Anomaly Detection

Documentation
Brief explanation of parameters

Nice People at Work

Introduction

Automatic anomaly detection and classification is a useful tool for any company, but especially so for those ones which depend on quality. In order to best suit this tool for our clients, our new version of the SmartAlerts module has been designed with customization in mind. The goal of this documentation is explaining the most important parameters for SmartAlerts, their meaning and their relevance in detecting quality problems.

Contents

1 A brief explanation of the system

This system was constructed with three separate parts in mind: detecting, classifying and watching. Each part is independent of the rest, with its own input and output.

1.1 Detecting

By analyzing historical data, the system is able to set up a general shape that events take. This allows us to know whether, for example, a buffer ratio of 30% is something to be expected or not.

input A stream of events from Kafka

output A collection of anomalous events

1.2 Classifying

A machine learning process allows us here to find which combination of entities (such as Barcelona + Akamai) is the most relevant every minute. An issue is created for every combination, storing some information such as importance, time of creation or views affected.

input A collection of anomalous events

output A collection of issues to watch

1.3 Watching

Finally, the issues are watched over time. Every minute, their importance gets updated with the information coming from the previous part, and, if a threshold is overcome, an alert is created.

input A collection of issues to watch

output Alerts

2 Parameters

Right now the parameters need to be changed directly in the database via a script (see commands in section 3). The table `anomaly_detection_system_properties` holds the following:

Field	Type	Null	Key
system_id	int(10) unsigned	NO	PRI
cooldown	float unsigned	NO	
upper_bound	float unsigned	NO	
low_bound	float unsigned	NO	
ceiling	float unsigned	NO	
dimensions	varchar(150)	NO	
sliding_max_importance	float unsigned	NO	
min_views	int(10)	NO	

2.1 Definitions

`system_id` the identifier for the system

`cooldown` every minute, the importance of an issue gets multiplied by the cooldown value. This ensures that, over time, issues which aren't repeatedly watched get dropped.

`upper_bound` upon reaching the uppper bound, an issue is opened and the client receives an alert.

`low_bound` an open issue which drops below the lower bound is closed and saved in the historical table.

`ceiling` maximum total importance of an alert.

`dimensions` the possible filters on which the system operates

`sliding_max_importance` establishes the relative importance of the filter entity with the most concurrencies, with respect to the one with least.
For instance, in the case $sliding\_max\_importance = 3$ if the greatest entity in the filter city is Barcelona, and the smallest one is Roquetes, then an anomaly in Barcelona is three times as important as one in Roquetes.

`min_views` the minimum amount of anomalies necessary to open an alert.

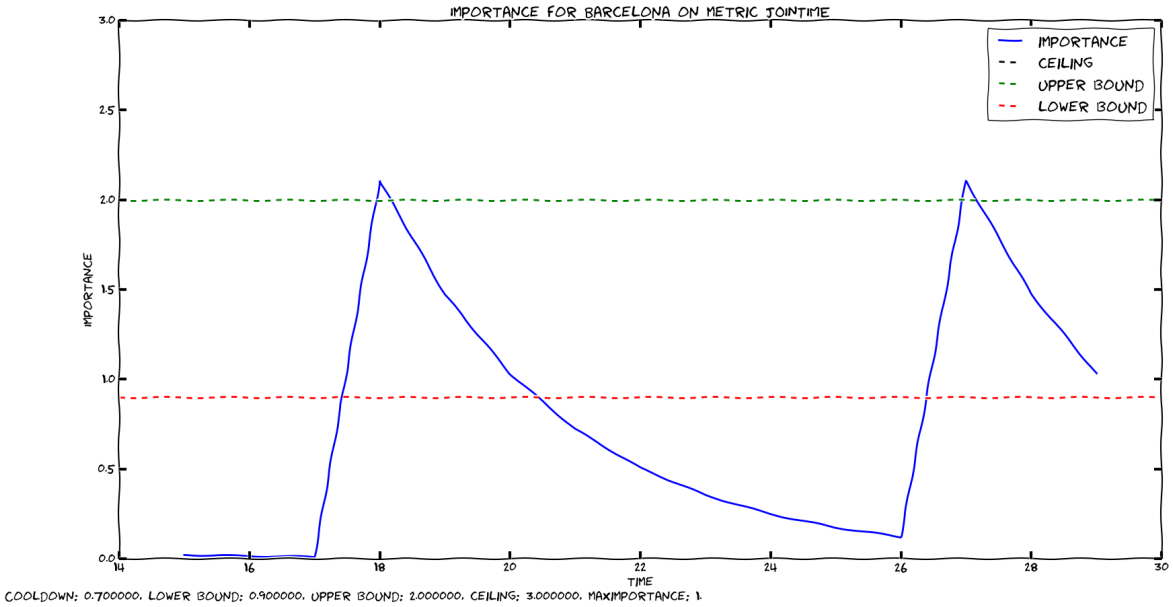

Figure 1: Importance for `cooldown 0.7, low_bound 0.9, upper_bound 2, max_importance 1`

2.2 Effects

As seen in Figure 1, the issue becomes more important when it gets new information, and it progressively loses importance otherwise. The green and red boundaries mark the points in time when the alert gets opened and closed. In this case, it opened twice.

3 Commands

Here's what happens upon calling the `./anomalyDetectionDatabaseScript.sh`:

```
This is a script to set system options in SmartAlerts.
Please start by specifying a system code. Then, the options to set are:

-t|--toggle [0/1]
Toggles a system on or off.

-m|--modify [field] [value]
Modifies a field to the value set.

-c|--check
Shows the values set in the system.
```

nicepeopleatwork.com

NPAW
NICEPEOPLEATWORK