ANACONDACON

Harness the Power of Data Science

Austin, Texas | April 8-11, 2018

Data Engineering for Data Scientists

Max Humber

David Smith
@revodavid

The art of writing a talk proposal that's interesting enough for the conference organizers to accept, but noncommittal enough for whatever I will actually want to talk about 6 months from now.

10:05 AM - 23 Feb 2018

49 Retweets **265** Likes

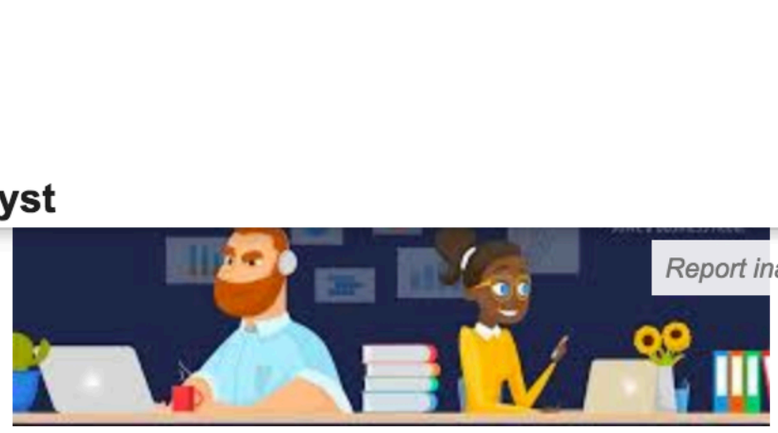
When models and data applications are pushed to production, they become brittle black boxes that can and will break. In this talk you'll learn how to **one-up** your data science workflow with a little **engineering**! Or more specifically, about how to improve the reliability and quality of your data applications... all so that your models **won't break** (or at least won't break as often)! Examples for this session will be in Python 3.6+ and will rely on: **logging** to allow us to debug and diagnose things while they're running, Click to develop "beautiful" **command line interfaces** with minimal boiler-plating, and pytest to write short, elegant, and maintainable **tests**.

Google

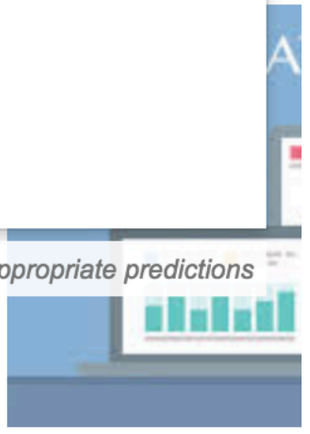
data engineer

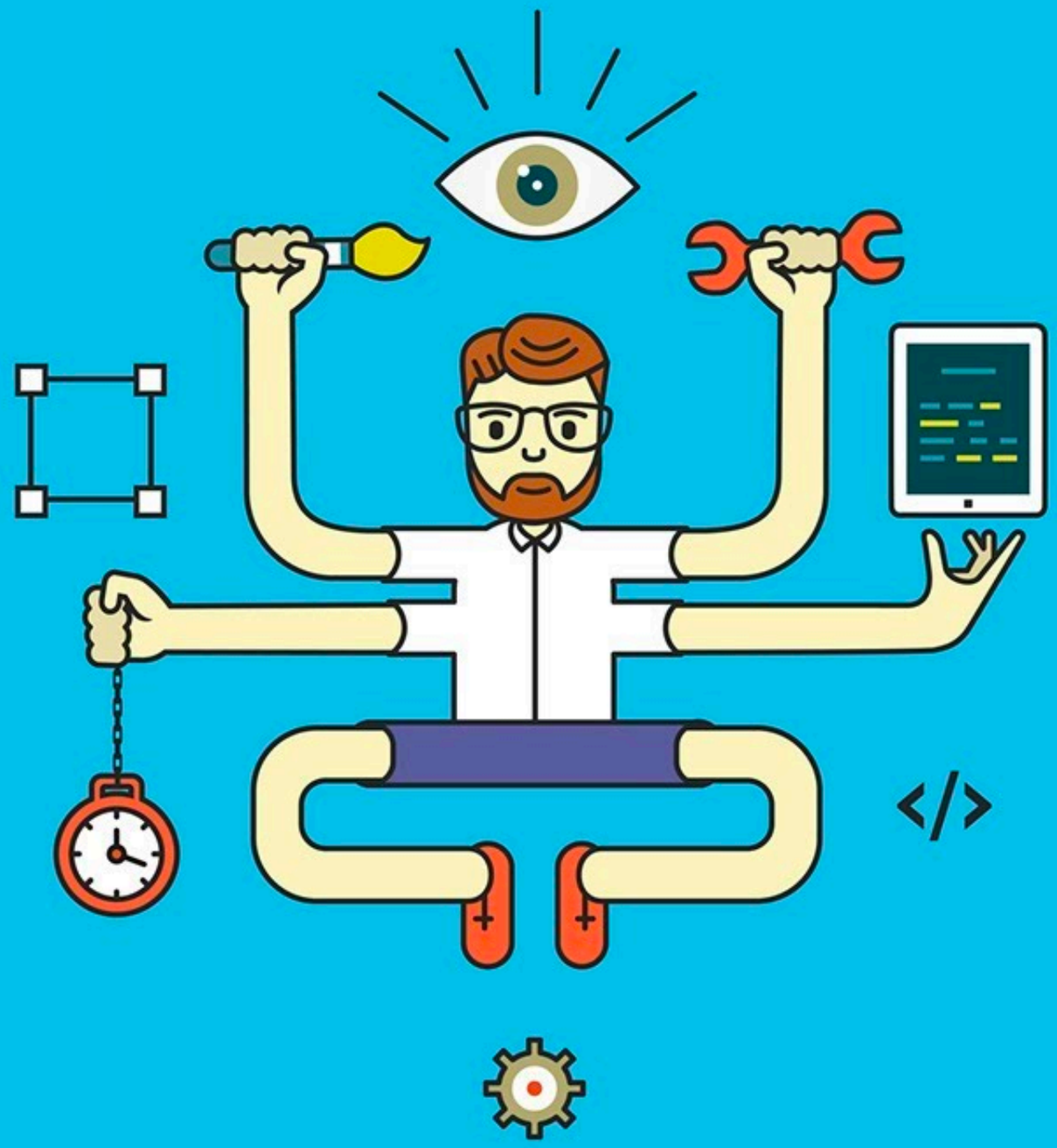
data engineer
data engineering icon
data engineering logo
data engineer job description
data engineering and analytics
data engineer tools
data engineering process
data engineer career path
data engineer meme
data engineer vs data analyst

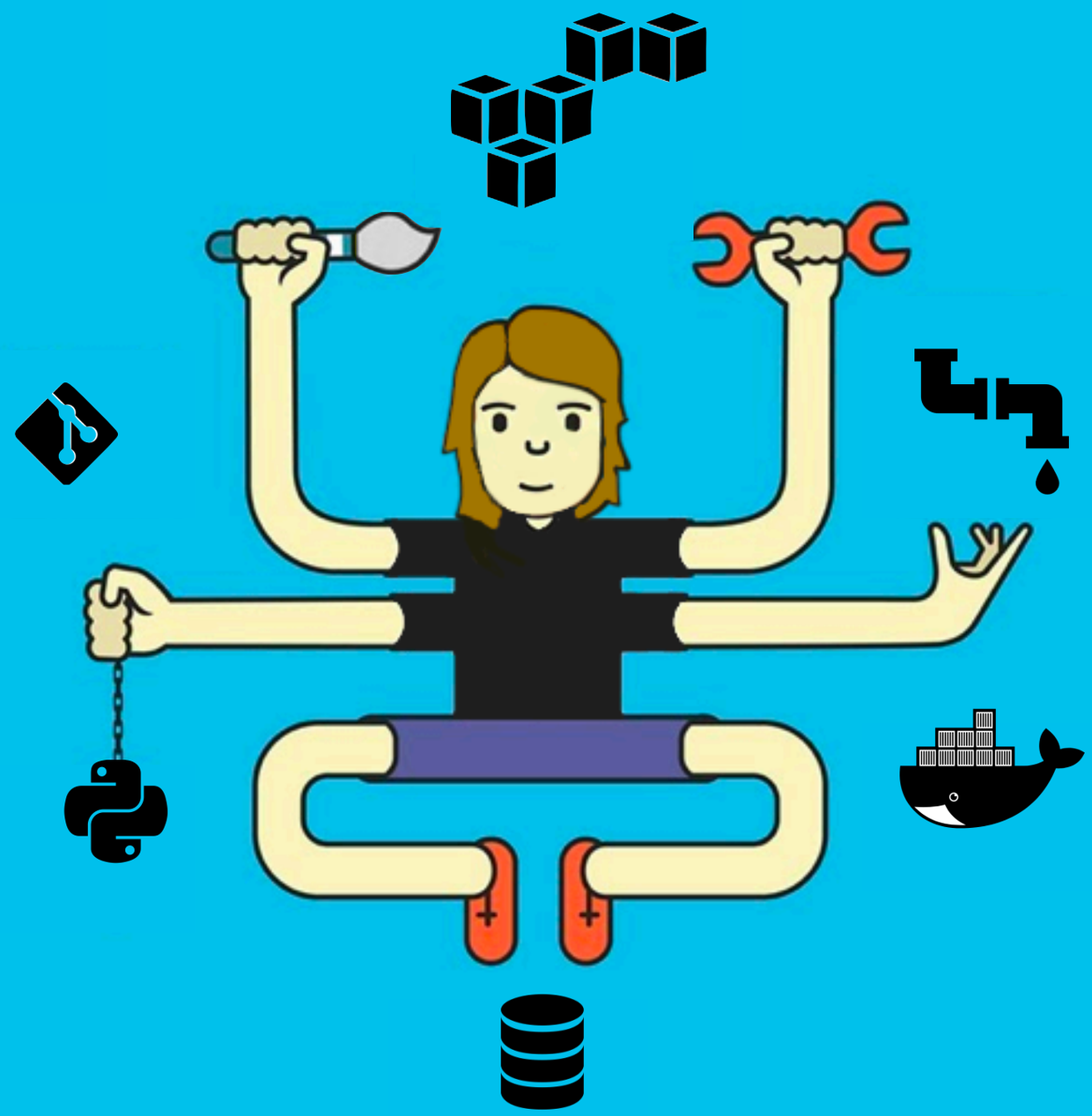
data science

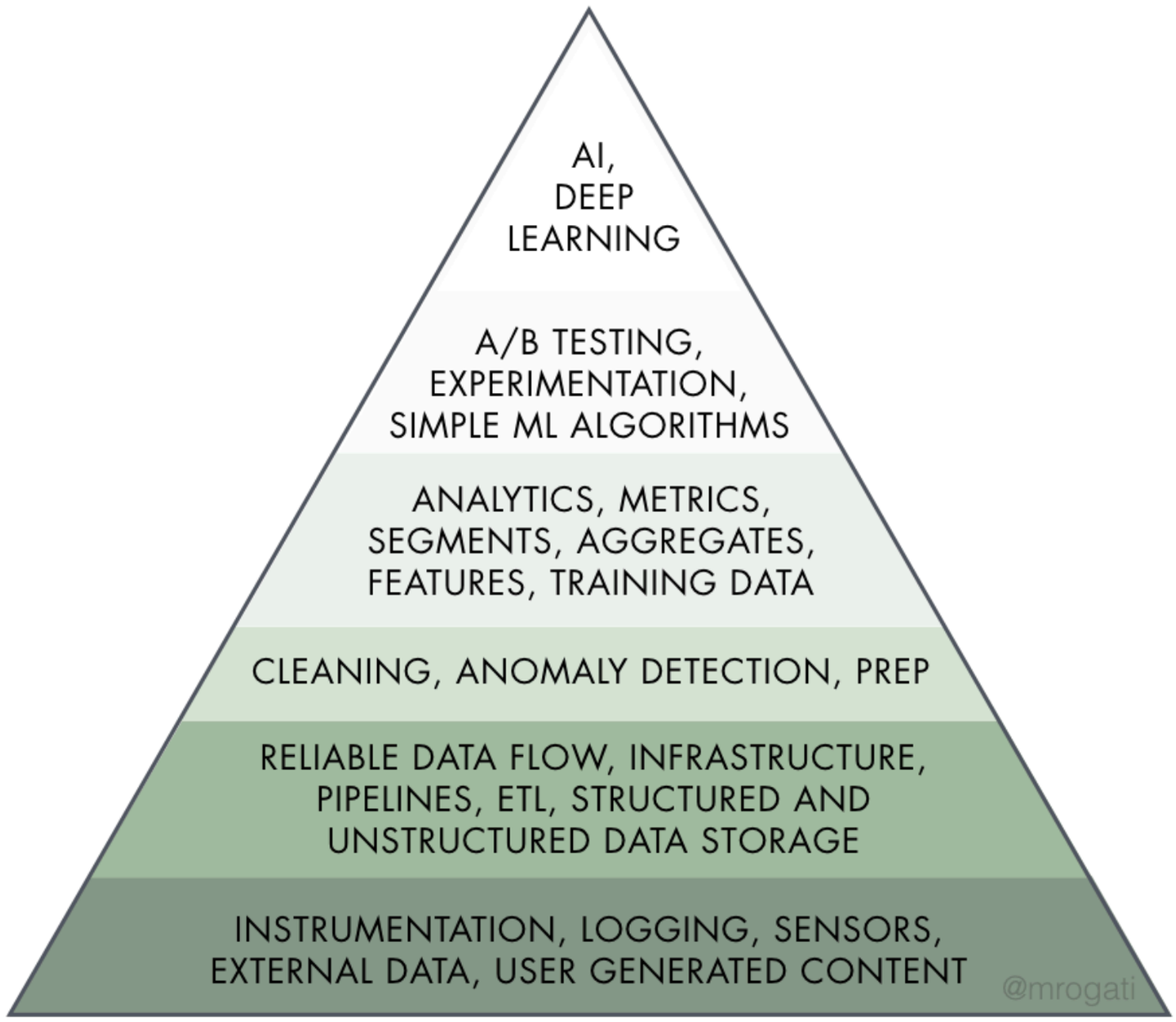


Report inappropriate predictions

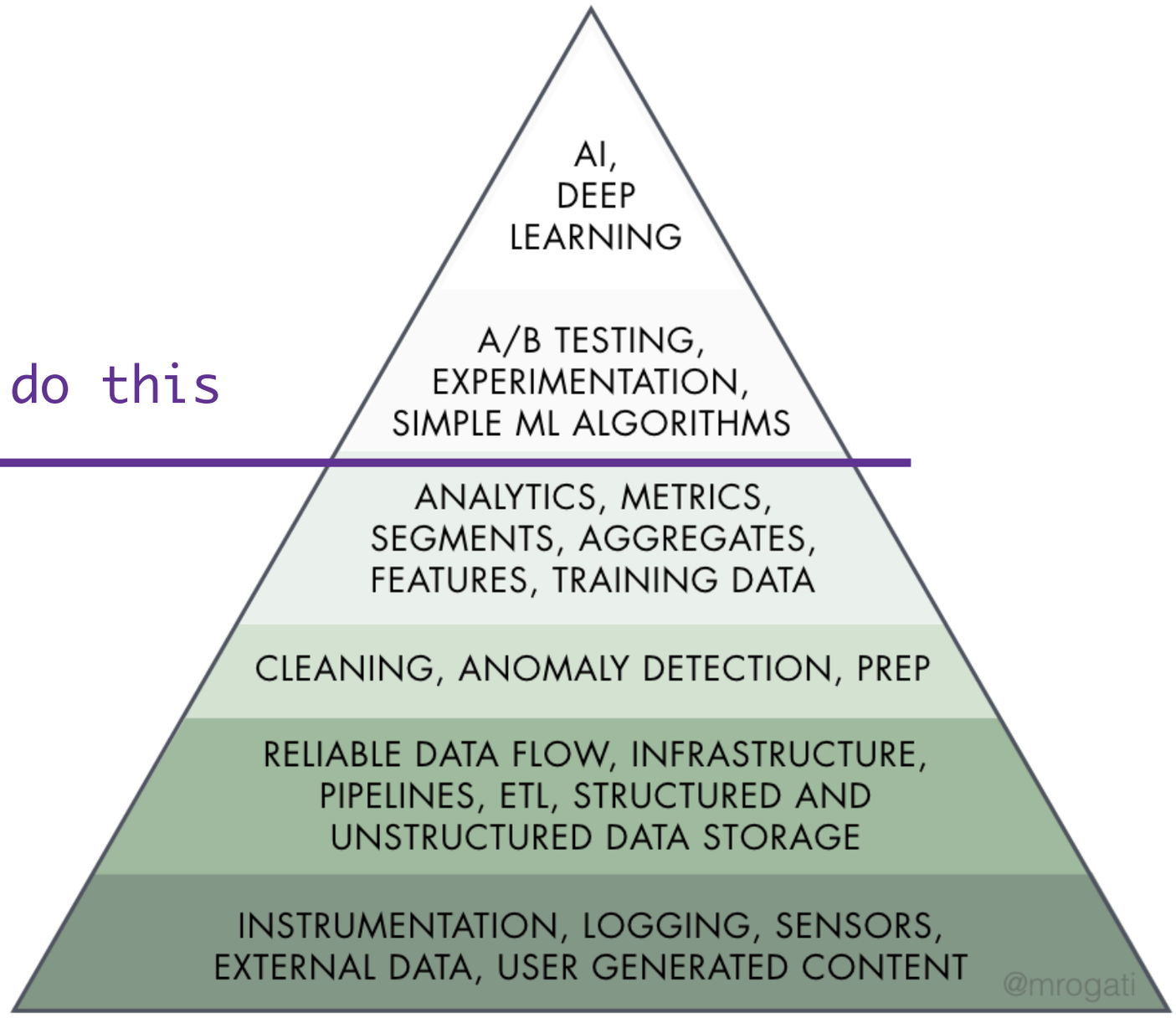






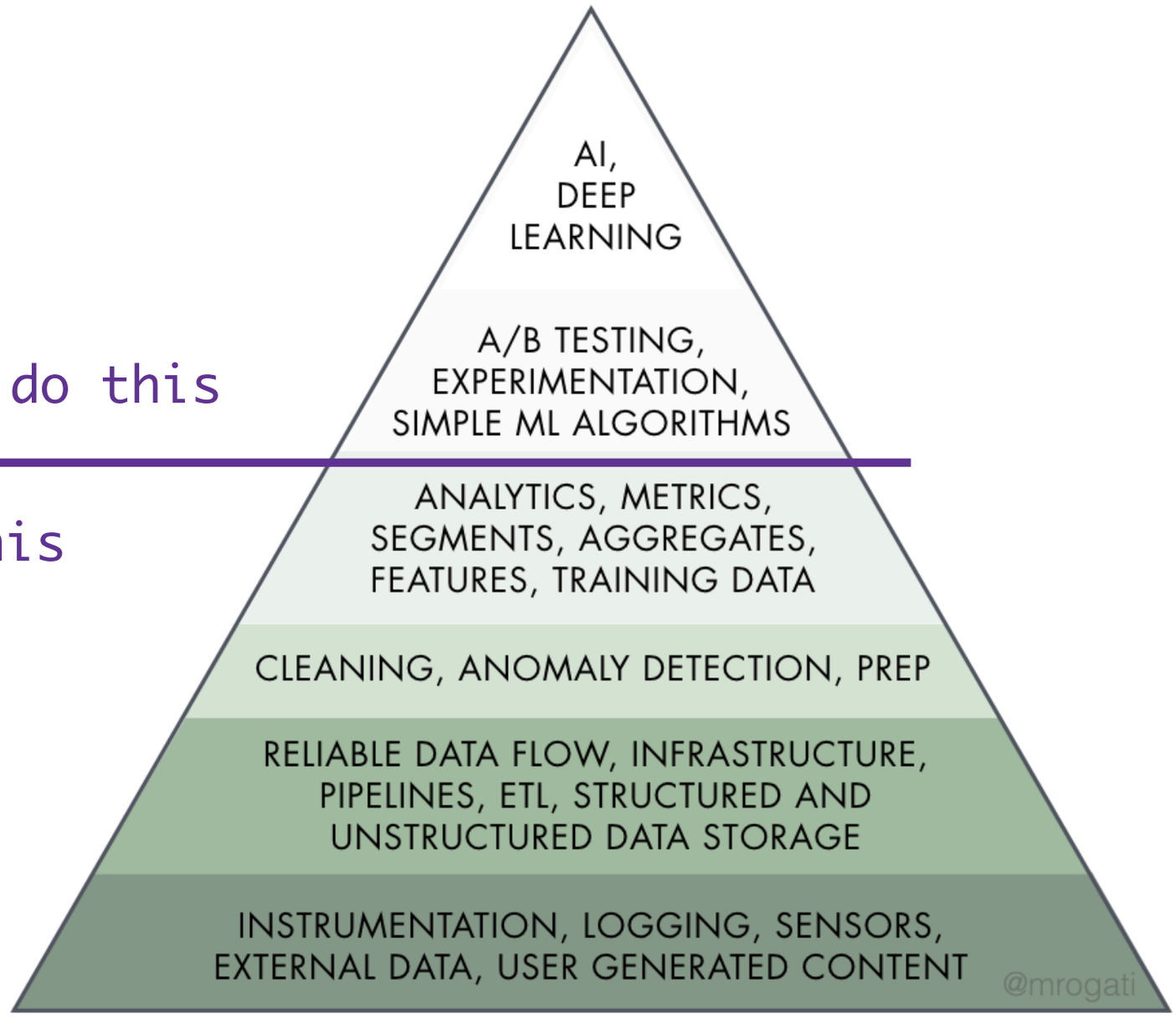

AI,
DEEP
LEARNING
A/B TESTING,
EXPERIMENTATION,
SIMPLE ML ALGORITHMS
ANALYTICS, METRICS,
SEGMENTS, AGGREGATES,
FEATURES, TRAINING DATA
CLEANING, ANOMALY DETECTION, PREP
RELIABLE DATA FLOW, INFRASTRUCTURE,
PIPELINES, ETL, STRUCTURED AND
UNSTRUCTURED DATA STORAGE
INSTRUMENTATION, LOGGING, SENSORS,
EXTERNAL DATA, USER GENERATED CONTENT
@mrogati

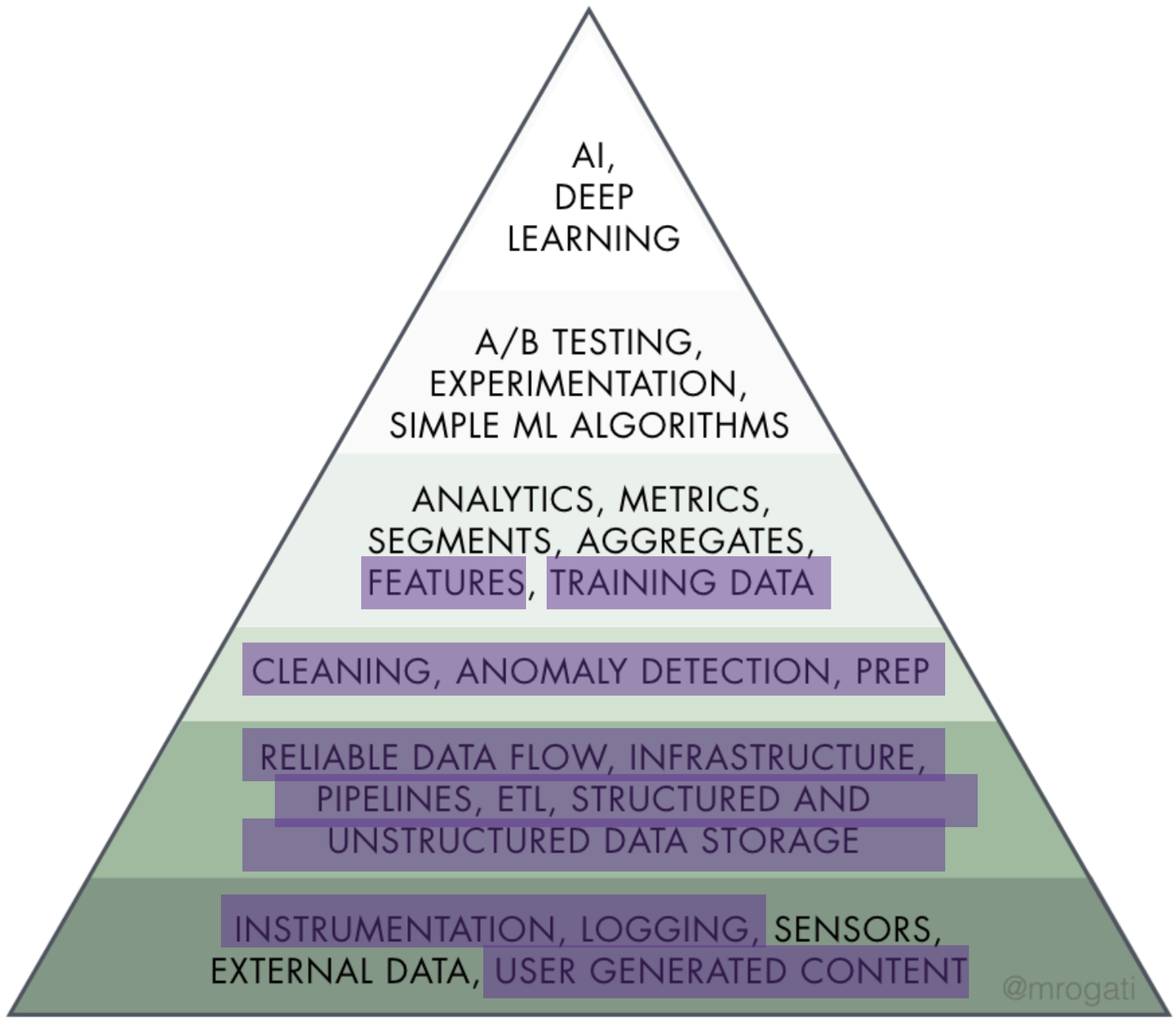
you can't do this



you can't do this

without this

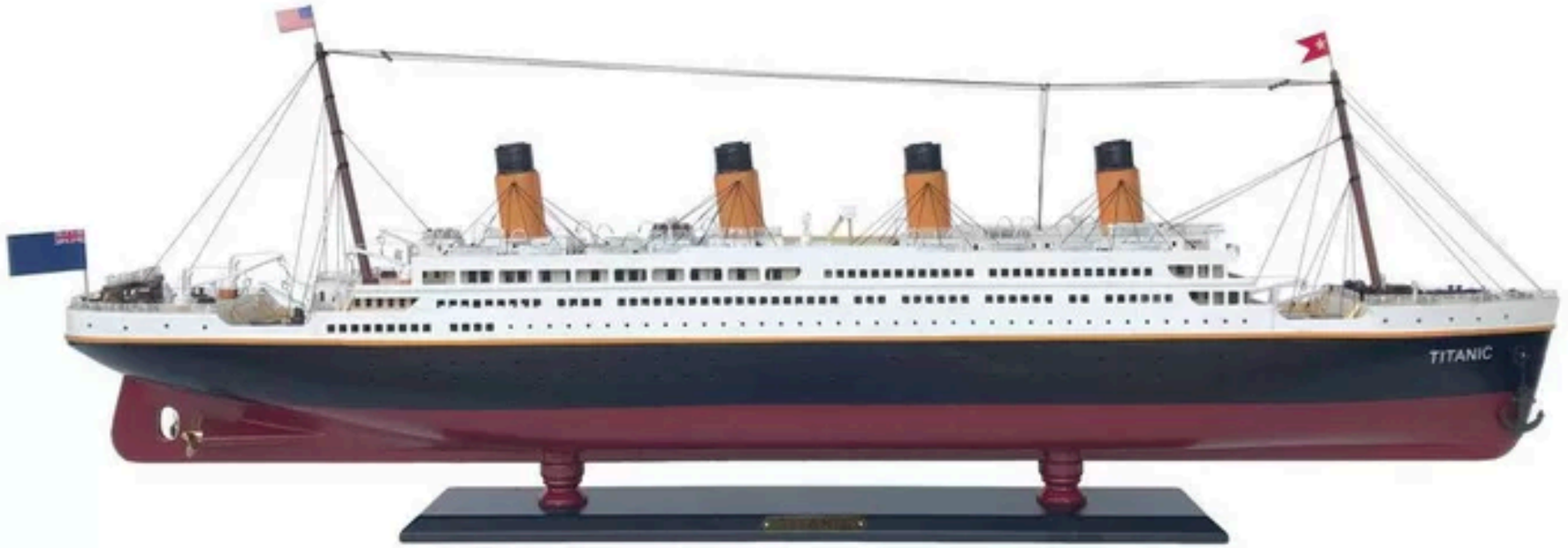



AI,
DEEP
LEARNING
A/B TESTING,
EXPERIMENTATION,
SIMPLE ML ALGORITHMS
ANALYTICS, METRICS,
SEGMENTS, AGGREGATES,
FEATURES, TRAINING DATA
CLEANING, ANOMALY DETECTION, PREP
RELIABLE DATA FLOW, INFRASTRUCTURE,
PIPELINES, ETL, STRUCTURED AND
UNSTRUCTURED DATA STORAGE
INSTRUMENTATION, LOGGING, SENSORS,
EXTERNAL DATA, USER GENERATED CONTENT
@mrogati

#1 .py
#2 defence
#3 log
#4 cli

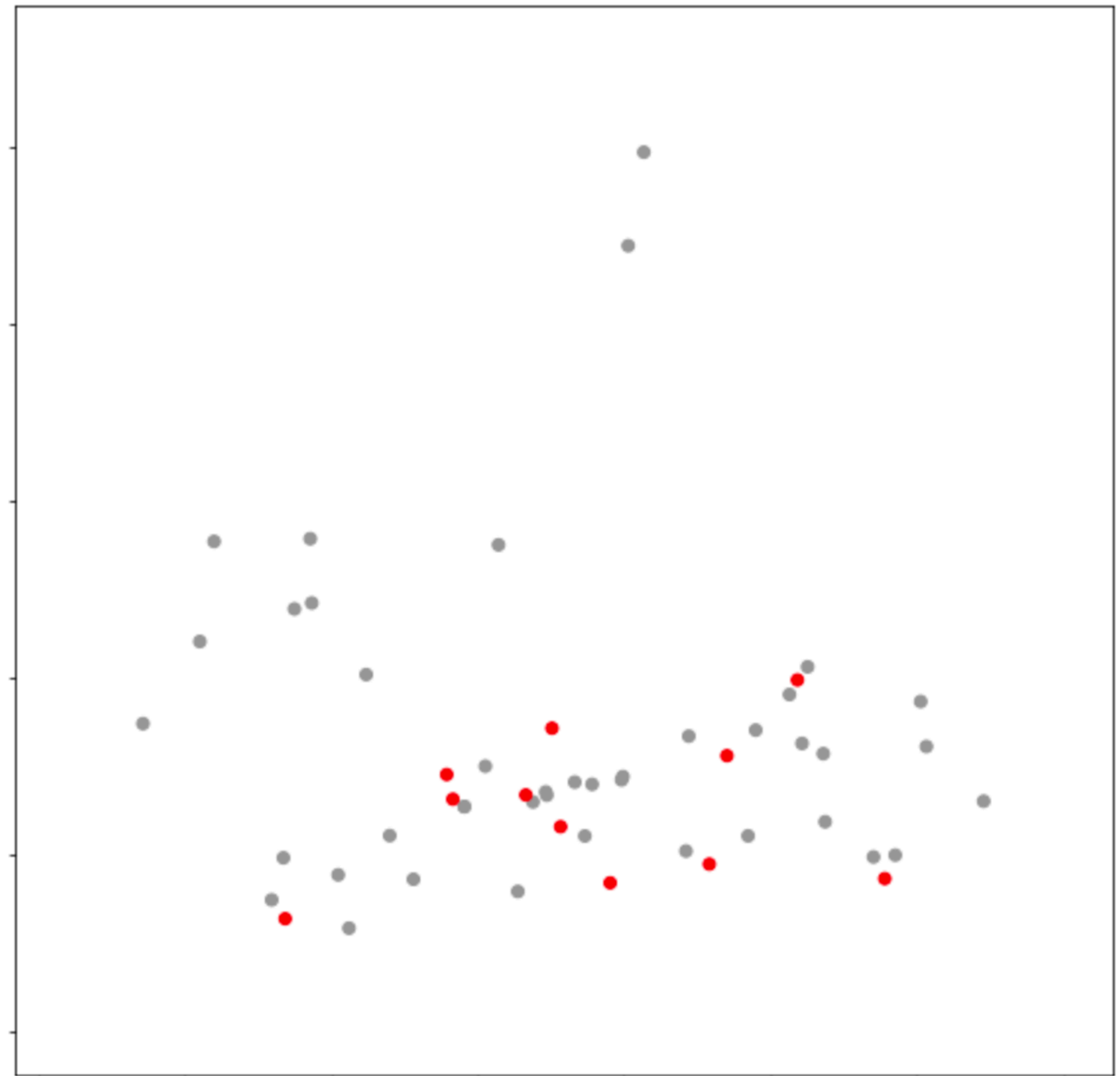
#1 .py
#2 defence
#3 log
#4 cli
#5 😱

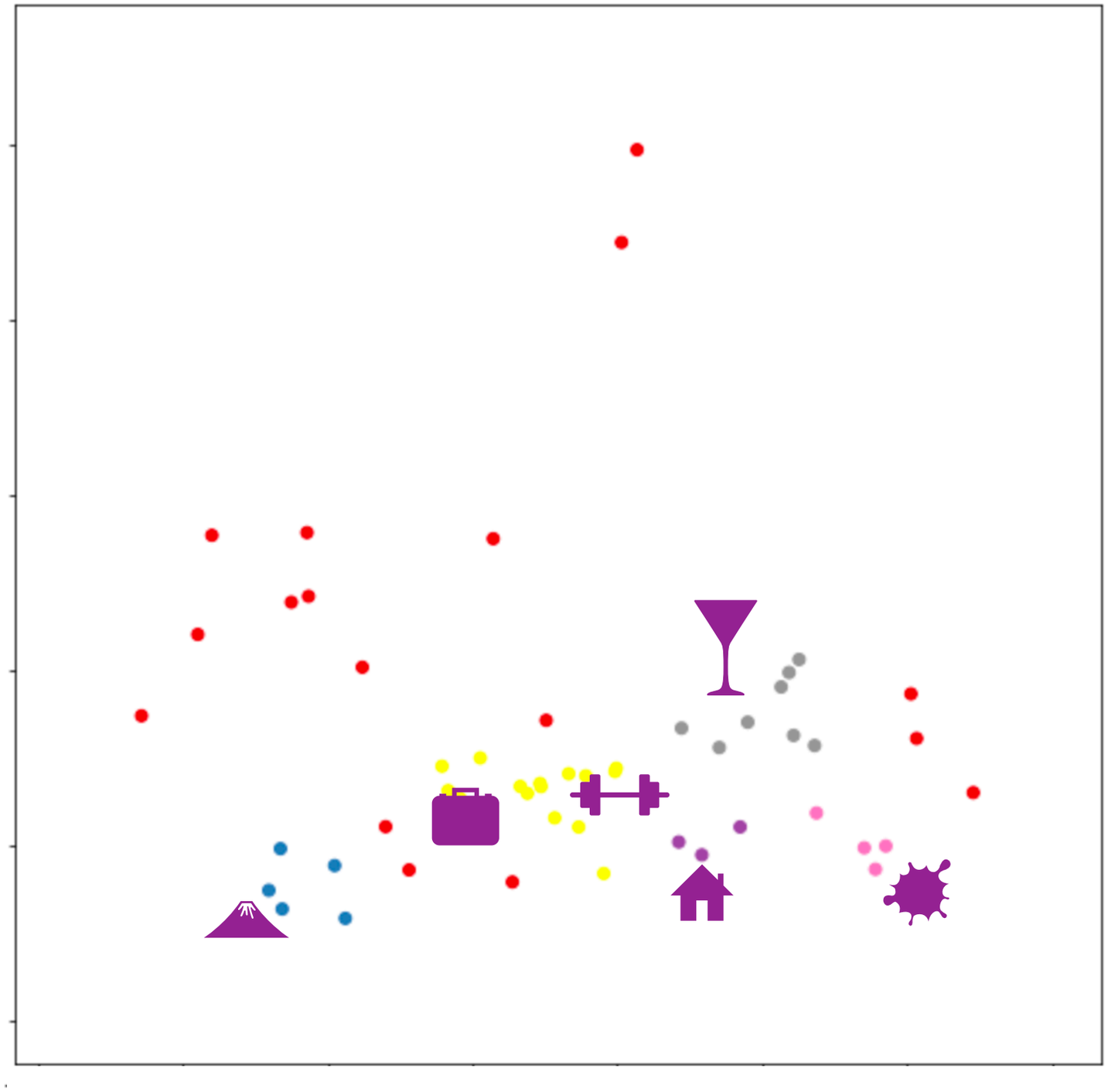

ELMER'S
Glue-All
Multi-Purpose Glue
Bonds Strong
For All Your Needs
Safe, Non-Toxic
1.25 FL OZ (36.9mL)


TITANIC


TITANIC

	drop_off	time	pick_up	last_drop_off	last_pick_up
277	work	2018-03-27 08:42:59	other	other	NaN
278	climbing	2018-03-27 17:18:49	west_core	work	other
279	home	2018-03-27 18:23:39	climbing	climbing	west_core
280	work	2018-03-28 09:08:11	home	home	climbing
281	home	2018-03-28 17:39:16	work	work	home
283	work	2018-03-29 09:17:35	home	NaN	home
284	home	2018-03-29 17:10:19	work	work	home
285	climbing	2018-03-30 11:53:09	home	home	work
286	other	2018-03-30 13:13:20	climbing	climbing	home
287	home	2018-03-30 13:18:11	other	other	climbing





Dupont St
Ossington Ave
Bathurst St
Spadina Ave
Bloor St
Yonge St
Church St
Carlton St
Yonge-Dundas Sq
Gerrard St
Jarvis St
Shuter St
College St
Dundas St W
Toronto
Queen St E
Adelaide St
Trinity Bellwoods Park
Queen St W
OLD TORONTO
King St W
ENTERTAINMENT DISTRICT
Fort York National Historic Site
Gardiner Expy
Queens Quay W
Budweiser Stage
Lake Shore Blvd W


GO

#1

Lose the Notebook

	drop_off	time	pick_up	last_drop_off	last_pick_up
277	work	2018-03-27 08:42:59	other	other	NaN
278	climbing	2018-03-27 17:18:49	west_core	work	other
279	home	2018-03-27 18:23:39	climbing	climbing	west_core
280	work	2018-03-28 09:08:11	home	home	climbing
281	home	2018-03-28 17:39:16	work	work	home
283	work	2018-03-29 09:17:35	home	NaN	home
284	home	2018-03-29 17:10:19	work	work	home
285	climbing	2018-03-30 11:53:09	home	home	work
286	other	2018-03-30 13:13:20	climbing	climbing	home
287	home	2018-03-30 13:18:11	other	other	climbing

[1]
```
import pandas as pd
from sklearn.model_selection import train_test_split
```

[2]
```
df = pd.read_csv('max_bike_data.csv')
```

[1]
```
import pandas as pd
from sklearn.model_selection import train_test_split
```

[2]
```
df = pd.read_csv('max_bike_data.csv')
```

[3]
```
for col in df:
    percent_missing = df[df[col].isnull() == True].shape[0] / df.shape[0]
    print(f'percent missing for column {col}: {percent_missing:.3f}')
```

```
percent missing for column drop_off: 0.007
percent missing for column time: 0.000
percent missing for column pick_up: 0.007
percent missing for column last_drop_off: 0.010
percent missing for column last_pick_up: 0.010
```

[4]
```
df = df[df['drop_off'].notnull()]
```

- ✅ exploratory analysis
- ✅ visualizing ideas
- ✅ prototyping

`.ipynb`

`.ipynb`

✅ exploratory analysis

✅ visualizing ideas

✅ prototyping

❌ messy

❌ bad at versioning

❌ not ideal for production

.ipynb

.py

```
$ jupyter nbconvert --to script [NOTEBOOK_NAME].ipynb
```

```
$ jupyter nbconvert --to script [NOTEBOOK_NAME].ipynb
```

[1]
```
import pandas as pd
from sklearn.model_selection import train_test_split
```

[2]
```
df = pd.read_csv('max_bike_data.csv')
```

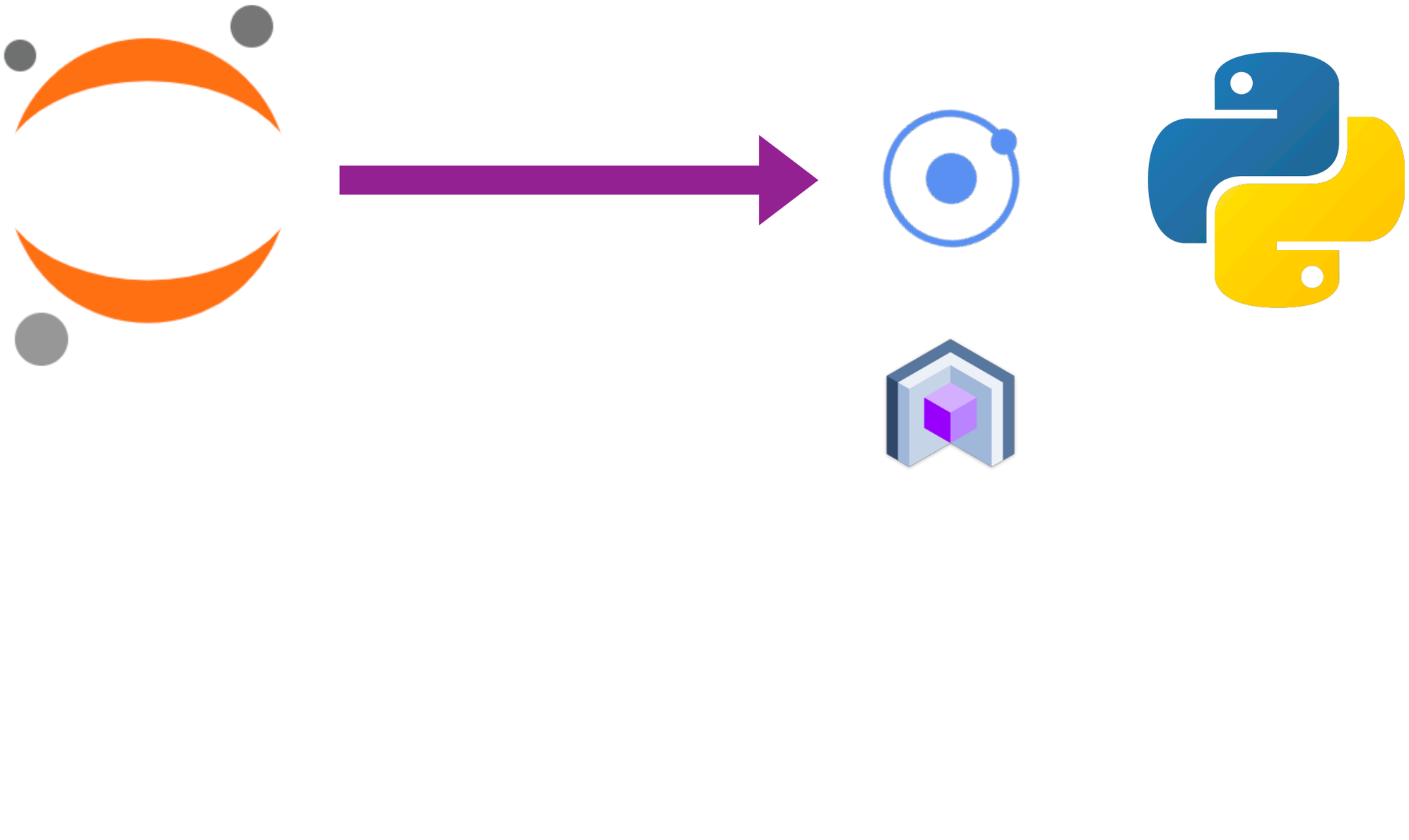
[3]
```
for col in df:
    percent_missing = df[df[col].isnull() == True].shape[0] / df.shape[0]
    print(f'percent missing for column {col}: {percent_missing:.3f}')
```

```
percent missing for column drop_off: 0.007
percent missing for column time: 0.000
percent missing for column pick_up: 0.007
percent missing for column last_drop_off: 0.010
percent missing for column last_pick_up: 0.010
```

[4]
```
df = df[df['drop_off'].notnull()]
```

```
$ jupyter nbconvert --to script [NOTEBOOK_NAME].ipynb
```

```
# coding: utf-8

# In[1]:

import pandas as pd
from sklearn.model_selection import train_test_split

# In[2]:

df = pd.read_csv('max_bike_data.csv')

# In[3]:

for col in df:
    percent_missing = df[df[col].isnull() == True].shape[0] / df.shape[0]
    print(f'percent missing for column {col}: {percent_missing:.3f}')

# In[4]:

df = df[df['drop_off'].notnull()]

```



hydrogen

Run code interactively, inspect data, and plot. All the power of Jupyter kernels, inside your favorite text editor.

#execute #run #jupyter #ipython #julia

nteract 430,058 461

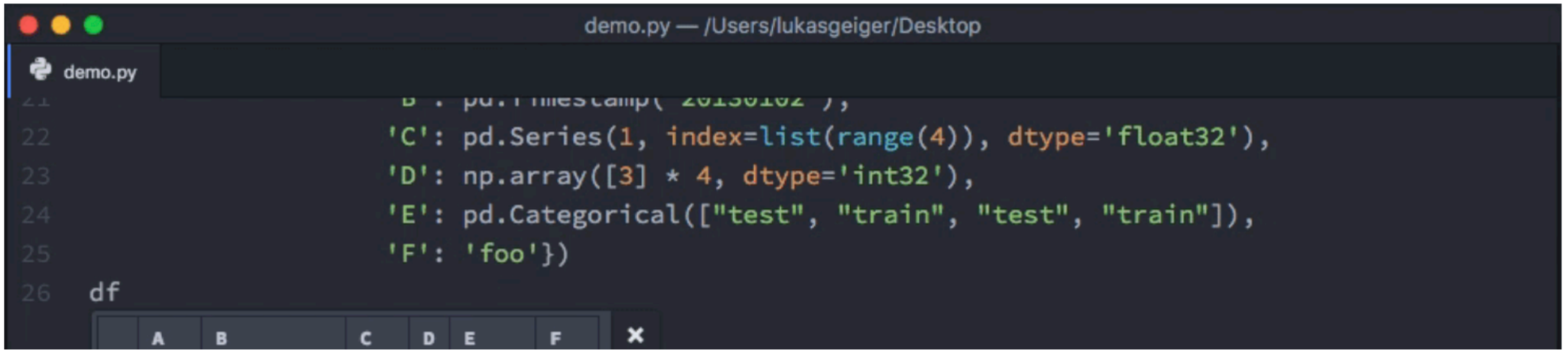
Repo Bugs Versions License

Flag as spam or malicious

Hydrogen

Hydrogen is an interactive coding environment that supports Python, R, JavaScript and other Jupyter kernels.

Checkout our Documentation and Medium blog post to see what you can do with Hydrogen.



cmd+enter

notebook_to_hydrogen.py

```
# coding: utf-8

# In[1]:

import pandas as pd
from sklearn.model_selection import train_test_split

# In[2]:

df = pd.read_csv('max_bike_data.csv')

# In[3]:

for col in df:
    percent_missing = df[df[col].isnull() == True].shape[0] / df.shape[0]
    print(f'percent missing for column {col}: {percent_missing:.3f}')
```

Hydrogen Kernels updated:

Python 3

```
9
10  df = pd.read_csv('max_bike_data.csv')
```

	drop_off	time	pick_up	last_drop_off	last_pick_up
0	other	2017-10-22 18:23:19	home	NaN	NaN
1	home	2017-10-22 19:51:52	climbing	other	home
2	work	2017-10-23 09:06:08	home	home	climbing
3	home	2017-10-23 17:46:26	work	work	home
4	work	2017-10-24 09:05:50	home	home	work
5	home	2017-10-24 17:57:51	work	work	home
6	work	2017-10-25 09:09:27	home	home	work
7	other	2017-10-25 17:28:48	work	work	home
8	climbing	2017-10-25 18:45:38	west_core	other	work
9	home	2017-10-25 19:38:44	climbing	climbing	west_core
10	west_core	2017-10-25 20:08:52	home	home	climbing

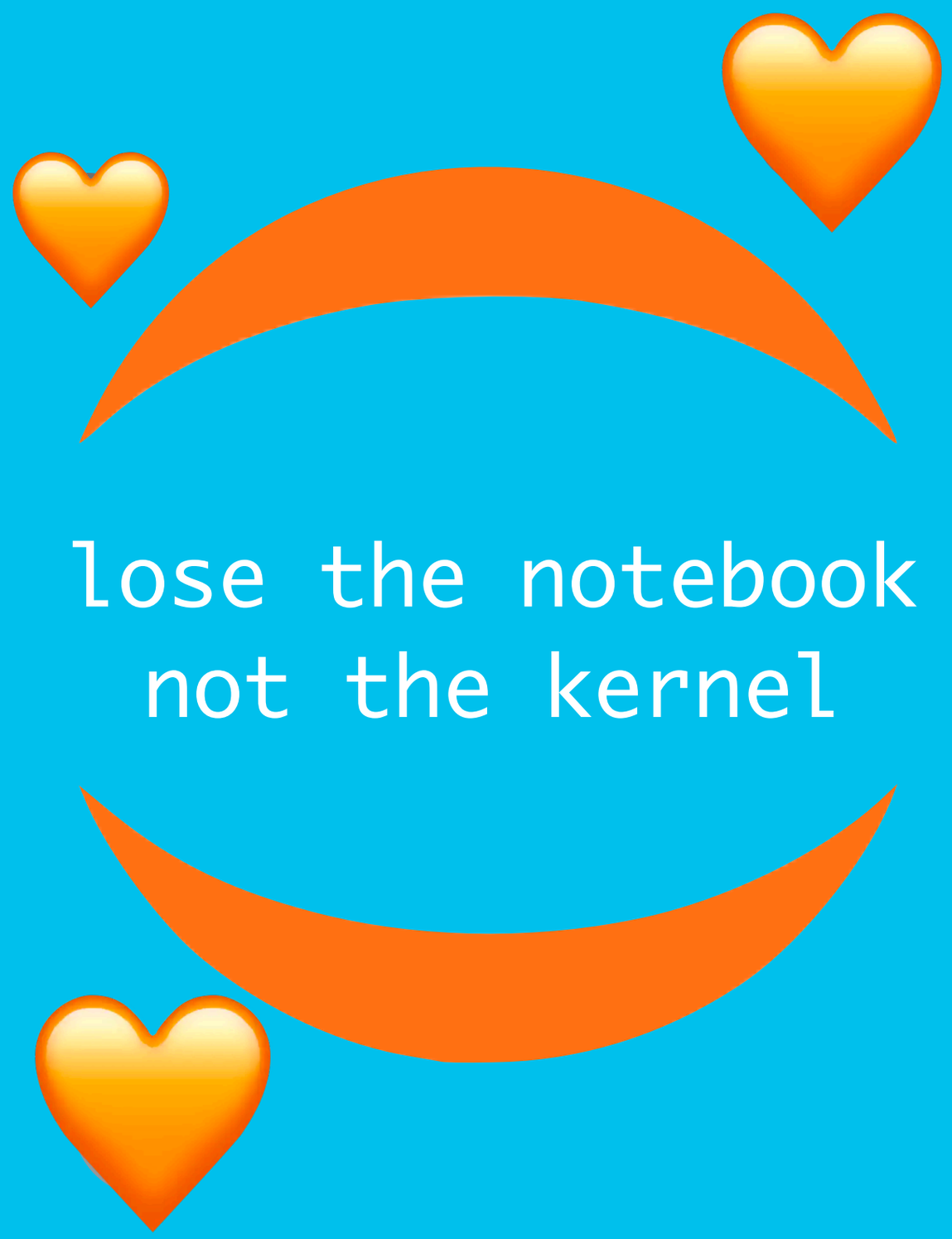
```python
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv('max_bike_data.csv')

for col in df:
    percent_missing = df[df[col].isnull() == True].shape[0] / df.shape[0]
    print(f'percent missing for column {col}: {percent_missing:.3f}')

df = df[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)
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv('max_bike_data.csv')

for col in df:
    percent_missing = df[df[col].isnull() == True].shape[0] / df.shape[0]
    print(f'percent missing for column {col}: {percent_missing:.3f}')

df = df[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)
```

	time	pick_up	last_drop_off	last_pick_up
243	2018-03-12 19:51:03	climbing	climbing	climbing
254	2018-03-17 12:51:35	home	climbing	west_core
162	2018-01-30 09:02:36	home	climbing	west_core
100	2017-12-14 09:05:38	home	home	downtown_core

lose the notebook

lose the notebook
not the kernel



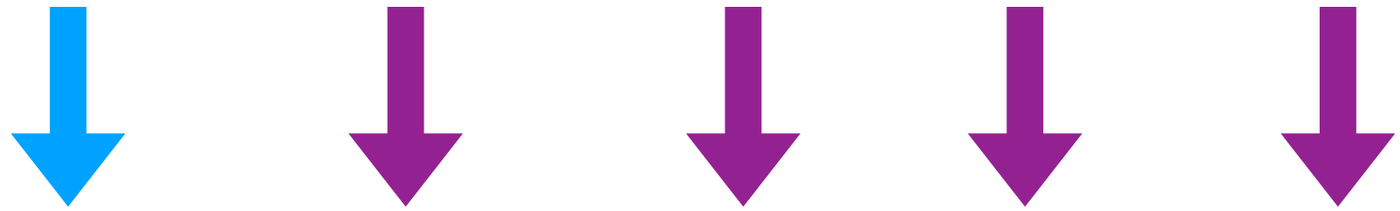
lose the notebook
not the kernel

#2

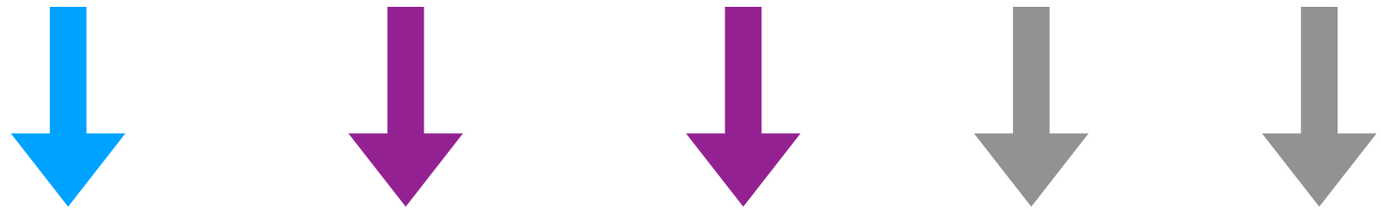
Get Defensive

time	pick_up	last_drop_off	last_pick_up
2018-03-27 08:42:59	other	other	NaN
2018-03-27 17:18:49	west_core	work	other
2018-03-27 18:23:39	climbing	climbing	west_core
2018-03-28 09:08:11	home	home	climbing
2018-03-28 17:39:16	work	work	home
2018-03-29 09:17:35	home	NaN	home
2018-03-29 17:10:19	work	work	home
2018-03-30 11:53:09	home	home	work
2018-03-30 13:13:20	climbing	climbing	home
2018-03-30 13:18:11	other	other	climbing

	drop_off	time	pick_up	last_drop_off	last_pick_up
277	work	2018-03-27 08:42:59	other	other	NaN
278	climbing	2018-03-27 17:18:49	west_core	work	other
279	home	2018-03-27 18:23:39	climbing	climbing	west_core
280	work	2018-03-28 09:08:11	home	home	climbing
281	home	2018-03-28 17:39:16	work	work	home
283	work	2018-03-29 09:17:35	home	NaN	home
284	home	2018-03-29 17:10:19	work	work	home
285	climbing	2018-03-30 11:53:09	home	home	work
286	other	2018-03-30 13:13:20	climbing	climbing	home
287	home	2018-03-30 13:18:11	other	other	climbing



	drop_off	time	pick_up	last_drop_off	last_pick_up
277	work	2018-03-27 08:42:59	other	other	NaN
278	climbing	2018-03-27 17:18:49	west_core	work	other
279	home	2018-03-27 18:23:39	climbing	climbing	west_core
280	work	2018-03-28 09:08:11	home	home	climbing
281	home	2018-03-28 17:39:16	work	work	home
283	work	2018-03-29 09:17:35	home	NaN	home
284	home	2018-03-29 17:10:19	work	work	home
285	climbing	2018-03-30 11:53:09	home	home	work
286	other	2018-03-30 13:13:20	climbing	climbing	home
287	home	2018-03-30 13:18:11	other	other	climbing



	drop_off	time	pick_up	last_drop_off	last_pick_up
277	work	2018-03-27 08:42:59	other	other	NaN
278	climbing	2018-03-27 17:18:49	west_core	work	other
279	home	2018-03-27 18:23:39	climbing	climbing	west_core
280	work	2018-03-28 09:08:11	home	home	climbing
281	home	2018-03-28 17:39:16	work	work	home
283	work	2018-03-29 09:17:35	home	NaN	home
284	home	2018-03-29 17:10:19	work	work	home
285	climbing	2018-03-30 11:53:09	home	home	work
286	other	2018-03-30 13:13:20	climbing	climbing	home
287	home	2018-03-30 13:18:11	other	other	climbing

```
from sklearn.preprocessing import LabelBinarizer
lb = LabelBinarizer()
lb.fit_transform(X_train.pick_up)
```

```
from sklearn.preprocessing import LabelBinarizer
lb = LabelBinarizer()
lb.fit_transform(X_train.pick_up)

lb.transform(['home'])
```

```
array([[0, 0, 0, 1, 0, 0, 0]])
```

```
from sklearn.preprocessing import LabelBinarizer
lb = LabelBinarizer()
lb.fit_transform(X_train.pick_up)

lb.transform(['home'])
array([[0, 0, 0, 1, 0, 0, 0]])

lb.transform([np.NaN])
```

```
from sklearn.preprocessing import LabelBinarizer
lb = LabelBinarizer()
lb.fit_transform(X_train.pick_up)

lb.transform(['home'])
```

```
array([[0, 0, 0, 1, 0, 0, 0]])
```

```
lb.transform([np.NaN])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call l
<ipython-input-20-d50d390f7983> in <module>()
----> 1 lb.transform([np.NaN])

~/anaconda3/lib/python3.6/site-packages/sklea      /      el.py
    336                                        pos_lab
    337                                        neg_l
--> 338                                        spa          utput)
    339
    340     def inverse_transform(self, Y

~/anaconda3/lib/python3.6/site-packages/sklearn/prep      ng      bel.py
    517     else:
```

```
$ pip install sklearn-pandas
```

DataFrameMapper

CategoricalImputer

```
from sklearn_pandas import DataFrameMapper, CategoricalImputer

mapper = DataFrameMapper([
    ('time', None),
    ('pick_up', None),
    ('last_drop_off', CategoricalImputer()),
    ('last_pick_up', CategoricalImputer())
])

mapper.fit(X_train)
```

```python
from sklearn_pandas import DataFrameMapper, CategoricalImputer

mapper = DataFrameMapper([
    ('time', None),
    ('pick_up', None),
    ('last_drop_off', CategoricalImputer()),
    ('last_pick_up', CategoricalImputer())
])

mapper.fit(X_train)
```

```python
mapper.transform(
    pd.DataFrame({
        'pick_up': ['work'],
        'time': [pd.Timestamp(2018, 4, 1, 9, 30)],
        'last_drop_off': [None],
        'last_pick_up': [None]
    })
)
```

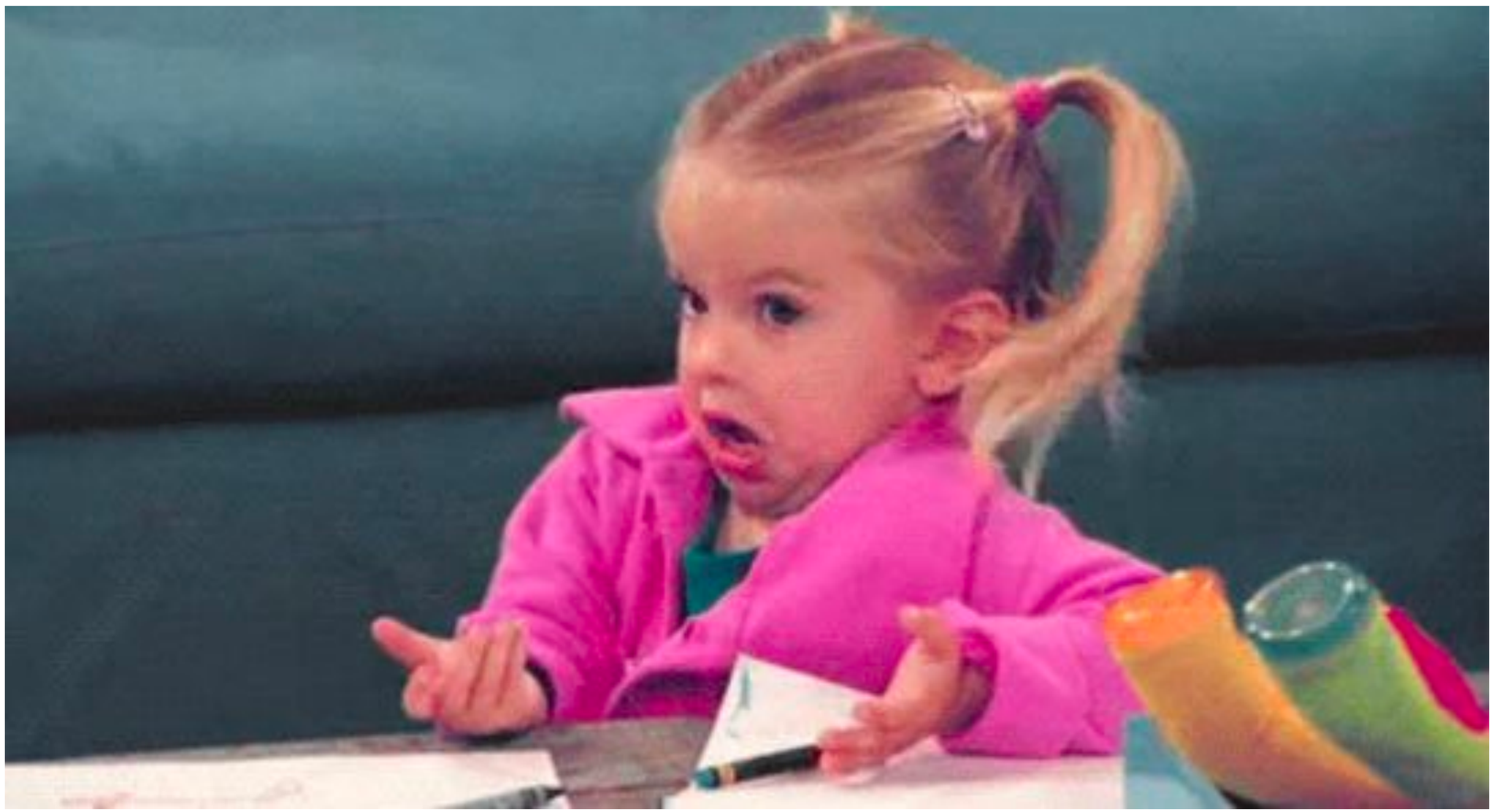
```
from sklearn_pandas import DataFrameMapper, CategoricalImputer

mapper = DataFrameMapper([
    ('time', None),
    ('pick_up', None),
    ('last_drop_off', CategoricalImputer()),
    ('last_pick_up', CategoricalImputer())
])
```

```
mapper.fit(X_train)
```

```
mapper.transform(
    pd.DataFrame({
        'pick_up': ['work'],
        'time': [pd.Timestamp(2018, 4, 1, 9, 30)],
        'last_drop_off': [None],
        'last_pick_up': [None]
    })
)
```

```
array([[1522575000000000000, 'work', 'home', 'home']], dtype=object)
```

```
array([[1522575000000000000, 'work', 'home', 'home']], dtype=object)
```



```
array([[1522575000000000000, 'work', 'home', 'home']], dtype=object)
```

```
mapper = DataFrameMapper([
    # ('time', None),
    ('pick_up',
    ('last_drop_off', [CategoricalImputer(),
    ('last_pick_up', [CategoricalImputer(),
], df_out=True)

mapper.fit_transform(X_train)
```

```
mapper = DataFrameMapper([
    # ('time', None),
    ('pick_up', LabelBinarizer()),
    ('last_drop_off', [CategoricalImputer(), LabelBinarizer()]),
    ('last_pick_up', [CategoricalImputer(), LabelBinarizer()])
], df_out=True)

mapper.fit_transform(X_train)
```

```
mapper = DataFrameMapper([
    # ('time', None),
    ('pick_up', LabelBinarizer()),
    ('last_drop_off', [CategoricalImputer(), LabelBinarizer()]),
    ('last_pick_up', [CategoricalImputer(), LabelBinarizer()])
], df_out=True)

mapper.fit_transform(X_train)
```

pick_up_home	pick_up_other	pick_up_west_core	pick_up_work	last_drop_off_art
0	0	0	1	0
1	0	0	0	0
0	0	1	0	0
1	0	0	0	0
0	1	0	0	0
1	0	0	0	0

```
array([[1522575000000000000, 'work', 'home', 'home']], dtype=object)
```

```
from sklearn.base import TransformerMixin

class DateEncoder(TransformerMixin):
```

```
from sklearn.base import TransformerMixin

class DateEncoder(TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
        dt = X.dt
        return pd.concat([dt.month, dt.dayofweek, dt.hour],
                         axis=1)
```

```python
from sklearn.base import TransformerMixin

class DateEncoder(TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
        dt = X.dt
        return pd.concat([dt.month, dt.dayofweek, dt.hour],
                         axis=1)
```

```
X = pd.DataFrame({'time': [pd.Timestamp(2018, 4, 1, 9, 30)]})
```

```
X = pd.DataFrame({'time': [pd.Timestamp(2018, 4, 1, 9, 30)]})

de = DateEncoder()
de.fit(X.time)
de.transform(X.time)
```

```
X = pd.DataFrame({'time': [pd.Timestamp(2018, 4, 1, 9, 30)]})

de = DateEncoder()
de.fit(X.time)
de.transform(X.time)
```

	time	time	time
0	4	6	9

month, dayofweek, hour

```
mapper = DataFrameMapper([
    ('time'                                          ),
    ('pick_up', LabelBinarizer()),
    ('last_drop_off', [CategoricalImputer(), LabelBinarizer()]),
    ('last_pick_up', [CategoricalImputer(), LabelBinarizer()])
])
```

```
mapper = DataFrameMapper([
    ('time', DateEncoder(), {'input_df': True}),
    ('pick_up', LabelBinarizer()),
    ('last_drop_off', [CategoricalImputer(), LabelBinarizer()]),
    ('last_pick_up', [CategoricalImputer(), LabelBinarizer()])
])
```

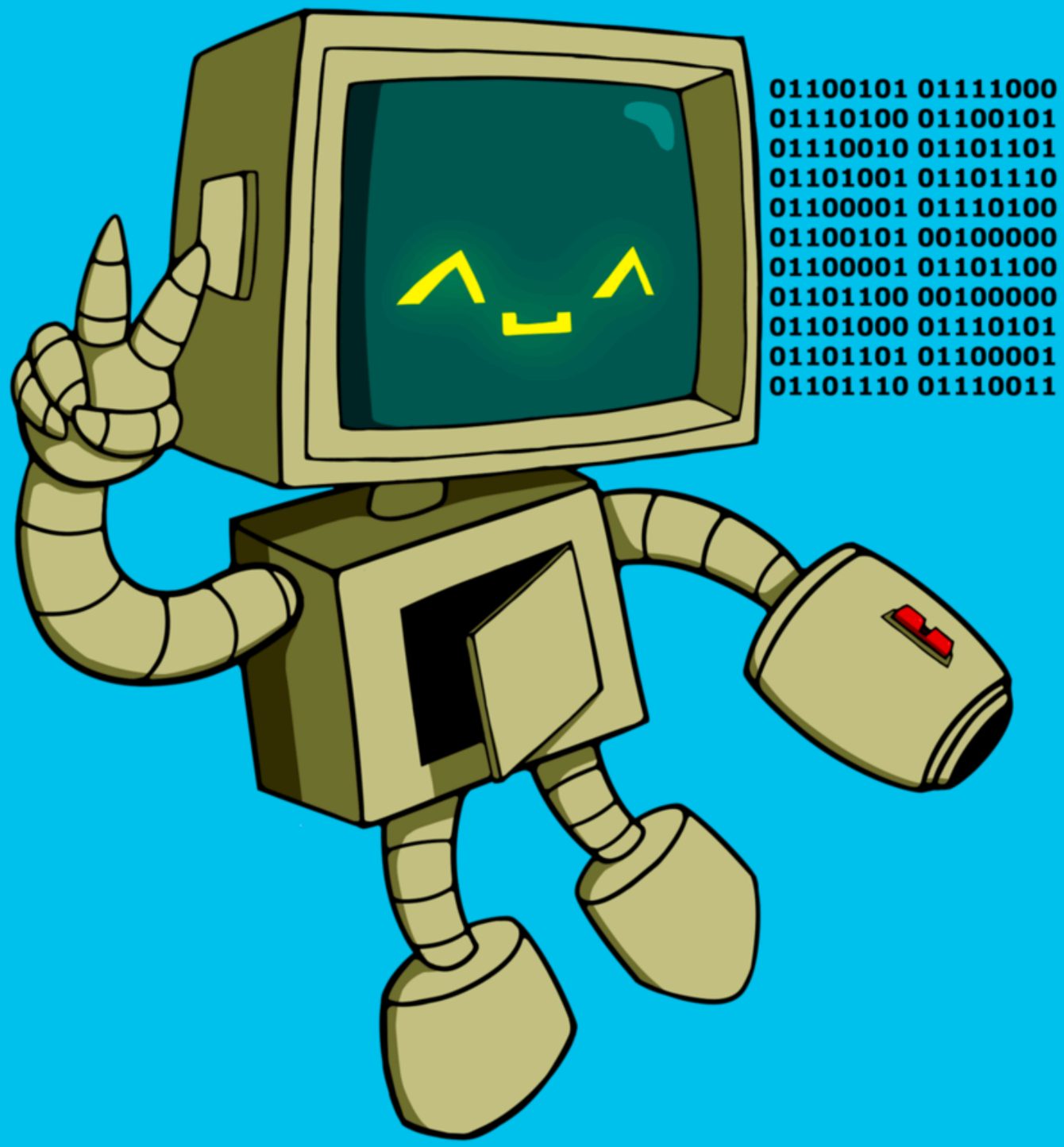
```
mapper = DataFrameMapper([
    ('time', DateEncoder(), {'input_df': True}),
    ('pick_up', LabelBinarizer()),
    ('last_drop_off', [CategoricalImputer(), LabelBinarizer()]),
    ('last_pick_up', [CategoricalImputer(), LabelBinarizer()])
])

mapper.fit_transform(X_train)
```

```
array([[ 2,  6, 11, ...,  0,  0,  0],
       [ 3,  3,  9, ...,  0,  0,  0],
       [12,  4, 16, ...,  0,  0,  1],
       ...,
       [ 3,  4, 17, ...,  0,  0,  0],
       [11,  2, 17, ...,  0,  0,  0],
       [ 2,  2, 18, ...,  0,  1,  0]])
```

```
mapper.transform(
    pd.DataFrame({
        'pick_up': ['work'],
        'time': [pd.Timestamp(2018, 4, 1, 9, 30)],
        'last_drop_off': ['home'],
        'last_pick_up': ['fljkkflkjflsanfsadas']
    })
)
```

```
mapper.transform(
    pd.DataFrame({
        'pick_up': ['work'],
        'time': [pd.Timestamp(2018, 4, 1, 9, 30)],
        'last_drop_off': ['home'],
        'last_pick_up': ['fljkkflkjflsanfsadas']
    })
)
```

```
array([[4, 6, 9, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0]])
```


01100101 01111000
01110100 01100101
01110010 01101101
01101001 01101110
01100001 01110100
01100101 00100000
01100001 01101100
01101100 00100000
01101000 01110101
01101101 01100001
01101110 01110011

#3

LOG

#3

LOG ALL THE THINGS

```
mapper.transform(
    pd.DataFrame({
        'pick_up': ['work'],
        'time': [pd.Timestamp(2018, 4, 1, 9, 30)],
        'last_drop_off': ['home'],
        'last_pick_up': ['fljkkflkjflsanfsadas']
    })
)
```

```
mapper.transform(
    pd.DataFrame({
        'pick_up': ['work'],
        'time': [pd.Timestamp(2018, 4, 1, 9, 30)],
        'last_drop_off': ['home'],
        'last_pick_up': ['fljkkflkjflsanfsadas']
    })
)
```

Cerberus is a lightweight and extensible data validation library for Python

Cerberus is a lightweight and extensible data validation library for Python

```
$ pip install cerberus
```

```
from cerberus import Validator
```

```
example = {
    'pick_up': 'work',
    'time': pd.Timestamp(2018, 4, 1, 9, 30),
    'last_drop_off': 'home',
    'last_pick_up': 'west_core'
}
```

```python
from cerberus import Validator

schema = {
    'time': {'type': 'datetime', 'nullable': False},
    'pick_up': {'type': 'string', 'nullable': False, 'allowed': [
        'home', 'work', 'climbing', 'west_core', 'downtown_core', 'other', 'art']
    },
    'last_drop_off': {'type': 'string', 'nullable': True},
    'last_pick_up': {'type': 'string', 'nullable': True}
}

example = {
    'pick_up': 'work',
    'time': pd.Timestamp(2018, 4, 1, 9, 30),
    'last_drop_off': 'home',
    'last_pick_up': 'west_core'
}
```

```python
from cerberus import Validator

schema = {
    'time': {'type': 'datetime', 'nullable': False},
    'pick_up': {'type': 'string', 'nullable': False, 'allowed': [
        'home', 'work', 'climbing', 'west_core', 'downtown_core', 'other', 'art']
    },
    'last_drop_off': {'type': 'string', 'nullable': True},
    'last_pick_up': {'type': 'string', 'nullable': True}
}

example = {
    'pick_up': 'work',
    'time': pd.Timestamp(2018, 4, 1, 9, 30),
    'last_drop_off': 'home',
    'last_pick_up': 'west_core'
}

v = Validator()
v.validate(example, schema)
```

```
from cerberus import Validator

schema = {
    'time': {'type': 'datetime', 'nullable': False},
    'pick_up': {'type': 'string', 'nullable': False, 'allowed': [
        'home', 'work', 'climbing', 'west_core', 'downtown_core', 'other', 'art']
    },
    'last_drop_off': {'type': 'string', 'nullable': True},
    'last_pick_up': {'type': 'string', 'nullable': True}
}

example = {
    'pick_up': 'work',
    'time': pd.Timestamp(2018, 4, 1, 9, 30),
    'last_drop_off': 'home',
    'last_pick_up': 'west_core'
}

v = Validator()
v.validate(example, schema) True
v.errors {}
```

```
bad_example = {
    'pick_up': 'sakjfnskffs',
    'time': [],
    'last_drop_off': 42,
    'last_pick_up': 'west_core'
}
```

```
bad_example = {
    'pick_up': 'sakjfnskffs',
    'time': [],
    'last_drop_off': 42,
    'last_pick_up': 'west_core'
}
```

```
v.validate(bad_example, schema)
v.errors
```

```
{'last_drop_off': ['must be of string type'],
 'pick_up': ['unallowed value sakjfnskffs'],
 'time': ['must be of datetime type']}
```

```
df = pd.DataFrame({
    'pick_up': ['work'],
    'time': [pd.Timestamp(2018, 4, 1, 9, 30)],
    'last_drop_off': ['home'],
    'last_pick_up': ['downtown_core']
})
df
```

	last_drop_off	last_pick_up	pick_up	time
0	home	downtown_core	work	2018-04-01 09:30:00

```
df = pd.DataFrame({
    'pick_up': ['work'],
    'time': [pd.Timestamp(2018, 4, 1, 9, 30)],
    'last_drop_off': ['home'],
    'last_pick_up': ['downtown_core']
})
df
```

	last_drop_off	last_pick_up	pick_up	time
0	home	downtown_core	work	2018-04-01 09:30:00

```
df_list = df.to_dict(orient='list')
```

```
{'last_drop_off': ['home'],
 'last_pick_up': ['downtown_core'],
 'pick_up': ['work'],
 'time': [Timestamp('2018-04-01 09:30:00')]}
```

```
from cerberus import Validator
from copy import deepcopy

class PandasValidator(Validator):
```

```python
from cerberus import Validator
from copy import deepcopy

class PandasValidator(Validator):

    def validate(self, document, schema,
                 update=False, normalize=True):
        document = document.to_dict(orient='list')
        schema = self.transform_schema(schema)
        super().validate(document, schema,
                         update=update, normalize=normalize)

    def transform_schema(self, schema):
        schema = deepcopy(schema)
        for k, v in schema.items():
            schema[k] = {'type': 'list', 'schema': v}
        return schema
```

```python
from cerberus import Validator
from copy import deepcopy

class PandasValidator(Validator):

    def validate(self, document, schema,
                 update=False, normalize=True):
        document = document.to_dict(orient='list')
        schema = self.transform_schema(schema)
        super().validate(document, schema,
                         update=update, normalize=normalize)

    def transform_schema(self, schema):
        schema = deepcopy(schema)
        for k, v in schema.items():
            schema[k] = {'type': 'list', 'schema': v}
        return schema
```

```
schema = {
    'time': {'type': 'datetime', 'nullable': False},
    'pick_up': {'type': 'string', 'nullable': False, 'allowed': [
        'home', 'work', 'climbing', 'west_core', 'downtown_core', 'other', 'art']
    },
    'last_drop_off': {'type': 'string', 'nullable': True},
    'last_pick_up': {'type': 'string', 'nullable': True}
}

df = pd.DataFrame({
    'pick_up': ['work'],
    'time': [None],
    'last_drop_off': ['home'],
    'last_pick_up': ['downtown_core']
})
```

```
schema = {
    'time': {'type': 'datetime', 'nullable': False},
    'pick_up': {'type': 'string', 'nullable': False, 'allowed': [
        'home', 'work', 'climbing', 'west_core', 'downtown_core', 'other', 'art']
    },
    'last_drop_off': {'type': 'string', 'nullable': True},
    'last_pick_up': {'type': 'string', 'nullable': True}
}

df = pd.DataFrame({
    'pick_up': ['work'],
    'time': [None],
    'last_drop_off': ['home'],
    'last_pick_up': ['downtown_core']
})

pv = PandasValidator()
```

```
schema = {
    'time': {'type': 'datetime', 'nullable': False},
    'pick_up': {'type': 'string', 'nullable': False, 'allowed': [
        'home', 'work', 'climbing', 'west_core', 'downtown_core', 'other', 'art']
    },
    'last_drop_off': {'type': 'string', 'nullable': True},
    'last_pick_up': {'type': 'string', 'nullable': True}
}

df = pd.DataFrame({
    'pick_up': ['work'],
    'time': [None],
    'last_drop_off': ['home'],
    'last_pick_up': ['downtown_core']
})

pv = PandasValidator()
pv.validate(df, schema)
```

```
schema = {
    'time': {'type': 'datetime', 'nullable': False},
    'pick_up': {'type': 'string', 'nullable': False, 'allowed': [
        'home', 'work', 'climbing', 'west_core', 'downtown_core', 'other', 'art']
    },
    'last_drop_off': {'type': 'string', 'nullable': True},
    'last_pick_up': {'type': 'string', 'nullable': True}
}

df = pd.DataFrame({
    'pick_up': ['work'],
    'time': [None],
    'last_drop_off': ['home'],
    'last_pick_up': ['downtown_core']
})

pv = PandasValidator()
pv.validate(df, schema)
pv.errors {'time': [{0: ['null value not allowed']}]}
```

FREE EDITION

5,000 events per month.
Includes all Standard features.

Try it free

```
import rollbar
rollbar.init(' 78asd86d876ad8678sdadsa687d ')

pv = PandasValidator()
pv.validate(df, schema)
pv.errors
```

```
import rollbar
rollbar.init(' 78asd86d876ad8678sdadsa687d ')

pv = PandasValidator()
pv.validate(df, schema)
pv.errors

if pv.errors:
    rollbar.report_message(f'Got a schema error: {str(pv.errors)}', 'warning')
```

bike Dashboard Items Deploys People RQL Settings

Top 10 items in last 24 hours

view all

24hr Trend	Count		Title
	1	-	#6 Got a schema error: {'time': [{0: ['null value not allowed']}]}

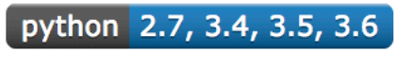
#4

Learn how to CLI

input →  → output


$ click_

$ click_

Python Fire 

Python Fire is a library for automatically generating command line interfaces (CLIs) from absolutely any Python object.

- Python Fire is a simple way to create a CLI in Python. [1]
- Python Fire is a helpful tool for developing and debugging Python code. [2]
- Python Fire helps with exploring existing code or turning other people's code into a CLI. [3]
- Python Fire makes transitioning between Bash and Python easier. [4]
- Python Fire makes using a Python REPL easier by setting up the REPL with the modules and variables you'll need already imported and created. [5]

Installation

To install Python Fire with pip, run: `pip install fire`

To install Python Fire with conda, run: `conda install fire -c conda-forge`

```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score

df = pd.read_csv('max_bike_data.csv')
```

```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score

df = pd.read_csv('max_bike_data.csv')

df['time'] = pd.to_datetime(df['time'])
df = df[df['drop_off'].notnull()]

def predict(X):
    return np.where(X.time.dt.hour <= 10, 'work', 'home')

y_pred = predict(df)
```

```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score

df = pd.read_csv('max_bike_data.csv')

df['time'] = pd.to_datetime(df['time'])
df = df[df['drop_off'].notnull()]

def predict(X):
    return np.where(X.time.dt.hour <= 10, 'work', 'home')

y_pred = predict(df)
accuracy_score(df['drop_off'], y_pred)
```

```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score

df = pd.read_csv('max_bike_data.csv')

df['time'] = pd.to_datetime(df['time'])
df = df[df['drop_off'].notnull()]

def predict(X):
    return np.where(X.time.dt.hour <= 10, 'work', 'home')

y_pred = predict(df)
accuracy_score(df['drop_off'], y_pred)
```

```
0.593103448275862
```

```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score

df = pd.read_csv('max_bike_data.csv')

df['time'] = pd.to_datetime(df['time'])
df = df[df['drop_off'].notnull()]

def predict(X):
    return np.where(X.time.dt.hour <= 10, 'work', 'home')

y_pred = predict(df)
accuracy_score(df['drop_off'], y_pred)
```

```
0.593103448275862
```

```
output = pd.DataFrame({'time': df.time, 'pick_up': df.pick_up,
    'drop_off': df.drop_off, 'predicted_drop_off': y_pred})

output.to_csv('output.csv', index=False)
```

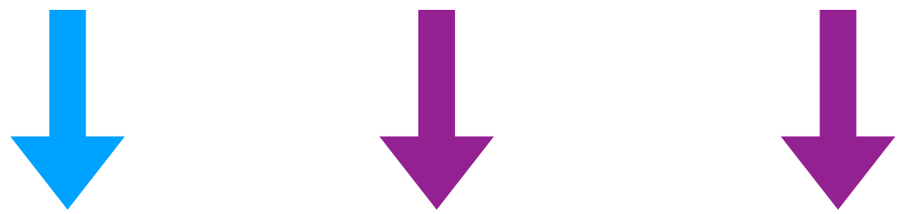
< refactor >

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


def _predict(X):
    return np.where(X.time.dt.hour <= 10, 'work', 'home')
```

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


def _predict(X):
    return np.where(X.time.dt.hour <= 10, 'work', 'home')

def predict(file):
    df = pd.read_csv(file)
    df['time'] = pd.to_datetime(df['time'])
    df = df[df['drop_off'].notnull()]
    output = pd.DataFrame({
        'time': df.time,
        'pick_up': df.pick_up,
        'drop_off': df.drop_off,
        'predicted_drop_off': _predict(df)})
    output.to_csv('output.csv', index=False)
    print('success!')
```

```python
import pandas as pd
import numpy as np
from fire import Fire


def _predict(X):
    return np.where(X.time.dt.hour <= 10, 'work', 'home')


def predict(file):
    df = pd.read_csv(file)
    df['time'] = pd.to_datetime(df['time'])
    df = df[df['drop_off'].notnull()]
    output = pd.DataFrame({
        'time': df.time,
        'pick_up': df.pick_up,
        'drop_off': df.drop_off,
        'predicted_drop_off': _predict(df)})
    output.to_csv('output.csv', index=False)
    print('success!')


if __name__ == '__main__':
    Fire()
```

```
$ python model.py
```



```
$ python model.py predict
```

```
$ python model.py predict --file=max_bike_data.csv
```

```
$ python model.py predict my_bike_data.csv
```

```
$ python model.py predict sunny_bike_data.csv
```

```
$ python model.py predict sunny_bike_data.csv
```

```python
if __name__ == '__main__':
    Fire()
```

#5

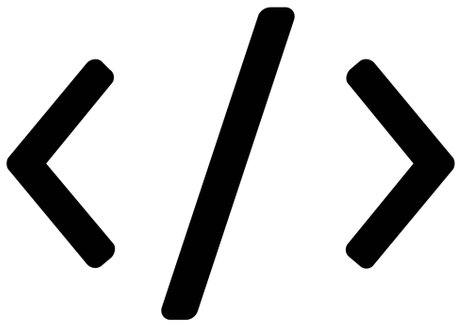
#5

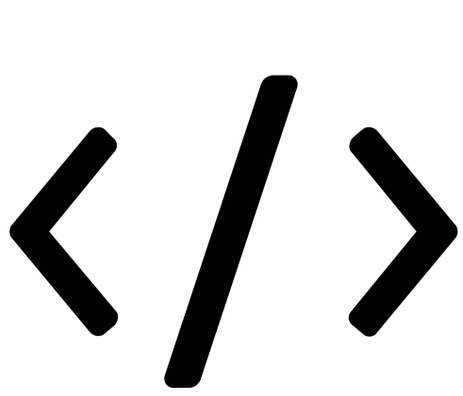
mummify

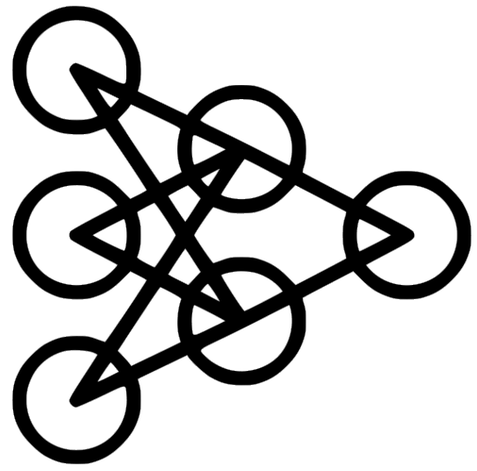
you suck at git

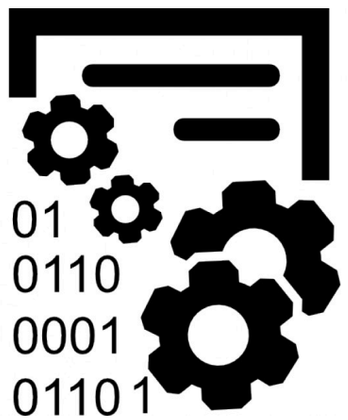
you suck at git
and logging

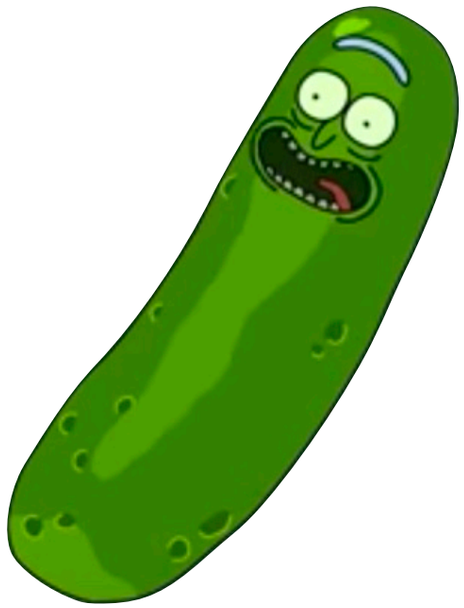
you suck at git
and logging
but it’s not your fault








01
0110
0001
0110 1




TIME FOR A LIVE
DEMO
38


TIME FOR A LIVE DEMO
38
WHAT COULD GO WRONG?

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
from sklearn.pipeline import make_pipeline
from sklearn_pandas import DataFrameMapper, CategoricalImputer
from helpers import DateEncoder

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)

mapper = DataFrameMapper([
    ('time', DateEncoder(), {'input_df': True}),
    ('pick_up', LabelBinarizer()),
    ('last_drop_off', [CategoricalImputer(), LabelBinarizer()]),
    ('last_pick_up', [CategoricalImputer(), LabelBinarizer()])
])

lb = LabelBinarizer()
y_train = lb.fit_transform(y_train)
```

model.py base

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
pipe = make_pipeline(mapper, model)
pipe.fit(X_train, y_train)

acc_train = pipe.score(X_train, y_train)
acc_test = pipe.score(X_test, lb.transform(y_test))

print(f'Training: {acc_train:.3f}, Testing: {acc_test:.3f}')
```

model.py add

```
import mummify

mummify.log(f'Training: {acc_train:.3f}, Testing: {acc_test:.3f}’)
```

model.py mummify

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
```

model.py model swap 1

```
from sklearn.neural_network import MLPClassifier
model = MLPClassifier()
```

model.py model swap 2

```
from sklearn.neural_network import MLPClassifier
model = MLPClassifier(max_iter=2000)
```

model.py model swap 2 + max_iter

```
mummify history
```

```
mummify switch

mummify history
```

mummify command line

```
git --git-dir=.mummify status
```

mummify is just git

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=6)
```

mummify adjust hypers on 1

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=4)
```

mummify adjust hypers on 1

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=1000)
```

mummify switch back to rf

```
import pickle
with open('rick.pkl', 'wb') as f:
    pickle.dump((pipe, lb), f)
```

pickle model

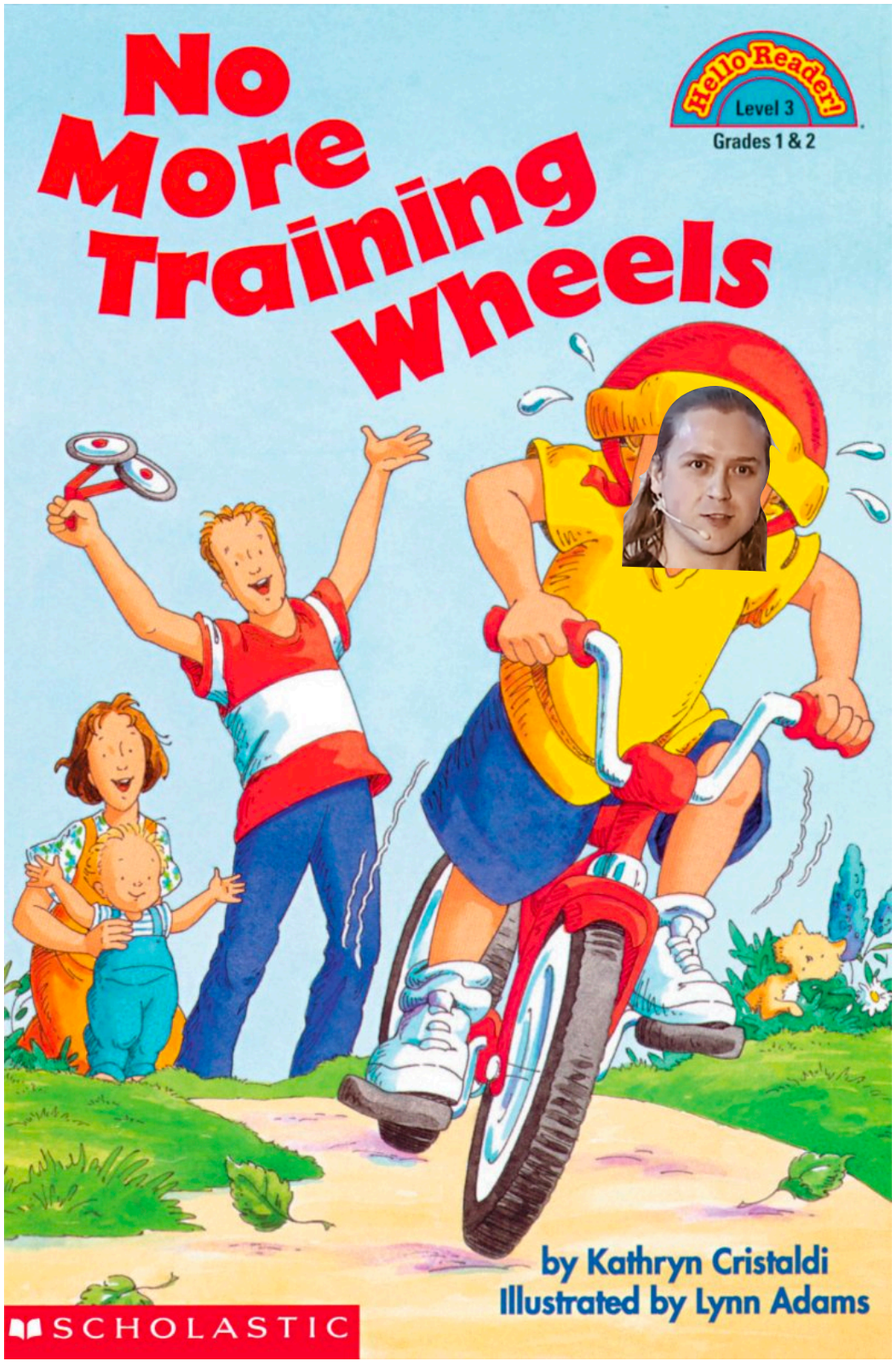
```
import pickle
from fire import Fire
import pandas as pd

with open('rick.pkl', 'rb') as f:
    pipe, lb = pickle.load(f)

def predict(file):
    df = pd.read_csv(file)
    df['time'] = pd.to_datetime(df['time'])
    y = pipe.predict(df)
    y = lb.inverse_transform(y)[0]
    return f'Max is probably going to {y}'

if __name__ == '__main__':
    Fire(predict)
```

predict.py

```
$ git --git-dir=.mummify add .
$ git --git-dir=.mummify commit -m 'add predict'
```

```
time,pick_up,last_drop_off,last_pick_up
2018-04-09 9:15:52,home,other,home
```

new_data.csv


No More Training Wheels
Hello Reader!
Level 3
Grades 1 & 2
by Kathryn Cristaldi
Illustrated by Lynn Adams
SCHOLASTIC

https://github.com/maxhumber/mummify

pip install mummify

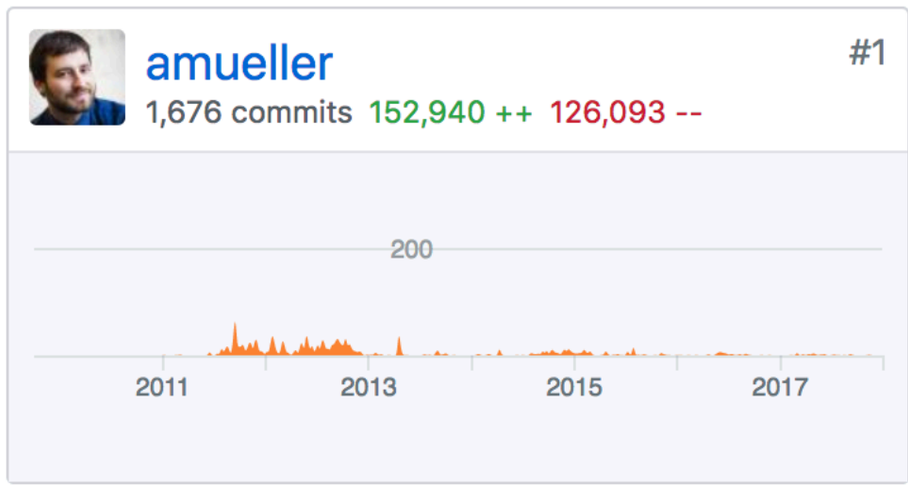
https://github.com/maxhumber/mummify

```
pip install mummify
```

```
conda install -c maxhumber mummify
```

#END

hydrogen



sklearn

cerberus

sklearn-pandas

mummify

First 50 get it free!

This book is 99% complete

https://leanpub.com/personal_finance_with_python/c/anaconda

maxhumber