Forecast Project Requirements

One of the most important parts of the algorithms developed in Invent Analytics use historical sales data to predict future sales, i.e. Forecasting. There are many approaches to this problem. In this project, we want you to implement one way of calculating forecasts.

The project can be implemented using C# or Java. A simple Console application will suffice. No UI is required.

In order to calculate forecasts, the algorithm requires historical sales data. We have generated two months of sales data exported it as a zipped CSV file. You can download the zip file from the following url:

https://s3-eu-west-1.amazonaws.com/invent-misc/interview/2MonthSalesData.zip

The provided CSV file has the following columns:

- **Product**: an item sold inside stores.
- **Store**: a location where items are sold.
- **Date**: the date of the transaction.
- **SalesQuantity**: the quantity of products sold at the store in the given date.

Steps

1. Calculate average weekly sales of all items in all stores:
    a. Find total sales quantity of all items in all stores.
    b. Divide total sales by number of weeks in the dataset.
2. Calculate weekly standard deviation of all items in all stores:
    a. Calculate each week's sales for a product in all stores: Sales [x1, x2, x3, x4,. ...] N weeks for each product.
    b. Get Average (Mean) Weekly Sales from previous calculation. *mean*
    c. Calculate the standard deviation:
        i. `SQRT ( 1/N * (SUM ((x(i) - mean)^2)) )` (Note: N, number of weeks, i is index of week, SUM function to run on all indexes)
        ii. `x(i)`: i indicates the index of the week, so `x(1)` is the total sales of product X in week 1.
3. Estimate following week's sales for each product in each store. In previous first two question we have calculated for products on whole stores but in this question we as for each product's sale estimate on each store separately.
    a. Let's say data ends in time 't'; in order to estimate we have the following formula:
        i. `S1 = TotalSales[t:t-7] * 0.5`

ii. `S2 = TotalSales[t-7:t-14] * 0.4`
iii. `S3 = TotalSales[t-14:t-21] * 0.1`
iv. `EstimatedSales = S1 + S2 + S3`
v. S1 correspond to last week's total sales of productX in storeX, S2 and S3 are for the preceding weeks. `EstimatedSales` is the estimated sales of productX in storeX.

The result of steps 1 and 2 should be saved as a CSV file with the following format:

- Product
- TotalSales
- AverageWeeklySales
- WeeklyStandardDeviation

The result of step 3 should be saved as a CSV file with the following format:

- Product
- Store
- EstimatedSales

One can develop completely different approaches which use different coefficient values, or which do not use coefficients at all. For example, instead of using the last three weeks, one can use the last two weeks with [t:t-7] having a coefficient of 0.7, and [t-7:t-14] having a coefficient of 0.3. Or, one can use the last four weeks with coefficients 0.4, 0.3, 0.2 and 0.1 moving backwards.

We would like you to implement this application in an object oriented structure which allows a user to use different algorithms as he/she wishes. The application can accept a command line parameter which specifies the algorithm to run, or the algorithm can be specified in a configuration file. The application will dynamically load the corresponding algorithm implementation in runtime based on the configuration.

The algorithm can parse and load the CSV file in each run. Or you can write a script to load the data into a database (e.g. SQL Server Express) and the algorithm can skip the overhead of CSV parsing on each run. Either way, the details of data access must be abstracted so that the forecasting algorithm is not concerned with where the data is coming from. Feel free to use any library you wish for CSV parsing, database access etc.

You can deliver the implementation and output files via email, or share it through a service such as Github or Bitbucket etc.