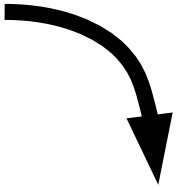
Building scikit-learn

Andreas Müller
NYU, scikit-learn

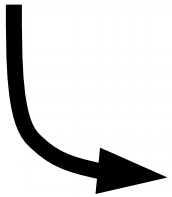
✉ amueller@nyu.edu
@amuellerml
@amueller
Amueller.io

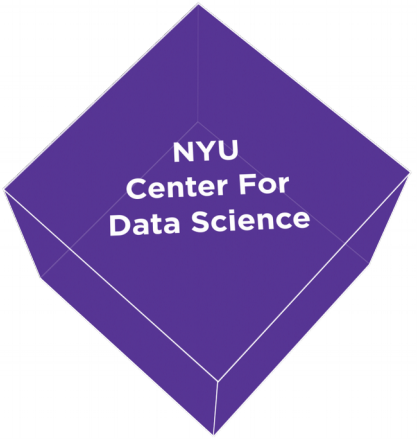
Me


universitätbonn
ais




Amazon
Machine
Learning




NYU
Center For
Data Science

Classification
Regression
Clustering
Semi-Supervised Learning
Feature Selection
Feature Extraction
Manifold Learning
Dimensionality Reduction
Kernel Approximation
Hyperparameter Optimization
Evaluation Metrics
Out-of-core learning
……


agramfort
Alexandre Gramfort


AlexanderFabisch
Alexander Fabisch


alextp
Alexandre Passos


amueller
Andreas Mueller


arjoly
Arnaud Joly


bdholt1
Brian Holt


bthirion
bthirion


chrisfilo
Chris Filo Gorgole...


cournape
David Cournapeau


duchesnay
Duchesnay


dwf
David Warde-Farley


fabianp
Fabian Pedregosa


GaelVaroquaux
Gael Varoquaux


glouppe
Gilles Louppe


jakevdp
Jake Vanderplas


jaquesgrobler
Jaques Grobler


jnothman


kastnerkyle
Kyle Kastner


kuantkid
Wei LI


larsmans
Lars


lucidfrontier45
Shiqiao Du


mblondel
Mathieu Blondel


MechCoder
Manoj Kumar


ndawe
Noel Dawe


NelleV
Varoquaux


ogrisel
Olivier Grisel


paolo-losi
Paolo Losi


pprett
Peter Prettenhofer


robertlayton
Robert Layton


ronw
Ron Weiss


satra
Satrajit Ghosh


sklearn-ci


vene
Vlad Niculae


VirgileFritsch
Virgile Fritsch


vmichel
Vincent Michel


yarikoptic
Yaroslav Halchenko

Mission

Commoditize and Democratize Machine Learning

Simplicity

```
lr = LogisticRegression()
lr.fit(X_train, y_train)
lr.score(X_test, y_test)
```

Consistency

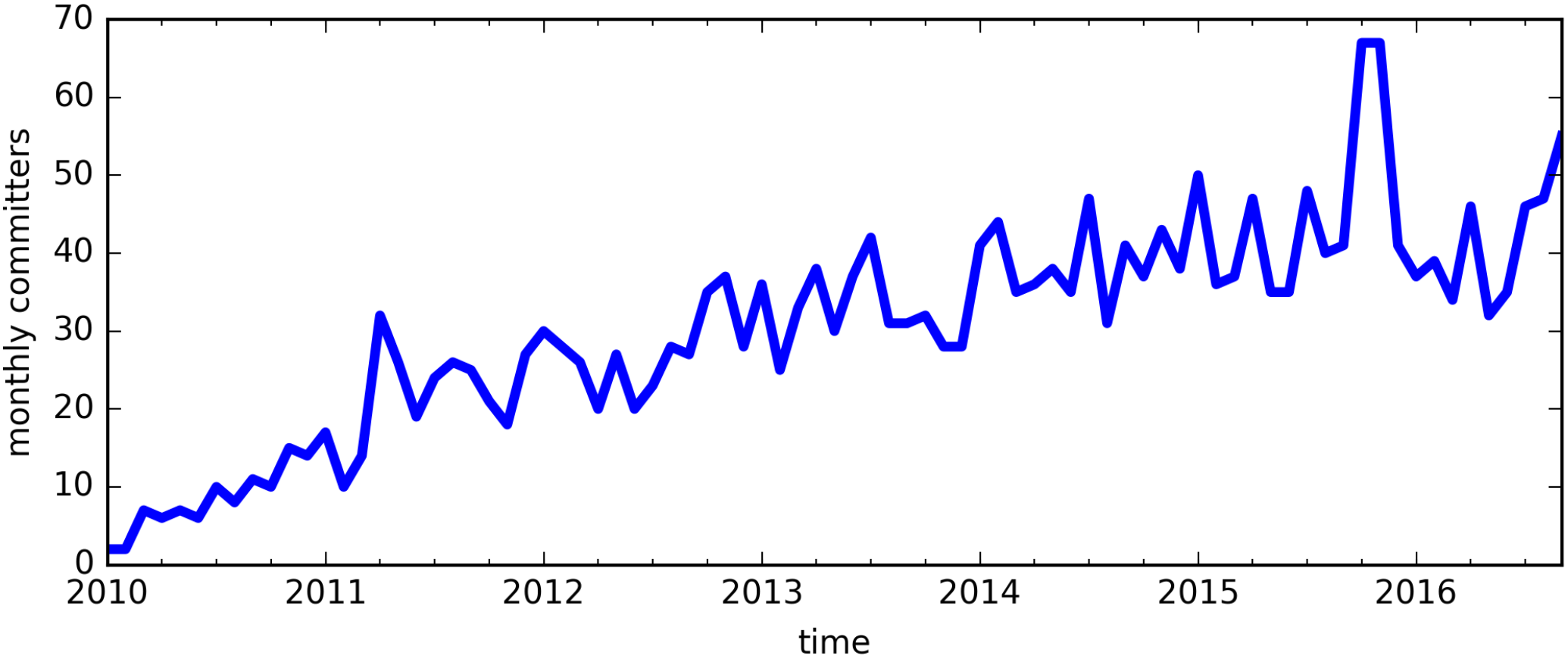
```
grid = GridSearchCV(svm,param_grid)
grid.fit(X_train, y_train)
grid.score(X_test, y_test)
```

Flat Class Hierarchy, Few Types

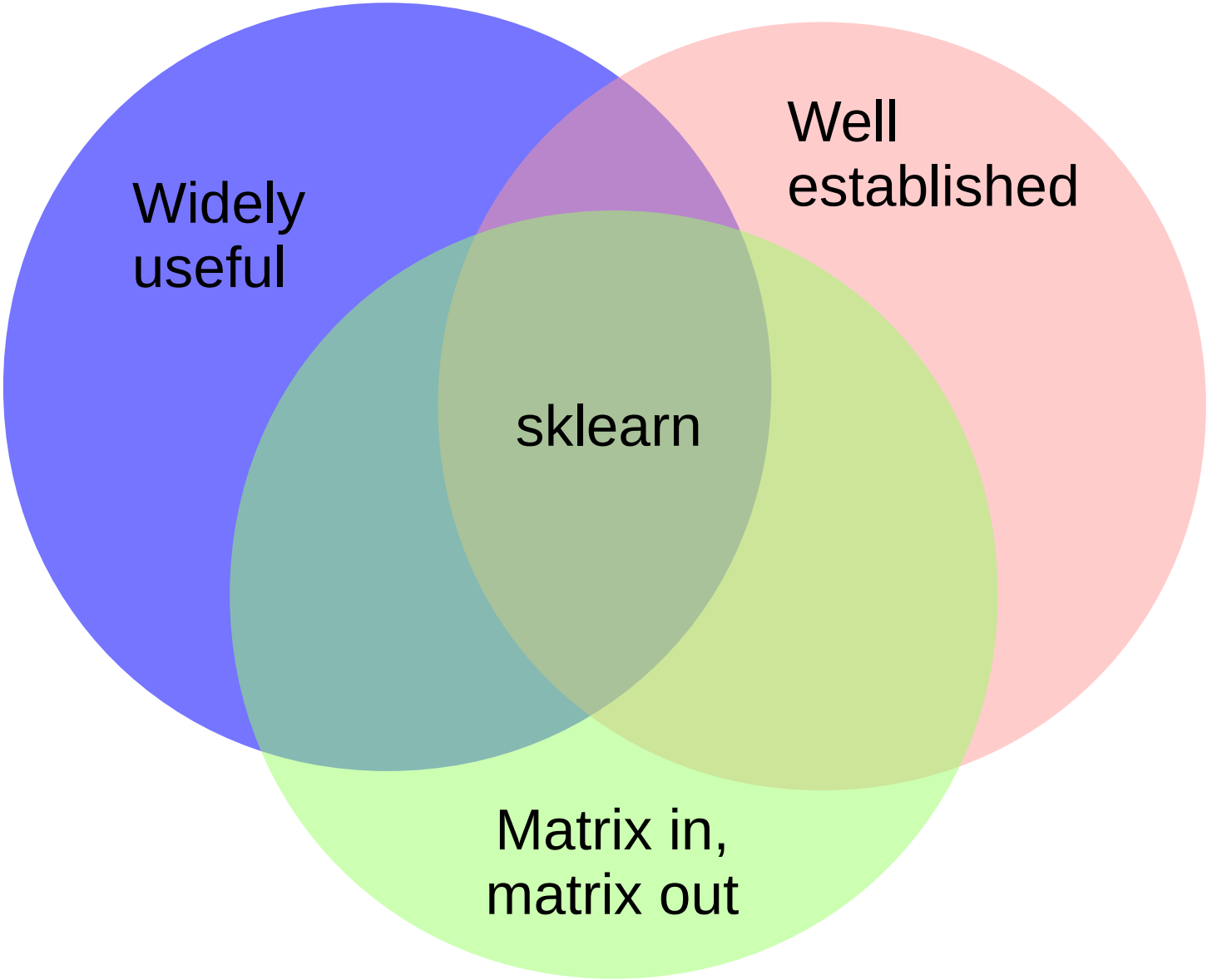
- Numpy arrays / sparse matrices
- Estimators
- [Cross-validation objects]
- [Scorers]

Maintainability

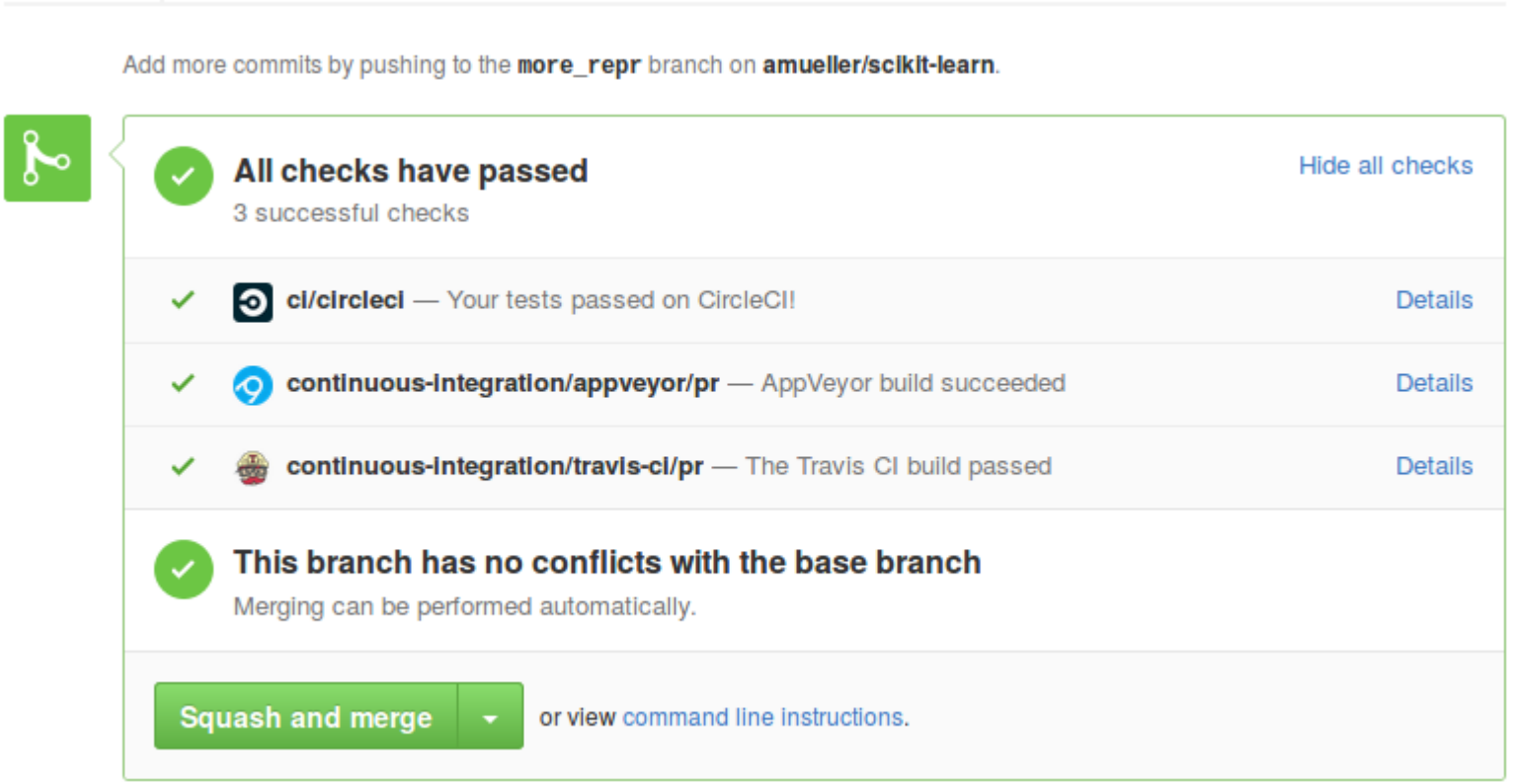
Code Contributors Over Time



Scoping


Widely
useful
Well
established
sklearn
Matrix in,
matrix out

Testing & Continuous Integration



Three way documentation

1.9. Ensemble methods

The goal of **ensemble methods** is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

Two families of ensemble methods are usually distinguished:

- In **averaging methods**, the driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.

  Examples: *Bagging methods*, *Forests of randomized trees*, ...

- By contrast, in **boosting methods**, base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble.

  Examples: *AdaBoost*, *Gradient Tree Boosting*, ...

`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*) [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.

Parameters:

n_estimators : integer, optional (default=10)

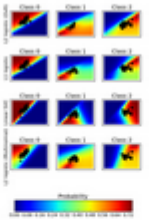
The number of trees in the forest.

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

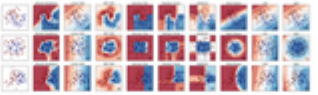
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

Examples

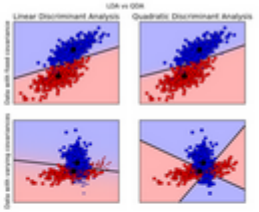
Recognizing hand-written digits



Plot classification probability



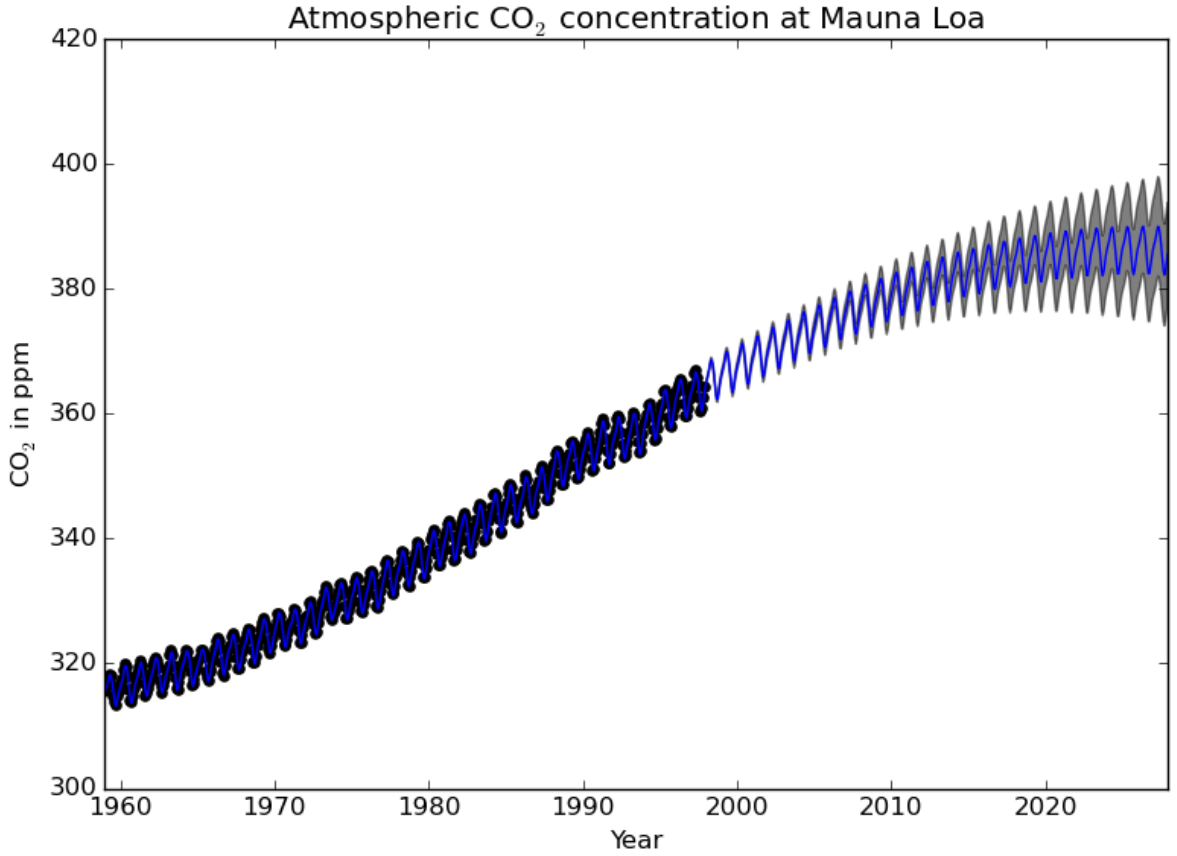
Classifier comparison



Linear and Quadratic Discriminant Analysis with confidence ellipsoid

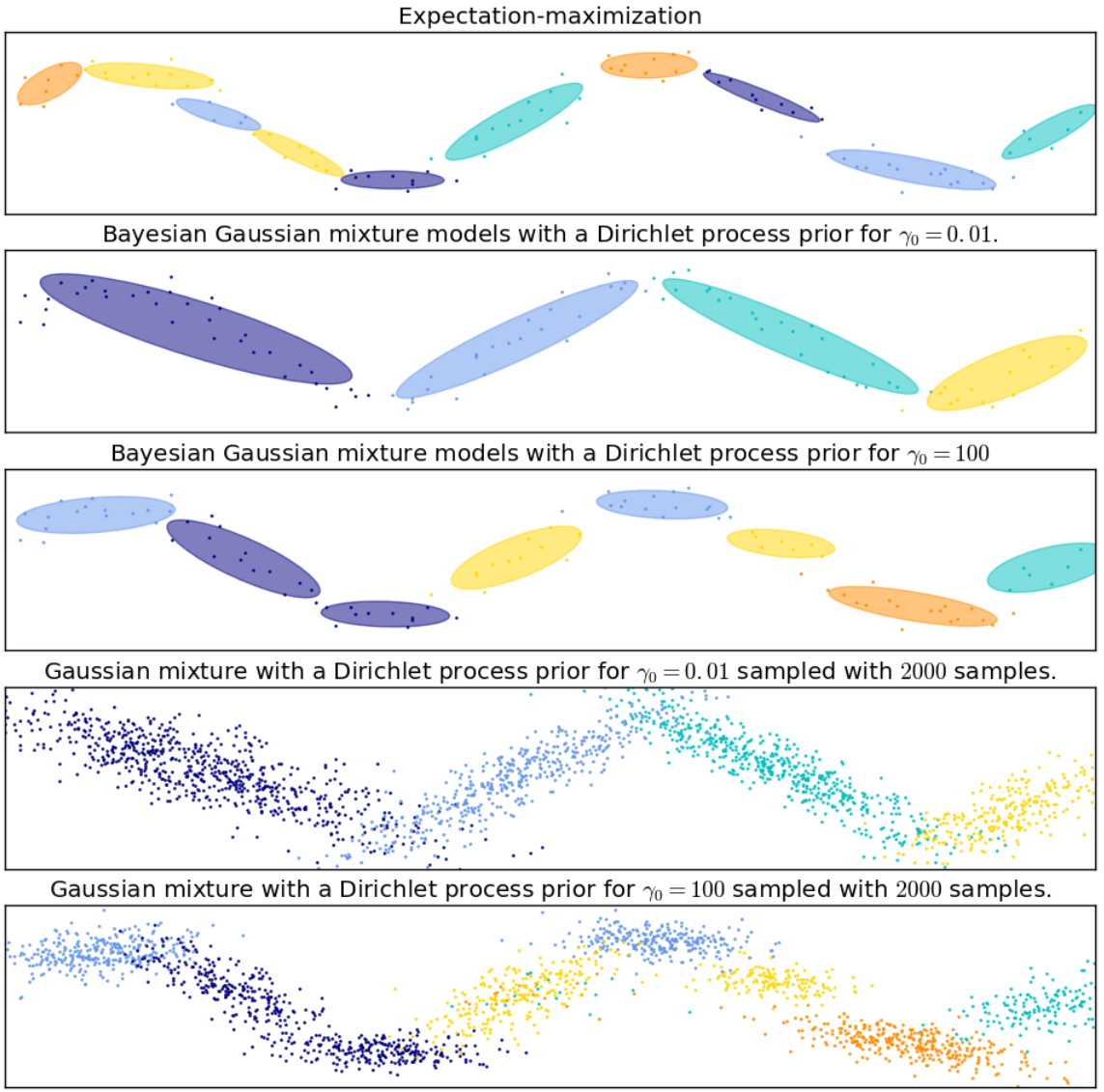
Additions in the 0.18 release

Gaussian Process Rewrite

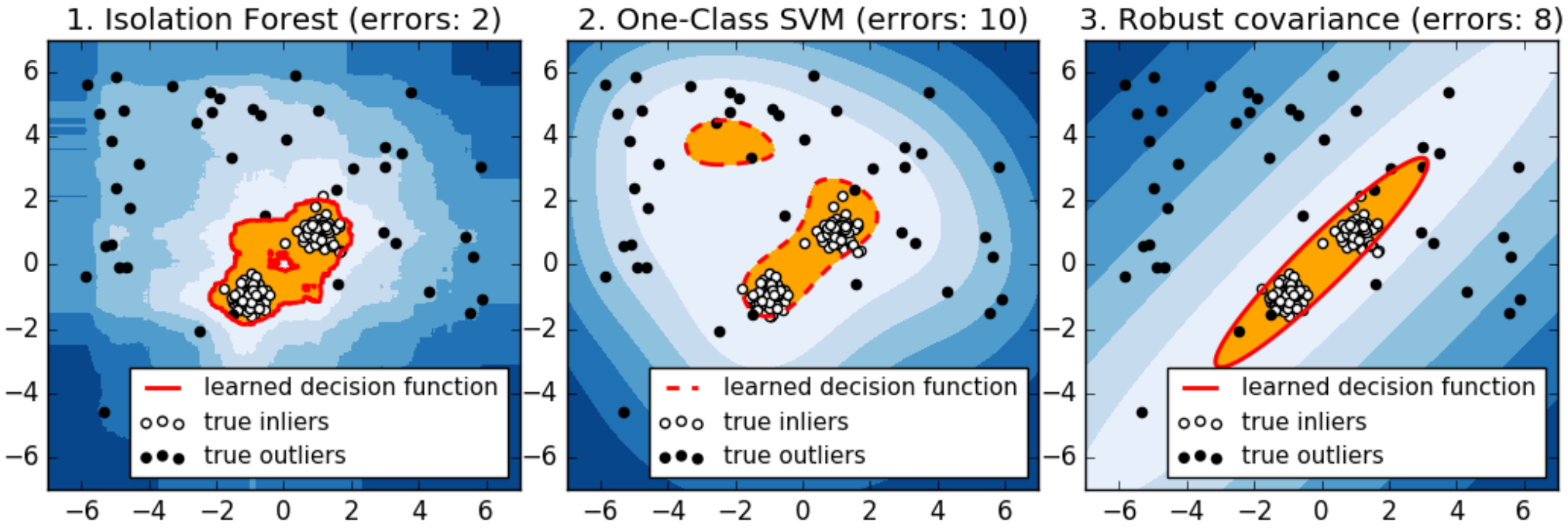



```
34.4**2 * RBF(length_scale=41.8)
+ 3.27**2 * RBF(length_scale=180)
    * ExpSineSquared(length_scale=1.44, periodicity=1)
+ 0.446**2 * RationalQuadratic(alpha=17.7, length_scale=0.957)
+ 0.197**2 * RBF(length_scale=0.138) +
WhiteKernel(noise_level=0.0336)
```

```
from sklearn.mixture import GaussianMixture
from sklearn.mixture import BayesianGaussianMixture
```

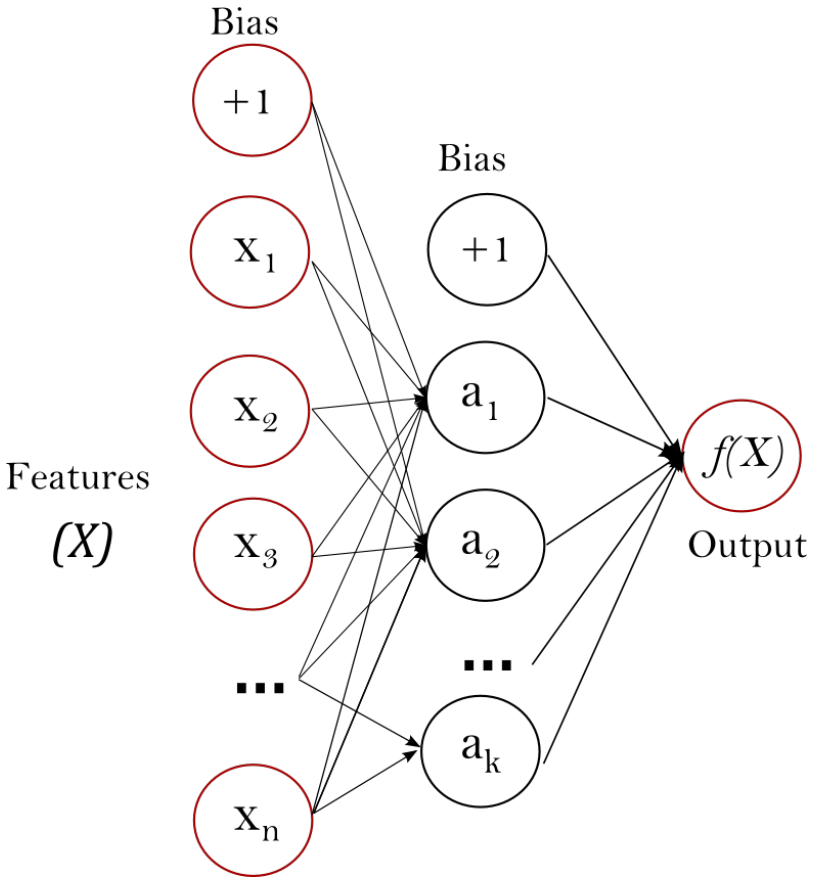


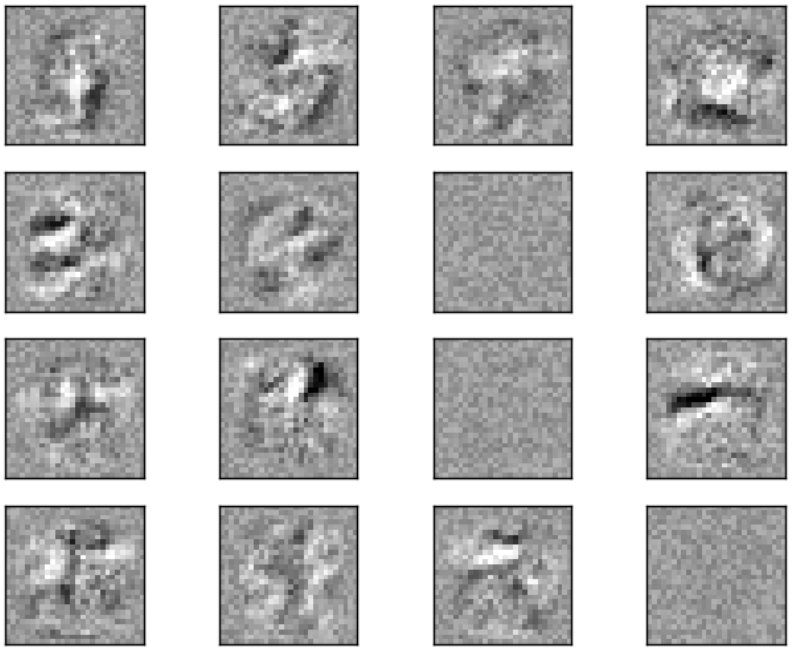
Isolation Forests



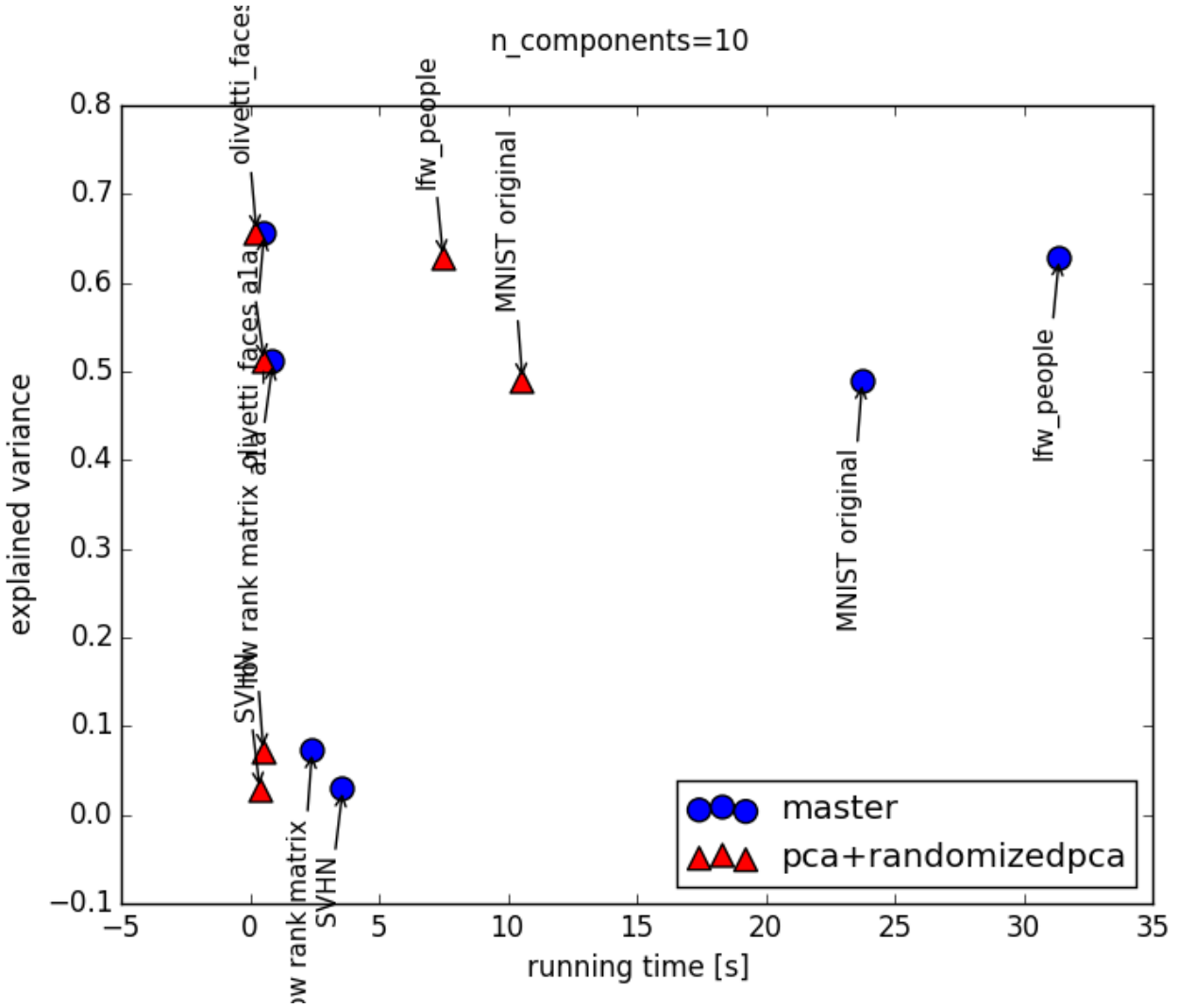
```
pipe = Pipeline([('preprocessing', StandardScaler()),
                 ('classifier', SVC())])

param_grid = {'preprocessing': [StandardScaler(), None]}

grid = GridSearchCV(pipe, param_grid)
```

Neural Networks





Faster Principal Component Analysis



Other software projects

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)
```

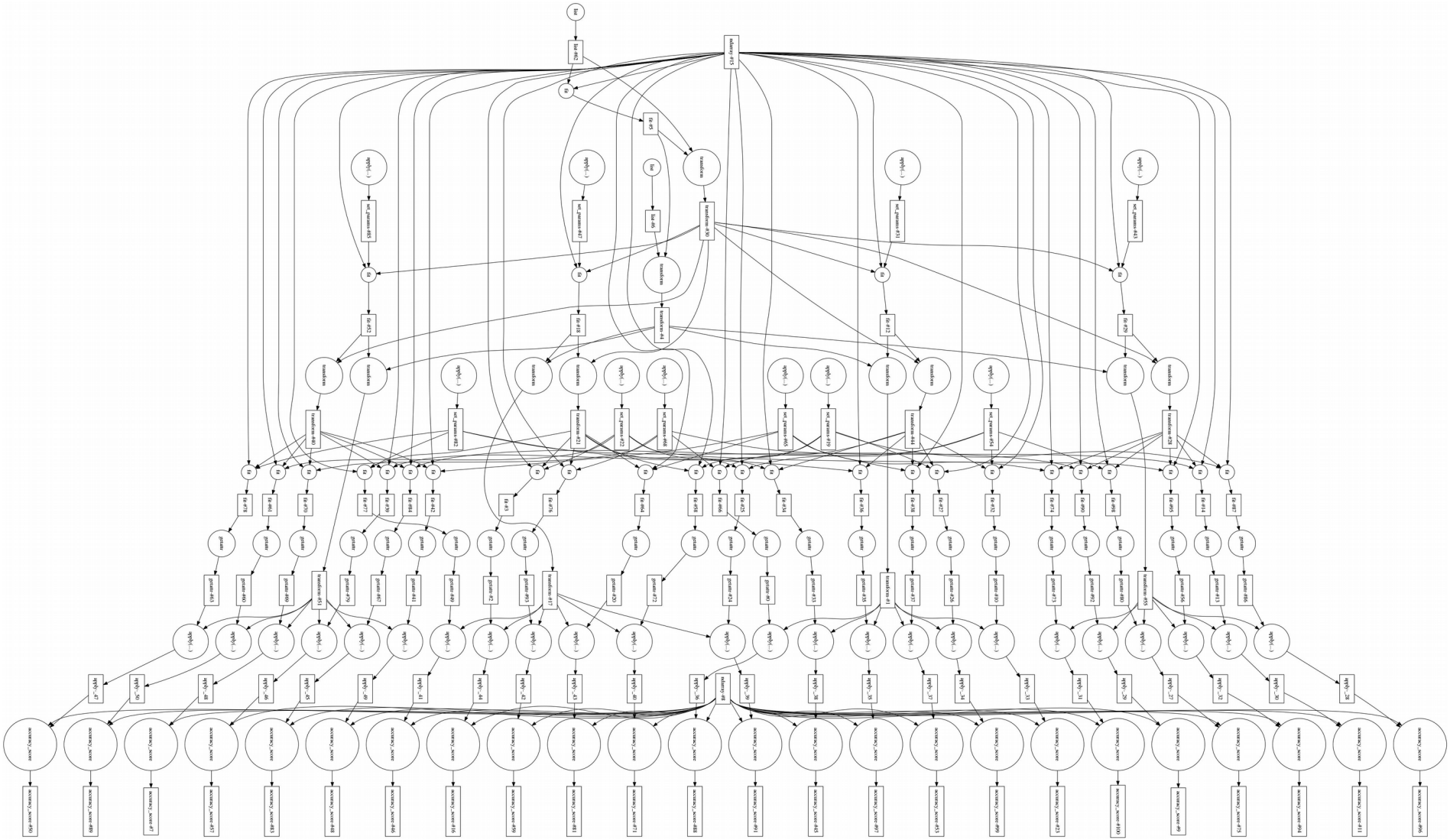
```
model = PatsyModel(LogisticRegression(),
                   "species ~ sepal_length + petal_length")

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

Dask-learn (with Matt Rocklin)

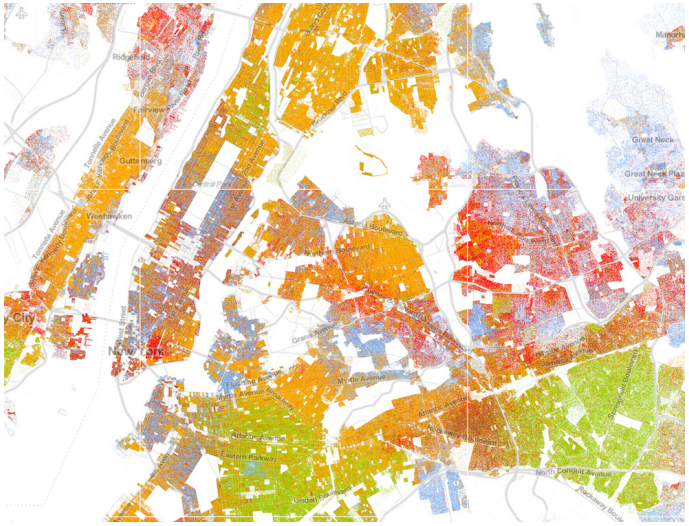
```
# from sklearn.pipeline import Pipeline
from dasklearn.pipeline import Pipeline

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 Matt Rocklin)



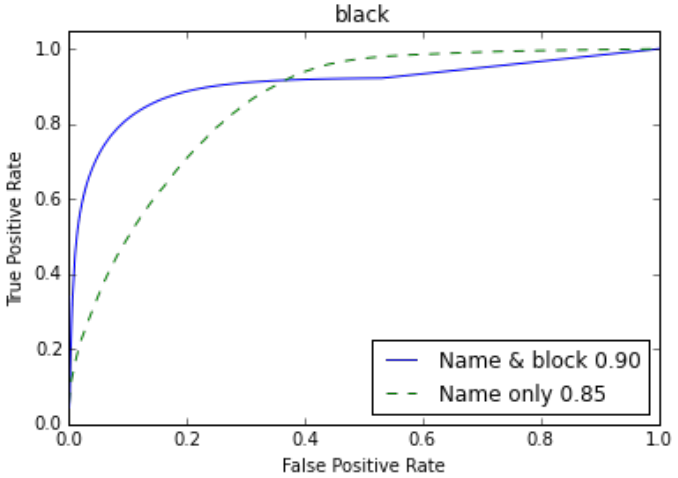
Voter Ethnicity Prediction (Tian Wang)



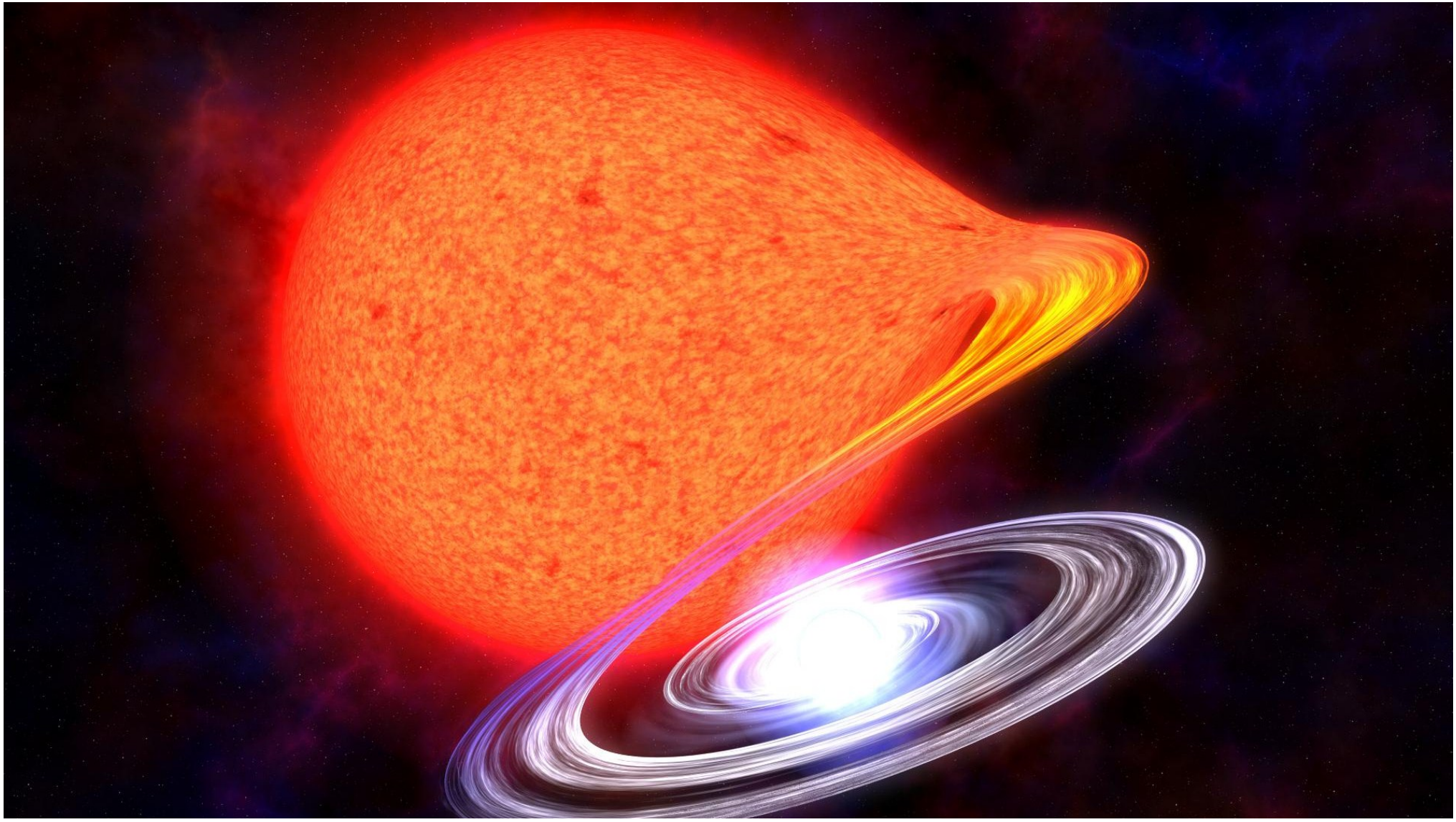
name	rank	count	prop100k	cum_prop100k	pctwhite	pctblack	pctapi	pctaian	pct2prace	pcthispanic
SMITH	1	2376206	880.85	880.85	73.35	22.22	0.4	0.85	1.63	1.56
JOHNSON	2	1857160	688.44	1569.3	61.55	33.8	0.42	0.91	1.82	1.5
WILLIAMS	3	1534042	568.66	2137.96	48.52	46.72	0.37	0.78	2.01	1.6
BROWN	4	1380145	511.62	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

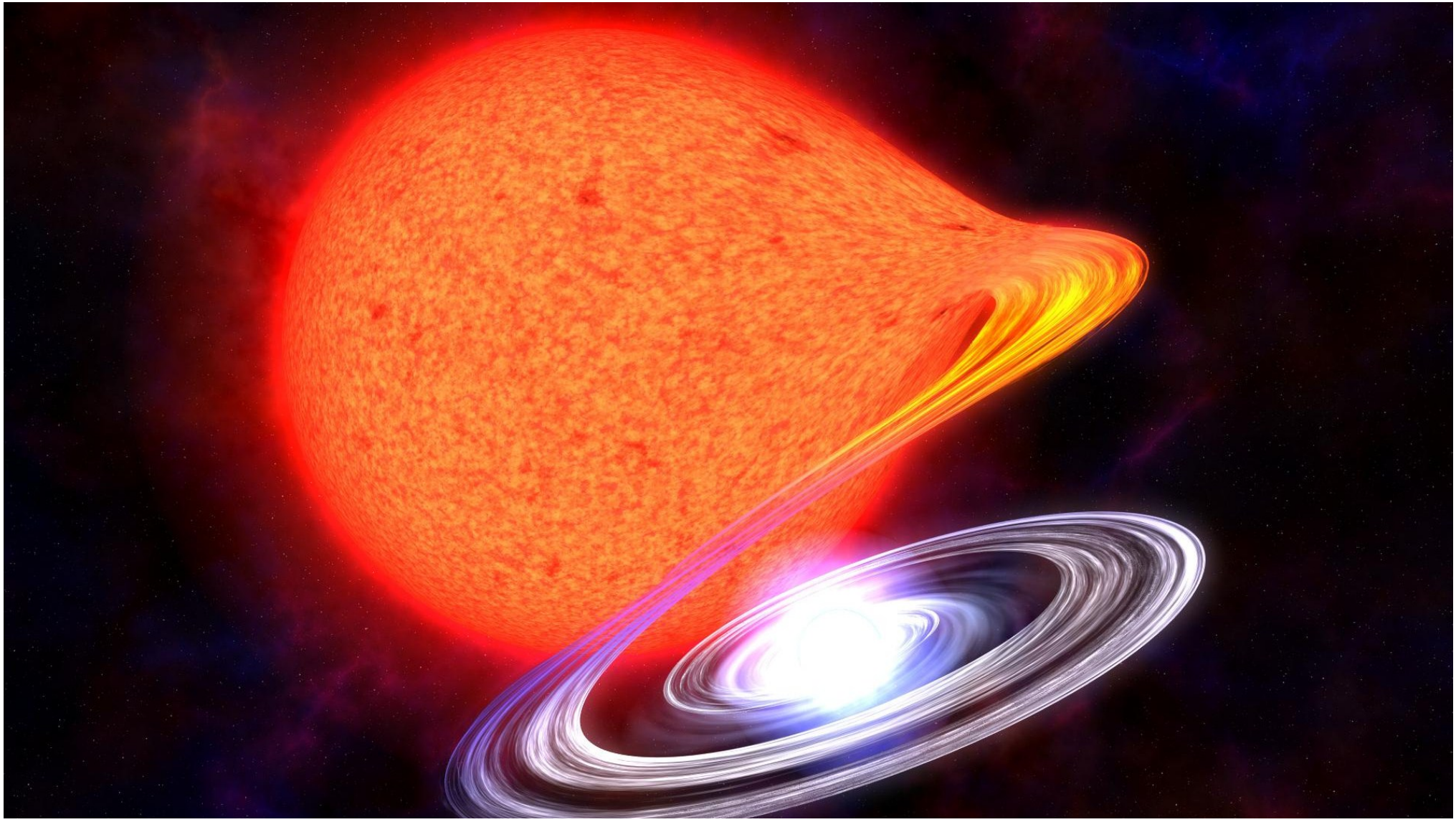
~ Voter ethnicity



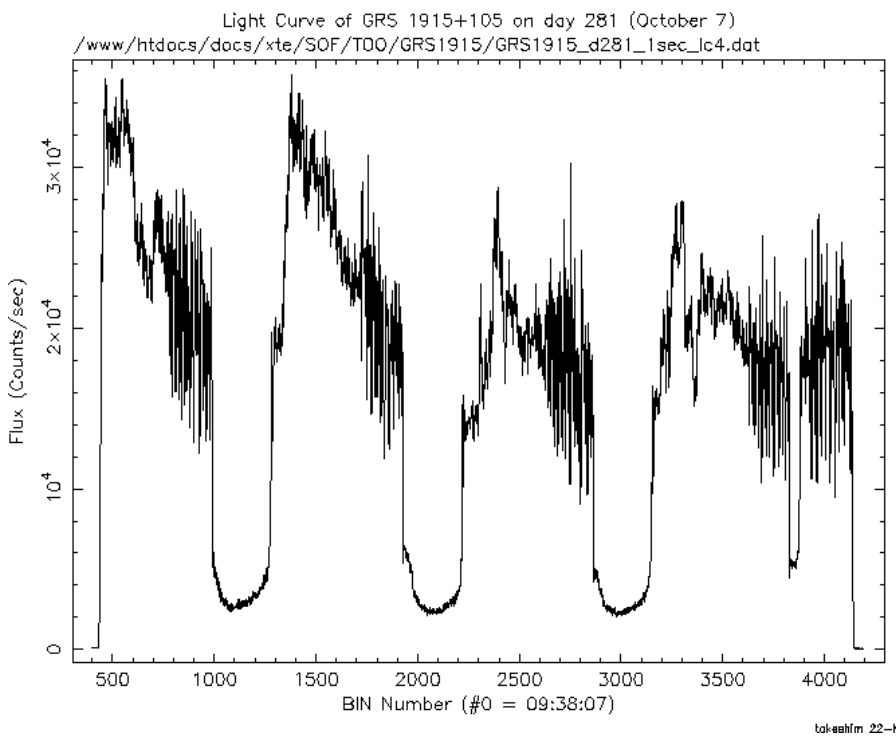
X-Ray Binaries
(Daniela Huttenkoppen, Nicolas Goix)

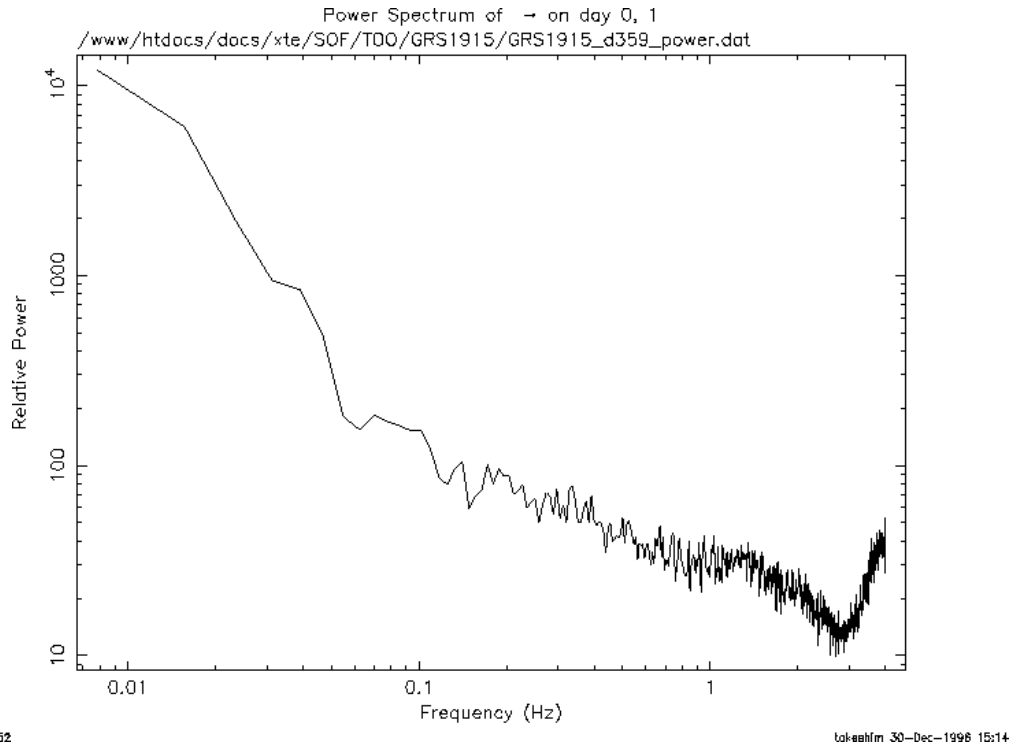


X-Ray Binaries
(Daniela Huttenkoppen, Nicolas Goix)



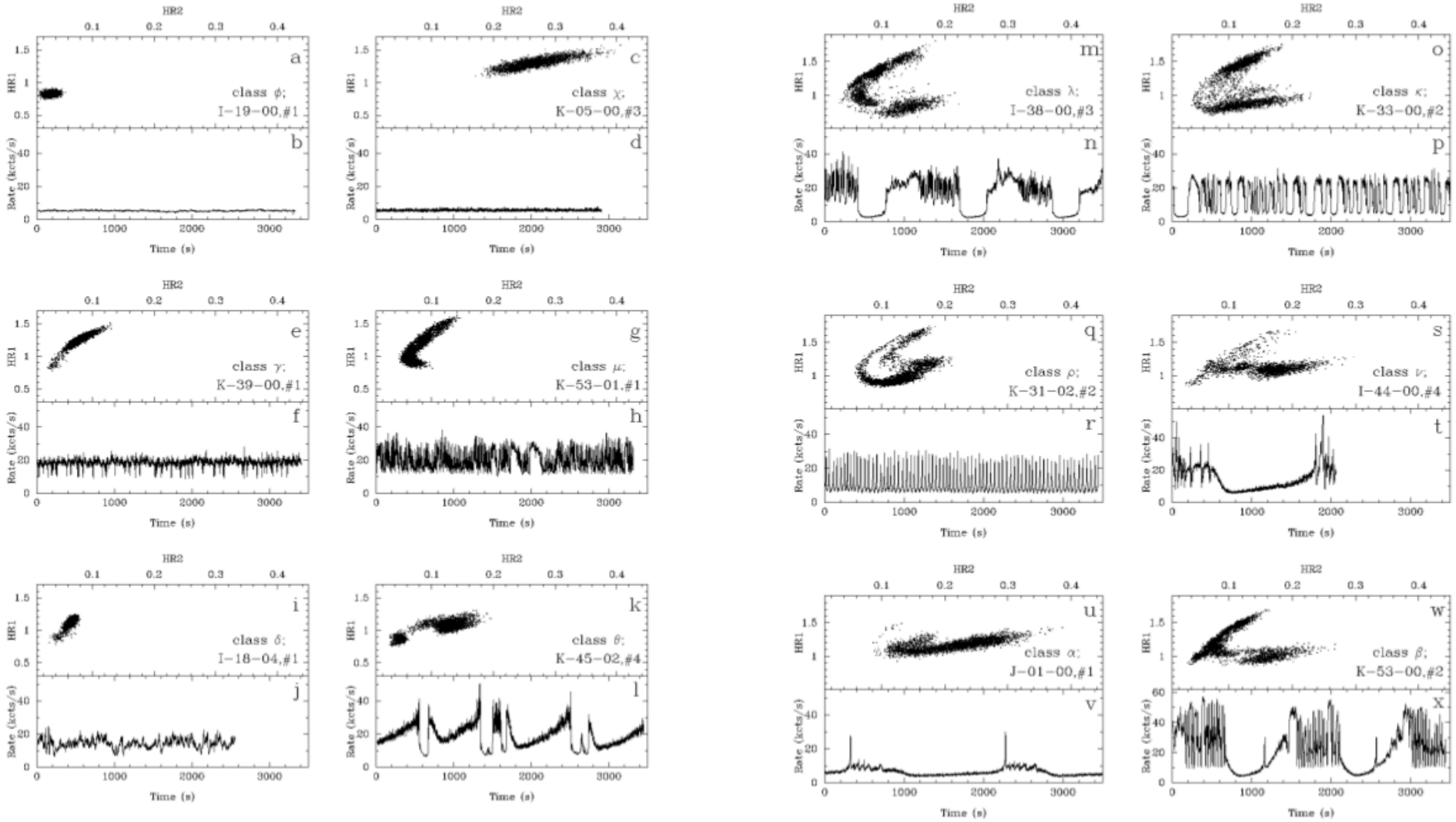
Measurements and Observations





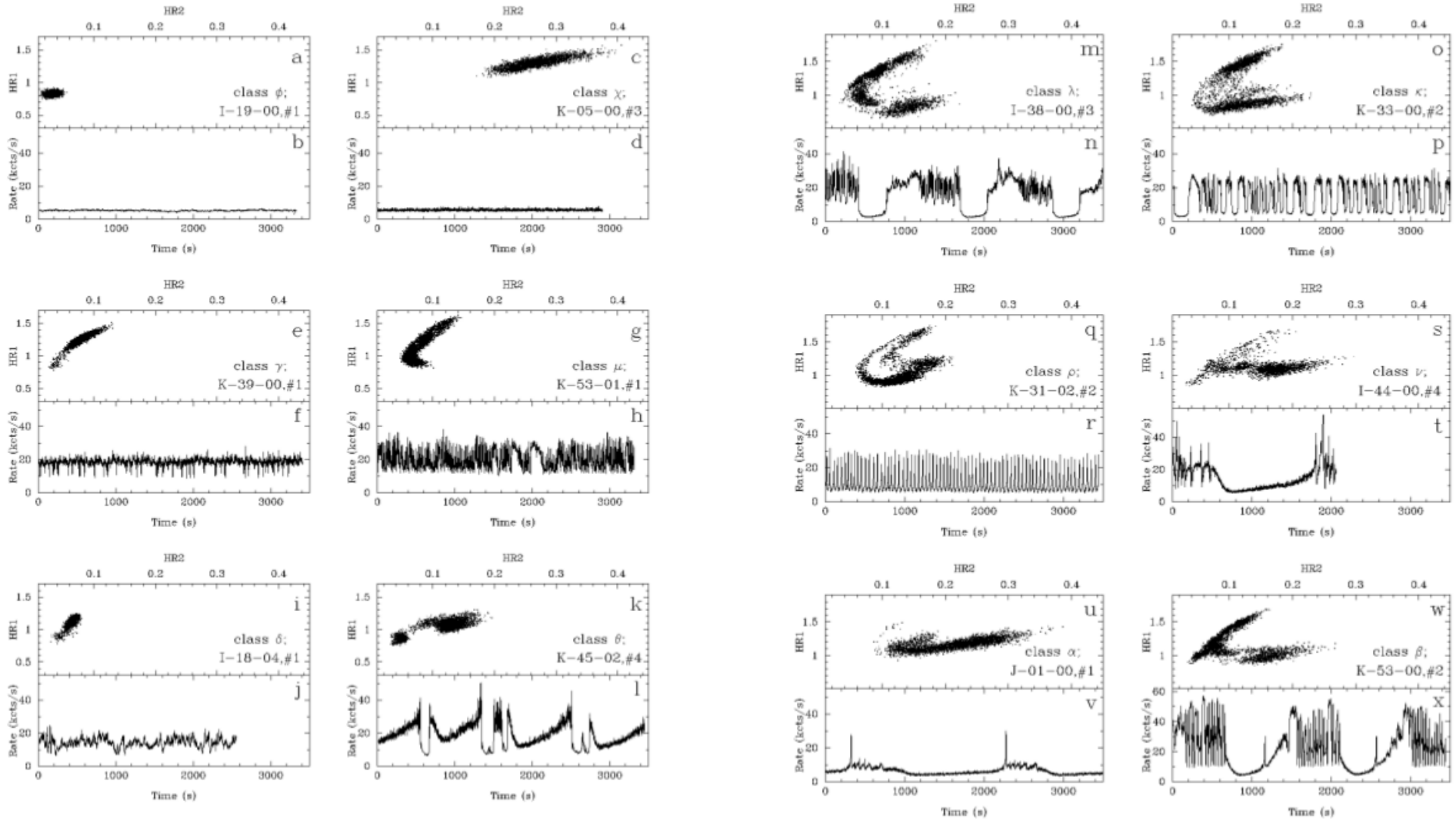
Black Hole States and Transitions
GRS 1915+105

(Huppenkothen, Heil, Hogg, Mueller)



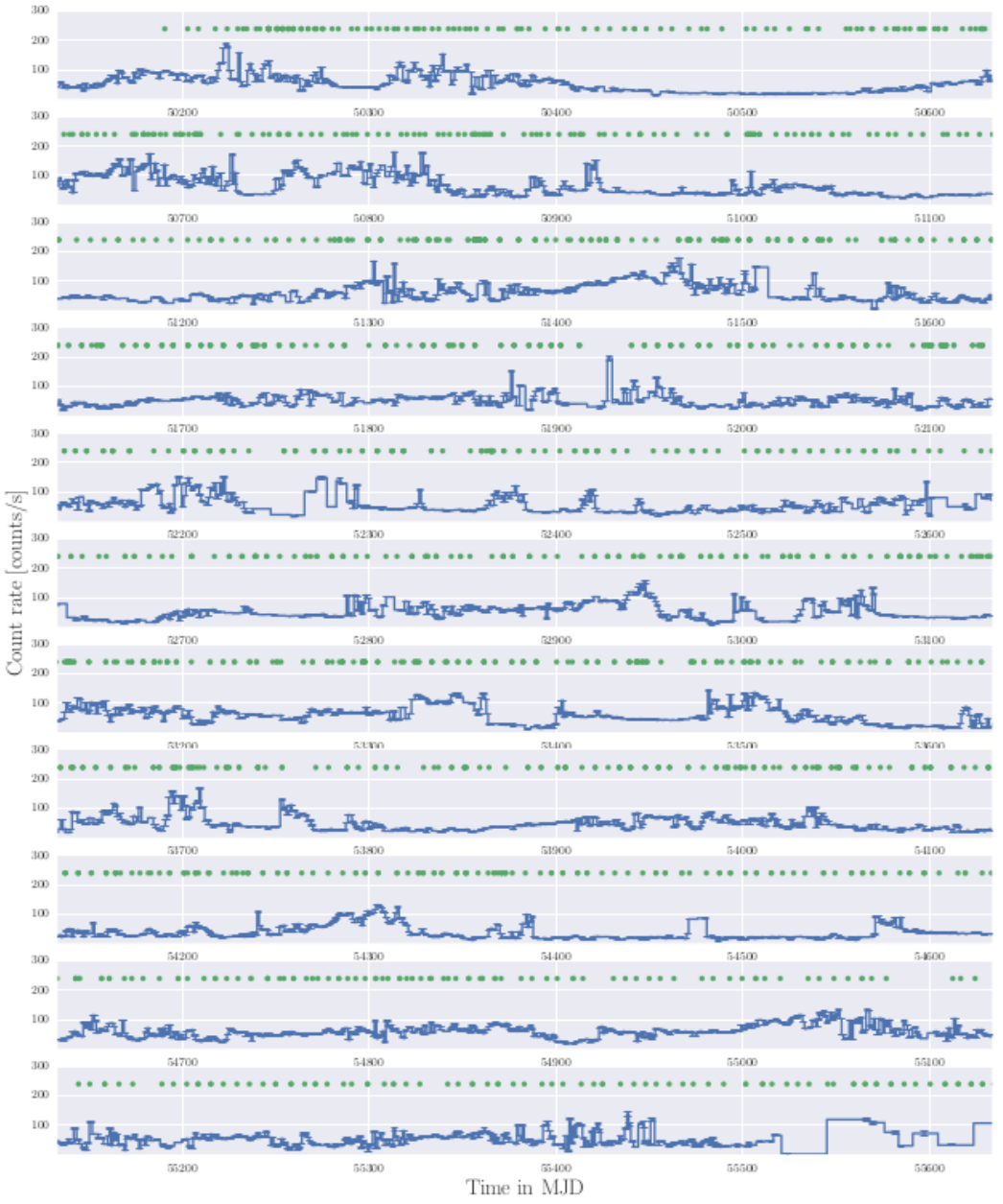
Belloni et. al. 2000

Black Hole States and Transitions
GRS 1915+105

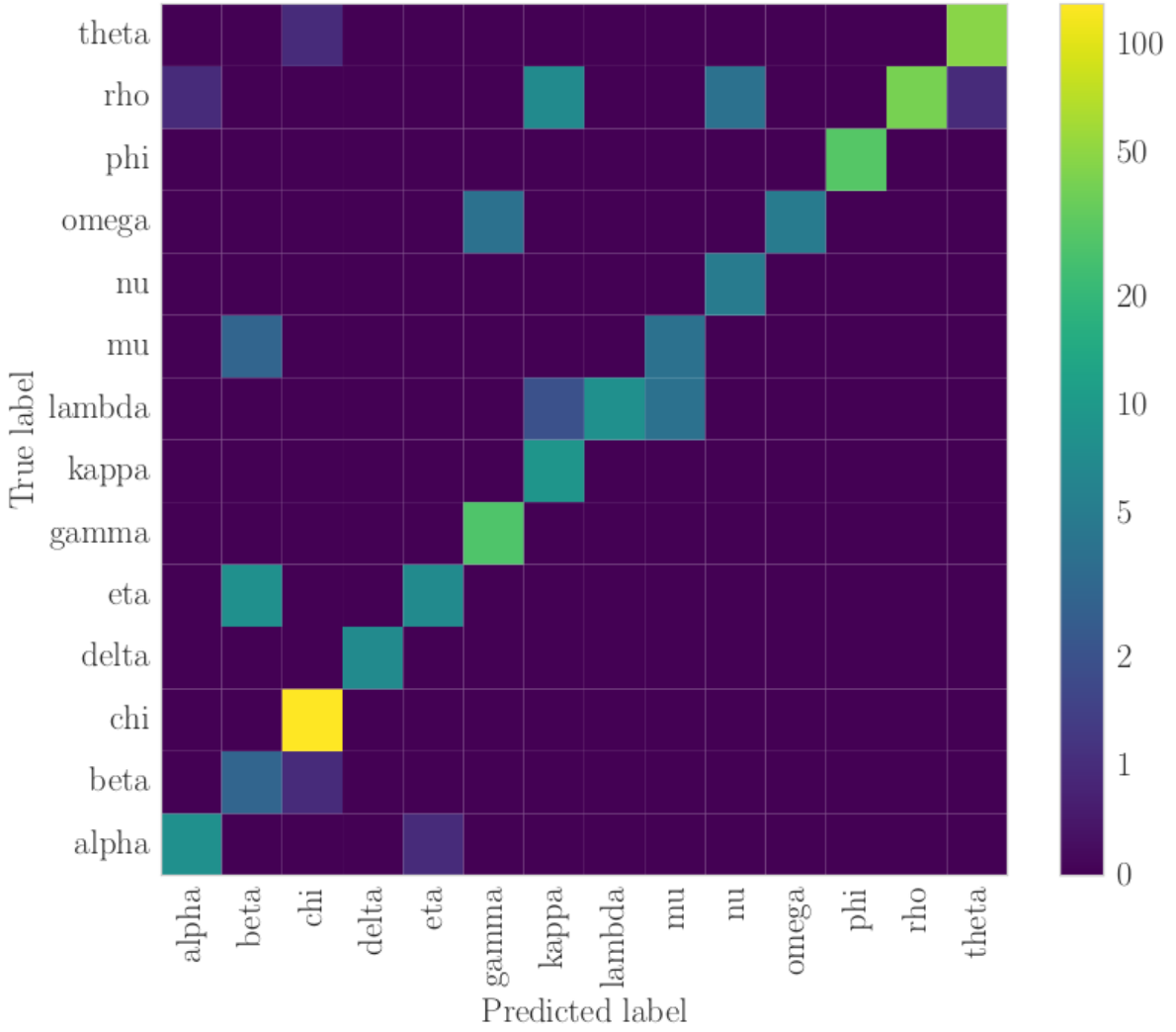


Belloni et. al. 2000

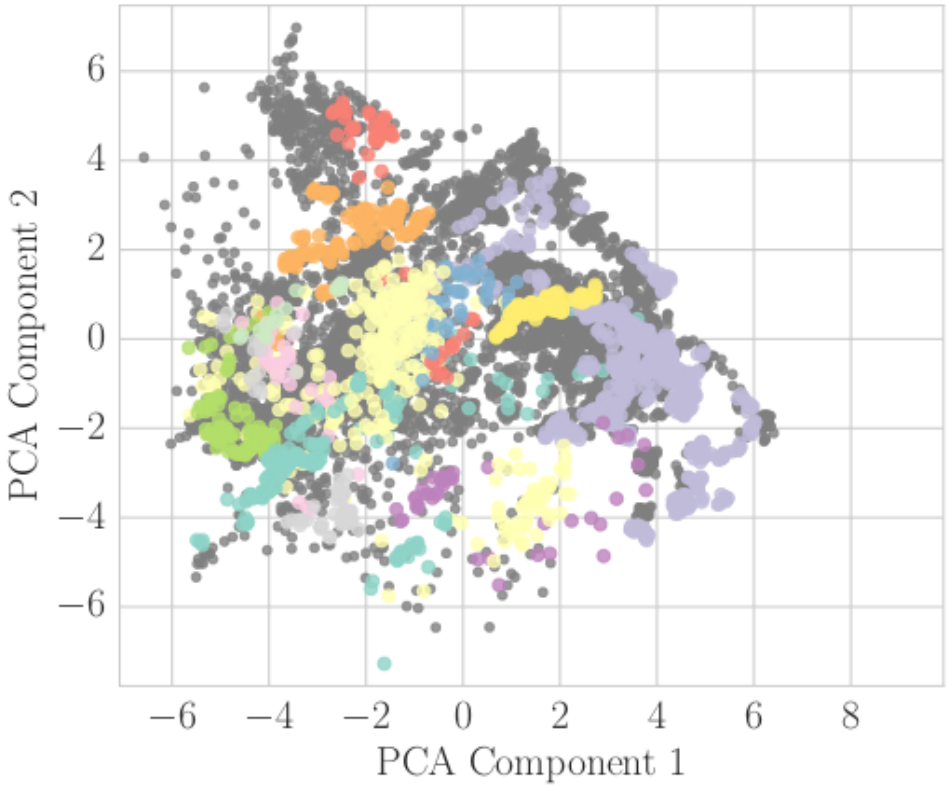
Black Hole States and Transitions

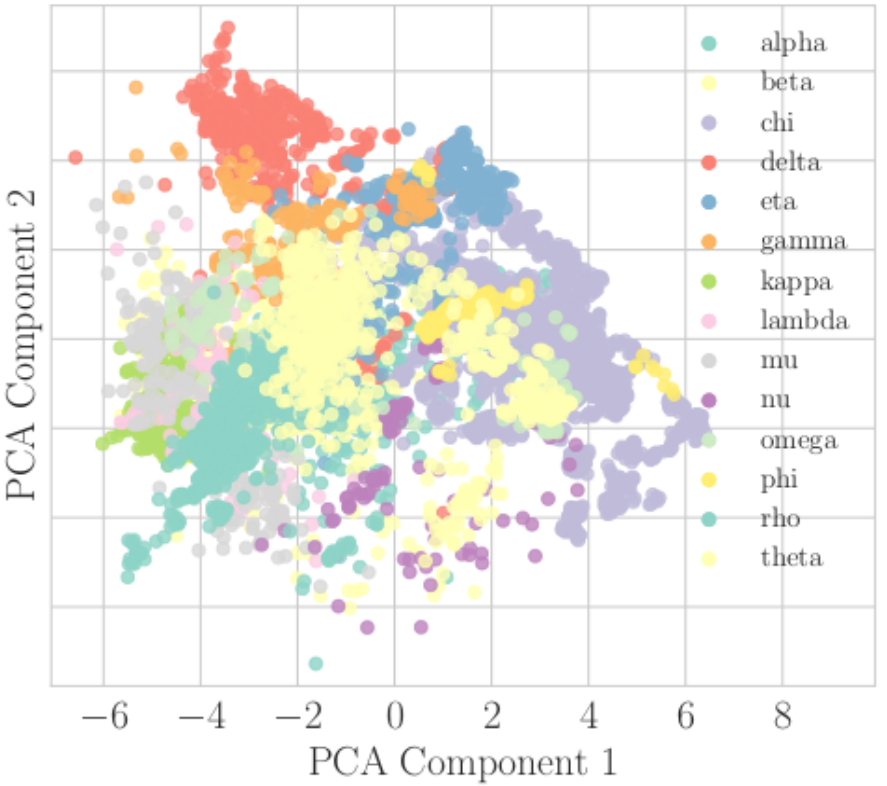


State Classification Results

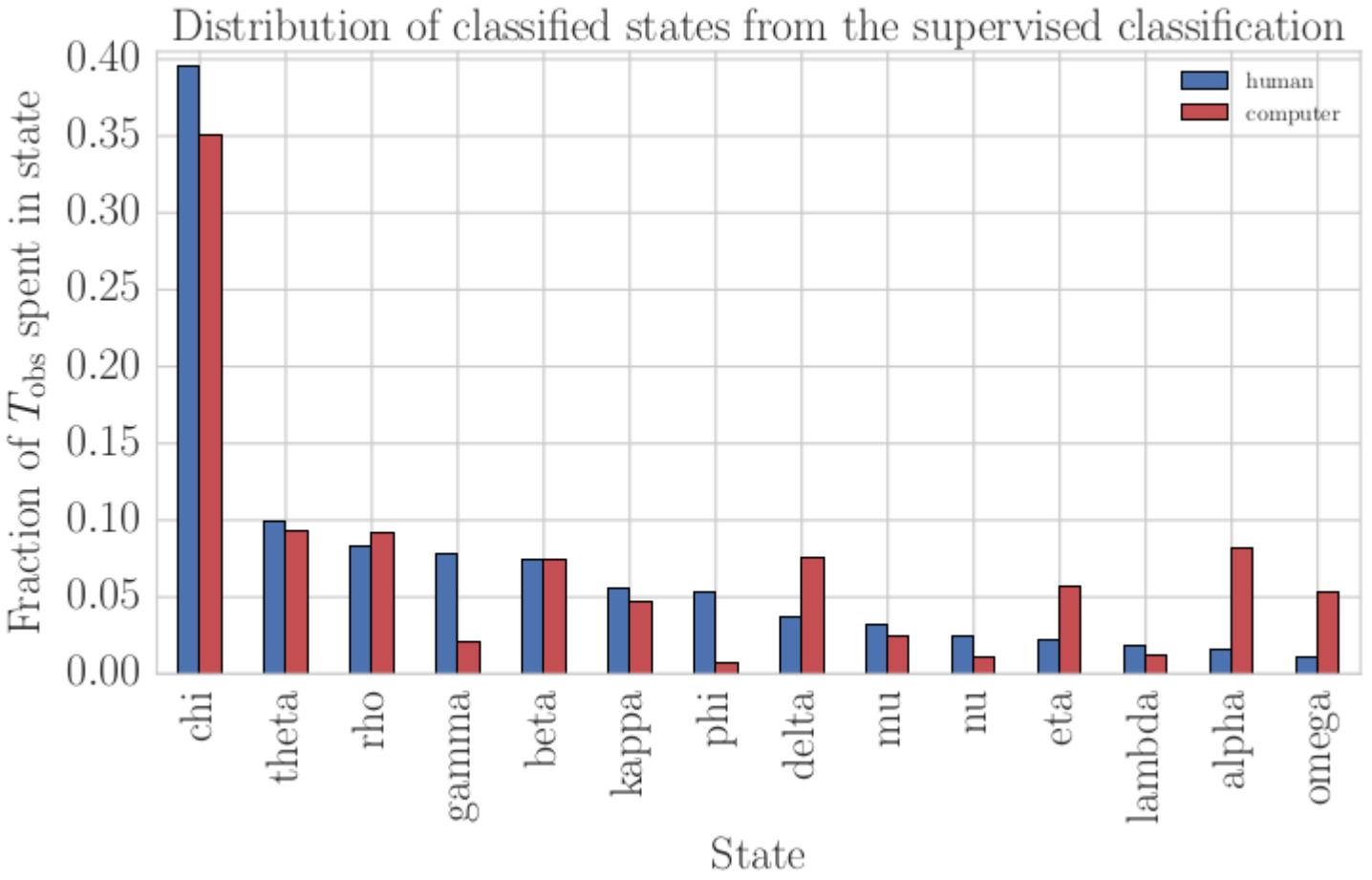


Validation of Inferred States

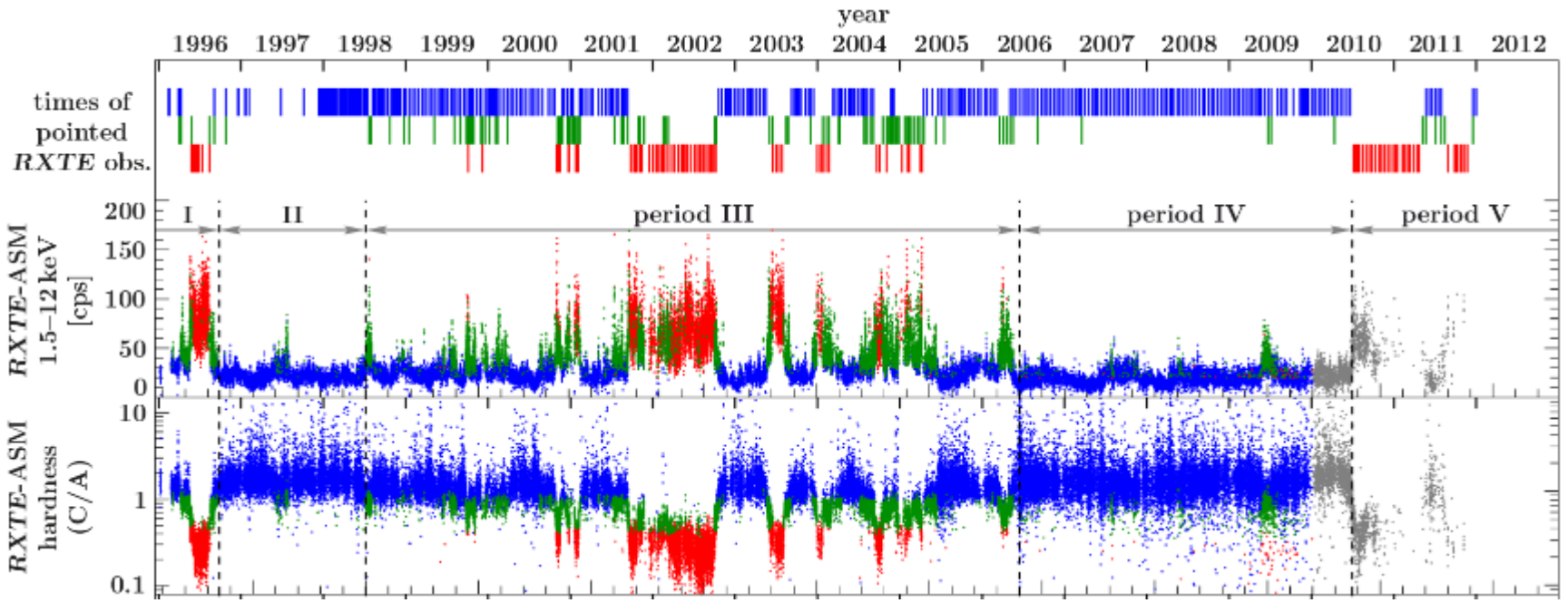




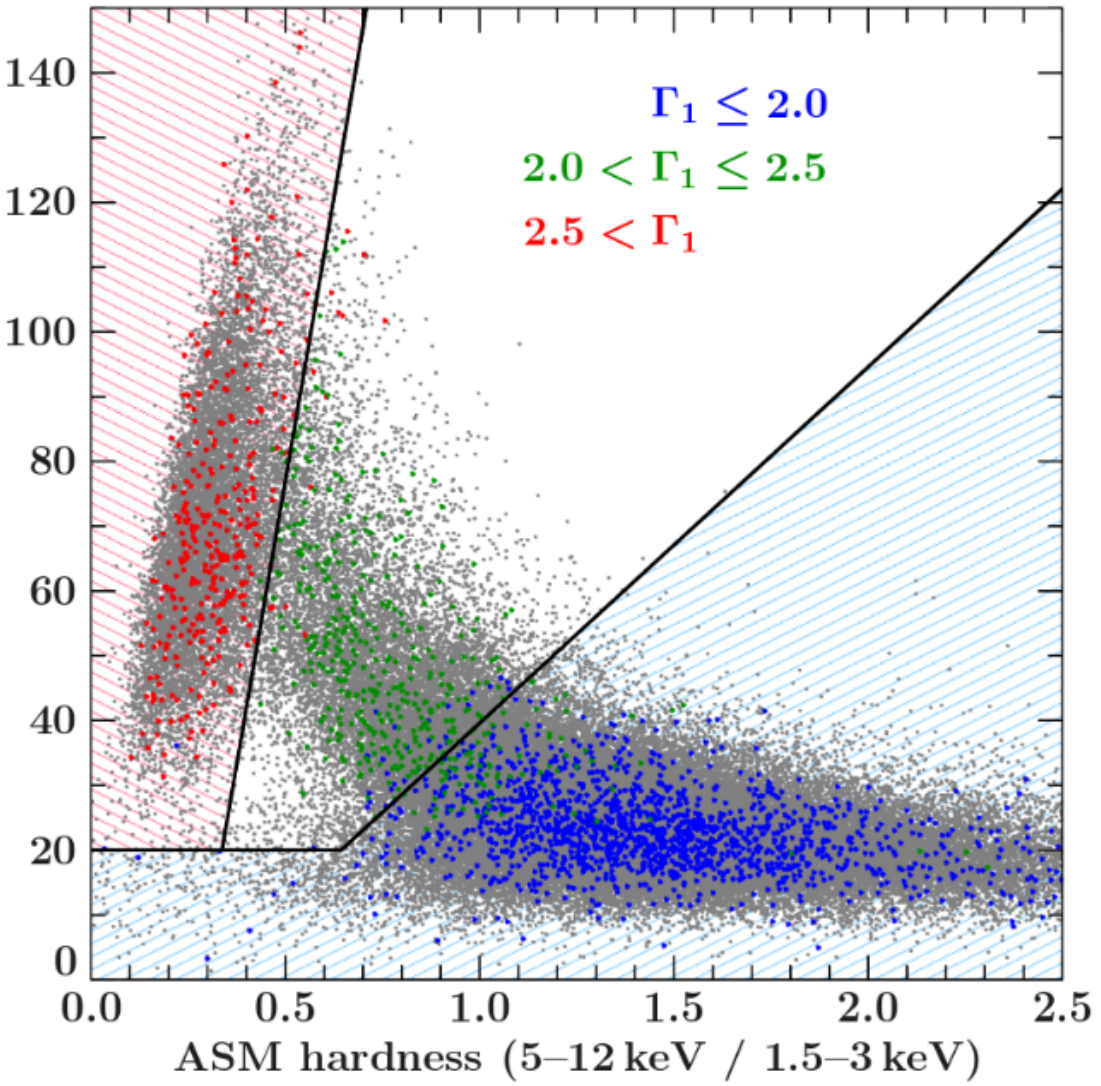
State Frequency Comparison



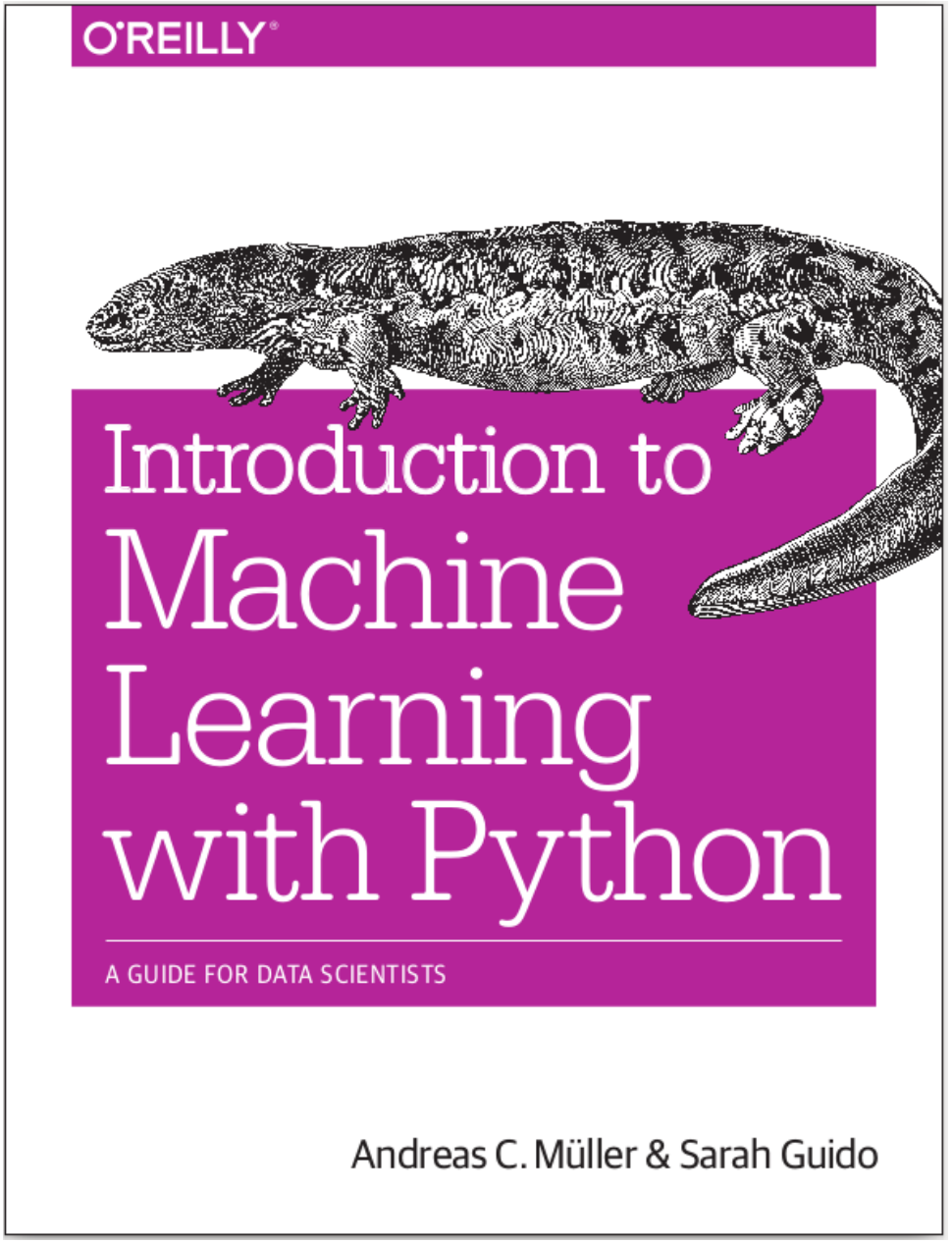
Finding States for CYG-X1



Human Annotations



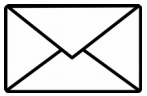
Demo
(Nicolas Goix)


O'REILLY
Introduction to
Machine
Learning
with Python
A GUIDE FOR DATA SCIENTISTS
Andreas C. Müller & Sarah Guido

Thank you.


@amuellerml


@amueller


amueller@nyu.edu


http://amueller.io