Common Machine Learning Solutions Everyone Needs to Know

Eurus Kim | Amir Malekpour

Wednesday, October 23, 2019

Eurus Kim
Staff ML Architect | Splunk

Amir Malekpour
Principal Software Engineer | Splunk

splunk> .conf19

Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or plans of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results may differ materially. The forward-looking statements made in the this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, it may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements made herein.

In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only, and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionalities described or to include any such feature or functionality in a future release.

Agenda

Understanding Machine Learning with Splunk

Solution 1 – Outlier Detection using DensityFunction

- Use cases covered
- What is Density Function?
- How to use DensityFunction

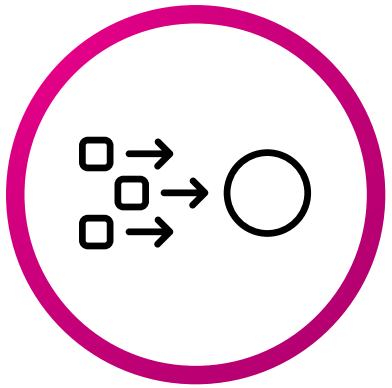
Solution 2 – Forecasting using StateSpaceForecast

- Understanding forecasting
- How to use StateSpaceForecast
- Caveats and considerations

Understanding Machine Learning with Splunk

splunk> .conf19

What is Machine Learning?



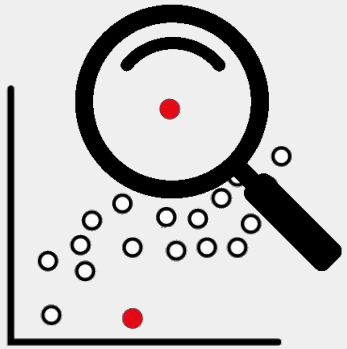
Use mathematical models to learn patterns in information

Catalog the patterns (and in some cases, iterate them as new data is received)

Use learned patterns to understand and interpret new data or make predictions

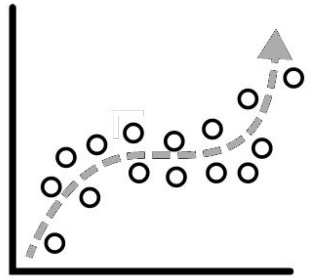
Splunk Customers Want Answers from their Data

Anomaly detection



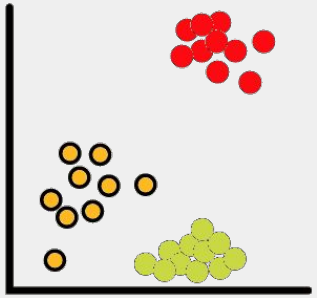
- Deviation from past behavior
- Deviation from peers
- (aka Multivariate AD or Cohesive AD)
- Unusual change in features

Predictive Analytics



- Predict Service Health Score/Churn
- Predicting Events
- Trend Forecasting
- Detecting influencing entities
- Early warning of failure

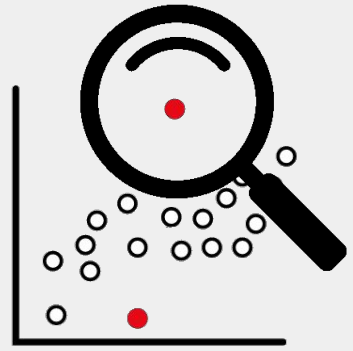
Clustering



- Identify peer groups
- Event Correlation
- Reduce alert noise
- Behavioral Analytics

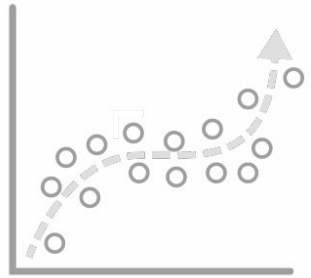
Splunk Customers Want Answers from their Data

Anomaly detection



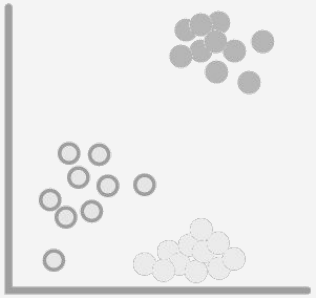
- Deviation from past behavior
- Deviation from peers
- (aka Multivariate AD or Cohesive AD)
- Unusual change in features

Predictive Analytics



- Predict Service Health Score/Churn
- Predicting Events
- Trend Forecasting
- Detecting influencing entities
- Early warning of failure

Clustering



- Identify peer groups
- Event Correlation
- Reduce alert noise
- Behavioral Analytics

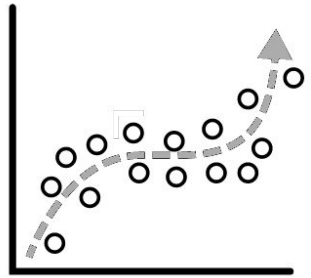
Solution #1

Splunk Customers Want Answers from their Data

Anomaly detection

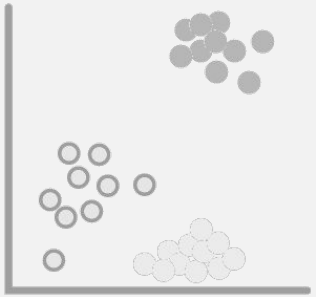
- Deviation from past behavior
- Deviation from peers
- (aka Multivariate AD or Cohesive AD)
- Unusual change in features

Predictive Analytics



- Predict Service Health Score/Churn
- Predicting Events
- Trend Forecasting
- Detecting influencing entities
- Early warning of failure

Clustering



- Identify peer groups
- Event Correlation
- Reduce alert noise
- Behavioral Analytics

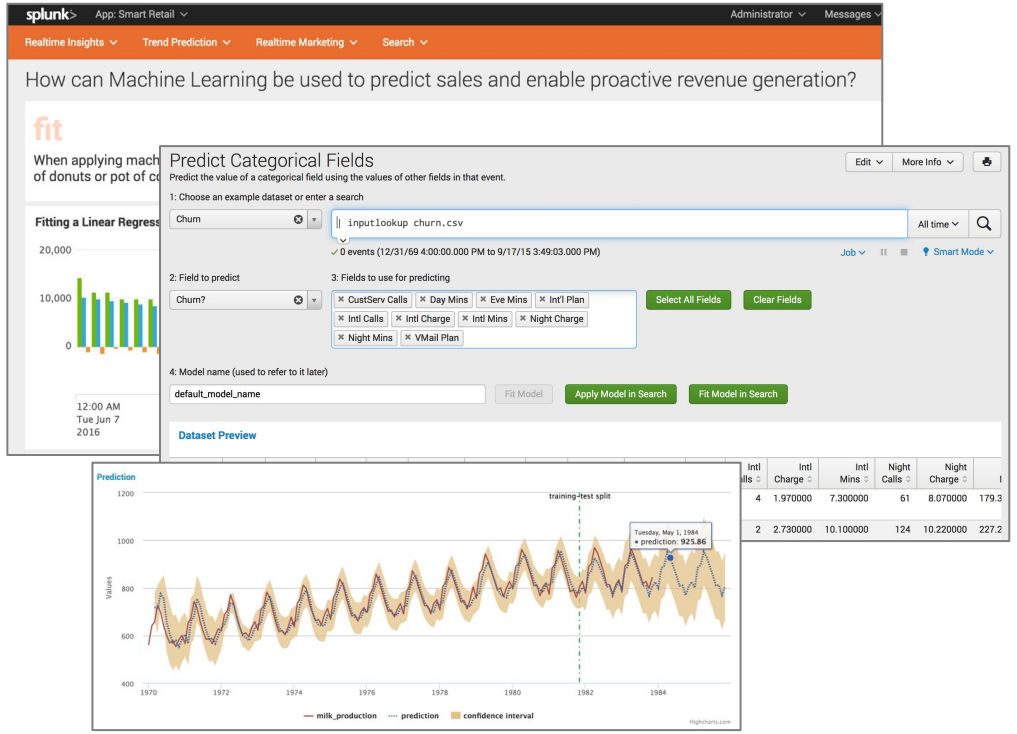
Solution #2

Overview of ML at Splunk

Overview of ML at Splunk

Splunk Machine Learning Toolkit (MLTK)

- **Experiments and Assistants:** Guided model building, testing, and deployment for common objectives
- **Showcases:** Interactive examples for typical IT, security, business, and IoT use cases
- **Algorithms:** 80+ standard algorithms (supervised & unsupervised)
- **ML Commands:** New SPL commands to fit, test, score and operationalize models
- **ML-SPL API:** Extensibility to easily import any algorithm (proprietary / open source)
- **Python for Scientific Computing Library:** Access to 300+ open source algorithms
- **Apache Spark MLLib:** Support large scale model training via Spark Add-on for MLTK (LAR)
- **Tensorflow Container:** Supports NN and GPU accelerated machine learning



Build custom analytics for any use case

splunk> .conf19

Solution #1: Outlier Detection Using DensityFunction

splunk> .conf19

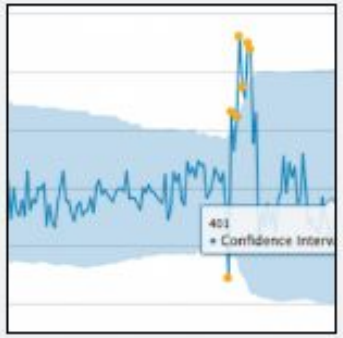
Solution 1 – Using DensityFunction

What type of use cases are we talking about?

Outlier in some numerical value

- Number of transactions
- Transaction latency
- System utilization (CPU/memory)
- Number of logins
- Amount of data transfer
- Time between actions
- Sensor measurement

Detect Numeric Outliers Assistant in MLTK



Detect Numeric Outliers

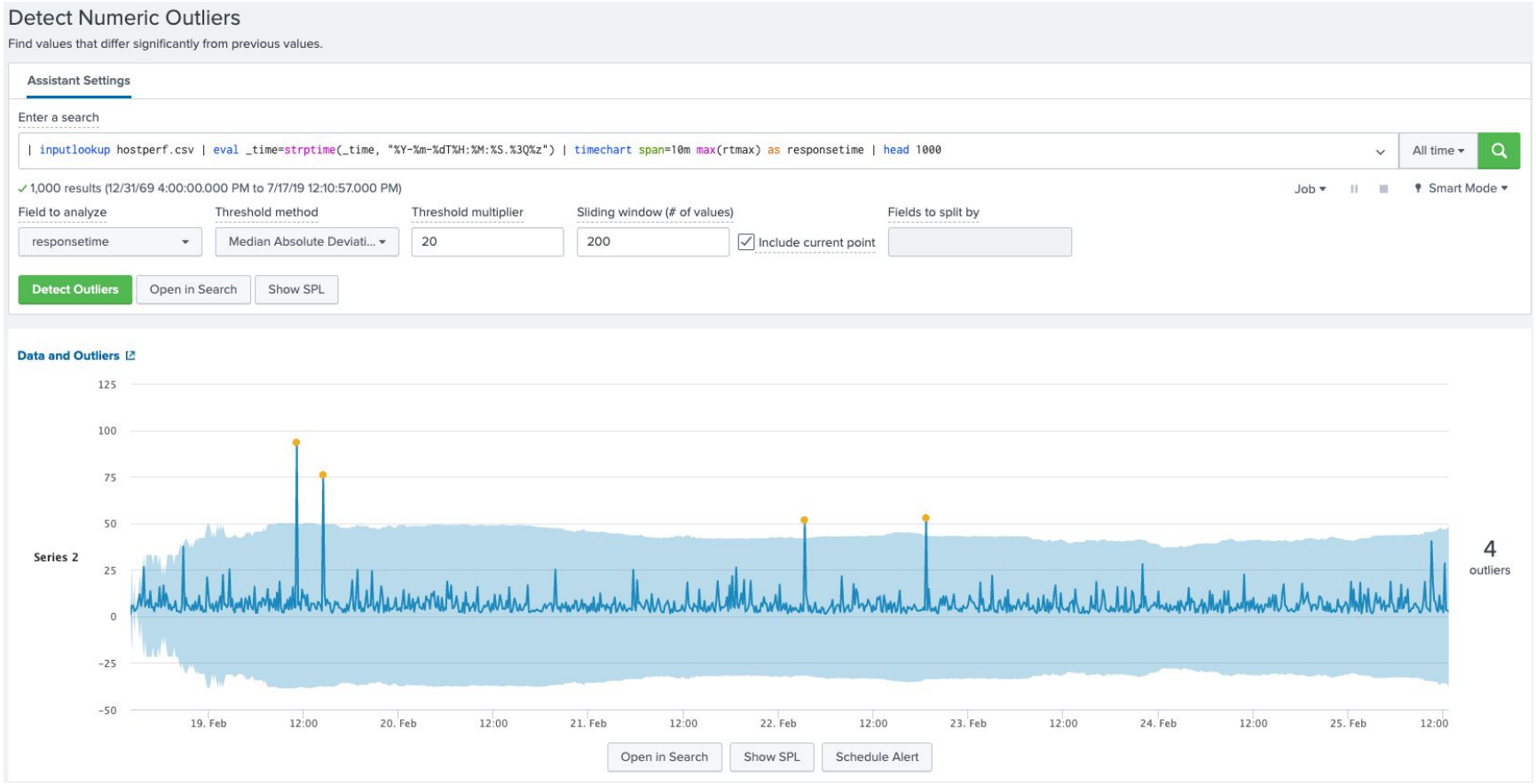
Find values that differ significantly from previous values.

Examples

- Detect Outliers in Server Response Time
- Detect Outliers in Number of Logins (vs. Predicted Value)
- Detect Outliers in Supermarket Purchases
- Detect Outliers in Power Plant Humidity
- Detect Cyclical Outliers in Call Center Data
- Detect Cyclical Outliers in Logins

We can do this today with the MLTK

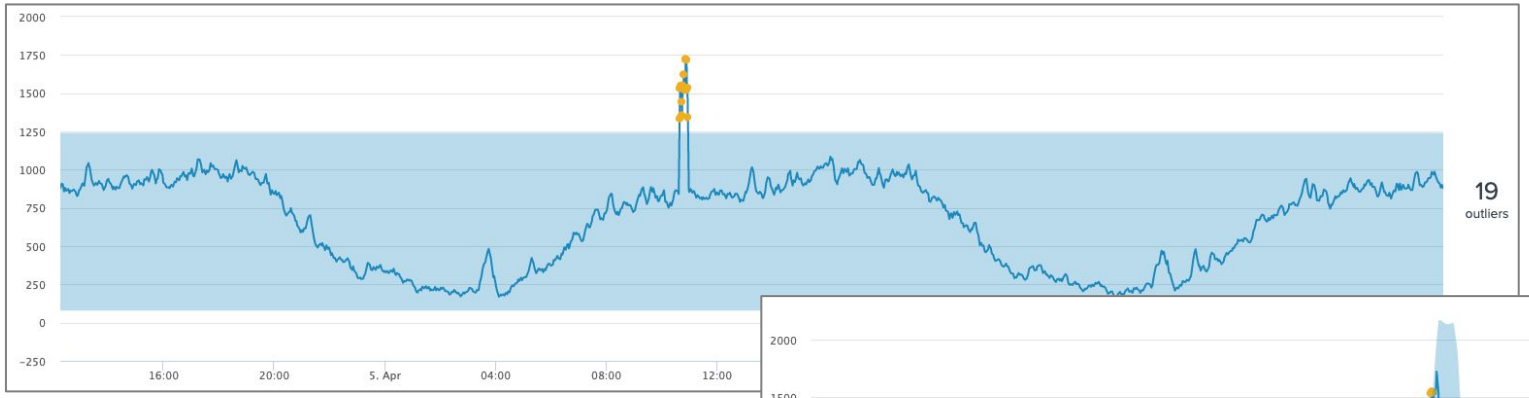
Using the Detect Numeric Outliers assistant



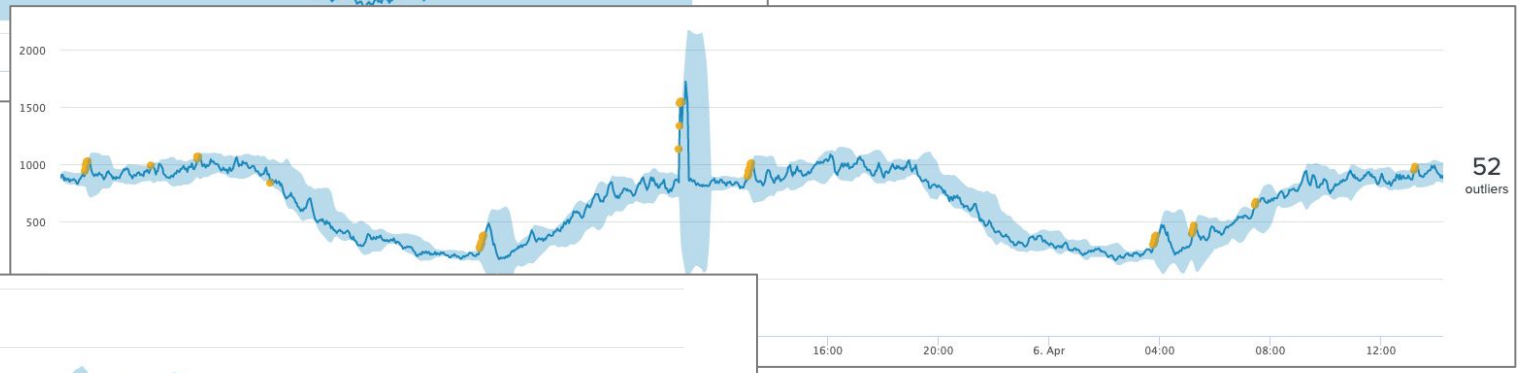
splunk> .conf19

But it can be hard to figure out how to use

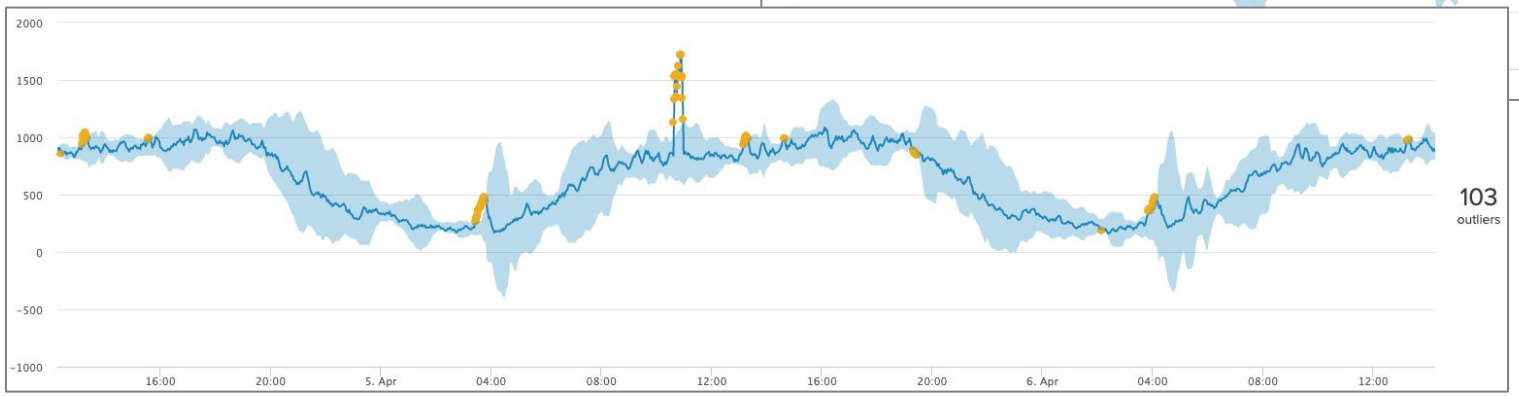
Which method works best for my data?



Using Standard Deviation with no sliding window



Using Standard Deviation with a sliding window



Using Median Absolute Deviation with a sliding window

splunk> .conf19

And there is no model created

You have to run your search on all your data every time

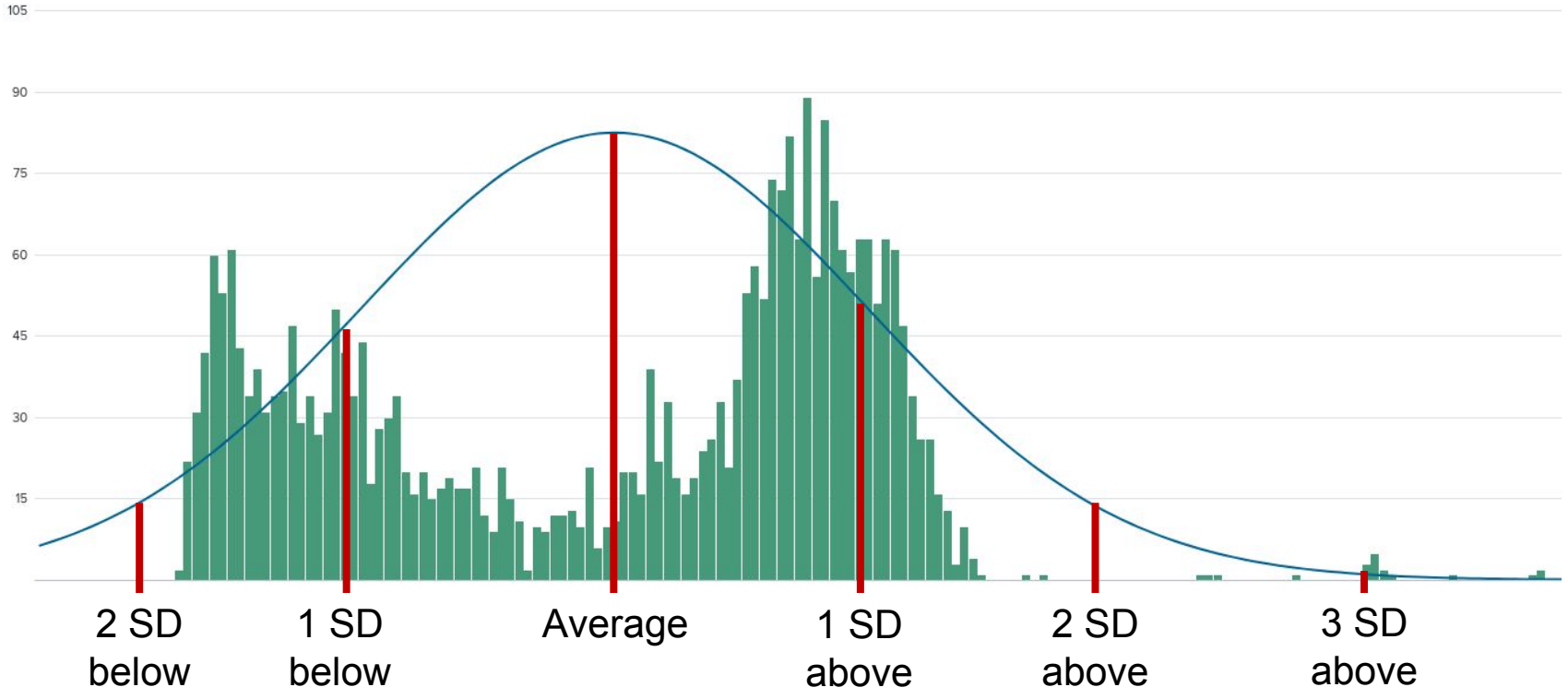
Calculate the outliers

```
| inputlookup hostperf.csv | eval _time=strptime(_time, "%Y-%m-%dT%H:%M:%S.%3Q%z") | timechart span=10m max(rtmax) as responsetime | head 1000
| streamstats window=200 current=true median("responsetime") as median
| eval absDev=(abs('responsetime'-median))
| streamstats window=200 current=true median(absDev) as medianAbsDev
| eval lowerBound=(median-medianAbsDev*exact(20)), upperBound=(median+medianAbsDev*exact(20))
| eval isOutlier=if('responsetime' < lowerBound OR 'responsetime' > upperBound, 1, 0)
```

Where's the **fit** command?

Why is it so hard?

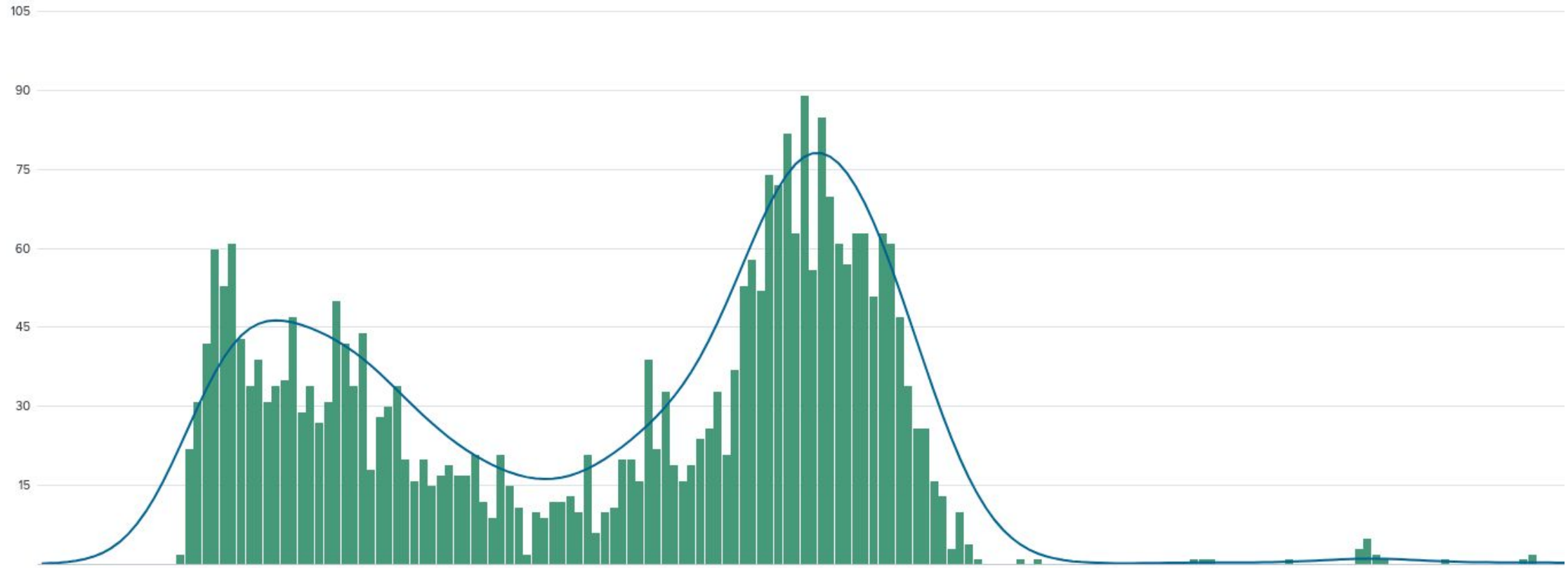
Your data may not be so “Normal”



When viewing our data as a histogram, the average may not be so ”average”

What if we could follow the shape of our data?

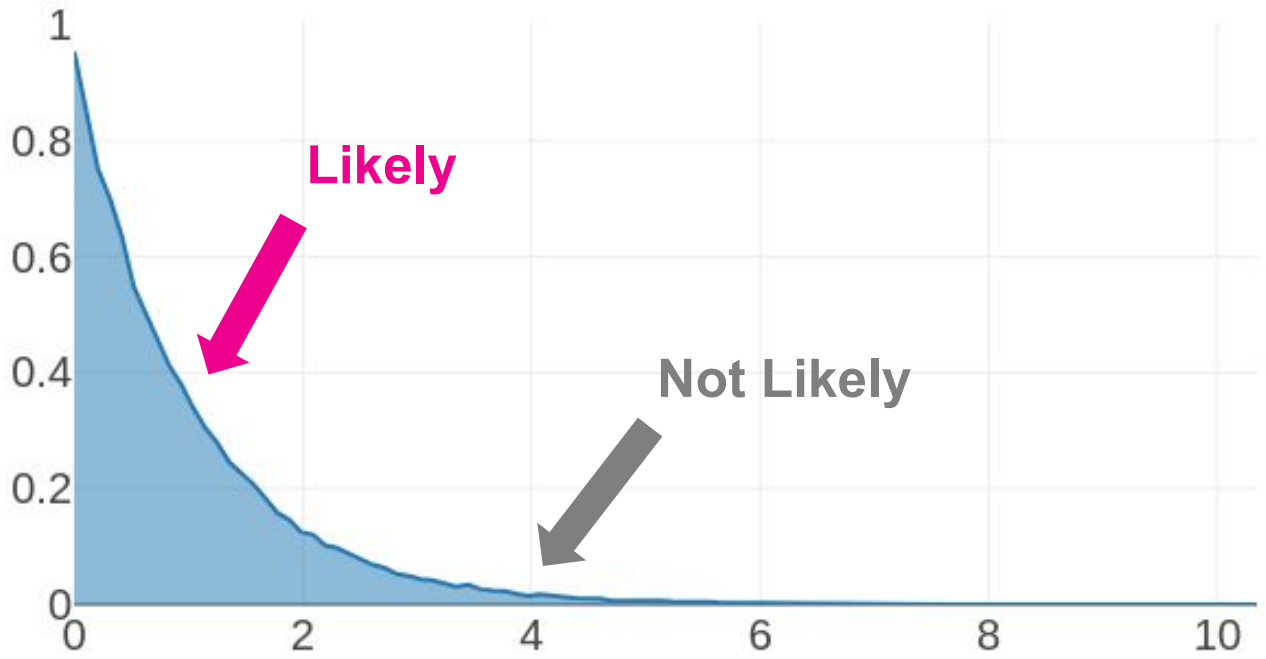
We can with the DensityFunction algorithm!



What is a Density Function Anyway?

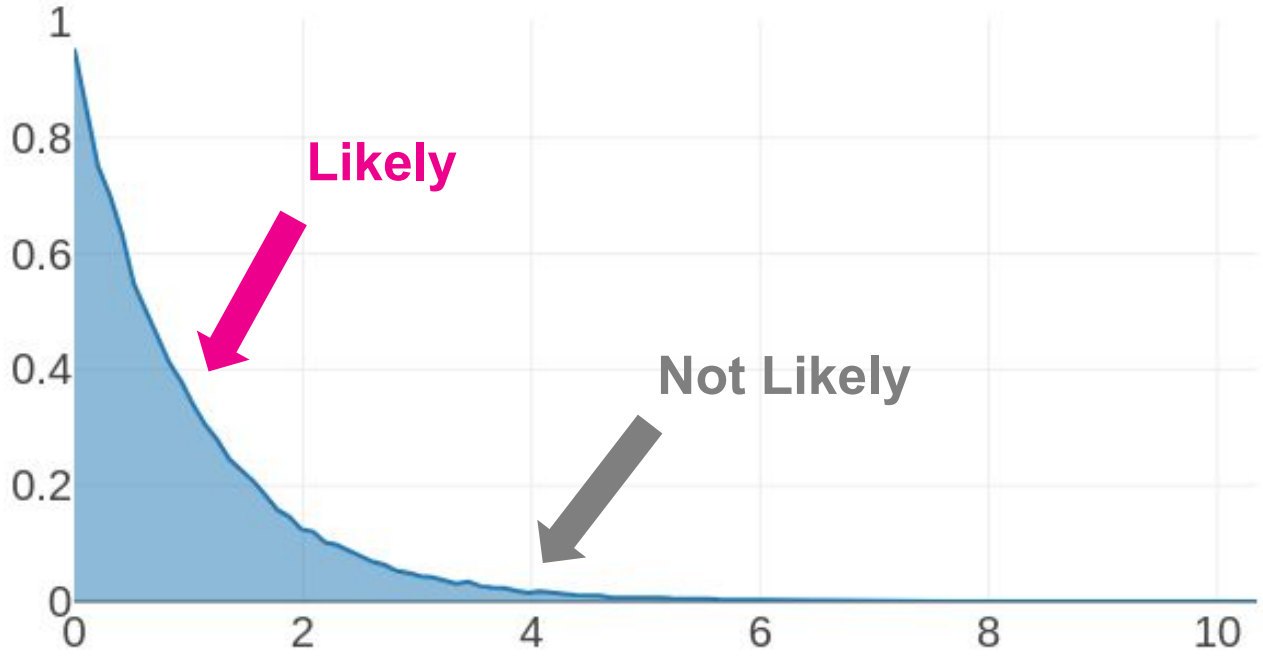
splunk> .conf19

What is a Density Function Anyway?



A mathematical function that maps outcomes to their relative likelihood

What is a Density Function Anyway?



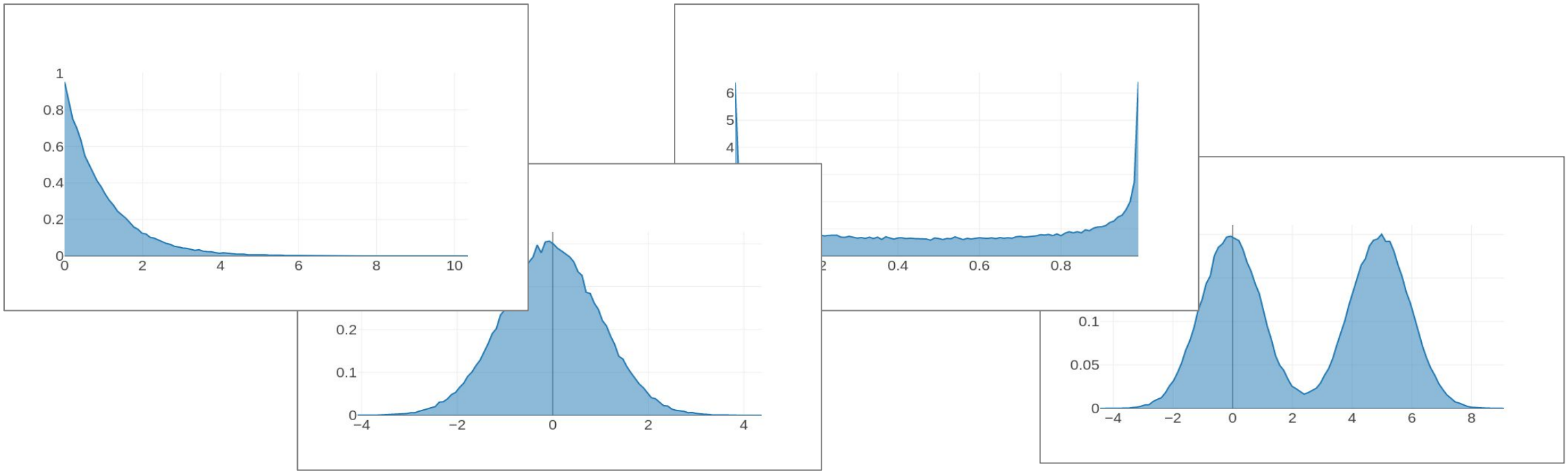
$$P(x) = \lambda e^{-\lambda x}$$

Parameter

A mathematical function that maps outcomes to their relative likelihood

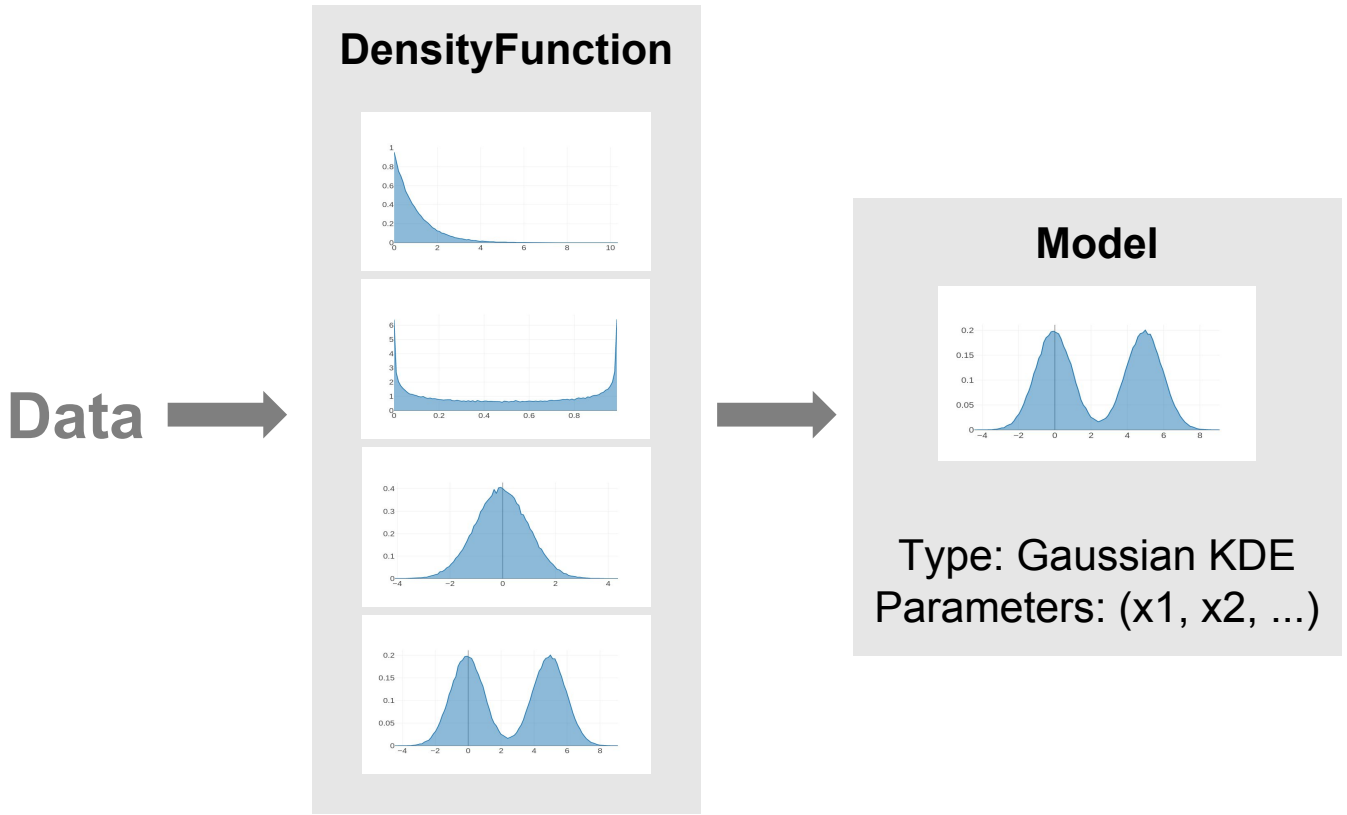
Fitting with DensityFunction

With a set of values, we'd like to know their **distribution type** and **parameters**



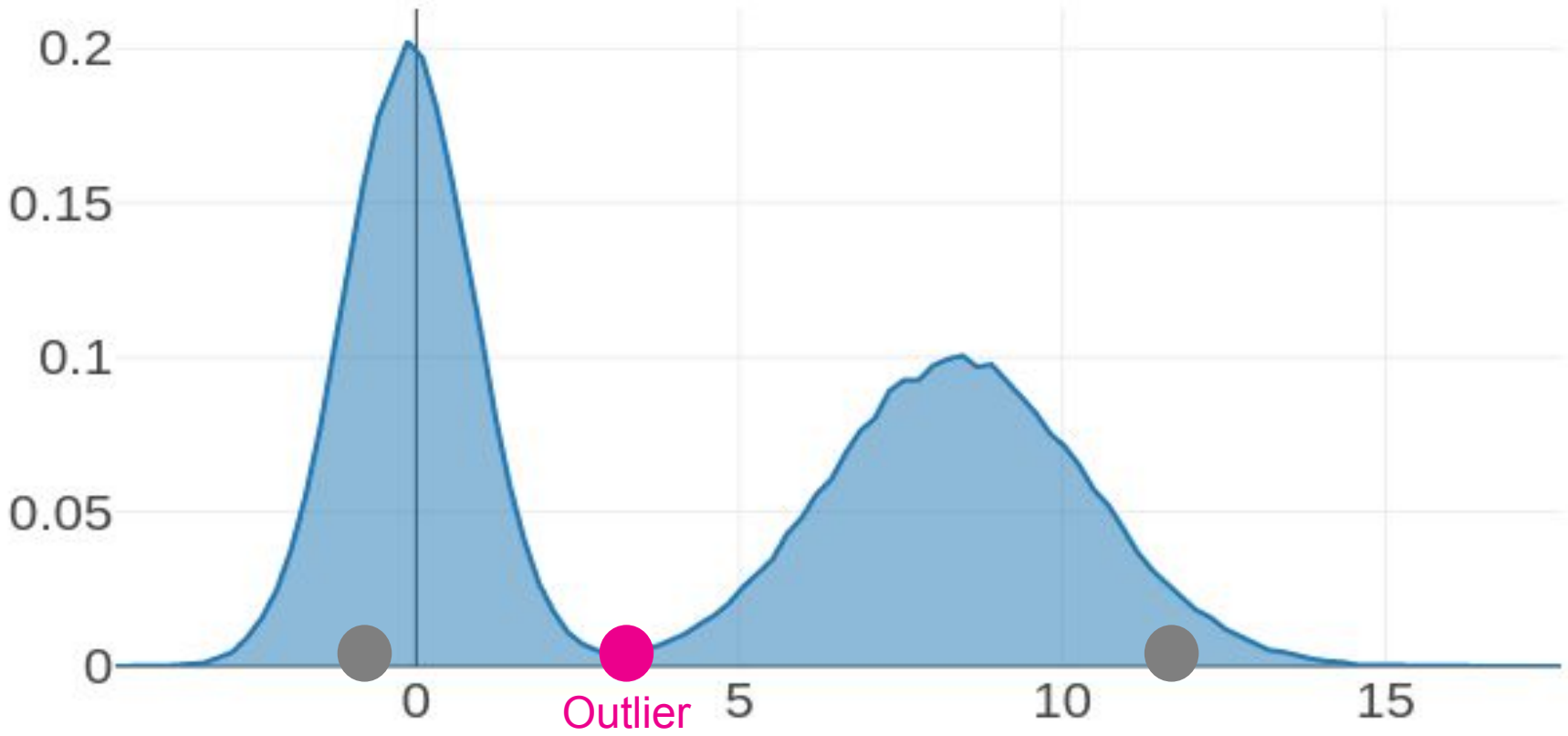
Fitting with DensityFunction

DensityFunction fits your data over a set of distributions and picks the best fit



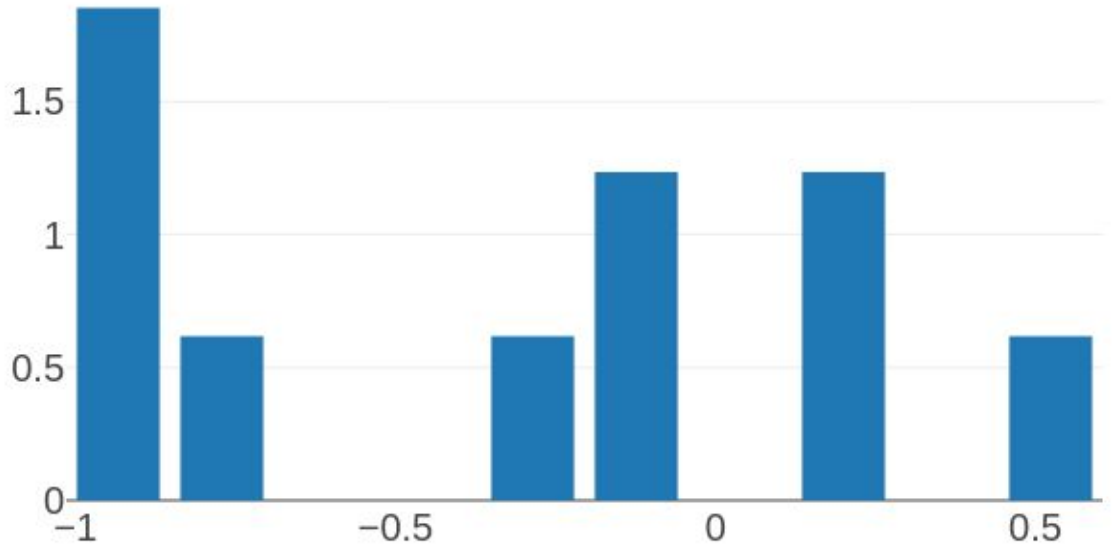
Outlier Detection with DensityFunction

When new data comes in, we use our density function to determined its likelihood



Caveats with DensityFunction

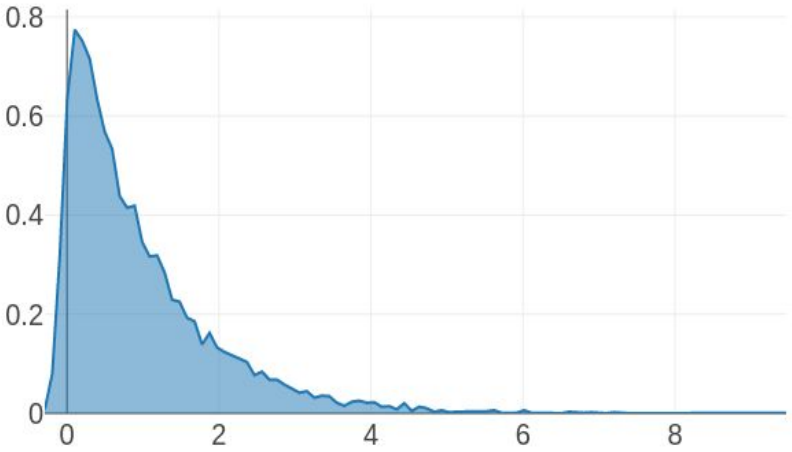
Don’t fit on noise!



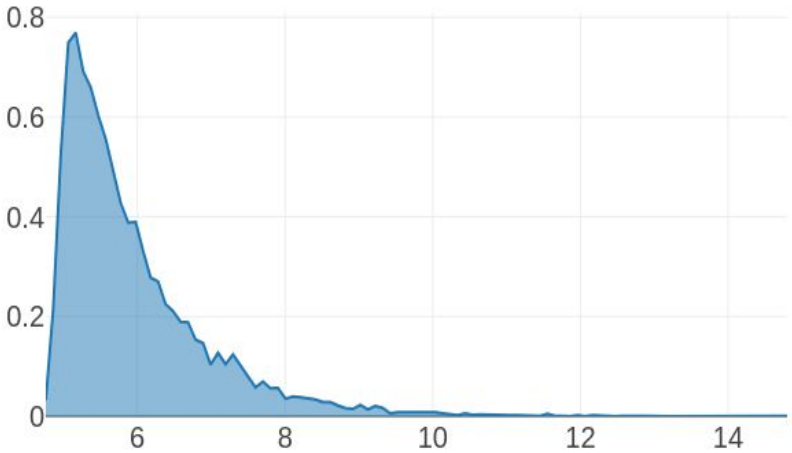
If you have only a **few data points** it’s likely you’re fitting on noise

Caveats with DensityFunction

Beware of shifting mean!



Total number of logins/month: Day 5



Total number of logins/month: Day 25

If your measure is **cumulative**, your distribution mean **shifts**

How do you use DensityFunction?

splunk> .conf19

Using DensityFunction

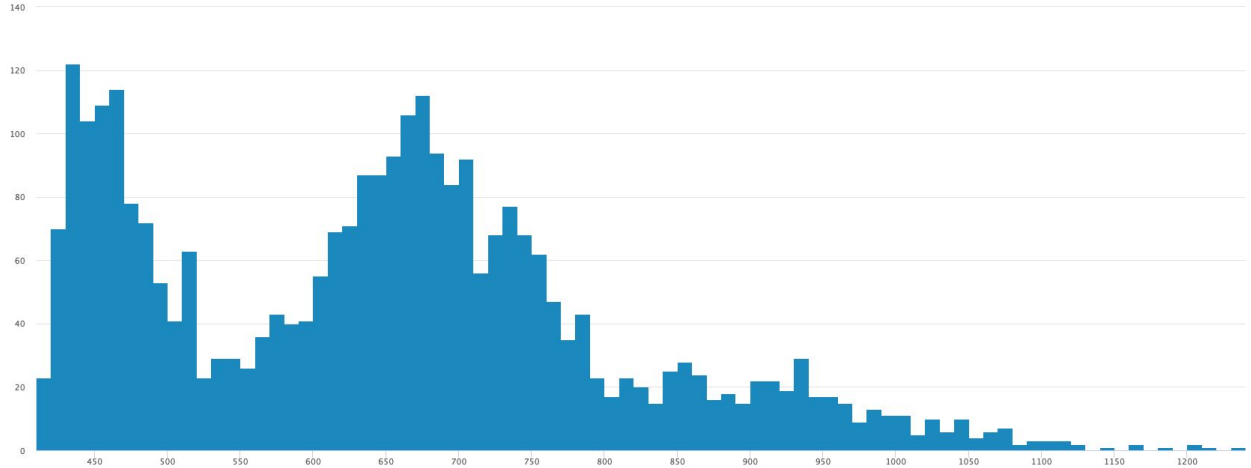
First you should understand the shape (distribution) of your data

```
index=your-index field=value
| stats count as my_field by dim1 dim2
...
| bin my_field bins=1000
| stats count by my_field
| makecontinuous my_field
| fillnull
| sort my_field
```

Or use the `histogram` macro in MLTK!

```
...
| `histogram(my_field,1000)`
```

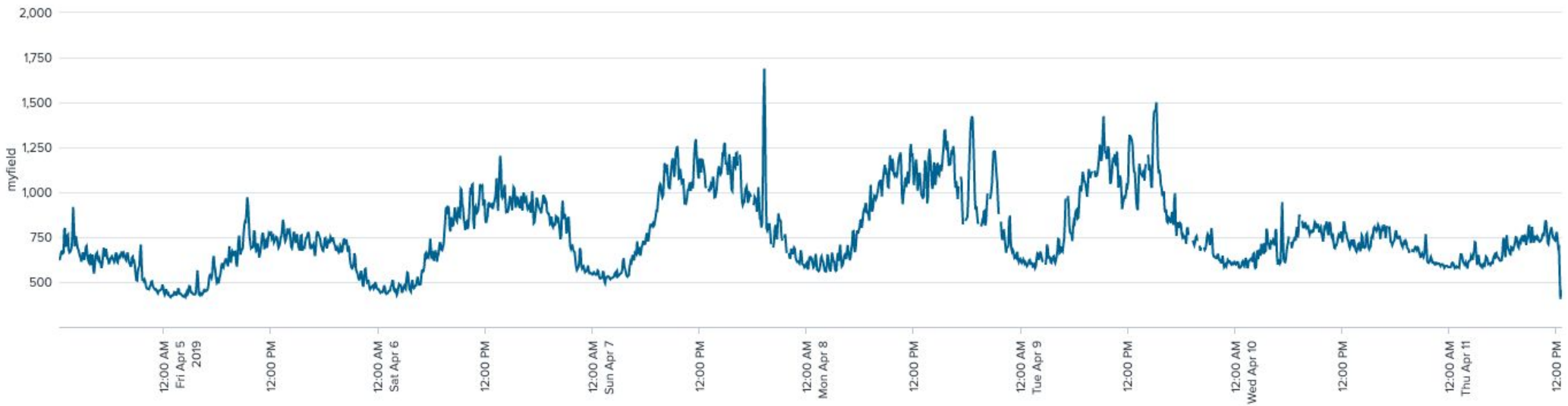
Use the Column Chart or Histogram Chart (in MLTK) Viz



Using DensityFunction

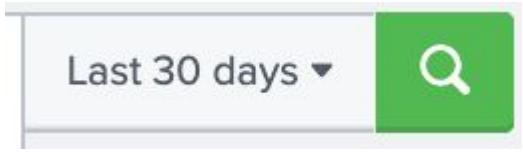
Possibly also understand the shape of your data over time

```
index=your-index other search terms
...
| timechart span=5m avg(my_field) as my_field
```



Using DensityFunction

Create a DensityFunction model



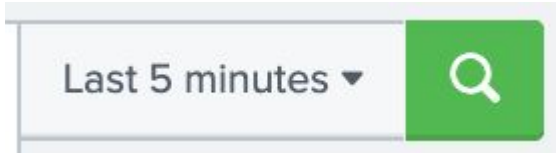

```
index=your-index other search terms
| stats count as my_field by dim1 dim2
...
| fit DensityFunction my_field by "dim1,dim2" into MyDFModel as IsOutlier
  threshold=0.01 dist=auto
```

_time	my_field	BoundaryRanges	IsOutlier	ProbabilityDensity(my_field)	date_hour
2019-04-06 00:16:00	442	[[-Infinity, 405.4724, 0.0044], [491.6502, Infinity, 0.0059]]	0.0	0.024080211386712812	0
2019-04-05 01:54:00	430	[[-Infinity, 407.6309, 0.0045], [521.7347, Infinity, 0.0057]]	0.0	0.017206205983126367	1
2019-04-05 02:24:00	426	[[-Infinity, 399.3642, 0.0046], [508.0576, Infinity, 0.0053]]	0.0	0.015769377838471113	2
2019-04-05 03:35:00	432	[[-Infinity, 394.9896, 0.0038], [592.5486, Infinity, 0.006]]	0.0	0.009509951666819004	3

Using DensityFunction

Applying your DensityFunction model

```
index=your-index other search terms
| stats count as my_field by dim1 dim2
...
| apply MyDFModel threshold=0.005
| search "IsOutlier(my_field)"=1
```



_time	my_field	BoundaryRanges	IsOutlier	ProbabilityDensity(my_field)	date_hour
2019-04-05 23:42:00	507	[[-Infinity, 426.7756, 0.005], [504.2077, Infinity, 0.005]]	1.0	0.00058596722040639	23
2019-04-06 00:41:00	489	[[-Infinity, 399.0446, 0.005], [485.6387, Infinity, 0.005]]	1.0	0.0005037745951562878	0
2019-04-06 13:40:00	1235	[[1234.5309, Infinity, 0.0102]]	1.0	0.00024044592989006216	13

You can change your threshold at apply

The BoundaryRanges designates where there are outliers

```
| apply MyDFModel threshold=0.01
```

input_field	BoundaryRanges
755	[[-Infinity, 368.9846, 0.0014], [1335.1528, Infinity, 0.0086]]

```
| apply MyDFModel threshold=0.001
```

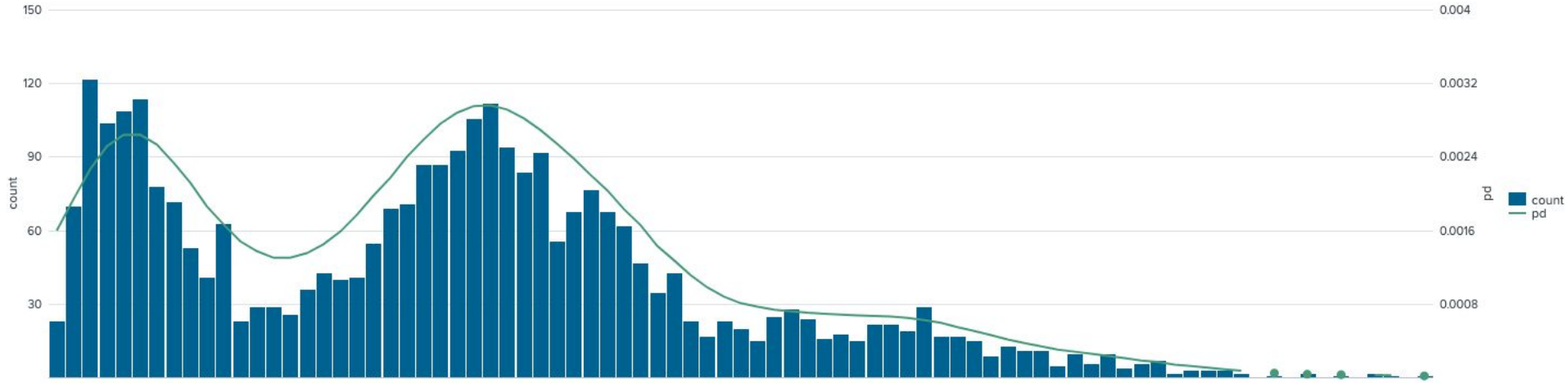
input_field	BoundaryRanges
755	[[-Infinity, 324.6197, 0.0001], [1540.5456, 1609.5577, 0.0004], [1660.4951, Infinity, 0.0004]]

```
| apply MyDFModel threshold=0.0001
```

input_field	BoundaryRanges
755	[[-Infinity, 296.6863, 0.0], [1744.2954, Infinity, 0.0001]]

Visualizing the Probability Density Estimate

```
...
| fit DensityFunction my_field show_density=true
| bin my_field bins=100
| stats count avg("ProbabilityDensity(my_field)") as pd by my_field
| makecontinuous my_field
| sort my_field
```



Visualize as a **Bar Chart**, and put the **pd** field on a **separate axis**

Get more advanced and Create an Anomaly Score

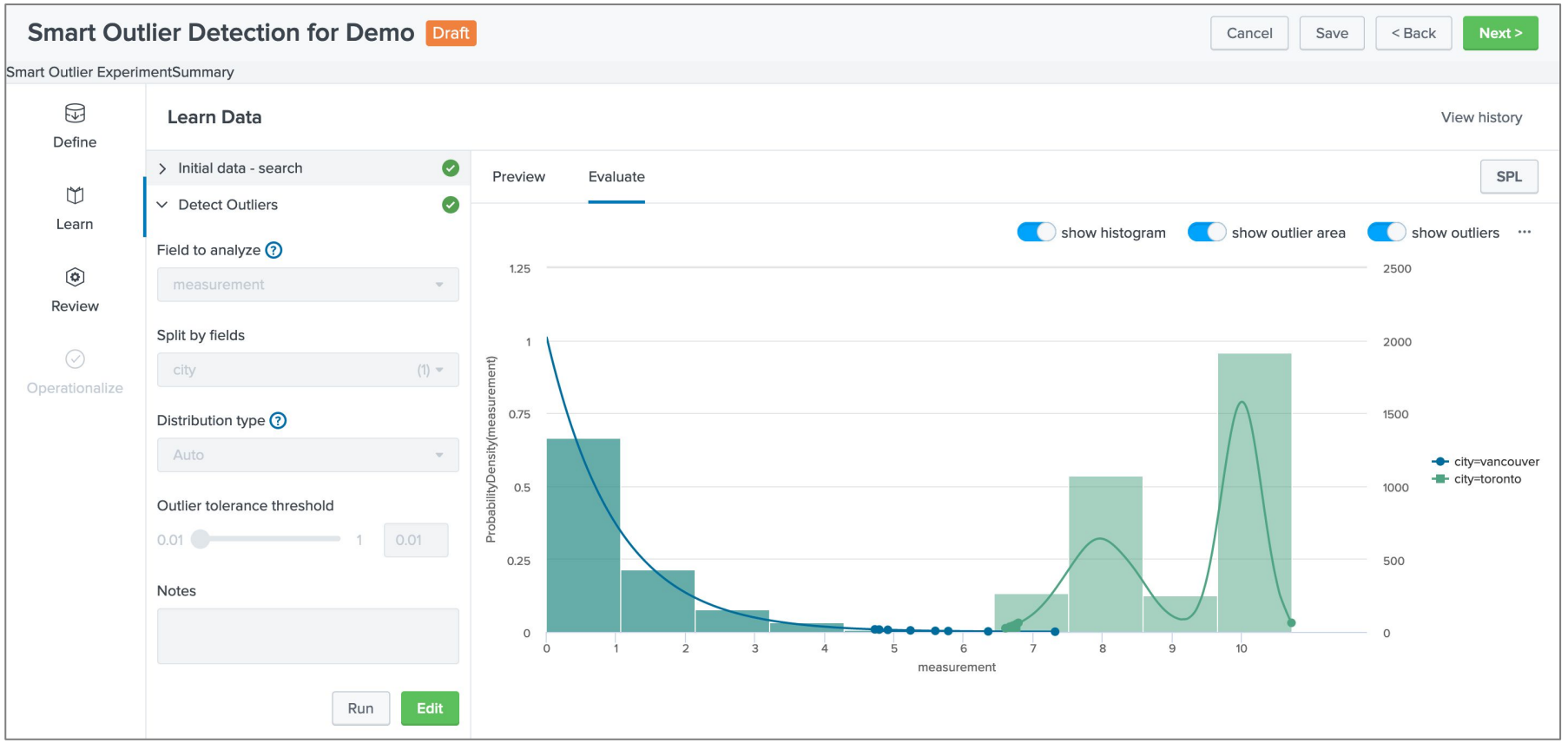
Apply different pivots of your data with different models

```
...
| fit DensityFunction my_field as IsOutlierOverall
| fit DensityFunction my_field by "dim1" as IsOutlierByDim1
| fit DensityFunction my_field by "dim2" as IsOutlierByDim2
| eval AnomalyScore=0
| foreach IsOutlier* [eval AnomalyScore=AnomalyScore+<<FIELD>>]
```

_time	my_field	AnomalyScore	IsOutlierByDim1	IsOutlierByDim2	IsOutlierOverall	dim1	dim2
2019-04-09 14:59:00	1487	3.0	1.0	1.0	1.0	14	2
2019-04-11 12:29:00	326	3.0	1.0	1.0	1.0	12	4
2019-04-08 18:25:00	1399	2.0	0.0	1.0	1.0	18	1
2019-04-08 18:30:00	1404	2.0	0.0	1.0	1.0	18	1
2019-04-08 20:56:00	1236	1.0	1.0	0.0	0.0	20	1
2019-04-09 15:03:00	1439	1.0	0.0	0.0	1.0	15	2
2019-04-05 11:00:00	752	0.0	0.0	0.0	0.0	11	5
2019-04-05 22:38:00	570	0.0	0.0	0.0	0.0	22	5

Using the Smart Outlier Detection Assistant

Putting it all together with an “easier” button



splunk> .conf19

Solution #2: Forecasting using StateSpaceForecast

splunk> .conf19

Let’s clarify some nomenclature

Forecast ≠ Prediction

Forecast vs Prediction

What is the difference?

Forecast

- Given the past values of a metric, tell me what the value will looks like X time periods from now (e.g. tomorrow, next week, etc).
- Forecasting relies on time and the historical values of a measurement in question as its inputs.

Prediction

- Given the past values of a set of fields, estimate (or predict) what the value of one of those fields will be, given the other fields as inputs.
- Prediction relies on many other inputs to try and explain the relationship between those inputs and the measurement you are trying to predict.

But both of these fall under the category of “Predictive Analytics”

Forecast vs Prediction

What is the difference?

Forecast

- Given the past values of a metric, tell me what the value will looks like X time periods from now (e.g. tomorrow, next week, etc).
- Forecasting relies on time and the historical values of a measurement in question as its inputs. **We are covering.**

Prediction

- Given the past values of a set of fields, estimate (or predict) what the value of one of those fields will be, given the other fields as inputs.
- Prediction relies on many other inputs to try and explain the relationship between those inputs and the measurement you are trying to predict.

But both of these fall under the category of “Predictive Analytics”

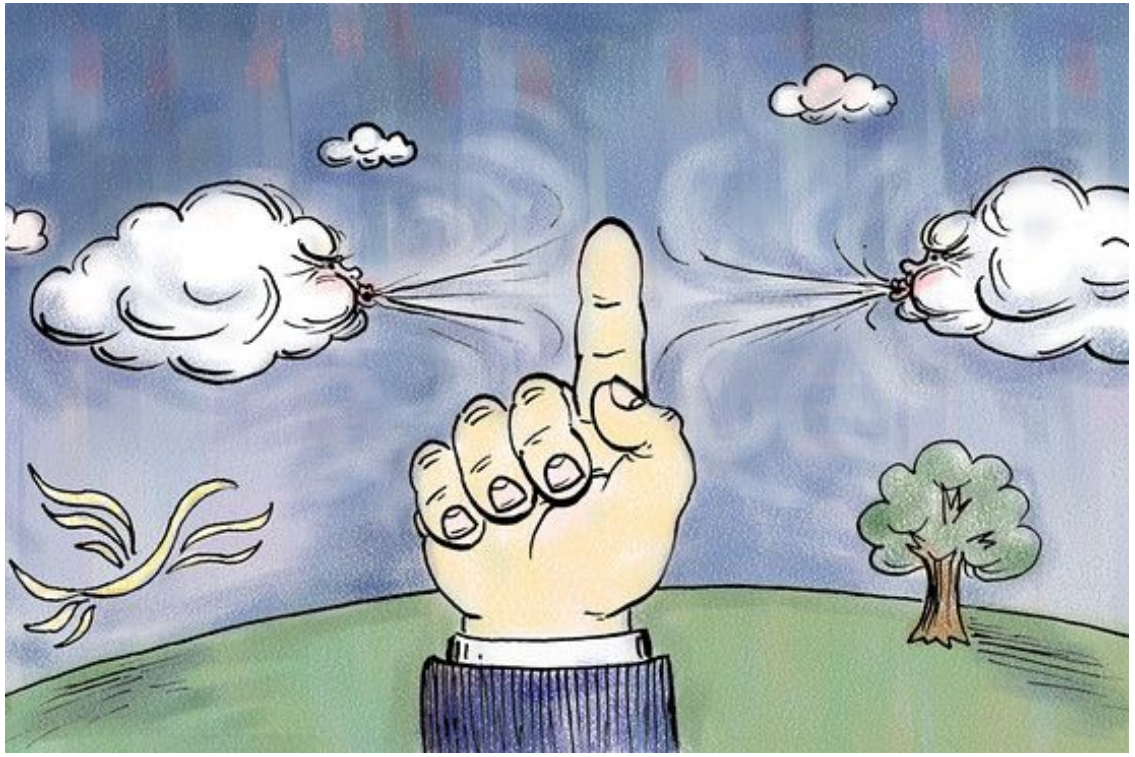
Why would I use forecasting?

Typically used for planning

- Based on past trends, what do we expect next week/month/quarter/year to look like?
- Capacity planning (hard drive, operating temperature)

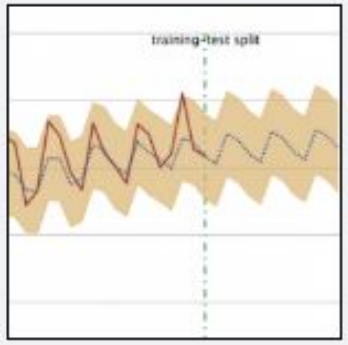
Forecasting is not a crystal ball, but it gives you a quantitative estimate on future values

- Getting a picture of what the future might look like



The old way of forecasting in MLTK

| predict my_field algorithm=LLP5 holdback=112 future_timespan=224

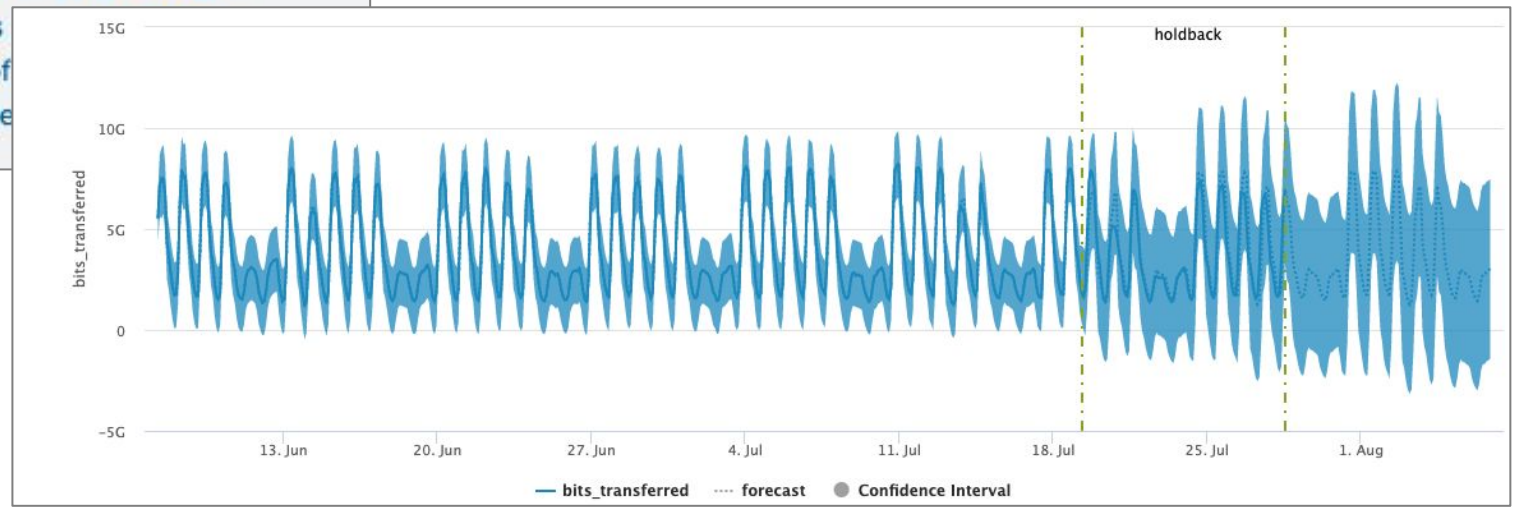


Forecast Time Series

Forecast future values given past values of a metric (numeric time series).

Examples

- Forecast Internet Traffic
- Forecast the Number of Employee Logins
- Forecast Monthly Sales
- Forecast the Number of
- Forecast Exchange Rate



Using the old way for forecasting

There's nothing wrong with the old way, it's just often improperly used

You have to be an expert at the math

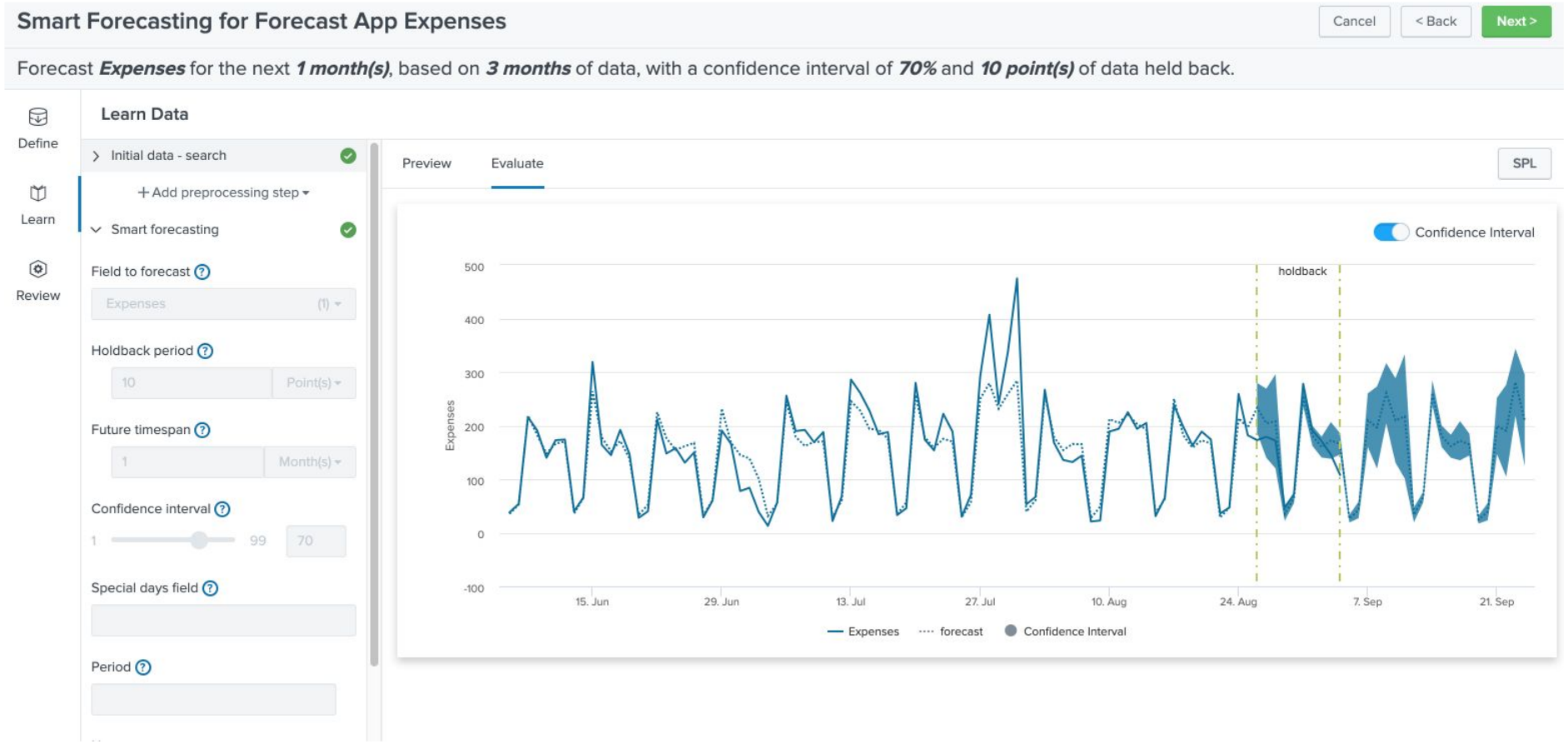
- You have to specify the algorithm to use for the predict command
- You have to know how to optimize on P, D, and Q parameters for ARIMA

There is no model file created, which means you can't "apply" your model to future data

Doesn't consider special days (holidays)

The new way of forecasting in MLTK

| fit StateSpaceForecast my_field holdback=112 forecast_k=224



Using StateSpaceForecast

Applying more real-time operational use cases

- Uses basically the same math (Kalman filter) as the predict command, but it will try to figure out the parameters and mode (algorithm in predict)
- You can “apply” your model to future data
- You can account for special days
- You can use incremental fit (continuously update your model with new data)
- You can do multivariate analysis
- It will automatically impute the missing values (null values)

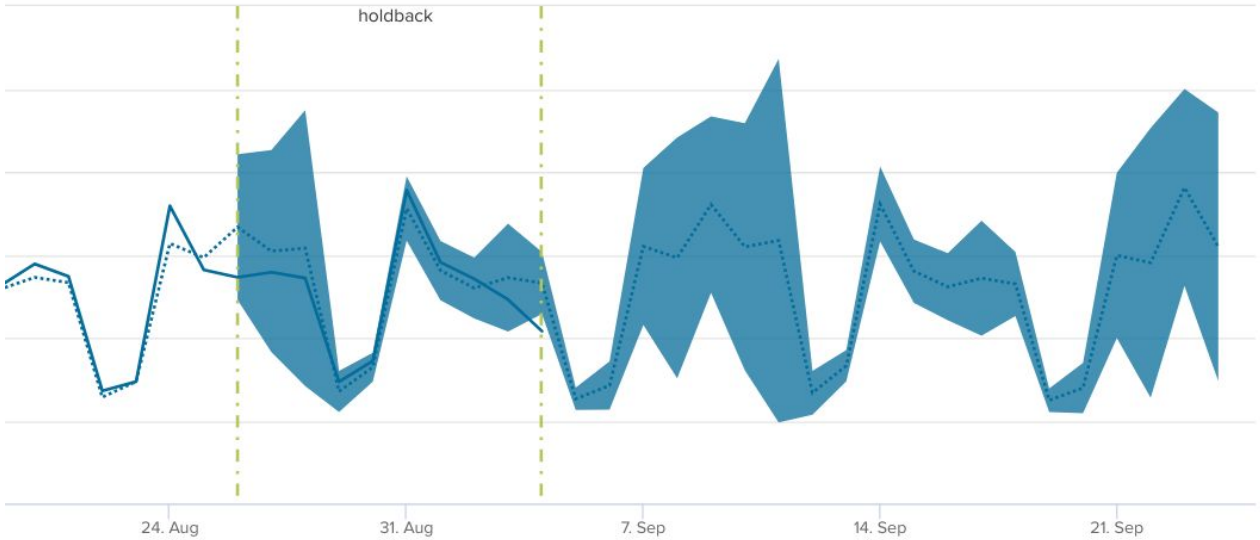
StateSpaceForecast Caveats and Considerations

splunk> .conf19

Confidence Level and Confidence Interval

What's the difference?

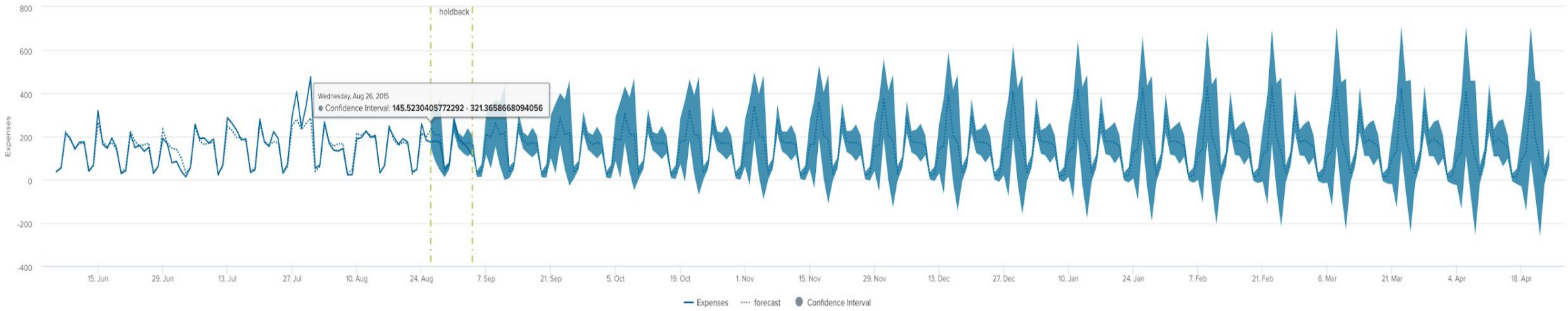
- Confidence level is how confident we are about the prediction that our confidence interval includes the real value
- Confidence interval and confidence level need to be interpreted together
- 95% confidence level means we are 95% confident that the confidence interval includes the true value



Confidence Level and Confidence Interval

Interpreting the data further into the future

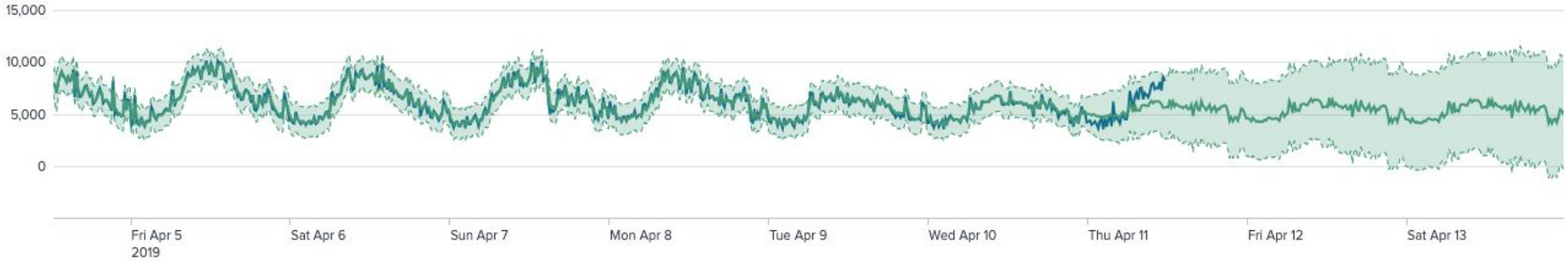
The confidence interval increases over time because the algorithm needs more “leeway” to fulfill its promise of 95% confidence level



Confidence interval is not about if the prediction is an outlier or not. It’s about accuracy of prediction.

Caveats with StateSpaceForecast

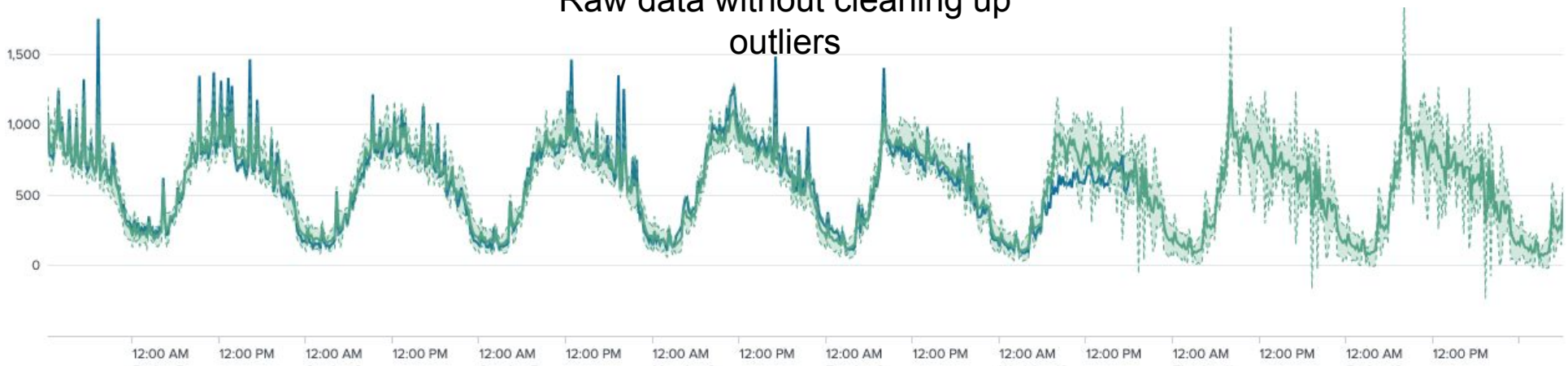
- Don’t project too far into the future
- Choose a large confidence level (e.g., 95%)
- If the confidence interval is too wide be careful about the reliability of the forecast



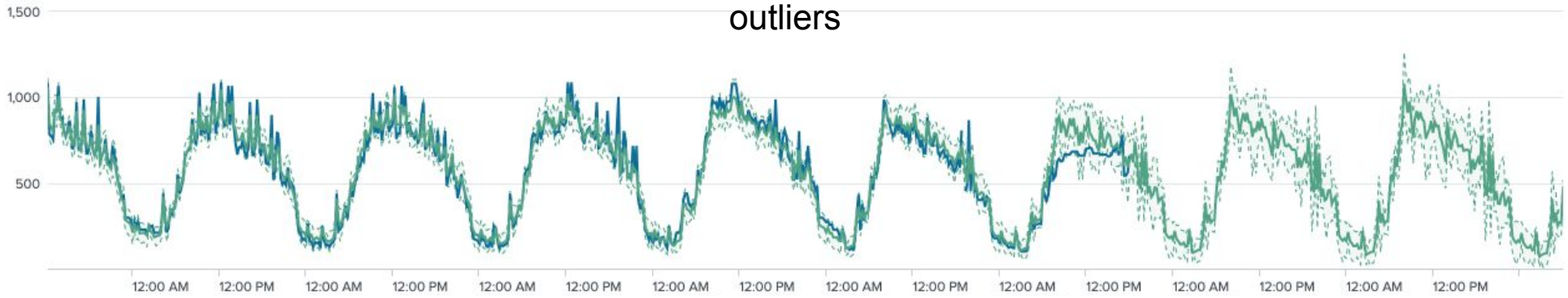
Forecasting is Sensitive to Outliers

Make sure you do some data cleansing first

Raw data without cleaning up outliers



After cleaning up some of the outliers



Key Takeaways

This is where the subtitle goes

1. Use DensityFunction for finding outliers
   - Visually inspect fit and tune threshold
   - Don’t fit over noise
2. Use StateSpaceForecast for projection and planning
   - Remove outliers before fitting
   - Pay attention to confidence interval

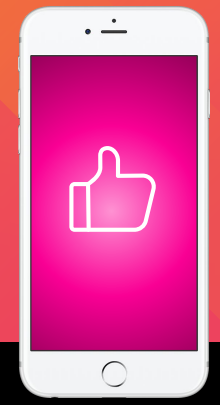
.conf19

splunk>

Thank You !

Go to the .conf19 mobile app to

RATE THIS SESSION



Q&A

Eurus Kim | Staff ML Architect

Amir Malekpour | Principal Software Engineer

splunk> .conf19