Monitoring ML applications in production

An overview

Alexander Kim

Production AI Conference | April 13th, 2019

Outline

About me

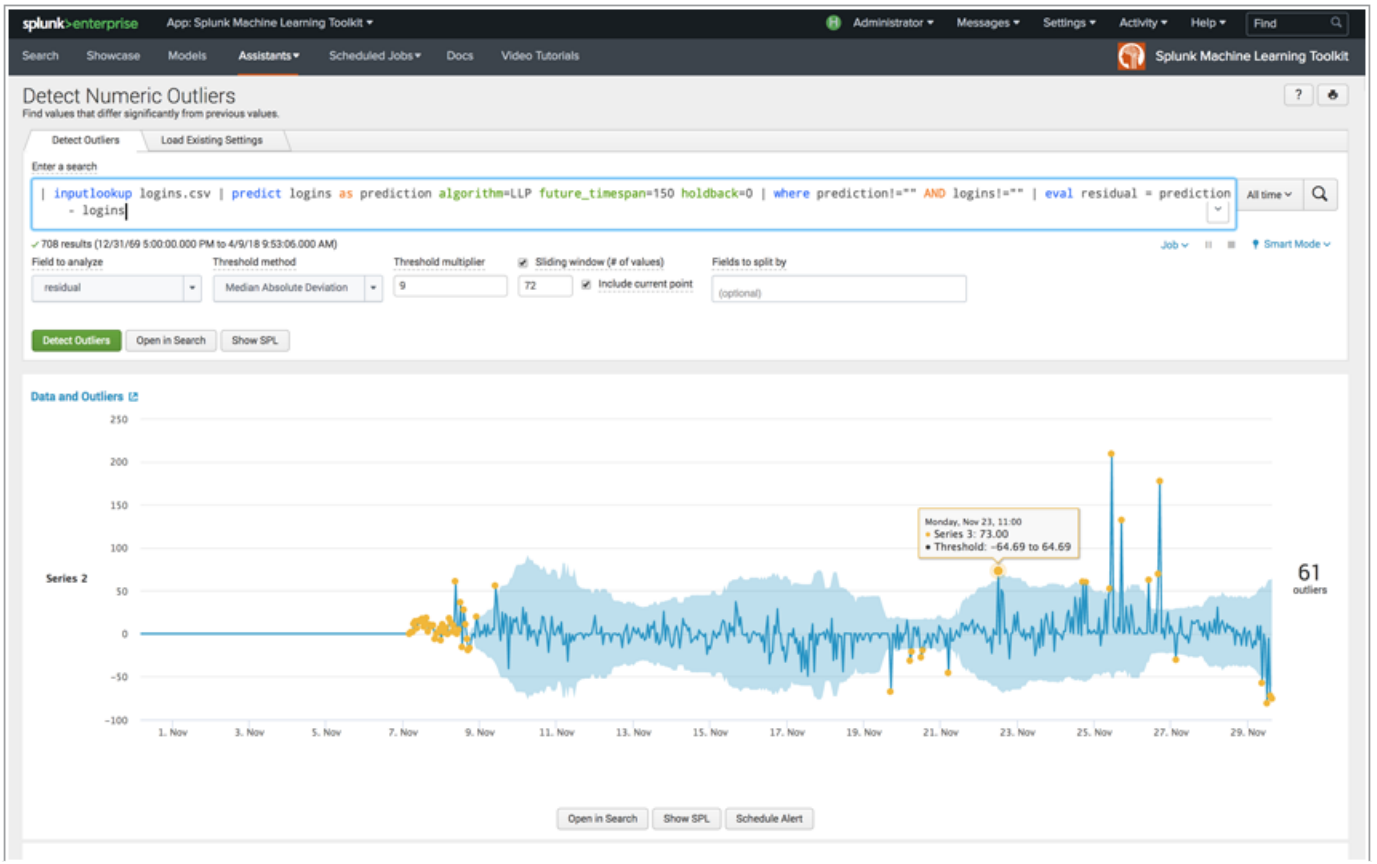
Past:

- B.Sc. & M.Sc. in Physics @ (Moscow State University, University of Alberta)
- Data Analysis & Image Processing @ UrtheCast
- Data Science @ Splunk

Currently:

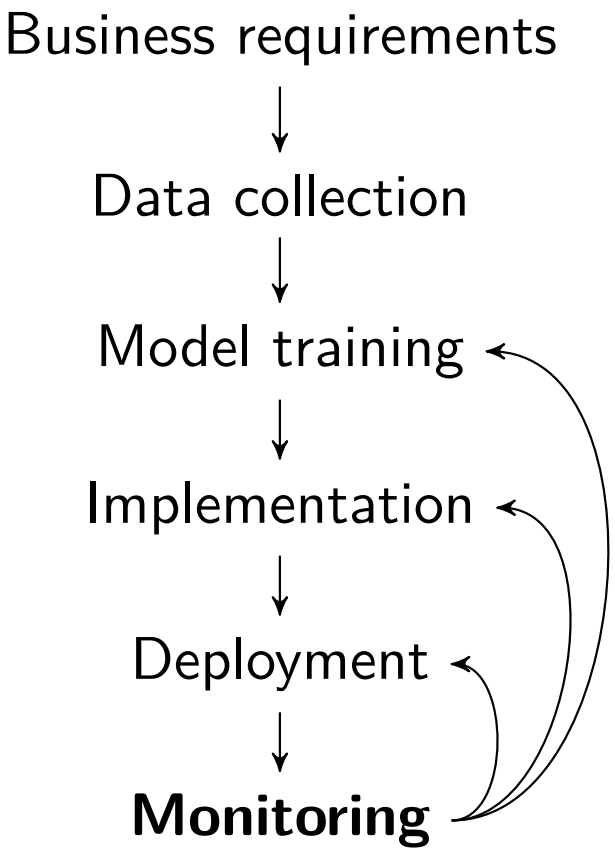
- Data Science @ [some private company ¯_(ツ)_/¯]
- Organizer @ PyData Montreal

source: urthecast.com



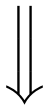
source: splunk.com

ML application life-cycle



Why monitor?

No monitoring



End-user monitoring

Why monitor?

No monitoring

⇓

End-user monitoring

⇓

"Twitter monitoring"

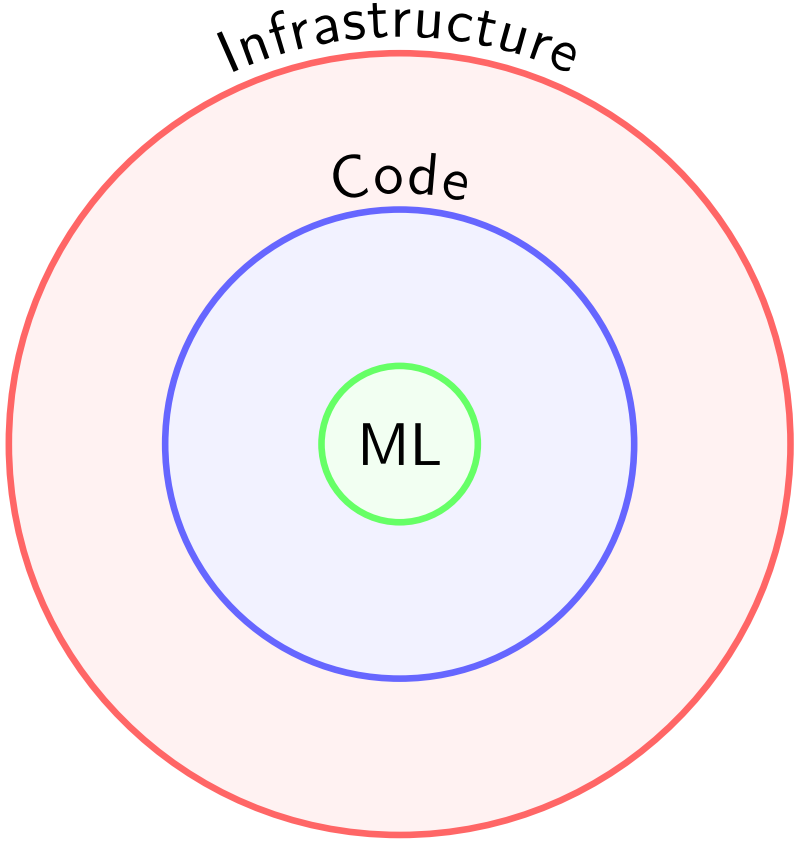
Why monitor?


No monitoring
End-user monitoring
"Twitter monitoring"
restic
@resticbackup
Follow
It's a bit ironic that today, on @WorldBackupDay, the website worldbackupday.com is down. We shou introduce "World Restore Day", a day four weeks later, where everybody should try disaster recovery from scratch...
AM - 31 Mar 2019
20 Retwe kes
20

ML application layers

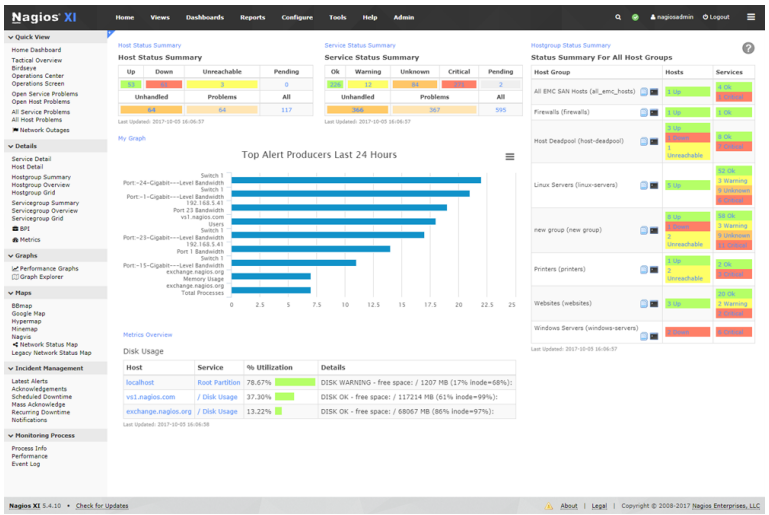
What are we monitoring?

- Infrastructure
- Code: performance and logic
- ML performance



Infrastructure monitoring

- Resource utilization & Security: CPU, storage, network, etc.
- Tools: Zabbix, Nagios, Amazon CloudWatch, etc.



source: zabbix.com, nagios.com

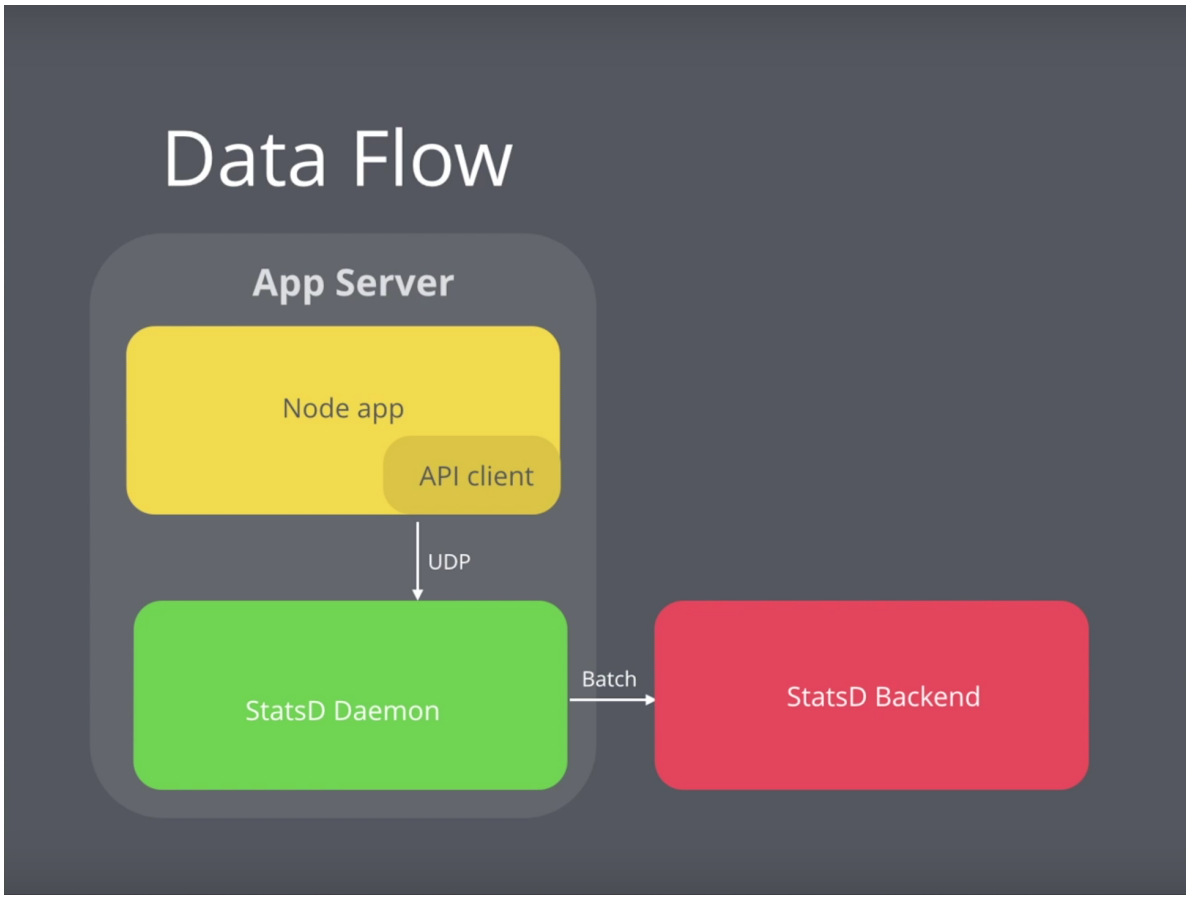
Code monitoring

- Instrumentation & metrics: **statsd, prometheus**, etc.
- Event logging & tracing: **logstash, splunk**, etc.

Instrumentation & metrics: Statsd

- API client →UDP protocol →Daemon →Collection backend
- Lightweight and simple, non-blocking, dimensional data model
- Metric types: counters, timers, gauges, sets
- External data storage

Instrumentation & metrics: Statsd

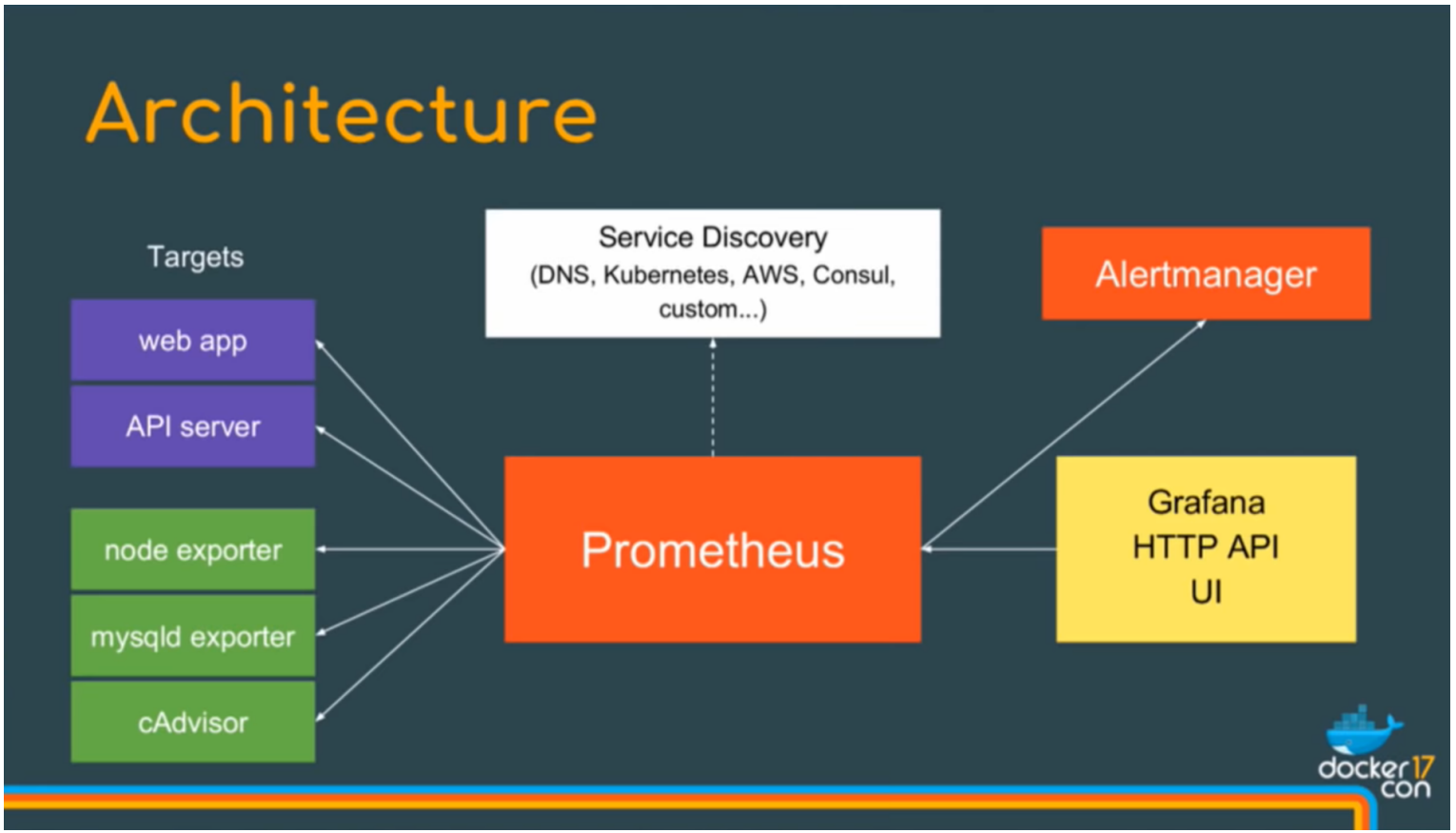


source: https://www.youtube.com/watch?v=NydIHi8Y224

Instrumentation & metrics: Prometheus

- Statsd-like functionality and more
- Built-in TSDB and dashboard, local storage*
- PromQL, Alerting

Instrumentation & metrics: Prometheus



source: https://www.youtube.com/watch?v=PDxcEzu62jk

Instrumentation & metrics: Statsd vs Prometheus

Statsd

```
...# other imports here
import statsd
c = statsd.StatsClient("my_host_name", 8125)
...# application code here
c.incr('http_requests_total.home.400')
```

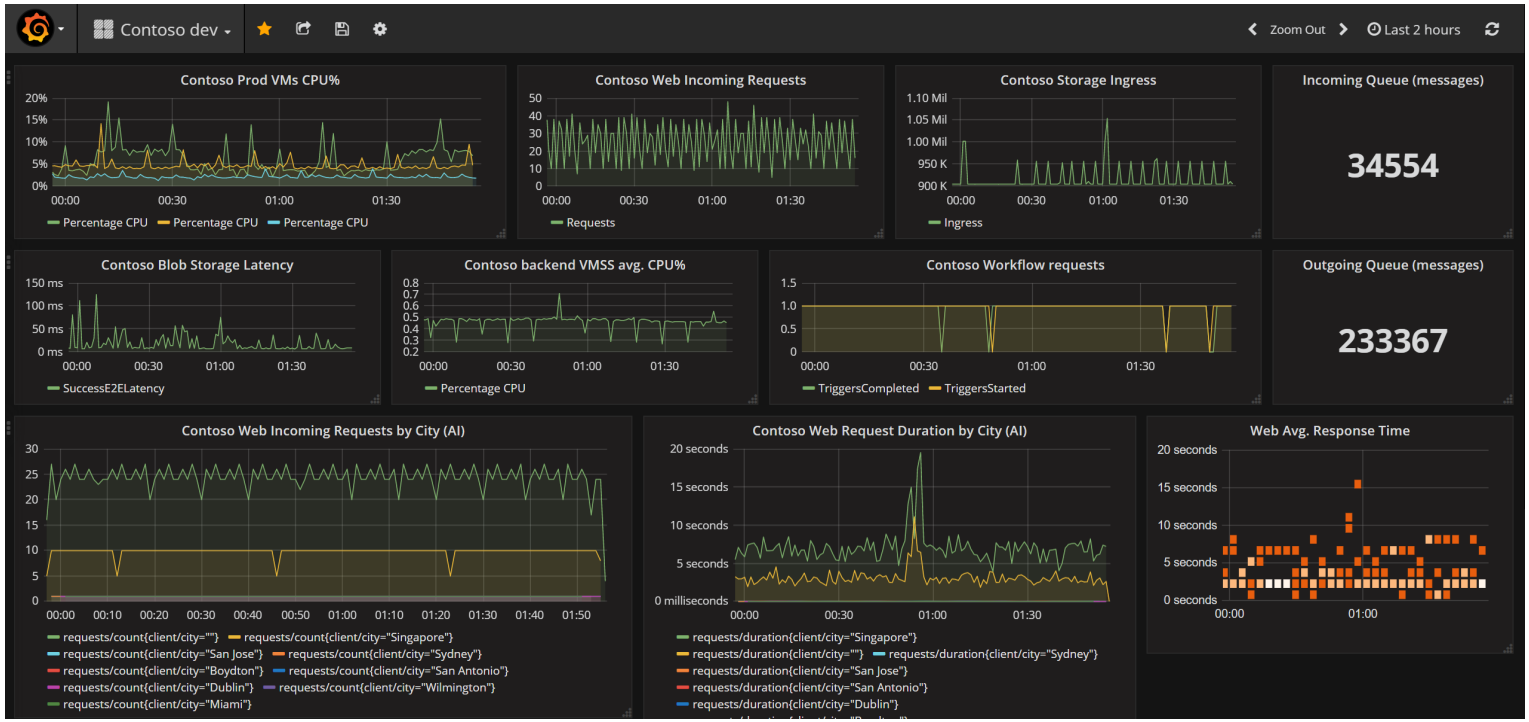
Prometheus

```
...# other imports here
from prometheus_client import Counter
c = Counter('http_requests_total', 'Total␣HTTP␣Requests␣(count)', ['method', '
     endpoint', 'status_code'])
...# application code here
c.labels(method='GET', endpoint="/home", status_code=400).inc()
```

Instrumentation & metrics: Statsd vs Prometheus

- Simplicity and low overhead →Statsd
- Large number of service instances →Prometheus
- from Statsd to Prometheus: `https://github.com/prometheus/statsd_exporter`

Instrumentation & metrics: Visualization



source: https://azure.microsoft.com/en-us/blog/monitor-azure-services-and-applications-using-grafana

Event logging & tracing

Elastic Stack

- Logstash & Beats
- Elasticsearch
- Kibana
- Elastic Stack Features (X-Pack)

source: https://medium.com/oneclicklabs-io

Event logging & tracing

Elastic Stack

- Logstash & Beats
- Elasticsearch
- Kibana
- Elastic Stack Features (X-Pack)

source: https://medium.com/oneclicklabs-io

Elastic Stack

Logstash

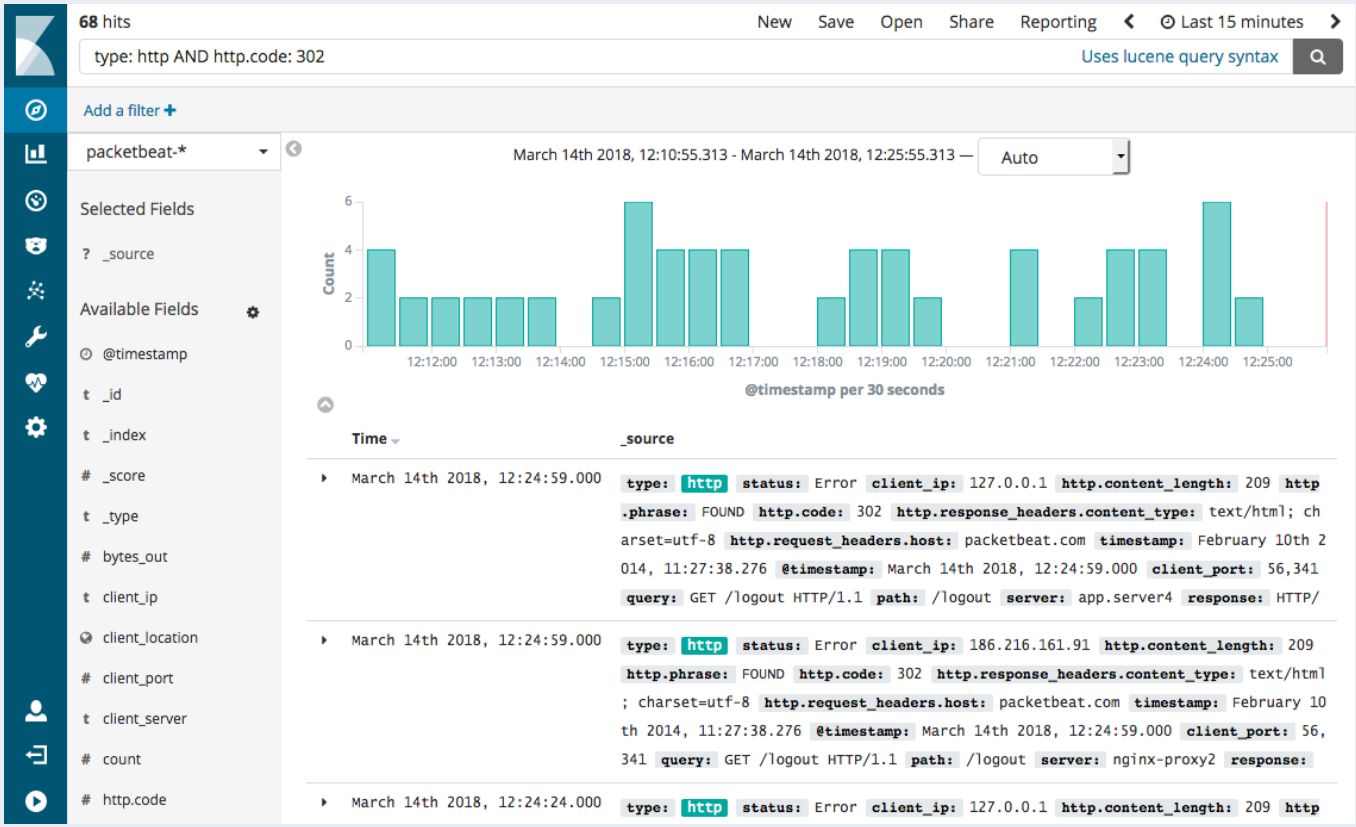
Logstash config

```
input {
  file {
    path => "/tmp/access_log"
    start_position => "beginning"
  }
}

filter {
  if [path] =~ "access" {
    mutate { replace => { "type" => "apache_access" } }
    grok {
      match => { "message" => "%{COMBINEDAPACHELOG}" }
    }
  }
  date {
    match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
  }
}

output {
  elasticsearch {
    hosts => ["localhost:9200"]
  }
  stdout { codec => rubydebug }
}
```

Elastic Stack

Kibana

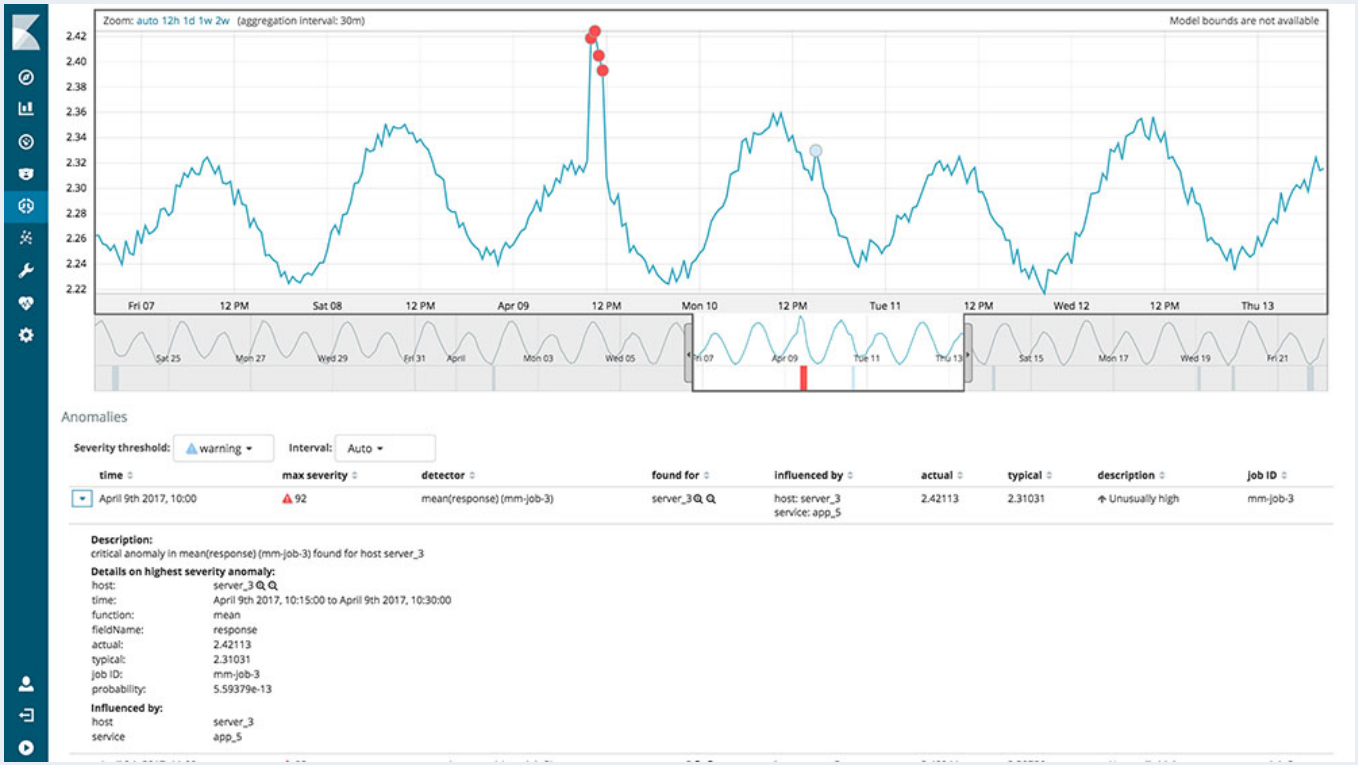
Kibana Dashboard



Elastic Stack

X-Pack

Elastic Stack Features (X-Pack)



source: https://www.elastic.co

Elastic Stack vs Splunk, Sumo Logic, etc.

- Open-source vs proprietary
- Customization vs off-the-shelf features
- Pay developers vs pay company

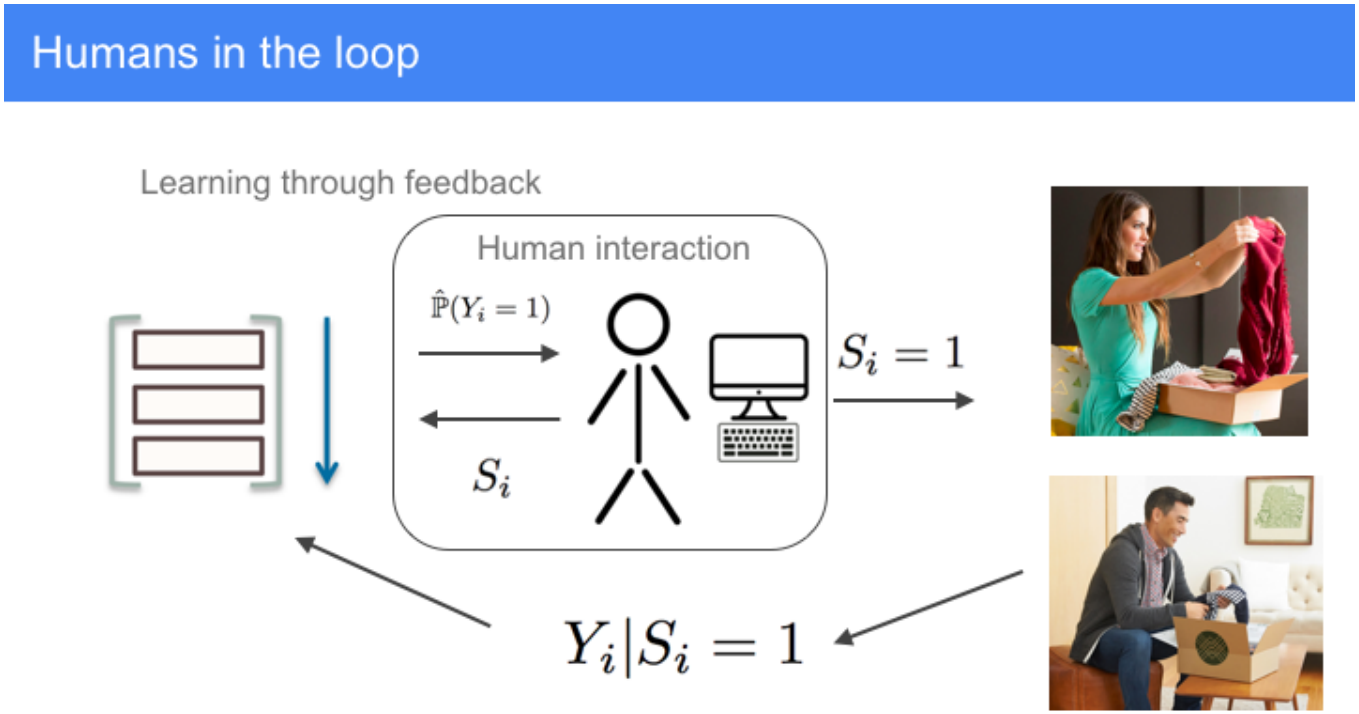
ML monitoring

- Comparison with ground truth
- Human-in-the-loop ML
- Model decay

Human-in-the-loop ML

- Frees engineers from edge cases
- Might be critical in some industries or mandated by law
- Content moderation teams, medical professionals, stylists, etc.

Human-in-the-loop ML



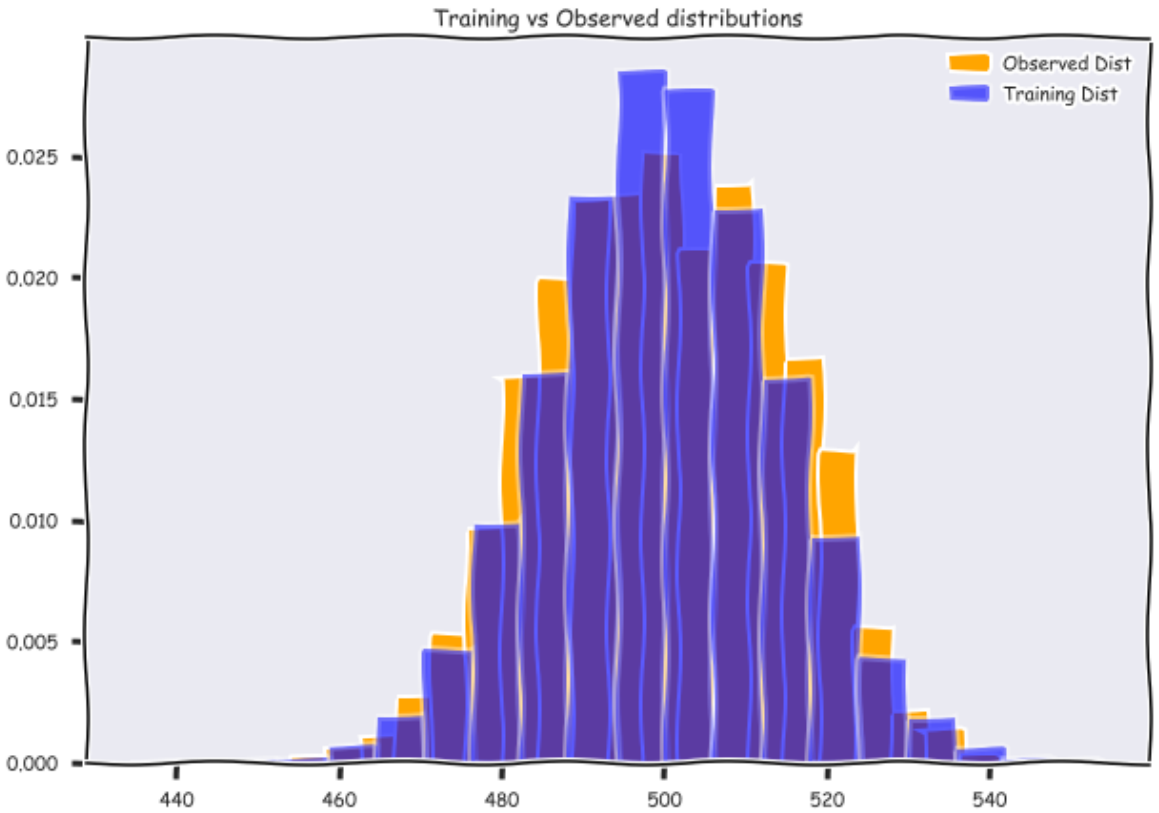
source: youtube.com/watch?v=_m8YOfnv-sg

Model decay

- Distributions change over time:
  - Macroeconomic factors
  - Data sources/integration
  - Internal changes (policy, strategy, UX, etc.)
- Statistical tests

Statistical tests: feature X

training and observed samples



Statistical tests: feature X

training and observed samples

```
N = 30000
M = 3000
scale_1 = 14.4
scale_2 = 10
x_train = np.random.normal(loc=500, scale=scale_1, size=N)

x_new_1 = np.random.normal(loc=490, scale=scale_2, size=int(M/2))
x_new_2 = np.random.normal(loc=510, scale=scale_2, size=int(M/2))
x_new = np.concatenate((x_new_1,x_new_2))
```

```
print(x_train[:5])
print(x_new[:5])
```

```
[525.40235378 505.7622638  514.09382697 532.26886207 526.89283506]
[493.71232144 493.04783891 495.04124602 491.35299604 496.5375878 ]
```

```
print(x_train.mean(), x_train.std())
print(x_new.mean(), x_new.std())
```

```
499.9370789319391 14.29489720125223
499.99017939635786 14.299418992619533
```

Population Stability Index (PSI)

$$PSI = \sum((X_{train}\% - X_{observed}\%) * ln(\frac{X_{train}\%}{X_{observed}\%}))$$

PSI Value	**Recommendation**
less than 0.1	No action required
between 0.1 and 0.25	Need to investigate and understand the changes
greater than 0.25	Feature X is no longer a good feature for this model

Population Stability Index (PSI)

```
df.head()
```

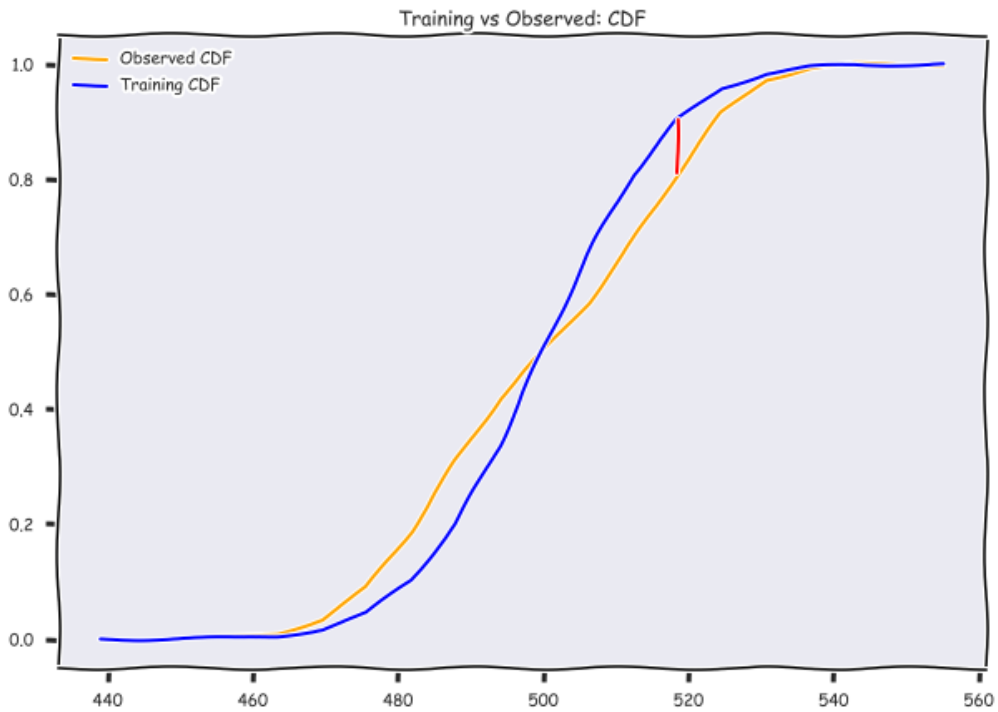
	bin_edges	training count	observed count	training percent	observed percent	psi
0	462.570797	74	7	0.002467	0.002333	0.000007
1	466.410157	159	11	0.005300	0.003667	0.000602
2	470.249517	288	28	0.009600	0.009333	0.000008
3	474.088877	463	44	0.015433	0.014667	0.000039
4	477.928237	835	80	0.027833	0.026667	0.000050

```
df['psi'].sum()
```

```
0.021066222095270176
```

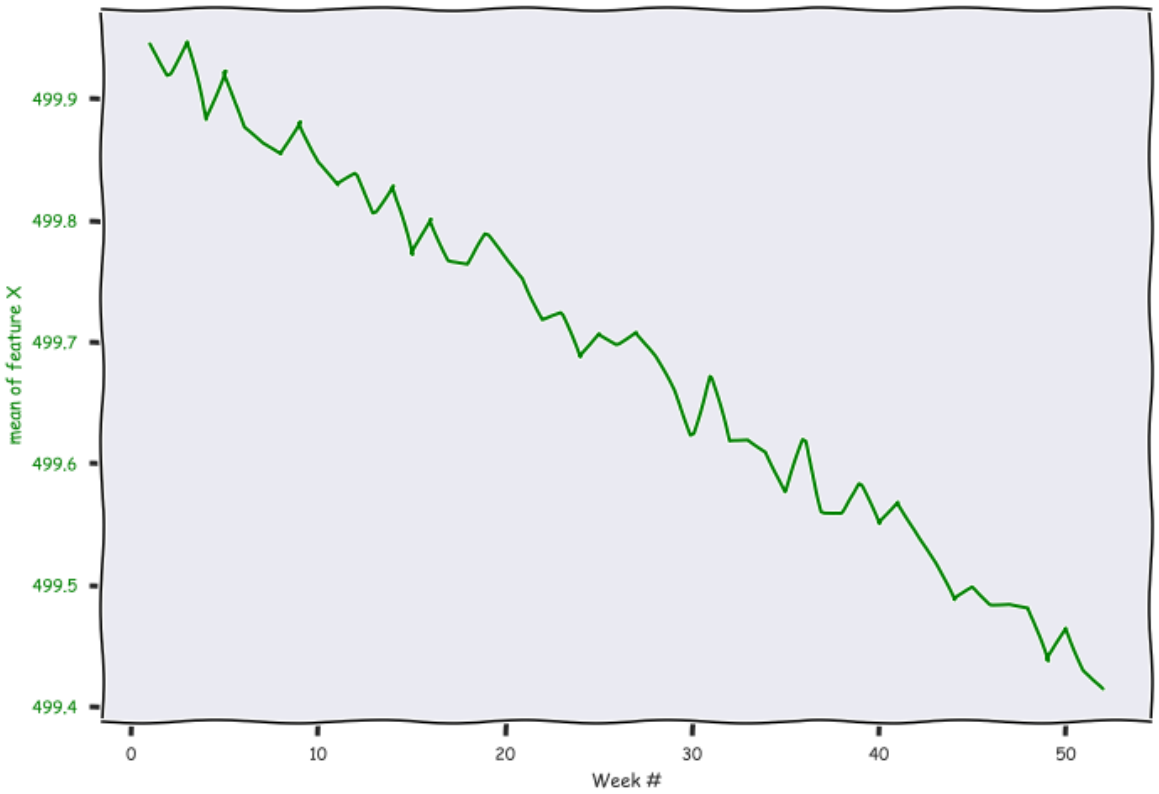
Kolmogorov–Smirnov test

$$D = max(abs(CDF_{training} - CDF_{observed}))$$



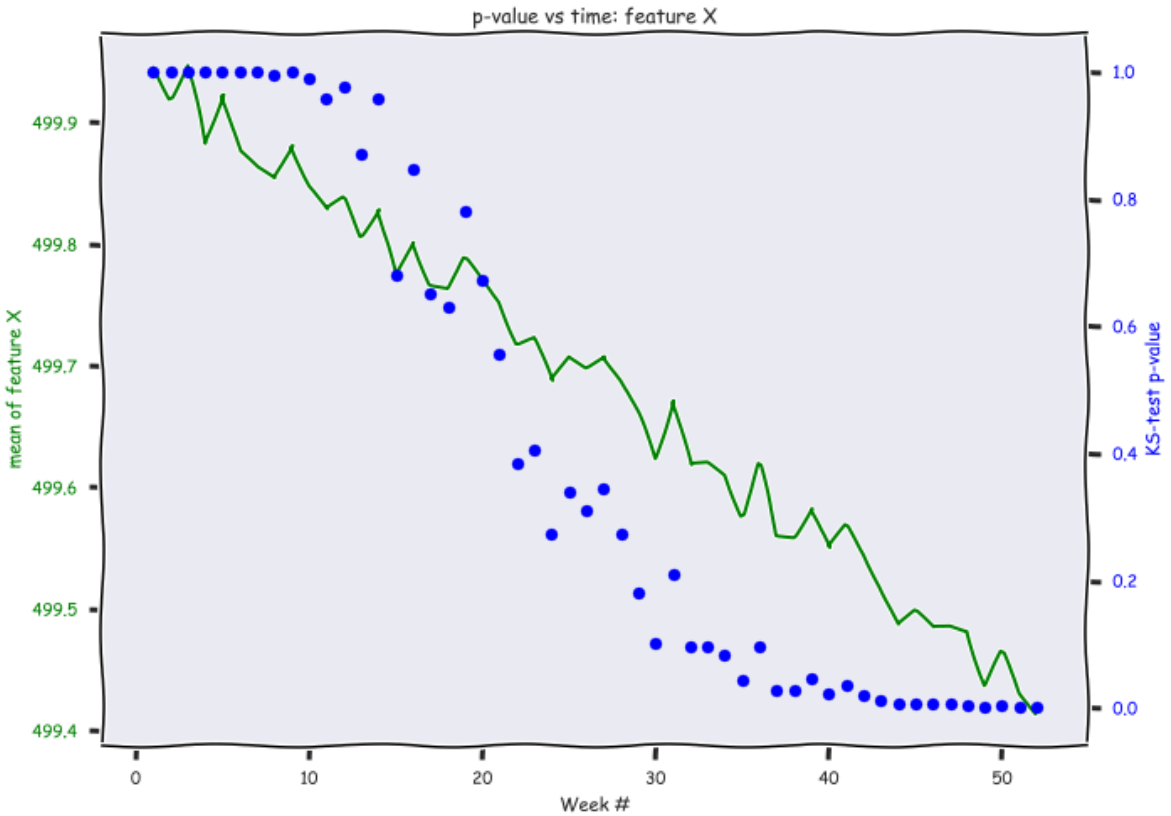
KS test

Scenario: mean of feature X decreases over time



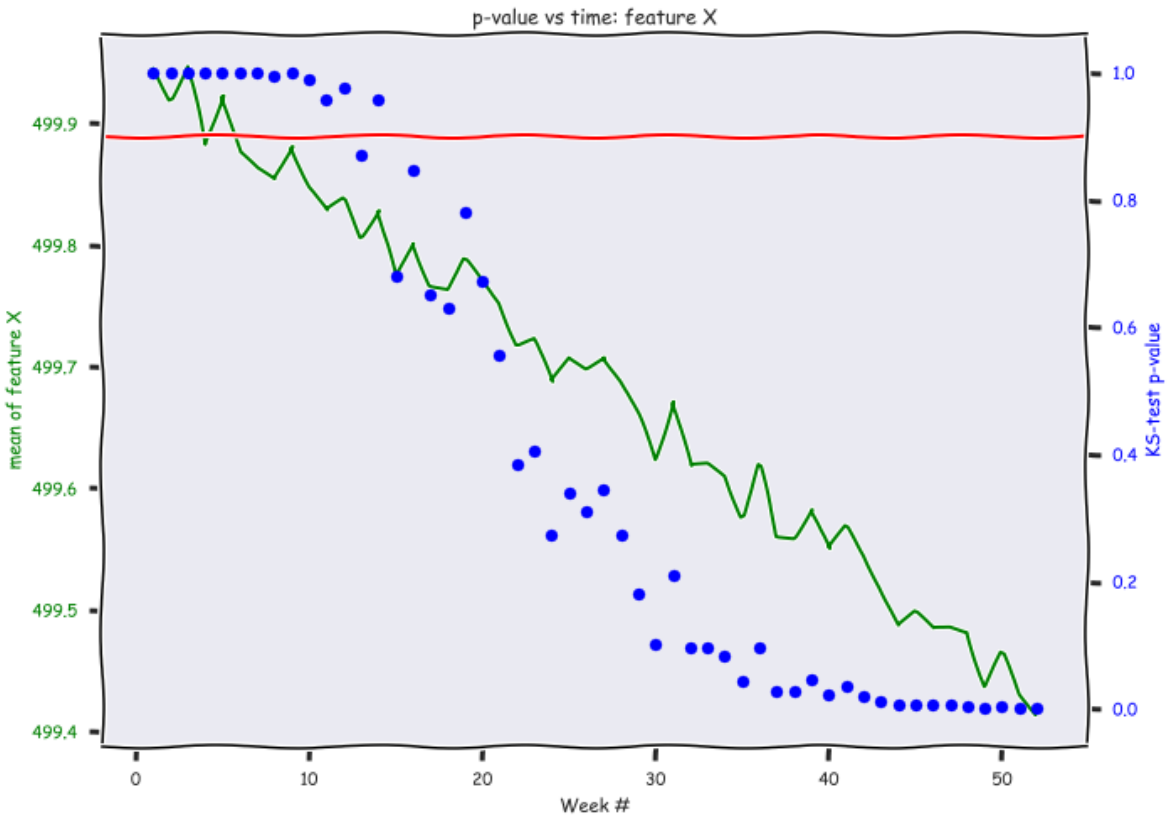
KS test

Scenario: mean of feature X decreases over time



KS test

Scenario: mean of feature X decreases over time



Other ways to detect change

- Loss function values vs time
- Model uncertainty vs time

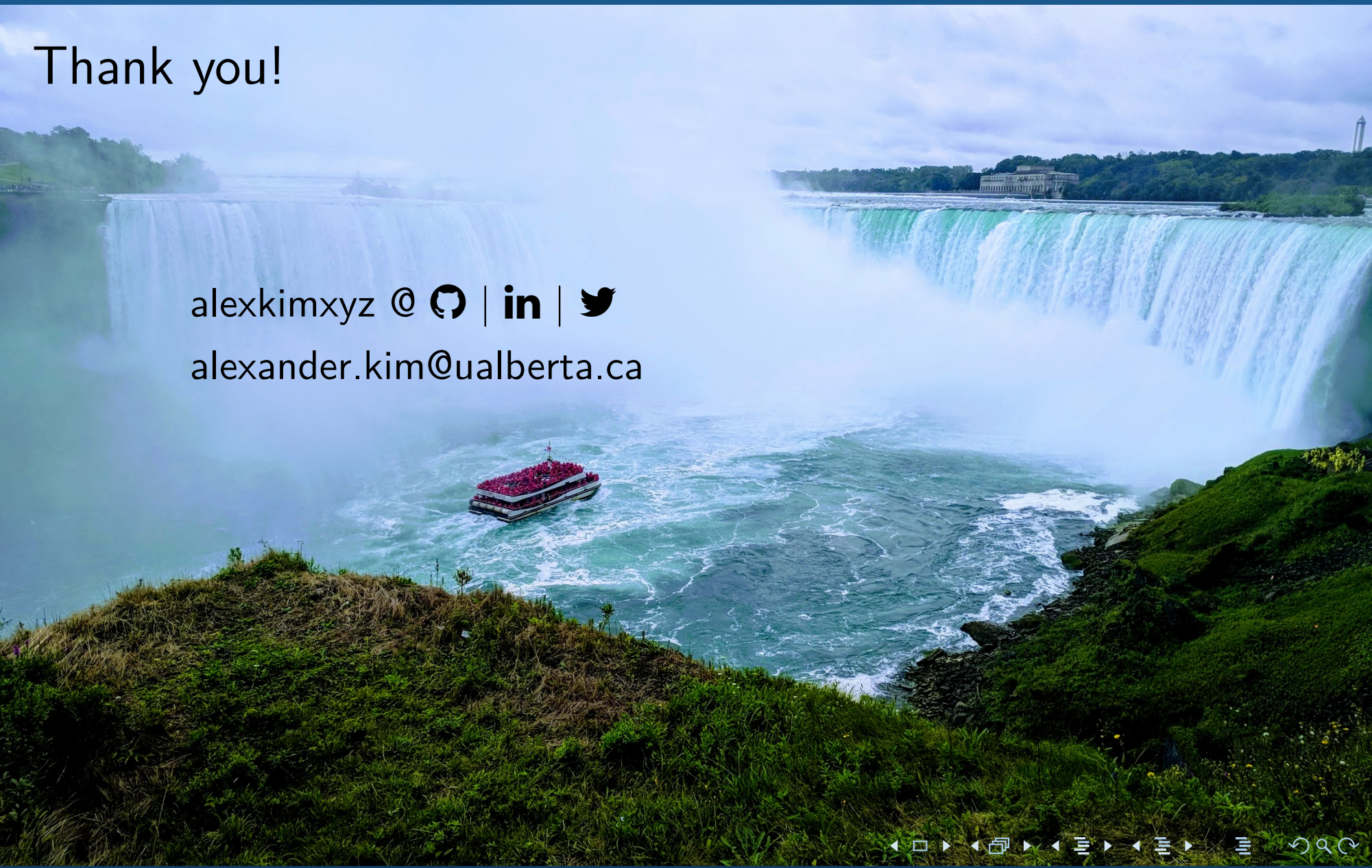
Summary

- Production is an opportunity for learning
- Good monitoring = automated monitoring
- Monitoring will evolve along-side your application: start simple
- Monitoring in phases e.g.:
  1. File logging + simple metrics + dashboards
  2. + logging to data store systems + threshold-based alerting
  3. + ML-based monitoring and alerting
  4. + model decay monitoring

Additional Resources

- Sculley, David, et al. "Hidden technical debt in machine learning systems." Advances in neural information processing systems. 2015.
- Breck, Eric, et al. "What's your ML Test Score? A rubric for ML production systems." (2016).
- Polyzotis, Neoklis, et al. "Data management challenges in production machine learning." Proceedings of the 2017 ACM International Conference on Management of Data. ACM, 2017.

Thank you!



alexkimxyz @ | in |

alexander.kim@ualberta.ca