Software and Engineering Efforts at NYU Center for Data Science

Andreas Mueller

About me

- Research Engineer @ NYU since 2014
- Spend most on my time on scikit-learn

About the NYU Center for Data Science

Master of Data Science

GORDON AND BETTY MOORE FOUNDATION

ALFRED P. SLOAN FOUNDATION

Moore-Sloan
Data Science
Environment

Master of Data Science

Class of 2015

Class of 2016

Master of Data Science

Class of 2015

Class of 2016

Class of 2017

66 Students

Statistics

Inference

Big Data Technologies

Python

Machine Learning

Data management

Probabilistic Modeling

Statistics

Inference

Big Data Technologies

Python

Machine Learning

Data management

Probabilistic Modeling

Urban Science *Business* *Economics*

Biology *Politics* *Sociology*

Neuroscience *Psychology*

Moore-Sloan
Data Science Environment

UNIVERSITY of WASHINGTON
eScience Institute

BIDS
BERKELEY INSTITUTE
FOR DATA SCIENCE

NYU
Center For
Data Science

Moore-Sloan
Data Science Environment

UNIVERSITY of WASHINGTON
eScience Institute

BIDS
BERKELEY INSTITUTE
FOR DATA SCIENCE

NYU
Center For
Data Science

Build bridges between methods and applications.

Moore-Sloan Data Science Environment

UNIVERSITY of WASHINGTON
eScience Institute

BIDS
BERKELEY INSTITUTE FOR DATA SCIENCE

NYU Center For Data Science

Build bridges between methods and applications.
Create career paths for data scientists.

Moore-Sloan
Data Science Environment

UNIVERSITY *of* WASHINGTON
eScience Institute

BIDS
BERKELEY INSTITUTE FOR DATA SCIENCE

NYU Center For Data Science

Build bridges between methods and applications.
Create career paths for data scientists.
Provide an ecosystem of open tools for data driven research.

MSDSE @ NYU

Fellows

Research Engineers

Ethnography

Connection to Library Sciences

- New position 50/50 CDS and Library
- Data provenance
- Metadata conventions
- Data handling, storage and versioning

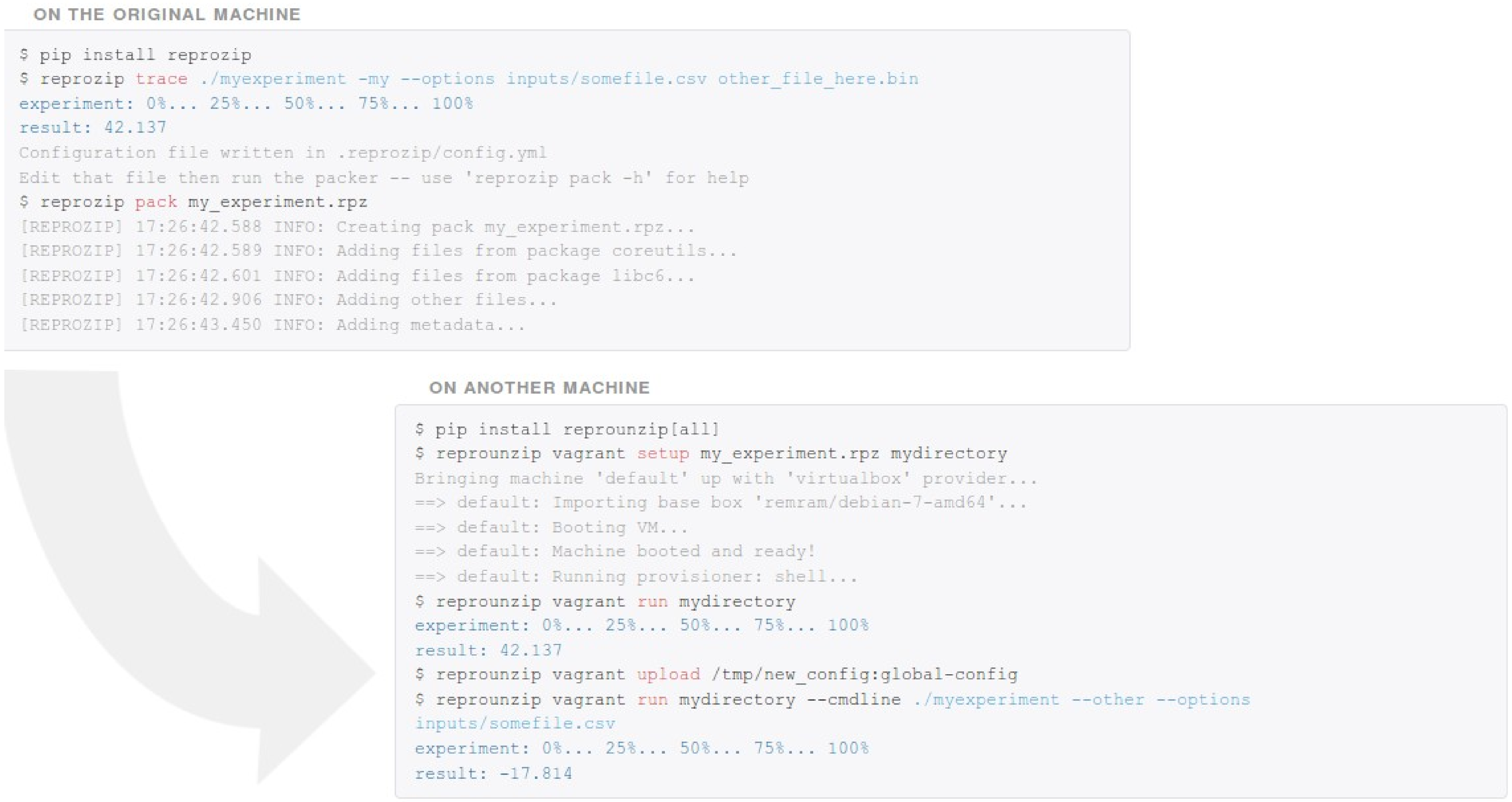
Reproducibility and Open Science Working Group

Juliana Freire

Kyle Cranmer

Remi Rampin

ReproZip (Remi Rampin)



ON THE ORIGINAL MACHINE

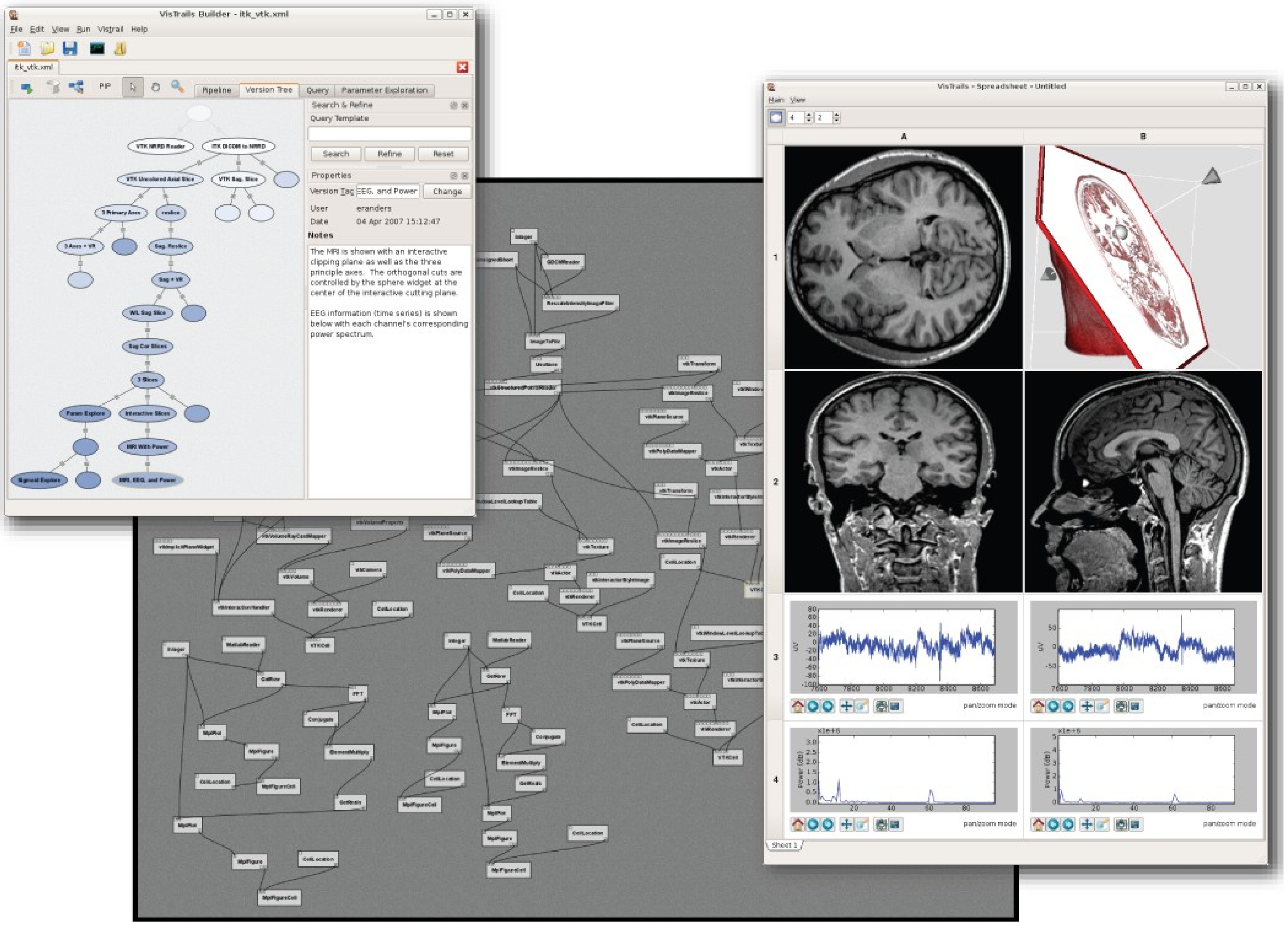
```
$ pip install reprozip
$ reprozip trace ./myexperiment -my --options inputs/somefile.csv other_file_here.bin
experiment: 0%... 25%... 50%... 75%... 100%
result: 42.137
Configuration file written in .reprozip/config.yml
Edit that file then run the packer -- use 'reprozip pack -h' for help
$ reprozip pack my_experiment.rpz
[REPROZIP] 17:26:42.588 INFO: Creating pack my_experiment.rpz...
[REPROZIP] 17:26:42.589 INFO: Adding files from package coreutils...
[REPROZIP] 17:26:42.601 INFO: Adding files from package libc6...
[REPROZIP] 17:26:42.906 INFO: Adding other files...
[REPROZIP] 17:26:43.450 INFO: Adding metadata...
```

ON ANOTHER MACHINE

```
$ pip install reprounzip[all]
$ reprounzip vagrant setup my_experiment.rpz mydirectory
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'remram/debian-7-amd64'...
==> default: Booting VM...
==> default: Machine booted and ready!
==> default: Running provisioner: shell...
$ reprounzip vagrant run mydirectory
experiment: 0%... 25%... 50%... 75%... 100%
result: 42.137
$ reprounzip vagrant upload /tmp/new_config:global-config
$ reprounzip vagrant run mydirectory --cmdline ./myexperiment --other --options
inputs/somefile.csv
experiment: 0%... 25%... 50%... 75%... 100%
result: -17.814
```

VisTrails (Remi, Juliana, Claudio)

Workflow and Provenance for Visualization and Data Exploration



Version Tag EEG, and Power

User eranders

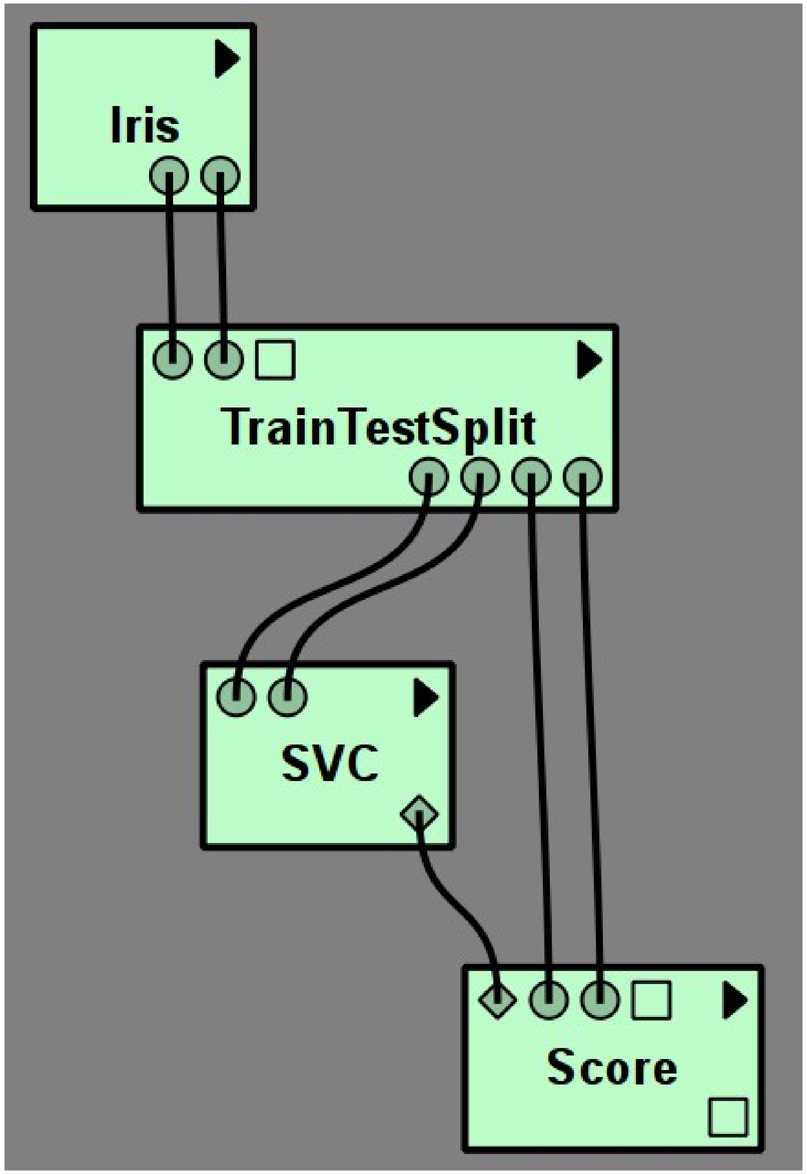
Date 04 Apr 2007 15:12:47

Notes

The MRI is shown with an interactive clipping plane as well as the three principle axes. The orthogonal cuts are controlled by the sphere widget at the center of the interactive cutting plane.

EEG information (time series) is shown below with each channel's corresponding power spectrum.

VisTrails sklearn plugin



Software Working Group

Goals:

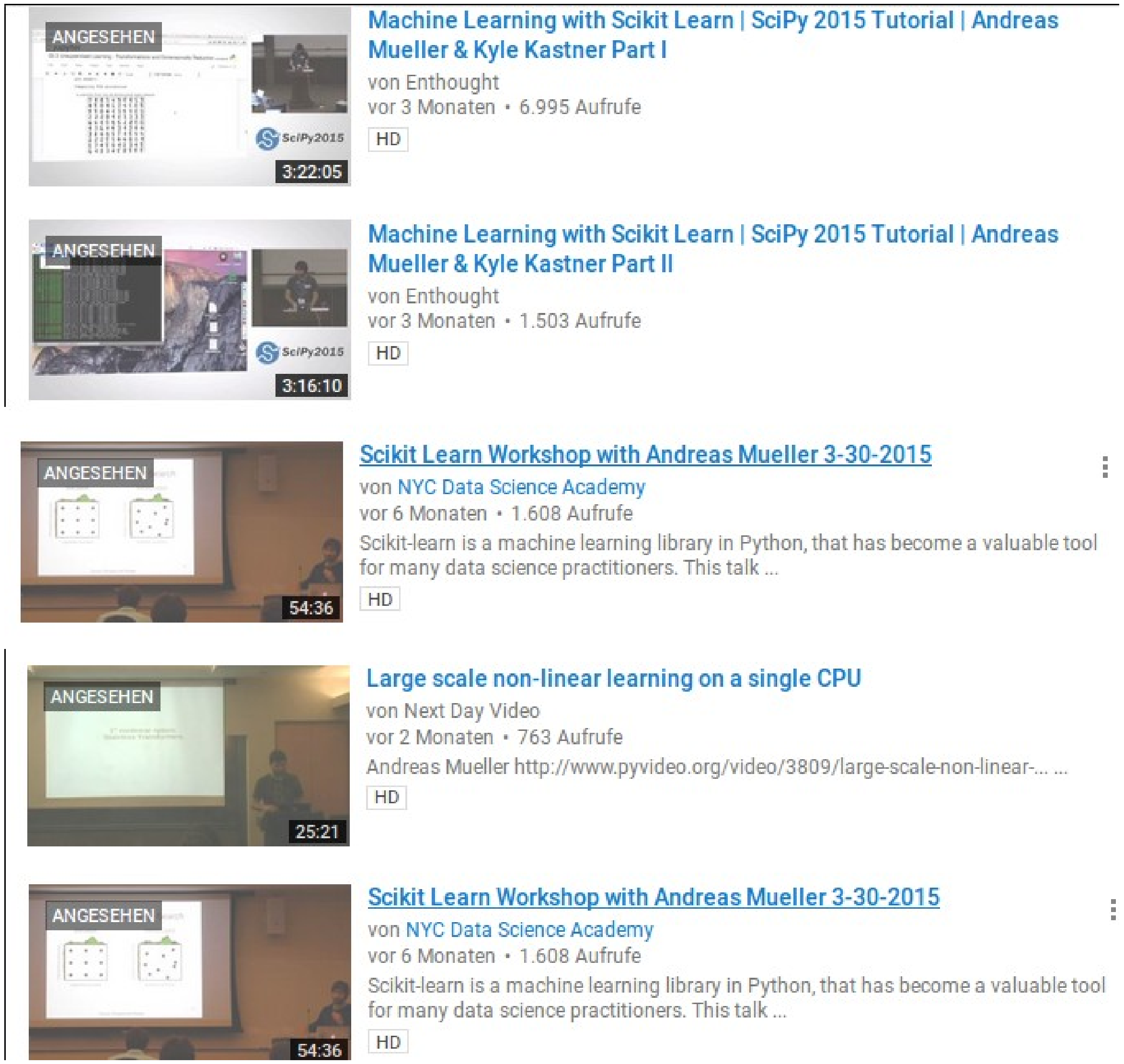
- Outreach to “Domain Sciences” to help with data sciences
- Education about software development
- Build and Support infrastructure for science
- Build community around software development at NYU

Software Working Group

- Chair: Claudio Silva
- Research Engineers:
  - Me (for the first year)
  - Stefan Karpinski (started this summer)
  - Heiko Mueller (starting now)
  - Two more to come
- Six master students started this summer.

Current Efforts

Current Efforts



Machine Learning with Scikit Learn | SciPy 2015 Tutorial | Andreas Mueller & Kyle Kastner Part I

von Enthought
vor 3 Monaten • 6.995 Aufrufe
HD

Machine Learning with Scikit Learn | SciPy 2015 Tutorial | Andreas Mueller & Kyle Kastner Part II

von Enthought
vor 3 Monaten • 1.503 Aufrufe
HD

Scikit Learn Workshop with Andreas Mueller 3-30-2015

von NYC Data Science Academy
vor 6 Monaten • 1.608 Aufrufe
Scikit-learn is a machine learning library in Python, that has become a valuable tool for many data science practitioners. This talk ...
HD

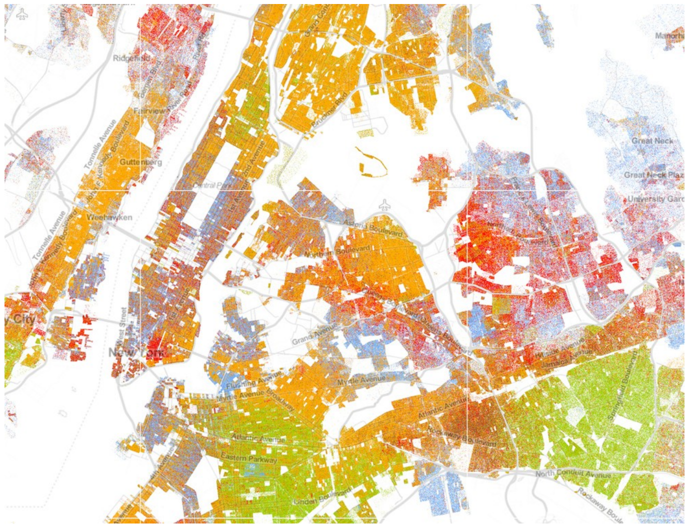
Large scale non-linear learning on a single CPU

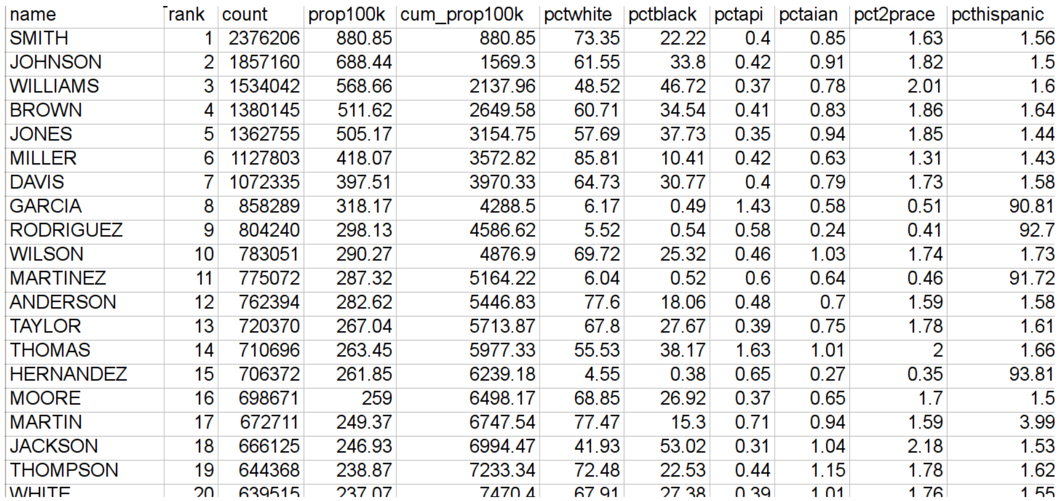
von Next Day Video
vor 2 Monaten • 763 Aufrufe
Andreas Mueller http://www.pyvideo.org/video/3809/large-scale-non-linear-... ...
HD

Scikit Learn Workshop with Andreas Mueller 3-30-2015

von NYC Data Science Academy
vor 6 Monaten • 1.608 Aufrufe
Scikit-learn is a machine learning library in Python, that has become a valuable tool for many data science practitioners. This talk ...
HD

Voter Ethnicity Prediction (Tian Wang)

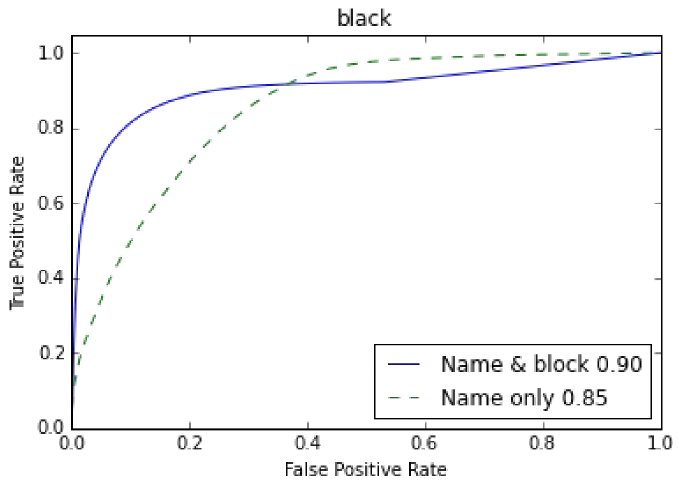




name	rank	count	prop100k	cum_prop100k	pctwhite	pctblack	pctapi	pctaian	pct2prace	pcthispanic
SMITH	1	2376206	880.85	880.85	73.35	22.22	0.4	0.85	1.63	1.56
JOHNSON	2	1857160	688.44	1569.3	61.55	33.8	0.42	0.91	1.82	1.5
WILLIAMS	3	1534042	568.66	2137.96	48.52	46.72	0.37	0.78	2.01	1.6
BROWN	4	1380145	511.62	2649.58	60.71	34.54	0.41	0.83	1.86	1.64
JONES	5	1362755	505.17	3154.75	57.69	37.73	0.35	0.94	1.85	1.44
MILLER	6	1127803	418.07	3572.82	85.81	10.41	0.42	0.63	1.31	1.43
DAVIS	7	1072335	397.51	3970.33	64.73	30.77	0.4	0.79	1.73	1.58
GARCIA	8	858289	318.17	4288.5	6.17	0.49	1.43	0.58	0.51	90.81
RODRIGUEZ	9	804240	298.13	4586.62	5.52	0.54	0.58	0.24	0.41	92.7
WILSON	10	783051	290.27	4876.9	69.72	25.32	0.46	1.03	1.74	1.73
MARTINEZ	11	775072	287.32	5164.22	6.04	0.52	0.6	0.64	0.46	91.72
ANDERSON	12	762394	282.62	5446.83	77.6	18.06	0.48	0.7	1.59	1.58
TAYLOR	13	720370	267.04	5713.87	67.8	27.67	0.39	0.75	1.78	1.61
THOMAS	14	710696	263.45	5977.33	55.53	38.17	1.63	1.01	2	1.66
HERNANDEZ	15	706372	261.85	6239.18	4.55	0.38	0.65	0.27	0.35	93.81
MOORE	16	698671	259	6498.17	68.85	26.92	0.37	0.65	1.7	1.5
MARTIN	17	672711	249.37	6747.54	77.47	15.3	0.71	0.94	1.59	3.99
JACKSON	18	666125	246.93	6994.47	41.93	53.02	0.31	1.04	2.18	1.53
THOMPSON	19	644368	238.87	7233.34	72.48	22.53	0.44	1.15	1.78	1.62
WHITE	20	639515	237.07	7470.4	67.91	27.38	0.39	1.01	1.76	1.55

Census block ethnical distribution x Last name ethnical distribution

~ Voter ethnicity



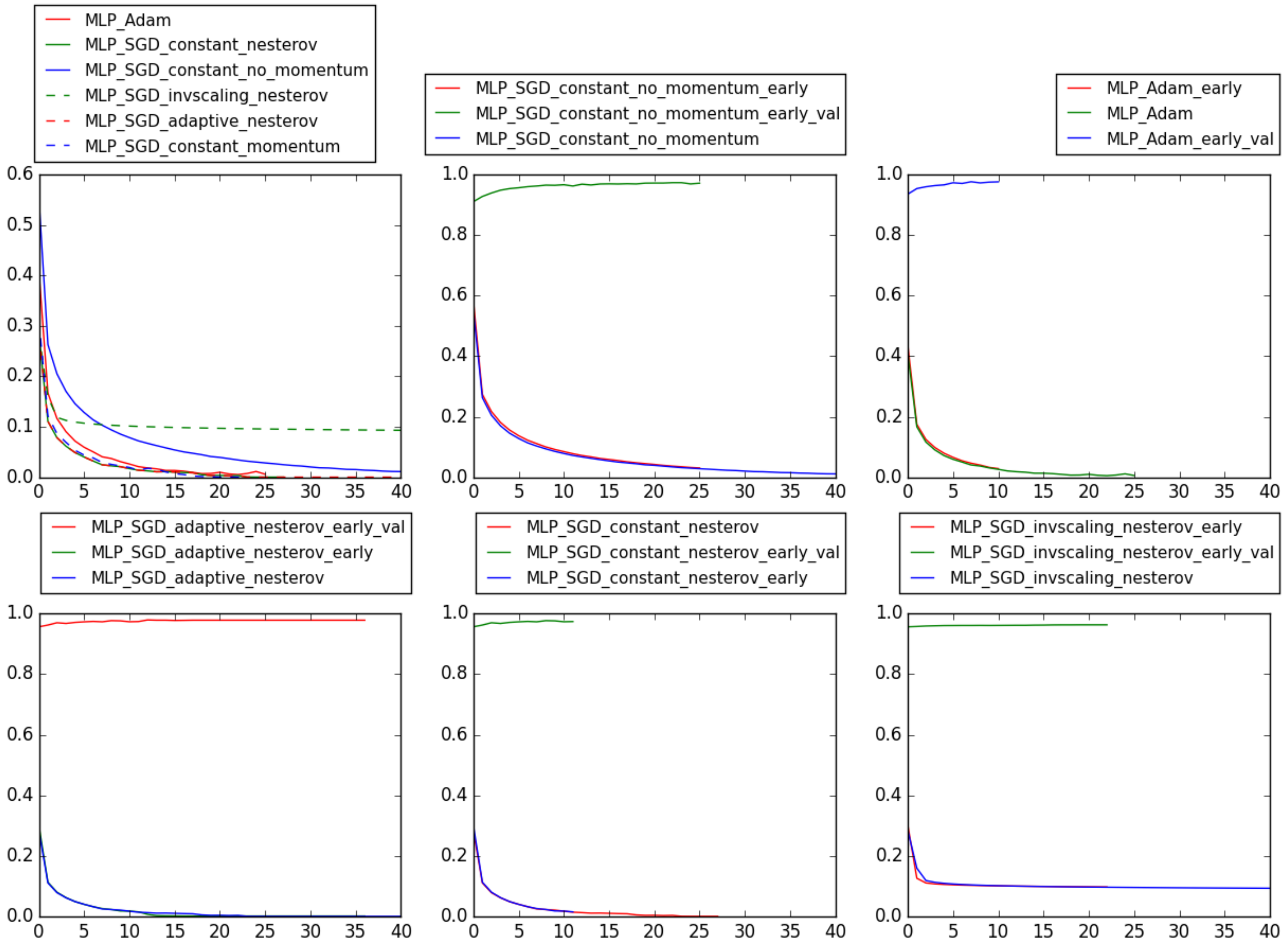
KNN Imputation (Tian Wang)

$$\begin{pmatrix} 6 & 3 & 7 & 3 & 6 \\ 7 & \boxed{?} & 7 & 2 & 5 \\ 0 & 6 & \boxed{?} & 4 & 6 \\ 1 & 6 & 3 & 3 & 8 \end{pmatrix}$$

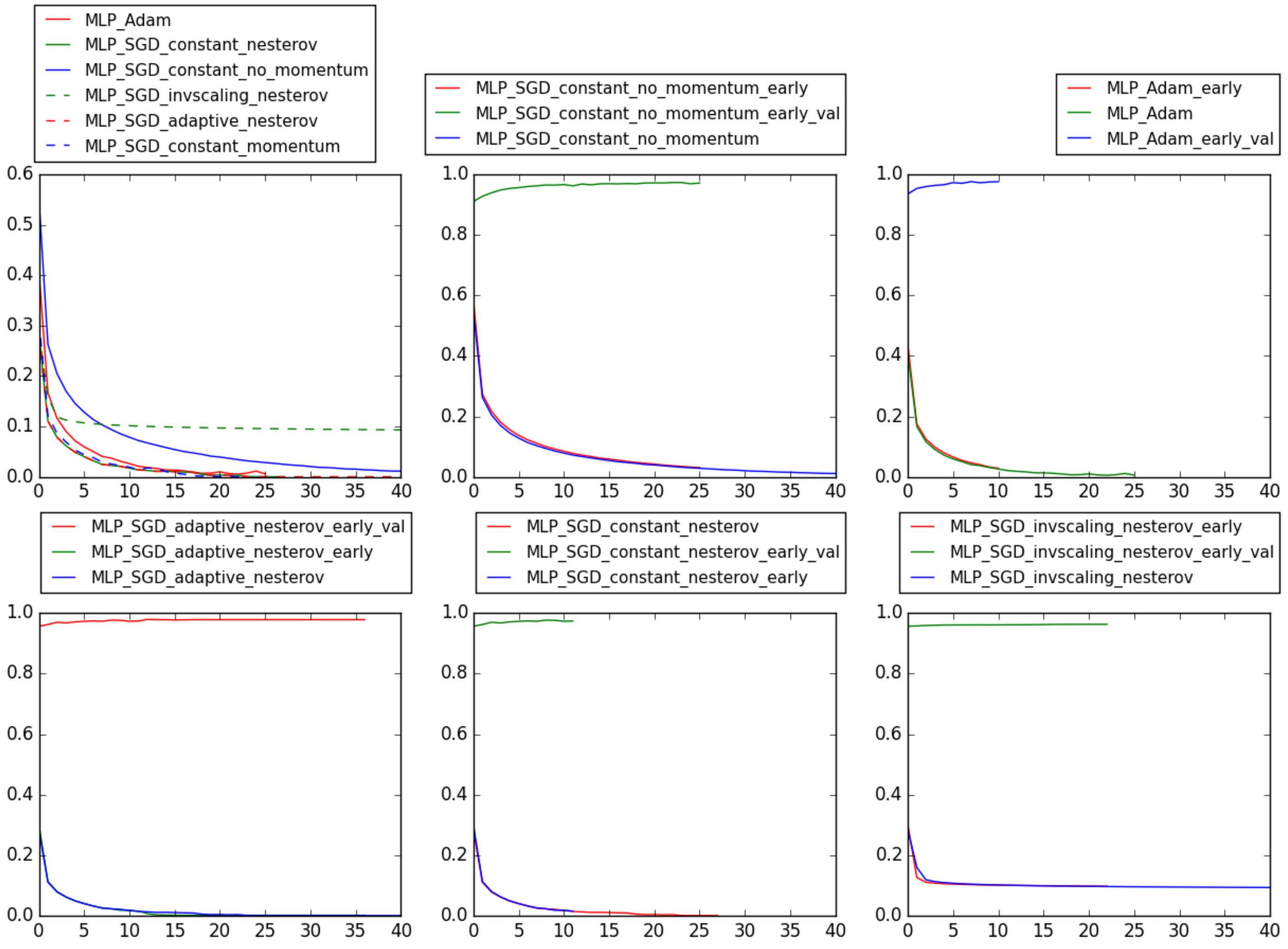
KNN Imputation (Tian Wang)

$$\begin{tikzcd}
6 & 3 \arrow[d, bend left] & 7 & 3 & 6 \\
7 & \boxed{?} & 7 & 2 & 5 \\
0 & 6 & \boxed{?} & 4 & 6 \\
1 & 6 & 3 \arrow[u, bend right] & 3 & 8
\end{tikzcd}$$

Neural Network (Jiyuan Qian)



Neural Network (Jiyuan Qian)

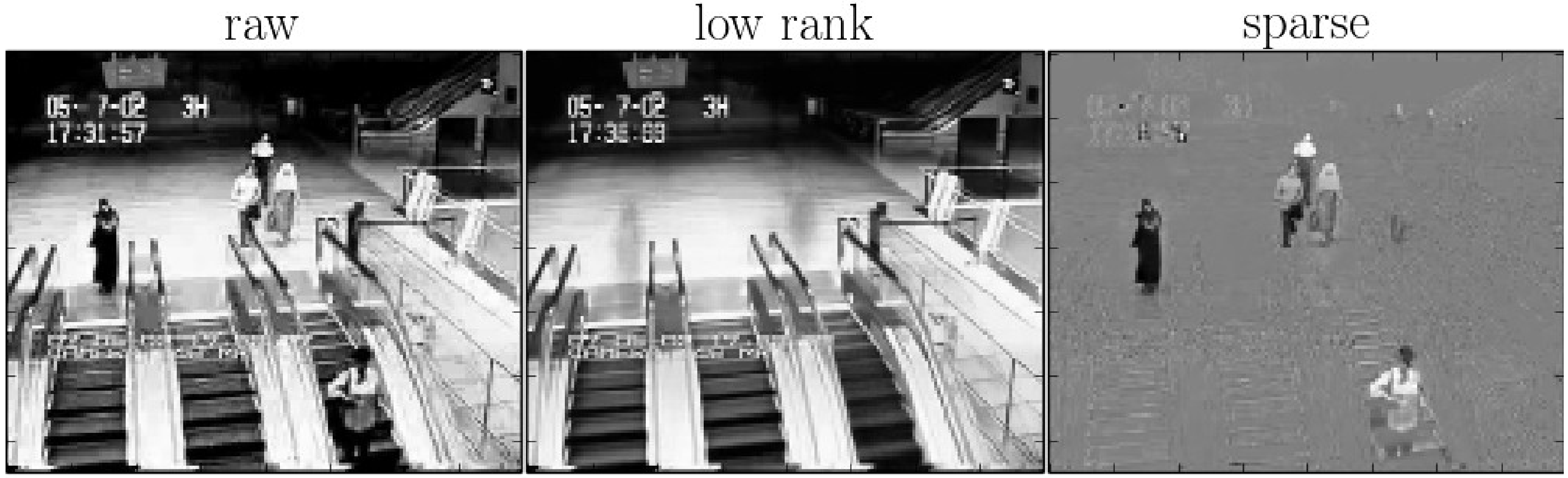


Merged on Friday!

Robust PCA (Jiyuan Quian)

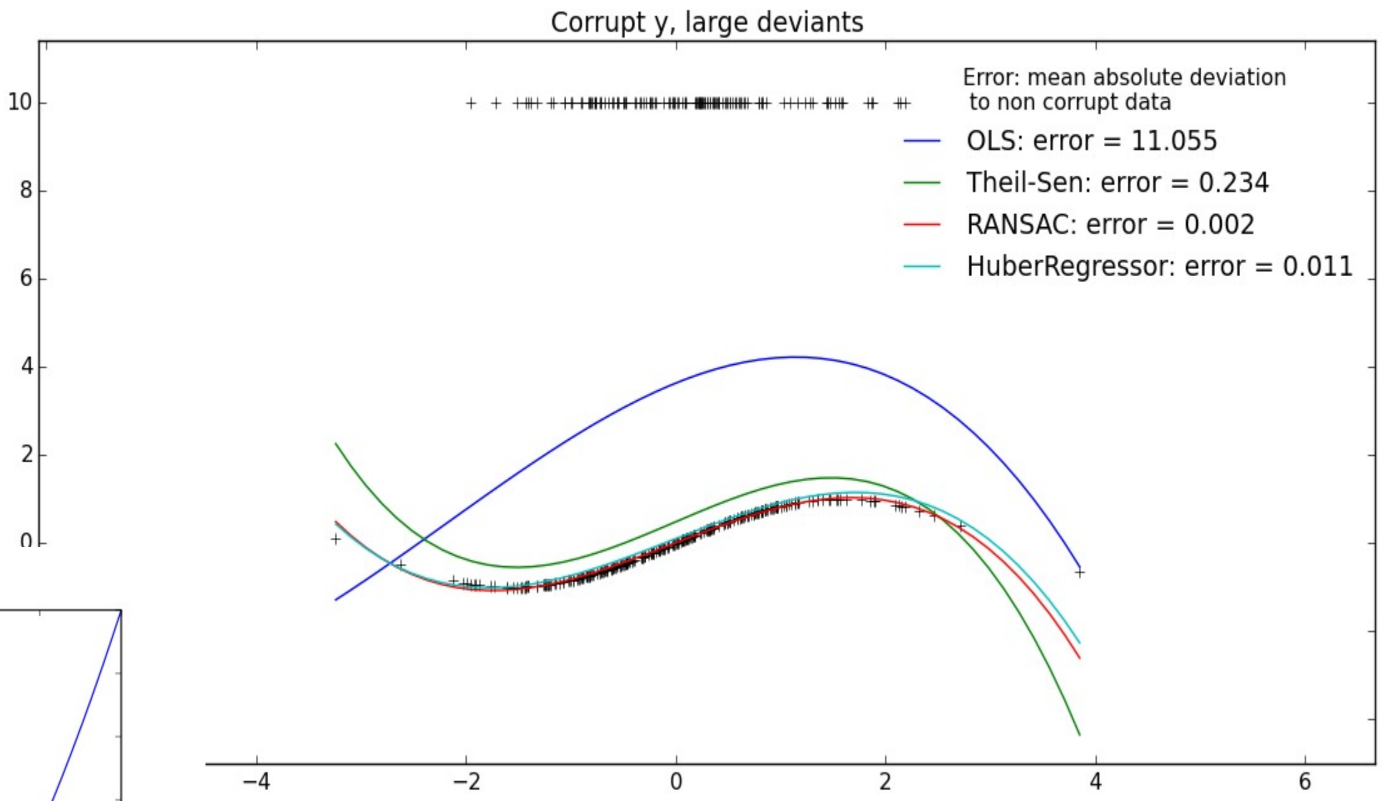
$$X = L + S$$

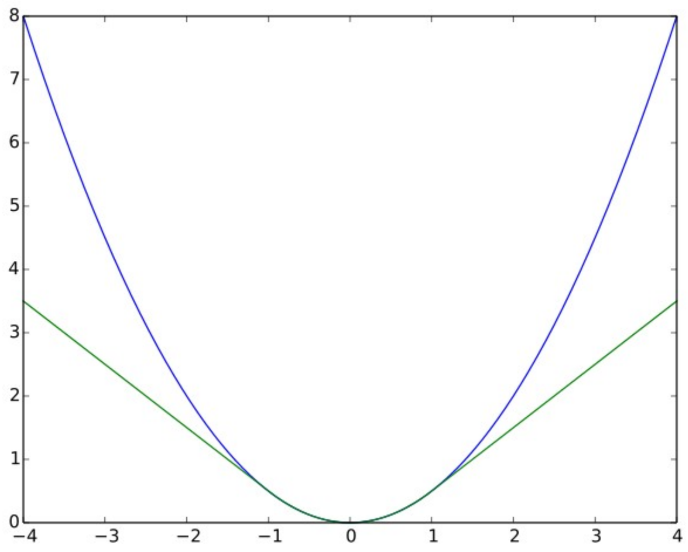
L low rank, S sparse



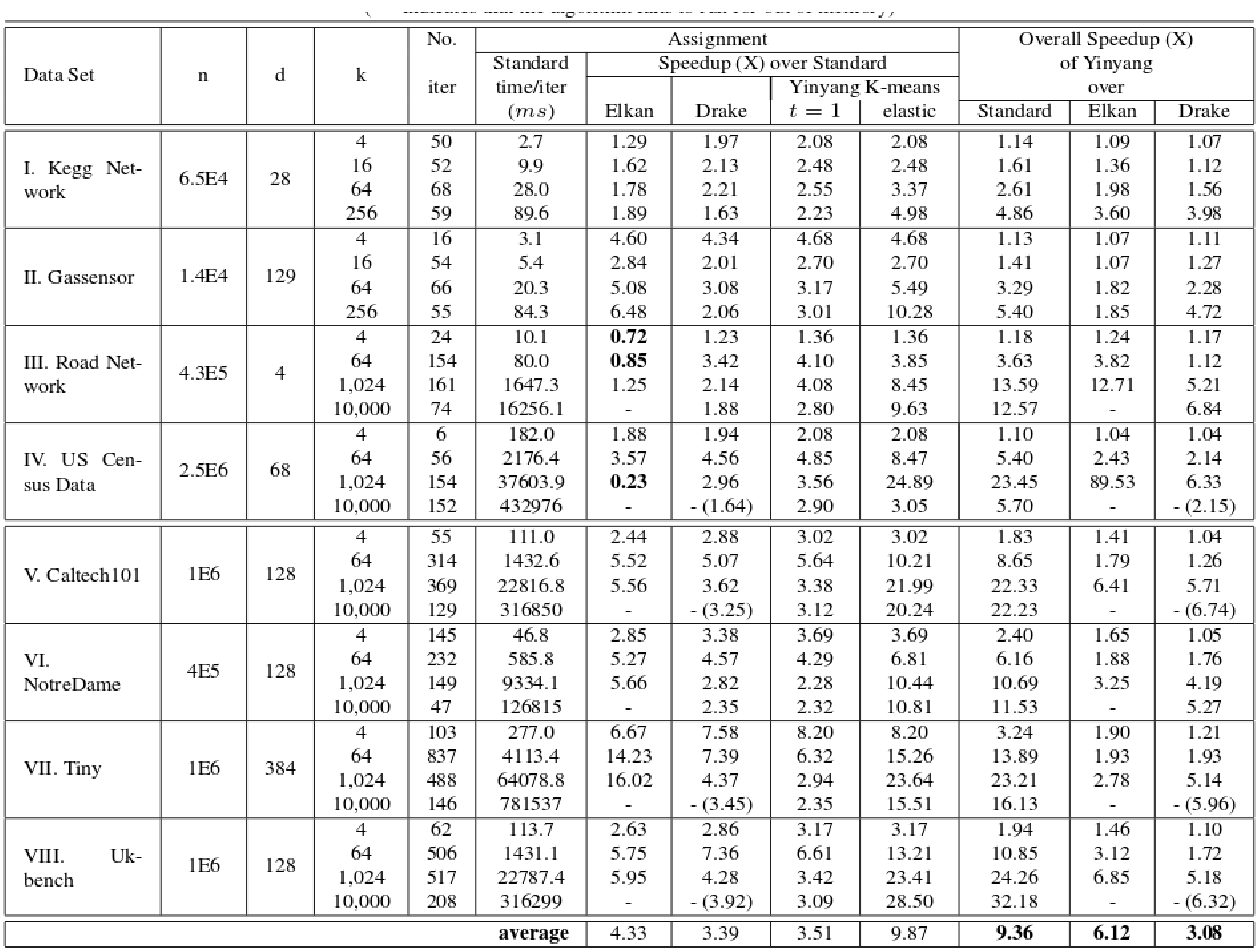
Solved via Trimmed Grassmann Average (TGA) by Hauberg et al

Huber Regression (Manoj Kumar)





Ying-Yan Kmeans (Manoj Kumar)



Data Set	n	d	k	No. iter	Assignment					Overall Speedup (X) of Yinyang over		
					Standard time/iter (ms)	Speedup (X) over Standard		Yinyang K-means				
						Elkan	Drake	$t = 1$	elastic	Standard	Elkan	Drake
I. Kegg Network	6.5E4	28	4	50	2.7	1.29	1.97	2.08	2.08	1.14	1.09	1.07
			16	52	9.9	1.62	2.13	2.48	2.48	1.61	1.36	1.12
			64	68	28.0	1.78	2.21	2.55	3.37	2.61	1.98	1.56
			256	59	89.6	1.89	1.63	2.23	4.98	4.86	3.60	3.98
II. Gassensor	1.4E4	129	4	16	3.1	4.60	4.34	4.68	4.68	1.13	1.07	1.11
			16	54	5.4	2.84	2.01	2.70	2.70	1.41	1.07	1.27
			64	66	20.3	5.08	3.08	3.17	5.49	3.29	1.82	2.28
			256	55	84.3	6.48	2.06	3.01	10.28	5.40	1.85	4.72
III. Road Network	4.3E5	4	4	24	10.1	**0.72**	1.23	1.36	1.36	1.18	1.24	1.17
			64	154	80.0	**0.85**	3.42	4.10	3.85	3.63	3.82	1.12
			1,024	161	1647.3	1.25	2.14	4.08	8.45	13.59	12.71	5.21
			10,000	74	16256.1	-	1.88	2.80	9.63	12.57	-	6.84
IV. US Census Data	2.5E6	68	4	6	182.0	1.88	1.94	2.08	2.08	1.10	1.04	1.04
			64	56	2176.4	3.57	4.56	4.85	8.47	5.40	2.43	2.14
			1,024	154	37603.9	**0.23**	2.96	3.56	24.89	23.45	89.53	6.33
			10,000	152	432976	-	- (1.64)	2.90	3.05	5.70	-	- (2.15)
V. Caltech101	1E6	128	4	55	111.0	2.44	2.88	3.02	3.02	1.83	1.41	1.04
			64	314	1432.6	5.52	5.07	5.64	10.21	8.65	1.79	1.26
			1,024	369	22816.8	5.56	3.62	3.38	21.99	22.33	6.41	5.71
			10,000	129	316850	-	- (3.25)	3.12	20.24	22.23	-	- (6.74)
VI. NotreDame	4E5	128	4	145	46.8	2.85	3.38	3.69	3.69	2.40	1.65	1.05
			64	232	585.8	5.27	4.57	4.29	6.81	6.16	1.88	1.76
			1,024	149	9334.1	5.66	2.82	2.28	10.44	10.69	3.25	4.19
			10,000	47	126815	-	2.35	2.32	10.81	11.53	-	5.27
VII. Tiny	1E6	384	4	103	277.0	6.67	7.58	8.20	8.20	3.24	1.90	1.21
			64	837	4113.4	14.23	7.39	6.32	15.26	13.89	1.93	1.93
			1,024	488	64078.8	16.02	4.37	2.94	23.64	23.21	2.78	5.14
			10,000	146	781537	-	- (3.45)	2.35	15.51	16.13	-	- (5.96)
VIII. Ukbench	1E6	128	4	62	113.7	2.63	2.86	3.17	3.17	1.94	1.46	1.10
			64	506	1431.1	5.75	7.36	6.61	13.21	10.85	3.12	1.72
			1,024	517	22787.4	5.95	4.28	3.42	23.41	24.26	6.85	5.18
			10,000	208	316299	-	- (3.92)	3.09	28.50	32.18	-	- (6.32)
					average	4.33	3.39	3.51	9.87	**9.36**	**6.12**	**3.08**

Other scikit-learn contributions (Vighnesh Birodkar)

- Fixes to input validation in scikit-learn
- Fixes to K-Means clustering
- Fixes to preprocessing

Patsy-sklearn

```
y ~ x_1 + x_2 + x_3
log(y) ~ x_1 + x_2 + x_3
log(y) ~ x_1:x_2 + x_3
log(y) ~ x_1:x_2 + log(x_3)
```

Patsy-sklearn

```
y ~ x_1 + x_2 + x_3
log(y) ~ x_1 + x_2 + x_3
log(y) ~ x_1:x_2 + x_3
log(y) ~ x_1:x_2 + log(x_3)
```

```
model = PatsyModel(LogisticRegression(),
```

Patsy-sklearn

```
y ~ x_1 + x_2 + x_3
log(y) ~ x_1 + x_2 + x_3
log(y) ~ x_1:x_2 + x_3
log(y) ~ x_1:x_2 + log(x_3)
```

```
model = PatsyModel(LogisticRegression(),
                   "species ~ sepal_length + petal_length")
```

Patsy-sklearn

```
y ~ x_1 + x_2 + x_3
log(y) ~ x_1 + x_2 + x_3
log(y) ~ x_1:x_2 + x_3
log(y) ~ x_1:x_2 + log(x_3)
```

```
model = PatsyModel(LogisticRegression(),
                   "species ~ sepal_length + petal_length")

transformer = PatsyTransformer(
```

Patsy-sklearn

```
y ~ x_1 + x_2 + x_3
log(y) ~ x_1 + x_2 + x_3
log(y) ~ x_1:x_2 + x_3
log(y) ~ x_1:x_2 + log(x_3)
```

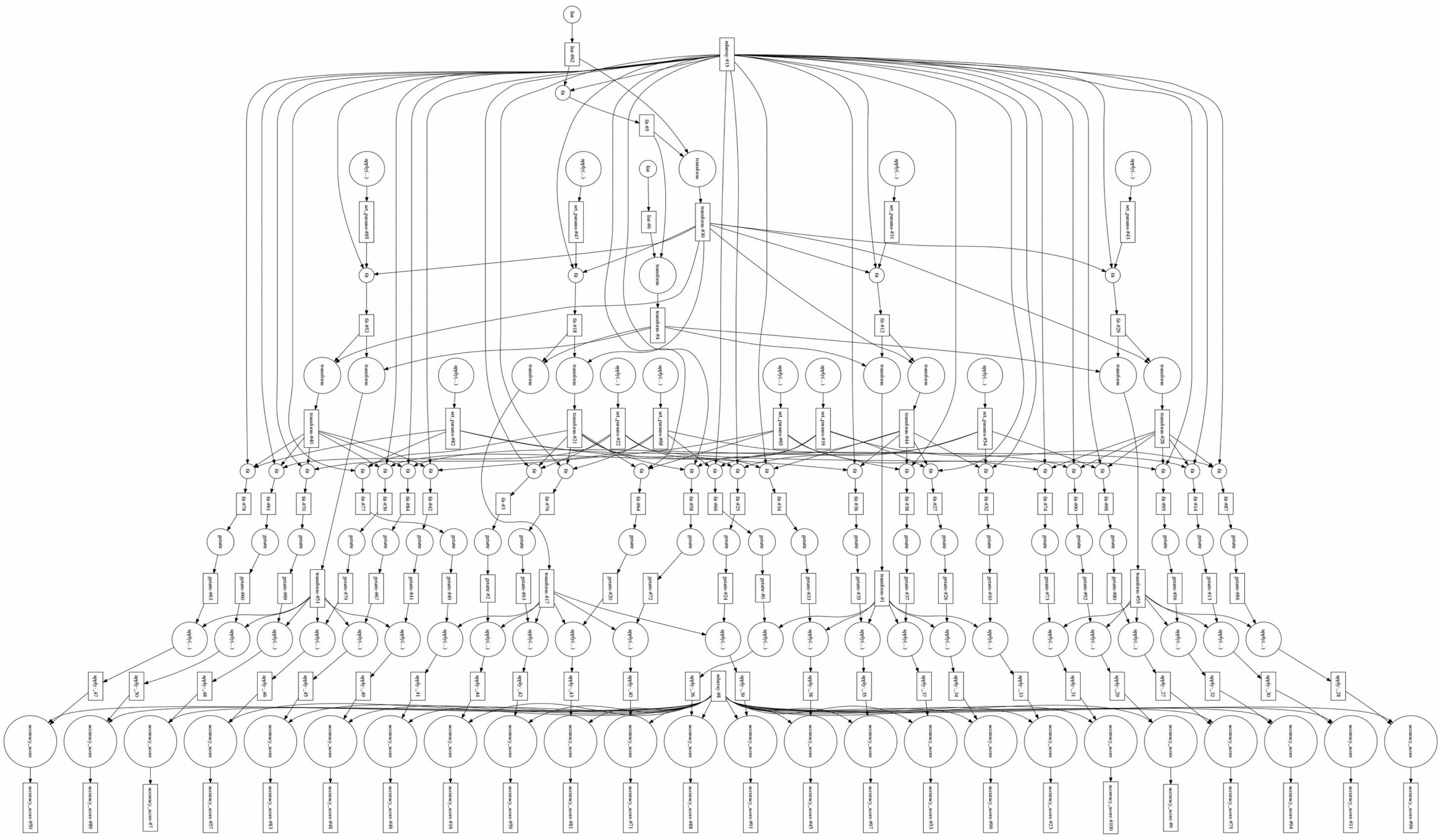
```
model = PatsyModel(LogisticRegression(),
                   "species ~ sepal_length + petal_length")

transformer = PatsyTransformer(
 "sepal_length + log(petal_length) + petal_length:sepal_width")
```

Dask-learn (with @mrocklin)

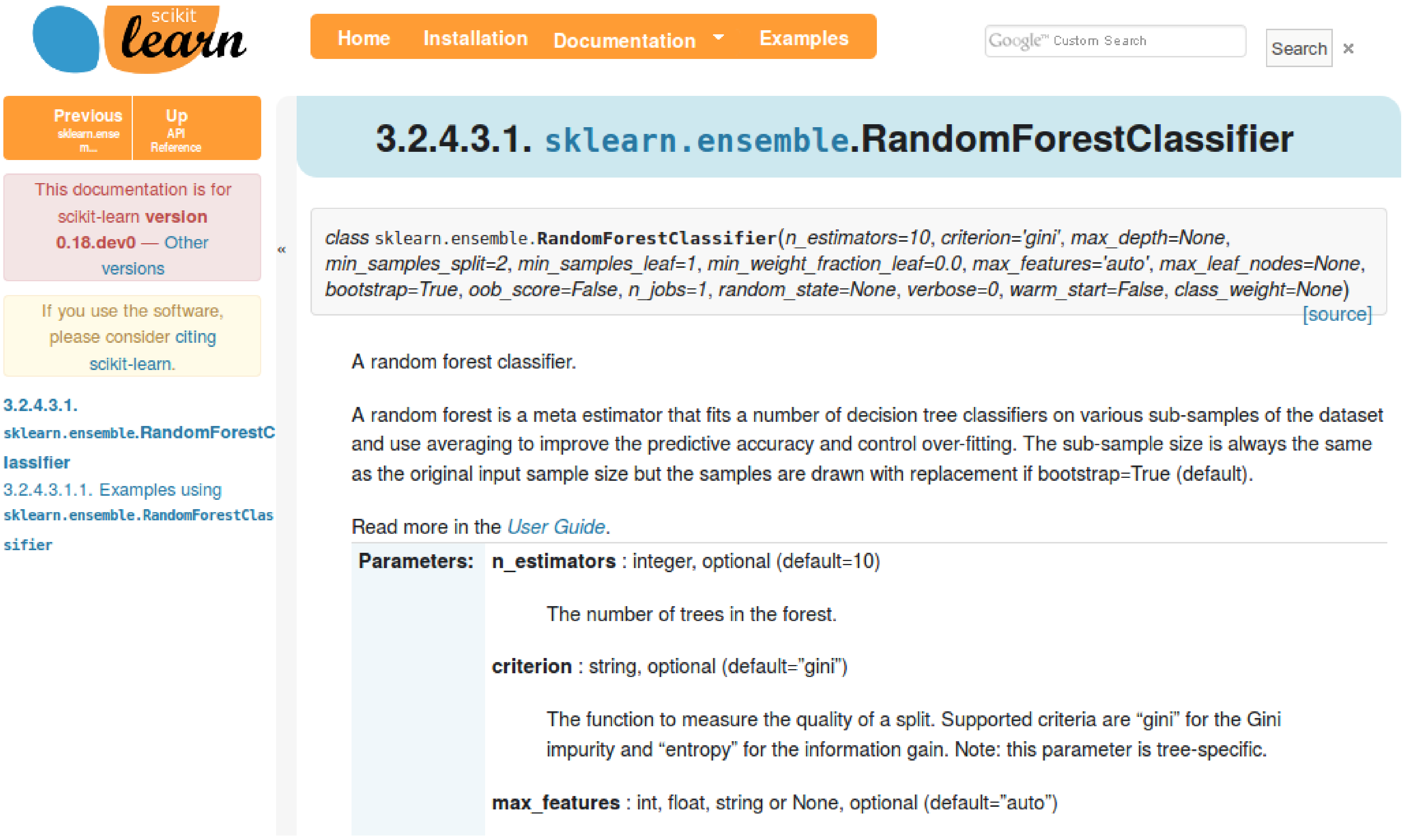
```
# from sklearn.pipeline import Pipeline
from dasklearn.pipeline import Pipeline

pipeline = Pipeline([("count", CountVectorizer()),
                     ("select_fdr", SelectFdr()),
                     ("svm", LinearSVC())])
```

```
from dask.imperative import value
X_train, y_train, X_test, y_test = map(value, [X_train, y_train, X_test, y_test])

scores = [pipeline.set_params(**params)
                  .fit(X_train, y_train)
                  .score(X_test, y_test)
          for params in parameters]

result = compute(scores, get=get_sync)
```

Dask-learn (with @mrocklin)



Add Backlinks to Docs



This documentation is for scikit-learn **version 0.18.dev0** — Other versions

If you use the software, please consider citing scikit-learn.

3.2.4.3.1. `sklearn.ensemble.`RandomForestClassifier
3.2.4.3.1.1. Examples using **`sklearn.ensemble.RandomForestClassifier`**

3.2.4.3.1. `sklearn.ensemble`.RandomForestClassifier

class `sklearn.ensemble.`**`RandomForestClassifier`**(*n_estimators=10*, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *bootstrap=True*, *oob_score=False*, *n_jobs=1*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*) [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

Read more in the *User Guide*.

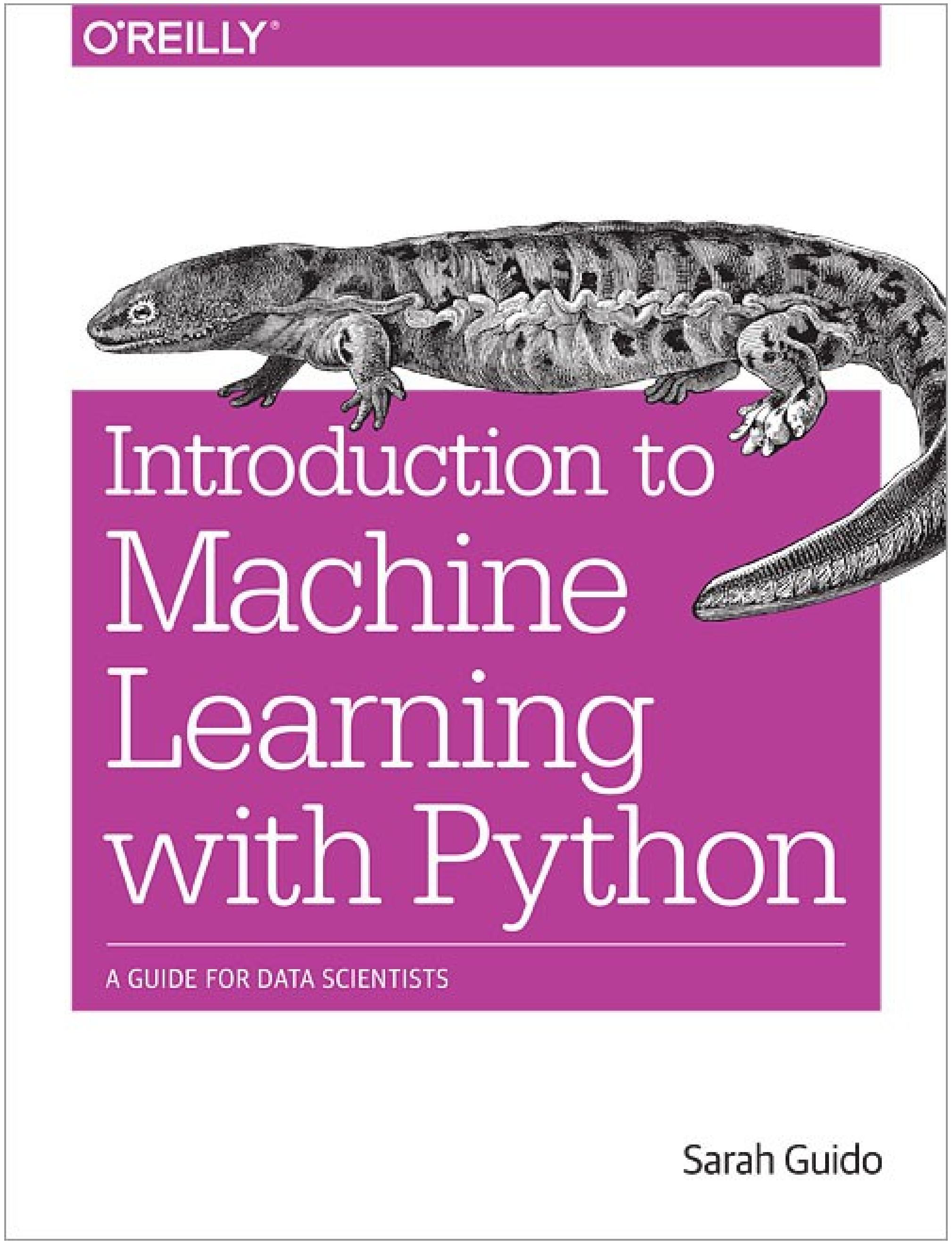
Parameters: **n_estimators** : integer, optional (default=10)

The number of trees in the forest.

criterion : string, optional (default="gini")

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

max_features : int, float, string or None, optional (default="auto")


O'REILLY

Introduction to Machine Learning with Python

A GUIDE FOR DATA SCIENTISTS

Sarah Guido

Thank you for your attention.

@t3kcit

@amueller

importamueller@gmail.com

http://amueller.github.io