OPEN DATA SCIENCE CONFERENCE

Boston | May 1 - 4 2018

Anomaly Detection at Scale:

Data Science Difficulties

Data Engineering Challenges

Dusan Randjelovic, Senior Data Scientist **@SmartCat**

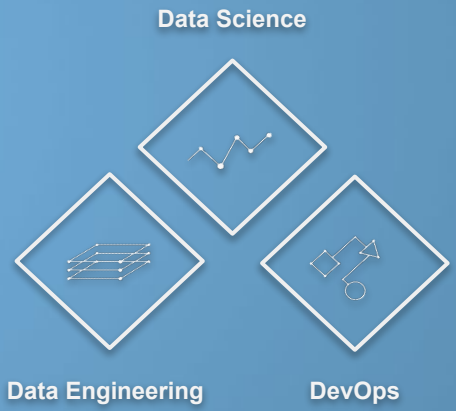
Anomaly Detection

Agenda (stripped down version)

- Why?
- What?
- How?
- How?
- How?

Agenda explained

- **Why** do we do AD (anomaly detection)?
- **What** are we looking at?
- **How:** Data science methods
- **How:** Data engineering setup
- **How:** DevOps automation



Some notes

- This is a hands-off overview with sporadic math or code teasers, focus on data science methods and data science management
- Takeaway conclusions and open questions
- Ask your question or share use-cases
- Track progress on the right

Why do we do anomaly detection?

- Problems AD is solving
- Use-cases / Applications
- Takeaway

Problems that need AD - monitoring

Complex systems produce big data

Volume (ex.: 2TB daily of user interactions for a mobile game)
Velocity (real time KPIs)
Variety (structured, unstructured, external, incidental)

COULD SCALE

“Measure anything, measure everything”, Ian Malpass

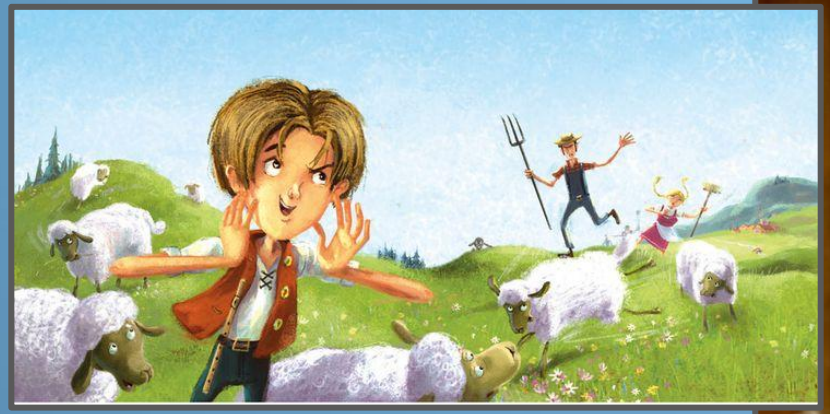
CANNOT

“Watch everything”, Nathaniel Cook

Problems that need AD - alerting

With many metrics come many false positives

Cry-wolf ⟷ Alert fatigue



Problems that need AD - insights

Not even core KPIs are independent or easy to interpret

Problems ⟷ (types of) solutions mapping

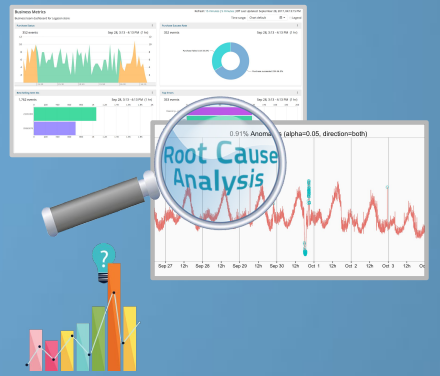
Monitoring → AD acts as ... → **Descriptive**

Alerting → AD acts as ... → **Diagnostic**
Predictive

Insights → AD acts as ... → **Prescriptive**

ANALYTICS SYSTEM

results →



AD use-cases - fraud detection

- Detect unusual behaviour in user events, like bank transactions, login attempts, etc.
- Malicious intent
- Detection should rather be prevention
- Similar: Intrusion detection, DDOS attacks

AD use-cases - predictive maintenance

- Models IoT (or other) devices lifetime
- Saves cost compared to preventive maint.
- (Pattern of) Anomalies are taken as signals

AD use-cases - energy consumption optimization

- Modelling of (temporal) patterns helps reduce waste
- AD helps to model outages and load shedding needs

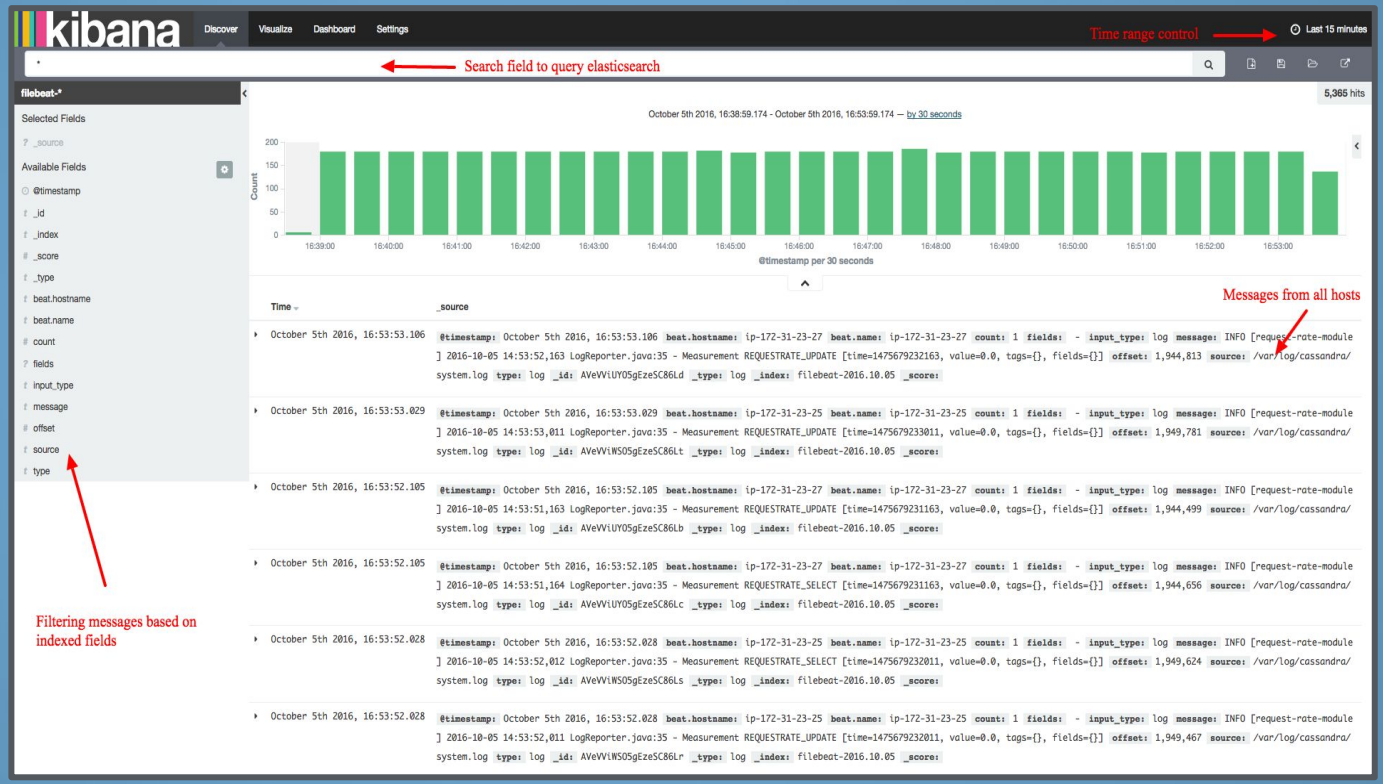
Example use-case - IT systems smart monitoring

Support distributed systems health, security, elasticity and scalability

From practice: 12 node Cassandra cluster
SLA - latency of queries < 100ms in 99.999% req.

Metrics stack: Riemann + InfluxDB + Grafana

Logs stack: ELK (Elasticsearch, Logstash, Kibana)



Cause of query latency (left) is hidden. Can be disk latency (middle), network or background process in DB

SLA example takeaway

- Not trivial to reason about causes
- Need automatic way to take actions
- Need smart alerts - predictive analytics
- Need to be robust to “normal” changes

Takeaway

Whatever the business case - start gathering the data about the system as early as you can, but

`at the same time`

use that data in the best way to `monitor` **and** `optimize`

What data are we looking at?

- Metrics + logs
- Time series
- Types of anomalies
- Takeaway

IT systems: Metrics vs. Logs

- **Metrics** are real-valued real-time data streams about system's components' performance
- **Logs** are textual, often irregular, informations about different activities and events within the system

Metrics + Logs

Metrics data tells us about a system behaviour and logs usually provide more contextual info

Context revealed from logs:

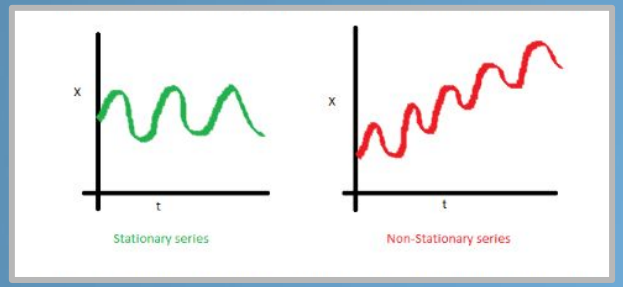
- Specific queries/users/components that influence metrics increase/decrease
- Components responses and tracebacks
- Order or out-of-order of events / dependencies

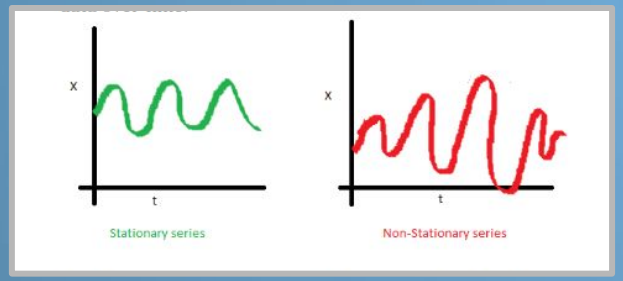
But dominate context around metrics + logs is **TEMPORAL**

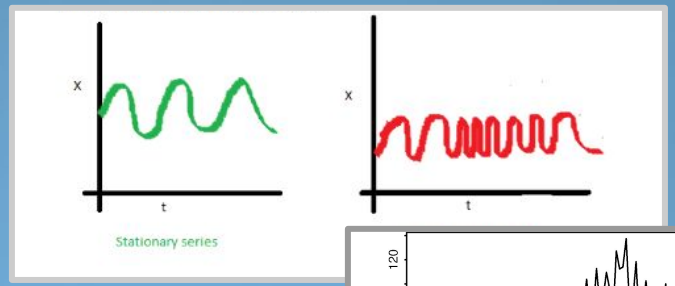
Time series

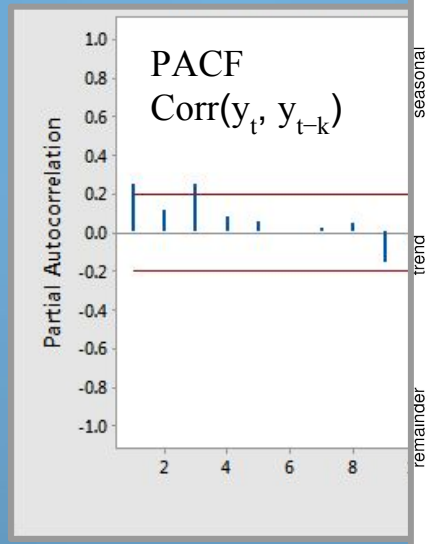
- Sequence of measurements made over time
- Data points are generally not *i.i.d.*
  - Non-stationarity (mean or variance)
  - Autocorrelation
  - Hypothesis testing and modelling (for example ADF test, Box-Jenkins)

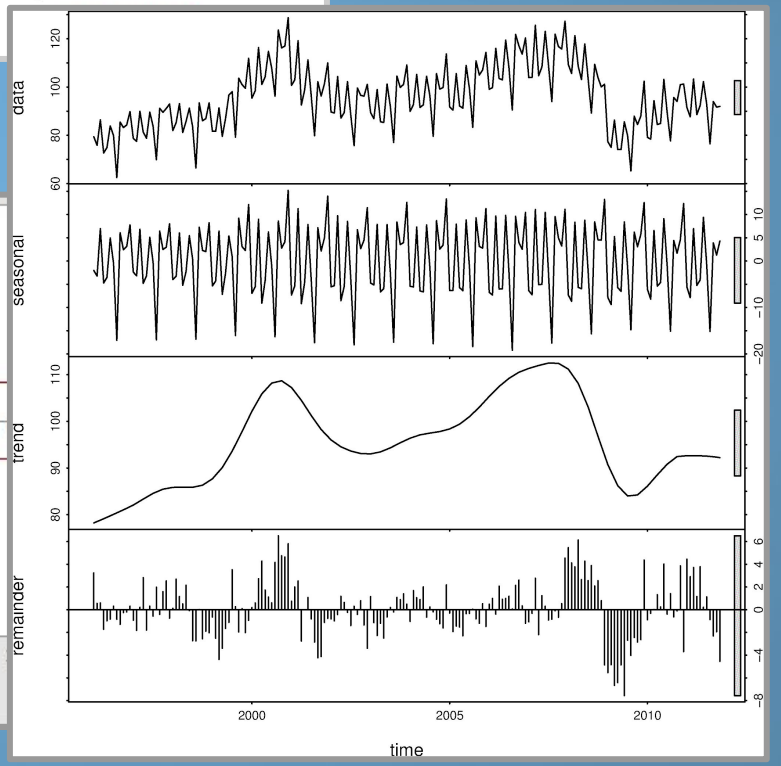
Time series


x
t
Stationary series
x
t
Non-Stationary series


x
t
Stationary series
x
t
Non-Stationary series

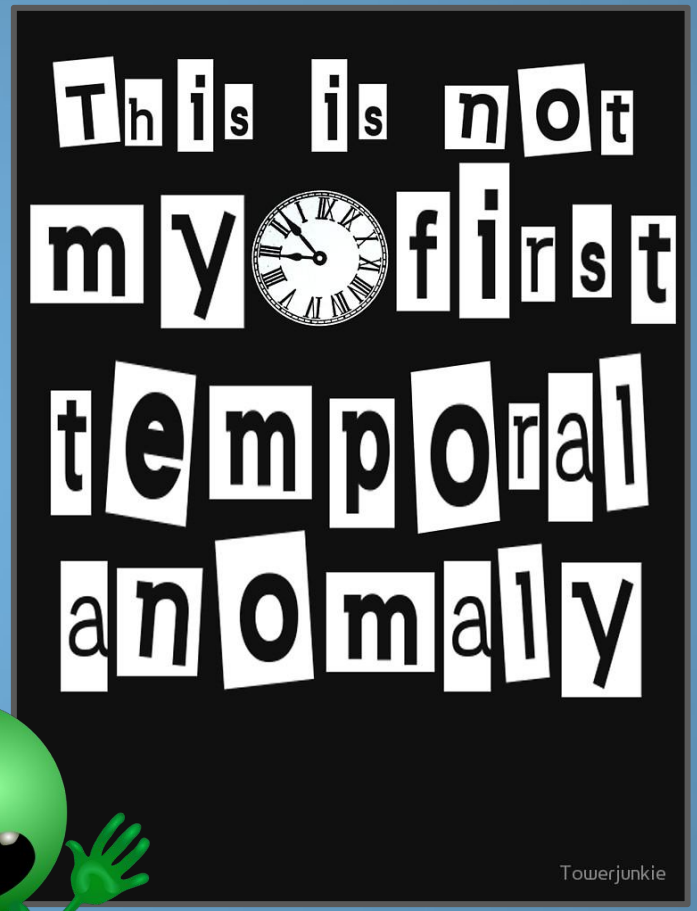

x
t
Stationary series
x
t


PACF
Corr(y_t, y_t−k)
Partial Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
-0.8
-1.0
2
4
6
8


data
seasonal
trend
remainder
2000
2005
2010
time

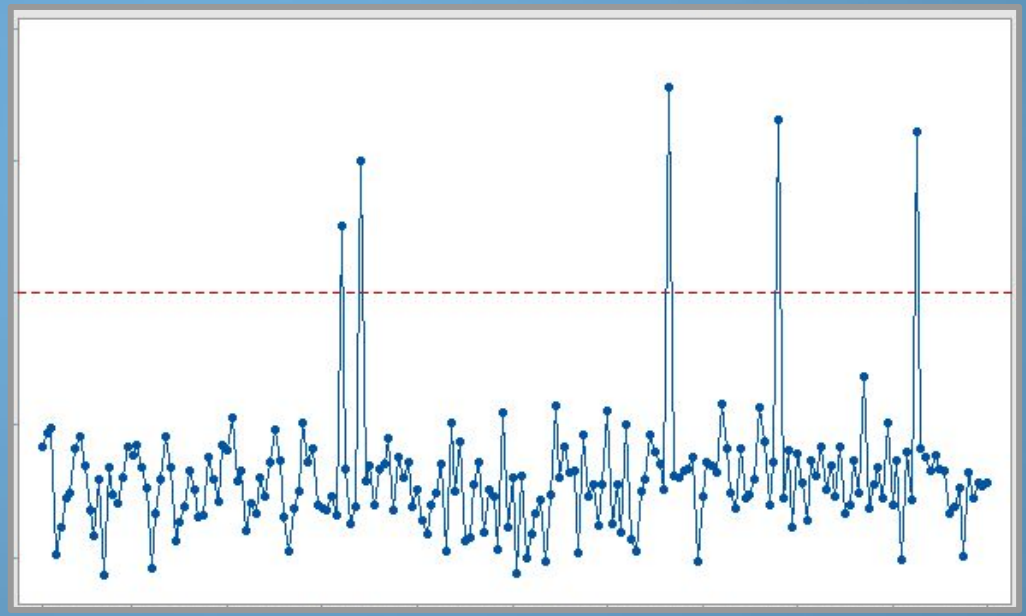
Time series anomalies

DON'T PANIC



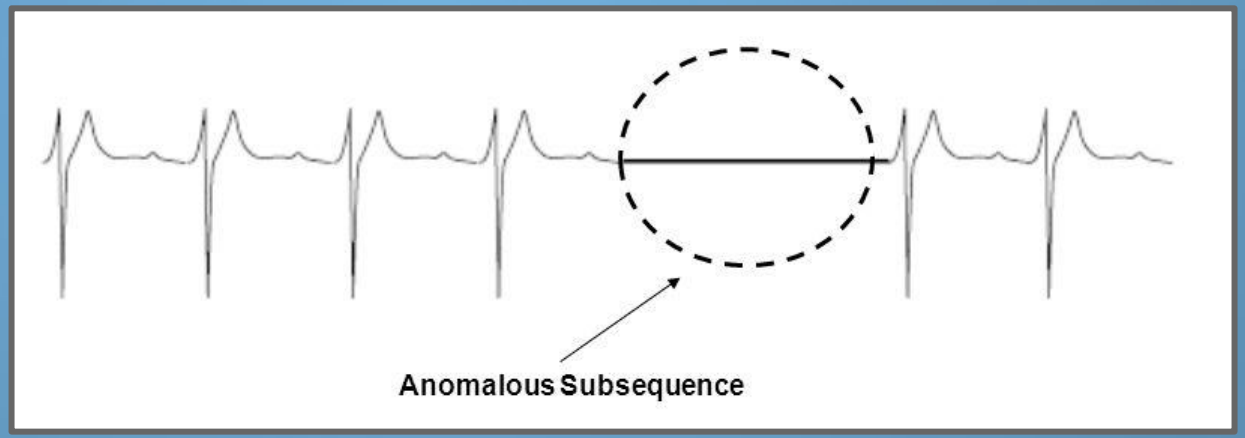
Types of (time series) anomalies

- Outliers = point anomalies



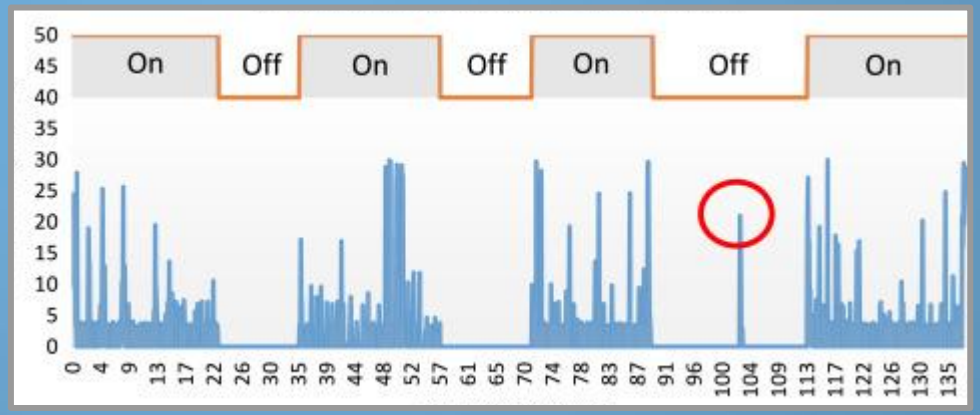
Types of (time series) anomalies

- Not every anomaly is outlier - collective anomalies



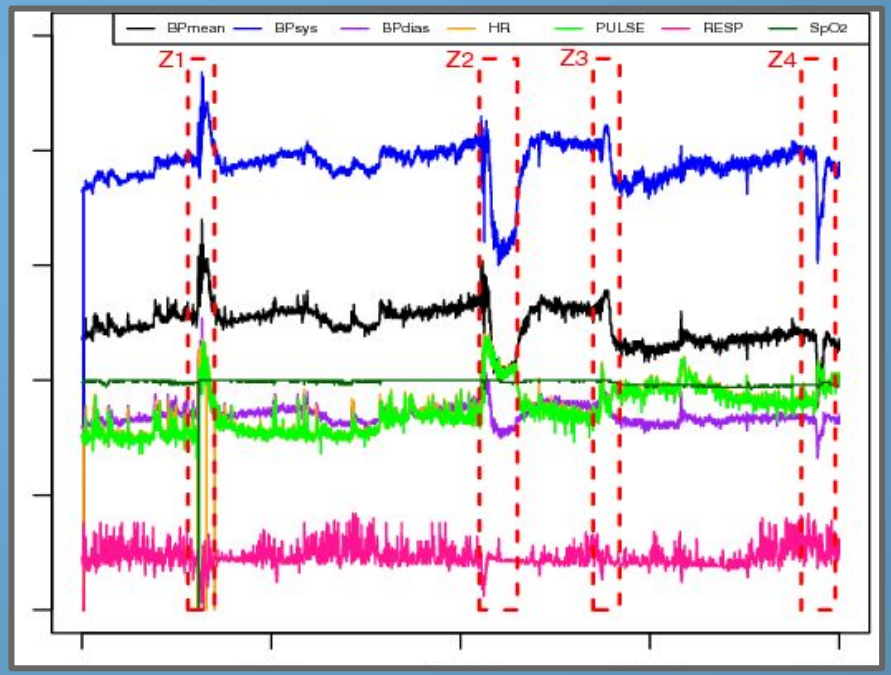
Types of (time series) anomalies

- **Not every anomaly is outlier - contextual anomalies**



Types of (time series) anomalies

- Univariate vs. multivariate

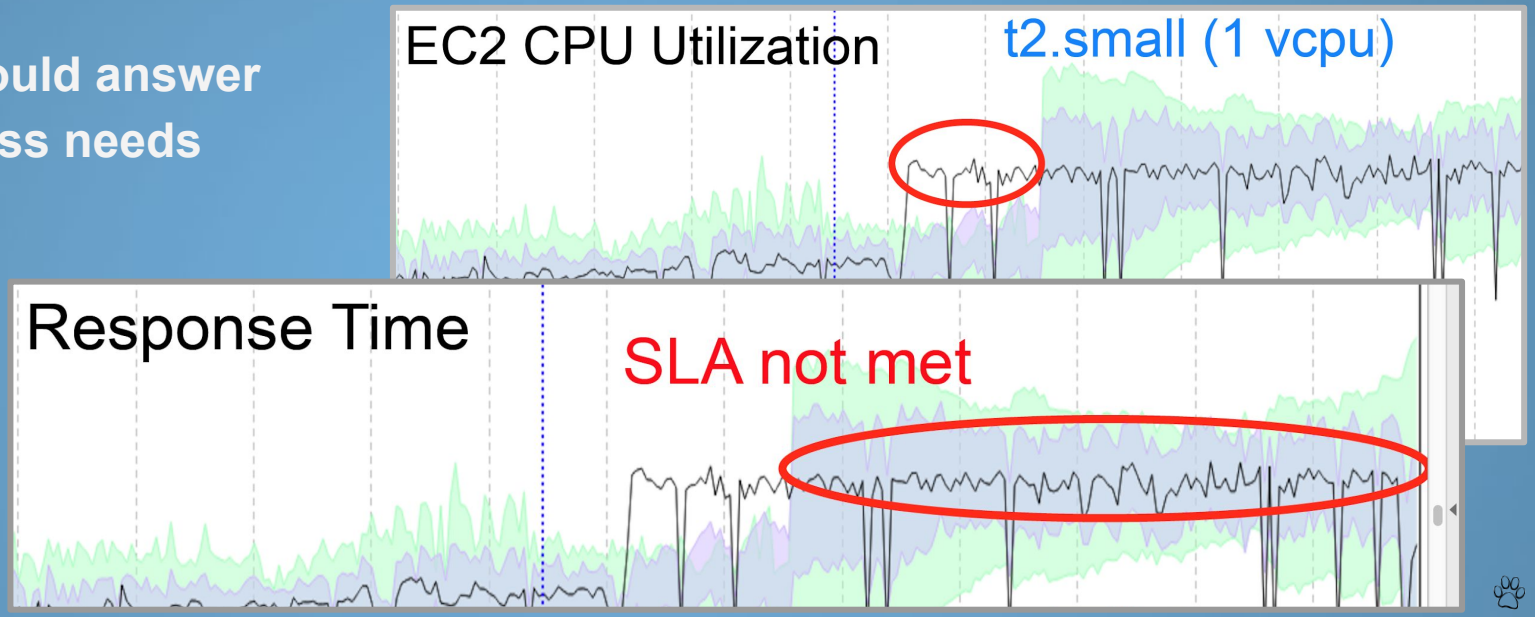


Anomalies are rare?

- Example: minute metrics with 3 sigma around mean threshold gives daily ~4 just-by-chance false positives
- For 12 node distributed system:
  - AWS CloudWatch gives 14 minute metrics/EC2 ~ 12*14*4 daily alerts
  - StatsD can collect 100+ metrics/server!

Context matters

AD should answer business needs



source: https://speakerdeck.com/eanichols/elizabeth-betsy-nichols

Takeaway

- **Time series are wild beasts**
- **Distributed systems are full of them**

Regardless of technical difficulties, first decision for any AD system has to be `time resolution` and `context` under which AD operates

AD methods

- Transformations and thresholds
- Statistical modelling and forecasting
- Machine learning
- Takeaway

Transformations and thresholds

Simplest AD: static threshold

- **Lower/upper bounds or # sigmas around mean or median**
- **Could be quite effective if metric is *chosen* right (meaning - made stationary)**
  - **Univariate transformations**
  - **Combination of metrics**

Transformations

Delta: `raw[n] -> raw[n] - raw[n-1]`

Rate: `raw[n] -> raw[n] / time`

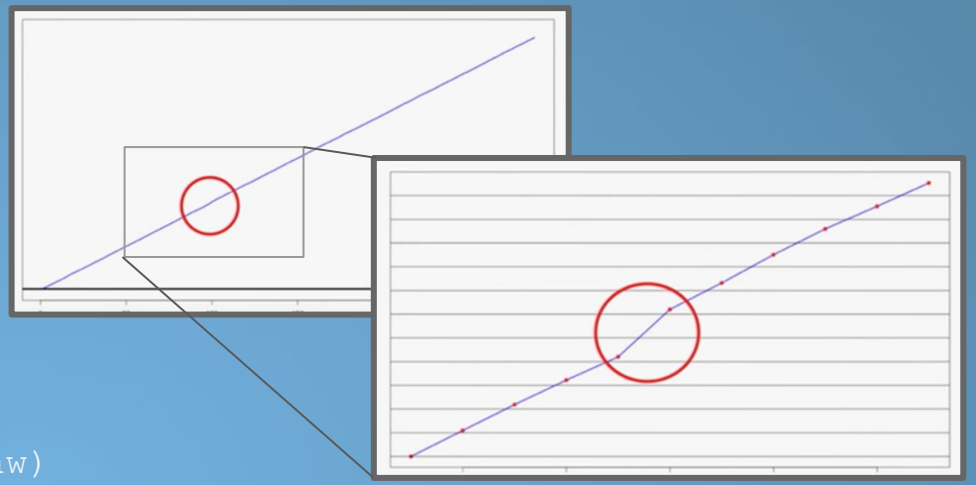
Scale: `raw[n] -> raw[n] * const.`

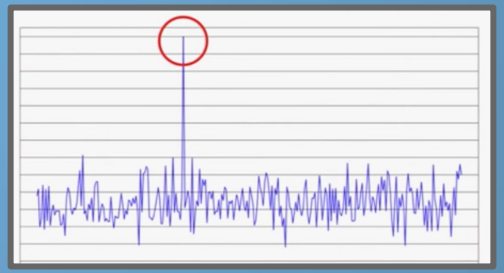
Min/Max: `raw[n] -> min(raw), max(raw)`

Ratios: `raw[n] -> raw[n] / max(raw)`

Summary: `raw[n] ->`

`(25th percentile, median(raw), ...)`





Example:
monotonic metrics (# of reads of popular blog post) -> delta

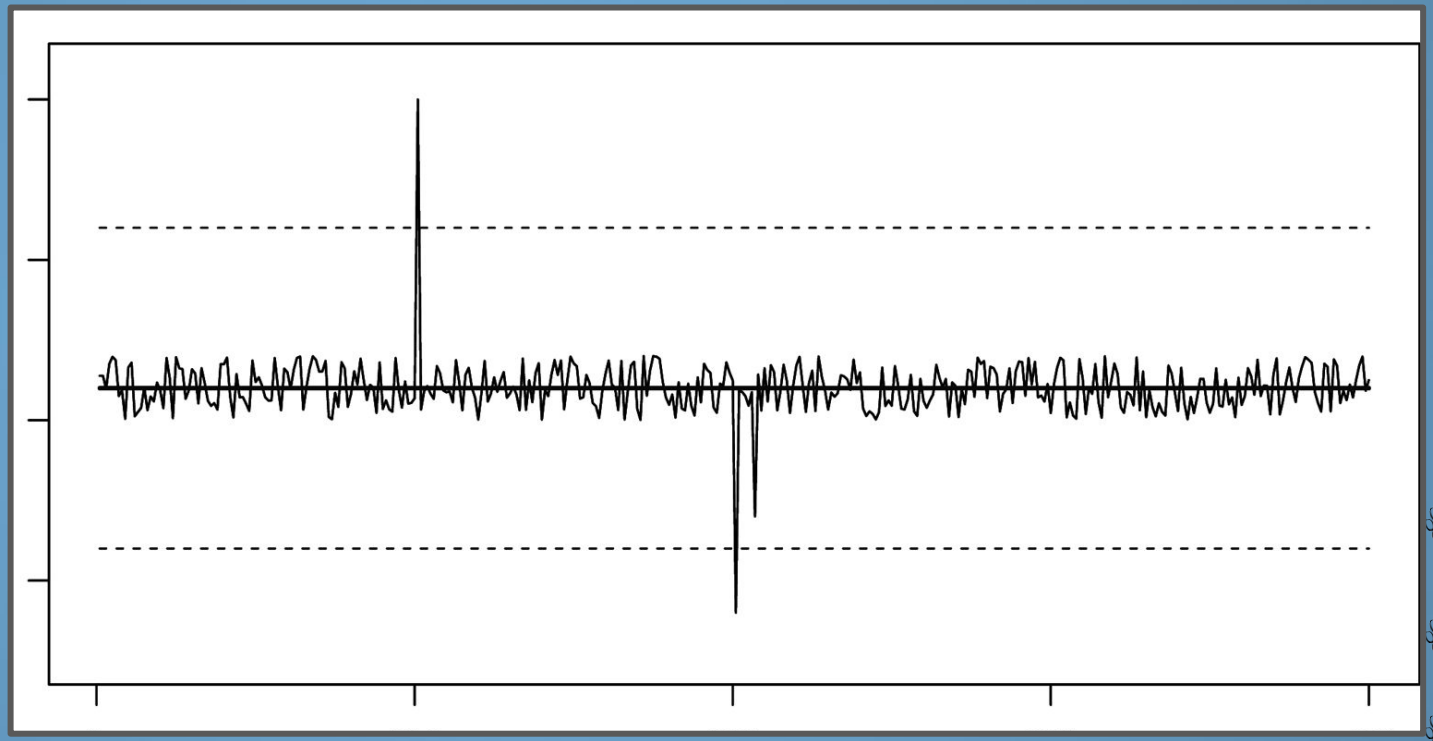
Combination of metrics

Some metric may not be anomalous per se, but taken together with some other...

Example:

If # of purchases increases over time but revenue decreases -> price is not adjusted properly and we should look at ratio as metric

Simplest anomaly detector



Adjust for trend/seasonality

Differencing

```
import pandas as pd
import numpy as np
from scipy.stats import norm

def variance_detector(df):
    """
    :param ts: pandas DataFrame with columns 'timestamp' and 'value'
    :return: pandas DataFrame with added columns for detected anomalies
        and probabilities
    """
    new = df.copy()
    diffs = np.abs(new.value.diff(periods=1))
    new['diff'] = diffs
    new['predicted_anomaly'] = diffs > 3*diffs.std()
    new['probability'] = norm.cdf(diffs, 0, 1)

    return new
```

Assumption: Residuals follow normal distribution

Same variance detector in pySpark 2.3.0

```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count, rand, collect_list, explode, struct, count, lit

spark = SparkSession.builder.master("local").appName("ODSC-2.3.0").getOrCreate()

df = spark.range(0, 10 * 1000 * 1000).withColumn('id', (col('id') / 10000).cast('integer')).withColumn('value', rand())
df.cache()

print(df.count())
print(spark.version, pyspark.__version__)
```

```
10000000
2.3.0 2.3.0
```

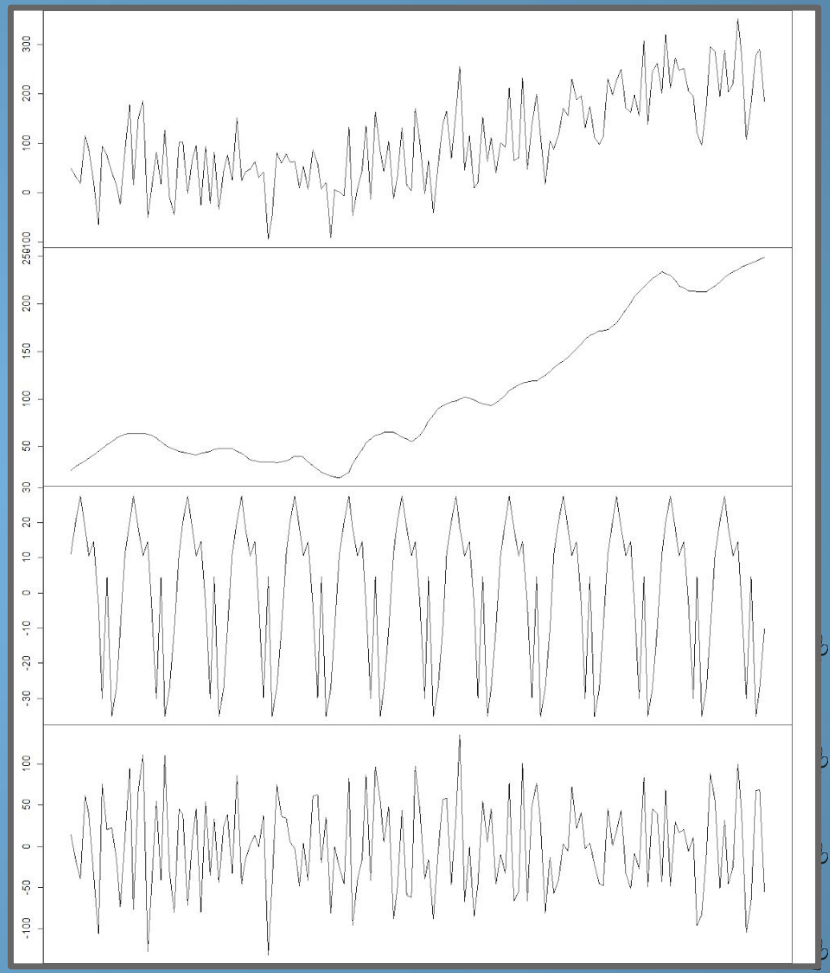
```
import pandas as pd
from scipy import stats
from pyspark.sql.types import *
from pyspark.sql.functions import pandas_udf, PandasUDFType

@pandas_udf("id long, v double, is_anomaly double, probability double", PandasUDFType.GROUPED_MAP)
def pandas_cdf(pdf):
    diffs = pdf.value.diff(periods=1)
    pdf['is_anomaly'] = diffs > 3*diffs.std()
    pdf['probability'] = stats.norm.cdf(diffs, 0, 1)

    return pdf

df.groupby('id').apply(pandas_cdf).show()
```

```
+---+-------------------+----------+-------------------+
| id|                  v|is_anomaly|        probability|
+---+-------------------+----------+-------------------+
|148| 0.3433535847367343|       0.0|               null|
```

Adjust for seasonality

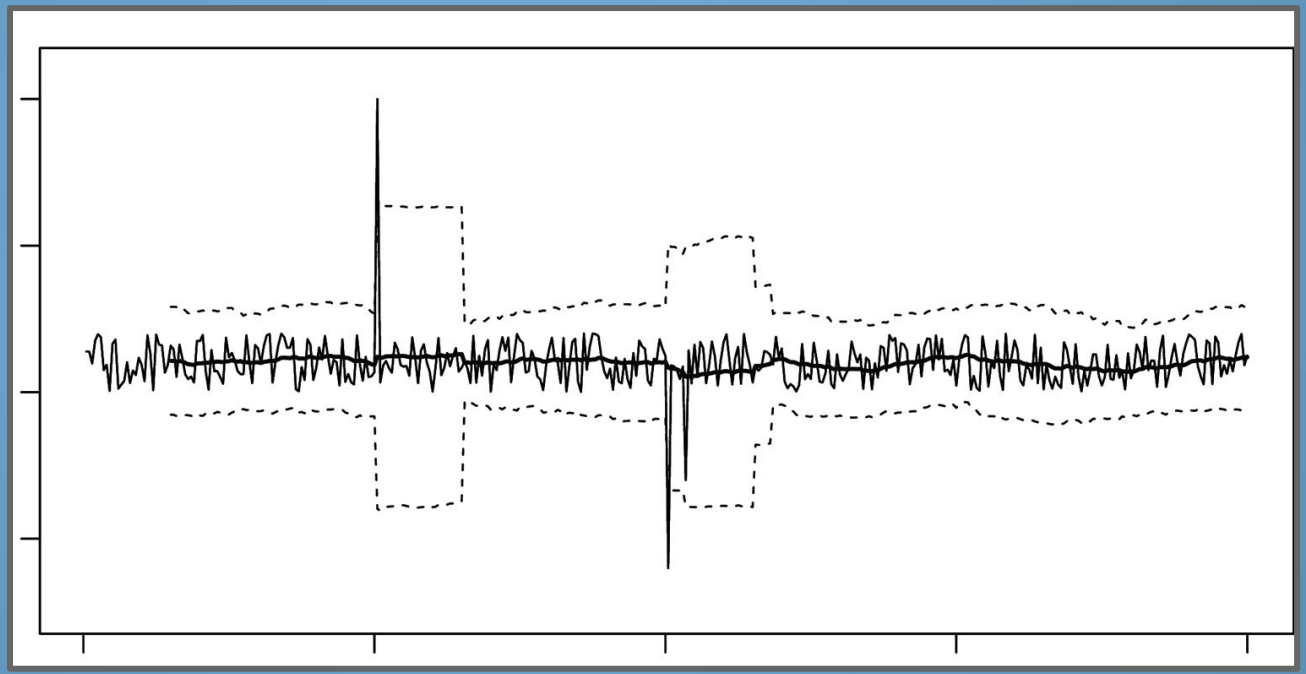
STL decomposition

- Trend
- Seasonality
- Residual



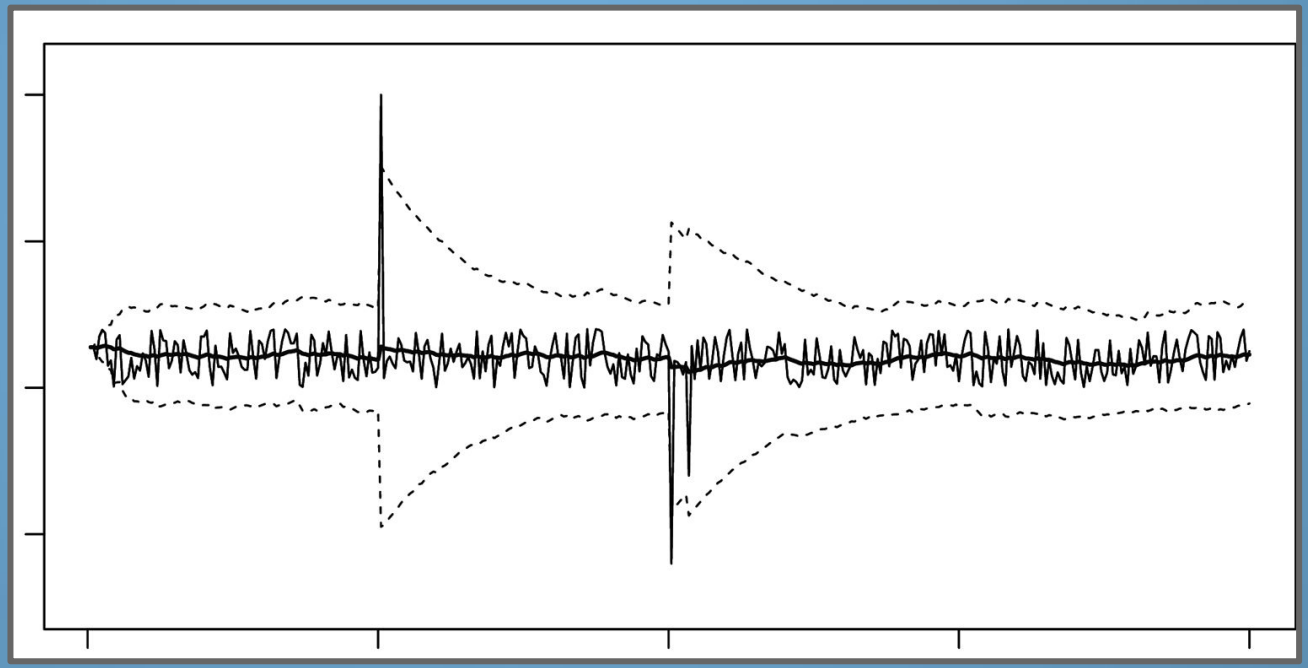
Moving average (MA)

Requires history of values



Exponentially weighted MA

$$S_t = \alpha y_{t-1} + (1-\alpha) S_{t-1} \quad 0 < \alpha \leq 1 \quad t \geq 3$$



Statistical modelling

(S)ARIMA(X) - generalization of autocorrelation/non-stationarity

Holt-Winters (triple exponential smoothing)

Forecasting and AD

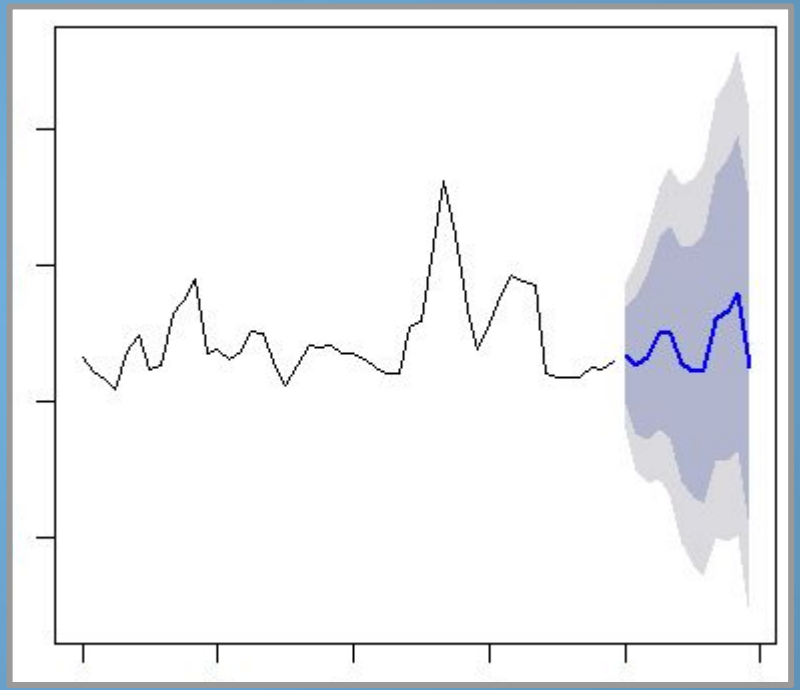
Confidence intervals

II

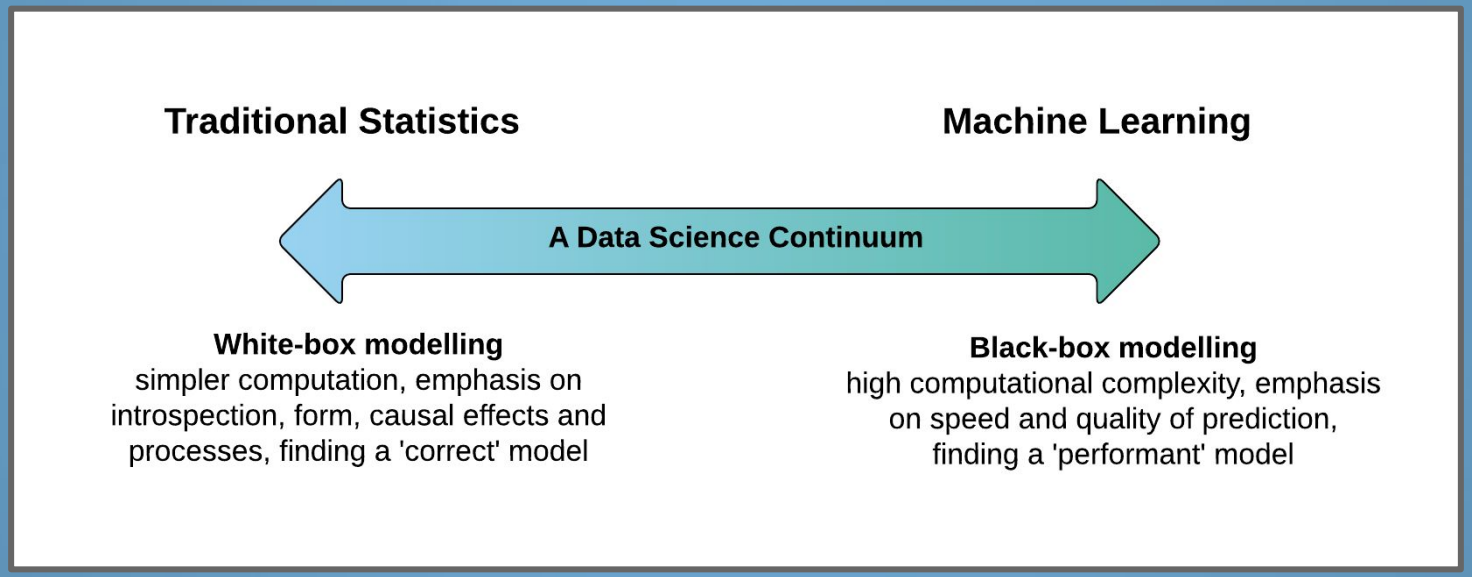
Anomaly bounds

Not robust to outliers or change of distribution

Often useful to “forget” history if data is volatile



Machine learning for AD

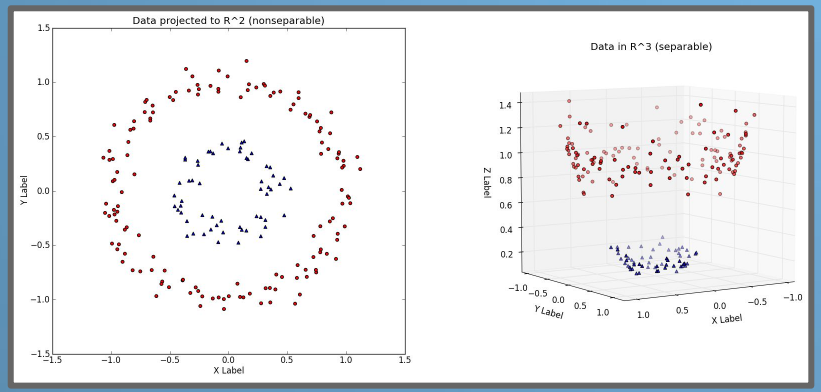


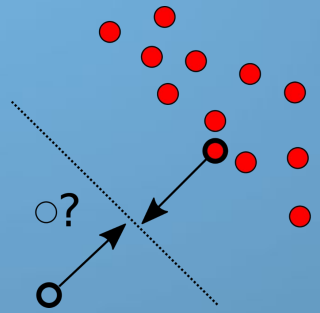
Machine learning for AD

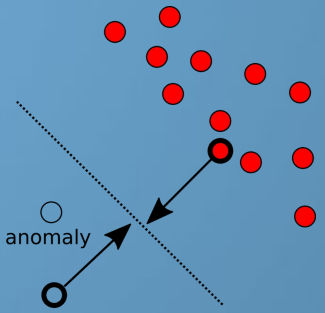
- Supervised vs. unsupervised
- Very unbalanced classes for classification
- Strong candidate for Human-in-the-loop approach
- Example methods: OC-SVM, rPCA, LSTM, Autoencoders

OC-SVM: One-class Support vector machine

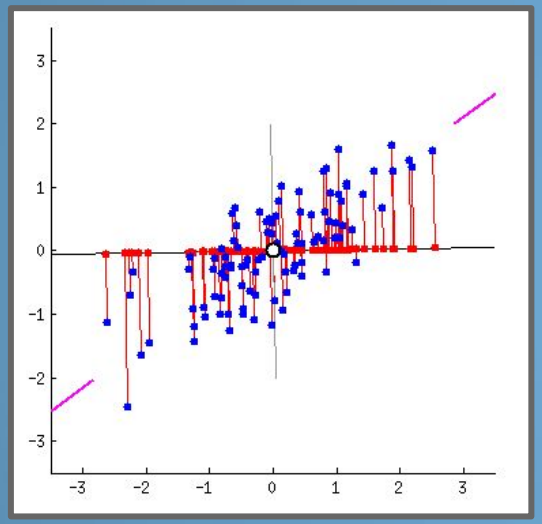
- Hyperplane separates normal class from rest
- Could have non-linear kernel

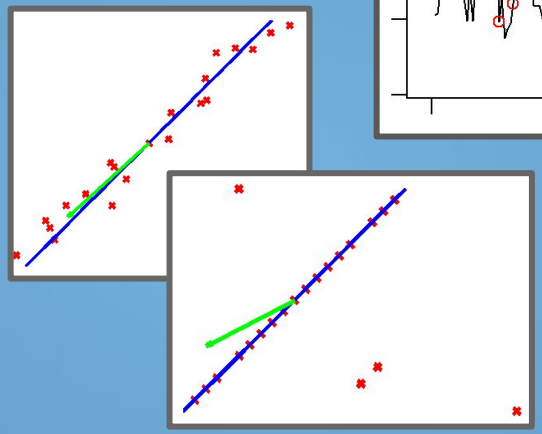


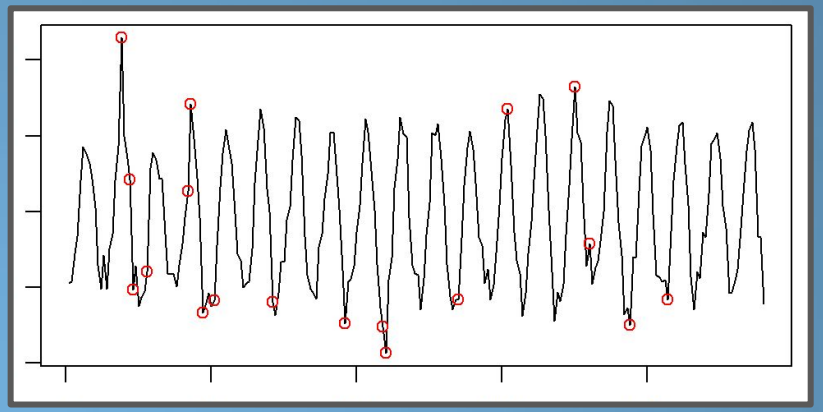




Robust PCA

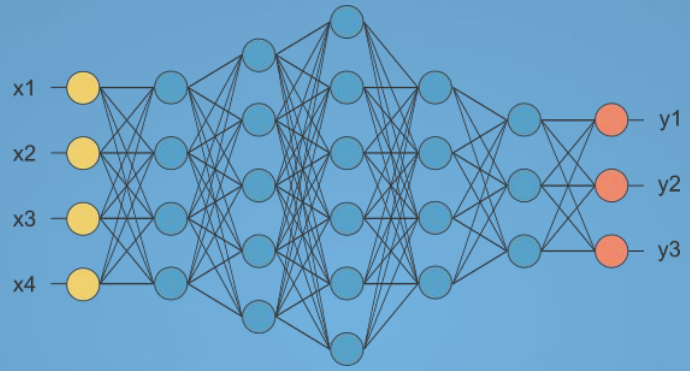






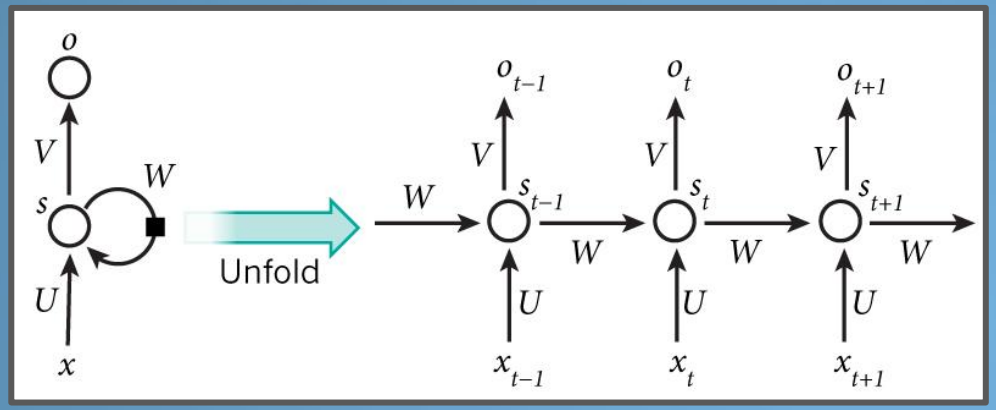
PCA is sensitive to outliers, rPCA models outliers too

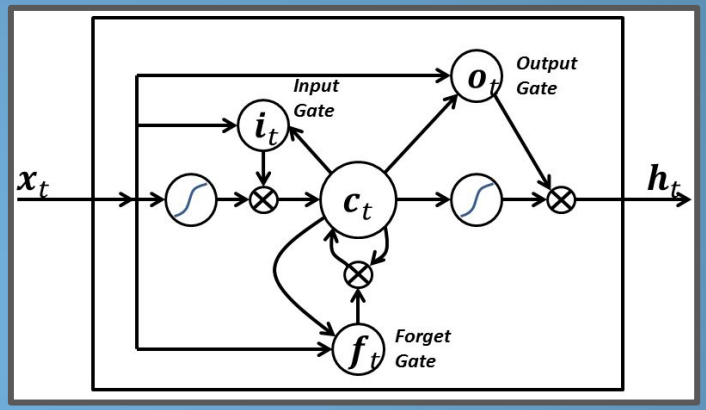
Deep learning for AD



- Neural networks, applied to sequence prediction
- Model non-linearities
- Model values order with recurrent NN blocks

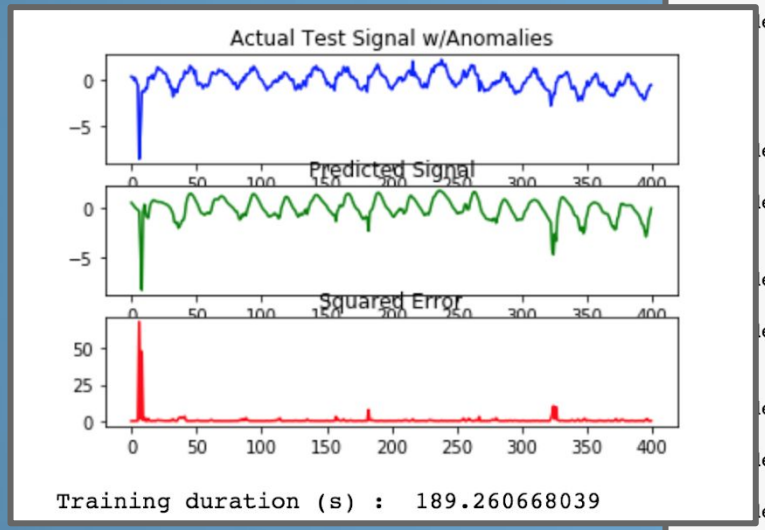
LSTM: Long short-term memory





- Predicted sequence is compared with incoming values
- Online training is hard

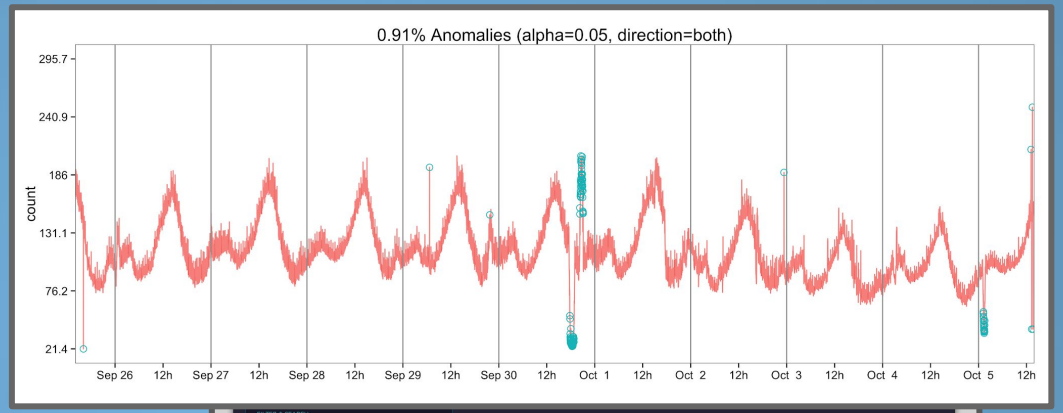
Simple LSTM for AD

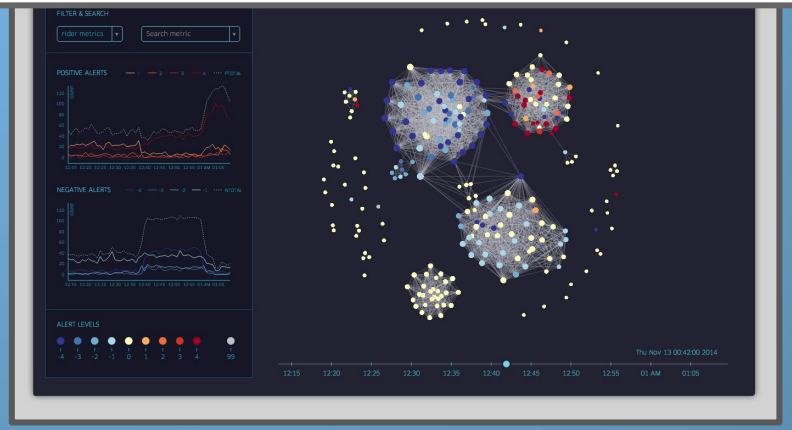



```
def build_model():
    model = Sequential()
    layers = {'input': 1, 'hidden1': 64, 'hidden2': 256, 'hidden3': 100, 'output': 1}

    el.add(LSTM(
          input_length=sequence_length - 1,
          input_dim=layers['input'],
          output_dim=layers['hidden1'],
          return_sequences=True))
    el.add(Dropout(0.2))

    el.add(LSTM(
          layers['hidden2'],
          return_sequences=True))
    el.add(Dropout(0.2))

    el.add(LSTM(
          layers['hidden3'],
          return_sequences=False))
    el.add(Dropout(0.2))

    el.add(Dense(
          output_dim=layers['output']))
    el.add(Activation("linear"))

    start = time.time()
    model.compile(loss="mse", optimizer="rmsprop")
    print "Compilation Time : ", time.time() - start
    return model
```

Twitter, Netflix, Uber

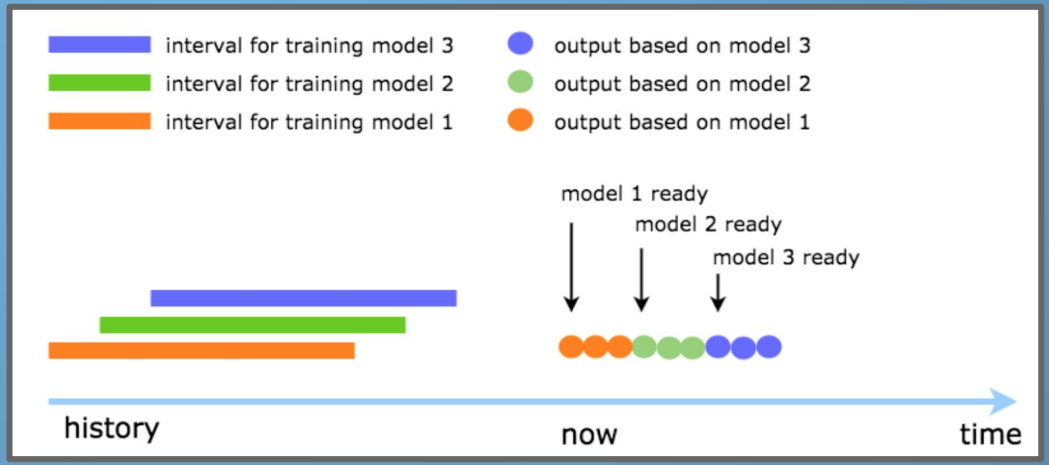
- STL + ESD
- rPCA
- Outlier+clustering
  LSTM





AD models training

- Streaming (online) training
- Batch
  - Incremental
  - Rolling

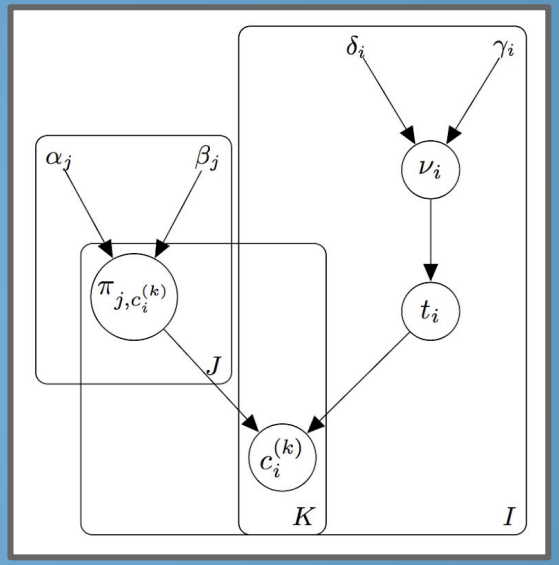


How to lower FPR

- Filters
  - Alert when *N* anomalies are detected
  - Don’t alert if variance-to-mean is high
  - Don’t alert if previous alert is unresolved
- Add context (additional data, rules)

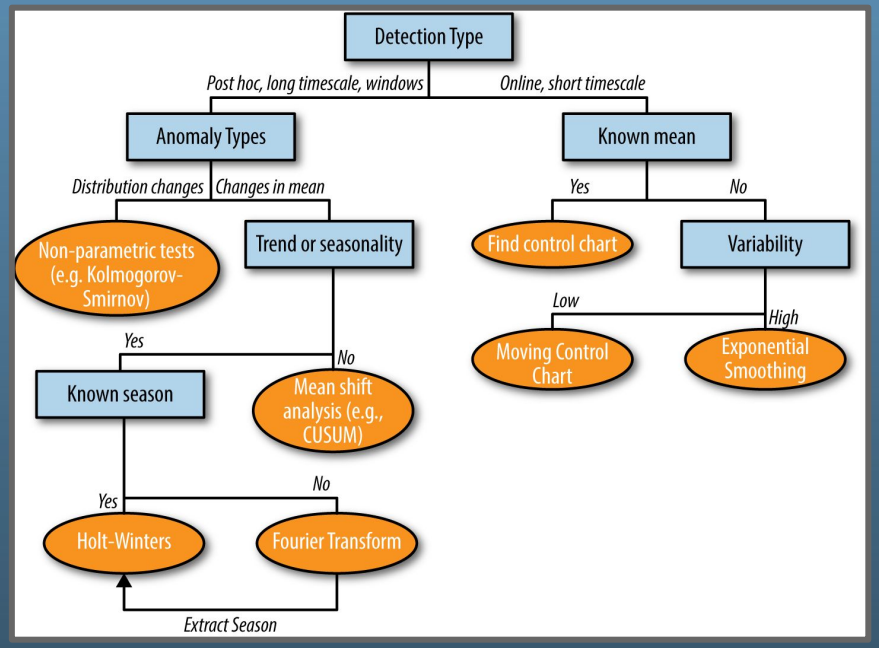
Ensemble methods

- Another way to lower FPR
- Unsupervised ensembles?



- Bayesian Classifier Combination for AD

Takeaway



+ If labeled: OC-SVM

+ If multivariate or
+ nonlinear: LSTM

Source: Anomaly Detection for Monitoring: A Statistical Approach to Time Series Anomaly Detection

AD at scale

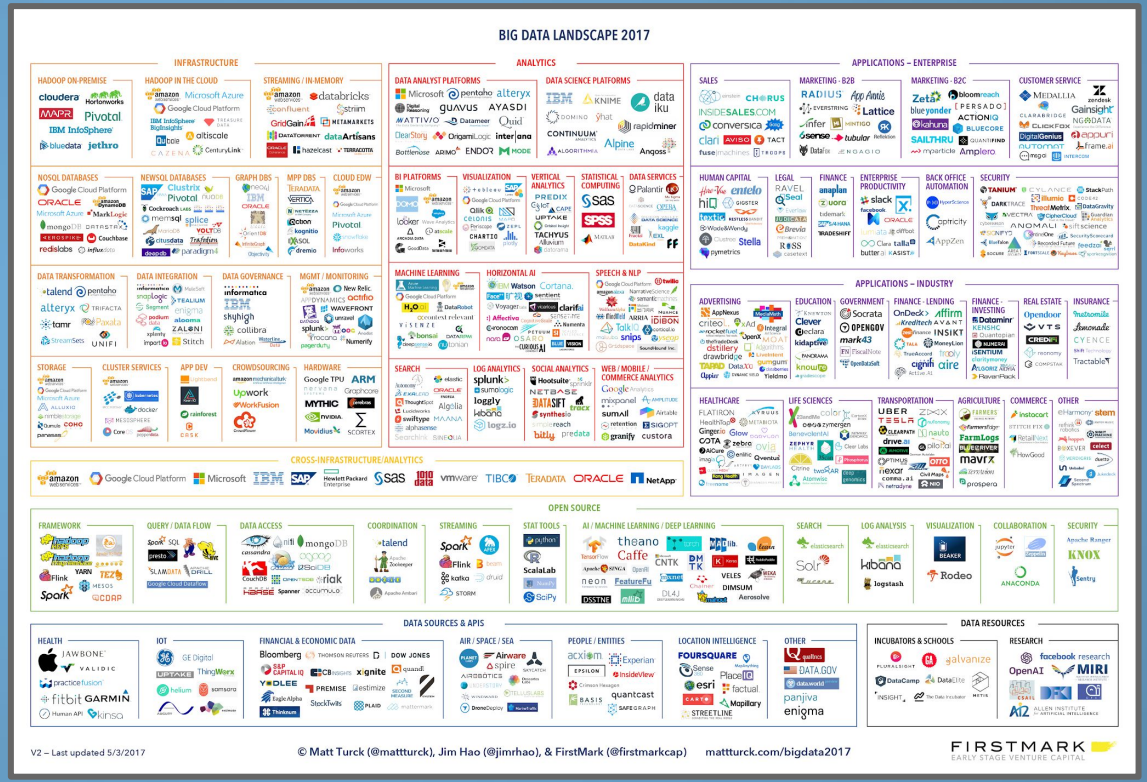
- Big Data technologies
- Selection of right tools
- Example stack
- Takeaway

AD system components

- Metrics and logs collection - connectors, APIs
- Time series databases, messaging
- Processing (Streaming and/or Batch)
- Downstream storage
- Visualization, alerting, administration

Big Data Landscape

- DBs
- Warehouse
- Caching
- Processing
- Visualization

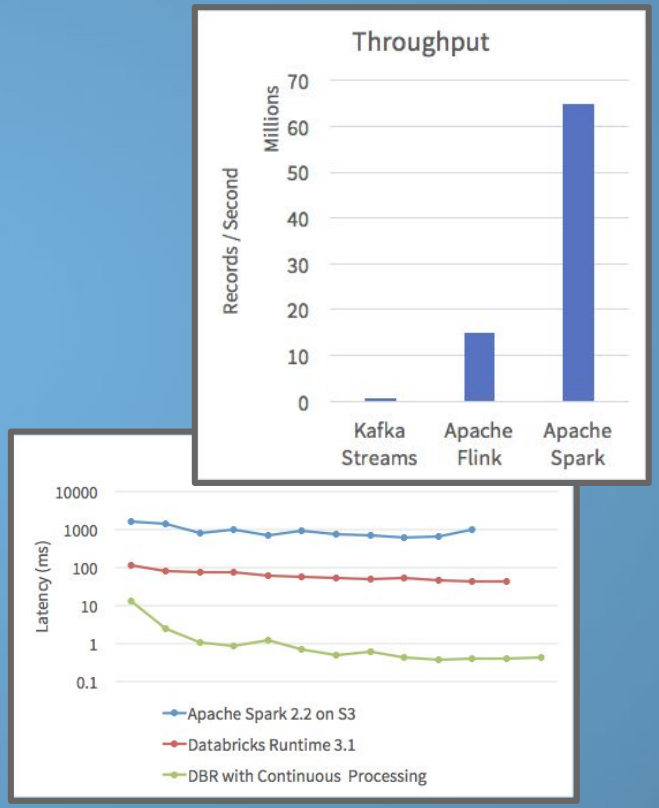


Open source DBs and streaming data

- Cassandra - NoSQL fault-tolerant distributed DB
- Kafka - data streams storage and messaging
- InfluxDB - Time series DB

Open source (streaming) processing

- Kafka Streams
- Apache Flink
- Spark Streaming
  - DStreams
  - Structured Streaming



Deep learning on Spark

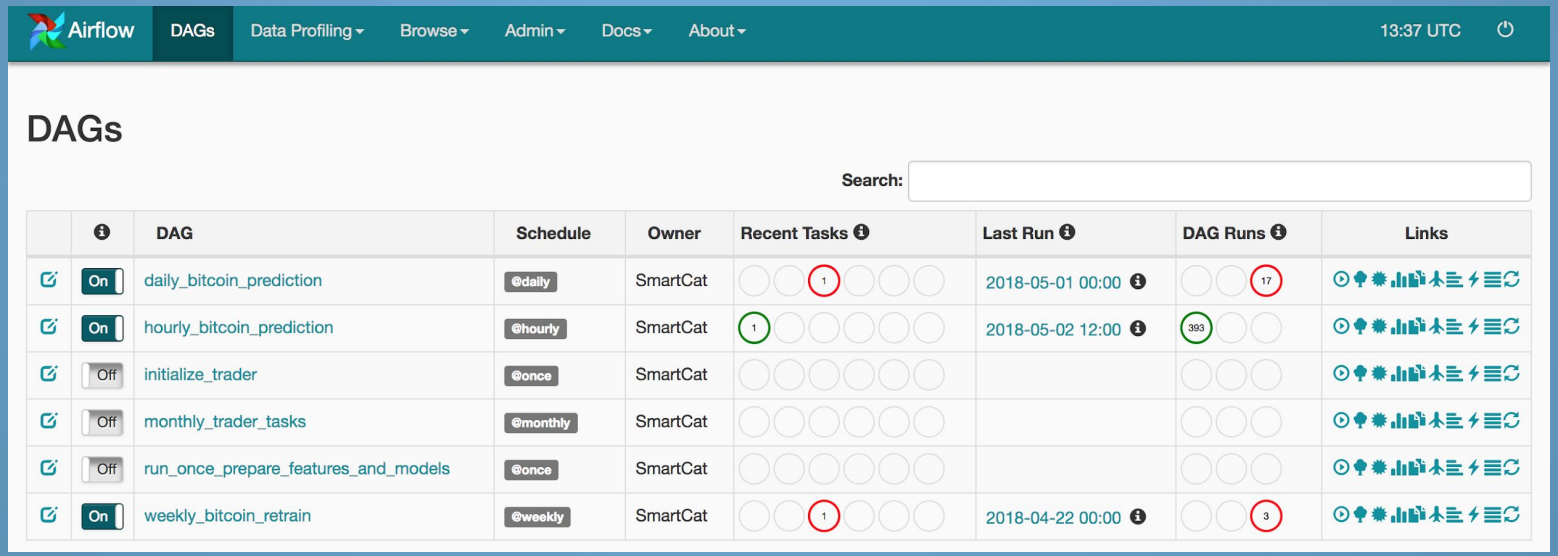
- Distributed Keras (CERN)
- TensorFlowOnSpark (Yahoo! Inc.)
- Elephas: http://maxpumperla.github.io/elephas/
- Deep Learning Pipelines (Databricks)

Open source visualization

- Kibana from ELK
- Grafana

Open source scheduling

- Apache Airflow



Selection

- On premise vs cloud
- Self-managed vs SaaS
- Technical/business considerations

Technical considerations

- Detection delay
- Out-of-order fault tolerance
- (Very) Low latency
- Exactly-once

Takeaway

- **Choose right tools**
- **Don’t over engineer**
- **Your experience?**

DevOps

- From DevOps to DataOps
- Automation helps experimentation
- Takeaway

DevOps in Big Data era

- Automation, deployment, reproducibility
- Orchestrate data, tools, code and environments
- Bridges a gap between theory and practice or local tests and production-like env
- Fast experimentation and iterations

Technologies

- docker-compose & docker swarm
- Terraform
- Ansible
- AWS Cloud Formation

Takeaway

Automate infrastructure provisioning and data flow so data scientist could `experiment` **often and in** `production-like` **environment**

SmartCat open source github.com/smartcat-labs

- Ansible playbooks
- Data generator
- Load generator
- Cassandra diagnostics

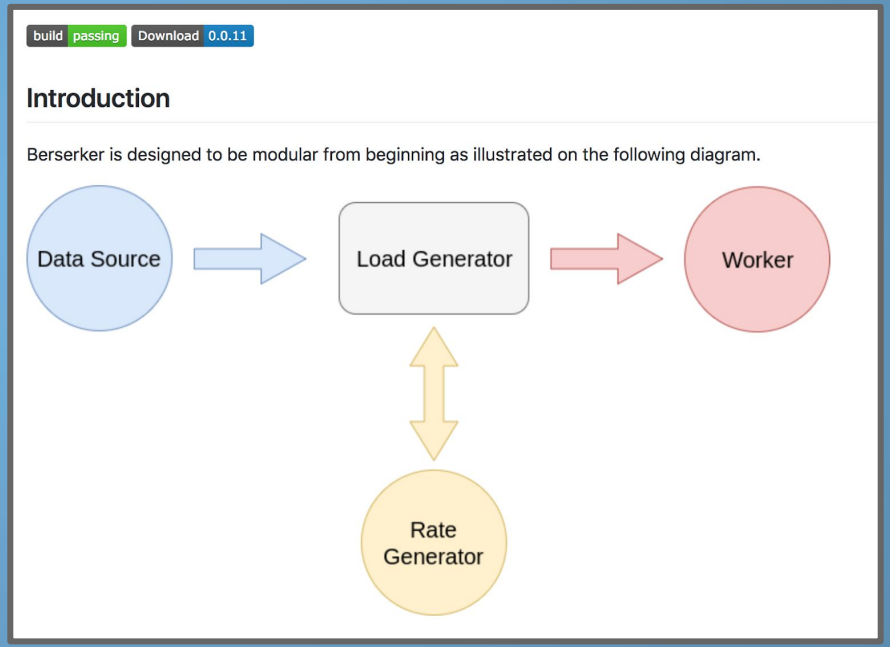


Image credits and references

- **LSTM: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/**
- **Big Data Landscape: http://mattturck.com/bigdata2017/**
- **Anomaly Detection for Monitoring: A Statistical Approach to Time Series Anomaly Detection, Preetam Jinka & Baron Schwartz**
- **A Bayesian Ensemble for Unsupervised Anomaly Detection, Edward Yu and Parth Parekh**

Thank you!

www.smartcat.io
dusan.randjelovic@smartcat.io