Real-time Stock Data Analysis Using Apache Spark Streaming

Predicting the stock market is, without a doubt, one of the most challenging tasks in the finance industry. It is difficult to keep track of the market as many variables play an essential role in controlling it.

Stock data analysis is used by investors and traders to make critical decisions related to the stocks. Investors and traders study and evaluate past and current stock data and attempt to gain an edge in the market by making decisions based on the insights obtained through the analyses.

Suppose you are working in an angel broking company. You have been provided real-time global equity data. The data contains the following information:

symbol - id of the stock

timestamp - time at which we are getting the data

open - the price at which the particular stock opens in the time period

high - highest price of the stock during the time period

low - lowest price of the stock during the time period

close - the price at which the particular stock closes in the time period

volume - indicates the total number of transactions involving the given stock in the time period

Based on the data, you need to perform some real-time analyses to generate insights that can be used to make informed decisions.

How to get the data?

To get the data, we will make use of an API provided by Alpha Vantage. The API returns intraday time series (timestamp, open, high, low, close, volume) of the equity specified in JSON format.

API:

https://www.alphavantage.co/query?function=TIME_SERIES_INTRADAY&symbol=MSFT&interval=1min&apikey=demo

API Parameters:

- function - the time series of your choice. In this case, function=TIME_SERIES_INTRADAY
- symbol - the name of the equity of your choice. For example, here symbol=MSFT (Microsoft)
- interval - time interval between two consecutive data points in the time series. The following values are supported: 1min, 5min, 15min, 30min, 60min. Here, interval=1min
- apikey - your API key. Follow the steps given here to get your key
- outputsize - by default, outputsize=compact. Strings compact and full are accepted with the following specifications: compact returns only the latest 100 data points in the intraday time series; full returns the full-length intraday time series. This parameter is optional.

Response:


```
{
   "Meta Data": {
      "1. Information": "Intraday (1min) open, high, low, close prices and volume",
      "2. Symbol": "MSFT",
      "3. Last Refreshed": "2018-08-24 15:59:00",
      "4. Interval": "1min",
      "5. Output Size": "Compact",
      "6. Time Zone": "US/Eastern"
   },
   "Time Series (1min)": {
      "2018-08-24 15:59:00": {
         "1. open": "108.3700",
         "2. high": "108.4700",
         "3. low": "108.3400",
         "4. close": "108.4200",
         "5. volume": "491979"
      },
      "2018-08-24 15:58:00": {
         "1. open": "108.4500",
         "2. high": "108.4600",
         "3. low": "108.3500",
         "4. close": "108.3600",
         "5. volume": "238561"
      },

     ...

   }

}
```

As you can see, once you hit the API you will get the data as shown above. The above API hit uses a demo API key. You have to get your own API key by following the steps given here. Get your key as this will be used in the python script to fetch the data ahead.

Problem Statement

Once the NYSE opens, a script should run to fetch the data relating to the stocks every minute inside a folder. The script will make use of the API provided by Alpha Vantage as described above. In parallel, a Spark application should run to stream data from the folder every minute and then perform the analyses on the data. The results of the analyses should be written in an output file. These results will act as insights to make informed decisions related to the stocks.

Let's now look into the specifics of the problem statement:

Fetch data every minute relating to the following four stocks:
Facebook (FB)
Google (GOOGL)
Microsoft (MSFT)
Adobe (ADBE)

It can be achieved by using the python script that hits the intraday API for each stock every minute. We will provide you with a python script and the steps to run in the resources section ahead. On hitting the API for each stock every minute, the script will get the response corresponding to each stock every minute. From the response received, it will extract the latest minute data and will dump that data for each stock every minute in a new file inside a folder. A sample file is provided.

Each minute a file like this will be generated inside the folder.

Note: The above stocks are listed on NYSE (New York Stock Exchange) located in EDT timezone. Daily opening hours for NYSE are from 09:30 AM to 04:00 PM GMT(-4) and it remains closed on Saturday and Sunday. While running the script, keep note of the timezone and the opening hours to get the data. Run the script during the time NYSE remains open. Once the exchange closes, you will not get the real-time data. It will remain open from 7:00 PM IST to 1:30 AM IST (IST is 9.5 hours ahead of EDT).

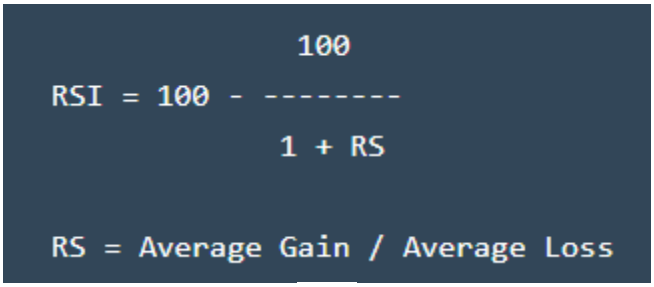
- Read the data file generated inside the folder every minute and convert the data into DStreams in Spark
- Using Spark Streaming, perform the following analyses:

1. Calculate the simple moving average closing price of the four stocks in a 5-minute sliding window for the last 10 minutes. Closing prices are used mostly by

the traders and investors as it reflects the price at which the market finally settles down. The SMA (Simple Moving Average) is a parameter used to find the average stock price over a certain period based on a set of parameters. The simple moving average is calculated by adding a stock's prices over a certain period and dividing the sum by the total number of periods. The simple moving average can be used to identify buying and selling opportunities

2. Find the stock out of the four stocks giving maximum profit (average closing price - average opening price) in a 5-minute sliding window for the last 10 minutes

3. Find out the Relative Strength Index or RSI of the four stocks in a 1-minute sliding window for the last 10 minutes. RSI is considered overbought when above 70 and oversold when below 30. The formula to calculate the RSI is as follows:



RSI

To simplify the calculation explanation, RSI has been broken down into its basic components: RS, Average Gain and Average Loss. This RSI calculation is based on 14 periods, which is the default suggested by Wilder in his book. Losses are expressed as positive values, not negative values.

The very first calculations for average gain and average loss are simple 14-period averages.

First Average Gain = Sum of Gains over the past 14 periods / 14. First Average Loss = Sum of Losses over the past 14 periods / 14

The second, and subsequent, calculations are based on the prior averages and the current gain loss:

Average Gain = [(previous Average Gain) x 13 + current Gain] / 14. Average Loss = [(previous Average Loss) x 13 + current Loss] / 14.

Taking the prior value plus the current value is a smoothing technique similar to that used in calculating an exponential moving average. Here, in our case, we will use 10-period averages (not 14).

4. Calculate the trading volume of the four stocks every 10 minutes and decide which stock to purchase out of the four stocks. Volume plays a very important role in technical analysis as it helps us to confirm trends and patterns. You can think of volumes as a means to gain insights into how other participants perceive the market. Volumes are an indicator of how many stocks are bought and sold over a given period of time. Higher the volume, more likely the stock will be bought

- Build the Spark application using Maven
- Generate a fat jar/s as required corresponding to the Spark application code to generate DStreams and perform the analyses

- Run the python script and the Spark application jar/s and store the results of the analyses in an output file. The results should correspond to data for at least 30 minutes
- Write the logic in simple words for the Spark Application code developed and how to run it and the jar/s in a document

References

https://zerodha.com/
https://www.investopedia.com/