#ODSC

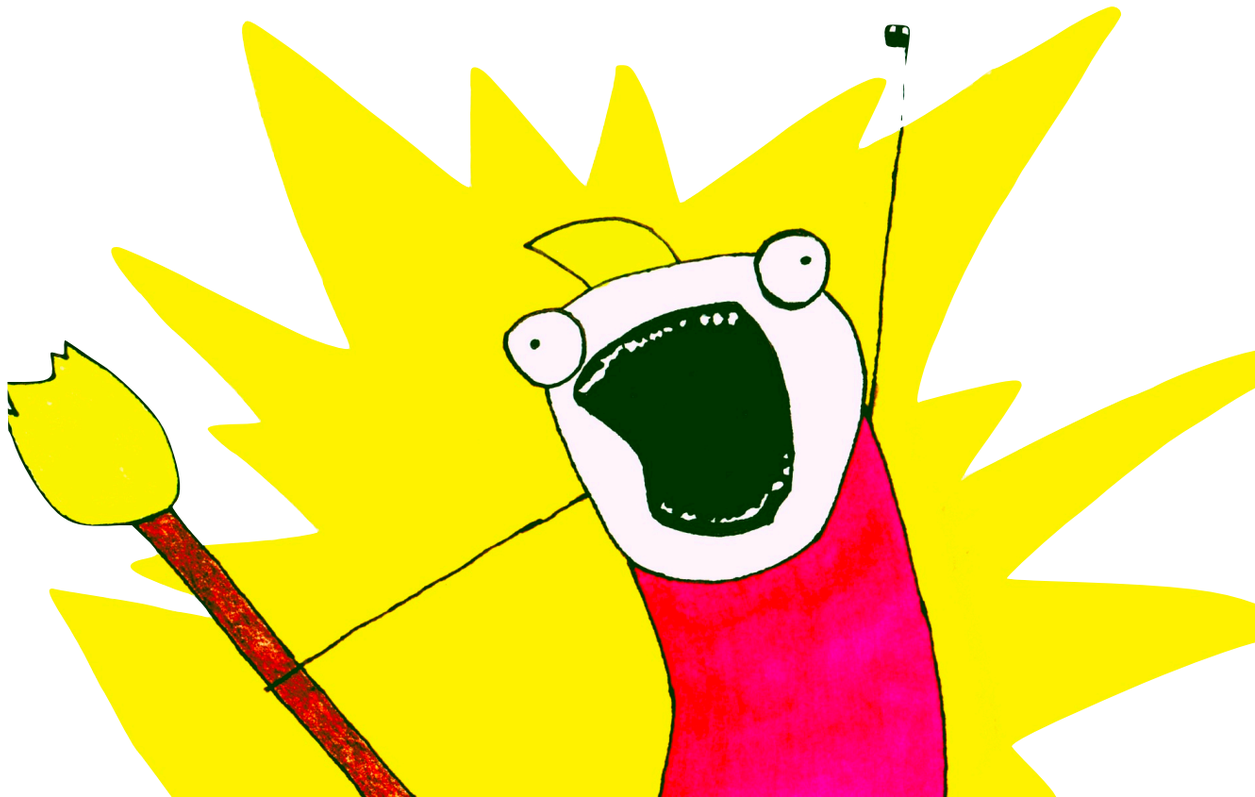
Boston | May 1 - 4 2018

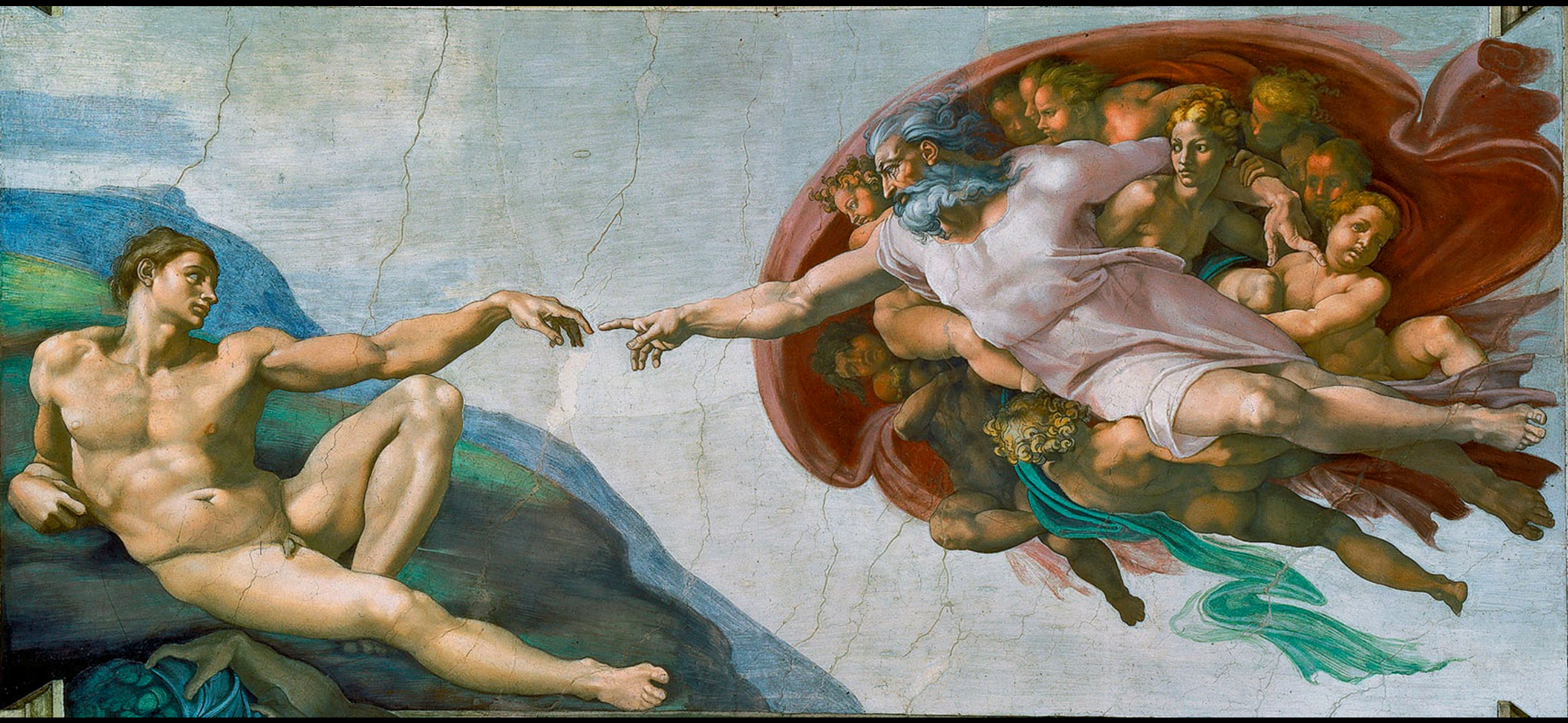
DATAFY

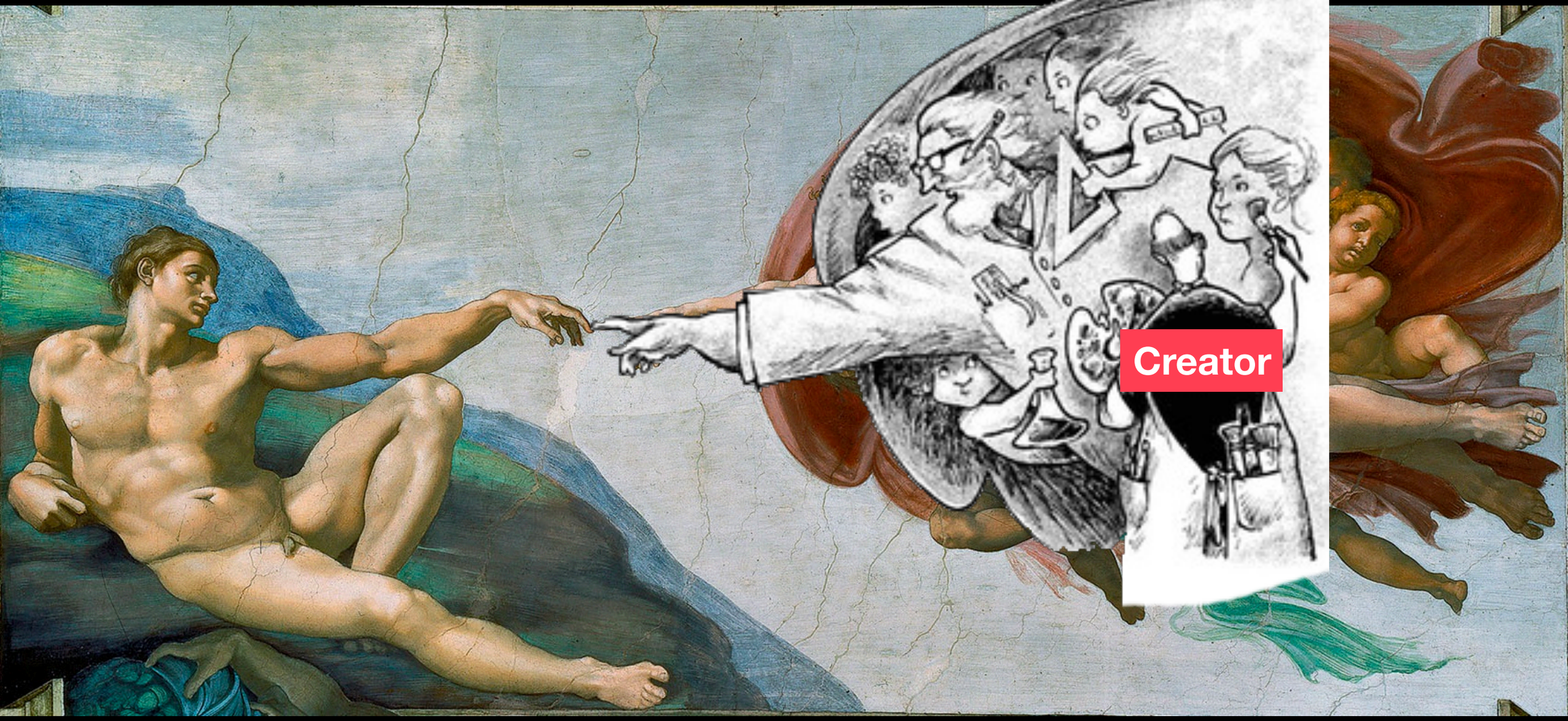
Max Humber

DATAFY ALL THE THINGS

Max Humber

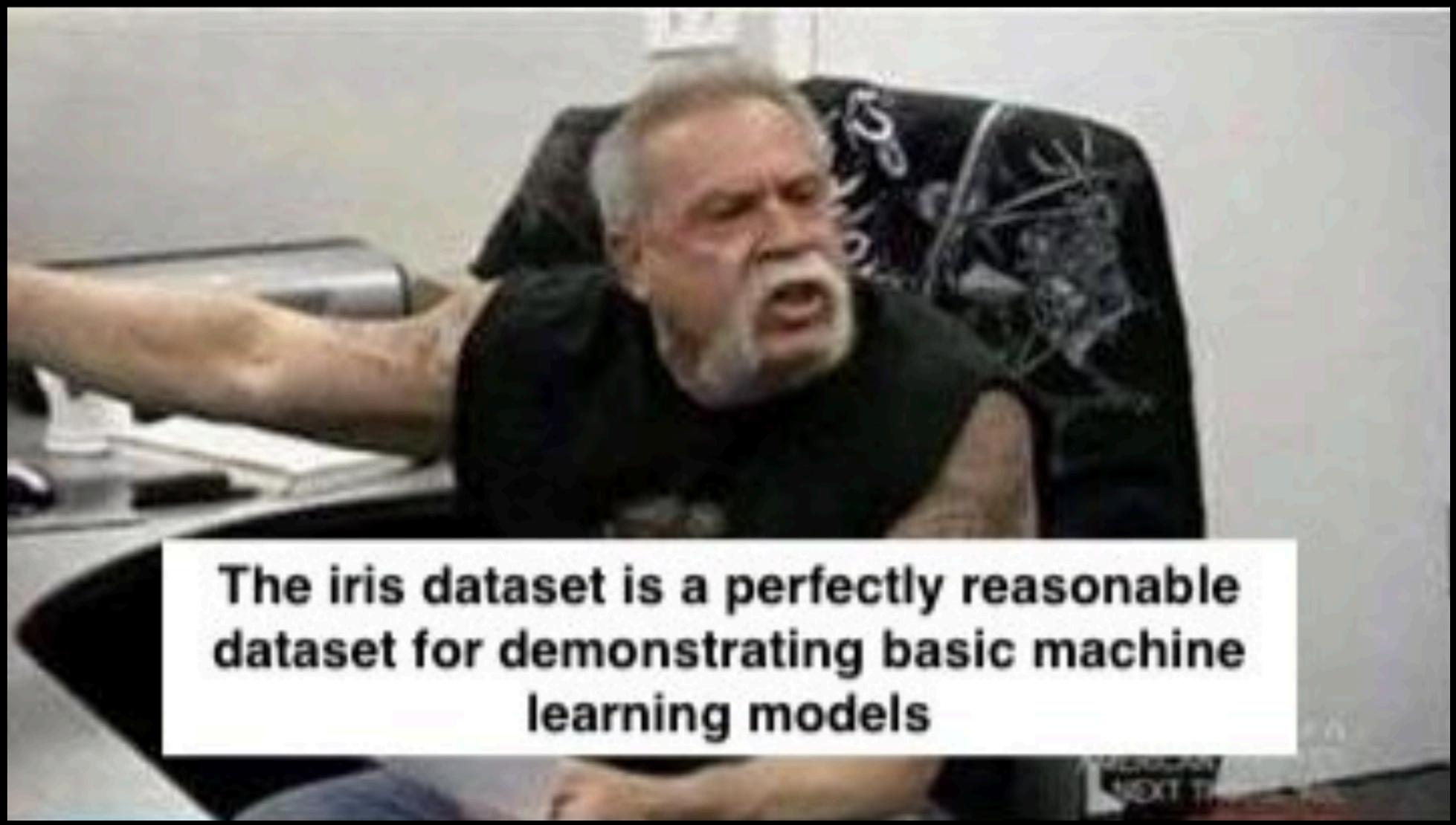






Data Creationism

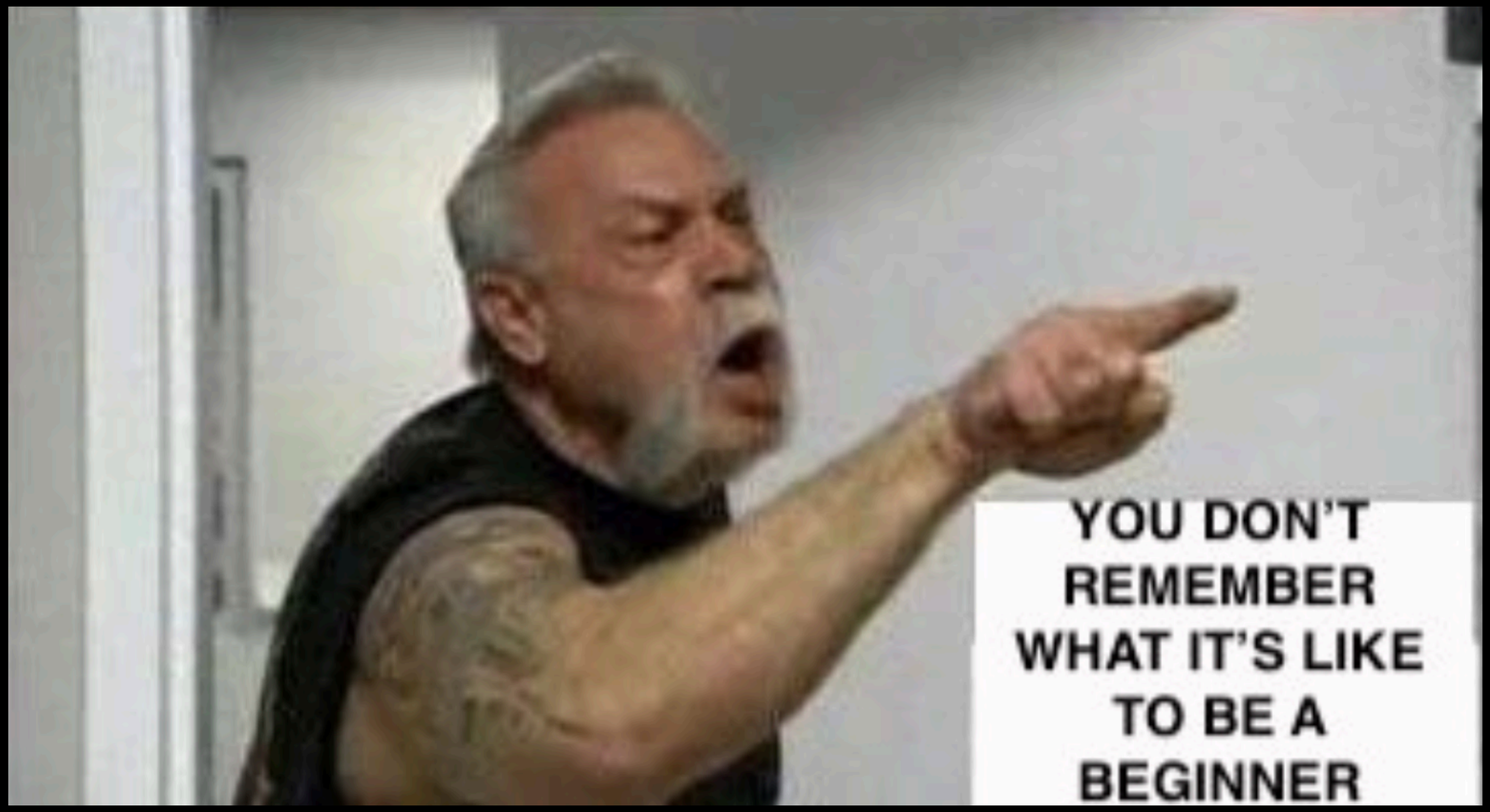
Data is everywhere. And it's everything (if you're creative)! So it makes me so sad to see **Iris** and **Titanic** in every blog, tutorial and book on data science and machine learning. In DATAFY ALL THE THINGS I'll empower you to **curate** and **create** your own data sets (so that we can all finally let Iris die). You'll learn how to parse **unstructured text**, harvest data from interesting **websites** and public **API**s and about **capturing** and dealing with **sensor** data. Examples in this talk will be provided and written in python and will rely on requests, beautifulsoup, mechanicalsoup, pandas and some 3.6+ magic!

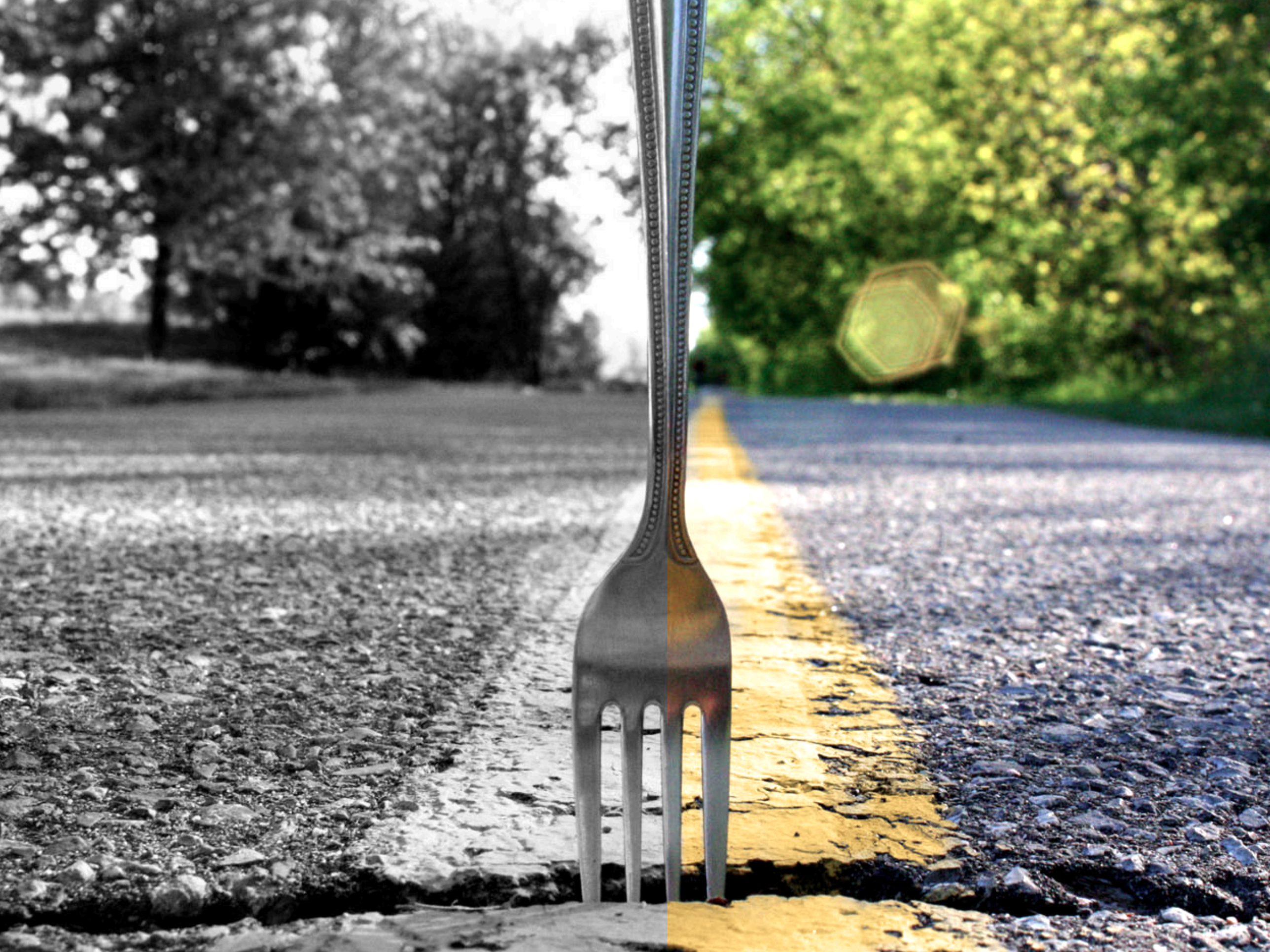

The iris dataset is a perfectly reasonable dataset for demonstrating basic machine learning models


It lulls beginners into falsely thinking working with data is easy


EVERYONE NEEDS TO START
SOMEWHERE


IT'S THE WRONG PLACE FOR BEGINNERS TO START


YOU DON'T
REMEMBER
WHAT IT'S LIKE
TO BE A
BEGINNER



ROBERT KOSARA / MARCH 21, 2018

How to Get Excited About Standard Datasets

…Who hasn't stared at an iris plant and gone crazy trying to decide whether it's an iris setosa, versicolor, or maybe even virginica? It's the stuff that keeps you up at night for days at a time.

Luckily, the iris dataset makes that super easy. All you have to do is measure the length and width of your particular iris's petal and sepal, and you're ready to rock! What's that, you still can't decide because the classes overlap? Well, but at least now you have data!


Iris
Bespoke data

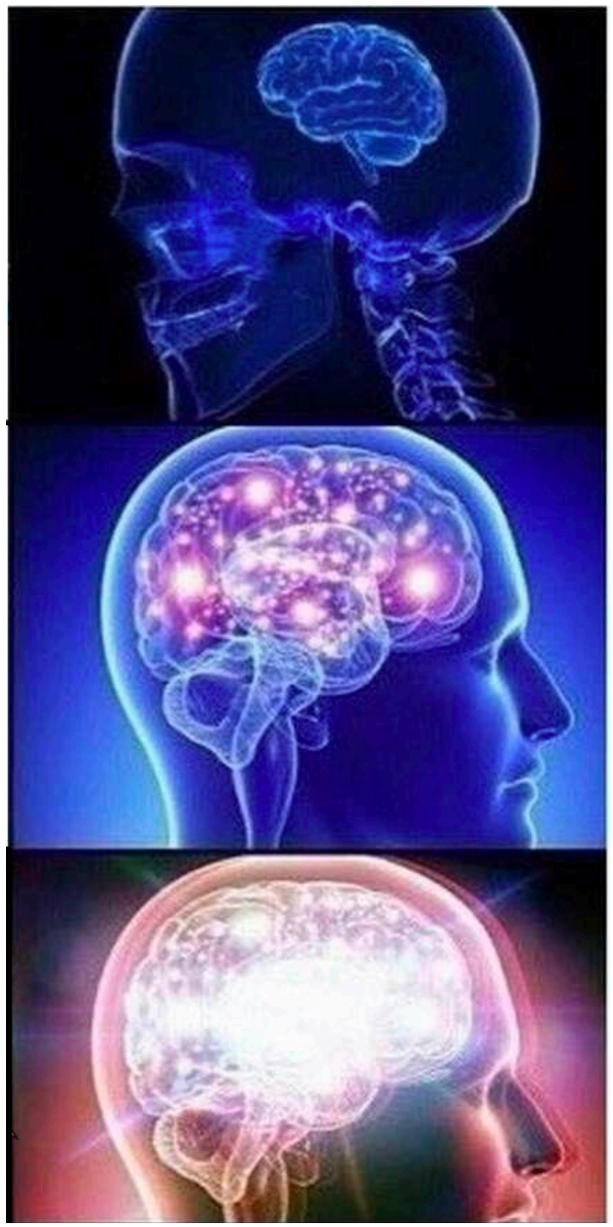

Bespoke data
Iris

This presentation…

create

curate

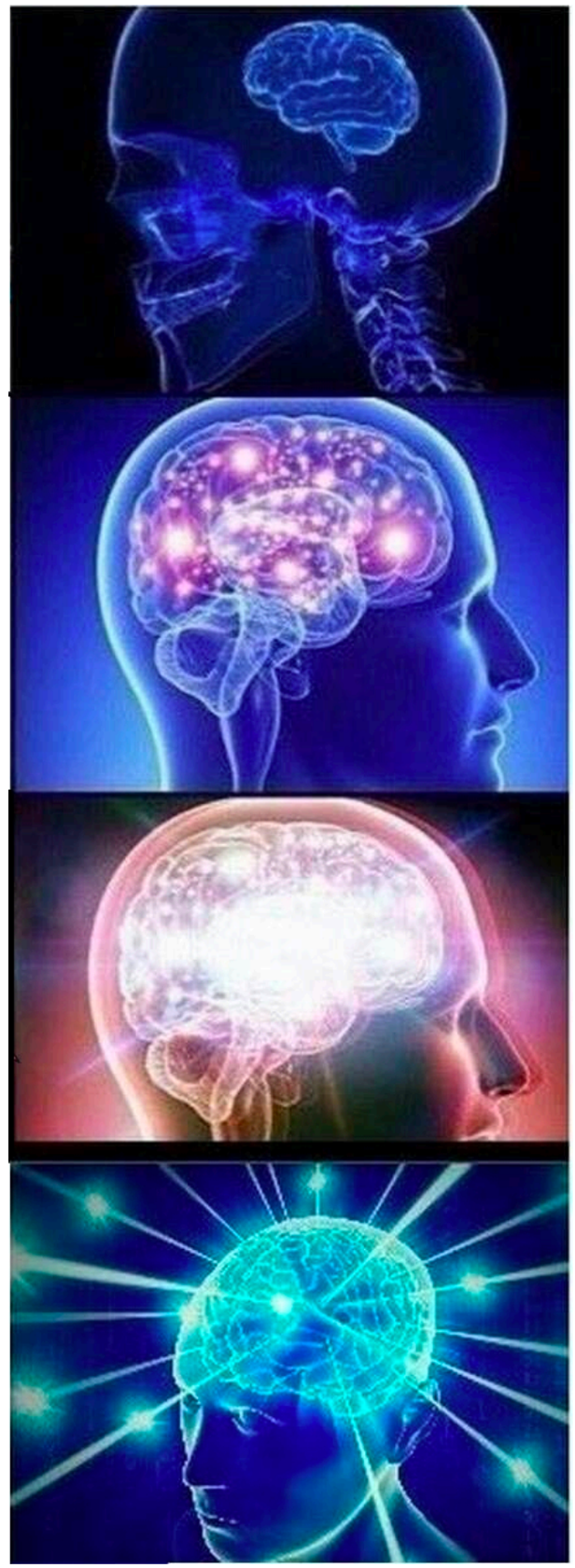
capture

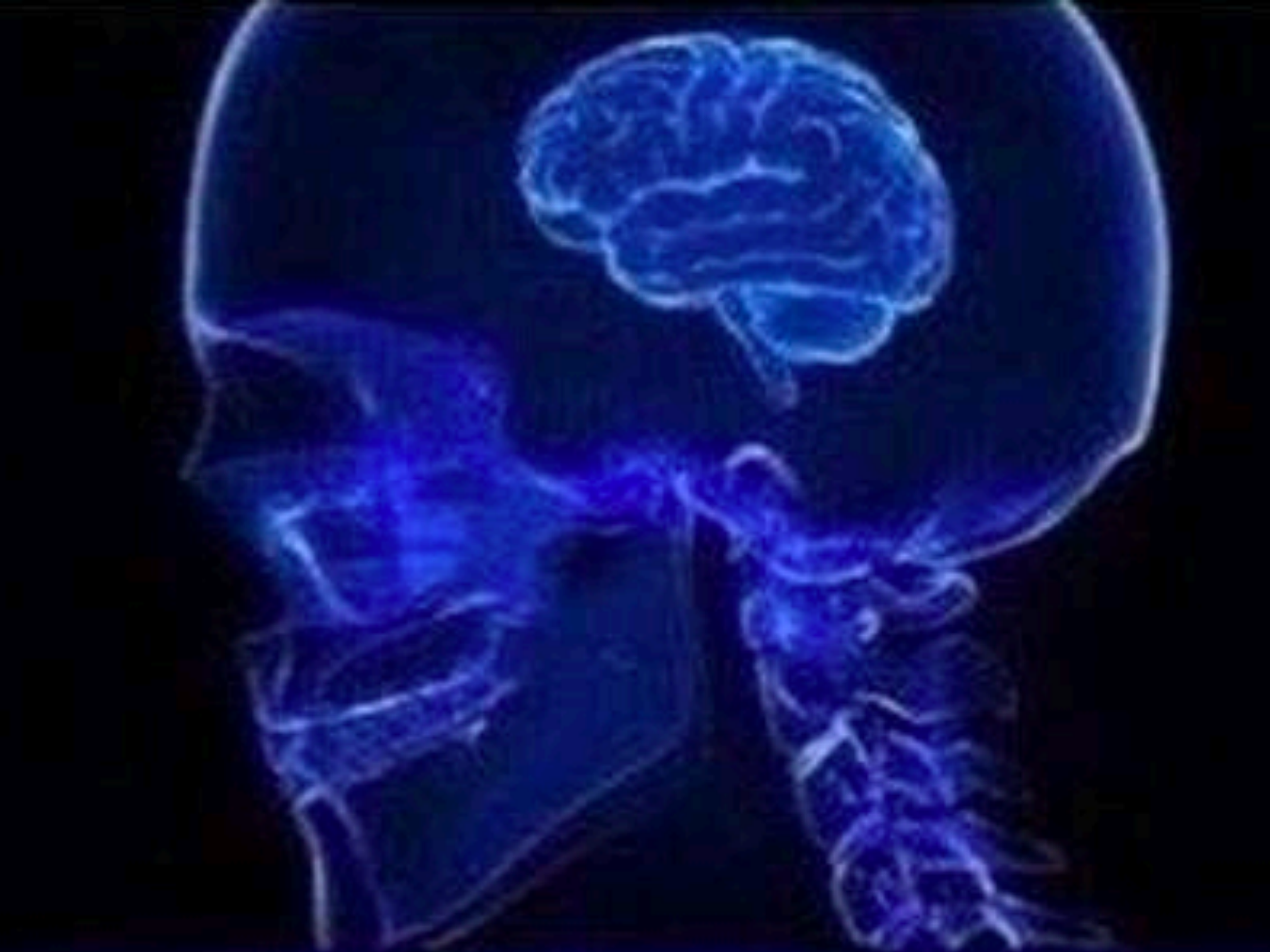


create

curate

capture






pd.DataFrame()

1

```
import pandas as pd

data = [
    ['conference', 'month', 'attendees'],
    ['ODSC', 'May', 5000],
    ['PyData', 'June', 1500],
    ['PyCon', 'May', 3000],
    ['useR!', 'July', 2000],
    ['Strata', 'August', 2500]
]
df = pd.DataFrame(data, columns=data.pop(0))
```

```python
import pandas as pd

data = [
    ['conference', 'month', 'attendees'],
    ['ODSC', 'May', 5000],
    ['PyData', 'June', 1500],
    ['PyCon', 'May', 3000],
    ['useR!', 'July', 2000],
    ['Strata', 'August', 2500]
]
df = pd.DataFrame(data, columns=data.pop(0))
```

```
import pandas as pd

data = [
    ['conference', 'month', 'attendees'],
    ['ODSC', 'May', 5000],
    ['PyData', 'June', 1500],
    ['PyCon', 'May', 3000],
    ['useR!', 'July', 2000],
    ['Strata', 'August', 2500]
]
df = pd.DataFrame(data, columns=data.pop(0))
```

	conference	month	attendees
0	ODSC	May	5000
1	PyData	June	1500
2	PyCon	May	3000
3	useR!	July	2000
4	Strata	August	2500

2

```
data = {
    'package': ['requests', 'pandas', 'Keras', 'mummify'],
    'installs': [4000000, 9000000, 875000, 1200]
}
df = pd.DataFrame(data)
```

```
data = {
    'package': ['requests', 'pandas', 'Keras', 'mummify'],
    'installs': [4000000, 9000000, 875000, 1200]
}
df = pd.DataFrame(data)
```

	installs	package
0	4000000	requests
1	9000000	pandas
2	875000	Keras
3	1200	mummify

3

```
df = pd.DataFrame([
    {'artist': 'Bino', 'plays': 100_000},
    {'artist': 'Drake', 'plays': 1_000},
    {'artist': 'ODESZA', 'plays': 10_000},
    {'artist': 'Brasstracks', 'plays': 100}
])
```

```
df = pd.DataFrame([
    {'artist': 'Bino', 'plays': 100_000},
    {'artist': 'Drake', 'plays': 1_000},
    {'artist': 'ODESZA', 'plays': 10_000},
    {'artist': 'Brasstracks', 'plays': 100}
])
```

	artist	plays
0	Bino	100000
1	Drake	1000
2	ODESZA	10000
3	Brasstracks	100

```python
df = pd.DataFrame([
    {'artist': 'Bino', 'plays': 100_000},
    {'artist': 'Drake', 'plays': 1_000},
    {'artist': 'ODESZA', 'plays': 10_000},
    {'artist': 'Brasstracks', 'plays': 100}
])
```

	artist	plays
0	Bino	100000
1	Drake	1000
2	ODESZA	10000
3	Brasstracks	100

```python
df = pd.DataFrame([
    {'artist': 'Bino', 'plays': 100_000},
    {'artist': 'Drake', 'plays': 1_000},
    {'artist': 'ODESZA', 'plays': 10_000},
    {'artist': 'Brasstracks', 'plays': 100}
])
```

	artist	plays
0	Bino	100000
1	Drake	1000
2	ODESZA	10000
3	Brasstracks	100

4

```
csv = '''\
    food,fat,carbs,protein
    avocado,0.15,0.09,0.02
    orange,0.001,0.12,0.009
    almond,0.49,0.22,0.21
    steak,0.19,0,0.25
    peas,0,0.04,0.1
'''
```

```
pd.read_csv(csv)
```

```
csv = '''\
    food,fat,carbs,protein
    avocado,0.15,0.09,0.02
    orange,0.001,0.12,0.009
    almond,0.49,0.22,0.21
    steak,0.19,0,0.25
    peas,0,0.04,0.1
'''
```

```
pd.read_csv(csv)
```

```
# ---------------------------------------------------------------------------
# FileNotFoundError                         Traceback (most recent call last)
# <ipython-input-22-b8ca875b07d1> in <module>()
# ----> 1 pd.read_csv(csv)
#
# FileNotFoundError: File b'food,fat,carbs,protein\n...' does not exist
```

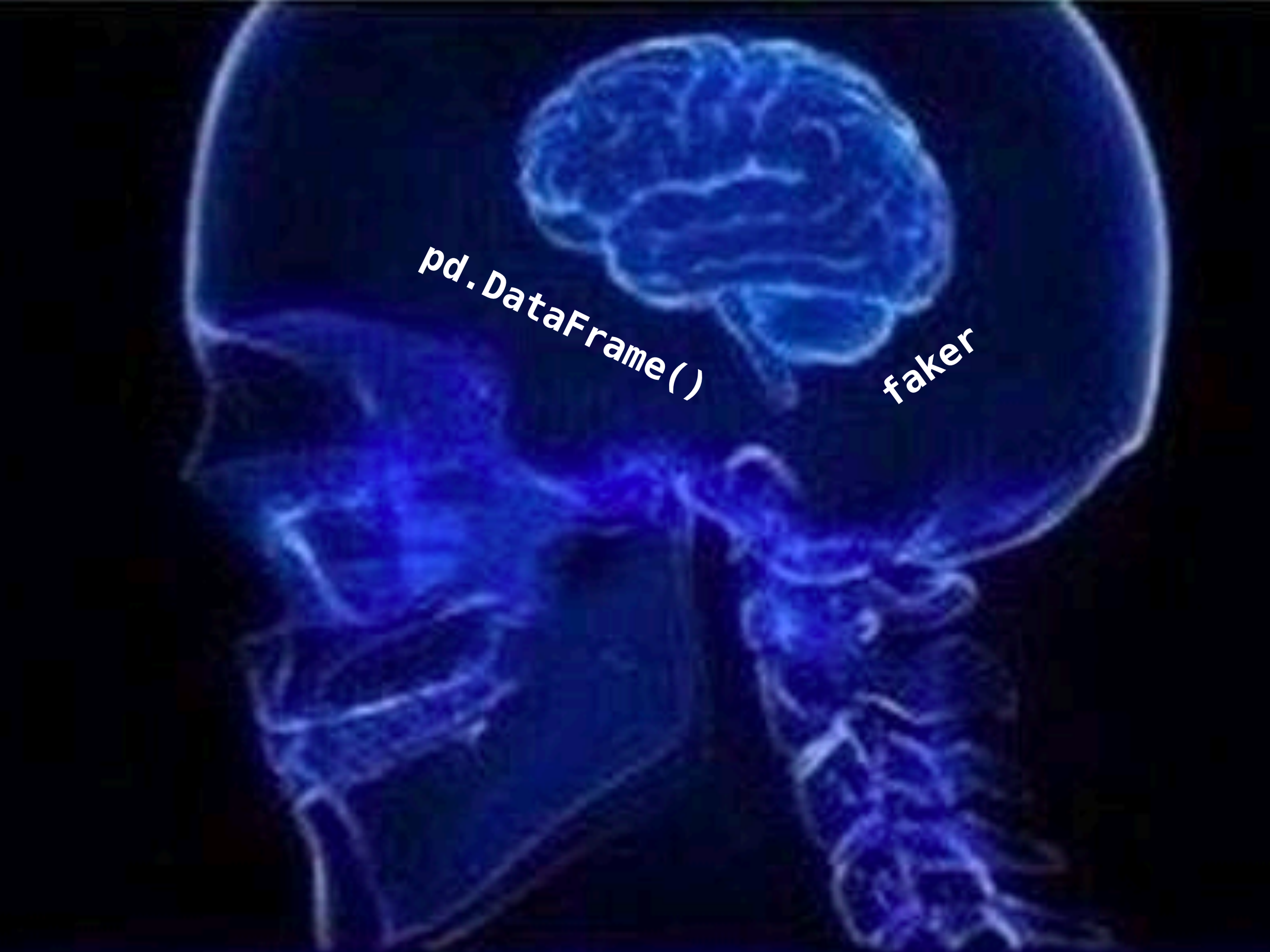
```
from io import StringIO

csv = '''\
    food,fat,carbs,protein
    avocado,0.15,0.09,0.02
    orange,0.001,0.12,0.009
    almond,0.49,0.22,0.21
    steak,0.19,0,0.25
    peas,0,0.04,0.1
'''

df = pd.read_csv(StringIO(csv))
```

```
from io import StringIO

csv = '''\
    food,fat,carbs,protein
    avocado,0.15,0.09,0.02
    orange,0.001,0.12,0.009
    almond,0.49,0.22,0.21
    steak,0.19,0,0.25
    peas,0,0.04,0.1
'''
```

```
df = pd.read_csv(StringIO(csv))
```

	food	fat	carbs	protein
0	avocado	0.150	0.09	0.020
1	orange	0.001	0.12	0.009
2	almond	0.490	0.22	0.210
3	steak	0.190	0.00	0.250
4	peas	0.000	0.04	0.100


pd.DataFrame()


pd.DataFrame()
faker

```
# pip install Faker
from faker import Faker
fake = Faker()

fake.name()
```

```
# pip install Faker
from faker import Faker
fake = Faker()

fake.name()
```
```
'Stephanie Lane'
```

```
# pip install Faker
from faker import Faker
fake = Faker()

fake.name()
```
```
'Stephanie Lane'
```
```
fake.phone_number()
```

```python
# pip install Faker
from faker import Faker
fake = Faker()
```

```python
fake.name()
```

```
'Stephanie Lane'
```

```python
fake.phone_number()
```

```
'466-579-9676'
```

```python
# pip install Faker
from faker import Faker
fake = Faker()
```

```python
fake.name()
```

```
'Stephanie Lane'
```

```python
fake.phone_number()
```

```
'466-579-9676'
```

```python
fake.bs()
```

```python
# pip install Faker
from faker import Faker
fake = Faker()
```

```python
fake.name()
```

```
'Stephanie Lane'
```

```python
fake.phone_number()
```

```
'466-579-9676'
```

```python
fake.bs()
```

```
'engage global users'
```

```
'unleash leading-edge synergies'
```

```
fak 'leverage visionary niches'
```

```
fake.phone_num 'repurpose magnetic e-commerce'
```

```
fake.bs() 'engage global users'
```

```
're-intermediate user-centric infrastructures'
```

```
'drive enterprise networks'
```

```python
# pip install Faker
from faker import Faker
fake = Faker()
```

```python
fake.name()
```

```
'Stephanie Lane'
```

```python
fake.phone_number()
```

```
'466-579-9676'
```

```python
fake.bs()
```

```
'engage global users'
```

```python
fake.profile()
```

```python
# pip install Faker
from faker import Faker
fake = Faker()
```

```python
fake.name()
```

```
'Stephanie Lane'
```

```python
fake.phone_number()
```

```
'466-579-9676'
```

```python
fake.bs()
```

```
'engage global users'
```

```python
fake.profile()
```

```
{'address': '2581 West Vista\nLake Jenniferchester, MH 05640',
 'birthdate': '2017-09-03',
 'blood_group': 'A+',
 'company': 'Wall, Hernandez and Thomas',
 'current_location': (Decimal('-61.270794'), Decimal('127.522539')),
 'job': 'Regulatory affairs officer',
 'mail': 'lewiskim@hotmail.com',
 'name': 'Joshua Perkins',
 'residence': '7443 Wendy Station\nPaulburgh, VT 64427-9810',
```

```
output =  {
    'created_at': fake.past_datetime(start_date='-365d'),
    'name': fake.name(),
    'occupation': fake.job(),
    'address': fake.street_address(),
    'credit_card': fake.credit_card_number(card_type='visa'),
    'company_bs': fake.bs(),
'city': fake.city(),
'ssn': fake.ssn(),
'paragraph': fake.paragraph()}
```

```
def create_rows(n=1):
    output = [{
        'created_at': fake.past_datetime(start_date='-365d'),
        'name': fake.name(),
        'occupation': fake.job(),
        'address': fake.street_address(),
        'credit_card': fake.credit_card_number(card_type='visa'),
        'company_bs': fake.bs(),
        'city': fake.city(),
        'ssn': fake.ssn(),
        'paragraph': fake.paragraph()}
        for x in range(n)]
    return pd.DataFrame(output)

df = create_rows(10)
```

```
import pandas as pd
import sqlite3

con = sqlite3.connect('data/fake.db')
cur = con.cursor()

df.to_sql(name='users', con=con, if_exists="append", index=True)
```

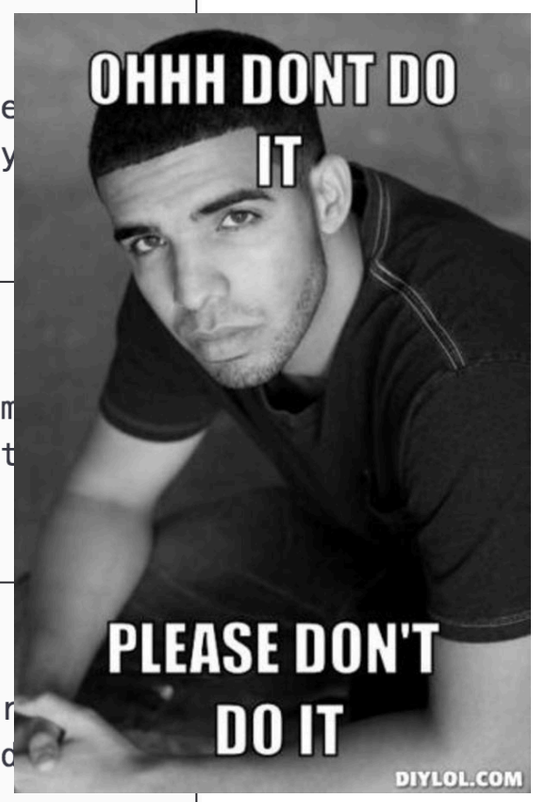
```
import pandas as pd
import sqlite3

con = sqlite3.connect('data/fake.db')
cur = con.cursor()

df.to_sql(name='users', con=con, if_exists="append", index=True)

pd.read_sql('select * from users', con)
```

```
import pandas as pd
import sqlite3

con = sqlite3.connect('data/fake.db')
cur = con.cursor()

df.to_sql(name='users', con=con, if_exists="append", index=True)

pd.read_sql('select * from users', con)
```

index	address	city	company_bs	created_at	credit_card	name
0	995 Steven Corner	West Brian	brand synergistic niches	2017-09-16 23:57:04	4354918050874459	Alexander Doyle
1	31835 Martinez Islands	South Heathertown	revolutionize front-end metrics	2017-09-03 23:57:36	4802837467063555	Tammy Peters
2	8620 Gabriel Branch Apt. 054	South Ronaldtown	re-contextualize plug-and-play applications	2017-12-29 11:19:04	4980799271663703	Christopher Mcdowell

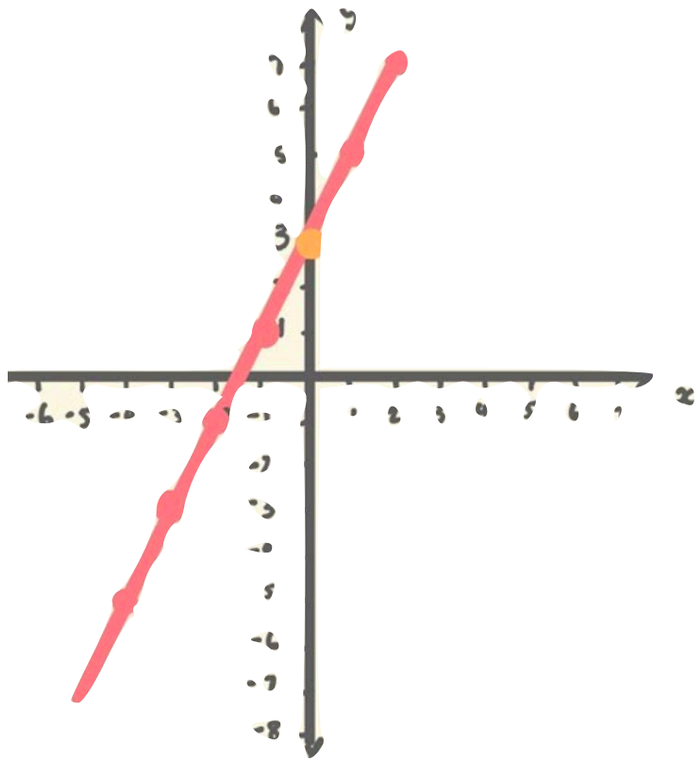
```python
import pandas as pd
import sqlite3

con = sqlite3.connect('data/fake.db')
cur = con.cursor()

df.to_sql(name='users', con=con, if_exists="append", index=True)

pd.read_sql('select * from users', con)
```

index	address	city	company_bs	created_at	credit_card	name
0	995 Steven Corner	West Brian	brand synergistic niches	2017-09-16 23:57:04	4354918050874459	Ale Doy
1	31835 Martinez Islands	South Heathertown	revolutionize front-end metrics	2017-09-03 23:57:36	4802837467063555	Tam Pet
2	8620 Gabriel Branch Apt. 054	South Ronaldtown	re-contextualize plug-and-play applications	2017-12-29 11:19:04	4980799271663703	Cha Mic




pd.DataFrame()
faker


sklearn
pd.DataFrame()
faker

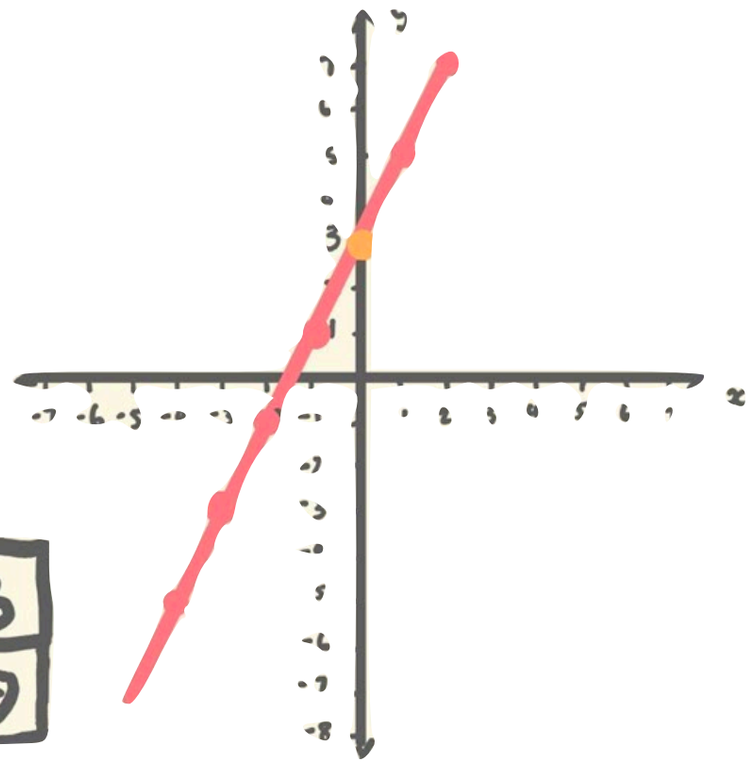


$$y = mx + b$$

$$y = 2x + 3$$

x	-3	-2	-1	0	1	2	3
y	-3	-1	1	3	5	7	9

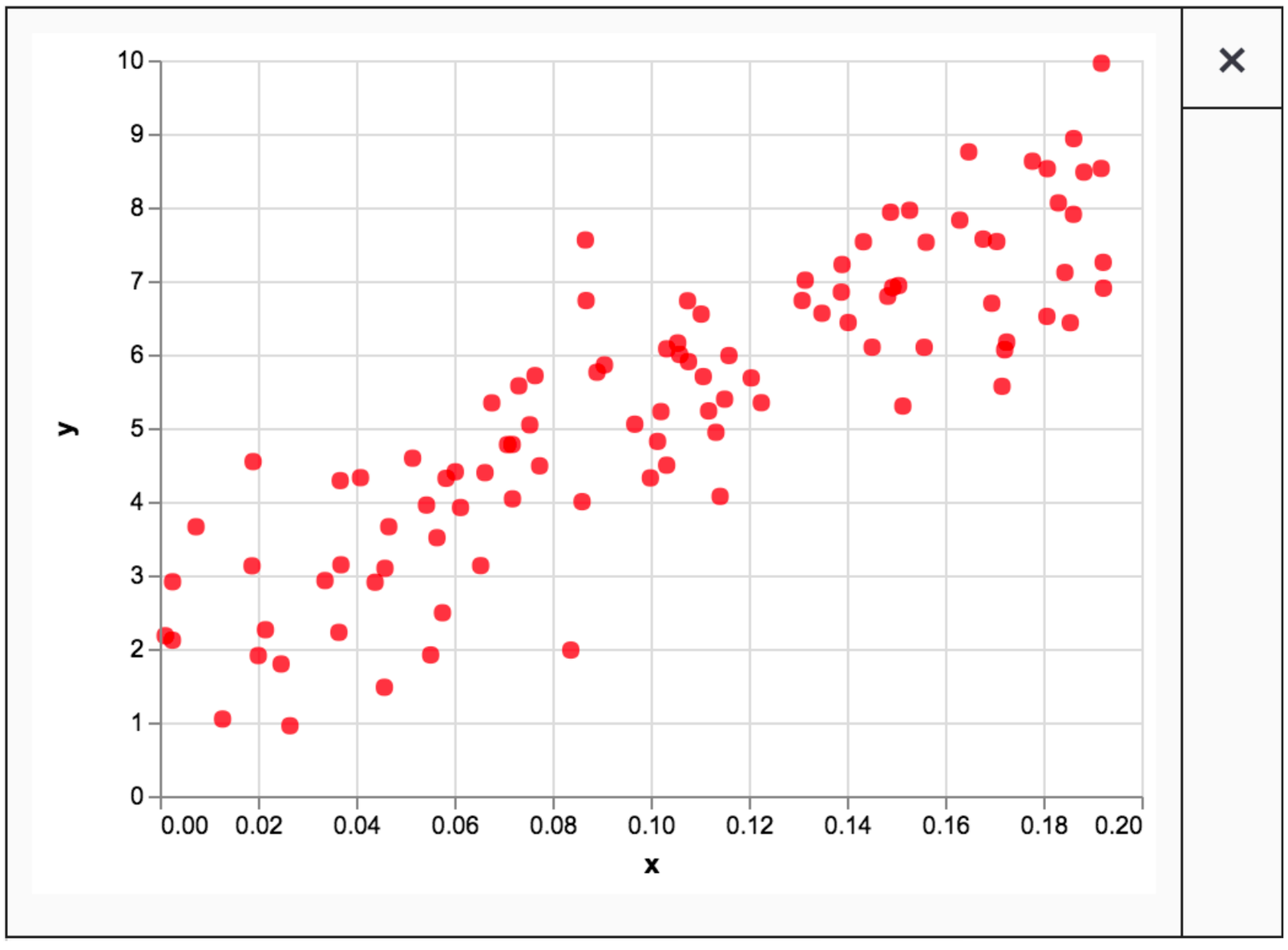
+2



```
import numpy as np
import pandas as pd

n = 100
rng = np.random.RandomState(1993)
x = 0.2 * rng.rand(n)
y = 31*x + 2.1 + rng.randn(n)

df = pd.DataFrame({'x': x, 'y': y})
```

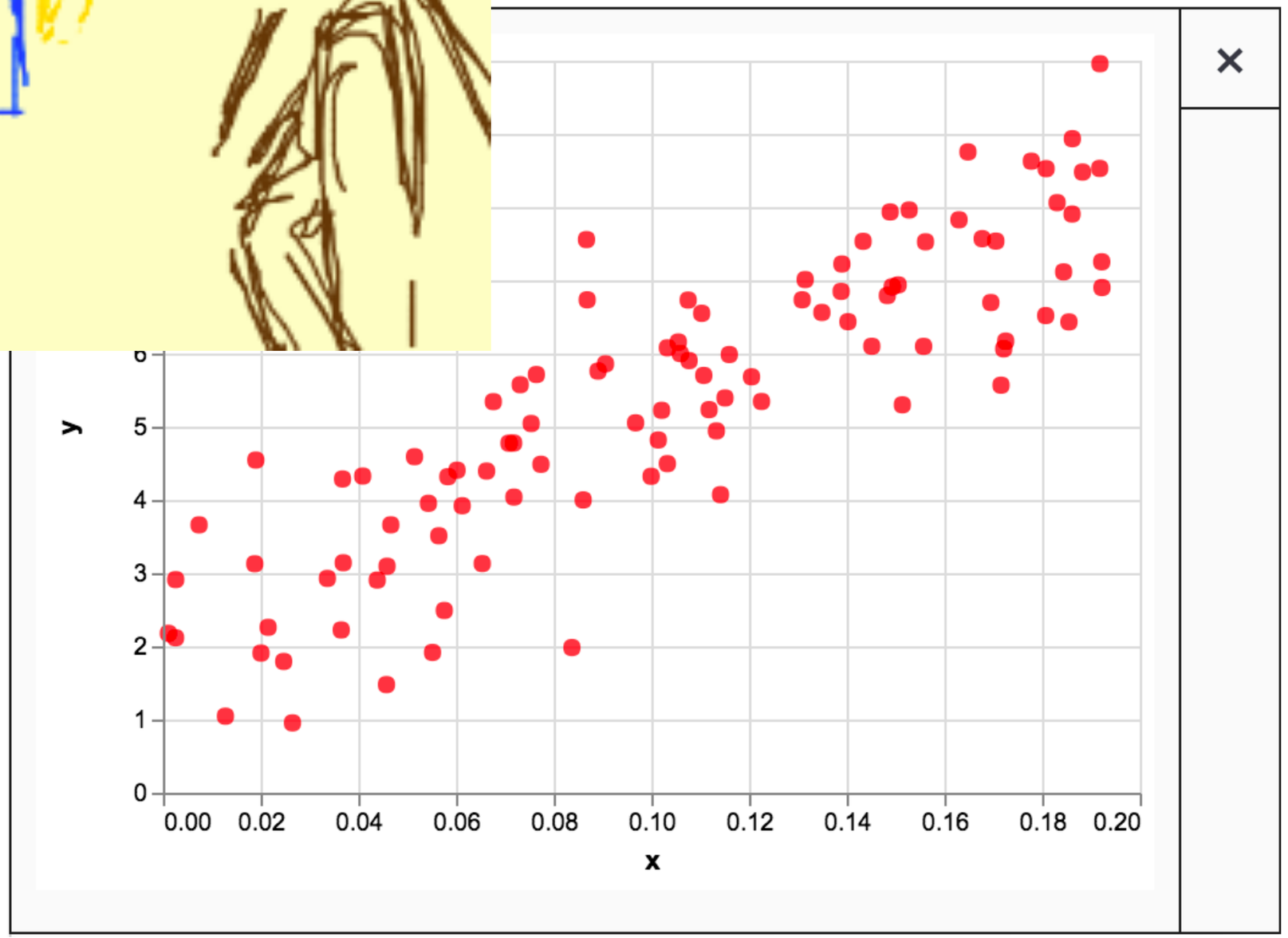
```
df = pd.DataFrame({'x': x, 'y': y})

import altair as alt

(alt.Chart(df, background='white')
    .mark_circle(color='red', size=50)
    .encode(
        x='x',
        y='y'
    )
)
```

```
df = pd.DataFrame({'x': x, 'y': y})

import altair as alt

(alt.Chart(df, background='white')
     .mark_circle(color='red', size=50)
     .encode(
         x='x',
         y='y'
     )
)
```




Let's Get Creative!

```
})


)
e=50)
```


y
x
0
1
2
3
4
5
6
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0.18
0.20

</create>


<curate>

2017 Reading Challenge
Congrats! You read 52 books of your goal of 52!

100%

2016 Reading Challenge
Congrats! You read 52 books of your goal of 52!

100%

2015 Reading Challenge
Congrats! You read 52 books of your goal of 52!

100%

==========
Bad Choices: How Algorithms Can Help You Think (Ali Almossawi)
- Your Highlight on page 3 | Location 46-46 | Added on Thursday, June 1, 2017 11:29:25 PM

An algorithm is a series of unambiguous steps that achieves some meaningful objective in finite time.
==========
The Moral Landscape (Sam Harris)
- Your Highlight on page 112 | Location 1703-1705 | Added on Sunday, June 4, 2017 11:24:24 PM

While anxiety and fear are emotions that most of us would prefer to live without, they serve as ancho
==========
The Moral Landscape (Sam Harris)
- Your Highlight on page 118 | Location 1797-1800 | Added on Sunday, June 4, 2017 11:34:12 PM

We are conscious of only a tiny fraction of the information that our brains process in each moment. Wh
==========
The Moral Landscape (Sam Harris)

File uses CRLF (Windows) line endings

CRLF UTF-8 Plain Text 0 files

```
==========
Bad Choices
- Your High
¤¬
An algorith
==========
The Moral L
- Your High
¤¬
While anxie
==========
The Moral L
- Your High
¤¬
We are cons
==========
The Moral L
```

Chris Albon

@chrisalbon

Following

#TeamStructured

Rachel Thomas @math_rachel

What do *you* call the type of data that is found in a SQL table or an Excel sheet? (I've heard many different opinions on this)

7:42 PM - 28 Apr 2018

9 Likes

5 9

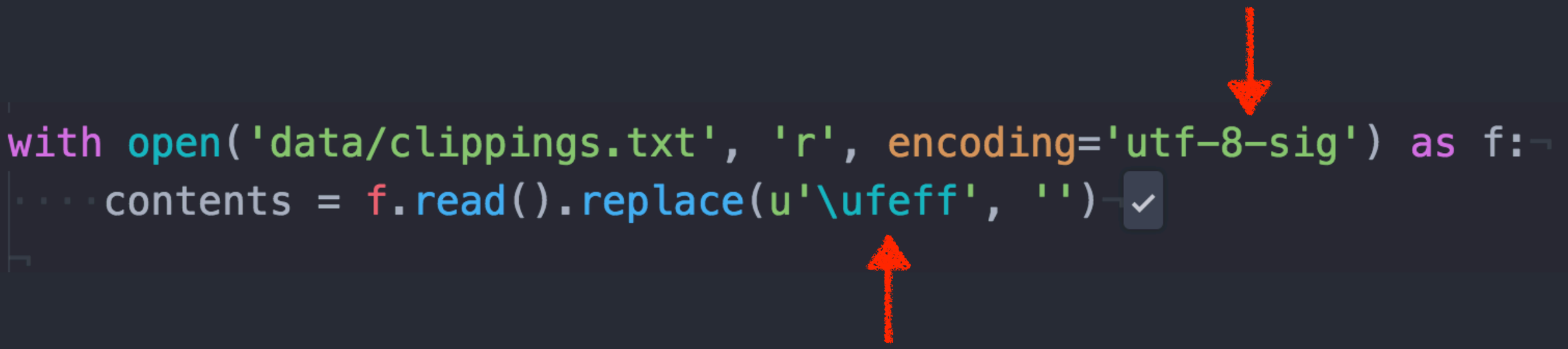
==========¤¬
Bad Choices: How Algorithms Can Help Y
- Your Highlight on page 3 | Location
¤¬
An algorithm is a series of unambiguou
==========¤¬
The Moral Landscape (Sam Harris)¤¬
- Your Highlight on page 112 | Locatio
¤¬
While anxiety and fear are emotions th
==========¤¬
The Moral Landscape (Sam Harris)¤¬
- Your Highlight on page 118 | Locatio
¤¬
We are conscious of only a tiny fracti
==========¤¬

```
with open('data/clippings.txt', 'r') as f:
    contents = f.read()
```

```
with open('data/clippings.txt', 'r') as f:
    contents = f.read()
```

```
---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent call last)
<ipython-input-5-f60260b28de1> in <module>()
      1 with open('data/clippings.txt', 'r') as f:
----> 2     contents = f.read()

~/anaconda3/lib/python3.6/encodings/ascii.py in decode(self, input, final)
     24 class IncrementalDecoder(codecs.IncrementalDecoder):
     25     def decode(self, input, final=False):
---> 26         return codecs.ascii_decode(input, self.errors)[0]
     27
     28 class StreamWriter(Codec,codecs.StreamWriter):

UnicodeDecodeError: 'ascii' codec can't decode byte 0xef
```

==========¤¬
Bad Choices: How Algorithms Can Help Y
- Your Highlight on page 3 | Location
¤¬
An algorithm is a series of unambiguou
==========¤¬
The Moral Landscape (Sam Harris)¤¬
- Your Highlight on page 112 | Locatio
¤¬
While anxiety and fear are emotions th
==========¤¬
The Moral Landscape (Sam Harris)¤¬
- Your Highlight on page 118 | Locatio
¤¬
We are conscious of only a tiny fracti
==========¤¬


with open('data/clippings.txt', 'r', encoding='utf-8-sig') as f:
contents = f.read().replace(u'\ufeff', '')

```python
with open('data/clippings.txt', 'r', encoding='utf-8-sig') as f:
    contents = f.read().replace(u'\ufeff', '')

print(contents)
```

```
The Happiness Hypothesis (Jonathan Haidt)
- Your Highlight on page 167 | Location 2554-2557 | Added on Sunday
Shakespeare wrote: Sweet are the uses of adversity, Which like the
==========
The Happiness Hypothesis (Jonathan Haidt)
- Your Highlight on page 169 | Location 2591-2594 | Added on Sunday

The life story is not the work of a historian— remember that the ri
==========
The Happiness Hypothesis (Jonathan Haidt)
- Your Highlight on page 171 | Location 2612-2613 | Added on Sunday
```

```python
with open('data/clippings.txt', 'r', encoding='utf-8-sig') as f:
    contents = f.read().replace(u'\ufeff', '')

print(contents)
```

```
The Happiness Hypothesis (Jonathan Haidt)
- Your Highlight on page 167 | Location 2554-2557 | Added on Sunday

Shakespeare wrote: Sweet are the uses of adversity, Which like the
==========
The Happiness Hypothesis (Jonathan Haidt)
- Your Highlight on page 169 | Location 2591-2594 | Added on Sunday

The life story is not the work of a historian— remember that the ri
==========
The Happiness Hypothesis (Jonathan Haidt)
- Your Highlight on page 171 | Location 2612-2613 | Added on Sunday
```

```
with open('data/clippings.txt', 'r', encoding='utf-8-sig') as f:
    contents = f.read().replace(u'\ufeff', '')
    lines = contents.rsplit('==========')
```

```
with open('data/clippings.txt', 'r', encoding='utf-8-sig') as f:
    contents = f.read().replace(u'\ufeff', '')
    lines = contents.rsplit('==========')
    lines[400:405]
```

```
['\nExit West (Mohsin Hamid)\n- Your Highlight on page 69 | Locatio
 '\nExit West (Mohsin Hamid)\n- Your Highlight on page 74 | Locatio
 '\nExit West (Mohsin Hamid)\n- Your Highlight on page 77 | Locatio
 '\nExit West (Mohsin Hamid)\n- Your Highlight on page 79 | Locatio
 '\nExit West (Mohsin Hamid)\n- Your Highlight on page 94 | Locatio
```

```
with open('data/clippings.txt', 'r', encoding='utf-8-sig') as f:
    contents = f.read().replace(u'\ufeff', '')
    lines = contents.rsplit('==========')
    lines[400:405]
```

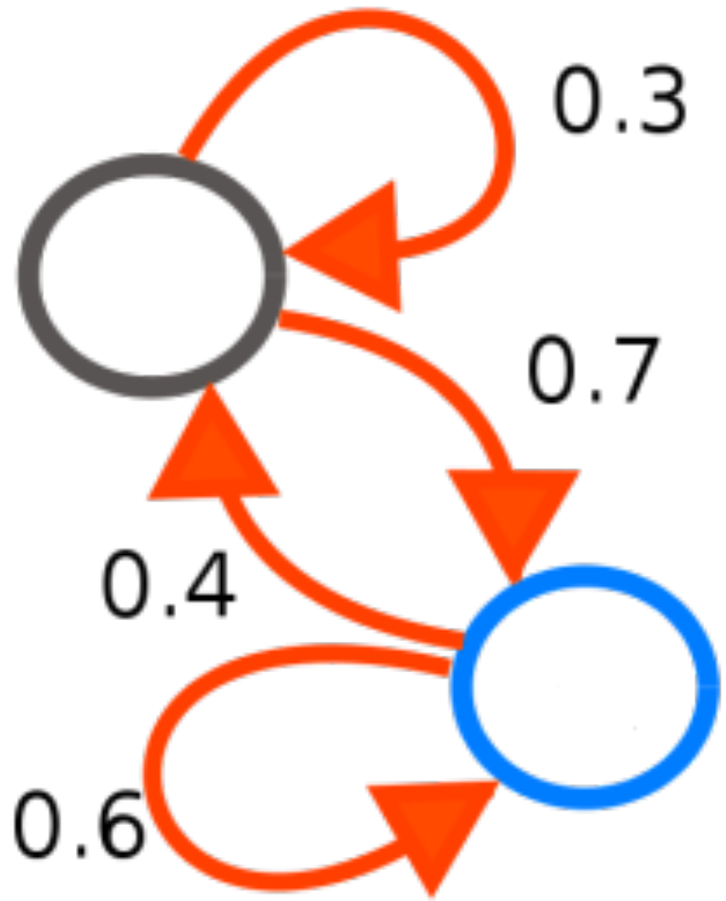
```
['\nExit West (Mohsin Hamid)\n- Your Highlight on page 69 | Locatio
 '\nExit West (Mohsin Hamid)\n- Your Highlight on page 74 | Locatio
 '\nExit West (Mohsin Hamid)\n- Your Highlight on page 77 | Locatio
 '\nExit West (Mohsin Hamid)\n- Your Highlight on page 79 | Locatio
'\nExit West (Mohsin Hamid)\n- Your Highlight on page 94 | Locatio
```

```
 2017 11:10:06 PM\n\nShe tried to recover her former go
 2017 11:19:01 PM\n\nShe was uncertain what to do to di
 2017 11:25:08 PM\n\nthe calm that is called the calm b
 2017 11:27:25 PM\n\nBut liberated from claustrophobic
2017 11:46:39 PM\n\nIn exchange for their labor in clear
```

```
meta, quote = line.split(')\n- ', 1)
title, author = meta.split(' (', 1)
_, quote = quote.split('\n\n')
```

```
with open('data/clippings.txt', 'r', encoding='utf-8-sig') as f:
    contents = f.read().replace(u'\ufeff', '')
    lines = contents.rsplit('==========')
    store = {'author': [], 'title': [], 'quote': []}
    for line in lines:
        try:
meta, quote = line.split(')\n- ', 1)
title, author = meta.split(' (', 1)
_, quote = quote.split('\n\n')
```

```
with open('data/clippings.txt', 'r', encoding='utf-8-sig') as f:
    contents = f.read().replace(u'\ufeff', '')
    lines = contents.rsplit('==========')
    store = {'author': [], 'title': [], 'quote': []}
    for line in lines:
        try:
            meta, quote = line.split(')\n- ', 1)
            title, author = meta.split(' (', 1)
            _, quote = quote.split('\n\n')
            store['author'].append(author.strip())
            store['title'].append(title.strip())
            store['quote'].append(quote.strip())
        except ValueError:
            pass
```


0.3
0.7
0.4
0.6

```
import markovify
import pandas as pd

df = pd.read_csv('data/highlights.csv')

text = '\n'.join(df['quote'].values)

model = markovify.NewlineText(text)

model.make_short_sentence(140)
```

```python
import markovify
import pandas as pd

df = pd.read_csv('data/highlights.csv')

text = '\n'.join(df['quote'].values)

model = markovify.NewlineText(text)
```

```python
model.make_short_sentence(140)
```

```
The person you aspire to be, you'll never become that person.
```

```
model.make_short_sentence(140)
```

Early Dates are Interviews; don't waste the opportunity to actually move toward a romantic relationship.

```
model.make_short_sentence(140)
```

Early Dates are Interviews; don't waste the opportunity to actually move toward a romantic relationship.

Pick a charity or two and set up autopay.

```
model.make_short_sentence(140)
```

Early Dates are Interviews; don't waste the opportunity to actually move toward a romantic relationship.

Pick a charity or two and set up autopay.

Everyone always wants money, which means you can implement any well-defined function simply by connecting with people's experiences.

```
model.make_short_sentence(140)
```

Early Dates are Interviews; don't waste the opportunity to actually move toward a romantic relationship.

Pick a charity or two and set up autopay.

Everyone always wants money, which means you can implement any well-defined function simply by connecting with people's experiences.

The more you play, the more varied experiences you have, the more people alive under worse conditions.

```
model.make_short_sentence(140)
```

Early Dates are Interviews; don't waste the opportunity to actually move toward a romantic relationship.

Pick a charity or two and set up autopay.

Everyone always wants money, which means you can implement any well-defined function simply by connecting with people's experiences.

The more you play, the more varied experiences you have, the more people alive under worse conditions.

Everything can be swept away by the bear to avoid losing your peace of mind.

```
model.make_short_sentence(140)
```

Early Dates are Interviews; don't waste the opportunity to actually move toward a romantic relationship.

Pick a charity or two and set up autopay.

Everyone always wants money, which means you can implement any well-defined function simply by connecting with people's experiences.

The more you play, the more varied experiences you have, the more people alive under worse conditions.

Everything can be swept away by the bear to avoid losing your peace of mind.

Make a spreadsheet. The cells of the future.

```
model.make_short_sentence(140)
```

Early Dates are Interviews; don't waste the opportunity to actually move toward a romantic relationship

in

T

E

Make a spreadsheet. The cells of the future.

Year Progress

@year_progress

Following

7:00 AM - 1 May 2018

2,286 Retweets **4,805** Likes

39 2.3K 4.8K

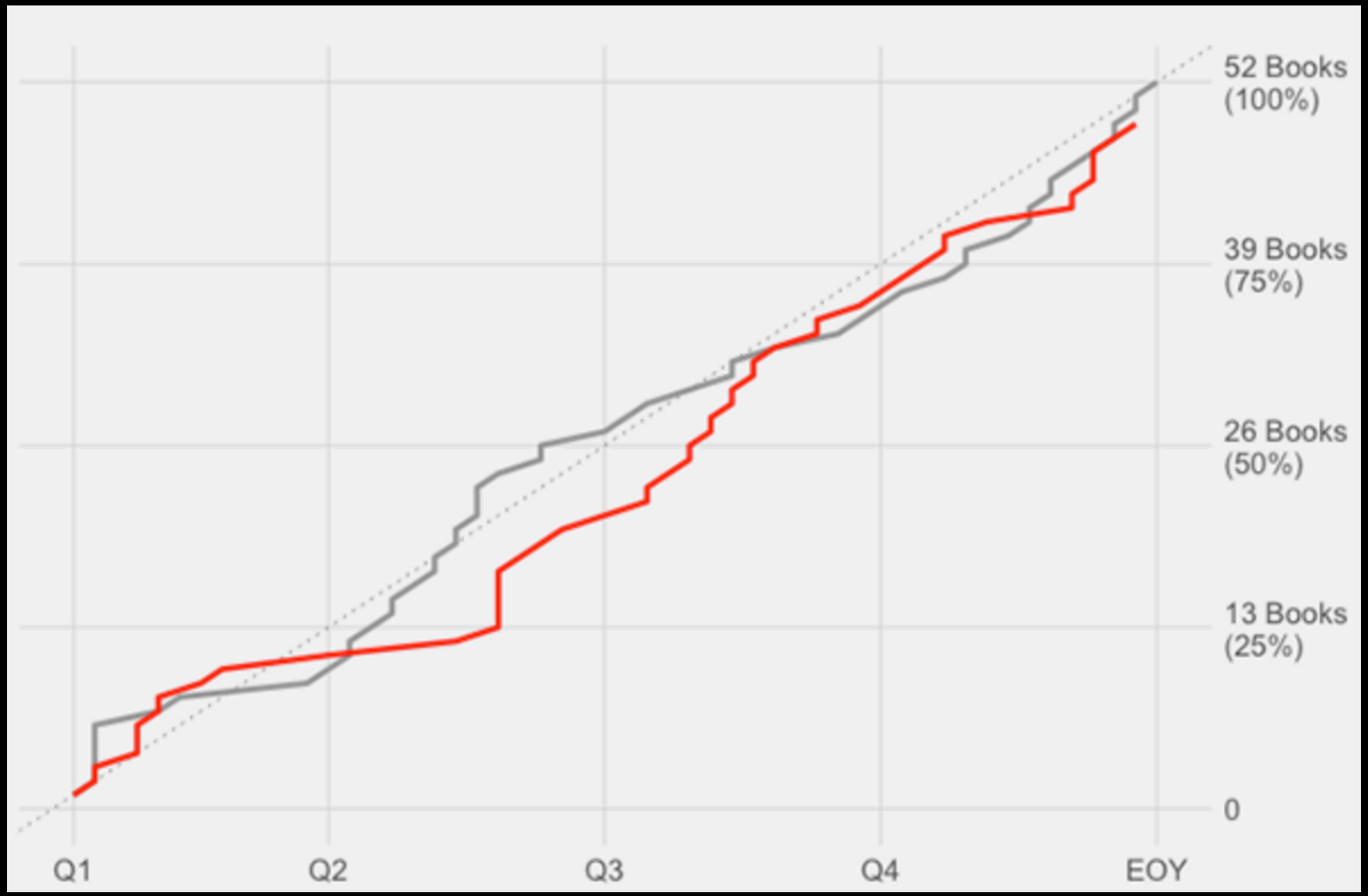
2018 READING CHALLENGE

11
books completed

6 books behind schedule

11/52 (21%)

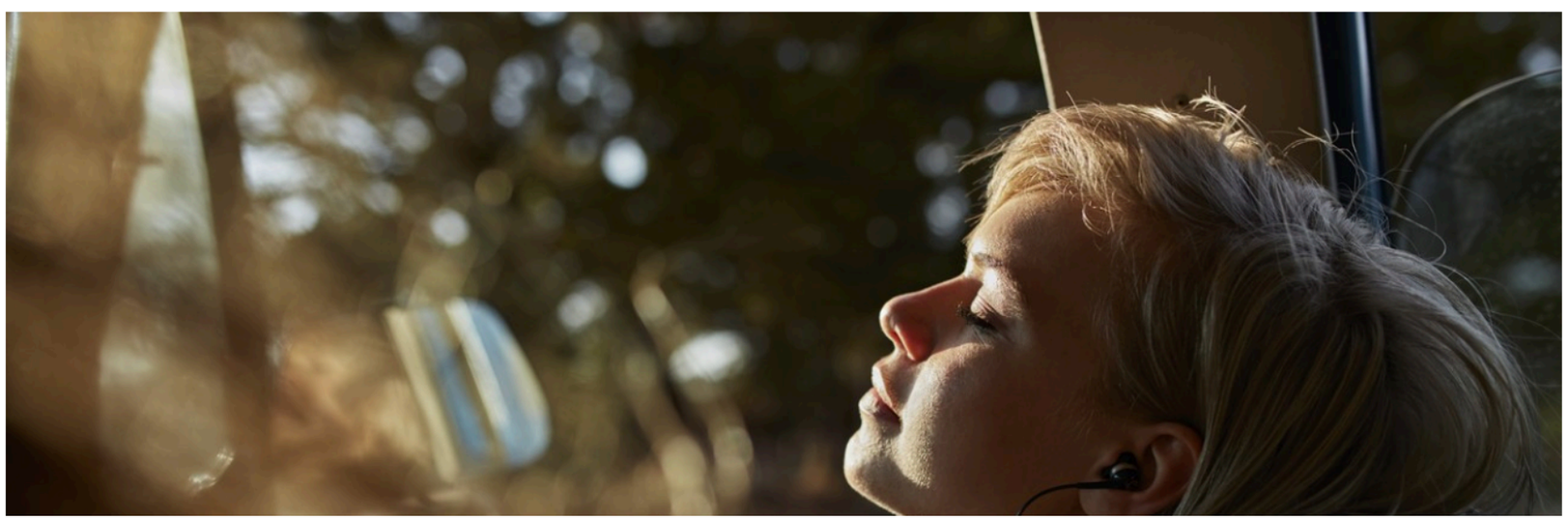
View Challenge


52 Books
(100%)
39 Books
(75%)
26 Books
(50%)
13 Books
(25%)
0
Q1
Q2
Q3
Q4
EOY

DAILY SHOUTS

LISTENING TO AUDIOBOOKS IS JUST AS GOOD AS READING, IF NOT BETTER, SO BACK THE HELL OFF

By Alex Watt October 5, 2017



Find Quotes

Find quotes by keyword, author
Search

All Quotes | My Quotes | A

BROWSE BY TAG

Search for a Tag Sea

Love Quotes 66k
Life Quotes 53.5k
Inspirational Quotes 52k
Humor Quotes 33.5k
Philosophy Quotes 22.5k
God Quotes 17.5k
Truth Quotes 17.5k
Wisdom Quotes 15.5k
Inspirational Quotes Quotes 15k
Happiness Quotes 14k
Romance Quotes 14k
Hope Quotes 13.5k
Death Quotes 13k
Quotes Quotes 12.5k
Poetry Quotes 12.5k

https://www.goodreads.com/quotes/search?q=Fluke%3A+Or%2C+I+Know+Why+the...

goodreads | Search books

Home | My Books | Browse | Community

Find Quotes

Find quotes by keyword, author | Search

All Quotes | My Quotes | A

RESULTS FOR "FLUKE: OR, I KNOW WHY THE WINGED WHALE SINGS" (SHOWING 1-20 OF 45) (0.05 SECONDS)

"There's some heinous fuckery goin' on mon."
— Christopher Moore, Fluke: Or, I Know Why the Winged Whale Sings

tags: fluke, humor

399 likes | Like

"She's so small, yet she contains so much evil."
— Christopher Moore, Fluke: Or, I Know Why the Winged Whale Sings

tags: humor

201 likes | Like

"Actually, orcas aren't quite as complex as scientists imagine. Most killer whales are just four tons of doofus dressed up like a police car."
— Christopher Moore, Fluke: Or, I Know Why the Winged Whale Sings

tags: christopher-moore, fluke

148 likes | Like

BROWSE BY TAG

Search for a Tag | Sea

Love Quotes 66k
Life Quotes 53.5k
Inspirational Quotes 52k
Humor Quotes 33.5k
Philosophy Quotes 22.5k
God Quotes 17.5k
Truth Quotes 17.5k
Wisdom Quotes 15.5k
Inspirational Quotes Quotes 15k
Happiness Quotes 14k
Romance Quotes 14k
Hope Quotes 13.5k
Death Quotes 13k
Quotes Quotes 12.5k
Poetry Quotes 12.5k

goodreads

My Books Browse Community

Find Quotes

Find quotes by keyword, author

RESULTS FOR "FLUKE: OR, I KNOW
SECONDS)

"There's some heinous f
— Christopher Moore, Fluk

tags: fluke, humor

"She's so small, yet she
— Christopher Moore, Fluk

tags: humor

"Actually, orcas aren't q
Most killer whales are ju
police car."
— Christopher Moore, Fluk

tags: christopher-moore, fluke

"Shoes off in the whale!
anus."
— Christopher Moore, Fluk

tags: humor

"Animals might put up v
eventually kill you for it.
— Christopher Moore, Fluk

“Animals might put up with that smiley shit, but people will eventually kill you for it.”

— Christopher Moore, Fluke: Or, I Know Why the Winged Whale Sings

Like

59 likes

“All killer whales are named Kevin. You knew that, right?”

— Christopher Moore, Fluke: Or, I Know Why the Winged Whale Sings

Like

tags: humor, science

26 likes

“Animals might put up with that smiley shit, but people will eventually kill you for it.”

— Christopher Moore, Fluke: Or, I Know Why the Winged Whale Sings

Like

59 likes

“All killer whales are named Kevin. You knew that, right?”

— Christopher Moore, Fluke: Or, I Know Why the Winged Whale Sings

tags: humor, science

Like

26 likes

```
import requests


book = 'Fluke: Or, I Know Why the Winged Whale Sings'

payload = {'q': book, 'commit': 'Search'}
r = requests.get('https://www.goodreads.com/quotes/search',
                 params=payload)
```

```
import requests
from bs4 import BeautifulSoup

book = 'Fluke: Or, I Know Why the Winged Whale Sings'

payload = {'q': book, 'commit': 'Search'}
r = requests.get('https://www.goodreads.com/quotes/search',
                 params=payload)
soup = BeautifulSoup(r.text, 'html.parser')

for s in soup(['script']):
    s.decompose()

soup.find_all(class_='quoteText')
```

```python
import requests
from bs4 import BeautifulSoup

book = 'Fluke: Or, I Know Why the Winged Whale Sings'

payload = {'q': book, 'commit': 'Search'}
r = requests.get('https://www.goodreads.com/quotes/search',
                 params=payload)
soup = BeautifulSoup(r.text, 'html.parser')
```

```
</div>, <div class="quoteText">
      “Animals might put up with that smiley shit, but people will eventually kill you for it.”
  <br/>  –
    <a class="authorOrTitle" href="/author/show/16218.Christopher_Moore">Christopher Moore</a>,
    <span id="quote_book_link_33441">
<a class="authorOrTitle" href="/work/quotes/1684116">Fluke: Or, I Know Why the Winged Whale Sin
</span>

</div>, <div class="quoteText">
      “All killer whales are named Kevin. You knew that, right?”
  <br/>  –
    <a class="authorOrTitle" href="/author/show/16218.Christopher_Moore">Christopher Moore</a>,
    <span id="quote_book_link_33441">
<a class="authorOrTitle" href="/work/quotes/1684116">Fluke: Or, I Know Why the Winged Whale Sin
</span>
```

```
s = soup.find_all(class_='quoteText')[5]
```

```
<div class="quoteText">
      “All killer whales are named Kevin. You knew that, right?”
  <br/>  –
    <a class="authorOrTitle" href="/author/show/16218.Christopher_Moore">Christopher Moore</a>,
    <span id="quote_book_link_33441">
<a class="authorOrTitle" href="/work/quotes/1684116">Fluke: Or, I Know Why the Winged Whale Sings</a>
</span>

</div>
```

```
s = soup.find_all(class_='quoteText')[5]
```

```
<div class="quoteText">
      “All killer whales are named Kevin. You knew that, right?”
  <br/>  -
    <a class="authorOrTitle" href="/author/show/16218.Christopher_Moore">Christopher Moore</a>,
    <span id="quote_book_link_33441">
<a class="authorOrTitle" href="/work/quotes/1684116">Fluke: Or, I Know Why the Winged Whale Sings</a>
</span>

</div>
```

```
s = s.text.replace('\n', '').strip()
```

```
'“All killer whales are named Kevin. You knew that, right?”    -    Christopher Moo
```

```
quote = re.search('“(.*)”', s, re.IGNORECASE).group(1)
```

```
'All killer whales are named Kevin. You knew that, right?'
```

```
meta = re.search('”(.*)', s, re.IGNORECASE).group(1)
```

```
'    -    Christopher Moore,    Fluke: Or, I Know Why the Winged Whale Sings'
```

```
s = soup.find_all(class_='quoteText')[5]
```

```
<div class="quoteText">
      “All killer whales are named Kevin. You knew that, right?”
  <br/>  –
    <a class="authorOrTitle" href="/author/show/16218.C
    <span id="quote_book_link_33441">
<a class="authorOrTitle" href="/work/quotes/1684116">Fl
</span>

</div>
```

```
s = s.text.replace('\n', '').strip()
```

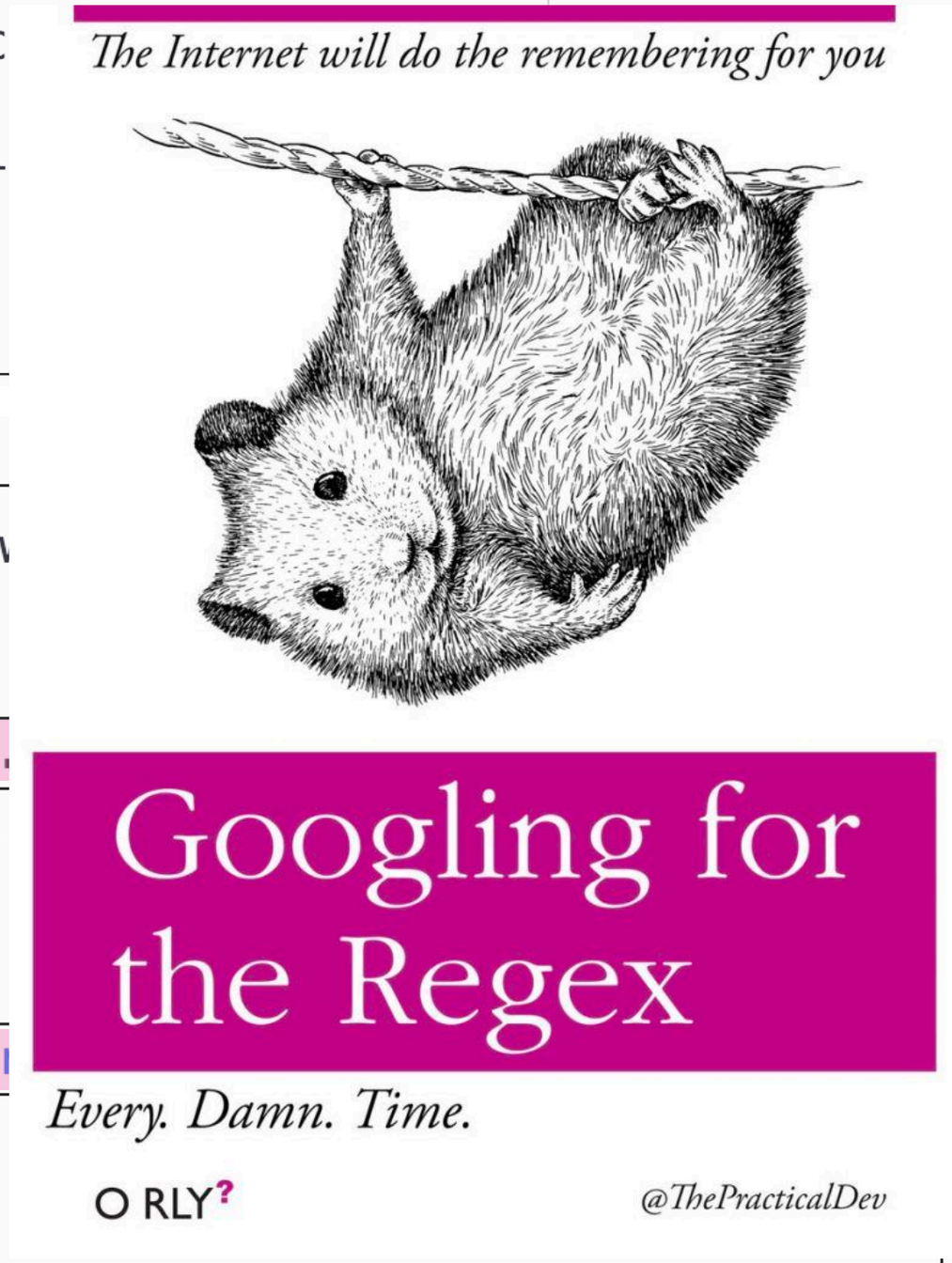
```
'“All killer whales are named Kevin. You knev
```

```
quote = re.search('“(.*)”', s, re.IGNORECASE).
```

```
'All killer whales are named Kevin. You knew
```

```
meta = re.search('”(.*)', s, re.IGNORECASE).gr
```

```
'    –    Christopher Moore,    Fluke: Or, I
```



```
def get_quotes(book):
    payload = {'q': book, 'commit': 'Search'}
    r = requests.get('https://www.goodreads.com/quotes/search', params=payload)
    soup = BeautifulSoup(r.text, 'html.parser')
    # remove script tags
    for s in soup(['script']):
        s.decompose()
    # parse text
    book = {'quote': [], 'author': [], 'title': []}
    for s in soup.find_all(class_='quoteText'):
        s = s.text.replace('\n', '').strip()
        quote = re.search('(.*)', s, re.IGNORECASE).group(1)
        meta = re.search('(.*)', s, re.IGNORECASE).group(1)
        meta = re.sub('[^,.a-zA-Z\s]', '', meta)
        meta = re.sub('\s+', ' ', meta).strip()
        meta = re.sub('^\s', '', meta).strip()
        try:
            author, title = meta.split(',')
        except ValueError:
            author, title = meta, None
        book['quote'].append(quote)
        book['author'].append(author)
        book['title'].append(title)
    return book
```

```
def get_quotes(book):
    payload = {'q': book, 'commit': 'Search'}
    r = requests.get('https://www.goodreads.com/quotes/search', params=payload)
    soup = BeautifulSoup(r.text, 'html.parser')
    # remove script tags
    for s in soup(['script']):
        s.decompose()
    # parse text
    book = {'quote': [], 'author': [], 'title': []}
    for s in soup.find_all(class_='quoteText'):
        s = s.text.replace('\n', '').strip()
        quote = re.search('(.*)', s, re.IGNORECASE).group(1)
        meta = re.search('(.*)', s, re.IGNORECASE).group(1)
        meta = re.sub('[^,.a-zA-Z\s]', '', meta)
        meta = re.sub('\s+', ' ', meta).strip()
        meta = re.sub('^\s', '', meta).strip()
        try:
            author, title = meta.split(',')
        except ValueError:
            author, title = meta, None
        book['quote'].append(quote)
        book['author'].append(author)
        book['title'].append(title)
return book
```

```
def get_quotes(book):
    payload = {'q': book, 'commit': 'Search'}
    r = requests.get('https://www.goodreads.com/quotes/search', params=payload)
    soup = BeautifulSoup(r.text, 'html.parser')
    # remove script tags
    for s in soup(['script']):
        s.decompose()
    # parse text
    book = {'quote': [], 'author': [], 'title': []}
    for s in soup.find_all(class_='quoteText'):
        s = s.text.replace('\n', '').strip()
        quote = re.search('(.*)', s, re.IGNORECASE).group(1)
        meta = re.search('(.*)', s, re.IGNORECASE).group(1)
        meta = re.sub('[^,.a-zA-Z\s]', '', meta)
        meta = re.sub('\s+', ' ', meta).strip()
        meta = re.sub('^\s', '', meta).strip()
        try:
            author, title = meta.split(',')
        except ValueError:
            author, title = meta, None
        book['quote'].append(quote)
        book['author'].append(author)
        book['title'].append(title)
return book
```

```
books = [
    'Fluke: Or, I Know Why the Winged Whale Sings',
    'Shades of Grey Fforde',
    'Neverwhere Gaiman',
    'The Graveyard Book'
]

all_books = {'quote': [], 'author': [], 'title': []}
for b in books:
    print(f"Getting: {b}")
    b = get_quotes(b)
    all_books['author'].extend(b['author'])
    all_books['title'].extend(b['title'])
    all_books['quote'].extend(b['quote'])
```

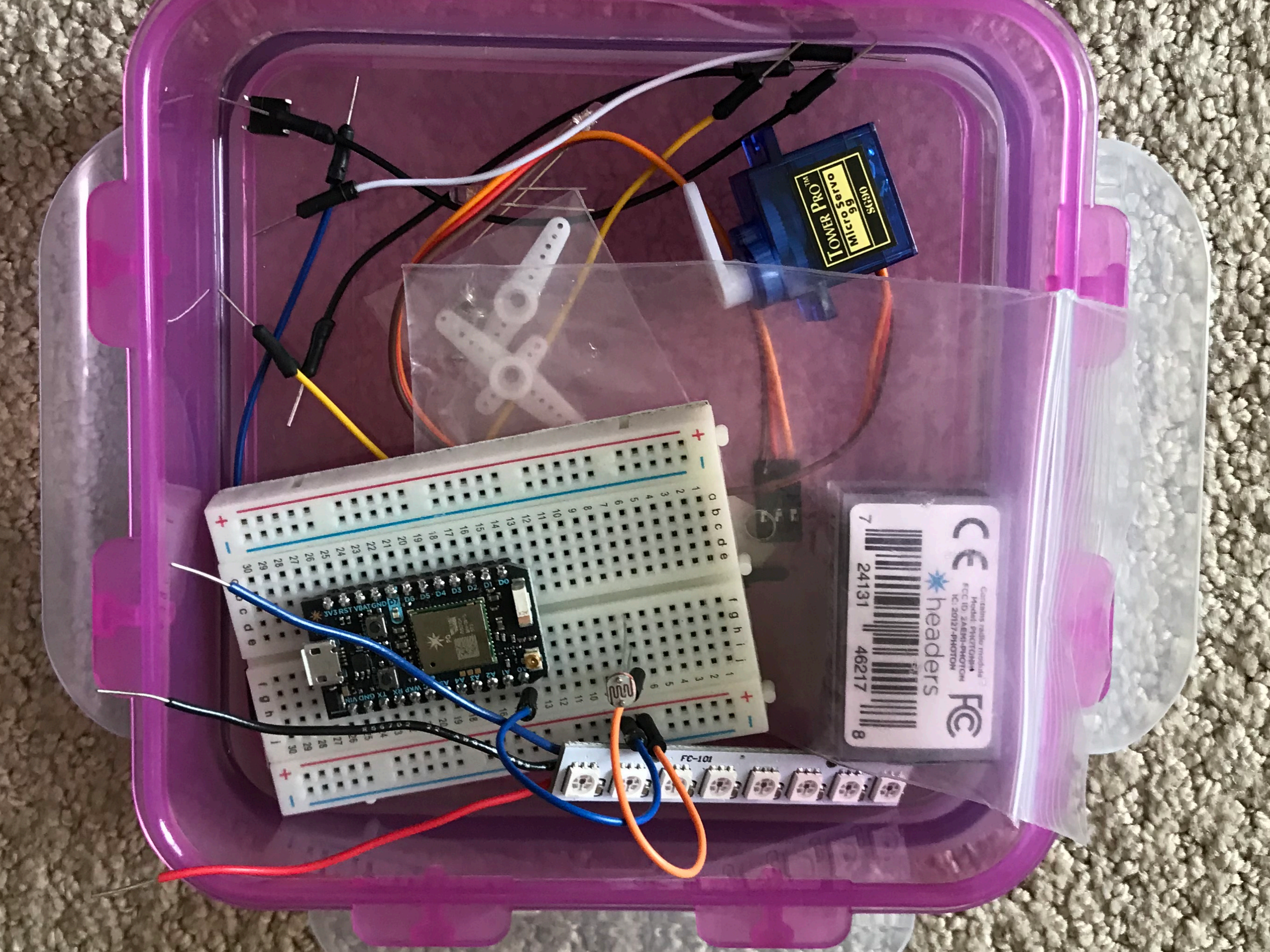
```
books = [
    'Fluke: Or, I Know Why the Winged Whale Sings',
    'Shades of Grey Fforde',
    'Neverwhere Gaiman',
    'The Graveyard Book'
]

all_books = {'quote': [], 'author': [], 'title': []}
for b in books:
    print(f"Getting: {b}")
    b = get_quotes(b)
    all_books['author'].extend(b['author'])
    all_books['title'].extend(b['title'])
    all_books['quote'].extend(b['quote'])

audio = pd.DataFrame(all_books)
audio.to_csv('audio.csv', index=False, encoding='utf-8-sig')
```

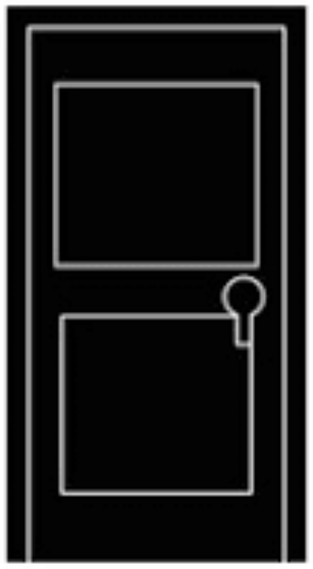
```
audio = pd.DataFrame(all_books)
```

16	Christopher Moore, Fluke Or, I Know Why the Wi...	[Acknowledgments] I recommend them all for fur...
17	Christopher Moore, Fluke Or, I Know Why the Wi...	[Conservation] Barring that, just yell at peop...
18	Christopher Moore, Fluke Or, I Know Why the Wi...	Fine, fuck it," Clay said, tossing the plate i...
19	Christopher Moore, Fluke Or, I Know Why the Wi...	Ninety-five percent of all the species that ha...
20	Jasper Fforde	Okay, this is the wisdom. First, time spent on...
21	Jasper Fforde	The cucumber and the tomato are both fruit; th...
22	Jasper Fforde	The safest course was actually the simplest-do...
23	Jasper Fforde	the best lies to tell are the ones people want...
24	Jasper Fforde	You see? I know where every single book used t...

</curate>


<capture>


TOWER PRO
MicroServo
9g
SG90
3V3 RST VBAT GND D7 D6 D5 D4 D3 D2 D1 D0
TX GND VIN
WKP RX
FC-101
headers
24131
46217



If nonchalant-boomer published, then add row to Max Humber's Google Drive spreadsheet

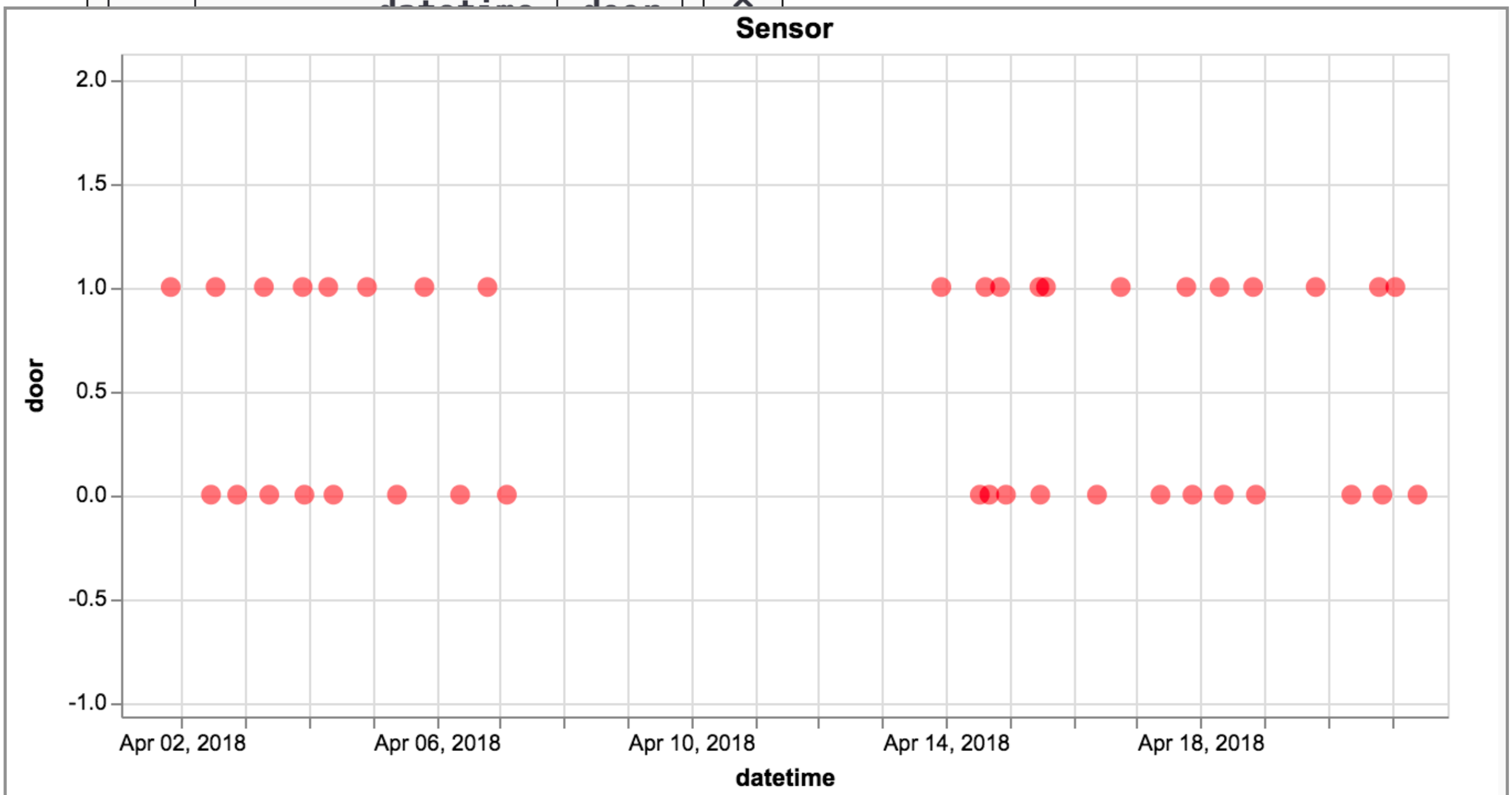
by maxhumber

On

```
df = pd.read_csv('data/in_and_out.csv')
```

	datetime	door
0	2018-04-01 19:52	1
1	2018-04-02 11:02	0
2	2018-04-02 12:46	1
3	2018-04-02 20:55	0
4	2018-04-03 6:58	1
5	2018-04-03 9:01	0
6	2018-04-03 21:36	1
7	2018-04-03 22:15	0
8	2018-04-04 7:14	1
9	2018-04-04 9:12	0
10	2018-04-04 21:50	1
11	2018-04-05 9:09	0
12	2018-04-05 19:29	1

```
df = pd.read_csv('data/in_and_out.csv')
```



11	2018-04-05 9:09	0
12	2018-04-05 19:29	1

```
from datetime import datetime

d = {}
for i, row in df.iterrows():
    date = pd.Timestamp(row['datetime']).to_pydatetime()
    door = row['door']
    d[date] = door
```

```
from traces import TimeSeries as TTS
from datetime import datetime

d = {}
for i, row in df.iterrows():
    date = pd.Timestamp(row['datetime']).to_pydatetime()
    door = row['door']
    d[date] = door

tts = TTS(d)
```

```
from traces import TimeSeries as TTS
from datetime import datetime

d = {}
```

```
tts[datetime(2018, 4, 9)]

tts[datetime(2018, 4, 17, 8)]

tts[datetime(2018, 4, 17, 8, 59)]
```

```
from traces import TimeSeries as TTS
from datetime import datetime

d = {}
```

```
tts[datetime(2018, 4, 9)] 0
```

```
tts[datetime(2018, 4, 17, 8)] 1
```

```
tts[datetime(2018, 4, 17, 8, 59)] 0
```

```
tts.distribution(
    start=datetime(2018, 4,  1),
    end=datetime(2018, 4, 21)
)
```

```
tts.distribution(
    start=datetime(2018, 4,  1),
    end=datetime(2018, 4, 21)
)
```

```
Histogram({0: 0.682, 1: 0.318})
```

$$10^{-fi}$$

12 BEERS OF SUMMER

TICKETS **$20** ON EVENTBRITE

TASTE ALL YOU CAN!

FRI. JULY 17TH | 6:00–10:00

12 BEERS OF SUMMER

TICKETS **$20** ON EVENTBRITE

TASTE ALL YOU CAN!

FRI. JULY 17TH | 6:00–10:00

Fido 5:26 PM 58%

Notes

Ace Hill (Pale Ale)
8:00
M: -5%, S: 5%

-Amsterdam (3 Speed)
7:05
M: -5%, S: 5%

-Beaus (Cherry Stout)
6:30
M: -30%, S: 15%
Fancy

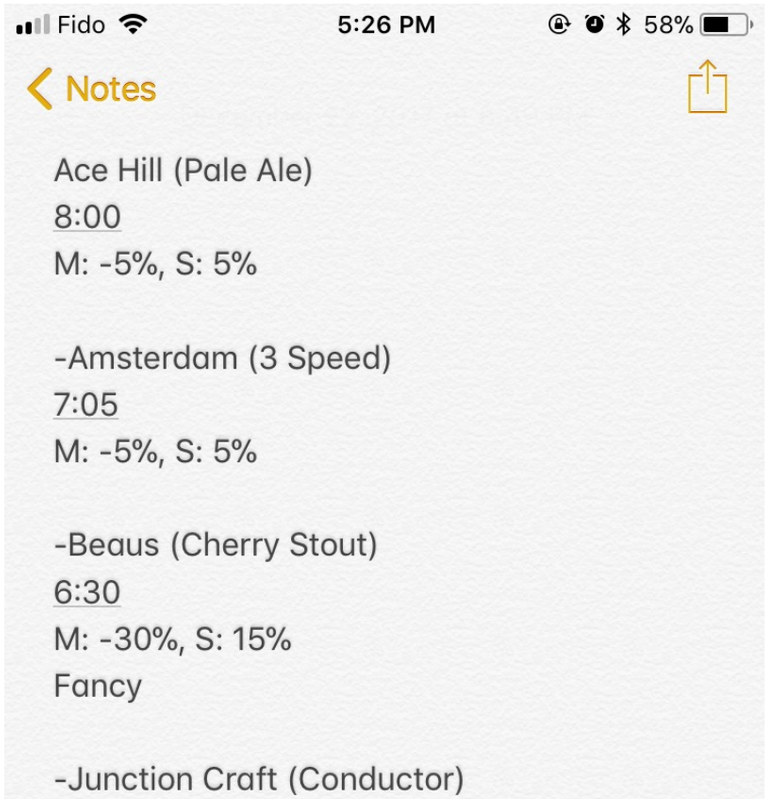
-Junction Craft (Conductor)

12 BEERS OF SUMMER

TICKETS **$20** ON EVENTBRITE

TASTE ALL YOU CAN!

FRI. JULY 17TH | 6:00–10:00



A1 | *fx* time

	A	B	C	D	E	F	G
1	time	beer	ml	abv	Mark	Max	Adam
2	18:47	Honkers, Goose Islanc	200	4.3	1	1	1
3	18:51	Beat the Heat, Black C	200	4.5	1	1	1
4	18:58	Ambre de la Chaudier	200	7	1	1	1
5	19:03	Queen Street 501, Bri	250	5	1	1	1
6	19:14	Farm Table: Grisette,	200	4.9	1	1	2
7	19:26	Loose Lips, Longslice	200	5.5	1	1	1
8	19:48	Food Truck Beer, Hen	400	4.5	1	1	1
9	20:15	Sweetwater Squeeze,	180	3.8	1	1	1
10	20:30	Golden Ale, Sweetgra	200	4.9	1	1	1
11	20:48	Original, Innis & Gunn	180	6.6	0	1	1
12	20:54	Pilsner, Steam Whistle	200	5	1	1	1
13	21:02	Midnight Mass, Brims	60	5.5	1	1	1
14							

```
df = pd.read_csv('data/beer.csv')
df['time'] = pd.to_timedelta(df['time'] + ':00')
```

	time	beer	ml	abv	Mark	Max	Adam
0	18:47:00	Honkers, Goose Island	200	4.3	1	1	1
1	18:51:00	Beat the Heat, Black Oak	200	4.5	1	1	1
2	18:58:00	Ambre de la Chaudiere, Mill Street	200	7.0	1	1	1
3	19:03:00	Queen Street 501, Brickworks	250	5.0	1	1	1
4	19:14:00	Farm Table: Grisette, Beau's	200	4.9	1	1	2
5	19:26:00	Loose Lips, Longslice	200	5.5	1	1	1
6	19:48:00	Food Truck Beer, Henderson	400	4.5	1	1	1
7	20:15:00	Sweetwater Squeeze, Amsterdam	180	3.8	1	1	1
8	20:30:00	Golden Ale, Sweetgrass	200	4.9	1	1	1
9	20:48:00	Original, Innis & Gunn	180	6.6	0	1	1
10	20:54:00	Pilsner, Steam Whistle	200	5.0	1	1	1
11	21:02:00	Midnight Mass, Brimstone	60	5.5	1	1	1

```python
df = pd.melt(df,
    id_vars=['time', 'beer', 'ml', 'abv'],
    value_vars=['Mark', 'Max', 'Adam'],
    var_name='name', value_name='quantity'
)

weight = pd.DataFrame({
    'name': ['Max', 'Mark', 'Adam'],
    'weight': [165, 155, 200]
})

df = pd.merge(df, weight, how='left', on='name')
```

7	20:15:00	Sweetwater Squeeze, Amsterdam	180	3.8	Mark	1	155
8	20:30:00	Golden Ale, Sweetgrass	200	4.9	Mark	1	155
9	20:48:00	Original, Innis & Gunn	180	6.6	Mark	0	155
10	20:54:00	Pilsner, Steam Whistle	200	5.0	Mark	1	155
11	21:02:00	Midnight Mass, Brimstone	60	5.5	Mark	1	155
12	18:47:00	Honkers, Goose Island	200	4.3	Max	1	165
13	18:51:00	Beat the Heat, Black Oak	200	4.5	Max	1	165
14	18:58:00	Ambre de la Chaudiere, Mill Street	200	7.0	Max	1	165
15	19:03:00	Queen Street 501, Brickworks	250	5.0	Max	1	165

```
df['standard_drink'] = (
    df['ml'] * (df['abv'] / 100) * df['quantity']) / 17.2)

# standard drink has 17.2 ml of alcohol

df['cumsum_drinks'] = (
    df.groupby([‘name’])[‘standard_drink'].apply(lambda x: x.cumsum()))

df['hours'] = df['time'] - df[‘time'].min()

df['hours'] = df['hours'].apply(lambda x: x.seconds / 3600)
```

```
df['standard_drink'] = (
    df['ml'] * (df['abv'] / 100) * df['quantity']) / 17.2)
```

time	beer	ml	abv	name	quantity	weight	standard_drink	cumsum_drinks	hours
18:47:00	Honkers, Goose Island	200	4.3	Mark	1	155	0.500000	0.500000	0.000000
18:51:00	Beat the Heat, Black Oak	200	4.5	Mark	1	155	0.523256	1.023256	0.066667
18:58:00	Ambre de la Chaudiere, Mill Street	200	7.0	Mark	1	155	0.813953	1.837209	0.183333
19:03:00	Queen Street 501, Brickworks	250	5.0	Mark	1	155	0.726744	2.563953	0.266667

```
def ebac(standard_drinks, weight, hours):
    # https://en.wikipedia.org/wiki/Blood_alcohol_content
    BLOOD_BODY_WATER_CONSTANT = 0.806
    SWEDISH_STANDARD = 1.2
    BODY_WATER = 0.58
    META_CONSTANT = 0.015

    def lb_to_kg(weight):
        return weight * 0.4535924

    n = BLOOD_BODY_WATER_CONSTANT * standard_drinks * SWEDISH_STANDARD
    d = BODY_WATER * lb_to_kg(weight)

    bac = (n / d - META_CONSTANT * hours)
    return bac
```

```
def ebac(standard_drinks, weight, hours):
    # https://en.wikipedia.org/wiki/Blood_alcohol_content
    BLOOD_BODY_WATER_CONSTANT = 0.806
    SWEDISH_STANDARD = 1.2
    BODY_WATER = 0.58
    META_CONSTANT = 0.015

    def lb_to_kg(weight):
        return weight * 0.4535924

    n = BLOOD_BODY_WATER_CONSTANT * standard_drinks * SWEDISH_STANDARD
    d = BODY_WATER * lb_to_kg(weight)

    bac = (n / d - META_CONSTANT * hours)
    return bac


df['bac'] = df.apply(
    lambda row: ebac(
        row['cumsum_drinks'], row['weight'], row['hours']
    ), axis=1
)
```

	time	bac
12	18:47:00	0.011141
13	18:51:00	0.021799
14	18:58:00	0.038185
15	19:03:00	0.053128
16	19:14:00	0.063073
17	19:26:00	0.074323
18	19:48:00	0.092140
19	20:15:00	0.094251
20	20:30:00	0.103196
21	20:48:00	0.114086
22	20:54:00	0.125540
23	21:02:00	0.127815

	time	bac
12	18:47:00	0.011141
13	18:51:00	0.021799
14	18:58:00	0.038185
15	19:03:00	0.053128
16	19:14:00	0.063073
17	19:26:00	0.074323
18	19:48:00	0.092140
19	20:15:00	0.094251
20	20:30:00	0.103196
21	20:48:00	0.114086
22	20:54:00	0.125540
23	21:02:00	0.127815

	time	bac
12	18:47:00	0.011141
13	18:51:00	0.021799
14	18:58:00	0.038185
15	19:03:00	0.053128
16	19:14:00	0.063073
17	19:26:00	0.074323
18	19:48:00	0.092140
19	20:15:00	0.094251
20	20:30:00	0.103196
21	20:48:00	0.114086
22	20:54:00	0.125540
23	21:02:00	0.127815

Secure | https://bikesharetoronto.com/members/trips

Trip id	Start Station	Start Time	End Station	End Time	Duration
2731930	Niagara St / Richmond St W	2018-05-02 17:14:50	Fort York Blvd / Capreol Crt	2018-05-02 17:22:14	7m 24s
2728077	Fort York Blvd / Capreol Crt	2018-05-02 09:05:40	Niagara St / Richmond St W	2018-05-02 09:14:14	8m 34s
2725769	Queens Quay W / Lower Simcoe St	2018-05-01 20:26:15	Navy Wharf Ct. / Bremner Blvd.	2018-05-01 20:32:55	6m 40s
2724447	Lower Spadina Ave / Lake Shore Blvd	2018-05-01 18:11:27	Queens Quay W / Lower Simcoe St	2018-05-01 18:14:54	3m 27s
2718293	Simcoe St / Queen St W	2018-05-01 06:38:50	Front St W / Blue Jays Way	2018-05-01 06:44:46	5m 56s
2718012	Spadina Ave / Bremner Blvd	2018-04-30 22:39:49	Simcoe St / Queen St W	2018-04-30 22:48:12	8m 23s

Secure | https://bikesharetoronto.com/members/login

BIKE SHARE TORONTO

LOG IN

Username

Password

LOGIN

Forgot Password

JOIN OUR MAILING LIST

Enter email address

SUBSCRIBE

Secure | https://bikesharetoronto.com/members/login

BIKE SHARE TORONTO

LOG IN

input#userName.form-control | 460 × 40

maxhumber

Password

••••••••••••

LOGIN

Forgot Password

Elements Console Sources Network Performance Memory Application Security Audits

```
▼<form method="POST" action="https://bikesharetoronto.com/members/login" accept-charset="UTF-8" _lpchecked="1">
  <input name="_token" type="hidden" value="kJrJ3oaOEpbi1C5VTNQ0O1hvUabWXqXUnpebytNJ">
  ▼<div class="centerpanel clearfix">
    ::before
    ▶<h1>…</h1>
    ▼<div class="form-group fields ">
      <label for="userName" class="label-form">Username</label>
      <input class="form-control" name="userName" type="text" id="userName" style="background-image: url("
```

Styles Computed Event Listeners

Filter :hov .cls +

```
element.style {
  background-image: url(
  background-repeat: ▶ no-repeat;
  background-attachment: scroll;
  background-size: 16px 18px;
  background-position: ▶ 98% 50%;
  cursor: auto;
}
```

.login input, .login .frm submit body.scss:261

Secure | https://bikesharetoronto.com/members/trips

BIKE SHARE TORONTO

PROFILE | BILLING | PAYMENTS | TRIPS | KEY

Start Date

2017-08-01

End Date

2018-04-01

SHOW RESULTS

Elements Console Sources Network Performance Memory Application Security Audits

```
<!DOCTYPE html>
<html lang="en">
▶#shadow-root (open)
▶<head>…</head>
…▼<body> == $0
  ▶<script>…</script>
  ▶<header>…</header>
  ▶<script type="text/javascript">…</script>
  ▶<div class="centerregisters clearfix">…</div>
  ▶<script>…</script>
  ▶<section>…</section>
  ▶<footer>…</footer>
  ▶<script>…</script>
  ▶<script>…</script>
```

Styles Computed Event Listeners

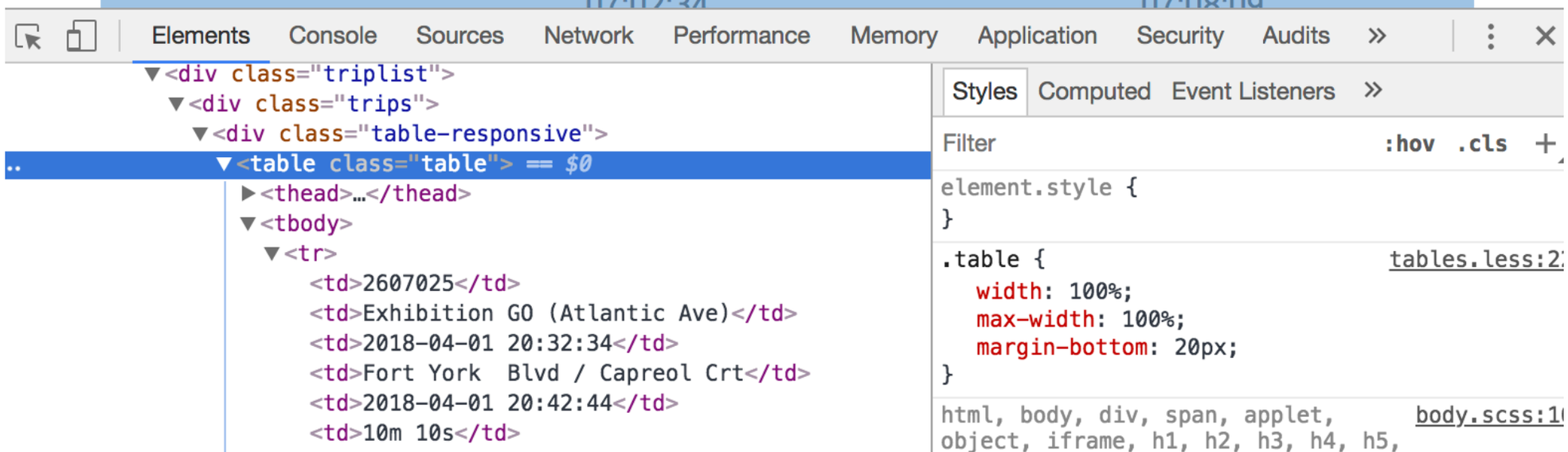
Filter :hov .cls +

```
element.style {
}
body {                                   body.scss:39
  color: #323232;
  font-family: "RobotoRegular";
  font-size: 18px;
  line-height: 1.6;
  width: 100%;
  display: flex;
  min-height: 100vh;
```

SHOW RESULTS

table.table | 690 × 24939

Trip id	Start Station	Start Time	End Station	End Time	Duration
2607025	Exhibition GO (Atlantic Ave)	2018-04-01 20:32:34	Fort York Blvd / Capreol Crt	2018-04-01 20:42:44	10m 10s
2603872	Niagara St / Tecumseth St	2018-03-31 14:06:49	Fort York Blvd / Capreol Crt	2018-03-31 14:11:24	4m 35s
2603473	Fort York Blvd / Capreol Crt	2018-03-31 12:50:31	Niagara St / Tecumseth St	2018-03-31 12:55:55	5m 24s
2602684	Nelson St / Duncan St	2018-03-31 07:02:34	Bremner Blvd / Spadina Ave	2018-03-31 07:08:09	5m 35s



```python
import mechanicalsoup

def fetch_data():
    browser = mechanicalsoup.StatefulBrowser(
        soup_config={'features': 'lxml'},
        raise_on_404=True,
        user_agent='MyBot/0.1: mysite.example.com/bot_info',
    )
```

Secure | https://bikesharetoronto.com/members/login

BIKE SHARE TORONTO

LOG IN

input#userName.form-control | 460 × 40

maxhumber

Password

••••••••••••

LOGIN

Forgot Password

Elements Console Sources Network Performance Memory Application Security Audits

```
▼<form method="POST" action="https://bikesharetoronto.com/members/login" accept-charset="UTF-8" _lpchecked="1">
  <input name="_token" type="hidden" value="kJrJ3oaOEpbi1C5VTNQ0O1hvUabWXqXUnpebytNJ">
  ▼<div class="centerpanel clearfix">
    ::before
    ▶<h1>…</h1>
    ▼<div class="form-group fields ">
      <label for="userName" class="label-form">Username</label>
      <input class="form-control" name="userName" type="text" id="userName" style="background-image: url("
```

Styles Computed Event Listeners

Filter :hov .cls +

```
element.style {
  background-image: url(
  background-repeat: ▶ no-repeat;
  background-attachment: scroll;
  background-size: 16px 18px;
  background-position: ▶ 98% 50%;
  cursor: auto;
}
```

.login input, .login .frm submit body.scss:261

```
import mechanicalsoup

def fetch_data():
    browser = mechanicalsoup.StatefulBrowser(
        soup_config={'features': 'lxml'},
        raise_on_404=True,
        user_agent='MyBot/0.1: mysite.example.com/bot_info',
    )
    browser.open('https://bikesharetoronto.com/members/login')
    browser.select_form('form')
    browser['userName'] = BIKESHARE_USERNAME
    browser['password'] = BIKESHARE_PASSWORD
```

```python
import mechanicalsoup

def fetch_data():
    browser = mechanicalsoup.StatefulBrowser(
        soup_config={'features': 'lxml'},
        raise_on_404=True,
        user_agent='MyBot/0.1: mysite.example.com/bot_info',
    )
    browser.open('https://bikesharetoronto.com/members/login')
    browser.select_form('form')
    browser['userName'] = BIKESHARE_USERNAME
    browser['password'] = BIKESHARE_PASSWORD
    browser.submit_selected()
```

```
import mechanicalsoup

def fetch_data():
    browser = mechanicalsoup.StatefulBrowser(
        soup_config={'features': 'lxml'},
        raise_on_404=True,
        user_agent='MyBot/0.1: mysite.example.com/bot_info',
    )
    browser.open('https://bikesharetoronto.com/members/login')
    browser.select_form('form')
    browser['userName'] = BIKESHARE_USERNAME
    browser['password'] = BIKESHARE_PASSWORD
    browser.submit_selected()
    browser.follow_link('trips')
    browser.select_form('form')
    browser['startDate'] = '2017-10-01'
    browser['endDate'] = '2018-04-01'
    browser.submit_selected()
```

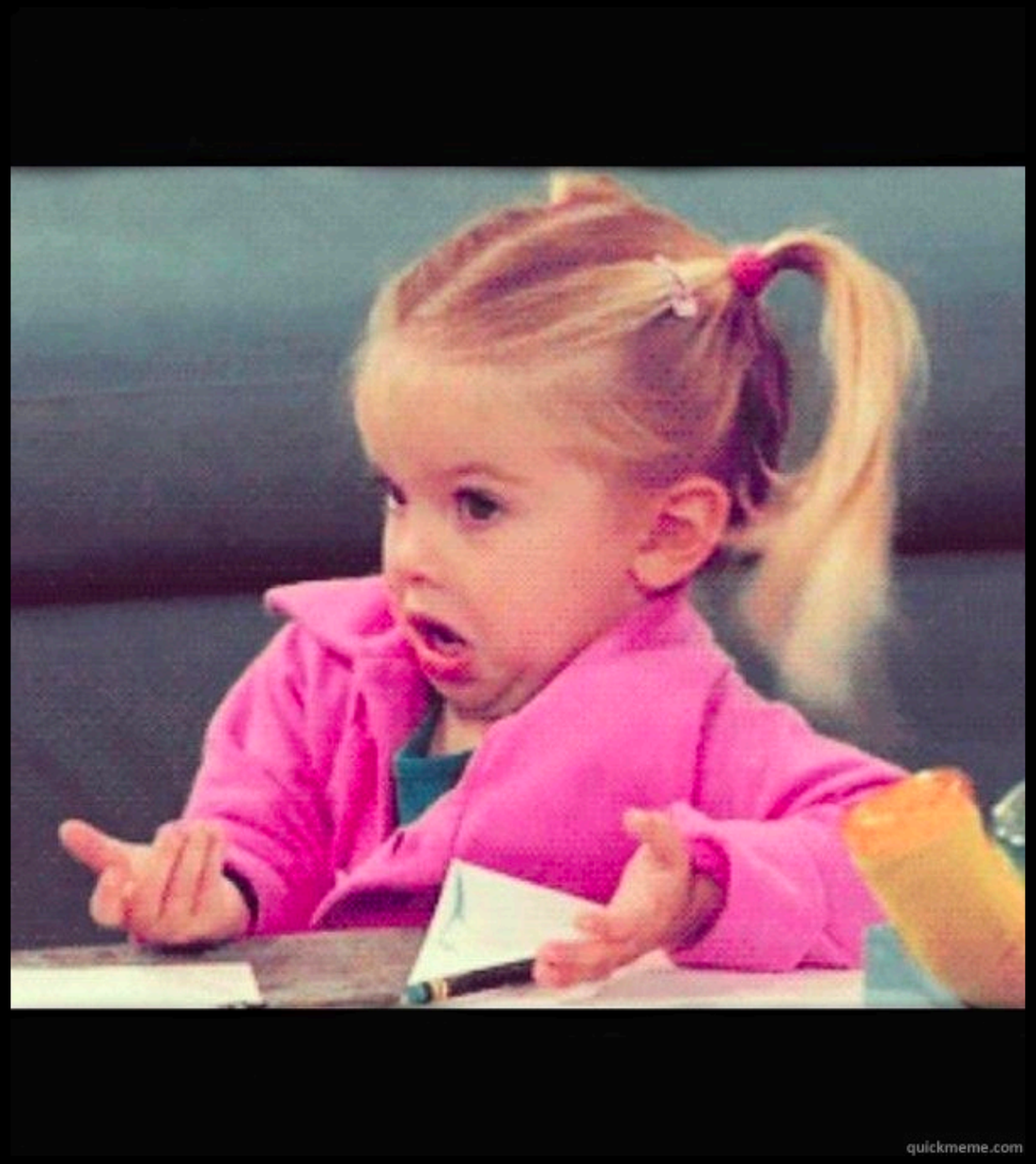
```
import mechanicalsoup

def fetch_data():
    browser = mechanicalsoup.StatefulBrowser(
        soup_config={'features': 'lxml'},
        raise_on_404=True,
        user_agent='MyBot/0.1: mysite.example.com/bot_info',
    )
    browser.open('https://bikesharetoronto.com/members/login')
    browser.select_form('form')
    browser['userName'] = BIKESHARE_USERNAME
    browser['password'] = BIKESHARE_PASSWORD
    browser.submit_selected()
    browser.follow_link('trips')
    browser.select_form('form')
    browser['startDate'] = '2017-10-01'
    browser['endDate'] = '2018-04-01'
    browser.submit_selected()
    html = str(browser.get_current_page())
    df = pd.read_html(html)[0]
    return df

df = fetch_data()
```

```
df = fetch_data()
```

	Trip id	Start Station	Start Time	End Station	End Time	Duratio
0	2607025	Exhibition GO (Atlantic Ave)	2018-04-01 20:32:34	Fort York Blvd / Capreol Crt	2018-04-01 20:42:44	10m 10s
1	2603872	Niagara St / Tecumseth St	2018-03-31 14:06:49	Fort York Blvd / Capreol Crt	2018-03-31 14:11:24	4m 35s
2	2603473	Fort York Blvd / Capreol Crt	2018-03-31 12:50:31	Niagara St / Tecumseth St	2018-03-31 12:55:55	5m 24s
3	2602684	Nelson St / Duncan St	2018-03-31 07:02:34	Bremner Blvd / Spadina Ave	2018-03-31 07:08:09	5m 35s
4	2600654	King St W / Douro St	2018-03-30 13:18:11	Bremner Blvd / Spadina Ave	2018-03-30 13:27:34	9m 23s
5	2600643	Liberty St / Fraser Ave Green P	2018-03-30 13:13:20	King St W / Douro St	2018-03-30 13:16:57	3m 37s
6	2600324	Fort York Blvd / Capreol Crt	2018-03-30 11:53:09	Liberty St / Fraser Ave Green P	2018-03-30 12:05:27	12m 18s

```
['Exhibition GO (Atlantic Ave)',
 'Niagara St / Tecumseth St',
 'Fort York Blvd / Capreol Crt',
 'Nelson St / Duncan St',
 'King St W / Douro St',
 'Liberty St / Fraser Ave',
 'Niagara St / Richmond St W',
 'Queens Quay W / Lower Simcoe St',
 'King St W / Stafford St',
 'Front St W / Bay St (North Side)',
 'Lower Spadina Ave / Lake Shore Blvd',
 'Dundas St W / Crawford St',
 'King St W / Spadina Ave',
 'King St W / Tecumseth St',
 'Tecumseth St / Queen St W',
 'Hanna Ave / Liberty St',
 'Queen St W / Shaw St',
 'Stewart St / Bathurst St',
```

```
['Exhibition GO (Atlantic Ave)',
 'Niagara St / Tecumseth St',
 'Fort York Blvd / Capreol Crt',
 'Nelson St / Duncan St',
 'King St W / Douro St',
 'Liberty St / Fraser Ave',
 'Niagara St / Richmond St W',
 'Queens Quay W / Lower Simcoe St',
 'King St W / Stafford St',
 'Front St W / Bay St (North Side)',
 'Lower Spadina Ave / Lake Shore Blvd',
 'Dundas St W / Crawford St',
 'King St W / Spadina Ave',
 'King St W / Tecumseth St',
 'Tecumseth St / Queen St W',
 'Hanna Ave / Liberty St',
 'Queen St W / Shaw St',
 'Stewart St / Bathurst St',
```



Overview
Get Started
Get API Key

Web Services
Best Practices
Client Libraries

Dev Guides
Geocoding Addresses
Geocoder FAQ

Policies and Terms
Usage Limits
Optimizing Quota Usage

What is Geocoding?

Geocoding is the process of converting addresses (like "1600 Amphitheatre Parkway, Mountain View, CA") into geographic coordinates (like latitude 37.423021 and longitude -122.083739), which you can use to place markers on a map, or position the map.

Reverse geocoding is the process of converting geographic coordinates into a human-readable address.

You can also use the Maps Geocoding API to find the address for a given place ID.

The Maps Geocoding API provides a direct way to access these services via an HTTP request. The following example uses the Geocoding service through the Maps JavaScript API to demonstrate the basic functionality.

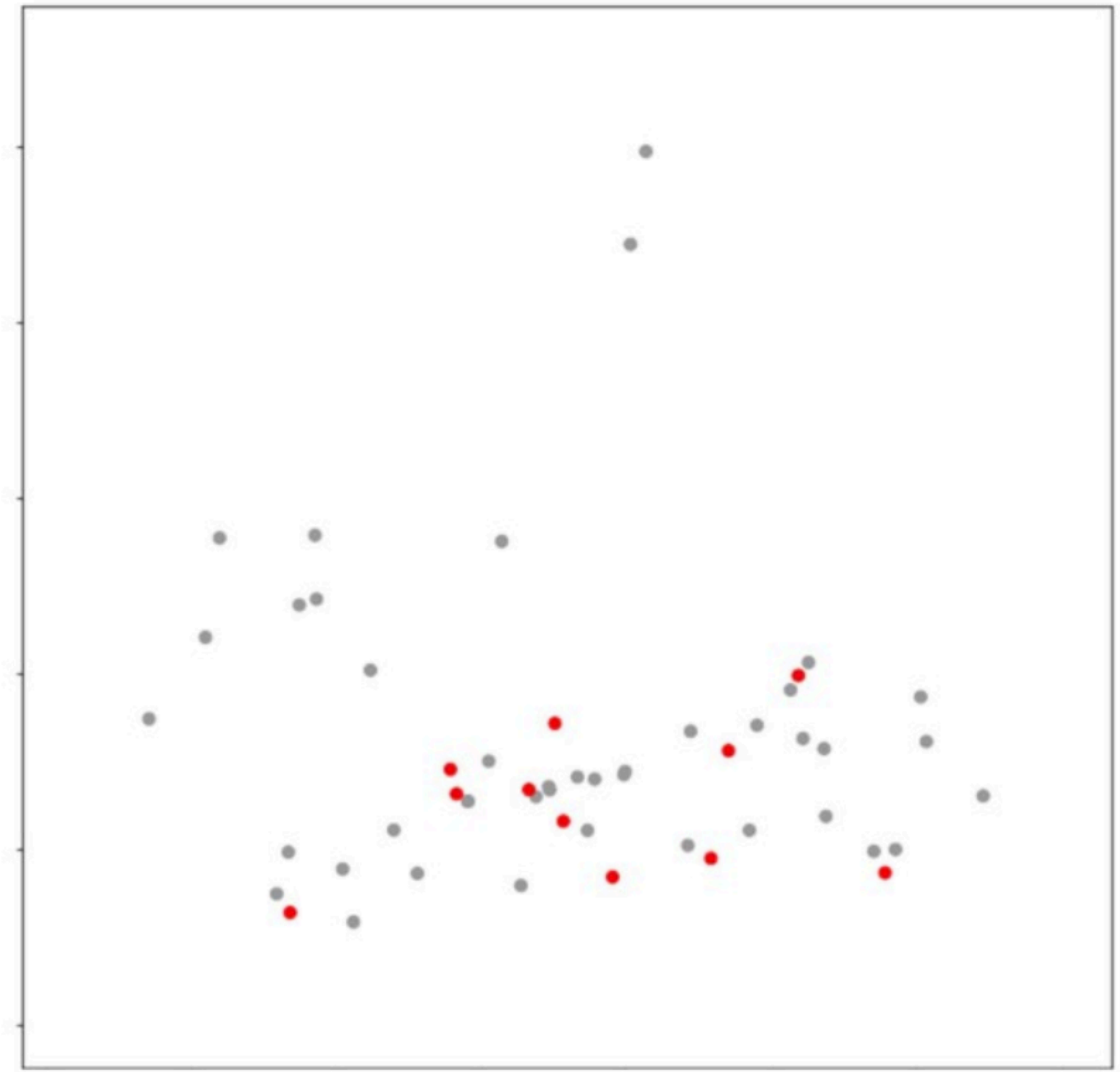
```
def get_geocode(query):
    url = 'https://maps.googleapis.com/maps/api/geocode/json?'
    payload = {'address': query + 'Toronto', 'key': GEOCODING_KEY}
```

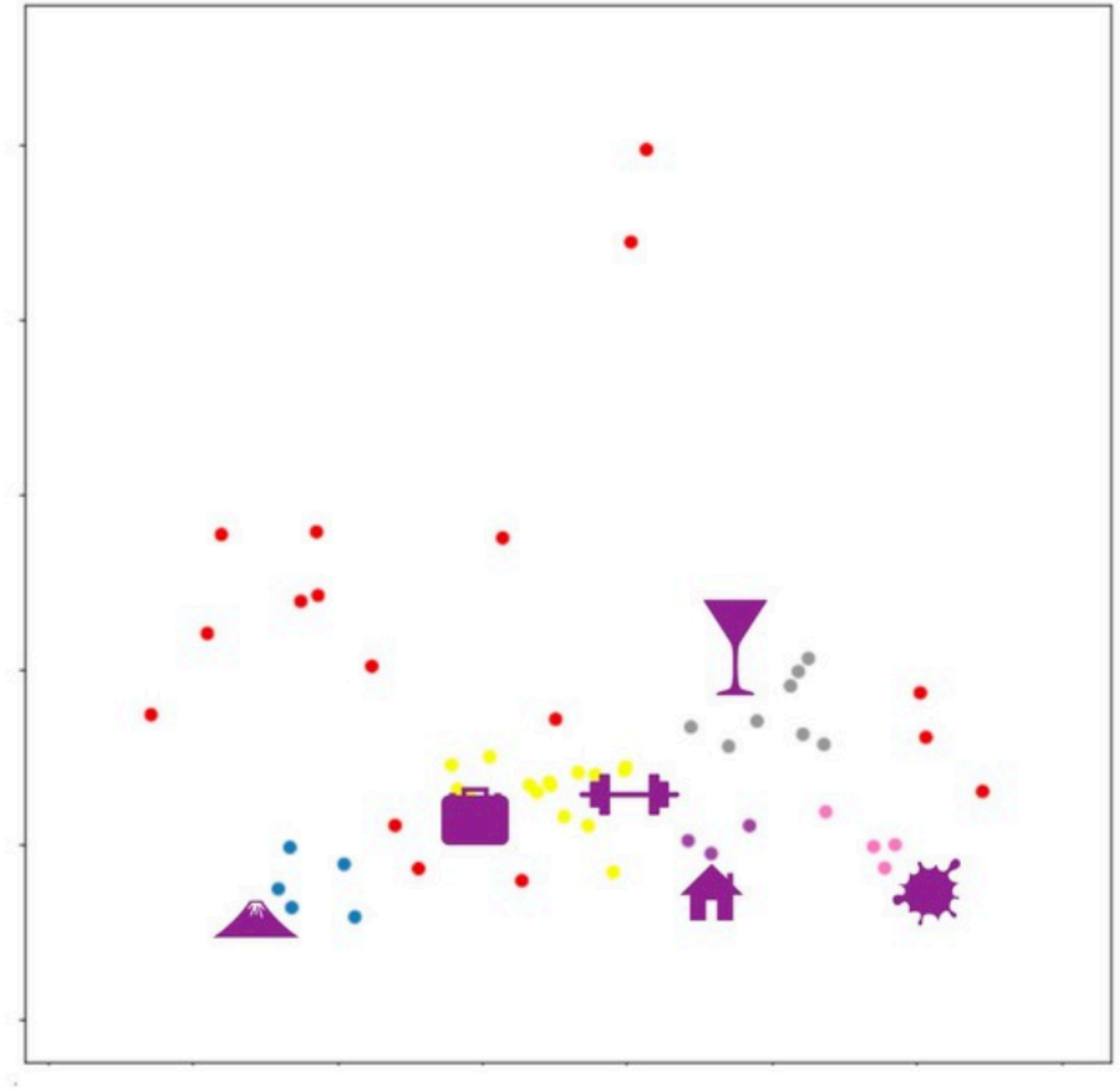
```
def get_geocode(query):
    url = 'https://maps.googleapis.com/maps/api/geocode/json?'
    payload = {'address': query + 'Toronto', 'key': GEOCODING_KEY}
    r = requests.get(url, params=payload)
    results = r.json()['results'][0]
```

```python
def get_geocode(query):
    url = 'https://maps.googleapis.com/maps/api/geocode/json?'
    payload = {'address': query + 'Toronto', 'key': GEOCODING_KEY}
    r = requests.get(url, params=payload)
    results = r.json()['results'][0]
    return {
        'query': query,
        'place_id': results['place_id'],
        'formatted_address': results['formatted_address'],
        'lat': results['geometry']['location']['lat'],
        'lng': results['geometry']['location']['lng']
    }
```

```
get_geocode('joe rockheads')
```

```
 {'formatted_address': '29 Fraser Ave, Toronto, ON M6K 1Y7, Canada',
  'lat': 43.6364182,
  'lng': -79.4231483,
  'place_id': 'ChIJWejHUAY1K4gRHUUQL18qt8E',
  'query': 'joe rockheads'}
```





Branch: master ▾ presentations / anacondacon2018 / 06-0_keras_bonus / model.py

Find file | Copy path

maxhumber re-add anaconda — da627a4 18 days ago

1 contributor

68 lines (50 sloc) | 1.89 KB

Raw | Blame | History

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
from sklearn_pandas import DataFrameMapper, CategoricalImputer
from helpers import DateEncoder
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier

df = pd.read_csv('../max_bike_data.csv')
df['time'] = pd.to_datetime(df['time'])
df = df[(df['pick_up'].notnull()) & (df['drop_off'].notnull())]

TARGET = 'drop_off'
y = df[TARGET].values
X = df.drop(TARGET, axis=1)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

```

maxhumber / **presentations**

Watch 1 | Star 3 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

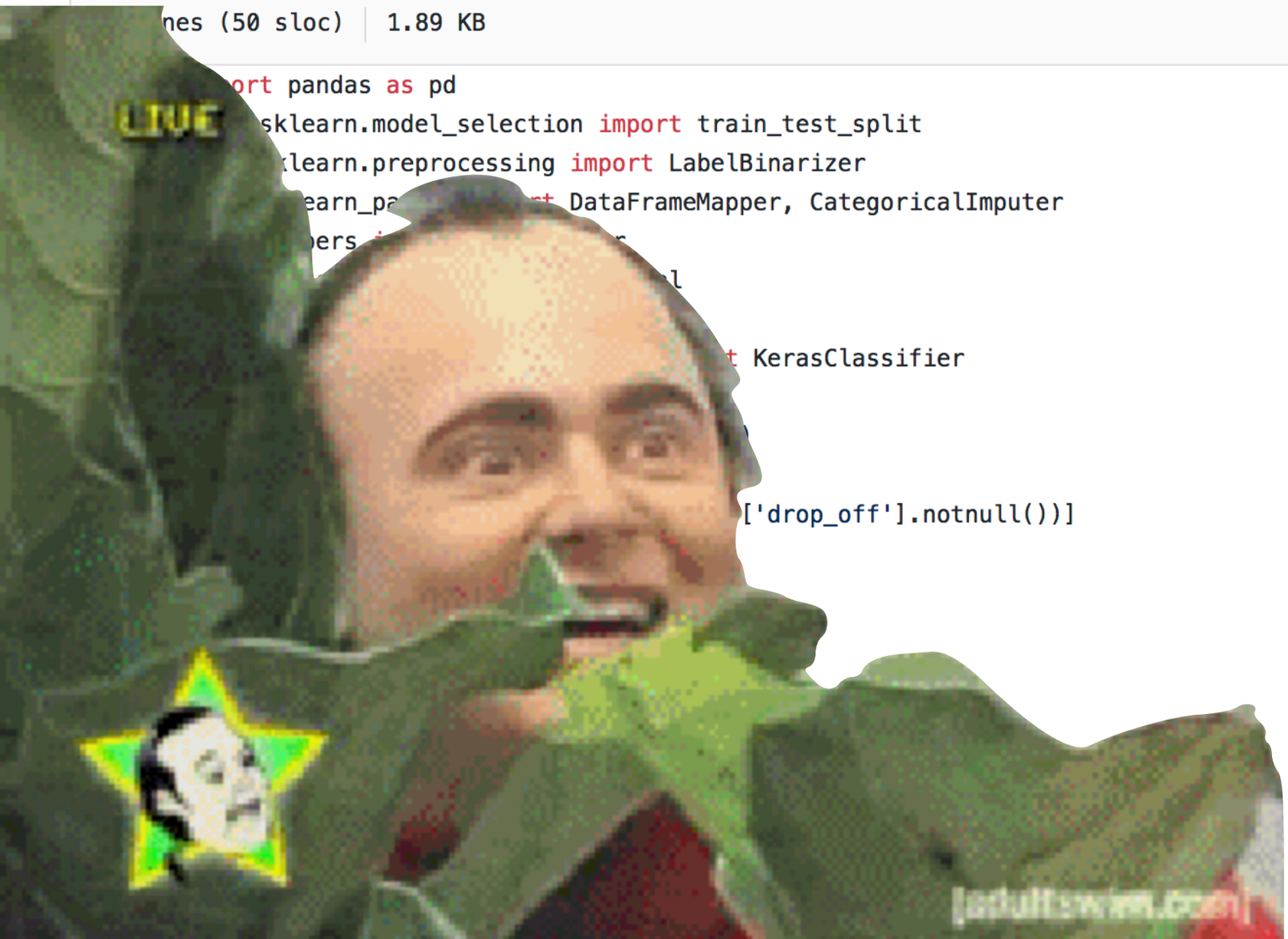
Branch: **master** | **presentations** / anacondacon2018 / 06-0_keras_bonus / **model.py** | Find file | Copy path

maxhumber re-add anaconda — da627a4 18 days ago

1 contributor

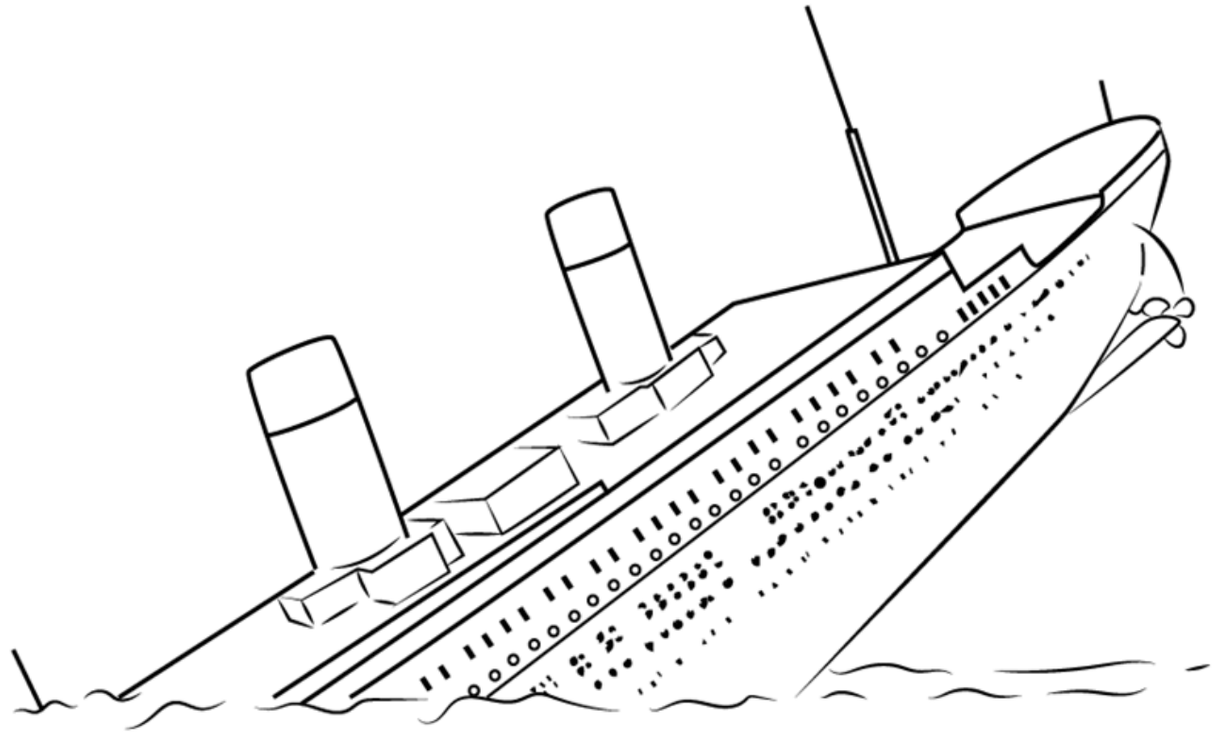
nes (50 sloc) | 1.89 KB | Raw | Blame | History

```
ort pandas as pd
sklearn.model_selection import train_test_split
klearn.preprocessing import LabelBinarizer
earn_pa  t DataFrameMapper, CategoricalImputer
bers i  r
l

t KerasClassifier


['drop_off'].notnull())]
```



</capture>

```python
import pandas as pd
import numpy as np
import seaborn as sns

df = sns.load_dataset('titanic')
```



```python
import pandas as pd
import numpy as np
import seaborn as sns

df = sns.load_dataset('titanic')
df = df[['survived', 'pclass', 'sex', 'age', 'fare']].copy()
df
```

	survived	pclass	sex	age	fare
0	0	3	male	22.0	7.2500
1	1	1	female	38.0	71.2833
2	1	3	female	26.0	7.9250
3	1	1	female	35.0	53.1000
4	0	3	male	35.0	8.0500
5	0	3	male	NaN	8.4583
6	0	1	male	54.0	51.8625
7	0	3	male	2.0	21.0750
8	1	3	female	27.0	11.1333
9	1	2	female	14.0	30.0708
10	1	3	female	4.0	16.7000
11	1	1	female	58.0	26.5500

```python
import pandas as pd
import numpy as np
import seaborn as sns

df = sns.load_dataset('titanic')
df = df[['survived', 'pclass', 'sex', 'age', 'fare']].copy()
df
```

	survived	pclass	sex	age	fare
0	0	3	male	22.0	7.2500
1	1	1	female	38.0	71.2833
2	1	3	female	26.0	7.9250
3	1	1	female	35.0	53.1000
4	0	3	male	35.0	8.0500
5	0	3	male	NaN	8.4583
6	0	1	male	54.0	51.8625
7	0	3	male	2.0	21.0750
8	1	3	female	27.0	11.1333
9	1	2	female	14.0	30.0708
10	1	3	female	4.0	16.7000
11	1	1	female	58.0	26.5500

```
df.rename(
    columns={
        'survived': 'mummified',
        'pclass': 'class',
        'fare': 'debens'
    }, inplace=True)
```

```
df.rename(
    columns={
        'survived': 'mummified',
        'pclass': 'class',
        'fare': 'debens'
    }, inplace=True)

df['debens'] = round(df['debens'] * 10, -1)
```

```
df.rename(
    columns={
        'survived': 'mummified',
        'pclass': 'class',
        'fare': 'debens'
    }, inplace=True)

df['debens'] = round(df['debens'] * 10, -1)
```

```
# inverse
df['mummified'] = np.where(df['mummified'] == 0, 1, 0)
```

```
df.rename(
    columns={
        'survived': 'mummified',
        'pclass': 'class',
        'fare': 'debens'
    }, inplace=True)

df['debens'] = round(df['debens'] * 10, -1)
```

```
# inverse
df['mummified'] = np.where(df['mummified'] == 0, 1, 0)
```

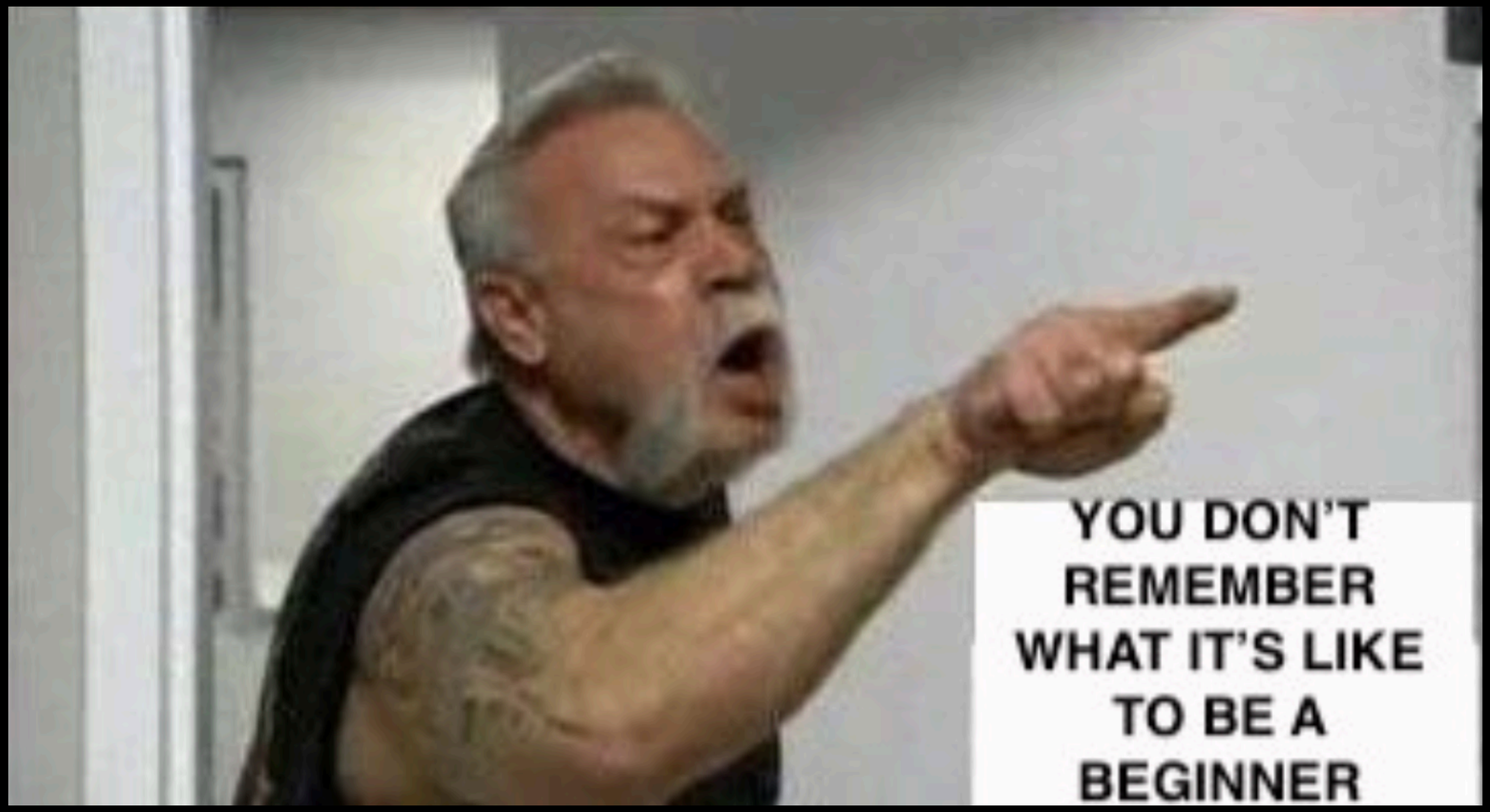
```
df = pd.get_dummies(df)
df = df.drop('sex_female', axis=1)
df.rename(columns={'sex_male': 'male'}, inplace=True)
```

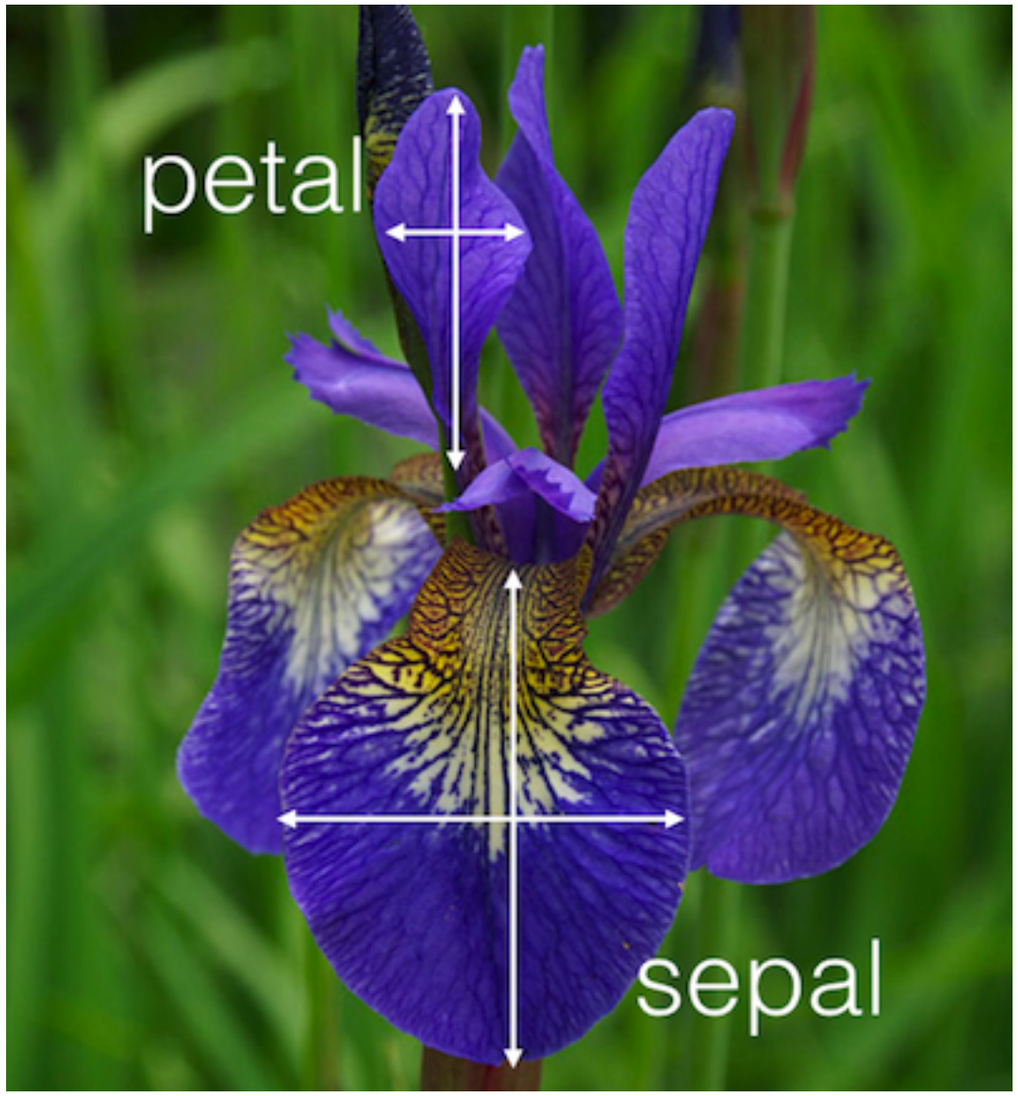
```python
df.rename(
    columns={
        'survived': 'mummified',
        'pclass': 'class',
        'fare': 'debens'
    }, inplace=True)

df['debens'] = round(df['debens'] * 10, -1)
```

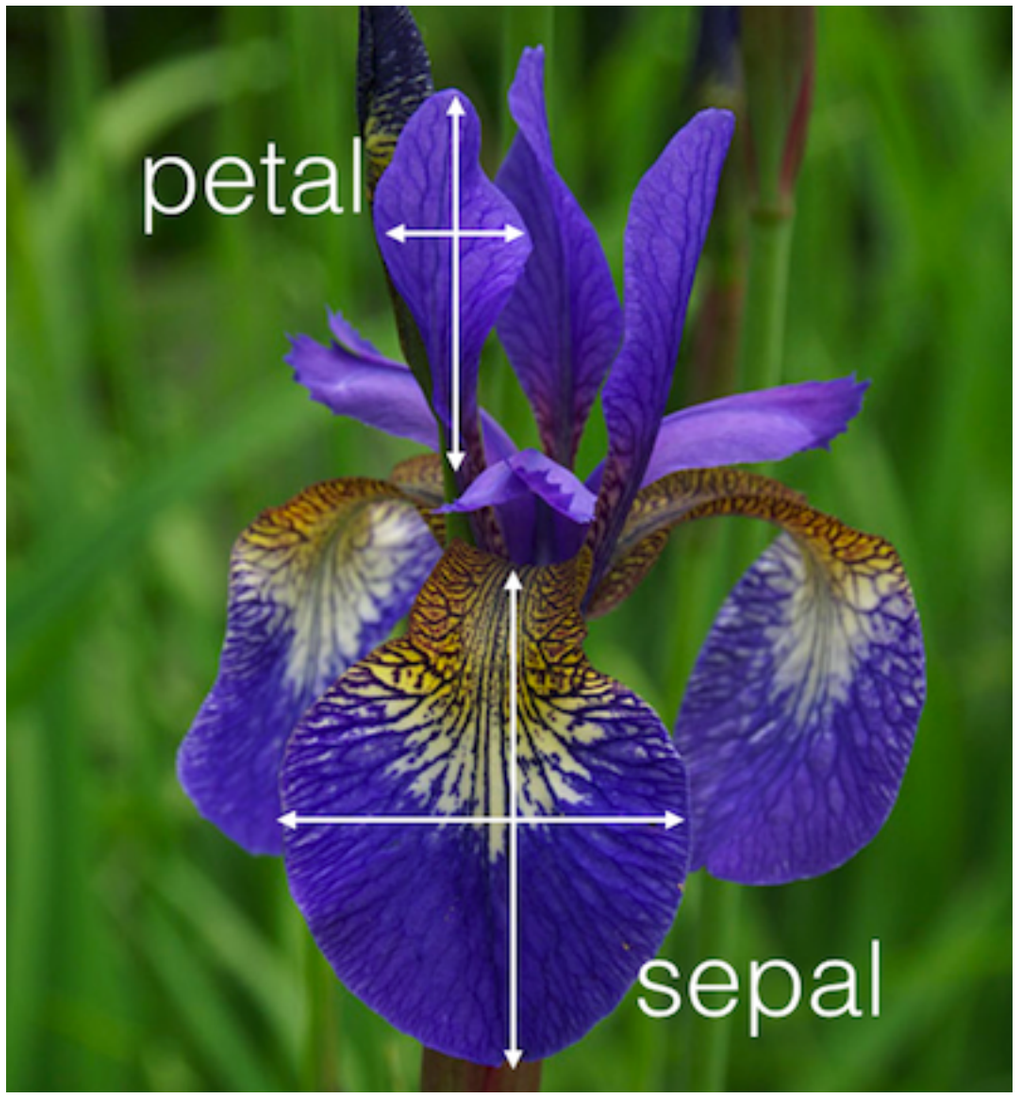
```python
# inverse
df['mummified'] = np.where(df['mummified'] == 0, 1, 0)
```

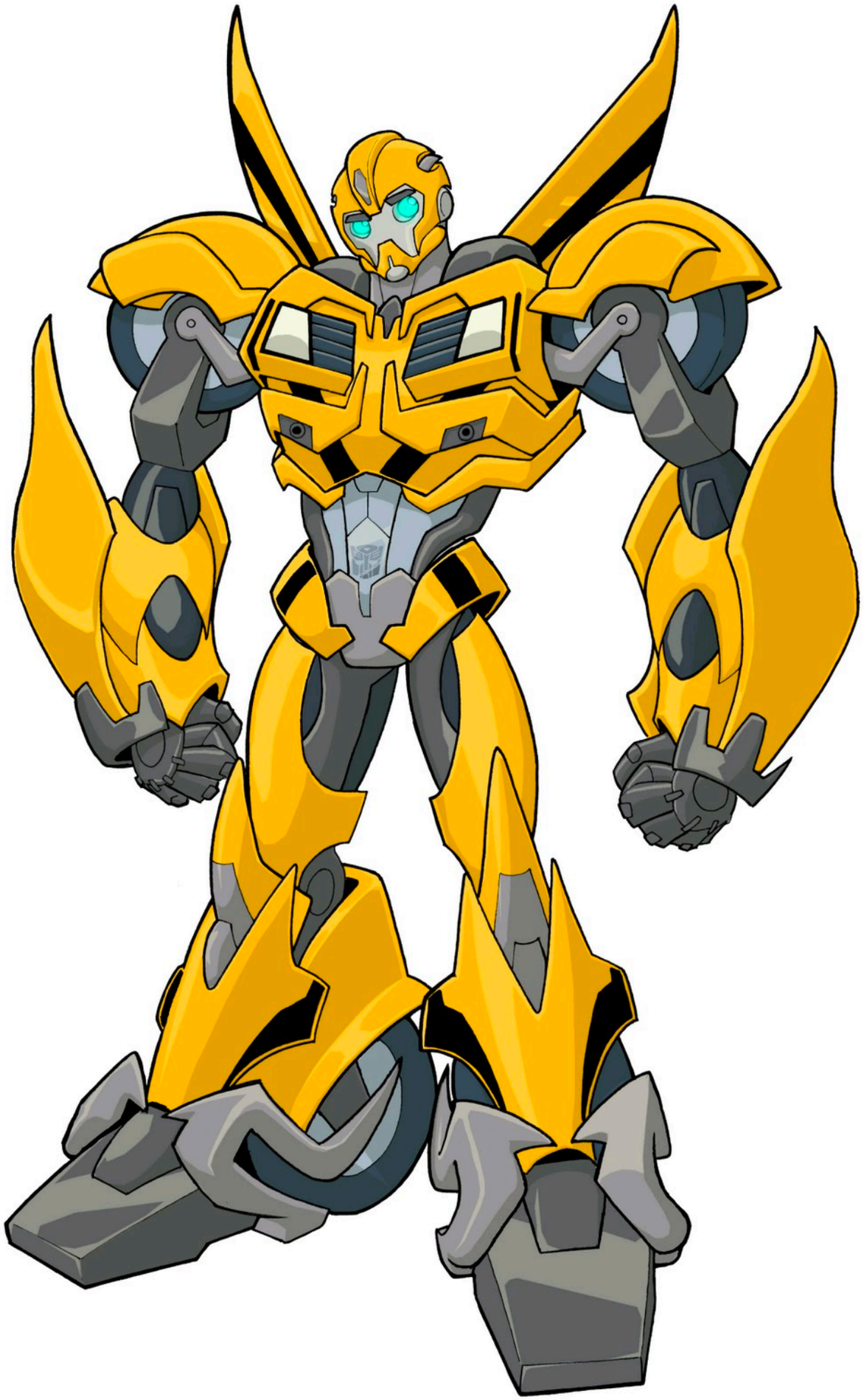
	mummified	class	age	debens	male
0	1	3	22.0	70.0	1
1	0	1	38.0	710.0	0
2	0	3	26.0	80.0	0
3	0	1	35.0	530.0	0
4	1	3	35.0	80.0	1
5	1	3	NaN	80.0	1
6	1	1	54.0	520.0	1

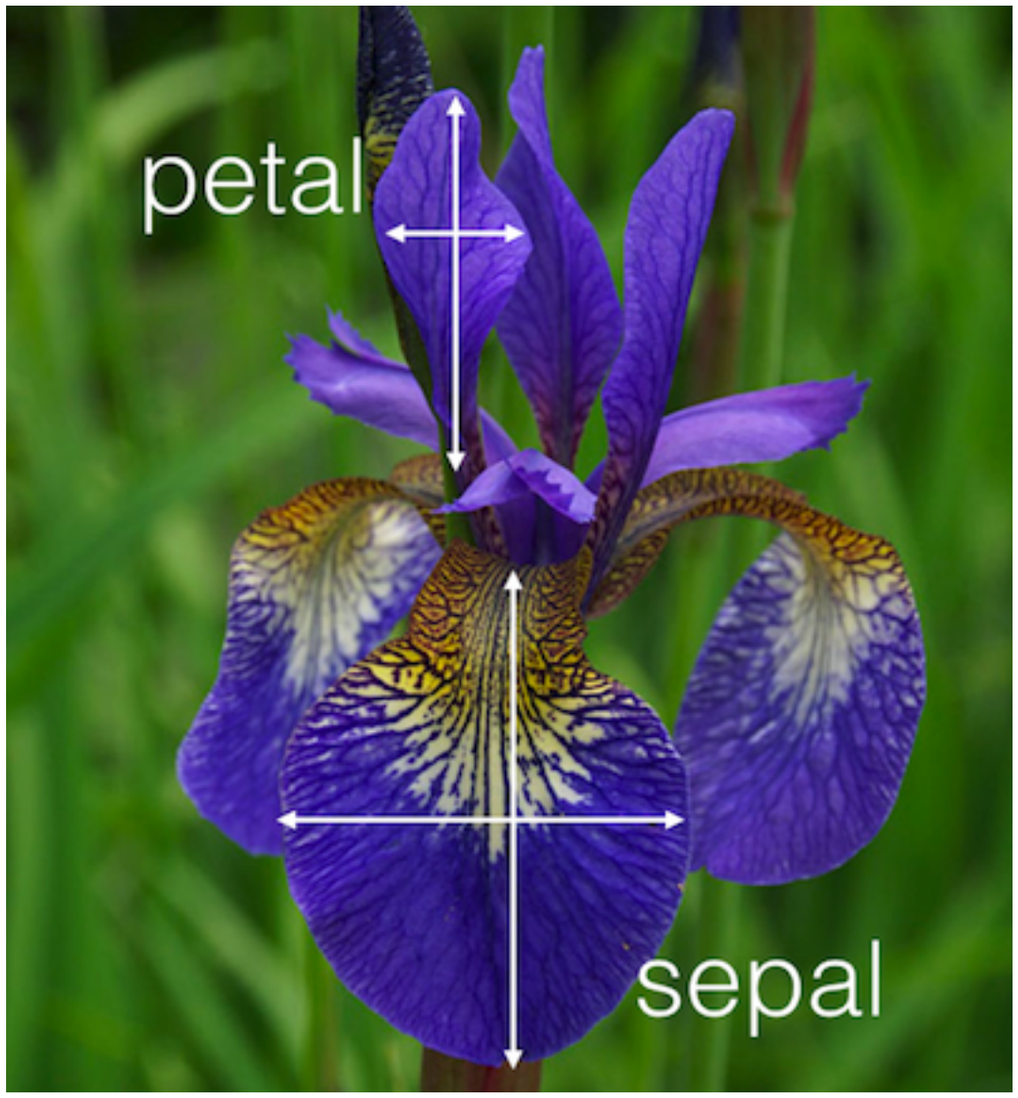
```python
}, inplace=True)
```

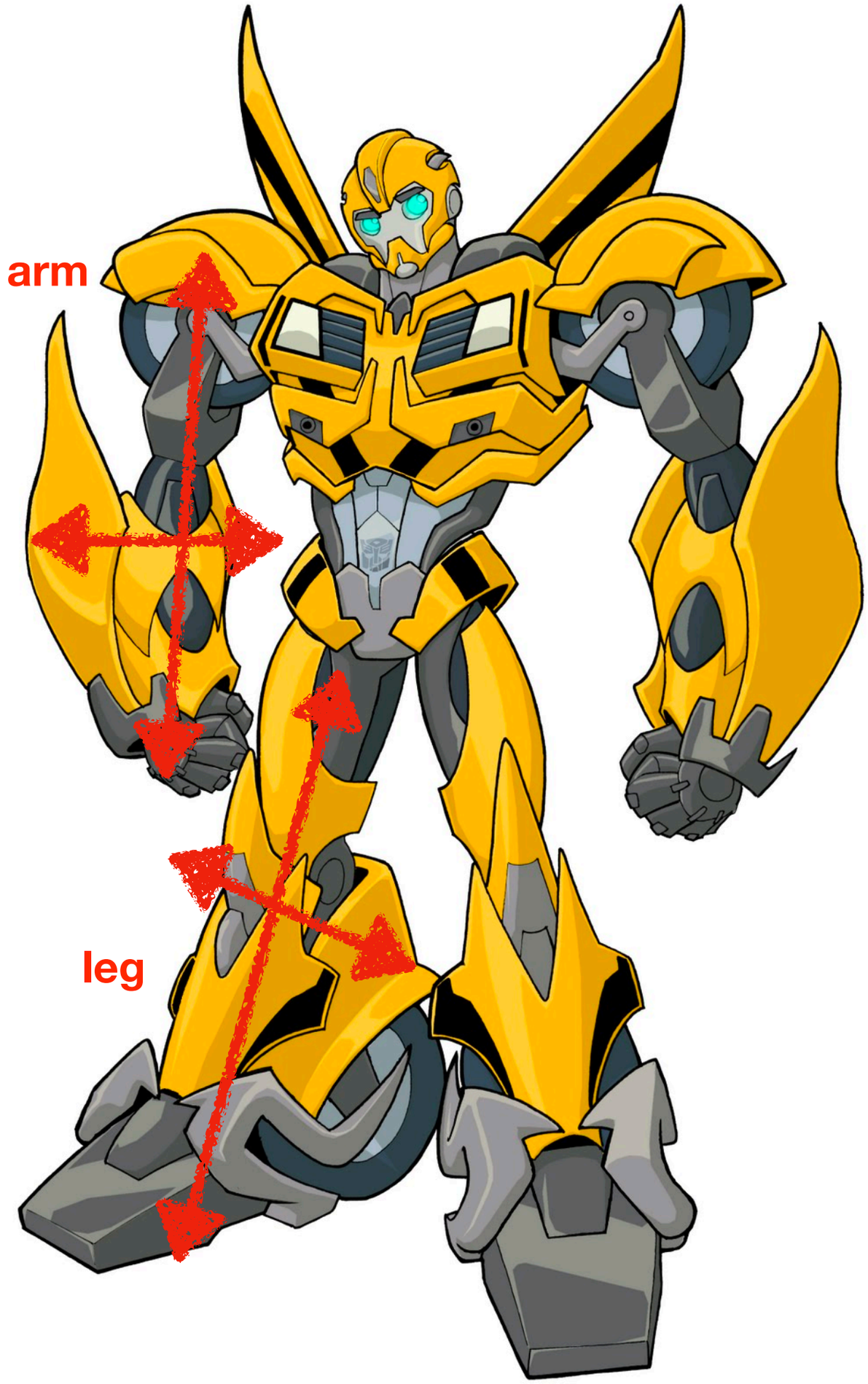

YOU DON'T
REMEMBER
WHAT IT'S LIKE
TO BE A
BEGINNER


petal
sepal


petal
sepal

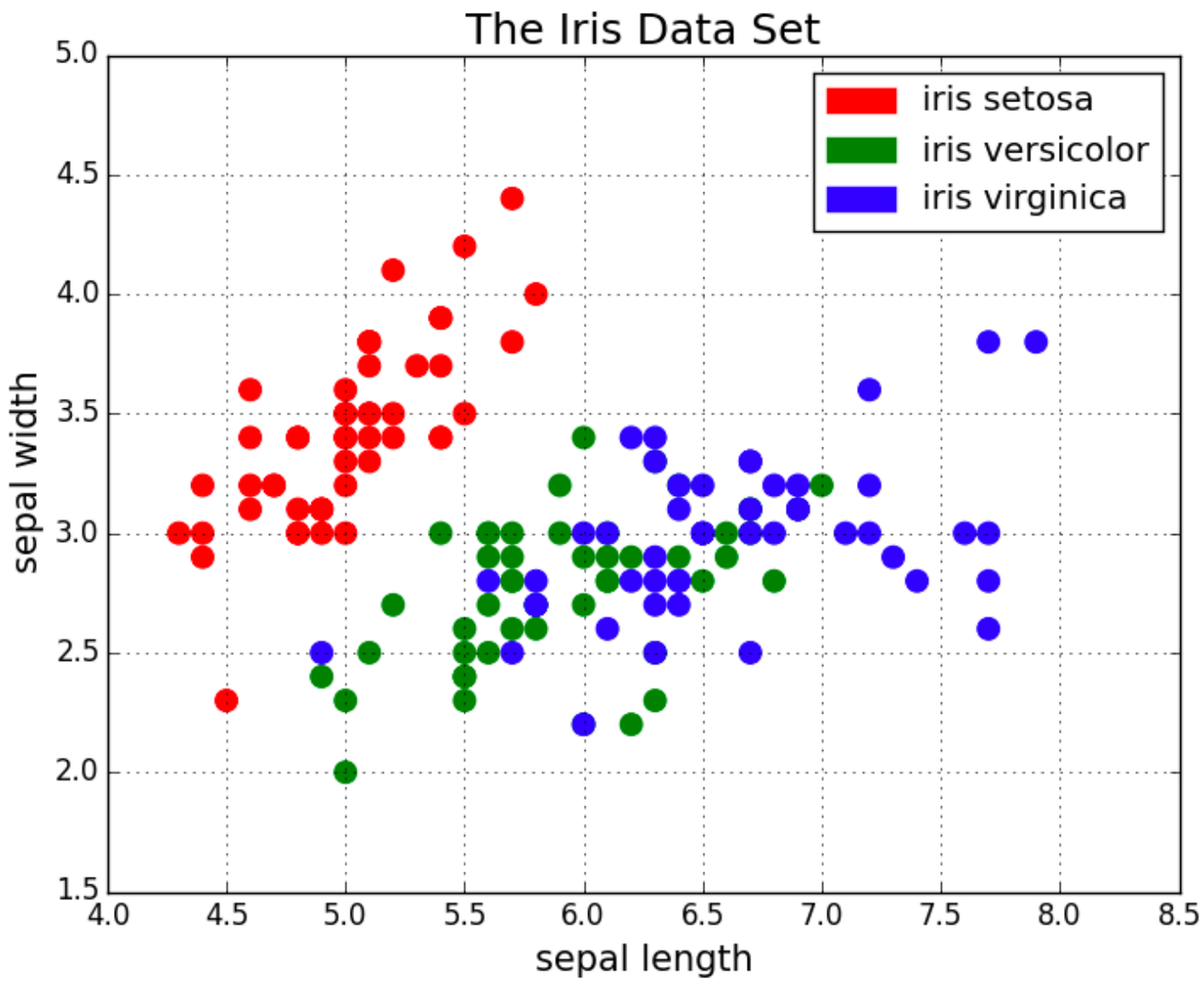


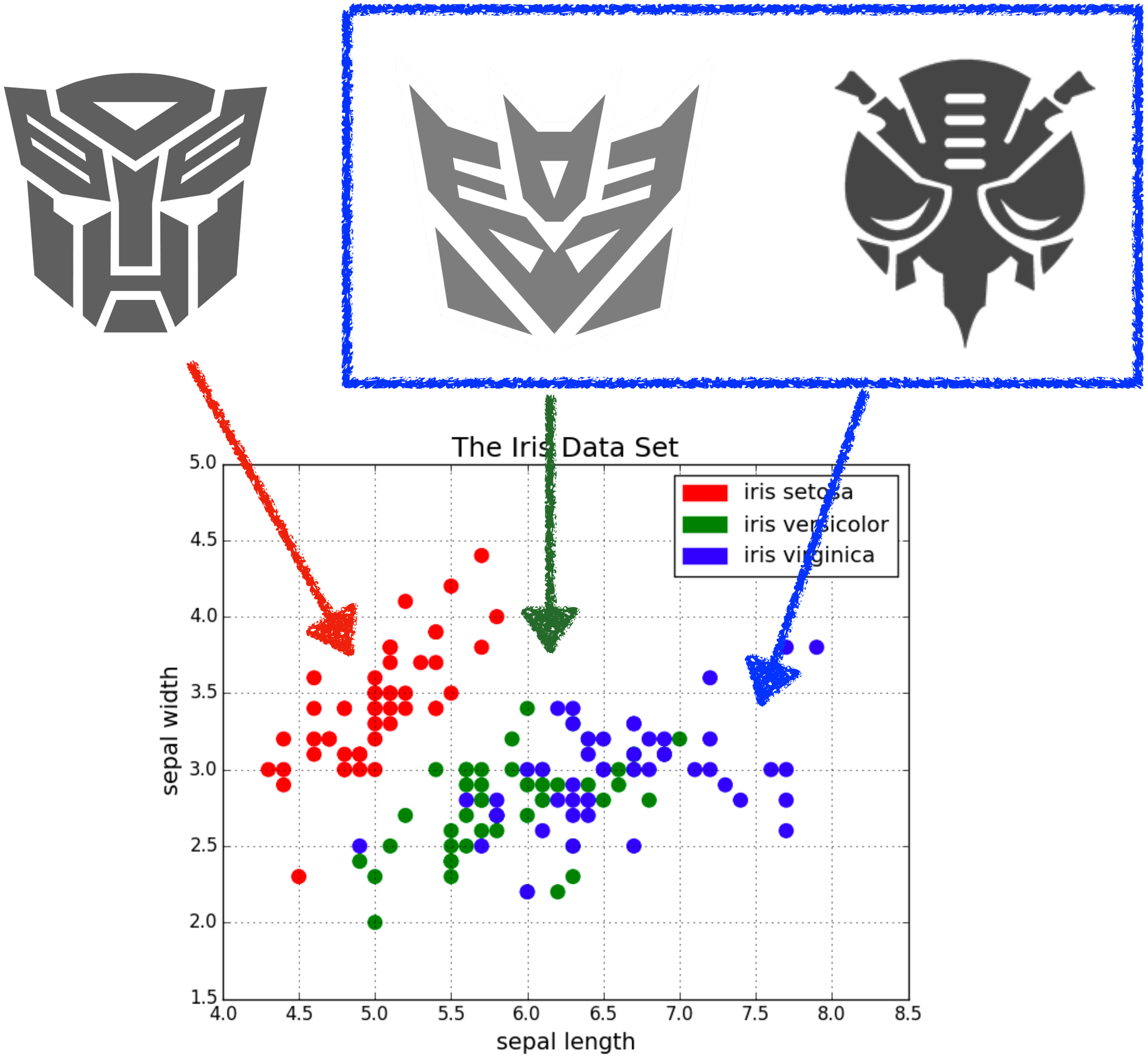

petal
sepal


arm
leg

```python
import seaborn as sns
df = sns.load_dataset('iris')
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa
10	5.4	3.7	1.5	0.2	setosa
11	4.8	3.4	1.6	0.2	setosa
12	4.8	3.0	1.4	0.1	setosa
13	4.3	3.0	1.1	0.1	setosa
14	5.8	4.0	1.2	0.2	setosa


The Iris Data Set
iris setosa
iris versicolor
iris virginica
sepal width
sepal length
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5


The Iris Data Set
iris setosa
iris versicolor
iris virginica
sepal width
sepal length
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5

```
transformers = {
    'setosa': 'autobot',
    'versicolor': 'decepticon',
    'virginica': 'predacon'}

df['species'] = df['species'].map(transformers)
```

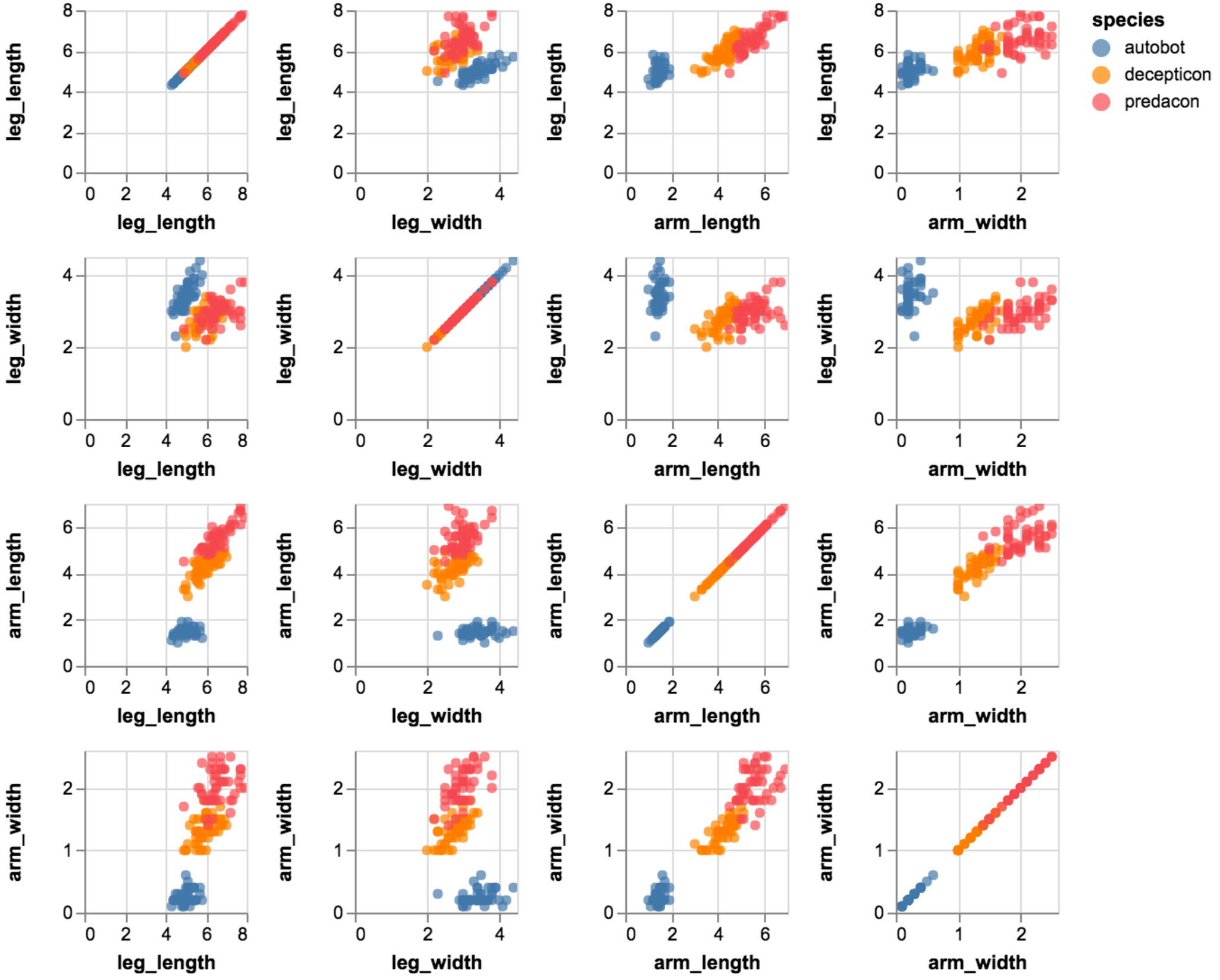
```python
transformers = {
    'setosa': 'autobot',
    'versicolor': 'decepticon',
    'virginica': 'predacon'}

df['species'] = df['species'].map(transformers)
```

```
df.sample(10)
```

	sepal_length	sepal_width	petal_length	petal_width	species
136	6.3	3.4	5.6	2.4	predacon
48	5.3	3.7	1.5	0.2	autobot
39	5.1	3.4	1.5	0.2	autobot
146	6.3	2.5	5.0	1.9	predacon
64	5.6	2.9	3.6	1.3	decepticon
77	6.7	3.0	5.0	1.7	decepticon
144	6.7	3.3	5.7	2.5	predacon
52	6.9	3.1	4.9	1.5	decepticon
91	6.1	3.0	4.6	1.4	decepticon
38	4.4	3.0	1.3	0.2	autobot

```
df.rename(
    columns={
        'sepal_length': 'leg_length',
        'sepal_width': 'leg_width',
        'petal_length': 'arm_length',
        'petal_width': 'arm_width'
    },
    inplace=True
)
```

```
(alt.Chart(df)
    .mark_circle().encode(
        x=alt.X(alt.repeat('column'), type='quantitative'),
        y=alt.Y(alt.repeat('row'), type='quantitative'),
        color='species:N')
    .properties(
        width=90,
        height=90)
    .repeat(
        background='white',
        row=['leg_length', 'leg_width', 'arm_length', 'arm_width'],
        column=['leg_length', 'leg_width', 'arm_length', 'arm_width'])
    .interactive()
)
```

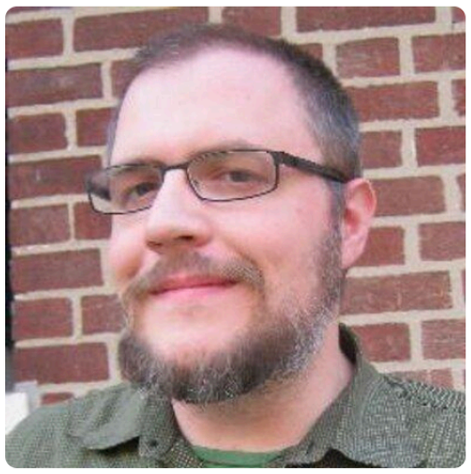

leg_length
leg_width
arm_length
arm_width
species
autobot
decepticon
predacon


END

Jeremy Singer-Vine

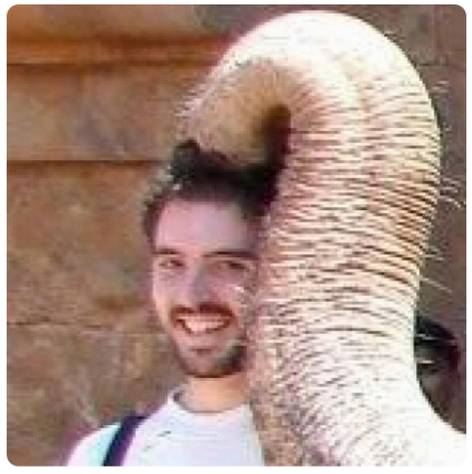
Daniele Faraglia

Jake Vanderplas



Leonard

Kenneth Reitz



Matthieu Moy

Mike Stringer


#ODS

"You suck at Git. And logging. But it's not your fault."

```
pip install mummify
```

maxhumber