



Detection and Visualization of Anomalies in Cloud-Based Enterprise Applications

Master Thesis for High Integrity Systems (M.Sc.)

By: Mohammad Faiz Usmani, 1323197.

First Supervisor: Prof. Dr. rer. nat. Josef Fink.

Second Supervisor : Msc. Bastian Stock.

- **Personal:**

- To take up a challenge on a large real-world dataset for future career prospect in data analytics and data engineering.

- **Practical:**

- To analyse the OTC events dataset and come up with a process that can be used in the future to mark anomalies.

- Test the capabilities of Python libraries and Azure Data Explorer specifically for this business use-case.

- **Thesis Writing:**

- To write a documentation (specifically sections before Case-Study in documentation) that provides a 'One-stop-shop' for Anomaly detection.

- This includes Use-cases of big tech companies, various terminologies, and explaining the effective and popular techniques.

RESEARCH QUESTION

"Is it possible to detect and visualize so far unknown, valid, explainable and actionable anomalies in event communication between multi cloud-based enterprise applications?"

Keywords:

Exploratory Data Analysis, Descriptive Analytics, Time-Series Anomaly Detection

Data Characteristics:

EventID: 40 unique events

EventIDCompact: 22 unique values (clubbing some eventIDs together like 16a, 16b, 16c, etc.)

eventPhase: 2 values (Order and Contract)

System: 3 values (System 1, System 2, System 3)

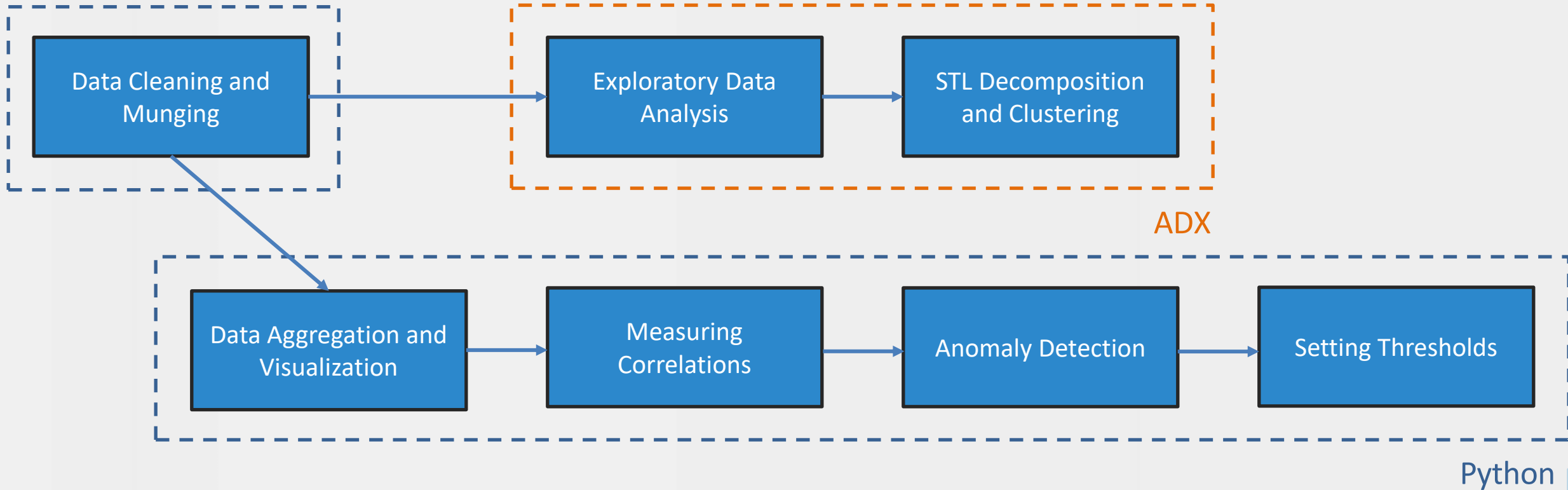
Timestamp: datetime of event occurrence till microseconds.

eventIDSteps: unique reference number that distinguishes every event.

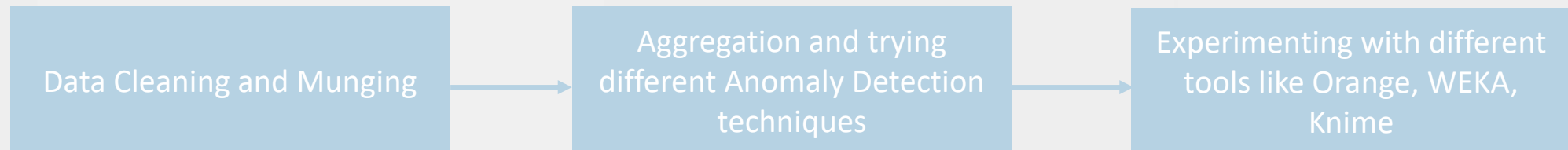
retryCounter: upto 15 retries per eventIDSteps.

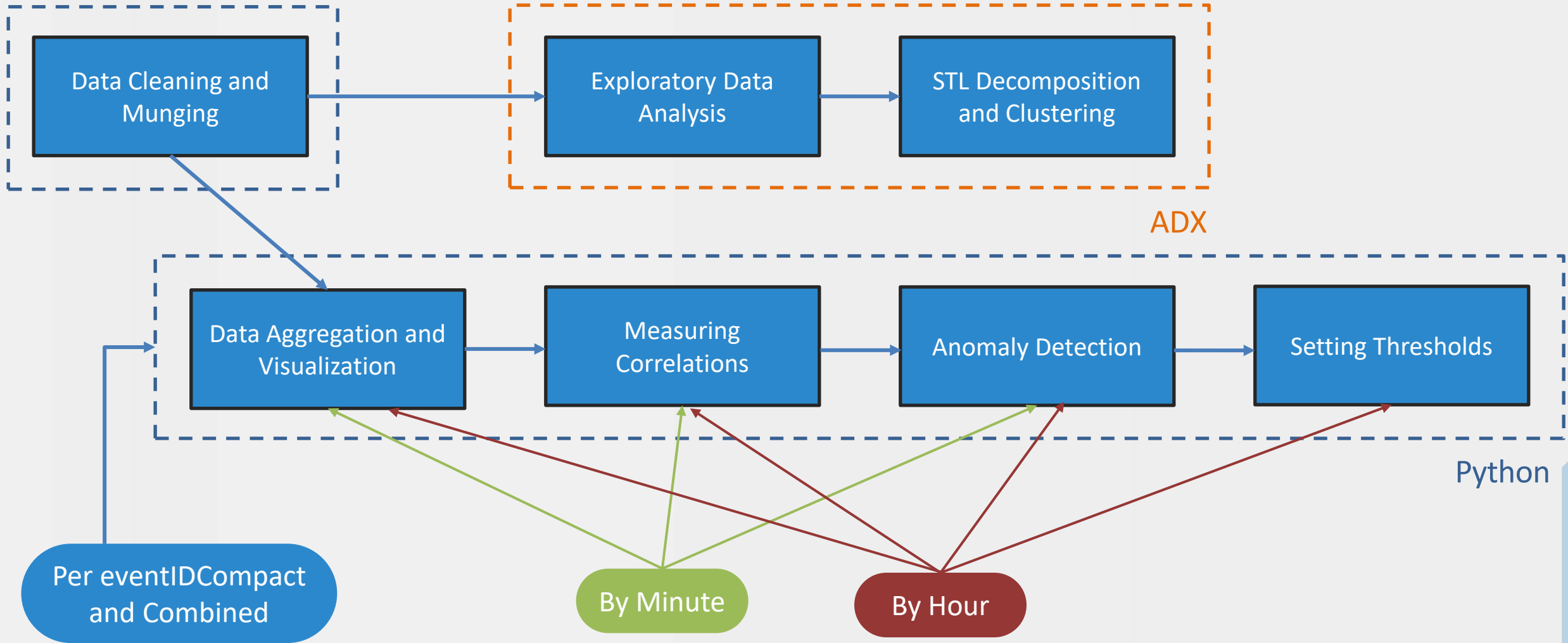
latencyInSeconds: latency from start time until processing time of the event including retries (if any).

APPROACH



Old Dataset





- **Data Cleaning and Munging:**

- Change DateTime format, setting index, checking distribution and missing values, removing entire duplicate rows.

- **Data Aggregation and Visualization:**

- Aggregate time series on counting events by day, hour, minute and visualize them.

- Also apply aggregation functions (count, sum, avg) on retryCounter and latencyInSeconds by hour and minute and visualize them.

- **Measuring Correlations:**

- Various aggregated parameters from the above step ex- eventCount, retryCounter were combined in one dataframe (minute and hour separately) and correlations were found out using Pearson and Kendall's Tau Coefficients.

- **Anomaly Detection:**

- ADTK and Scikit libraries were used for various methods like Persistence, AutoRegression, Double Rolling Windows, Isolation Forest on multivariate series with proper window sizes based on seasonality.

- **Setting Thresholds:**

- Data aggregated on 'hourly' base was identified as a good candidate for setting threshold.
- Using Median + 2.5*Median Absolute Deviation, the thresholds were calculated considering seasonality.
- Based on the IQR of the difference between outliers and threshold, low, medium and high priority was assigned to the outliers.

- **Exploratory Data Analysis (ADX):**

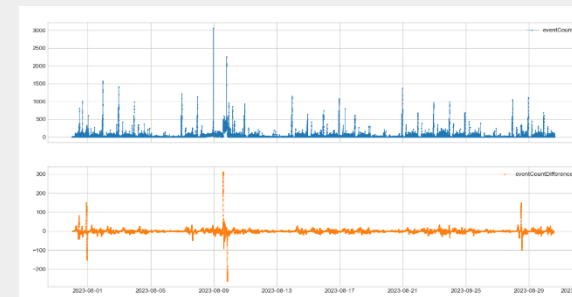
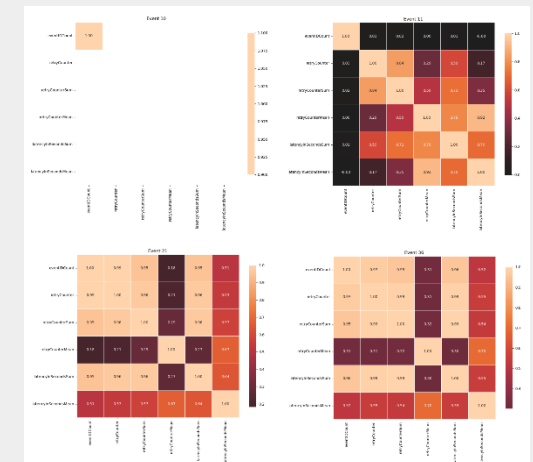
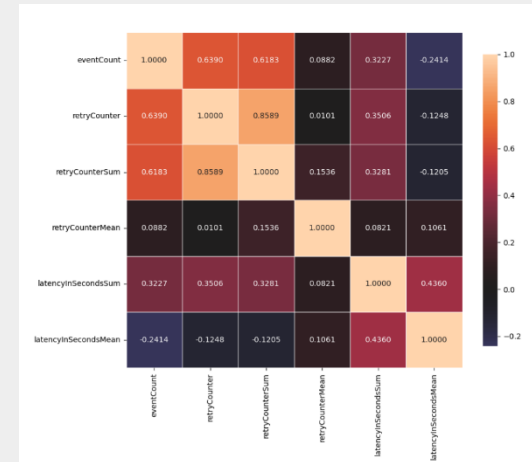
- Made visualizations on summarizing the events by various features. Made time-series for eventCount segregated by system, Phase and eventIDCompact on the hourly data.
- Applied the similar approach for retryCounter and LatencyInSeconds.

- **STL Decomposition and Clustering (ADX):**

- Using functions in KQL for STL decomposition, found out the anomalous time interval.
- Using clustering feature values were identified that were contributing to high count in the anomalous time period.
- Diffpattern function of KQL was used to identify feature values that were different between a normal and an anomalous time interval.

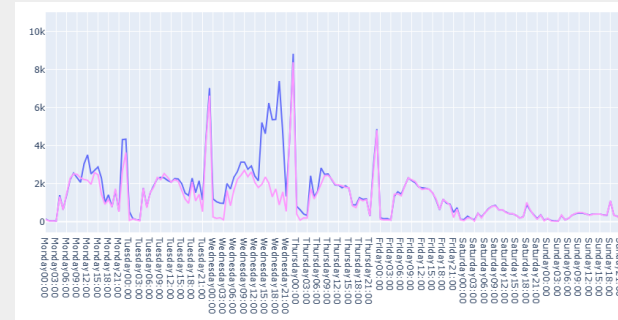
RESULTS

- Strong correlation between eventCount and Retries, decent correlation between Latency (sum) and eventCount and Retries per hour. Reason- There are events with no retries and latency present. Also, there are some events with very high values present for latency. (Pic 1. – Kendall's tau Correlation, Pic 2.- Correlation for parameters belonging to events with no, very less and significant retries)
- In detecting anomalies Persistence, Autoregression, and Isolation Forest were all right in the hourly approach but got worse in the per minute approach. Double Rolling windows performed better in the per minute aggregation. (Pic 3- Persistence model for detecting anomalies in eventCount and retries per hour, Pic 4- Isolation Forest on eventCount and retries combined per hour and per minute, Pic 5- Double rolling windows per minute on eventCount)

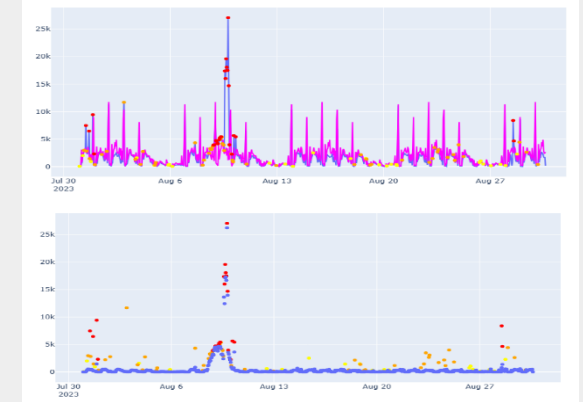


RESULTS (contd.)

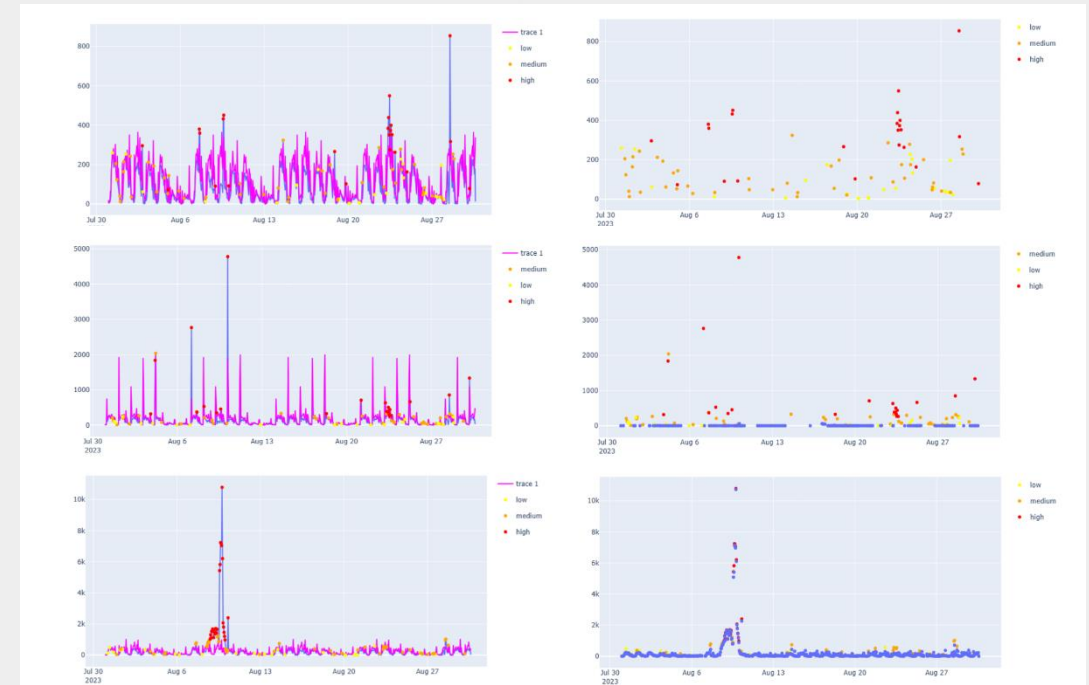
- Mean and Median of the eventCounts based on the seasonality (day of week, hour of the day) per hour were plotted against each other and skewness was revealed. (Pic 6.)
- MAD thresholds were calculated based on the seasonality and visualized in pink along with the eventCount in blue. Based on the IQR of the difference outliers were marked with red, orange and yellow. And the actual points marked in red were in line with actual retries. (Pic 7.)
- For events with less retries the actual retries are somewhere around the orange and red marked points. And the red points kindof completely overlay the actual retries for events with high retries. (Pic 8.)



6.



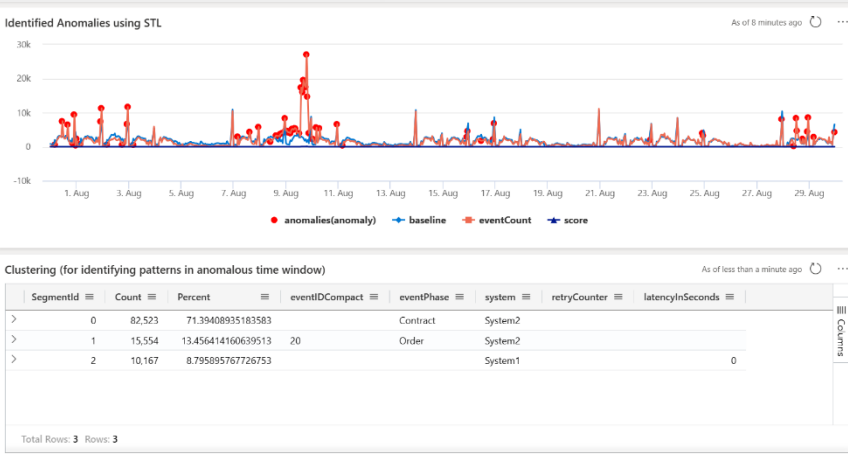
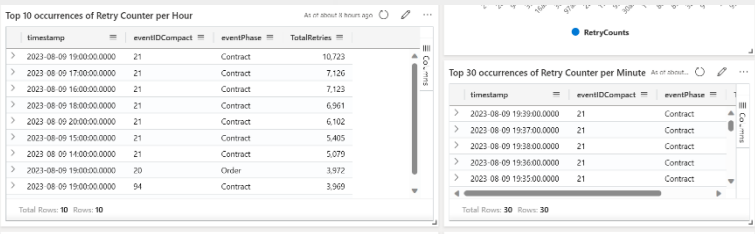
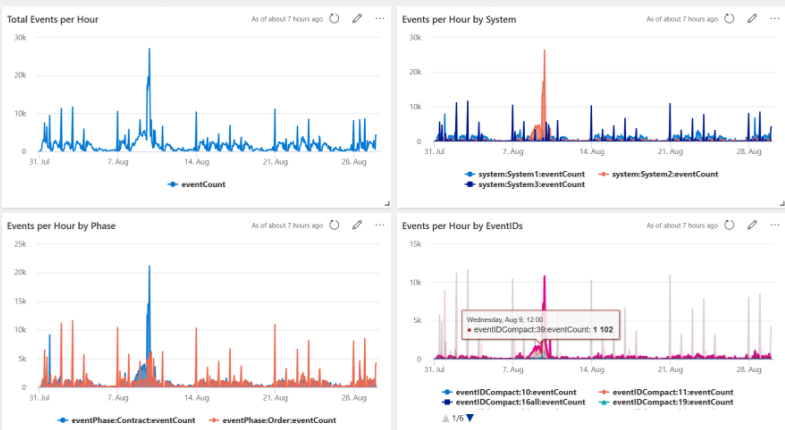
7.



8.

RESULTS (contd.)

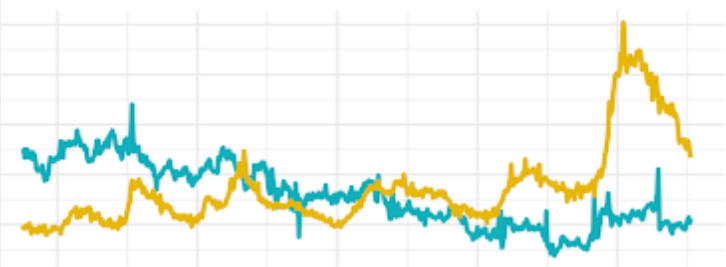
- In ADX segregating events by Phase, System and eventIDs revealed a lot of interesting patterns. Same can be said for the number of retries by the above features. (Pic. 9)
- Counting top occurrences by combining the features together revealed further info. (Pic. 10)
- Used STL to identify time interval where a lot of outliers are clubbed together. Used that time interval to identify top features that are contributing to the pattern using Clustering. (Pic. 11)



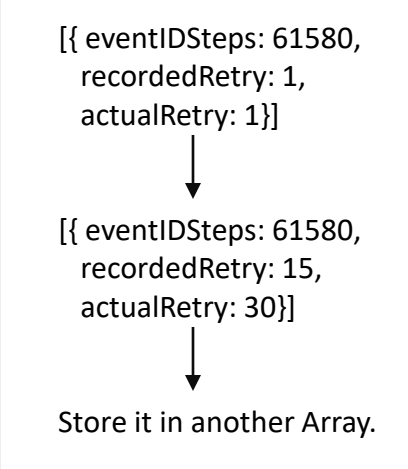
- Events with no retries (10, 31, 87, 99s), Events with very less retries (9, 11, 20, 95), and Events with a significant number of retries (16all, 19, 21, 22, 30all, 32, 36, 39, 86, 89, 91, 94, 97all, 99).
- Events 16all, 30all and 97all had latency without retries.
- Almost for all the events the data showed seasonality where eventCount was high on the morning and afternoon part of weekdays but eventually got low from the night of Friday till the night of Sunday.
- Event20 showed behavior against this pattern where the eventCount was highest at 23:00 but there were no retries, hence it could be attributed to a real use-case being executed.
- There were high counts of retries around July 31 22:00, Aug 8 – Aug 10 (2 days), and Aug 28 12:00. And these were depicted correctly through steep peaks and troughs in the visualization for double rolling windows and high/medium severity anomalies using MAD.
- Using Exploratory Data Analysis and Clustering in ADX- System 2 then Contract (phase) and Events 21, 20 (Order), 94, 39, 16all (Order) were identified as major causes for anomalies.
- Almost all of the retries came from System 2 and during the anomalous time period around August 9, the actual retries were 4 times the registered ones.

FURTHER EXPLORATIONS DONE

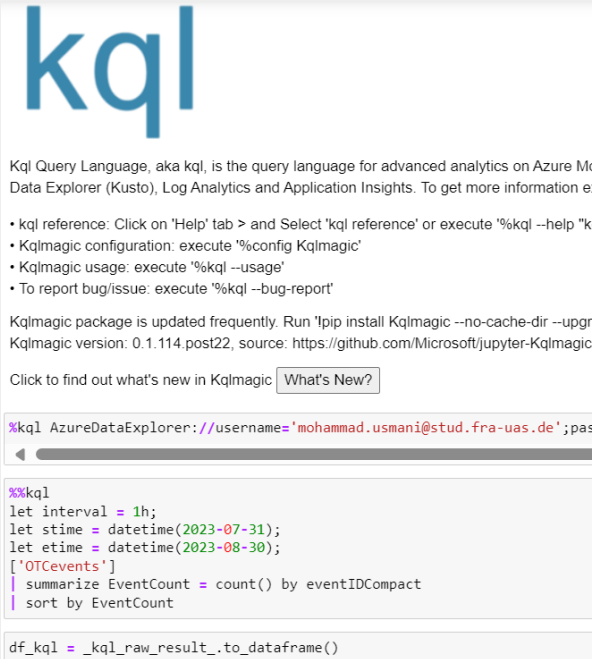
- Using Time Differencing in Python to set thresholds and identify events that are occurring very close together. Can also make it granular. (Pic 12.)
- Storing EventIDSteps in a data structure (array of objects) when the retries start and keeping a counter on actual and recorded retries. (Pic 13.)
- Using Python in Azure is possible like a pipe operator in Sandbox images but is not that flexible. Also added cost and execution time is high. (Pic 15.)
- Ingesting data in Jupyter Notebooks through ADX and using KQL is possible through KQLMagic extension and Microsoft Entra connection strings. Other extensions: azure-kusto-data (Cluster connection, running queries), azure-kusto-ingest (for ingesting data). (Pic 14.)
- AI Anomaly Detector (being retired in 2 years)
- Tools- Orange (nice GUI, good execution speed), Knime (rich in functionalities and integrations), WEKA (Old GUI, Not so fast)



12.



14.



Kql Query Language, aka kql, is the query language for advanced analytics on Azure Microsoft Data Explorer (Kusto), Log Analytics and Application Insights. To get more information e

- kql reference: Click on 'Help' tab > and Select 'kql reference' or execute '%kql --help "k
- Kqlmagic configuration: execute '%config Kqlmagic'
- Kqlmagic usage: execute '%kql --usage'
- To report bug/issue: execute '%kql --bug-report'

Kqlmagic package is updated frequently. Run '!pip install Kqlmagic --no-cache-dir --upgr Kqlmagic version: 0.1.114.post22, source: https://github.com/Microsoft/jupyter-Kqlmagic

Click to find out what's new in Kqlmagic: [What's New?](#)

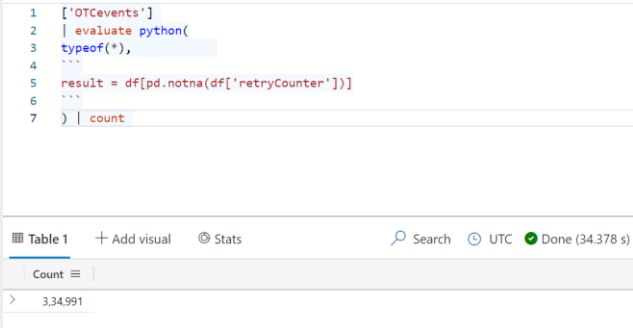
```

%%kql AzureDataExplorer://username='mohammad.usmani@stud.fra-uas.de';pas
let interval = 1h;
let stime = datetime(2023-07-31);
let etime = datetime(2023-08-30);
['OTCEvents']
| summarize EventCount = count() by eventIDCompact
| sort by EventCount

df_kql = _kql_raw_result_.to_dataframe()
  
```

14.

15.



```

1 ['OTCEvents']
2 | evaluate python(
3   typeOf(*),
4   ...
5   result = df[pd.notna(df['retryCounter'])]
6   ...
7   ) | count
  
```

Count
3,34,991

15.

- **What went well?**

- Learning ADX, various techniques of anomaly detection and data analytics in general, and successfully implementing them to find anomalies.
- Investigative approach: Learn about the data behavior and then apply techniques, not the other way around.

- **What didnt go well?**

Time constraints, first dataset was faulty and had to change the approach once the current dataset was received.

- **What could improve?**

- More data: better setting of thresholds (considering multiple seasonalities) and identifying outliers, other techniques like LSTM Autoencoders could be then used effectively to identify outliers.
- Automation, real-time analysis.

Thank You.