# HPCC Systems Summer Internship 2019

By Vannel Zeufack
zvanel@live.fr

# Final Report

# Theme: "Develop and Assess Unsupervised Anomaly Detection Methods"

**Supervised by:**

- Chala Arjuna (arjuna.chala@lexisnexisrisk.com)
- Xu Lili (lili.xu@lexisnexisrisk.com)
- McCall, Trish (trish.mccall@lexisnexisrisk.com)
- Chapman, Lorraine Adele (Lorraine.Chapman@lexisnexisrisk.com)

## Project description

The project is about creating an ECL Bundle encompassing several widely used Anomaly Detection algorithms, identify at least one algorithm that is parallelizable and implement efficiently in ECL. For this, I had to:

- Identify a publicly available dataset and method for assessing anomalies in the data
- Research state of the art Anomaly Detection algorithms and choose two or more for implementation
- Implement methods in ECL on HPCC Systems cluster and assess results in identifying target anomalies.

## Dataset

To work through this project, I, in coordination with my supervisors, agreed to detect anomalies in HPCC Systems log files. However, because of the lack of a sufficiently large HPCC Systems log file, we decided to use another log file with similar features as HPCC Systems log file. The dataset ended up being: ***NASA-HTTP Web Server Log***

**Description:** NASA-HTTP Web Server Log data set contains all HTTP requests to the NASA Kennedy Space Center WWW server in Florida for the month of July 1995. Records consists of the following fields:

- host making the request
- timestamp in the format "DAY MON DD HH:MM:SS YYYY"
- request
- HTTP reply code
- bytes in the reply.

**Source**: [http://bytequest.net/index.php/2017/01/03/freely-available-large-datasets-to-try-out-hadoop/](http://bytequest.net/index.php/2017/01/03/freely-available-large-datasets-to-try-out-hadoop/)

**File Type**: ASCII

**File Size**: Compressed GZ archive: 19.7 MB; Uncompressed ASCII: 205.2 MB

| ## | line |
|---|---|
| 1 | 199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/ HTTP/1.0" 200 6245 |
| 2 | unicomp6.unicomp.net - - [01/Jul/1995:00:00:06 -0400] "GET /shuttle/countdown/ HTTP/1.0" 200 3985 |
| 3 | 199.120.110.21 - - [01/Jul/1995:00:00:09 -0400] "GET /shuttle/missions/sts-73/mission-sts-73.html HTTP/1.0" 200 4085 |
| 4 | burger.letters.com - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/countdown/liftoff.html HTTP/1.0" 304 0 |
| 5 | 199.120.110.21 - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/missions/sts-73/sts-73-patch-small.gif HTTP/1.0" 200 4179 |
| 6 | burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 304 0 |
| 7 | burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /shuttle/countdown/video/livevideo.gif HTTP/1.0" 200 0 |
| 8 | 205.212.115.106 - - [01/Jul/1995:00:00:12 -0400] "GET /shuttle/countdown/countdown.html HTTP/1.0" 200 3985 |
| 9 | d104.aa.net - - [01/Jul/1995:00:00:13 -0400] "GET /shuttle/countdown/ HTTP/1.0" 200 3985 |
| 10 | 129.94.144.152 - - [01/Jul/1995:00:00:13 -0400] "GET / HTTP/1.0" 200 7074 |

*Figure 1 - RawLog_Sample*

# Objectives

Two anomalies were targeted:

- ***User anomaly***: this anomaly aims at detecting users having a significantly different behavior from the majority of users.
- ***Workflow anomaly***: this anomaly aims at detecting abnormal activity windows.

# Methodology and results

The main unsupervised algorithm at the heart of detecting our target anomalies was ***outlier detection with K-Means Clustering***. Each anomaly detection system was made of the following main steps: ***feature extraction, feature visualization, K-Means Clustering and anomaly detection***. The approach for each of the anomalies is as follows:

## User anomaly

The following steps describe the process to detect abnormal users.

### Feature Extraction

From the raw log file, the seven following features were extracted for each user:

- ✓ ***avgReqInFirst8Hrs***: the average number of requests submitted by the user between 00:00AM and 08:00AM excluded.
- ✓ ***avgReqInSecond8Hrs***: the average number of requests submitted by the user between 08:00AM and 04:00PM excluded.
- ✓ ***avgReqInLast8Hrs***: the average number of requests submitted by the user between 04:00PM and 00:00AM excluded.
- ✓ ***numberOfActiveDays***: the number of days the user has been active
- ✓ ***avgUniqDailyReq***: the average number of requests submitted by the user daily, not considering repeated requests.
- ✓ ***avgDailyReq***: the average number of requests submitted by the user daily, considering repeated requests.
- ✓ ***avgDailyBIR***: the average size (in bytes) in reply of daily requests for each user.

### Feature visualization

Incapable of plotting a 7D graph to represent the whole feature set, we made a 3D scatter plot by selecting the 3 most representative features: ***numberOfActiveDays, avgDailyReq and avgDailyBIR***. Moreover, since the features were having very different ranges, we applied a min-max normalization. We ended up with the following scatter plot:

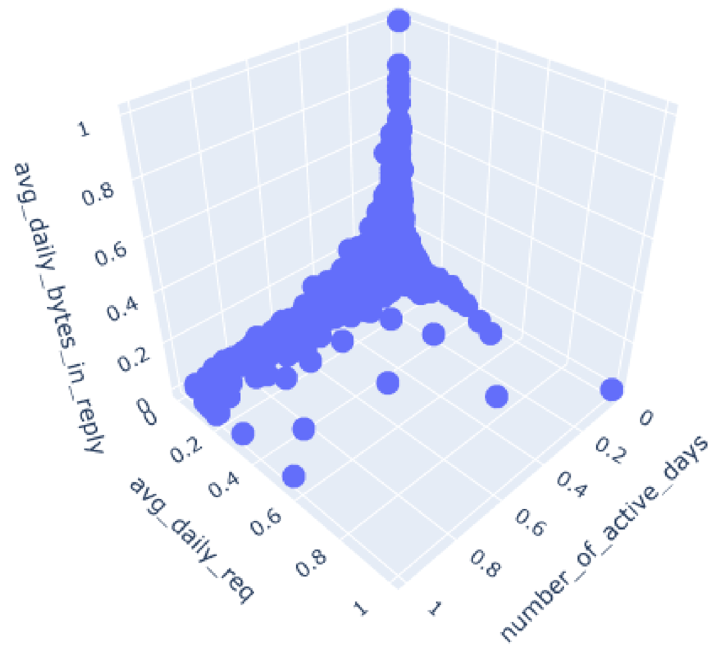



*Figure 2 – User anomaly detection_features Scatter Plot*

## K-Means Clustering

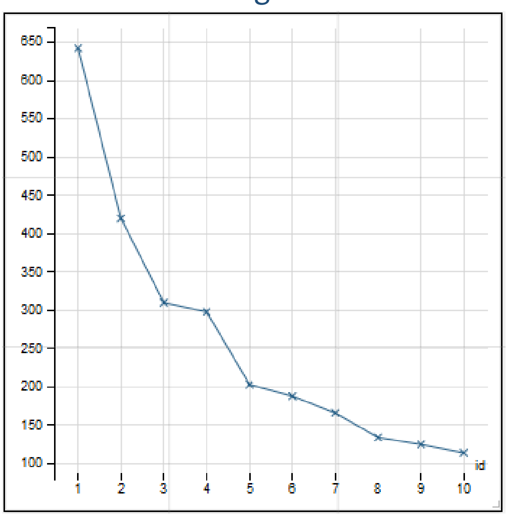



*Figure 3 - Elbow Plot*

With no prior knowledge about the dataset, we used both the scatter plot and the elbow method (graph shown on the left) to have a good estimate of the number of clusters to use.

Considering the preceding scatter plot and the elbow plot (which do not really show a clear elbow), we chose to run K-Means Clustering with ***one cluster***.

The centroid ended up at the following position:

| avgReqInFirst8hrs | avgReqInSecond8hrs | avgReqInLast8hrs | numberOfActiveDays | avgUniqDailyReq | avgDailyReq | avgDailyBIR |
|---|---|---|---|---|---|---|
| 2.648264 | 5.567384 | 3.537103 | 1.604514 | 10.31377 | 11.75275 | 23605.27 |

## Anomaly detection

To detect abnormal users, we computed the distance of each point (user) to the centroid. The following graph shows the distance (y-axis) to the centroid for each user (x-axis):

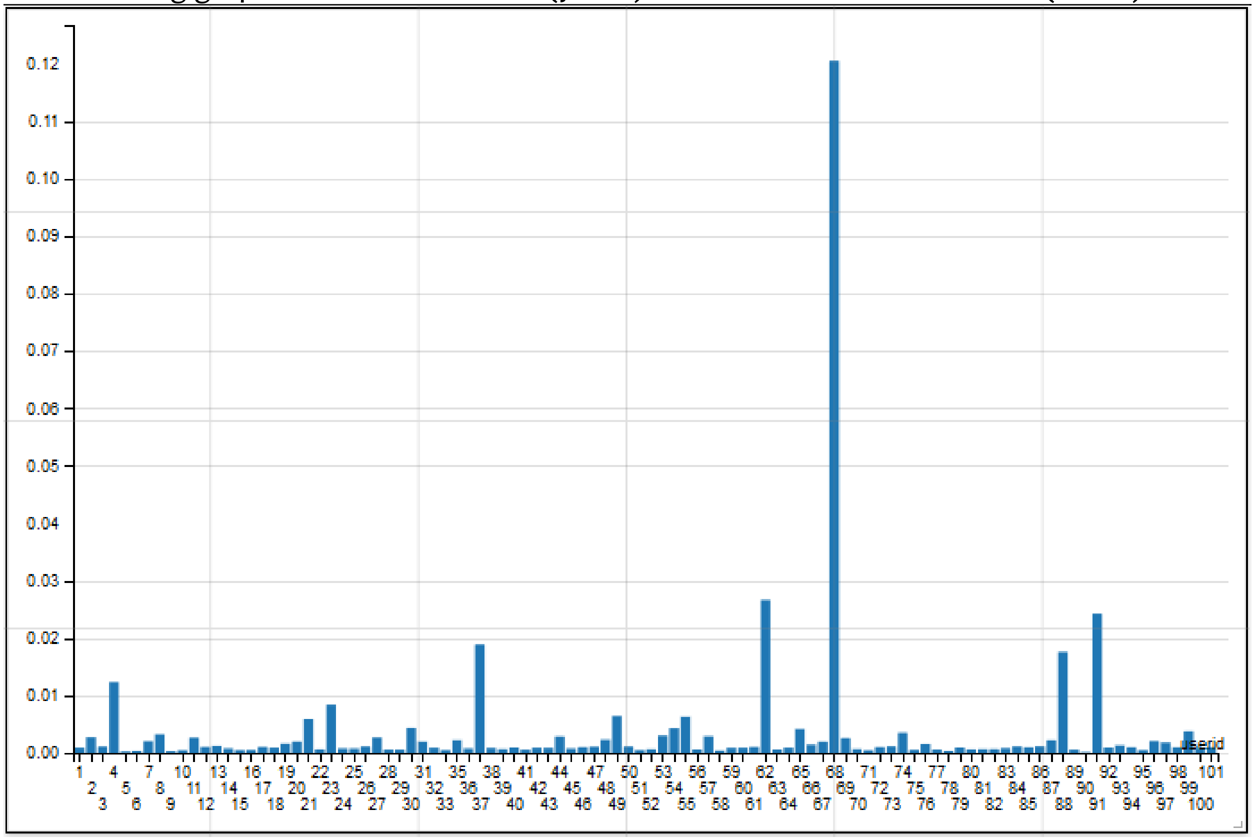



*Figure 4 - Distance to Centroid by user using Visualizer Bundle (Sample)*

One can remark that the number of users in the graph is limited to 101. This is due to the fact that, the plot including all users (81621 in total) was almost not visible in ECL Watch and also was slowing down the browser. The result for all users can be seen in the figure below:

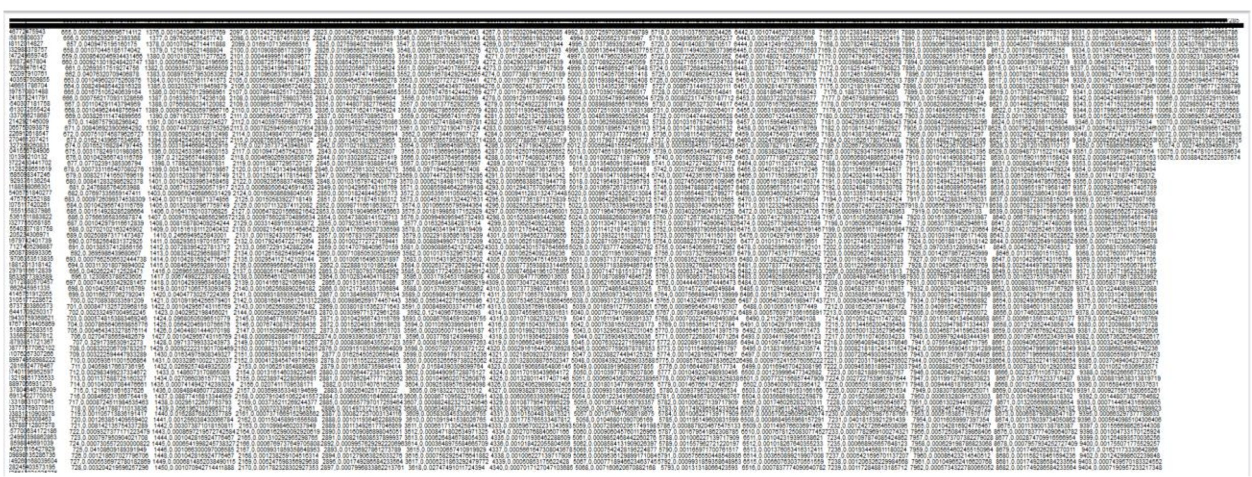


*Figure 5 - Distance to Centroid by user using Visualizer Bundle (Full)*

Due to this limitation, we exported the distance tables and used python's plotly bundle. The result is shown in the figure below:

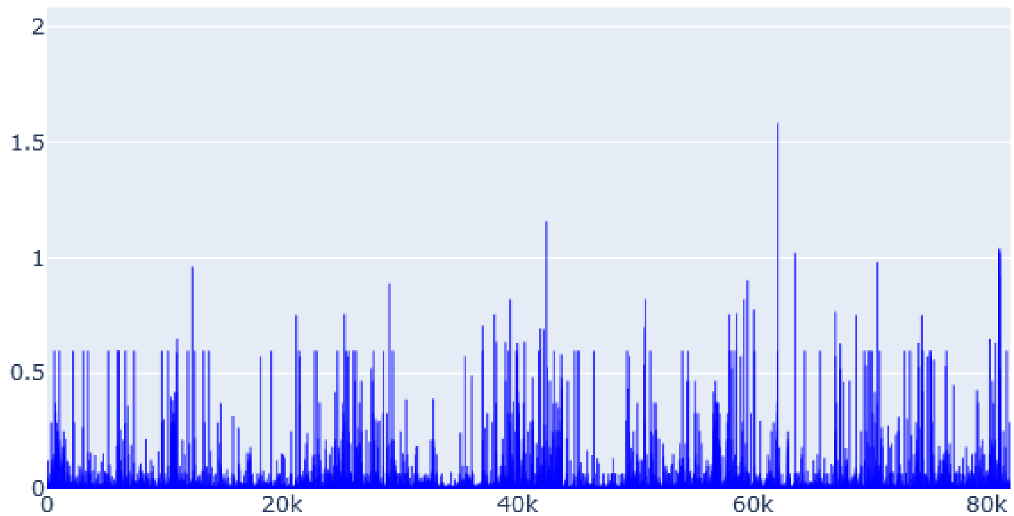



*Figure 6 - Distance to Centroid by user using Plotly*

Considering the preceding graph, we selected **1** as a distance threshold above which users would be considered abnormal. After applying that filter, we got **20** potential abnormal users:

| userid | avgreqinfirst8hrs | avgreqinsecond8hrs | avgreqinlast8hrs | numberofactivedays | avguniqdailyreq | avgdailyreq | avgdailybir |
|---|---|---|---|---|---|---|---|
| 35224 | 43.89285714285715 | 79.5 | 32.07142857142857 | 28 | 64.42857142857143 | 155.4642857142857 | 17001.32504298419 |
| 37334 | 52.39285714285715 | 124.8571428571429 | 16.82142857142857 | 28 | 38.92857142857143 | 194.0714285714286 | 7299.689653718146 |
| 42492 | 1 | 366 | 0 | 1 | 359 | 367 | 14944.5 |
| 58077 | 25.82142857142857 | 96.82142857142857 | 51.03571428571428 | 28 | 84.42857142857143 | 173.6785714285714 | 38834.64352473291 |
| 62178 | 62.17857142857143 | 119.6785714285714 | 170.5714285714286 | 28 | 137.8214285714286 | 352.4285714285714 | 23608.28620254894 |
| 62179 | 35.57142857142857 | 65.07142857142857 | 110.8571428571429 | 28 | 88.71428571428571 | 211.5 | 25405.40376539441 |
| 62180 | 132.9285714285714 | 220.6428571428571 | 274 | 28 | 175.1071428571429 | 627.5714285714286 | 23008.07335774272 |
| 62181 | 94.58333333333333 | 150.5416666666667 | 237.8333333333333 | 24 | 147.5416666666667 | 482.9583333333333 | 21402.41859983557 |
| 63677 | 75.89285714285714 | 65.82142857142857 | 5.321428571428571 | 28 | 81.53571428571429 | 147.0357142857143 | 32731.98365290124 |
| 69151 | 1116 | 3 | 0 | 1 | 25 | 1119 | 710.5773011617516 |
| 81013 | 33.35714285714285 | 33.82142857142857 | 62.10714285714285 | 28 | 87 | 129.2857142857143 | 27945.41091163142 |
| 81014 | 32.78571428571428 | 42.35714285714285 | 74.10714285714286 | 28 | 97.46428571428571 | 149.25 | 26673.28533591622 |
| 81016 | 27.75 | 49.64285714285715 | 55.89285714285715 | 28 | 87.28571428571429 | 133.2857142857143 | 27869.38533675763 |
| 81017 | 29.5 | 38.64285714285715 | 65.96428571428571 | 28 | 90.78571428571429 | 134.1071428571429 | 29214.54746571996 |
| 81018 | 30.10714285714286 | 40.35714285714285 | 60.92857142857143 | 28 | 88.60714285714286 | 131.3928571428571 | 26970.16246288058 |
| 81019 | 28.07142857142857 | 39.85714285714285 | 63.10714285714285 | 28 | 89.17857142857143 | 131.0357142857143 | 28385.11364646626 |
| 81031 | 41.10714285714285 | 48.75 | 58.64285714285715 | 28 | 100 | 148.5 | 23659.45376052417 |
| 81032 | 35.39285714285715 | 35.75 | 56.71428571428572 | 28 | 83.60714285714286 | 127.8571428571429 | 23767.75312013502 |
| 81033 | 29.28571428571428 | 45.17857142857143 | 54.07142857142857 | 28 | 84.10714285714286 | 128.5357142857143 | 26982.39403272956 |
| 81034 | 32.67857142857143 | 42.07142857142857 | 67.28571428571429 | 28 | 97.28571428571429 | 142.0357142857143 | 27289.76113368105 |

*Figure 7 - List_Of_Potential_Abnormal_Users*

### Discussion

From our centroid, we get that most users have been active between 1-2 days. However, for 18 users (except user#42492 and user#69151) in our list of potential abnormal users, we find that they have been active 28 days. That is the whole month for which the dataset was recorded. Moreover, all other features sustain that they have been extremely active. ***Therefore, security analysts may investigate their activities to understand why they have been so active every day***.

For the other 2 users, we remark that their number of active days seem normal (1 day). However, for that particular day, they show a really high activity within second 8 hours of the day for user#42492 and within first 8 hours of the day for user#69151. ***For these users, security analysts may want investigate why they have been so active that day and within a specific portion of that day***.

## Workflow anomaly

The following steps describe the process to detect abnormal activity periods.

### Feature extraction

For this anomaly, we decided to focus on ***one hour periods***. Our features were the count of events occurring in each hour period. To extract our features, we:

- divided the log file into fixed hourly windows (00AM – 01AM, 01AM – 02AM etc.).
- Next, for each time window, we had to count the number of occurrence of each request. However, the log file contains 22432 unique requests. That means our features would be of dimension 22432. That is too high.
- To handle this issue, we added a level of abstraction by no more focusing on each request but on each type of request. To get request's types, we used Spell, a streaming parser for system event logs developed by Min Du, Feifei Li [1]. We reused an implementation of Spell, made by Logpai and available at https://github.com/logpai/logparser.git. From Spell, we got 3 request types: ***GET ****, ***HEAD * HTTP/1.0*** and ***POST * HTTP/1.0***. The figure below shows a sample of the enriched dataset with each request attached to its request type.

| recID | Date | Time | Timezone | UserIP | Content | replyCode | bytesInReply | EventId | EventTemplate |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1-Jul-95 | 0:00:01 | 400 | 199.72.81.55 | GET /history/apollo/ HTTP/1.0 | 200 | 6245 | 37e3d9fa | GET * |
| 2 | 1-Jul-95 | 0:00:06 | 400 | unicomp6.unicomp.net | GET /shuttle/countdown/ HTTP/1. | 200 | 3985 | 37e3d9fa | GET * |
| 3 | 1-Jul-95 | 0:00:09 | 400 | 199.120.110.21 | GET /shuttle/missions/sts-73/mis: | 200 | 4085 | 37e3d9fa | GET * |
| 4 | 1-Jul-95 | 0:00:11 | 400 | burger.letters.com | GET /shuttle/countdown/liftoff.ht | 304 | 0 | 37e3d9fa | GET * |
| 5 | 1-Jul-95 | 0:00:11 | 400 | 199.120.110.21 | GET /shuttle/missions/sts-73/sts- | 200 | 4179 | 37e3d9fa | GET * |
| 6 | 1-Jul-95 | 0:00:12 | 400 | burger.letters.com | GET /images/NASA-logosmall.gif I | 304 | 0 | 37e3d9fa | GET * |
| 7 | 1-Jul-95 | 0:00:12 | 400 | burger.letters.com | GET /shuttle/countdown/video/li | 200 | 0 | 37e3d9fa | GET * |
| 8 | 1-Jul-95 | 0:00:12 | 400 | 205.212.115.106 | GET /shuttle/countdown/countdo | 200 | 3985 | 37e3d9fa | GET * |
| 9 | 1-Jul-95 | 0:00:13 | 400 | d104.aa.net | GET /shuttle/countdown/ HTTP/1. | 200 | 3985 | 37e3d9fa | GET * |
| 10 | 1-Jul-95 | 0:00:13 | 400 | 129.94.144.152 | GET / HTTP/1.0 | 200 | 7074 | 37e3d9fa | GET * |

*Figure 8 - Sample Data with request types*

After this dimensionality reduction, we made a count of the occurrence of each request type in each time window. A sample of our features can be seen in the table on the right.

| windowid | cntevent1 | cntevent2 | cntevent3 |
|---|---|---|---|
| 1 | 3563 | 1 | 1 |
| 2 | 3002 | 2 | 0 |
| 3 | 2259 | 9 | 0 |
| 4 | 1729 | 5 | 0 |
| 5 | 1482 | 0 | 0 |

*Figure 9 - Sample features*

The figure below shows a scatter plot of our extracted features.

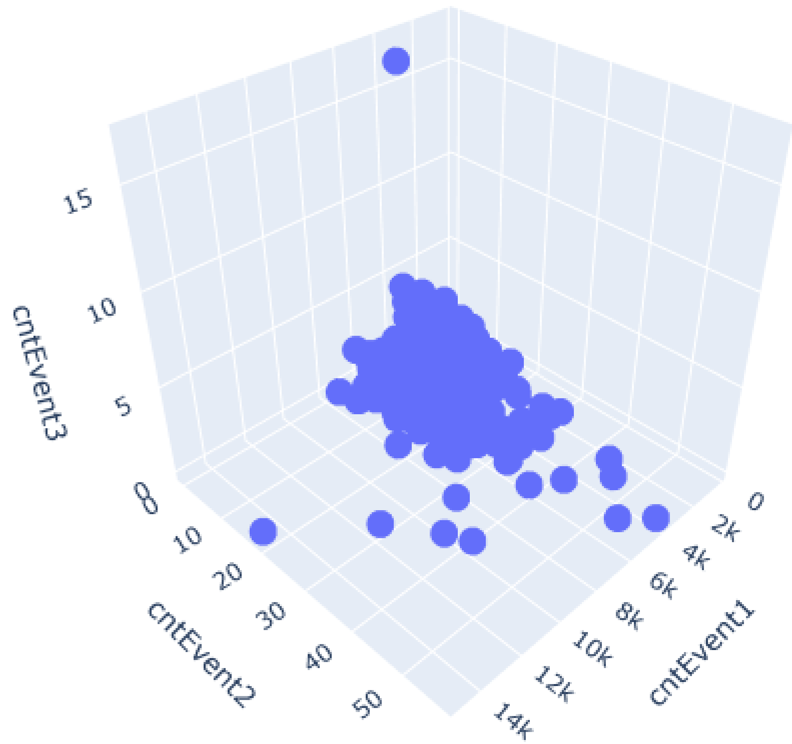
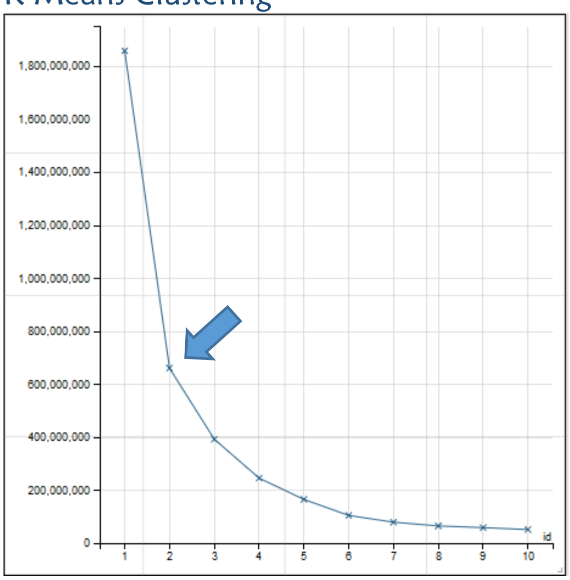



*Figure 10 - Workflow Anomaly Detection_features Scatter Plot*

## K-Means Clustering




From the scatter plot, we would consider one cluster. However, the elbow plot suggests 2 centroids. Therefore, we run K-Means with ***two centroids***.

The centroids ended at the following positions:

| CentroidID | cntEvent1 | cntEvent2 | cntEvent3 |
|---|---|---|---|
| 1 | 4845.343 | 12.019324 | 0.270531 |
| 2 | 1944.0462 | 3.2131868 | 0.120879 |

## Anomaly detection

As we did to detect abnormal users, we computed the distance of each point (window) to its centroid. The following graphs show the distance (y-axis) to the centroid for each window (x-axis) and for each cluster. This time, the amount of data to display was manageable by the Visualizer.

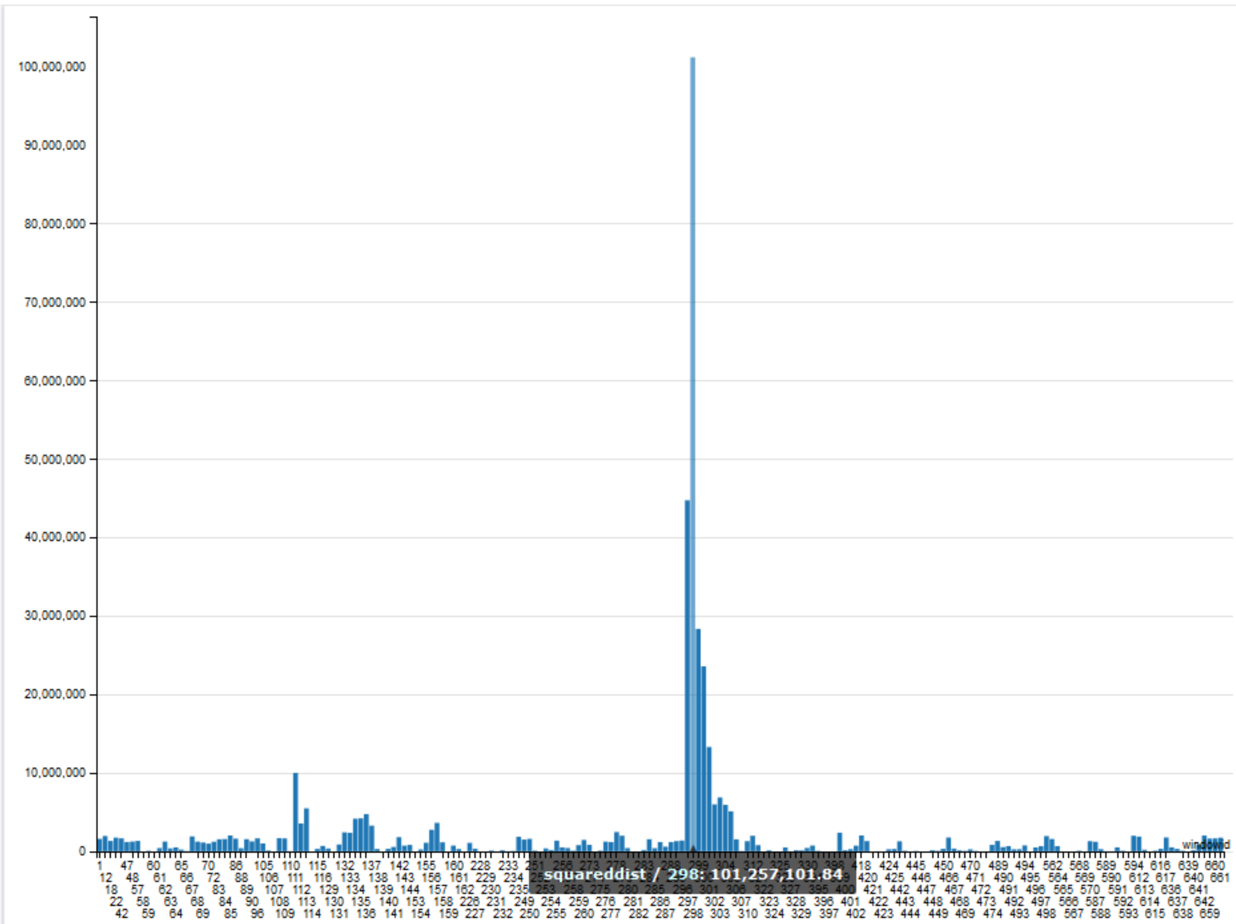



*Figure 11 - WorflowAD_Distance Plot #1*

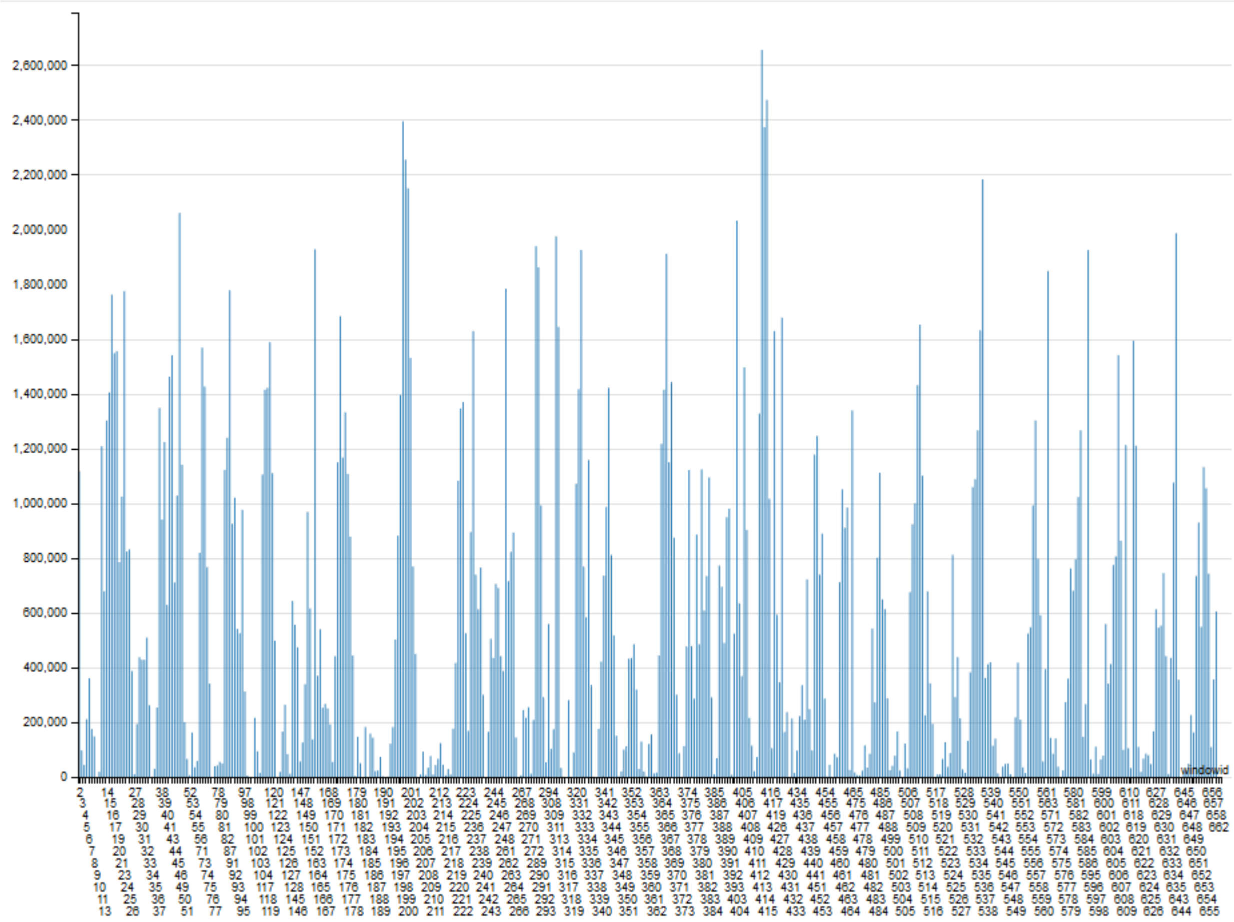



*Figure 12 - WorflowAD_Distance Plot #2*

Considering the preceding graphs, we selected **10,000,000** (for cluster #1) and **2,200,000** (for cluster #2) as distance thresholds above which users would be considered abnormal. After applying those filters, we got **11 potential abnormal windows** (6 related to cluster #1 and 5 related to cluster #2). The figures below show our potential abnormal windows:

| windowid | cntevent1 | cntevent2 | cntevent3 |
|---|---|---|---|
| 111 | 8016 | 18 | 0 |
| 297 | 11538 | 29 | 0 |
| 298 | 14908 | 18 | 0 |
| 299 | 10175 | 37 | 0 |
| 300 | 9707 | 41 | 0 |
| 301 | 8500 | 33 | 0 |

*Figure 13 - Potential abnormal windows #1*

| windowid | cntevent1 | cntevent2 | cntevent3 |
|---|---|---|---|
| 198 | 396 | 1 | 0 |
| 199 | 442 | 0 | 0 |
| 413 | 314 | 2 | 0 |
| 414 | 403 | 6 | 0 |
| 415 | 371 | 0 | 0 |

*Figure 14 - Potential abnormal Windows #2*

### Discussion

Our two centroids seem to represent two types of windows:

1. windows with an average number of event 1 (Get requests) submissions and the highest number of event 2 (Head) submissions.
2. The second centroid represent windows with a high number of submissions for event 1.

Regarding the occurrence of each event type (Get, Head and Post), it should be noted that Get events occurs the most followed by Head and finally Post.

Considering our interpretations of each centroid, we can say that our six potential abnormal windows related to cluster #1 shows a high number submission for both event 1 and event 2. ***Those windows can therefore be considered as the busiest ones***. Those busy windows could be related to a particular day like the black Friday or a possible cyber-attack.

For the five potential abnormal windows related to window #2, we see that they show a relatively low submission rate for both event 1 and event 2. ***They can therefore be considered as the "idle" periods***. These almost idle periods could be normal or associated with some service being down or a downgrade in system performance or internet connection.

## Conclusion

Our goal was to develop and assess unsupervised anomaly detection methods. To achieve this goal, we applied ***K-Means Clustering*** in two different but similar approaches to detect anomalies in a log file (NASA-HTTP Web Server Log for this experiment). We had two target anomalies: to find abnormal users and to find abnormal activity windows. The two algorithms designed to detect our target anomalies followed four main steps: ***feature extraction, feature visualization, K-Means Clustering and anomaly detection.***

**At the end of our project, we can conclude that the designed unsupervised anomaly detection methods were effective**. We were able to detect potential abnormal users. Those users showed a relatively high activity compared to the majority of users. On the other hand, we also got some potential abnormal activity windows. Those windows showed the highest and lowest activity rate compared to most activity windows.

## Future work

Our experiments were conducted on a single log file. This work can be extended to other files like HPCC log files in order to confirm the effectiveness of the approaches and also make the algorithms more robust, applicable to a variety of log files.

Also, this work could be extended by adding modules to determine root causes of the detected anomalies.

Finally, this work can be leveraged by the implementation of a streaming anomaly detection system. Indeed, the algorithm designed in this project works in batch mode. A streaming version would analyze the log file in real-time and be able to detect anomalies timely as the logs are recorded.

## References

[1] Min Du and Feifei Li. 2016. Spell: Streaming Parsing of System Event Logs. In Proc. IEEE International Conference on Data Mining (ICDM). 859–864.