One model in production is worth *two* in the notebook

Ivan Marin

Daitan GROUP
Accelerating™

The task

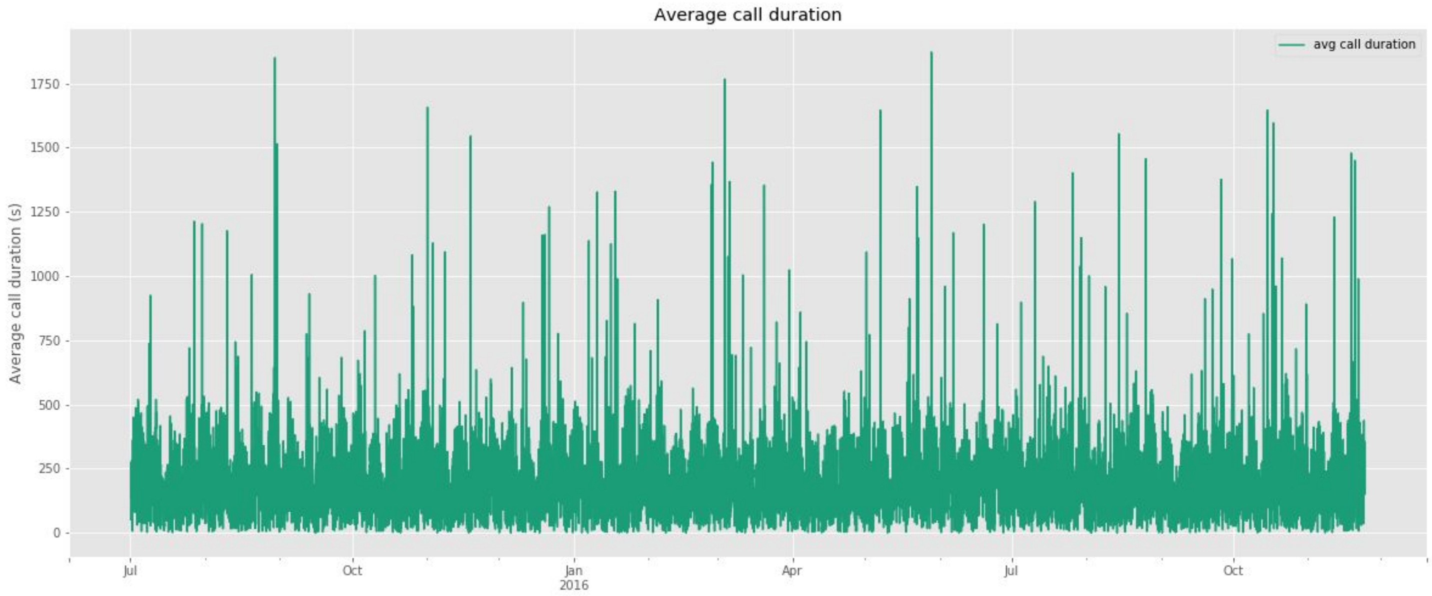
Detect anomalies in CDR data in VOIP/SIP to **identify** and **block attacks** and **fraud**

The data

	callingnum	callednum	record_type	ip	trunk_group	duration	MOS	gateway
2015-07-01 00:00:00	193303371	785879295	1	219.245.183.194	4lUz1b10L33fG	30.042124	3.951128	PM6wtCwaCuH5t9
2015-07-01 01:00:00	842294305	122111267	1	10.126.24.188	quXEtDXaK2	92.967500	3.954967	Sr6HibmQl0B
2015-07-01 02:00:00	291759817	527913027	1	210.63.7.55	taHigBvViXJ2	80.563725	3.785789	Y6HWS1B0Qg9Vt4r
2015-07-01 03:00:00	309013819	926158872	1	17.122.205.13	Tv3zqzYmzurRD	52.071217	3.499770	KST6XwU2c7uJM
2015-07-01 04:00:00	157320309	203758568	1	7.207.194.242	Pm3TnCThN9	61.975864	3.961631	QJmiSZtxdu1UDQ
2015-07-01 05:00:00	977804131	269439711	1	246.3.250.158	4lUz1b10L33fG	143.047730	3.992199	p4s79l5qD2zHOUx
2015-07-01 06:00:00	687866301	196609304	1	13.101.27.59	oMUv3mZn7uMUXn	272.676990	3.947686	KO15HluQHo
2015-07-01 07:00:00	098428369	703244667	1	253.169.82.113	N0y2NQsOuCQeLa	140.243935	3.993461	0XqlQ6Em4Om4n
2015-07-01 08:00:00	557868839	507158133	1	253.169.82.113	N0y2NQsOuCQeLa	200.851378	3.996015	0DzRGMEKXgcaS
2015-07-01 09:00:00	254369388	172132628	1	253.169.82.113	eMFOlieV5HeXpy	259.480745	3.999703	Sr6HibmQl0B

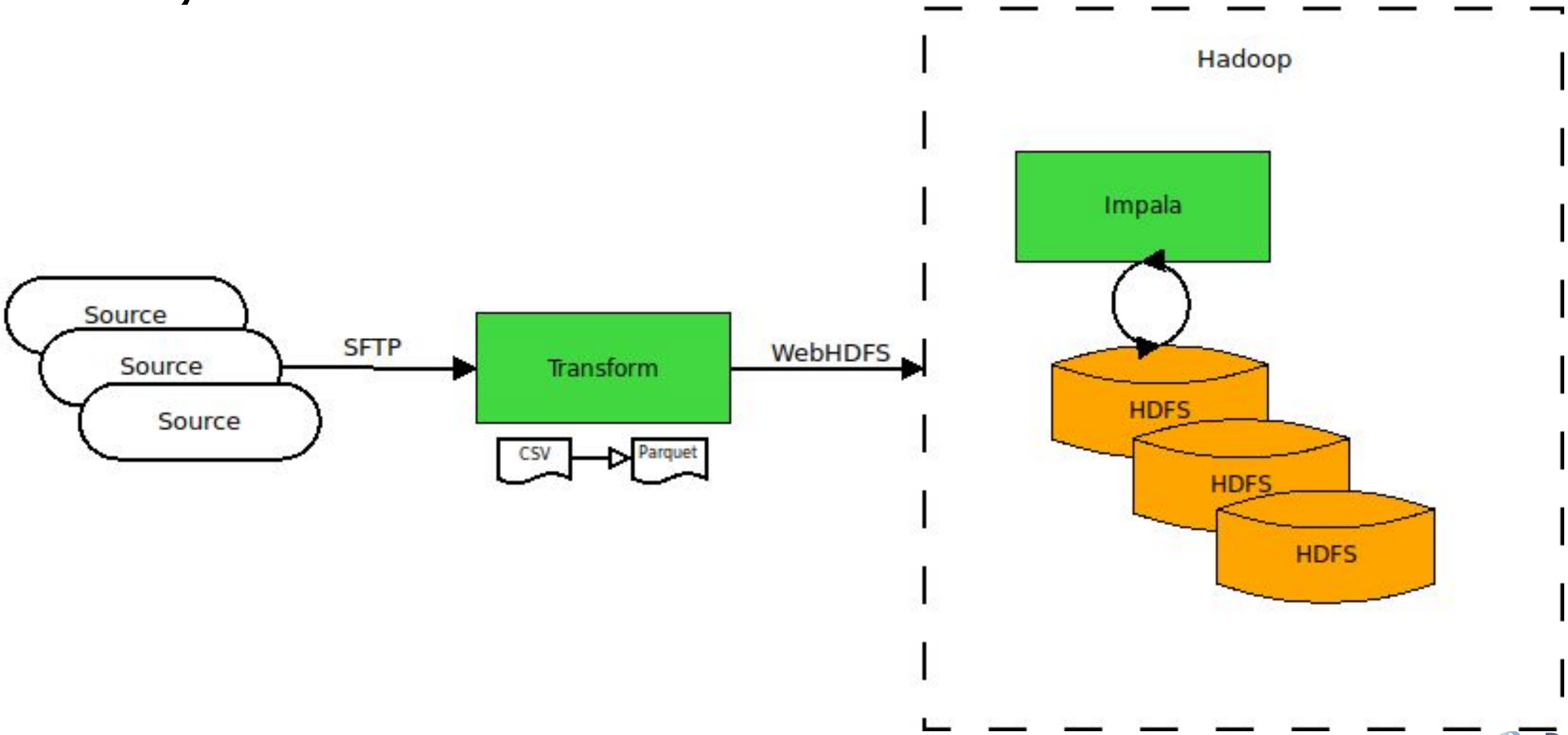
Daitan GROUP Accelerating

The data



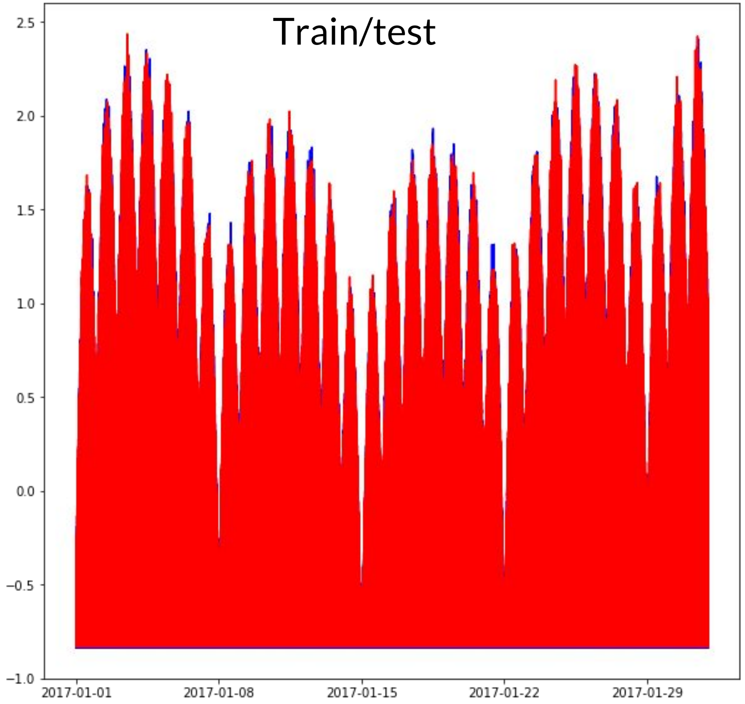
- *unsupervised* (no labeled data)
- flexible (different customers with different behaviors)

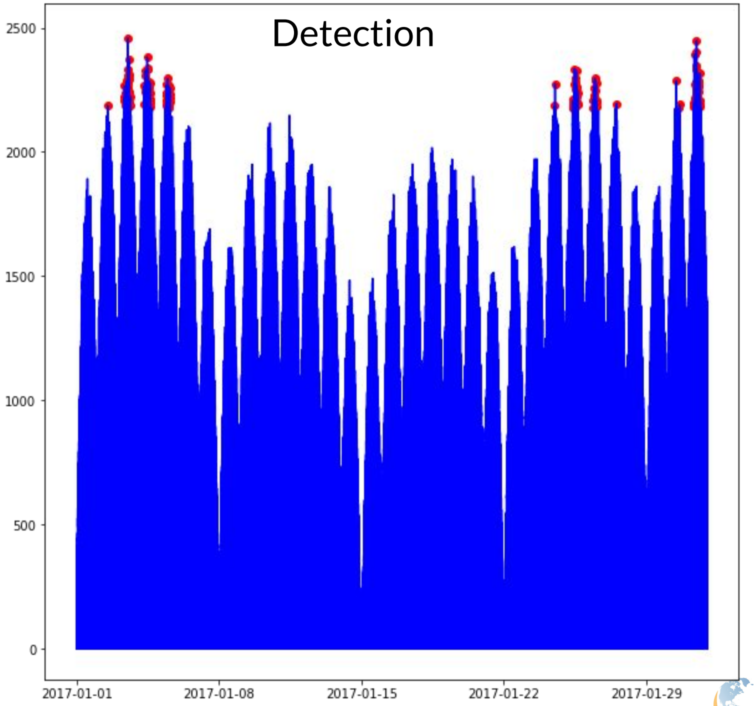
The layout


Hadoop
Impala
Source
Source
Source
SFTP
Transform
WebHDFS
CSV
Parquet
HDFS
HDFS
HDFS

Developing models

The first model: One Class SVM





The first model: One Class SVM

The One Class SVM approach failed.

- Low accuracy (less than 60%)
- Feature engineering didn't help (and boy we tried)

To make matters worse, there is no implemented parallel version of OCSVM.

The KPI approach

Instead of going first with raw data, we decided to go then with some KPIs:

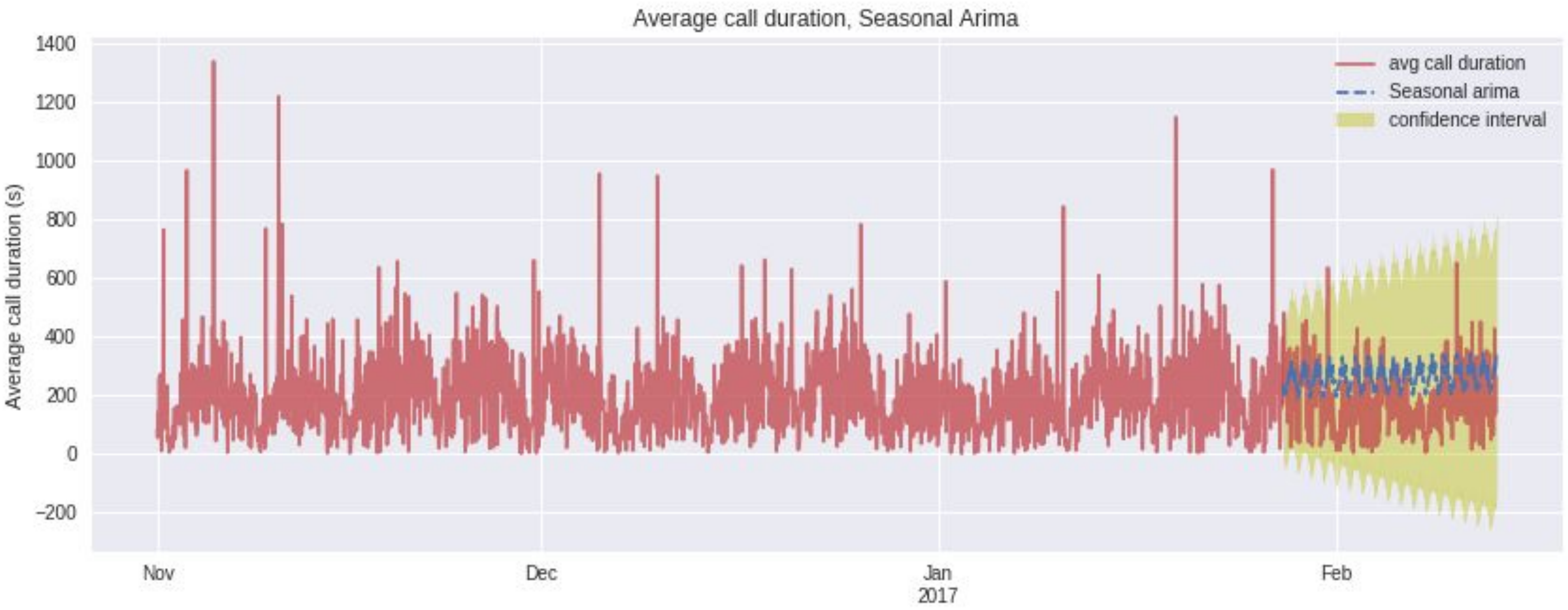
- **Average Call Duration**
- Bids
- MOS (Mean Opinion Score)
- ACHT (Average Call Holding Time)
- Post Dial Delay

The second model: KPI time series models

Box-Jenkins approach for Arima model:

- Check for stationarity
- Autocorrelation plots
- Partial autocorrelation plots
- Differentiate the series
- Fit the model

The second model: KPI time series models



The second model: KPI time series models

The ARIMA approach didn't perform well:

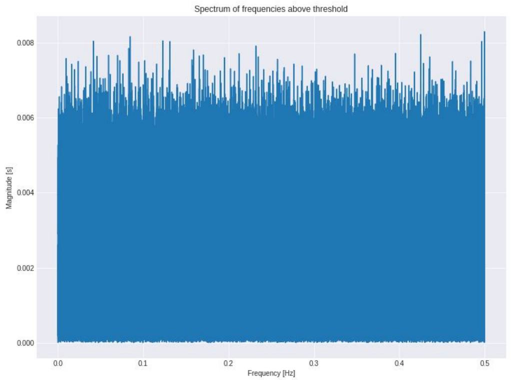
- High RMSE and MAE
- Small forecast horizon
- Not useful as baseline for anomaly detection

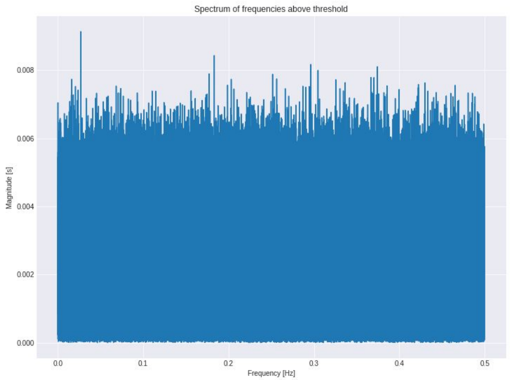
The third model: KPI frequency based model

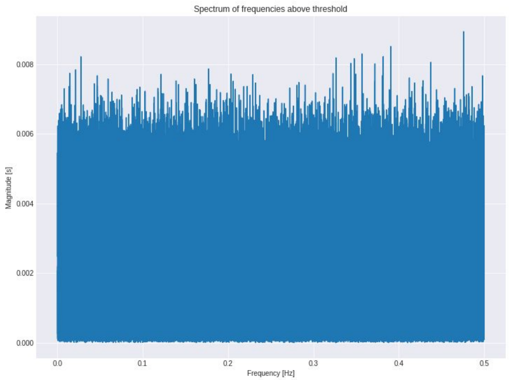
Hypothesis: Normal traffic has different frequency distribution from anormal traffic

- Decompose each KPI into the frequency domain
- Analyse the spectral signature
- Apply a threshold that separates anomalies from normal data

The third model: KPI frequency based model

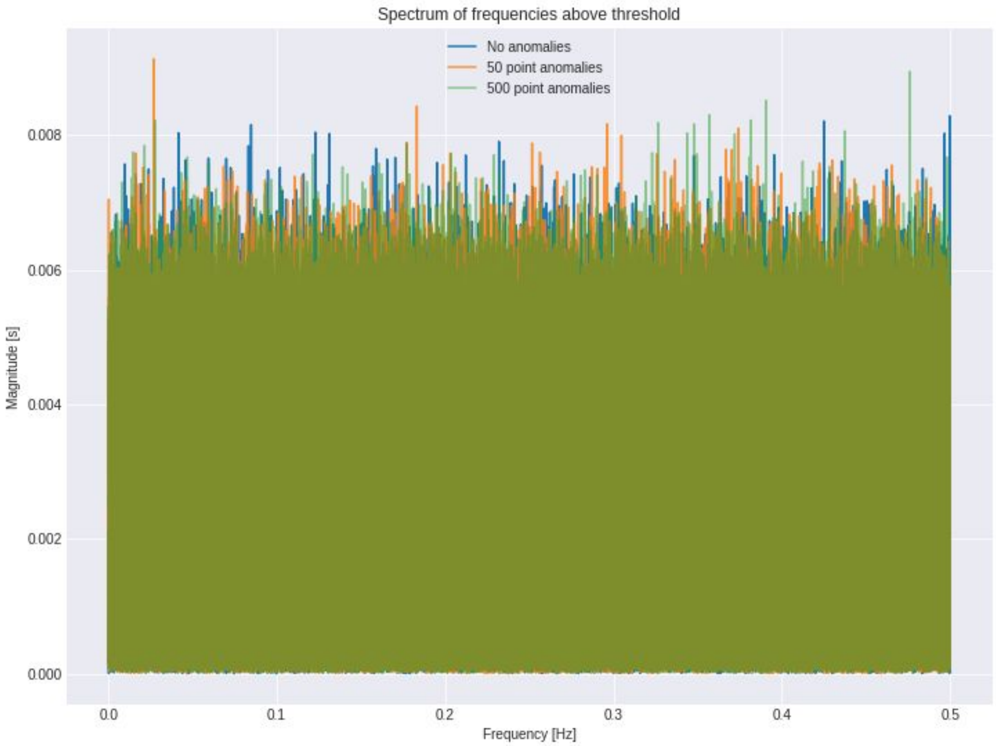






The third model: KPI frequency based model

Well, it failed again.



The Nth model

We continued testing other modeling approaches for detecting anomalies:

- Using more than one KPI at the same time
- Deep Learning (Feed forward)
- Entropy based methods

They all were not acceptable.


NEVER GIVE UP
NEVER SURRENDER

Going simple

What if we are going in the wrong direction?

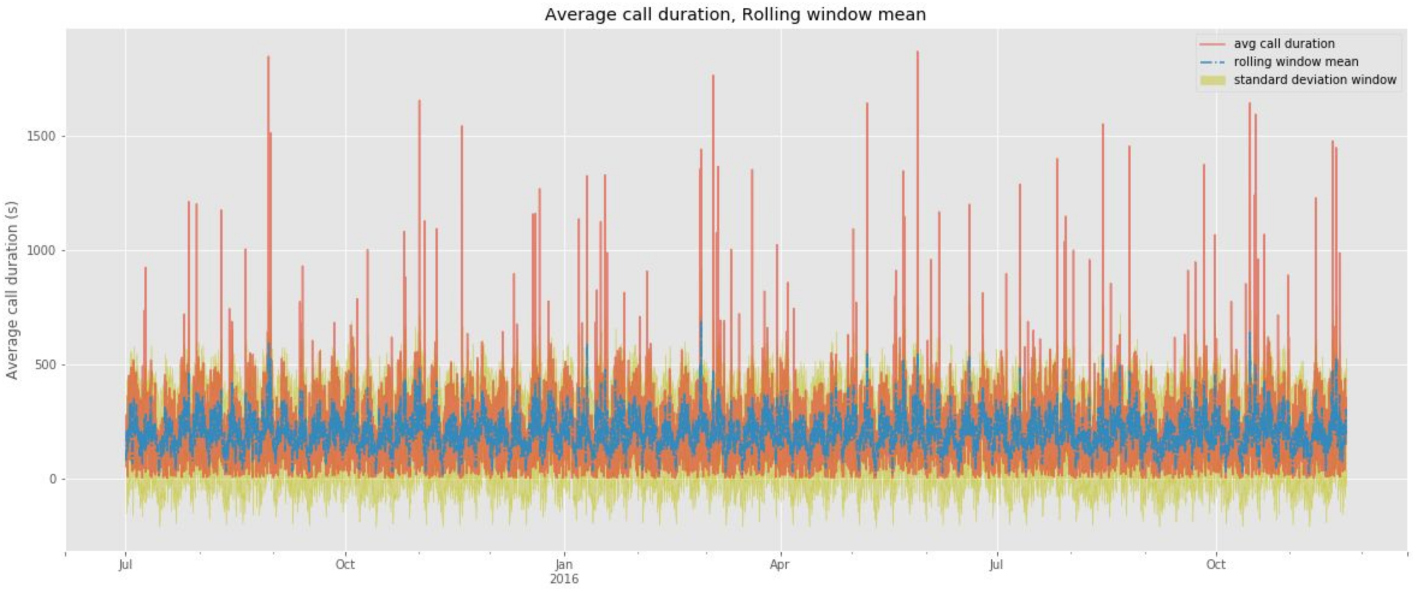
All models that we tested for anomaly detection so far are

- (relatively) complex
- possibly slow
- depend on external tools and frameworks (Spark, SkLearn, TensorFlow)

Daitan GROUP Accelerating

Going simple: rolling average model

We reverted back to simple statistics: the *rolling average.*



Going simple: rolling average model

The threshold was determined by a combination of sensitivity and deviation from the mean:

- Count the number of anomalies given a threshold
- Sum the difference between the anomalies and the rolling average
- Adjust the threshold to a customer comfort level

Going simple: rolling average model

Yes!

- But the rolling mean was not very sensitive to long tail events
- We need to take into account events that have a longer time window

Going simple: Exponential moving average

Exponential moving average

$$EM_m = \alpha * y[m] + (1 - \alpha) * S_{m-1}$$

$$\delta = y_i - EM_{i-1}$$

$$EM_i = EM_{i-1} + \alpha * \delta$$

$$S_i = (1 - \alpha) * (S_{i-1} + \alpha * \delta^2)$$

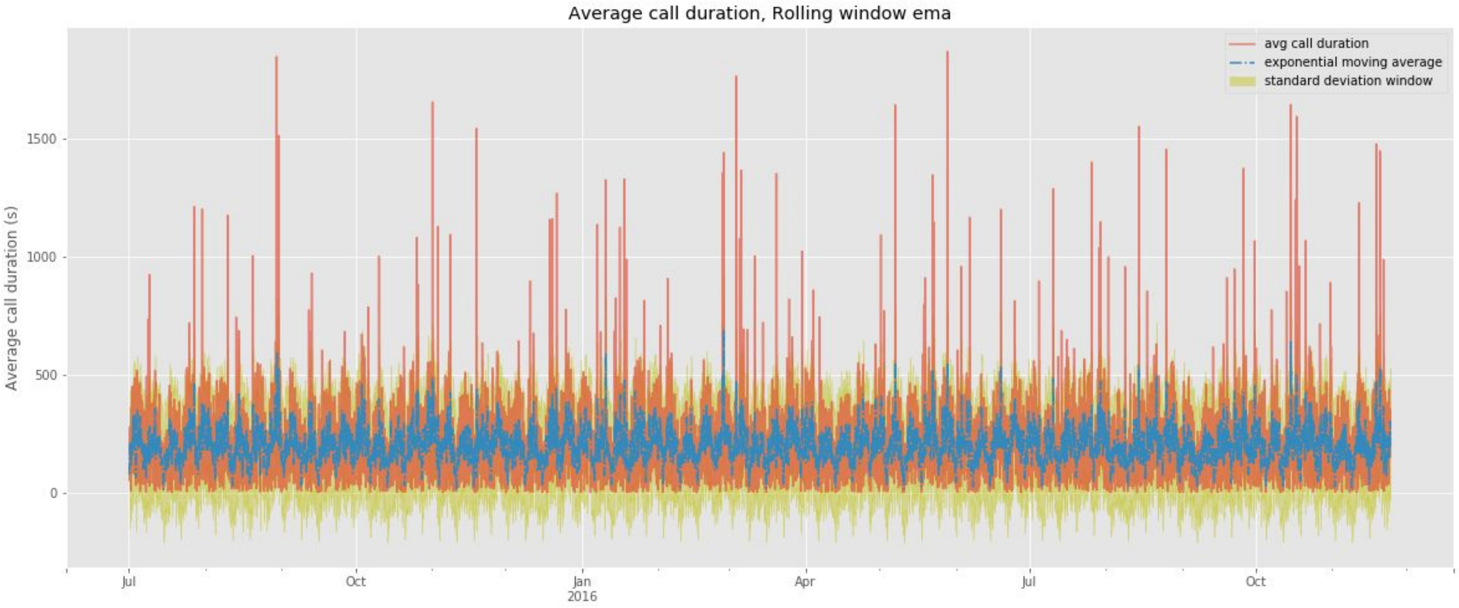
The method to calibrate the threshold is the same as the rolling average

Going simple: Exponential moving average

It works.

- Acceptable number of false positives and false negatives
- two parameters to calibrate: alpha and number of deviations
- Simple to implement

Going simple: Exponential moving average



The question

Should we invest more time in refining the working models or keep searching for a better one?

And how should we implement the chosen model?

The decision

We decided to go with the simple model:

- We had a short time to get the model out of the notebooks
- Another team would handle the transition to production
- There were several constraints in how we could deploy any model

So, how should we implement it?

Getting to Production

Getting to Production

Or, going from the notebook to production. How?

Constraints:

- Quick to be implemented by the Dev team
- Had to read and write to PostgreSQL
- Would be called multiple times
- No new services

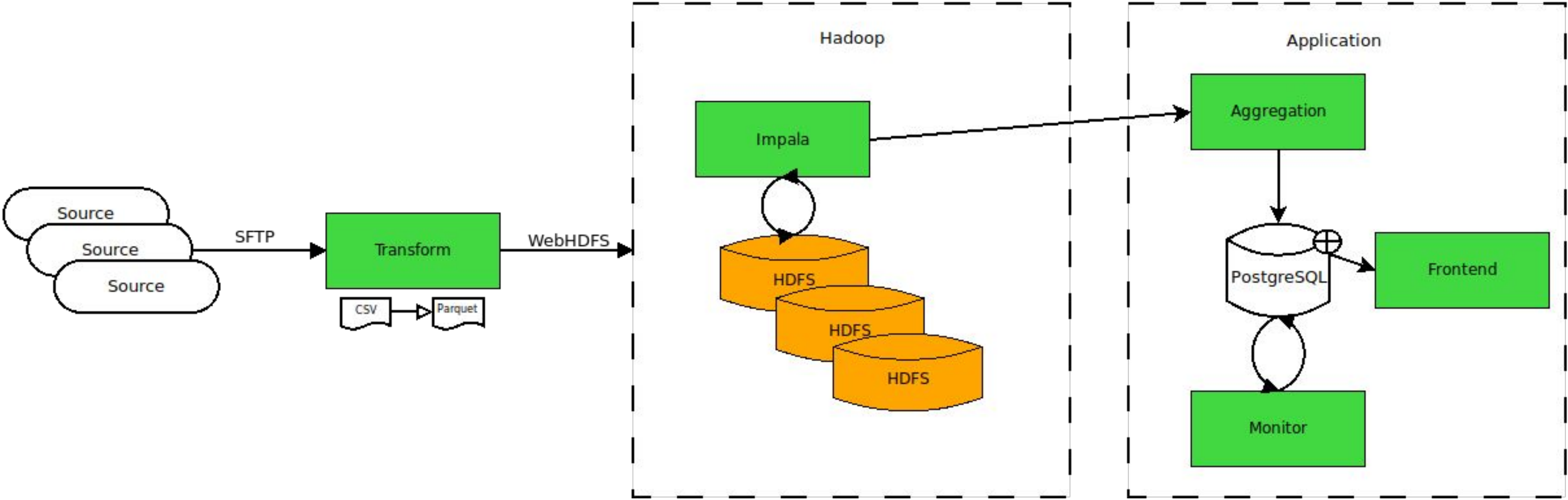
Getting to Production

The solution:

Java application and ***PostgreSQL stored procedures***

- The Java application keeps track of the anomalies and the alert flow
- The EMA/EMV algorithms are implemented in PSQL and called using JDBC
- The input tables and anomaly profile tables were written directly on PostgreSQL

Getting to Production



Getting to Production

Yes, we implemented the online EMA/EMV algorithms in stored procedures.

- Satisfied all requirements
- This is the system currently being in use in production
- Used the skills already available on the team

Considerations

- It's important to test your hypothesis and models

Considerations

- It's important to test your hypothesis and models

- Careful with what tools you use to develop your model - they may be not available in production!

Considerations

- It's important to test your hypothesis and models

- Careful with what tools you use to develop your model - they may be not available in production!

- Even the best model may not enter in production

Considerations

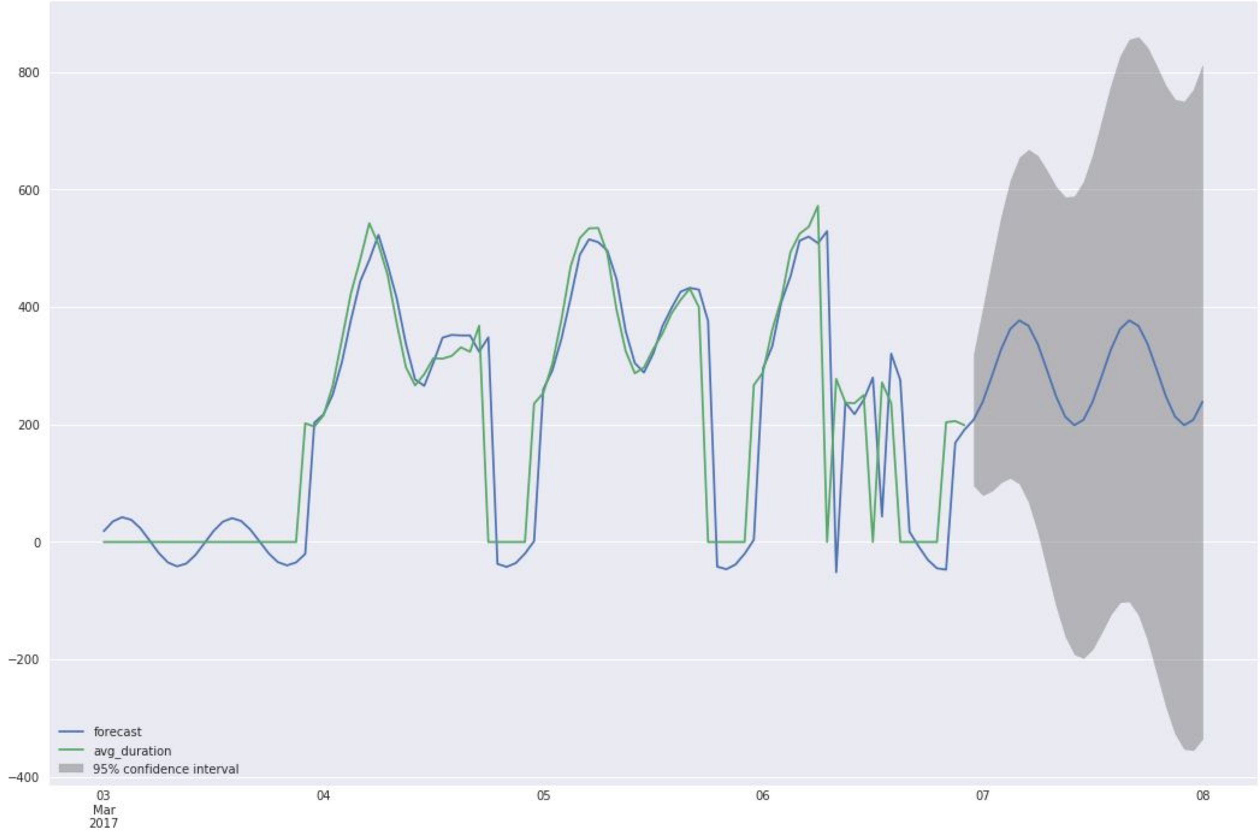
- It's important to test your hypothesis and models
- Careful with what tools you use to develop your model - they may be not available in production!
- Even the best model may not enter in production
- Think of the developers.

So, to finish, a simple question:

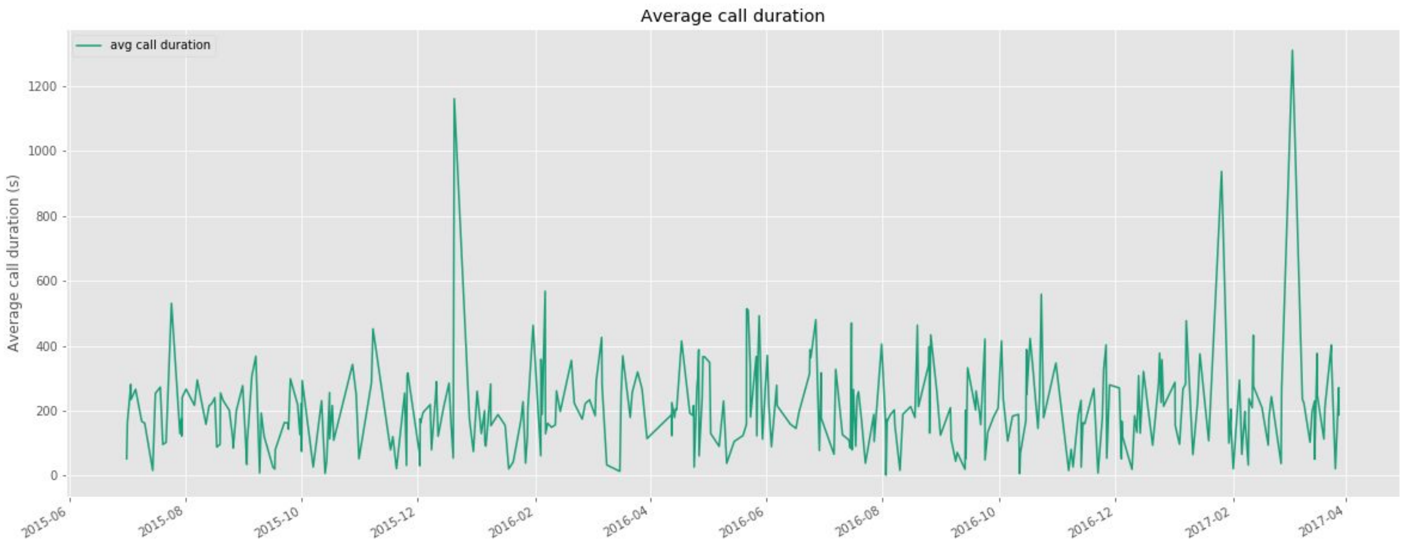
How do you get your models out of the notebooks and into production?

Extra

The second model: KPI time series models

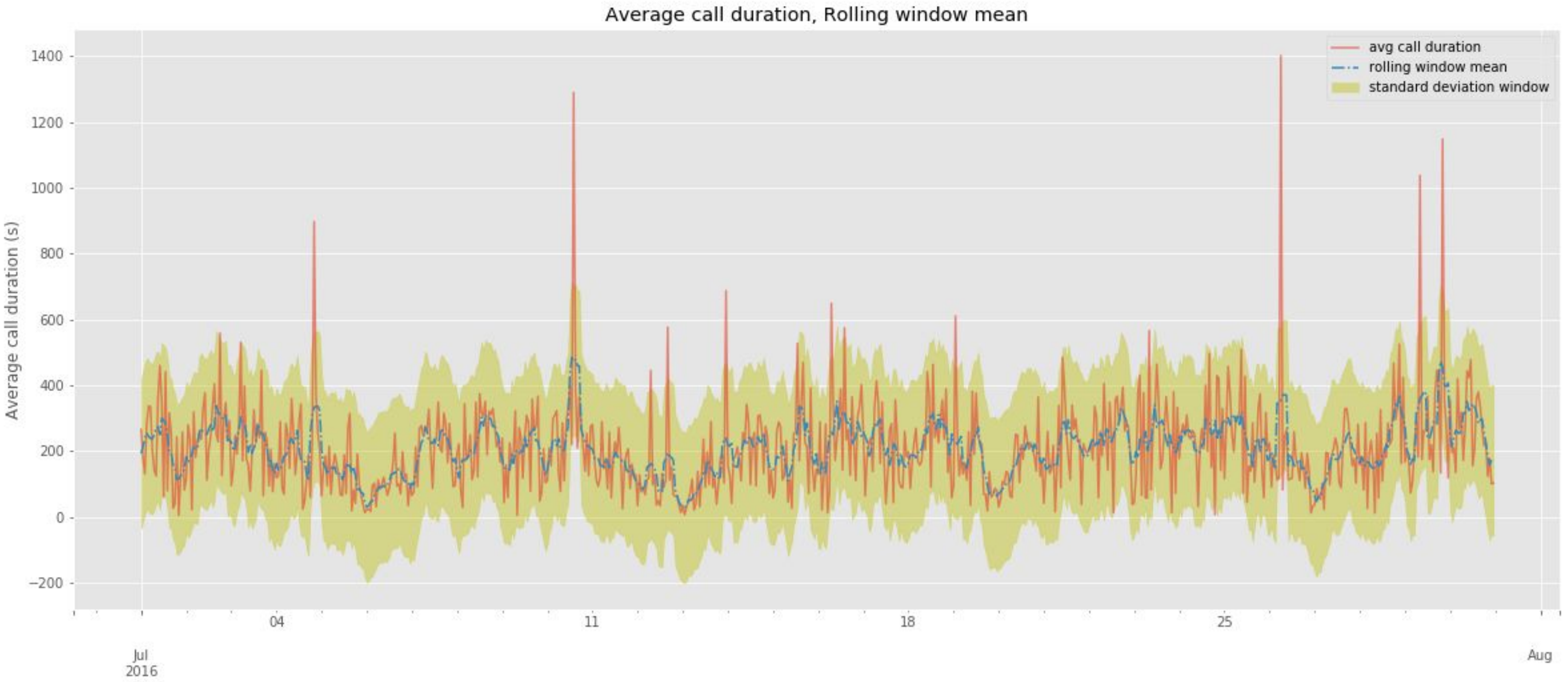


The data



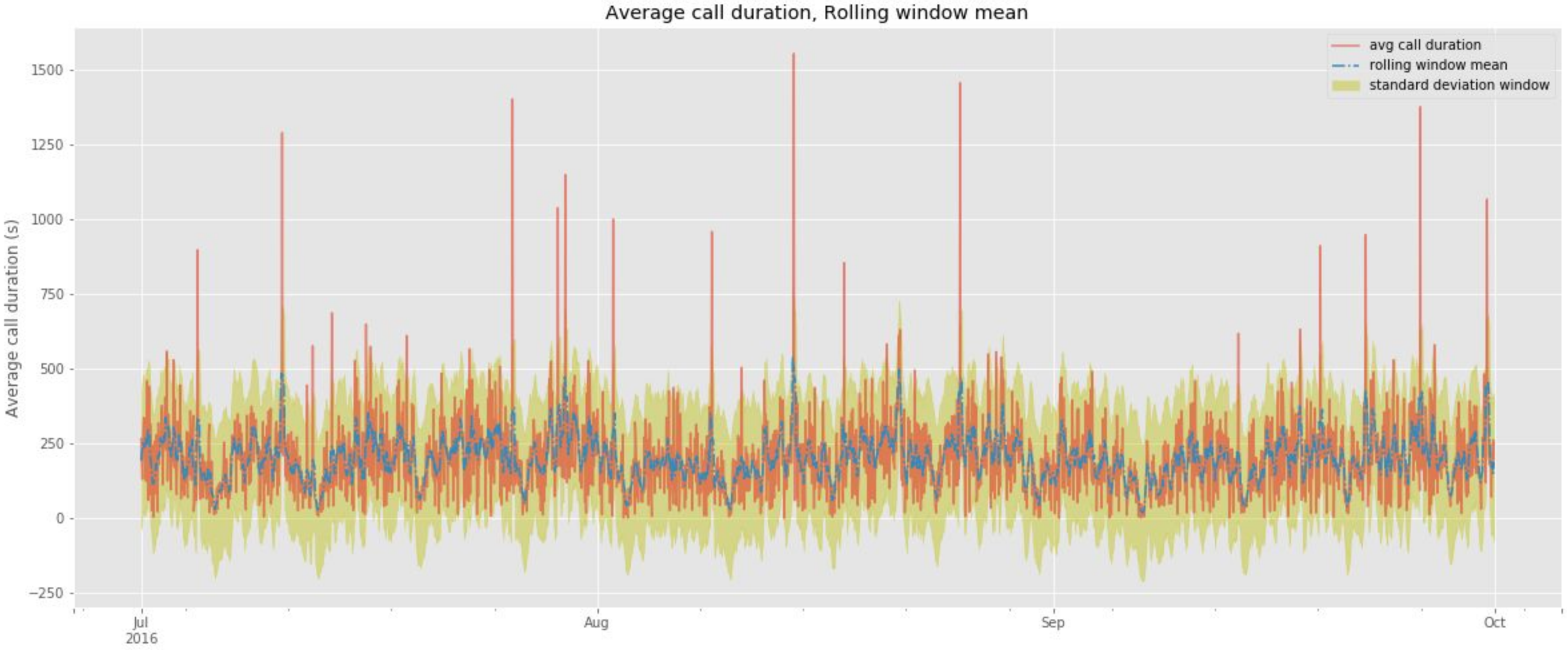
- One trunk group

Models

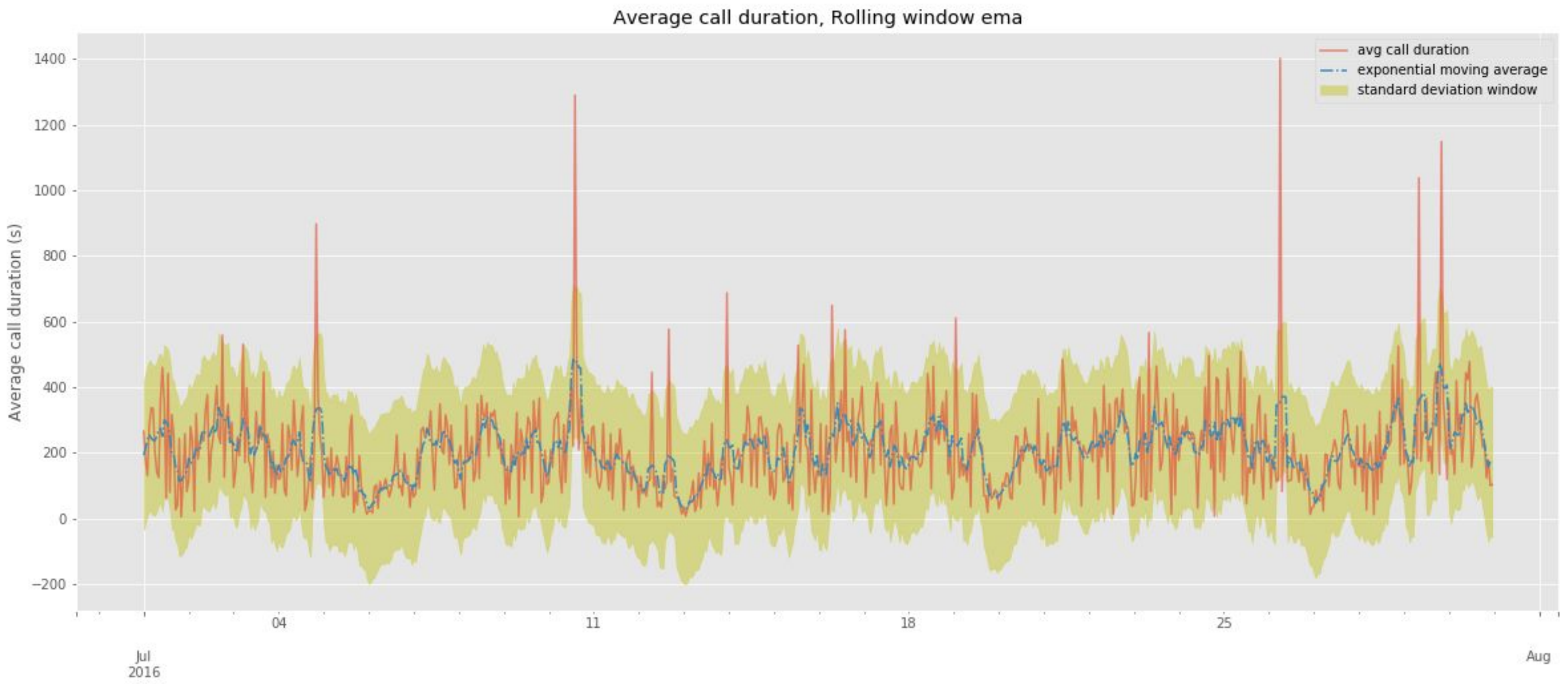


Average call duration, Rolling window mean
avg call duration
rolling window mean
standard deviation window
Average call duration (s)
1400
1200
1000
800
600
400
200
0
−200
04
11
18
25
Jul
2016
Aug

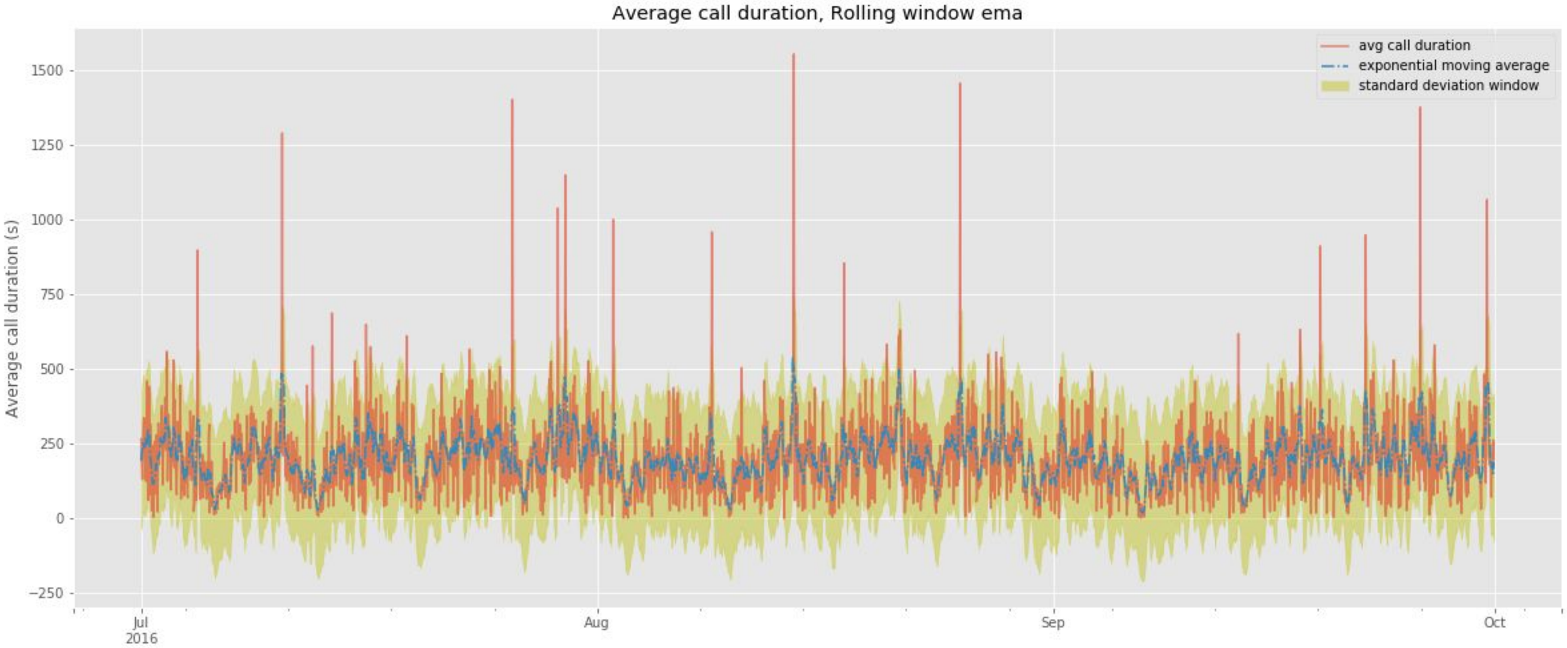
Models


Average call duration, Rolling window mean
avg call duration
rolling window mean
standard deviation window
Average call duration (s)
1500
1250
1000
750
500
250
0
−250
Jul
2016
Aug
Sep
Oct

Models


Average call duration, Rolling window ema
avg call duration
exponential moving average
standard deviation window
Average call duration (s)
1400
1200
1000
800
600
400
200
0
−200
Jul
2016
04
11
18
25
Aug

Models



Average call duration, Rolling window ema
avg call duration
exponential moving average
standard deviation window
Average call duration (s)
1500
1250
1000
750
500
250
0
−250
Jul
2016
Aug
Sep
Oct

EMA/EMV algorithm