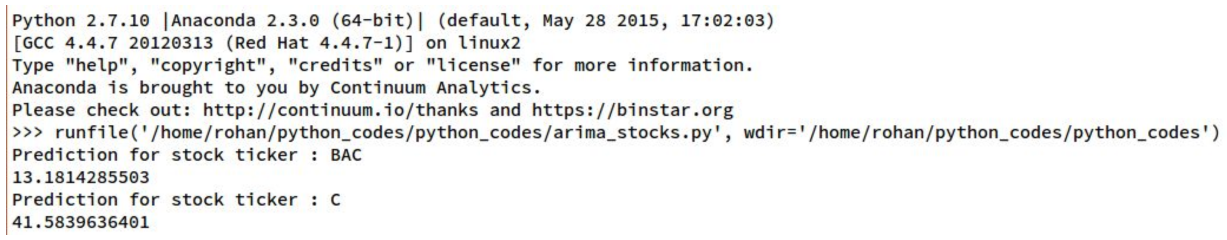
Nike (NKE) :



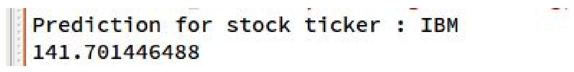
Following are the predictions of the various stock prices based on the ARIMA model for 11th March at 10:00 am

```
BAC = 13.1814285503

C =  41.5839636401

IBM = 141.701446488

AAPL = 102.692440868

GE = 30.2762729387

T = 37.9398334365

MCD = 118.779930537

NKE = 59.0906862258

TWTR = 18.276919312

TSLA = 204.081162818
```

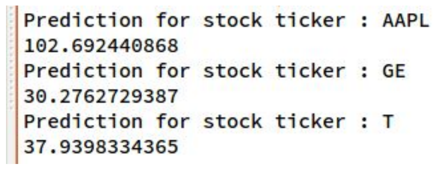
Code Screen Shots:



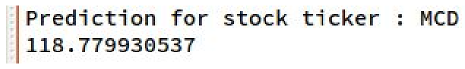
```
Python 2.7.10 |Anaconda 2.3.0 (64-bit)| (default, May 28 2015, 17:02:03)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://binstar.org
>>> runfile('/home/rohan/python_codes/python_codes/arima_stocks.py', wdir='/home/rohan/python_codes/python_codes')
Prediction for stock ticker : BAC
13.1814285503
Prediction for stock ticker : C
41.5839636401
```



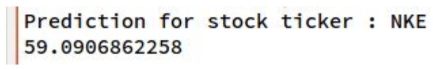
```
Prediction for stock ticker : IBM
141.701446488
```



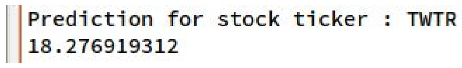
```
Prediction for stock ticker : AAPL
102.692440868
Prediction for stock ticker : GE
30.2762729387
Prediction for stock ticker : T
37.9398334365
```



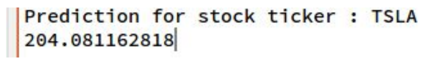
```
Prediction for stock ticker : MCD
118.779930537
```



```
Prediction for stock ticker : NKE
59.0906862258
```



```
Prediction for stock ticker : TWTR
18.276919312
```



```
Prediction for stock ticker : TSLA
204.081162818
```

References :

ARIMA wikipedia
Python Statsmodels
Yahoo Finance api documentation