2/19/2019

Stock Data Analysis Logic & Steps to Execute the Jar

Case Study-1

Name:Ruksana Bhanu

STEPS TO EXECUTE THE JAR

COMMAND TO EXECUTE THE JAR FILE:

Step1: To maintain the naming convention provided by the Upgrad, the jar generated from the build "StockDataAnalysis-0.0.1-SNAPSHOT" is renamed to SparkApplication

Step2: Upon renaming the jar, execute the below command. Ensure the paths exists

java -jar SparkApplication.jar C:\\\\Users\\\\Public\\\\JSON C:\\\\Users\\\\Public\\\\output 1>> LogInfo.txt 2>> ConsoleOutput.txt

The Above Command will generate two logs. One is log.info related logs and the other is console output. Single PDF version of these files is submitted as Output.pdf

Step3: Right after running the above jar, in the other terminal execute the python script to have the stream of data.

Command: python getdata.py

Note:

1. Please pass the arguments in the above mentioned format only.(take care of slashes)
2. The **SparkApplication.zip** provided will have the code and the jar files that are generated from the maven build (Without rename). Whereas SparkApplication.jar is the file name renamed version of build generated by version. (Original name was StockDataAnalysis-0.0.1-SNAPSHOT.jar)

LOGIC FOR THE ANALYSES

ANALYSES1: SIMPLE MOVING AVERAGE:

PROBLEM STATEMENT:

Calculate the simple moving average closing price of the four stocks in a 5-minute sliding window for the last 10 minutes.

SOLUTION APPROACH:

1. Input files are converted into DStreams of type **EquityData** using json parsing
2. The DStreams of type **EquityData** are transformed to mapToPair which return the DStream of stockname and corresponding closing price in the below format. The attribute used is close from EquityData object.
   ("stocksymbolname", Tuple<1000.00, 1>)
3. Perform ReduceByKey Operation on above DStream which returns the below Format
   ("stocksymbolname", Tuple<1000.00, 4>)
   1000.00 is the sum of elements and 4 is the number of elements.
4. Perform MapToPair on above DStream to calculate the average.
   Tuple2 of above DStream holds sum and no.of elements. So avg = sum/no.of elements.
   Hence the resultant of this computation would be in below format.
   (MSFT,104.89244)
   (FB,146.77)
   (GOOGL,1077.50734)
   (ADBE,241.70914)

ANALYSES2: MAXIMUM PROFIT

PROBLEM STATEMENT:

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.

SOLUTION APPROACH:

1. Input files are converted into DStreams of type **EquityData** using json parsing
2. The DStreams of type **EquityData** are transformed to mapToPair which return the DStream of stockname and corresponding closing price in the below format. The attribute used is close from EquityData object.
   ("Stocksymbolname", Tuple<1000.00, 1>)
3. Perform ReduceByKey Operation on above DStream which returns the sum of the closing prices in the below format
   ("Stocksymbolname", Tuple<1000.00, 4>)
   1000.00 is the sum of elements and 4 is the number of elements.
4. Calculate average of the closing prices using above DStream which returns below format
   ("Stocksymbolname", Tuple<300.00, 1>)
5. Likewise perform the same steps to get the average of opening price also.
6. Now join the two DStreams avrageofclosingPrice and averageofopeningPrice.
7. Calculate profit by subtracting the AvgOfClosingstockprice-AvgOfOpeniningstockprice
   This would result in below format.
   ("stocksymbolname1", 100.0)
   ("stocksymbolname2", 200.0)
   ("stocksymbolname3", 300.0)
   ("stocksymbolname4", 400.0)
8. Swap the key pairs and perform sortByKey Operation and return first tuple for the output.

ANALYSES3: RELATIVE STRENGTH INDEX

PROBLEM STATEMENT:

Find out the Relative Strength Index or RSI of the four stocks in a 1-minute sliding window for the last 10 minutes.

SOLUTION APPROACH:

1. Input files are converted into DStreams of type **EquityData** using json parsing
2. The DStreams of type **EquityData** are transformed to mapToPair which return the DStream of stockname and corresponding profit/loss by subtracting open-close. Open is nothing but previous days closing price. Therefore, the calculation is previous day closing price-current day closing price.The attribute used is close and open from EquityData object. The Tuple may have positive values or negative Values. Positive implies gain and Negative implies Loss
   ("Stocksymbolname", Tuple<1000.00>)
3. Perform Filter and reduce Operation on above DStream to get the sum of gain stream.
4. Perform Filter and reduce Operation on above DStream to get the sum of loss stream.
5. Join above two DStreams.
6. Now calculate the RSI on the above DStream.
   For the first time, average gain = sum of gains/period
   Average loss= sum of loss/period
   The sum of gains/loss can be attained from the DStream obtained in point5.
   Period is initially set to 10
7. These calculated average gain /loss are stored in a hashmap corresponding to stockname. A hashmap with key as stockname and value as an object. This object will hold the values of average gain/loss calculated
8. From the second and subsequent steps, period value gets decremented.
   If the hashmap contains the details for the stock, then it fetches the previous details from that hashmap and calculates the RSI with below formula

avgGain = ((AvgGainPrevious() * period) + avgGainCurrent) / 10;

avgLoss= ((AvgLossPrevious() * period) + avgLossCurrent) / 10;

AvgGainPrevious and AvgLossPrevious can be fetched from the hashmap for that stockname.

9. After this calculation hasmap is added with latest values of avg gain and loss for that stockname. This way hashmap will hold the recently calculated previous price details which can be used in calculating further computations.

ANALYSES4: TRADING VOLUME

PROBLEM STATEMENT:

Calculate the trading volume of the four stocks every 10 minutes and decide which stock to purchase out of the four stocks..

SOLUTION APPROACH:

1. Input files are converted into DStreams of type **EquityData** using json parsing
2. The DStreams of type **EquityData** are transformed to mapToPair which return the DStream of stockname and corresponding volume in the below format. The attribute used is close from EquityData object.
   ("Stocksymbolname", 5000.00)
3. Perform ReduceByKey Operation on above DStream which returns the sum of the volumes in the below format
   ("stocksymbolname1", 100.0)
   ("stocksymbolname2", 200.0)
   ("stocksymbolname3", 300.0)
4. Swap the key pairs and perform sortByKey Operation and return first tuple for the output.