ELASTICSEARCH MACHINE LEARNING

Here we discuss ElasticSearch Machine Learning. ML is an add-on to ElasticSearch that you can purchase with a standalone installation or pay as part of the monthly Elastic Cloud subscription.

(This article is part of our ElasticSearch Guide. Use the right-hand menu to navigate.)

ElasticSearch Machine Learning

The term **machine learning** has a broad definition. It is a generic term handed over to the laymen as a way of avoiding discussing the specifics of the various models.

To be specific what ElasticSearch ML does is **unsupervised learning time series analysis**. That means it draws conclusions from a set of data instead of using training a model (i.e., **supervised learning**) to make predictions, like you would with regression analysis using different techniques, including **neural networks**, **least squares**, or **support vector machines**.

How is this Useful

To see the value of **unsupervised learning time series analysis** consider the typical approach to cybersecurity or application performance monitoring. That is to assume data follows a normal (gaussian) distribution and then select some threshold that one considers significant when flagging outliers.

This is the familiar **bell curve** approach. Even if one does not think of it that way, that's what they are doing. This curve is described by terms the highschool math student should know: **mean**, **variance**, and **standard deviation**. But that is not machine learning.

Not Better than Guessing

Typically someone doing application monitoring or cybersecurity flags an event when it lies at either end of the curve, which is where the probability of such an event is low. That's usually referred to as some multiple of σ (**sigma**), where sigma is a multiple of standard deviations, where the probability of an event lying there is very low.

You could also with derision call this approach **guessing**.

The threshold approach is flawed, because it leads to **false positives**. That causes analysts to spend time tracking down events that are not truly **statistically significant**.

For example, with cybersecurity, just because someone is sending data to a particular IP address more now than before does not mean that event is out of bounds. They need to consider the cyclical nature of events and look at current events in light of what has come before them. It could be this happens every month and is normal. A clever algorithm can do that by, for example, applying a **least squares** method and looking to minimize the **error** (i.e., the difference between what is observed and what is expected), against a shifting subset of data. This is how ElasticSearch does it. And they apply several algorithms, not just least squares.

How to use ElasticSearch to do Machine Learning

Here we show how to use the tool. In another blog post we will explain some of the logic and algorithms behind it. But the tool is supposed to make it not necessary to understand all of that. Still, you should so you can trust what it is telling you.

We are going to draw from this video by ElasticSearch and slow it down to single steps to make it simpler to understand.

For data we will use the New York City taxi cab dataset that you can download from Kaggle here. The data gives us pick up and drop off times and locations for NYC cabs over a period of a few years. We want to see when traffic falls off or increases in such a way that is an abnormality, like a taxi strike or snowstorm.

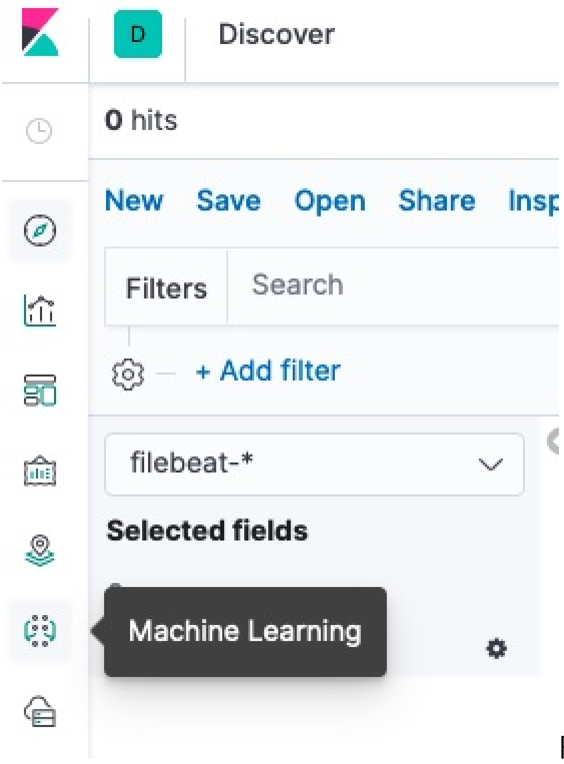
There are some limits with loading this data in the Kibana ML screen, which is a feature with ES ML. You can only load 100 MB of data at a time. The taxi cab data is much larger than that. It includes a **test** and **train** dataset. Since we are not training a model with unsupervised learning, we will just pick one of them. And since we need to limit it to 100 mb we will split the 200 mb data file like this and just pick one 90MB dataset.

```
unzip train.zip
split -b 90000000 train.csv
```

Also note that ElasticSearch tends to freeze up when you load data like this, unless you have a large cluster. But it still loads the data. So once it looks like it has finished loading, meaning the screen no longer updates, just click out of it and to go to **Index management** to see how many records are in the new index. It should be about 100,000 for each 90MB of data.

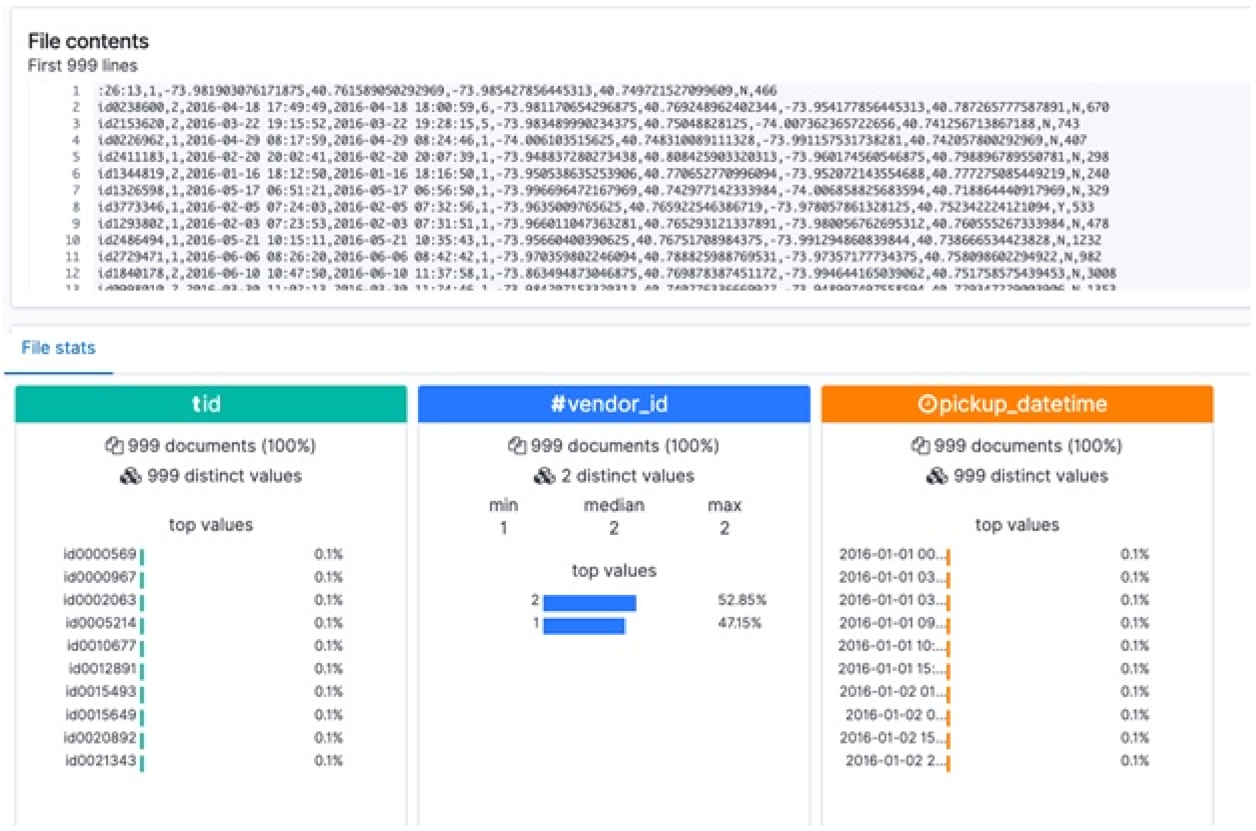
Upload the Data into the Machine Learning Screens

Open Kibana and click on the **Machine Learning** icon. You will have that icon even if you don't have a trial or paid license. But what will be missing is the **Anomaly Detection, job creation**, and other screens. So ElasticSearch will guide you through signing up for a trial.



From the **Data Visualizer** select **Import a file** and import one of the files you split from train.csv. Or if you loaded the data a different way you can use the **Select an Index Pattern** option.

ElasticSearch will show you the first 1,000 rows and then make some quick record counts.


File contents
First 999 lines

```
:26:13,1,-73.981903076171875,40.761589050292969,-73.985427856445313,40.749721527099609,N,466
id0238600,2,2016-04-18 17:49:49,2016-04-18 18:00:59,6,-73.981170654296875,40.769248962402344,-73.954177856445313,40.787265777587891,N,670
id2153620,2,2016-03-22 19:15:52,2016-03-22 19:28:15,5,-73.983489990234375,40.75048828125,-74.007362365722656,40.741256713867188,N,743
id0226962,1,2016-04-29 08:17:59,2016-04-29 08:24:46,1,-74.006103515625,40.748310089111328,-73.991157531738281,40.742057800292969,N,407
id2411183,1,2016-02-20 20:02:41,2016-02-20 20:07:39,1,-73.948837280273438,40.808425903320313,-73.960174560546875,40.798896789550781,N,298
id1344819,2,2016-01-16 18:12:50,2016-01-16 18:16:50,1,-73.950538635253906,40.770652770996094,-73.952072143554688,40.777275085449219,N,240
id1326598,1,2016-05-17 06:51:21,2016-05-17 06:56:50,1,-73.996696472167969,40.742977142333984,-74.006858825683594,40.718864440917969,N,329
id3773346,1,2016-02-05 07:24:03,2016-02-05 07:32:56,1,-73.963500976562500,40.765922546386719,-73.978057861328125,40.752342224121094,Y,533
id1293802,1,2016-02-03 07:23:53,2016-02-03 07:31:51,1,-73.966011047363281,40.765293121337891,-73.980056762695312,40.760555267333984,N,478
id2486494,1,2016-05-21 10:15:11,2016-05-21 10:35:43,1,-73.95660400390625,40.76751708984375,-73.991294860839844,40.738666534423828,N,1232
id2729471,1,2016-06-06 08:26:20,2016-06-06 08:42:42,1,-73.970359802246094,40.788825988769531,-73.97357177734375,40.758098602294922,N,982
id1840178,2,2016-06-10 10:47:50,2016-06-10 11:37:58,1,-73.863494873046875,40.769878387451172,-73.994644165039062,40.751758575439453,N,3008
```

File stats

tid	#vendor_id	@pickup_datetime
999 documents (100%)	999 documents (100%)	999 documents (100%)
999 distinct values	2 distinct values	999 distinct values

tid — top values

id0000569	0.1%
id0000967	0.1%
id0002063	0.1%
id0005214	0.1%
id0010677	0.1%
id0012891	0.1%
id0015493	0.1%
id0015649	0.1%
id0020892	0.1%
id0021343	0.1%

#vendor_id

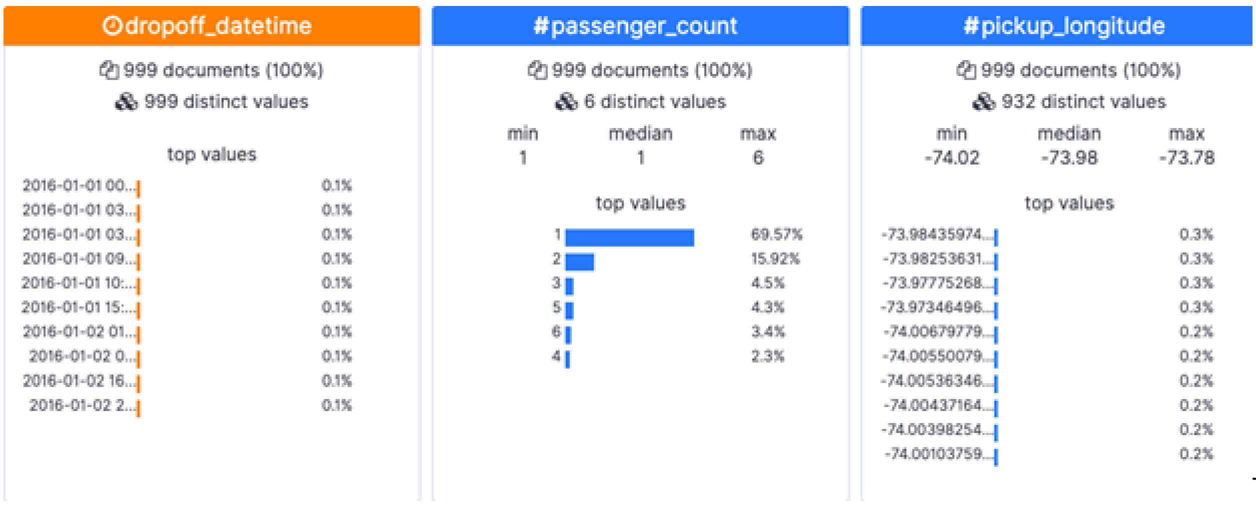
min	median	max
1	2	2

top values

2	52.85%
1	47.15%

@pickup_datetime — top values

2016-01-01 00...	0.1%
2016-01-01 03...	0.1%
2016-01-01 03...	0.1%
2016-01-01 09...	0.1%
2016-01-01 10:...	0.1%
2016-01-01 15:...	0.1%
2016-01-02 01...	0.1%
2016-01-02 0...	0.1%
2016-01-02 15...	0.1%
2016-01-02 2...	0.1%



@dropoff_datetime

999 documents (100%)
999 distinct values

top values

2016-01-01 00...	0.1%
2016-01-01 03...	0.1%
2016-01-01 03...	0.1%
2016-01-01 09...	0.1%
2016-01-01 10:...	0.1%
2016-01-01 15:...	0.1%
2016-01-02 01...	0.1%
2016-01-02 0...	0.1%
2016-01-02 16...	0.1%
2016-01-02 2...	0.1%

#passenger_count

999 documents (100%)
6 distinct values

min	median	max
1	1	6

top values

1	69.57%
2	15.92%
3	4.5%
5	4.3%
6	3.4%
4	2.3%

#pickup_longitude

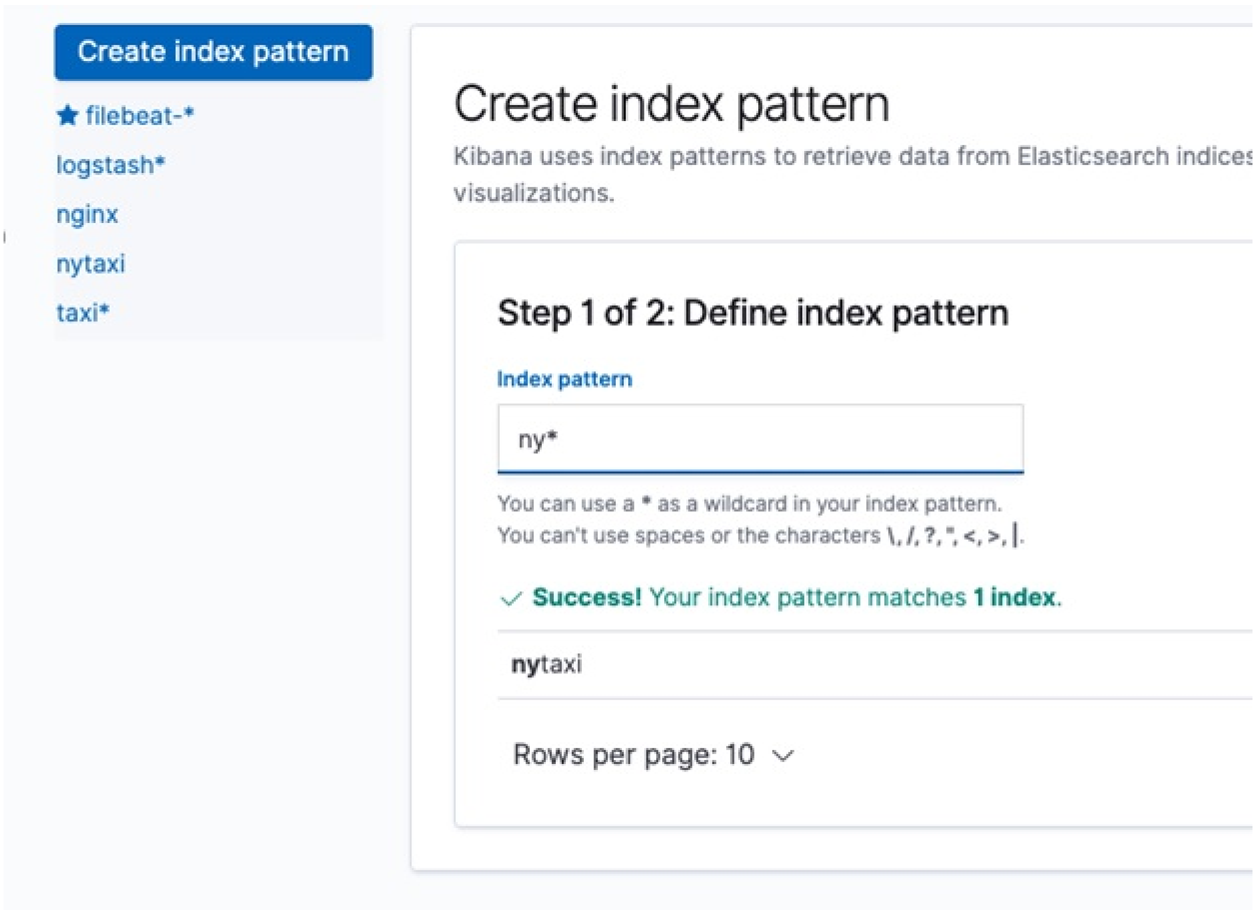
999 documents (100%)
932 distinct values

min	median	max
-74.02	-73.98	-73.78

top values

-73.98435974...	0.3%
-73.98253631...	0.3%
-73.97775268...	0.3%
-73.97346496...	0.3%
-74.00679779...	0.2%
-74.00550079...	0.2%
-74.00536346...	0.2%
-74.00437164...	0.2%
-74.00398254...	0.2%
-74.00103759...	0.2%

Then click **Import** at the bottom of the screen. Give it a name for the **Index Pattern** name, like **ny***.



Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices visualizations.

Step 1 of 2: Define index pattern

Index pattern

ny*

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

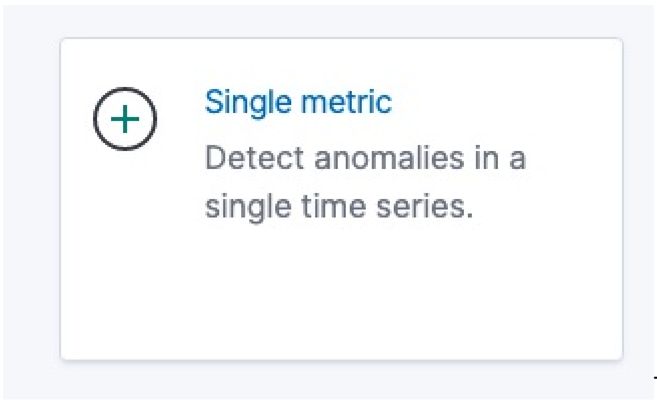
Success! Your index pattern matches 1 index.

nytaxi

Rows per page: 10

Anomaly Detection

Now we get to the interesting part. We want ElasticSearch to look at this time series data. Pick the **Single Metric** option.


Single metric

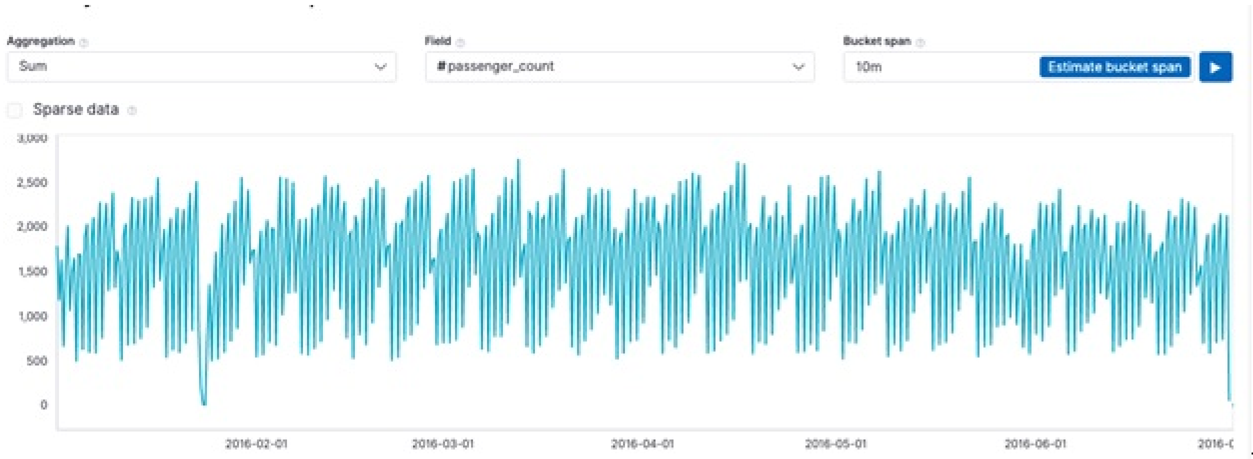
Detect anomalies in a single time series.

Then we have to create a job. ElasticSearch will run an **aggregation query** in the background. We tell it to **sum passenger_count** over time. (If the drop down box does not work just copy and paste the field name.)

We use this aggregation to observe when there is a drop off or spike in passengers that lies outside the normal range and that takes into consideration the normal rise and fall of passenger count over time.

Click the button Use **Full Range of Data**, so it will pick all the data available and not the date interval you manually put. You can change the **bucket option** to 10m.

Then the screen will fill out like this:

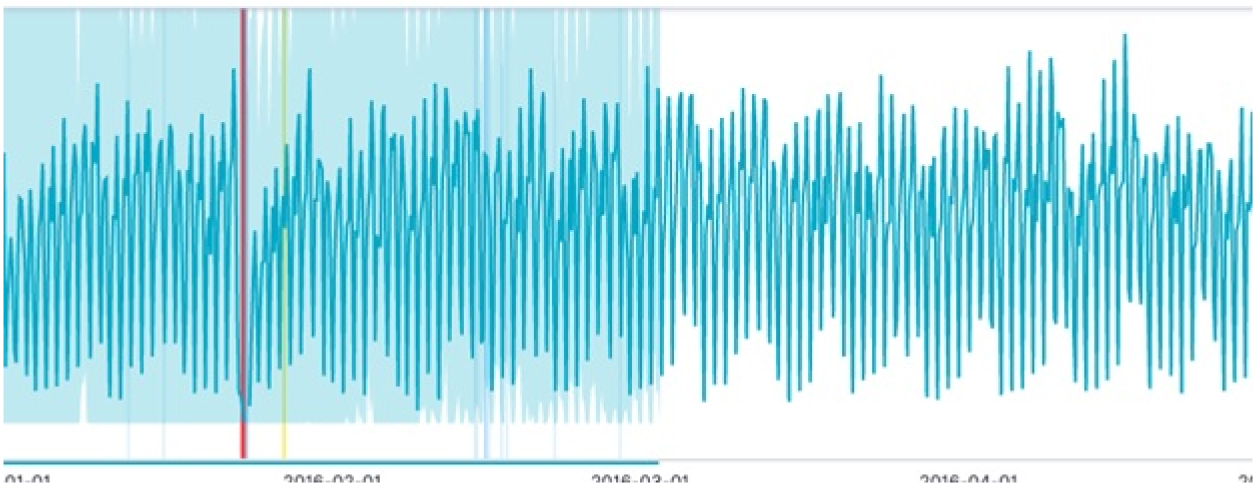

Aggregation: Sum

Field: #passenger_count

Bucket span: 10m

Sparse data

Then click Create Job. It starts running its algorithms and updating the display with vertical colored lines as it completes its logic.



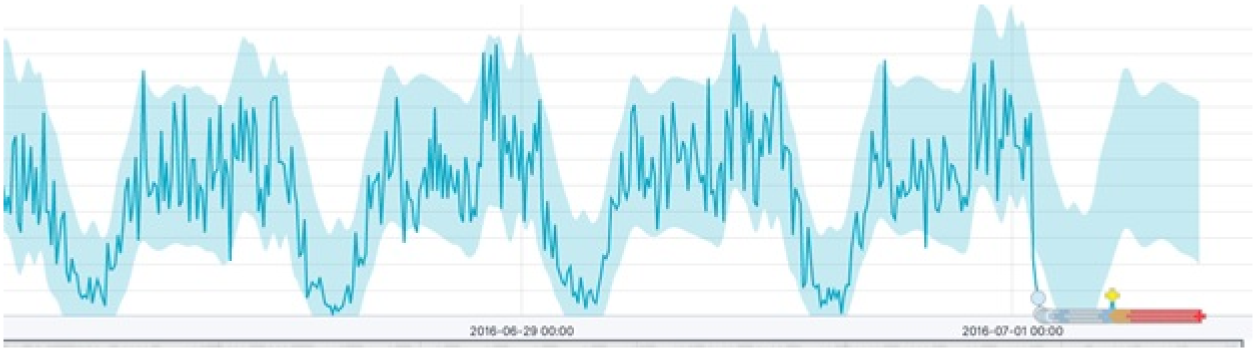
View the Anomaly Detection Results

When it is done click **View Results**.

We will explain in another post exactly what calculations it has done, after all you should not trust an ML model without having some understanding of how it drew its conclusions.

But you can think of it like this. Taxi rides go up and down during the week versus the weekend and rush hour versus not. If you were working with sales data you would call that **trends** or **seasonality**. So if you used a **gaussian** (normal) distribution against that it would be wrong, as the mean and variance would be over the whole set including the high and low points. So the improved algorithm slides along making multiple normal curves (and other frequency distributions) against subsets of the data to eliminate this up and down pattern.

This produces the results below. The light blue area is the shifting probability distribution function. The red area is where you can see is the anomaly, the point where the plot has gone outside the curve. It's also the point where we have run out of data. It has flagged that as an outlier by calculating an anomaly score, a point we will cover in the next post.


2016-06-29 00:00
2016-07-01 00:00