Visualizing the Model Selection Process

CEB Day 5: February 3, 2017


54th SAPPORO SNOW FESTIVAL
NETFLIX
So I read about this great ML model

$$R \approx P \times Q^T = \hat{R}$$

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{k} p_{ik} q_{kj}$$

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{k} p_{ik} q_{kj})^2$$

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2 e_{ij} q_{kj} \qquad p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij} q_{kj}$$

$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2 e_{ij} q_{ik} \qquad q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij} p_{ik}$$

$$E = \sum_{(p_i, q_j, r_{ij})} (r_{ij} - \sum_{k=1}^{k} p_{ik} q_{kj})^2$$

Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 42.8 (2009): 30-37.

```
def nnmf(R, k=2, steps=5000, alpha=0.0002, beta=0.02):

    n, m = R.shape
    P = np.random.rand(n,k)
    Q = np.random.rand(m,k).T

    for step in range(steps):
        for idx in range(n):
            for jdx in range(m):
                if R[idx][jdx] > 0:
                    eij = R[idx][jdx] - np.dot(P[idx,:], Q[:,jdx])
                    for kdx in range(K):
                        P[idx][kdx] = P[idx][kdx] + alpha * (2 * eij * Q[kdx][jdx] - beta * P[idx][kdx])
                        Q[kdx][jdx] = Q[kdx][jdx] + alpha * (2 * eij * P[idx][kdx] - beta * Q[kdx][jdx])

        e  = 0
        for idx in range(n):
            for jdx in range(m):
                if R[idx][jdx] > 0:
                    e += (R[idx][jdx] - np.dot(P[idx,:], Q[:,jdx])) **  2

        if e < 0.001:
            break

    return P, Q.T
```


easy
Life with
Scikit-Learn

```
from sklearn.decomposition import NMF
model = NMF(n_components=2, init='random', random_state=0)
model.fit(R)
```

```python
from sklearn.decomposition import NMF, TruncatedSVD, PCA

models = [
    NMF(n_components=2, init='random', random_state=0),
    TruncatedSVD(n_components=2),
    PCA(n_components=2),
]

for model in models:
    model.fit(R)
```

So now I’m all

Made Possible by the Scikit-Learn API

```
class Estimator(object):

    def fit(self, X, y=None):
        """
        Fits estimator to data.
        """
        # set state of self
        return self

    def predict(self, X):
        """
        Predict response of X
        """
        # compute predictions pred
        return pred
```

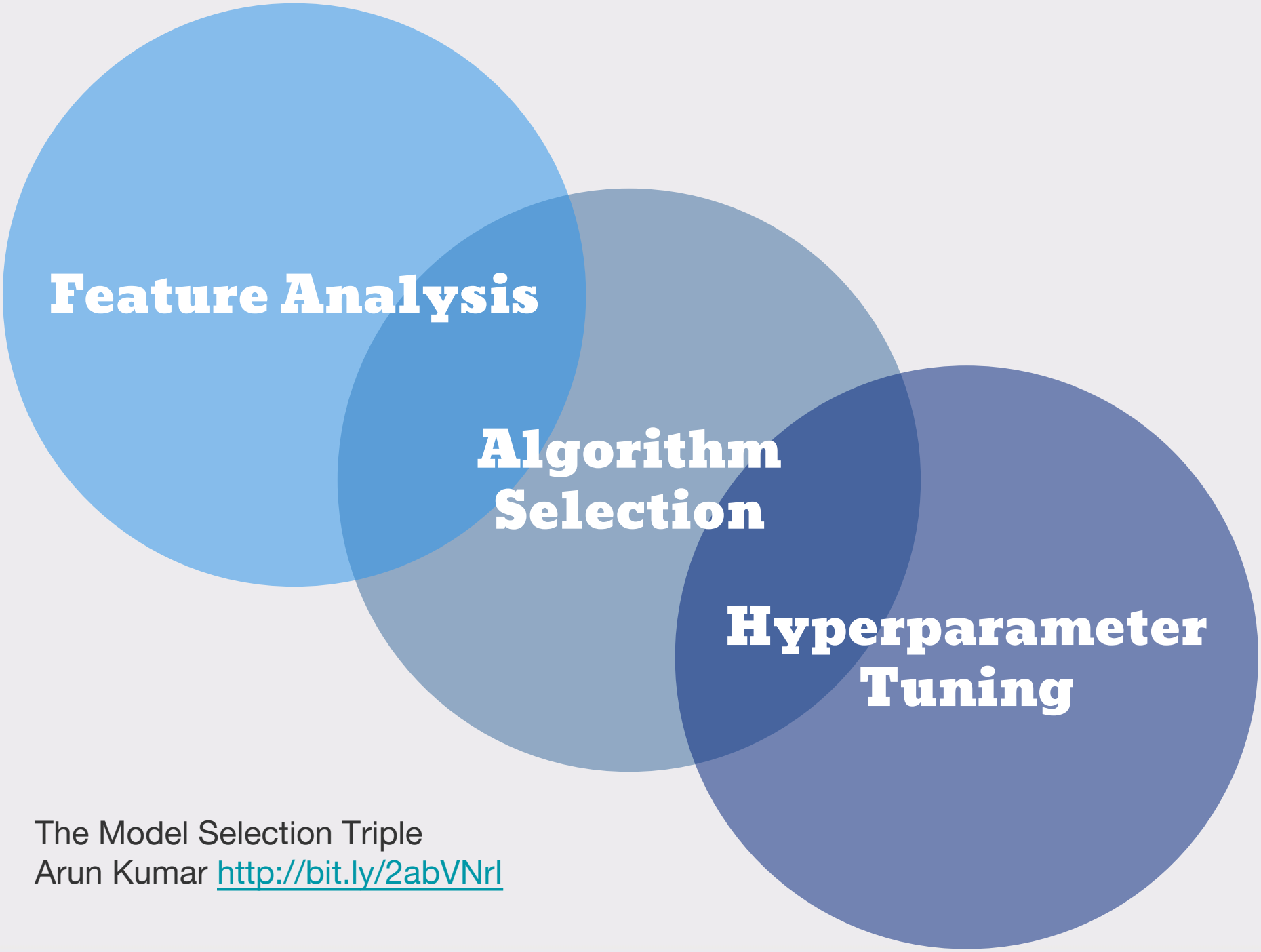
```
class Transformer(Estimator):

    def transform(self, X):
        """
        Transforms the input data.
        """
        # transform X to X_prime
        return X_prime
```

```
class Pipeline(Transfomer):

    @property
    def named_steps(self):
        """
        Returns a sequence of estimators
        """
        return self.steps

    @property
    def _final_estimator(self):
        """
        Terminating estimator
        """
        return self.steps[-1]
```

Buitinck, Lars, et al. "API design for machine learning software: experiences from the scikit-learn project." arXiv preprint arXiv:1309.0238 (2013).


Algorithm design
stays in the hands of
Academia


Wizardry When
Applied



The Model Selection Triple
Arun Kumar http://bit.ly/2abVNrI

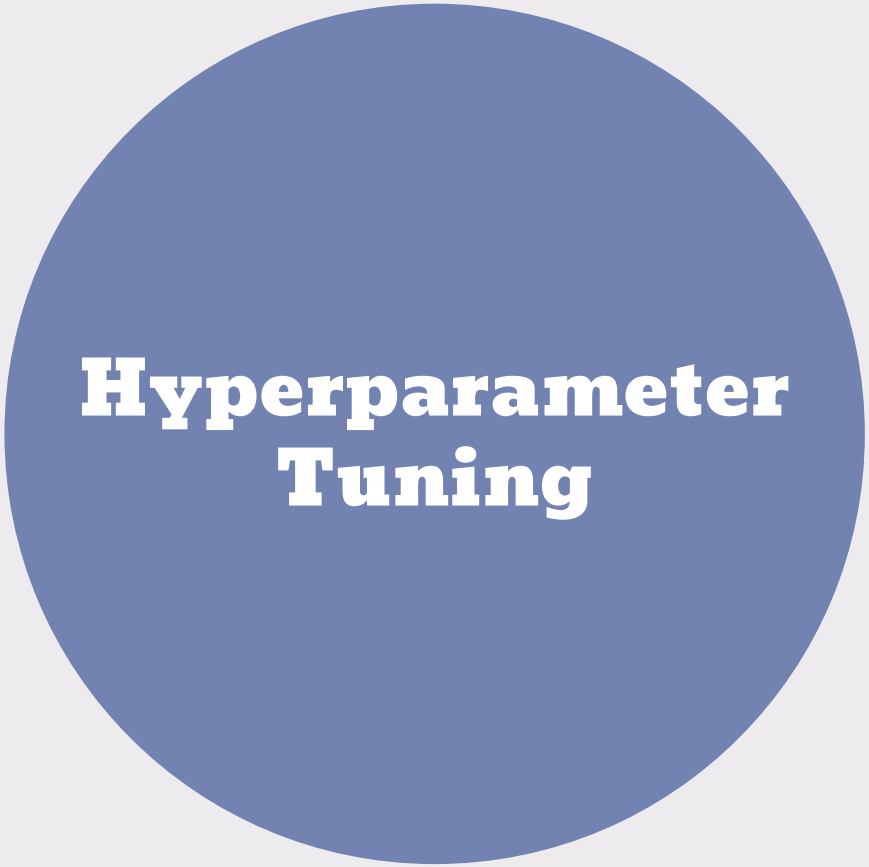
The Model Selection Triple

- Define a bounded, high dimensional feature space that can be effectively modeled.
- Transform and manipulate the space to make modeling easier.
- Extract a feature representation of each instance in the space.

The Model Selection Triple

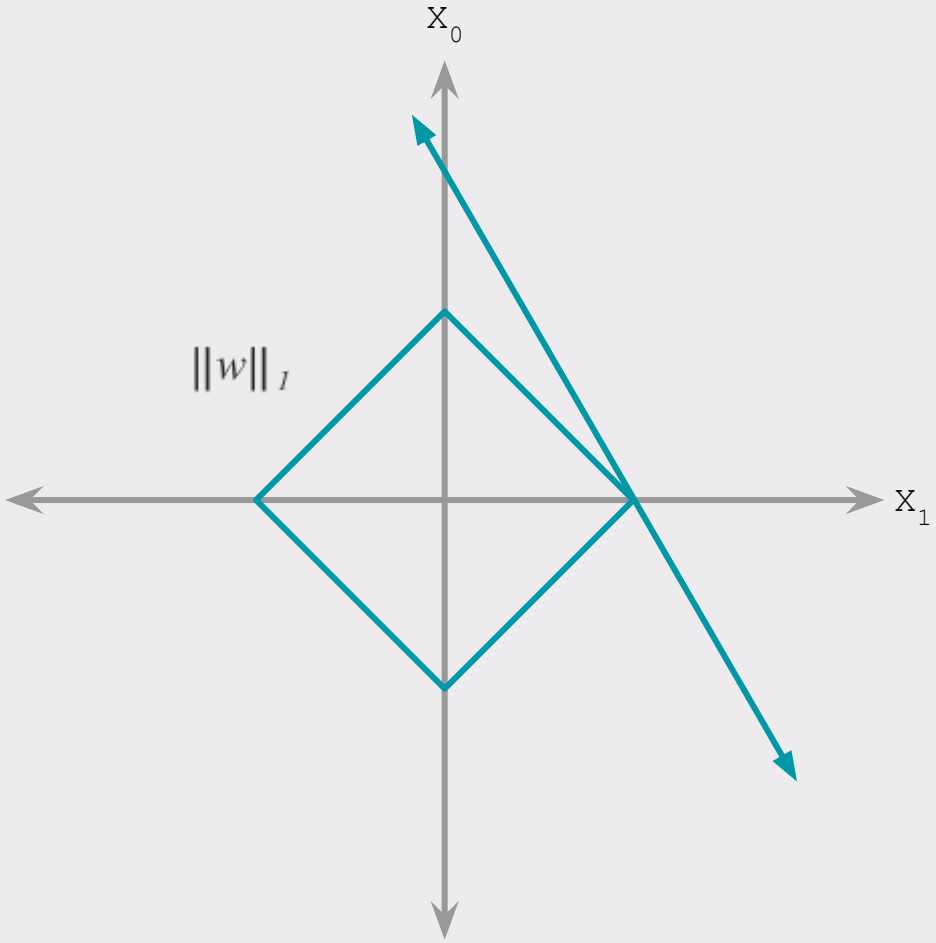
- Select a model family that best/correctly defines the relationship between the variables of interest.
- Define a model form that specifies exactly how features interact to make a prediction.
- Train a fitted model by optimizing internal parameters to the data.

The Model Selection Triple



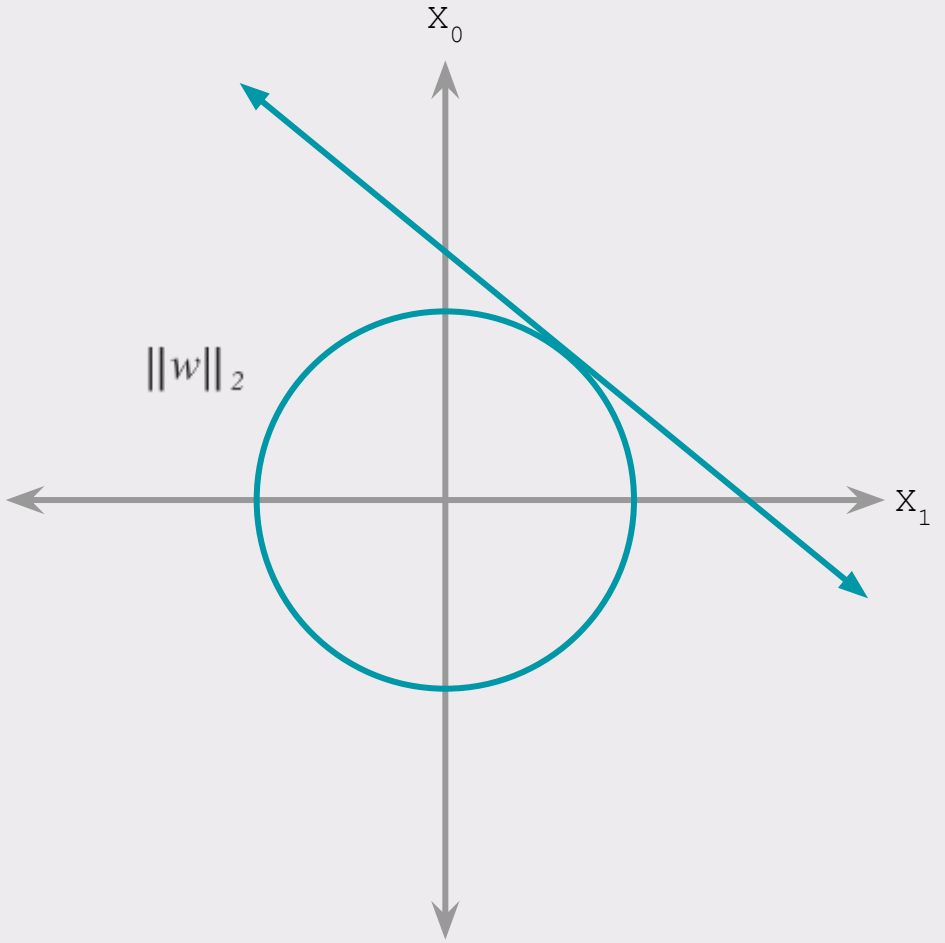
- Evaluate how the model form is interacting with the feature space.
- Identify hyperparameters (parameters that affect training or the prior, not prediction)
- Tune the fitting and prediction process by modifying these params.


Can it be automated?



L1 Normalization

Possibility that a feature is eliminated by setting its coefficient equal to zero.



L2 Normalization

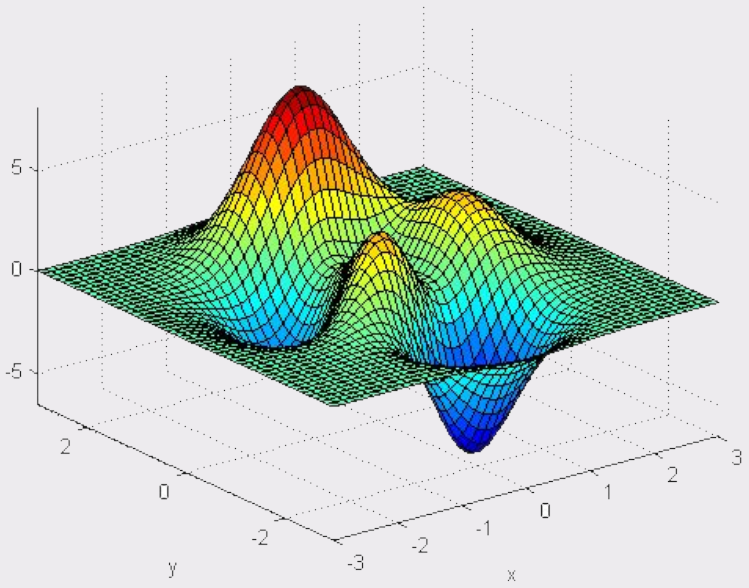
Features are kept balanced by minimizing the relative change of coefficients during learning.

Regularization is a form of automatic feature analysis.

Automatic Model Selection Criteria

```
from sklearn.cross_validation import KFold

kfolds = KFold(n=len(X), n_folds=12)

scores = [
    model.fit(
        X[train], y[train]
    ).score(
        X[test], y[test]
    )

    for train, test in kfolds
]
```

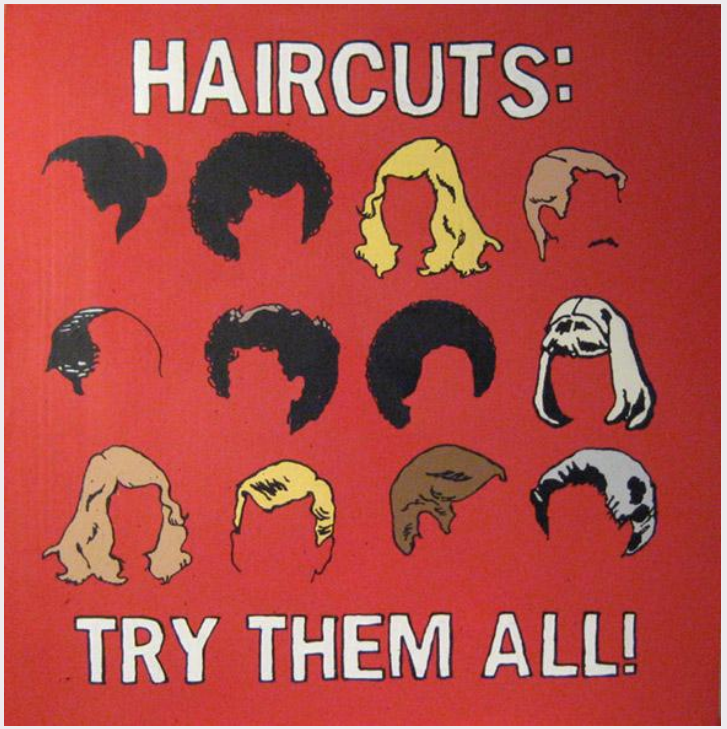
F1



R^2

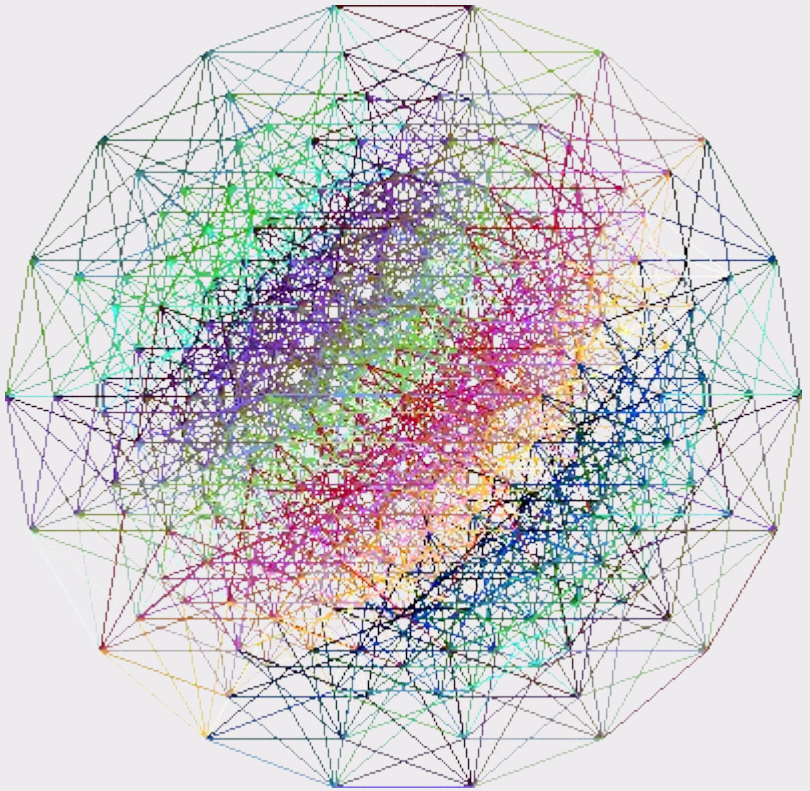
Automatic Model Selection: Try Them All!

```
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn import cross_validation as cv

classifiers = [
    KNeighborsClassifier(5),
    SVC(kernel="linear", C=0.025),
    RandomForestClassifier(max_depth=5),
    AdaBoostClassifier(),
    GaussianNB(),
]

kfold  = cv.KFold(len(X), n_folds=12)
max([
    cv.cross_val_score(model, X, y, cv=kfold).mean
    for model in classifiers
])
```



Automatic Model Selection: Search Param Space

```
from sklearn.feature_extraction.text import *
from sklearn.linear_model import SGDClassifier
from sklearn.grid_search import GridSearchCV
from sklearn.pipeline import Pipeline

pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('model', SGDClassifier()),
])

parameters = {
    'vect__max_df': (0.5, 0.75, 1.0),
    'vect__max_features': (None, 5000, 10000),
    'tfidf__use_idf': (True, False),
    'tfidf__norm': ('l1', 'l2'),
    'model__alpha': (0.00001, 0.000001),
    'model__penalty': ('l2', 'elasticnet'),
}

search = GridSearchCV(pipeline, parameters)
search.fit(X, y)
```



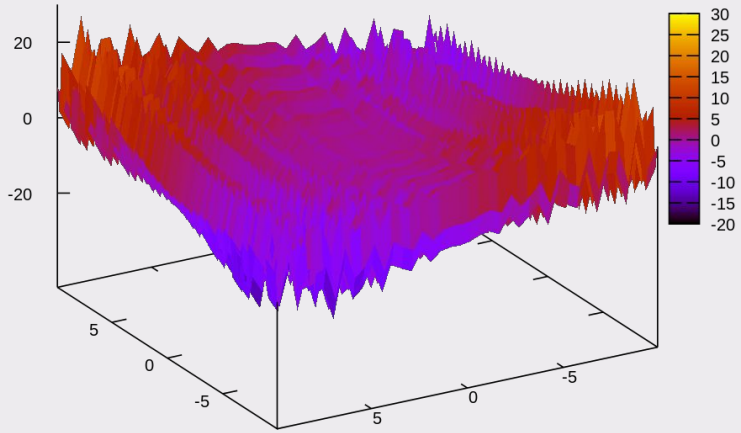

Maybe not so Wizard?

Automatic Model Selection: Search?

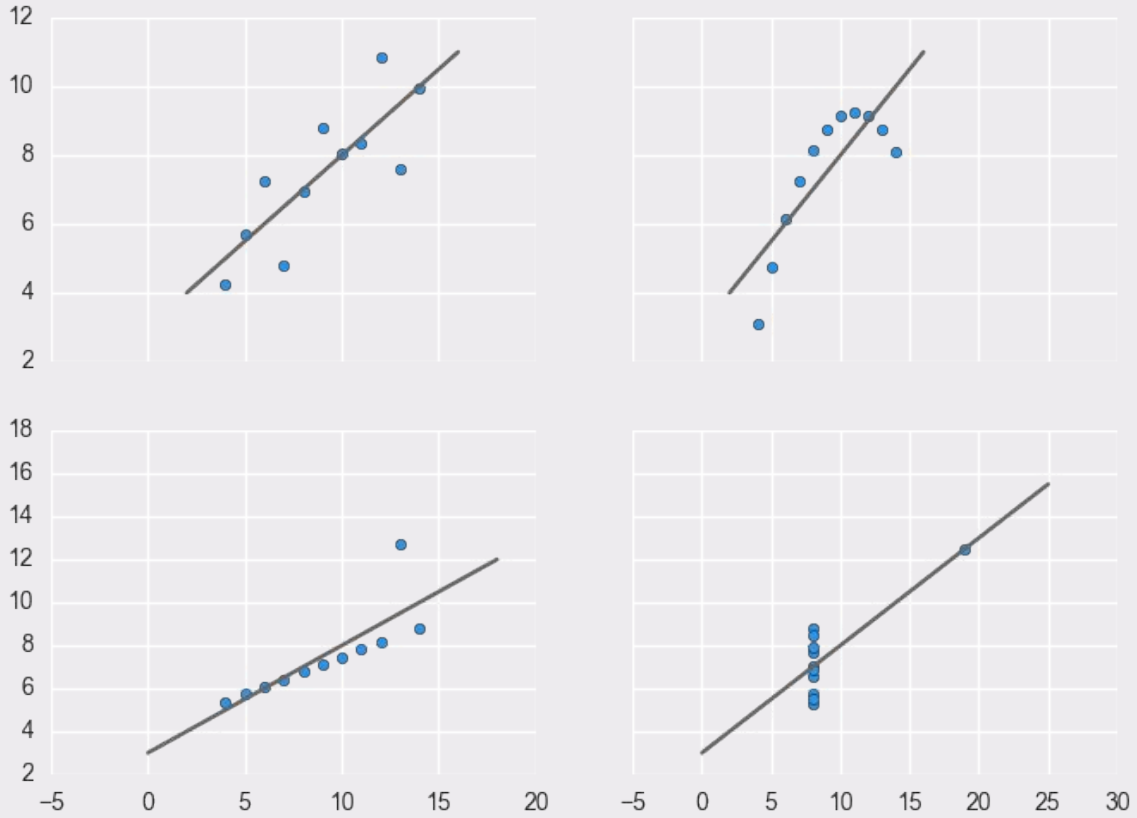
Search is difficult particularly in high dimensional space.

Even with techniques like genetic algorithms or particle swarm optimization, there is no guarantee of a solution.

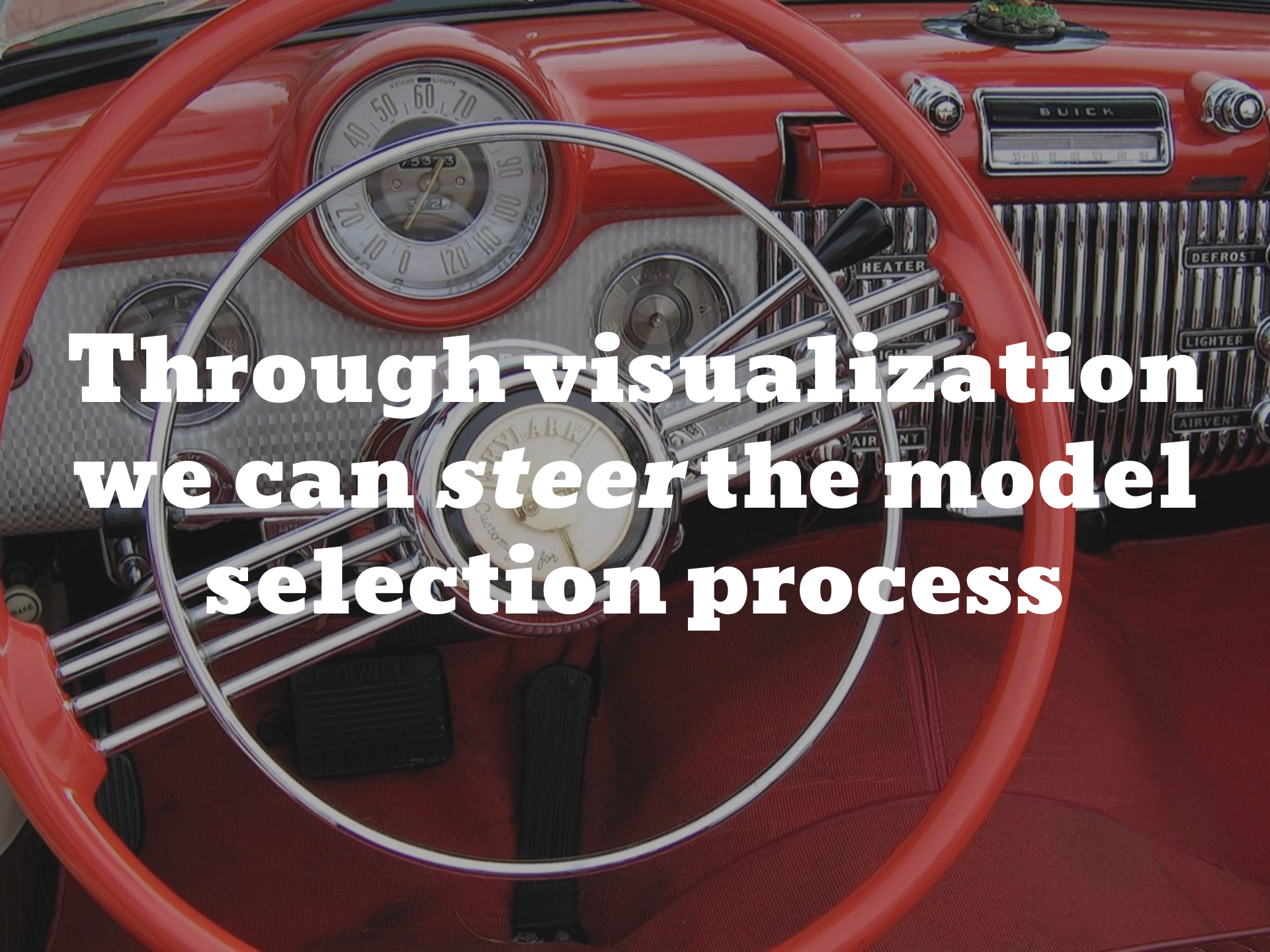
As the search space gets larger, the amount of time increases exponentially.



Anscombe's Quartet

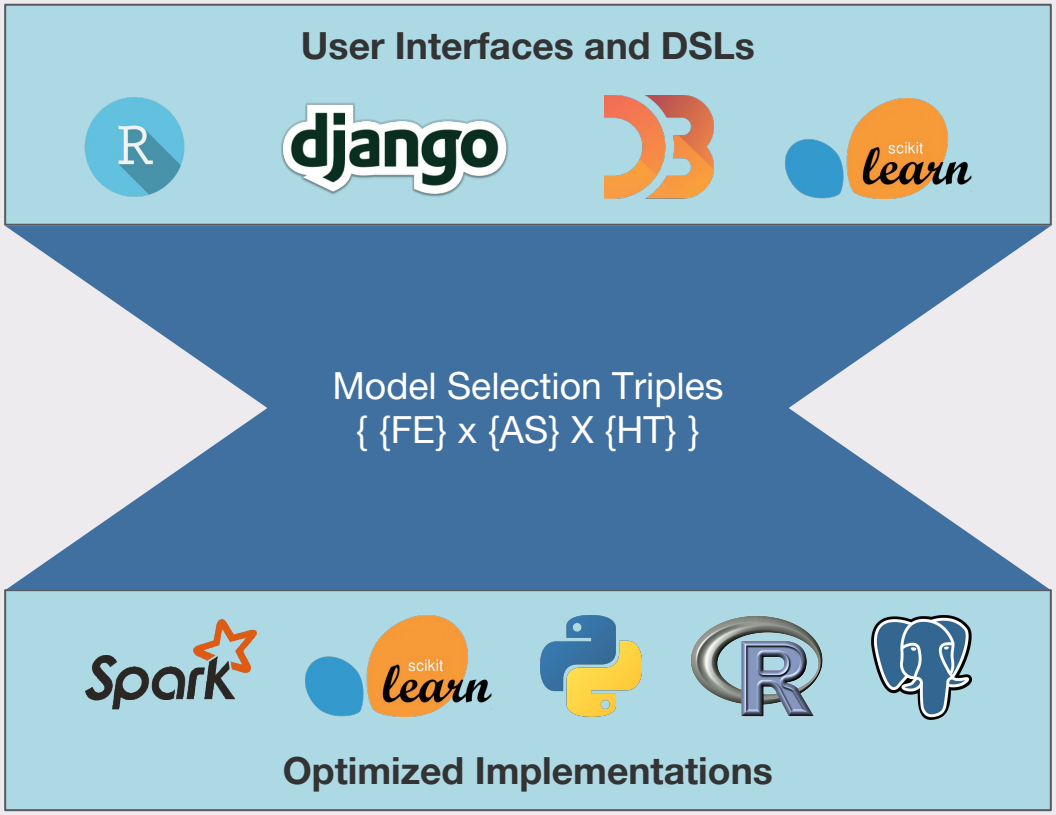


Anscombe, Francis J. "Graphs in statistical analysis."
The American Statistician 27.1 (1973): 17-21.



Through visualization we can *steer* the model selection process

Model Selection Management Systems

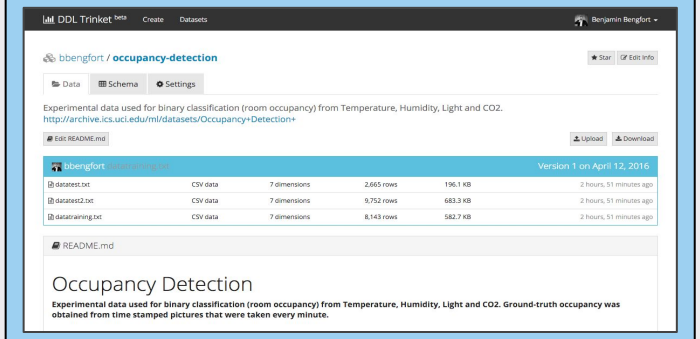


Kumar, Arun, et al. "Model selection management systems: The next frontier of advanced analytics." ACM SIGMOD Record 44.4 (2016): 17-22.

Can we visualize *machine learning?*

Data Management

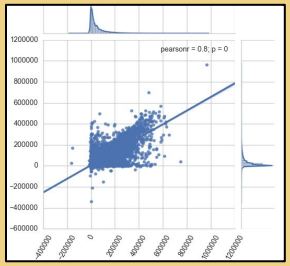
Wrangling
Standardization
Normalization
Selection & Joins

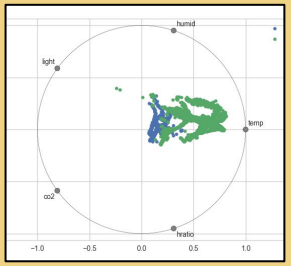


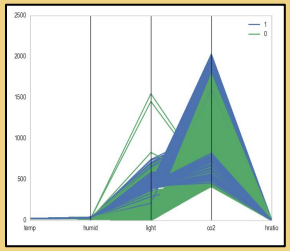
Feature Analysis

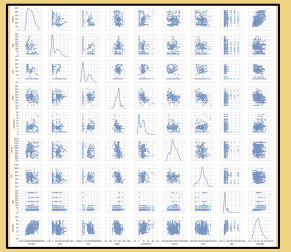
Feature Selection

Feature Analysis







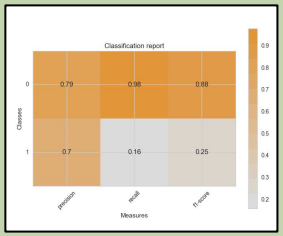


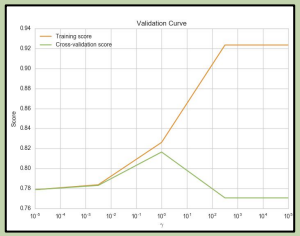
Model Storage

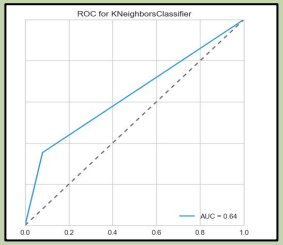
Revisit Features

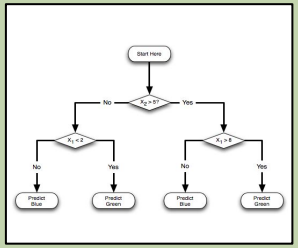
Model Selection

Model Evaluation + Hyperparameter Tuning







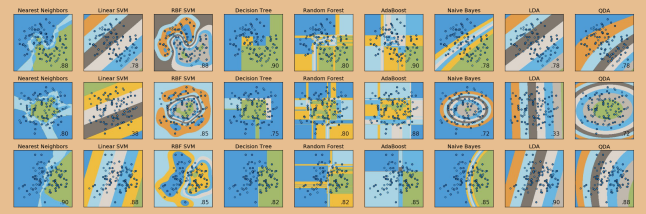


Initial Model

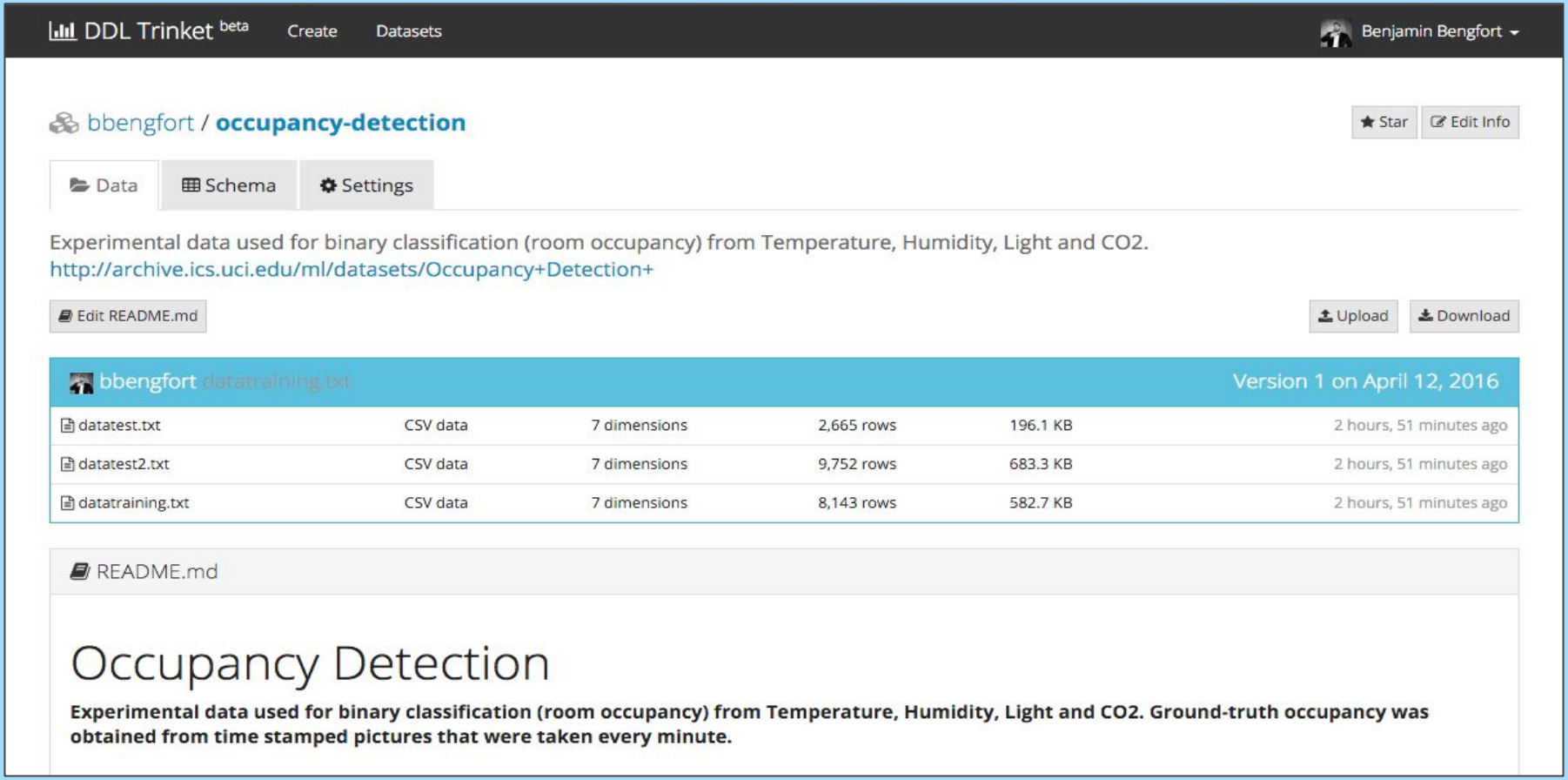
Iterate!

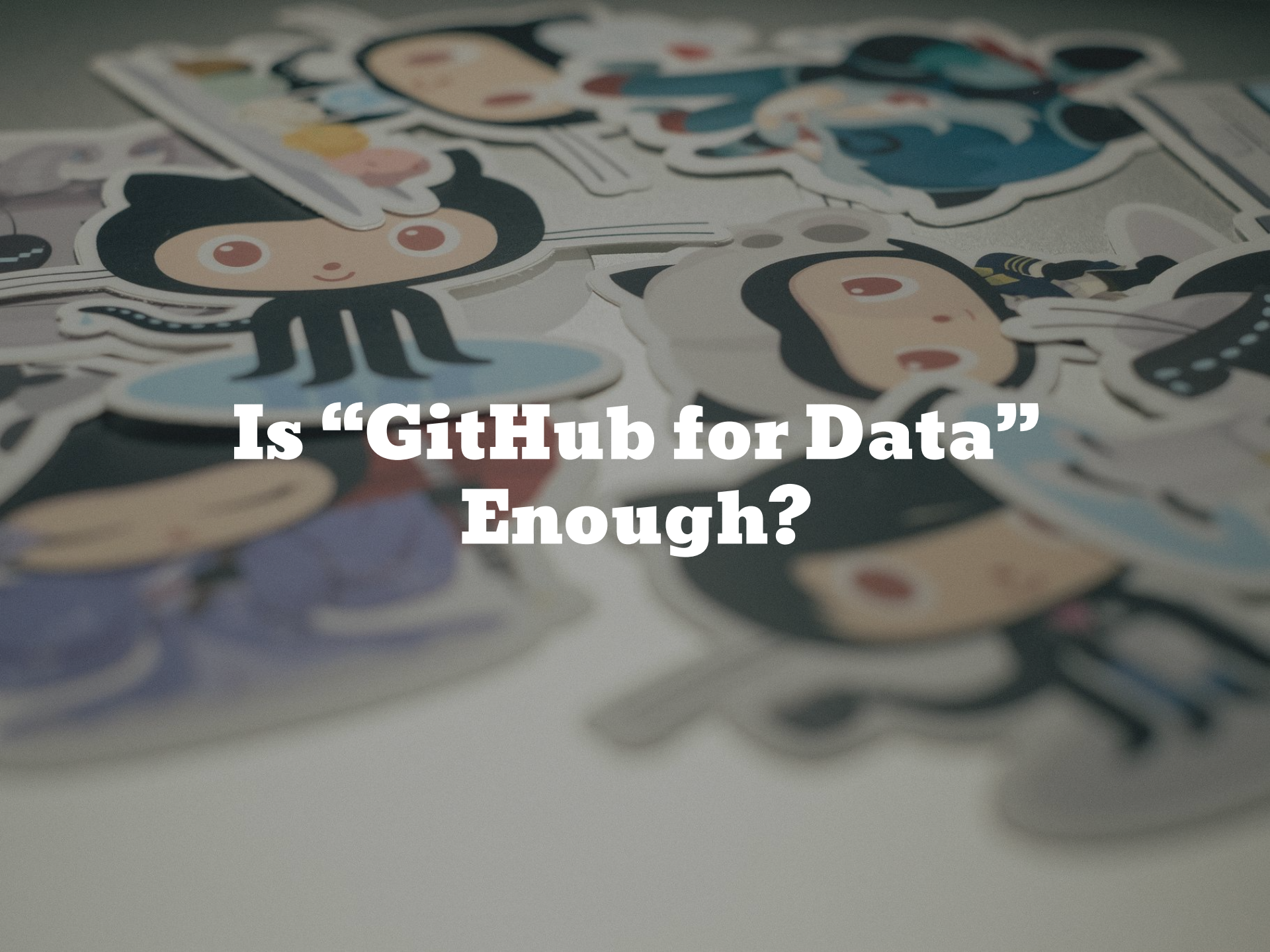
Model Selection

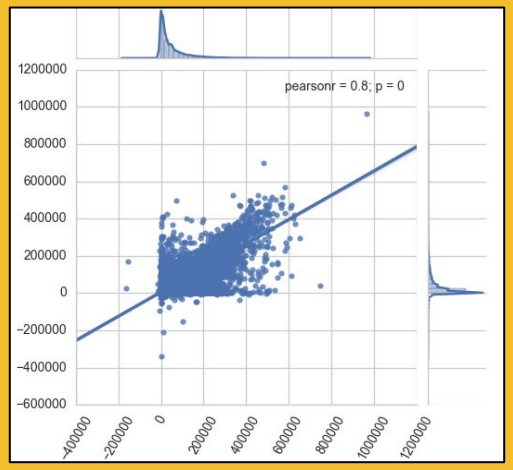
Linear Models	Nearest Neighbors	SVM
Ensemble	Trees	Bayes

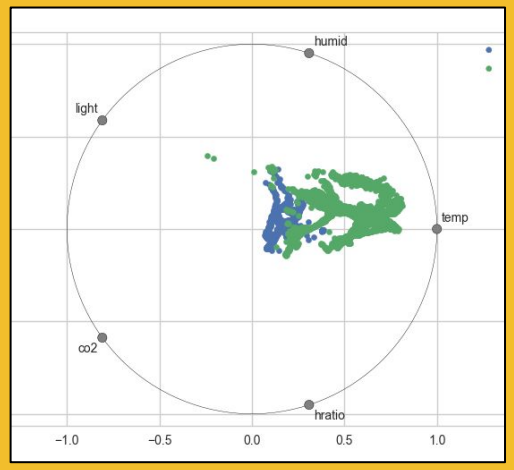


Data and Model Management

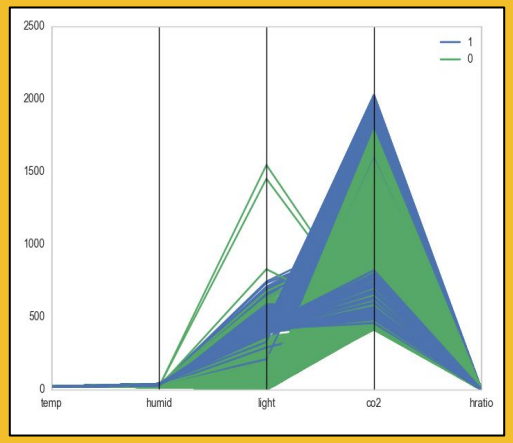


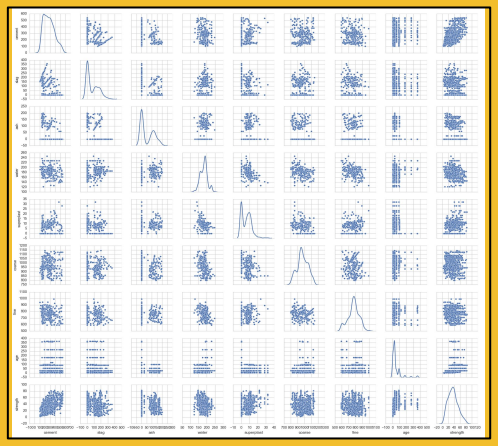

Is “GitHub for Data”
Enough?

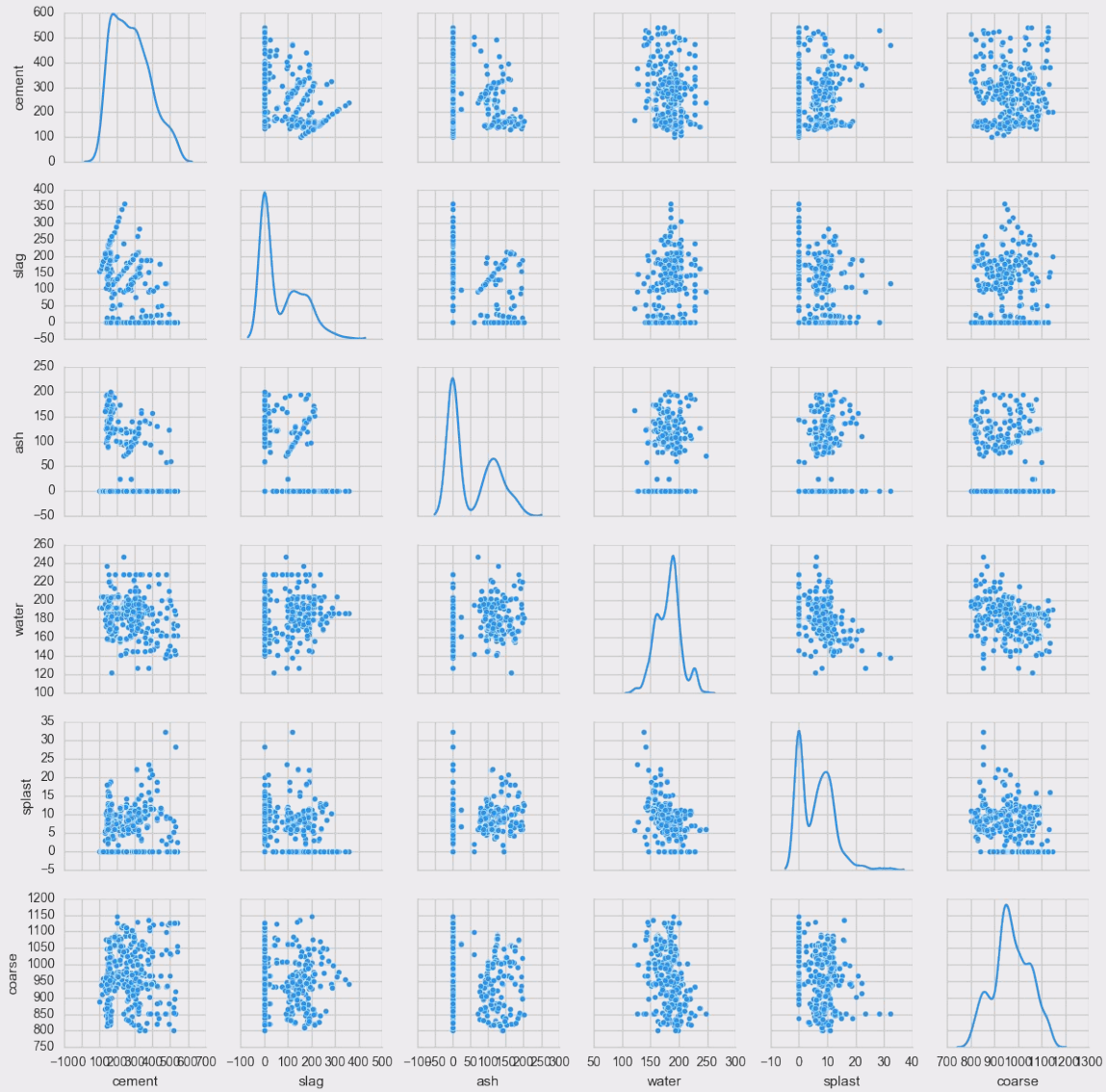




Visualizing Feature Analysis





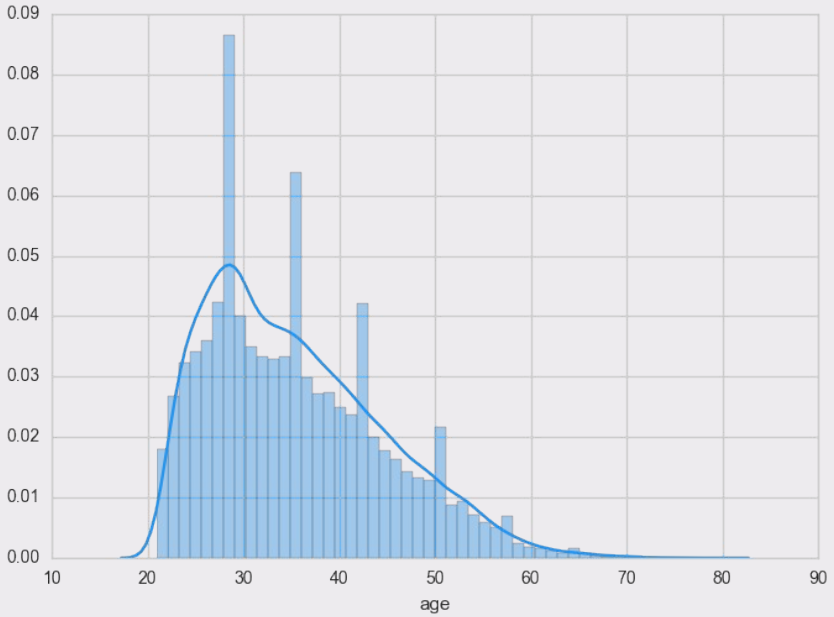


SPLOM (Scatterplot Matrices)

Visual Rank by Feature: 1 Dimension

Rank by:

1. Normality of distribution (Shapiro-Wilk and Kolmogorov-Smirnov)
2. Uniformity of distribution (entropy)
3. Number of potential outliers
4. Number of hapaxes
5. Size of gap

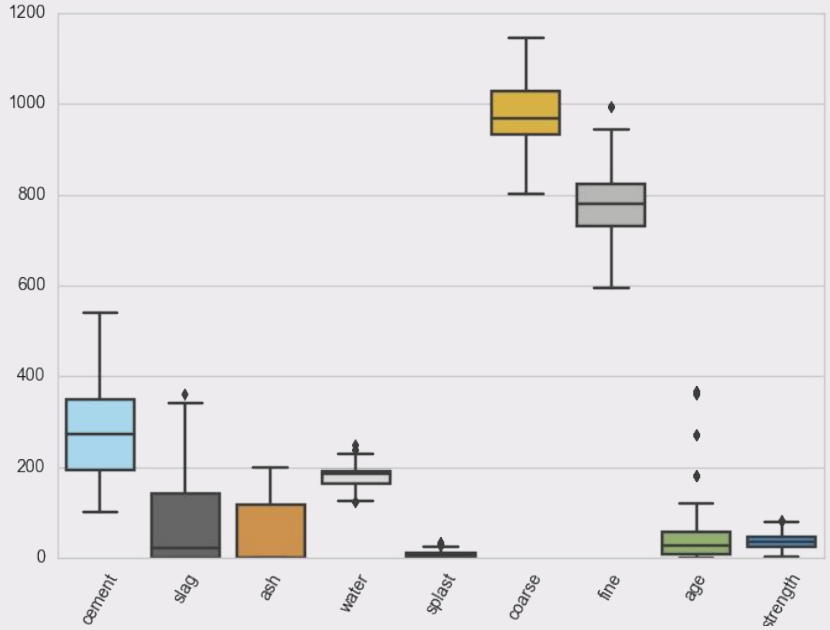


Seo, Jinwook, and Ben Shneiderman. "A rank-by-feature framework for interactive exploration of multidimensional data." Information visualization 4.2 (2005): 96-113.

Visual Rank by Feature: 1 Dimension

Rank by:

1. Normality of distribution (Shapiro-Wilk and Kolmogorov-Smirnov)
2. Uniformity of distribution (entropy)
3. Number of potential outliers
4. Number of hapaxes
5. Size of gap

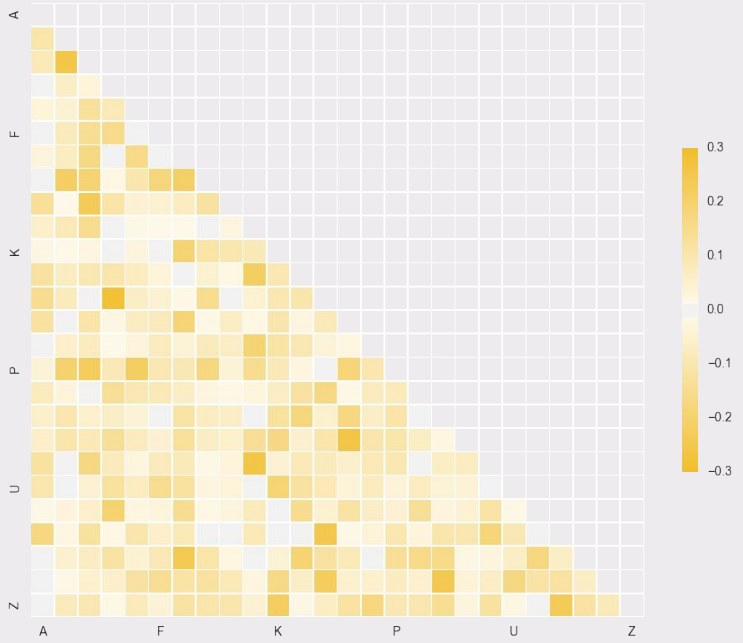


Seo, Jinwook, and Ben Shneiderman. "A rank-by-feature framework for interactive exploration of multidimensional data." Information visualization 4.2 (2005): 96-113.

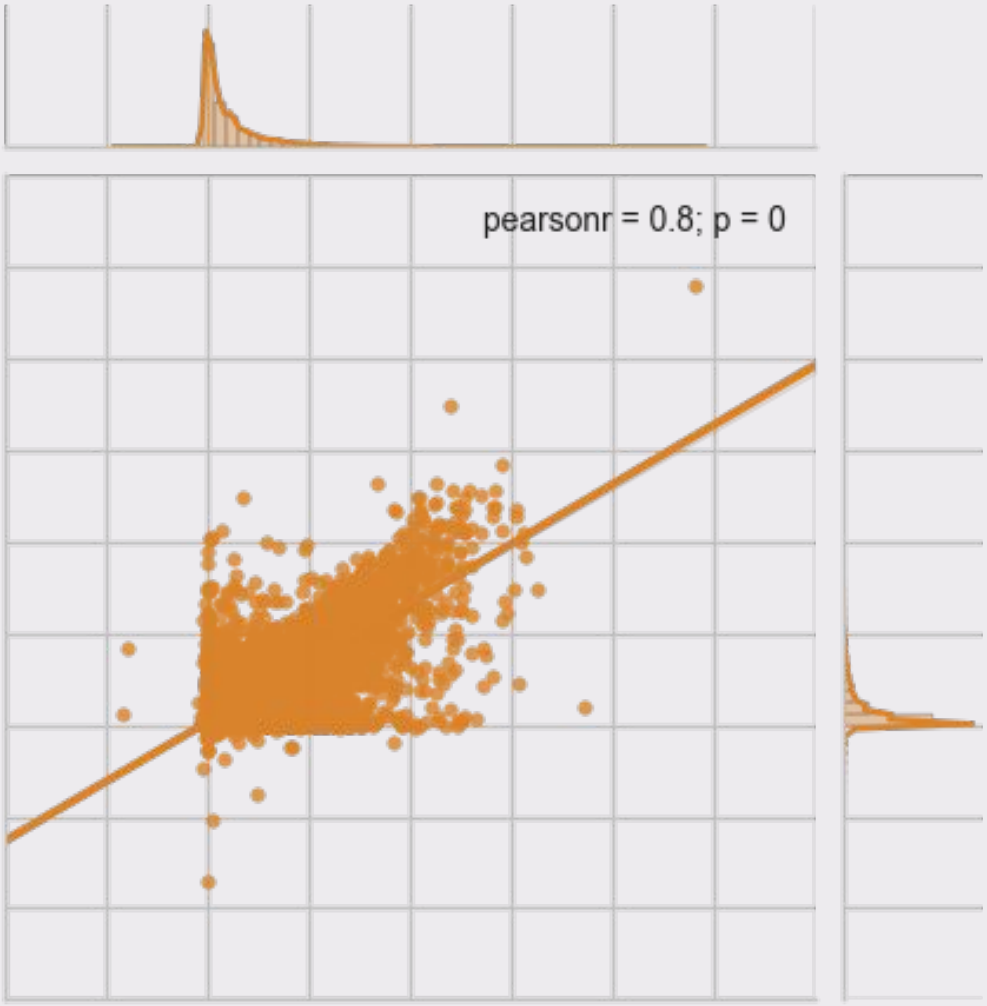
Visual Rank by Feature: 2 Dimensions

Rank by:

1. Correlation Coefficient (Pearson, Spearman)
2. Least-squares error
3. Quadracity
4. Density based outlier detection.
5. Uniformity (entropy of grids)
6. Number of items in the most dense region of the plot.

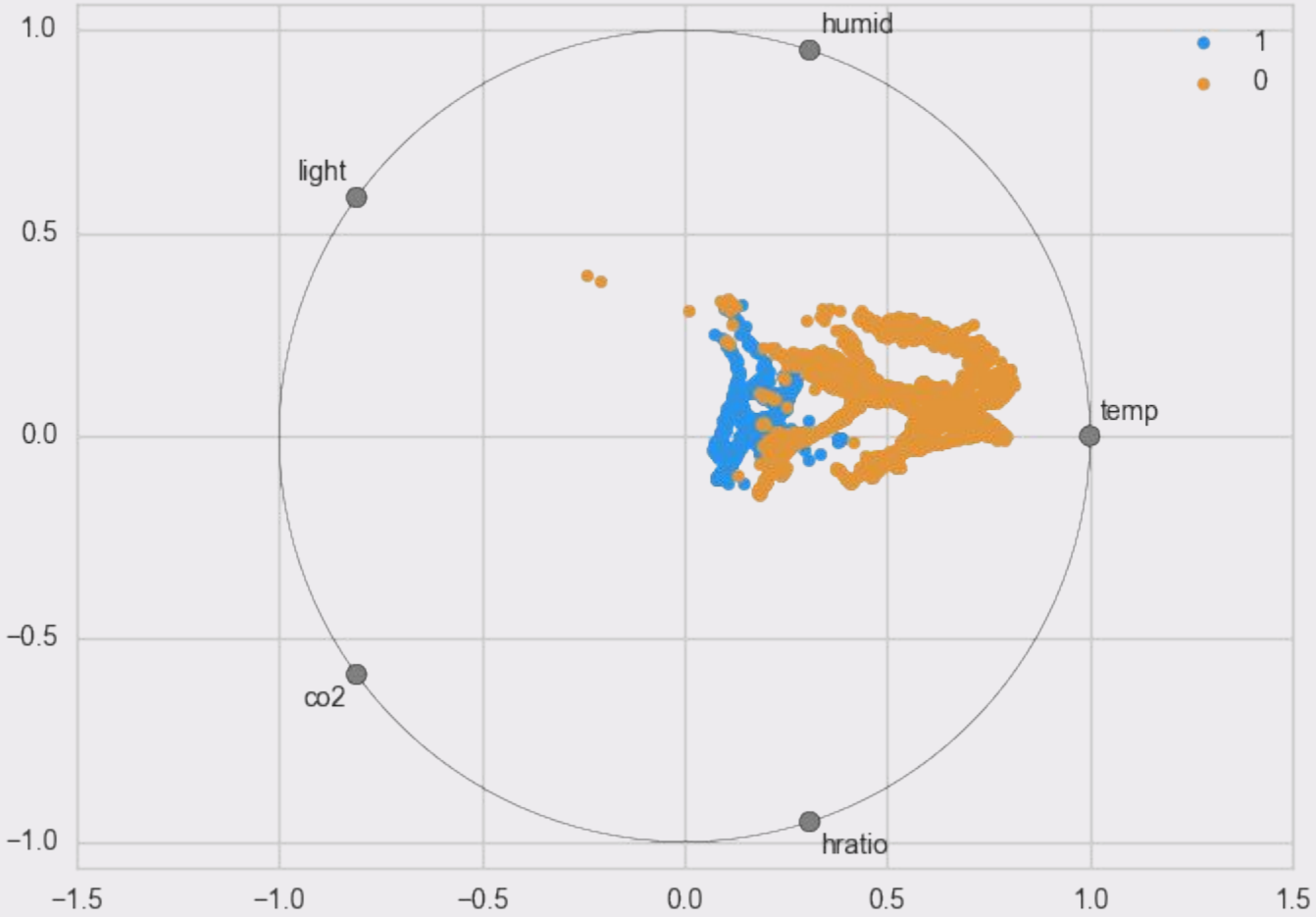


Seo, Jinwook, and Ben Shneiderman. "A rank-by-feature framework for interactive exploration of multidimensional data." Information visualization 4.2 (2005): 96-113.

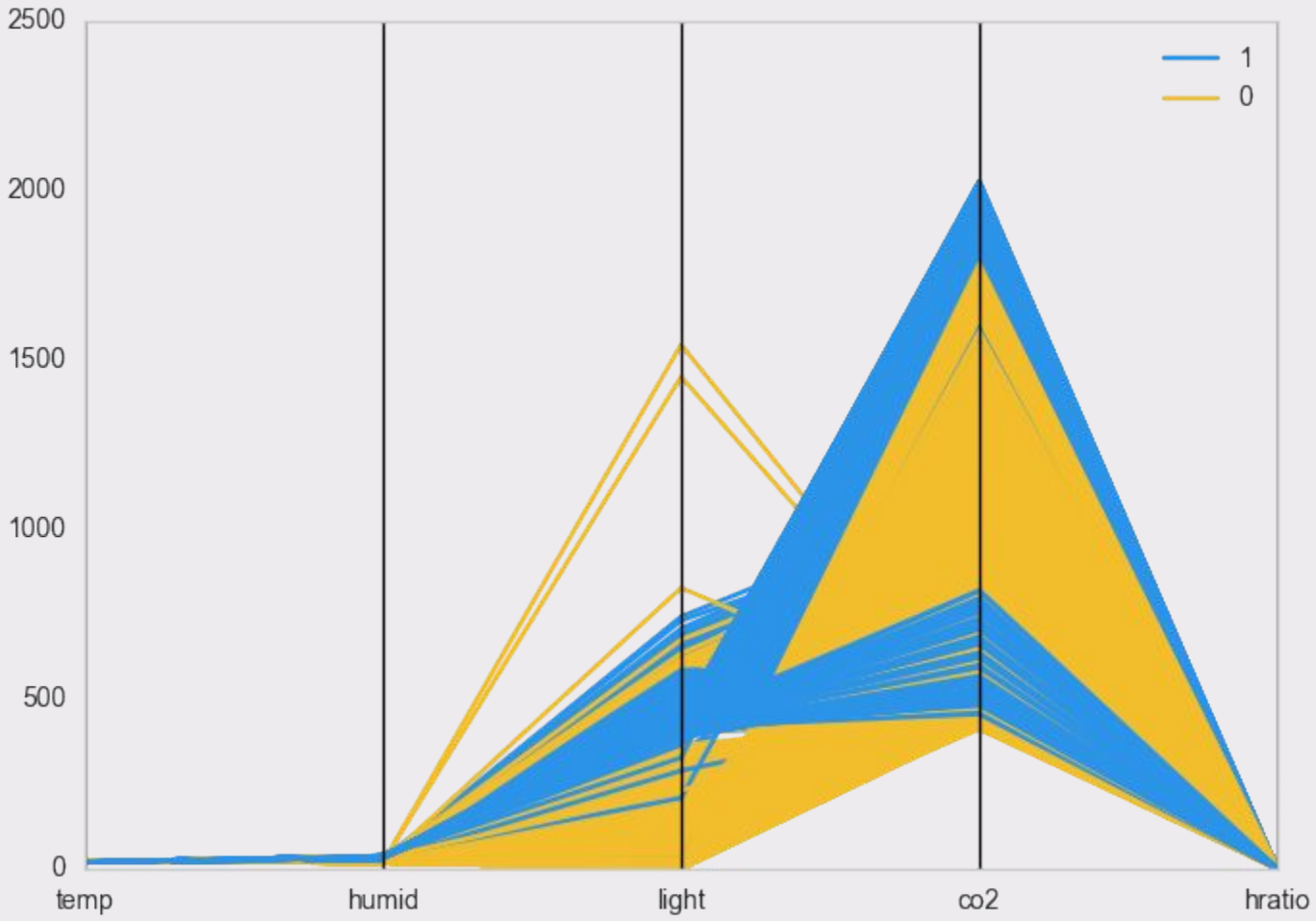


Joint Plots: Diving Deeper after Rank by Feature
Special thanks to Seaborn for doing statistical visualization right!

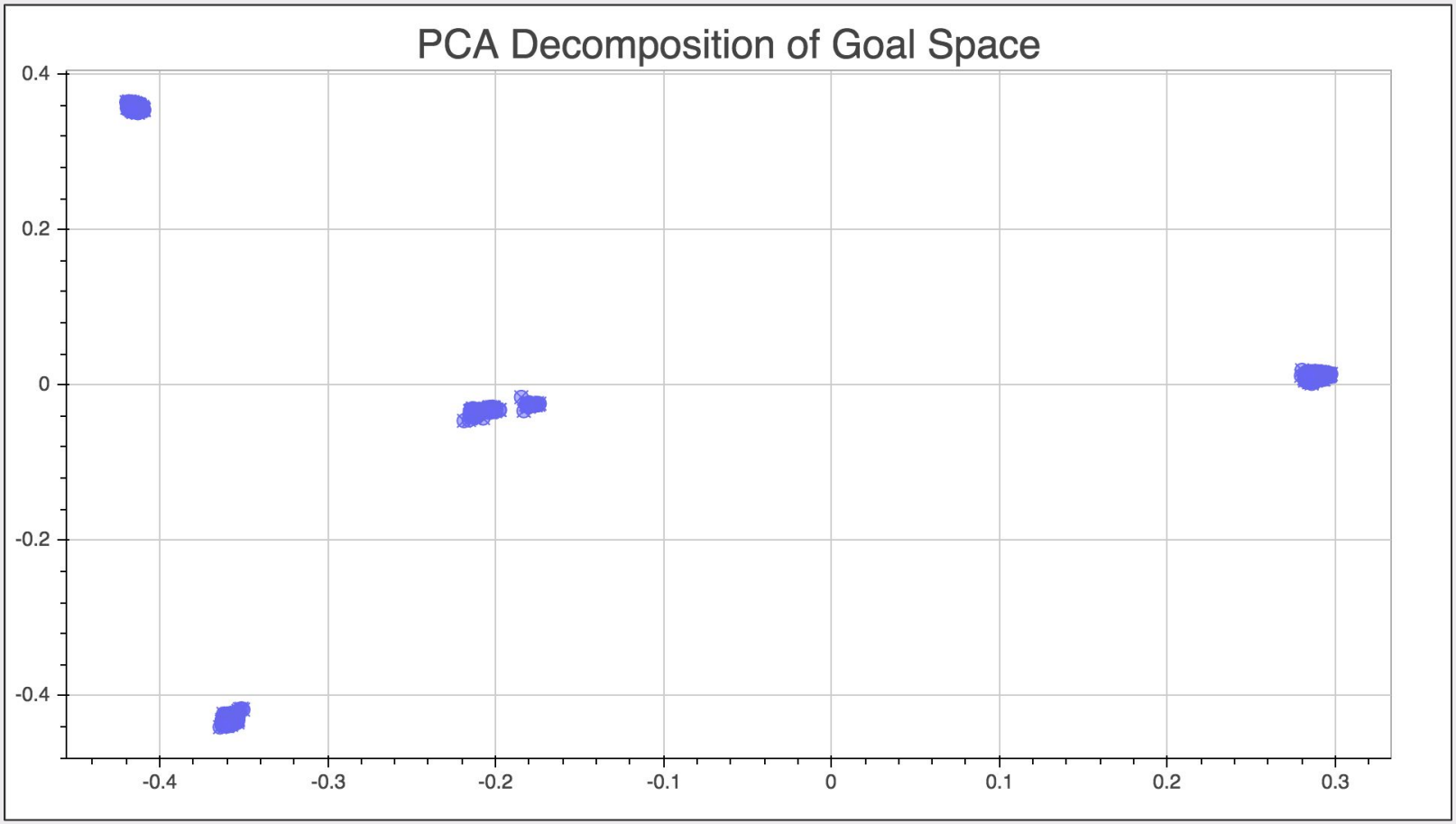

Detecting Separablity



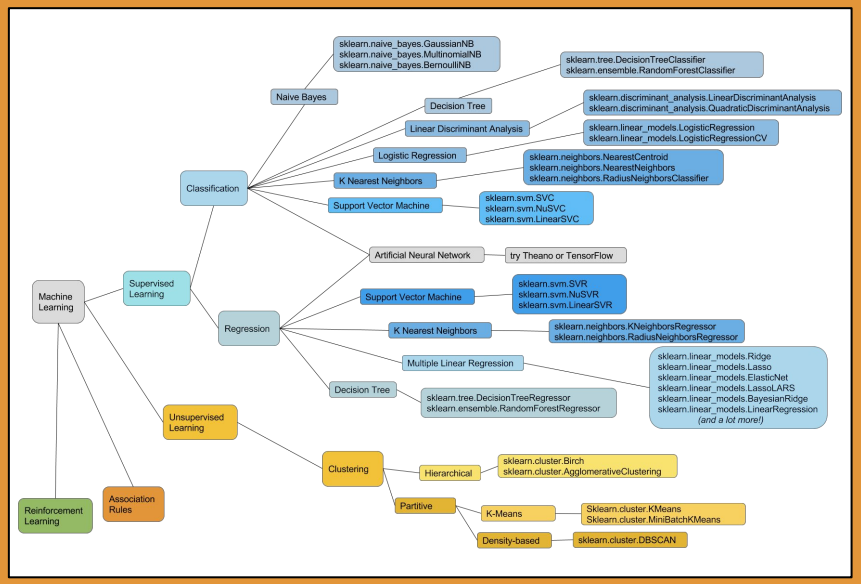
Radviz: Radial Visualization

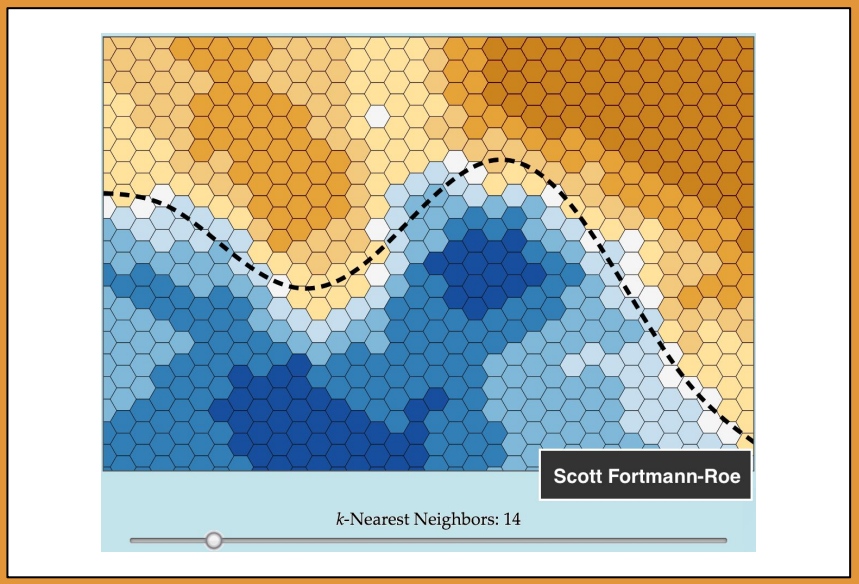


Parallel Coordinates

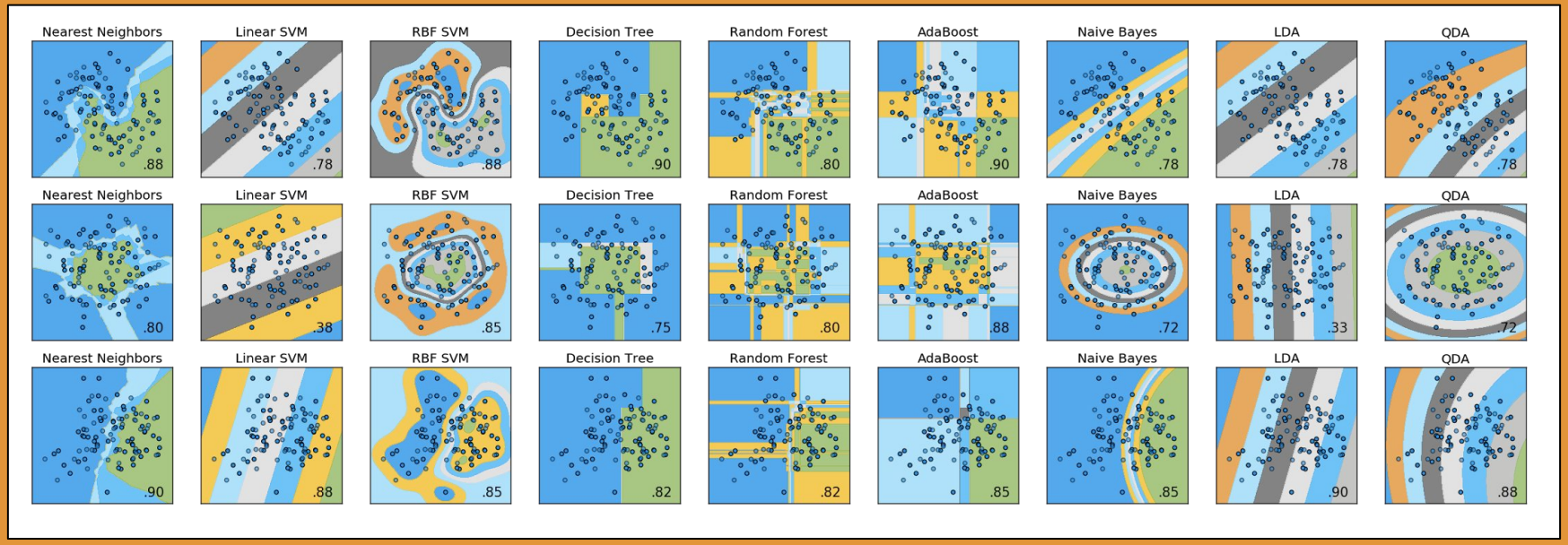


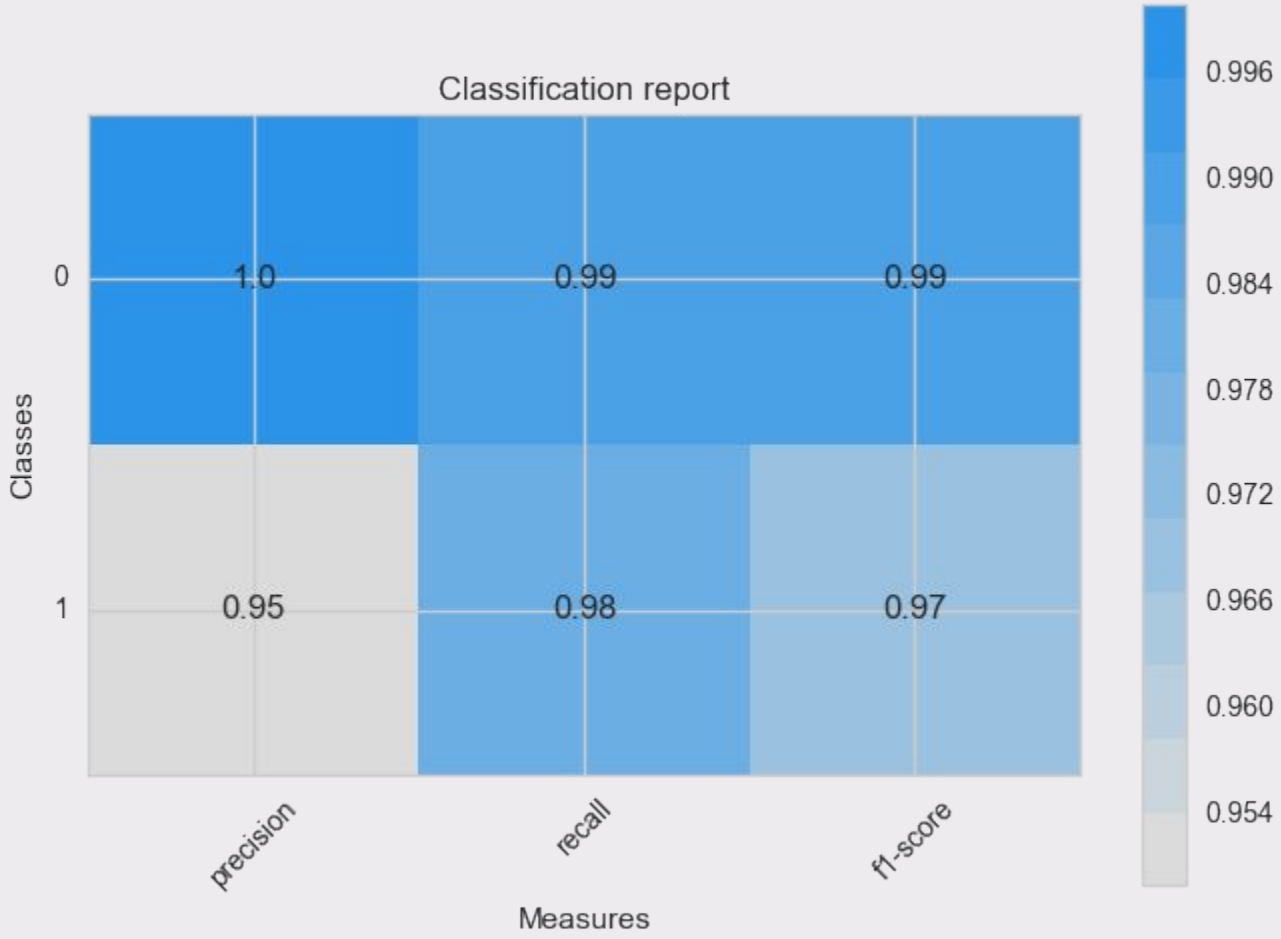
Decomposition (PCA, SVD) of Feature Space



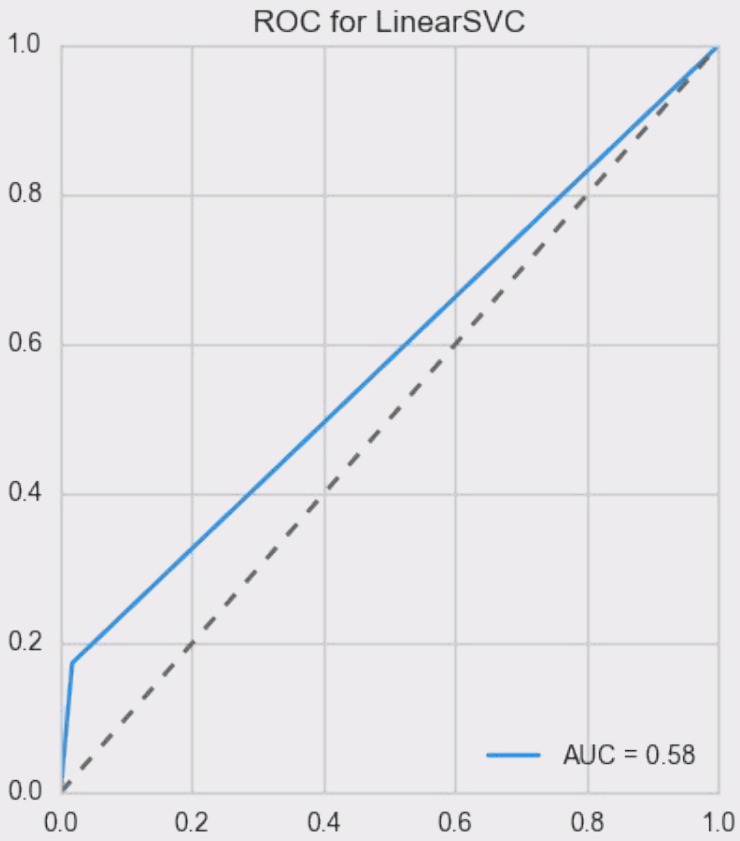


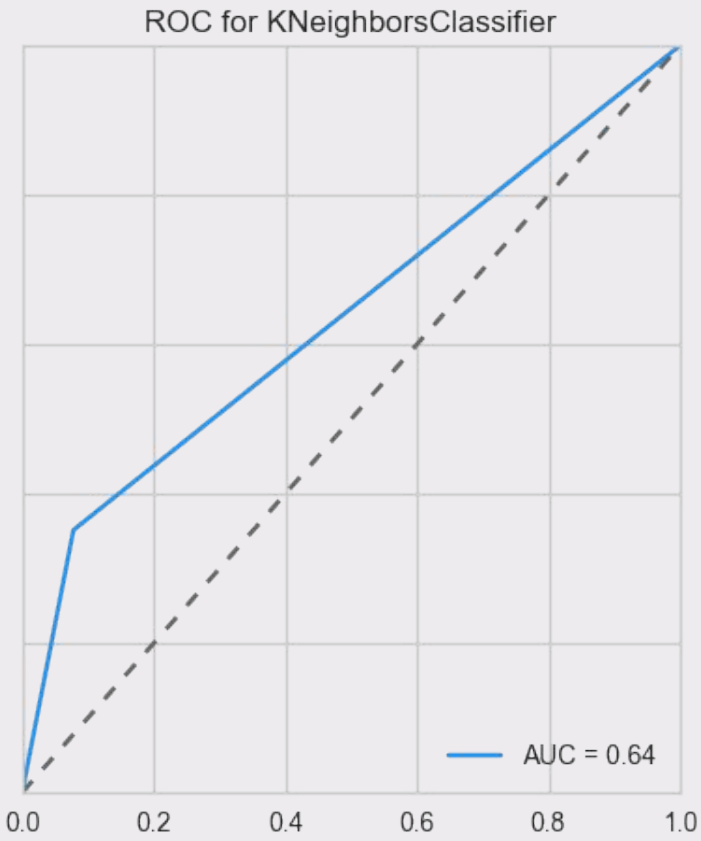
Visualizing Model Selection



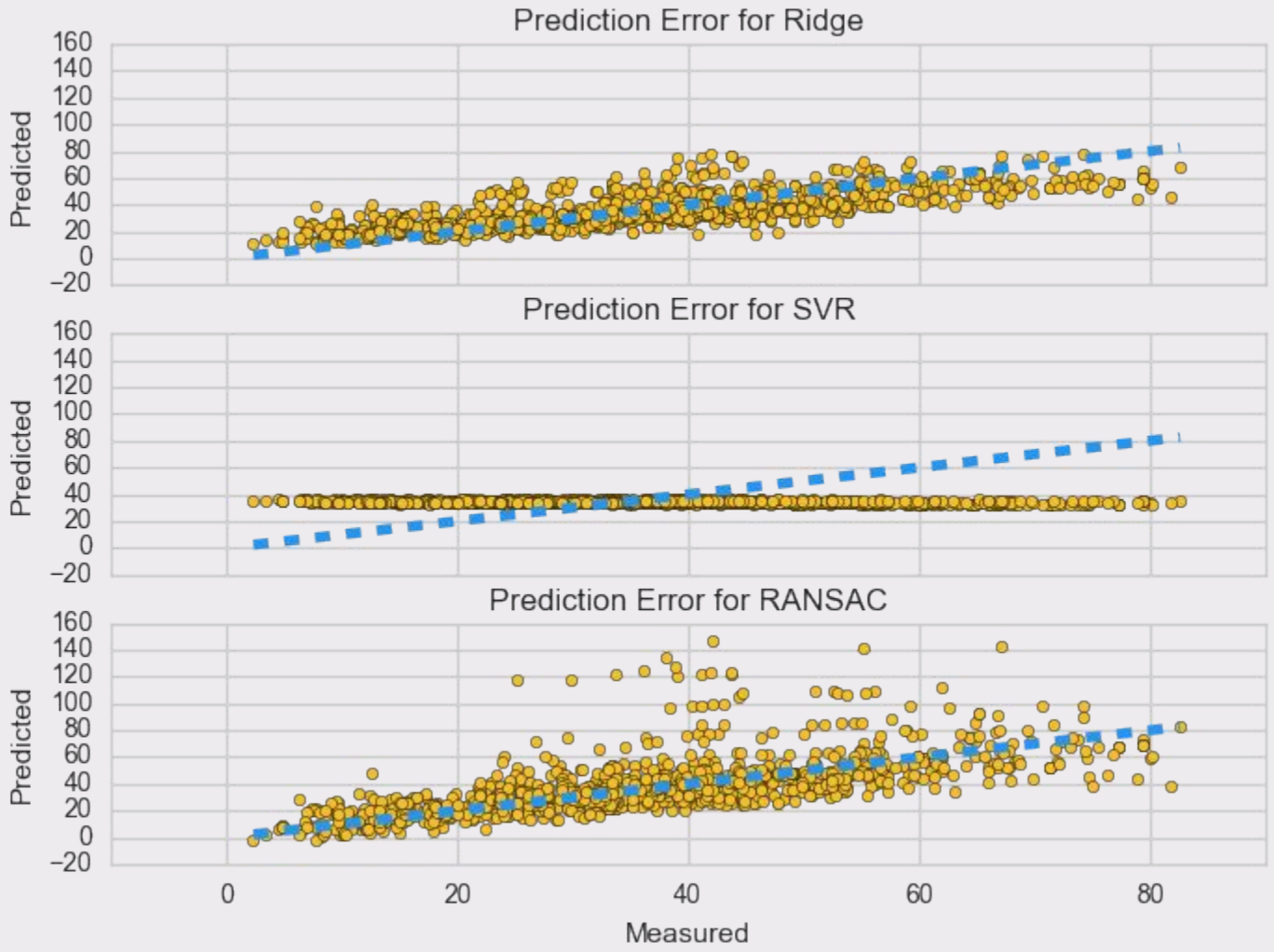


Confusion Matrices

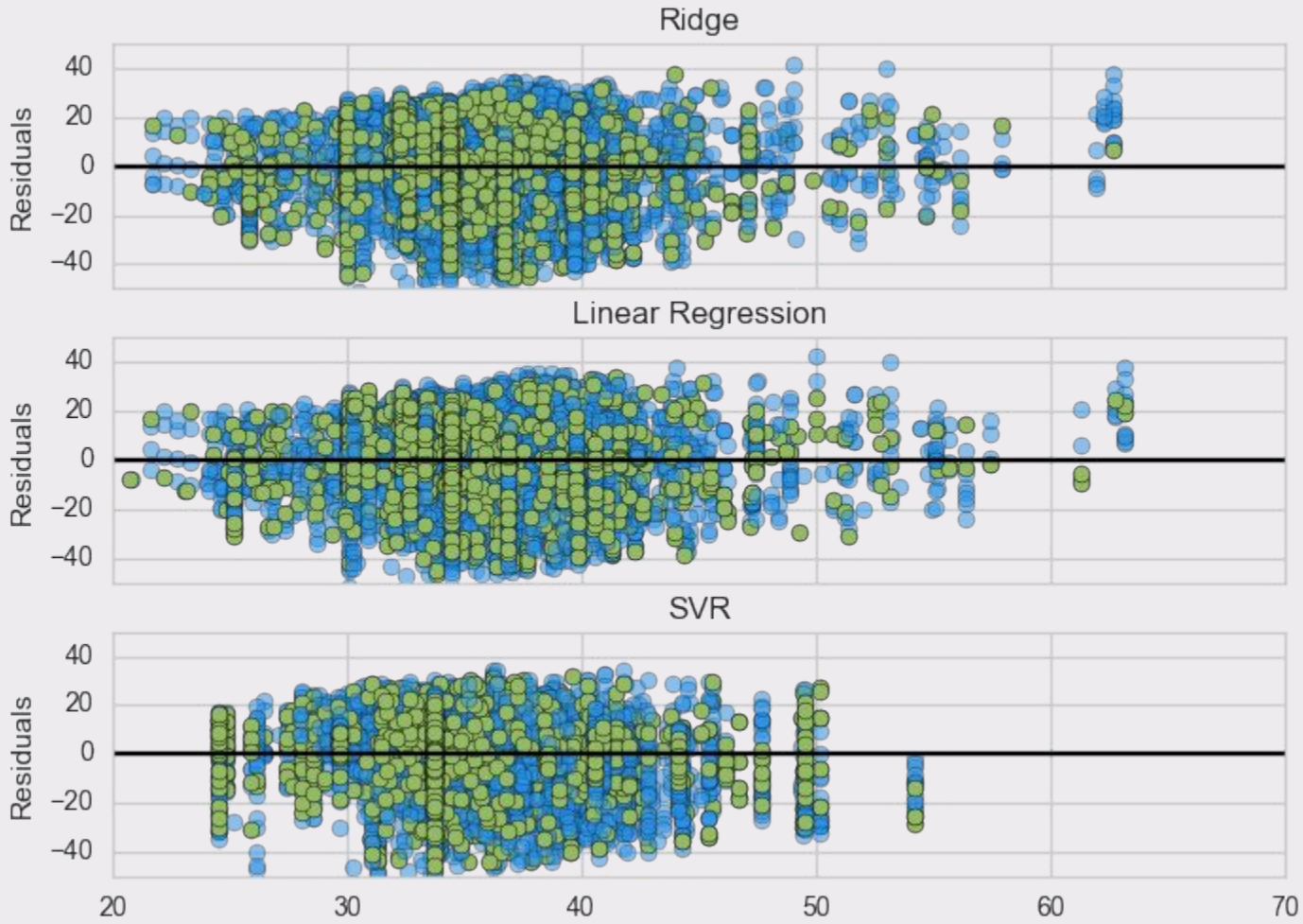




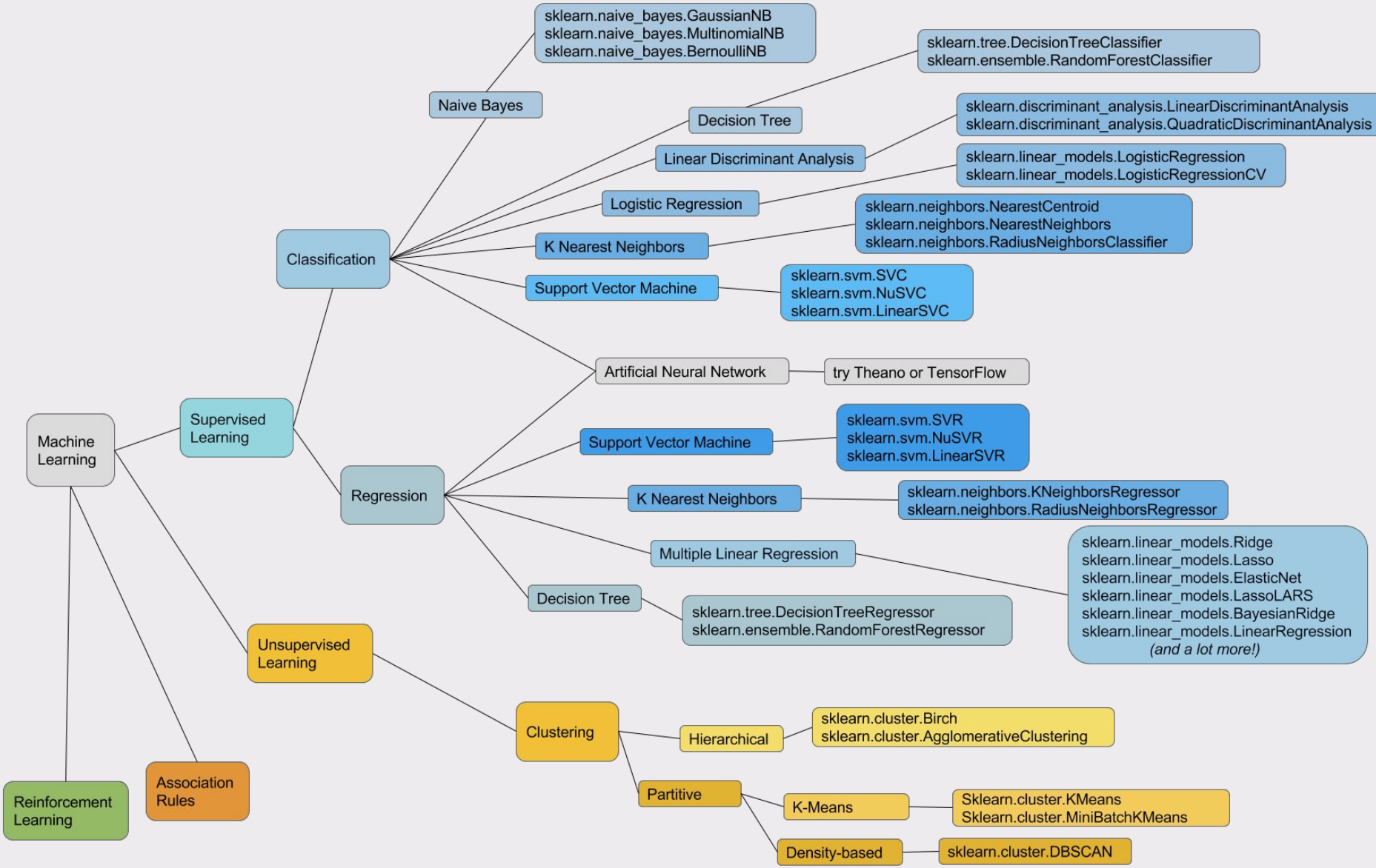
Receiver Operator Characteristic (ROC) and Area Under Curve (AUC)



Prediction Error Plots

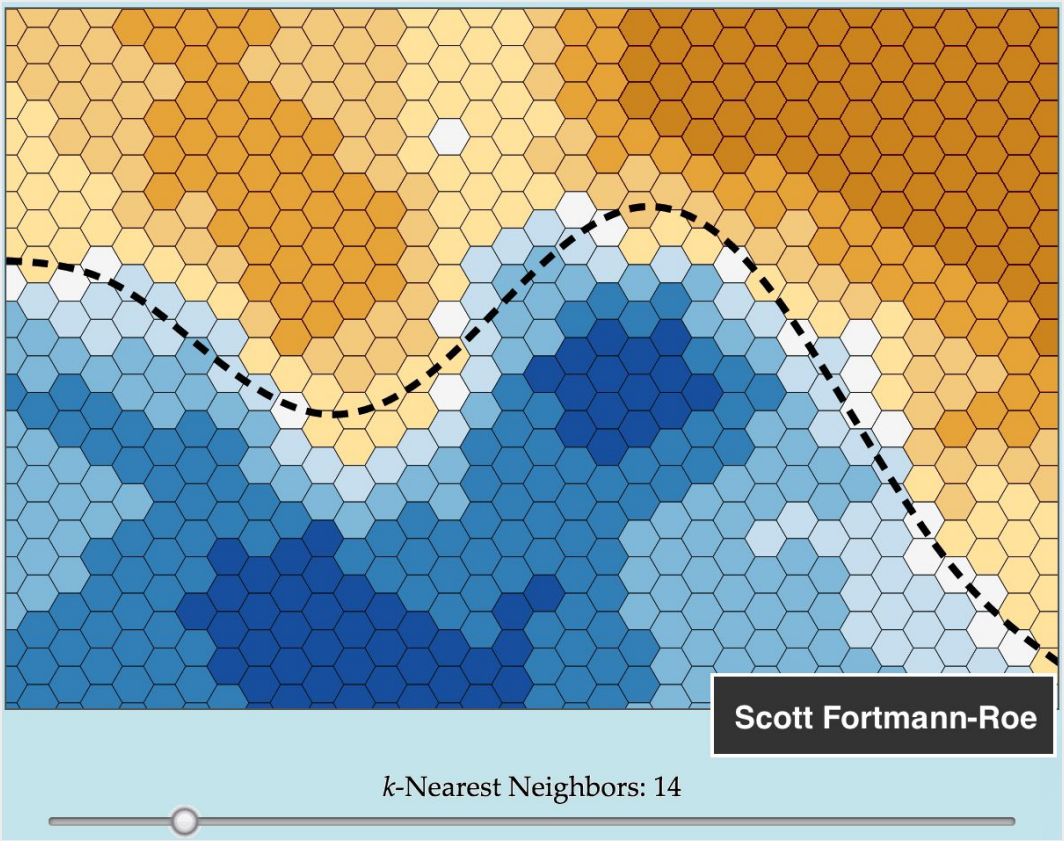


Visualizing Residuals

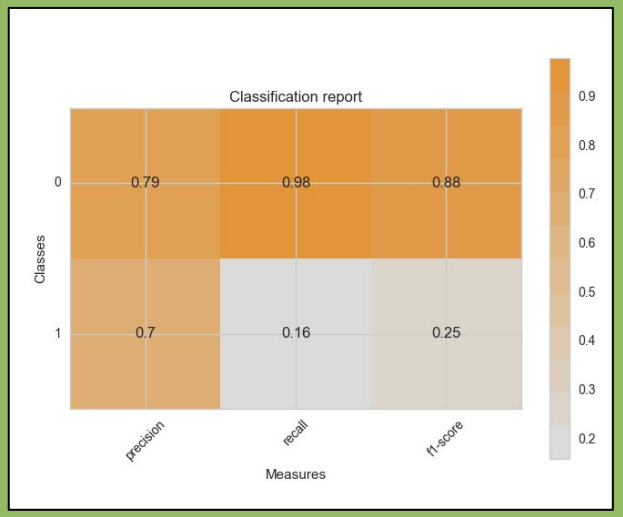


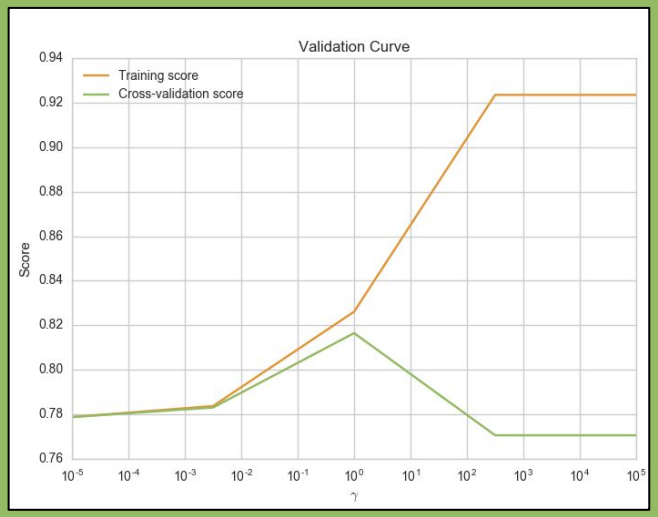
Model Families vs. Model Forms vs. Fitted Models
Rebecca Bilbro http://bit.ly/2a1YoTs

kNN Tuning Slider in 2 Dimensions

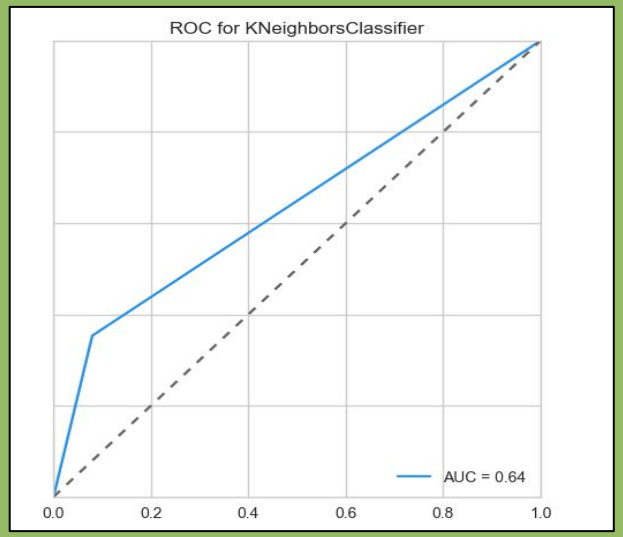


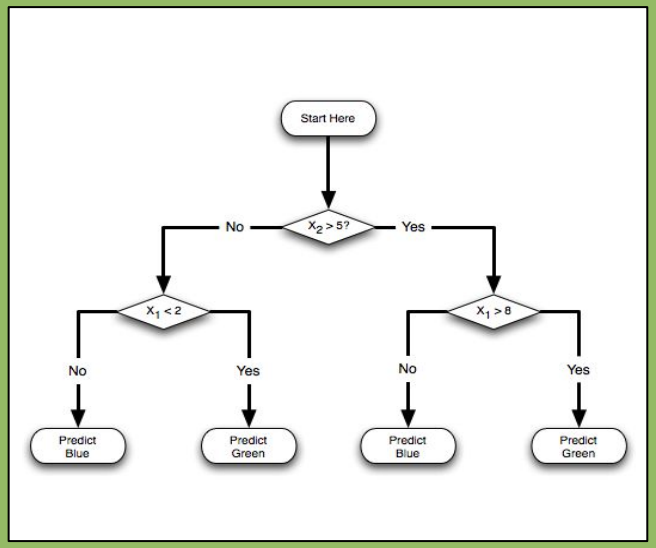
Scott Fortmann-Roe http://bit.ly/29P4SS1

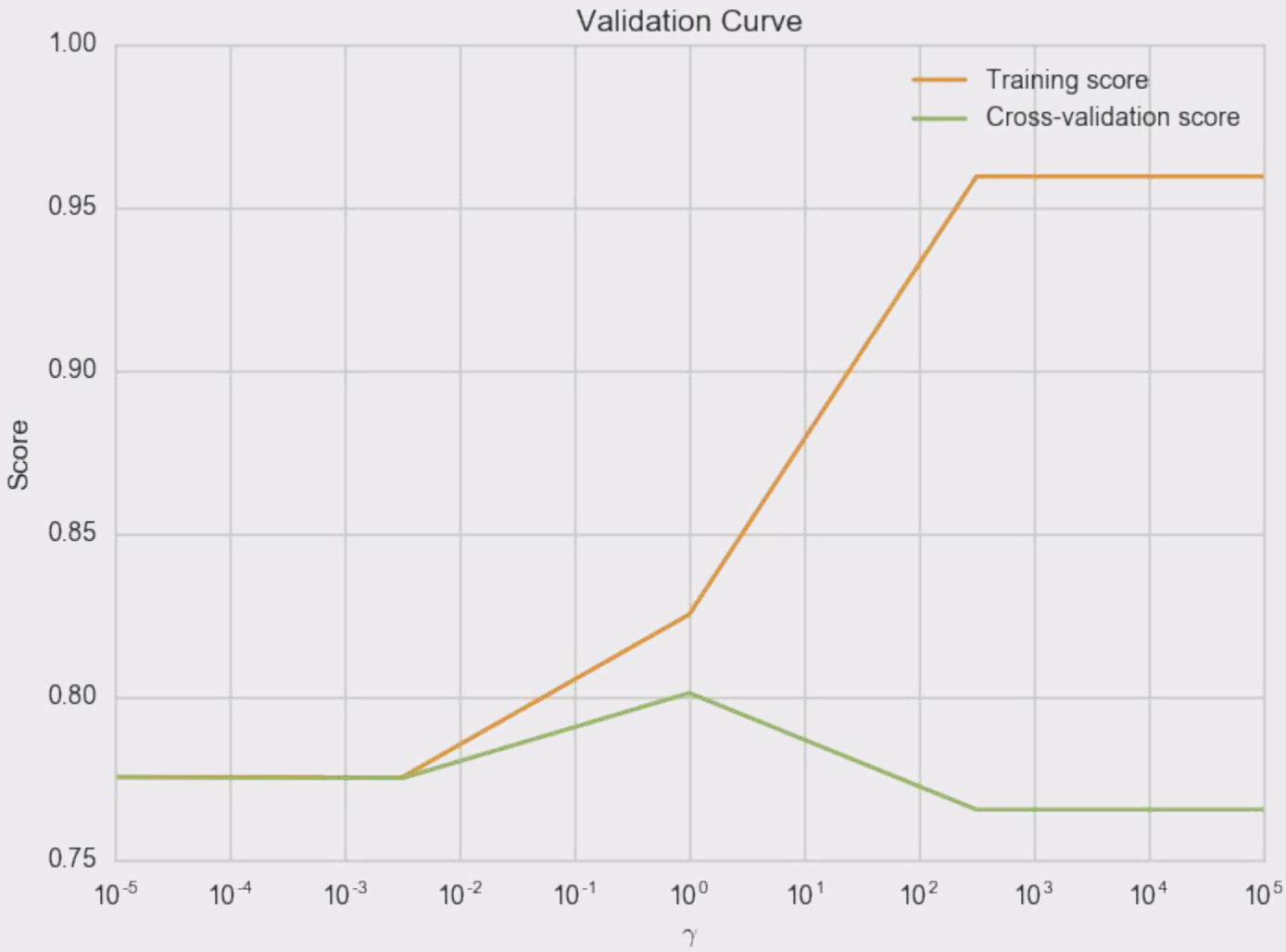




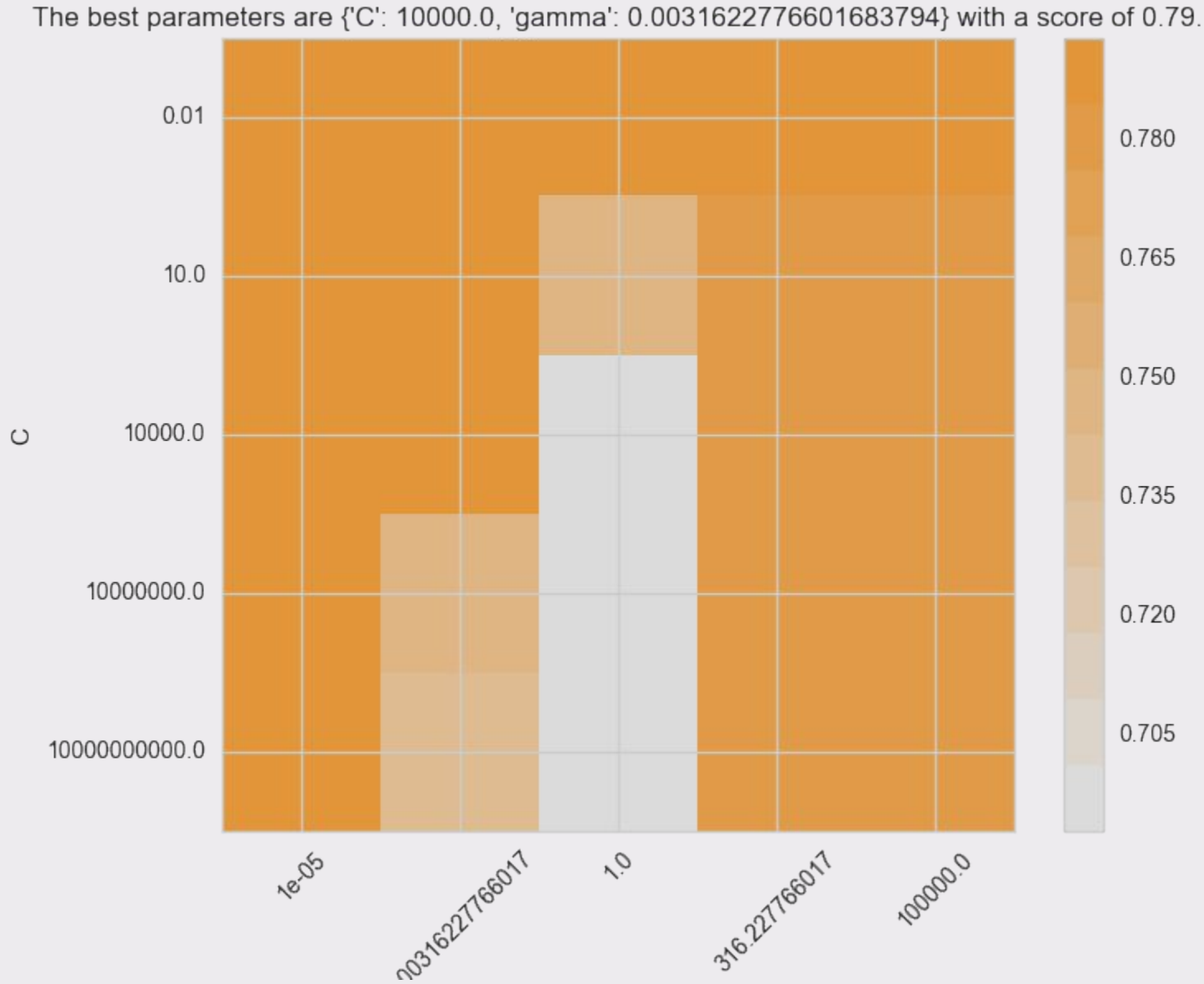
Visualizing Evaluation/Tuning



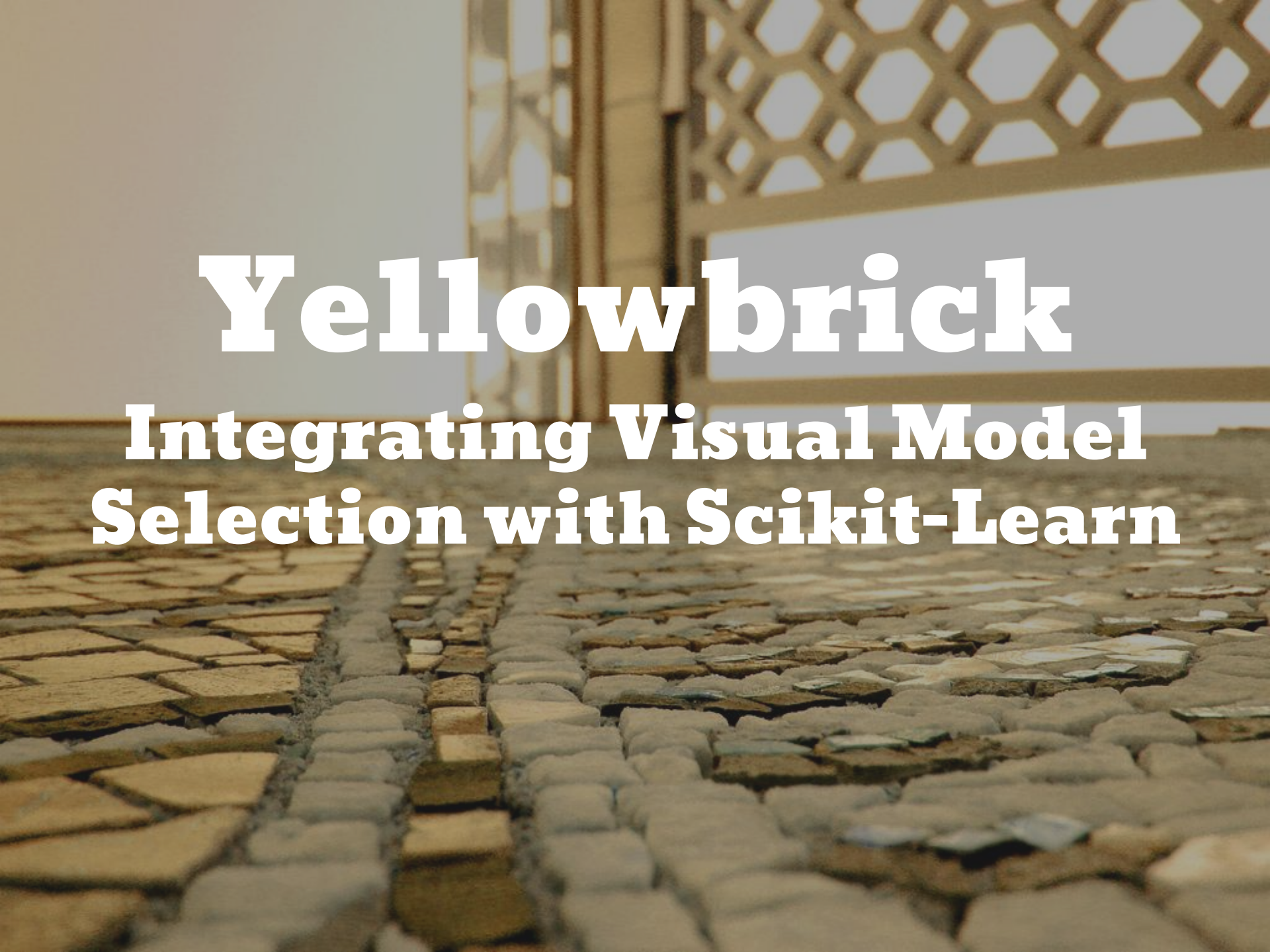


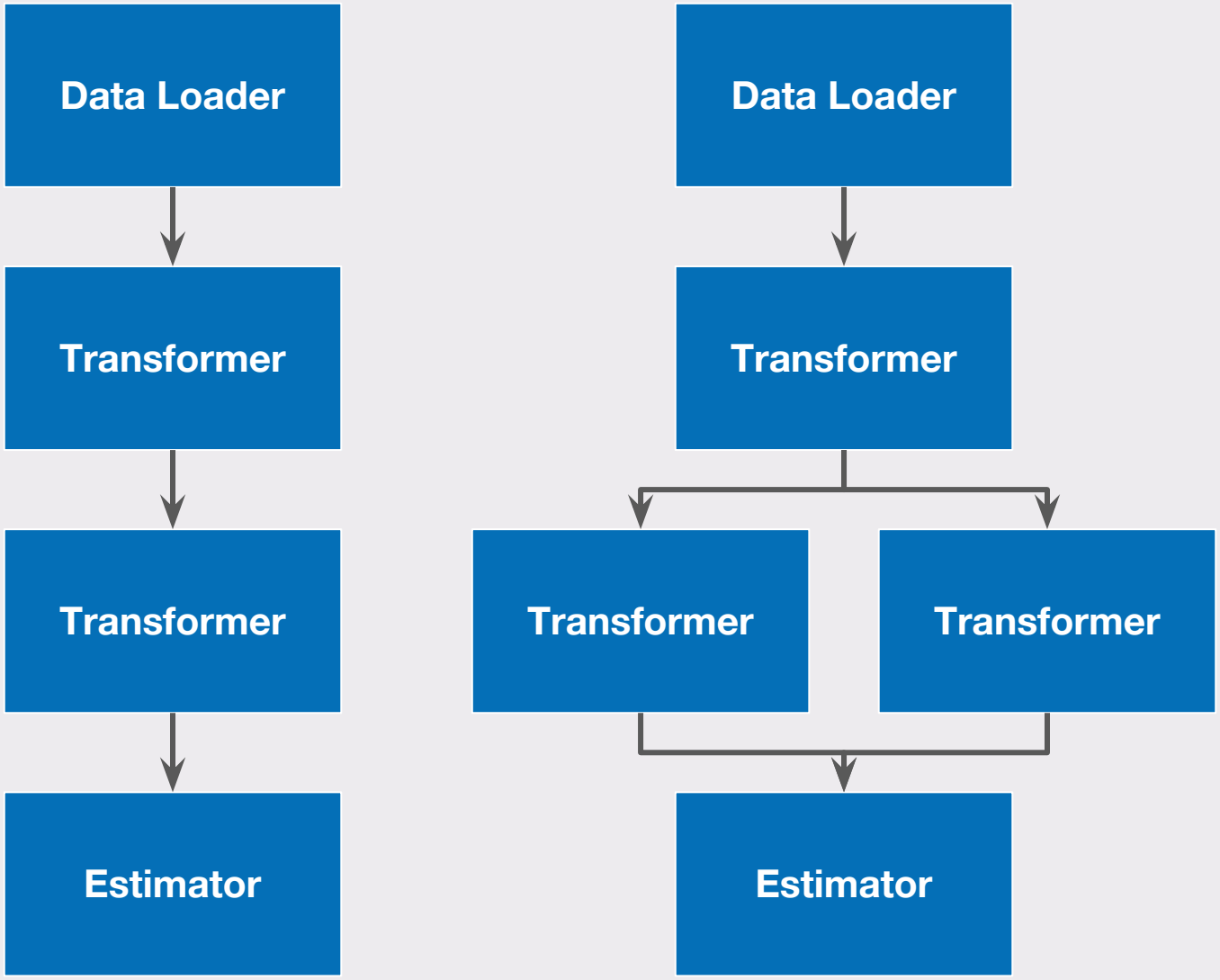


Cross Validation Curves

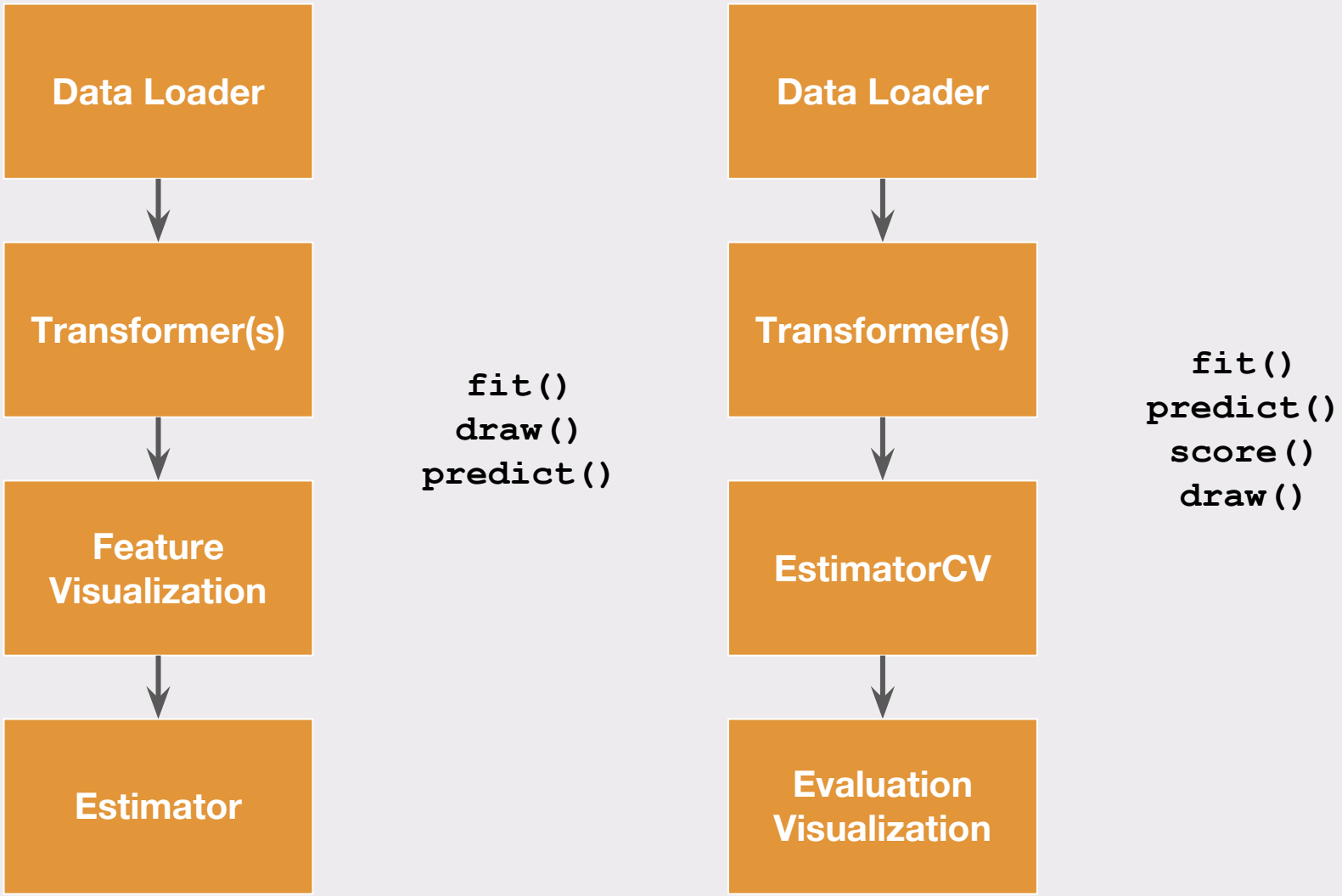


Visual Grid Search

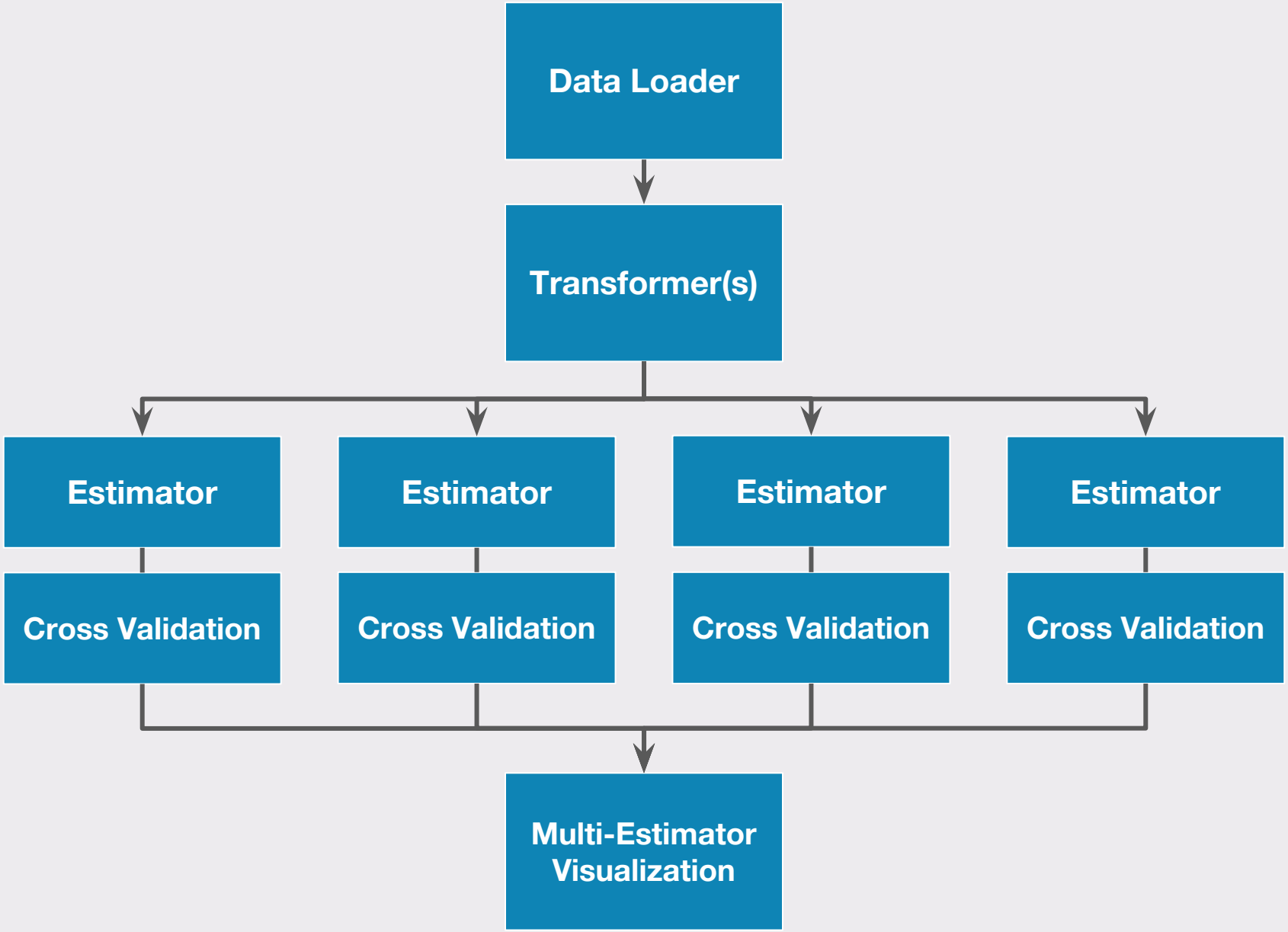

Yellowbrick
Integrating Visual Model Selection with Scikit-Learn



Scikit-Learn Pipelines: `fit()` and `predict()`

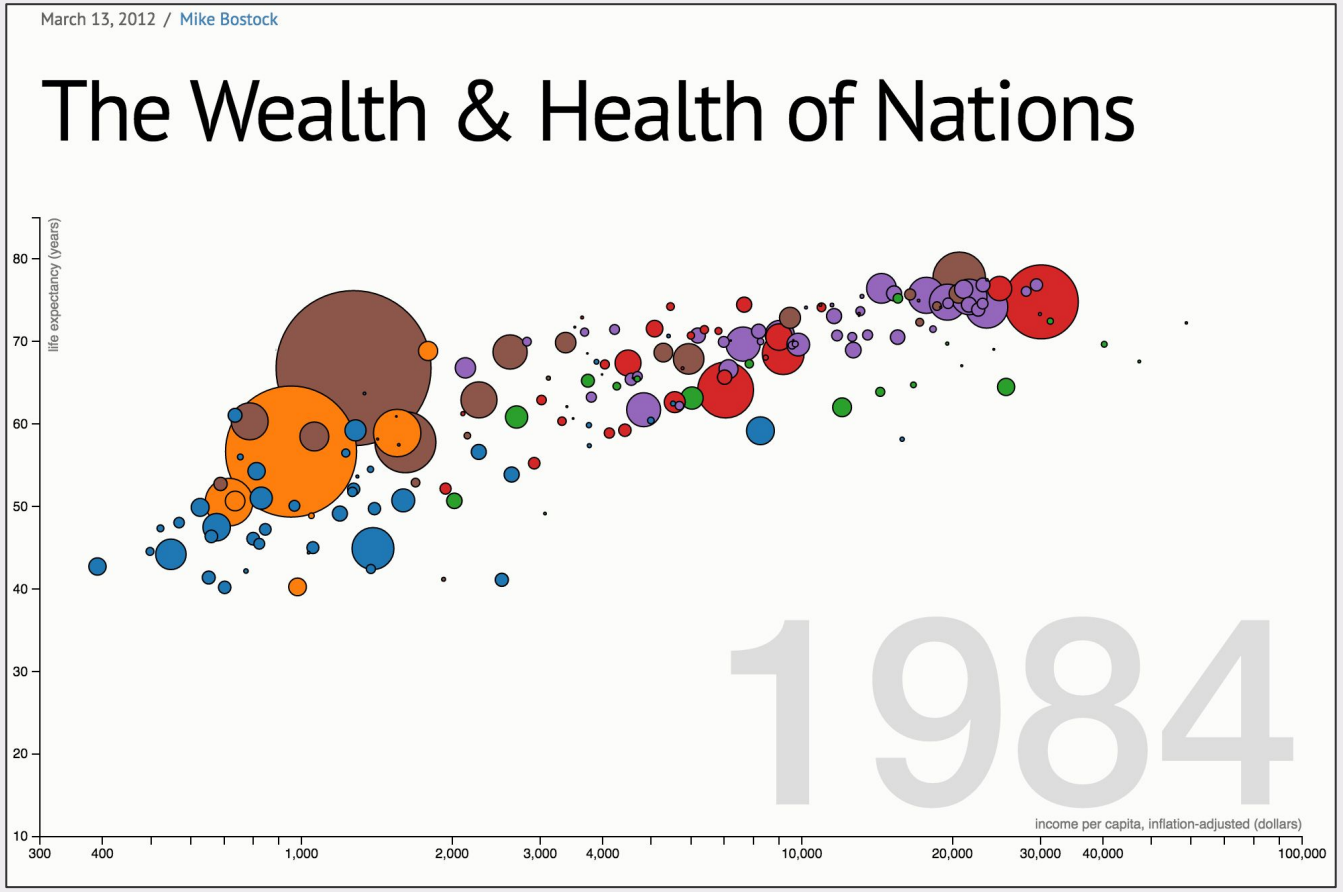


Yellowbrick Visual Transformers



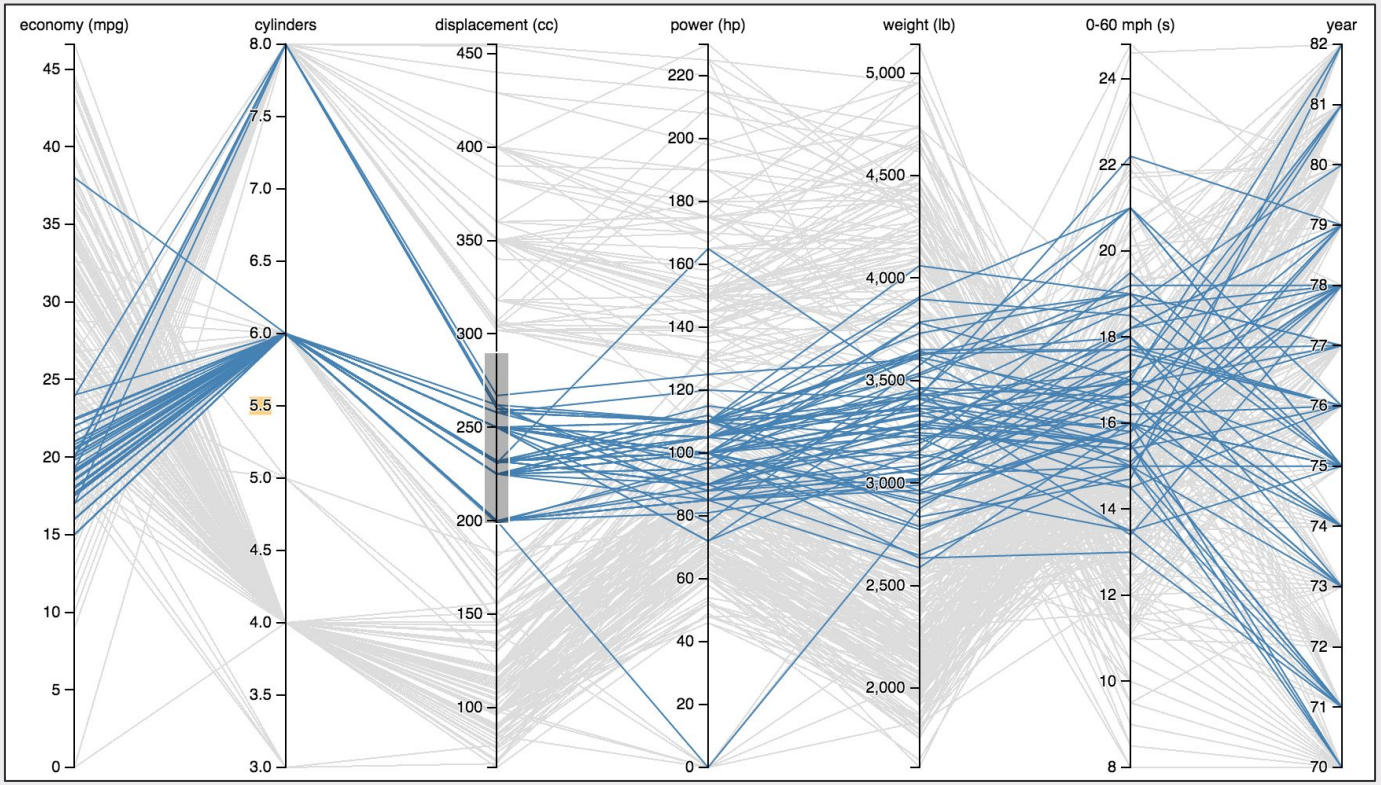
Model Selection Pipelines

Employ Interactivity to Visualize More



Health and Wealth of Nations Recreated by Mike Bostock
Originally by Hans Rosling http://bit.ly/29RYBJD

Visual Analytics Mantra:
Overview First; Zoom & Filter; Details on Demand



Heer, Jeffrey, and Ben Shneiderman. "Interactive dynamics for visual analysis." Queue 10.2 (2012): 30.

bbengfort / **occupancy-detection**

Star Edit Info

Files Schema Explore Settings

Experimental data used for binary classification (room occupancy) from Temperature, Humidity, Light and CO2.
http://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+

Edit README.md Upload Download

bbengfort datatraining.txt Version 1 on April 12, 2016

datatest.txt	CSV data				3 months, 1 week ago
datatest2.txt	CSV data				3 months, 1 week ago
datatraining.txt	CSV data	7 dimensions	8,143 rows	582.7 KB	3 months, 1 week ago

Cultivar
Visual Model Management System

README.md

Occupancy Detection

Experimental data used for binary classification (room occupancy) from Temperature, Humidity, Light and CO2. Ground-truth occupancy was obtained from time stamped pictures that were taken every minute.

Dataset

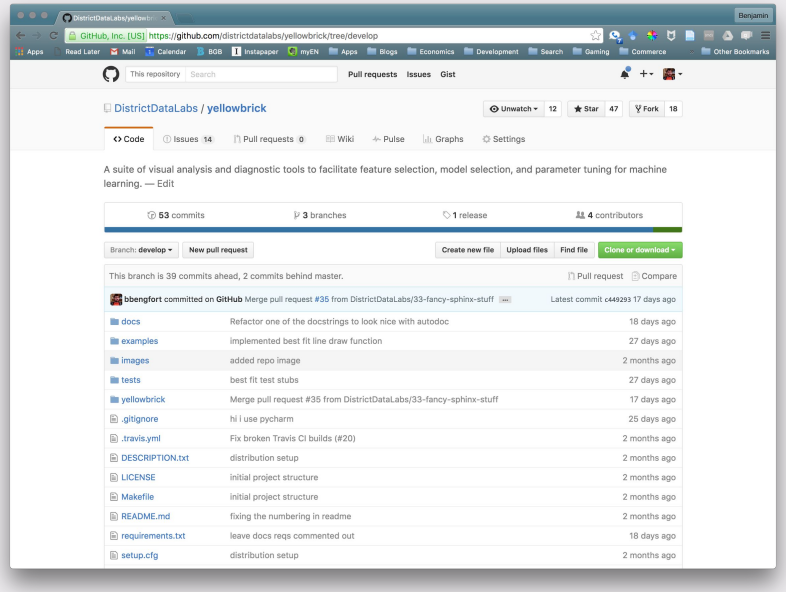
Three data sets are submitted, for training and testing. Ground-truth occupancy was obtained from time stamped pictures that were taken every minute.

For the journal publication, the processing R scripts can be found on github.com/LuisM78/occupancy-detection-data.

Attributes

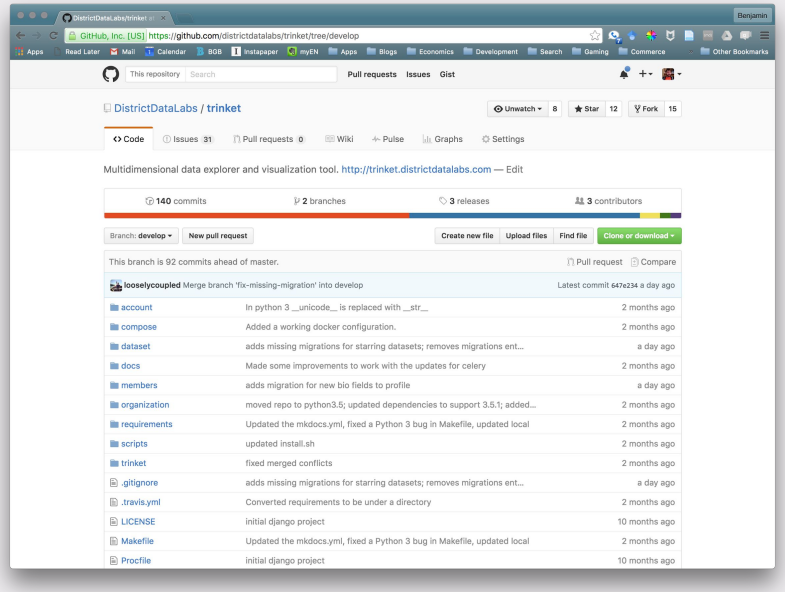
- date time year-month-day hour:minute:second

DDL Open Source Projects on GitHub



Yellowbrick

http://bit.ly/2a2Y0jy



Cultivar

http://bit.ly/2a5otxB