Moving Average:

1. [30 pts] Set up the stream to feed data into a pyspark DStream.

***Terminal sender: python dj30\_feeder.py | nc -lk 9999***

***Terminal listener: pyspark***

***Terminal listener: from dj30\_receiver import \****

***Terminal listener: setup\_stream()***

***Terminal listener:launch\_stream(10)***

***Code for dj30\_receiver:***

def setup\_stream():

    # Create a DStream that will connect to hostname:port, like localhost:9999

    lines = ssc.socketTextStream("localhost", 9999)

    # Split each line into words

    words = lines.flatMap(lambda line: line.split(" "))

    words.pprint()

1. [30 pts] Use DStream windowing to accumulate two DStreams: dj30sum and dj30ct, respectively the sum and count of prices.

    data = float(row['Close'])

    price\_data = np.append(price\_data, data)

    dj30sum += float(data)

    counter += 1

Pyspark code is the same as question 1

1. [30 pts] Divide these two DStreams (**dj30sum and dj30ct**) to produce dj30avg, thus creating moving average DStreams for both 10-day MA and 40-day MA .

***I used the np array matrix to compute the MA***

***I was stuck on the DSwindowing so used np array, hopefully it is worth for partial credits***

    data = float(row['Close'])

    price\_data = np.append(price\_data, data)

    dj30sum += float(data)

    counter += 1

    date = row['Long Date']

    # print(i, dj30sum, flush = True)

    if len(price\_data) >= 10:

        temp = np.convolve(price\_data, np.ones(10), 'valid')/10

        print('10daysMA: ', temp[-1])

    if len(price\_data) >= 40:

        temp = np.convolve(price\_data, np.ones(40), 'valid')/40

        print('40daysMA: ', temp[-1])

Pyspark code is the same as question 1

1. [60 pts]. Compare the two moving averages (shorter-term MA and the longer MA) to indicate buy and sell signals.

if len(price\_data) > 40:

        ten\_MA =  np.convolve(price\_data, np.ones(10), 'valid')/10

        fourty\_MA = np.convolve(price\_data, np.ones(40), 'valid')/40

        if (ten\_MA[-2] > fourty\_MA[-2]) != (ten\_MA[-1] > fourty\_MA[-1]):

            if ten\_MA[-1] > fourty\_MA[-1]:

                output.append((date, 'buy'))

            else:

                output.append((date, 'sell'))

Pyspark code is the same as question 1

Here is the result:

Time: 2022-04-12 21:12:55

-------------------------------------------

[('3/8/90',

'buy'),

('4/26/90',

'sell'),

('5/14/90',

'buy'),

('8/2/90',

'sell')]

Bloom Filter

1. Your pyspark (or scala) code and

class BloomFilter(object):

    def \_\_init\_\_(self, n):

        self.size = int(-(n \* math.log(0.05))/(math.log(2)\*\*2))

        self.size = int(8 \* n)

        self.hash\_count = int(8 \* math.log(2))

        self.bit\_array.setall(0)

    def add(self, item):

        digests = []

        for i in range(self.hash\_count):

            digest = mmh3.hash(item, i) % self.size

            digests.append(digest)

            # set the bit True in bit\_array

            self.bit\_array[digest] = True

    def check(self, item):

        for i in range(self.hash\_count):

            digest = mmh3.hash(item, i) % self.size

            if self.bit\_array[digest] == False:

                return False

        return True

1. A recorded session showing the streaming filter in action. The session should be no more than 120 seconds in length. [ To do this, create a zoom meeting, set it to *record to the cloud*, plan what you are going to say, then start the meeting with just yourself, share the screen, and go through the demo. Soon after you end the meeting, zoom will send you a recording URL, which you may watch – and submit the URL]

<https://drive.google.com/file/d/1s-Phf1buz5qKP-Cay2Gg-LSRlbs_NgmK/view?usp=sharing>

My tufts zoom account doesn’t allow me to record virtually, so I shared with google hope it will work.