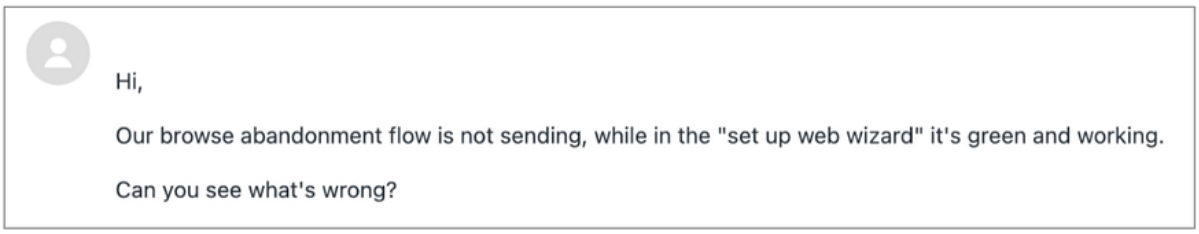
Developing our first anomaly detection algorithm

Author: Olof Jacobson

Claps: 248

Date: Jan 31

This support ticket was filed with Klaviyo in July 2021:



The customer had set up a flow to automatically email their contacts, but it stopped working, hurting their sales. In late 2021 we set out to understand why our customers would sometimes run into this problem. Our investigation resulted in Klaviyo building an anomaly detection system that detects and notifies customers when similar issues occur.

Building this system was challenging because it had to work for all Klaviyo customers despite their massive differences in scale. The system had to be able to evolve with gradual changes in customersâ€™ businesses but still raise the alarm when something changed too drastically. It also had to handle the wide range of different trends, seasonalities, and business quirks unique to individual Klaviyo customers.

Introduction

Iâ€™m a data scientist at Klaviyo and my job is to build product features that help customers better understand and make decisions based on data.

In this blog post I tell the story of how my team and I developed the algorithm that forms the core of our anomaly detection system. I will show problems we encountered along the way, the tradeoffs we made, and how we modified a standard algorithm for our use case.

Background

A flow in Klaviyo is a sequence of actions that are performed automatically when certain conditions are met. A flow has a trigger condition that specifies when the logic in the flow should start being executed.

As an example, a flow may be set up to message customers for purchases they make at an e-commerce store. The trigger in this example would be someone placing an order, and the action taken by the flow would be to message that customer thanking them for their purchase.

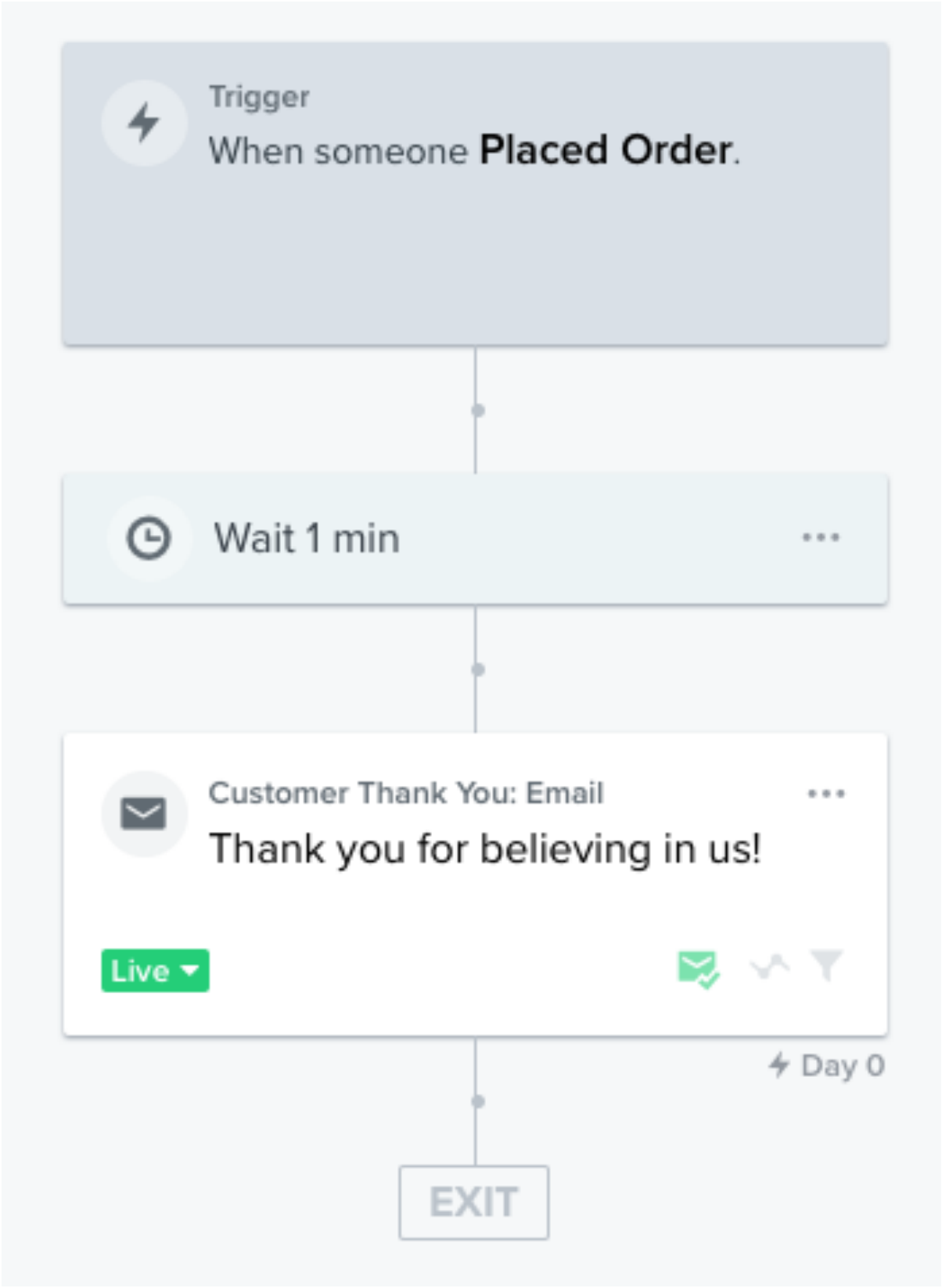


Fig 1: An example of a flow that thanks customers for ordering in Klaviyoâ€™s flow builder.

When a flow is set up, one of the fundamental expectations is that it will keep sending messages according to the configured logic. Our customers view them as â€œset and forget.â€ You set them up once and then they continuously bring in value over time. Unfortunately, there are situations when this doesnâ€™t happen and a flow stops sending messages even though the flow itself has not changed.

How did we know this was a problem that needed solving

Our customer success managers, who work with our larger accounts, were the first to ask for a solution. From time to time, when reviewing accounts, they would come across flows that had stopped sending out messages. Or worse, angry customers would contact them asking why they hadnâ€™t noticed that a flow stopped sending.

We reviewed a large set of flow related support tickets, like the one at the start of this blog post, and found that flows that unexpectedly stopped sending messages were a problem for customers of all sizes. The problem was rare, but it could often take a long time for customers to notice that something was wrong, months in some cases.

First pass

When customers contacted customer support it was because they noticed their flows were not sending as expected. Our initial approach was to detect anomalies in the number of flow messages sent by flows since that appeared to be what our customers were monitoring themselves. Our thinking was that if flow messages were what customers cared about, that was also what we should be monitoring.

Although all the customer problems reported were about flows that stopped sending, we set out to detect both abnormally low and abnormally high message send volumes. Because why not? If something abnormal was going on, wouldnâ€™t our customers want to know about it?

However, we quickly ran into issues with this approach. We noticed that it was common for customers to have large positive spikes in the number of daily messages sent for their flows.

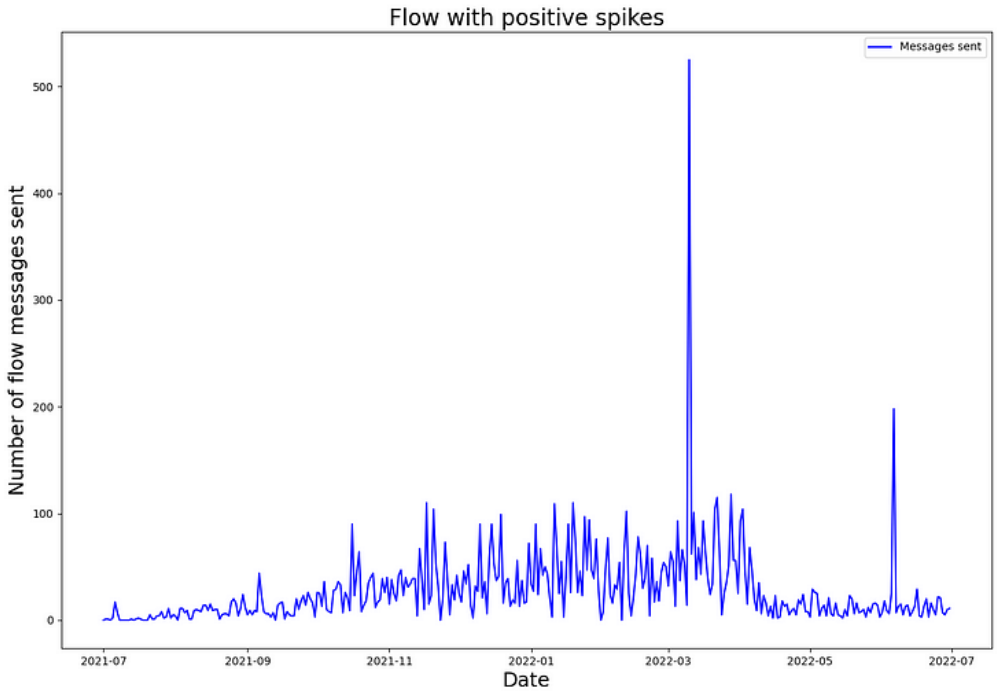


Fig 2: The daily count of sent messages from a Klaviyo flow. There are two large positive spikes in the data.

For example, this type of spike could be caused by the customer sending a promotional campaign to a wide audience. That action would drive more of their contacts to visit the website, browse products, add products to their cart, and so on. As a consequence, all the flows triggered by these events would start having higher volumes of messages. Although the flow send numbers in these cases were abnormally high, this seemed to be expected behavior from the clientâ€™s perspective, and not something they needed to be alerted to.

We also encountered many flows where smart sending was enabled. When customers sent campaigns to the majority of their contact lists, their flows with smart sending turned on would completely stop sending messages for some time, but that was expected, and not something they needed to be alerted to.

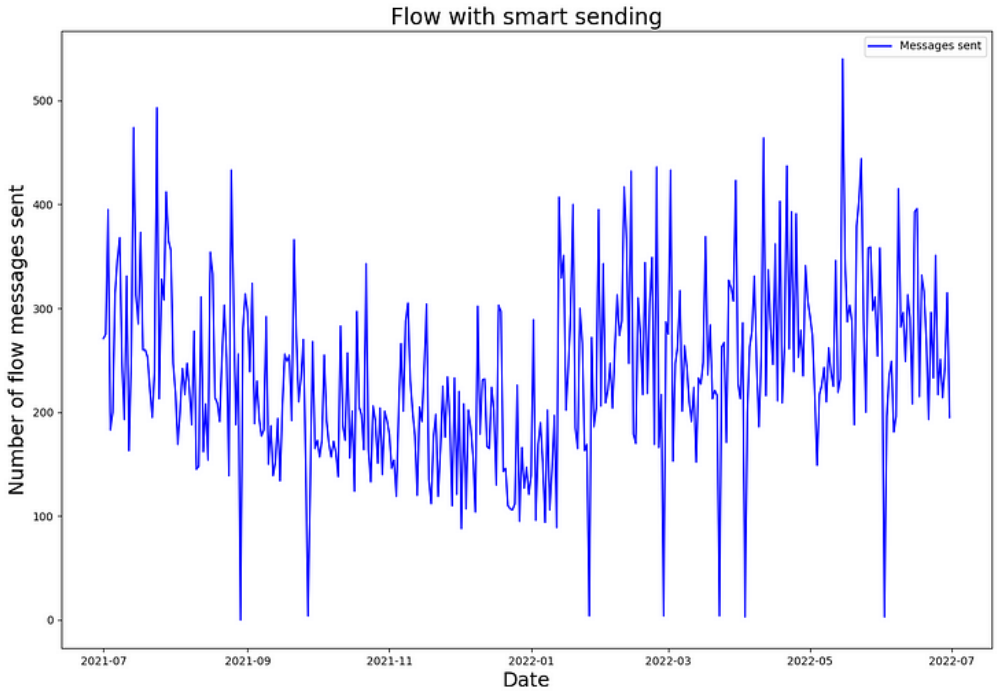


Fig 3: The daily message send count for a Klaviyo flow with smart sending enabled. The drops to 0 are not in fact anomalies. They occur when the account sends campaigns to all of their contacts and they want flow messages to be blocked to avoid over communicating.

We also encountered many flows that were configured to send messages on only certain days of the week. That was definitely not something that customers needed to be alerted to since the customers themselves must have intended that functionality when configuring the flow. Nevertheless, flows of that kind caused many issues with various anomaly detection techniques.

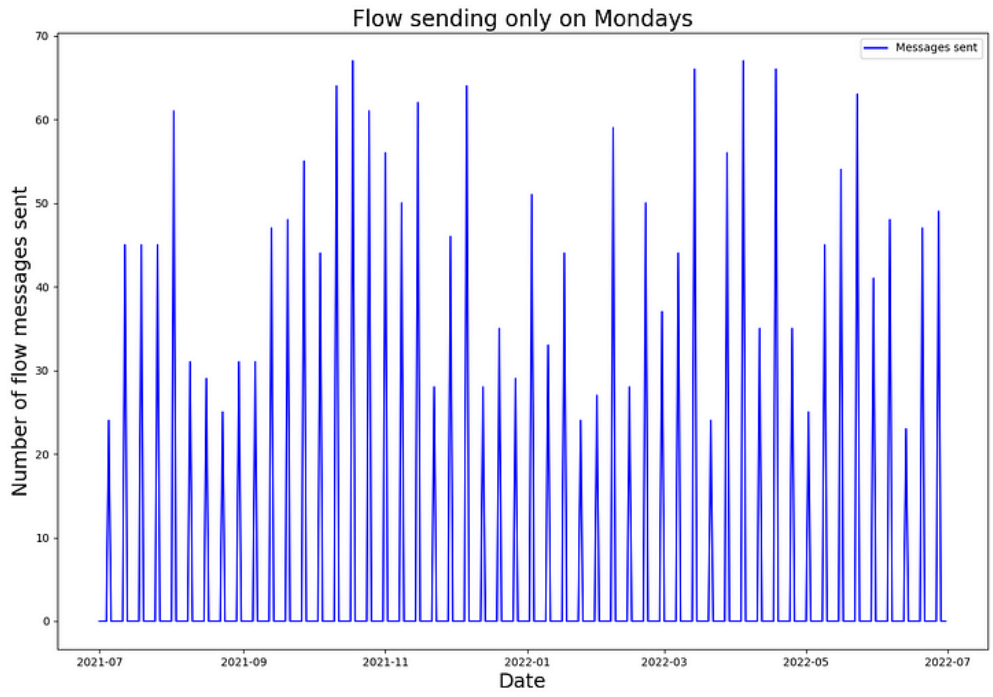


Fig 4: The daily message send count for a Klaviyo flow that only sends on a particular weekday.

All of these factors made anomaly detection more difficult.

Changing scope

One of the first changes we made to the scope of the project was to ignore all the positive spikes in daily event counts. Although we might want to detect and notify our customers of that kind of activity in theory, we hadnâ€™t seen any example of customers asking us to detect elevated sending in support tickets, nor had we heard of this request from customer success managers.

We also realized it would be easier to detect true anomalies if we looked at how often a flow was triggered, rather than how often it sent messages. By looking at triggering events we didnâ€™t have to worry about smart sending or intentional flow delays impacting the results. This approach would also let us give customers a clearer indication of where the root problem was.

Hereâ€™s why. If the metric that is used to trigger a flow abruptly stops getting events then, assuming the decrease is not explainable as random variation, something has likely gone wrong with an external integration or an API call.

On the other hand, if a flow stops sending messages, there could be a wide range of root causes. In addition to the reasons mentioned above, it could be that something was changed in the flow itself. Or even that some customer property that the flow depends on stopped getting updated.

While we thought that it might be valuable to notify customers of all of these issues, some of them proved to be difficult to reliably detect. And even if we could detect all of them with a single algorithm, it was going to be challenging to guide customers through fixing the issues since there was such a wide range of possible root causes. Instead, we decided to focus on one of the issues, build out a solid anomaly detection system for that particular issue, and then break out the other problems into separate projects that we could tackle later.

Labeling metrics

We were able to learn a lot about how well various anomaly detection techniques worked by running them on a dataset of production data and checking which trigger metrics were classified as anomalous and non-anomalous.

However, in order to be able to quantify how different methods performed against each other we needed to be more systematic. We created a dataset of manually labeled anomalies by going through and plotting thousands of triggering metrics for real Klaviyo accounts, and for each one labeling if and where the metric was in an anomalous state.

The vast majority of metrics were non-anomalous and easy to label. However, there was a significant chunk of metrics that were more difficult. The two most common types of difficult-to-label metrics had either low daily event volume or only had events during a short time period.

Low daily event volume made it difficult to judge whether event volume tapered off naturally or if something more problematic had occurred.

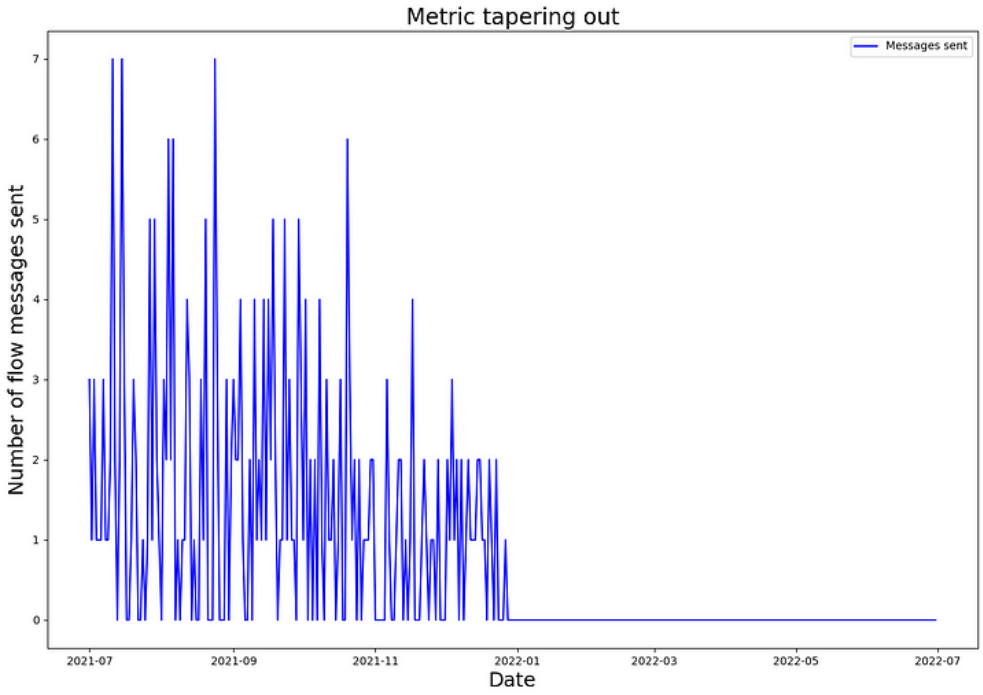


Fig 5: Daily messages sent by a Klaviyo flow. The number of messages is low and gradually decreases to 0.

For metrics that only had events for a limited time period, the question of whether or not they were anomalous became confounded by the question of why they started receiving events in the first place.

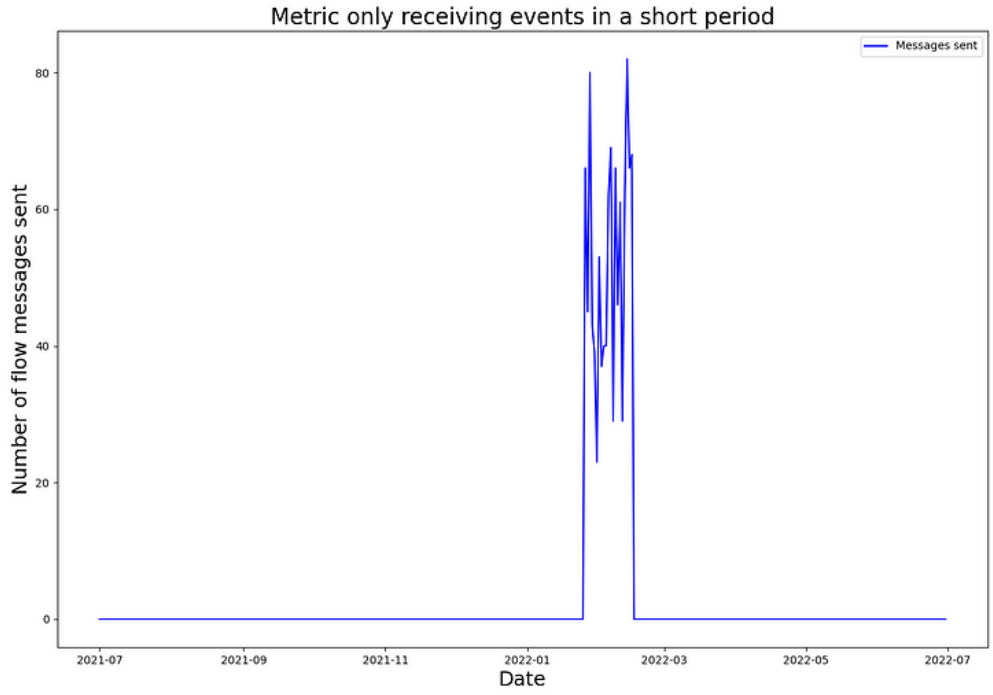


Fig 6: Daily messages sent by a Klaviyo flow. The flow starts sending messages abruptly for a limited period and then stops sending again.

For the metric above, for example, when judging whether or not the decrease to zero daily event volume should be considered anomalous, itâ€™s reasonable to first ask what made the triggering event start out just a few days prior. Were the account owners just testing something out? If that was the case then we wouldnâ€™t want to notify the customer.

Second attempt

Armed with our manually labeled dataset, we proceeded to compare various anomaly detection algorithms against each other.

We first attempted to develop simple rules for what should count as an anomaly. One such rule was that the event count on a given day was considered anomalous if the event count was zero and the preceding mean was greater than a number X which we would experiment setting to different values. A low value of X meant that the alarm would be sensitive and trigger easily, but would generate a lot of false positives. A high value of X resulted in fewer false positives but made the system unable to detect anomalies for low event volume metrics.

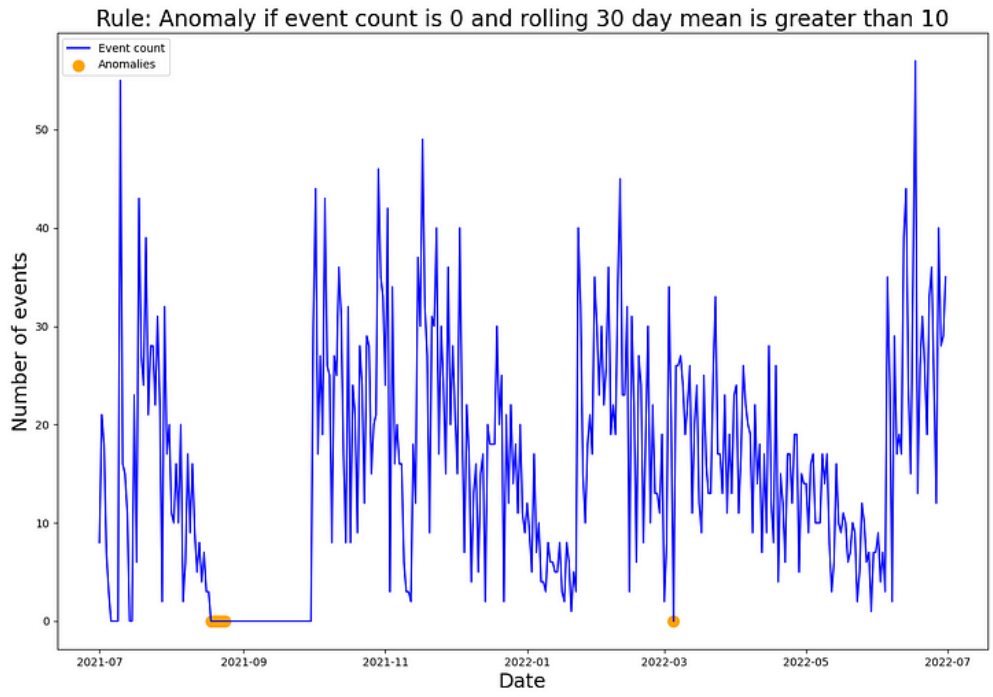


Fig 7: Daily viewed product events in a Klaviyo account. A simple rule based anomaly detection system has been applied to the data and anomalous days, according to the rule, have been highlighted on the graph.

The difference in daily event volume between different Klaviyo customers can be massive. Large Klaviyo accounts may have tens of thousands of flow triggers on a given day. And small accounts with limited customer activity might have daily event counts in the single digits. Ideally, we wanted an anomaly detection system to be able to help any Klaviyo customer regardless of their size and event volume.

After the rule based approaches, we experimented with rolling window techniques where a past historical daily mean and standard deviation were calculated based on preceding data points in a window of fixed size. The reasoning behind using a rolling window approach was that we needed a system which could adjust over time to new behaviors in a flow, but also mark a flow as anomalous if abrupt changes occurred. By windowing the data, the mean and standard deviation would adjust to changes in the daily event volume.

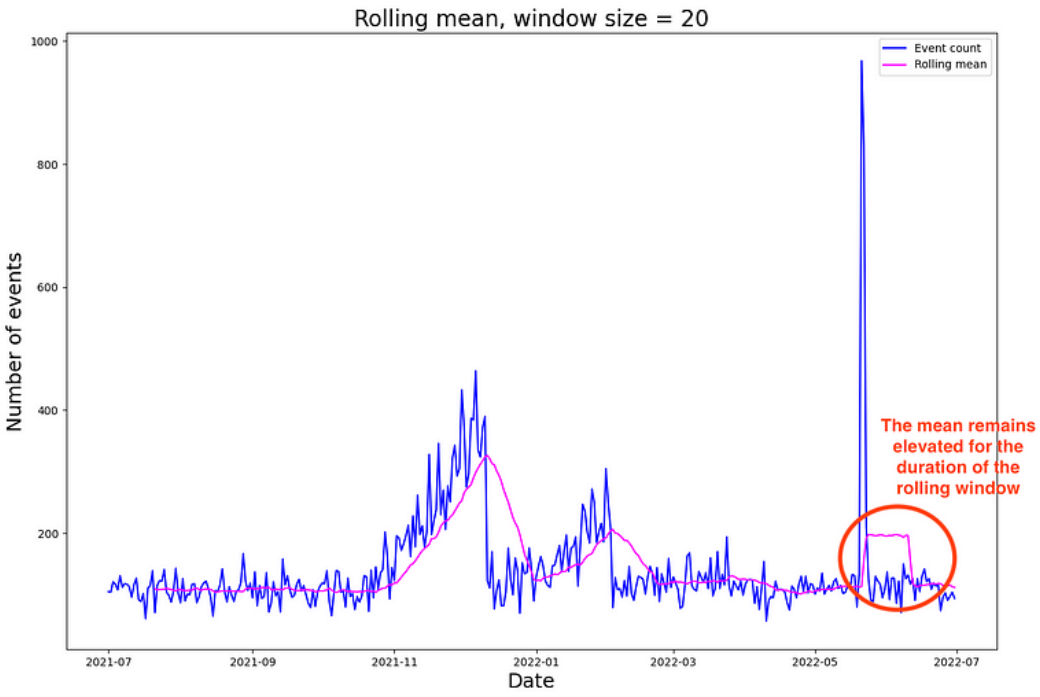


Fig 8: Daily viewed product events in a Klaviyo account. A rolling mean has been applied to the data and is shown in the graph.

We eventually transitioned to exponential smoothing, both single and double exponential smoothing, instead of rolling windows. In exponential smoothing techniques, averages and standard deviations are calculated with exponential weighting where the most recent data points are considered more important and thus given larger weights.

The reasoning behind moving to exponential smoothing was that with rolling windows it is possible that data points far in the past have an outsized impact on the calculations. It was commonly observed that the previously mentioned large positive spikes in event data often lead to undesirable non-smooth behavior of the mean.

Compare the following single exponentially smoothed mean (Fig 9) to the previous rolling mean (Fig 8) for the same metric. The large positive spike towards the end of the data causes a sharp increase in the mean in both cases. For the rolling mean the contribution of the positive spike remains constant for 30 days, whereas the exponentially smoothed mean decreases gradually. In most cases we preferred a mean that decreased gradually.

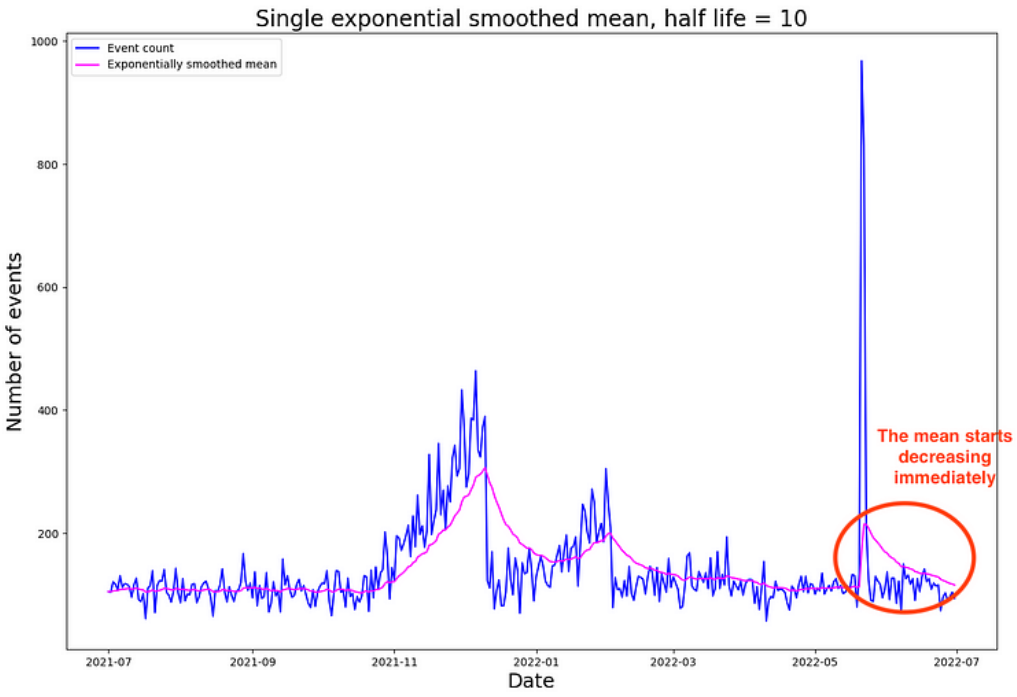


Fig 9: Daily viewed product events in a Klaviyo account. An exponentially weighted mean has been applied to the data and is shown in the graph.

We tried out a few standard rules for how to classify a metric as anomalous or non-anomalous based on preceding mean and standard deviation. For example, any value more than X standard deviations away from the historical mean should be considered anomalous.

In our case we were only looking for decreases in event volume so we only considered a metric anomalous when an observation was more than X standard deviations below the historical mean. This type of system is generally referred to as a control chart system and the value of the threshold below which a metric is considered anomalous is referred to as the lower control line (LCL).

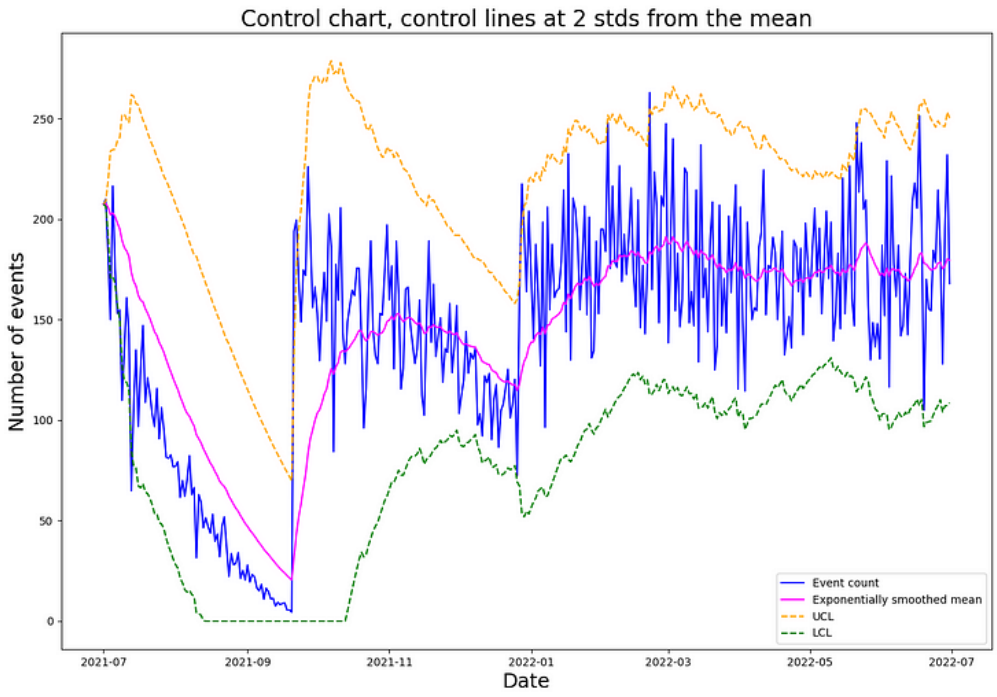


Fig 10: Daily viewed product events in a Klaviyo account. An exponentially weighted mean has been applied to the data and is shown in the graph. Lower and upper control lines are also shown based on an exponentially weighted standard deviation.

One problem with the rule described above is that any change in the daily event volume of a metric that is less than X standard deviations outside of the mean cannot be detected.

It is common for control chart systems to implement not just one rule but several. If a metric is more than three standard deviations below the mean on a given day we might be able to say immediately that it is anomalous. A metric that is just two standard deviations below the mean might be a strong indicator that something is wrong, but not quite strong enough to trigger an anomaly. However if we observe values that are two standard deviations outside of the historical mean for three subsequent days, that might be enough to consider the metric anomalous.

By stacking different rules like this, it is possible to construct an anomaly detection system that is both quick to react to large changes in daily event volume, but also able to wait for extended periods for changes that are less extreme.

We attempted to construct rules like this for both the simple rule based systems described earlier as well as the control chart systems. This improved performance of all the methods but it was still not at satisfactory levels.

Standard deviation compared to mean tends to be relatively high for low volume metrics. So for a low volume metric to be considered anomalous, the lower control line must be a small multiple of the standard deviation below the mean. Otherwise it risks dipping below zero, at which point anomalies can no longer be detected.

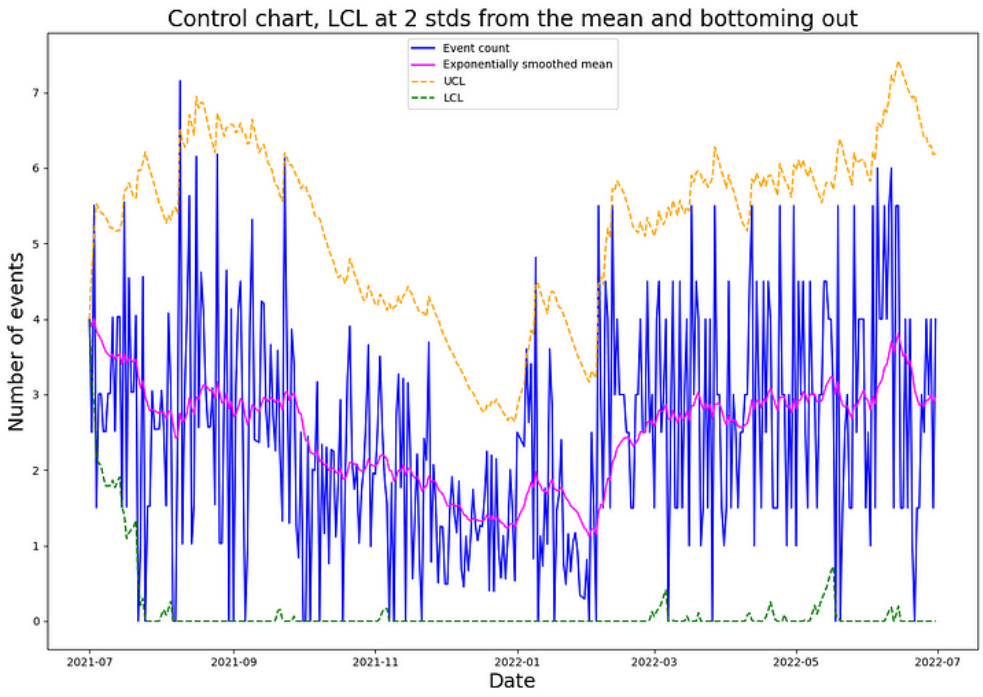


Fig 11: Daily viewed product events in a Klaviyo account. An exponentially weighted mean and control lines are shown. For large parts of the year the lower control line (which is not allowed to take on negative values) bottoms out at 0. This means that the system is unable to detect anomalies.

Although initial stacking of control chart rules lead to performance improvements, we were worried about the complexity of the resulting system. The huge variety in event volume across different Klaviyo accounts, as well as the characteristics of different triggering metrics, meant that we would potentially need to have a very large number of rules to cover all the cases where we wanted to alert.

Cusum

We started experimenting with another technique called cumulative sum control chart (CUSUM). This is a delightfully simple yet powerful method. It retains most of the power of the multiple stacked control chart rules, without all the complexity.

In the CUSUM method, an observation that is outside of an expected bound adds a contribution to a factor called the cumulative sum. The size of the contribution depends on how far outside of the expected range the observation is. The system considers the metric anomalous when the cumulative sum value breaches a predefined threshold.

Informally, we can say that any observation which is surprising adds the amount of surprise to a cumulative surprise value. When the system has accumulated enough surprise, it declares the underlying data to be anomalous.

The elegance of the CUSUM method is that it can trigger an alarm quickly for very surprising observations, and wait longer to trigger for sequences of mildly surprising observations.

Formally, the standard lower limit CUSUM algorithm can be defined by these two simple rules:

For a process with mean ð‡ and standard deviation ðŽ, the lower cumulative sum Sâ™ for observation xâ™, is given by

$$S_0 = 0$$

$$S_{n+1} = min(0, S_n + \frac{(x_{n+1} - \mu)}{\sigma})$$

The CUSUM method immediately yielded better results than the other anomaly detection methods that we had tried. After the hyperparameters of the system had been tuned, we went through and reviewed all the misclassified examples to find out if we could identify patterns of when the algorithm struggled.

The first system improvements that we attempted were based on the realization that daily event counts of zero were more likely to represent an anomaly in a metric than non-zero counts.

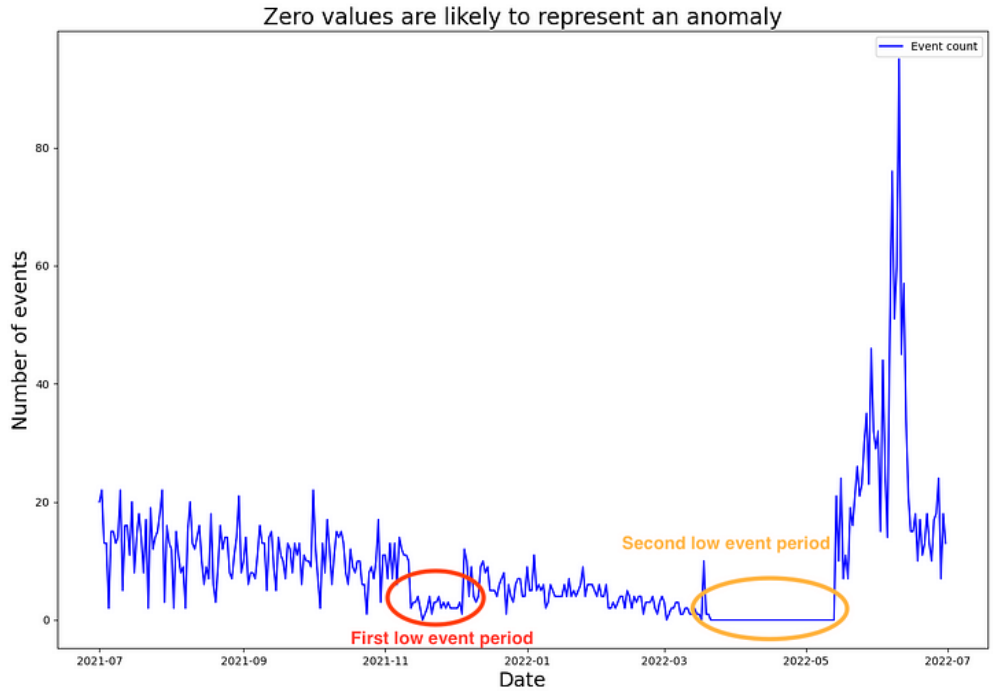


Fig 12: Daily viewed product events in a Klaviyo account. Note the two highlighted periods in the graph. Both periods have unexpectedly low daily event counts compared to previous data. However the second period is more indicative of an underlying data integration or ingestion issues than the first.

The above graph is a good example of a metric that has at least two periods of abnormally low event counts. The first period is in late November to early December where event count is low but not zero. And then in late March to May 2022 there is a period of no events at all.

By implementing separate reference values and threshold factors for these different types of observations, we made the system more sensitive to zero values.

After the system had been made extra sensitive to zero event counts we noticed some problems for metrics with daily event volumes that were close to zero for extended periods. See the graph below (Fig 13) for an example. From approximately the middle of the graph, the daily event count remains low for many months. Towards the end of the graph the metric receives no events at all for an extended period. This looks like the metric naturally tapers off and not like something we need to alert the customer to.

However, not only had the alarm been made extra sensitive to observations that were zero, the standard deviation had also decreased to a low value. Since the standard deviation is used in the denominator in the CUSUM method, the low value of the standard deviation inflated the results of the equation. We saw that our anomaly detection system generated a lot of false positives alerts for this kind of metric.

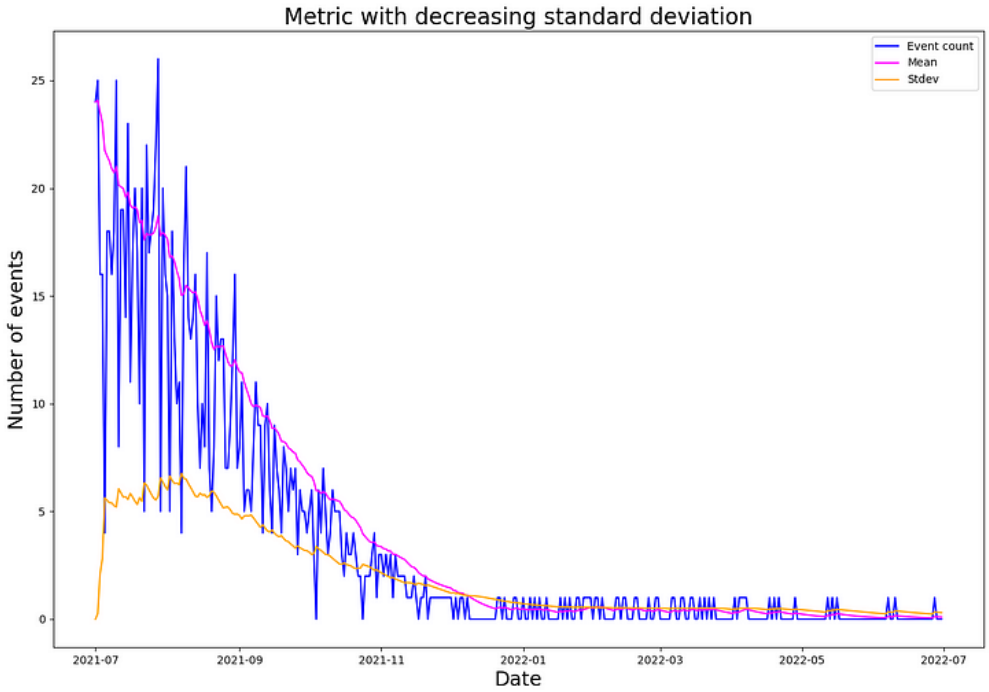


Fig 13: Daily viewed product events in a Klaviyo account. The event counts slowly approach 0 causing both the exponentially weighted mean and standard deviation to become low. The low value of the standard deviation can cause some issues in the CUSUM calculation.

We also noticed that the large positive spikes in many triggering metrics inflated both mean and standard deviation and made it harder for alarms to trigger.

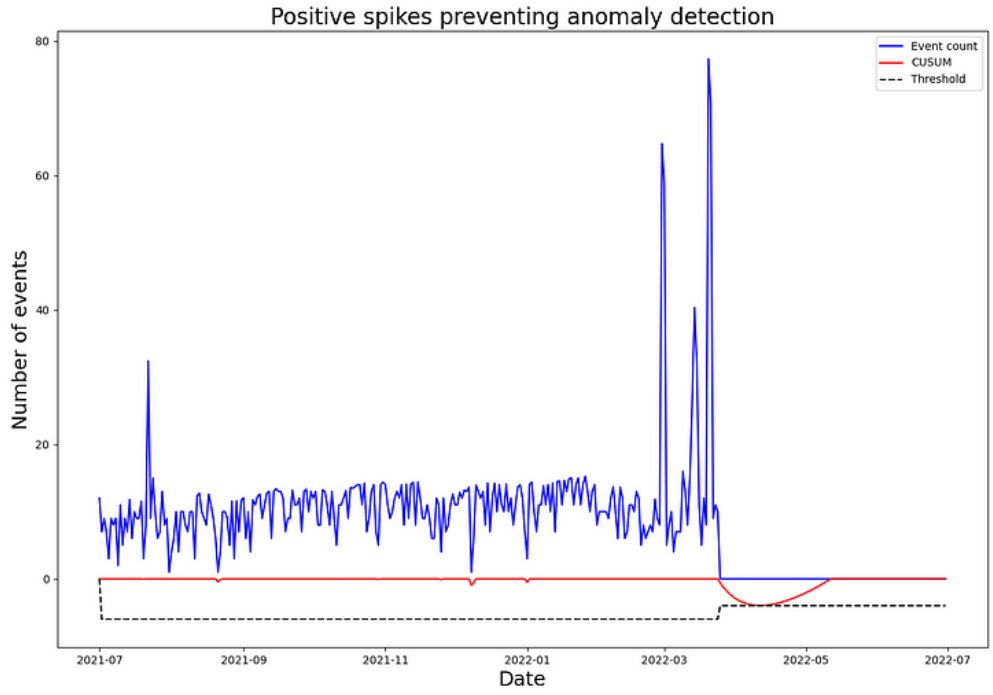


Fig 14: Daily viewed product events in a Klaviyo account. A notification is triggered if the CUSUM value breaches the threshold. The large positive spikes that precede the drop to 0 daily events, inflate the standard deviation just enough so that the CUSUM value does not breach the alarm threshold.

We addressed all of these issues by making small modifications to the base CUSUM algorithm. The instability caused by low standard deviation was resolved by implementing a minimum standard deviation used in calculations. The adverse effects of large positive spikes were diminished by limiting how much impact a daily value much higher than the mean was allowed to have on the CUSUM calculations. After each change made to the algorithm we re-reviewed all of the misclassified examples in the dataset. Each change led to performance gains, admittedly at the expense of making the algorithm slightly more complicated.

At the end of the process, almost all of the misclassified examples were ambiguous, the kind of examples that had caused us headaches during the manual labeling process. This was a great sign as the CUSUM method had essentially become as good at classifying metrics as the output of our own manual labeling. We considered this system to be a success and good enough to start using in production.

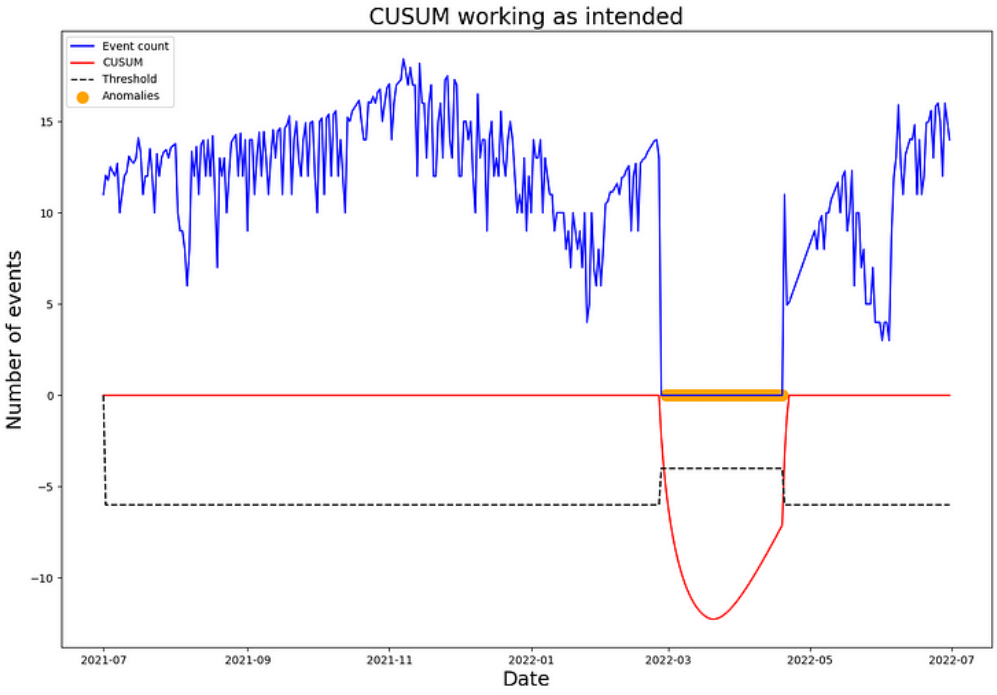


Fig 15: Example of the CUSUM algorithm working as intended. When the underlying metric stops receiving events the CUSUM value quickly accumulates until it breaches the notification threshold. Note also how the threshold is set to a higher value when the event count is 0. This makes the anomaly detection system more sensitive and quicker to trigger.

The response

When we went live with the anomaly detection feature we were eager to see how customers would interact with the new notifications. Fortunately, we saw the behavior we were hoping to see. Customers who were shown anomaly detection notifications interacted with them and best of all, fixed their underlying issue and started receiving events and sending flow messages again.

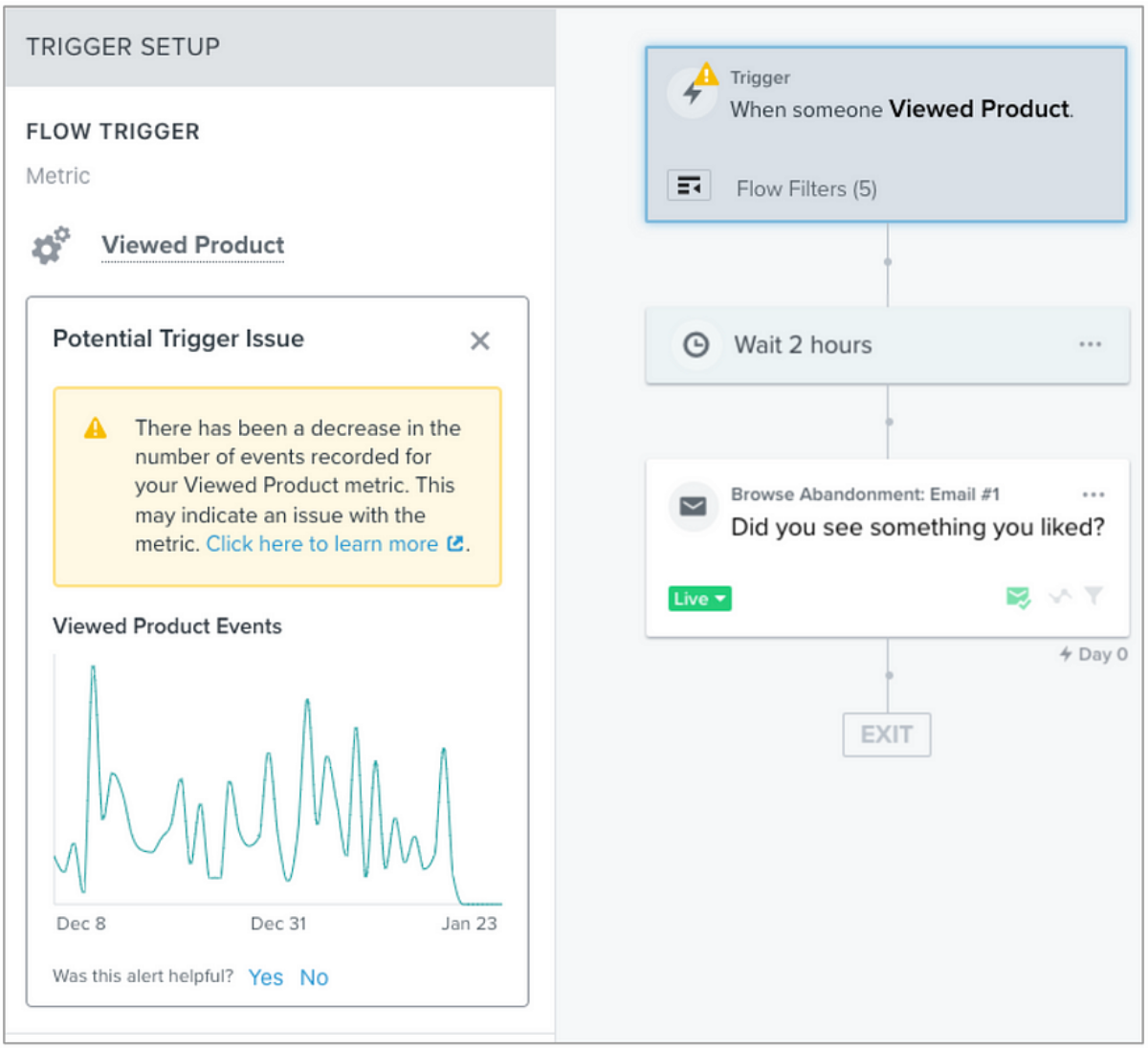


Fig 16: Production UI for a customer with an anomaly in their Viewed product metric.

To help us understand if we had hit or missed the mark, we gave customers the option to provide feedback on whether or not the notifications were useful. In the nine months that the feature has been live, 92% of respondents marked the alert as helpful.

Closing remarks

Anomaly detection is complicated, especially when anomalies can take many different forms and there isnâ€™t always a clear answer for what it is that we want to detect. The development process described in this article is what led us to build the first anomaly detection algorithm that was productionized and used to message customers. Although we consider the first algorithm a success, itâ€™s only a start. Our team is working hard at refining and expanding the coverage of anomaly detection in Klaviyo. If you have not yet received a notification from us, that probably means your Klaviyo account is working as expected =)

Want to hear more about anomaly detection at Klaviyo? Listen to me and my colleagues on Klaviyo Data Science podcast episode 29: Detecting the Unexpected.