Stock Advisor RPI

Embedded Project Report Submitted To: Dr. Imran Amin

1212106

1212132

1212102

1212139

1212118

Anwer Amin Uzair Abdullah Akhtar Edhi Daniyal Idris Motiwala Muhammad Bilal

4/26/2016

# Introduction

The project will be an always-on raspberry pi based solution designed to provide a low cost deployment of a personal stock investment advisor. The solution will get publicly available stock data which will be passed thought stock analysis algorithms to providing notifications to a user on potentially profitable future investments. The solution will build its own dataset by scrapping data from public website belonging to the Pakistan Stock Exchange. Standard stock analysis algorithms will be used to analyses the data for and generate “advice” notifications delivered to the users via a SMS. The project will use Python for web scrapping as well as data analysis. The web application and SMS notifications will be made using complementing technologies.

# Features

* Low cost automated solution for stock data analysis and investment advice.
* All data extracted from public domain websites without buying access to expensive stock exchange databases or APIs.
* Delivers notifications using simple web application and SMS.

# Implementation

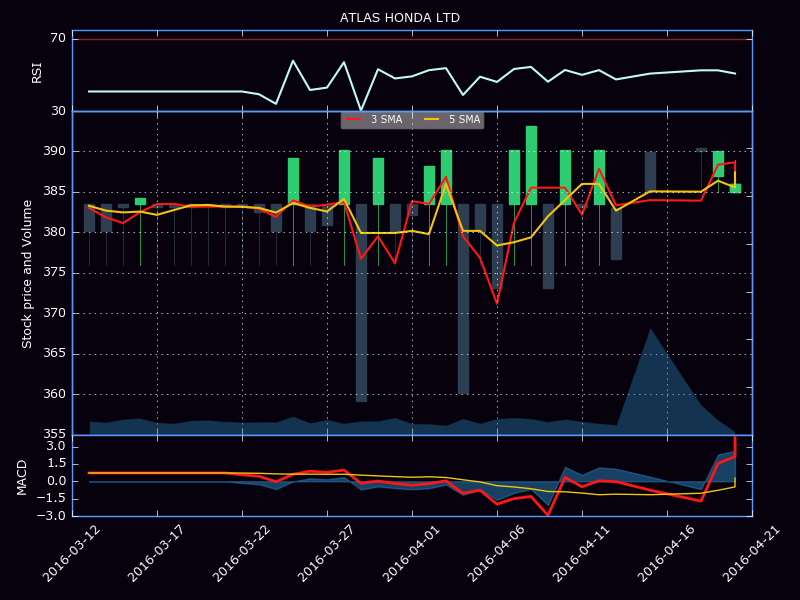
The following section defines the various components of the project

## Data Gathering

The first component required to build the solution was a way of gathering stock data, traditionally such systems use online databases or APIs in order to get this day, however to keep the system “free”, the project has its own automated data gathering mechanism. The component uses Selenium web drivers and Python’s BeautifulSoup4 library to extract data from the Pakistan Stock Exchange website storing it in an online MongoDB database. The script uses a scheduling system running automatically at 1:00 pm on every weekday to extract that day’s data.

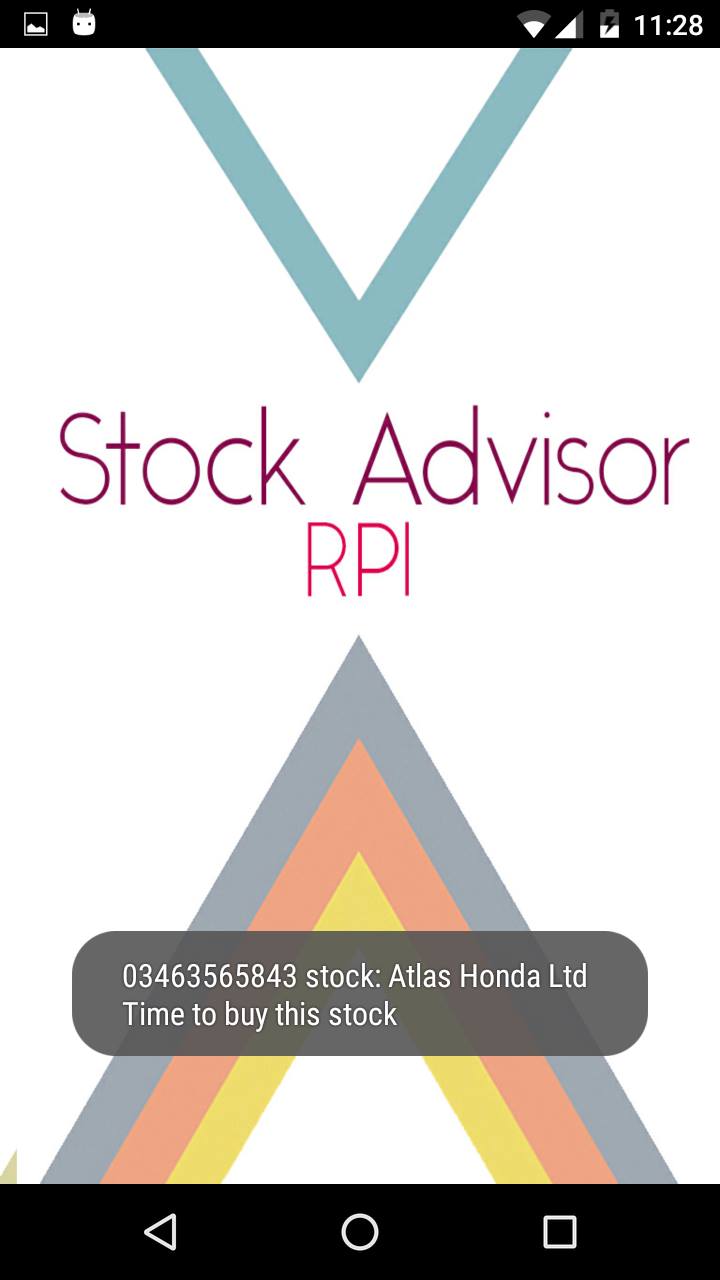
## Data Analysis

The second subsystem is responsible for 2 major tasks, built using the Numpy library, the forex\_main.py and forex\_worker.py modules. The forex\_main is responsible for inserting sms notifications to the mongo db collection by analyzing the scrapped data. The content of message will be advice to user what to do with the individual stock. The forex\_worker code is responsible for generating graphical user interface in the form of graph for individually selected stock. The graph includes MACD, EMS, SMA and other stock related functions. The generation of the graph is based on the pre-scrapped data of 26 days of individual stock.



## SMS Gateway

The third part of our project is sending the notification to user; this part was handled by android application which uses a REST API to fetch data from mongodb. After fetching the records whose sent flag is false, the application use mobile sms service to send notifications to the provided cell number. After sending the messages the application updates the “is\_sended” flag to true.



# Conclusion

The proposed project was a small personal stock advisor which can run on an always-on raspberry pi, the final project achieved this goal not only as a single user application but the code allows multiple device to run the program at the same time. The frame work of analysis allows for additions of more advanced algorithms without any change to the existing data structure due to the schema’s flexible nature.

# Code

Forex\_main.py

1. import numpy as np
2. import mongo\_data as mdata
3. import forexCal as fx
4. import schedule
5. import time
6. import easygui
7. def forexRoutine():
8. pnumber = ""
9. pnumber = easygui.enterbox(msg='Enter phone number:', title='Forex plotter ', default='03463565843', strip=True)
10. print("Fetching stock data")
11. namelist = mdata.getstockNameList()
12. counter = 0
13. for stockname in namelist:
14. if counter == 5:
15. break
16. closep = mdata.getClosingRates(stockname)
17. if len(closep) < 26:
18. continue
19. MA5 = fx.movingaverage(closep,5)
20. MA10 = fx.movingaverage(closep,10)
21. EMA7 = fx.ExpMovingAverage(closep,7)
22. EMA10 = fx.ExpMovingAverage(closep,10)
23. rsi = fx.rsiFunc(closep)
24. macd = fx.computeMACD(closep)
25. print('Stock Name: ' + stockname)
26. print ('MA 5:')
27. print(MA5)
28. print ('MA 10:')
29. print(MA10)
30. print('EMA 7:')
31. print(EMA7)
32. print('EMA 10:')
33. print(EMA10)
34. print('RSI:')
35. print(rsi)
36. messagestr = "stock: "+ stockname
37. nl = "\n"
38. # RSI calculation
39. rsiarray = rsi.tolist()
40. if rsiarray[len(rsiarray) -1] > 70:
41. messagestr = messagestr + nl + "stock is Overbought"
42. print('stock is Overbought')
43. if rsiarray[len(rsiarray) -1] < 30 :
44. messagestr = messagestr + nl + "stock is Oversold"
45. print('stock is Oversold')
46. print('MACD:')
47. print(macd)
48. macdres = macd[2]
49. if macdres[len(macdres)-1] > 0:
50. #time to sell bullish crossover
51. messagestr = messagestr + nl + "Time to buy this stock"
52. print("Time to buy this stock")
53. if macdres[len(macdres)-1] < 0:
54. #time to buy bearish crossover
55. messagestr = messagestr + nl + "Time to sell this stock"
56. print("Time to sell this stock")
57. if messagestr == "stock: "+ stockname:
58. print("no noticeable trend change")
59. else:
60. #favorable tread did appear
61. mdata.createSMS(messagestr,pnumber)
63. # counter+=1
64. schedule.every().day.at("14:30").do(forexRoutine) #2:30 pm
65. schedule.every(24).hours.do(forexRoutine)
66. while True:
67. try:
68. schedule.run\_pending()
69. time.sleep(1)
70. except Exception as e:
71. print ("Exception occured: ", e)
72. forexRoutine()

### forex\_Cal.py

1. import numpy as np
2. import mongo\_data as mdata
3. def movingaverage(values,window):
4. weigths = np.repeat(1.0, window)/window # to save up on processing power.
5. smas = np.convolve(values, weigths, 'valid') # only run on valid points
6. return smas # as a numpy array
7. def ExpMovingAverage(values, window):
8. weights = np.exp(np.linspace(-1., 0., window))
9. weights /= weights.sum()
10. a = np.convolve(values, weights, mode='full')[:len(values)]
11. a[:window] = a[window]
12. return a
13. def rsiFunc(prices, n=14):
14. deltas = np.diff(prices)
15. seed = deltas[:n+1]
16. up = seed[seed>=0].sum()/n
17. down = -seed[seed<0].sum()/n
18. rs = up/down
19. rsi = np.zeros\_like(prices)
20. rsi[:n] = 100. - 100./(1.+rs)
21. for i in range(n, len(prices)):
22. delta = deltas[i-1] # cause the diff is 1 shorter
23. if delta>0:
24. upval = delta
25. downval = 0.
26. else:
27. upval = 0.
28. downval = -delta
29. up = (up\*(n-1) + upval)/n
30. down = (down\*(n-1) + downval)/n
31. rs = up/down
32. rsi[i] = 100. - 100./(1.+rs)
33. return rsi
34. def computeMACD(x, slow=26, fast=12):
35. emaslow = ExpMovingAverage(x, slow)
36. emafast = ExpMovingAverage(x, fast)
37. return emaslow, emafast, emafast - emaslow # tuple

### Scrapping\_sch.py

1. from bs4 import BeautifulSoup
2. from selenium import webdriver
3. import selenium
4. import time
5. import pymongo
6. import datetime
7. import schedule
8. def scraping():
9. d = datetime.datetime.now()
10. # correction later 9,17
11. if d.hour in range(9,17) and d.isoweekday:
12. # logger.info('STARING SCRAPING')
13. driver = webdriver.Firefox()
14. driver.get('http://www.psx.com.pk/')
15. driver.implicitly\_wait(10)
16. WebElement = driver.find\_element\_by\_name('mainFrame')
17. print(WebElement.text)
18. driver.switch\_to.frame(WebElement)
19. button = driver.find\_element\_by\_xpath('//\*[@id="Head"]/table/tbody/tr[3]/td[2]/div/table/tbody/tr/td[1]/div/a[3]').click()
20. driver.implicitly\_wait(30)
21. table1 = driver.find\_element\_by\_xpath('//\*[@id="contentdiv1"]/table/tbody/tr[2]/td[1]/table[2]/tbody/tr[4]/td')
22. soup = BeautifulSoup(table1.get\_attribute('outerHTML'))
23. driver.implicitly\_wait(10)
24. stockCatagory = ""
25. marketSummary = []
26. rowlist = []
27. title\_arry = ["SYMBOL" ,"LDCP", "OPEN" ,"HIGH" ,"LOW", "CURRENT", "CHANGE", "VOLUME"]
28. tables = soup.find\_all('tbody')
29. for tbody in tables:
30. rows = tbody.find\_all('tr')
31. stockCatagory = rows[0::]
32. for tr in rows[2::]:
33. rowsdata = tr.find\_all('td')
34. del rowlist
35. rowlist = []
36. json\_obj = {}
37. counter = 0
38. for data in rowsdata:
39. #print(data.text)
40. json\_obj[title\_arry[counter]] = data.text
41. counter += 1
42. marketSummary.append(json\_obj)
43. #print(marketSummary)
44. # logger.info('DATA RETRIVED')
45. marketsummary = marketSummary
46. client = pymongo.MongoClient('mongodb://stockadvisor:anwer123@ds015919.mlab.com:15919/stockadvisordb?authMechanism=SCRAM-SHA-1')
47. #connect to the rightDB in the cluster
48. db = client['stockadvisordb']
49. #the stocks object to be inserted
50. post = {
51. "stocks": marketsummary,
52. "date": datetime.datetime.now()}
53. #get table name, this is only for simplfication for reuse
54. stocks = db.stocks
55. #perfrom insert operations
56. stock\_id = stocks.insert\_one(post).inserted\_id
57. print(stock\_id)
58. print(stock\_id)
59. driver.quit()
60. return 1
61. else:
62. print ("job pendding process in correct interval")
63. schedule.every().day.at("13:00").do(scraping) # 01:00 pm
64. while True:
65. try:
66. # print("Wait mode")
67. # schedule.run\_pending()
68. time.sleep(1)
69. except Exception as e:
70. print ("Exception occured: ", e)

### SMSMONGODb.cs

1. using MongoDB.Bson;
2. using MongoDB.Driver;
3. using System;
4. using System.Collections.Generic;
5. using System.Linq;
6. using System.Text;
7. using System.Threading.Tasks;
8. namespace mongoDB
9. {
10. class Program
11. {
12. static void Main(string[] args)
13. {
14. var notification = new Notification();
15. List<Notification> notifications = notification.notificationList();
16. ObjectId Id = notifications.First().Id;
17. Notification.sent\_message(Id).Wait();
18. }
19. public class Notification
20. {
21. public ObjectId Id { get; set; }
22. public string body { get; set; }
23. public bool sent\_flag { get; set; }
24. public DateTime date { get; set; }
25. public List<Notification> notificationList() {
26. var client = new MongoClient("mongodb://stockadvisor:anwer123@ds015919.mlab.com:15919/stockadvisordb?authMechanism=SCRAM-SHA-1");
27. var db = client.GetDatabase("stockadvisordb");
28. var col = db.GetCollection<Notification>("notifications");
29. var notifications = col
30. .Find(notification => notification.sent\_flag == false)
31. .ToListAsync()
32. .Result;
33. foreach (var notification in notifications)
34. {
35. Console.WriteLine("{");
36. Console.WriteLine(notification.Id);
37. Console.WriteLine(notification.body);
38. Console.WriteLine(notification.date);
39. Console.WriteLine("}");
40. }
41. return notifications;
42. }
43. public static async Task sent\_message(ObjectId Id)
44. {
45. var client = new MongoClient("mongodb://stockadvisor:anwer123@ds015919.mlab.com:15919/stockadvisordb?authMechanism=SCRAM-SHA-1");
46. var db = client.GetDatabase("stockadvisordb");
47. var col = db.GetCollection<Notification>("notifications");
48. var updateFilter = Builders<Notification>.Filter.Eq("\_id", Id);
49. var update = Builders<Notification>.Update.Set("sent\_flag", true);
50. await col.UpdateOneAsync(updateFilter, update);
51. }
52. }
53. }
54. }