Stringlifier

Or how to simplify your data science life with one tool

whoami

Adobe Security Intelligence Team

- Tiberiu Boros
  - Data Scientist / Machine Learning Engineer
  - boros@adobe.com
- Andrei Cotaie
  - Technical Lead - Security Engineer
  - cotaie@adobe.com
- Kumar Vikramjeet
  - Security Engineer
  - vikrakum@adobe.com
- Vivek Malik
  - Incident Responder
  - vivmalik@adobe.com

What's Stringlifier?!

Why Stringlifier?

LIFE IS… NOT NUMERICAL

TEXT IS EVERYWHERE

TEXT IS NOT ALWAYS WHAT YOU EXPECT IT TO BE

Anomaly Detection

Principle

```
s6-svscan -t0 /var/run/s6/services
```

```
/sbin/udevd -d
```

```
containerd-shim -namespace moby -workdir
/mnt/data/docker/root/666666.666666/containerd/daemon/io.containerd.runtime.v1.linux/moby/874ea4a3be6a4cf14f886931b97262bf8a52e91a4af599d8f679c18045fba358 -address
/var/run/docker/containerd/containerd.sock -containerd-binary /usr/bin/containerd -runtime-root /var/run/docker/runtime-runc
```

```
python3 /usr/bin/splunkfuscator
```

```
/bin/bash -c ulimit -S -c 0 >/dev/null 2>&1 ; nohup /usr/bin/splunkfuscator
```

Anomaly Detection - data

/bin/bash /opt/octopus/worker-addins/video-splitting/frameExtraction.sh /tmp/karaf_data/tmp/Tentakel-3-5061908559013761025.octo/f99829d5-87b7-4543-b034-7fe55a725463.mp4 /tmp/karaf_data/tmp/Tentakel-3-5061908559013761025.octo f99829d5-87b7-4543-b034-7fe55a725463 f99829d5-87b7-4543-b034-7fe55a725463_timestamps.txt 5 0.1

/bin/bash /opt/octopus/worker-addins/video-splitting/frameExtraction.sh /tmp/karaf_data/tmp/Tentakel-3-4893939540994376643.octo/295d7e0b0b3507b78afbccfbbc24fafa /tmp/karaf_data/tmp/Tentakel-3-4893939540994376643.octo 295d7e0b0b3507b78afbccfbbc24fafa 295d7e0b0b3507b78afbccfbbc24fafa_timestamps.txt 5 0.1

https://github.com/hubblestack

The Engineering solution


```python
import re

def command_cleaner(x):
    regex = re.compile('((?!_)[\w\d\-]){30,100}')
    return regex.sub('<RANDOMSTRING>', x)

```

```python
string="""/bin/bash /opt/octopus/worker-addins/video-spl
action.sh /tmp/karaf_data/tmp/Tentakel-3-506190855901376
/f99829d5-87b7-4543-b034-7fe55a725463.mp4
/tmp/karaf_data/tmp/Tentakel-3-5061908559013761025.octo
f99829d5-87b7-4543-b034-7fe55a725463 f99829d5-87b7-4543-
```

```python
command_cleaner(string)
```

```
'/bin/bash /opt/octopus/worker-addins/video-splitting/frameE
/tmp/karaf_data/tmp/<RANDOMSTRING>.octo/<RANDOMSTRING>.mp4 /
a/tmp/<RANDOMSTRING>.octo <RANDOMSTRING> <RANDOMSTRING>_time
0.1'
```

```python
import re
def command_cleaner(x):
    regex = re.compile('((?!_)[\w\d\-]){30,100}')
    return regex.sub('<RANDOMSTRING>', x)

```

```python
list_of_paths=[
    '/47259192097be0fe6b9dae0cfa0648a4788ee13bc01ee6a2026c361fdffe7a0f/layer.tar',
    '/config01/data/collection-29--1284350413225293969.wt',
    '/cgroup/memory/docker/3c3ce2e8478c683c129865ab962038cf12a981d48aec4c4ddb4bdea191337a0b/memory.stat',
    '/var/tmp/etilqs_fffa1f2557b44647 (deleted)',
    '/config01/data/diagnostic.data/metrics.2020-07-25T14-53-19Z-00000',
    '/tmp/octopus/logs/Flite-1938.log',
    '/bin/prometheus-config-reloader'
      ]
```

```python
for path in list_of_paths:
    print(command_cleaner(path))
```

```
/<RANDOMSTRING>/layer.tar    <-------- OK
/config01/data/<RANDOMSTRING>.wt    <-------- OK
/cgroup/memory/docker/<RANDOMSTRING>/memory.stat    <-------- OK
/var/tmp/etilqs_fffa1f2557b44647 (deleted)    <-------- FAIL
/config01/data/diagnostic.data/metrics.2020-07-25T14-53-19Z-00000    <-------- FAIL
/tmp/octopus/logs/Flite-1938.log    <-------- FAIL
/bin/prometheus-config-reloader    <-------- OK
```

Here was regex…

```python
import re
def command_cleaner(x):
    regex = re.compile('((?!_)[\w\d\-]){20,100}')
    return regex.sub('<RANDOMSTRING>', x)

```

```python
list_of_paths=[
    '/47259192097be0fe6b9dae0cfa0648a4788ee13bc01ee6a2026c361fdffe7a0f/layer.tar',
    '/config01/data/collection-29--1284350413225293969.wt',
    '/cgroup/memory/docker/3c3ce2e8478c683c129865ab962038cf12a981d48aec4c4ddb4bdea191337a0b/memory.stat',
    '/var/tmp/etilqs_fffa1f2557b44647 (deleted)',
    '/config01/data/diagnostic.data/metrics.2020-07-25T14-53-19Z-00000',
    '/tmp/octopus/logs/Flite-1938.log',
    '/bin/prometheus-config-reloader'
      ]
```

```python
for path in list_of_paths:
    print(command_cleaner(path))
```

```
/<RANDOMSTRING>/layer.tar        <-------- OK
/config01/data/<RANDOMSTRING>.wt      <-------- OK
/cgroup/memory/docker/<RANDOMSTRING>/memory.stat      <-------- OK
/var/tmp/etilqs_fffa1f2557b44647 (deleted)      <-------- FAIL
/config01/data/diagnostic.data/metrics.<RANDOMSTRING>      <-------- OK
/tmp/octopus/logs/Flite-1938.log      <-------- FAIL
/bin/<RANDOMSTRING>      <-------- TOTAL FAIL
```

Prometheus has left the building

And problems just started

- IPs
- IDs
- Passwords/Keys
- Timestamps

And now Stringlifier

Raw text input

Detects:

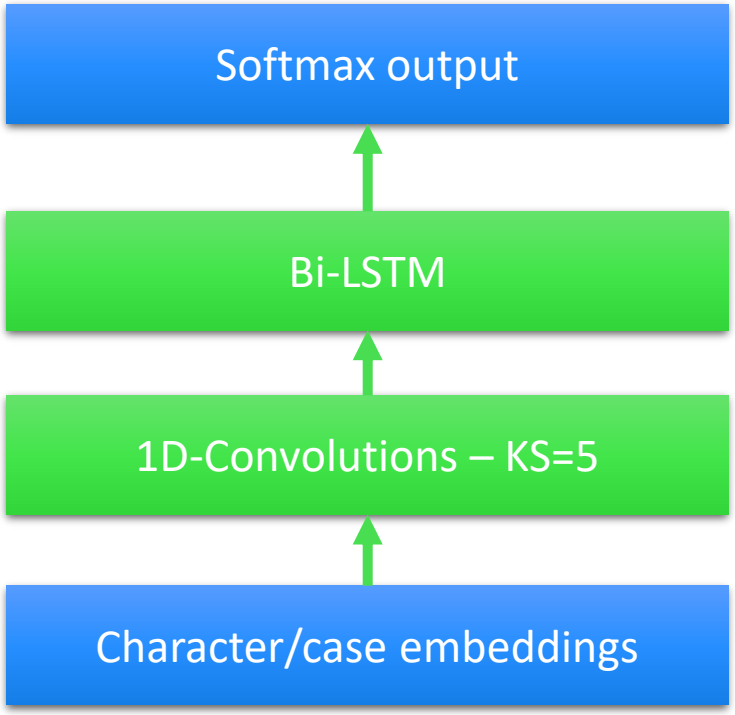
- Random strings
- Numbers
- IPs
- UUIDs

Open-sourced

PIP installation package

Stringlifier as a ML model

- Model
  - Character level tagging
  - Detect continuous labels at runtime
- Training
  - Synthetic generation of data
  - Use a list of 400K common English words
  - Generate numbers, random strings and UUIDS
  - Simulate commands and parameters



How to use Stringlifier

```
$ pip install stringlifier
```

```
from stringlifier.api import Stringlifier
stringlifier = Stringlifier()
s = stringlifier("com.docker.hyperkit -A -u -F vms/0/hyperkit.pid -c 8 -m 8192M -b 127.0.0.1
--pass=\"NlcXVpYWRvcg\" -s 0:0,hostbridge -s 31,lpc -s 1:0,virtio-
vpnkit,path=vpnkit.eth.sock,uuid=45172425-08d1-41ec-9d13-437481803412 -U c6fb5010-a83e-4f74-
9a5a-50d9086b9", return_tokens = True)
```

```
(['com.docker.hyperkit -A -u -F vms/0/hyperkit.pid -c 8 -m 8192M -b <IP_ADDR> --pass="<RANDOM_STRING>" -
s 0:0,hostbridge -s 31,lpc -s 1:0,virtio-vpnkit,path=vpnkit.eth.sock,uuid=<UUID> -U <UUID> A’],

 [[('127.0.0.1', 65, 74, '<IP_ADDR>'),
   ('NlcXVpYWRvcg', 83, 95, '<RANDOM_STRING>'),
   ('45172425-08d1-41ec-9d13-437481803412', 172, 208, '<UUID>'),
   ('c6fb5010-a83e-4f74-9a5a-50d9086b9', 212, 245, '<UUID>')]])
```

Stringlifier Demo

jupyter stringlifier-demo Last Checkpoint: 3 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

In [ ]:
```
import pandas as pd
import sys

sys.path.insert(1, 'stringlifier')
from stringlifier.api import Stringlifier

stringlifier = Stringlifier()
```

Use Case: Sanitize commands for DS/ML implementation

In [ ]:
```
# command df example
commands = ["com.docker.hyperkit -A -u -F vms/0/hyperkit.pid -c 8 -m 8192M -b 127.0.0.1 --pass=\"NlcXVxYWRvcg\" -s 0:0,
            "ssh -i ~/.ssh/id_rsa user1@w15550s0454q6774.store.repo.com",
            'bash -c source /usr/local/neo/6/env.sh; nserver javascript -instance:default -file /usr/local/neo/wi/lib/
            "iptables -A OUTPUT -p tcp -d 192.168.100.0/24 --dport 22 -j ACCEPT"]
df = pd.DataFrame(columns=['commands'], data=commands)
```

In [ ]:
```
display(df)
```

In [ ]:
```
def exec_stringlifier(text):
    s_out, tokens = stringlifier(text, return_tokens=True)
    return s_out[0]
```

In [ ]:
```
# santized commands feature
df['commands_clean'] = df['commands'].apply(exec_stringlifier)
```

In [ ]:
```
df.commands.to_list()
```

In [ ]:
```
df.commands_clean.to_list()
```

Use Case: Detect certain strings (secrets, keys, etc) in text (repo/folder)

In [ ]:
```
import os
import tqdm
```

Stringlifier use cases

Build a synthetic dataset

Advantages

- We control how the data looks like
- We know the number of classes
- We know the class of each datapoint

Disadvantages

- Not real data
- Biased
- You could achieve our goal just by using and if/else statement

Run unsupervised clustering

Pre-processing

- Regex cleanup
- Stringlifier cleanup

Post-processing

- Tokenization
- TF-IDF
- Dimensionality reduction
- Unsupervised clusters

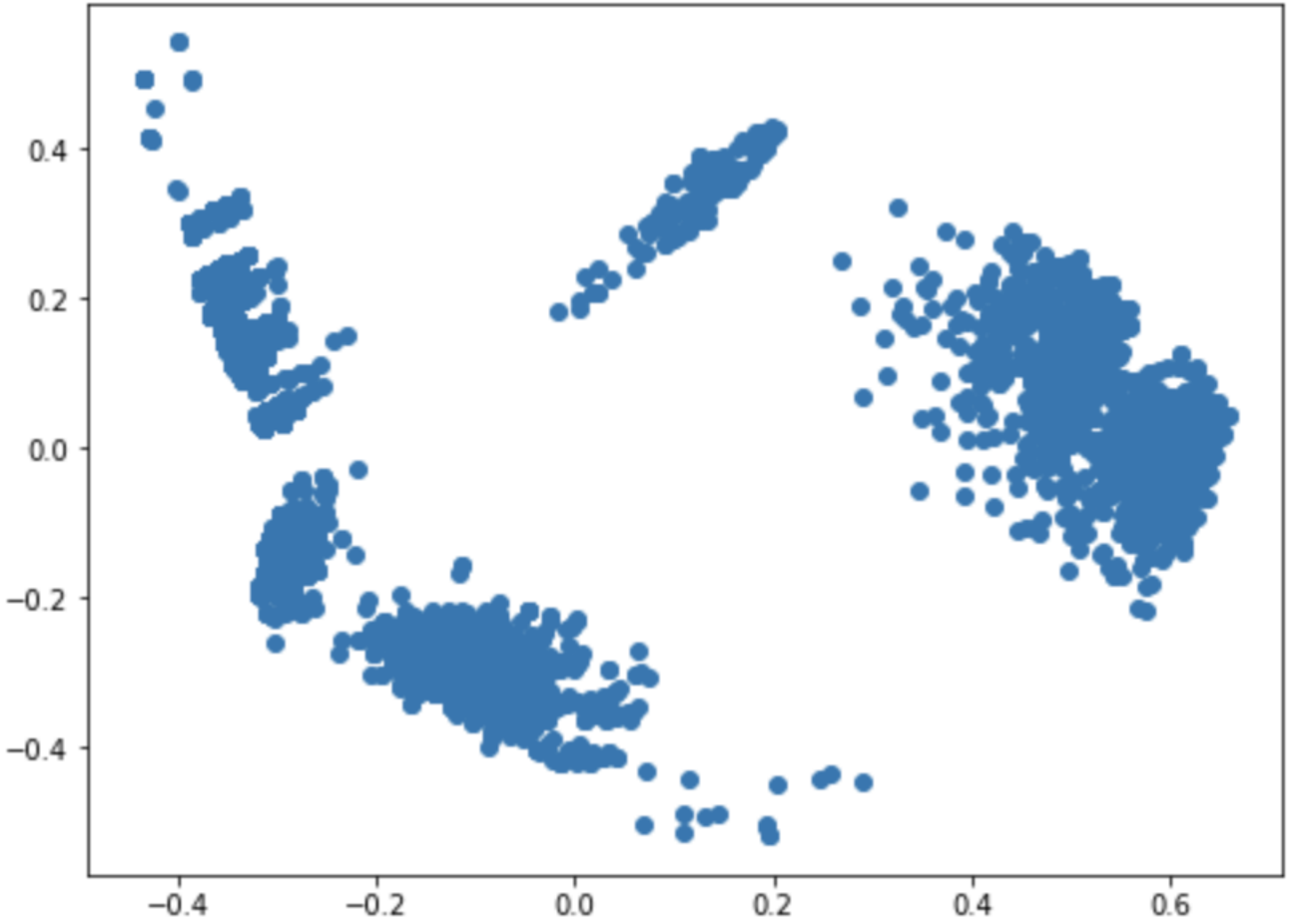
Compare results

See how well the automatically generated clusters compare to the original ones

Check if the distribution of data has changed

Let’s look at the data

And then talk about how we generated it



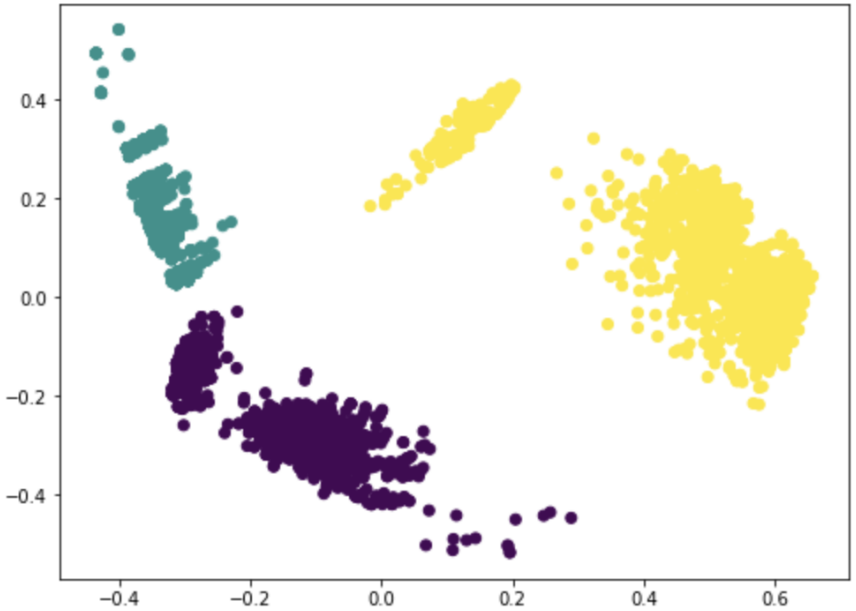
Simple cleanup

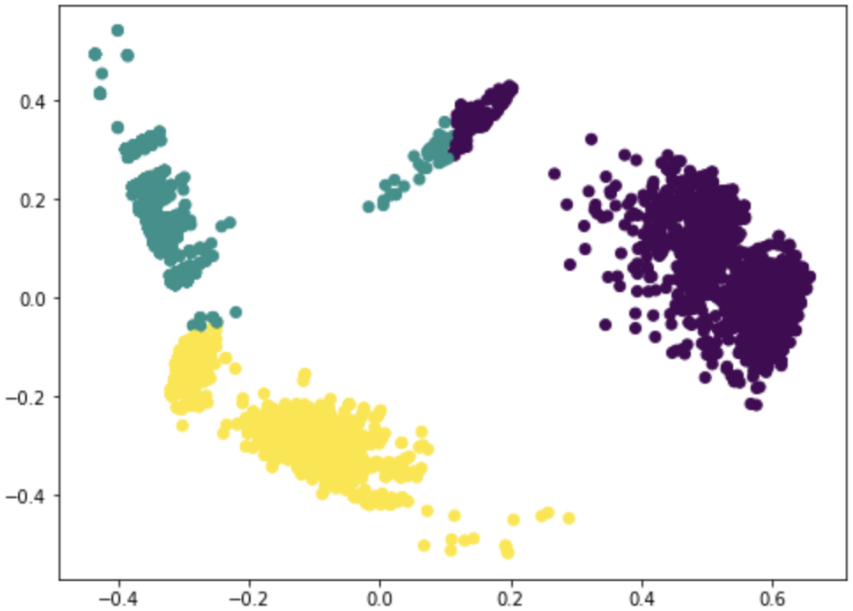
Stringlifier synthetic dataset

- The dataset:
  - 3 almost fictional commands:
    - /usr/bin/ffmpeg
    - /usr/local/bin/mpg321
    - /usr/bin/iptables
  - Each command has its own parameters (some of them overlap):
    - --input-file
    - --output-file
    - --salt
    - --codec
    - --username
    - --password
    - …
  - We use generation rules for parameters:
    - RANDOM, UUID, IP, from list etc.

Examples

- /usr/bin/iptables -j=SNAT --to-destination --dst=22.19.221.235 --dport=74767 -p udp
- /usr/bin/iptables -A postrouting --to-source --src=129.138.185.150 --dport=79234 -p tcp
- /usr/bin/ffmpeg --input-file=tmp-Lmo4uD --output-file=tmp-lBmL27tX8q3 --offset=95281 --server-address=129.153.1.208 --salt=1f7a9828bc4c469695e294963672f7f2 --codec=h264 --username=admin2 --password=tmp-OQHKNFF6G6UIBZZGYI
- /usr/bin/iptables -A prerouting -j=SNAT --to-source --dst=241.113.132.103 --sport=40210 -p tcp
- /usr/bin/ffmpeg --input-file=tmp-vcwkxm --output-file=tmp-E8FPPRECS --server-address=170.31.251.11 --salt=0123c359a4234d6f9eee4c4e1b6c3882 --username=admin1 --password=tmp-pW3UHUT
- /usr/bin/ffmpeg --input-file=tmp-VRHH06W5O --offset=70852 --server-address=117.5.180.1 --username=admin1 --password=tmp-P9OS4NWMZRI
- /usr/local/bin/mpg321 --input=tmp-WTPEBRJ --output=tmp-kwslsx5yakemaimv2 --trim-start=88546 --trim-end=55387 --username=admin4 --password=tmp-BATLM6ZBZYFUD662MLS5
- /usr/bin/ffmpeg --input-file=tmp-IGXNG8YMGOOWK8QXDIGN --output-file=tmp-FA6xGQ0 --offset=22440 --server-address=205.248.165.143 --codec=h264 --password=tmp-ccsv6wo
- /usr/bin/iptables -p tcp
- /usr/bin/iptables --to-destination --dst=121.222.36.88 --dport=80462 -p tcp

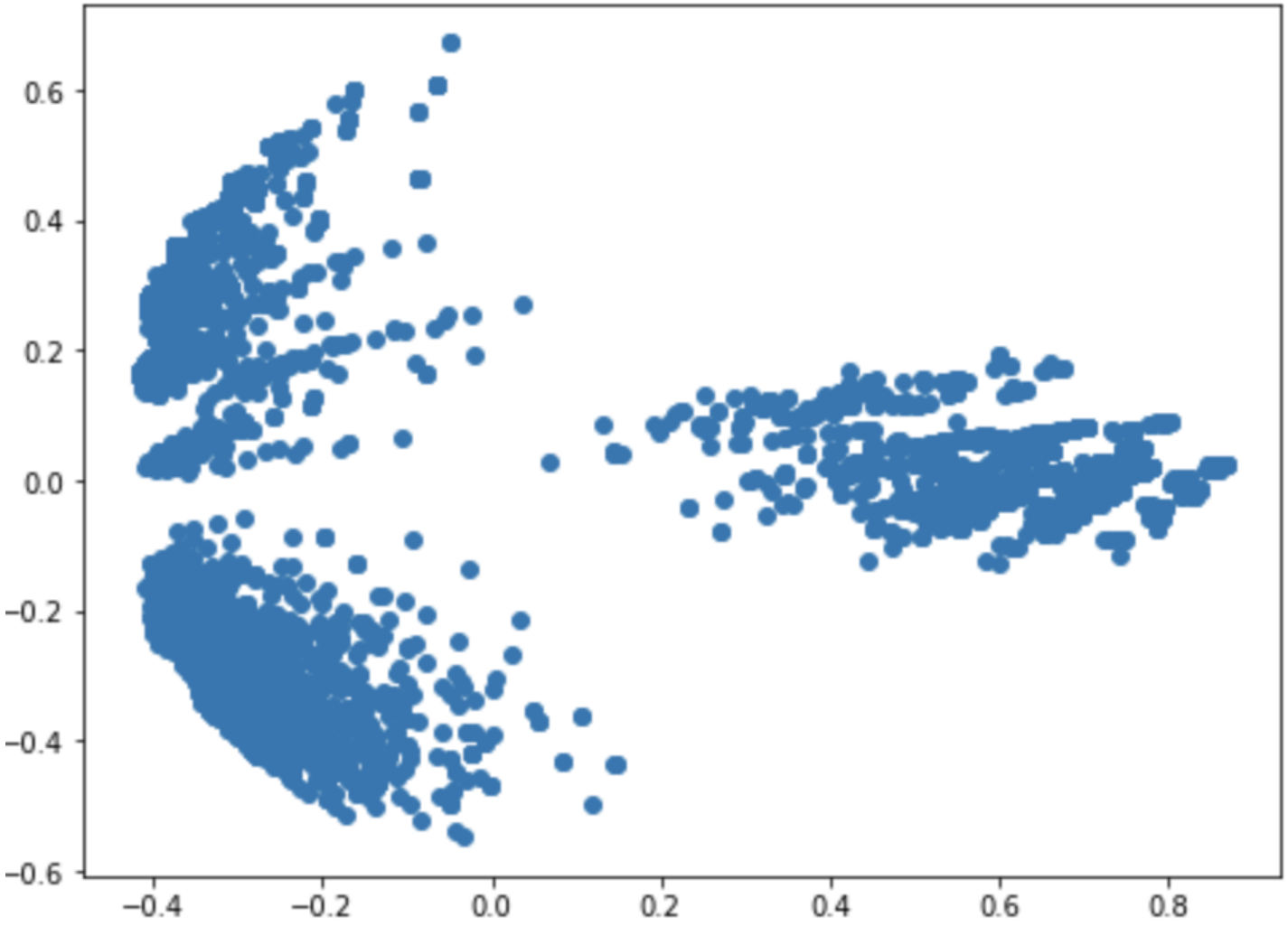




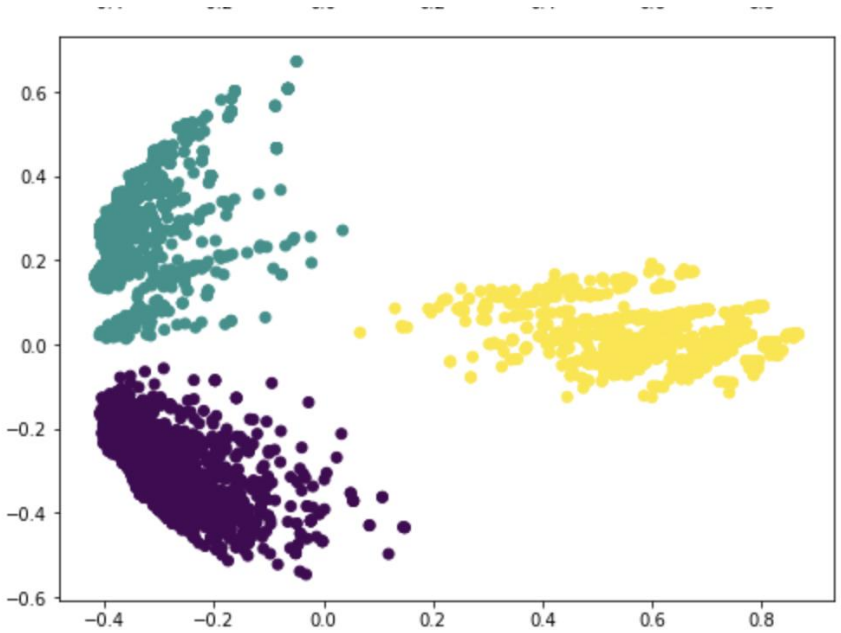
Actual vs. automatic clusters

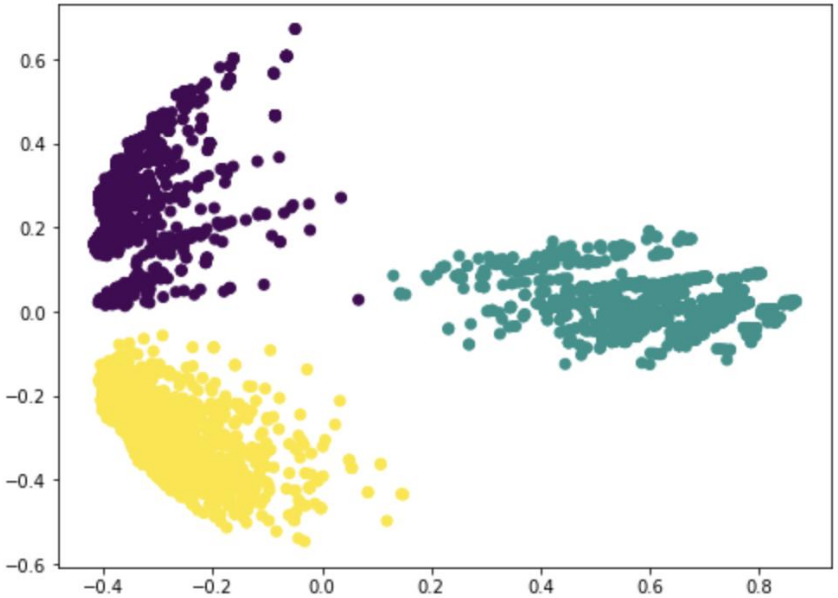
simple cleanup

Using Stringlifier



Stringlifier cleanup





Actual vs. automatic clusters

Stringlifier cleanup

How to Get and Use Stringlifier

GitHub: https://github.com/adobe/stringlifier

Quick start notebook:

https://colab.research.google.com/drive/1bgZQSKhVAYU4r46wqb0v8Sfvuo_yMOLA?usp=sharing

PIP package:

https://pypi.org/project/stringlifier/

Resources

Adobe Security Newsletter
adobe.com/go/securitynews

Twitter
@AdobeSecurity

Trust Center
trust.adobe.com

Open Source CCF v3.0
adobe.com/go/open-source-ccf

Security @ Adobe blog
blogs.adobe.com/security/

Adobe