



# Testing Software with ML

Jason Arbon

1



STPCon: Testing Software with ML

2



## Takeaways

“I’m not intimidated by ML.”

“I have performed ML-based testing!”

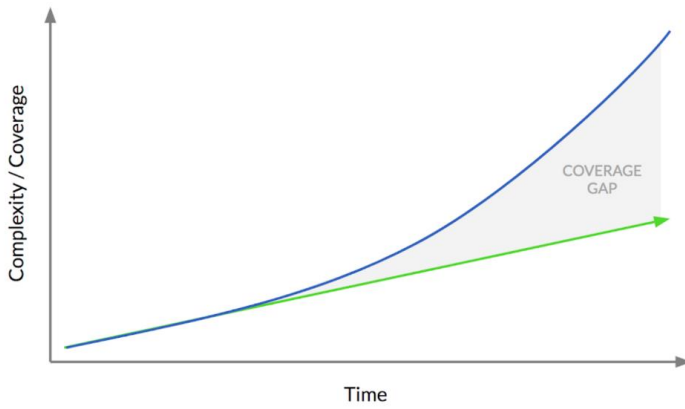
“I understand the limits and possibilities of ML for testing.”

3



Why ML?

4



**Features**

Complexity increases exponentially as new features and states combine

**Tests**

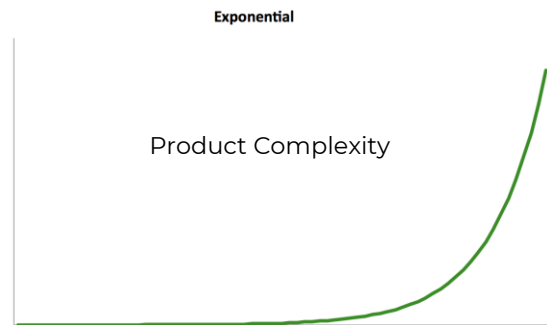
Test coverage grows linearly because tests can only be added one at a time

Desktop

Web

Mobile

Cloud



ML

Replicates Human Actions

Finally a tech to HELP Testing



STPCon: Testing Software with ML

7



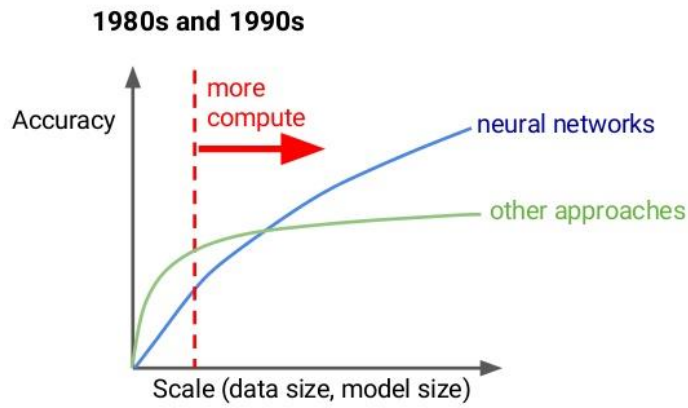
STPCon: Testing Software with ML

Machine learning is teaching  
computers to learn without  
having to program rules.



8

## Why Now?



--Jeff Dean, AI Frontiers

9

## Element Selection

10



11

## Human

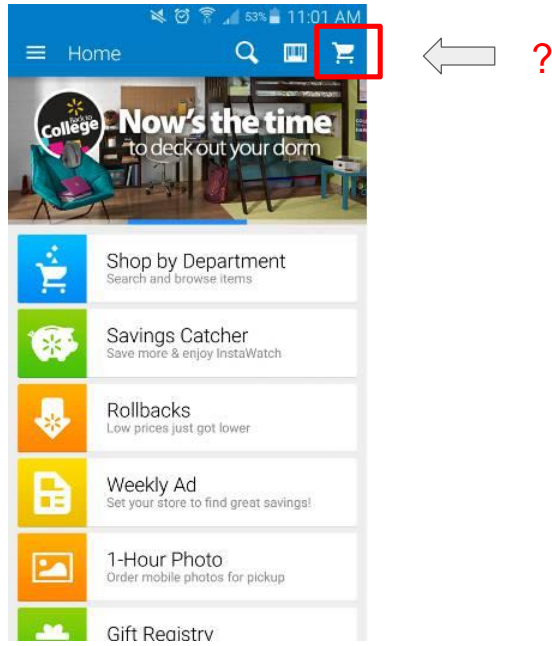
- Looks at Screen
- Sees all the 'Elements'
- Identifies elements by name/function
- Decides what to do next...



12

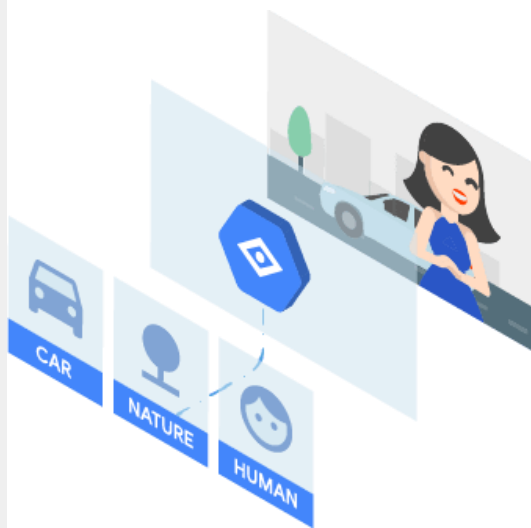
driver.find\_element\_by\_xpath('//\*[@id="main bar"]/div[1]/div/div[1]/div[3]/img')

Object  
Detection



13

Objects  
Detection



14



AutoML

<https://cloud.google.com/vision/>

Google Cloud Why Google Solutions Products Pricing Getting started

AI & Machine Learning Products [Contact sales](#)

### Powerful image analysis

Cloud Vision offers both pretrained models via an API and the ability to build custom models using AutoML Vision to provide flexibility depending on your use case.

Cloud Vision API enables developers to understand the content of an image by encapsulating powerful machine learning models in an easy-to-use REST API. It quickly classifies images into thousands of categories (such as, "selfie"), detects individual objects and faces within images, and reads printed words contained within images. You can build metadata on your image catalog, moderate offensive content, or enable new marketing scenarios through image sentiment analysis.

**AutoML Vision Beta** makes it possible for developers with limited machine learning expertise to train high-quality custom models. After uploading and labeling images, AutoML Vision will train a model that can scale as needed to adapt to demands. AutoML Vision offers higher model accuracy and faster time to create a production-ready model.

Try the API  Drag image file here or [Browse from your computer](#)

### Insight from your images

Easily detect broad sets of objects in your images, from flowers, animals, or transportation to thousands of other object categories commonly found within images. Vision API improves over time as new concepts are introduced and accuracy is improved. With AutoML Vision, you can create custom models that highlight specific concepts from your images. This enables use cases ranging from categorizing product images to

15

Object Detection\*

Label Detection

Web Locations

Optical Character Recognition

Handwriting Recognition

Logo Detection

Landmarks, Faces, Content Moderation

REST API



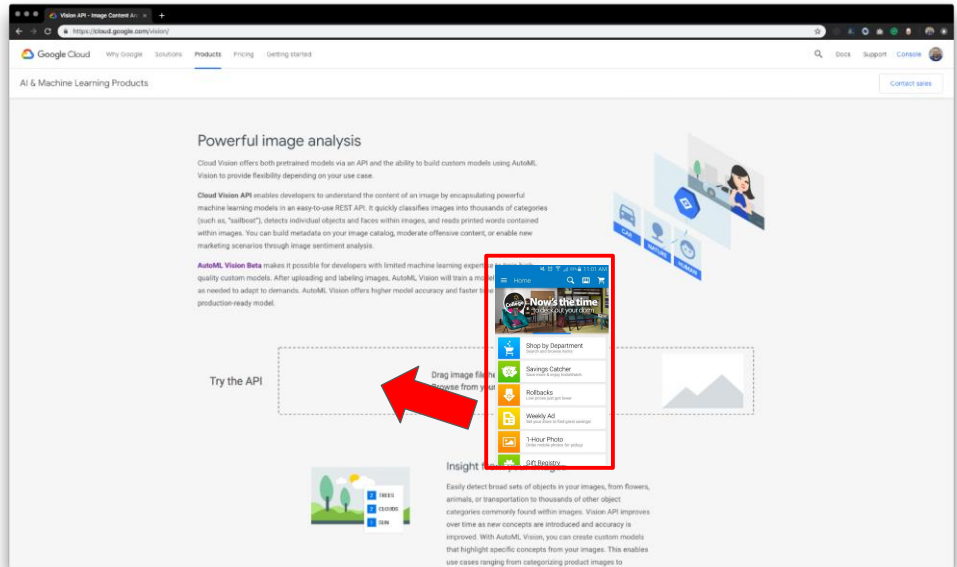
STPCon: Testing Software with ML

16



# Upload Screenshot

AutoML



17

Web

| Web | Document | Properties                       | Safe Search |
|-----|----------|----------------------------------|-------------|
|     |          |                                  |             |
|     |          | <b>Web Entities</b>              |             |
|     |          | Mobile app                       | 0.888       |
|     |          |                                  | 0.6196      |
|     |          | Walmart                          | 0.6129      |
|     |          | Shopping App                     | 0.6121      |
|     |          | Android                          | 0.5563      |
|     |          | iPhone                           | 0.54758     |
|     |          | Application software             | 0.537       |
|     |          | Web page                         | 0.44952     |
|     |          | Online advertising               | 0.39151     |
|     |          | Image                            | 0.3401      |
|     |          |                                  | 0.3192      |
|     |          | Shopping                         | 0.313       |
|     |          | Tablet Computers                 | 0.2982      |
|     |          | Gift                             | 0.295       |
|     |          | Mobile Phones                    | 0.12842     |
|     |          | <hr/>                            |             |
|     |          | <b>Pages with Matched Images</b> |             |

18

Questions

Are you surprised the ML figured out this was the Walmart app?

19

Questions

Are you surprised the ML realized this was a mobile app/screenshot?

20

Questions

Why do you think the ML Thinks this is 'Online Advertising'?

21

Questions

What did the ML miss?

22

Document

Web Document Properties Safe Search

Screen Shot 2019-03-29 at 9:06:22 PM.png

23

Questions

Are you impressed that the ML could find so many elements from just a Screenshot?  
similar to the human eye/brain?

24

Questions

What types of elements did the ML recognize that your brain did not?

25

Questions

What types of elements did the ML recognize that a typical code/DOM-based analysis might have missed?

26

Questions

What types of elements did the ML miss, and why do you think it missed them?

27

Questions

How could this data help “Manual Testing”?

28

Questions

Did the ML find the shopping cart as we'd hoped?

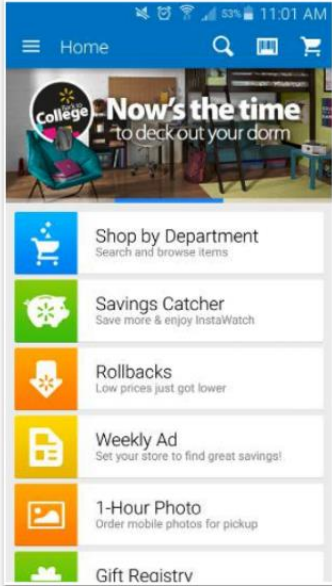
29

Questions

How could this ML response be useful in testing even if it didn't find the shopping cart, or 'every element'?

30

JSON

| Web   | Document | Properties   | Safe Search |
|---|----------|--|-------------|
|  <p>Screen Shot 2019-03-29 at 9.06.22 PM.png</p> |          | <pre> {   "labelAnnotations": [     {       "mid": "/m/07s6nbt",       "description": "Text",       "score": 0.9356654,       "topicality": 0.9356654     },     {       "mid": "/m/03gq5hm",       "description": "Font",       "score": 0.81951183,       "topicality": 0.81951183     },     {       "mid": "/m/03scnj",       "description": "Line",       "score": 0.72950023,       "topicality": 0.72950023     },     {       "mid": "/m/01zbnw",       "description": "Screenshot",       "score": 0.70882714,       "topicality": 0.70882714     },     {       "mid": "/m/086nh",       "description": "Web page",       "score": 0.5940073,       "topicality": 0.5940073     },     {       "mid": "/m/03bt1gh",       "description": "Games",       "score": 0.5211578,       "topicality": 0.5211578     }   ] } </pre> |             |

31

JSON

```

{
  "description": "Gift",
  "boundingPoly": {
    "vertices": [
      {
        "x": 95,
        "y": 590
      },
      {
        "x": 123,
        "y": 590
      },
      {
        "x": 123,
        "y": 609
      },
      {
        "x": 95,
        "y": 609
      }
    ]
  }
},
}

```

32



Safe Search

```
},  
  "safeSearchAnnotation": {  
    "adult": "VERY_UNLIKELY",  
    "spoof": "VERY_UNLIKELY",  
    "medical": "VERY_UNLIKELY",  
    "violence": "VERY_UNLIKELY",  
    "racy": "VERY_UNLIKELY"  
  },  
}
```

33

Questions

How could safe search help app testing?

34

# Google App

| Logos  | Web | Document | Properties  | Safe Sea |
|--|-----|----------|---|----------|
| <p>Screen Shot 2019-03-29 at 10.58.24 PM.png</p> |     |          | <p>+Page 1</p> <p>+Block 1</p> <p>Google 17 + Search news , images a nd videos Google LLC</p> <p>+Block 2</p> <p># 4 in Utilities * * * * 4 . 7 , 213 . 1 K Ratings</p> <p>+Block 3</p> <p>+Block 4</p> |          |

35

# Logos

| Logos  | Web | Document | Properties    | Sa |
|--|-----|----------|---------------|----|
| <p>Screen Shot 2019-03-29 at 10.58.24 PM.png</p> |     |          | <p>google</p> |    |

36

Web

|       |     |                 |            |             |
|-------|-----|-----------------|------------|-------------|
| Logos | Web | <b>Document</b> | Properties | Safe Search |
|-------|-----|-----------------|------------|-------------|

google\_home\_screen 2.png

37

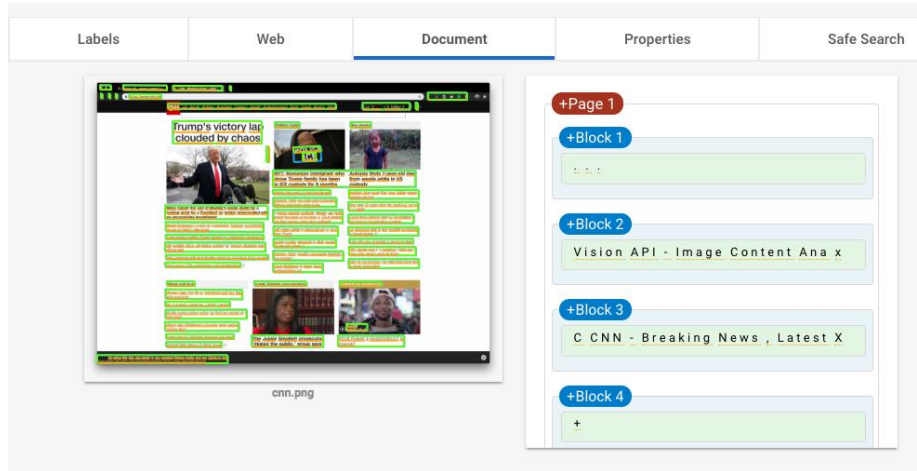
Facebook App

|      |       |     |                 |            |
|------|-------|-----|-----------------|------------|
| bels | Logos | Web | <b>Document</b> | Properties |
|------|-------|-----|-----------------|------------|

facebook\_app.png

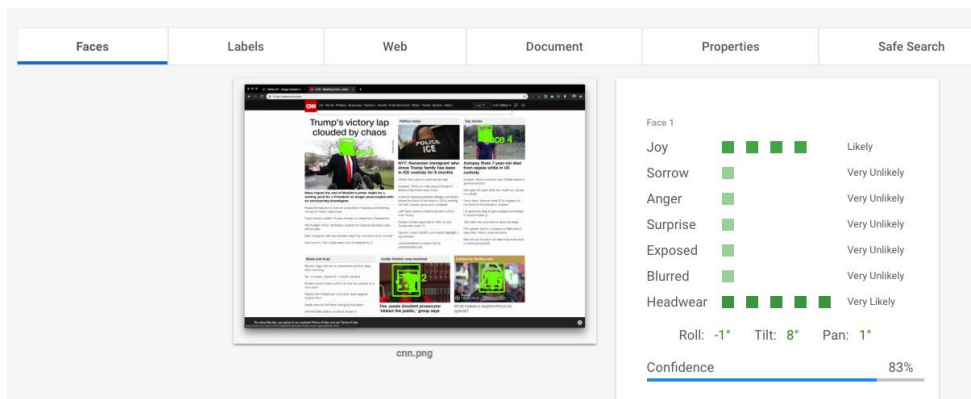
38

CNN



39

Face Detection



40

15 Min

<Your App Here>

41

Exercise

Share thoughts Applying ML to your app!

42



Still didn't find that shopping cart...  
Google can't find it--how can we?

43



We build our 'own' ML for testing

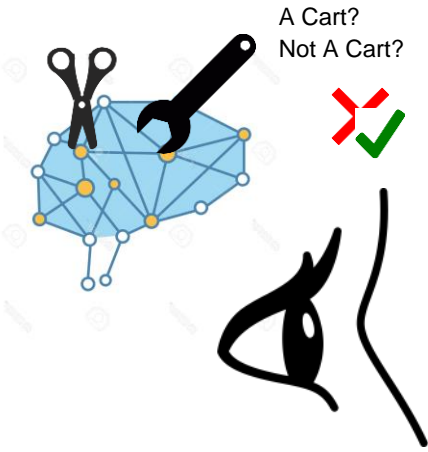
44

Train

Guess/Test

Features

Images

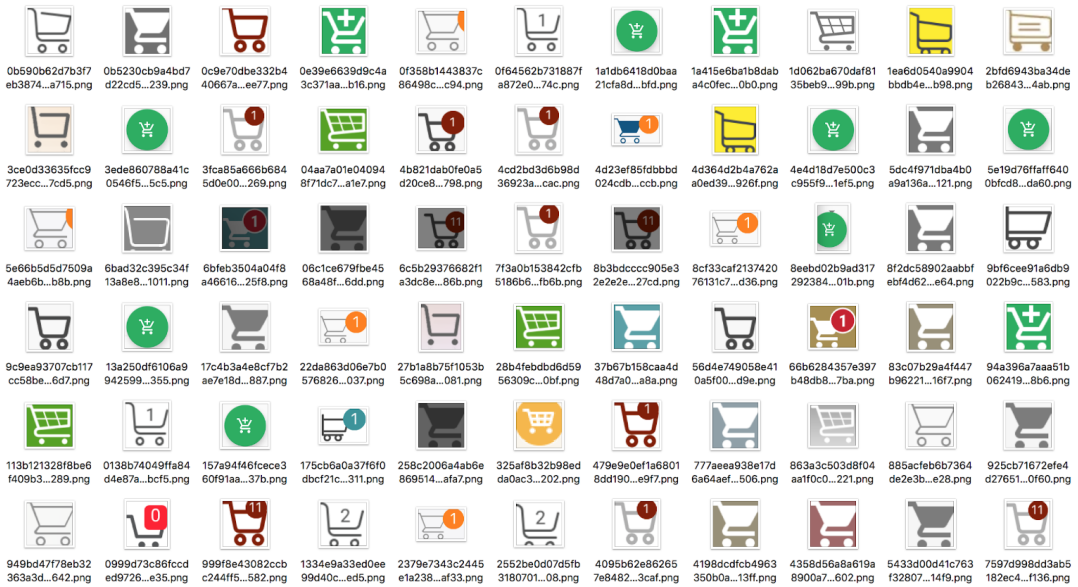


- ← AverageColor
- ← NumberOfEdges
- ← 128 X 128 Pixels



45

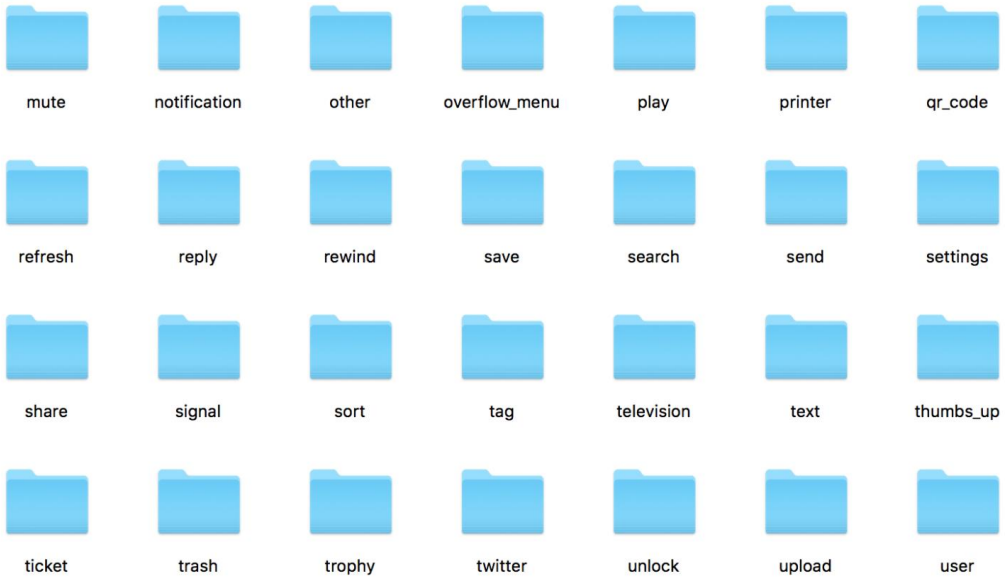
STPCon: Testing Software with ML

46



STPCon: Testing Software with ML



47

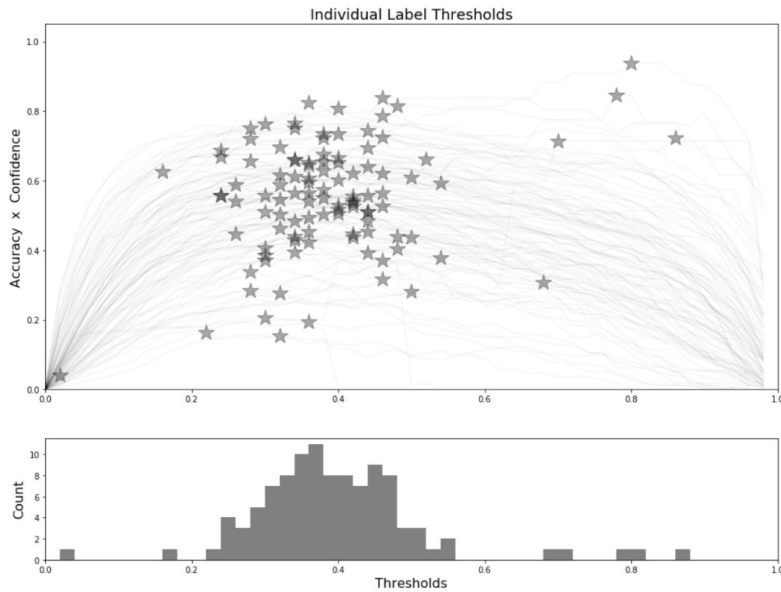
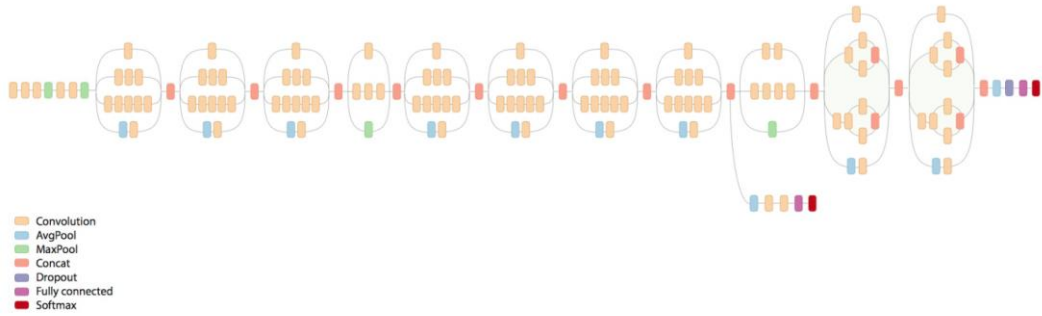


STPCon: Testing Software with ML



48





|                  | thresh | conf   | smm    |
|------------------|--------|--------|--------|
| fingerprint:     | 0.8000 | 1.0000 | 0.9375 |
| google:          | 0.7800 | 1.0000 | 0.8462 |
| fab:             | 0.4600 | 0.9000 | 0.9310 |
| barcode:         | 0.3600 | 0.8767 | 0.9412 |
| eye:             | 0.4800 | 0.9459 | 0.8621 |
| thumbs up:       | 0.4000 | 0.9478 | 0.8516 |
| trophy:          | 0.4600 | 0.9466 | 0.8322 |
| headphones:      | 0.3400 | 0.9213 | 0.8305 |
| car:             | 0.3000 | 0.9373 | 0.8154 |
| home:            | 0.2800 | 0.9405 | 0.8000 |
| calculator:      | 0.3400 | 0.8875 | 0.8458 |
| globe:           | 0.4400 | 0.9362 | 0.7942 |
| cart:            | 0.4000 | 0.9539 | 0.7708 |
| airplane:        | 0.2800 | 0.9388 | 0.7814 |
| facebook:        | 0.4600 | 0.8222 | 0.8810 |
| light bulb:      | 0.3800 | 0.9595 | 0.7532 |
| profile_picture: | 0.8600 | 0.8500 | 0.8500 |
| qr_code:         | 0.2800 | 0.8505 | 0.8476 |
| twitter:         | 0.7000 | 1.0000 | 0.7143 |
| camera:          | 0.3200 | 0.9419 | 0.7390 |
| ...              |        |        |        |
| rewind:          | 0.3600 | 0.7692 | 0.5511 |
| minimize:        | 0.3000 | 0.8000 | 0.5094 |
| sort:            | 0.4800 | 0.7108 | 0.5673 |
| fast forward:    | 0.3400 | 0.6092 | 0.6486 |
| download:        | 0.4600 | 0.6154 | 0.6371 |
| mute:            | 0.3000 | 0.7068 | 0.5446 |
| overflow_menu:   | 0.5400 | 0.6154 | 0.6154 |
| arrow_left:      | 0.3000 | 0.7746 | 0.4802 |
| attach:          | 0.4600 | 0.4332 | 0.8571 |
| reply:           | 0.2800 | 0.6300 | 0.5354 |
| maximize:        | 0.4600 | 0.6984 | 0.4536 |
| bag:             | 0.6800 | 0.7714 | 0.3971 |
| chart:           | 0.2800 | 0.6286 | 0.4531 |
| external_link:   | 0.5000 | 0.6667 | 0.4211 |
| alarm:           | 0.3200 | 0.6495 | 0.4254 |
| inbox:           | 0.3000 | 0.3902 | 0.5291 |
| delete:          | 0.3600 | 0.5350 | 0.3627 |
| arrow_right:     | 0.2200 | 0.3714 | 0.4375 |
| upload:          | 0.3200 | 0.2698 | 0.5690 |
| gmail:           | 0.0200 | 0.3333 | 0.1250 |



### Arrows



STPCon: Testing Software with ML

### Share Buttons

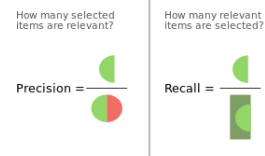
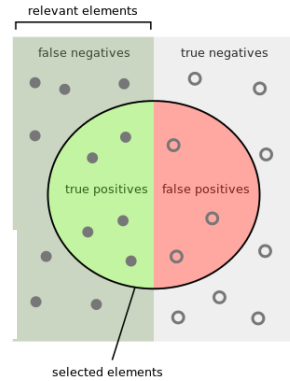


STPCon: Testing Software with ML



# F1 Score (thanks wikipedia)

$$F_1 = \left( \frac{\text{recall}^{-1} + \text{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$



## F1 at test.ai Today

Test the classifier on 1747 vapor validation samples: F1 score (97%).

|                          | precision | recall | f1-score | support |
|--------------------------|-----------|--------|----------|---------|
| _negative                | 0.85      | 0.85   | 0.85     | 48      |
| app_message_popup        | 0.99      | 0.95   | 0.97     | 78      |
| article                  | 0.96      | 0.91   | 0.94     | 176     |
| home                     | 0.99      | 0.98   | 0.99     | 524     |
| login                    | 0.98      | 0.99   | 0.99     | 187     |
| menu_drawer              | 1.00      | 0.99   | 1.00     | 106     |
| search                   | 0.99      | 0.95   | 0.97     | 149     |
| search_results           | 0.96      | 0.98   | 0.97     | 134     |
| settings                 | 1.00      | 1.00   | 1.00     | 76      |
| sign_in_up_options       | 1.00      | 1.00   | 1.00     | 64      |
| system_permissions_popup | 1.00      | 1.00   | 1.00     | 205     |
| micro avg                | 0.98      | 0.97   | 0.98     | 1747    |
| macro avg                | 0.97      | 0.96   | 0.97     | 1747    |
| weighted avg             | 0.98      | 0.97   | 0.98     | 1747    |
| samples avg              | 0.97      | 0.97   | 0.97     | 1747    |

Test the classifier on 76 log-in screenshots that Yash collected: Accuracy: 97%;

Only 2 misclassified as 'sign\_in\_up\_options' and 'unknown'.

```
[ 'login', 'login', 'login', 'login',
  'sign_in_up_options', 'login', 'login', 'login',
  'login', 'login', 'login', 'login', 'login', 'login',
  'login', 'login', 'login', 'login', 'login', 'login',
  'login', 'login', 'login', 'login', 'login', 'login',
  'login', 'login', 'login', 'login', 'login', 'login',
  'login', 'login', 'login', 'login', 'login', 'login',
  'login', 'login', 'login', 'login', 'login', 'unknown',
  'login', 'login', 'login', 'login', 'login', 'login',
  'login', 'login', 'login', 'login', 'login', 'login',
  'login', 'login', 'login', 'login', 'login', 'login',
  'login', 'login', 'login', 'login', 'login', 'login',
  'login', 'login', 'login', 'login', 'login', 'login',
  'login', 'login' ]
```



Now we can see shopping carts--and more!

55



Branch: master [appium-classifier-plugin / lib / labels.js](#)

 jlipps initial commit

1 contributor

111 lines (109 sloc) | 1.36 KB

```

1  const LABELS = [
2    " unclassified",
3    "add",
4    "airplane",
5    "alarm",
6    "arrow down",
7    "arrow left",
8    "arrow right",
9    "arrow up",
10   "attach",
11   "bag",
12   "barcode",
13   "battery",
14   "bluetooth",
15   "bookmark",
16   "brightness",
17   "calculator",
18   "calendar",
19   "call",
20   "camera",
21   "car",
22   "cart",
23   "chart",
24   "check mark",
25   "clock"

```

56

## Open Source: Code and Data

### Build Classifier

To build the classifier from the provided images run the following.

```
python retrain.py --image_dir training_images/ --output_graph output/saved_model.pb --output_labels
output/saved_model.pbtxt --how_many_training_steps 4000 --learning_rate 0.30 --testing_percentage 25 --
validation_percentage 25 --eval_step_interval 50 --train_batch_size 2000 --test_batch_size -1 --
validation_batch_size -1 --bottleneck_dir /tmp/bottleneck
```

### Run Classifier

To use the classifier to classify images, run the script under `sample_run/` directory.

```
python run_model.py cart.png
```



Your App's  
Elements?

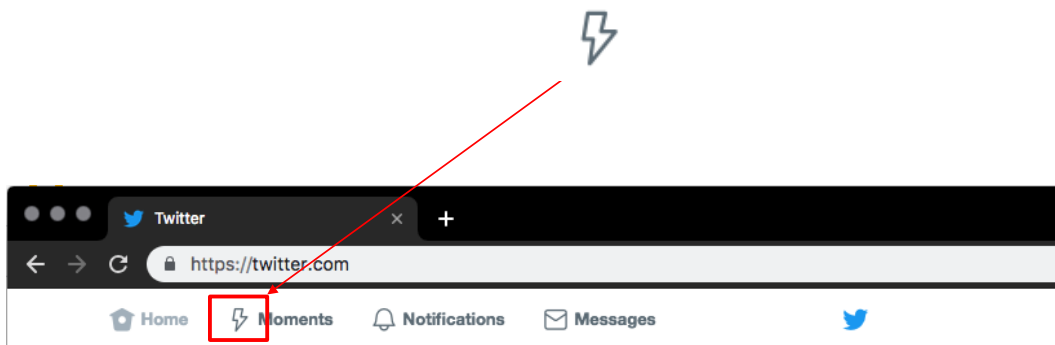
What does your app have that  
the open source classifier  
doesn't?

Your App's  
Elements?

1. Find An Element in your app that isn't in the standard list.
2. Get an image of this from your app.
3. Get similar images on the web

59

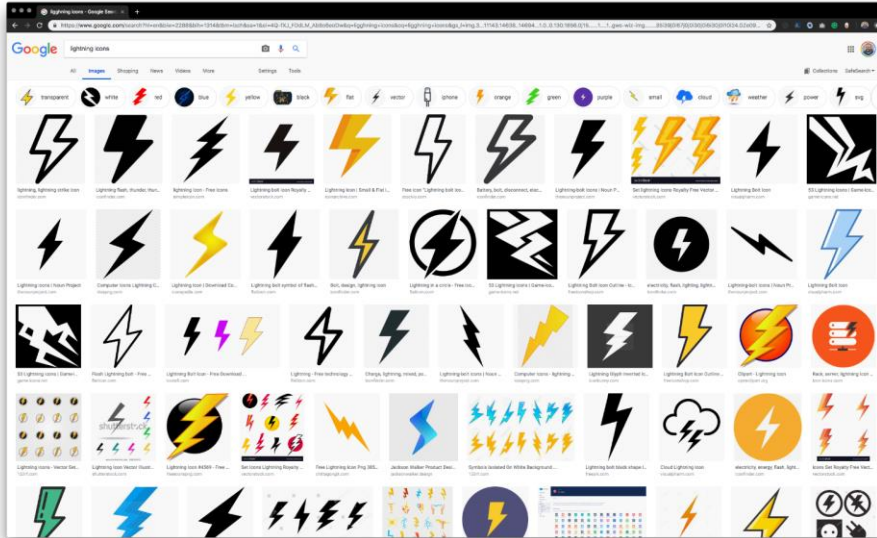
Add: "Moments/Lighting" Button



60



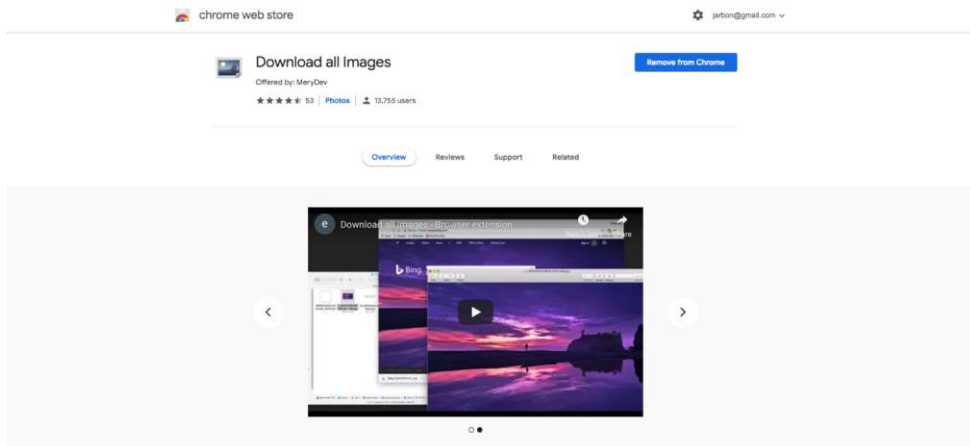
# Search for Similar Icons on Web



STPCon: Testing Software with ML

61

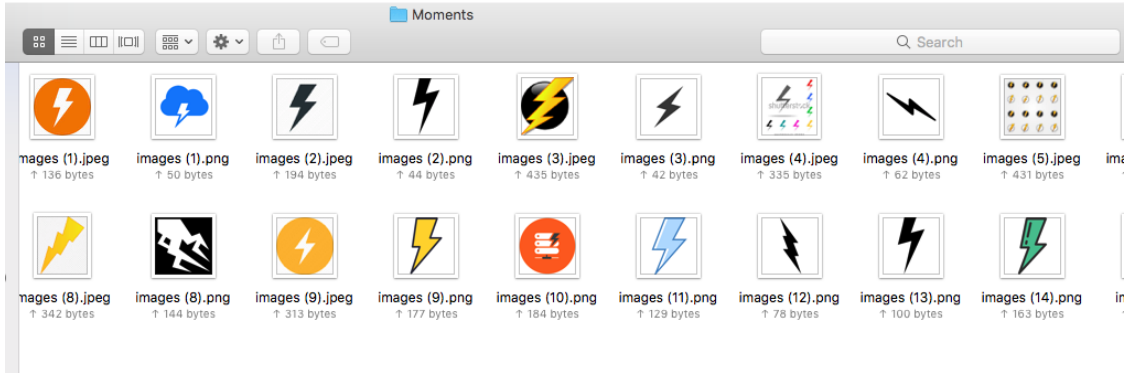
# Tip: Extension to Download All



STPCon: Testing Software with ML

62

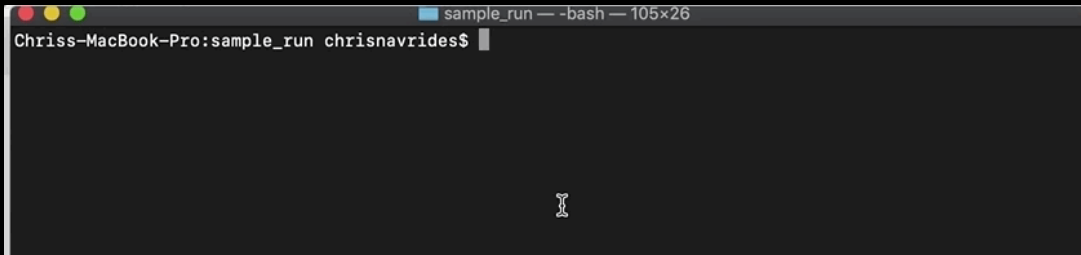
## Add folder called "Moments" with Images



Later :)







```

sample_run — -bash — 105x26
Chriss-MacBook-Pro:sample_run chrisnavrides$

```

65

## Retrain the ML

### Build Classifier

To build the classifier from the provided images run the following.

```
python retrain.py --image_dir training_images/ --output_graph output/saved_model.pb --output_labels
output/saved_model.pbtxt --how_many_training_steps 4000 --learning_rate 0.30 --testing_percentage 25 --
validation_percentage 25 --eval_step_interval 50 --train_batch_size 2000 --test_batch_size -1 --
validation_batch_size -1 --bottleneck_dir /tmp/bottleneck
```

### Run Classifier

To use the classifier to classify images, run the script under `sample_run/` directory.

```
python run_model.py --image cart.png
```



66





# Performance Analysis



# K-Means Clustering

## K Means

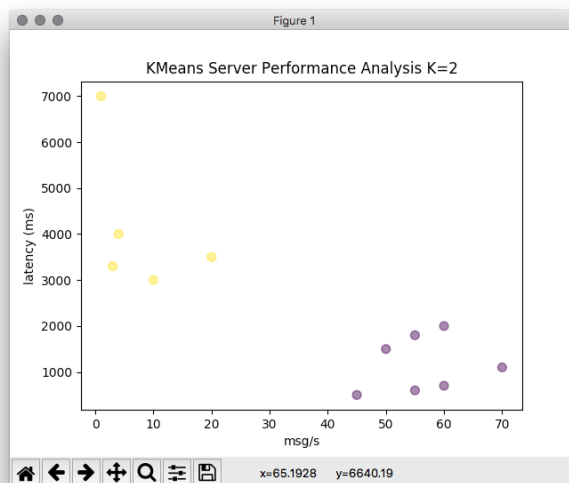
```

kmeans.py
#Jason Arbon
# Quick example of using Kmeans to identify servers
# with different performance characteristics
4
5 from pandas import DataFrame
6 import matplotlib.pyplot as plt
7 from sklearn.cluster import KMeans
8
9 Data = {'msg/s': [ 60, 55, 50, 60, 70, 45, 55, 10, 20, 1, 4, 3],
10        'latency': [ 2000, 1800, 1500, 700, 1100, 500, 600, 3000, 3500, 7000, 4000, 3300]}
11
12
13 def plot(Data, k):
14     df = DataFrame(Data, columns=['msg/s', 'latency'])
15
16     kmeans = KMeans(n_clusters=k).fit(df)
17     plt.scatter(df['msg/s'], df['latency'], c= kmeans.labels_.astype(float), s=50, alpha=0.5)
18     print(kmeans.labels_.astype(float))
19     plt.title('KMeans Server Performance Analysis K=' + str(k))
20     plt.ylabel('latency (ms)')
21     plt.xlabel('msg/s')
22     plt.show()
23
24 plot(Data, 2)
25 plot(Data, 3)
26
Line: 15/3 | Python | Soft Tabs: 2 | plot(Data, k)

```

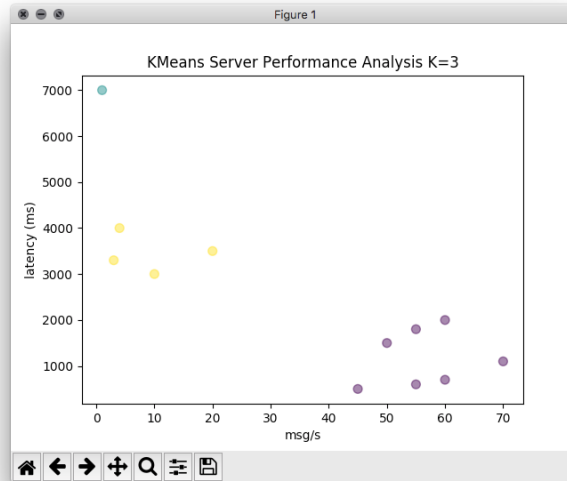
71

## Auto Identify Differently Behaving Servers



72

## Auto Identify Differently Behaving Servers



73

Questions

What data might be interesting to cluster on your team?

Perf  
Stress  
Crash logs  
Test Results

74



STPCon: Testing Software with ML

75

# Your Testing Problems AI Applicable?

jason@test.ai

Appdiff, Inc. 40 Brannan St #100, San Francisco, CA 94107, United States

Contents of this presentation are proprietary and confidential. The presentation, or any elements of the presentation may not be duplicated, distributed, shared or used in any way other than for purposes of review, while under NDA. All contents property of Appdiff Inc. unless otherwise assigned to another party. Copyright © Appdiff Inc, 2019.

76



STPCon: Testing Software with ML

# Test Generation