RSA Conference 2019

San Francisco | March 4–8 | Moscone Center

BETTER.

SESSION ID: HTA-F02

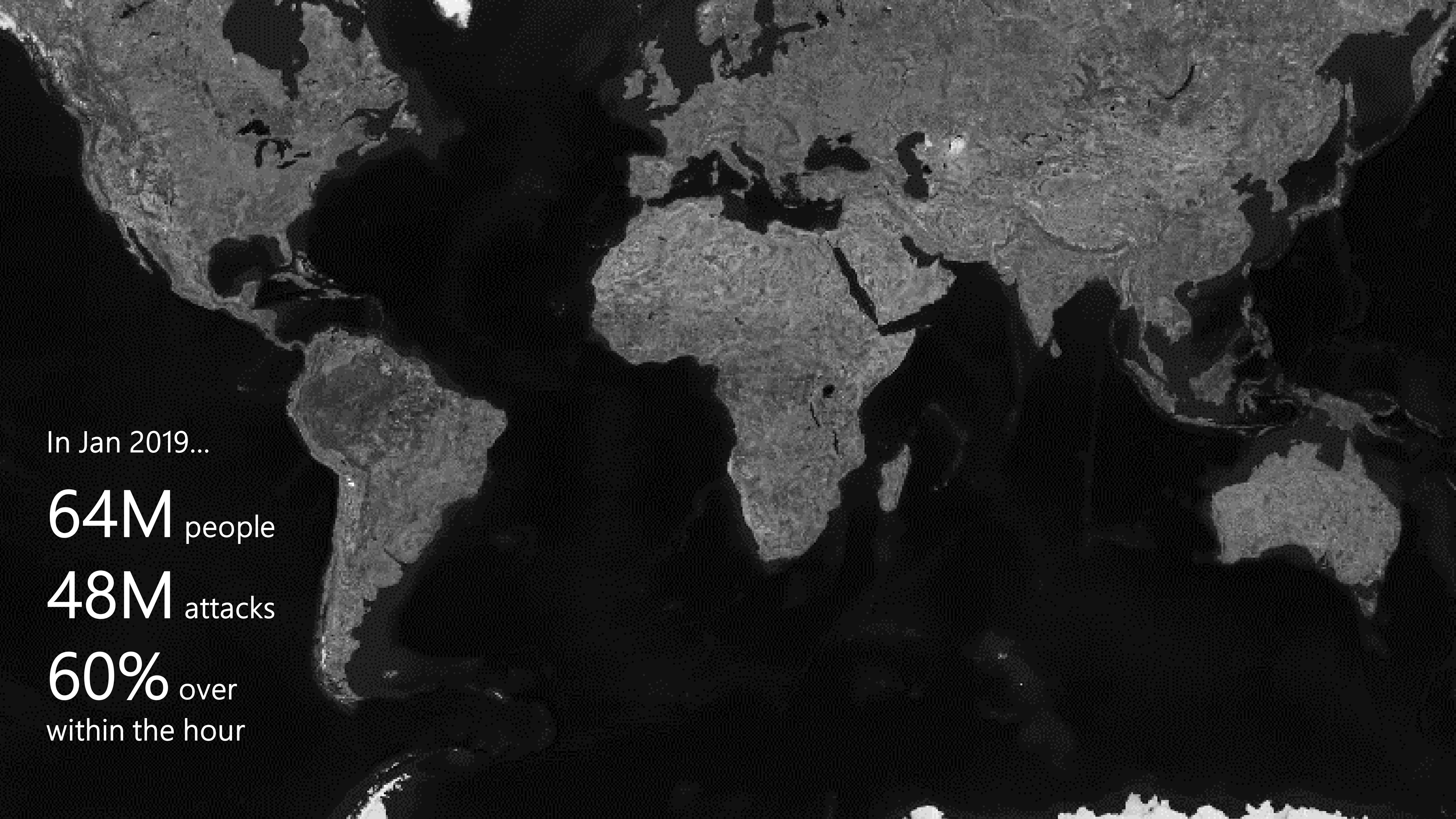
Blackbox Interpretability: Next Frontier in Adversarial ML Evasion

Holly Stewart, Greg Ellison

Microsoft Defender Research Team

Contributions by:
Sam Jenkins, Harsha Nori

#RSAC



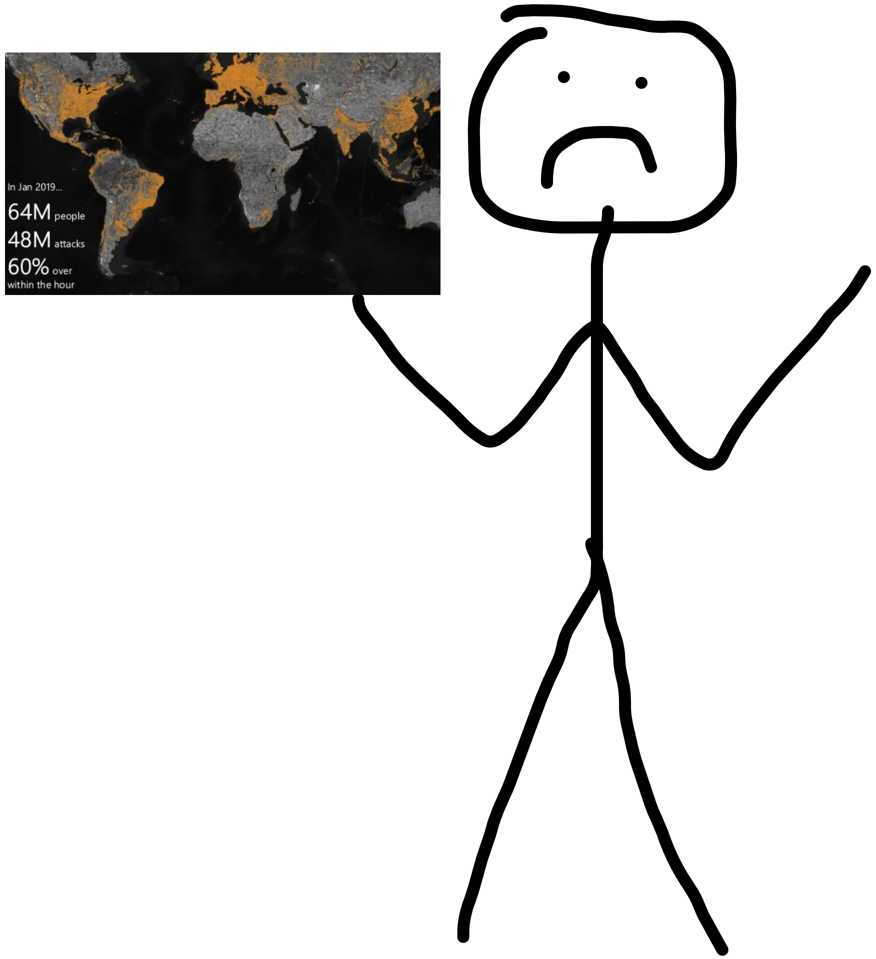
In Jan 2019…

64M people

48M attacks

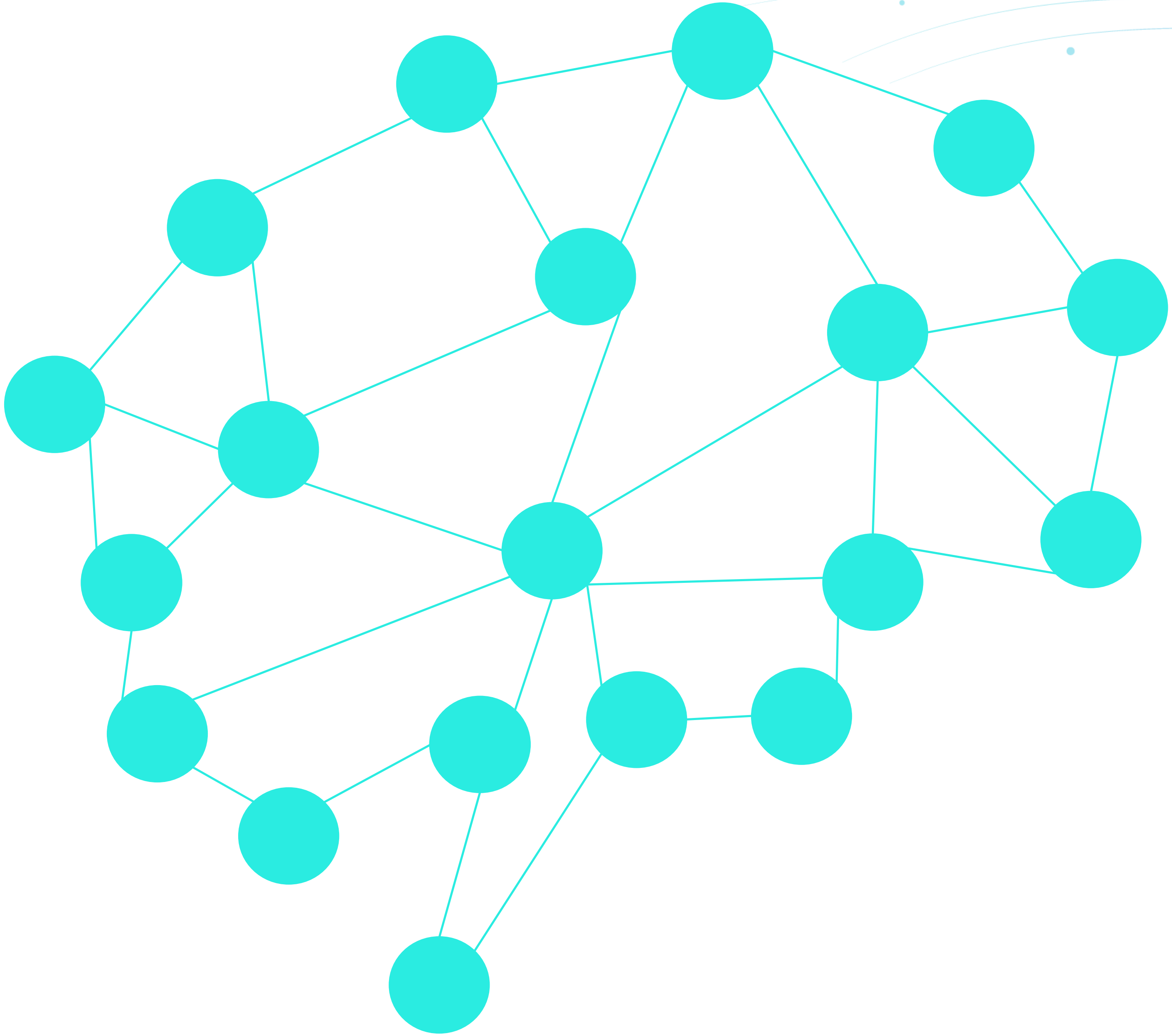
60% over within the hour

How do we address these zero day malware attacks?



Sticky

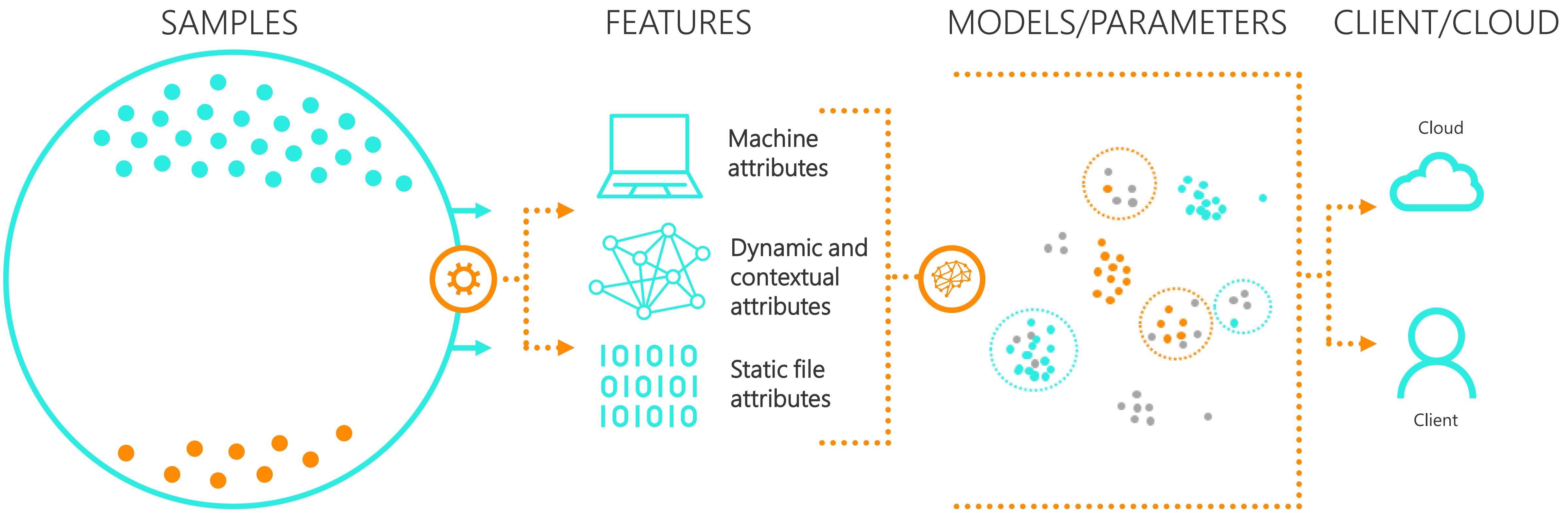
- Use attack surface reduction rules (ex. blocking all docs with macros)
- Detonate malware in an isolated environment
- Block when malware behaves badly on the system
- Use machine learning to predict and block the threat at first sight



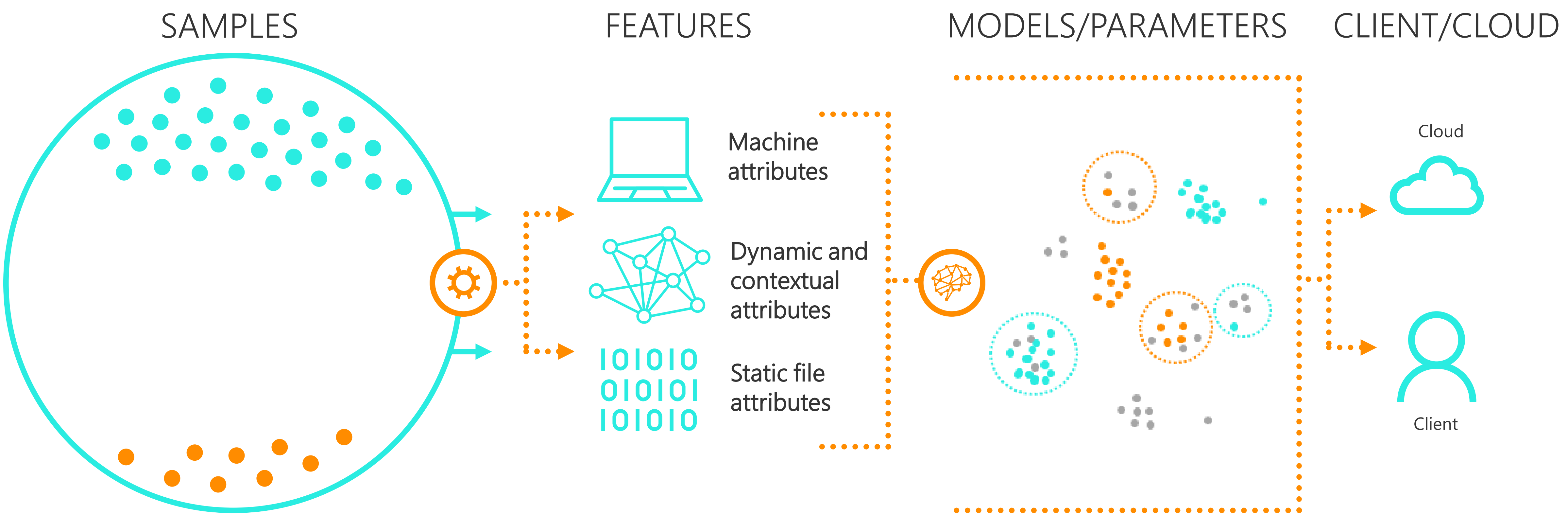
In Jan 2019…

30% threats first blocked through machine learning prediction

Supervised machine learning process

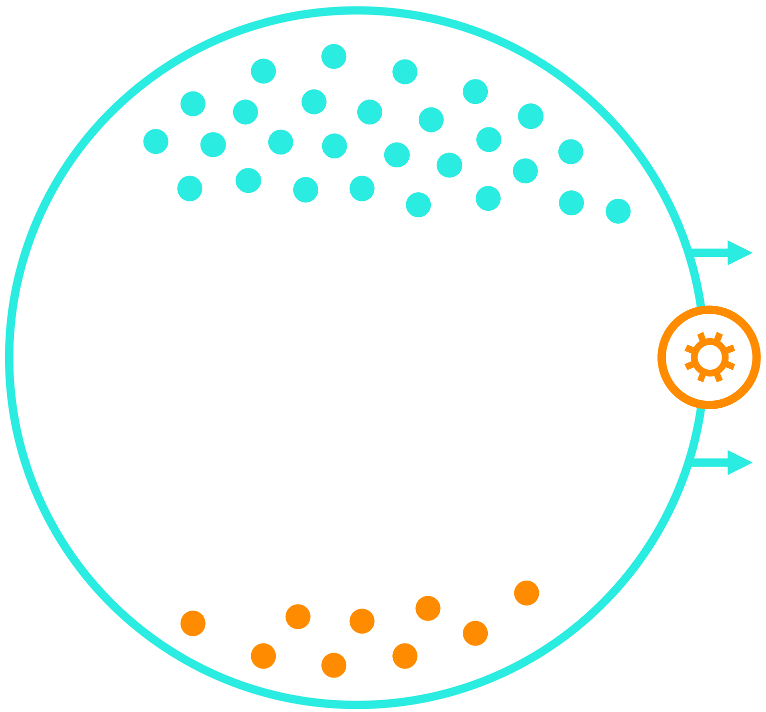


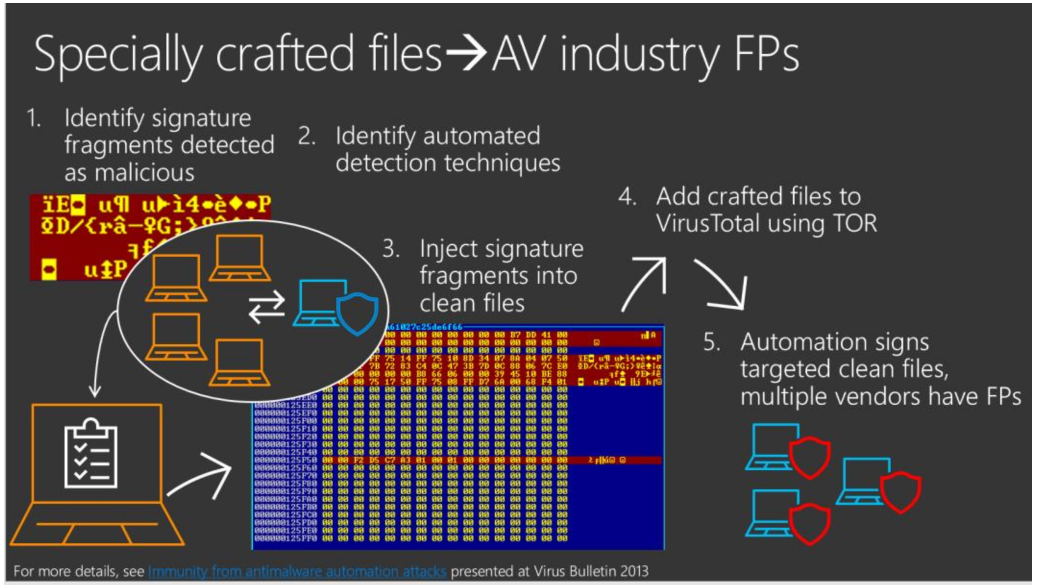
Supervised machine learning threat model

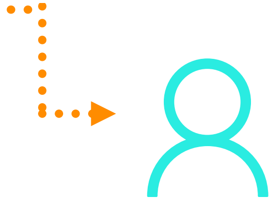


Supervised machine learning threat model

SAMPLES



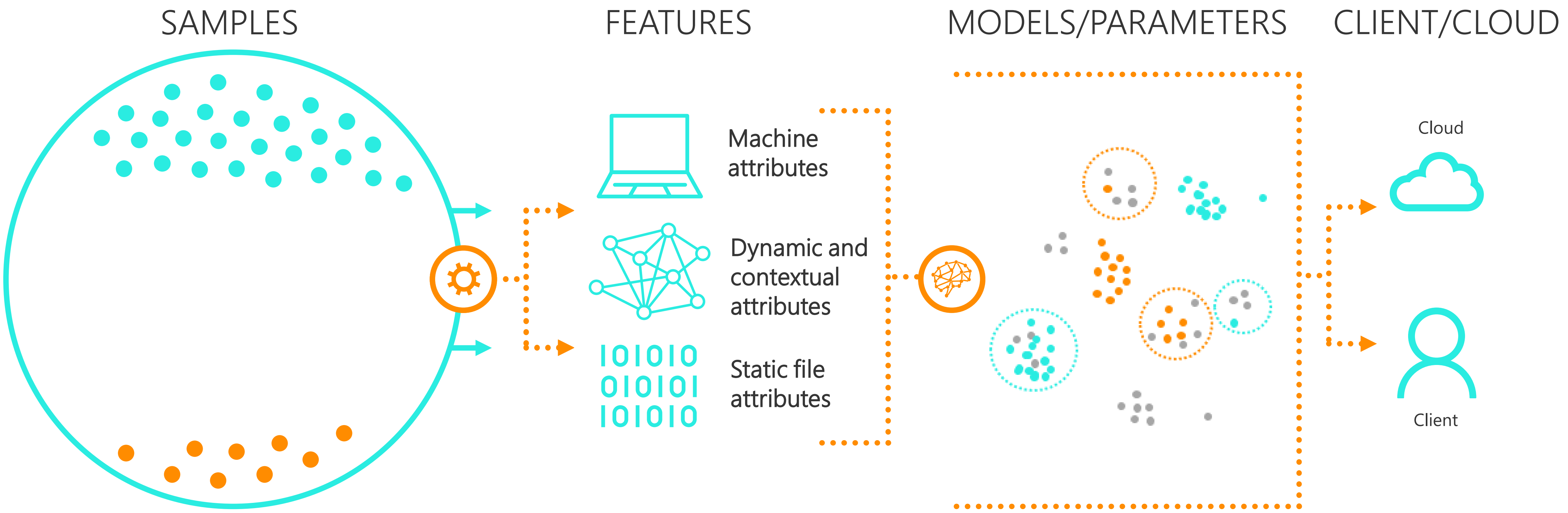




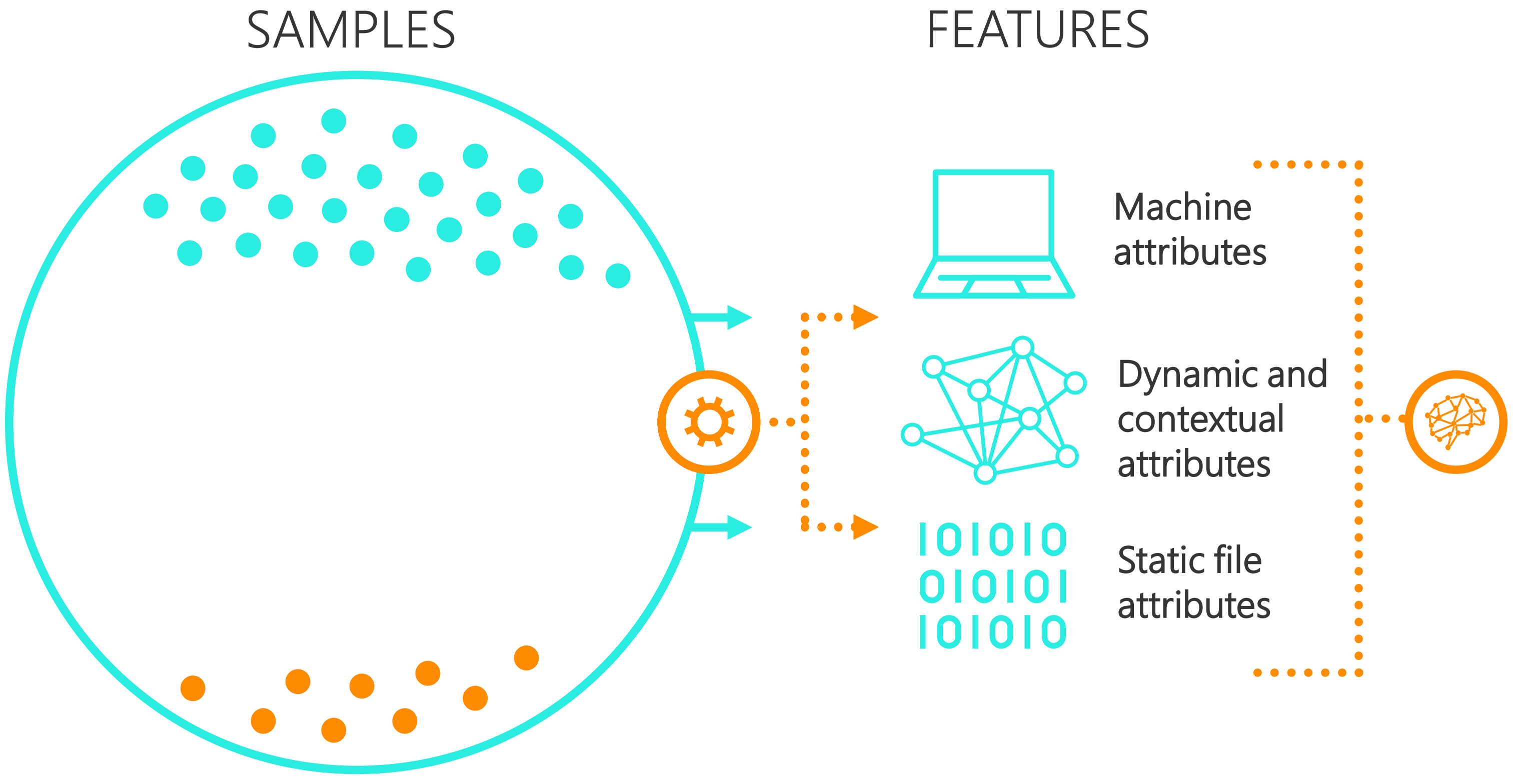
Client

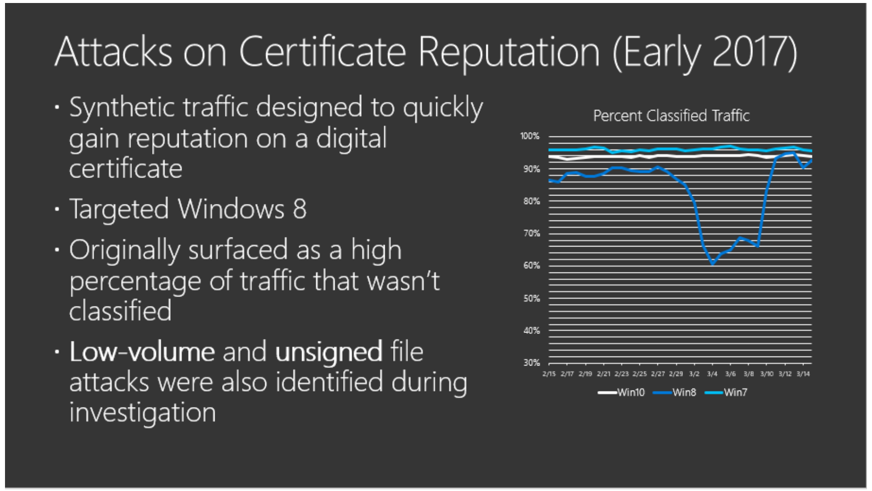
* More info on the blog:
https://aka.ms/hardening-ML
and **Black Hat USA**

Supervised machine learning threat model



Supervised machine learning threat model

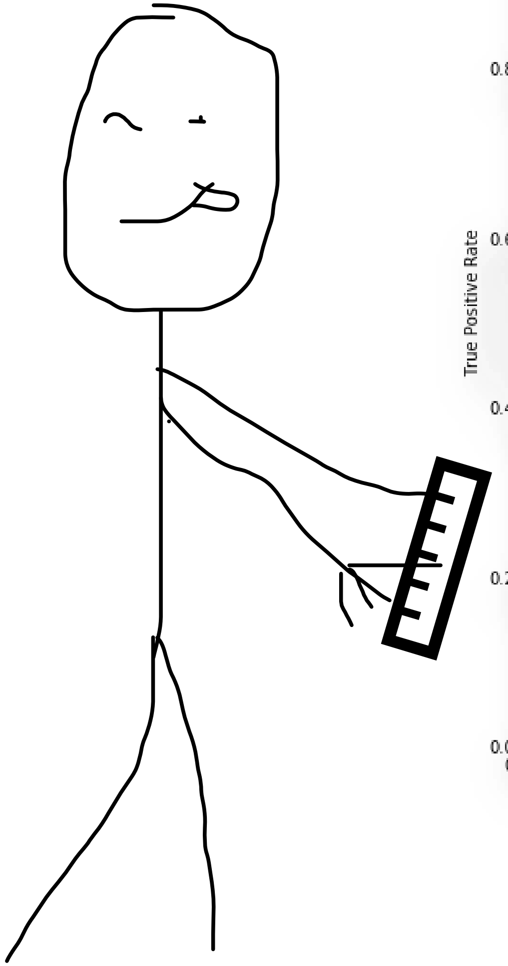


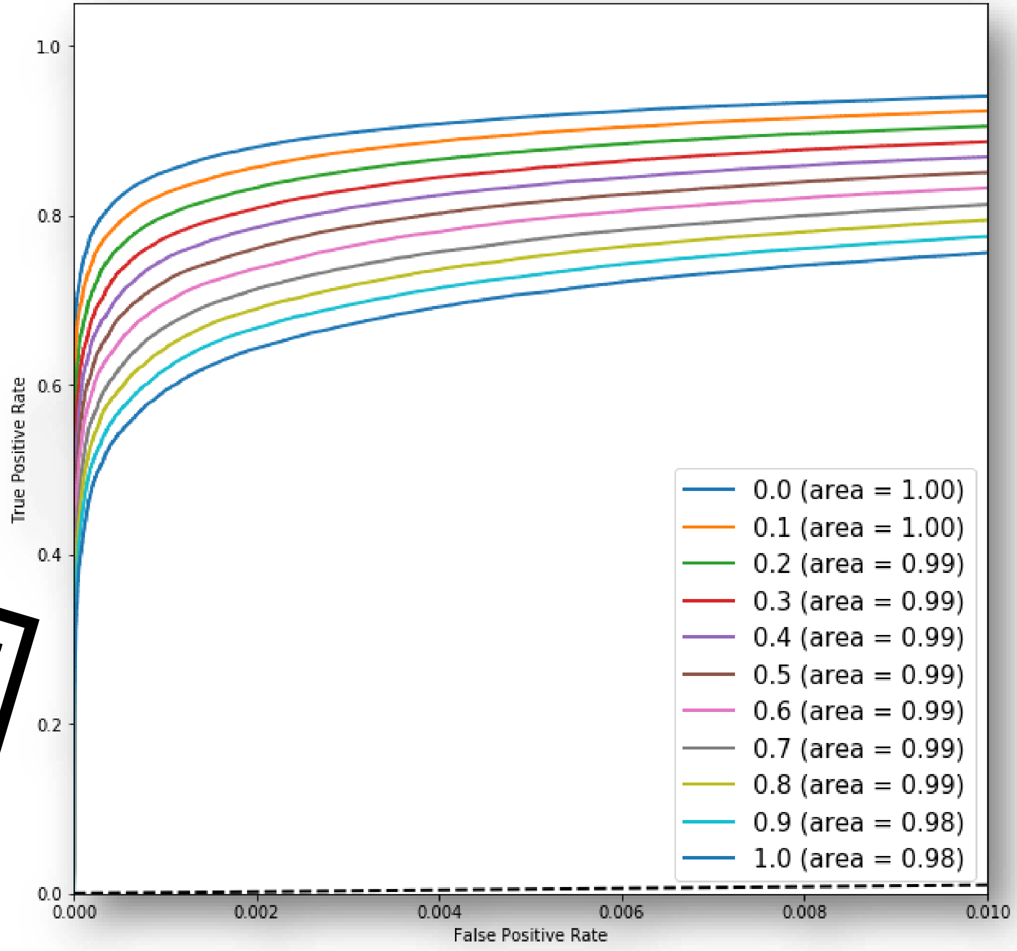


* More info on the blog: https://aka.ms/hardening-ML and **Black Hat USA**

Challenges with machine learning

False Positives





Challenges with machine learning

False Positives

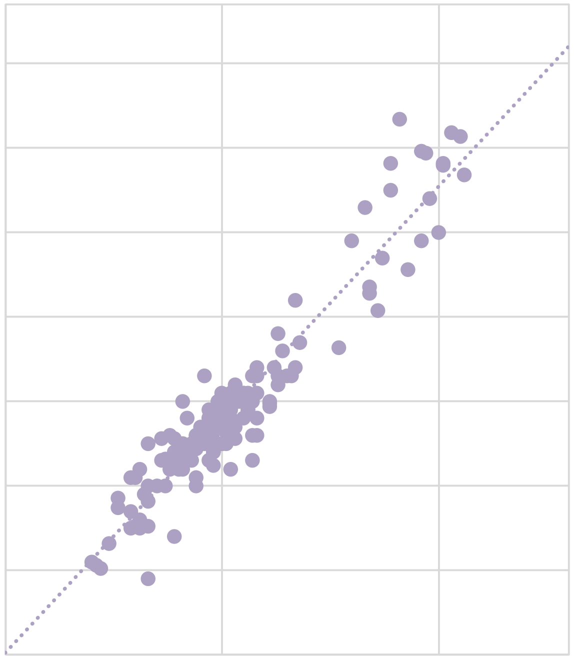
Compute

Linear Model

Computationally fast

Simple structure

Less precise predictions

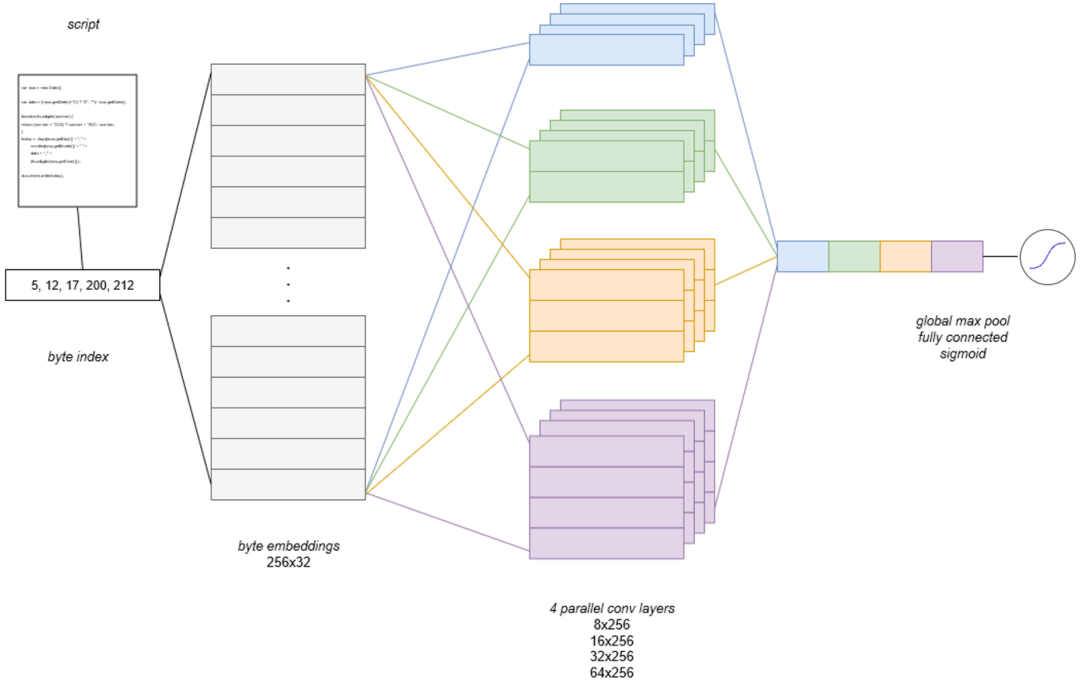


Deep Learning Model

Computationally slow

Complex structure

Better predictive performance



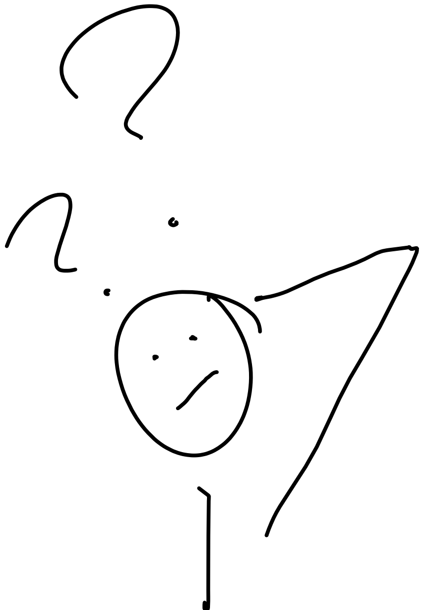
Challenges with machine learning

False Positives

Compute

Interpretability

This black box is intentional.

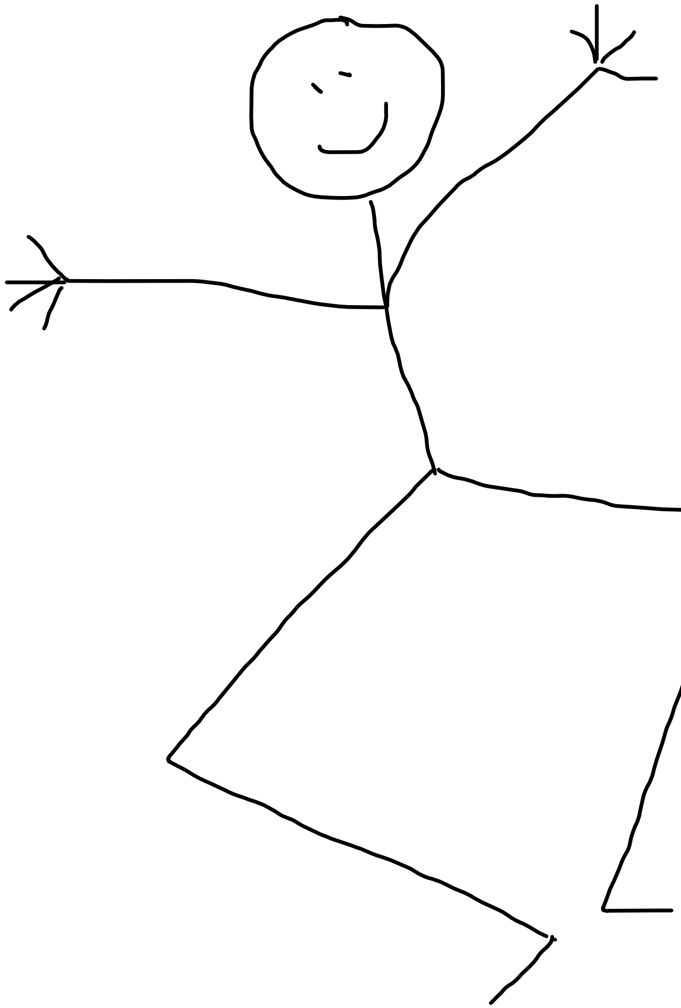


Enter black box interpretability methods

(Many) work with any model

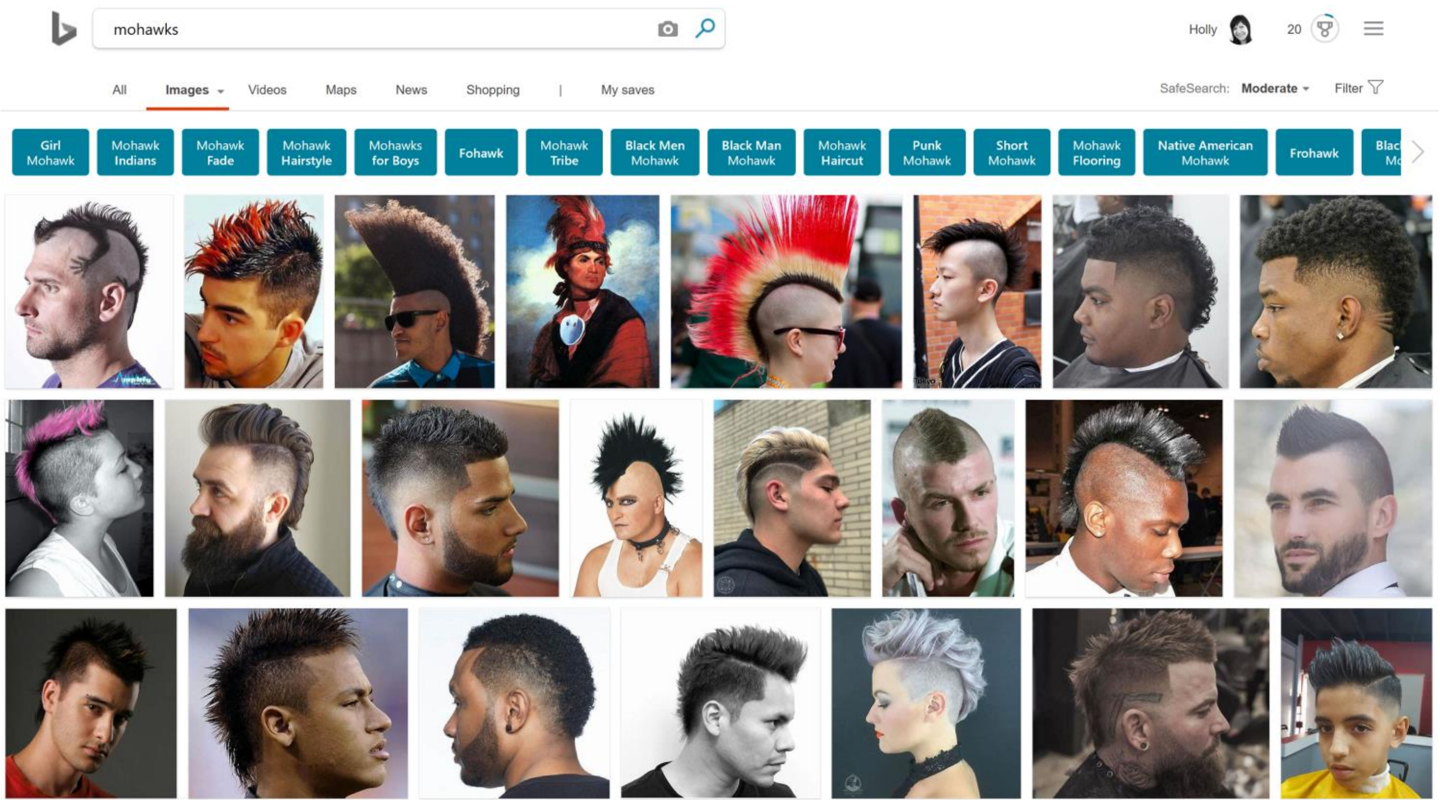
Help explain model decisions

Publicly available libraries



Benefits of black box interpretability

Missing signals
(feature evasion)

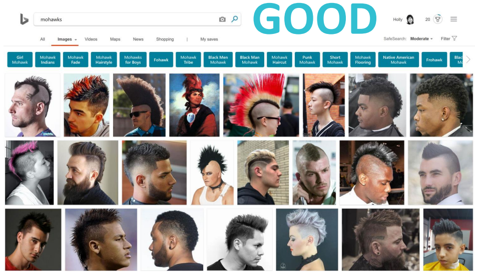


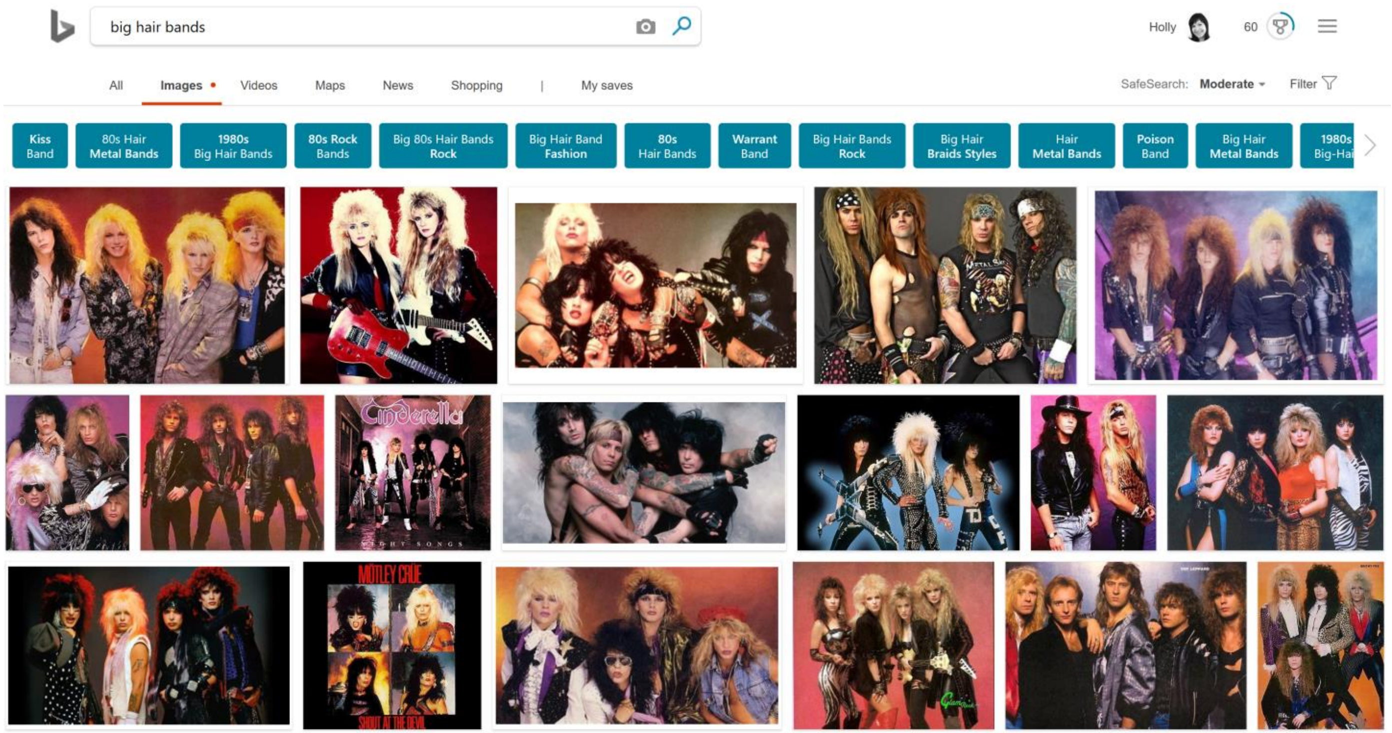
Mohawk = GOOD

Benefits of black box interpretability

Missing signals
(feature evasion)

GOOD



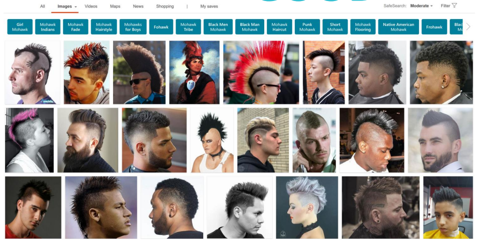


Big hair = BAD

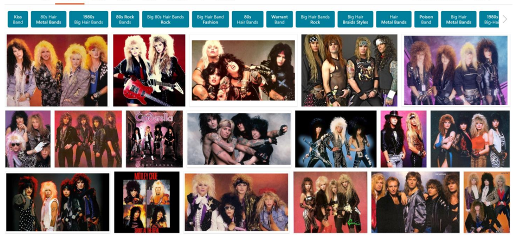
Benefits of black box interpretability

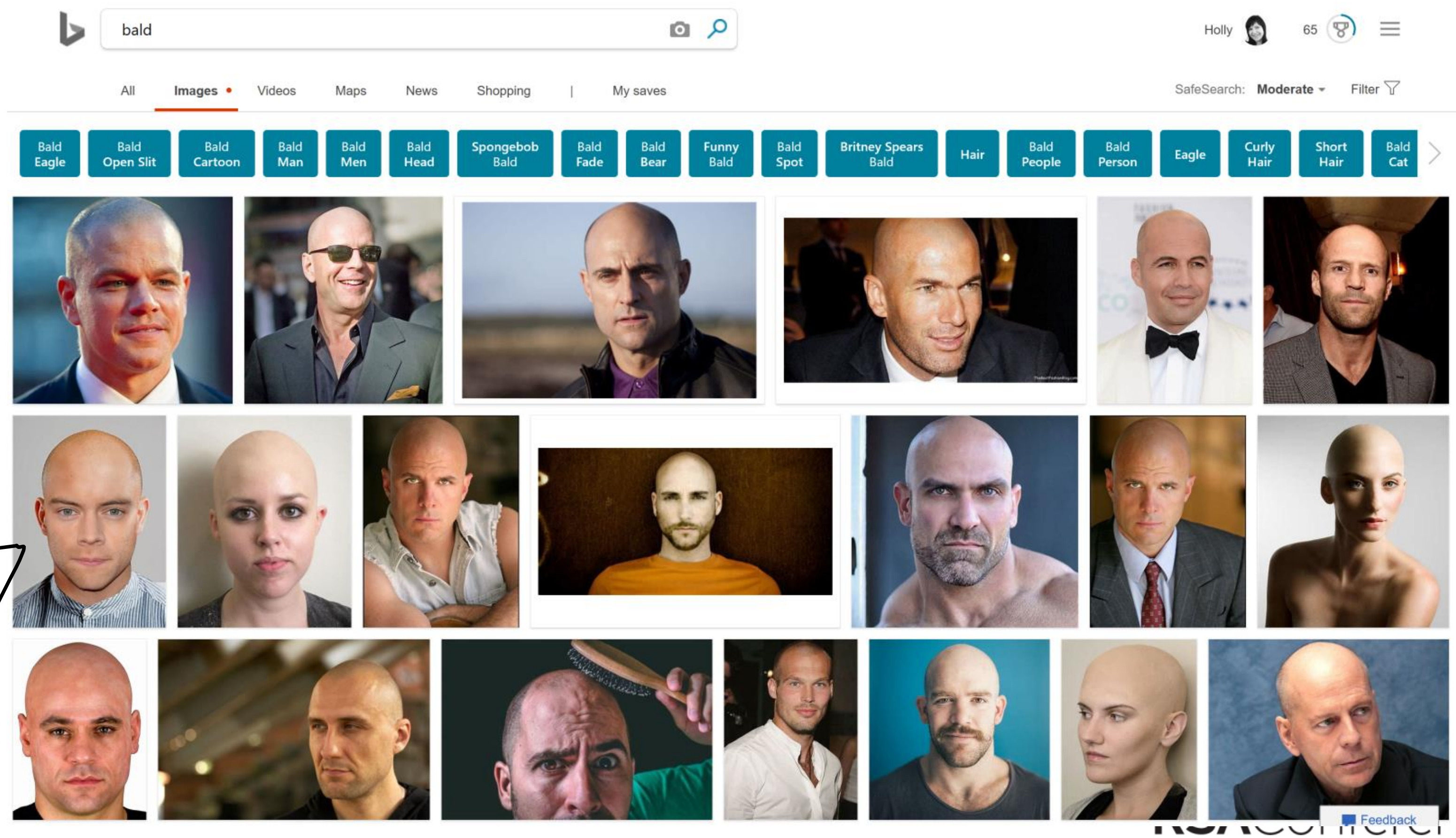
Missing signals
(feature evasion)

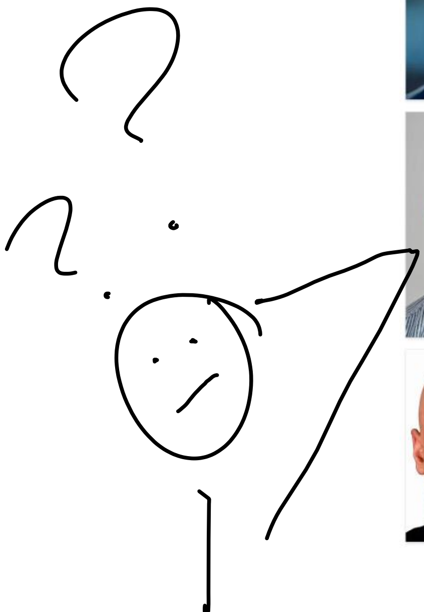
GOOD



BAD



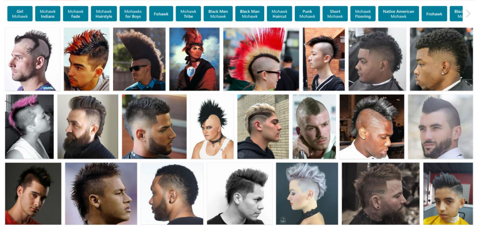




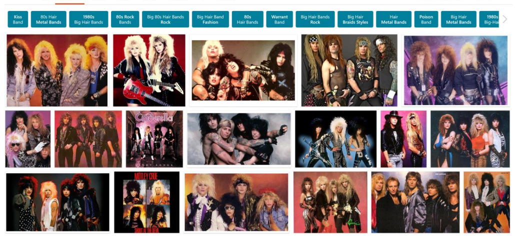
Benefits of black box interpretability

Missing signals
(feature evasion)

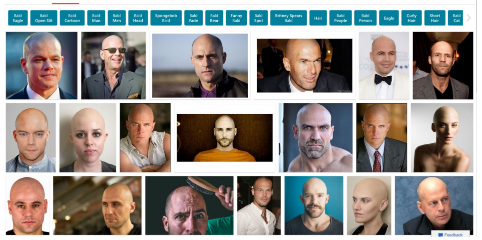
GOOD



BAD



??



Misleading signals
(ambiguous features)

How about fauxhawks…

??

← **Mohawk… GOOD**

← **Big hair… BAD**

Model improvement using black box interpretability

Missing signals → Add new features
(feature evasion)

Misleading signals → Remove misleading features
(ambiguous features)

RSAConference2019

Black Box Interpretability Methods

Notes on terminology

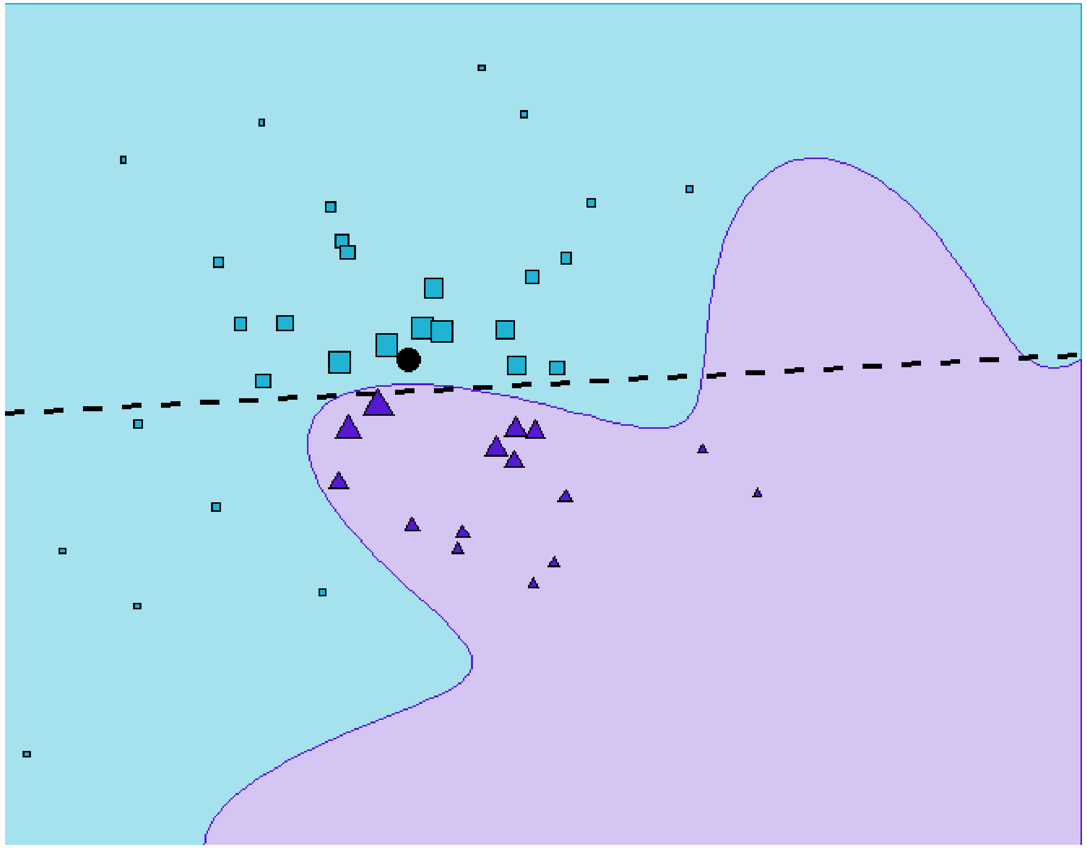
- We're considering binary malware classifiers
  - Decide if a file is malware (1) or clean (0)
- An instance is a single classifier decision (file, in this case)
- FP: False Positive – a clean file the classifier thinks is malware
- FN: False Negative – a malware file the classifier misses

Black box interpretability methods

- Current focus of publicly-available methods:
  - Instance level model decisions
- Two methodologies:
  - LIME
  - SHAP

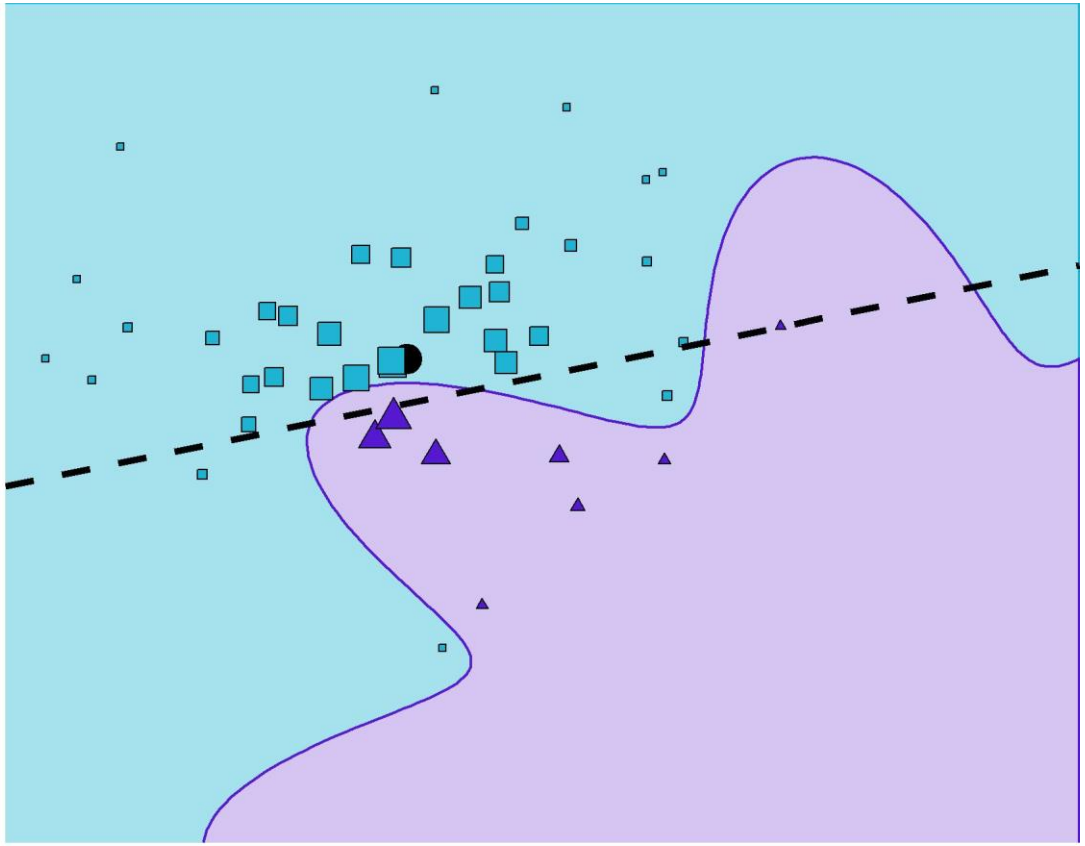
LIME

- Introduced in 2016 by Marco Ribero [1]
- Locally Interpretable Model-Agnostic Explanations



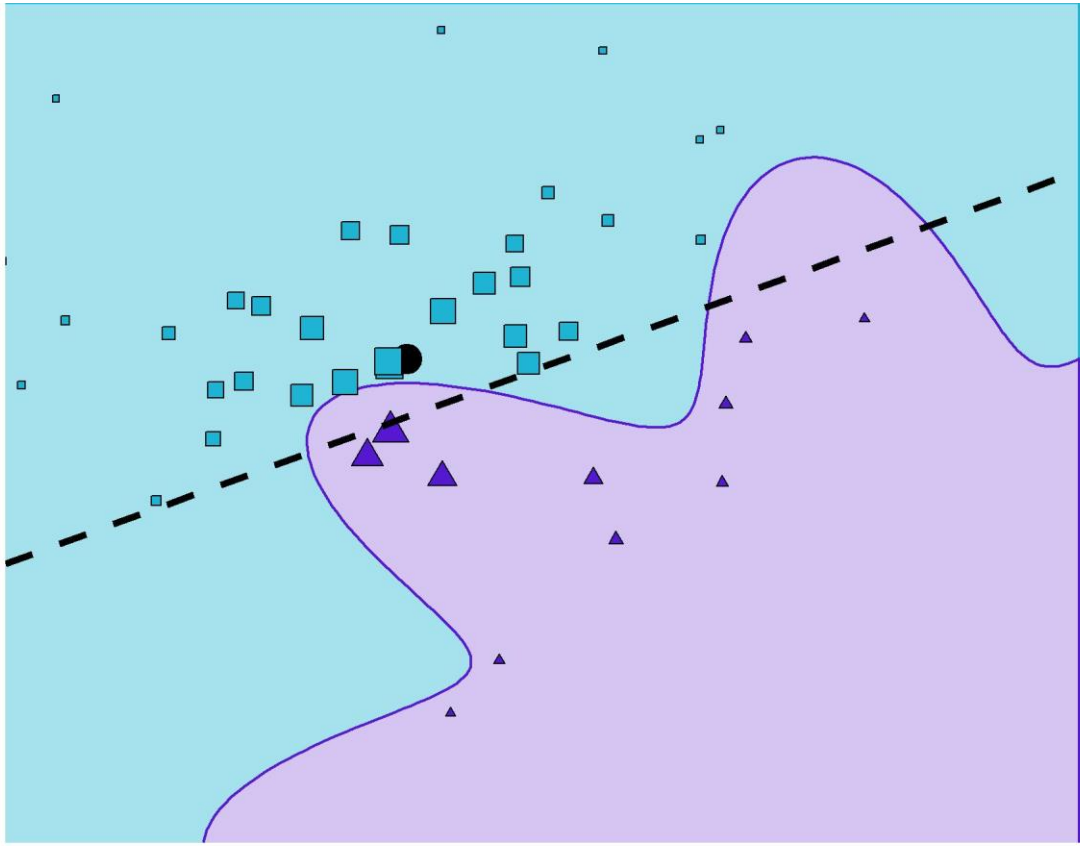
LIME

- LIME samples space around instance and fits a local linear (thus, interpretable) model
- The model weights the samples by distance from instance
- The local model is fit using K-LASSO



SHAP

- Introduced in 2017 By Scott Lundberg [2]
- Differs from LIME in weighting sample distance from *x'*, and estimating *f'(x)*



Implementation details

- LIME & SHAP frameworks are not computationally cheap
  - Focus on known FPs & FNs

- SHAP & LIME open source python libraries
  - https://github.com/marcotcr/lime
  - https://github.com/slundberg/shap

Combining multiple methods

- These methods are approximations!
- Unlike reading the coefficients of a linear model, we're estimating model behavior
- So, we check to see that results of LIME and SHAP are consistent

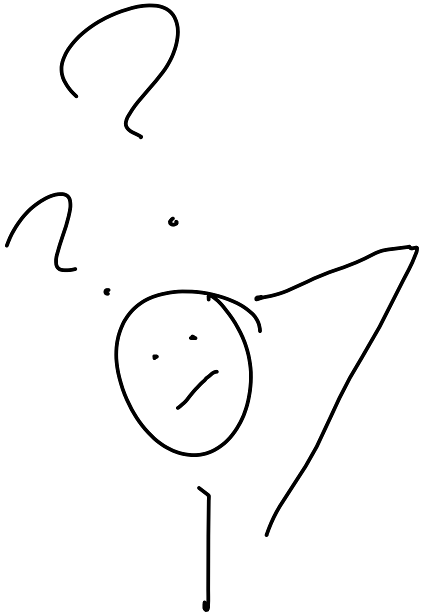
Black box interpretability methods

- LIME & SHAP both result in a feature contribution score, *s*
  - *s* > 0: the feature pushes the classifier toward malware
  - *s* < 0: the feature pushes the classifier toward clean
  - The closer *s* is to 0, the weaker the feature's contribution

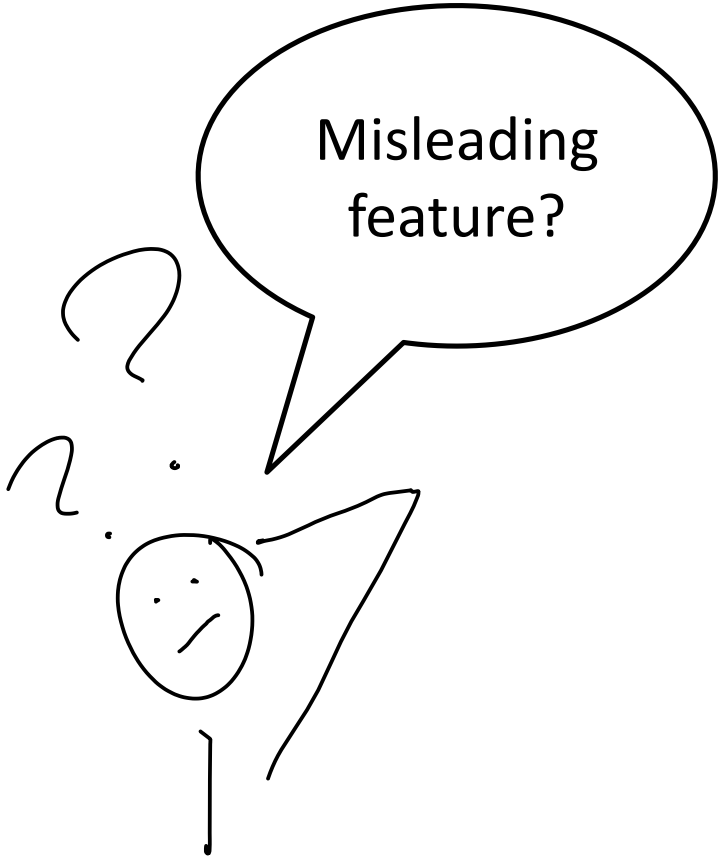
Example: Per instance interpretability

Cloud classifier false positive (FP)

Feature	LIME score	SHAP score
Emulation Event 1	0.22	0.03
Emulation Event 2	0.27	0.18
File Metadata Feature	0.22	0.07
Other Features	…	…



Example: Per instance interpretability



Cloud classifier false positive (FP)

Feature	LIME score	SHAP score
Emulation Event 1	0.22	0.03
Emulation Event 2	**0.27**	**0.18**
File Metadata Feature	0.22	0.07
Other Features	…	…

Aggregated interpretability

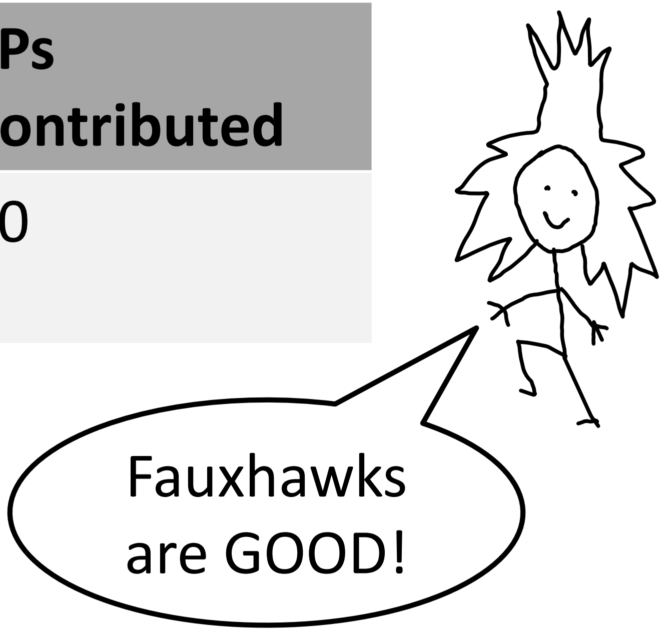
Should we remove this misleading feature to avoid FPs?

Not necessarily… this is only one instance – let’s aggregate!

Example: Emulation Event 2

Feature	Avg LIME score	Avg SHAP score	FPs Contributed	TPs Contributed
Emulation Event 2	0.115	0.114	1	30

Looks like this is a good feature!



Identifying misleading features

Example: Emulation Event 2

Feature	Avg LIME score	Avg SHAP score	FPs Contributed	TPs Contributed
Emulation Event Feature 2	0.115	0.114	1	30

We can extend this analysis to discover badly behaved features:

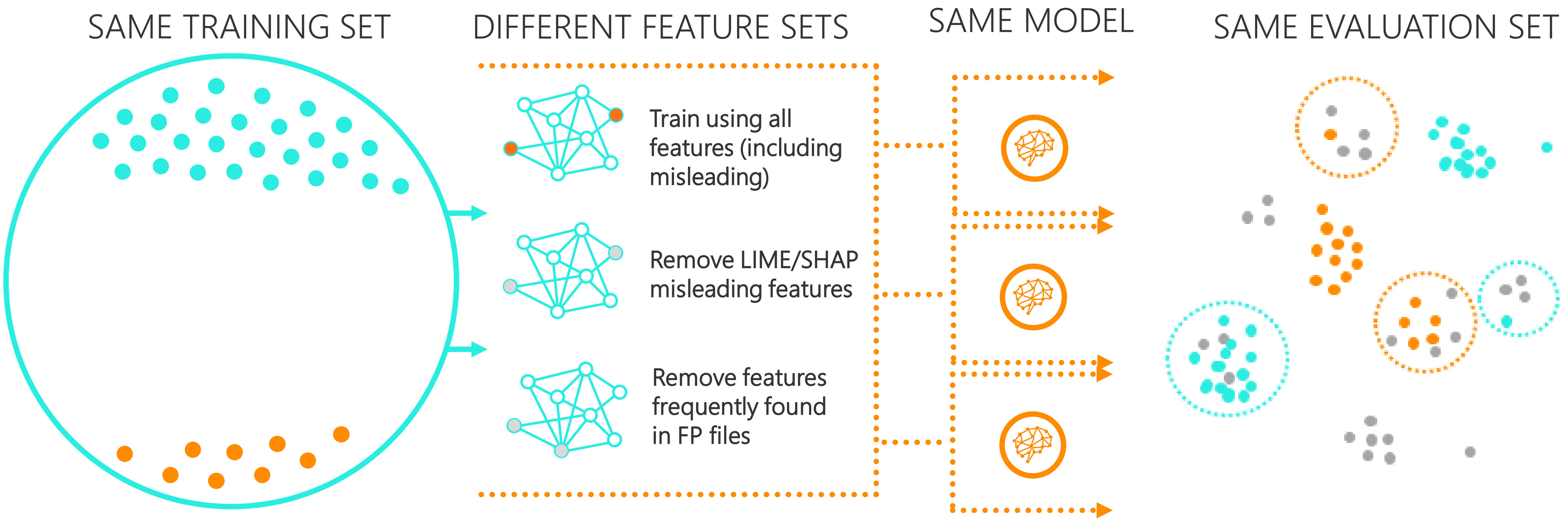
Example: Emulation Event 2

Feature	Avg LIME score	Avg SHAP score	FPs Contributed	TPs Contributed
Candidate 1	0.356	0.388	4	2
Candidate 2	0.943	0.874	4	0
Candidate 3	0.053	0.081	2	28
…				

Resolving misleading features

- The obvious solution is to exclude misleading features from the model entirely

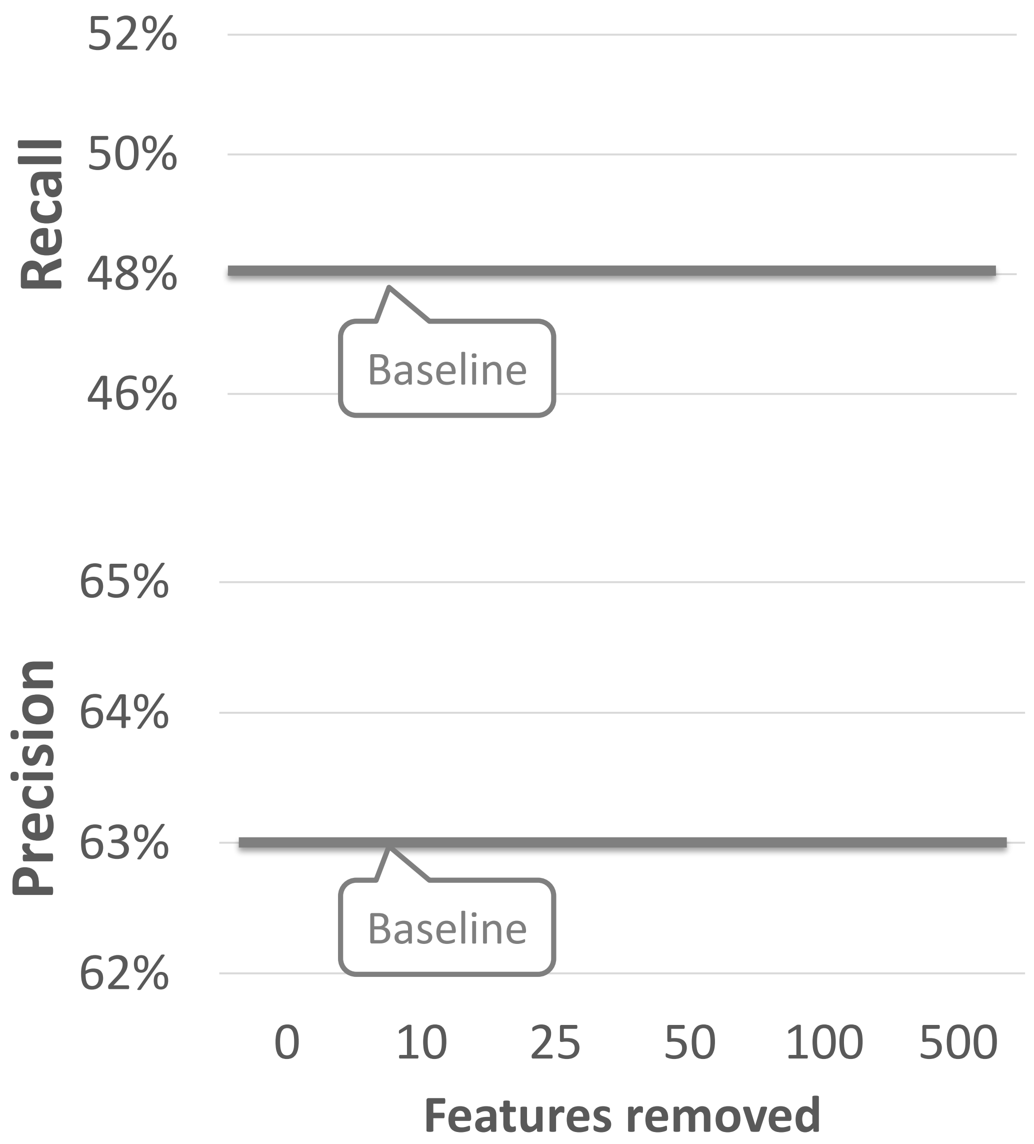
Experiment design: Excluding misleading features



Results: Excluding misleading features

Measurements

- **Recall** - % malware detected
- **Precision** - % actual malware/classified malware



Results: Excluding misleading features

Measurements

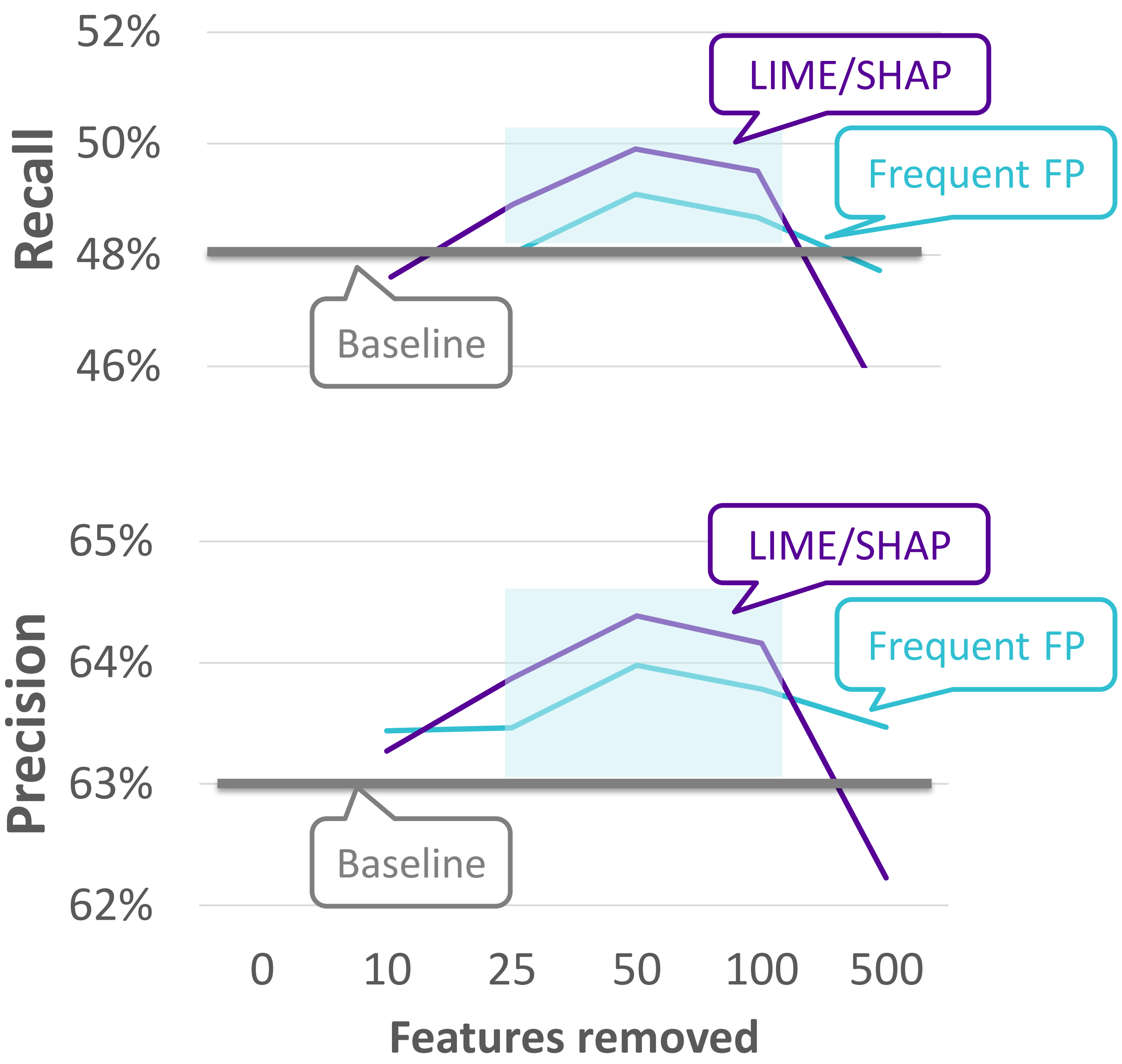
Recall - % malware detected

Precision - % actual malware/classified malware

Both techniques beat the baseline

Sweet spot: 50 features removed (out of 200k)

LIME/SHAP best model



Detecting feature evasion

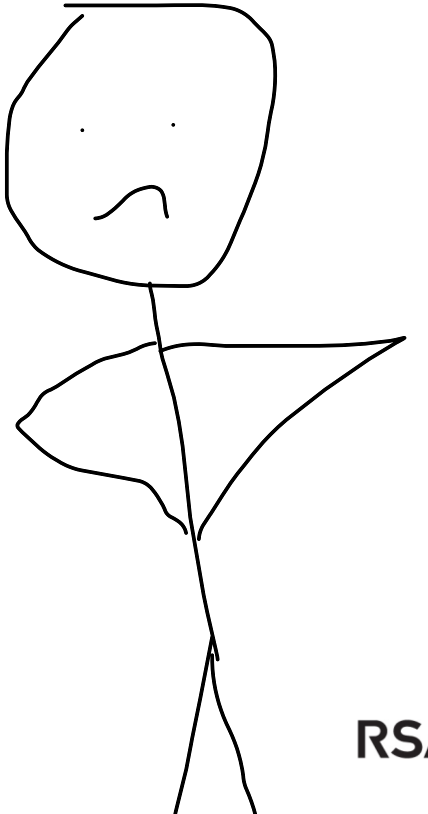
Reminder: LIME and SHAP feature contribution score, *s*

s > 0: the feature pushes the classifier toward malware

s < 0: the feature pushes the classifier toward clean

Example: Cloud Classifier FN

Feature	LIME score	SHAP score
Generic File Metadata 1	-0.002	0.028
Generic File Metadata 2	-0.018	-0.060
…		
File Content Feature 1	-0.002	0
File Content Feature 2	0.008	0

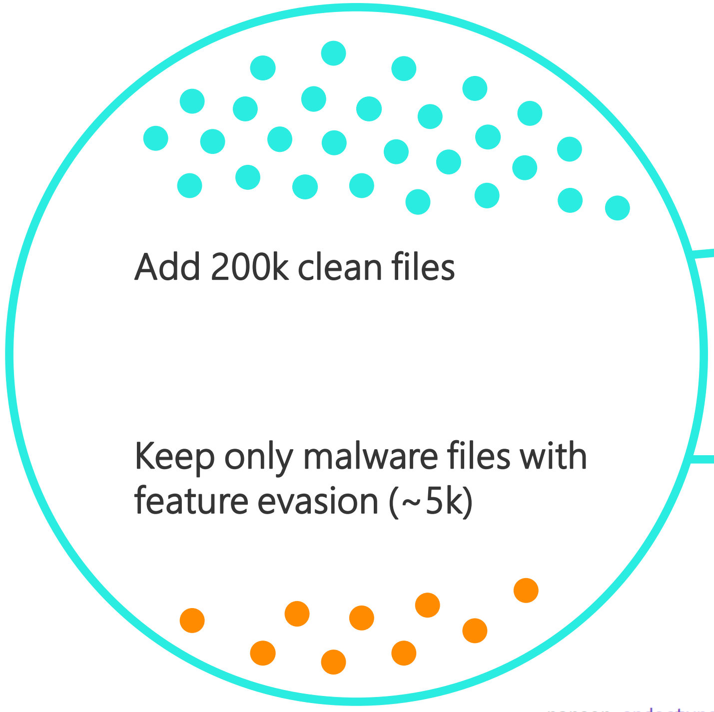


Automating feature generation

- What to do about these feature evaded files?
- Generate signal!
  - Collect files with feature evasion
  - Mine the file content for malware signals
- Target new feature development at previously unclassifiable malware

Automated feature mining (JavaScript files)

FILE SELECTION PROCESS



FEATURIZATION

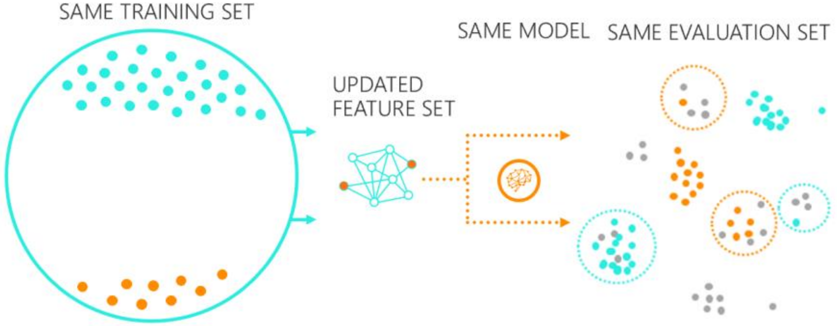
Parse text to create new n-gram features

IDENTIFY STRONG FEATURES

Feature	Score
File Content Feature 1	0.64
File Content Feature 2	0.33

TRAIN LINEAR MODEL

RETRAIN MODEL WITH NEW FEATURES




```
parser.ondoctype = dt => {
  appendChild(
    openStack[openStack.length - 1],
    parseDocType(currentDocument, `<!doctype ${dt}>`)
  );

  const entityMatcher = /<!ENTITY ([^ ]+) "([^"]+)">/g;
  let result;
  while ((result = entityMatcher.exec(dt))) {
    const [, name, value] = result;
    if (!(name in parser.ENTITIES)) {
      parser.ENTITIES[name] = value;
```

```
"nstack"
"tack.l"
"ck.len"
".lengt"
"ength "
"gth - "
"h - 1]"
"- 1],p"
"1],par"
",parse"
"arsedo"
"sedoct"
"doctyp"
"ctype("
"ype(cu"
```

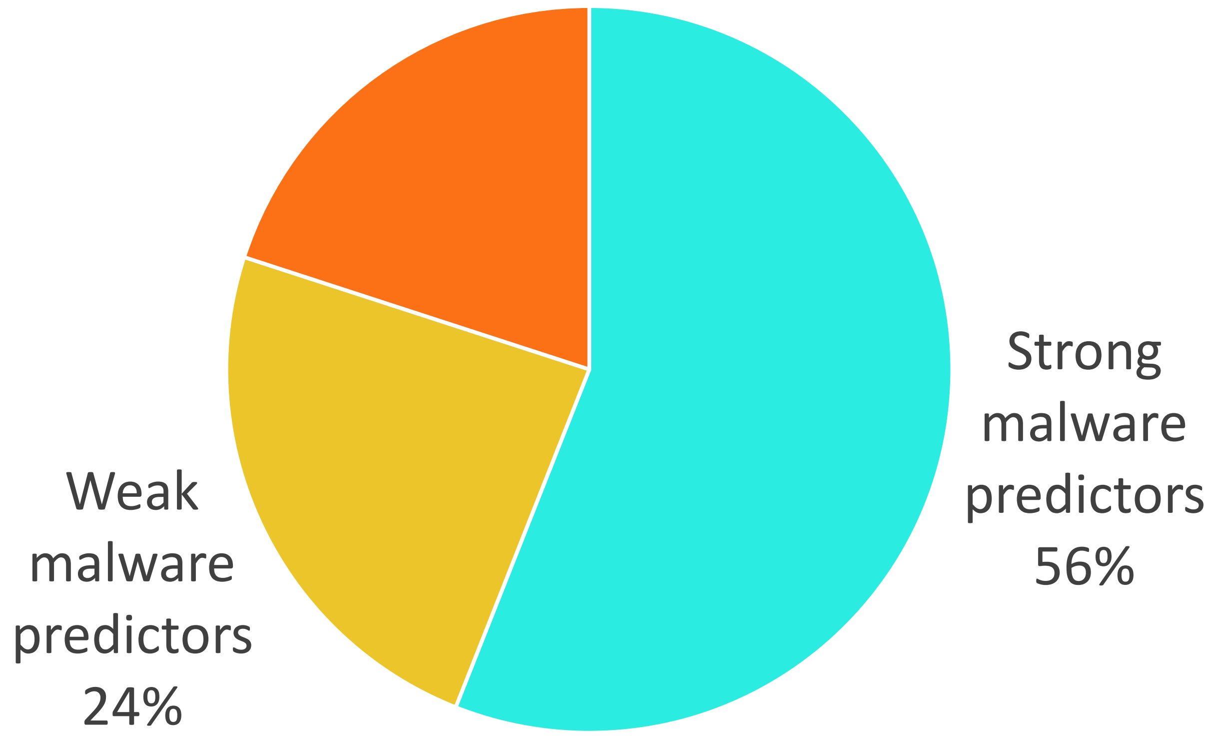
Black Box Interpretability Results

Impact and results (JavaScript)

Did any of the new features matter?

YES

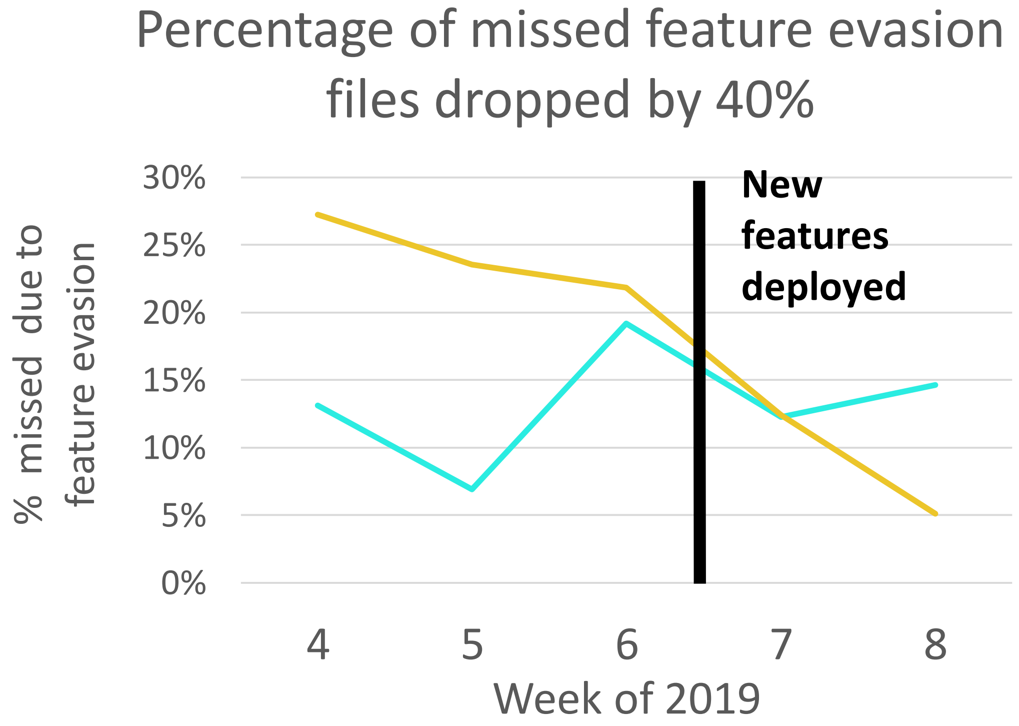
Malware prediction from new features



Impact and results (JavaScript)

Did any of the new features matter? YES

Did the features result in a reduction of feature-evasion misses? YES



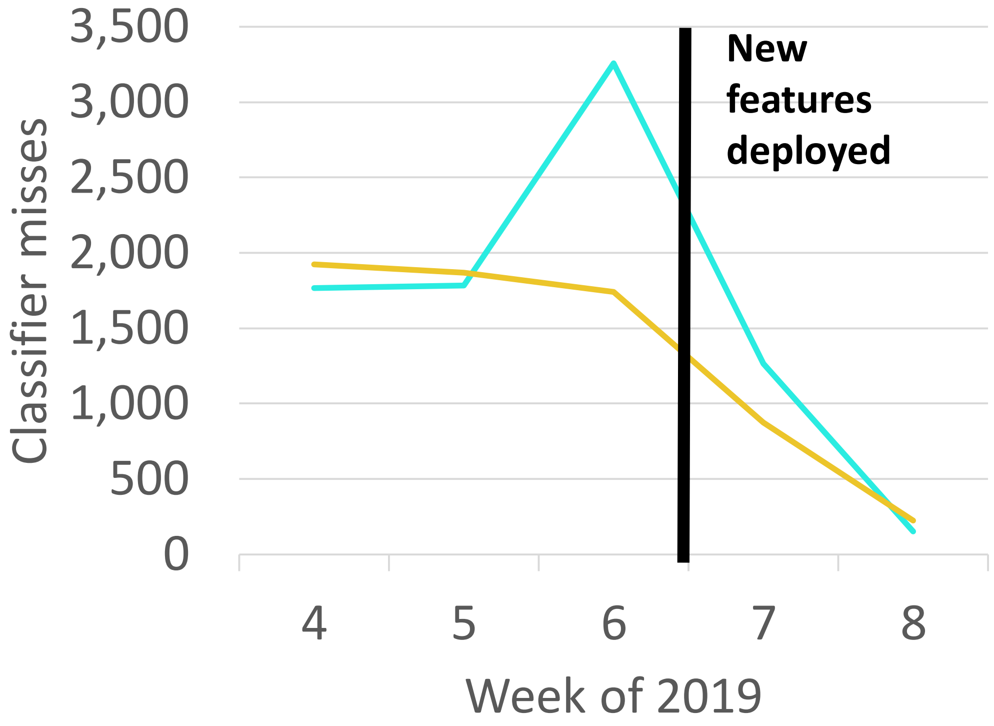
Impact and results (JavaScript)

Did any of the new features matter? YES

Did the features result in a reduction of feature-evasion misses? YES

Did our classifiers miss less malware? YES

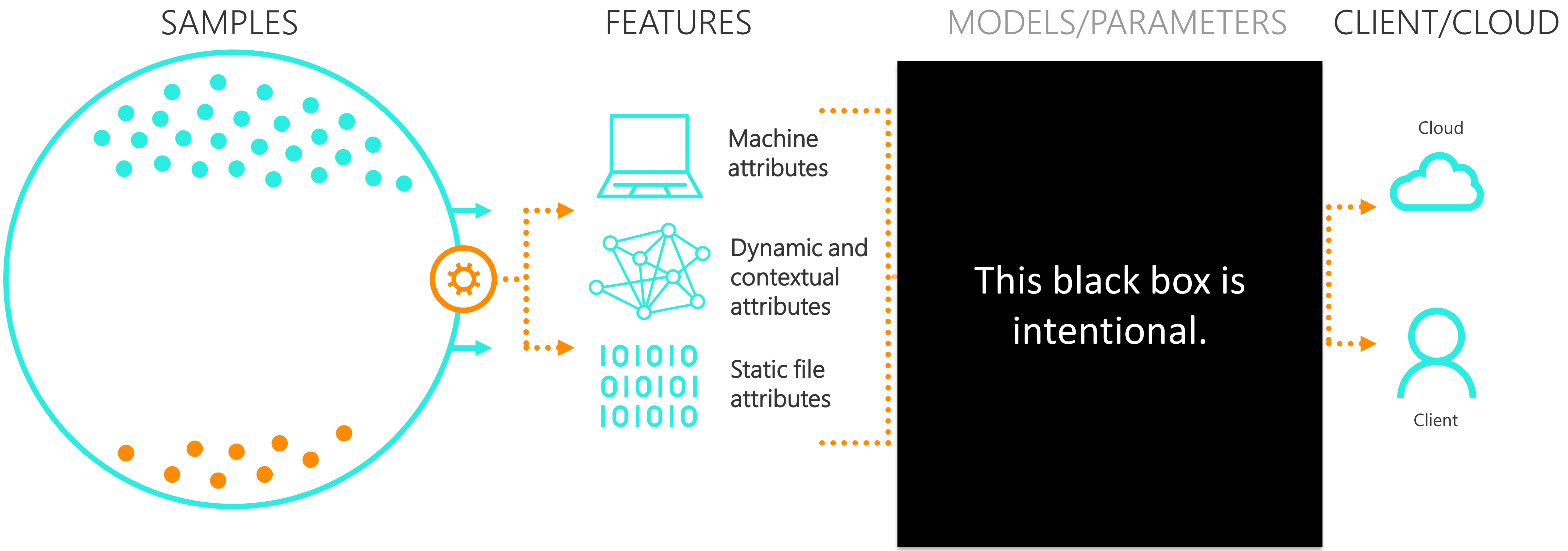
Total number of average daily classifier JavaScript misses declined

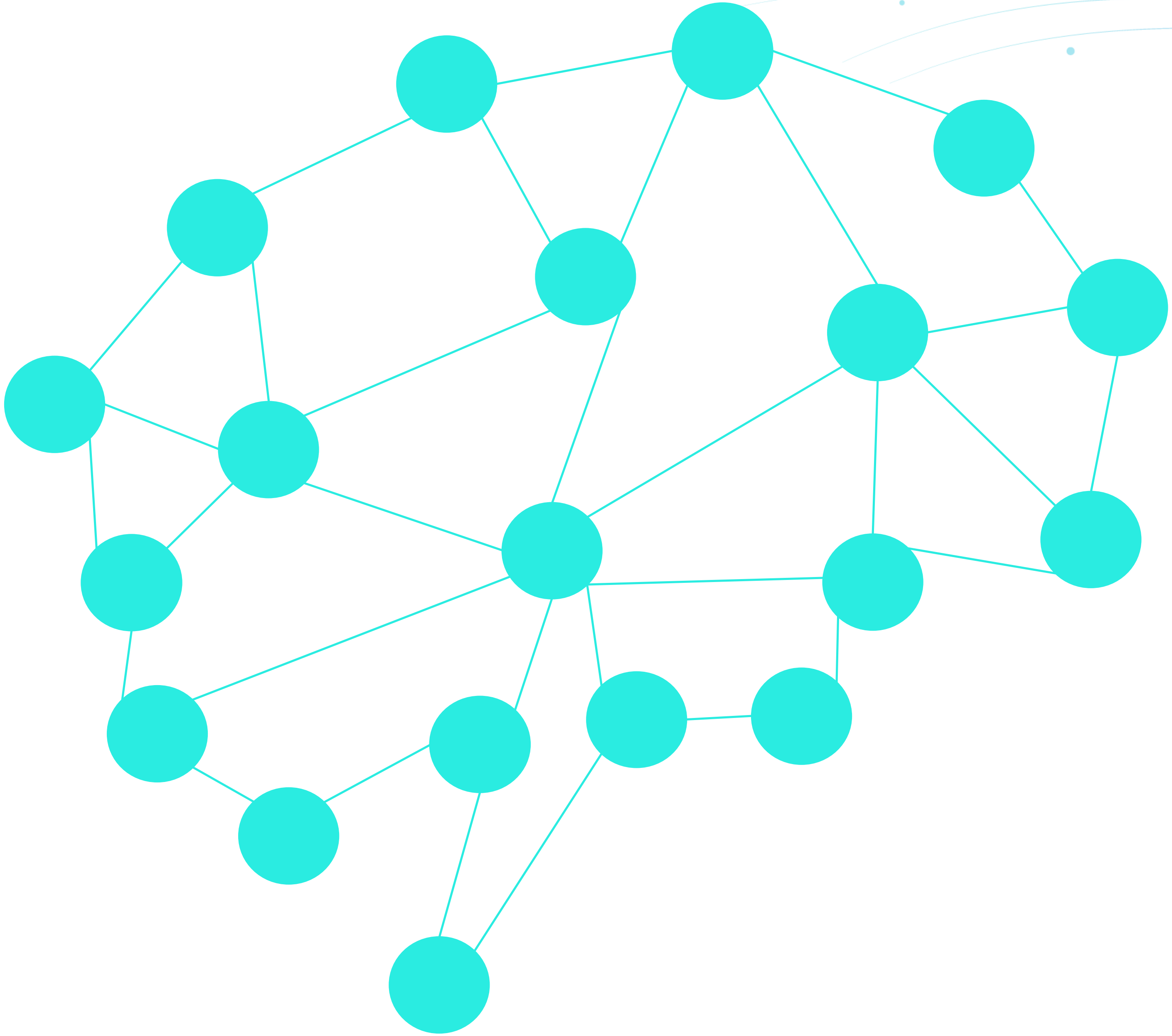


Adversaries can use it, too

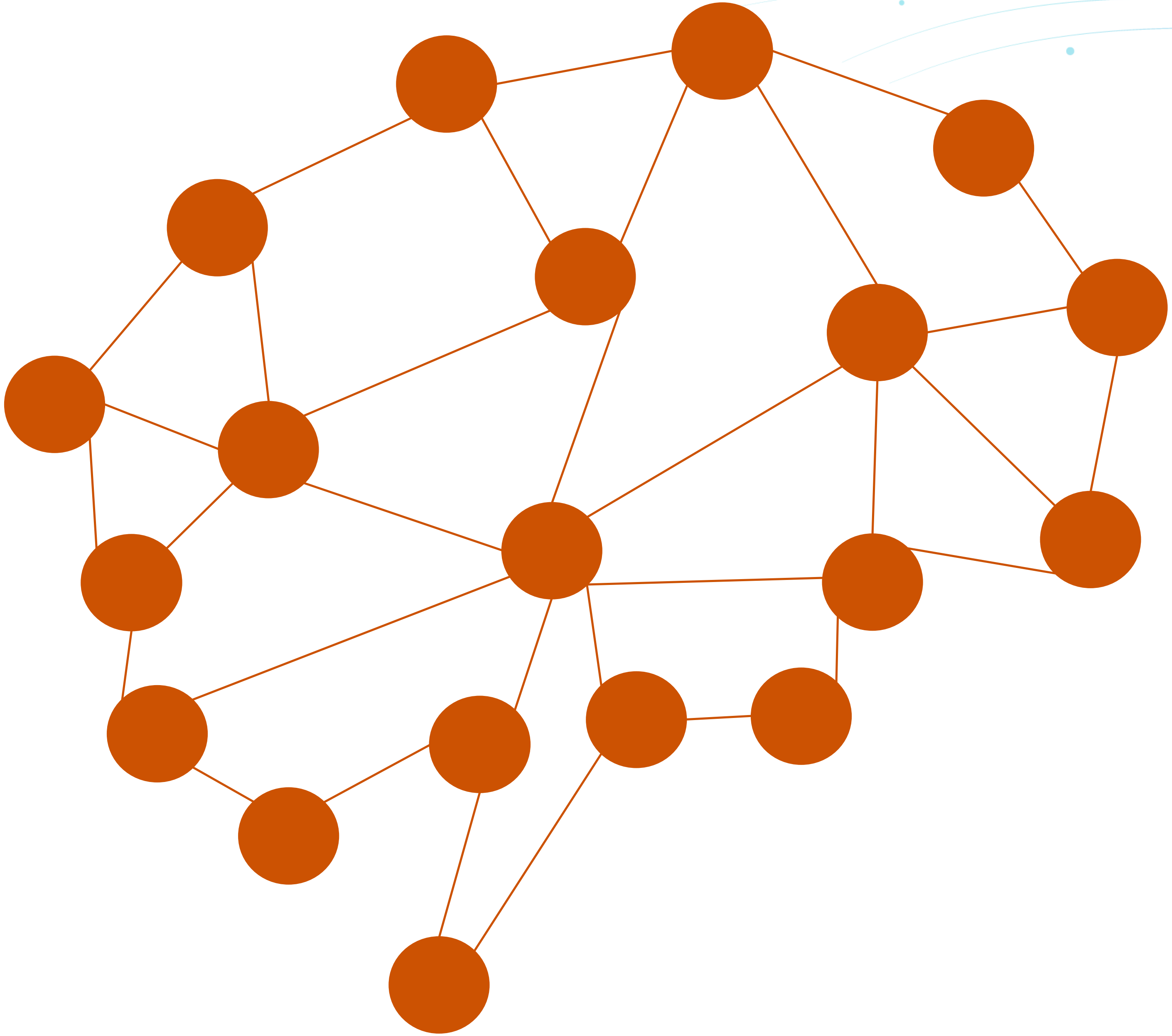
- Black box interpretability methods consider only the inputs and resulting decision of a classifier
  – This is exactly how an attacker would examine our model!

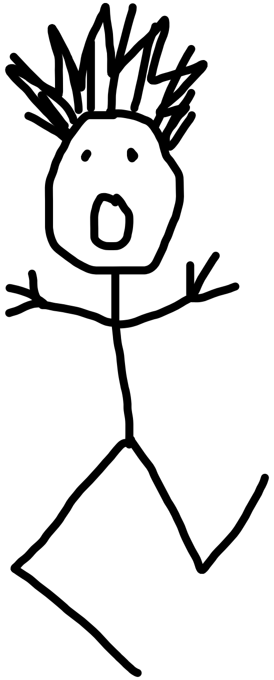
How attackers can use black box interpretability





30% threats first blocked through machine learning prediction





ALL YOUR ML ARE BELONG TO US

Key Takeaways

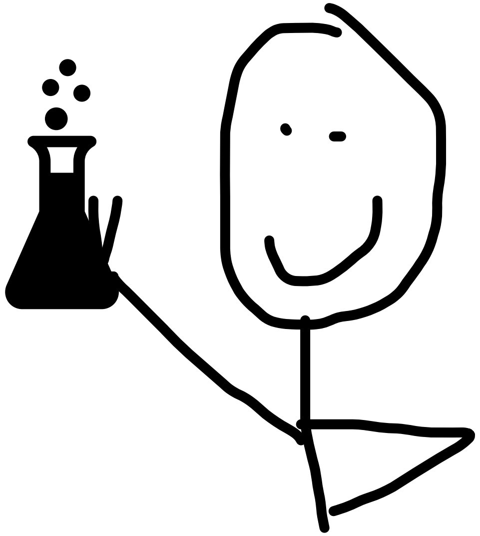
Interpretabilty methods are freely available

Make machine learning black box “explainable”

Help identify feature evasion and misleading features

But… attackers could also use these methods to their advantage

Start your experiments now before they do!



Questions?

References

- [1] "'Why Should I Trust You?': Explaining the Predictions of Any Classsifier“; Ribeiro, Singh, Geustrin (2016)
- [2] A Unified Approach to Interpreting Model Predictions; Lundberg, Lee (2017)
- [3] High-Precision Model-Agnostic Explanations; Ribeiro, Singh, Guestrin (2018)