Finding Fraud Events using Time Series Anomaly Detection Methods

Weronika (Nika) Jusczak, Staff Fraud & Risk Analyst at Intuit

Research Question

Topic: Detecting Fraud in Financial Services Applications over time

Question: What statistical methods can differentiate fraud events from other time series anomalies and behaviors?

Motivation.
As a leader in Risk Analytics at Intuit, finding new ways to identify and monitor fraudulent behavior is critical in mitigating losses. We want to be able to not only stop fraudulent transactions and activity, but detect and prevent fraud before a customer even transacts and losses are incurred.

E-Commerse fraud has been growing since the pandemic, with a reported $41 Billion in losses for 2022, and a projected $48 Billion in losses for 2023. (**Baluch2023Feb?**).

Data and Methodology

Getting Data.

Here is a sample of the kind of data we may (or may not) work with at Intuit.

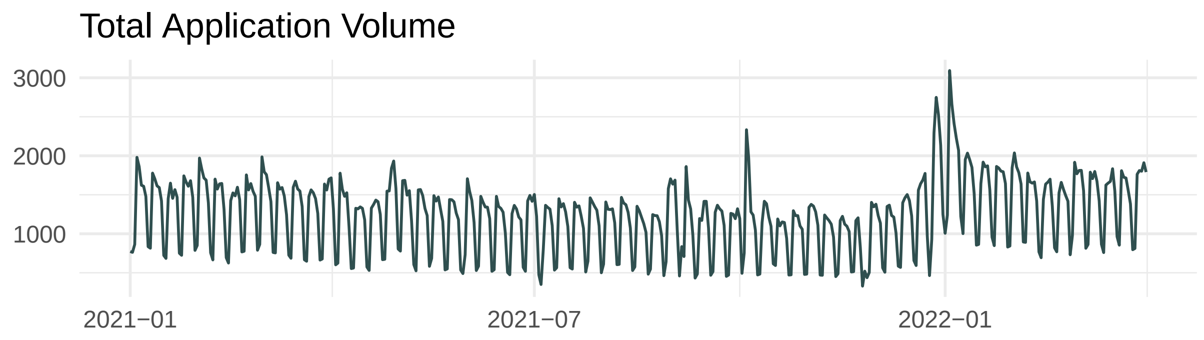
dt	product	industry	apps
2021-01-01	A	Agriculture	4
2021-01-01	A	Beauty Industry	14
2021-01-01	A	Communication Services	123
2021-01-01	A	Construction	4
2021-01-01	A	Consulting	13
2021-01-01	A	Financial Services	11

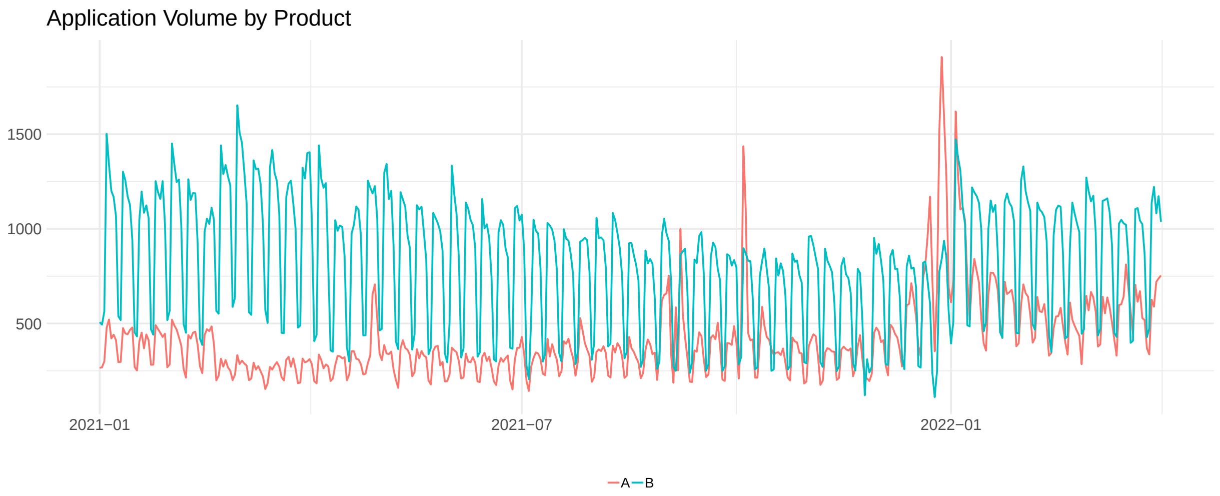
dt	product	industry	apps
2021-09-09	B	Construction	9
2021-09-09	B	Consulting	65
2021-09-09	B	Financial Services	57
2021-09-09	B	Janitorial Services	54
2021-09-09	B	Retail	104
2021-09-09	B	Transportation	137

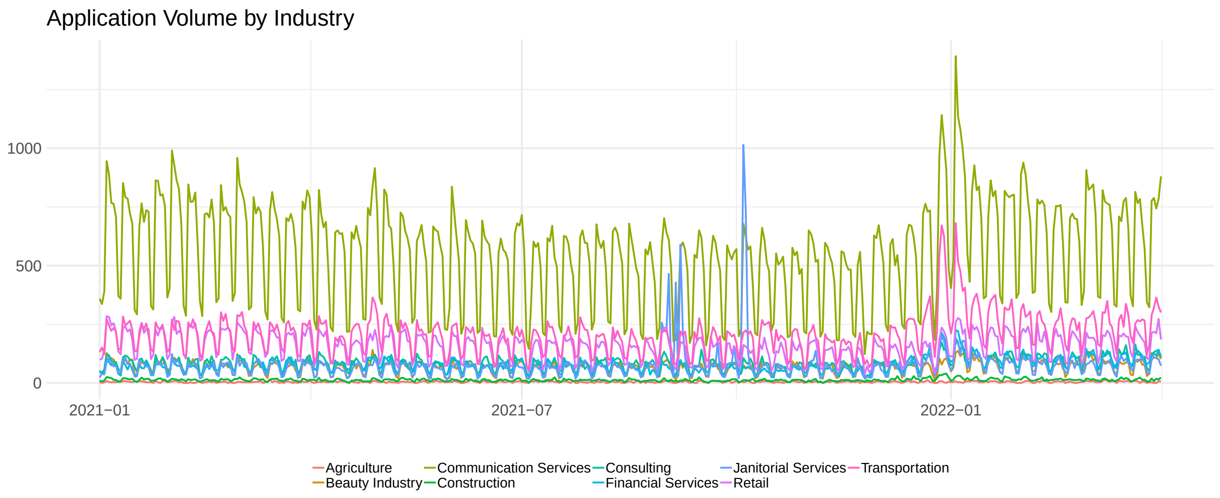
Exploratory Data Analysis

When plotting application volume over time, we see a clear weekly seasonal component, shifts in the mean over larger spans of time, as well as point anomalies.

Total application volume consists of daily aggregated applications for Product A and Product B across all Industries. If there is a vulnerability for fraud, will it appear in the aggregate or individual time series? We see similar patterns in the individual time series for seasonality and mean shifts, but differences in the point anomaly spikes.







Methodology

Decomposing time series Since we observed season and trend in the dataset, the next step is to determine whether to use additive or multiplicative decomposition. Additive decomposition means the time series is a sum of it's components, while multiplicative implies it is a product of it's components. An additive model produces better results when the seasonal variation is relatively constant, whereas a multiplicative model is more useful when the seasonal variation increases over time.

STL: STL decomposition uses LOESS to perform smoothing on the time series in 2 loops. In the inner

dt	observed	season	trend	remainder
2021-01-01	773	32.5892	1395.815	-655.40437
2021-01-02	762	-575.1652	1394.684	-57.51852
2021-01-03	864	-561.7287	1393.552	32.17637
2021-01-04	1980	306.7596	1392.421	280.81950
2021-01-05	1860	287.9398	1391.289	180.77071
2021-01-06	1622	278.4548	1390.158	-46.61286

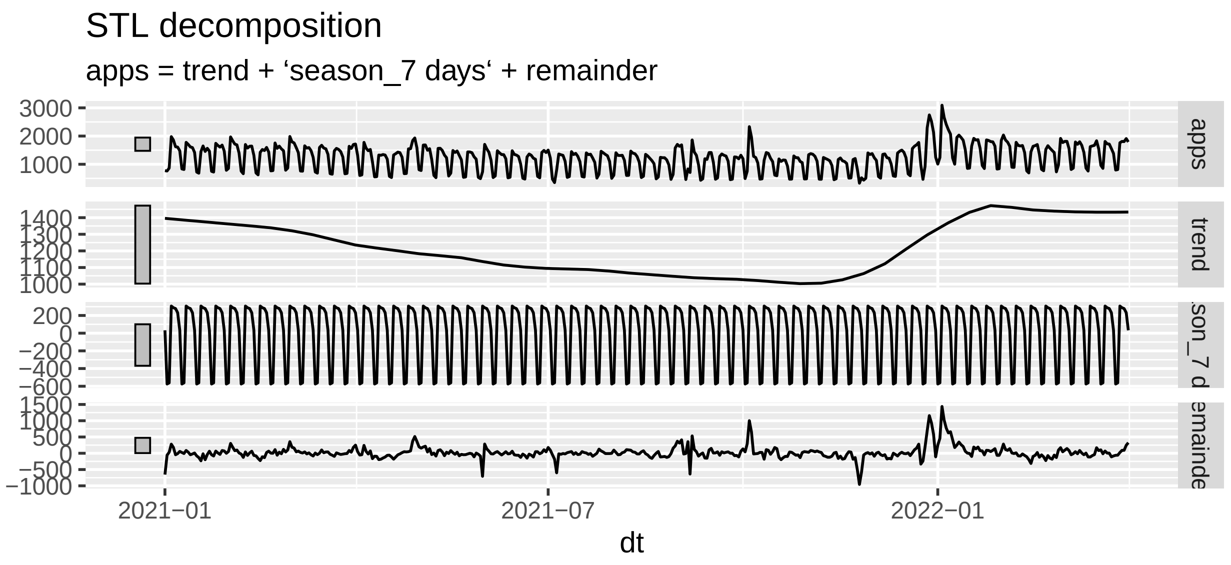
loop, the seasonal component is calculated and removed, then the trend component is found. The outer loop minimizes the effect of outliers. The remainder gives is found by subtracting the the seasonal and trend components. (**BibEntry2016Aug?**)

For the application to fraudulent activity, we are interested in the remainder.

The seasonality of applications is weekly, most application activity occurs on weekdays than weekends. Usually if a seasonal pattern exists, we see an oscillating wave pattern. In this case, we chose a 7 day period and a periodic window to find the seasonality component. The periodic window was chosen because the same seasonal pattern repeats across the entire time span of the data set.

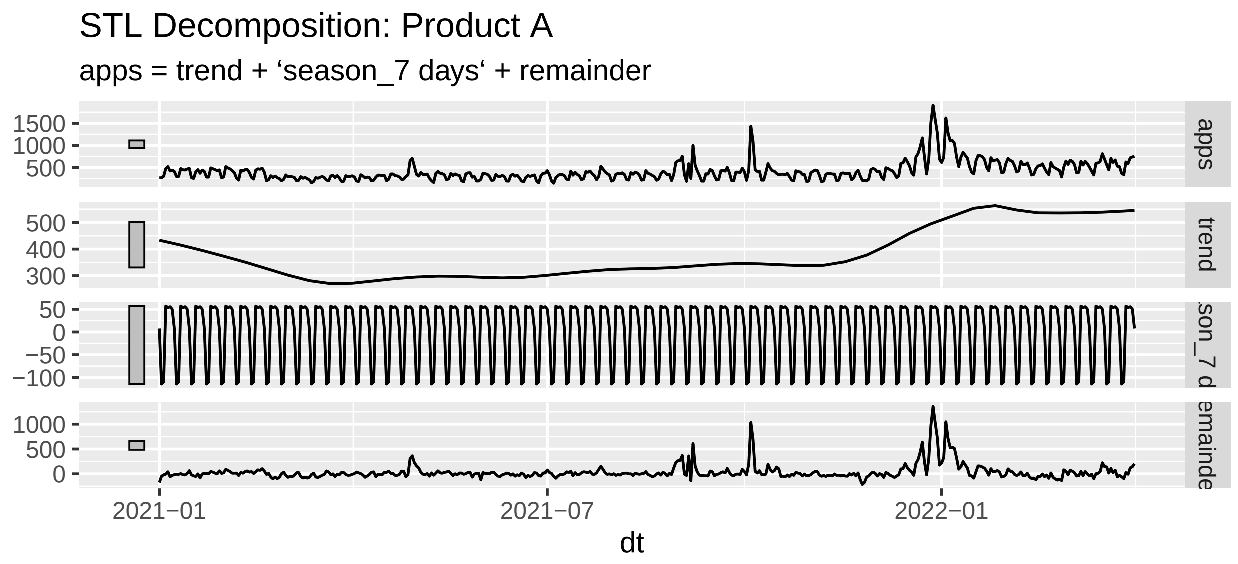
```
# grouping data as sum of applications by day
df_grp <- df %>% dplyr::group_by(dt) %>%
  dplyr::summarise(apps = sum(apps),.groups = 'drop') %>% as_tbl_time(dt)
# decomposing data
td <- df_grp %>% time_decompose(apps, method = "stl")
```

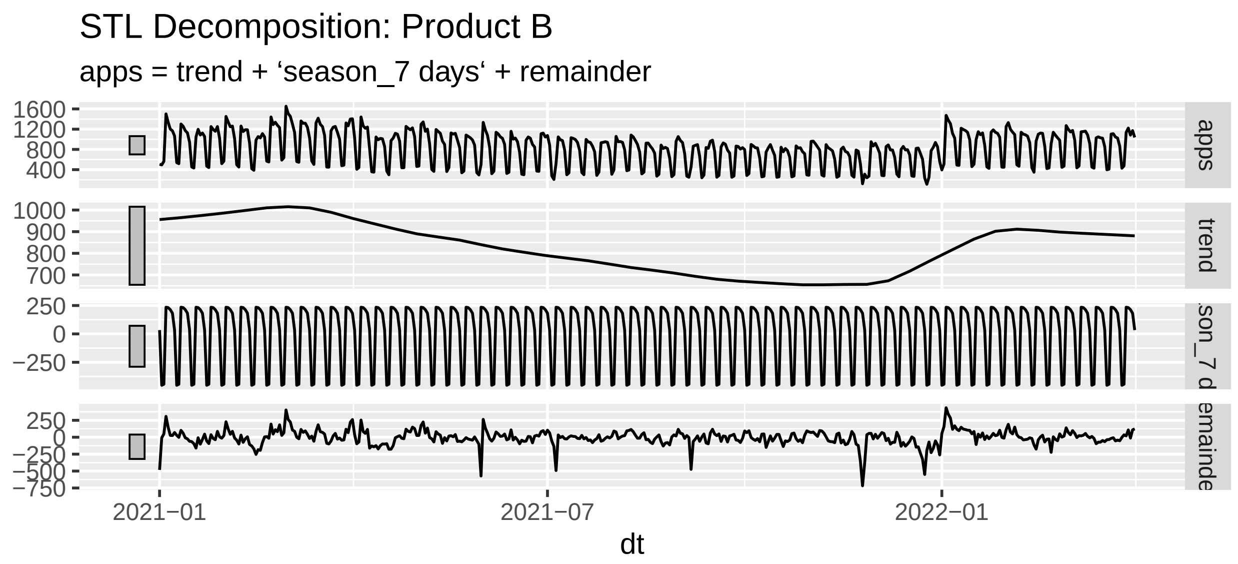
```
df_grp %>% as_tsibble() %>%
  model(STL(apps ~ season(window = "periodic", period = '7 days') + trend(window= 90.5), robust = TRUE)
  components() %>% autoplot()
```



When we look at the time series decompositions for products A and B separately,the seasonal and trend components have similar shapes as the aggregate, but the noise we see in the remainder closely mirroring the noise in the aggregate.

Most fraudulent activity at the application level is a product of bots creating several applications with the intent to defraud the company or buyer. We want to find the clusters within a time series to find the attributes that can tell us the exact products, industries, etc. the fraudulent applications are targeting. By taking out seasonal and trend components, we can compare the noise in the remainder to the aggregate to target high-risk fraud attacks.





Identifying Outliers in the STL Remainder Now that we've decomposed the time series, we want to detect the outliers. To do this, we chose to use the GESD Method.

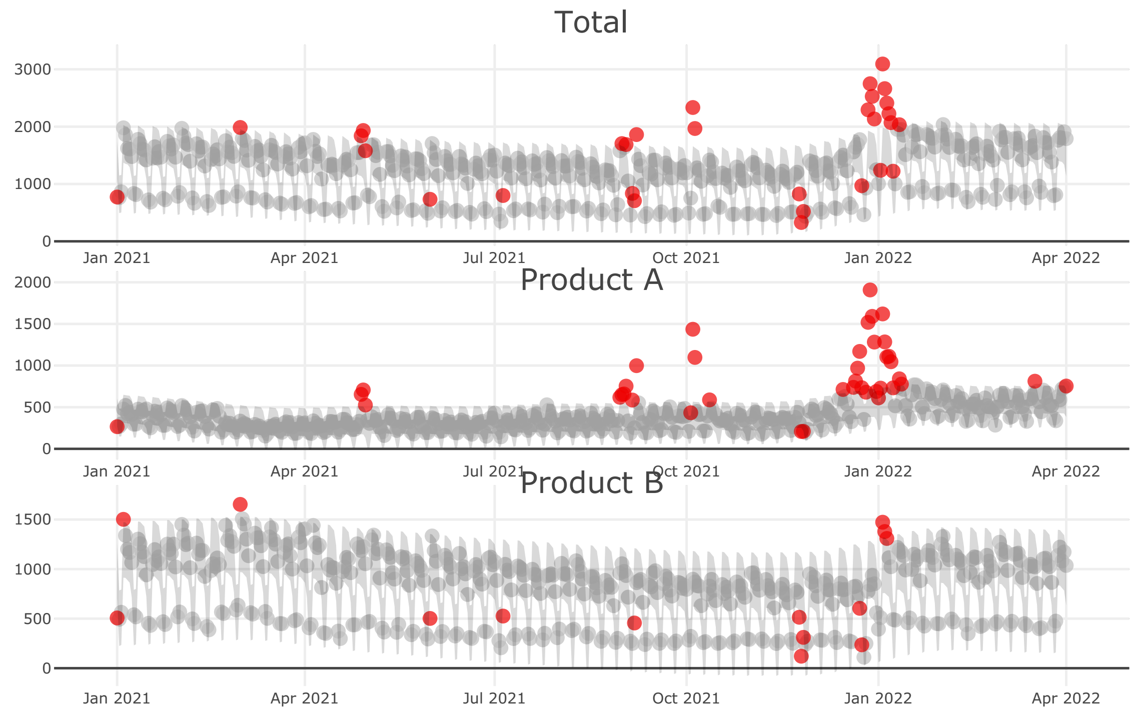
The GESD Method (Generlized Extreme Studentized Deviate Test) progressively eliminates outliers using a Student's T-Test comparing the test statistic to a critical value. Each time an outlier is removed, the test statistic is updated. Once test statistic drops below the critical value, all outliers are considered removed. (from anomalize documentation) (**Rosner1983May?**)

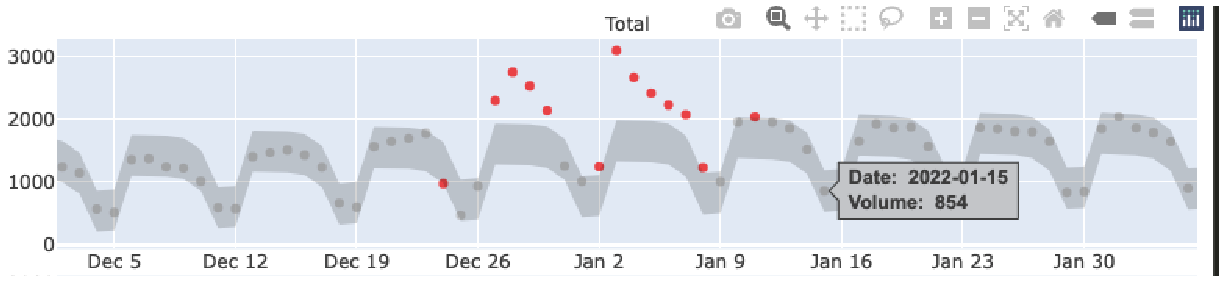
```
a <- td %>% anomalize(remainder, method = "gesd")
df_anom <- a %>% time_recompose()

knitr::kable(head(df_anom)) %>%
  kableExtra::kable_styling(font_size = 7)
```

dt	observed	season	trend	remainder	remainder_l1	remainder_l2	anomaly	recomposed_l1	recomposed_l2
2021-01-01	773	32.5892	1395.815	-655.40437	-327.8641	327.819	Yes	1100.5402	1756.223
2021-01-02	762	-575.1652	1394.684	-57.51852	-327.8641	327.819	No	491.6544	1147.338
2021-01-03	864	-561.7287	1393.552	32.17637	-327.8641	327.819	No	503.9595	1159.643
2021-01-04	1980	306.7596	1392.421	280.81950	-327.8641	327.819	No	1371.3164	2027.000
2021-01-05	1860	287.9398	1391.289	180.77071	-327.8641	327.819	No	1351.3651	2007.048
2021-01-06	1622	278.4548	1390.158	-46.61286	-327.8641	327.819	No	1340.7487	1996.432

When we plot the application volume series with the outliers and limits, at first glance it looks like there 2 distinct time series in each plot… this is a result of the weekly seasonality we observed when decomposing the series. If we look closer, we notice distinctly different behavior on weekends vs. weekdays.

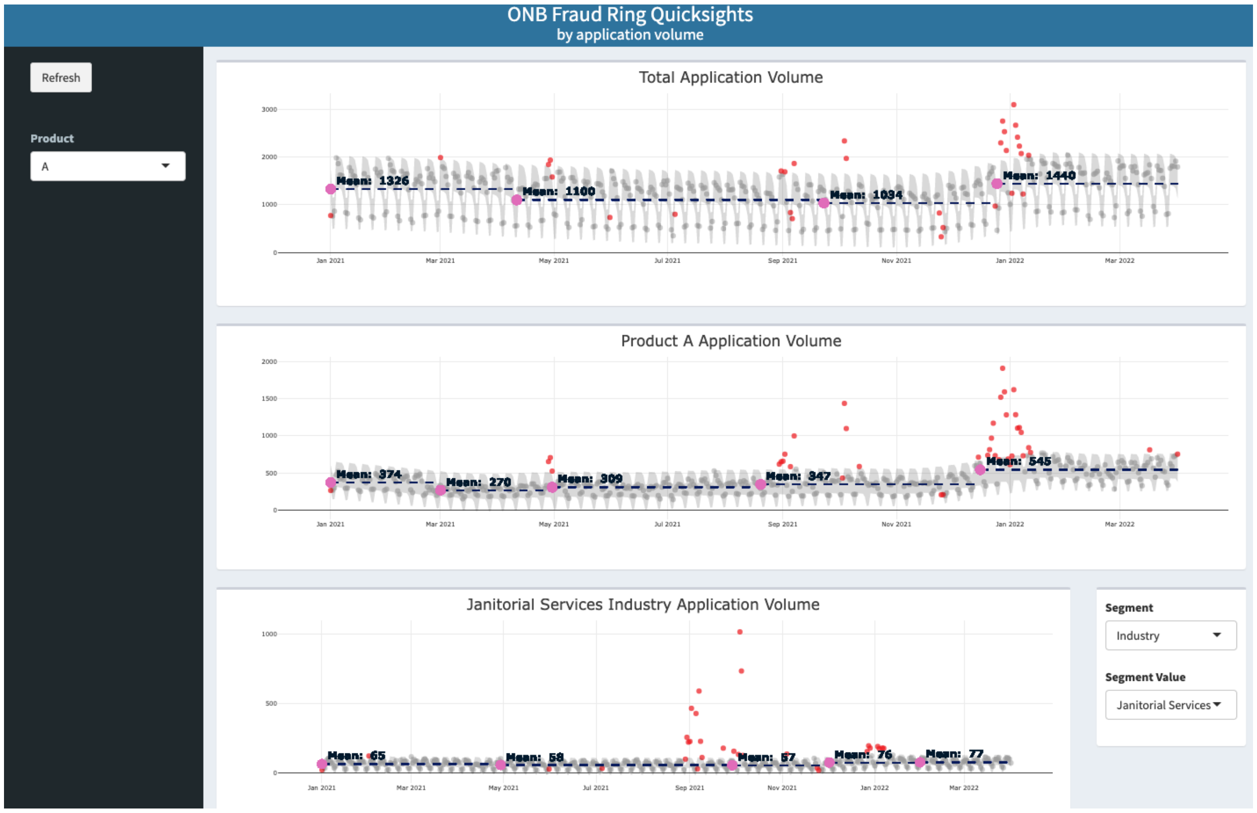




Findings and Results

We applied this method at scale, iterating across dozens of attributes, and deployed an R Shiny interactive dashboard hosted in Databricks for stakeholders within risk to monitor and quickly identify and target fraudulent behavior.

We also included tabs with insights into the trend and seasonal components, linking fraud data to marketing and customer growth data to explain large shifts in mean, and differentiate those from fraud anomalies. The simplicity of this method makes it easy to explain and share, but there are opportunities to enhance it even further…



Future Work

- Find the clusters of fraud. We were able to iterate time series decomposition and anomaly detection across several time series within the aggregate, but what are the specific profiles of applications that cause fraud events? Are there other statistical methods and tools that would could combine with this method to find what clusters of channels fraudulent applications are coming through?
- Another big opportunity we've started exploring is the link between fraudulent applications and marketing. How can we compound our savings, and optimize our marketing budget by looking at whether increases in customer growth from marketing campaigns are actually bad actors causing losses?

References