

Automatic Alert's Triage

Summer Student Project 2019

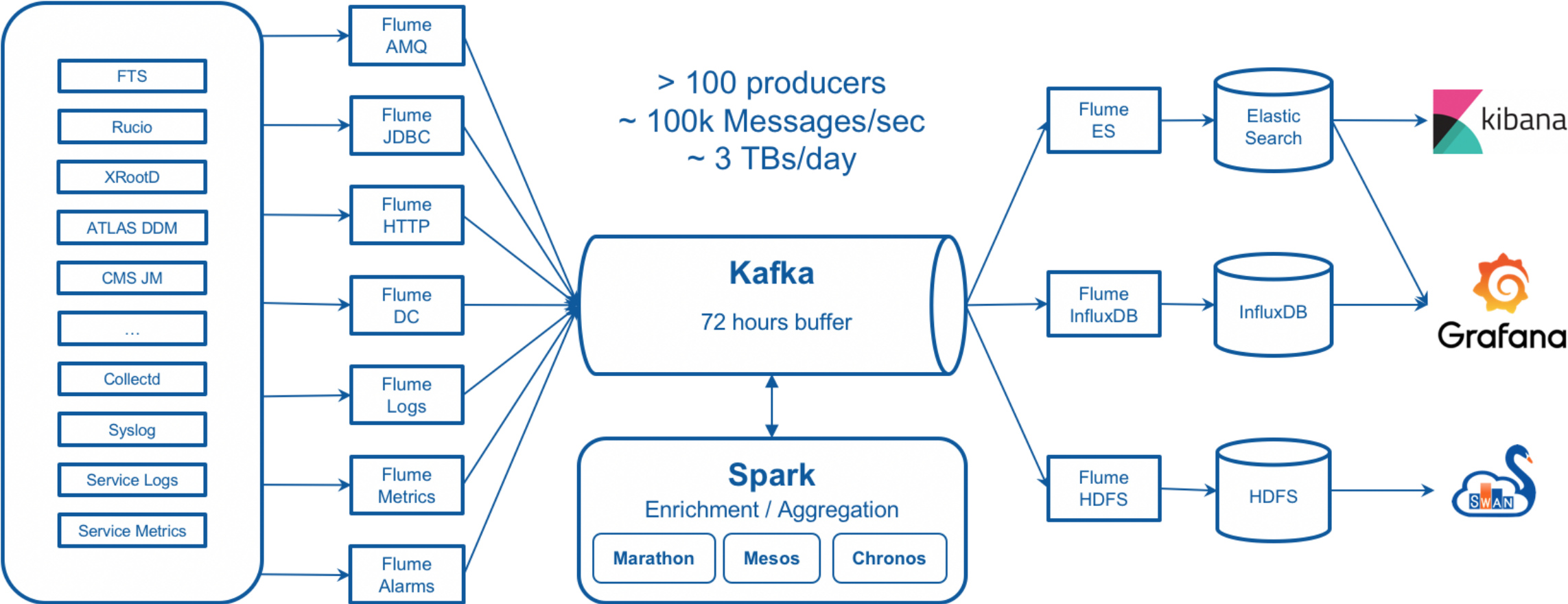
CMS Monitoring Team

International School of Engineering,
Chulalongkorn University

Yanisa Sunthornyotin

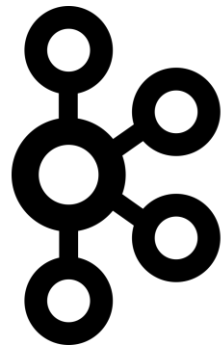





Sources > Transport > (Processing) > Storage > Access



Multiple sources and notification channels

- Each subsystem produce its own data, but we have a common messaging service: Kafka



- E-mail 
- Tickets (Jira/Snow) 
- Messaging Applications 

Different expected response time.

Original statement

A syntax to describe complex events and a model to classify their notification channels

Complex patterns

E.g.:

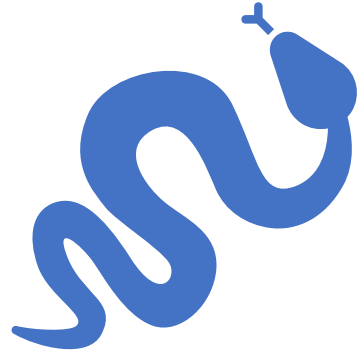
- Timeout message has been repeated 5 times in a 1 hour window for a given node.
- Memory increase alert but there is not a request increase alert.

Alerts Classification

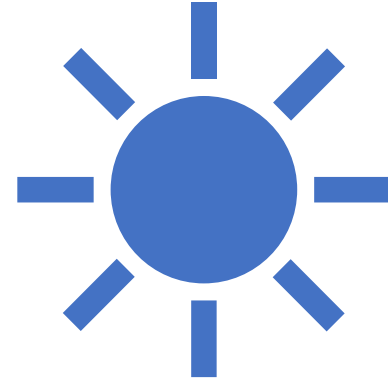
- Notification Channel as target variable
- Features to be determined
- Explore multiples alternatives (e.g. Decision trees, Random Forest, content based recommendation techniques)

Results visualization and presentation

- Storytelling with data guidelines.
- Paper with results



Python and Bash only.

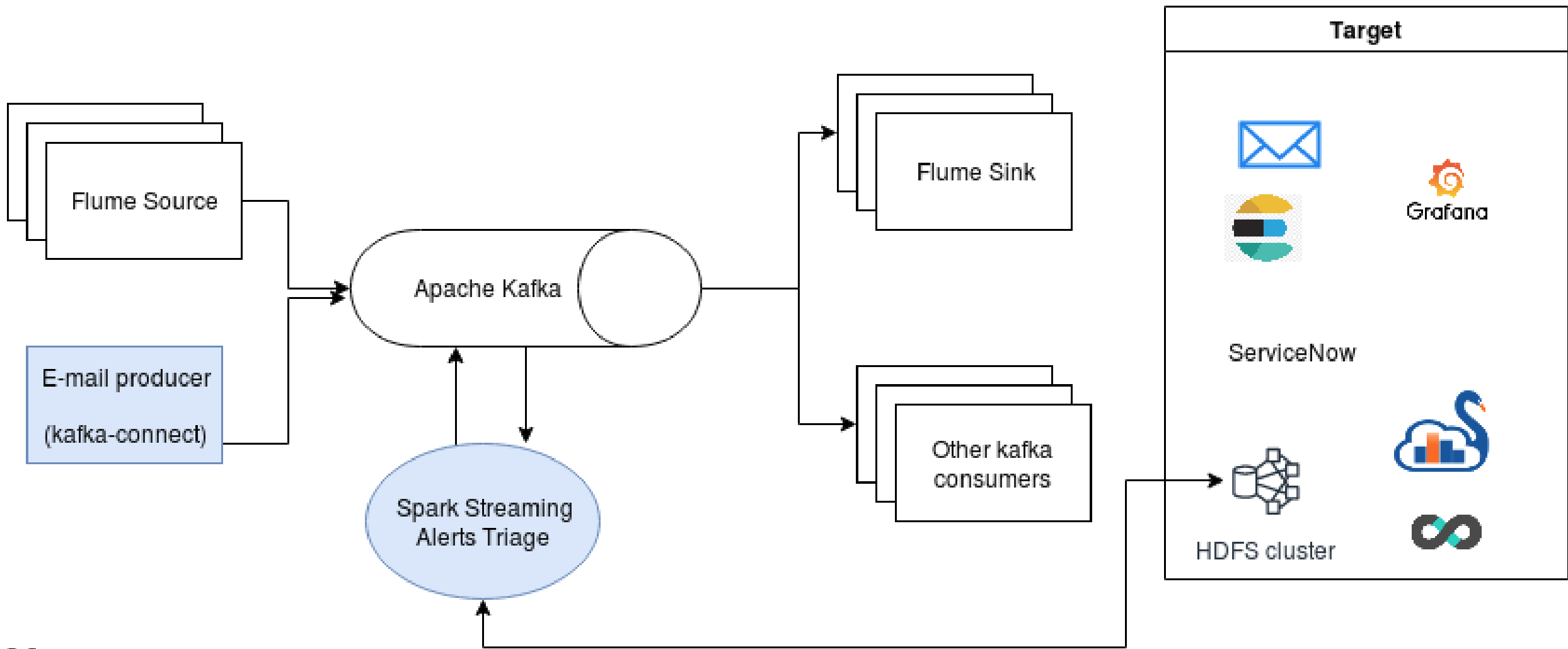


The project must be finished during
the summer stay.

Restrictions

Tools

- Apache Spark
- Scikit-Learn
- Spark streaming
- Apache Flink



Notes

Consuming from Kafka, we can use this architecture in a project-agnostic way. Initially, we will work with currently produced alerts, but the architecture will help to produce new alerts from other sources in the future. We are looking forward to new applications.

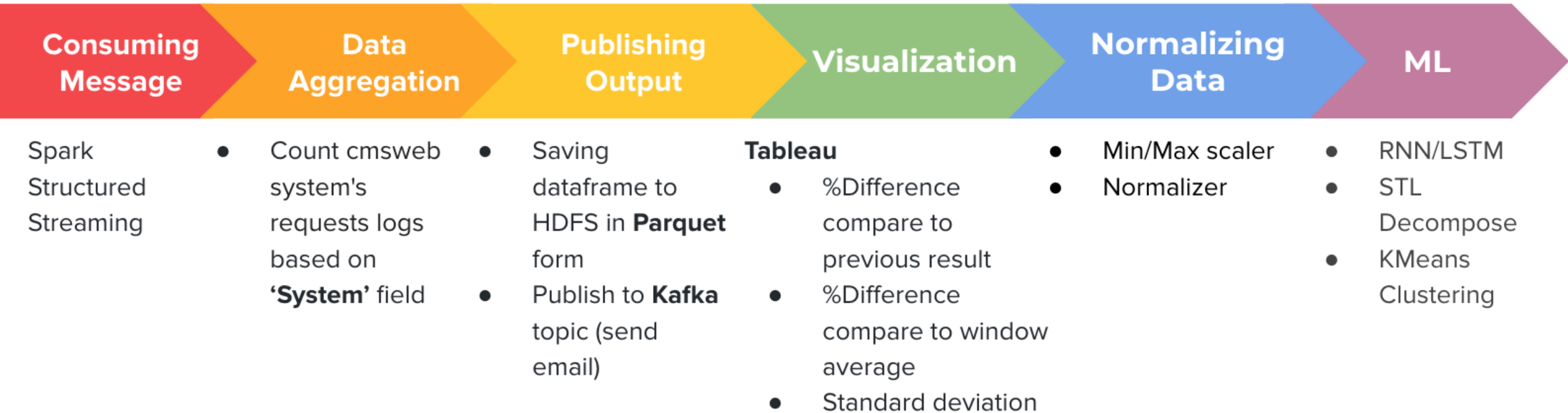
The project will be released with an Open Source license.

Current Status

Learn from past users
behavior on diverse systems
and APIs.

Find and report anomalies.

Project Progress





Use clustering to group similar behavior



Detect anomalies using clustering quality measures



Report anomalies using the Notifier module

New approach

Important Factor For Anomaly Detection

- Percentage different compared to average value

- Number of System being called
- Number of API being called
- Number of User who send a request to system

If some of these factor are lower than usual or higher than usual such as there's a spike occurred, it should be considered as anomaly

- Date/Time of the logs

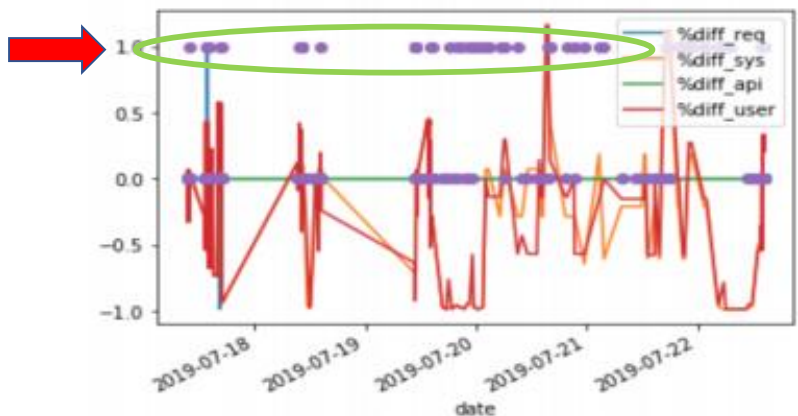
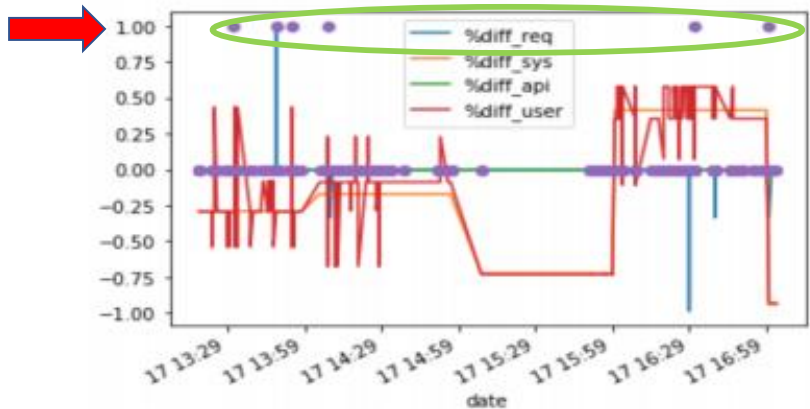
- Make the anomaly more personalized. In the particular time period some factor should be considered as an anomaly but some shouldn't.

By combining these features with the name of system, user, and API we can make anomaly detection system more personalize.

Anomaly can be detected by several reasons

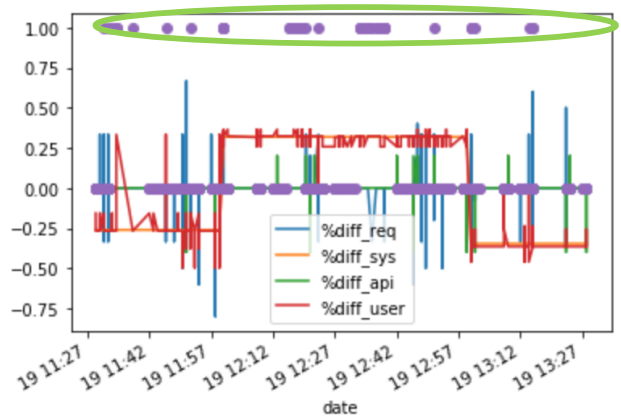
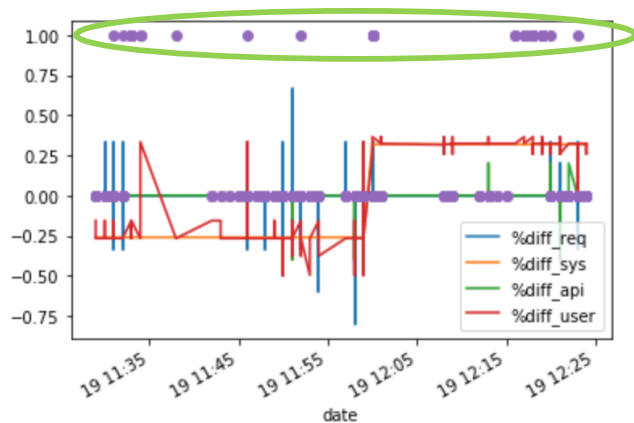
- Heavy user loads
- Heavy system called
- Heavy API called
- Abnormal date and time which the peak occurred
- Combination of these factors or altogether

Alert trigger



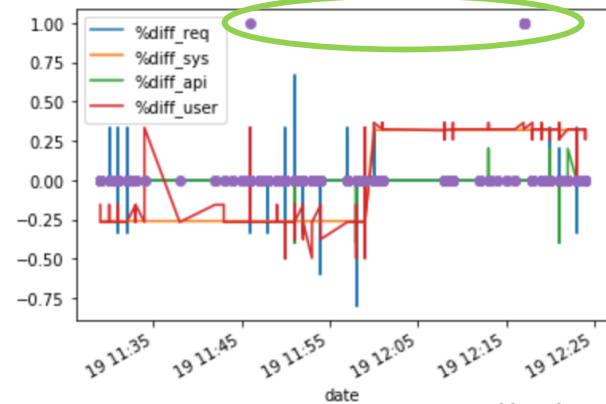
Benchmark: +/- 40% of Max Euclidean distance to the centre of cluster

Example: CouchDB system log

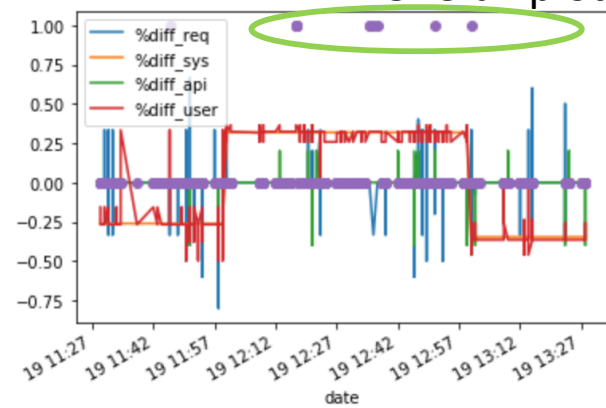


Benchmark: >15.D of the Euclidean distance mean of every data in cluster to the centre of its cluster

Details plot[Zoom in]



Overall plot[Zoom out]



Benchmark: >25.D of the Euclidean distance mean of every data in cluster to the centre of its cluster

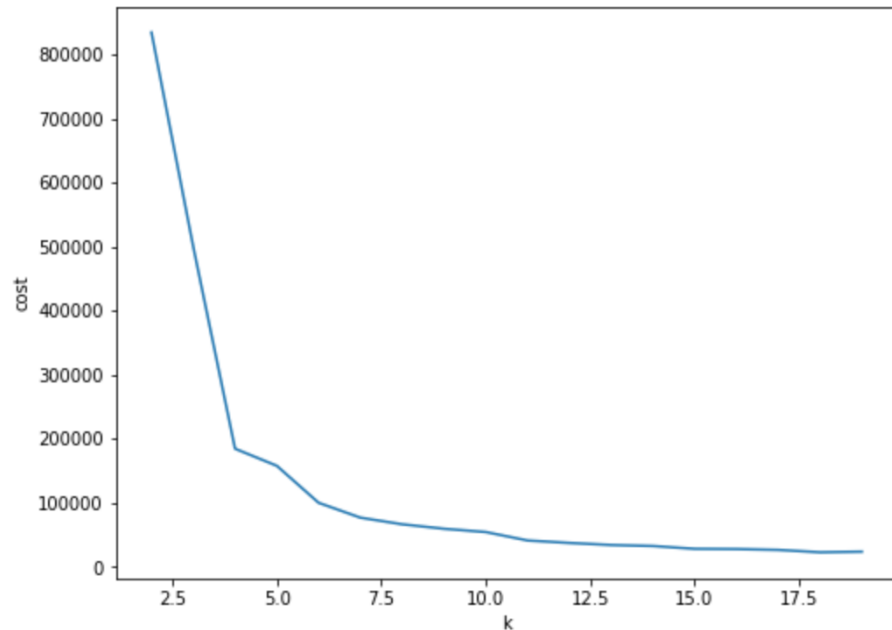
CouchDB anomaly detection with 40% uncertainty region

Towards interpretability

- Why are we getting the alert?
 - An user has changed his behaviour
 - A system is under attack
 - An API too popular

Model Evaluation

- Elbow Method



Plot of elbow in data 2000 samples determine that k value should be around 10

- Silhouette: Study the separation distance between the resulting clusters
 - +1 indicate that the sample is far away from the neighbouring clusters
 - 0 indicates that the sample is on or very close to the decision boundary between two neighbouring clusters
 - -1 indicate that those samples might have been assigned to the wrong cluster

Kmeans Model evaluation using Silhouette method

```
# Evaluate clustering by computing Silhouette score
evaluator = ClusteringEvaluator()
silhouette = evaluator.evaluate(predictions)
print("Silhouette with squared euclidean distance = " + str(silhouette))
```

```
Silhouette with squared euclidean distance = 0.6417416612740873
```

For more information
Please visit...

<https://github.com/operationalintelligence/EmailAlertingSystem/>