Stock Price Prediction

Big Data Analytics

Assignment 2

Palash Sushil Matey(pm2824)

Stock Price Prediction

There are a couple of ways to approach Stock Price Prediction. The methods discussed and tried were :

1. Sentiment Analysis
2. Regression using the Gaussian Kernel
3. Weighted Moving Average

Taking into account each of these condition it was interesting to see the stock value predictions. Going through each of these methods we get the following results.

Sentiment Analysis

So this was implemented trying to parse the headline data from the websites and then using sentiment analysis on the extracted text.
The extraction of web pages and relevant links was done using the snowball method used in Assignment 1.

```
#Code credit StackOverflow!
url = 'https://in.finance.yahoo.com/q/h?s=IBM&t=2016-03-10'
data = urllib2.urlopen(url)
soup = BeautifulSoup(data)

divs = soup.find('div',attrs={'id':'yfi_headlines'})
div = divs.find('div',attrs={'class':'bd'})
ul = div.find('ul')
lis = ul.findAll('li')
hls = []
for li in lis:
    headlines = li.find('a').contents[0]
    print headlines
```

Sentiment Analysis was done using Alchemy API
The code is as follows:

```
from alchemyapi import AlchemyAPI

KEY = open('../../api_key.txt','rb').read()[:-1]


def alcObj():
        return AlchemyAPI(KEY)


def sentAn(obj, text):
      response = obj.sentiment("text", text)
      print response
      try:
            score       = response[u'docSentiment'][u'score']
      except:
            score       = 0
      type   = response[u'docSentiment'][u'type']
      return score, type

def testAlc():
      obj = alcObj()
            for test_text in test_text_list:
            score, type = sentAn(obj, test_text)
            print "test_text:\t", test_text
            print "Sentiment is %s %s" %(str(score), str(type))

testAlc()
```

The effort was to get a score and get the average total score. With that information I intended to help classify All the tweets of a particular company as positive, negative or neutral. We get a value, however this value is not enough to get a quantitative value to predict the tweets.
Overall, using sentiment analysis is useful in the long term, in short term stock prediction especially with stable stocks like the ones that were given it does not play a significant role.

<u>Regression using Gaussian Kernel</u>

The idea behind using a Gaussian Kernel is to fit the data around the mean value. It is weighted more towards the value closest to the current value and In this case could help us get a better fit to the data.

```
################################################
#Idea and Code collaboration: Maanit Mehra, Marshall Van Loon, Zoltan
Onodi Scuz,
#                       Anubha Bhargava
#
#
#
######################################
from yahoo_finance import Share
import yahoo_finance as yhf
from datetime import date
import numpy as np
from sklearn import gaussian_process
import datetime
from datetime import dt


# Stock tickers to predict
```

```
stocks = ['BAC', 'C', 'IBM', 'AAPL', 'GE', 'T', 'MCD', 'NKE', 'TWTR',
'TSLA']

# Today String
#TODAY = '20' + date.today().strftime('%y-%m-%d')
TODAY = '2016-3-01'
BEGIN = '2016-2-14'

# Training Model
gp = gaussian_process.GaussianProcess(theta0=1e-2, thetaL=1e-10,
thetaU=10, random_start=100)

# Since we are predicting the stock price change in one hour, only recent
data is needed
stock_prices = {}
stock_predictions = {}
for stock in stocks:

    # Initialize list of share prices
    stock_prices[stock] = []

    # Initialize share
    share = yhf.Share(stock)


    # Reverse the order begin->today
    stock_prices[stock] = stock_prices[stock][::-1]

    # X axis = 0:(days * # of Observations per day)
    L = len(stock_prices[stock])
    X = np.atleast_2d(np.linspace(0, L-1, L)).T
    Y = np.array(stock_prices[stock])
    gp.fit(X, Y)

    stock_pred, sigma2_pred = gp.predict(x, eval_MSE=True)

    Today = float(share.get_price())
    Diff = EndOfNextDay - Today
    pred = Today + Diff/13.0

    current.append(yhf.Share(stock).get_price())
    # Save predictions
    stock_predictions[stock] = pred
    twoalgo.append(pred)
```

This approach while significantly better than most, with the Gaussian kernel taking care of the weighting it predicted value closer to the more recent value, however it still lacked accuracy (Trial and Error).

The stock prediction algorithm is the one that Is given below.

Exponential Moving Average using predictor hints from London Stock Exchange
The idea of using other Stock Markets as predictors appealed to me, it would seem that in most cases stock markets follow each other in terms of trends especially ones concentrated in a part of the world. So I took the FTSE which is the London Stock Exchange top 100 stocks which is open before NASDAQ. I use this as an hint to indicate the trend of the stock market generally. This is used to decide if it is a good idea to marginally increase or marginally decrease my stock value.

I am collecting minute by minute data using google finance API and then using the last twenty mins of trading data on the earlier day to assign some preliminary prediction to my stocks(using the hint from the predictor). This is weighted by a factor of 0.58(Reference found at the bottom of this document). This Is further refined later.

Next part of my code involves using a combination of Moving average and Exponential Moving Average weighted with the Golden Ratio. Initially I was using the Exponential Moving average as it performs the same function as that of Gaussian Kernel. In a similar manner it weighs recent data points more heavily than older points. The golden ratio is often used to predict stock market trends(experimental analysis), some traders are also know to use 75%,25% or 50%,50% approach. This depends on the time and luck mostly. Although there is some interesting theory behind the golden ratio. It is the number to which the Fibonacci series converges. It is basically the ratio of the first number and number that immediately follows. The other side of the weight is found by dividing any number in the series divided by the number two places to its right.

In this code the first ratio is used. Implications of this ratio are quite an interesting read, it seems to be the ratio that describes patterns in life and human behavior. The golden ratio is used to weight the moving average and exponential moving average. The gr weight assigned to the moving average is lower than that assigned to the exponential moving average.

Finally I use both the preliminary analysis value and the exponential moving average value , take their average and arrive at a value it is more stable and weighted. This gives the best prediction accuracy.

```
################################################################
#
#Code collaborations : Anubha Bhargava, Maanit Mehra, Zoltan Onodi Scuz
#                 Ashish Nanda
#Idea Collaborations: Paul Jaquet
#Original Code : Palash Sushil Matey
#
#
#############################################################
import csv
import numpy as np
import requests
import scipy as sp
import sklearn as sl
import math
import pandas as pd
from functools import reduce
import os
import os.path
mport sys
import datetime as dt
import pandas as pd
import pandas.io.data as pio
from pandas import Series
from yahoo_finance import Share
import yahoo_finance as yhf
from datetime import date
import numpy as np
from sklearn import gaussian_process
import datetime

INTERVAL = 60 #sec
GOOGLE_QUOTE_URL =
"http://www.google.com/finance/getprices?f=d,o,h,l,c,v&df=cpct&i="+str(INT
ERVAL)+"&q="
PATH = "./DATA/"
final = []
flag = False
## This function collects detailed data from the Google Finance API
## for each Ticker symbol. It saves this data in a file named after the
company
## input.
def getHistory(symbol,company):
      global final
      flag = True
      filename = PATH + company + ".csv"
      try:
            os.remove(filename)
      except:
            pass

      try:
            file = open (filename, "a+")
      except:
            file = open (filename, "w+")

      stock_url = GOOGLE_QUOTE_URL + symbol
      resp = requests.get(stock_url)
      resp_filter = resp.text.split(" ")
      global flag
      ## From 6 to exclude the textual information, upto -1 to exclude the
last \n
      text_list = resp_filter[0].split("\n")[7:-1]
      file.write("TIME,CLOSE,HIGH,LOW,OPEN,VOLUME\n")
      i = 0
      time1 = []
      open1 = []
      for row in text_list:
            file.write(row+"\n")
            line = row.split(',')
            time1.append(line[0])
            open1.append(line[4])
            #for i in range(0,):
            #     if i%10 == 0:
            #           print (line[i])
#     file.write(resp_filter)
      file.close()
      #print (time1)
      #print (open1)
      last = []
      arr = []
      for t,o in zip(time1,open1):
            try:
                  if float(t)%10 == 0:
                        o = float(o)
                        last.append(o)
            except:
                  pass
      #cprice = last
      #print (arr)

      if flag == True:
            avg = (float(last[-1])-float(last[-2]))*0.56 + float(last[-1])
            avg = round(avg,2)
      #print (avg)
            final.append(avg)
      else:
      avg = -(float(last[-1])-float(last[-2]))*0.56 + float(last[-1])
            avg = round(avg,2)
      #print (avg)
            final.append(avg)

      #print (company+ avg)
      #print (last)
def main():
        reader=csv.DictReader(open('Yahoo_symbols.csv','r'))
        company_list=[]
        sym_list=[]
        for sym in reader:
                company=sym["COMPANY"]
                symbol=sym["SYMBOL"]
                company_list.append(company)
                sym_list.append(symbol)
#        print (sym_list)
        for i in range(0,len(sym_list)):
#          print sym_list[i]
                getHistory(sym_list[i], company_list[i])

main()




onealgo = []
twoalgo = []
current = []

def Predict(company, today):
    global flag
    London = pio.get_data_yahoo(
        '^FTSE',
        start = dt.datetime(2016, 2, 19),
        end = dt.datetime(2016, 3, 10))
    #print (London['Open'])
    London = (London.sub(London['Close'],axis=0))
    #print (London)
    lprice = London['Open']
    l_mat = lprice.as_matrix()
    #print (len(l_mat))
    #print(l_mat.ndim)
    #print ('Printn')
    #print (l_mat)
    flag = True
    df = pio.get_data_yahoo(
        company,
        start = dt.datetime(2015, 1, 1),
        end = dt.datetime(today.year, today.month, today.day))
    df.loc[today] = (Share(company).get_open(),
Share(company).get_days_high(), Share(company).get_days_low(),
Share(company).get_price(), Share(company).get_volume(),
float(Share(company).get_price()))
    closing = df['Adj Close']
    moving_avg = pd.rolling_mean(closing, 10)
    ewma = pd.ewma(closing, span = 3)
    onealgo.append(str(getPrice(closing,moving_avg,ewma)))
    #print ("\n" + "Estimate Price of " + company + " @ " + str(today +
dt.timedelta(days=1)) + ": ")
    #print ("  " + str(predictPrice(closing, moving_avg, ewma)))
```

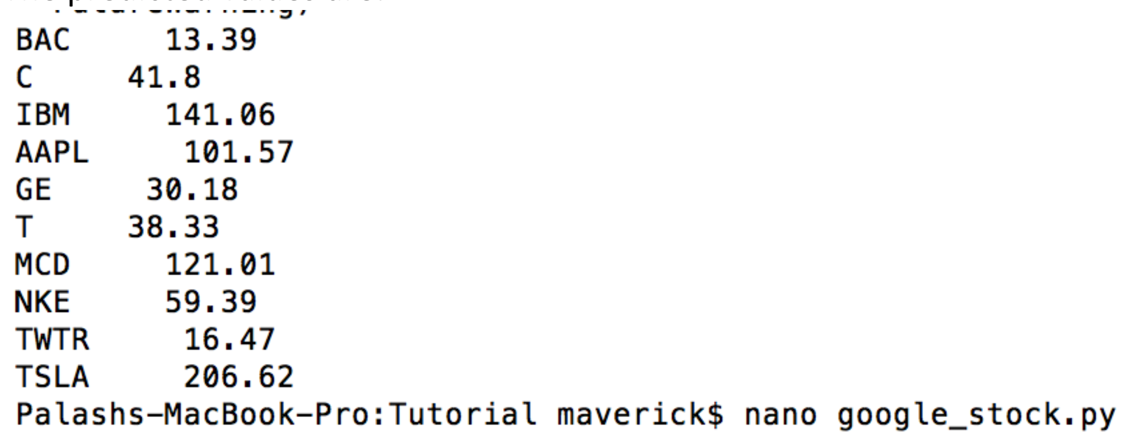
```
##########################################################################
###############################################################
###Formula online, paper link attached in the References.
##########################################################################
###############################################################
def getPrice(closing, moving_average, ewma):

    gr = (1 - 5 ** 0.5) / 2
    return ((ewma[-1] - ewma[-2]) * (gr)
            + (moving_average[-1] - moving_average[-2]) * (1 - gr)
            + closing[-1])

if __name__ == '__main__':
    stocks = ['BAC', 'C', 'IBM', 'AAPL', 'GE', 'T', 'MCD', 'NKE', 'TWTR',
'TSLA']
    today = dt.date.today() - dt.timedelta(days=0)
    #today = dt.datetime(2016,3,9)
    for stock in stocks:
        Predict(stock, today)
avg = 0.0
list1 = []
for i,n in zip(onealgo,final):
      i = float(i)
      n = float(n)
      i = round(i,2)
      #print (i,n)
      avg = (i+n)/2
      avg = round(avg,2)
      list1.append(avg)

#print (list1)
for i,n in zip(stocks,list1):
      print (str(i)+'     '+ str(n))
```

The predicted values are:



```
BAC     13.39
C     41.8
IBM     141.06
AAPL     101.57
GE     30.18
T     38.33
MCD     121.01
NKE     59.39
TWTR     16.47
TSLA     206.62
Palashs-MacBook-Pro:Tutorial maverick$ nano google_stock.py
```

References:
[1] http://www.goldennumber.net/fibonacci-stock-market-analysis/
[2] http://www.scientificpapers.org/wp-content/files/1134_How_to_use_Fibonacci_retracement_to_predict_forex_market.pdf
[3] http://www.investopedia.com/ask/answers/072715/how-are-double-exponential-moving-averages-applied-technical-analysis.asp