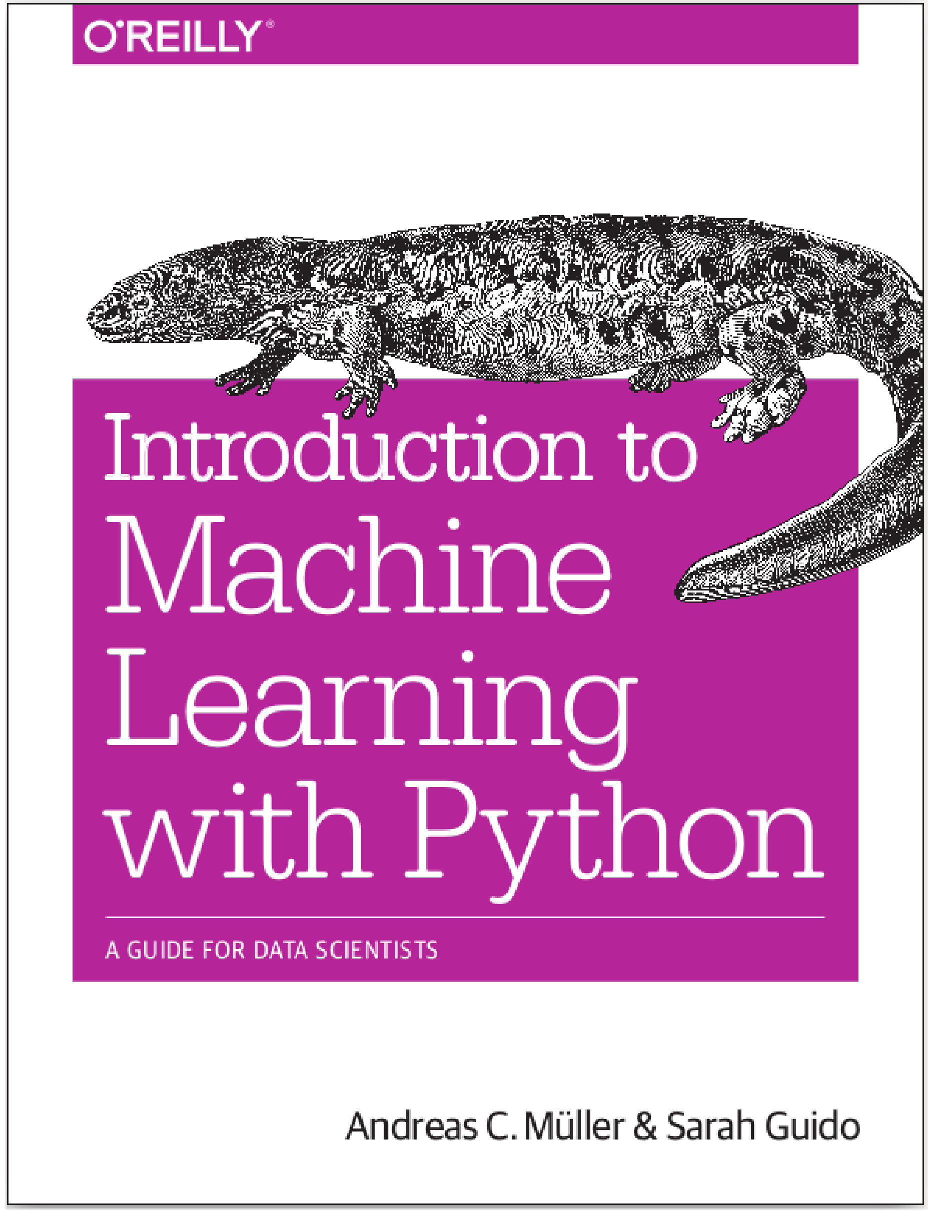
Introduction to Machine Learning with Scikit-Learn

Andreas Müller
Columbia University, scikit-learn

https://github.com/amueller/ml-training-intro

scikit learn

Columbia University in the City of New York



Lecture series:
http://www.cs.columbia.edu/~amueller/comsw4995s18/schedule/

What is machine learning?

Types of machine learning:
- supervised
- unsupervised
- reinforcement

Supervised Learning

$$(x_i, y_i) \propto p(x, y)$$ i.i.d.

$$x_i \in \mathbb{R}^n$$
$$y_i \in \mathbb{R}$$

$$f(x_i) \approx y_i$$

Classification and Regression

Classification:

- y discrete

Will you pass?

Regression:

- y continuous

How many points will you get in the exam?

Generalization

Not only $$f(x_i) \approx y_i$$

Also for new data: $$f(x) \approx y$$

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

Alexandre Gramfort
agramfort

Alexander Fabisch
AlexanderFabisch

Alexandre Passos
alextp

Andreas Mueller
amueller

Arnaud Joly
arjoly

Brian Holt
bdholt1

bthirion
bthirion

Chris Filo Gorgolewski
chrisfilo

David Cournapeau
cournape

Duchesnay
duchesnay

David Warde-Farley
dwf

Fabian Pedregosa
fabianp

Gael Varoquaux
GaelVaroquaux

Gilles Louppe
glouppe

Jake Vanderplas
jakevdp

Jaques Grobler
jaquesgrobler

Jan Hendrik Metzen
jmetzen

Jacob Schreiber
jmschrei

Joel Nothman
jnothman

Kyle Kastner
kastnerkyle

Lars
larsmans

Loïc Estève
lesteve

Shiqiao Du
lucidfrontier45

Mathieu Blondel
mblondel

Manoj Kumar
MechCoder

Noel Dawe
ndawe

Nelle Varoquaux
NelleV

Olivier Grisel
ogrisel

Paolo Losi
paolo-losi

Peter Prettenhofer
pprett

(Venkat) Raghav (Rajagopalan)
raghavrv

Robert Layton
robertlayton

Ron Weiss
ronw

Satrajit Ghosh
satra

sklearn-ci

sklearn-wheels

Tom Dupré la Tour
TomDLT

Vlad Niculae
vene

Virgile Fritsch
VirgileFritsch

Vincent Michel
vmichel

Wei Li
weilinear

Yaroslav Halchenko
yarikoptic

Get the notebooks!

nbviewer FAQ IPython

pydata-nyc-advanced-sklearn / Chapter 0 - Reminder.ipynb /

Scikit-Learn is simple

Classification

In [4]:
```
from sklearn.datasets import load_iris
from sklearn.cross_validation import train_test_split


iris = load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

In [5]:
```
from sklearn.svm import SVC
clf = SVC()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

Transformations

In [6]:
```
from sklearn.decomposition import PCA
```

In [7]:
```
pca = PCA(n_components=2)
pca.fit(X)
X_pca = pca.transform(X)
```

Tools

Cross-validation scoring

In [30]:
```
from sklearn.cross_validation import cross_val_score, StratifiedKFold
scores = cross_val_score(SVC(), X_train, y_train, cv=5)
print(scores)
```

```
[ 0.95652174  1.          0.95652174  0.91304348  0.9       ]
```

https://github.com/amueller/ml-training-intro

Home Installation Documentation Examples

Google™ Custom Search

Search ×

Documentation of scikit-learn 0.17

Quick Start

A very short introduction into machine learning problems and how to solve them using scikit-learn. Introduced basic concepts and conventions.

User Guide

The main documentation. This contains an in-depth description of all algorithms and how to apply them.

Other Versions

- scikit-learn 0.18 (development)
- scikit-learn 0.17 (stable)
- scikit-learn 0.16
- scikit-learn 0.15

Tutorials

Useful tutorials for developing a feel for some of scikit-learn's applications in the machine learning field.

API

The exact API of all functions and classes, as given by the docstrings. The API documents expected types and allowed features for all functions, and all parameters available for the algorithms.

Additional Resources

Talks given, slide-sets and other information relevant to scikit-learn.

Contributing

Information on how to contribute. This also contains useful information for advanced users, for example how to build their own estimators.

Flow Chart

A graphical overview of basic areas of machine learning, and guidance which kind of algorithms to use in a given situation.

FAQ

Frequently asked questions about the project and contributing.

http://scikit-learn.org/

Representing Data

one sample

$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix} \qquad y = \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

one feature

outputs / labels

Training and Testing Data

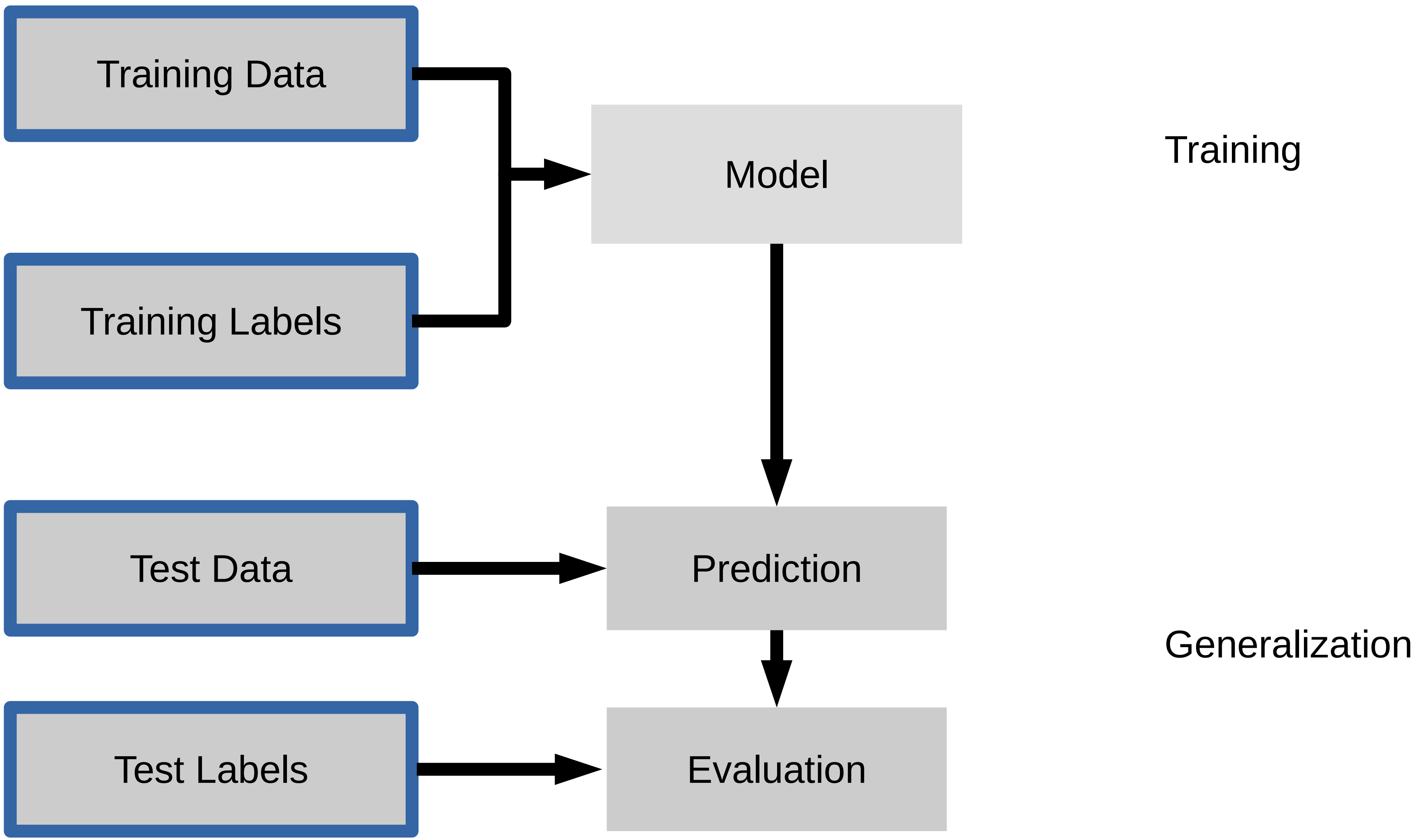
training set

$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix} \qquad y = \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

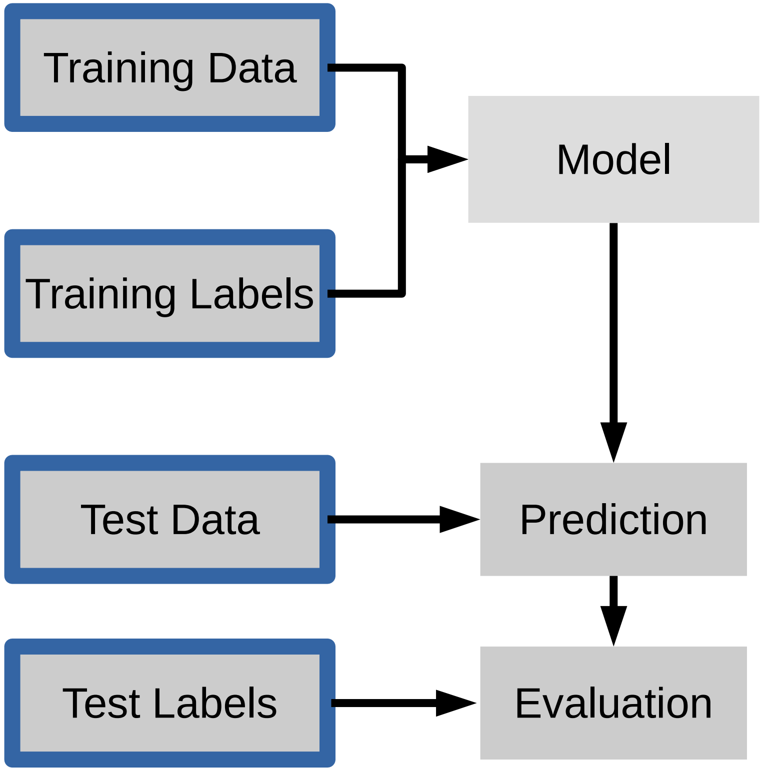
test set

IPython Notebook:
Part 0 – Data Loading

Supervised Machine Learning

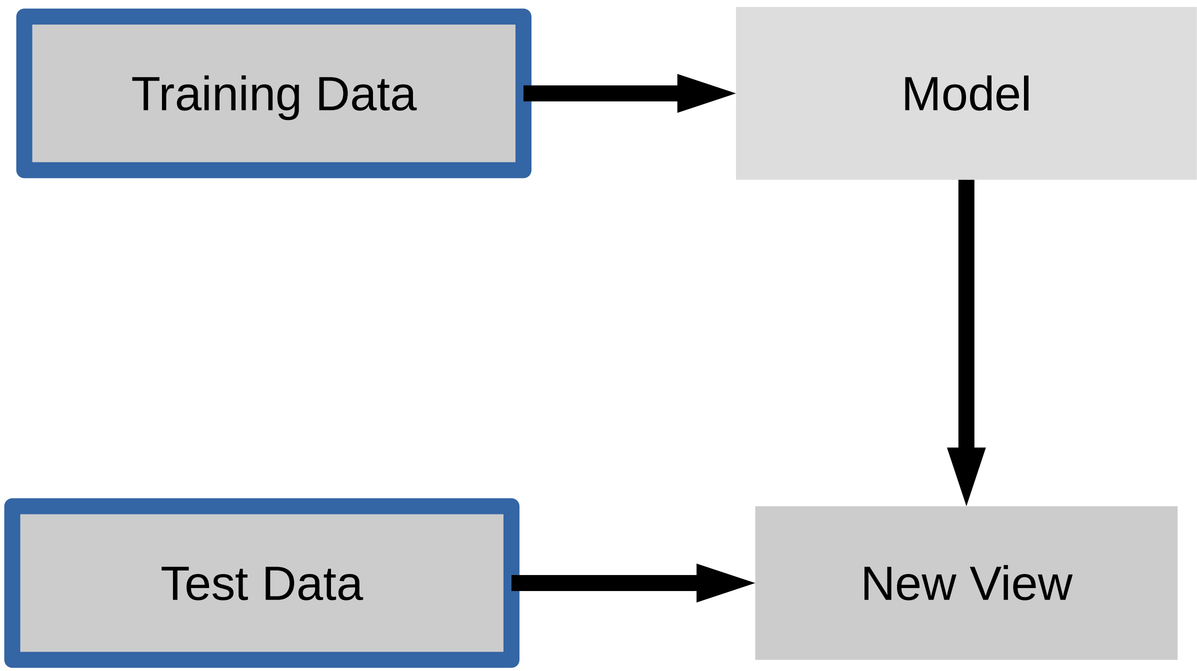


```python
clf = RandomForestClassifier()

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

clf.score(X_test, y_test)
```



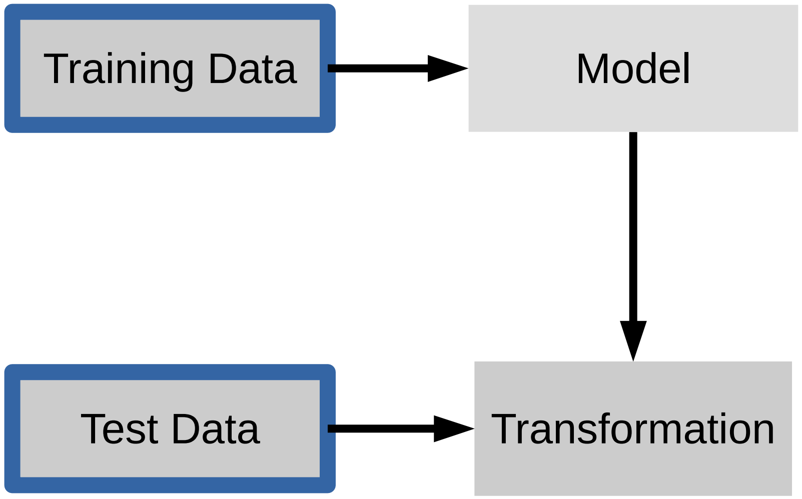
IPython Notebook:
Part 1 - Introduction to Scikit-learn

Unsupervised Machine Learning



Unsupervised Transformations

```
pca = PCA()

pca.fit(X_train)

X_new = pca.transform(X_test)
```



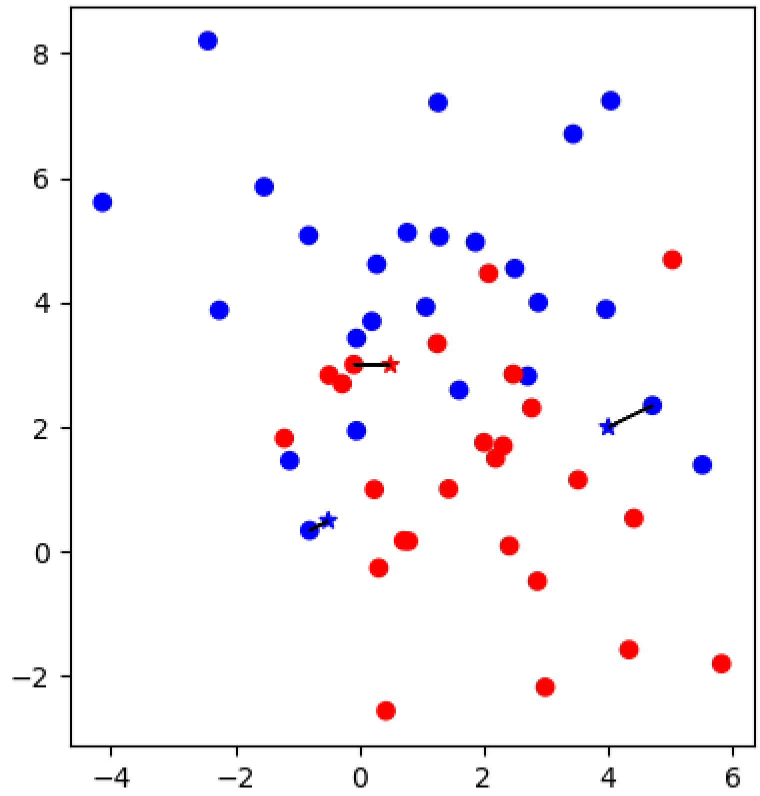
IPython Notebook:
Part 2 – Unsupervised Transformers

Basic API

`estimator.fit(X, [y])`

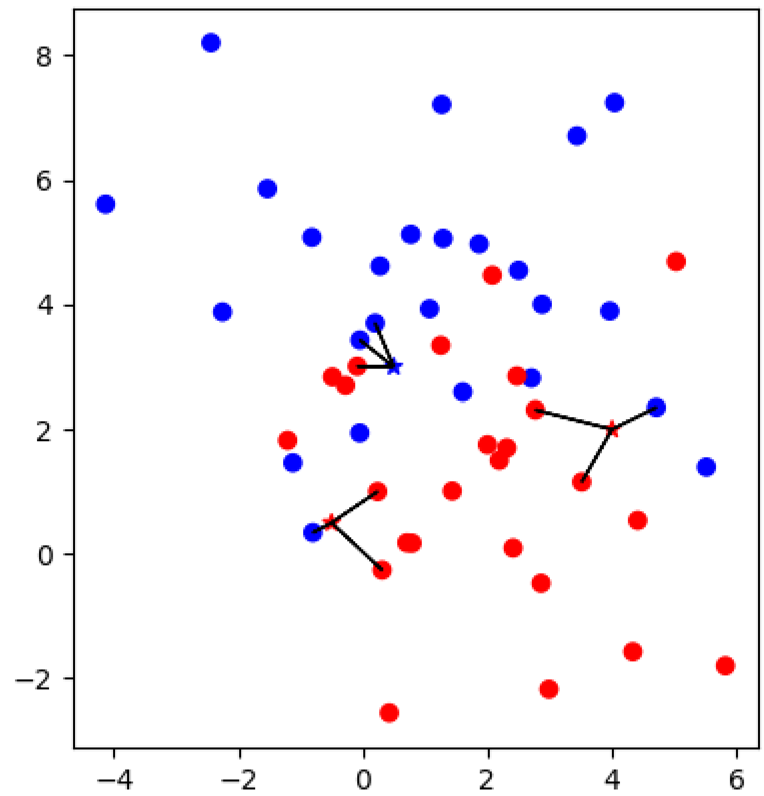
`estimator.predict`	**`estimator.transform`**
Classification	Preprocessing
Regression	Dimensionality reduction
Clustering	Feature selection
	Feature extraction

Nearest neighbors

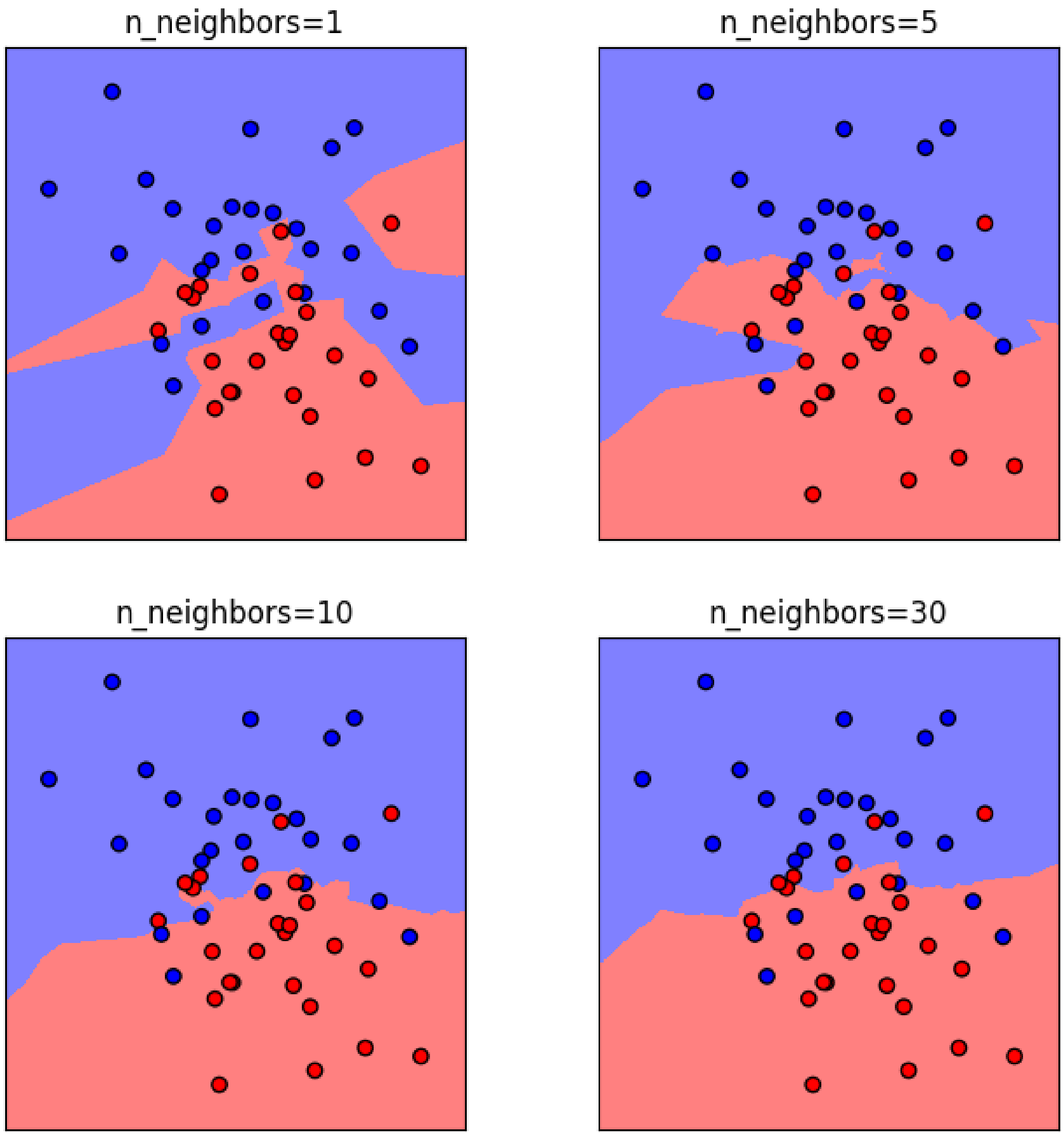


$$f(x) = y_i, i = \text{argmin}_j ||x_j - x||$$

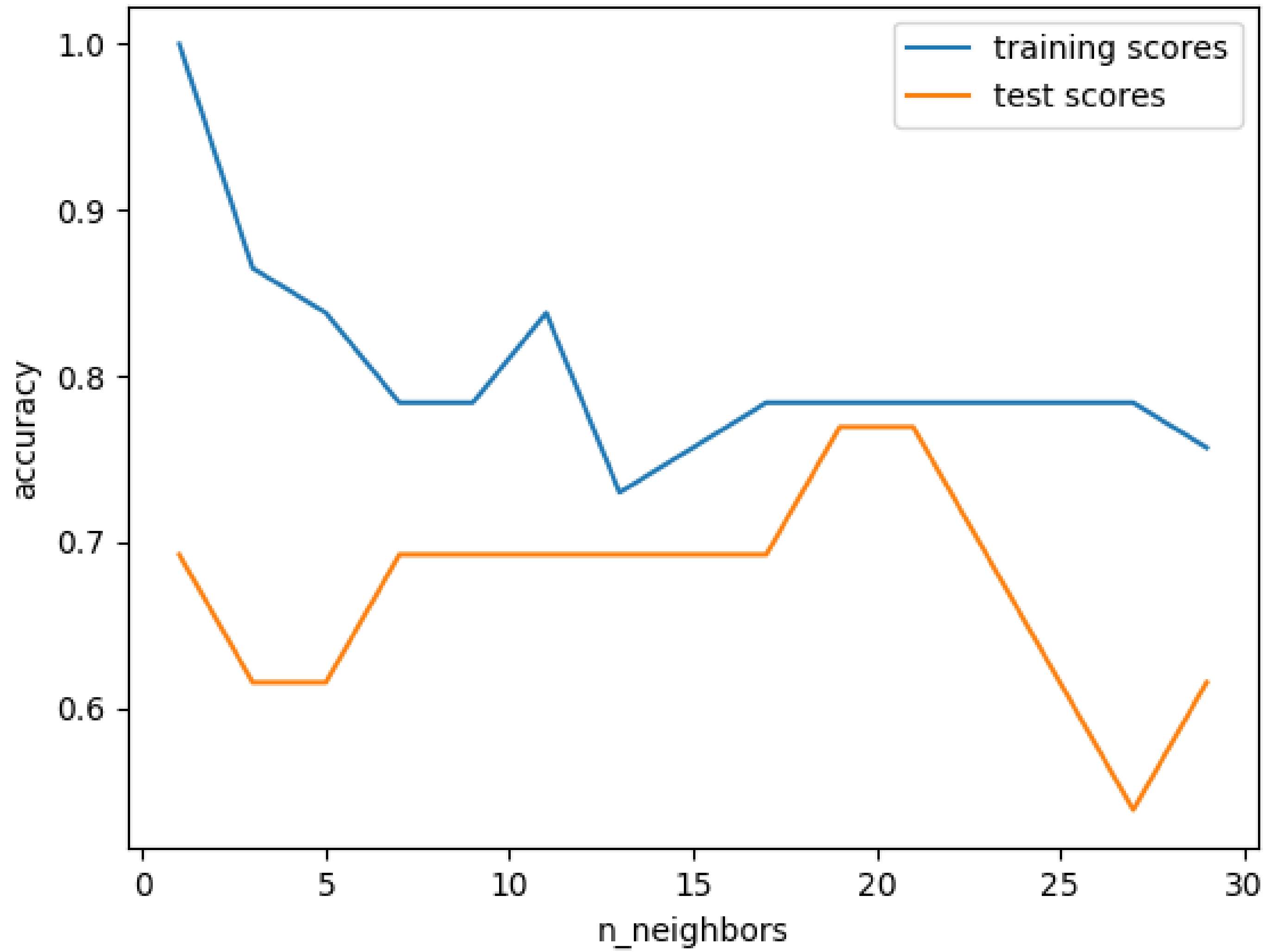
Nearest neighbors



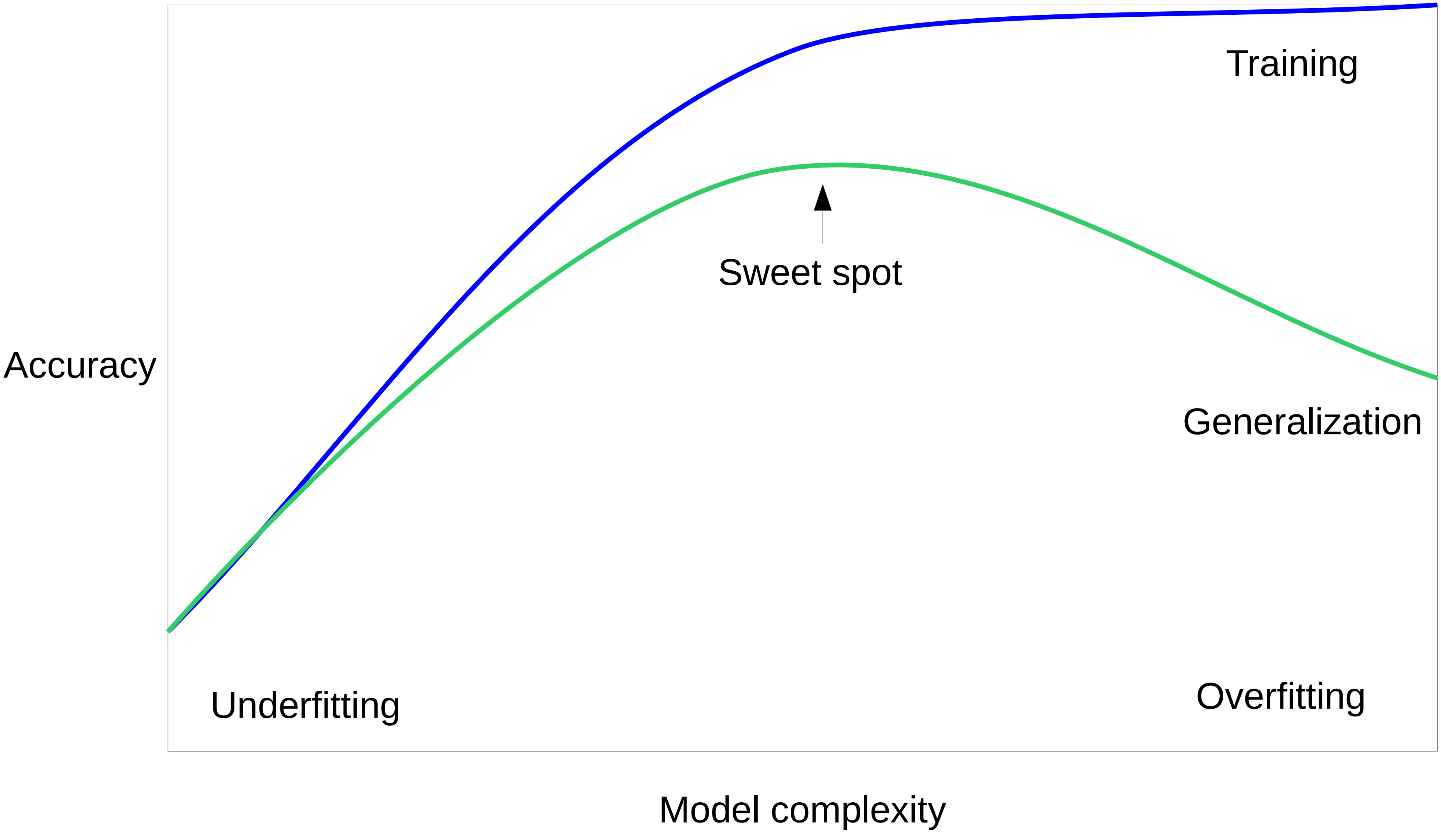
Influence of n_neighbors



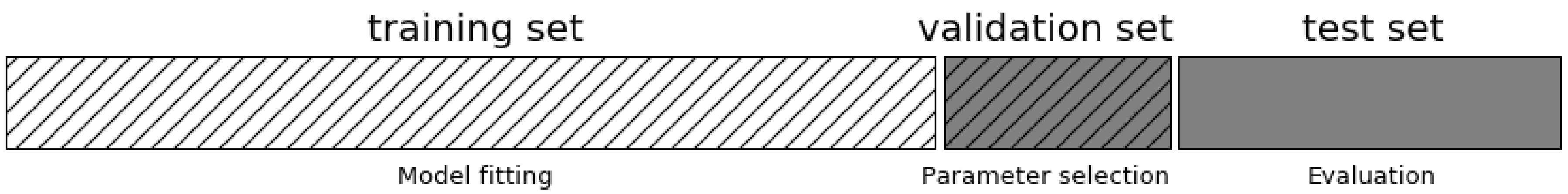
Model Complexity



Overfitting and Underfitting



Three-fold split

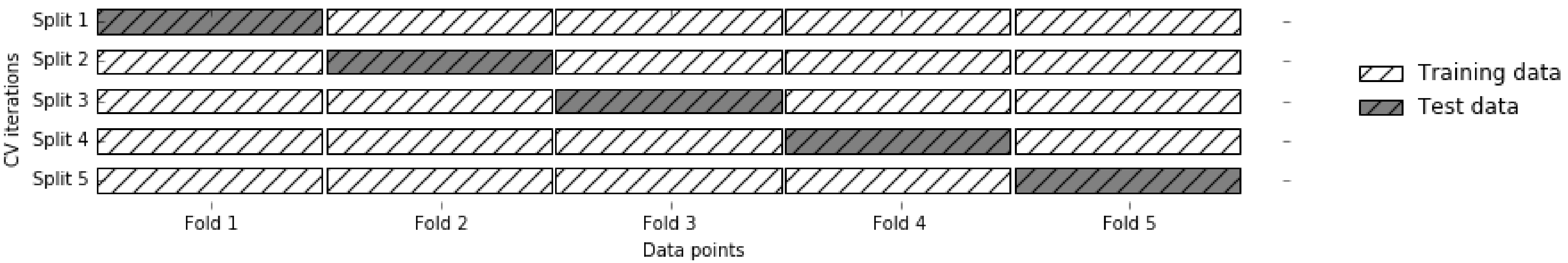


pro: fast, simple
con: high variance, bad use of data.

```python
val_scores = []
neighbors = np.arange(1, 15, 2)
for i in neighbors:
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    val_scores.append(knn.score(X_val, y_val))
print("best validation score: {:.3f}".format(np.max(val_scores)))
best_n_neighbors = neighbors[np.argmax(val_scores)]
print("best n_neighbors: {}".format(best_n_neighbors))

knn = KNeighborsClassifier(n_neighbors=best_n_neighbors)
knn.fit(X_trainval, y_trainval)
print("test-set score: {:.3f}".format(knn.score(X_test, y_test)))
```

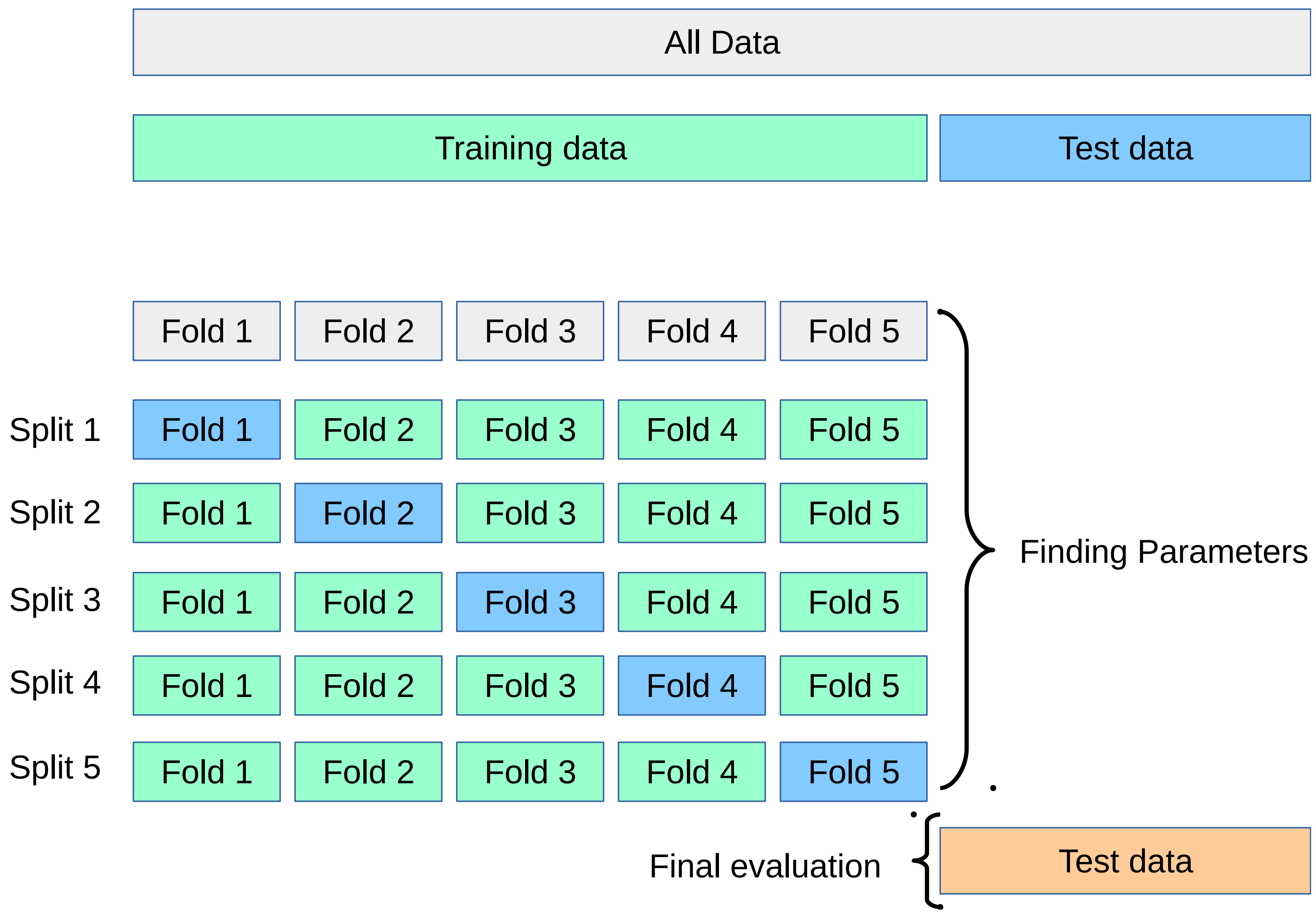
```
best validation score: 0.972
best n_neighbors: 3
test-set score: 0.965
```

Cross-validation



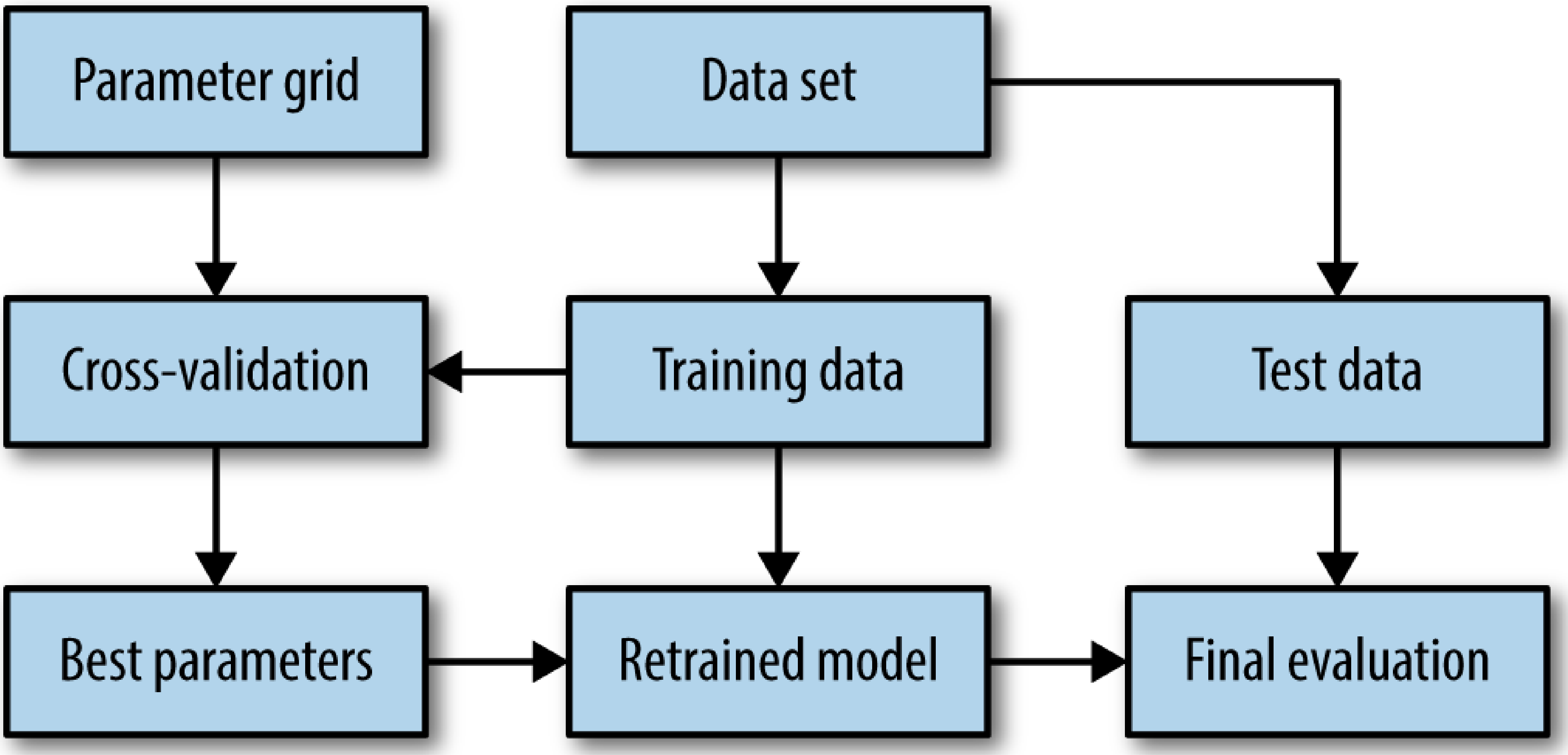
Pro: more stable, more data
con: slower

Cross-validation + test-set



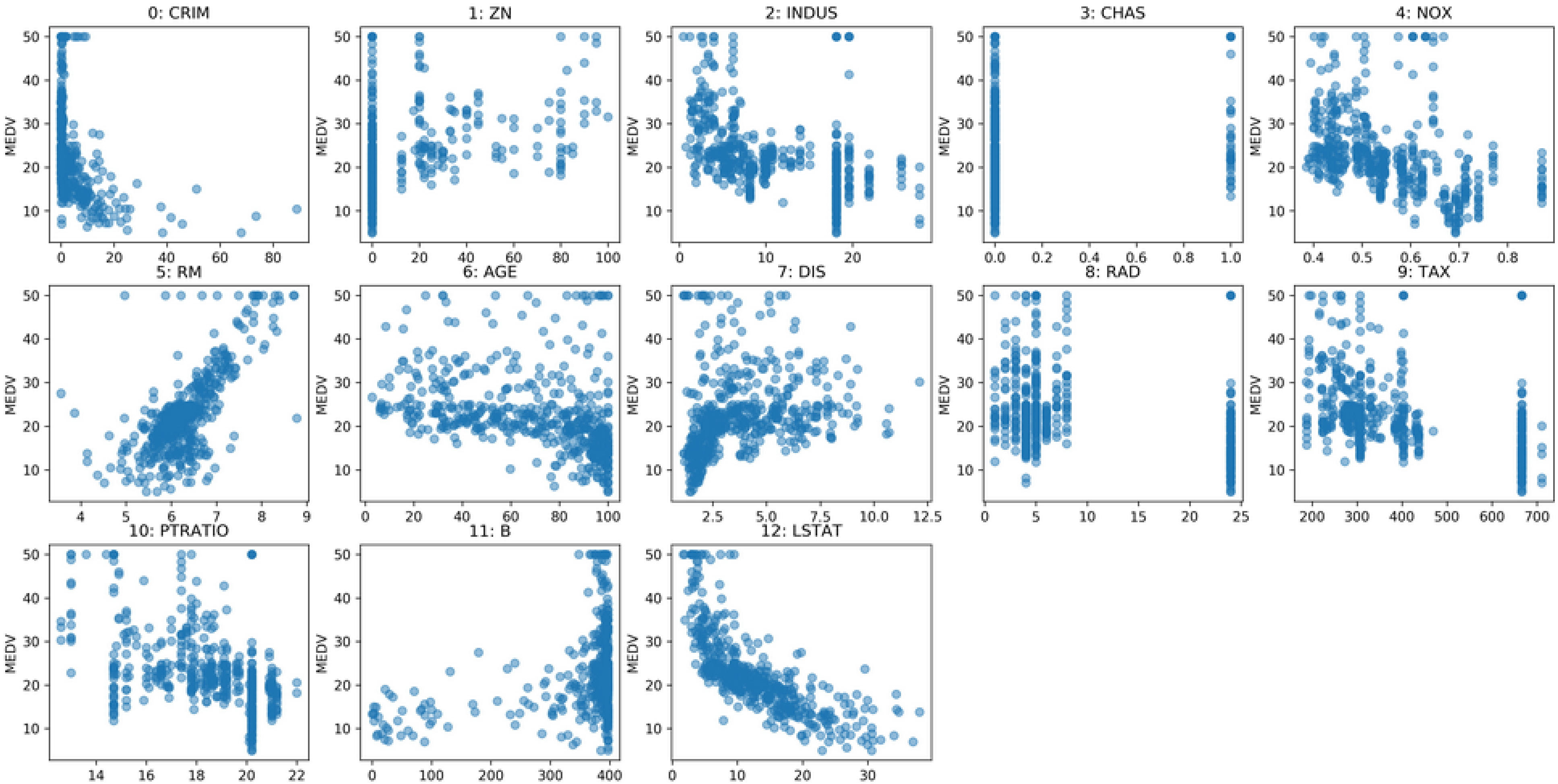
```python
from sklearn.model_selection import cross_val_score

X_train, X_test, y_train, y_test = train_test_split(X, y)

cross_val_scores = []

for i in neighbors:
    knn = KNeighborsClassifier(n_neighbors=i)
    scores = cross_val_score(knn, X_trainval, y_trainval, cv=10)
    cross_val_scores.append(np.mean(scores))

print("best cross-validation score: {:.3f}".format(np.max(cross_val_scores)))
best_n_neighbors = neighbors[np.argmax(cross_val_scores)]
print("best n_neighbors: {}".format(best_n_neighbors))

knn = KNeighborsClassifier(n_neighbors=best_n_neighbors)
knn.fit(X_train, y_train)
print("test-set score: {:.3f}".format(knn.score(X_test, y_test)))
```

```
best cross-validation score: 0.972
best n_neighbors: 3
test-set score: 0.972
```


Parameter grid
Data set
Cross-validation
Training data
Test data
Best parameters
Retrained model
Final evaluation

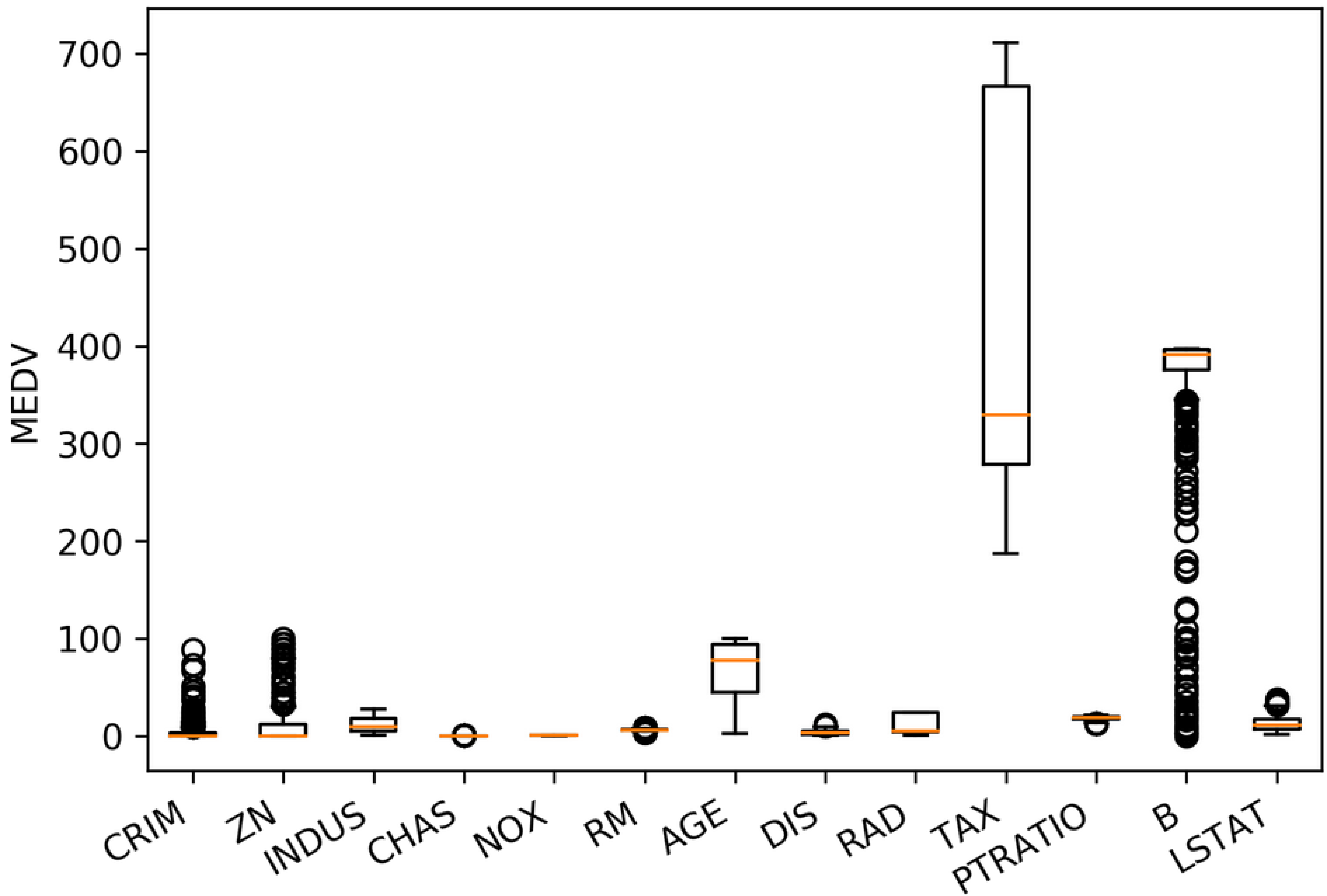
IPython Notebook:
Part 3 – Cross-validation and grid-search

Preprocessing


0: CRIM
1: ZN
2: INDUS
3: CHAS
4: NOX
5: RM
6: AGE
7: DIS
8: RAD
9: TAX
10: PTRATIO
11: B
12: LSTAT
MEDV

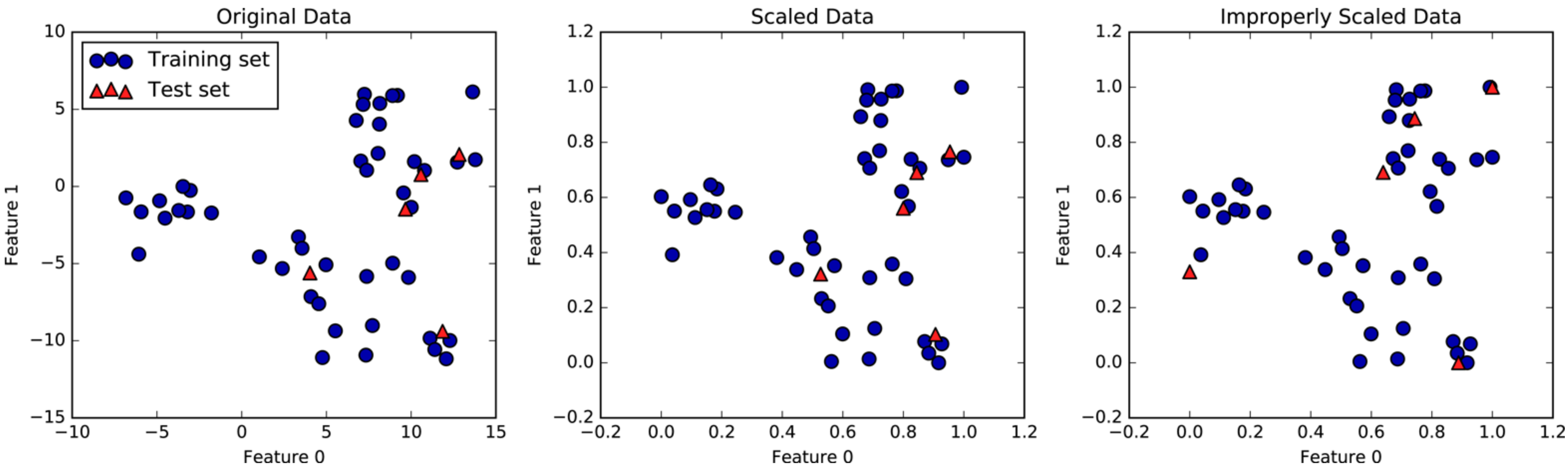
```
plt.boxplot(X)
plt.xticks(np.arange(1, X.shape[1] + 1), boston.feature_names, rotation=30, ha="right")
plt.ylabel("MEDV")
```

```
<matplotlib.text.Text at 0x7f580303eac8>
```



```python
from sklearn.linear_model import Ridge
X, y = boston.data, boston.target
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

scaler = StandardScaler()
scaler.fit(X_train)
X_train_scaled = transform(X_train)
ridge = Ridge().fit(X_train_scaled, y_train)

X_test_scaled = scaler.transform(X_test)
ridge.score(X_test_scaled, y_test)
```

```
0.6344846877867426
```


Original Data
Training set
Test set
Feature 0
Feature 1
Scaled Data
Feature 0
Feature 1
Improperly Scaled Data
Feature 0
Feature 1

Categorical Features

Categorical Features

$$\{\text{"red"}, \text{"green"}, \text{"blue"}\} \subset \mathbb{R}^p \quad ?$$

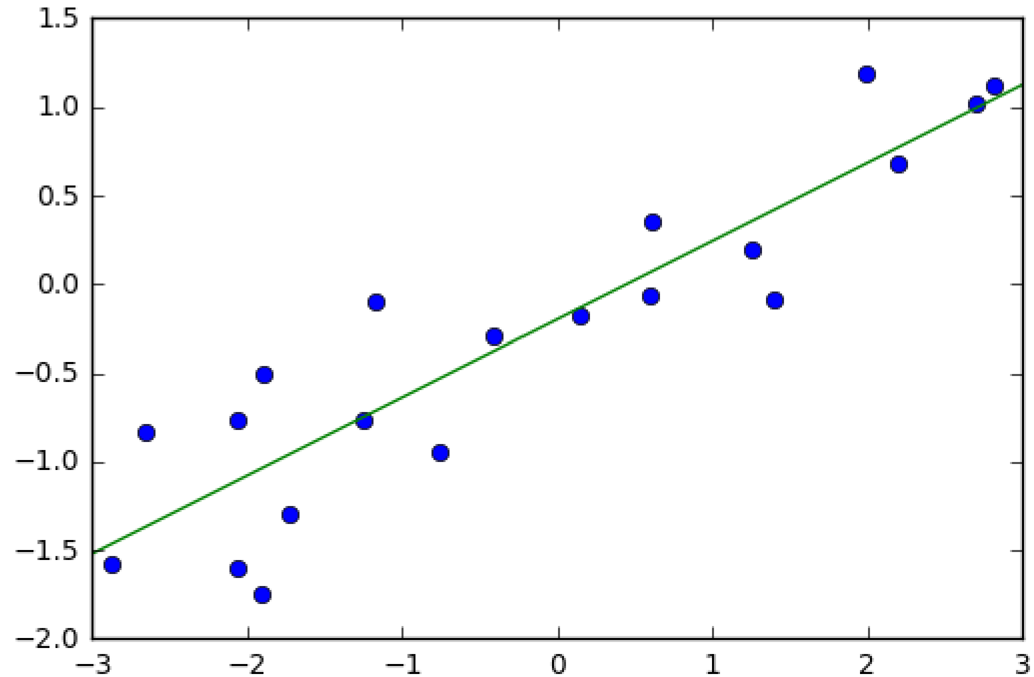
Categorical Variables

$$\begin{array}{ccc} \text{“red”} & \text{“green”} & \text{“blue”} \end{array}$$
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

IPython Notebook:
Part 4 – Preprocessing

Linear Models for Regression

Linear Models for Regression



$$\hat{y} = w^T \mathbf{x} + b = \sum_{i=1}^{p} w_i x_i + b$$

Linear Regression
Ordinary Least Squares

$$\hat{y} = w^T \mathbf{x} + b = \sum_{i=1}^{p} w_i x_i + b$$

$$\min_{w \in \mathbb{R}^p} \sum_{i=1}^{p} ||w^T \mathbf{x}_i - y_i||^2$$

Unique solution if $\mathbf{X} = (\mathbf{x}_1, ... \mathbf{x}_n)^T$ has full rank.

Ridge Regression

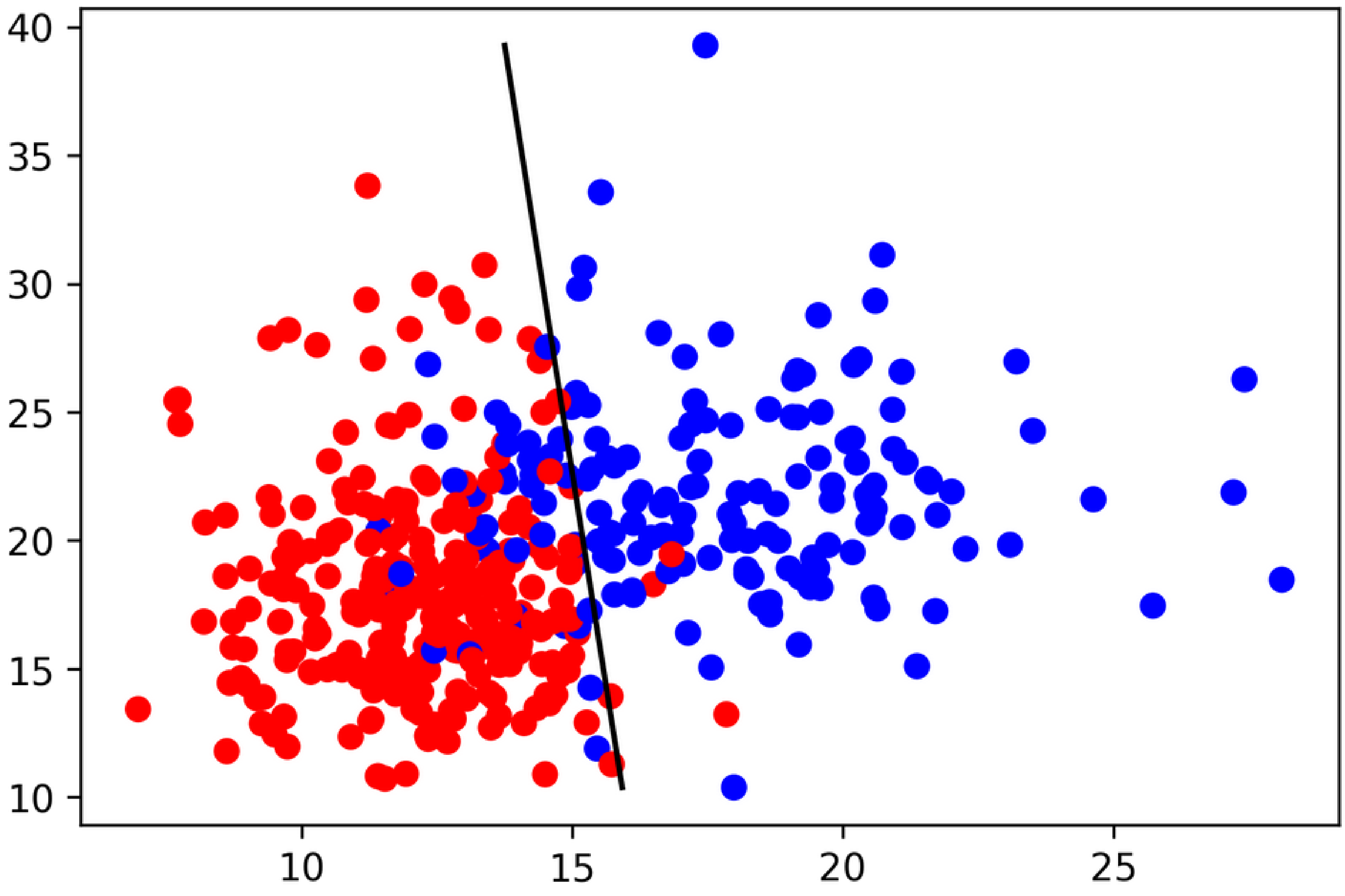
$$\min_{w \in \mathbb{R}^p} \sum_{i=1}^{n} ||w^T x_i - y_i||^2 + \alpha ||w||^2$$

Always has a unique solution.
Has tuning parameter alpha

IPython Notebook:
Part 5 – Linear Models for Regression

Linear Models for Classification

Linear models for **binary** classfiication

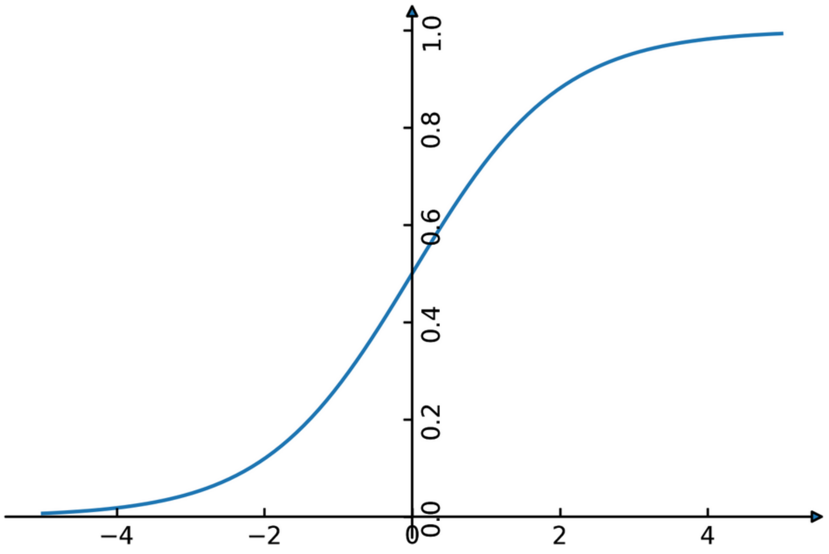


$$\hat{y} = \text{sign}(w^T \mathbf{x} + b) = \text{sign}(\sum_i w_i x_i + b)$$

Logistic Regression

$$\min_{w \in \mathbb{R}^p} -\sum_{i=1}^{n} \log(\exp(-y_i w^T \mathbf{x}_i) + 1)$$

$$p(y|\mathbf{x}) = \frac{1}{1 + e^{-w^T \mathbf{x}}}$$

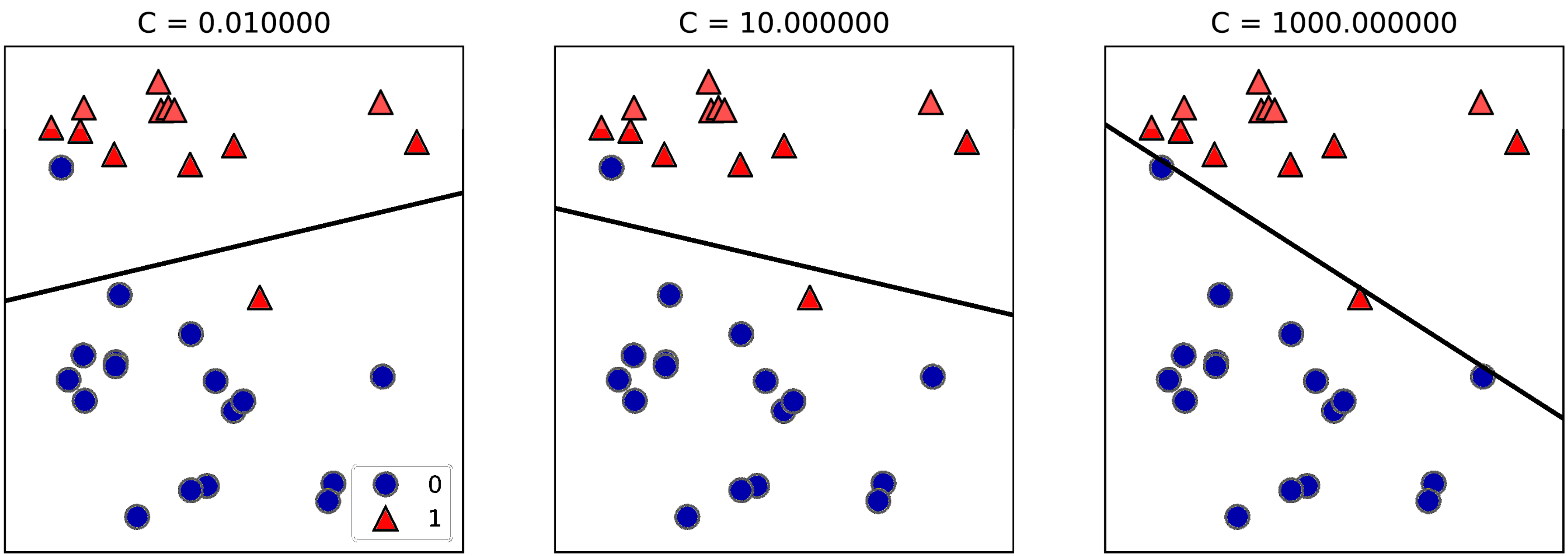


$$\hat{y} = \text{sign}(w^T \mathbf{x} + b)$$

Penalized Logistic Regression

$$\min_{w \in \mathbb{R}^p} -C \sum_{i=1}^{n} \log(\exp(-y_i w^T \mathbf{x}_i) + 1) + ||w||_2^2$$

C is inverse to alpha (or alpha / n_samples)


C = 0.010000
0
1
C = 10.000000
C = 1000.000000

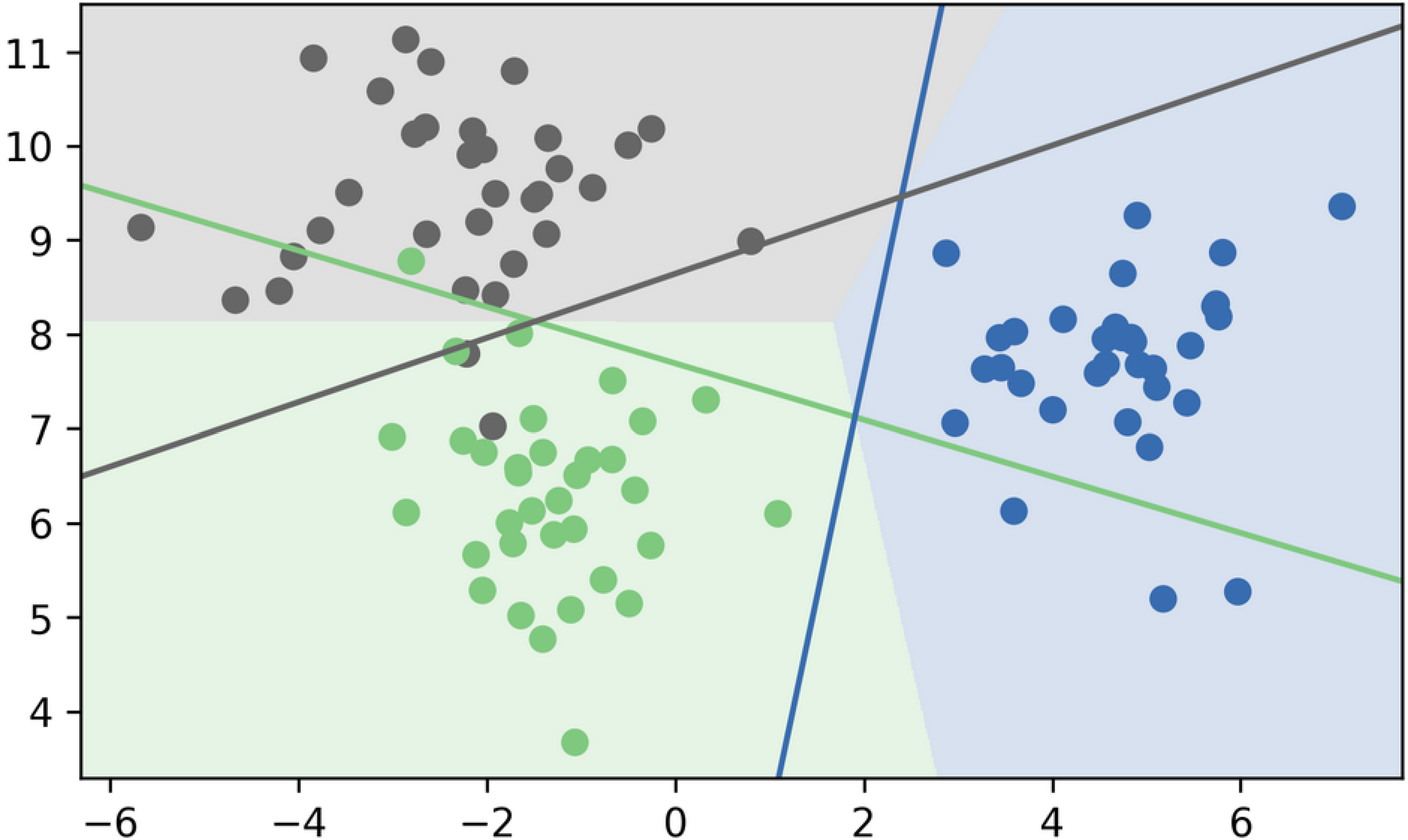
Multinomial Logistic Regression

Probabilistic multi-class model:

$$p(y = i|\mathbf{x}) = \frac{e^{-\mathbf{w}_i^T \mathbf{x}}}{\sum_{j \in Y} e^{-\mathbf{w}_j^T \mathbf{x}}}$$

$$\min_{w \in \mathbb{R}^p} - \sum_{i=1}^{n} \log(p(y = y_i|\mathbf{x}_i))$$

$$\hat{y} = \arg\max_{i \in Y} \mathbf{w}_i \mathbf{x}$$

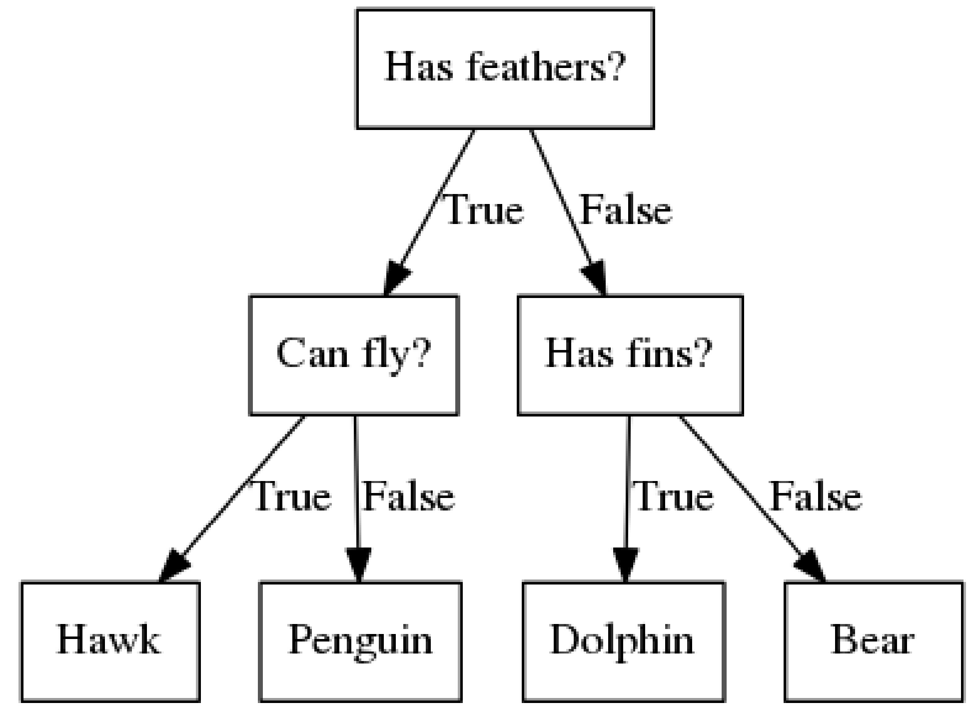


$$\hat{y} = \arg\max_{i \in Y} \mathbf{w}_i \mathbf{x}$$

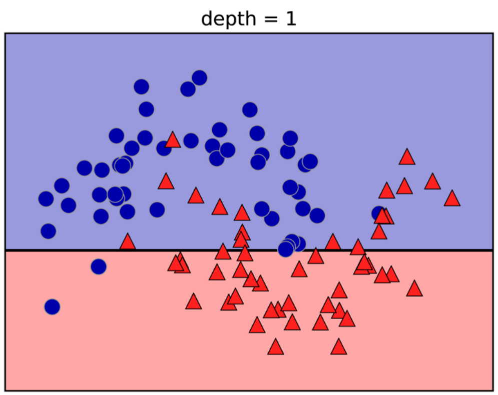
IPython Notebook:
Part 6 – Linear Models for Classification

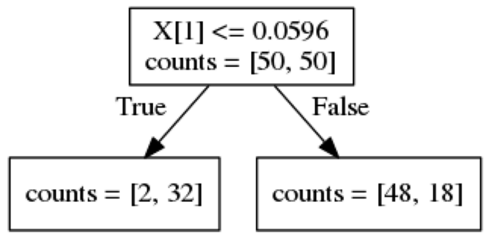
Decision Trees and Tree-based Models

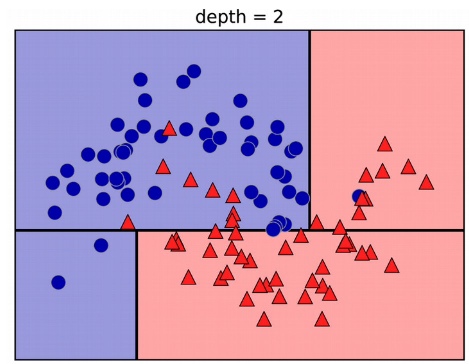
Idea: series of binary questions

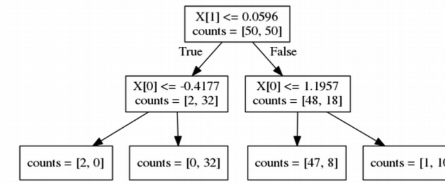


Building trees







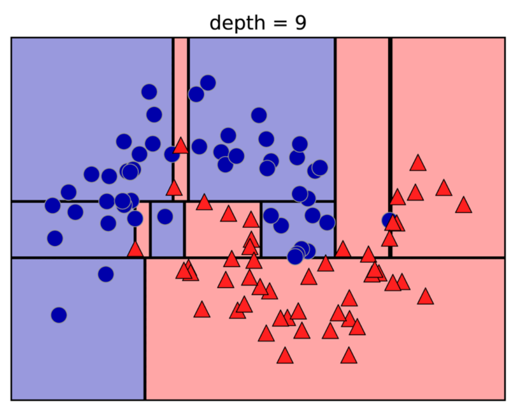


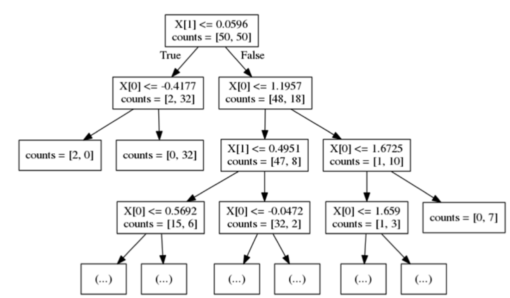
Continuous features:
"questions" are thresholds on single features.

[Other methods are possible but not as common]

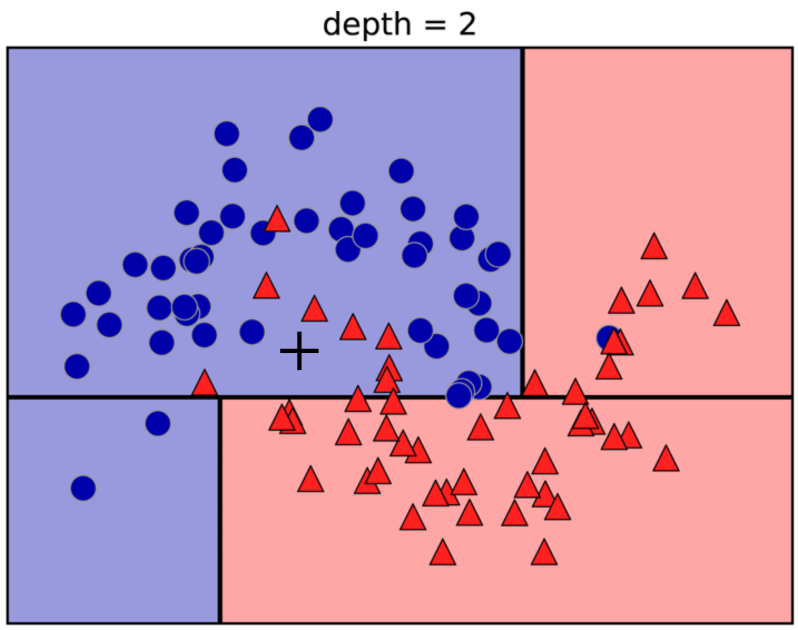
For each split:
exhaustive search over all features and thresholds!

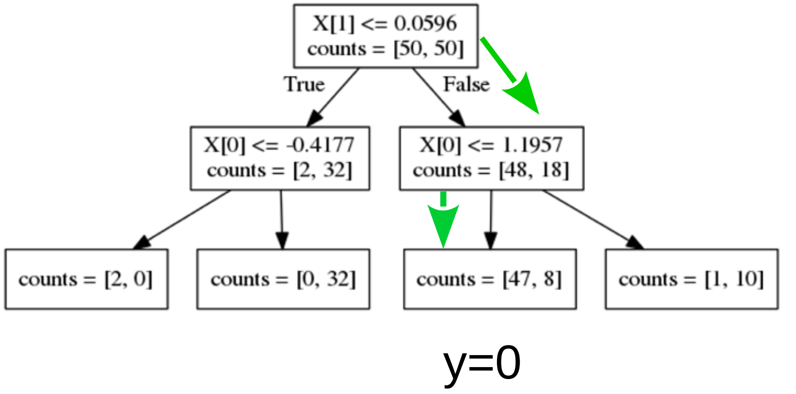
Minimize "impurity"





Prediction

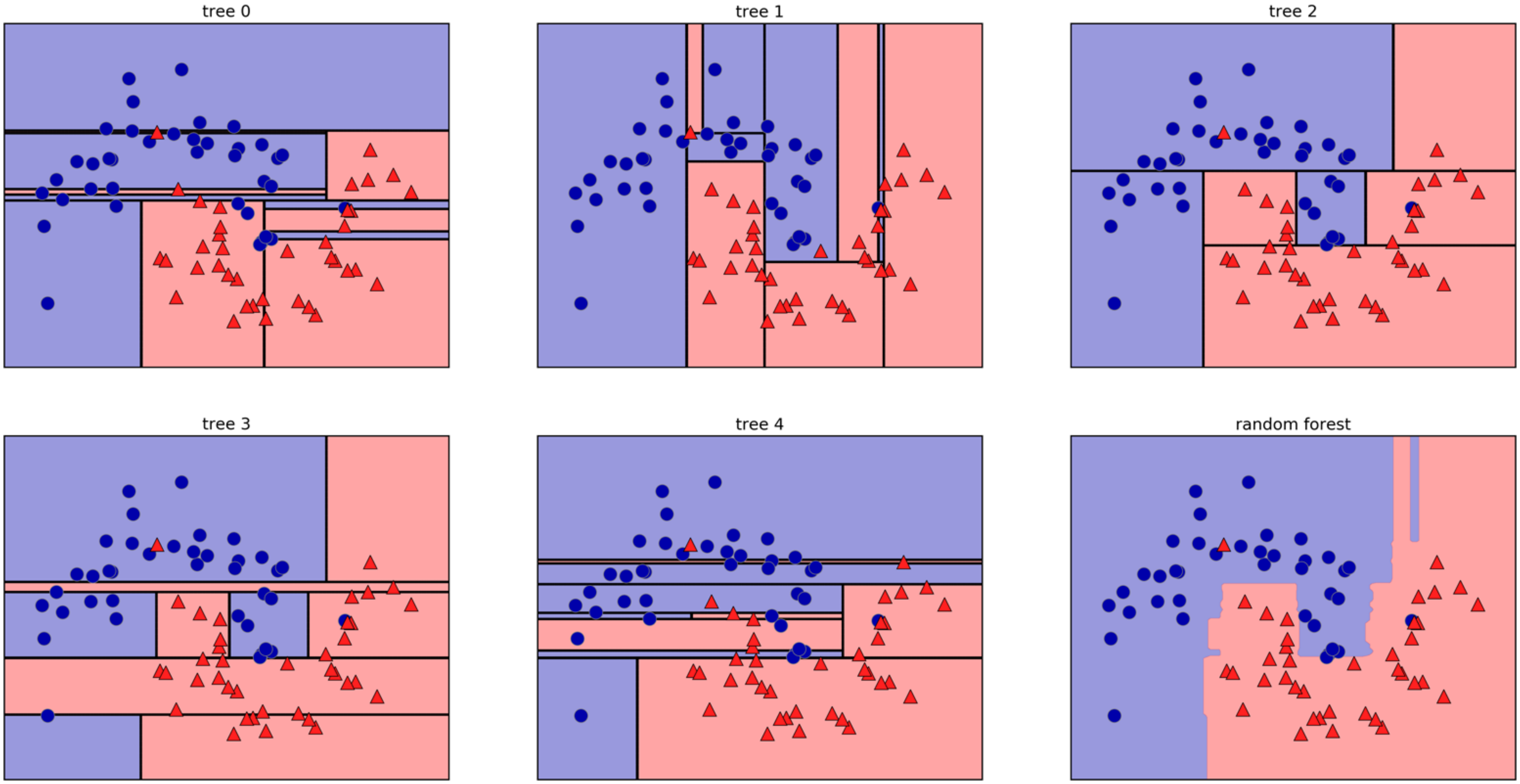




Traverse tree based on feature tests
Predict most common class in leaf

IPython Notebook:
Part 7 – Decision Trees

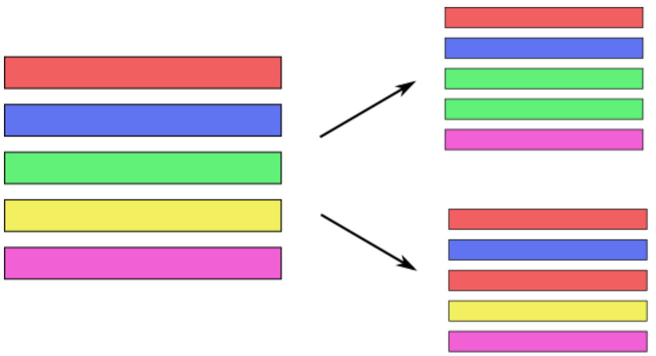
Random Forests



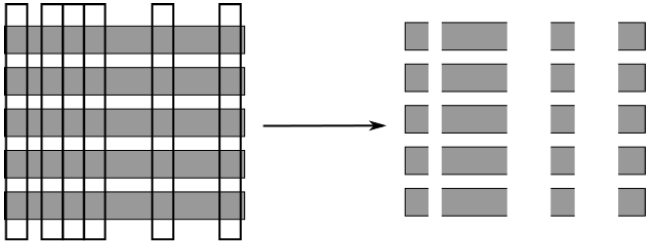
Randomize in two ways

- For each **tree**:

  Pick bootstrap sample of data



- For each **split**:
  Pick random sample of features

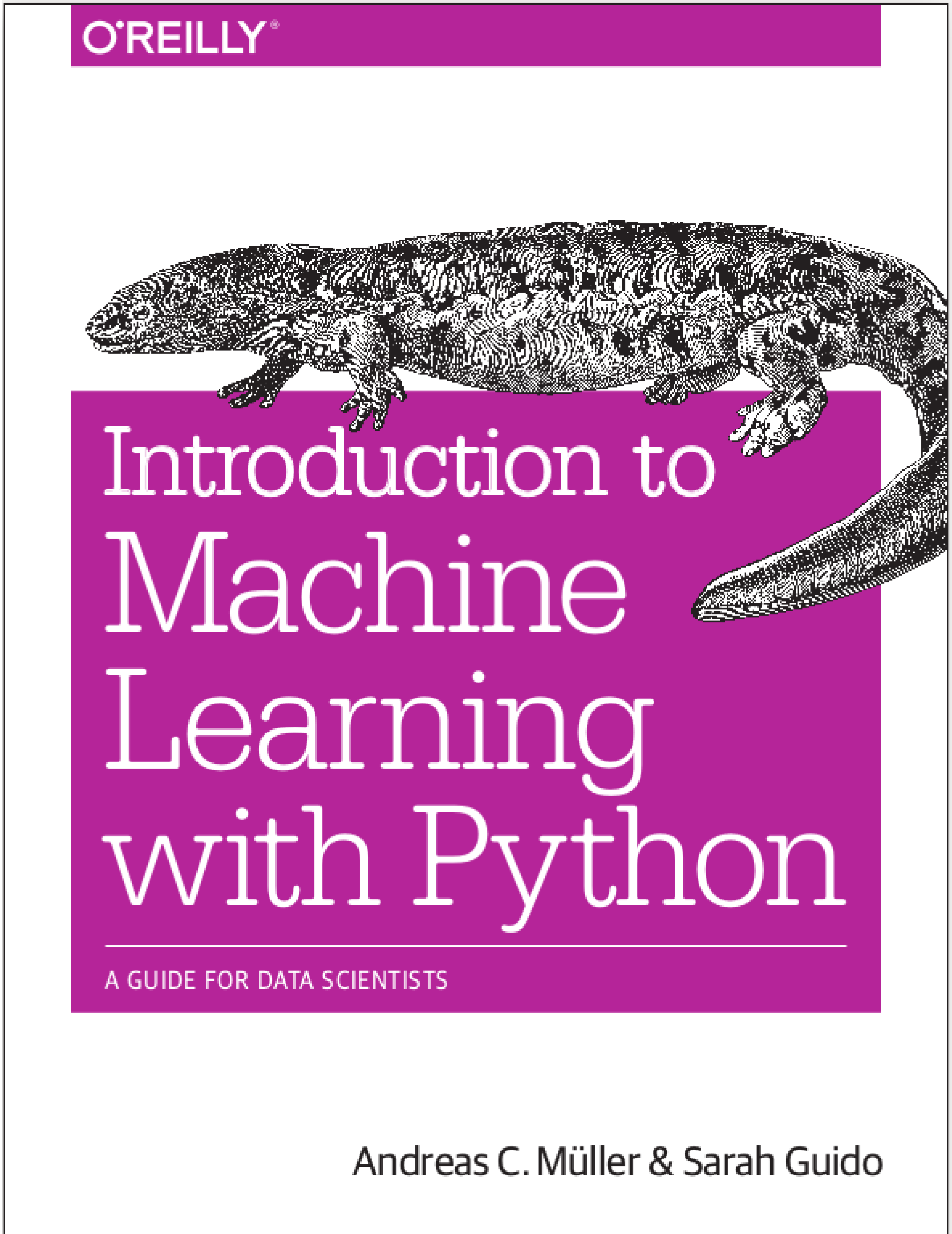


- More tree are always better

Tuning Random Forests

- Main parameter: max_features
  - around sqrt(n_features) for classification
  - Around n_features for regression

- n_estimators > 100
- Prepruning might help, definitely helps with model size!
- max_depth, max_leaf_nodes, min_samples_split again



amueller.github.io

@amuellerml

@amueller

andreas.mueller
@columbia.edu