

Data Science: It's Easy as Py!

Michael Becker

What My Coworkers Think I Do

$$h_{w,b}(x) = g(w^T x + b)$$

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x + b)$$

$$\hat{\gamma} = \min_{i=1,\dots,m} \hat{\gamma}^{(i)}$$

$$w^T \left(x^{(i)} - \gamma^{(i)} \frac{w}{\|w\|} \right) + b = 0$$

What I Actually Do

```
import pandas as pd
```

```
import seaborn as sns
```

```
from sklearn.svm import SVC
```

What I'll Cover

- What is Data Science?

What I'll Cover

- What is Data Science?
- The OSEMN process

What I'll Cover

- What is Data Science?
- The OSEMN process
- Writing a custom web scraper

What I'll Cover

- What is Data Science?
- The OSEMN process
- Writing a custom web scraper
- Scrubbing the data

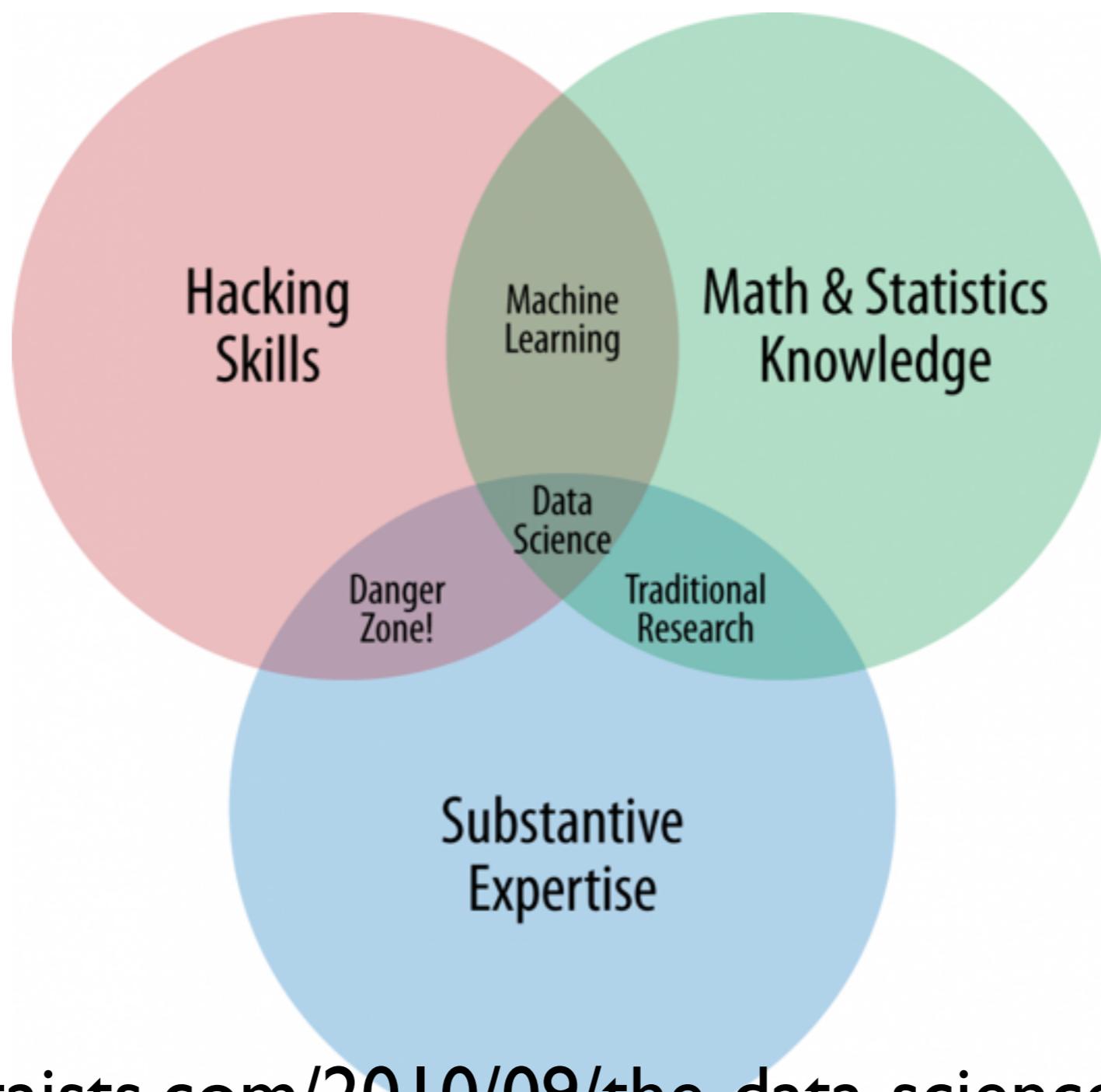
What I'll Cover

- What is Data Science?
- The OSEMN process
- Writing a custom web scraper
- Scrubbing the data
- Data Exploration

What I'll Cover

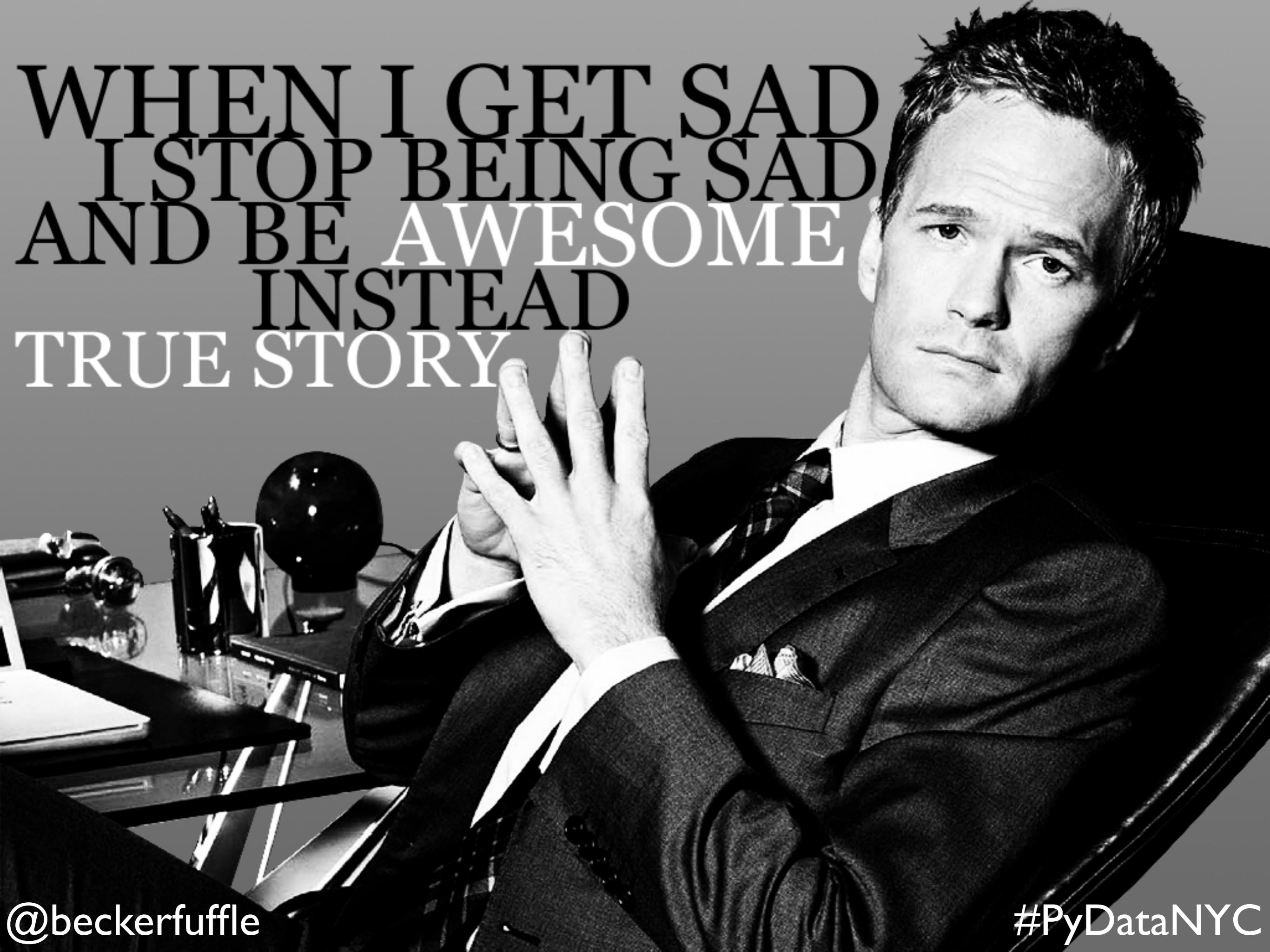
- What is Data Science?
- The OSEMN process
- Writing a custom web scraper
- Scrubbing the data
- Data Exploration
- Predictive Modeling

What is Data Science?



<http://www.dataists.com/2010/09/the-data-science-venn-diagram/>

WHEN I GET SAD
I STOP BEING SAD
AND BE AWESOME
INSTEAD
TRUE STORY



OSEMN!

- Obtain

OSEMN!

- Obtain
- Scrub

OSEMN!

- Obtain
- Scrub
- Explore

OSEMN!

- Obtain
- Scrub
- Explore
- Model

OSEMN!

- Obtain
- Scrub
- Explore
- Model
- iNterpret

Language Classification



Sign in

Translate



Japanese English Chinese Detect language ▾



English Chinese (Simplified) Latin ▾

Translate

yada yada yada



亚达内容十分重要

Translate from: [Turkish](#)

Yà dá nèiróng shí fèn zhòngyào

Examples of yada yada yada

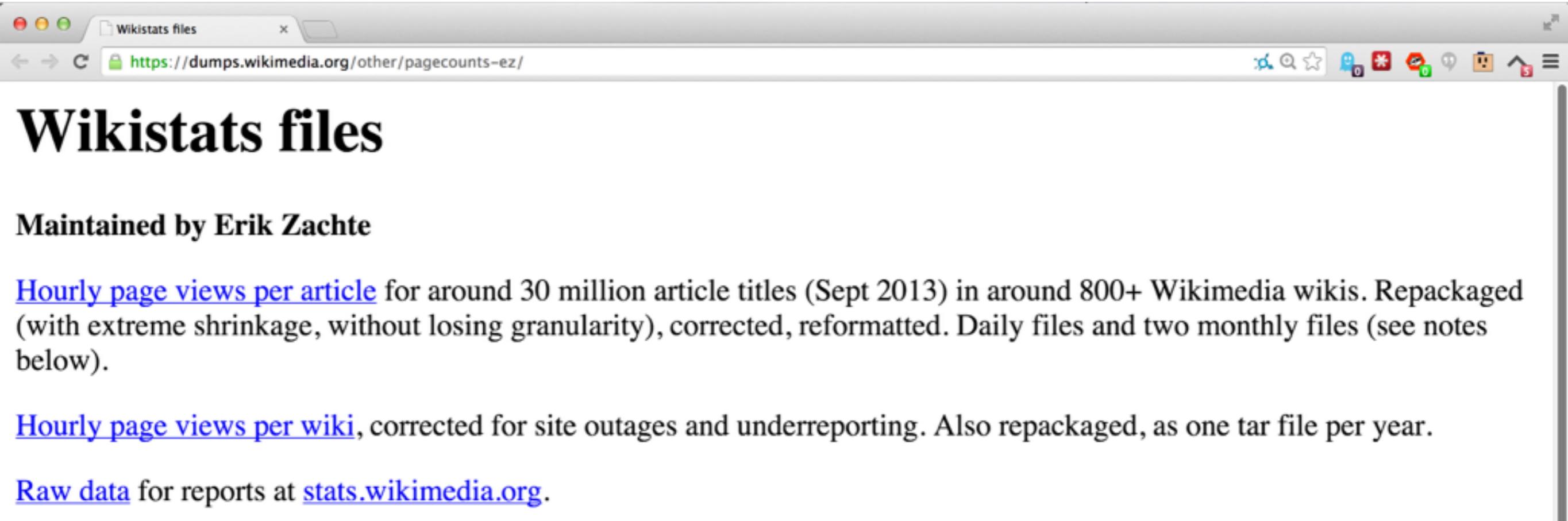
In the best scenario, a framework/teacher says, 'We do it this way because **yada yada yada**, so we want you to try it this way until you're comfortable with it.'

在最好的情况下，框架/老师说，“我们做的这种方式，因为亚达内容十分重要，所以我们希望你尝试一下这种方法，直到你舒服。”

automatically translated by Google

API?!
Where
we're
going we
don't
need APIs!

Scrape All The Things!



A screenshot of a web browser window titled "Wikistats files". The URL in the address bar is <https://dumps.wikimedia.org/other/pagecounts-ez/>. The page content includes the title "Wikistats files", a maintainer note by Erik Zachte, and descriptions of hourly page view data for articles and wikis.

Wikistats files

Maintained by Erik Zachte

[Hourly page views per article](#) for around 30 million article titles (Sept 2013) in around 800+ Wikimedia wikis. Repackaged (with extreme shrinkage, without losing granularity), corrected, reformatted. Daily files and two monthly files (see notes below).

[Hourly page views per wiki](#), corrected for site outages and underreporting. Also repackaged, as one tar file per year.

[Raw data](#) for reports at stats.wikimedia.org.

Scrape All The Things!

The screenshot shows a web browser window with the title bar "W Export pages – Wikipedia, X". The address bar contains the URL "https://en.wikipedia.org/wiki/Special:Export". The page itself is titled "Export pages" and is a "Special page". On the left sidebar, there's a "WIKIPEDIA The Free Encyclopedia" logo, a "Main page" link, and sections for "Contents", "Featured content", "Current events", "Random article", "Donate to Wikipedia", and "Wikimedia Shop". Below these are sections for "Interaction" (with links to "Help", "About Wikipedia", "Community portal", "Recent changes", and "Contact page") and "Tools" (with links to "Upload file", "Special pages", and "Printable version"). At the bottom, there's a "Languages" section with a gear icon and a row of colored bars (red, blue, green, yellow) with corresponding text below them: "Include only the current revision, not the full history" (checked), "Include templates", and "Include categories".

Scrape All The Things!

The screenshot shows a web browser interface with the following elements:

- Left Sidebar (Interaction):** Includes links for Help, About Wikipedia, Community portal, Recent changes, and Contact page.
- Top Bar:** Shows "Full history exports are limited to 1000 revisions." and a search bar with "Michael Becker".
- Tools Tab:** Contains links for Upload file and Special pages.
- Developer Tools Header:** Elements, Network, Sources, Timeline, Profiles, Resources, Audits, Console, HTTPS Everywhere.
- Network Tab Headers:** Preserve log (unchecked), Disable cache (checked).
- Table Headers (Network Tab):** Name, Path, Method, Status, Type, Initiator, Size, Time, Timeline.
- Table Data (Network Tab):** One row for "index.php?title=Special:Export&action=su... /w", showing Method: GET, Status: 200, Type: application/..., Initiator: Other, Size Content: 0 B, Time Latency: 383 ms, and Timeline bar spanning from 100 ms to 200 ms.
- Context Menu (Open Link in New Tab):** Options include Open Link in New Tab, Copy Link Address, Copy Request Headers, Copy Response Headers, Copy Response, Copy as cURL (selected), Copy All as HAR, Save as HAR with Content, Clear Browser Cache, and Clear Browser Cookies.
- Bottom Status Bar:** 1 requests | 0 B transferred.
- Bottom Footer:** Wikipedia-20141118....xml.

Scrape All The Things!

```
from uncurl import parse

print parse("curl 'https://en.wikipedia.org/w/index.php?title=Special:Export&action=submit'",

requests.post("https://en.wikipedia.org/w/index.php?title=Special:Export&action=submit",
    data='catname=&pages=Michael+Becker&curonly=1&wpDownload=1',
    headers={
        "Accept": "text/html,application/xhtml+xml",
        "Accept-Encoding": "gzip,deflate",
        "Accept-Language": "en-US,en;q=0.8",
        "Cache-Control": "no-cache",
        "Connection": "keep-alive",
        "Content-Type": "application/x-www-form-urlencoded",
        "DNT": "1",
        "Origin": "https://en.wikipedia.org",
        "Pragma": "no-cache",
        "Referer": "https://en.wikipedia.org/wiki/Special:Export",
```



@beckerfuffle

#PyDataNYC

Scrub

← → C GitHub, Inc. [US] <https://github.com/bwbaugh/wikipedia-extractor>

Wikipedia Extractor

Introduction

The project uses the *Italian Wikipedia* as source of documents for several purposes: as training data and as source of data to be annotated.

The Wikipedia maintainers provide, each month, an XML *dump* of all documents in the database: it consists of a single XML file containing the whole encyclopedia, that can be used for various kinds of analysis, such as statistics, service lists, etc.

Wikipedia dumps are available from [Wikipedia database download](#).

The Wikipedia extractor tool generates plain text from a Wikipedia database dump, discarding any other information or annotation present in Wikipedia pages, such as images, tables, references and lists.

Scrub

GitHub, Inc. [US] <https://github.com/bwbaugh/wikipedia-extractor>

For this document the Wikipedia extractor produces the following plain text:

```
<doc id="2" url="http://it.wikipedia.org/wiki/Armonium">
Armonium.
L'armonium (in francese, "harmonium") è uno strumento musicale azionato con
una tastiera, detta manuale. Sono stati costruiti anche alcuni armonium con
due manuali.

Armonium occidentale.
Come l'organo, l'armonium è utilizzato tipicamente in chiesa, per l'esecuzione
di musica sacra, ed è fornito di pochi registri, quando addirittura in certi
casi non ne possiede nemmeno uno: il suo timbro è molto meno ricco di quello
organistico e così pure la sua estensione.

...
</doc>
```

The extraction tool is written in Python and requires no additional library. it aims to achieve high accuracy in extraction task.

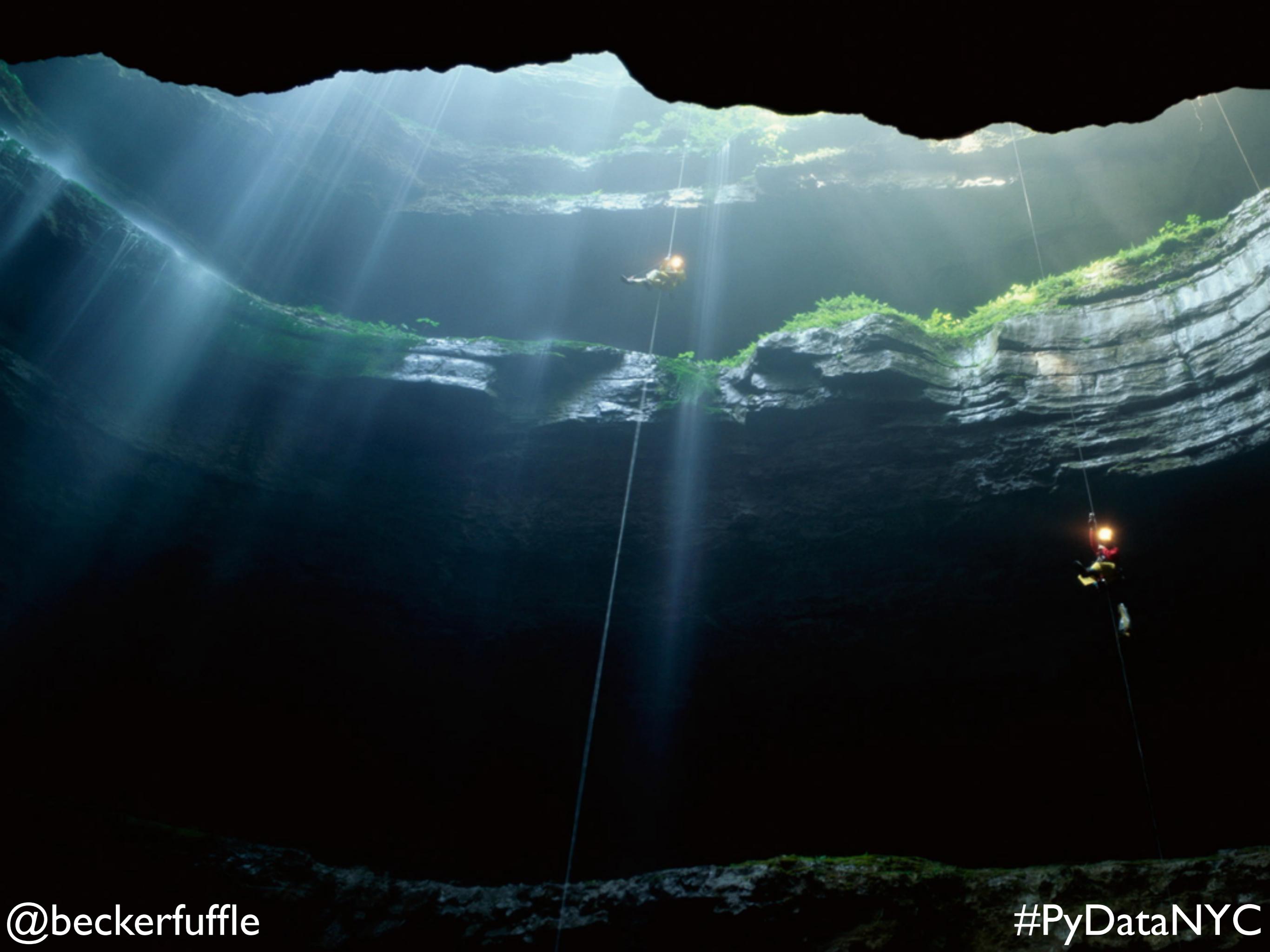
Wikipedia articles are converted to the Mediawiki Markup language, which provides a simple notation

Scrub

```
import bz2
import re

article = re.compile(r'<doc id="( ?P<id>\d+)" url="( ?P<url>[^"])+" t'

def parse(filename):
    data = ""
    with bz2.BZ2File(filename, 'r') as f:
        for line in f:
            line = line.decode('utf-8')
            data += line
            if line.count('</doc>'):
                m = article.search(data)
                if m:
                    yield m.groupdict()
                data = ""
```



@beckerfuffle

#PyDataNYC

Explore

```
files = [f for f in os.listdir('.') if os.path.isfile(f)]
top_letters = []
for lang in files:
    print(lang)
    c = Counter()
    for article in parse(lang):
        c['num_articles'] += 1
        for letter in article['content']:
            c[letter] += 1
        c['num_letters'] += 1
    d = dict(c.most_common(2000))
    top_letters.append(d)
```

Pandas

```
from pandas import DataFrame  
df = DataFrame(top_letters)  
df.fillna(0, inplace=True)  
df = df.set_index('lang')  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Index: 127 entries, ace to zh  
Columns: 10657 entries, to □  
dtypes: float64(10578), int64(79)
```

Pandas

```
df2 = df.join(df.div(df['num_letters'], axis='index'), rsuffix='_perc')
percs = [col for col in df2.columns if col.endswith('_perc')]
df3 = df2[percs]
df3
```

	_perc	_perc	_perc	!_perc	"_perc	#_perc	\$_perc	%_perc	&_perc
lang									
ace	0	0.009957	0.143409	0.000000	0.002081	0.000000	0.000018	0.000208	0.000000
af	0	0.004649	0.154431	0.000012	0.003242	0.000001	0.000017	0.000099	0.000013
als	0	0.006280	0.150704	0.000031	0.004069	0.000002	0.000001	0.000756	0.000014
am	0	0.011811	0.185362	0.000098	0.001537	0.000003	0.000010	0.000159	0.000050
ar	0	0.003733	0.163212	0.000031	0.001961	0.000002	0.000022	0.000354	0.000000

Pandas

```
df2 = df.join(df.div(df['num_letters'], axis='index'), rsuffix='_perc')
percs = [col for col in df2.columns if col.endswith('_perc')]
df3 = df2[percs]
df3
```

	_perc	_perc	_perc	!_perc	"_perc	#_perc	\$_perc	%_perc	&_perc
lang									
ace	0	0.009957	0.143409	0.000000	0.002081	0.000000	0.000018	0.000208	0.000000
af	0	0.004649	0.154431	0.000012	0.003242	0.000001	0.000017	0.000099	0.000013
als	0	0.006280	0.150704	0.000031	0.004069	0.000002	0.000001	0.000756	0.000014
am	0	0.011811	0.185362	0.000098	0.001537	0.000003	0.000010	0.000159	0.000050
ar	0	0.003733	0.163212	0.000031	0.001961	0.000002	0.000022	0.000354	0.000000

Visualize

```
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import seaborn as sns

num_clusters = 4

palette = sns.color_palette("colorblind", num_clusters)
est = KMeans(num_clusters, max_iter=30000) # 4 clusters
est.fit(df3.values)
y_kmeans = est.predict(df3.values)

pca = PCA(n_components=2)
pca.fit(df3.values)
X_trans = pca.transform(df3.values)
plt.scatter(X_trans[:, 0], X_trans[:, 1], c=[palette[y] for y in y_kmeans], s=50)
```

Visualize

```
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import seaborn as sns

num_clusters = 4

palette = sns.color_palette("colorblind", num_clusters)
est = KMeans(num_clusters, max_iter=30000) # 4 clusters
est.fit(df3.values)
y_kmeans = est.predict(df3.values)

pca = PCA(n_components=2)
pca.fit(df3.values)
X_trans = pca.transform(df3.values)
plt.scatter(X_trans[:, 0], X_trans[:, 1], c=[palette[y] for y in y_kmeans], s=50)
```

Visualize

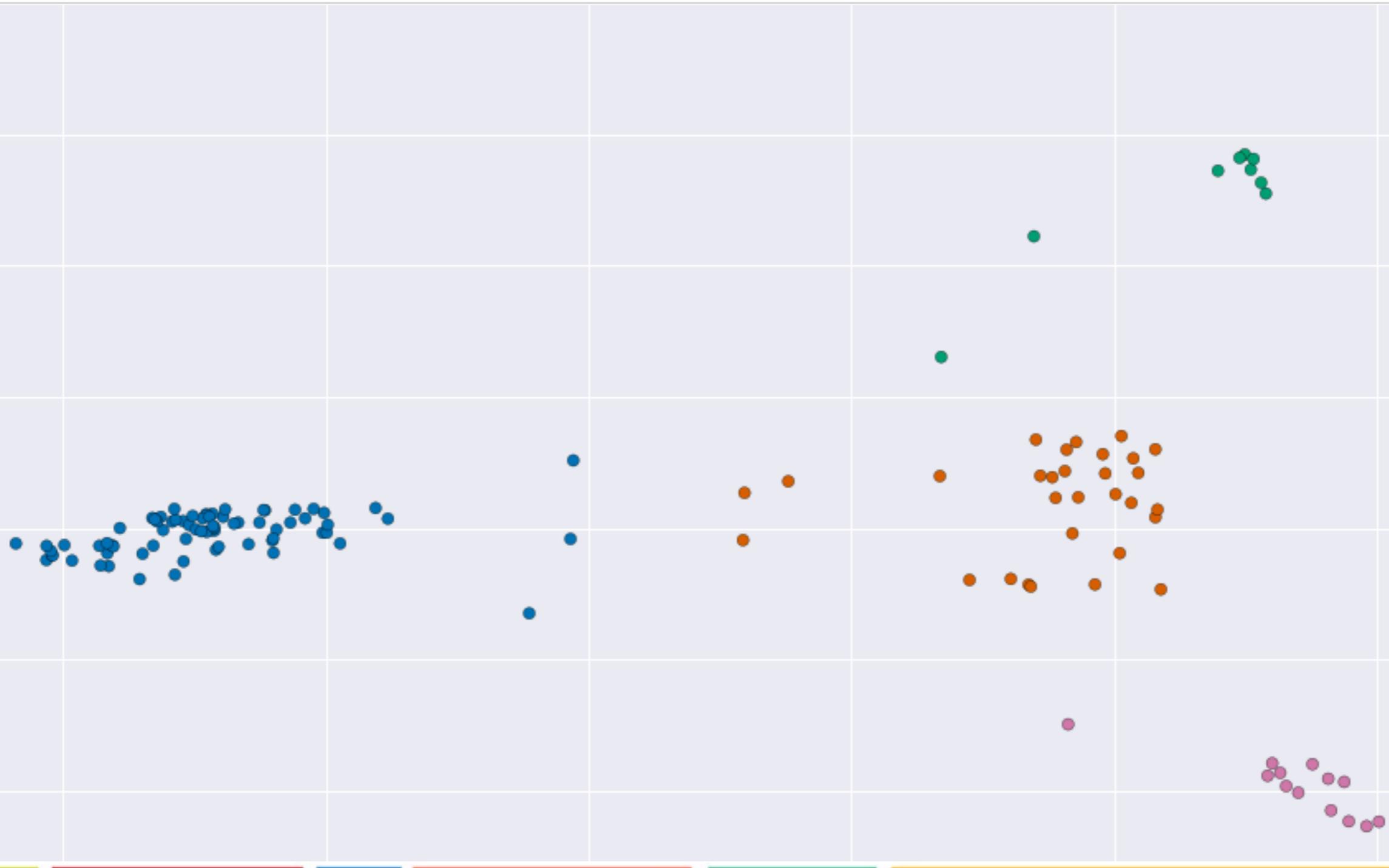
```
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import seaborn as sns

num_clusters = 4

palette = sns.color_palette("colorblind", num_clusters)
est = KMeans(num_clusters, max_iter=30000) # 4 clusters
est.fit(df3.values)
y_kmeans = est.predict(df3.values)

pca = PCA(n_components=2)
pca.fit(df3.values)
X_trans = pca.transform(df3.values)
plt.scatter(X_trans[:, 0], X_trans[:, 1], c=[palette[y] for y in y_kmeans], s=50)
```

Visualize



Explore More

```
from sklearn.metrics.pairwise import pairwise_distances

cluster_dfs = {}
cluster_langs = {}
cluster_distances = {}
for cluster_num in range(4):
    indexes = [i for i in range(y_kmeans.shape[0]) if y_kmeans[i] == cluster_num]
    cluster_langs[cluster_num] = [langs[i] for i in indexes]
    cluster_dfs[cluster_num] = df3.loc[cluster_langs[cluster_num],:]

# Calculate pairwise distances and display
print('Cluster #{0}'.format(cluster_num))
cluster_distances[cluster_num] = pairwise_distances(cluster_dfs[cluster_num].values)
n, m = cluster_distances[cluster_num].shape
distances = set([])
for i in range(n):
    for j in range(m):
        if i == j:
            continue
        distances.add((cluster_distances[cluster_num][i,j], tuple(sorted([i, j]))))
for a in sorted(distances)[:20]:
    print_sim(a[0], a[1][0], a[1][1], cluster_langs[cluster_num])
print()
```

Explore More

Cluster #0

0.00382418517864 ('Croatian (Hrvatski)', 'Serbo-Croatian (Srpskohrvatski / Српскохрватски)')
0.00520780738783 ('Bosnian (Bosanski)', 'Croatian (Hrvatski)')
0.00589451786297 ('Bosnian (Bosanski)', 'Serbo-Croatian (Srpskohrvatski / Српскохрватски)')
0.00834338647842 ('Indonesian (Bahasa Indonesia)', 'Malay (Bahasa Melayu)')
0.0192725635062 ('Danish (Dansk)', 'Norwegian (Bokmål) (Norsk (Bokmål))')
0.0232608076499 ('English (English)', 'Scots (Scots)')

Cluster #1

0.0392927583851 ('Arabic (العربية)', 'Egyptian Arabic (مصرى (Maṣri))')
0.0437481324993 ('Persian (فارسی)', 'Mazandarani (هزارونی)')
0.0508231659684 ('Western Panjabi (Shāhmukhī Pañjābī)', 'Urdu (اُردو)')

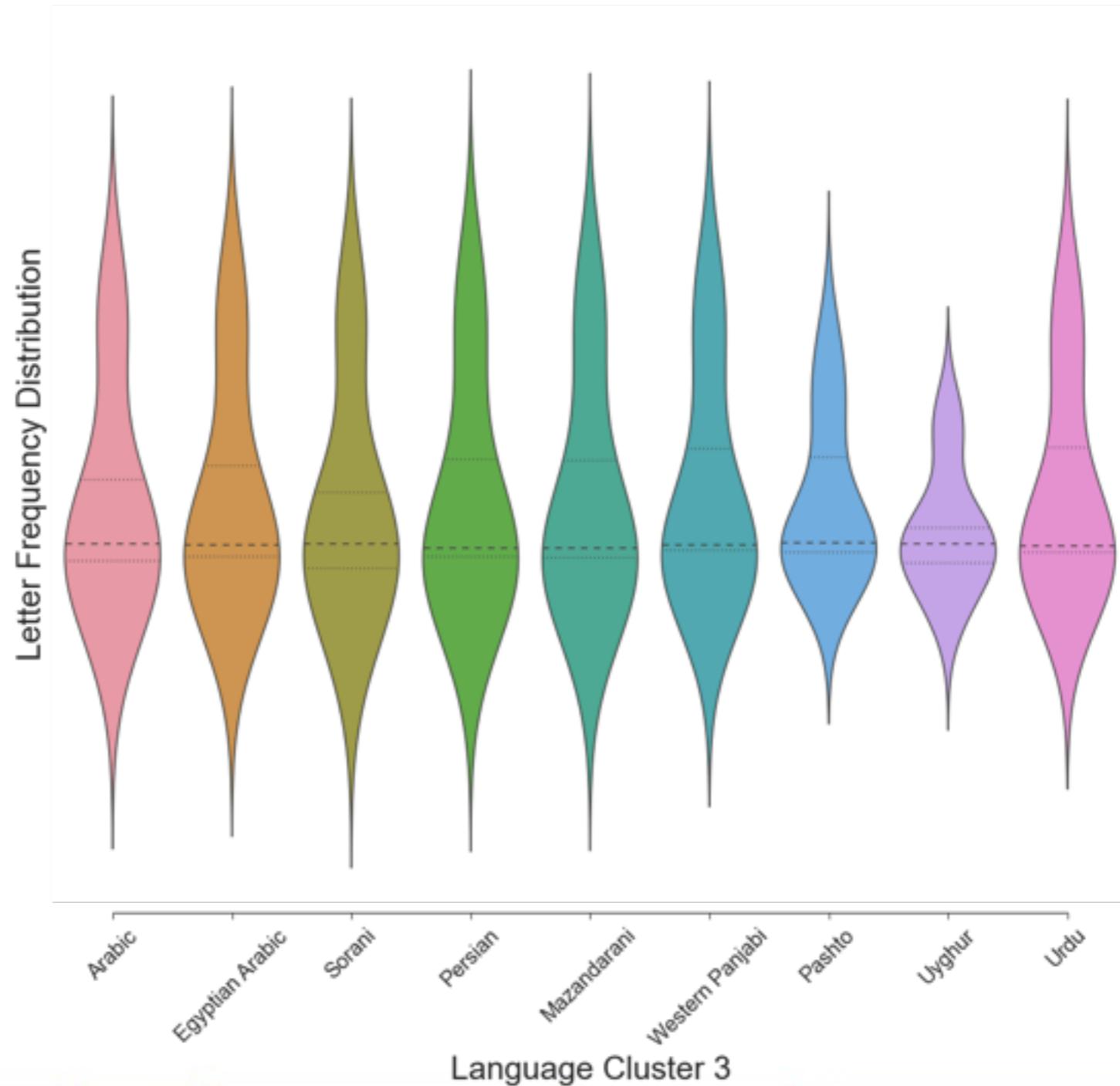
Cluster #2

0.0414121771519 ('Wu (吴语)', 'Chinese (中文)')
0.0425573179877 ('Wu (吴语)', 'Cantonese (粵語)')
0.0454926761733 ('Cantonese (粵語)', 'Chinese (中文)')
0.0468059566543 ('Gan (贛語)', 'Cantonese (粵語)')
0.0532862863375 ('Gan (贛語)', 'Wu (吴语)')

Cluster #3

0.025823417882 ('Chechen (Нохчийн)', 'Russian (Русский)')
0.0352906788461 ('Bulgarian (Български)', 'Macedonian (Македонски)')
0.0460701274904 ('Bulgarian (Български)', 'Chechen (Нохчийн)')
0.0495084304888 ('Bulgarian (Български)', 'Russian (Русский)')

Explore Even More

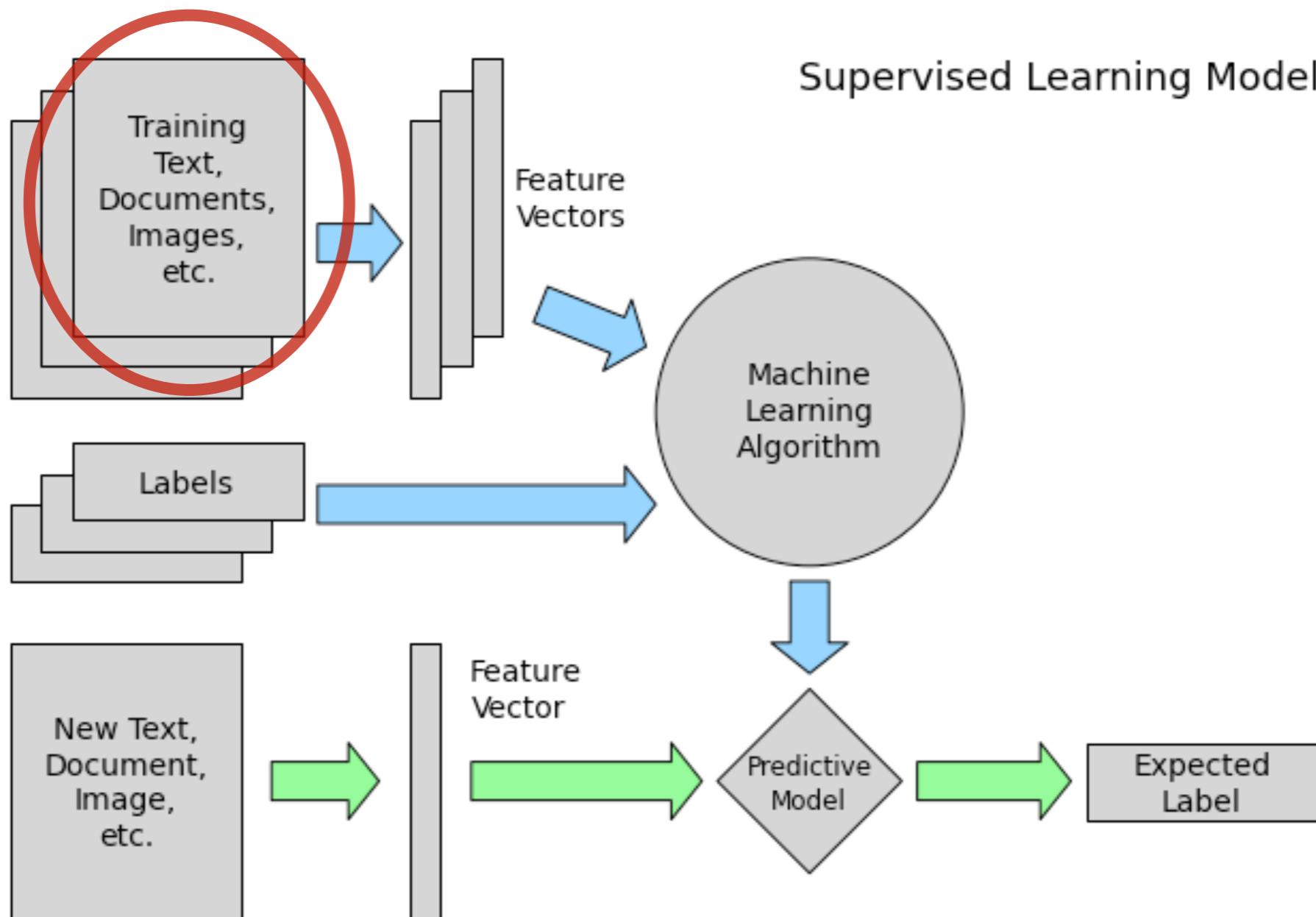




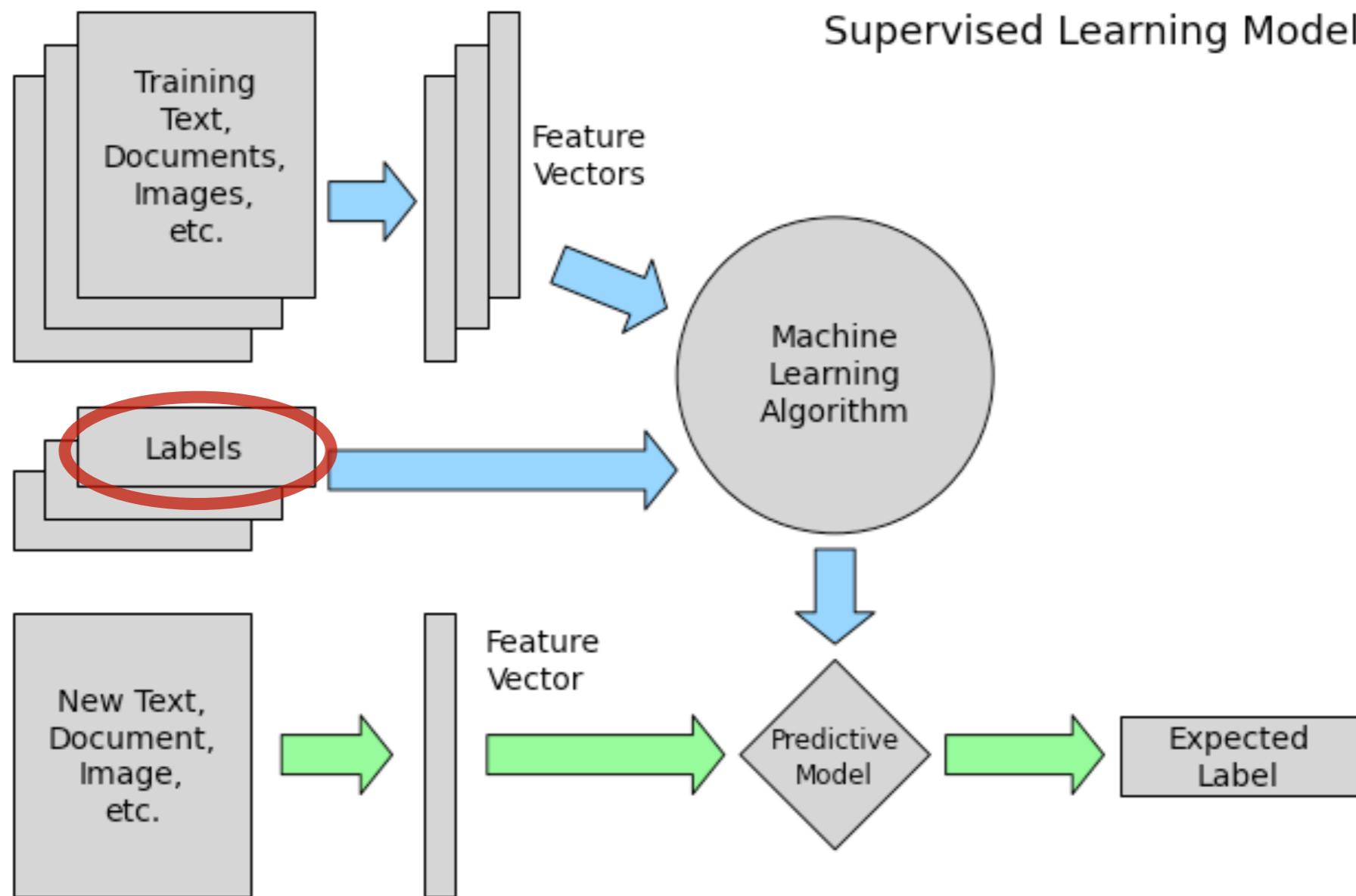
@beckerfuffle

#PyDataNYC

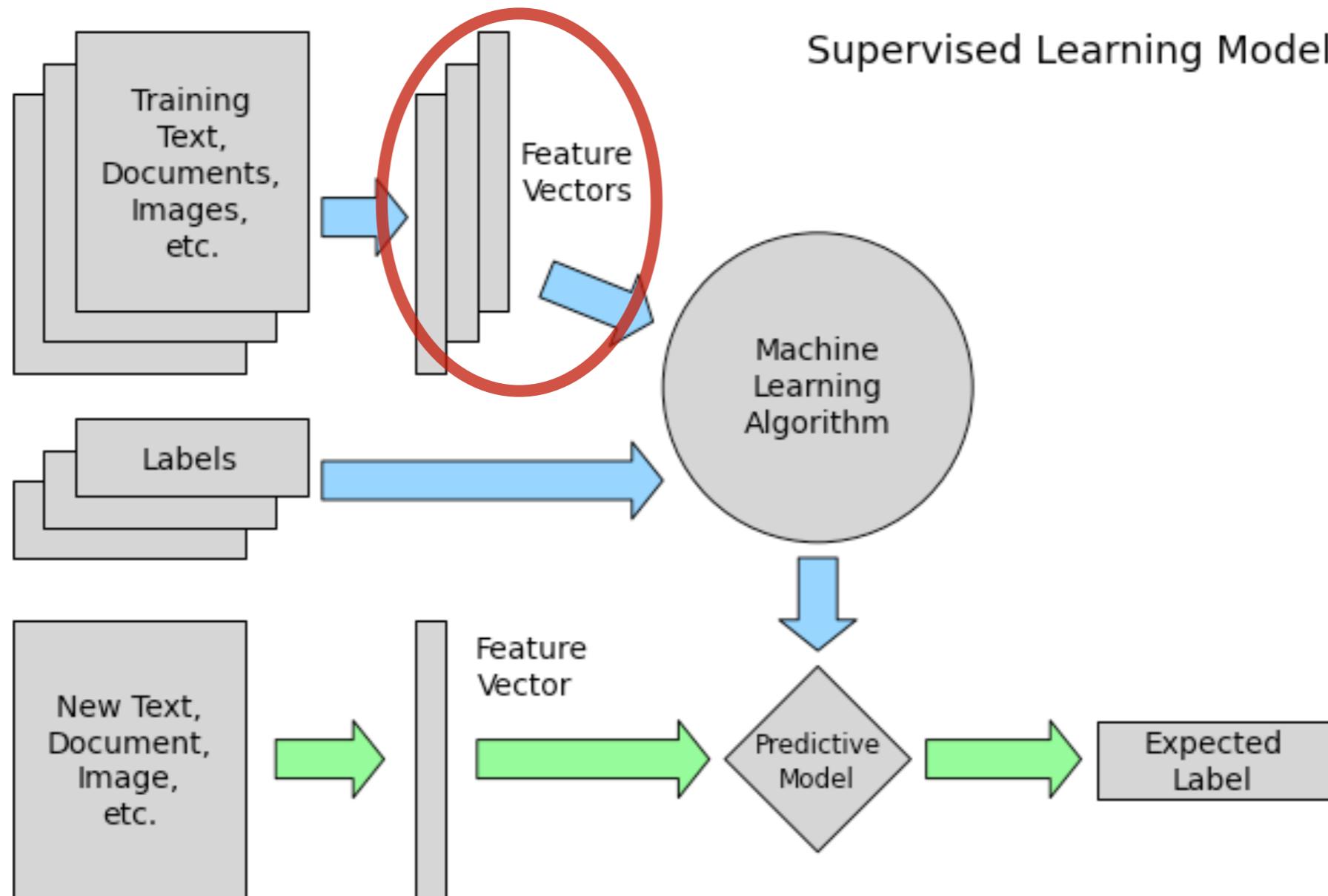
Supervised Learning



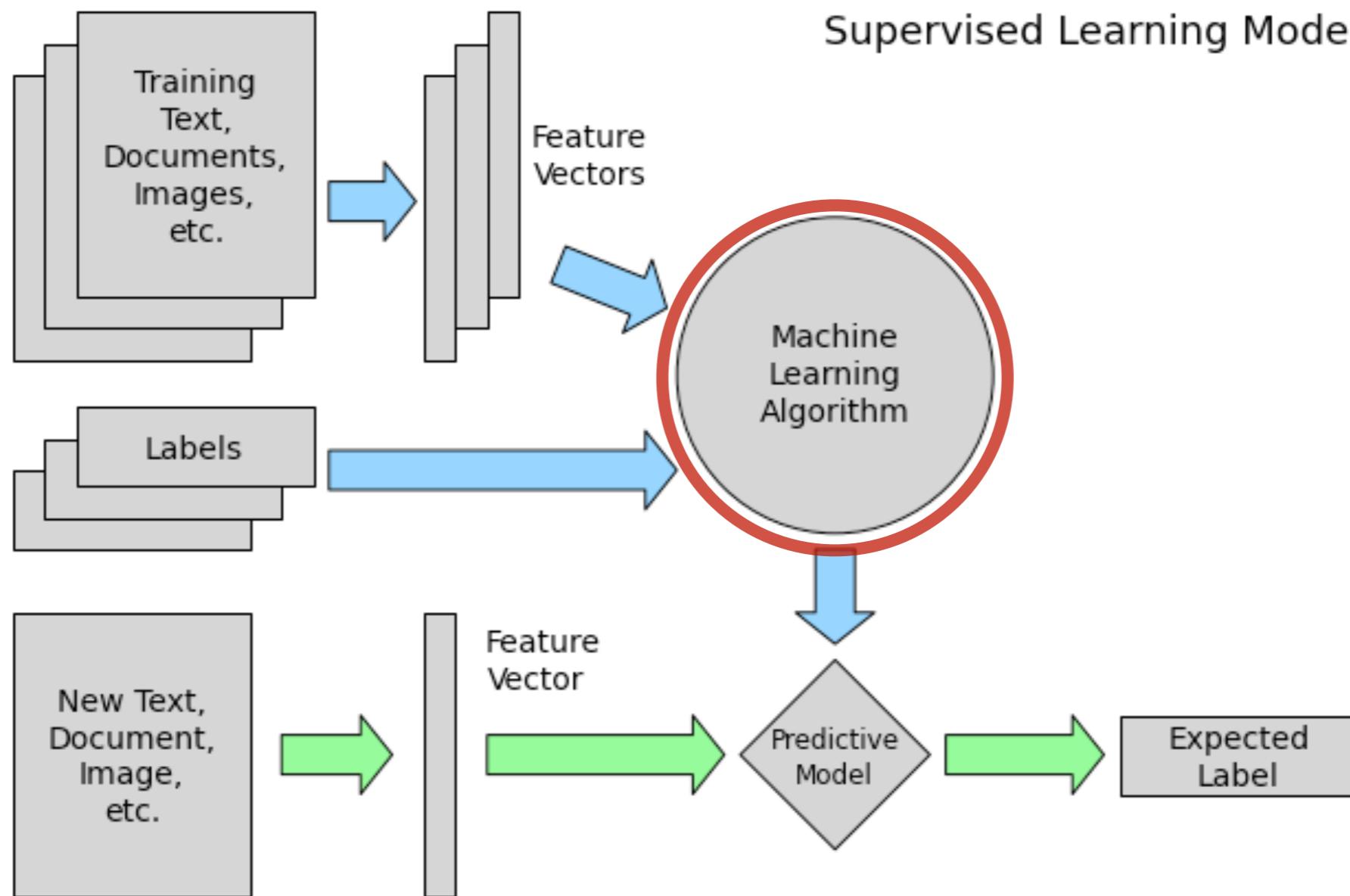
Supervised Learning



Supervised Learning



Supervised Learning



The Model

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC

vect = TfidfVectorizer(analyzer='char', ngram_range=(2, 3))
clf = SVC()
text_clf = Pipeline([
    ('vect', vect),
    ('clf', clf),
])
text_clf = text_clf.fit_transform(X_train, y_train)
```

The Model

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC

vect = TfidfVectorizer(analyzer='char', ngram_range=(2, 3))
clf = SVC()
text_clf = Pipeline([
    ('vect', vect),
    ('clf', clf),
])
text_clf = text_clf.fit_transform(X_train, y_train)
```

The Model

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC

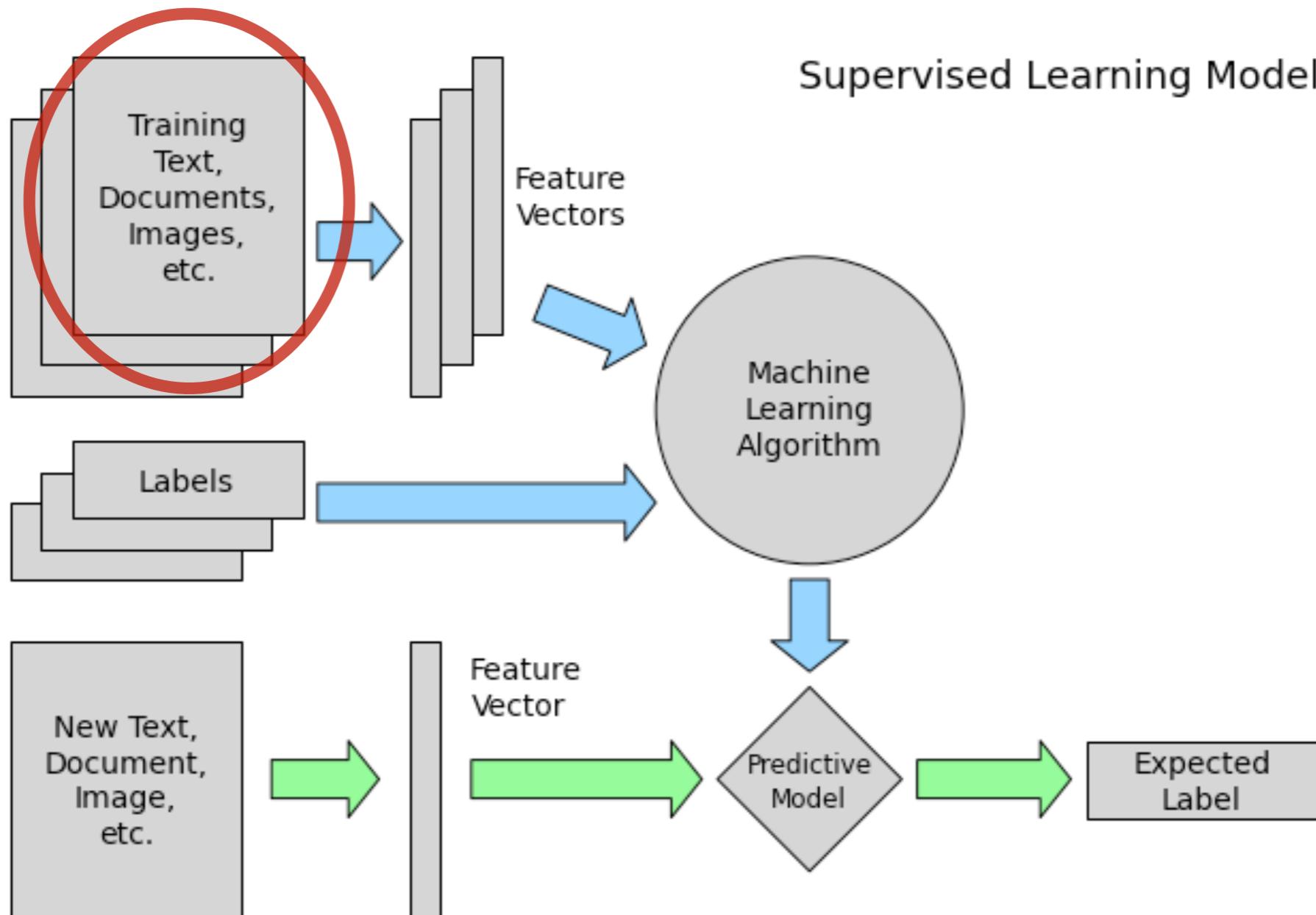
vect = TfidfVectorizer(analyzer='char', ngram_range=(2, 3))
clf = SVC()
text_clf = Pipeline([
    ('vect', vect),
    ('clf', clf),
])
text_clf = text_clf.fit_transform(X_train, y_train)
```

The Model

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC

vect = TfidfVectorizer(analyzer='char', ngram_range=(2, 3))
clf = SVC()
text_clf = Pipeline([
    ('vect', vect),
    ('clf', clf),
])
text_clf = text_clf.fit_transform(X_train, y_train)
```

Supervised Learning

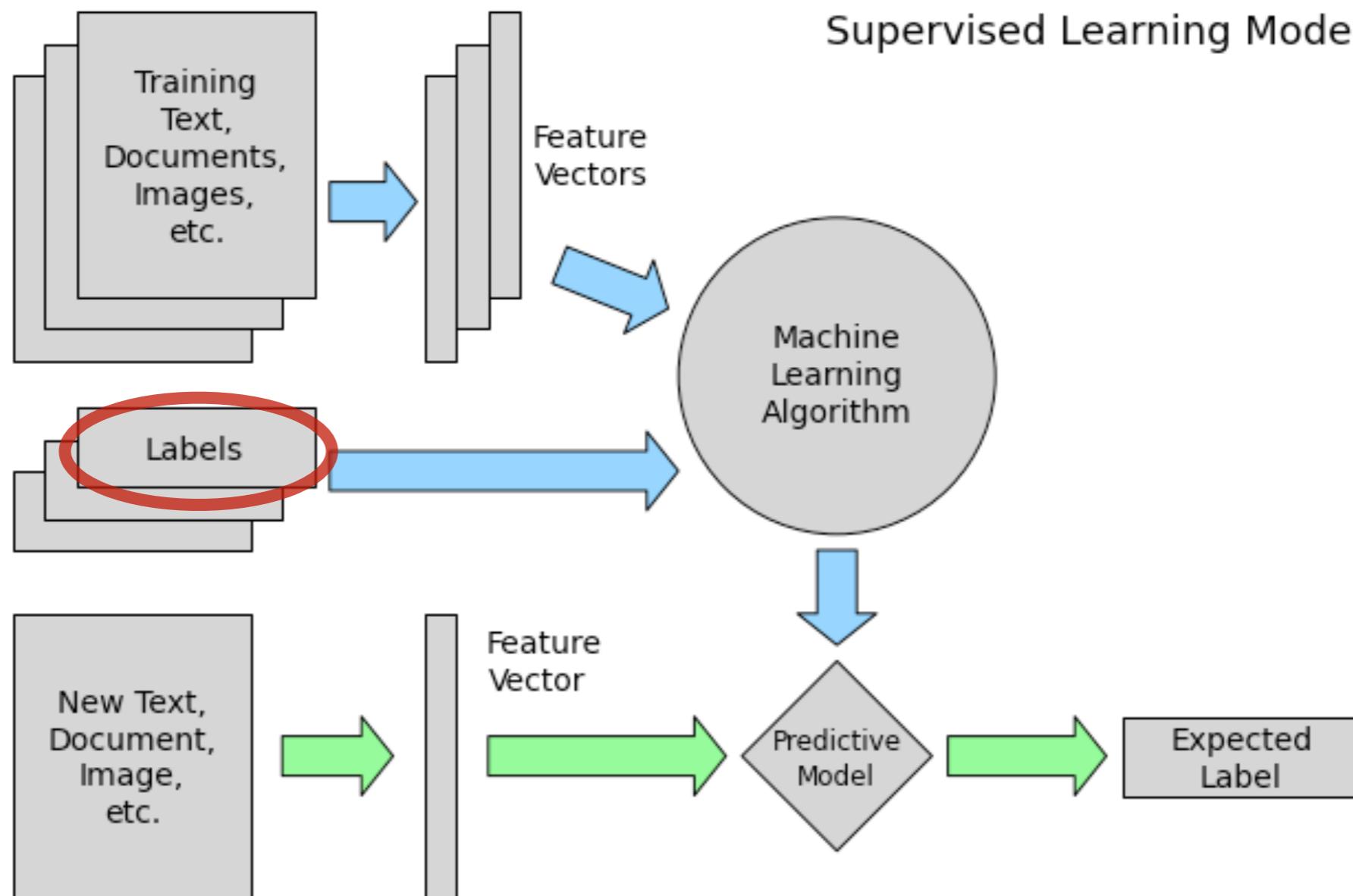


X_train

```
for x in X_train[:10]:  
    print ''.join(unicode(x, 'utf8')[:70].split())
```

Szöul Szöul (서울 특별시 Sǒul T'ǔkpyǒlsi, szoros átírásban: Szoul thukpjol Sean Connery Cecil B. DeMille Award (1996) | baftaawards Beste hov Nemecká demokratická republika Nemecká demokratická republika (NDR, h Plastic A plastic material is any of a wide range of synthetic or sem Phaistose ketas Phaistose ketas on põletatud savist ketas, mille leid Soustava SI Soustava SI (zkratka z francouzského Le Système Internati Баку Баку () је главни град Азербејџана. Налази се у јужном делу полу Gottlob Frege nume Friedrich Ludwig Gottlob Frege Belgrado Belgrado (Београд / Beograd em servo-croata ouça) é a capi Cálculo infinitesimal El cálculo infinitesimal o cálculo de infinites

Supervised Learning



y_train

y

```
['uk', 'cs', 'ko', 'es', 'war', 'de', 'de', 'pl', 'ar', 'cs']
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_train = le.fit_transform(y)
y_train
```

```
array([35,  3, 21, ...,  8, 18, 30])
```

Tuning

sklearn.svm.SVC

```
class sklearn.svm.SVC(C=1.0, kernel='rbf', degree=3, gamma=0.0, coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, random_state=None) ↴
```

C-Support Vector Classification.

The implementation is based on libsvm. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to datasets with more than a couple of 10000 samples.

The multiclass support is handled according to a one-vs-one scheme.

For details on the precise mathematical formulation of the provided kernel functions and how gamma, coef0 and degree affect each, see the corresponding section in the narrative documentation: [Kernel functions](#).

RandomizedSearchCV

```
from sklearn.grid_search import RandomizedSearchCV
from scipy.stats import uniform as sp_uniform

clf = SVC()
param_dist = {
    'C': sp_uniform(1, 4),
    'kernel': ['linear', 'rbf'],
}

random_search = RandomizedSearchCV(
    clf, param_distributions=param_dist, n_jobs=9)
random_search.fit(X_trans, y_train)
```

RandomizedSearchCV

```
random_search.best_score_
```

```
0.96647677939899623
```

```
random_search.best_params_
```

```
{'C': 3.3357996909950347, 'kernel': 'rbf'}
```

sklearn.metrics

- accuracy_score
- classification_report
- confusion_matrix

```
y_pred = clf.predict(X_test)
```

accuracy_score

```
from sklearn.metrics import accuracy_score  
ac = accuracy_score(y_true, y_pred)  
ac
```

0.95636036036036032

classification_report

```
from sklearn.metrics import classification_report  
cr = classification_report(y_true, y_pred, target_names=target_names)  
cr
```

	precision	recall	f1-score	support
af	0.99	1.00	1.00	1500
als	0.98	0.97	0.98	1500
az	1.00	1.00	1.00	1500
bar	0.97	0.98	0.98	1500
bcl	0.95	0.96	0.95	1500
bn	1.00	1.00	1.00	1500
bs	0.42	0.62	0.50	1500
vls	1.00	0.99	1.00	1500
war	0.98	0.96	0.97	1500
zh	1.00	1.00	1.00	1500
avg / total	0.96	0.96	0.95	111000

classification_report

```
from sklearn.metrics import classification_report  
cr = classification_report(y_true, y_pred, target_names=target_names)  
cr
```

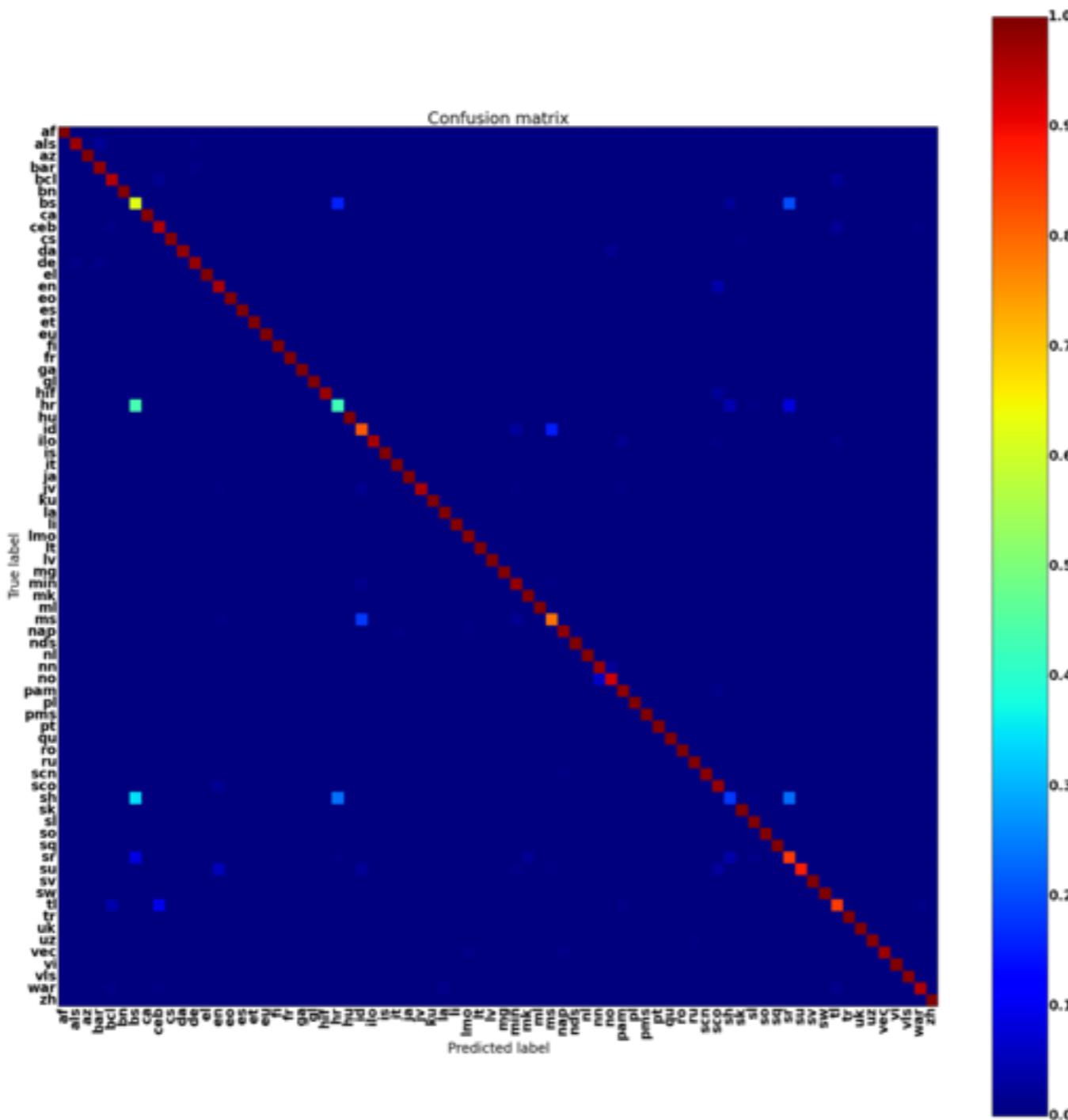
	precision	recall	f1-score	support
af	0.99	1.00	1.00	1500
als	0.98	0.97	0.98	1500
az	1.00	1.00	1.00	1500
bar	0.97	0.98	0.98	1500
bcl	0.95	0.96	0.95	1500
bn	1.00	1.00	1.00	1500
bs	0.42	0.62	0.50	1500
vls	1.00	0.99	1.00	1500
war	0.98	0.96	0.97	1500
zh	1.00	1.00	1.00	1500
avg / total	0.96	0.96	0.95	111000

I'm Confused

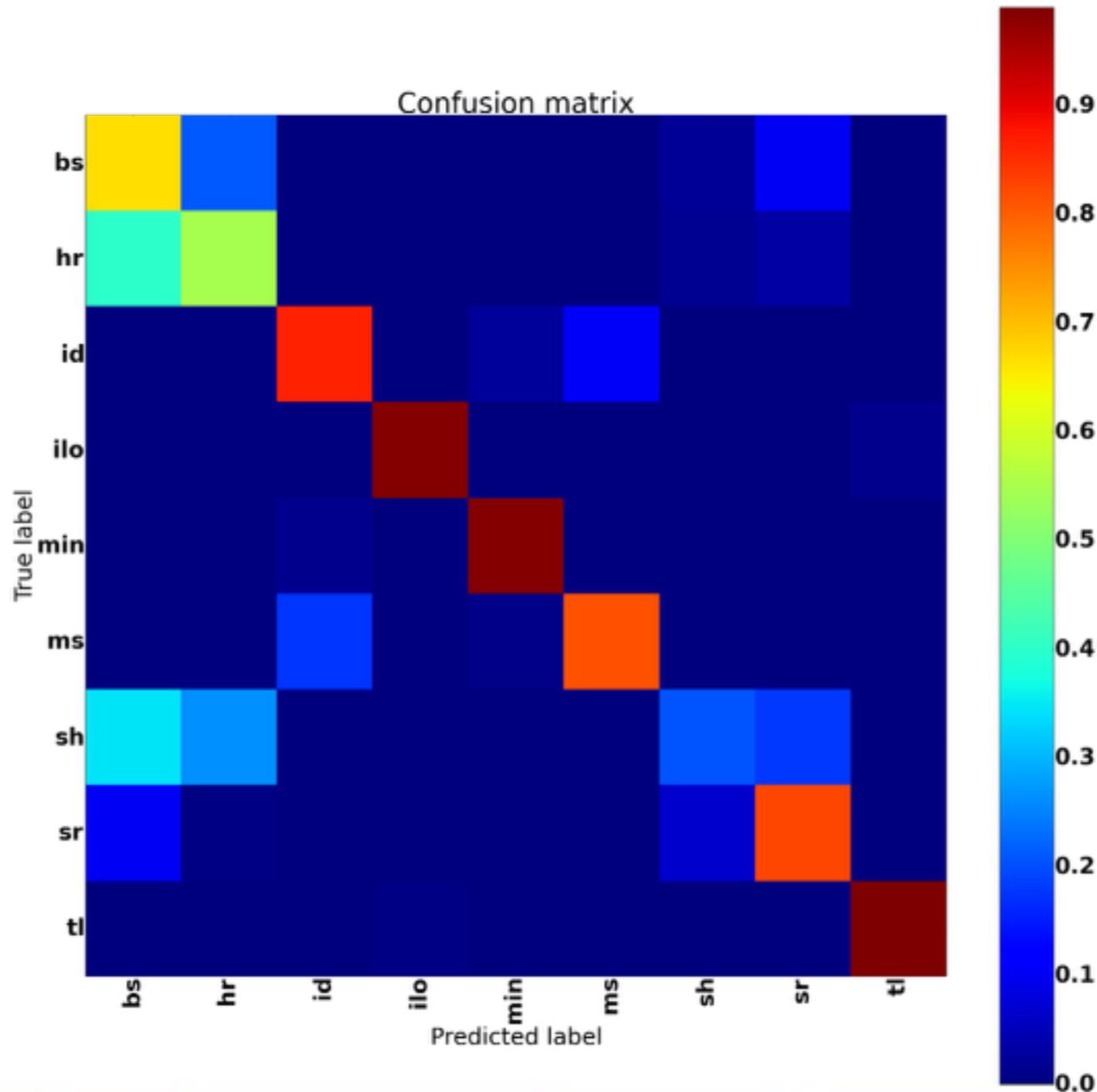
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_true, y_pred)
cm

array([[1497,      0,      0, ...,      0,      0,      0],
       [      0, 1461,      0, ...,      0,      0,      0],
       [      0,      0, 1499, ...,      0,      0,      0],
       ...,
       [      1,      0,      0, ..., 1487,      0,      0],
       [      0,      0,      0, ...,      0, 1439,      0],
       [      0,      0,      0, ...,      0,      0, 1499]])
```

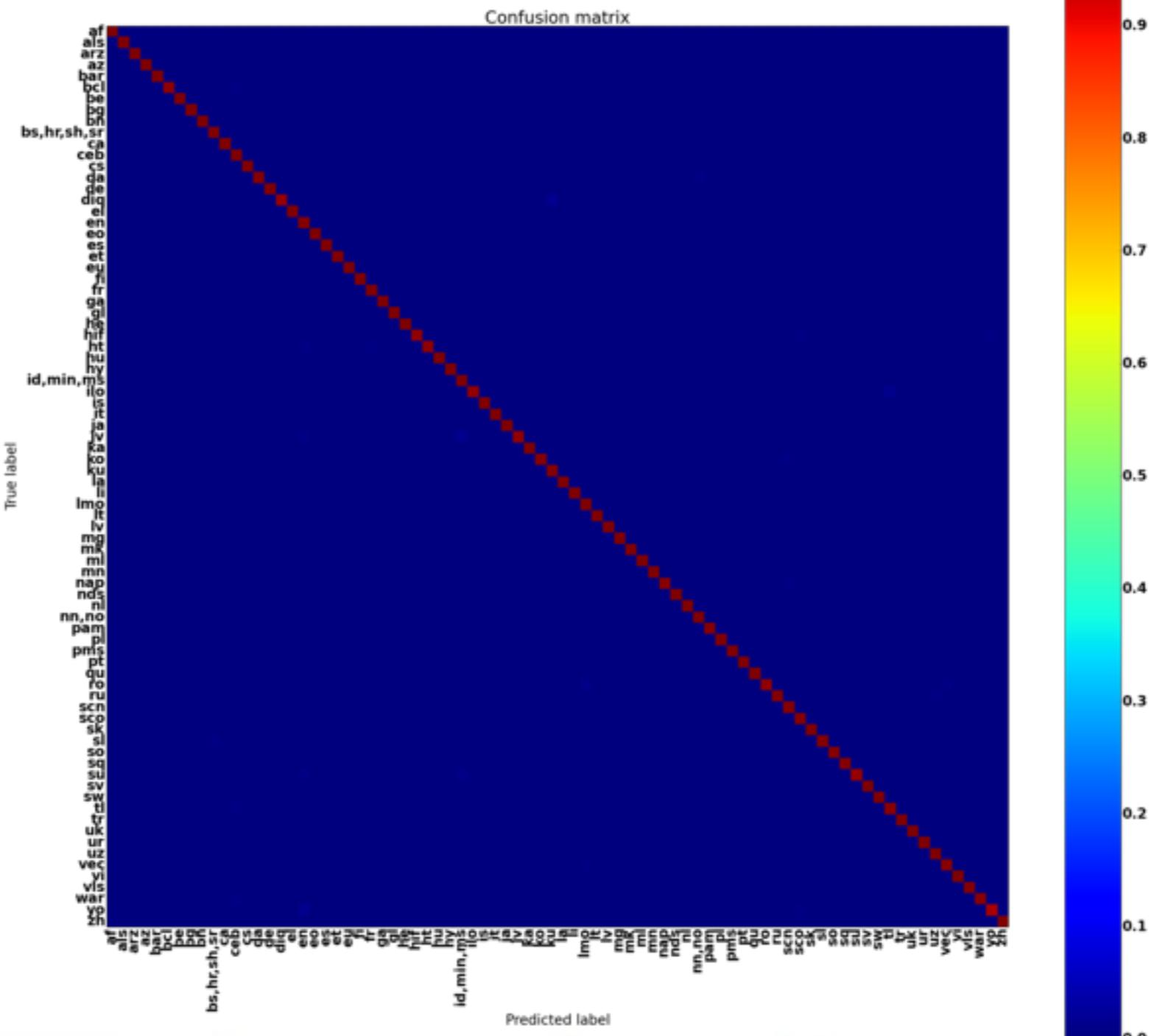
I'm Confused



I'm Confused



INterpret



My Info

Twitter: @beckerfuffle

Blog: beckerfuffle.com

Github: github.com/mdbecker

Work with me: <http://aweber.jobs>

