Recurrence Modeling

Owner: Nick Sohn
Created: 2020/12/21
Location: Waldeneers > Team Drives > Engineering > Projects > Website Redesign >
Summary: This is a project document for Rez, who is a Swatworks extern. It provides an initial scope and description of the project, and is intended to document any work done over the externship.

Overview

Swarthmore College provides a ‘microgrant’ opportunity, where students are funded to take on a short term industry project with alumni. These projects are intended to be a learning opportunity, and give the student exposure to real world applications of their interests. Walden Local is interested in sponsoring a student to explore machine learning applications. Due to the nature of COVID-19, this will be a fully remote project. However, we will provide some opportunities to interact with the larger team if desired.

Given the short nature of the proposed project, this problem has been abstracted to a simplified condition. The ideal outcome of the project would be a working script which simulates the desired test conditions, but a thorough and detailed test plan is also sufficient. Depending on the quality of work and level of interest from both parties, this project could potentially be extended into a collaboration for summer research, academic projects, or a future internship opportunity.

This is a proposal for a 40 hour project, stretched over 4 weeks (10 hours per week).

- Week 1: Introduction to systems, problem overview, solution space exploration
    - Investigate trends in delivery days and makeup days
    - Identify common patterns and edge cases for testing
- Week 2 + 3: Implementation, testing, and validation
    - Write a script for running algorithm through test conditions, validate that it works as expected
- Week 4:
    - Documentation, cleanup
    - Bonus: run through all scenarios outlined from week 1, make a recommendation for changes to the algorithm if necessary, or edge cases that are poorly handled
    - End of week review of deliverables

Outside of these constraints, the student is free to structure their schedule as required. The timeline may be extended (working less hours per week), but per the requirements of the grant active work time must be limited to 40 hours.

Final Deliverable

At the end of this project, any code will be checked into our git repository. As is described in our NDA, any work will be the intellectual property of Walden Local. We can discuss how to make aspects of this code shareable for the purposes of developing a portfolio for the student.

The student will also be responsible for maintaining this project document, and (if interested) the student will have the opportunity to present his findings to a subset of team members in the systems engineering group.

Project Context

Walden Local is a 6 year old startup building regenerative agriculture infrastructure in New England. We partner with small industrious farmers who use management intensive practices to raise grass-fed beef and pasture raised pork and chicken. Walden handles the logistics of processing, packaging, and fulfillment and delivery for a subscription service member base covering New England and parts of New York.

This project involves the subscription logic of Walden's operation. Our members typically receive a share once a month, although there are some out of scope edge cases that allow for deliveries every other month, or once a quarter (3 months). Outside of the member's normal delivery day, we create 2-4 makeup days per month. A member can elect to have their share delivered on the makeup day instead of their normal delivery day. This often happens with the member's first share (in order to get it as soon as possible), an upcoming holiday where the member won't be home to receive the delivery, or when a member misjudges how much they will consume and either have too little (resulting in an earlier makeup day) or too much (resulting in a later makeup day) product.

The issue with makeup days is that they are less efficient from a delivery perspective. A member's normally scheduled delivery day consists of only deliveries to that specific region (called a zip group). This means the distance between stops for our drivers is small, and the overall cost of delivery is low. On makeup days, we deliver to about half of our delivery area, so stops are more spread out and drivers have to drive further, which raises our delivery costs. On one hand, we are incentivized to allow members to use makeup days because it improves the quality of our service, and prevents members from skipping their share (better to delivery a share with a lower margin than no share at all). On the other, we want to get members back to their normal delivery day as fast as possible.

The main challenge with getting members back on their normally scheduled delivery day is keeping the share spacing close enough to 4 weeks such that the member does not end up with too little or too much meat by the time their next delivery comes around. The issue is further complicated by the fact that makeup days do not occur on the same day every month. They are scheduled based on driver availability, inventory, and other factors.

Acceptance Criteria

The goal of this project is to evaluate a simple algorithm for assigning a member's next delivery day. The evaluation criteria is the proportion of shares that take more than 2 month cycles to get back onto their normal delivery day. You will use real delivery data from the past year to identify the general patterns in the delivery schedule, and identify some representative shares for the most common initial conditions.

Your validation project should meet the following criteria:

- Contains a study of the distribution of makeup and normal delivery days to identify examples of normal and irregular service months
- Identifies the most common patterns for initial conditions (last delivery day compared to next scheduled normal service day)
- Includes a test script that implements the proposed algorithm, taking a date last delivered, and the attached makeup day and delivery info as input
- Bonus:
    - Complete validation and a recommendation as to whether the algorithm works
    - A generalized script that can run on any share in our order history
    - Data analysis illustrating trends in different member groups and average time to convergence with normal delivery day

Design Notes

Below are some plots which I generated via Excel and other similar tools to identify common patterns in Walden's deliveries for both regular and makeup delivery dates. I have eliminated the actual plots and my notes on what I visualized.

[REDACTED] Total deliveries to each makeup region by month

[REDACTED] Recurrence of different delivery routines over time

[REDACTED] Number of days between makeup deliveries to "name of region"

[REDACTED] Number of days between makeup deliveries to "name of region"

[REDACTED] % zip codes mapping to each makeup region for each normal delivery date.

The work shown above helps in identifying relationships between delivery dates and their corresponding makeup dates. Namely, the normalized tables derived from zip code information helps find the average offset of makeup dates relative to deliveries, or in other words, the average number of days between a delivery day and the following makeup day. This is done using the following summation over the four regions:

$$\Sigma(\text{Days to Next Makeup Date})*(\text{Percentage of Zip Codes in Delivery Date})$$

The information discovered from this analysis is illustrated below:

[REDACTED] Average no. of days before the next makeup day following a delivery (all Regions)

[REDACTED] Average of days between consecutive makeup deliveries (by month)

[REDACTED] Distribution of delivery dates by region (they are all similar as expected)

I. <u>Testing and Edge Cases</u>

- Without going into much detail, I found that certain months are either more "crowded" or more irregular than others, and I identified the sparser/more irregular months as potential edge cases to focus on.
- Zip codes which interchange between zip groups from time to time or otherwise just have irregular delivery intervals are also good edge cases for testing.

Implementation Notes

Below is a high-level sketch of how I plan to implement the algorithm.

- <u>Numbers</u> correspond to general tasks.
- <u>Letters</u> represent functions/methods.
- <u>Roman numerals</u> represent bottom-up implementation steps.

1. **[DONE]** Allocate memory to the program's input data
    a. Zip code and last delivery
        i. Both are passed in via the command line.
    b. Get Makeup Days
        i. Copy Service_day and region_lock from makeup days CSV.
        ii. Return an array of pairs of service dates and routines.
    c. Get Makeup Groups
        i. Copy [zip] code and makeup_group from mapping CSV.
        ii. Return a dictionary of codes and regions (fast lookup).
    d. Get Delivery Days
        i. Copy Service_day and group_concat from delivery info CSV.
        ii. Return an array pairing service dates with lists of eligible zip codes.
2. **[DONE]** Use Zip class to find/store relevant data for a given zip code
    a. Find Makeup Group
        i. Find the zip code in the mapping dictionary.
        ii. Return zip code's corresponding makeup group as a string.
    b. Find [Eligible] Makeup Days
        i. Find all makeup dates that deliver to the zip code's makeup group.
        ii. Return array of eligible makeup dates for zip code.
    c. Find [Eligible] Delivery Days
        i. See 2(b).
3. **[DONE]** Actual implementation of the algorithm

Validation Notes

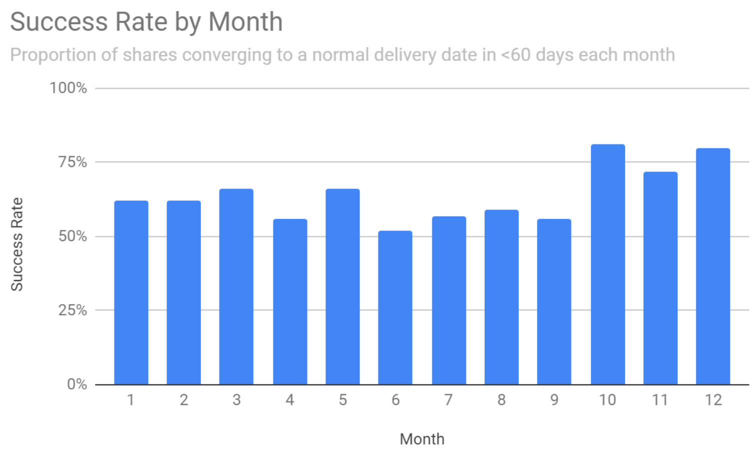
The next step is to test the algorithm in various ways. The primary evaluation criterion will be the “proportion of shares that take more than 2 month cycles to get back onto their normal delivery day.” The plan is outlined below.

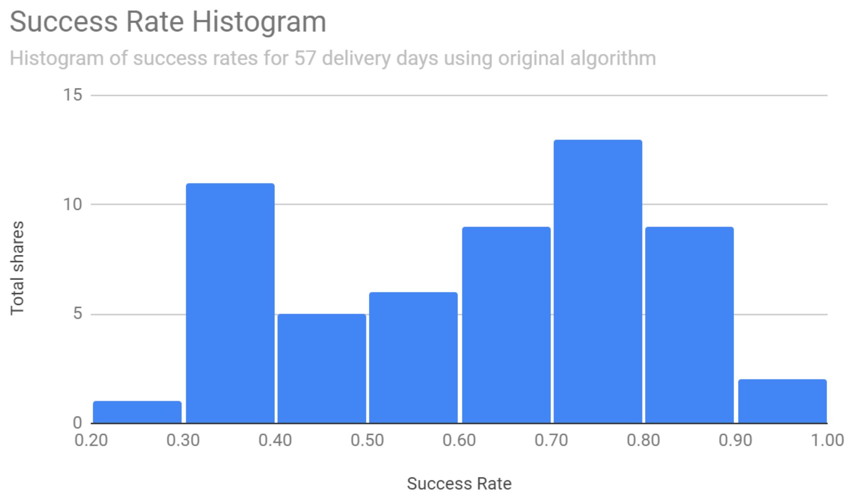
1. A benchmark will be obtained after examining 12 makeup dates between August and October--and averaging out their success rate. These functions/methods will be used:
    a. **Algorithm2**: recursive function that returns the next *normal* delivery date
        i. Using the same process as the first algorithm, find the next delivery date
        ii. If next delivery is a normal delivery day: return it.
        iii. If next delivery is a makeup day: check offset from initial condition
        iv. If offset is past a certain cutoff point from the initial condition: stop searching for a delivery day and return the initial date + cutoff
        v. If the next makeup day is not past the cutoff: recursively call Algorithm2 on the makeup day
    b. **Success Rate**: finds the success rate when starting from a given day
        i. Create an array of zip codes corresponding to a given makeup day
        ii. From Algorithm2, get the next normal delivery day for each zip code
        iii. Count the number of zip codes for which convergence takes <60 days
        iv. Divide convergence count by size of zip code array. Return it as a float
    c. **Convergence**: for a given zip code, finds the average number of days before convergence with a delivery for all makeup dates
        i. Create an array of eligible makeup dates for the zip codes within a certain range that an be studied
        ii. For each makeup date: find the time of convergence using Algorithm2
        iii. Return the average number of days to convergence as a float

Next-Next Delivery Day

[REDACTED] Outline of my tweak to the company's delivery algorithm and an example of a common trend in the data for which the algorithm behaves very strongly.

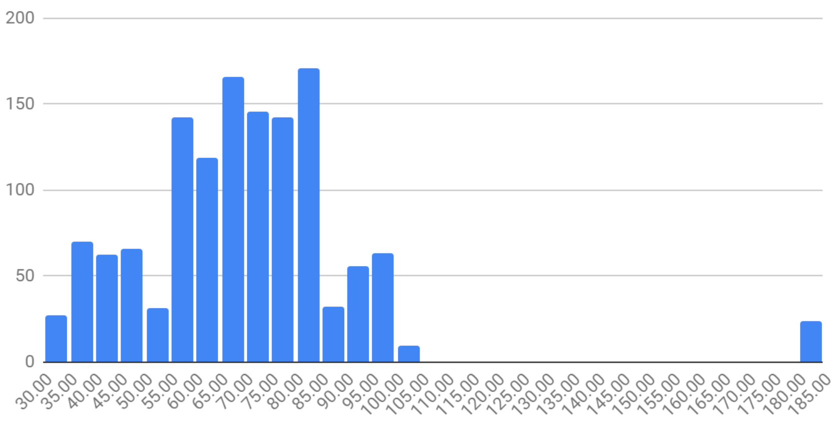
Data on the success and convergence rates are visualized below:





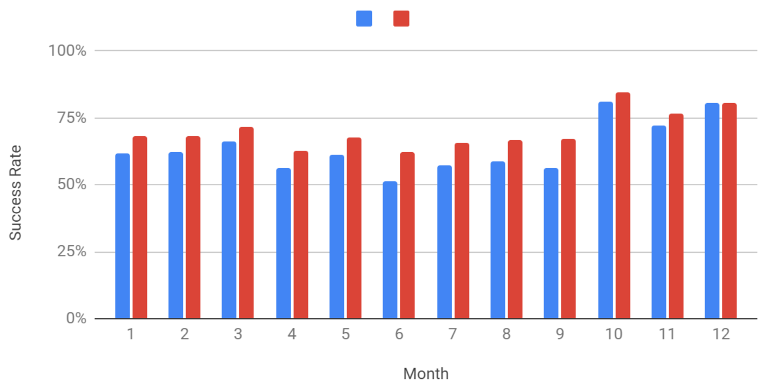
Convergence Histogram

Mean: 69.7; Median: 67.1; Std. Dev: 21.9



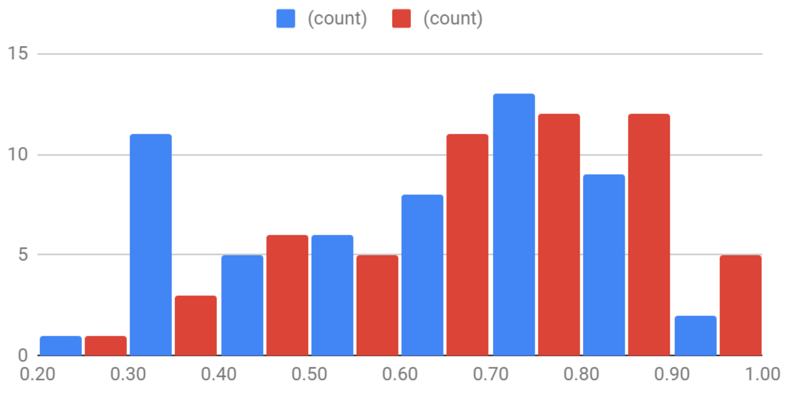
Success Rate Comparison

Compares the success rate of the original (blue) and new (red) algorithms



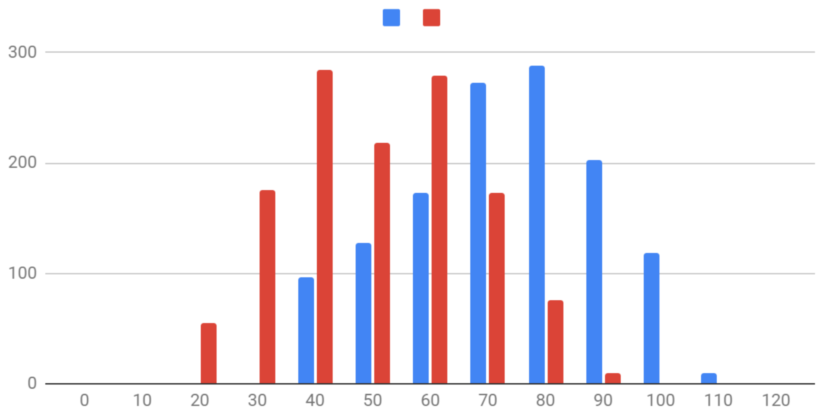
Success Rate Histograms

Compares the success rate of the original (blue) and new (red) algorithms



Convergence Histogram: Both Algorithms

Side-by-side comparison of average convergence for both algorithms across all shares.



TAKEAWAYS

The new algorithm, with a simple modification and without stretching the delivery time window, was compared to the original one. The results are summarized below:

- Average success rate was brought up from 62.3% to 69.0%.
- The modified algorithm was an improvement over the original for every month.
- The modified algorithm brought down the average convergence time by 15% from 69.7. This is a trimmed average: for shares with windows greater than 180, 180 was put down as the offset from the initial condition in order to avoid segmentation errors.
- When looking at the median, the improvement is even more defined (18%). This is because the original algorithm had more shares outside the 180-day window.
- The new algorithm has a smoother distribution of convergence times and success rates.

The analysis above shows that the changed version of the algorithm is a likely improvement above the original in terms of reducing convergence times and increasing the success rate (both across all shares and by each month), though more analysis may be necessary.

In terms of thinking about whether this change should be implemented, I would need to know more about the consumer demands associated with deliveries. Qualitatively, however, this algorithm does not seem like it compromises customer satisfaction for efficiency's sake: the windows for choosing delivery and makeup days have not been extended. If, however, this turns out to be a flaw, another version of the algorithm could be proposed and tested:

[REDACTED]