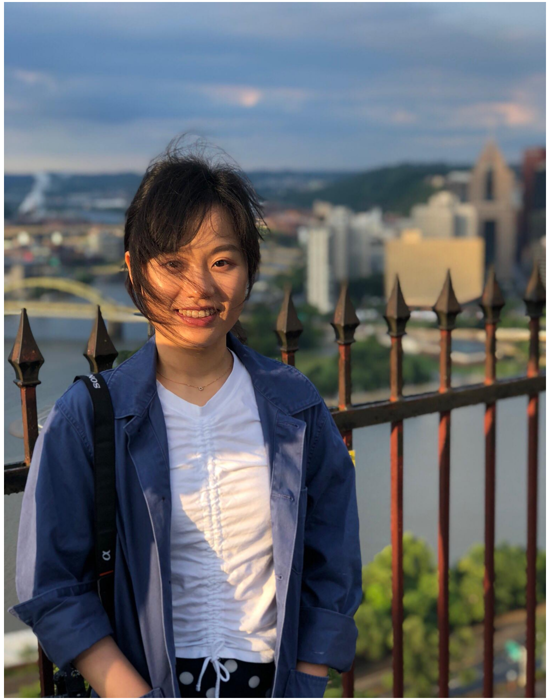
Visual Diagnostics for More Informed Machine Learning

Women in Data Science 2019

Dr. Rebecca Bilbro

Head of Data Science, ICX Media
Co-creator, Scikit-Yellowbrick
Author, Applied Text Analysis with Python

@rebeccabilbro



Tianshu Li

Ph.D. Student, University of Virginia
Crowd-sourced Urban Infrastructure Management
Text Analysis on Historical Inspection Reports

How to get the course materials:

```
git clone git@github.com:icxmedia/ml-teaching-materials.git
```

or, depending on the network:

```
git clone https://github.com/icxmedia/ml-teaching-materials.git
```

then:

```
cd ml-teaching-materials
pip install -r requirements.txt
```

How to get the data:

```
git clone git@github.com:DistrictDataLabs/yellowbrick.git
```

or, depending on the network:

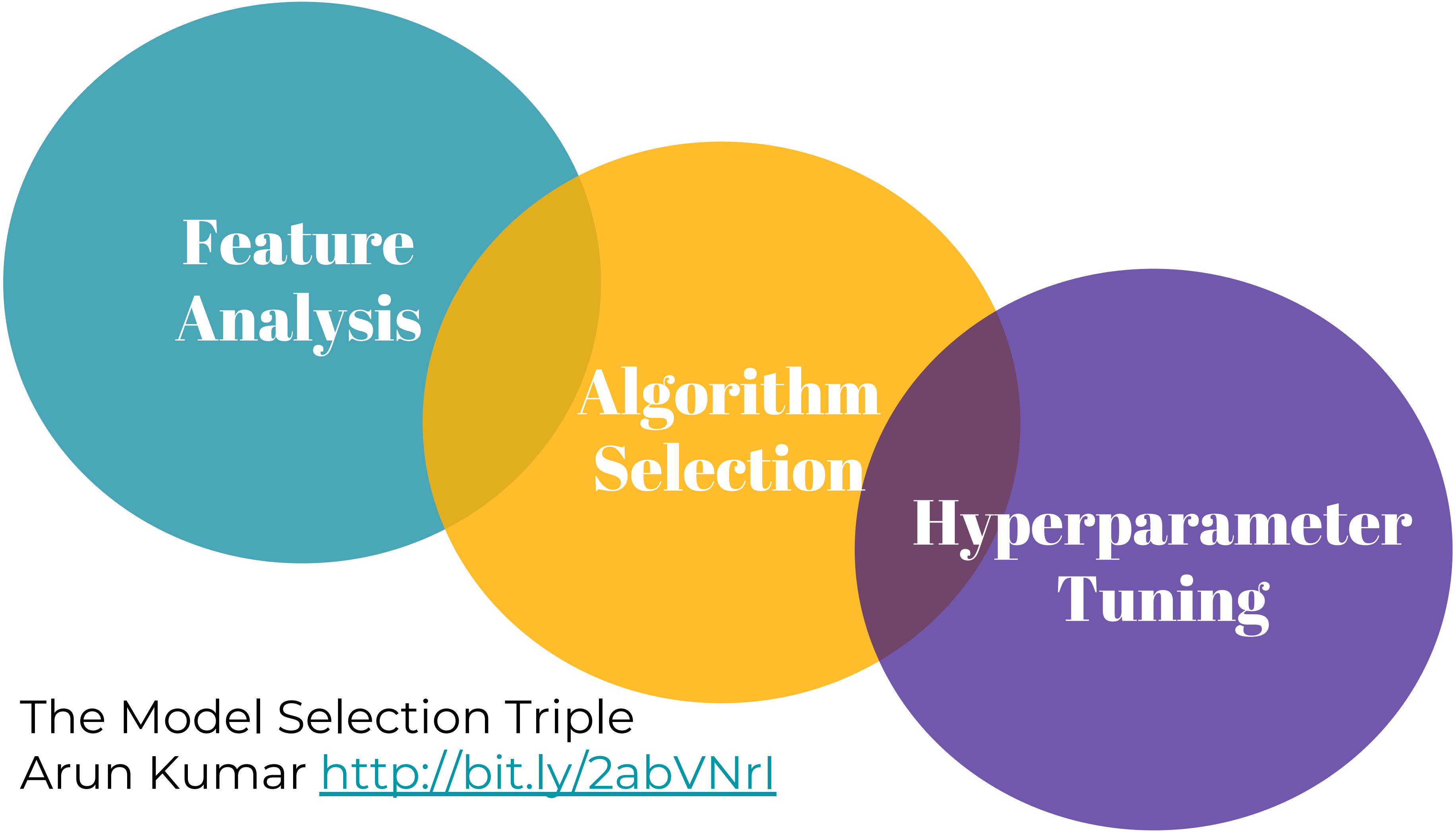
```
git clone https://github.com/DistrictDataLabs/yellowbrick.git
```

then:

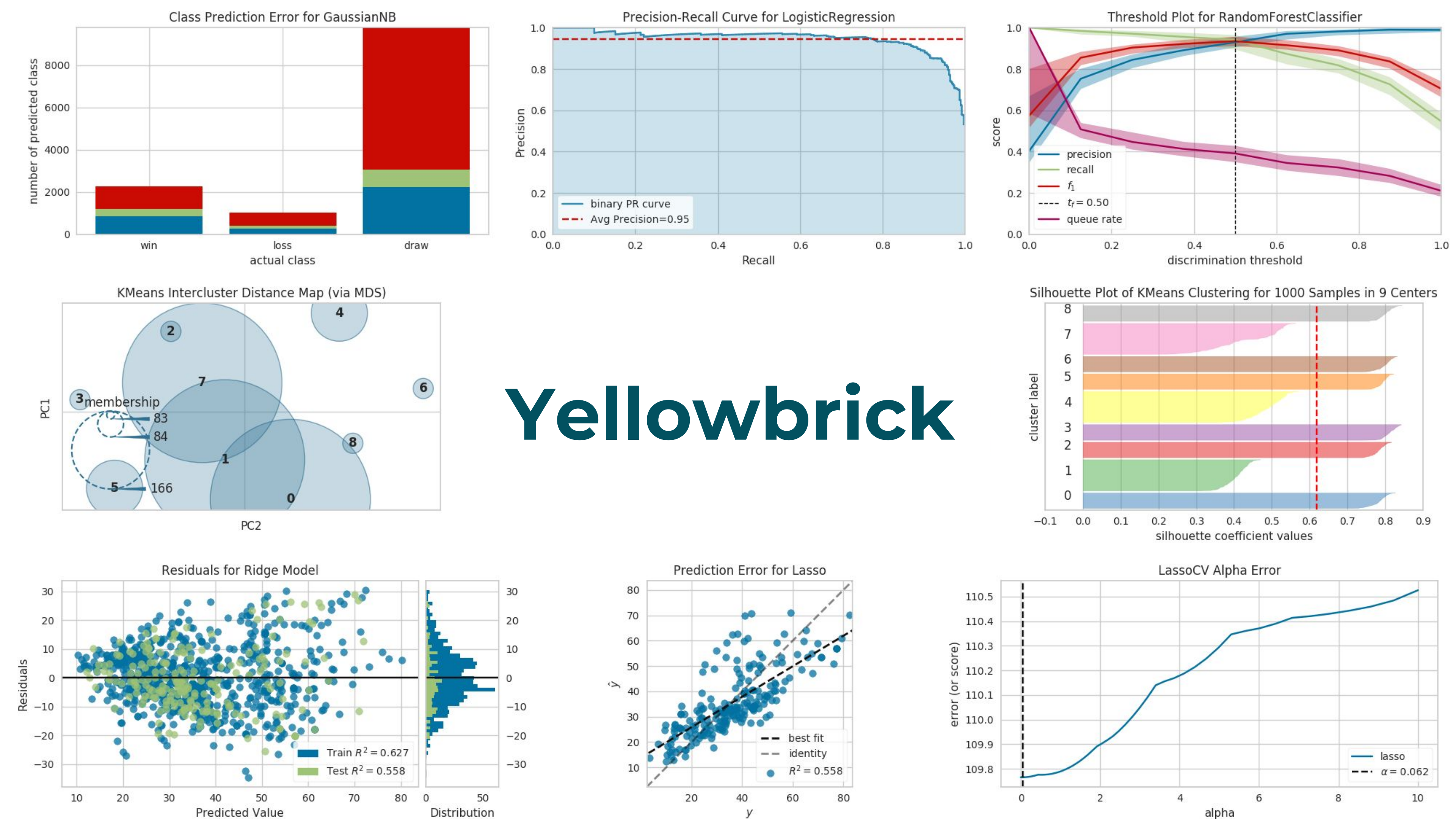
```
cd yellowbrick
pip install -r requirements.txt
pip install -e .
python -m yellowbrick.download
```

The machine learning problem:
Given a set of *n* data samples,
each represented by >1 number,
create a model that is able to predict
properties of as-yet unseen samples.

Ask me for my strong opinions about Random Forests.



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


Class Prediction Error for GaussianNB
number of predicted class
actual class
win
loss
draw
Precision-Recall Curve for LogisticRegression
Precision
Recall
binary PR curve
Avg Precision=0.95
Threshold Plot for RandomForestClassifier
score
discrimination threshold
precision
recall
f_1
$t_f = 0.50$
queue rate
KMeans Intercluster Distance Map (via MDS)
PC1
PC2
membership
83
84
166
Yellowbrick
Silhouette Plot of KMeans Clustering for 1000 Samples in 9 Centers
cluster label
silhouette coefficient values
Residuals for Ridge Model
Residuals
Predicted Value
Distribution
Train $R^2 = 0.627$
Test $R^2 = 0.558$
Prediction Error for Lasso
$\hat{y}$
y
best fit
identity
$R^2 = 0.558$
LassoCV Alpha Error
error (or score)
alpha
lasso
$\alpha = 0.062$

Scikit-Learn & Yellowbrick

Once upon a time...

And then it got...

Try them all!


```
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import model_selection as ms

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

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

scikit-learn Estimators

The main API implemented by scikit-learn is that of the estimator. An estimator is any object that learns from data;

it may be a classification, regression or clustering algorithm, or a transformer that extracts/filters useful features from raw data.

```
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
```

scikit-learn Transformers

Transformers are special cases of Estimators -- instead of making predictions, they transform the input dataset X to a new dataset X′.

```
class Transformer(Estimator):

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

scikit-learn interface

```python
# Import the estimator
from sklearn.linear_model import Lasso

# Instantiate the estimator
model = Lasso()

# Fit the data to the estimator
model.fit(X_train, y_train)

# Generate a prediction
model.predict(X_test)
```

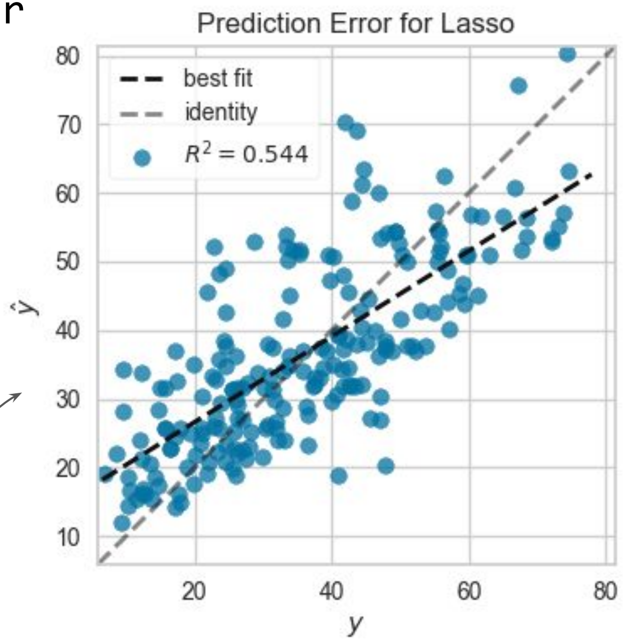
Yellowbrick interface

```python
# Import the model and visualizer
from sklearn.linear_model import Lasso
from yellowbrick.regressor import PredictionError

# Instantiate the visualizer
visualizer = PredictionError(Lasso())

# Fit
visualizer.fit(X_train, y_train)

# Score and visualize
visualizer.score(X_test, y_test)
visualizer.poof()
```



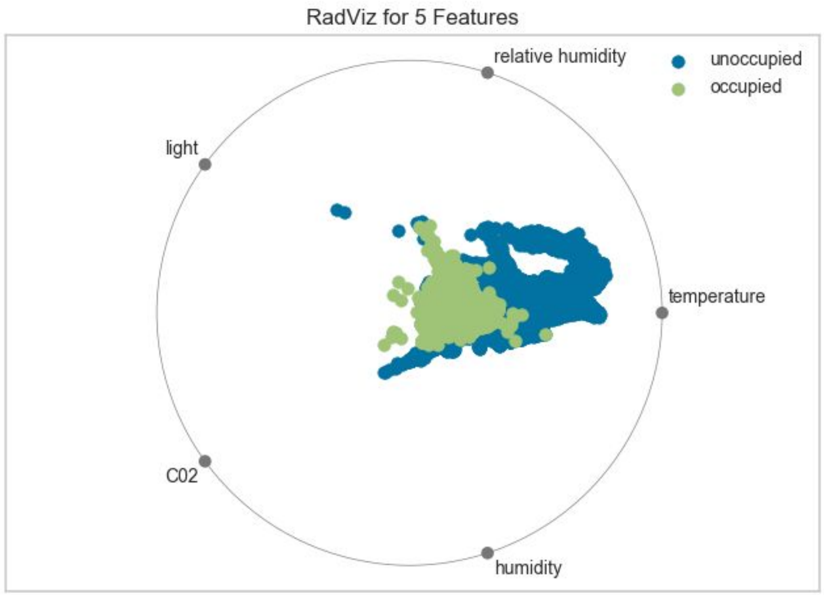
Steering the model selection triple

Feature Analysis

a.k.a. finding the ***smallest possible*** set of features that result in the ***most predictive*** model.

1. Look for separability

Radial Visualization



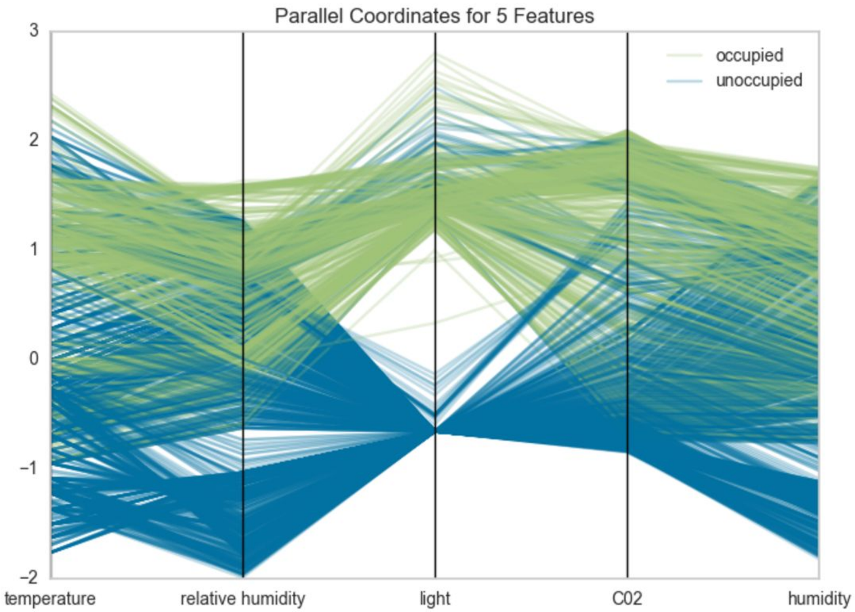
Features pull instances towards their position on the circle in proportion to their normalized numerical value for that instance.

Parallel Coordinates

Features represented as vertical lines.

Points represented as connected line segments.

Look for single-colored chords.

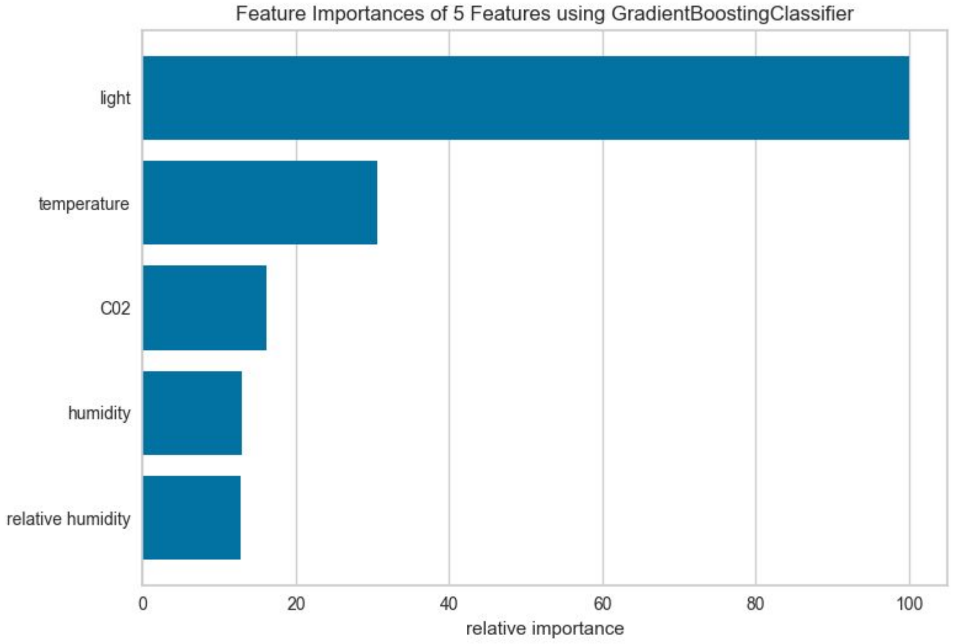


2. Look for correlation

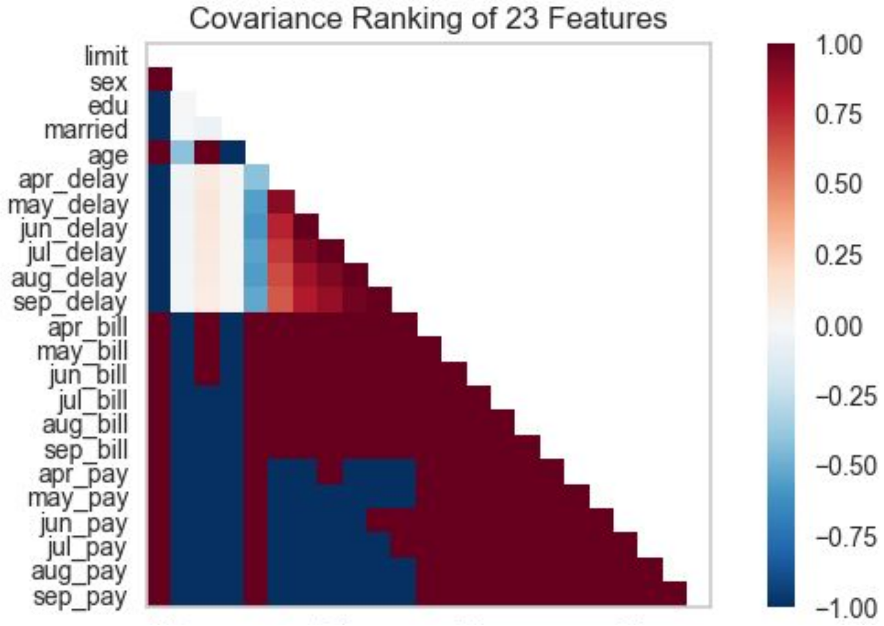
Feature Importance Plot

Visualize the relative importance of each feature to the model.

Identify weak features or combinations of features that are candidates for removal.



Rank2D



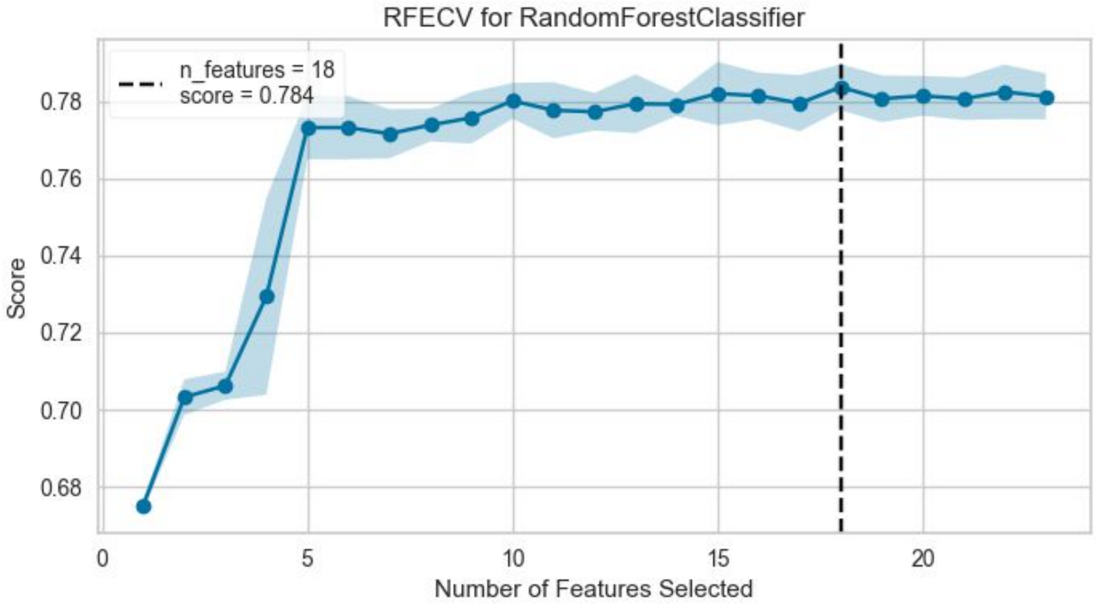
Visualize pairwise relationships as a heatmap.

Pearson shows us strong correlations, potential collinearity. Covariance helps us understand the sequence of relationships.

Recursive Feature Elimination

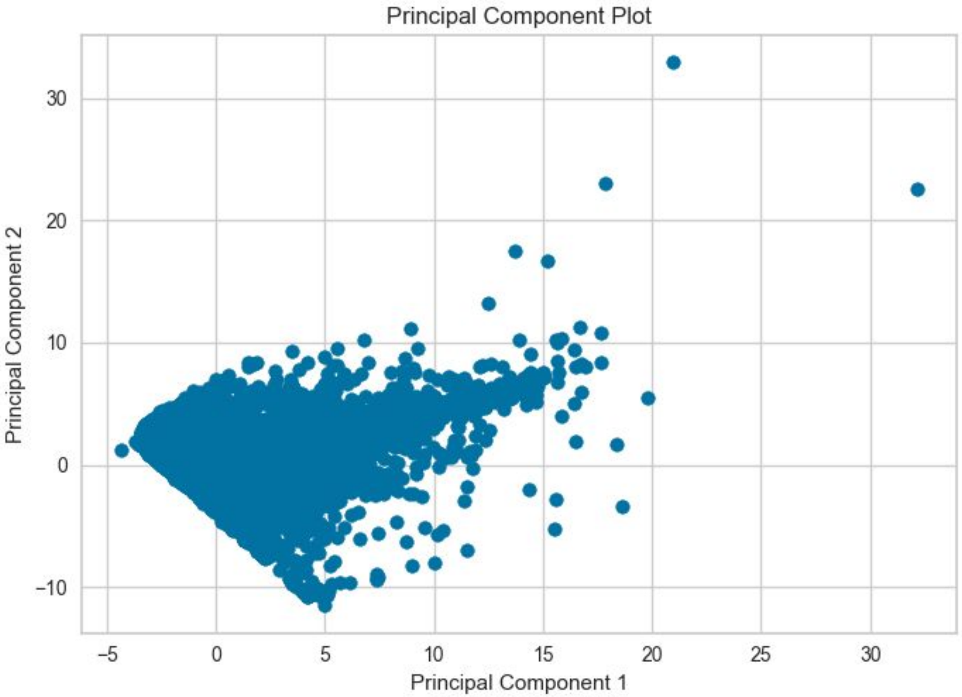
Iteratively drop the weakest feature(s) until desired number is reached.

Attempts to eliminate dependencies and collinearity.



3. Look at the distribution

PCA Projection Plots



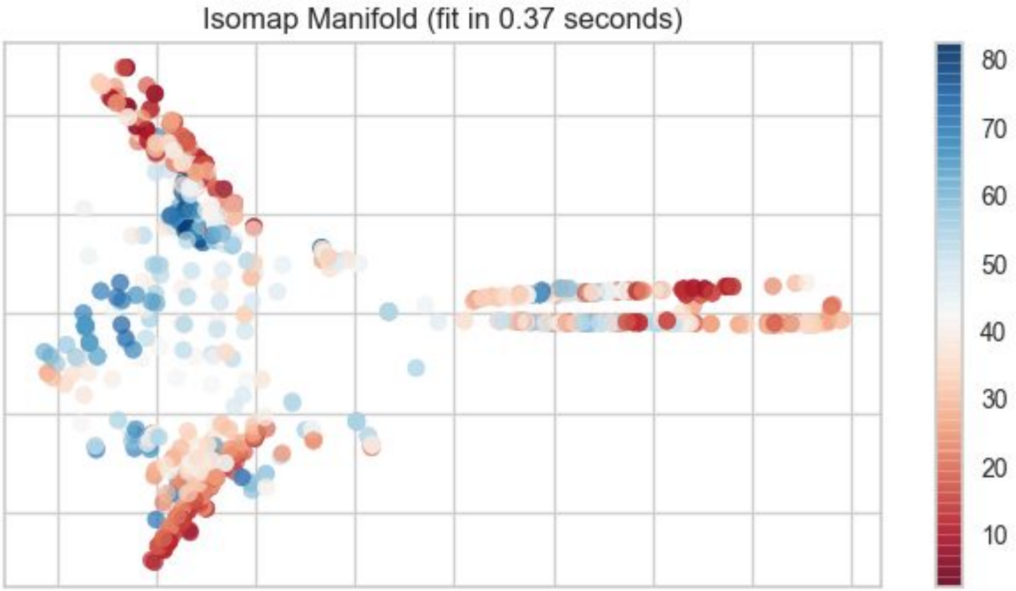
Decompose high dimensional data into two or three dimensions.

Visualize projected data along axes of principle variation.

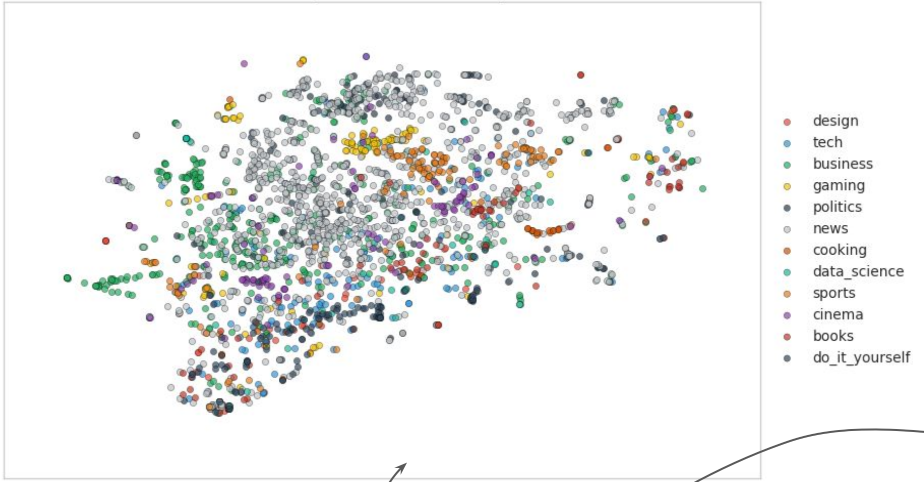
Manifolds

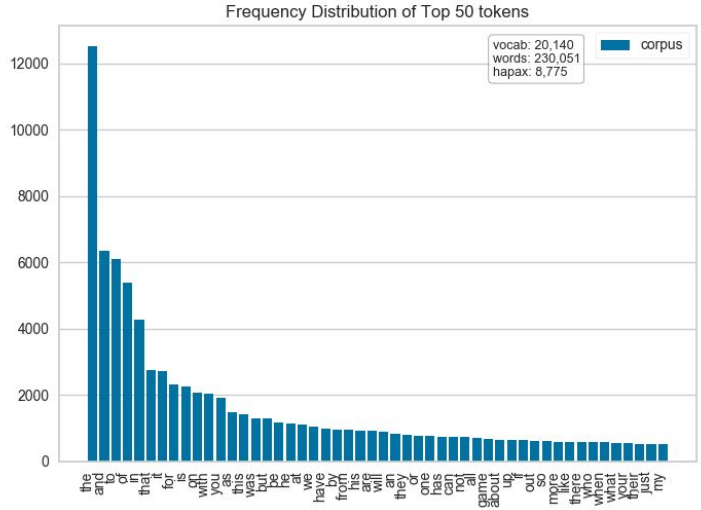
Embed instances into two dimensions

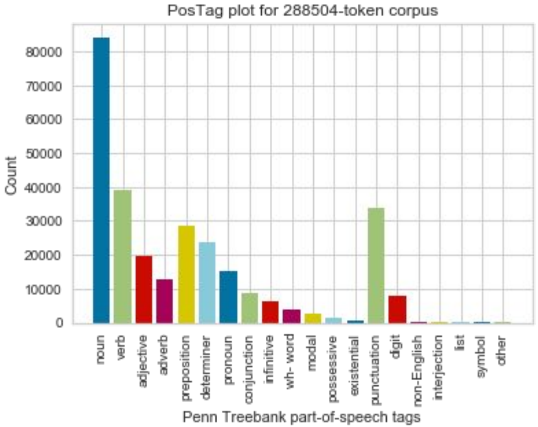
Look for latent (esp. non-linear) structures in the data, noise, separability.



Also for text!







visualize document distribution, top tokens & part-of-speech tags

How to launch the first lab:

```
cd notebooks
jupyter notebook feature-analysis.ipynb
```

Evaluation

Model ~~Selection~~

1. Decide what matters

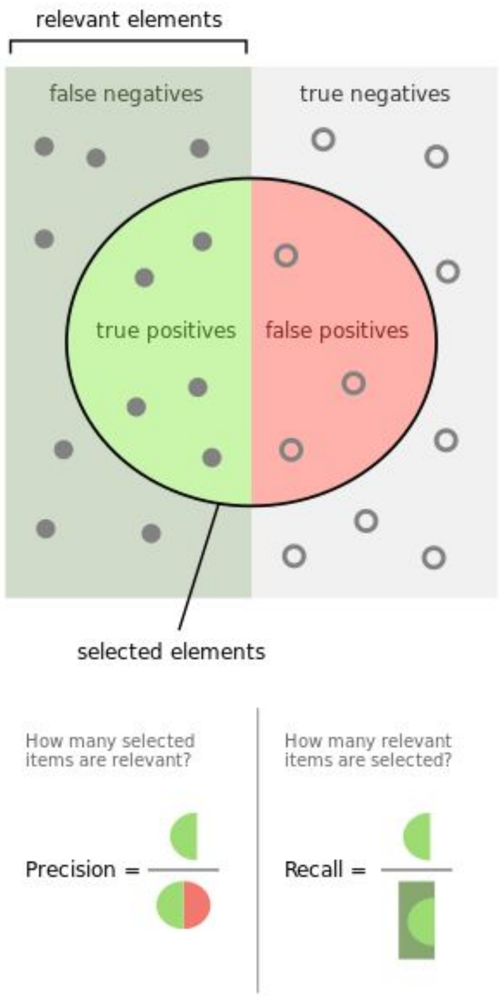
Evaluating Classifiers

Do we want to minimize false positives?
precision = true positives / (true positives + false positives)

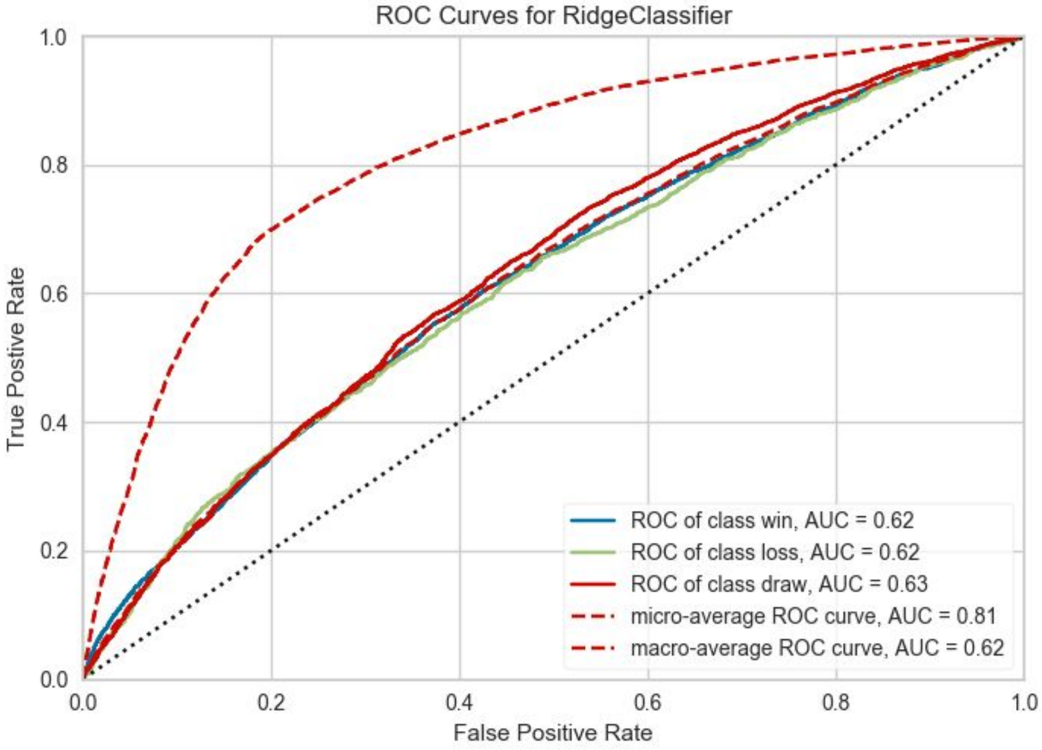
Do we want to minimize false negatives?
recall = true positives / (false negatives + true positives)

Will we need to compare many models?
F1 score = 2 * ((precision * recall) / (precision + recall))

Are the classes imbalanced?
support = number of training samples per class



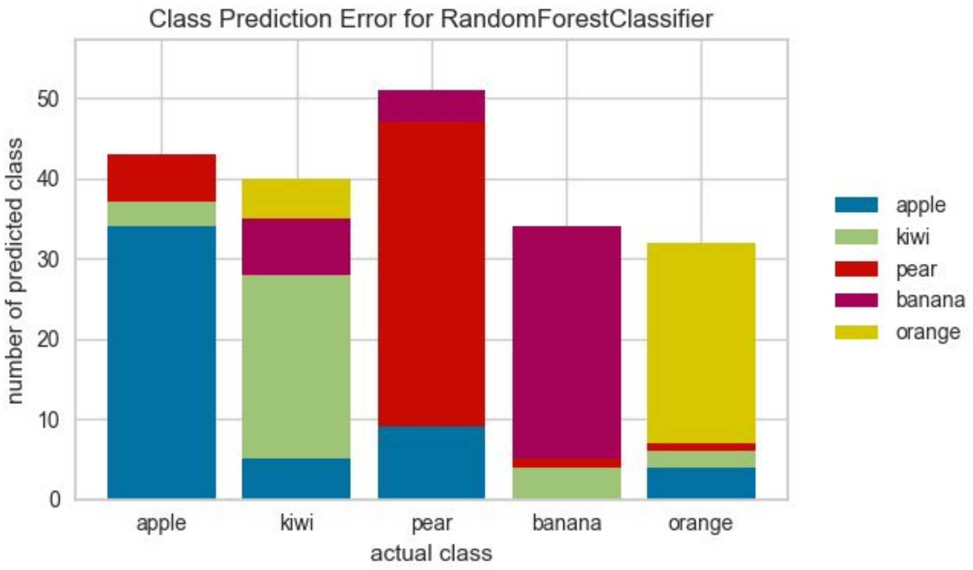
ROC-AUC



Getting more right **comes at the expense** of getting more wrong

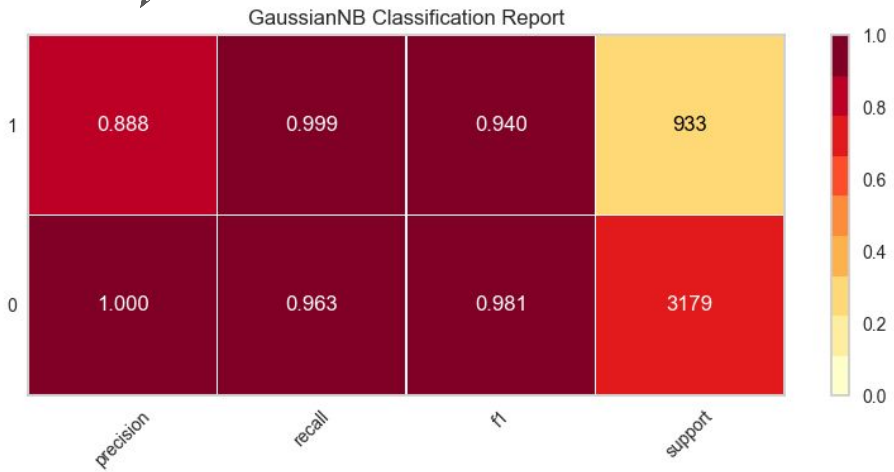
Class Prediction Error

Do I care about being right (or about **not being wrong**) for some categories more than for others?



Classification Heatmaps

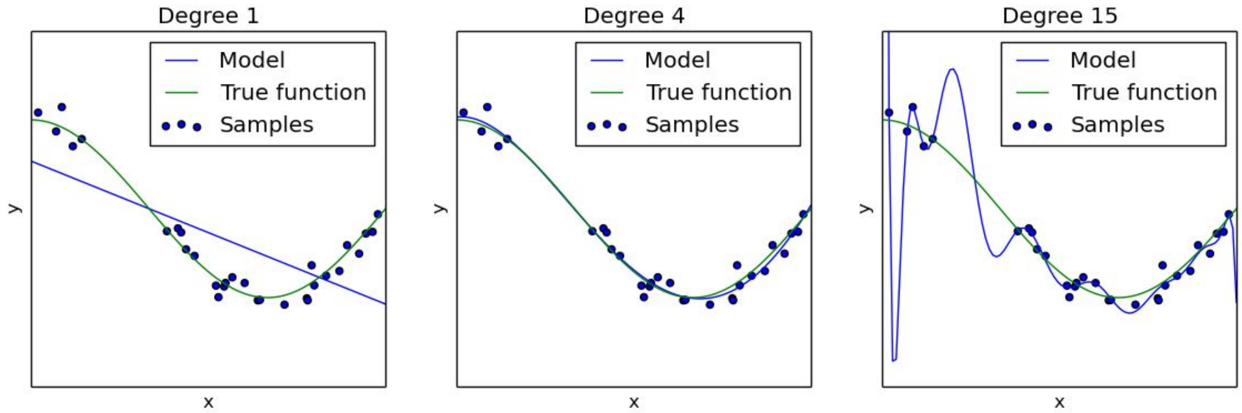
Is it better to have false positives here or here?



Precision: of those labelled edible, how many actually were?

Recall: how many of the poisonous ones did our model find?

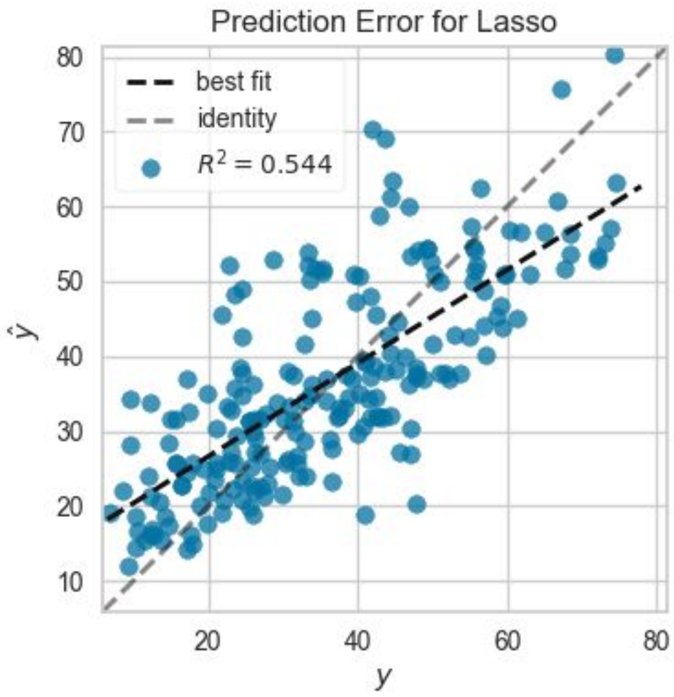
Evaluating Regressors



R^2 : How well does the model describe the training data? How well does the model predict out-of-sample data?

MSE/ASE: How sensitive is the model to outliers?

Prediction Error Plots



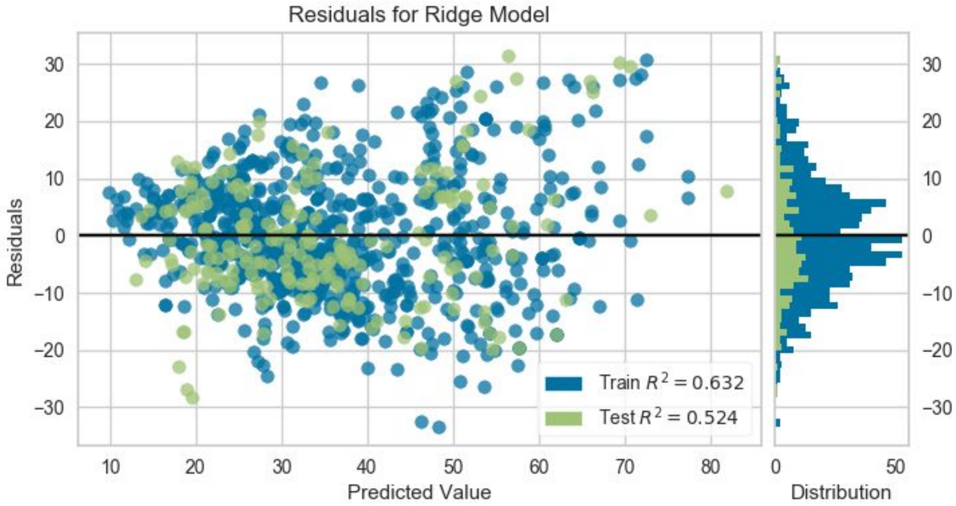
Visualize prediction errors as a scatterplot of the predicted & actual values.

Visualize the line of best fit & compare to the 45° line.

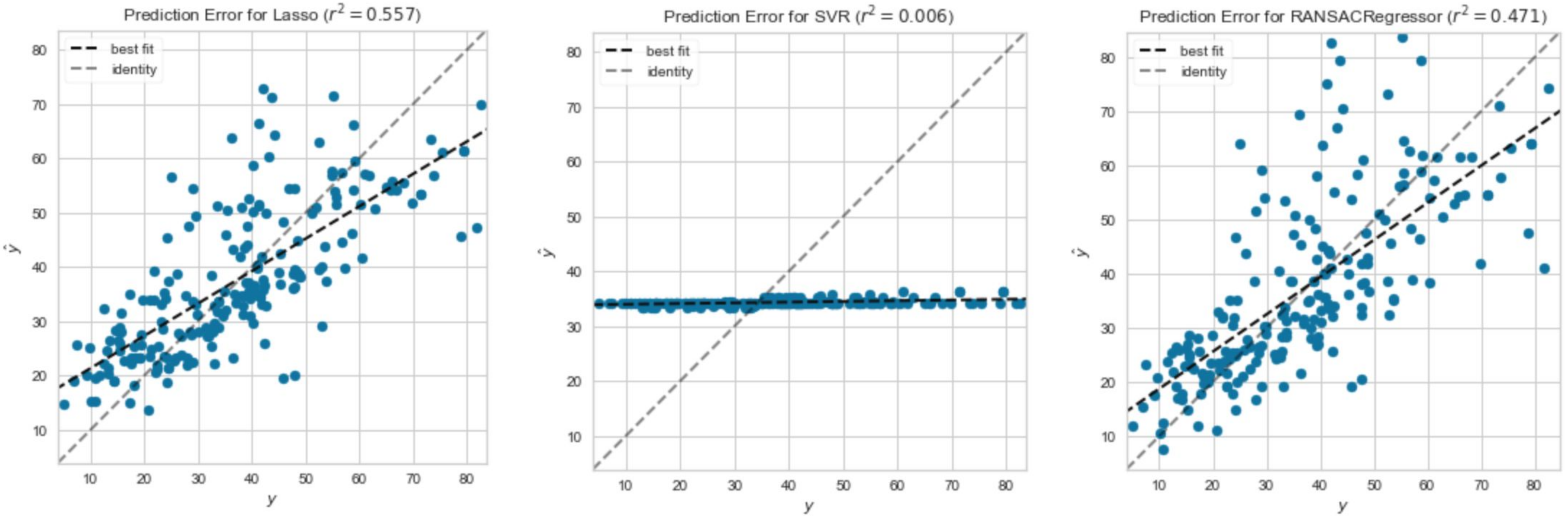
Plotting Residuals

Are residuals random?
we should not be able to predict error!

Visualize train and test data with different colors.



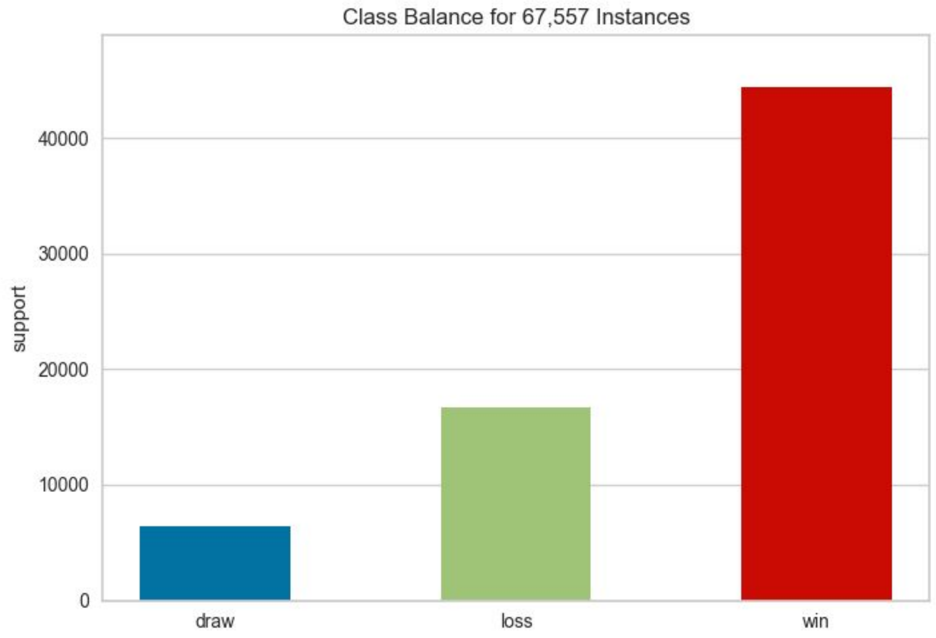
2. Try them all



Prediction Error for Lasso ($r^2 = 0.557$)
best fit
identity
$\hat{y}$
y
Prediction Error for SVR ($r^2 = 0.006$)
best fit
identity
$\hat{y}$
y
Prediction Error for RANSACRegressor ($r^2 = 0.471$)
best fit
identity
$\hat{y}$
y

3. Think horses

Class Balance



What to do with a low-accuracy classifier?

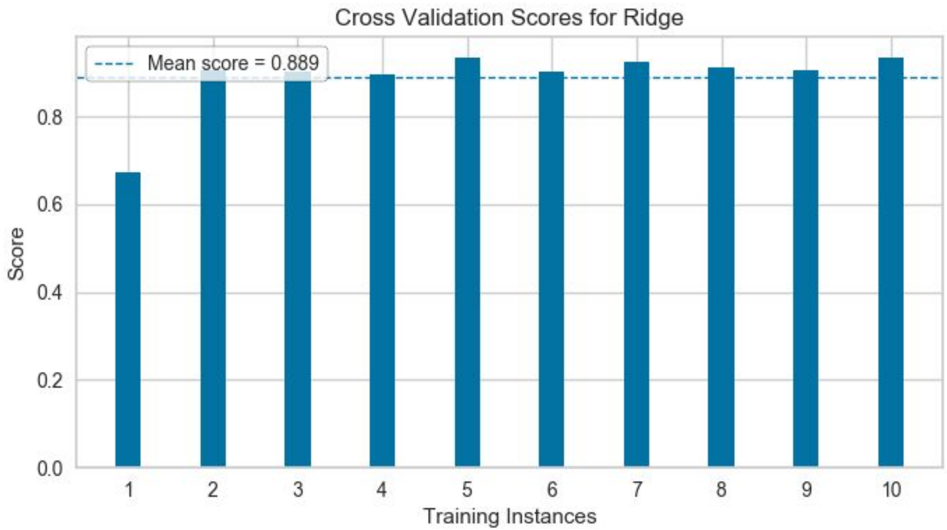
Check for class imbalance.

Try stratified sampling, oversampling, or need more data?

Cross Validation Scores

Real world data are distributed unevenly

Models are likely to perform better on some sections of data than others.



How to launch the second lab:

```
jupyter notebook model-evaluation.ipynb
```

Hyperparameter Tuning

1. Use the defaults

Hyperparameters

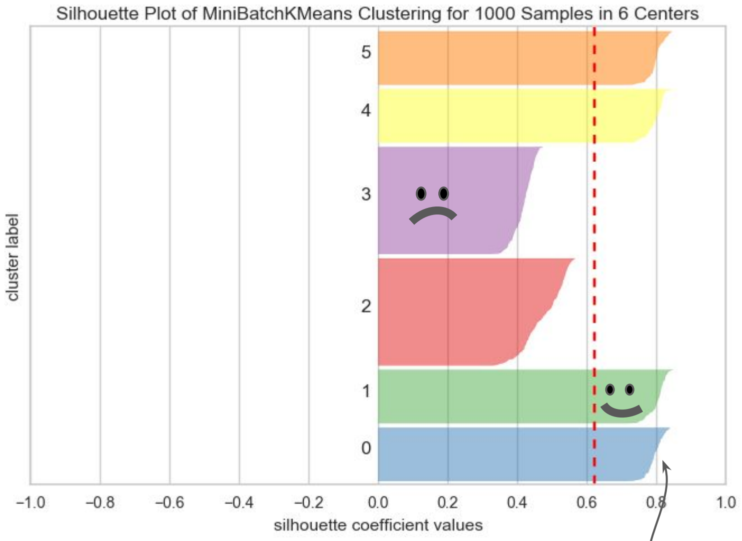
- When we call fit() on an estimator, it learns the parameters of the algorithm that make it fit the data best.
- However, some parameters are not directly learned within an estimator. E.g.
  - depth of a decision tree
  - alpha for regularization
  - kernel for support vector machines
  - number of clusters for centroidal clustering
- These parameters are often referred to as **hyperparameters**.

2. Gridsearch

Hyperparameter space is large and gridsearch is slow if you don't already know what you're looking for.

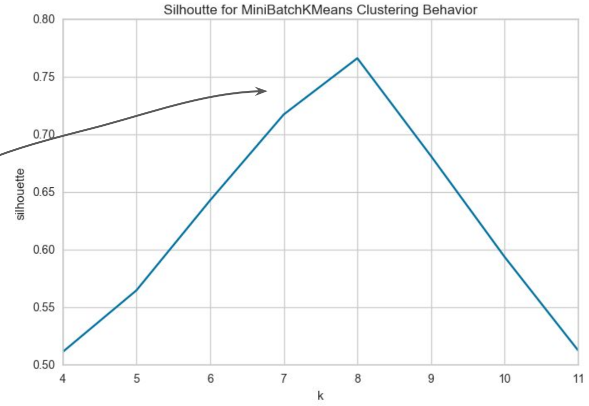
3. Visual gridsearch

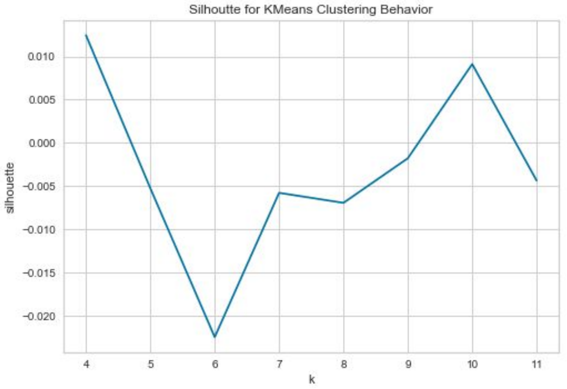
K-selection with Yellowbrick



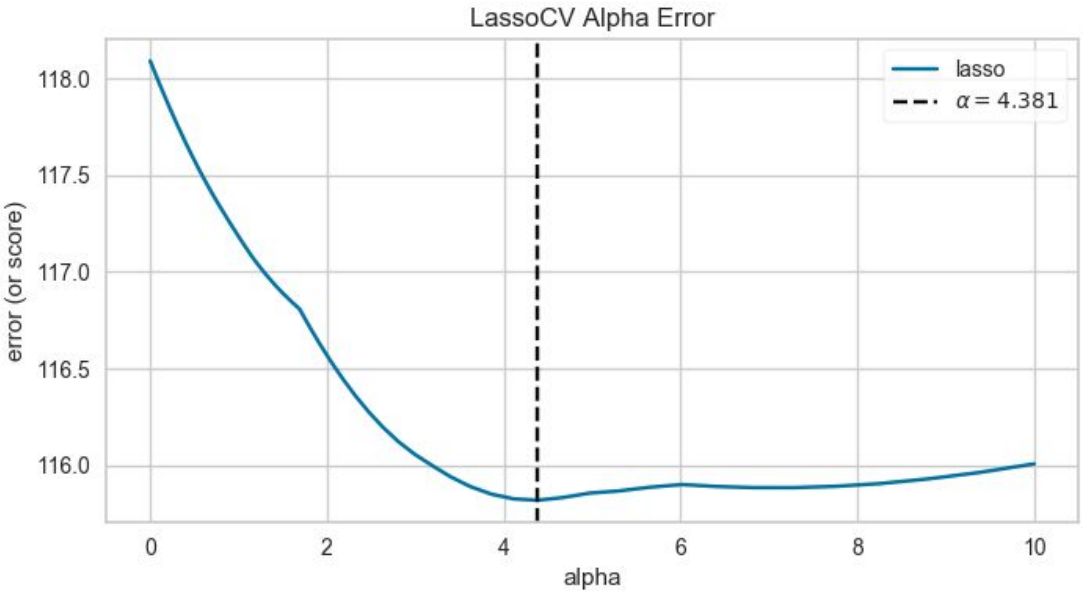
higher silhouette scores
mean denser, more
separate clusters

The elbow
shows the
best value
of k...
or suggests
a different
algorithm





Alpha selection with Yellowbrick



Should I use Lasso, Ridge, or ElasticNet? Is regularlization even working?

More alpha => less complexity

Reduced bias, but increased variance

How to launch the third lab:

```
jupyter notebook hyperparameter-tuning.ipynb
```

Contributing

Yellowbrick is an open source project supported by a community who will gratefully and humbly accept any contributions you might make to the project.

We want to be the **best first place** to contribute.

Star us on GitHub! bit.ly/yb-repo

Read the docs: bit.ly/scikit-yb