Executive Summary

DataFramers: Samira Ahmed, Giselle Kurniawan, Daniel Neufeldt, Roger Wilson

Our team, the Data Framers, conducted an analysis of the FreshStart program, which provides closed-end credit for customers, with the aim of helping them ‘graduate’ to a revolving line of credit. We first identified overarching trends in the customer journey, particularly focusing on drop-offs before the place_order event and the subsequent stabilization. Through clustering techniques, our analysis revealed distinct customer segments, including High Value Customers, Engaged Customers, and Window Shoppers, each exhibiting unique characteristics and preferences, highlighting the need for tailored marketing strategies.

Utilizing the XGBoost algorithm, we developed a predictive model achieving an accuracy score of approximately 93%, focused on forecasting customer order placements. Our approach incorporated numerical features like session duration and temporal features such as sequence modeling. Notably, we found that events such as "account down payment cleared," and "catalog" exerted significant influence on the model's outcomes.

Additionally, our investigation underscored the importance of timely communication with customers, particularly through campaign emails and promotions. We found that promptly sent promotions and campaign emails sent earlier in the journey increased the likelihood of a customer returning to the Fingerhut site, hopefully to complete an order.

In light of these findings, we offer recommendations to strengthen Fingerhut's market position and further its mission of providing support in building customer credit.

FingerHut Analysis

DataFramers: Samira Ahmed, Giselle Kurniawan, Daniel Neufeldt, Roger Wilson

Data Preparation

We first noticed that there were intermittent areas of duplicate data. Based on recommendations from Ben, we understood these to be the server's mistake in logging events, and removed all events with duplicate customer IDs, event IDs and timestamp. We decided to leave different events with the same timestamp in case some of these were logged by FingerHut. And, we decided to restructure the data into a long format. It has helped with aggregation, computational efficiency and data storage. As an example, customer -*727804037*'s information is

now stored in lists.

257	-727804037	388255280	21	catalog_(mail)	2021-08-16 00:00:00+00:00	1	NaN	Prospecting
258	-727804037	388255280	15	application_phone_approved	2021-08-28 13:15:26+00:00	2	1.0	Apply for Credit
259	-727804037	388255280	1	promotion_created	2021-10-08 12:14:48.023000+00:00	3	NaN	Promotion
260	-727804037	388255280	1	promotion_created	2021-11-12 21:04:14.683000+00:00	4	NaN	Promotion
261	-727804037	388255280	21	catalog_(mail)	2021-11-15 00:00:00+00:00	5	NaN	Prospecting
262	-727804037	388255280	29	account_activitation	2021-11-16 00:00:00+00:00	6	3.0	Credit Account
263	-727804037	388255280	18	place_order_phone	2021-11-16 17:10:09.773000+00:00	7	2.0	First Purchase
264	-727804037	388255280	24	campaignemail_clicked	2021-11-16 23:27:37+00:00	8	NaN	Discover
265	-727804037	388255280	27	account_downpaymentcleared	2021-11-18 00:00:00+00:00	9	5.0	Downpayment
266	-727804037	388255280	28	order_shipped	2021-11-22 00:00:00+00:00	10	6.0	Order Shipped

customer_id	account_id	ed_id	event_name	event_timestamp	journey_steps_until_end	milestone_number	stage
-727804037	[388255280]	[21, 15, 1, 1, 21, 29, 18, 24, 27, 28]	[catalog_(mail), application_phone_approved, p...	[2021-08-16 00:00:00+00:00, 2021-08-28 13:15:2...	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]	[nan, 1.0, nan, nan, nan, 3.0, 2.0, nan, 5.0, ...	[Prospecting, Apply for Credit, Promotion, Pro...

It was also difficult to consider which customers were still active during the dataset's timeframe. Again, as Ben mentioned, customers were labeled to have 'quit' after reaching 60 days of inactivity since submitting their application. So, we removed all customers with their first event occurring within the last 60 days of the dataset (9/20/2023). This removed 77,193 customers in total, or only 4.7% of the total customers. So, it's important to keep in mind that for the rest of this analysis, the population of interest is all customers that did not quit.

Another careful consideration was whether we should include phone users in the data. We noticed that phone users had different journey characteristics: they only took an average of 25.7 steps compared to web users' 33.1, and 42% of users had an order shipped, compared to 18.6% for web users. Our assumption was that phone users deliberately downloaded the Fingerhut app so they already had some intent in buying a product. However, seeing that the available data only includes customers that had their application approved (milestone 1), they would all have a similar intent in buying a product. And, phone users accounted for only 4.7% of all customers and rarely switched over to the website during their journey. So, phone users were not removed from our analysis.

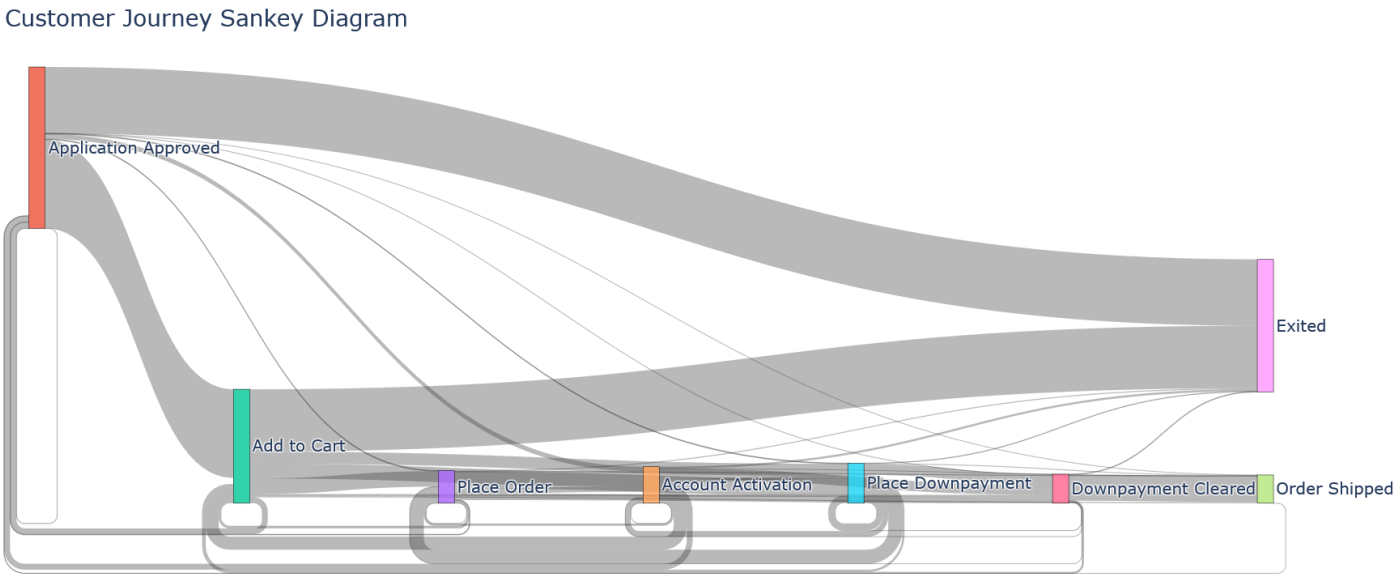
In terms of computational ability, some of us used Dask, a parallel computing library in Python that enables scalable and efficient processing. By saving the data as a Dask dataframe, it's been incredibly efficient to map data cleaning steps to lazily stored Pandas dataframes, or

partitions. Additionally, we can easily test computations on partitions before modeling the full dataset. While some of our members used Python, the remaining opted for R. We in particular used the `data.table` package which is known for its durable data manipulation features. This approach was particularly beneficial for many computations throughout our analysis since it allowed for rapid aggregation and complex operations while conserving significant memory.

Typical Customer Journey

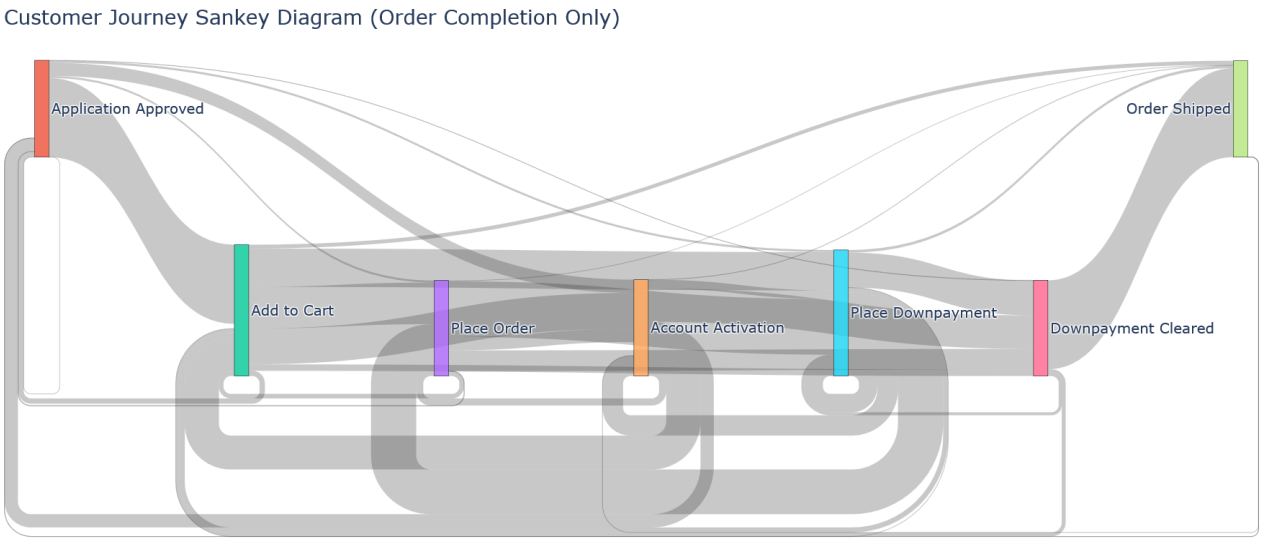
Complete vs. Incomplete Journeys

For our analysis, we considered the event ‘order shipped’ to be a sign of a complete journey, because this is the last step of the FreshStart program. Surprisingly, over 80% of web customers did not meet this criteria. We first wanted to visualize the typical journey to understand this dropoff, and to do so, we created Sankey diagrams of the milestone events (application approved, place order, account activation, down payment paid, down payment cleared, order shipped). We also wanted to consider ‘add to cart’ as an important milestone, because many customers would simply browse the FingerHut site without an intent of buying.



Of a random sample of 10,000 customers, 56% would add an item to their cart, but surprisingly, 41% would not, and would become inactive, or unsuccessful buyers (represented by the black exit node). Of customers that did add an item, 54% would become inactive after this step. However, those that go through with their purchase are very likely to complete the rest of the milestones, such as placing down payments and applying, with a reasonable amount of looping in between these later stages.

To hone in on customers that had a successful journey, we also created a Sankey diagram focusing just on a random sample of 3,000 customers. For starters, a majority of them follow with adding an item to their cart after their application is approved. But after that, there's almost equal likelihood of a customer placing the order, activating their account, and placing a down payment. At the same time, a fair amount of customers would go back to add other items to cart.

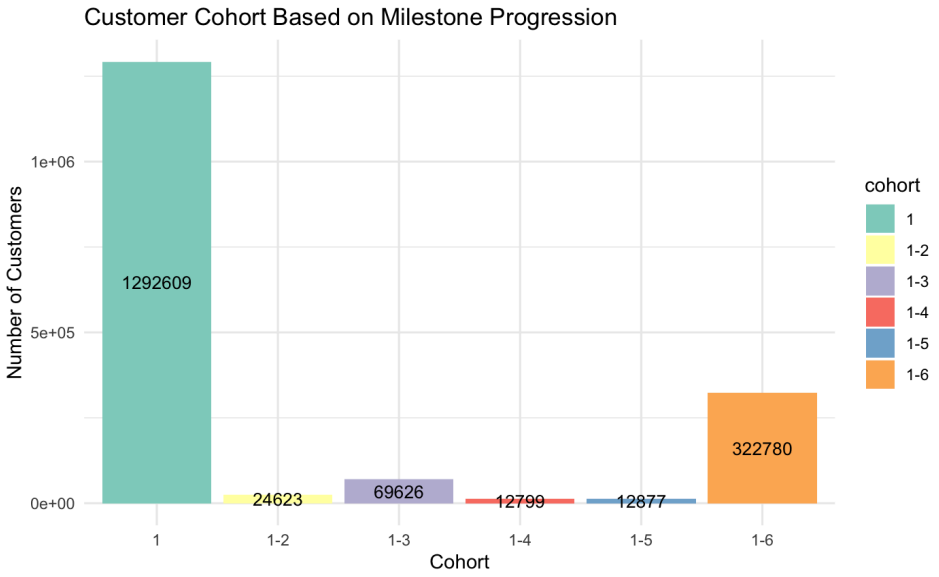


With this in mind, we also wanted to compare the sequences in which customers visited each stage. With a random sample of 4,000 customers, our visuals can be found in the appendix. Some key findings are as follows:

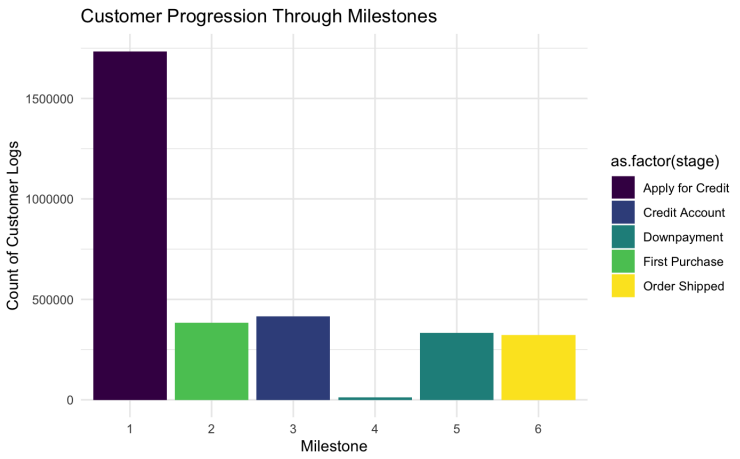
1. Successful customers (those with a purchase) overwhelmingly activated their account first (12.06%) compared to those without a purchase (1.54%). This result indicates that a fair amount of customers with a purchase signed into the FingerHut site already with the intent of making a purchase.
2. As we'll explore later, customers without a purchase receive significantly more promotions and prospecting emails compared to their counterparts because they require more incentive to continue their journey.
3. Successful customers spend significantly more steps on the FingerHut site. For example, by step 19, over half of customers without a purchase have already exited their journey, compared to 14.1% for those that will go on to make a purchase. By step 55, 90% of unsuccessful customers would quit compared to 65% of successful customers. Those still shopping for both groups are usually in the 'First Purchase' stage.

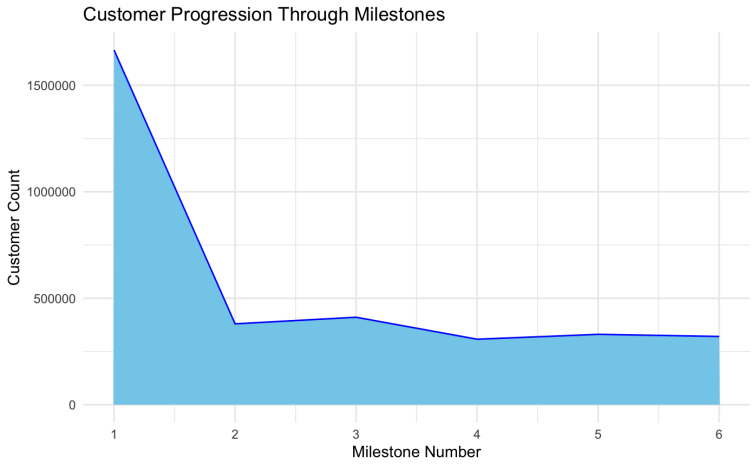
Key Areas of Drop Off

To better understand the customer journey, we took the approach of segmenting customers based on the earliest and latest milestones they reached. We then calculated the size of each cohort. This grouping allows us to quantify the customer base at each stage of the process. We recorded 1,292,609 customers who did not progress beyond the first milestone. When examining the transition from milestone 1 to milestone 6, we observed that only 18.60% of customers made it to the final milestone from the combined pool of customer and account identifications. This decline raised some questions about the barriers that may be preventing customer progression.



Our initial analysis revealed a decline in customer progression between the first and second milestones. While the initial engagement is high, suggesting effective initiation, there is a substantial decrease in the number of customers that transition to the second milestone. Our data also indicates a stabilization after milestone 2, suggesting that customers who surpass this initial milestone have a higher likelihood of continuing through subsequent stages. This pattern is important in understanding the customer journey between milestones 1 and 2.

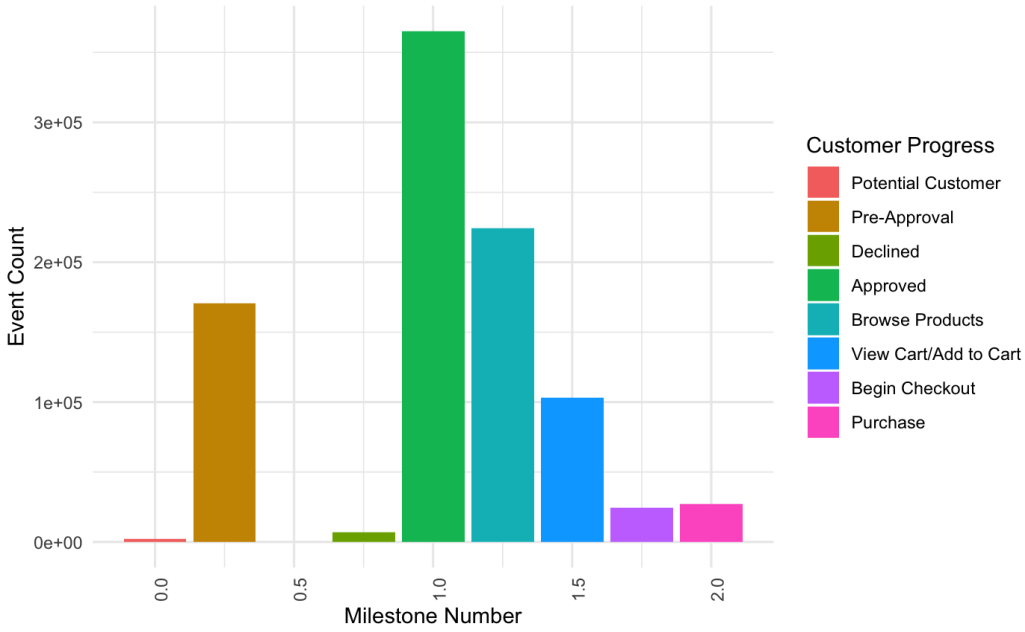




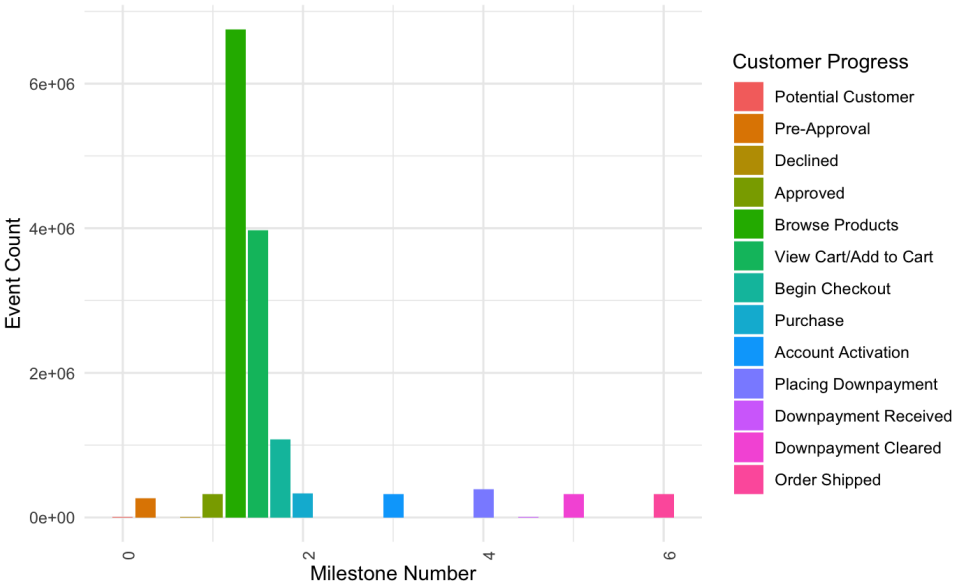
In our analysis, we have observed that not all customers follow a uniform path through the given milestones. We found that only 57 customers began their journey at a milestone greater than 1, indicating that the vast majority of customers did initiate at the starting point of our defined journey, i.e., milestone 1. This may be due to missing or incomplete data. To gain a deeper understanding, we introduced subcategories within a subset of our data to account for discrepancies in a customer's journey. This allowed us to incorporate events that fall outside the typical milestones (1 through 6) yet are significant in defining customer progression. This segmentation of data revealed distinct patterns and behaviors among different cohorts of customers.

For instance, for the event definition ‘site_registration’, we mapped this to an initial stage, denoted as milestone 0, labeling customers at this phase as 'Potential Customer'. For event definition identifiers ‘application_pending’, ‘application_view’ and ‘application_submit’ relate to actions taken during the 'Pre-Approval' phase, and are assigned a sub-milestone value of 0.25, reflecting their position in the early stages of the customer journey.

A noteworthy observation we made for the cohort of customers up to milestone 2 is that some of their logged events were consistent specifically at the 'Begin Checkout' and the 'Purchase' stage. This suggests that customers who initiate the checkout process for this group of customers are just as likely to follow through with placing an order.



As for the category of customers with a minimum milestone of 0 and a maximum milestone of 6, stability is evident after the second milestone. Another observation we made is there are approximately 6.8 million event counts associated with "Browse Products" by customers in this cohort. When considering the total number of 322,780 customers within this category, the consistent level of activity is remarkable. This pattern of engagement reflects positively on Fingerhut's ability to retain a customer's interest over an extended period. Moving forward, we will use the elements that contribute to this sustained activity and see if it could provide valuable insights about a customer's overall journey.

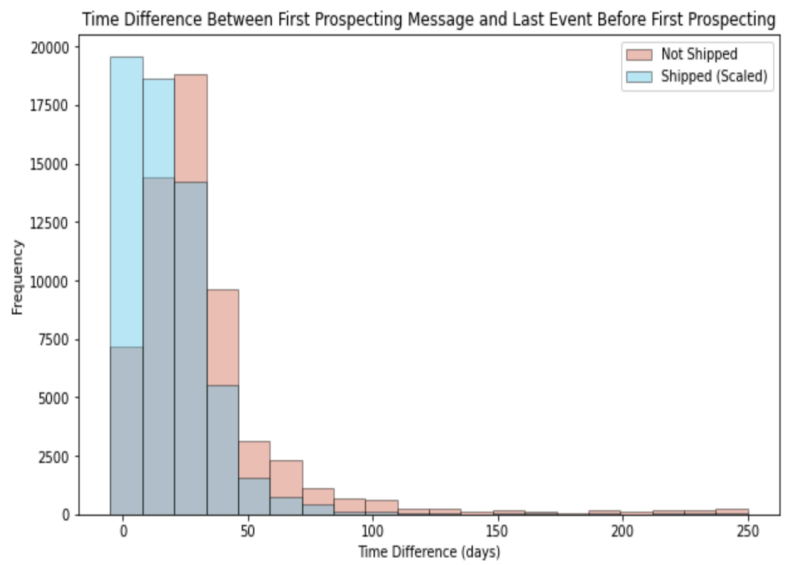


Actionable Ways to Increase Order Completions

Prospecting Stage

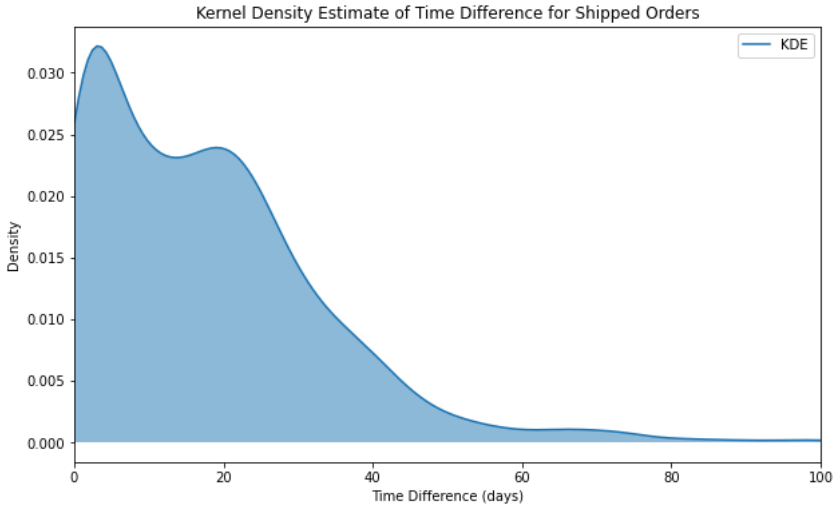
In this part of the analysis, we wanted to look into factors that Fingerhut could explicitly change, one being sending catalog mail. We specifically looked at both events in the Prospecting stage ("catalog_mail" and "catalog_mail_experian"). We first decided to see the percentage of customers who completed purchases with and without prospecting mail. Surprisingly, we found that 26% of customers in a random sample of 165,000 customers completed purchases without receiving prospecting mail while only 10% of customers in the same sample completed purchases having received prospecting mail. This can raise the question of how we can improve the percent of customers who make purchases after receiving catalog mail. Looking further into the data, we can see that only 32.75% of customers actually come back to the website after receiving prospecting mail and of those customers, 33.69% end up making a purchase. Taking a deeper look at the data, the two main controllable factors that Fingerhut can take into account when sending prospecting mail are the timing of the mail and the journey step that a customer is on.

To determine the best time to send out prospecting mail, we calculated the time difference between the first prospecting event of a customer and the previous event in that customer's journey, resulting in the graph below (note that the data for customers who completed orders is duplicated or scaled by a factor of 10 in order to show both distributions relative to each other). The median for customers who shipped orders

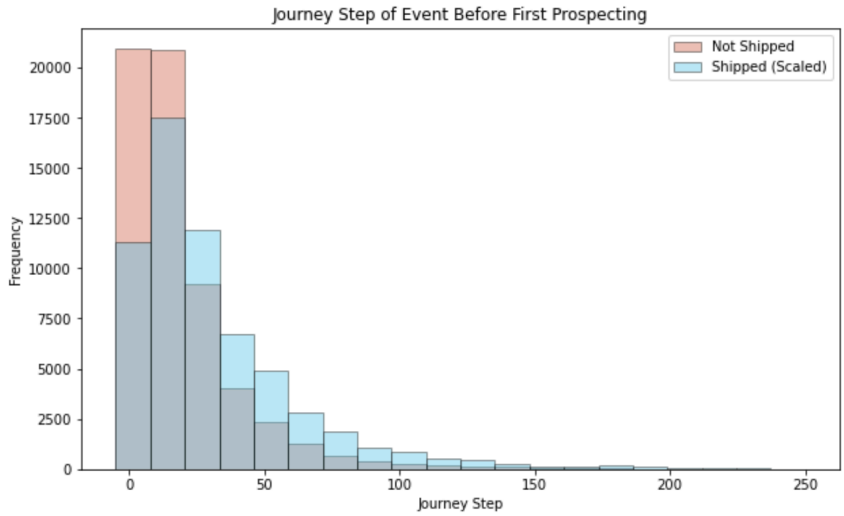


is 15 and the median for customers who have not is 26. To statistically confirm that there is a difference in the distributions, we ran a Kolmogorov-Smirnov Test that confirmed the two distributions were different. In order to statistically verify that there was also a difference in means and medians of the two distributions, we ran a bootstrap hypothesis test which confirmed statistical significance in the difference of means and medians.

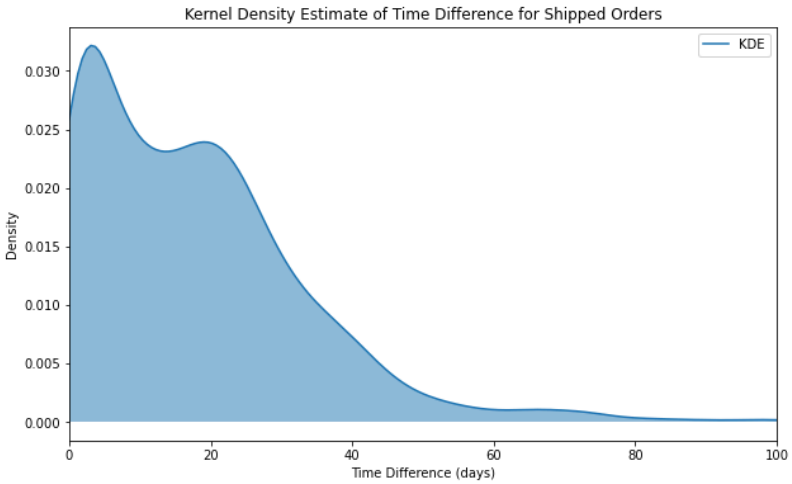
The next thing we did was use Gaussian Kernel Density Estimation, a non-parametric method for estimating the probability density function of a random variable, to find the optimal time difference. This process essentially smooths out the data points into a continuous distribution and then we looked for an interval within the middle 50% of data points that contained the greatest area under the distribution curve. Through this process, we determined that the best time to send prospecting mail is between one to six days after the customer's current journey step, with an optimal peak at approximately three days after.



We now have an idea of how long Fingerhut should wait before sending prospecting mail, so we then looked into at what point in the customer's journey should they send prospecting mail. Doing the exact



same procedure as before, except using “journey_steps_until_end” instead of time difference, we came up with the following graphs to the left. The median journey step of the last event before the first prospecting message for customers who completed orders was 22 whereas the median for customers who did not complete orders was 12. Running the same exact statistics tests as with time differences, we obtained the same results in terms of significance. Finally, according to the Kernel Density Estimation, the optimal journey step for customers who completed orders is between six and nineteen, with a peak optimal value at approximately 12 journey steps. These results could be because users had time to develop interest in Fingerhut.



With these results, we were able to find the best time and journey step customers were on to send prospecting mail to in a statistically significant manner. Our advice to Fingerhut for specifically prospecting mail is to time sending the mail to customers who are in the mid-early stages of their journey (journey steps between 6 and 19) within one to six days.

Promotions

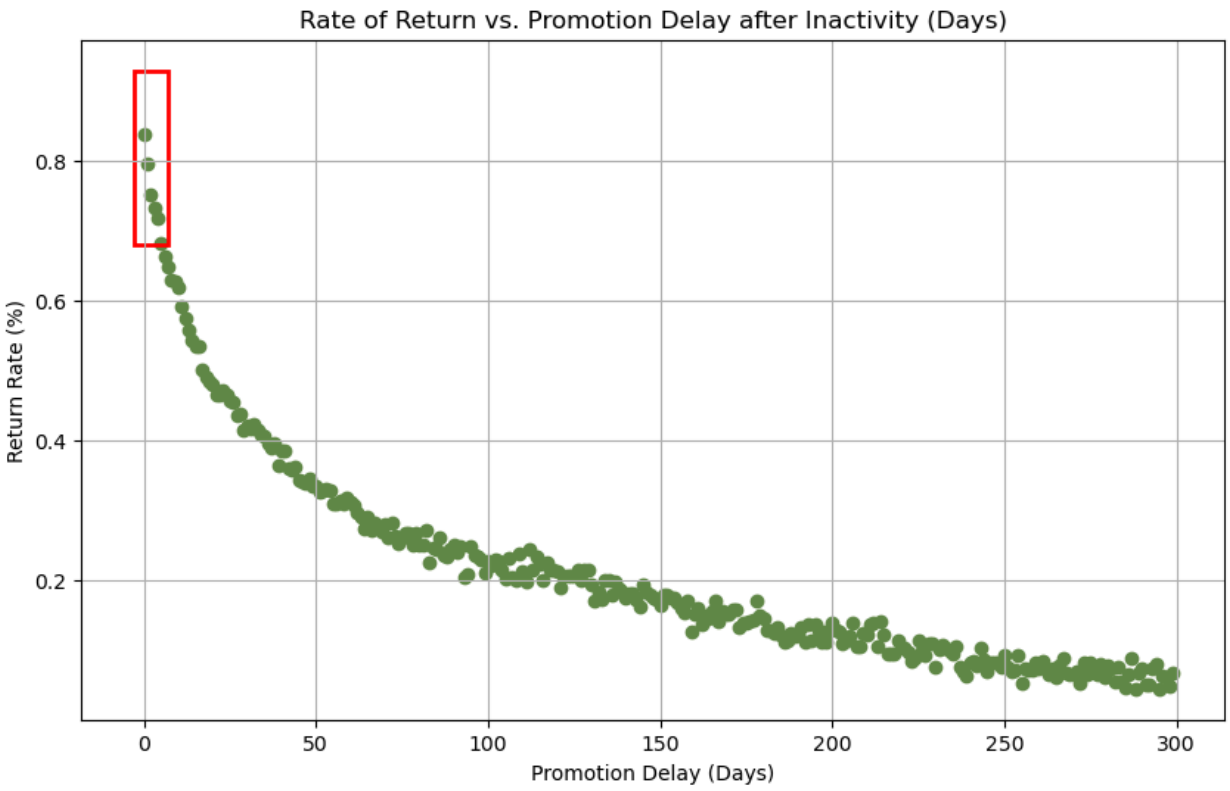
Another potential way to entice customers back to their journey is by sending promotions, or one-time deals to customers. While we do not have information on the promotions themselves, we were able to look into the frequency and timing of promotions being sent to customers.

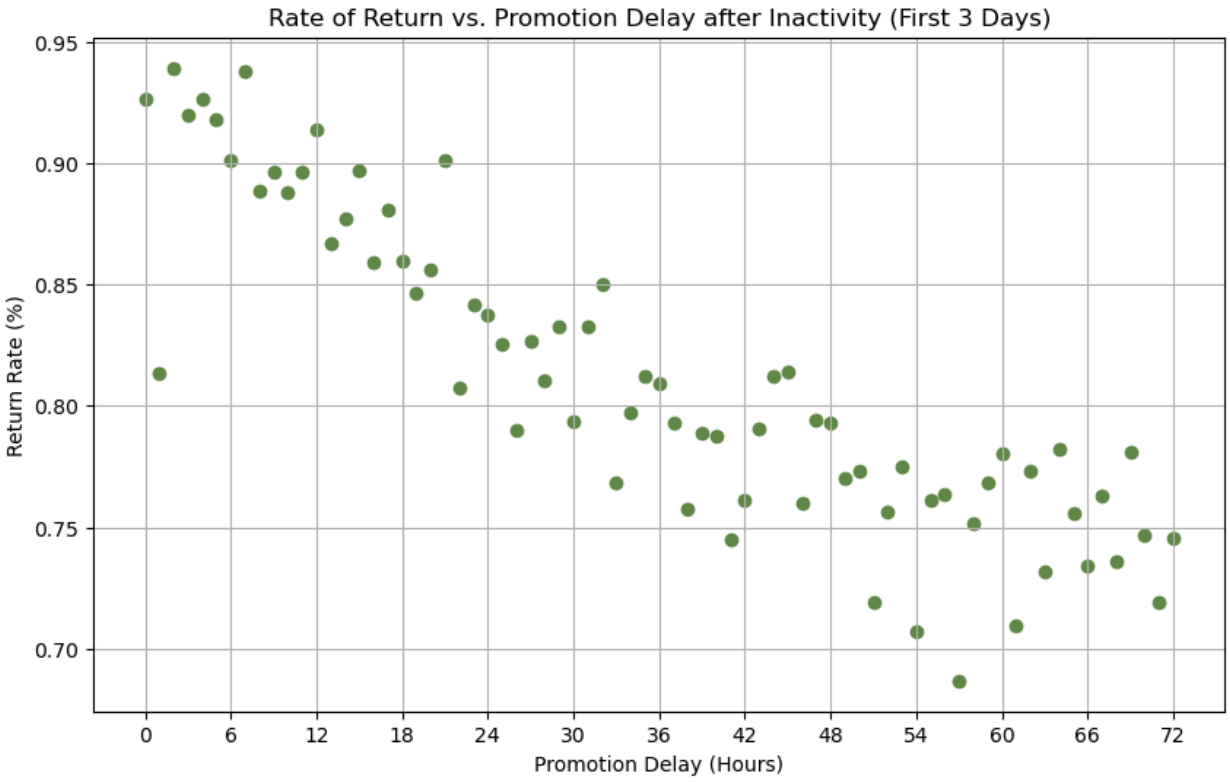
For starters, not every customer received a promotion. Out of a random sample of 240,000 customers, 19.8% of those who didn't make a purchase would not receive a promotion, while 28.8% of those who did make a purchase would also not receive one.

However, for those that did receive a promotion, we were interested in recording when the promotions were sent relative to the last customer event, and if the customer would return to the site. Our algorithm is as follows:

1. Randomly select 10% of the customers.
2. For each selected customer:
    a. Identify the steps in their journey where they received a promotion (If any).
    b. For each promotion received by the customer:
        i. Determine the customer's last event (excluding campaign emails, other promotions, and any clicks on these) and record its timestamp. Also determine the next customer event that isn't another promotional.
        ii. Calculate the time difference between the timestamp of the last non-promotional event and the promotion timestamp. This represents how much later the promotion was sent.
    c. For all promotions where the customer would return to the site (there exists a next event that isn't a promotion), record the average time of these successful promotions.
    d. Record the timestamp of the first promotion sent that resulted in no return. This is essentially Fingerhut's last contact with the customer.

For example, customer -2104674221's last event, browsing products, was on 3/24/21. They received two promotional discounts on 4/7/2021 and returned to the site two days later, but stopped browsing again. They would then receive a string of catalog emails and promotions starting on 5/18/2021, and it isn't until a promotion on 7/8/2021 that the customer promptly returns and eventually places an order. So, they received seven promotions, on average 40 days after inactivity, and all were successful.


Rate of Return vs. Promotion Delay after Inactivity (Days)
Return Rate (%)
0.8
0.6
0.4
0.2
0
50
100
150
200
250
300
Promotion Delay (Days)

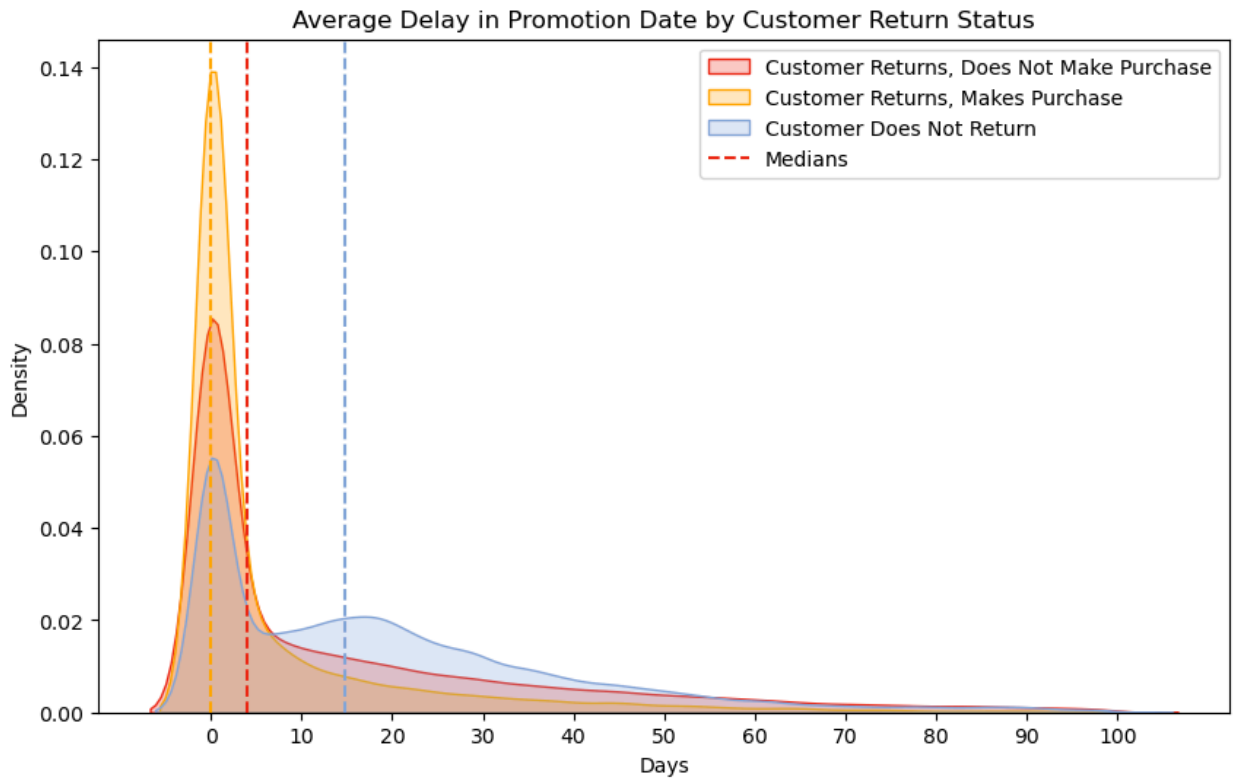

Rate of Return vs. Promotion Delay after Inactivity (First 3 Days)
Return Rate (%)
0.95
0.90
0.85
0.80
0.75
0.70
0
6
12
18
24
30
36
42
48
54
60
66
72
Promotion Delay (Hours)

With some data manipulation, we were able to calculate the likelihood of a customer returning to the Fingerhut site based on how delayed their promotion was in days. And it becomes instantly recognizable that the later a promotion is sent, the less likely a customer is to return to the site. In fact, even 20 days later, the likelihood of returning drops by 30%. The zoomed in graph also demonstrates that within the first 6 hours, promotions have over a 90% success rate in getting a customer to return.

We then compared the three possible groups of promotions:

1. The promotion sent does not get a customer to return to the site. We only recorded the FIRST promotion that met this criteria, because oftentimes the customer would receive multiple promotions and wouldn't click on any of them.
2. The promotion does get the customer to return to the site. This can be broken up into two groups:
    a. The customer returns to the site, but does not make a purchase.
    b. The customer returns to the site and does make a purchase.

When we compare the average delay for these three types of responses to promotions, we get interesting results:



Promotions that lead to:	Return, Made Purchase	Return, No Purchase	No Return
Mean delay (days)	8.6	19.7	23.7
Median delay (days)	0.08 (1.9 hours)	5.5	15.8
Number of Customers	19.3%	40.0%	40.7%

Our analysis yielded statistically significant results, indicating that promotions sent promptly after a customer's last non-promotional event are more likely to result in their return to the site and subsequent purchase. Specifically, promotions sent within the same day to approximately three days later were the most effective in incentivizing customer returns. It's also important to note that almost 30% of customers that made a purchase would not 'stall' on the site, meaning they do not delay their purchase decisions or require additional incentives or

promotions to complete their transactions. These findings emphasize the importance of timely promotion delivery in boosting customer engagement and thus success within the FreshStart program.

Clustering Algorithm

Goal

We wanted to use a clustering algorithm that groups customers with similar customer journeys and the objective was to distinguish ‘High Value Customers’—those who demonstrate commitment by advancing to at least the first purchase. If we could identify similar traits and behaviors within these clusters, we could then trace their journey to see if it culminated in an order being shipped. Once we defined the clusters, our next goal was to integrate this back into the FingerHut dataset. This reintegration would hopefully allow us to investigate whether the clusters we identified correlated with successfully completed purchases, i.e. milestone 6. By using a clustering algorithm, our aim was to anchor our previous analysis in the reality of FingerHut’s operational data to better understand a customers' pathways to successful transactions.

Feature Engineering

As mentioned in our previous analysis there is substantial stabilization past milestone 2, thus we tailored our algorithm to concentrate on customer behaviors up to milestone 2 and we eliminated the event logs of customers beyond this milestone. There are multiple event definitions for milestone 2 which are mapped to ‘place_order_phone’and ‘place_order_web’. We decided to combine phone and web user events given that phone users represented about 4% of our overall data. By merging the events for phone and web users for milestones 2 and below, we maintained the integrity of our dataset without losing significant insights, since our primary

focus was on customer's 'place_order', regardless of platform. These decisions also contributed to our goal to reduce dimensionality for computational efficiency.

We decided to use the K-means algorithm and we engineered our features by transforming many categorical variables, each representing a unique customer event definition. The dataset was pivoted wider such that each column corresponded to a specific event definition, and the rows detailed the frequency of these events per account. Initially we began with a set of 20 features to select from, including the account_id identifier column. Our goal was to reduce our preprocessed dataset to the most impactful features for our model. To achieve this, we used the variance of each column. We kept columns with high variance since these features typically meant that there is a substantial spread in the data points which in turn leads to meaningful differences in customer behavior that our clustering algorithm can leverage. Conversely, features with low variance may not contribute much to the model, as they don't vary much across customers.

```
> variances
                          add_to_cart                   application_approved                          begin_checkout
                         2.575006e+01                           2.380425e-03                           9.310889e+00
                      browse_products                         campaign_click                            place_order
                         6.020500e+02                           2.614305e-01                           2.188581e-01
                    promotion_created                              view_cart                  application_web_submit
                         1.395145e+01                           5.698789e+01                           3.033258e+00
                 application_web_view pre-application_(3rd_party_affiliates)                  campaignemail_clicked
                         4.094749e+01                           8.347408e-02                           8.018667e+00
                       catalog_(mail)                      site_registration                   application_declined
                         3.799848e+00                           1.281351e-02                           3.228419e-02
           catalog_(email)_(experian)                   application_pending                     fingerhut_university
                         3.497977e-04                           3.076789e-05                           4.229115e-04
     customer_requested_catalog_digital
                         1.206620e-06
```

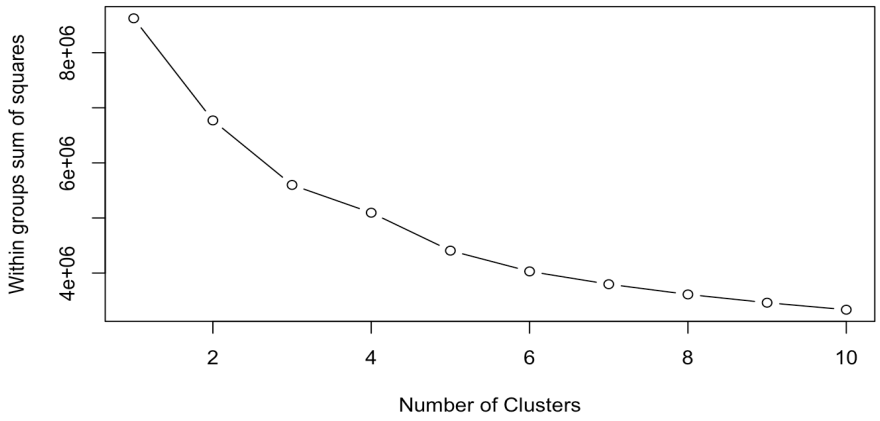
We then established a variance threshold of 0.1 to determine which features (columns) to keep. This variance threshold was chosen with a view to balance between retaining enough complexity in the data so that we may capture genuine customer behavior patterns, while also excluding features that would likely add noise to the clustering outcome. Applying this threshold, we cut the features down to 12 columns, account_id included. This refinement process

not only made the computational task more manageable but also maintained our focus on the features most likely to yield distinct and interpretable customer clusters for Fingerhut.

We had to address some outliers which are known to affect the sensitivity of the K-means algorithm. Removing these anomalies helped shape our dataset to better reflect the ‘typical’ customer pattern and in doing this it helped with some distortion we previously received from these extreme values. This removed a total of 151,021 customers accounts which is roughly 10% of our preprocessed data.

K-Means Algorithm

Prior to implementing our K-means algorithm, we utilized the elbow method to determine the optimal number of clusters. The elbow method involves plotting the within-group sum of squares (WSS) against the number of clusters and looking for a point where the rate of ‘decrease’ sharply changes. The plot generated didn’t reveal a clear elbow and in turn, the choice for the number of clusters was not immediately obvious. We experimented with 3 to 5 clusters to decide which provided the most meaningful clustering of customers . We chose to move forward with 4 clusters since it struck the best balance when grouping customers with similar customer journeys.



When implementing the K-means algorithm, we utilized the `nstart` parameter and set this to 25 to enable the algorithm to explore multiple starting centroids. This approach allowed our algorithm multiple opportunities to identify the best centroid start since K-means tends to be sensitive to the initial centroid starting points.

Results

Within our clustering results, we observed some consistent behaviors across the different customer groupings we identified within Fingerhut's data. Our 'High Value Customers' (Cluster 1) have the highest mean in the 'browse_products', 'add_to_cart', 'view_cart', 'begin_checkout', and 'place_order' features. To put into perspective the respective means for our 'High Value Customers' cluster are 15.131, 3.483, 5.836, 2.690 and 0.968. This signals not just interest, but an intent to purchase. This result is consistent with our previous analysis with key drop off points that customers who make it to milestone 6 have the highest 'browse_products', 'add_to_cart'/'view_cart', 'begin_checkout', and 'place_order' events logged in their respective cohort

	cluster	add_to_cart_mean	add_to_cart_median	add_to_cart_sd	begin_checkout_mean	begin_checkout_median	begin_checkout_sd	browse_products_mean	browse_products_median	browse_products_sd
1	1	3.482993	2	3.283318	2.6904217	2	2.245784	15.130836	10	15.642691
2	2	1.723960	1	2.582028	1.0414224	0	1.762740	10.749130	5	14.105291
3	3	1.018395	0	1.780022	0.6540606	0	1.275799	4.930002	2	8.584649
4	4	1.180133	0	2.006212	0.7693334	0	1.438952	5.623521	2	9.491248

place_order_mean	place_order_median	place_order_sd	promotion_created_mean	promotion_created_median	promotion_created_sd	view_cart_mean	view_cart_median	view_cart_sd
0.968283523	1	0.17524451	2.225144	2	2.368107	5.836125	4	5.538159
0.018846137	0	0.13598165	8.201403	8	2.358748	2.662821	1	4.117029
0.001019651	0	0.03191573	2.288650	2	2.316833	1.448178	0	2.700051
0.000000000	0	0.00000000	2.574032	2	2.491831	1.719206	0	3.069899

campaign_click_mean	campaign_click_median	campaign_click_sd	place_order_mean	place_order_median	place_order_sd	promotion_created_mean	promotion_created_median	promotion_created_sd
0.5670183	1	0.5062016	0.968283523	1	0.17524451	2.225144	2	2.368107
0.6568517	1	0.4838864	0.018846137	0	0.13598165	8.201403	8	2.358748
1.0145810	1	0.1198684	0.001019651	0	0.03191573	2.288650	2	2.316833
0.0000000	0	0.0000000	0.000000000	0	0.00000000	2.574032	2	2.491831

The ‘Engaged Customers’ (Cluster 2) stand out in several areas. They have the highest amount of promotional efforts made on their behalf and it is substantially more than any other cluster. This is demonstrated by the highest 'promotion_created' mean which resonates with our previous analysis about promotional efforts made for customers. This makes sense since this group has the second highest ‘browse_products’, ‘add_to_cart’, ‘view_cart’, and ‘begin_checkout’ means second to our ‘High Value Customers’ but only very few make it to place order which is indicated by the relatively low mean for ‘place_order’ in this cluster which is 0.018846137. ‘Engaged Customers’ cluster also leads in the mean for 'campaignemail_clicked' that is, 1.1846009, which reinforces the engagement (hence the name for this category) to the outreach efforts made by Fingerhut. Notably, this cluster has a significantly greater mean compared to the other clusters for 'catalog_(mail)' which is 3.8549212, indicating a substantial outreach on Fingerhut’s end to re-engage these customers. Although their 'place_order' mean is nowhere nearly as high as our ‘High Value Customers’ (Cluster 1), it's still considerable, underlining their importance as a target for conversion strategies.

campaignemail_clicked_mean	campaignemail_clicked_median	campaignemail_clicked_sd	catalog_(mail)_mean	catalog_(mail)_median	catalog_(mail)_sd
0.7483853	0	1.5357402	0.2997927	0	0.7270719
1.1846009	0	1.8854749	3.8549212	4	1.4581835
0.3856060	0	1.0778807	0.3429108	0	0.6762212
0.2999083	0	0.9385913	0.4871816	0	0.8581451

Our 'Window Shoppers' (Clusters 3 and 4) show varying levels of engagement. Cluster 4 outpaces Cluster 3 in 'begin_checkout' mean, suggesting a higher readiness to purchase, even though Cluster 3 eventually has a higher 'place_order' mean. This interesting dynamic could suggest a hesitation at the last moment for Cluster 4, while Cluster 3, despite fewer initiating the checkout process, has more customers who follow through with a purchase.

Looking at 'browse_products' means across all clusters, it’s evident that while browsing is a universal activity, the depth of engagement differs significantly. Cluster 1, again, has the highest mean, highlighting their intent to purchase. By examining these metrics it becomes clear where each cluster's behaviors and preferences lie. These insights are crucial as it can help Fingerhut strategize on personalization, targeted marketing, and resource allocation to elevate the customer experience and journey with FingerHut.

Evaluation of Algorithm

When we integrated the clustered data back into our original dataset by customer accounts (‘account_id’) we were able to see additional layers of insight into the customer behavior and journey at Fingerhut. There was a total ‘NA’ count of 151,021 but this is to be expected since these were the outliers we removed in the preprocessing part or our algorithm. Although computational limitations prevented us from extensively experimenting on finding the right amount of outliers, we've gained valuable information by matching the outliers with our existing cluster labels.

For 'Engaged Customers' in Cluster 2, the data shows an extremely high proportion of promotions created, at nearly one promotion per customer. This supports our earlier observations that this cluster, despite lower 'place_order' rates, is heavily targeted with promotional activities. The data suggests that Fingerhut recognizes the engagement level of these customers and is invested in converting their engagement into purchases.

cluster_label <chr>	Total_Customers <int>	Promotions_Created <int>	Proportion <dbl>
Engaged Customers	383707	383684	0.9999401
High Value Customers	317460	216870	0.6831412
Window Shoppers	805674	542583	0.6734523
NA	151021	139286	0.9222956

The 'High Value Customers' of Cluster 1 shows a high proportion of 0.803 for 'shipped_orders' given that these customers were only grouped together by features up to milestone 2 which is to place an order. This validates the importance of these features as these are the primary contributors that helped group ‘High Value Customers’. This high proportion indicates a successful conversion from placing orders to completing purchases, i.e., ‘order_shipped’ highlighting the effectiveness of our algorithm in identifying the most valuable customers.

cluster_label <chr>	Total_Customers <int>	Shipped_Orders <int>	Proportion <dbl>
Engaged Customers	383707	6118	0.015944458
High Value Customers	317460	254882	0.802879103
Window Shoppers	805674	6222	0.007722727
NA	151021	51222	0.339171374

In contrast, 'Window Shoppers' have a proportion of 0.0077 for 'shipped_orders', their lower likelihood to complete a purchase despite their high engagement in the early stages of the shopping process. The ‘NA’ category, initially removed due to the outlier status, shows a proportion of 0.339 for 'shipped_orders'. This suggests that some outlier accounts have a great impact on orders shipped. It points to the possibility that these outliers, although not fitting neatly within our existing clusters, may represent a segment of interest for Fingerhut. With increased computational resources, it is possible to refine an outlier function to potentially capture valuable patterns within this group.

Overall, our algorithm effectively differentiated between various levels of customer engagement and their impact on a customer’s journey through Fingerhut. The K-Means clusters have provided us with a broader perspective, indicating areas for further analysis with access to greater computational power.

XGBoost Algorithm

Goal

One of our primary objectives is to predict whether a customer will place an order, a key indicator of a successful customer journey. To achieve this, we have developed a hybrid model leveraging XGBoost algorithm. XGBoost, an abbreviation for eXtreme Gradient Boosting, is a supervised machine learning algorithm under ensemble learning. It builds a predictive model by combining predictions from several individual models in an iterative manner. The algorithm works by systematically introducing weak learners to the ensemble, with each new learner focusing on correcting the errors made by the existing ones. Employing gradient descent optimization, XGBoost minimizes a predefined loss function throughout the training process. Our decision to employ the XGBoost algorithm for our predictive modeling task stems from its remarkable capability to adeptly manage both numerical and categorical features. Furthermore, its gradient boosting framework empowers it to discern intricate relationships within the dataset, resulting in highly accurate predictions.

Feature Engineering

During our data preparation phase for modeling, we employed two key feature engineering methodologies: numerical features and 1-gram extraction. Our approach involved organizing the data by grouping it based on customer_id and sessionizing it, thereby permitting multiple sessions per customer_id with a session timeout of 2 days.

To capture the temporal aspect of customer interactions, we computed the duration of each session in minutes, providing us with valuable insights into the engagement levels and activity durations of individual customer sessions.

Concurrently, we performed 1-gram extraction by consolidating the event names for each customer's session into a list of strings. Here is an example of what our data looked like at this point:

	customer_id	session_id	events	session_duration
0	-2147452610	3945806	[application_web_approved, promotion_created, ...	7453.400000
1	-2147425125	235845	[campaign_click, application_web_view, applica...	1729.300000
2	-2147425125	235846	[browse_products, account_activitation, campai...	6550.700000
3	-2147417277	1727803	[application_web_approved, promotion_created, ...	407.916667
4	-2147395611	2295445	[account_activitation, campaign_click, applica...	5760.000000
5	-2147395574	4275402	[application_web_approved, promotion_created]	0.398667
6	-2147395574	4275403	[browse_products, browse_products, add_to_cart...	18274.801333
7	-2147379618	4356472	[campaign_click, application_web_view, applica...	5400.000000
8	-2147357371	6922986	[campaign_click, application_web_approved, bro...	1664.266667
9	-2147357371	6922987	[browse_products, browse_products, add_to_cart...	22426.863050

Leveraging scikit-learn's CountVectorizer, we transformed the events column into a sparse matrix. Under the hood, CountVectorizer facilitates the conversion of textual data into a numerical format, enabling us to represent each unique event as a feature and quantity its occurrence within each session. This transformation enables our model to discern and interpret the occurrence patterns of various events across sessions, thereby enhancing the overall predictive capability of the model.

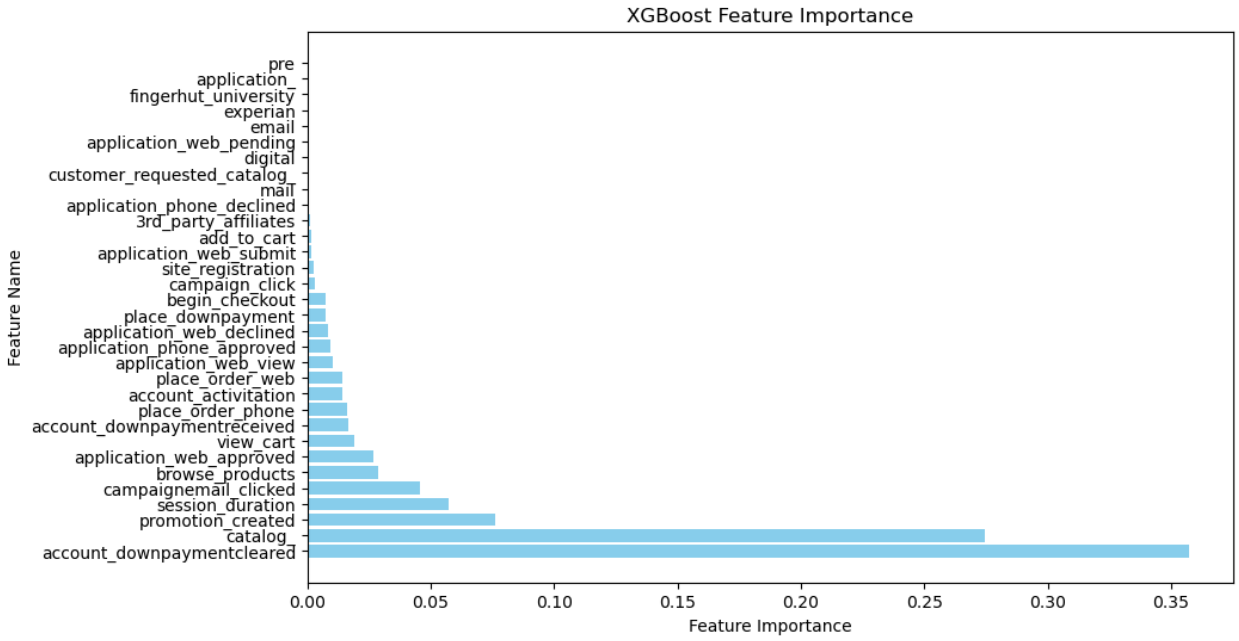
Furthermore, to construct our final input matrix, we concatenated the sparse 1-gram matrix with the session_duration column, thereby amalgamating both numerical and textual features into a unified input representation for our predictive model.

XGBoost Model

Before delving into the details of our XGBoost model's performance, it's imperative to discuss the evaluation process we employed. We began by partitioning the combined feature matrix into distinct training and testing sets. Specifically, 30% of our dataset was allocated for testing purposes, while the remaining 70% was dedicated to training our model. This division ensures that our model is trained on a substantial portion of the data while also being evaluated on unseen instances, thus enabling us to assess its generalization capability effectively. With our training data in hand, we proceeded to employ the XGBClassifier, a robust implementation of the XGBoost algorithm, to train our model. Once our model was trained, we utilized the predict function to generate predictions on the testing dataset.To gauge the effectiveness of our model, we employed the f1 score metric, a popular measure that balances precision and recall. Remarkably, our model achieved an impressive accuracy score of 93.9% when evaluated using the f1 score metric. This high level of accuracy underscores the efficacy of our XGBoost model in accurately predicting whether a customer will place an order, thus validating its utility in our predictive analytics endeavors.

Interpretability

XGBoost offers various tools for interpreting model predictions and assessing feature importance. Among these methods, we utilized feature importance ranking, enabling us to identify the most impactful features driving the model's predictions. Below is a visualization depicting XGBoost feature importance.



According to the plot, the event "account_downpaymentcleared" emerges as the most influential factor behind the model's predictions, followed by "catalog" events. This suggests that customers who have their account down payment cleared are more inclined to place an order. This insight sheds light on the significant drivers influencing customer order placements, offering valuable insights for strategic decision-making.

Final Thoughts

Our goal for this project was first to model the typical journey within the FreshStart program, but more importantly to uncover value insights into customer shopping behaviors and provide actionable ways to encourage customer retention. And, with approximately 80% of customers failing to have an order shipped, and most of them leaving their journey before even placing their order or adding items to cart, it's evident that there is room for improvement in helping customers with approved applications in their goals of improving credit.

While our clustering algorithm honed in on customer engagement prior to placing an order, we found that integrating our clusters back into the dataset presented an opportunity for a deeper understanding of customer behavior. It suggests that even those who deviate significantly from the 'typical' customer journey may hold key insights into purchasing behaviors and thus potential revenue streams. Our findings emphasize that a customer's journey to placing an order is complex. Promotional efforts, while abundant for 'Engaged Customers', do not always translate directly to completed journeys. This suggests a potential misalignment between marketing efforts and actual customer conversion, underlining an area for Fingerhut to optimize. For 'High Value Customers', the stabilization from 'place_order' to 'order_shipped' is much clearer, reinforcing the value of targeting these segments.

In addition, our XGBoost model was adept at predicting the likelihood of a customer reaching order completion, achieving an accuracy rate of 94%. It can hopefully be used on future customer data, or be expanded upon for different Fingerhut stages beyond the FreshStart program. While the model exhibited impressive performance, enhancing it with additional numerical features such as average session duration or session frequency per customer could offer deeper insights.

In identifying that most customer dropoff occurred while shopping, we dialed in on Fingerhut’s methods of communication with customers, particularly catalog emails and promotions. And, while both are aimed at incentivizing a customer in returning to their journey, they have different rates of efficacy.

Catalog emails, for example, were shown to have a higher customer order rate when specifically targeting users slightly later in their journey and when not waiting too long before sending prospecting mail out. This is clearly demonstrated in the histograms when the

distribution of journey steps for users who completed orders was clearly later (hovering around 10 to 42) in comparison to users who did not complete orders (with journey steps ranging from 5 to 24). In addition to this, we can see in the histogram for time between first prospecting and last user event before there is a difference in the distributions as users who completed orders were typically sent prospecting mail between 5 to 26 days after, whereas users who didn't complete orders were sent prospecting mail between 16 to 39 days later. We then used a Kernel Density Estimation technique to determine that the optimal time to send prospecting mail is within the week after a customer has reached steps 6 to 19 in their journey. Our recommendation to Fingerhut is to follow these intervals as a guideline and continue to refine the adjustments as new data becomes available.

Promotions, on the other hand, proved most effective the earlier they were sent. By analyzing the customer return rates based on promotion delay, it's apparent that those sent within the first six hours of inactivity had almost a 95% likelihood of success, with this rate consistently dropping afterwards. And, by highlighting the promotions involved in successful orders, it's apparent that stagnation in a customer journey can quickly turn to dropout. However, we have no external data on the promotions themselves, and would recommend that Fingerhut not only experiments with promotion timing, but with deadlines, personalization, and marketing as well.

Overall, Fingerhut's innovative approach to online retailing and its commitment to financial inclusion truly sets it apart as a trusted partner for individuals seeking to improve their financial standing. We hope that our recommendations not only aim to increase conversion rates and customer retention but also support Fingerhut's mission of empowering individuals to (re)build their credit and achieve their financial goals.

Current stage at each journey step, with customers that were successful (in blue) and unsuccessful (in orange) at completing a purchase. Stages are presented at steps 1, 2, 3, 5, 8, 13, 21, 34, 55, but the full animation is in our source code. Step number is visible above the sliding animation bar.