BSides

2023

ML Classification:

Orlando Barrera II

Who I am

Who I am not

- Sr Security TAM at Fastly
- Active member in the Austin security community
- Not a data scientist or data analyst
  - Disclaimer: The views and opinions expressed in this presentation are my own and do not necessarily reflect the official policy or position of my employer. My employer Fastly / Signal Sciences is not associated with this presentation.

Who is this talk for?

Keywords: XSS, Machine Learning, Python, scikit-learn, bots

- Beginners Who want to learn about Supervised Machine Learning and Binary Classification

Overview

I. Reflected XSS Classification
II. Supervised ML
III. Training/Testing ML Algorithm
IV. ML “Bot” Classification

Challenge

Yourself

What is a problem you can solve using Machine Learning?

What is Machine Learning

My definition -

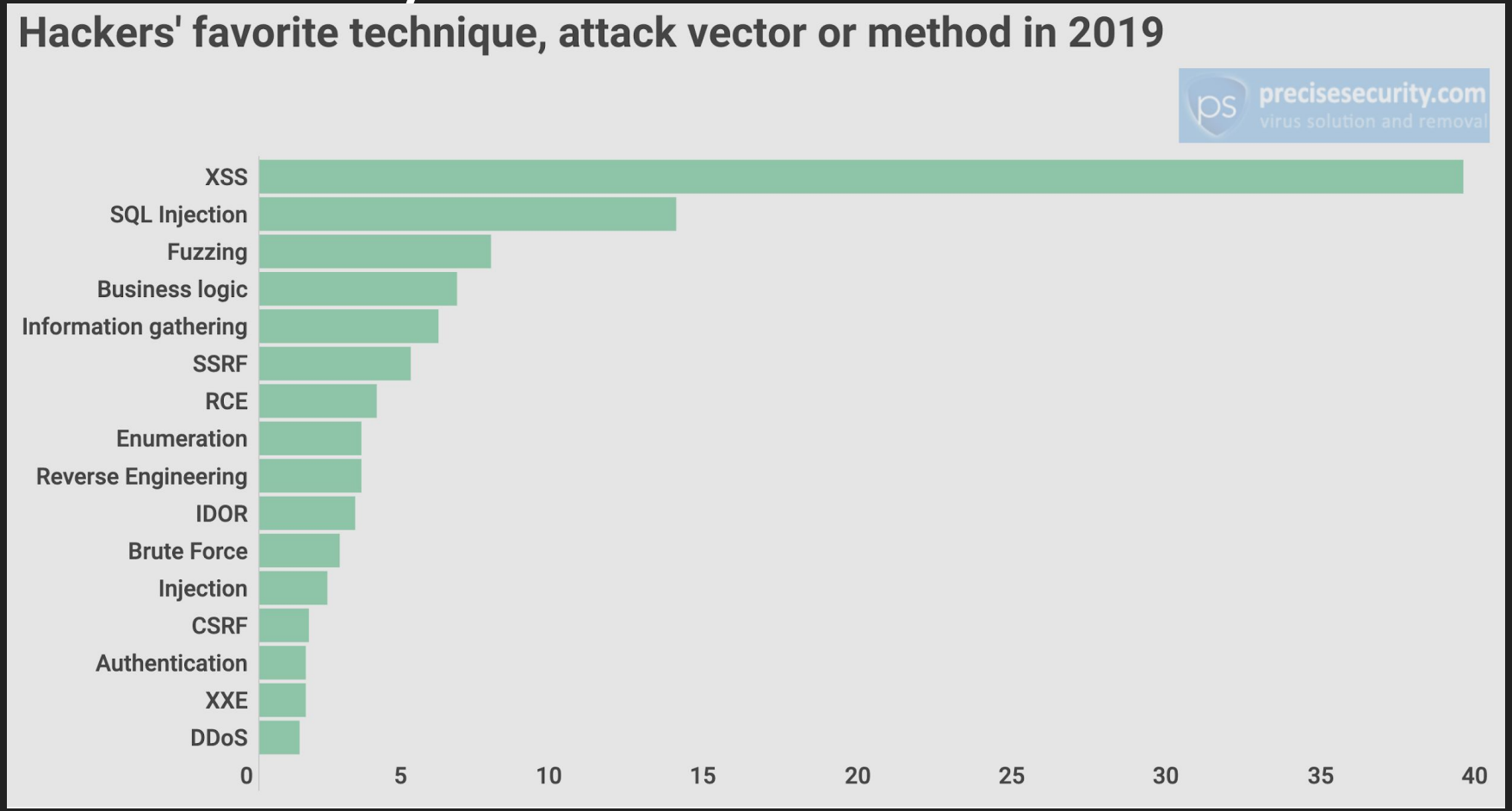
A tool to help semi-automate the validation of a hypothesis.

$f(x) = y$

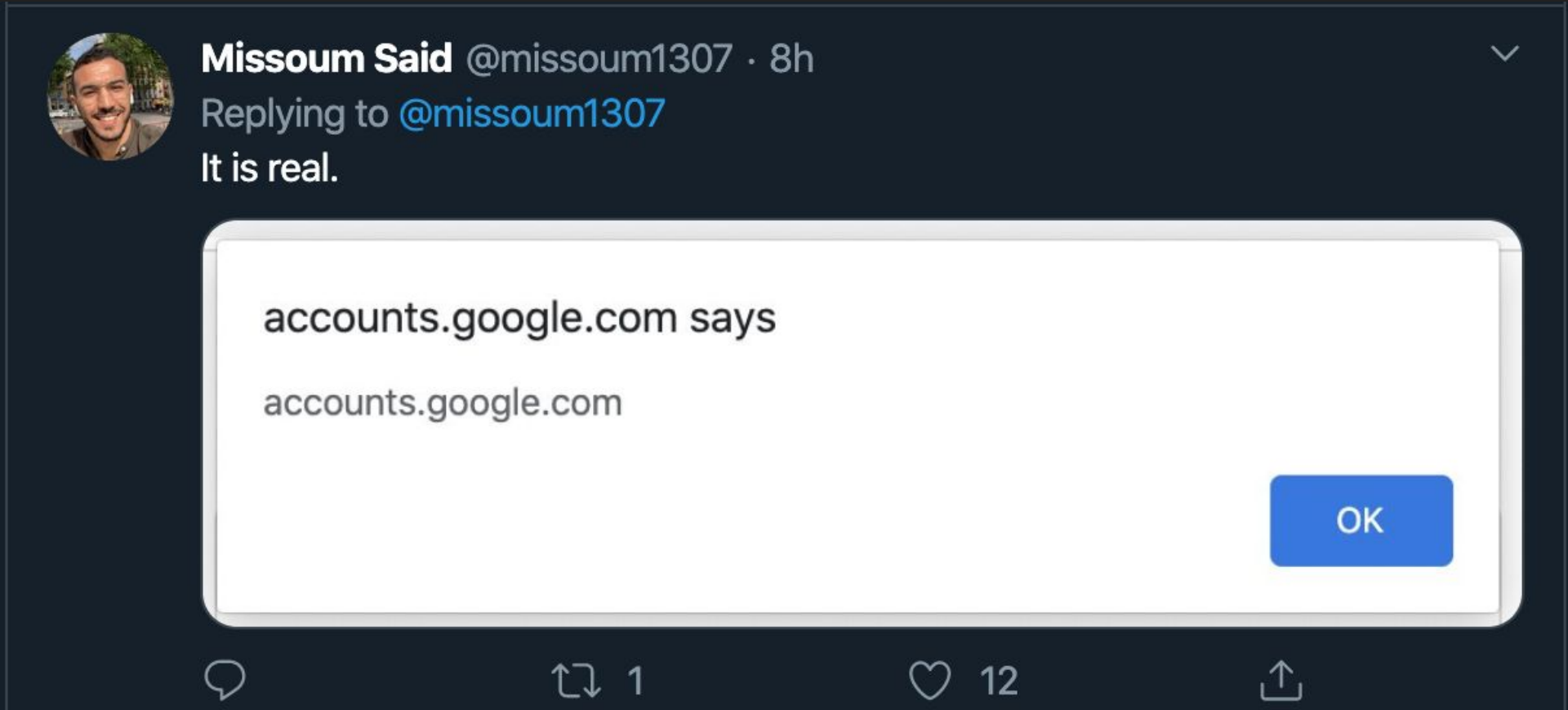
Reflected XSS Classification

http://example.com/page?var=<script>alert('xss')</script>

Why XSS Classification?



XSS Happens


Missoum Said @missoum1307 · 8h
Replying to @missoum1307
It is real.
accounts.google.com says
accounts.google.com
OK
1
12

XSS Classification: Regex

- Why use a regex to parse HTML at all?
- Use a parser to well… parse 🤷
- Just DON’T

<\s*script\b[^>]*>[^<]+<\s*/\s*script\s*>

<\s*script\b[^>]*>[^<]+<\s*/\s*script\s*>

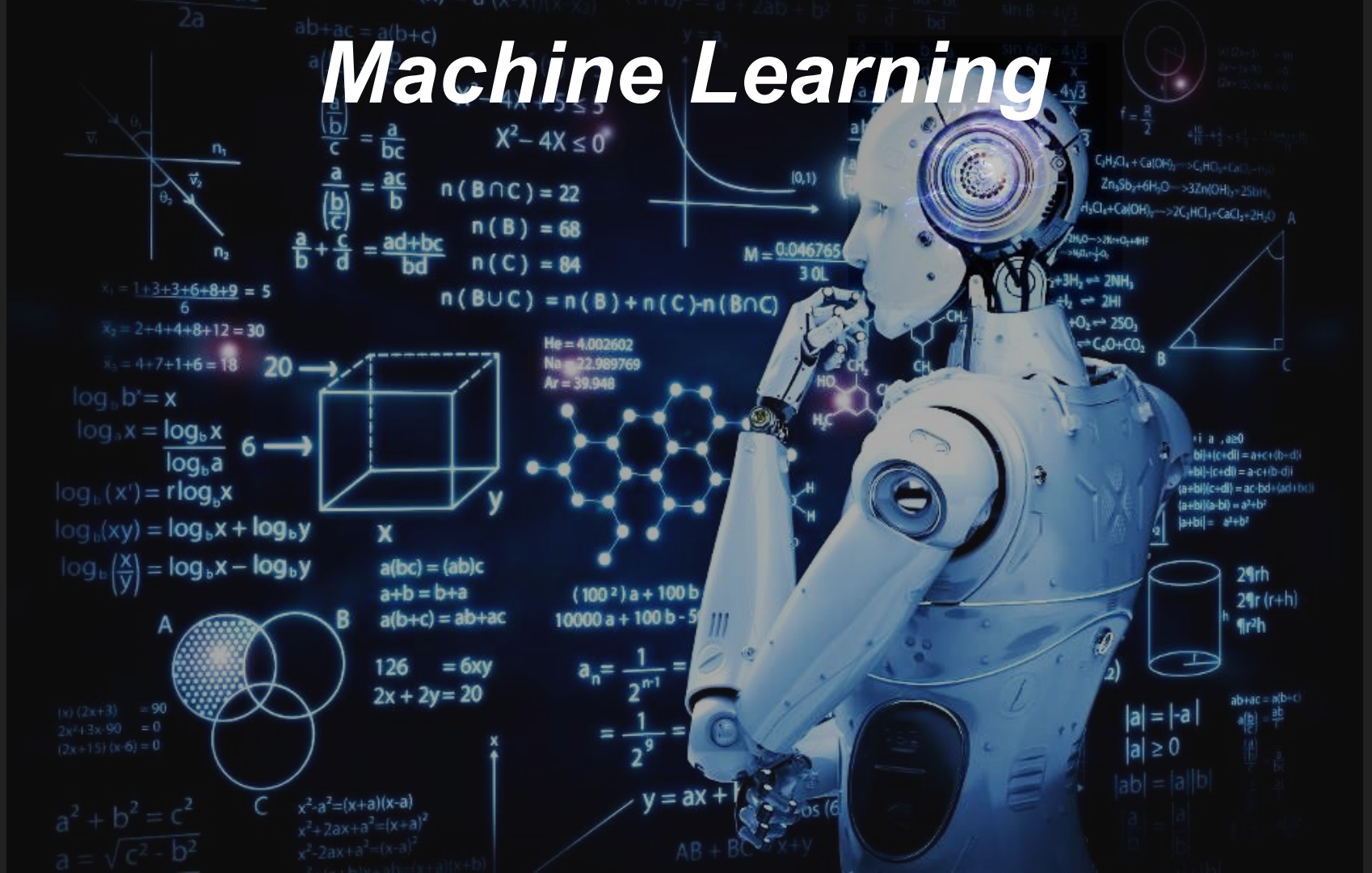
XSS Classification: Parsing and Tokenization

- libinjection XSS
- Used in OWASP ModSecurity Core Rule Set (CRS)

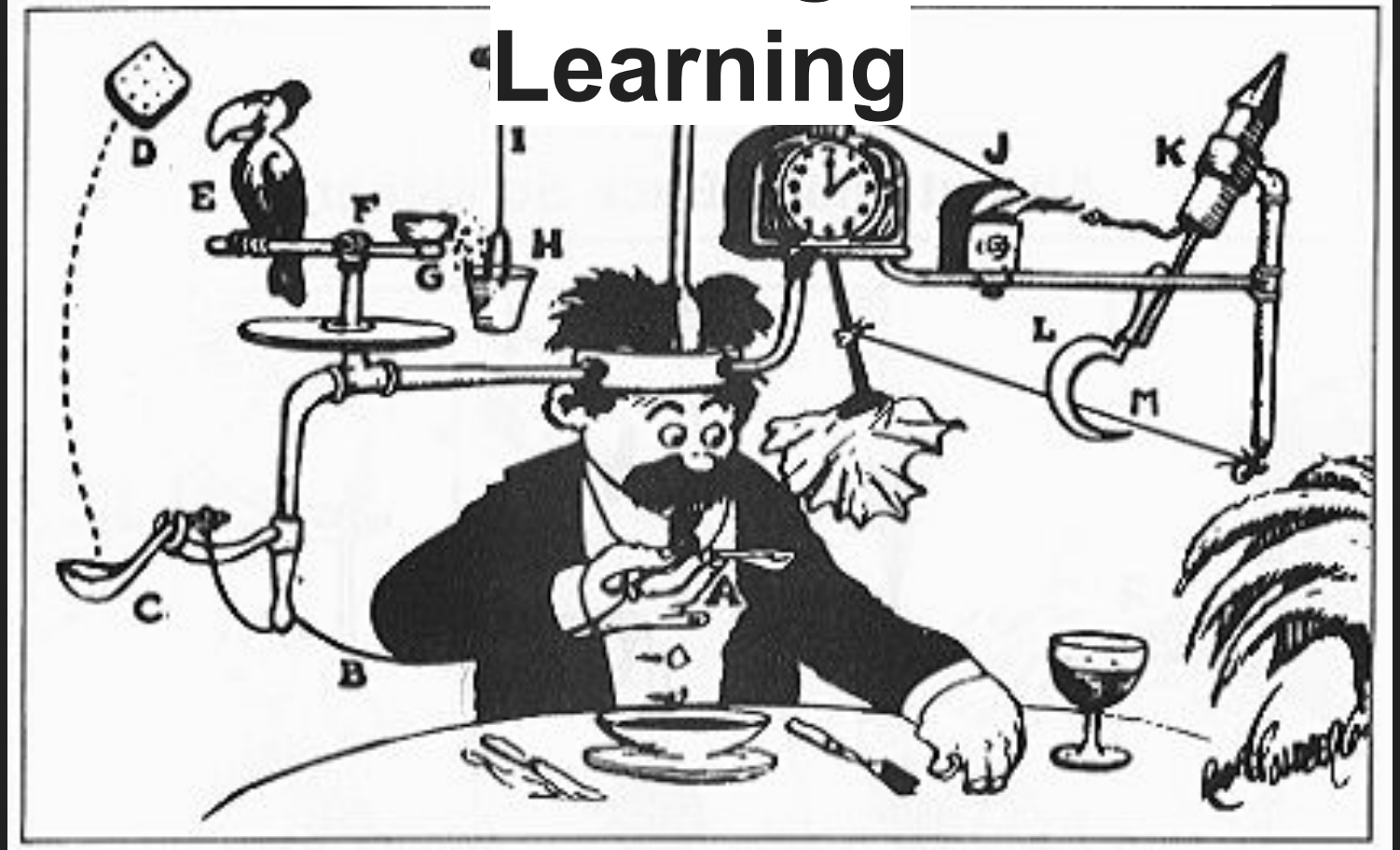
Why Machine Learning?

… because you get to work with models!

Machine Learning



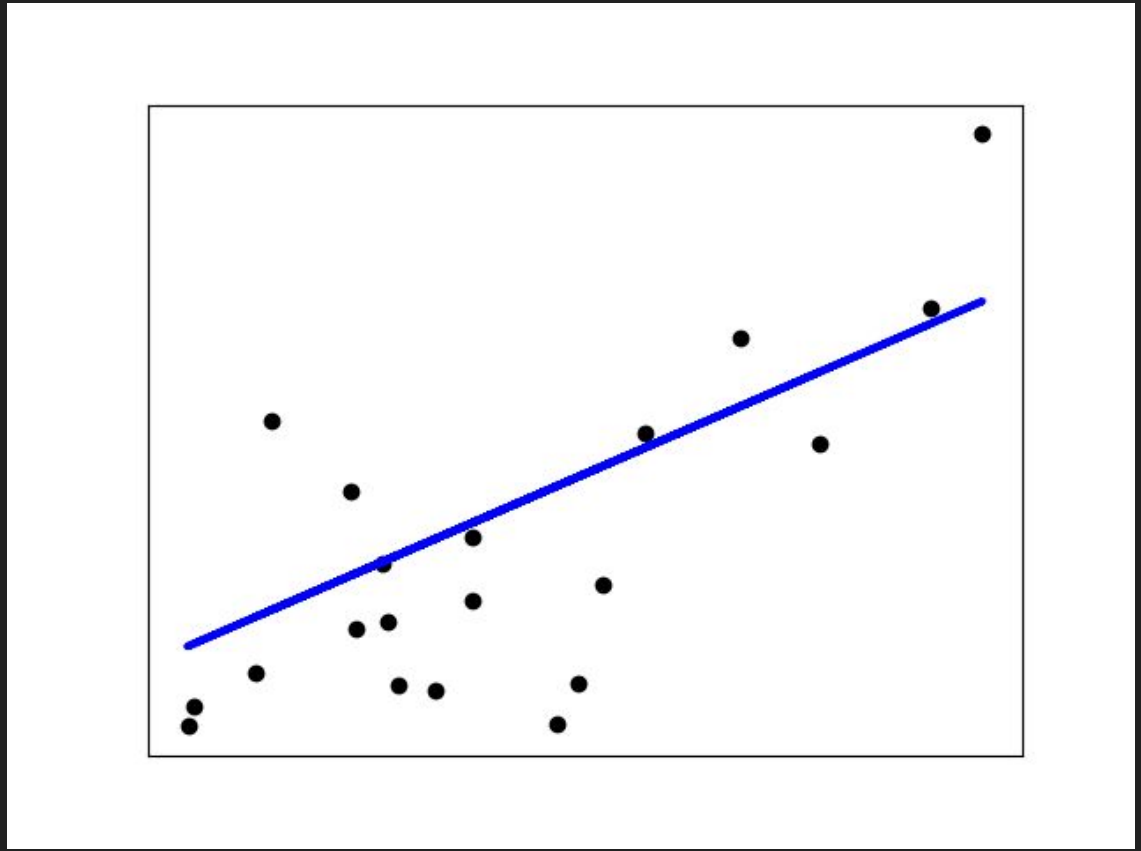
Rube Goldberg Machine Learning



Types of Supervised Machine Learning

I. Classification
II. Linear Regression

Linear Regression

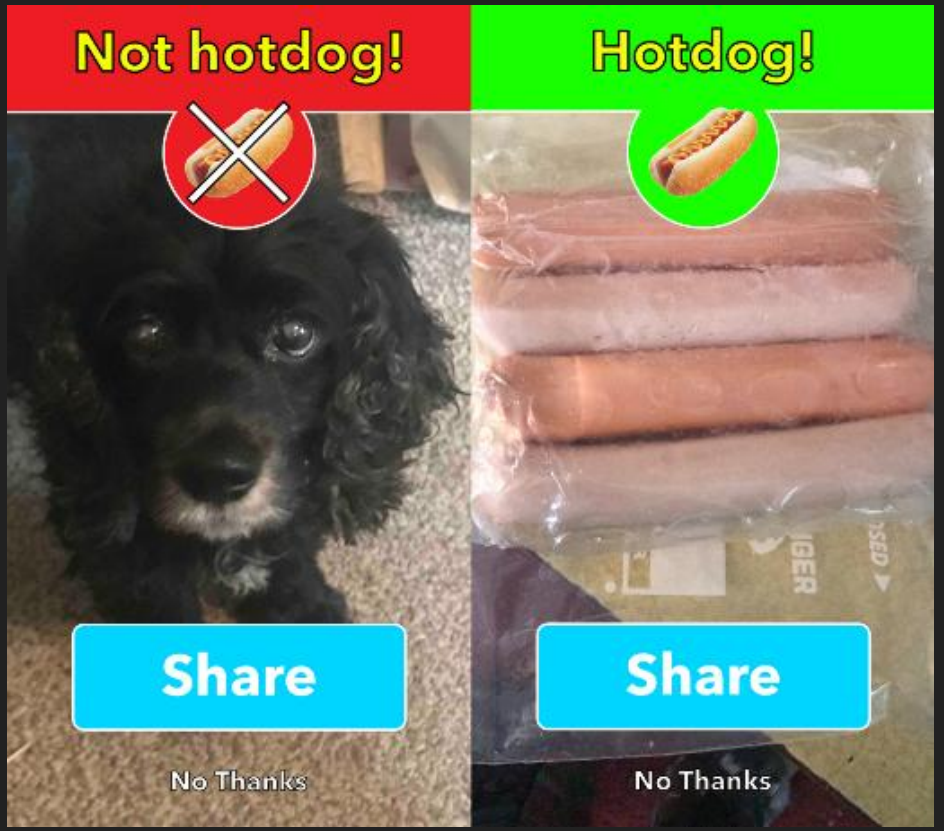


Machine Learning Terminology

- **Model**
- **Training**
- **Perdiction**

Machine Learning Terminology

- **Target (Label)**
- **Feature**



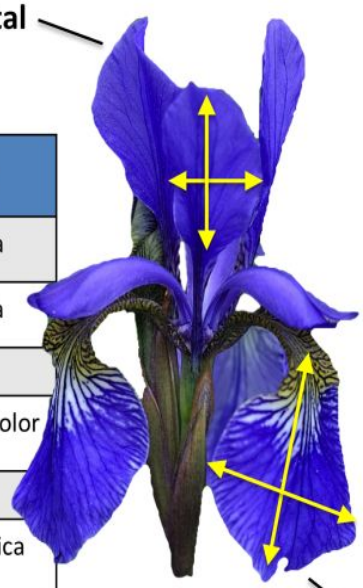

That's-- That's it?
It only does hotdogs?
HBO


No. And a "Not hotdog".

Iris Data Set

Samples
(instances, observations)

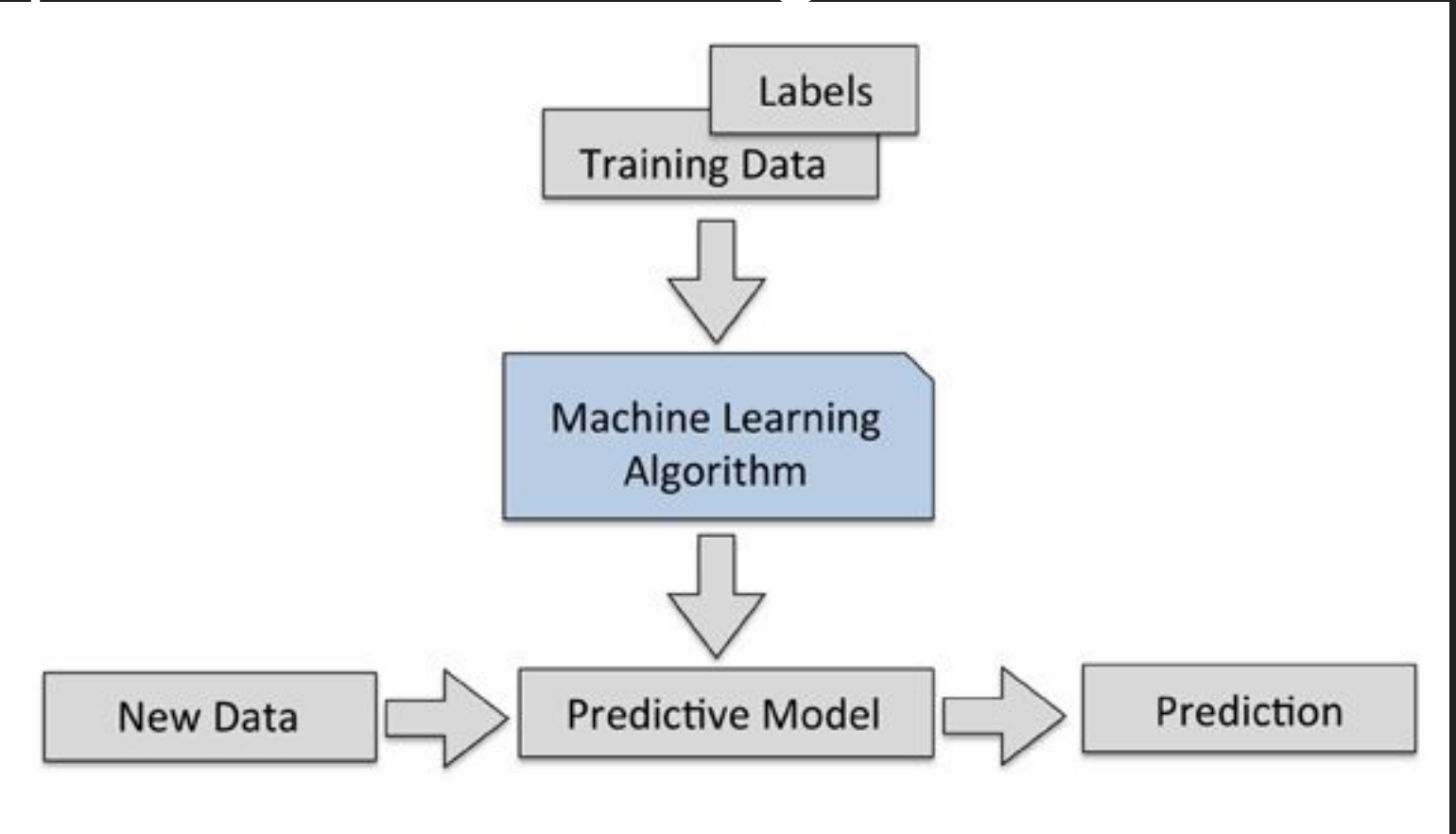
	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica



Features
(attributes, measurements, dimensions)

Class labels
(targets)

Supervised Learning: Classification



Previous and Related Work

J Inf Process Syst, Vol.13, No.4, pp.1014~1028, August 2017
https://doi.org/10.3745/JIPS.03.0079

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

XSSClassifier: An Efficient XSS Attack Detection Approach Based on Machine Learning Classifier on SNSs

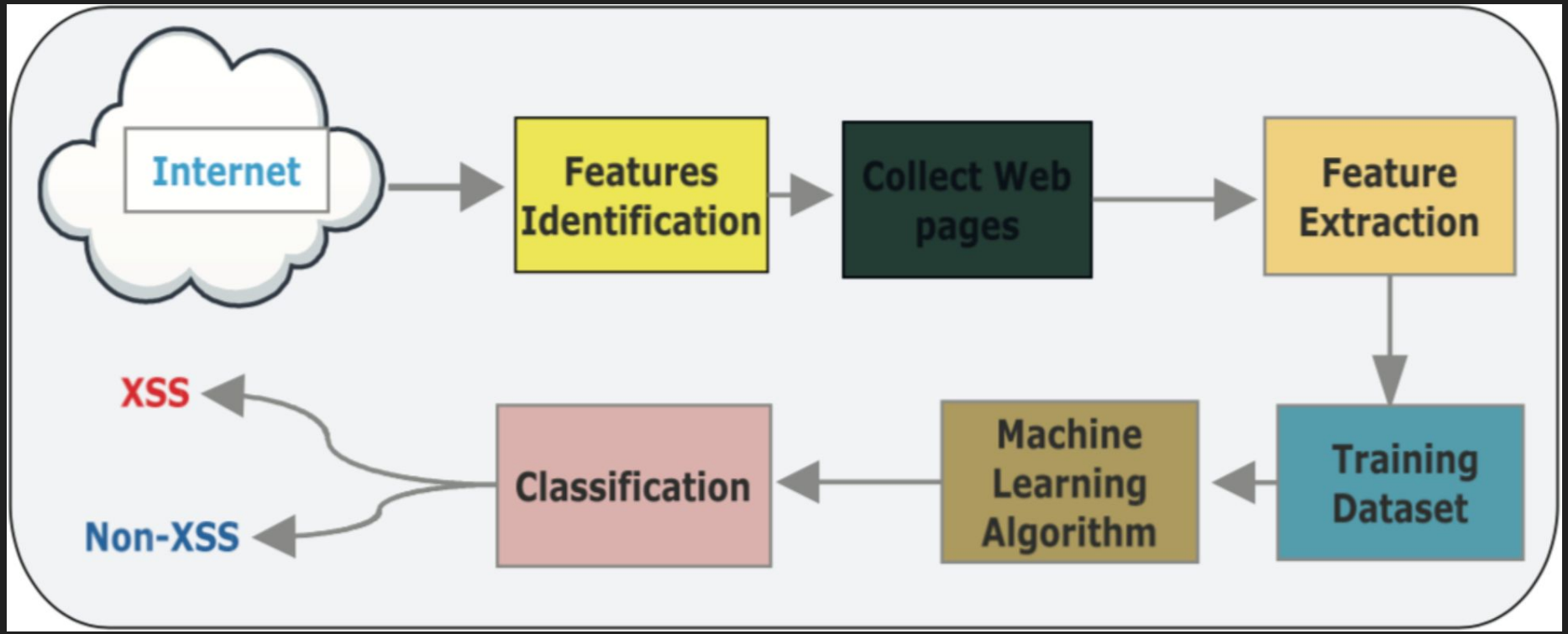
Shailendra Rathore*, Pradip Kumar Sharma*, and Jong Hyuk Park*

Previous and Related Work

Table 2. List of HTML tag features

No.	Feature	Type
1	Total count of harmful keywords	Integer
2	Total count of Iframe	Integer
3	Total count of external Iframe source	Integer
4	Total count of external links	Integer
5	Having trouble with event handlers	Boolean
6	Presence of malicious java script method	Boolean
7	Total count of DOM-modified keywords	Integer
8	Total count of String-decoding keywords.	Integer
9	Total count of AJAX keywords, and Other Keywords	Integer
10	The maximum size of the script	Integer
11	Total count of maximum size of the script	Integer
12	Total count of encoded links	Integer
13	Maximum count of encoded characters in JavaScript	Integer
14	Maximum size of HTML tags	Integer
15	Total count of maximum size HTML tags	Integer
16	Maximum count of JavaScript strings in HTML tags	Integer
17	The existence of obfuscation code	Boolean
18	Total count of external Script source	Integer

Previous and Related Work



Github Repo

https://github.com/obarrera/ML-XSS-Detection

XSSClassifier: XSS Attack Detection Approach Based on Machine Learning Classifier in Python

Research/Related Work

Based on the white paper, XSSClassifier: An Efficient XSS Attack Detection Approach Based on Machine Learning Classifier on SNSs:

https://pdfs.semanticscholar.org/2c74/d8e94b73c35e8651189262d0c7f32e6cfc7c.pdf?_ga=2.60428845.1871262773.1581209097-1873828646.1581209097

The Doc2Vec method, a concept that was presented in 2014 by Mikilov and Le, the underlying intuition of Doc2Vec is that the document representation should be good enough to predict the words in the document: https://cs.stanford.edu/~quocle/paragraph_vector.pdf

scikit-learn is a Python module for machine learning: https://github.com/scikit-learn/scikit-learn

XSS Machine Learning: https://github.com/Xyntax/ML/blob/master/xss_word_vector/xss_wordvector.ipynb

Data Collection - 44,873 “Not XSS” Samples

```
for geo in country:
    file = urlopen('https://trends.google.com/trends/trendingsearches/daily/rss?geo='+geo)
    #convert to string:
    data = file.read()
    #print(data)
    #close file because we dont need it anymore:
    file.close()

    #entire feed
    root = etree.fromstring(data)
    #print(root.attrib)

    for i in root.iter('item'):
        #print(i)
        for x in i:
            #print(x)
            for y in x:
                #print(y)
                for z in y.iter('{https://trends.google.com/trends/trendingsearches/daily}news_item_url'):
                    num += 1
                    print(str(num)+" => "+z.text)
                    url_List.append(z.text)
                    #f.write(z.text+"\n")
    time.sleep(1)
#f.close()

# Remove Duplicates

uniqlines = set(open(testFile).readlines())
open(testFile, 'w').writelines(set(uniqlines))

time.sleep(1)
```

```
57 => https://www.posttoday.com/social/general/587015
58 => https://www.khaosod.co.th/stock-monitor/news_2442568
59 => https://www.thaipost.net/main/detail/34235
60 => https://www.hoonsmart.com/archives/53502
```

Data Collection - 38,714 XSS Samples

```
    html = urlopen("http://www.xssed.com/mirror/"+str(i))
    #print(html)
    # parse the html using beautiful soup and store in variable `soup`
    soup = BeautifulSoup(html, 'html.parser')
    row = soup.find_all('th', attrs={'class': 'row3'})
    url = str(row).split('URL:')
    url = str(url[1]).split('</th>')
    urlStr = str(url[0]).replace("<br/>","")
    urlStr = urlStr.strip()
    if urlStr:
        query = parse.urlsplit(urlStr)[3]
        #if "http" in str(query):
        #    continue
        if query:
            print(str(i)+" => "+urlStr)
            # save the XSS URL to the file
            f.write(urlStr+"\n")
            time.sleep(1)

f.close()

time.sleep(15)
```

```
\';alert(String.fromCharCode(84,69,83,84,69))//%22;alert(String.fromCharCode(84,69,83,84,69))//\%22;alert(String.fromCharCode(84,69,83,84,69))//--%3E%3C/SCRIPT%3E%22%3E'%3E%3CSCRIPT%3Ealert(String.fromCharCode(84,69,83,84,69))%3C/SCRIPT%3E
53 => http://uniweb.legislature.ne.gov/legaldocs/search.php?search_by=keyword&amp;keywords=%27%3Balert%28String.fromCharCode%2884%2C69%2C83%2C84%2C69%29%29%2F%2F%5C%27%3Balert%28String.fromCharCode%2884%2C69%2C83%2C84%2C69%29%29%2F%2F%22%3Balert%28String.fromCharCode%2884%2C69%2C83%2C84%2C69%29%29%2F%2F%5C%22%3Balert%28String.fromCharCode%2884%2C69%2C83%2C84%2C69%29%29%2F%2F--%3E%3C%2FSCRIPT%3E%22%3E%27%3E%3CSCRIPT%3Ealert%28String.fromCharCode%2884%2C69%2C83%2C84%2C69%29%29%3C%2FSCRIPT%3E
52 => http://www.shoptime.com.br/apollo/vitrine.do?method=show&amp;lojaPrincipal=&amp;areaName=busca&amp;nomeLoja=&amp;tipoLoja=&amp;tipoBusca=comFoto&amp;fetch=30&amp;loja=&amp;palavra=';alert(String.fromCharCode(84,69,83,84,69))//
\';alert(String.fromCharCode(84,69,83,84,69))//%22;alert(String.fromCharCode(84,69,83,84,69))//\%22;alert(String.fromCharCode(84,69,83,84,69))//--%3E%3C/SCRIPT%3E%22%3E'%3E%3CSCRIPT%3Ealert(String.fromCharCode(84,69,83,84,69))%3C/SCRIPT%3E&amp;x=43&amp;y=24
51 => http://www.securitypronews.com/emailfriend.php?title='%22%3Cscript%3Ealert%28%22XSS_Vulnerable%20:-/%22%29%3C%2Fscript%3E
```

Data Collection - 83,587 Total Samples

testXSS.txt

```
http://www.sexyrosebuds.com/index.php?
task=search&amp;nocache=1&amp;SID=&amp;md=shop&amp;lang=fr&amp;catcode=4050&amp;keywords=%27%22%3E%3C
1%3EXSS+by+Xylitol%3C%2Fh1%3E%3C%2Fmarquee%3E&amp;go.x=29&amp;go.y=5
http://somesite.com?test=&lt;DIV STYLE=&quot;width: expression(alert(&apos;XSS&apos;));&quot;&gt;
http://www.guardiacivil.org/directorio/consulta_new.jsp?provincia="&gt;&lt;script&gt;alert(document.c
http://www.zcrack.com/crack_download_search.php?crack=%3Ch1%3EHacked%21%3Cp+align%3Dcenter%3E%3Cmarqu
```

testNORM.txt

```
https://api.addthis.com/oexchange/0.8/forward/google_plusone_share/offer?url=https://www.aciprensa.com/notici
invocan-y-lo-aman-38502&pubid=ra-533d9f87402355d5
http://sports.khan.co.kr/news/page/art_list.html?mcode=series&sec_id=so029
https://seeme.me?utm_source=mthai-music&utm_campaign=mainmenu&utm_medium=Seeme
https://vice.idnes.cz/novinari.aspx?idnov=2370
http://www.asahi.com/shimen/?iref=comtop_rnavi_top_p
/AjaxPage?jspName=neoSendToFriendForm.jsp&fromStatusBar=fromStatusBar&subject=%D7%96%D7%94+%D7%94%D7%96%D7%9E
%D7%94%D7%A8%D7%9B%D7%99%D7%91%D7%94+%D7%94%D7%A9%D7%95%D7%95%D7%99%D7%9D+%D7%91%D7%99%D7%95%D7%AA%D7%A8+
%D7%91%D7%A2%D7%A8%D7%91%D7%94&sendURL=http%3A%2F%2Fwww.mako.co.il%2Ftravel-israel%2Farava-taste-south-of-
Israel%2FArticle-94b0ba723039761006.htm%3FsCh%3D9257dce6ef0e1610%26pId%3D173113802%26partner%3Dsend_to_friend
https://twitter.com/intent/tweet?text=Brentford%202-0%20Leeds%20United%3A%20Championship%20-
%20as%20it%20happened&url=https%3A%2F%2Fwww.theguardian.com%2Ffootball%2Flive%2F2019%2Fapr%2F22%2Fbrentford-v
live%3FCMP%3Dshare_btn_tw%26page%3Dwith%3Ablock-5cbdfd068f08d5e277b52aca%23block-5cbdfd068f08d5e277b52aca
http://fotbal.idnes.cz/fotbal-online.aspx?online=7097916
https://twitter.com/intent/tweet?text=Dono+de+empresa+de+formatura+que+deu+golpe+em+alunos+tem+quase+R%24+500
de-empresa-de-formatura-que-deu-golpe-em-alunos-tem-quase-500-mil-em-dividas-23614488%3futm_source%3dTwitter%
https://www.ndtv.com/world-news/sri-lanka-bombings-three-children-of-denmarks-richest-man-killed-in-sri-lanka
https://isport.blesk.cz/galerie/fotbal-1-liga-rocnik-2018-19/378237/sestrihy-favorite-tapali-slavia-padla-kol
http://www.asahi.com/special/hataraku/?iref=com_sitemap
https://twitter.com/EnzoTroiano?ref_src=twsrc%5Etfw
https://plus.google.com/share?url=http://bit.ly/2VVCkPe
http://www.moneycontrol.com/india/newsarticle/news_email.php?autono=3876141
http://stock.hankyung.com/apps/analysis.current?nid=01&itemcode=027410
https://com.nicovideo.jp/search/%E3%81%82%E3%82%93%E3%81%9F?mode=s
http://stock.hankyung.com/apps/analysis.current?nid=01&itemcode=J55E258
https://www.facebook.com/dialog/share?app_id=132746074315&display=popup&href=https%3A%2F%2Fwww.engadget.com%2
```

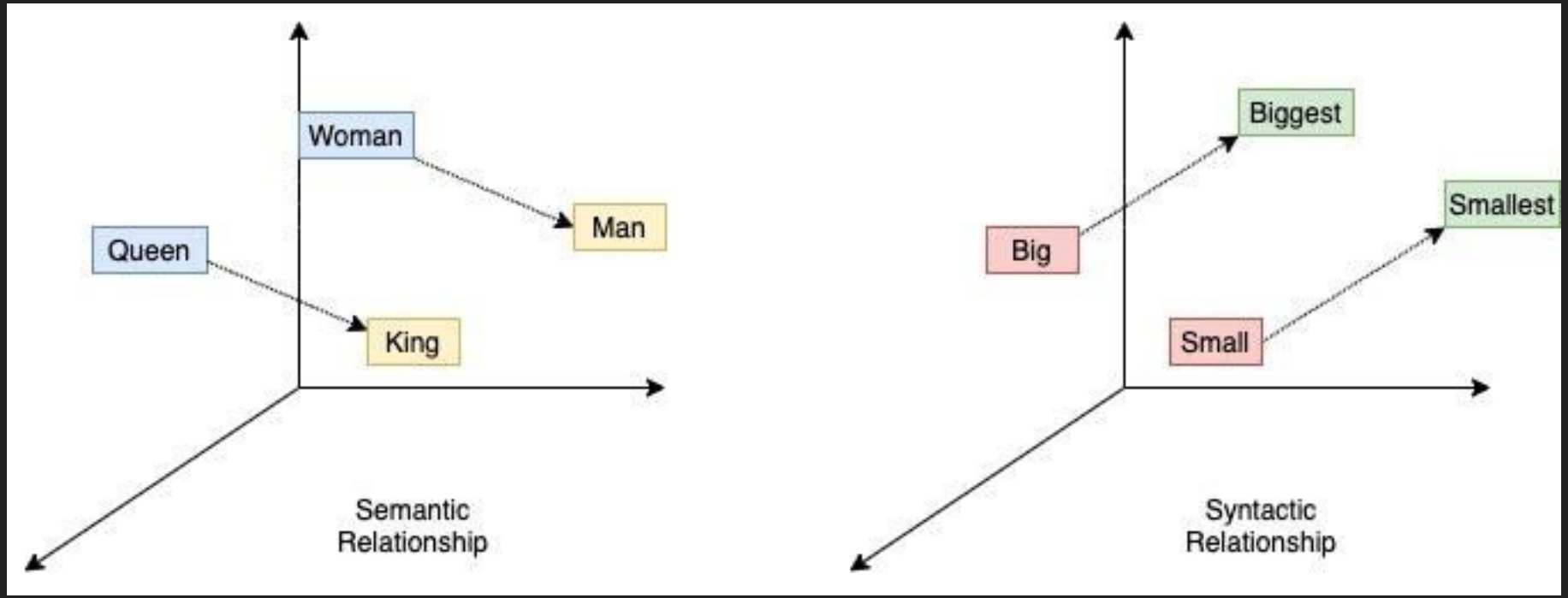
Data Vectorization and Features

Using the function getVec(['data'])

```
testText = [
                'Search=%22%3E\'%3E%3CSCRIPT%20SRC=http://br.zone-h.org/testes/xss.js%3E%3C/SCRIPT%3E?',
                'id=15%3Cscript%3Ealert%28document.cookie%29%3C/script%3E',
                'composite_search=1&keyword="/><script>alert("Xss:Vijayendra")</script>'
            ]

Xnew = getVec(testText)
```

Word2Vec

You may be familiar with the bag-of-words model.



Word2Vec

This model transforms each document to a fixed-length vector of integers.

For example, given the sentences:

- John likes to watch movies. Mary likes movies too.
- John also likes to watch football games. Mary hates football.

The model outputs the vectors:

- [1, 2, 1, 1, 2, 1, 1, 0, 0, 0, 0]
- [1, 1, 1, 1, 0, 1, 0, 1, 2, 1, 1]

Doc2Vec

Le and Mikolov in 2014 introduced the Doc2Vec algorithm, which usually outperforms such simple-averaging of Word2Vec vectors.

```
model = Doc2Vec(vector_size=vec_size,
                alpha=alpha,
                min_alpha=0.00025,
                min_count=1,
                dm=1)
model.build_vocab(tagged_data)
print("Building the sample vector model...")
features = []
for epoch in range(max_epochs):
    #print('Doc2Vec Iteration {0}'.format(epoch))
    print("*", sep=' ', end='', flush=True)
    model.random.seed(42)
    model.train(tagged_data,
                total_examples=model.corpus_count,
                epochs=model.epochs)
    # decrease the learning rate
    model.alpha -= 0.0002
    # fix the learning rate, no decay
    model.min_alpha = model.alpha
model.save("lib/d2v.model")
print()
print("Model Saved")
```

Data Vectorization and Features

```
# append the features
featureVec = np.append(featureVec,feature1)
featureVec = np.append(featureVec,feature2)
featureVec = n
featureVec = n
featureVec = n
featureVec = n
featureVec = n
featureVec = n
```

```
# add feature for malicious tag count
feature1 = int(lowerStr.count('link'))
feature1 += int(lowerStr.count('object'))
feature1 += int(lowerStr.count('form'))
feature1 += int(lowerStr.count('embed'))
feature1 += int(lowerStr.count('ilayer'))
feature1 += int(lowerStr.count('layer'))
feature1 += int(lowerStr.count('style'))
feature1 += int(lowerStr.count('applet'))
feature1 += int(lowerStr.count('meta'))
feature1 += int(lowerStr.count('img'))
feature1 += int(lowerStr.count('iframe'))
```

Data Vectorization and Features

Features Data

```
print("Test Sample: "+ X[0])
print("Features: " + str(features[0]))
print("\nLabel:\033[1;31;1m XSS(1)/\033[1;32;1m NOT XSS(0)\033[0;0m: " + str(y[0]))
```

```
Test Sample: http://search.rin.ru/cgi-bin/find.cgi?text=%3Cscript%3Ealert(%27HZ+iz+1337%27)%3B%3C%2Fscript%3E

Features: [ 0.37698153  0.43000719 -0.4600772   0.14952293 -1.10670066 -0.63367283
 -1.33330667 -0.57465571  2.15166187 -2.28842187 -0.59600937 -0.15612525
  0.97208875  1.63351858 -0.53387594 -1.16791904  0.57315004 -0.79807025
 -2.37217093  1.13317549  0.          1.          0.          0.
 97.          3.         15.          1.        ]

Label: XSS(1)/ NOT XSS(0): 1
```

Scikit-learn

- DecisionTree
- SVC
- GaussianNB
- KNeighbors
- RandomForest
- MLP

Training the Models

```
# import the class
from sklearn.neighbors import KNeighborsClassifier

# instantiate the model
knn = KNeighborsClassifier(n_neighbors=1)

# fit the model with data
knn.fit(X, y)

# predict the response for new observations
knn.predict([[3, 5, 4, 2]])
```

How to Compare Model Performance?

Testing the Models

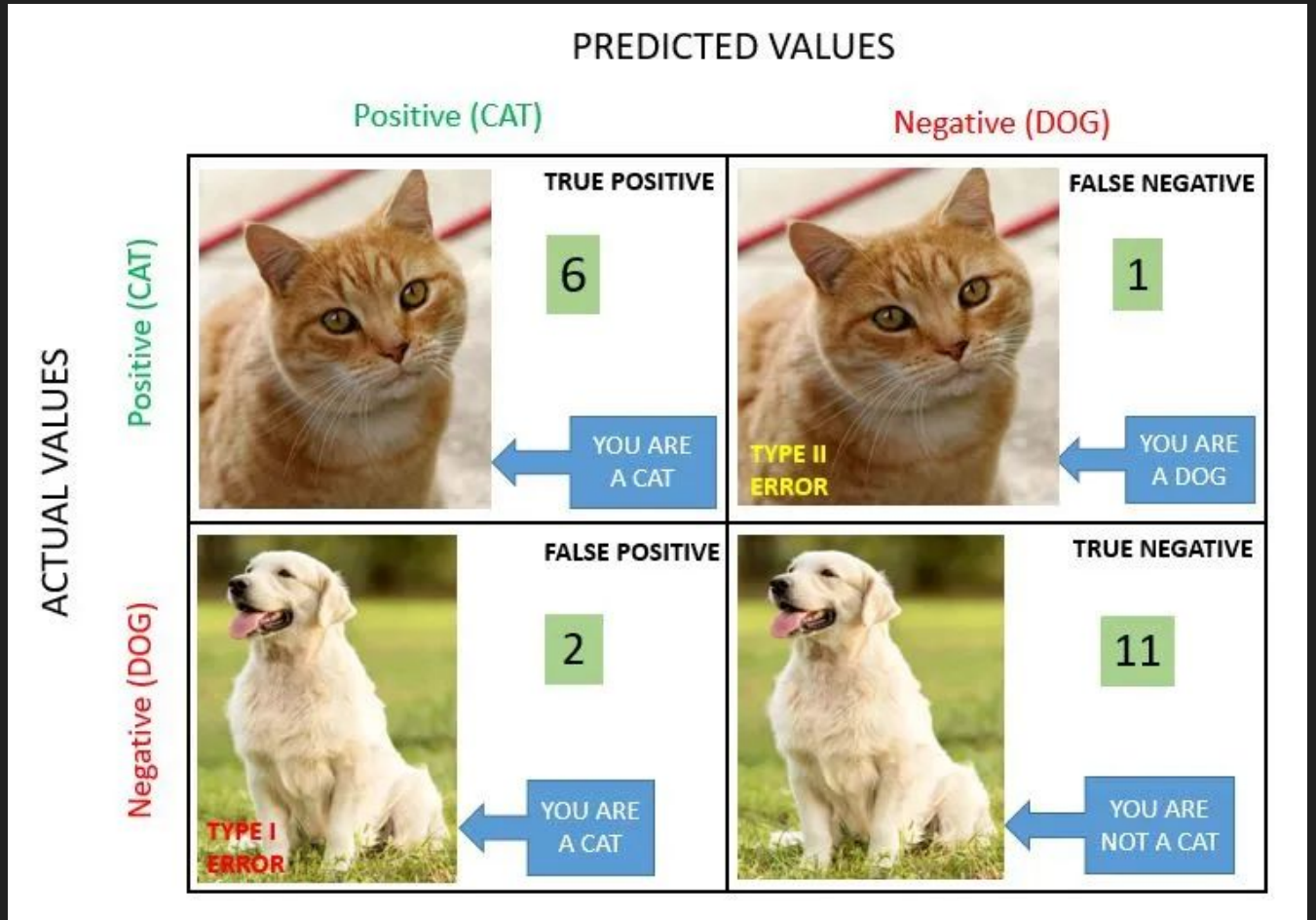
```
# Scikit-Learn Train Test Split Function
X_train, X_test, y_train, y_test =
train_test_split(features, y, test_size = .3, random_state=42)

# fit the model with data
my_classifier1.fit(X_train, y_train)

# predict the response for new observations
predictions1 = my_classifier1.predict(X_test)

score1 = accuracy_score(y_test, predictions1)))
```

Classification Report

Classification Report #2 SVC

	precision	recall	f1-score	support
0	0.97	0.99	0.98	13465
1	0.99	0.97	0.98	11588
accuracy			0.98	25053
macro avg	0.98	0.98	0.98	25053
weighted avg	0.98	0.98	0.98	25053

Confusion Matrix



Confusion Matrix

```
Confusion Matrix #6 MLPClassifier
[[13432    33]
 [   96 11492]]
```

Training Accuracy

- DecisionTree Classifier Accuracy Score: 98.5%
- SVC Accuracy Score: 98.2%
- GaussianNB Accuracy Score: 94.8%
- KNeighbors Classifier Accuracy Score: 96.0%
- RandomForest Classifier Accuracy Score: 99.5%
- MLP Classifier Accuracy Score: 99.5%

Make a Prediction

```
testXSS = [
'<script>alert(\'xss\')</script><script><script>', ]

Xnew = getVec(testXSS)

# make a prediction

ynew1 = loaded_model1.predict(Xnew)
```

Prediction Results

```
1
1
1
0
1
1
0.925
XSS => <script>alert('xss')</script><script><script>
```

What Next?

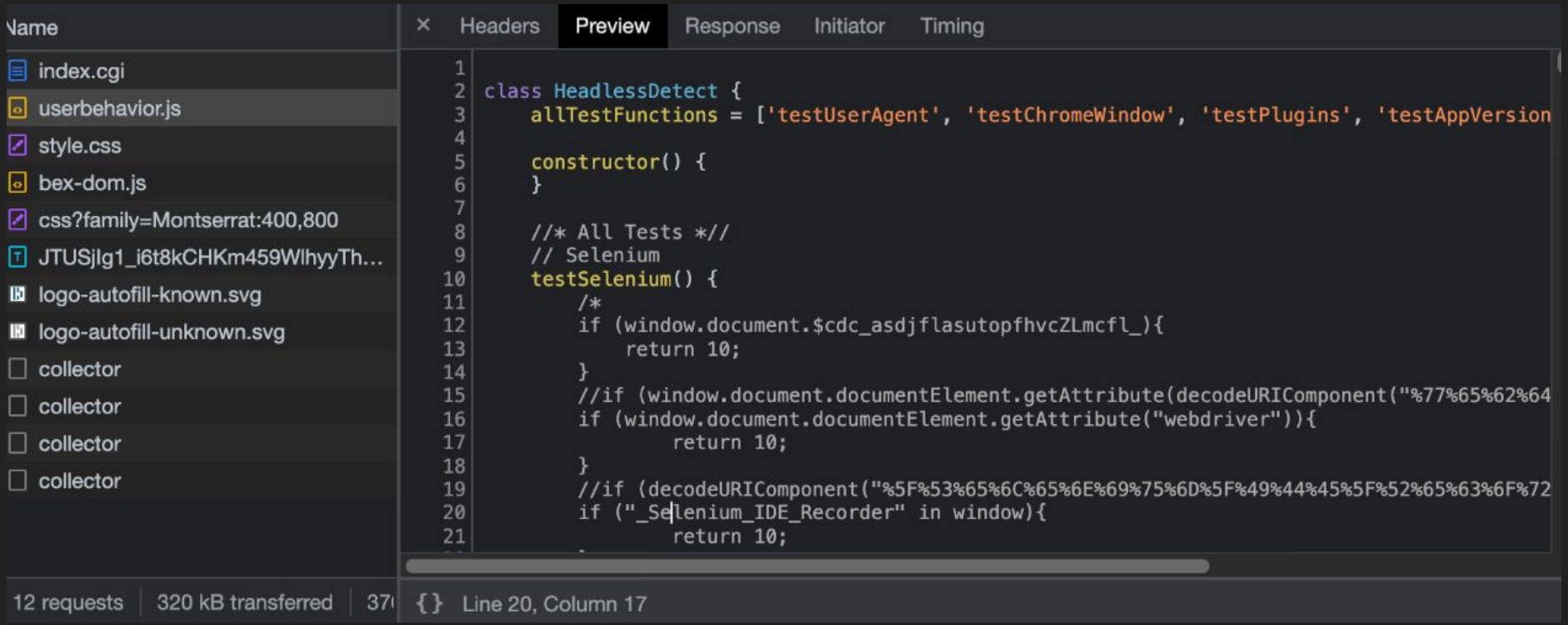
- OWASP ML Classification?
  - SQLi
  - CMD injection
  - Dir Traversal
  - etc…
- Libinjection parsing and tokenizing to train ML models
- Bot Classification?

Bot Classification

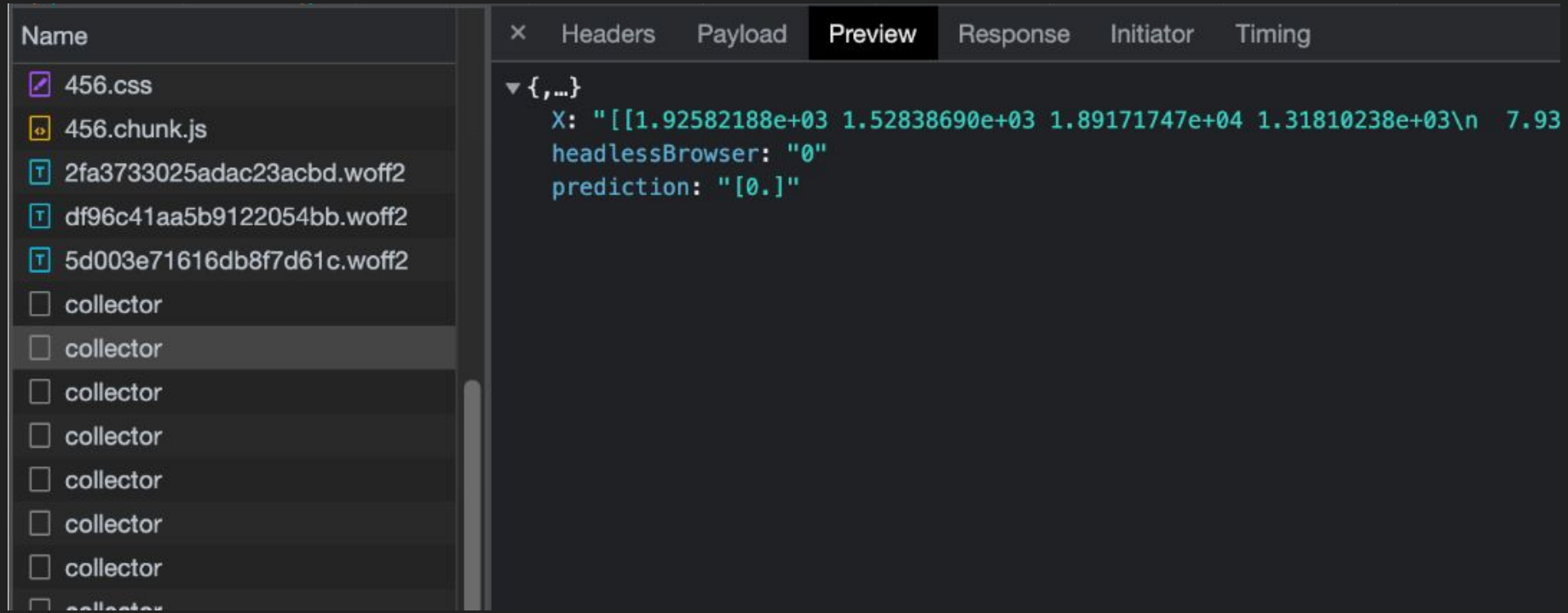
- Client Side JS to gather User Behavior
- Small ML model to run in AWS Lambda Serverless

```
// End results, what is shown to the user
results = {
    userInfo: {
        appCodeName: navigator.appCodeName || '',
        appName: navigator.appName || '',
        appVersion: navigator.appVersion|| '',
        vendor: navigator.vendor || '',
        platform: navigator.platform || '',
        userAgent: navigator.userAgent || '',
        plugins: navigator.plugins|| '',
        mediaDevices: navigator.mediaDevices || '',
        geolocation: navigator.geolocation || '',
        productSub: navigator.productSub || '',
        hardwareConcurrency: navigator.hardwareConcurrency || '',
        maxTouchPoints: navigator.maxTouchPoints || '',
        language: navigator.language || '',
        onLine: navigator.onLine || '',
        doNotTrack: navigator.doNotTrack || '',
        downlink: navigator.connection.downlink || '',
    },
    time: {
        totalTime: 0,
        timeOnPage: 0,
    },
    clicks: {
        clickCount:0,
        clickDetails: [],
    },
    mouseMovements: [],
    contextChange: [],
    keyLogger: [],
    headlessBrowser: headlessBrowser|| '',
    remoteIP: remoteIP|| '',
    collectorID: collectorID|| '',
    collectorKey: collectorKey|| '',
    windowLocation: windowLocation|| '',
```

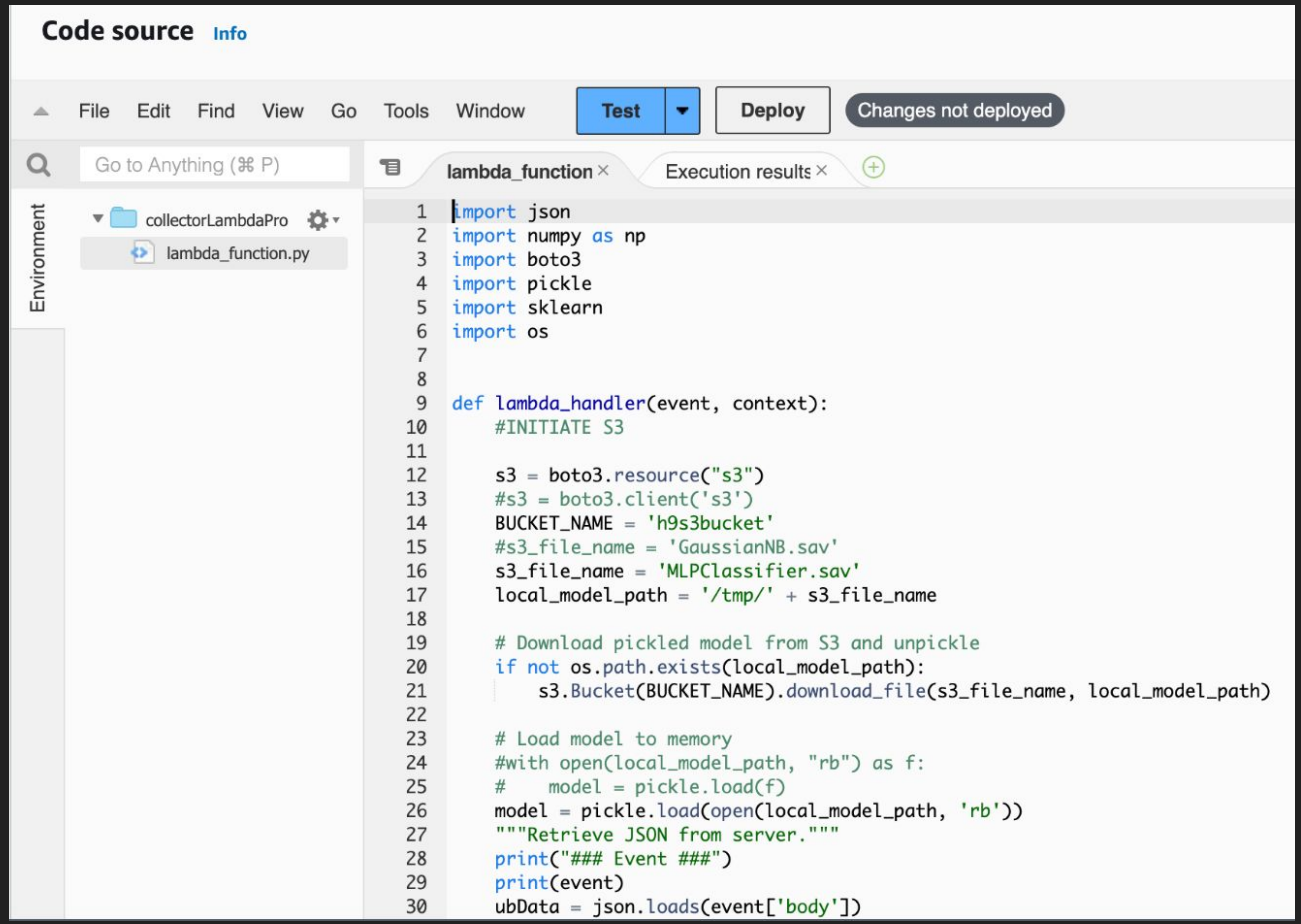
userbehavior.js



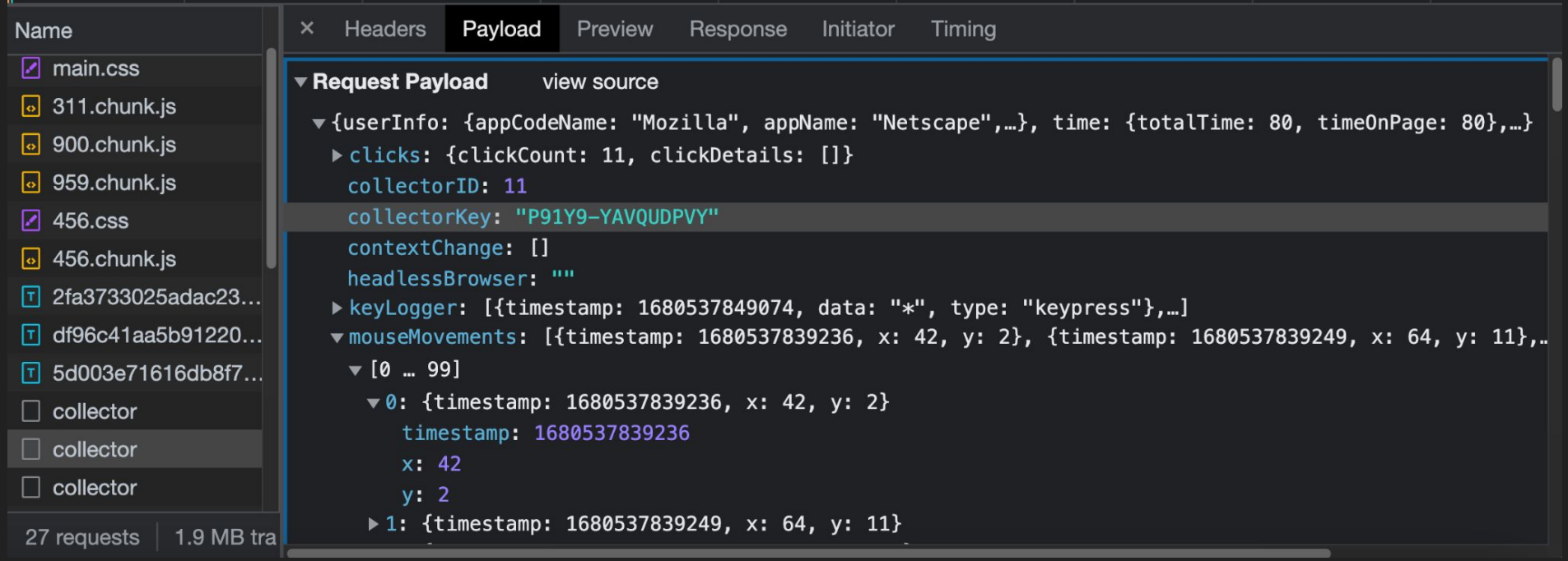
Collector API



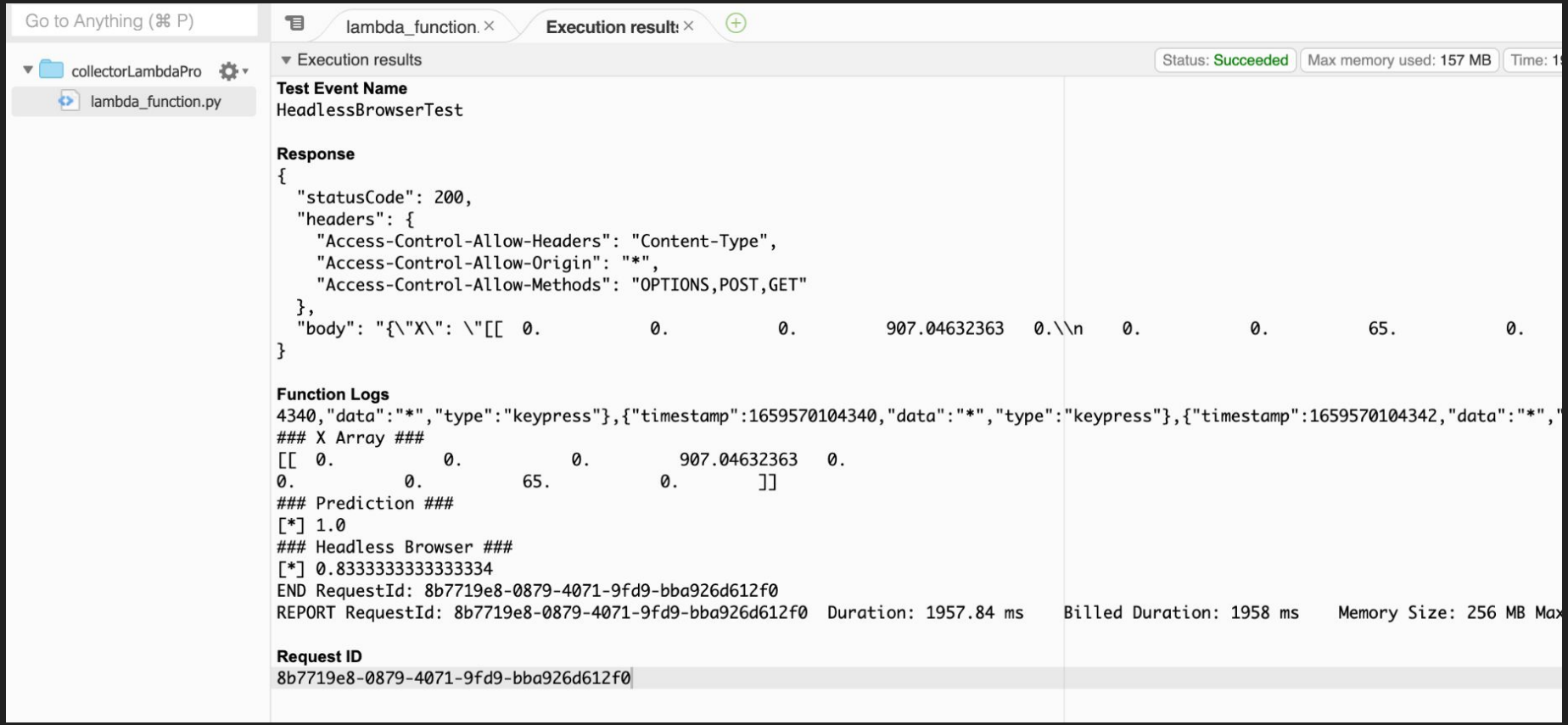
AWS Lambda Function



Collector API

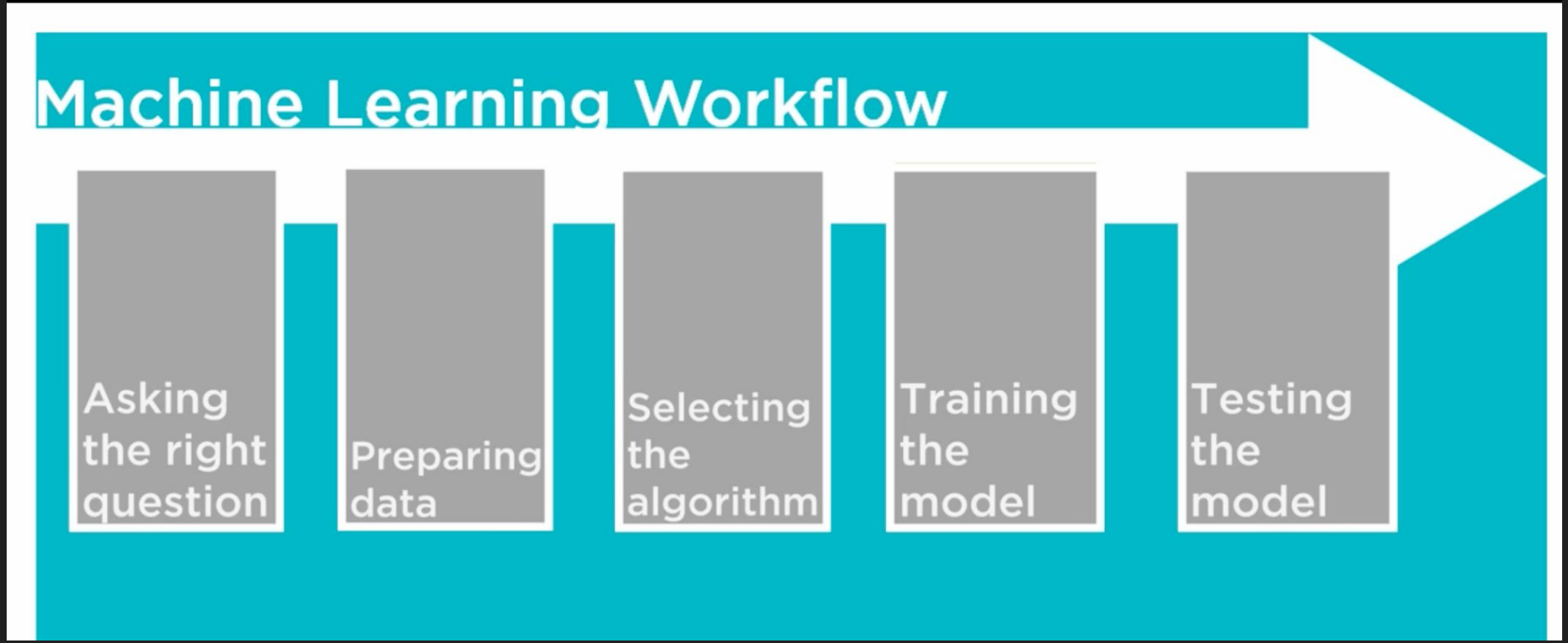


Prediction Based on User Behavior



CloudWatch Logs

▶	2023-04-02T11:12:45.670-05:00	### Event ###
▶	2023-04-02T11:12:45.671-05:00	{'resource': '/collector', 'path': '/collector', 'httpMethod': 'POST', 'headers': {'accept': '*/*', …
▶	2023-04-02T11:12:45.688-05:00	### X Array ###
▶	2023-04-02T11:12:45.688-05:00	[[1.00669472e+03 1.12617117e+03 5.42332865e+04 0.00000000e+00
▶	2023-04-02T11:12:45.688-05:00	1.11868191e+00 2.06272196e-05 0.00000000e+00 0.00000000e+00
▶	2023-04-02T11:12:45.688-05:00	0.00000000e+00]]
▶	2023-04-02T11:12:45.688-05:00	### Prediction ###
▶	2023-04-02T11:12:45.688-05:00	[*] 0.0
▶	2023-04-02T11:12:45.688-05:00	### Headless Browser ###
▶	2023-04-02T11:12:45.688-05:00	[*] 0

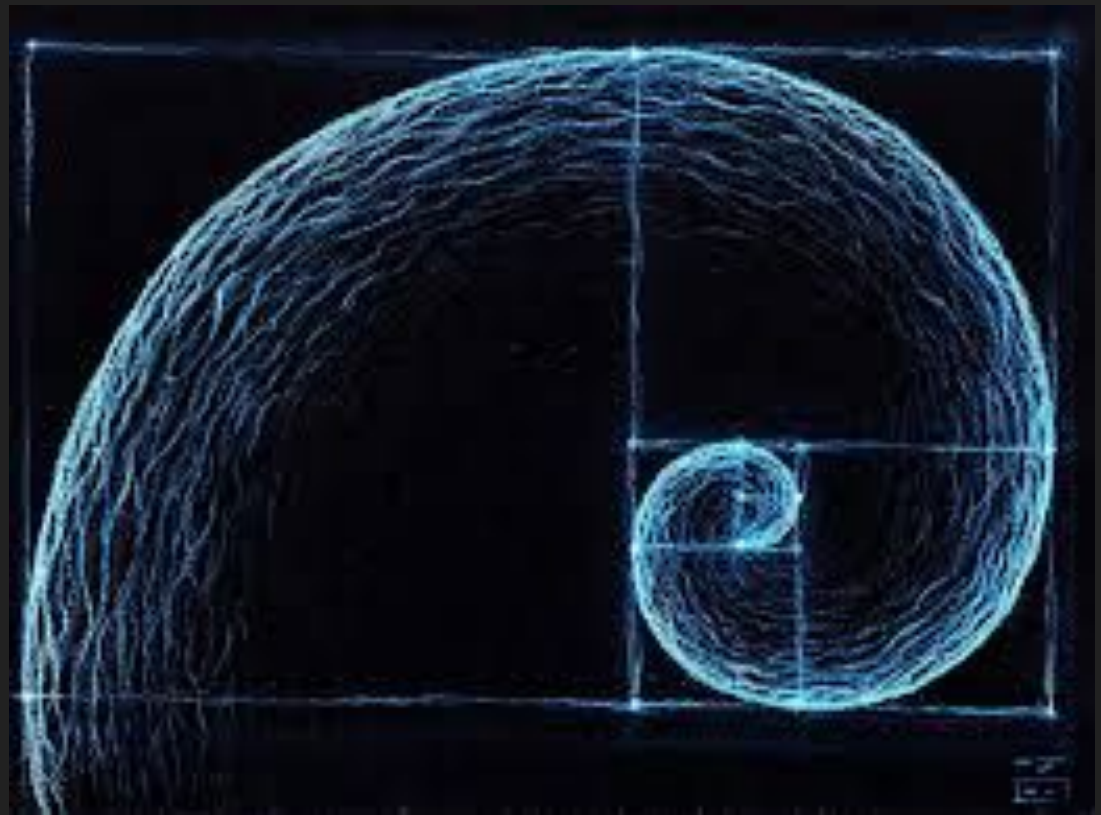

Machine Learning Workflow
Asking the right question
Preparing data
Selecting the algorithm
Training the model
Testing the model

Community Challenge

Is it feasible to have a web app security machine learning core rule set (MLCRS) model library, similar to the OWASP ModSecurity Core Rule Set and libinjection, that the community contributes to?

Questions?

https://github.com/obarrera/ML-XSS-Detection



Thank You for Attending