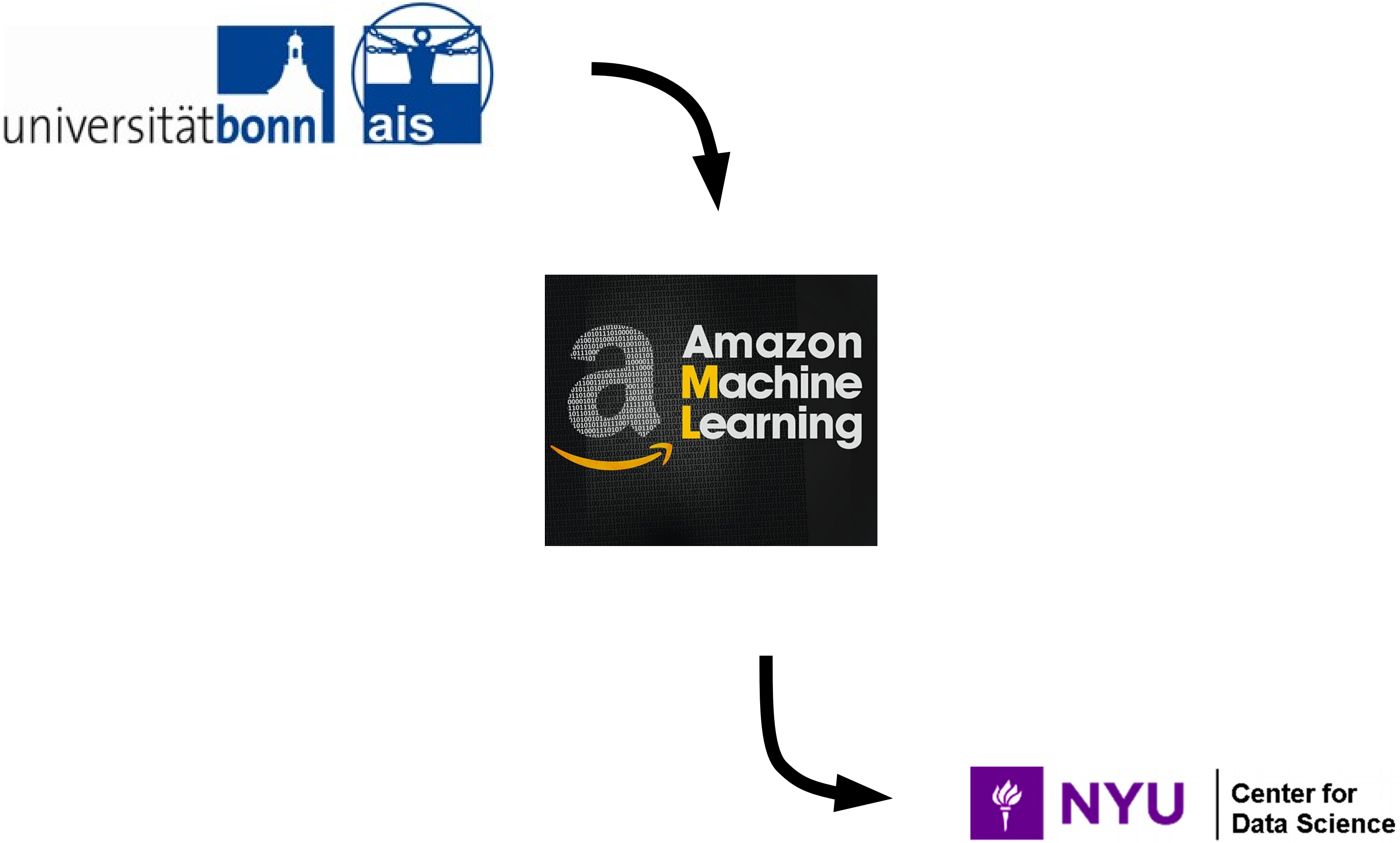
Machine Learning with Scikit-Learn

Andreas Mueller (NYU Center for Data Science, scikit-learn)

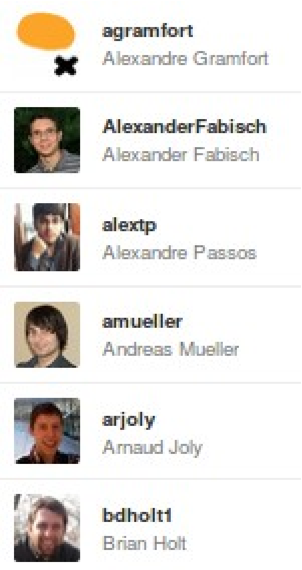
Material: http://bit.ly/sklsf

Me


universitätbonn
ais
Amazon
Machine
Learning
NYU
Center for
Data Science

scikit
learn

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

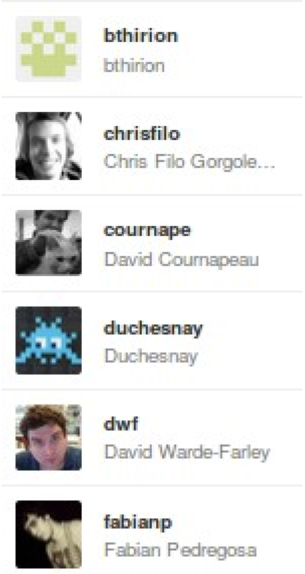
AlexanderFabisch
Alexander Fabisch

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

scikit
learn

GaelVaroquaux
Gael Varoquaux

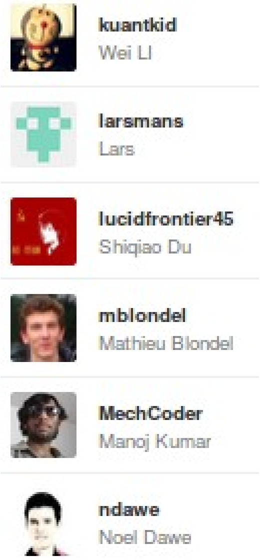
glouppe
Gilles Louppe

jakevdp
Jake Vanderplas

jaquesgrobler
Jaques Grobler

jnothman

kastnerkyle
Kyle Kastner



kuantkid
Wei LI

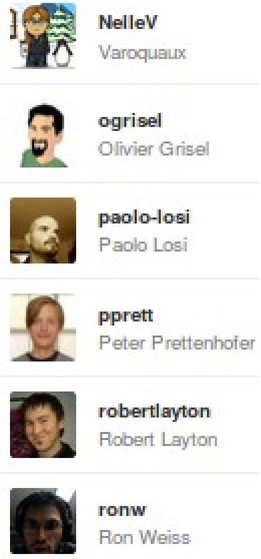
larsmans
Lars

lucidfrontier45
Shiqiao Du

mblondel
Mathieu Blondel

MechCoder
Manoj Kumar

ndawe
Noel Dawe



NelleV
Varoquaux

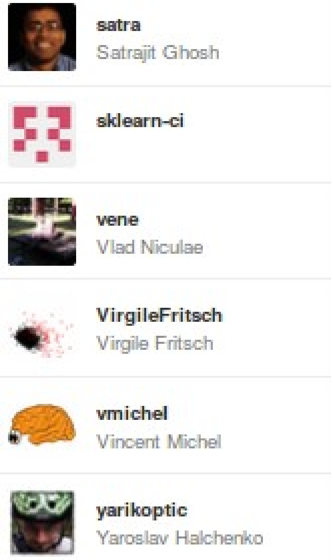
ogrisel
Olivier Grisel

paolo-losi
Paolo Losi

pprett
Peter Prettenhofer

robertlayton
Robert Layton

ronw
Ron Weiss



satra
Satrajit Ghosh

sklearn-ci

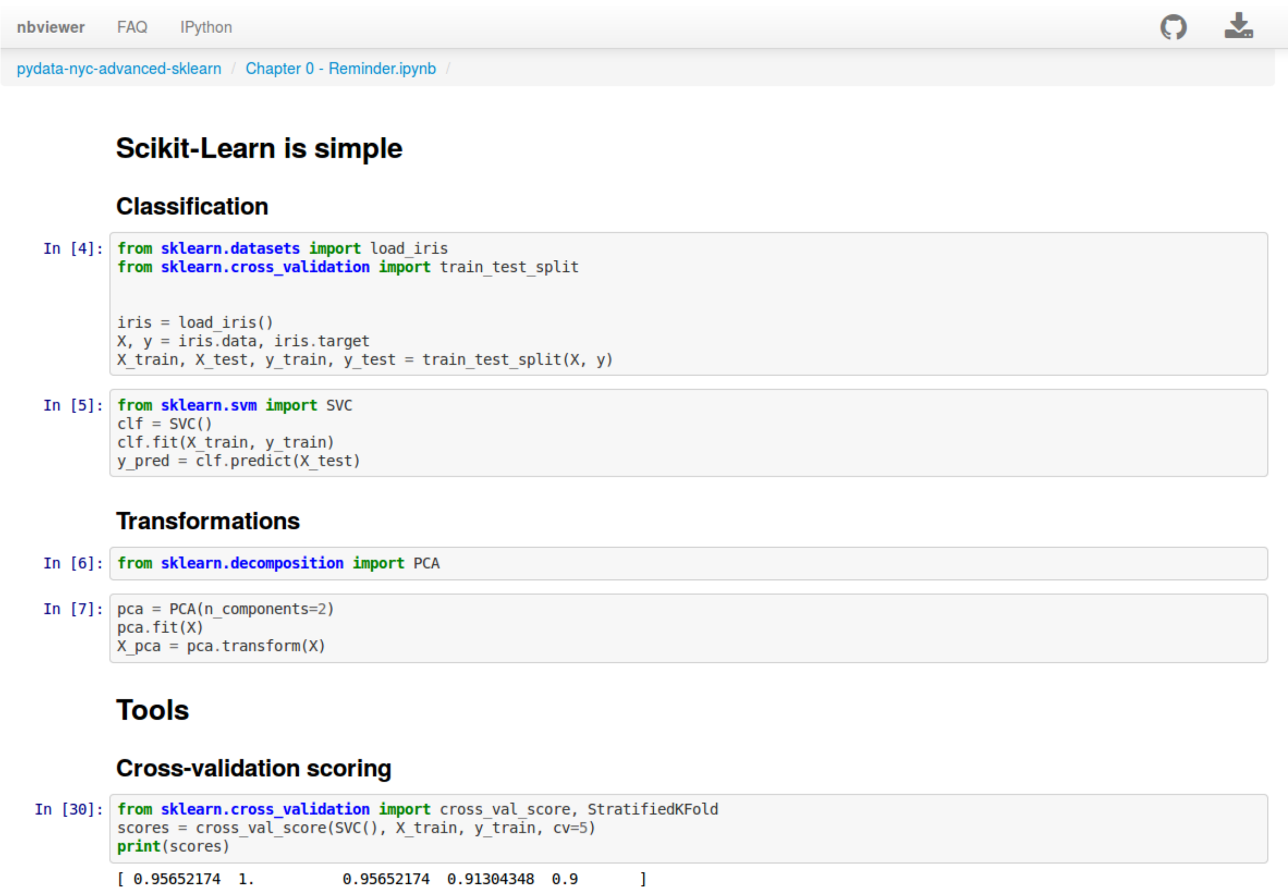
vene
Vlad Niculae

VirgileFritsch
Virgile Fritsch

vmichel
Vincent Michel

yarikoptic
Yaroslav Halchenko

Get the notebooks!



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)
```

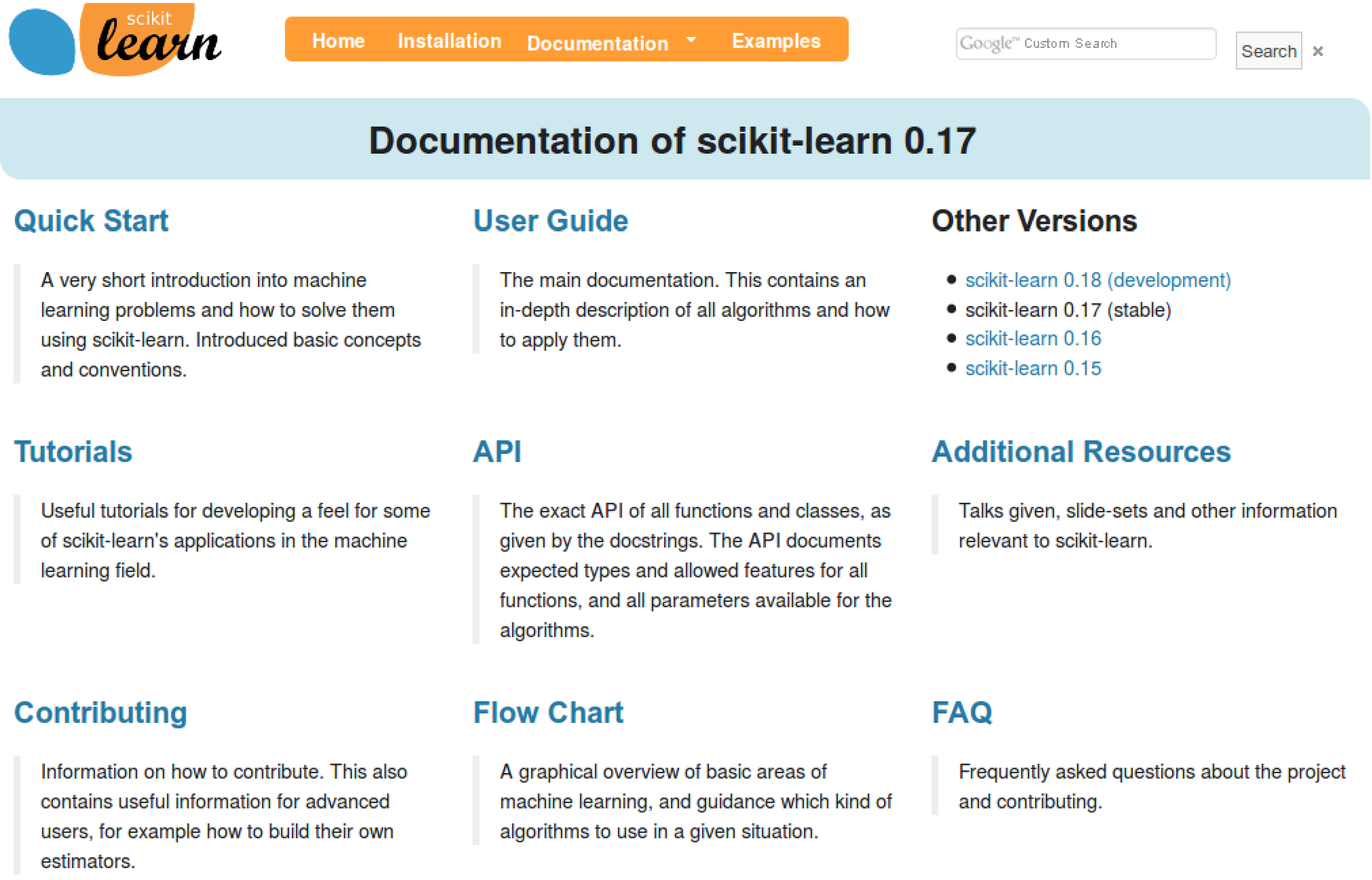
Tools

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       ]
```

http://bit.ly/sklsf



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/`

Hi Andy,

I just received an email from the first tutorial speaker, presenting right before you, saying he's ill and won't be able to make it.

I know you have already committed yourself to two presentations, but is there anyway you could increase your tutorial time slot, maybe just offer time to try out what you've taught? Otherwise I have to do some kind of modern dance interpretation of Python in data :-)
-Leah

Hi Andreas,

I am very interested in your Machine Learning background. I work for X Recruiting who have been engaged by Z, a worldwide leading supplier of Y. We are expanding the core engineering team and we are looking for really passionate engineers who want to create their own story and help millions of people.

Can we find a time for a call to chat for a few minutes about this?

Thanks

Hi Andy,

I just received an email fr first tutorial speaker, presenting r u, saying he's ill and won't be

I know you have a ourself to two presentations you could increase you , maybe just offer time to try out what you've taught? Otherwise I have to do some kind of modern dance interpretation of Python in data :-)
-Leah

Hi Andreas,

I am very interested in your Machine Learning background. I work for X Recruiting who have been engaged by Z, a worldwide leading supplier of Y. We are expanding the core engineering team and we are looking for really passionate engineers who own story and help millions of

Can we find a ti for a few minutes about t

Thanks

Doing Machine Learning With Scikit-Learn

Representing Data

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

Representing Data

$$X = \begin{array}{r} \text{one sample} \\ \\ \\ \\ \\ \\ \\ \end{array} \left(\begin{array}{ccccc} \multicolumn{5}{c}{\boxed{1.1 \quad 2.2 \quad 3.4 \quad 5.6 \quad 1.0}} \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{array}\right)$$

Representing Data

$$X = \begin{array}{r|ccccc} \text{one sample} & 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ & 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ & 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ & 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ & 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ & 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ & 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \\ & \text{one feature} & & & & \end{array}$$

Representing Data

one sample

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

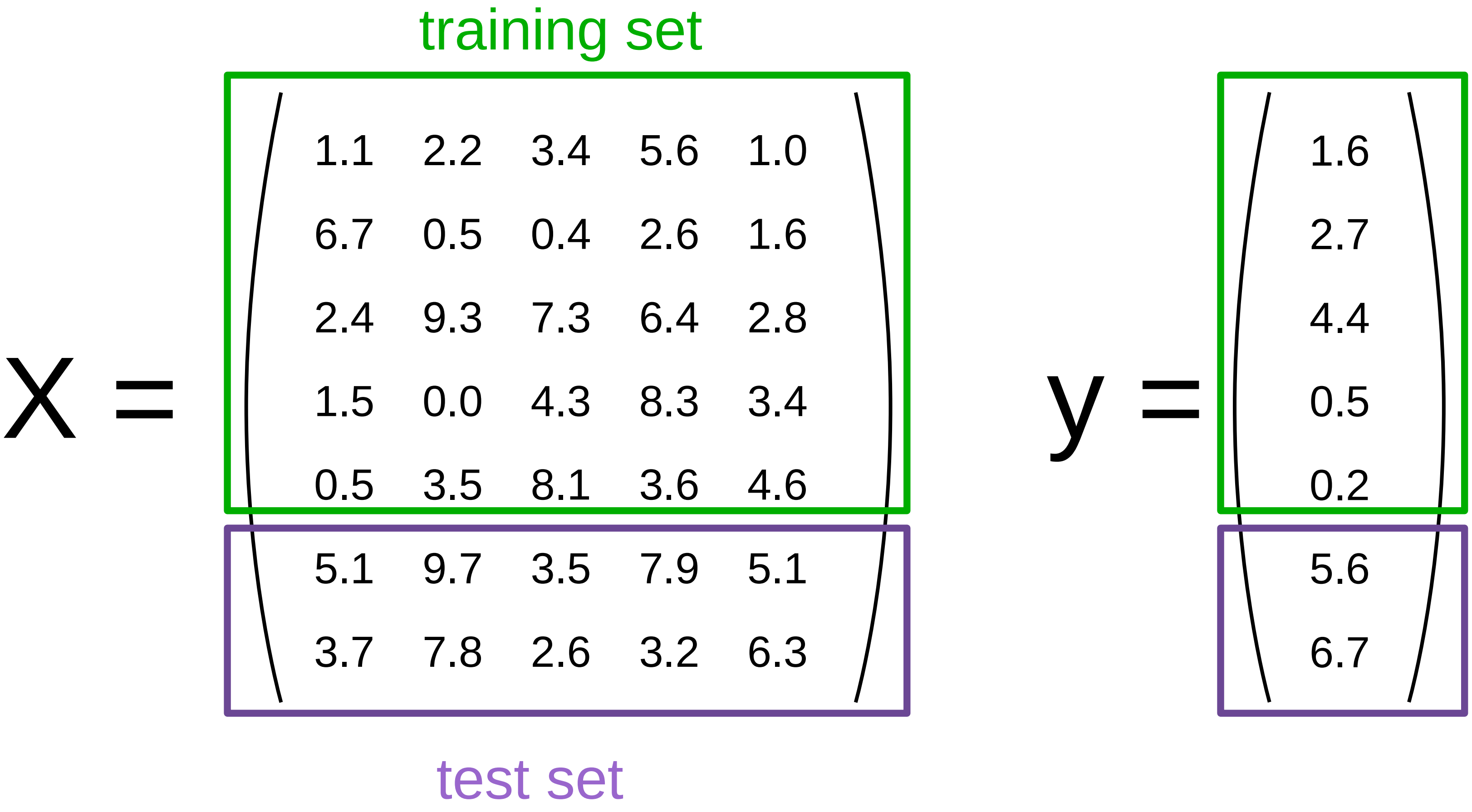
one feature

outputs / labels

Training and Testing Data

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

Training and Testing Data

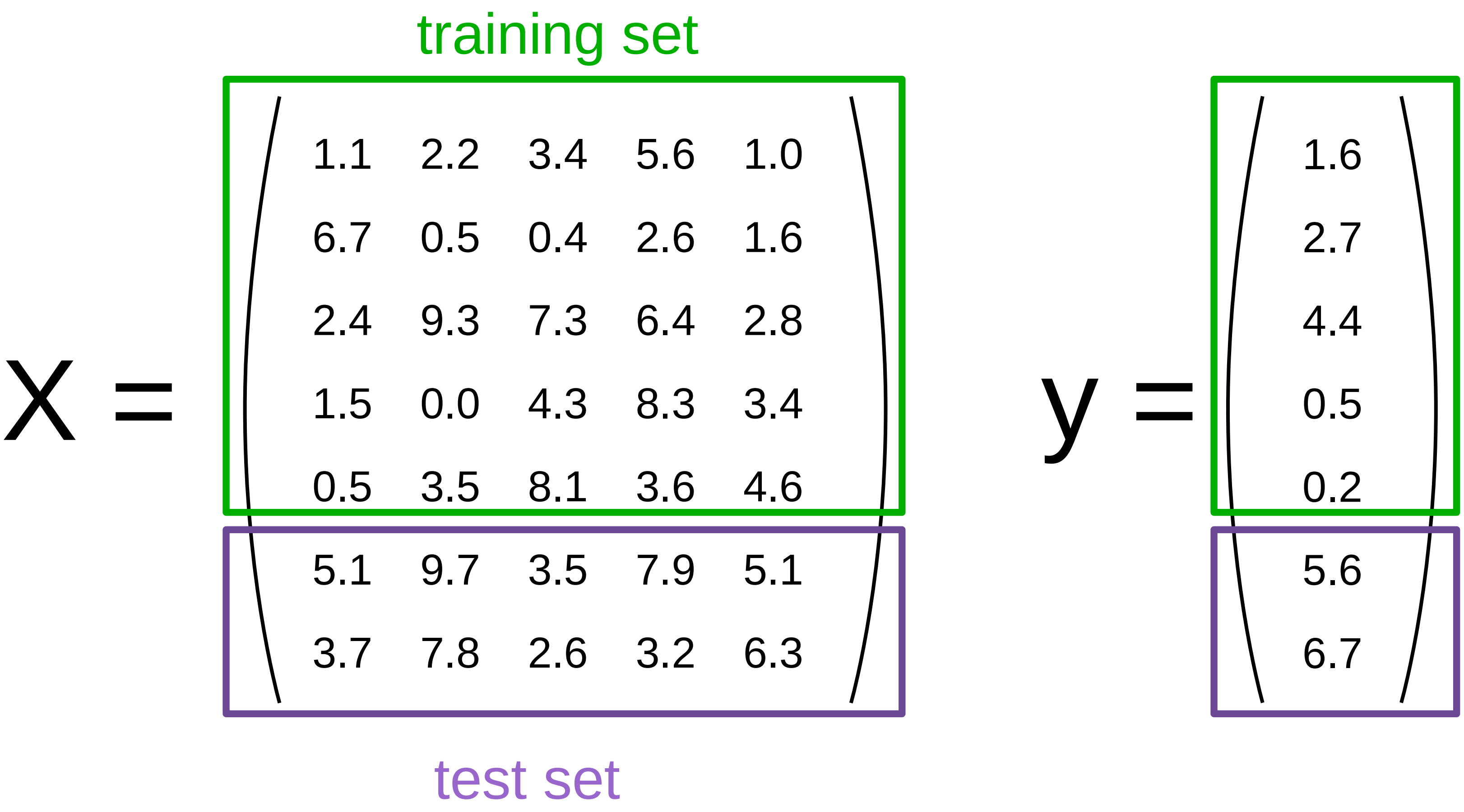


training set

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

test set

Training and Testing Data



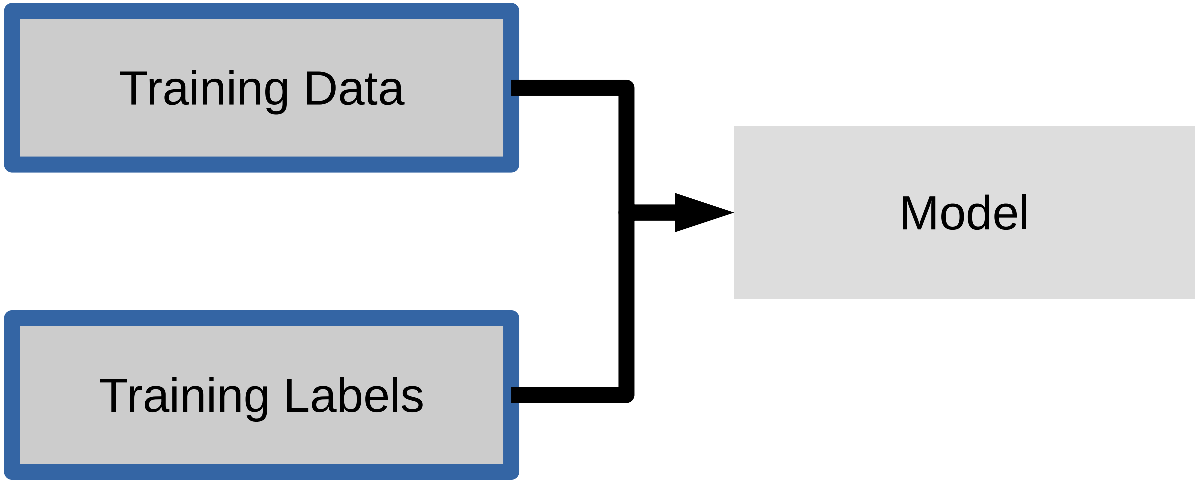
training set

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

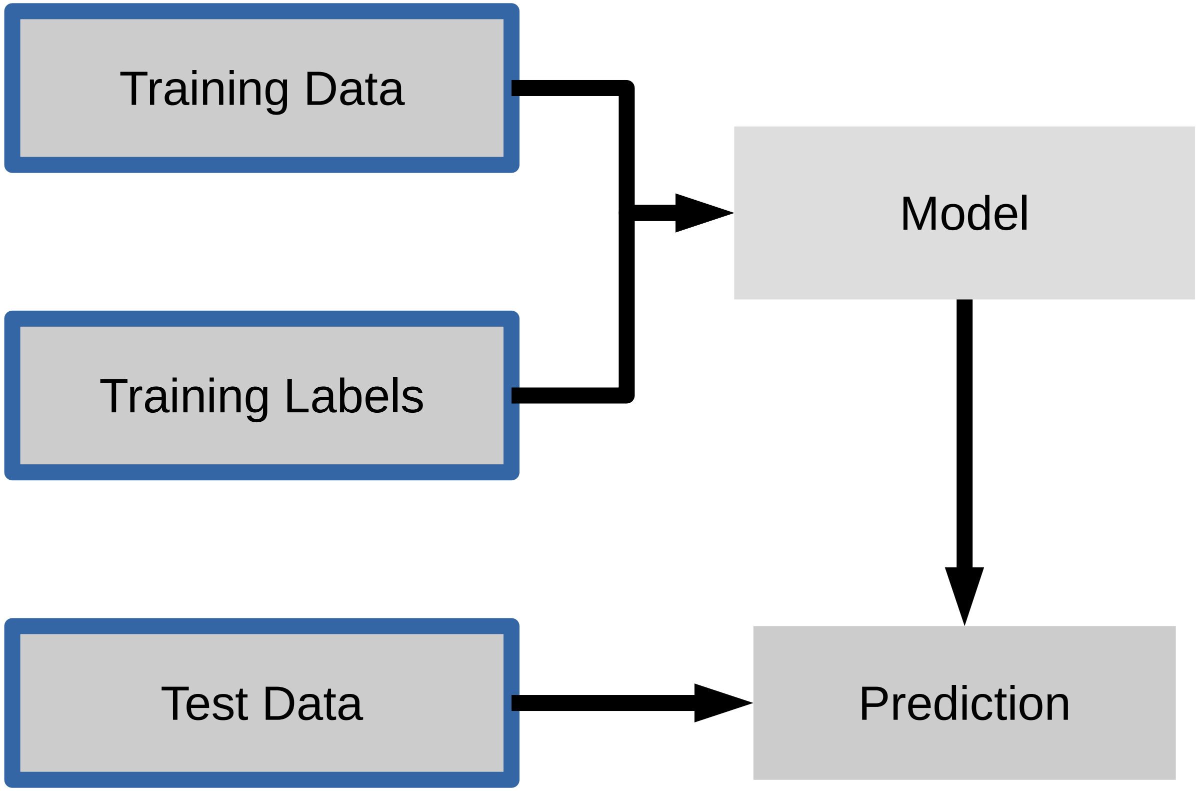
test set

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

Supervised Machine Learning



Supervised Machine Learning


Training Data

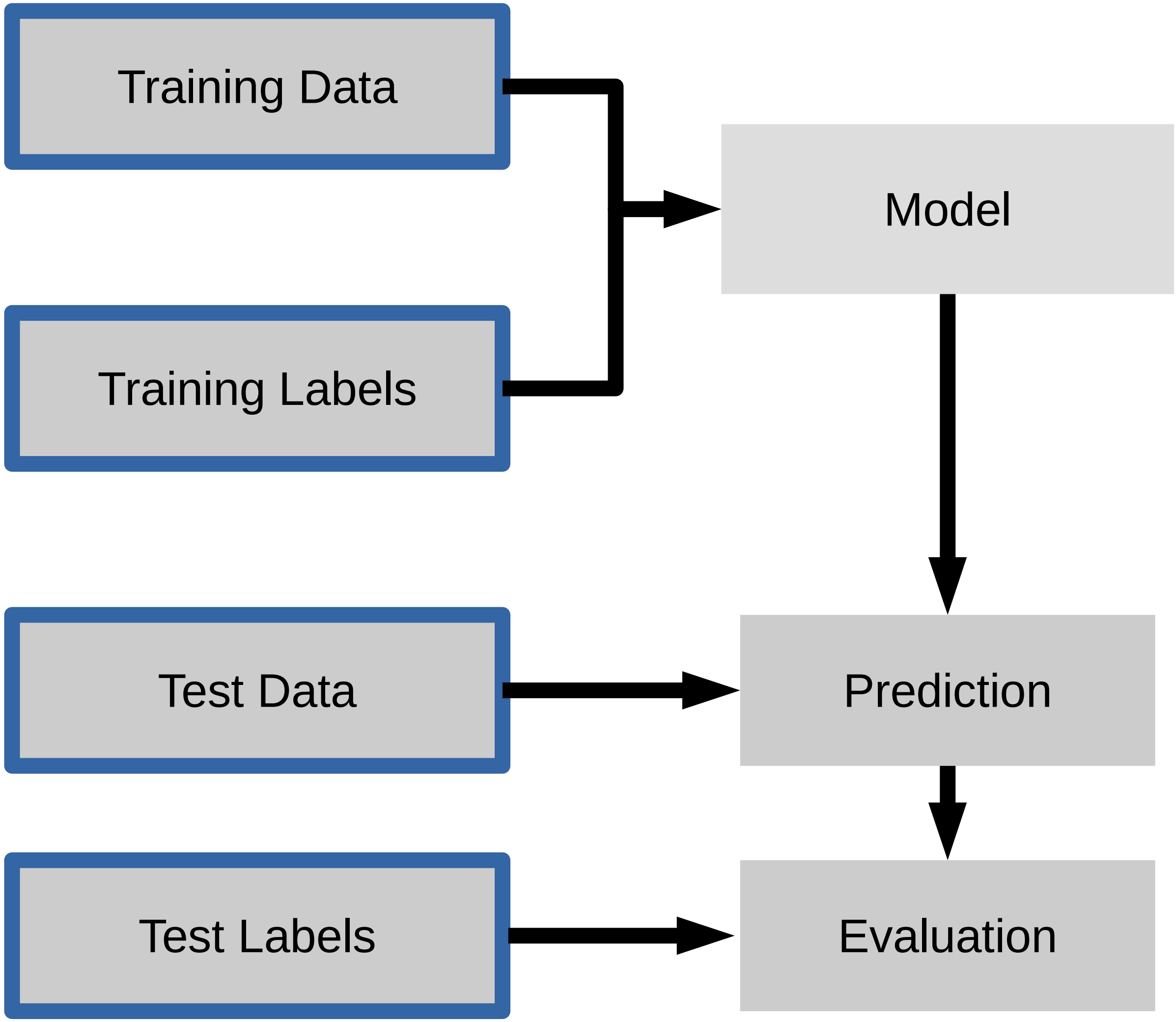
Training Labels

Test Data

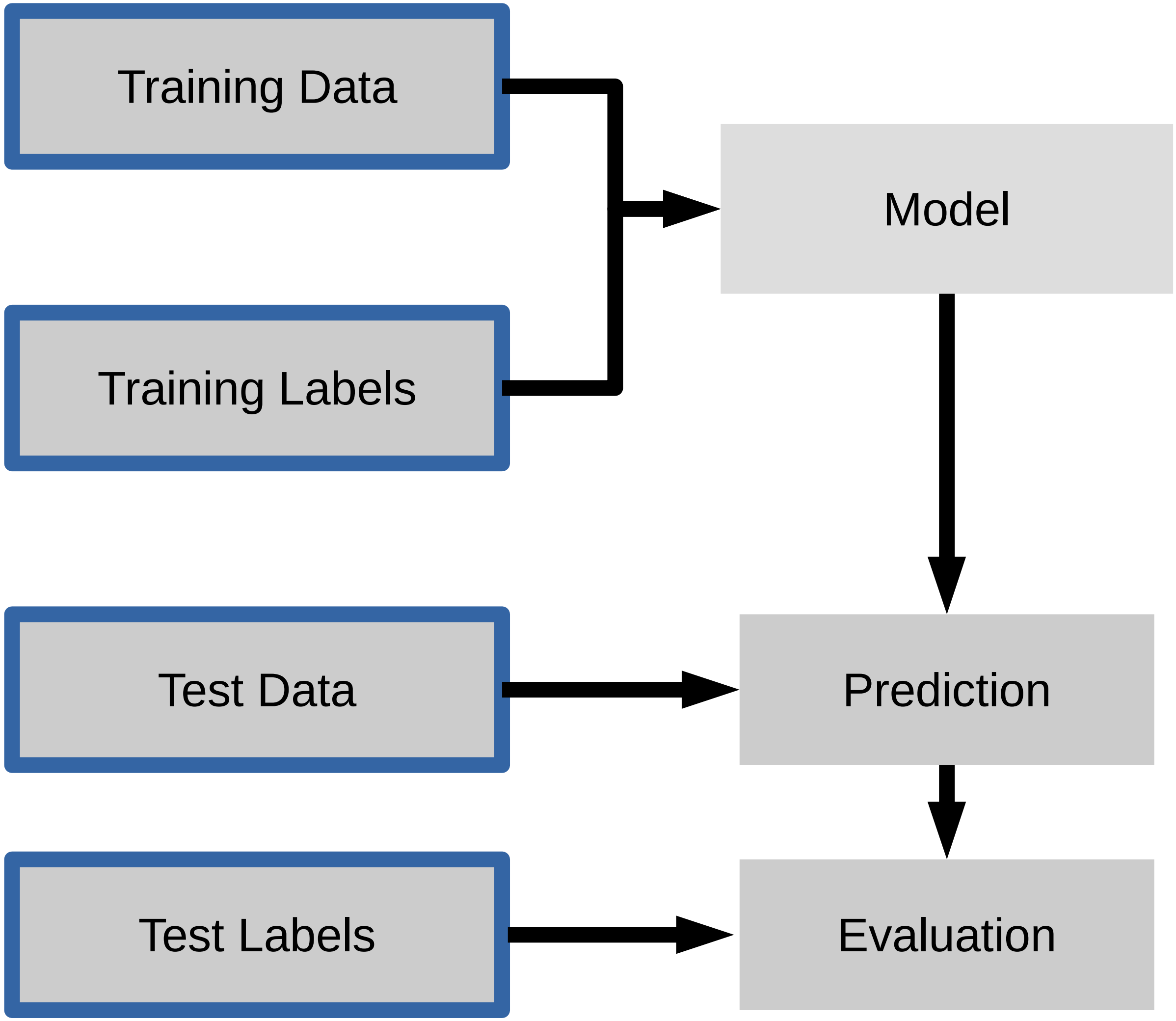
Model

Prediction

Supervised Machine Learning



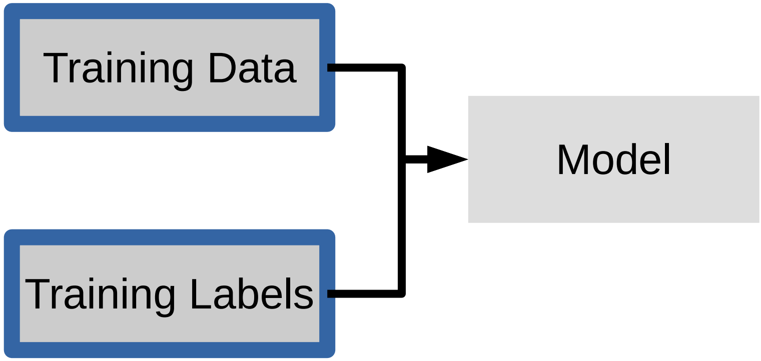
Supervised Machine Learning



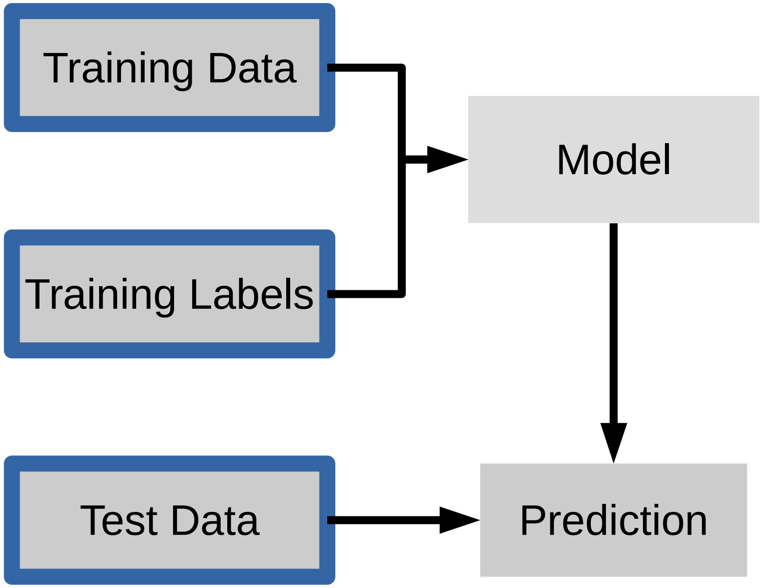
Training

Generalization

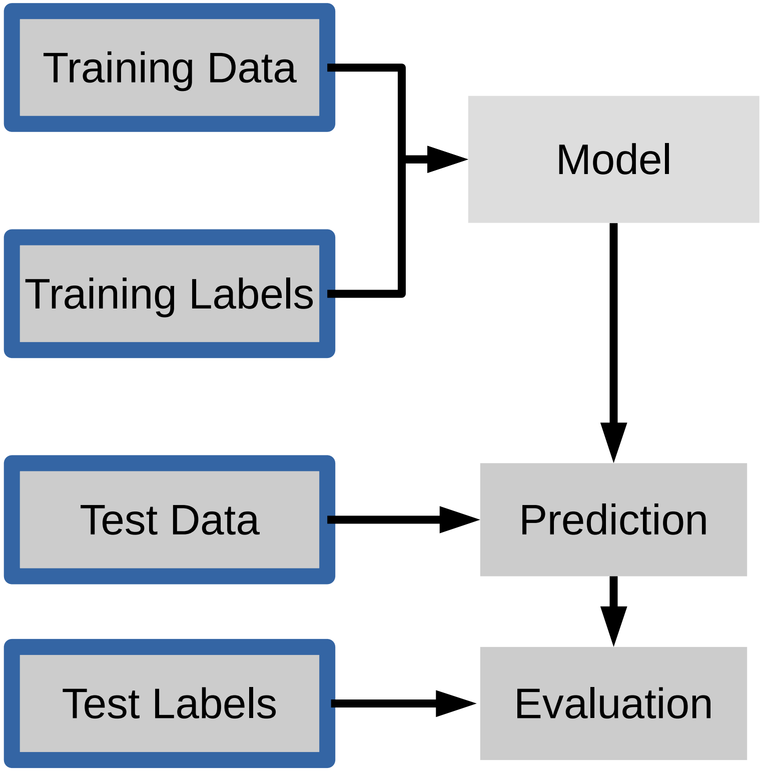
```
clf = RandomForestClassifier()

clf.fit(X_train, y_train)
```



```
clf = RandomForestClassifier()

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

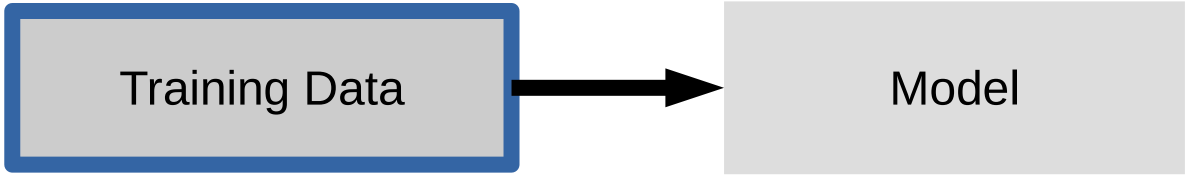


```
clf = RandomForestClassifier()

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

clf.score(X_test, y_test)
```

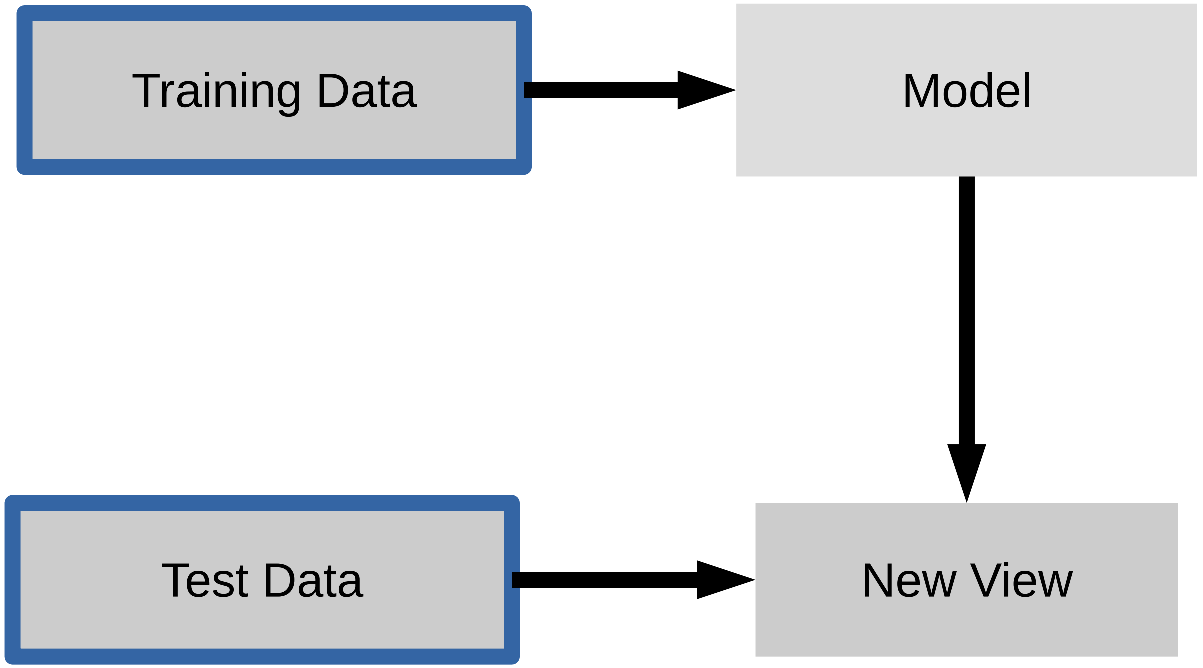


IPython Notebook:
Part 1 - Introduction to Scikit-learn

Unsupervised Machine Learning



Unsupervised Machine Learning

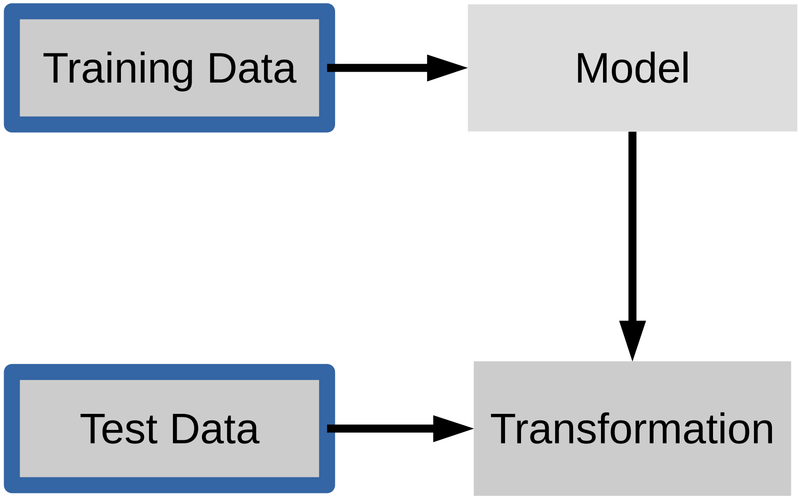


Unsupervised Transformations

```
pca = PCA()
```

```
pca.fit(X_train)
```

```
X_new = pca.transform(X_test)
```

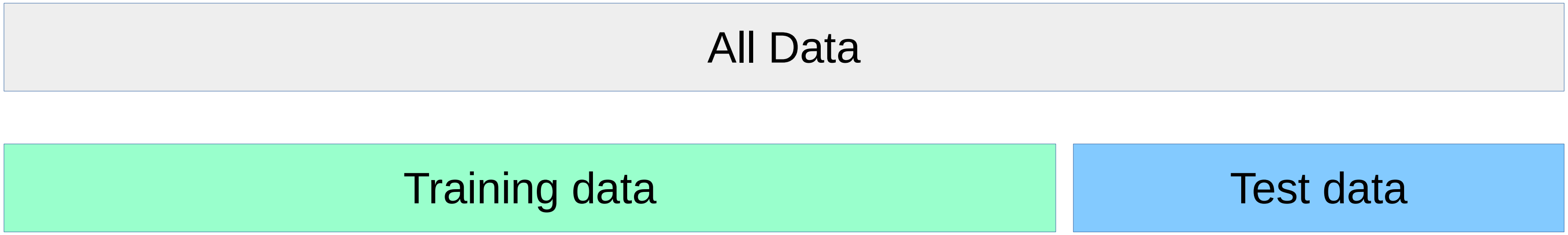


IPython Notebook:
Part 2 – Unsupervised Transformers

Basic API

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

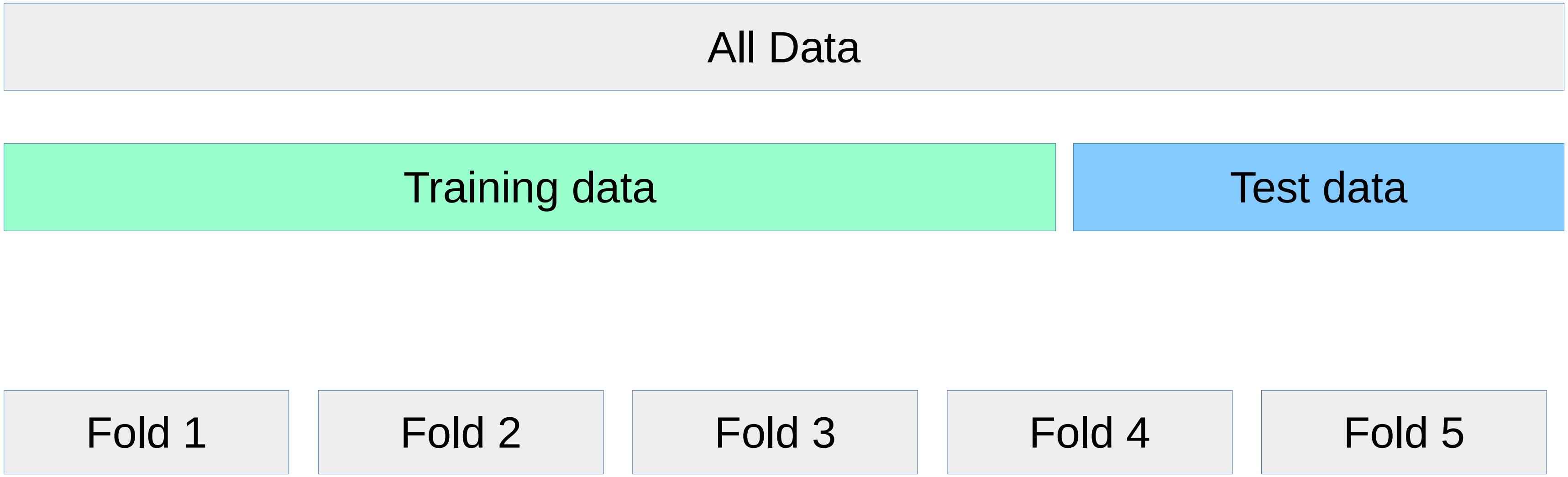
`estimator.predict`	`estimator.transform`
Classification	Preprocessing
Regression	Dimensionality reduction
Clustering	Feature selection
	Feature extraction



All Data

Training data

Test data



All Data

Training data

Test data

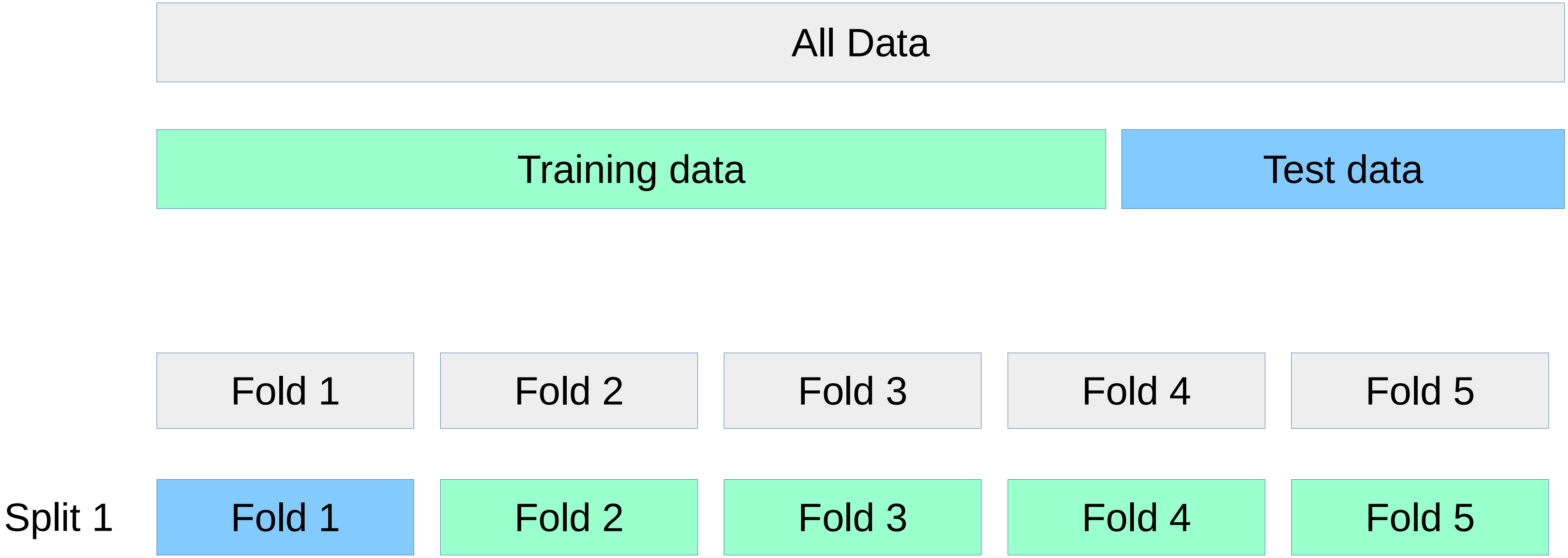
Fold 1

Fold 2

Fold 3

Fold 4

Fold 5



All Data

Training data

Test data

Fold 1

Fold 2

Fold 3

Fold 4

Fold 5

Split 1

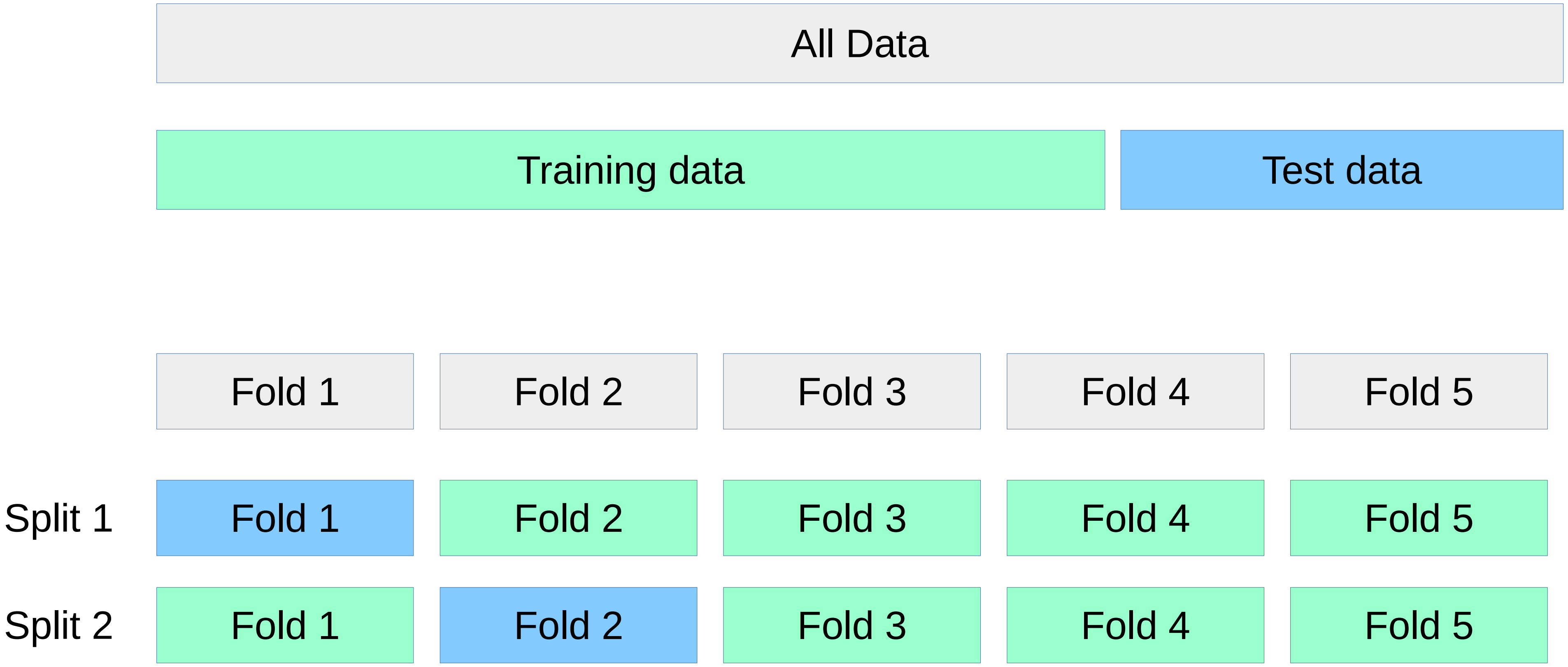
Fold 1

Fold 2

Fold 3

Fold 4

Fold 5



All Data

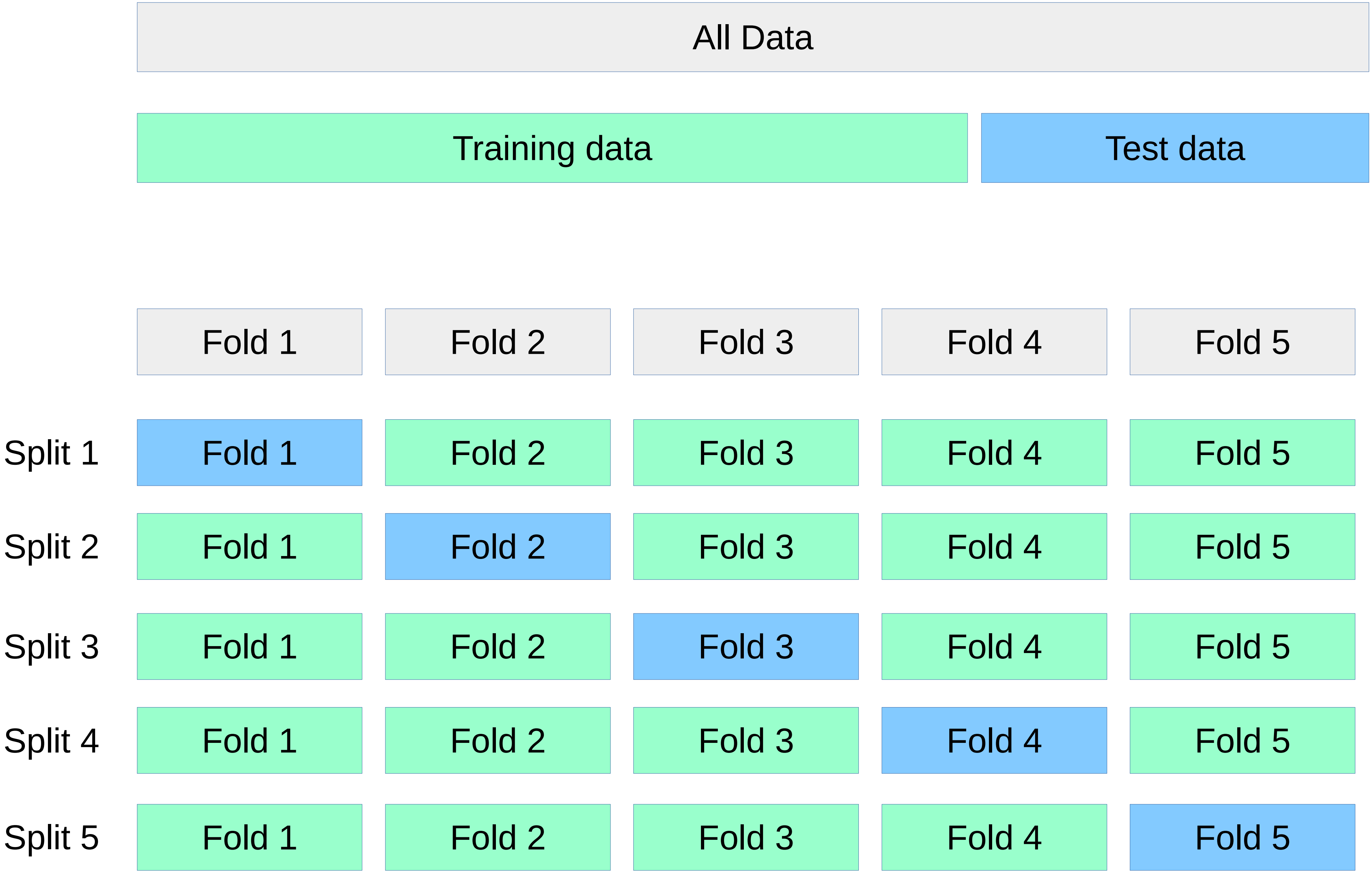
Training data

Test data

Fold 1 Fold 2 Fold 3 Fold 4 Fold 5

Split 1 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5

Split 2 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5

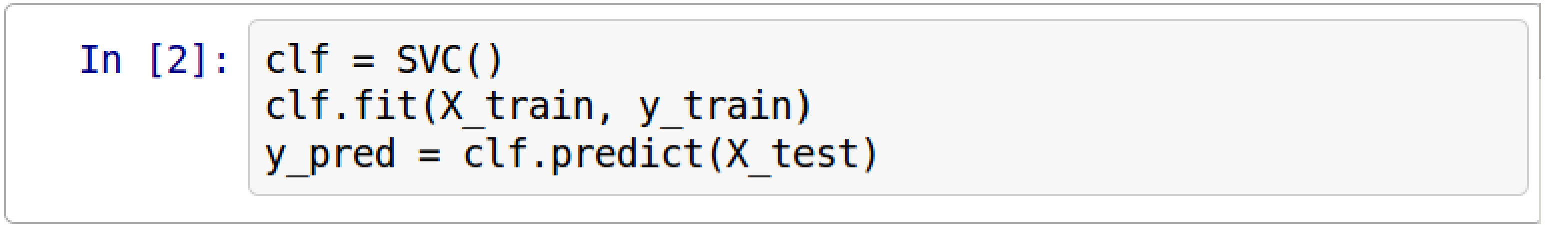


All Data

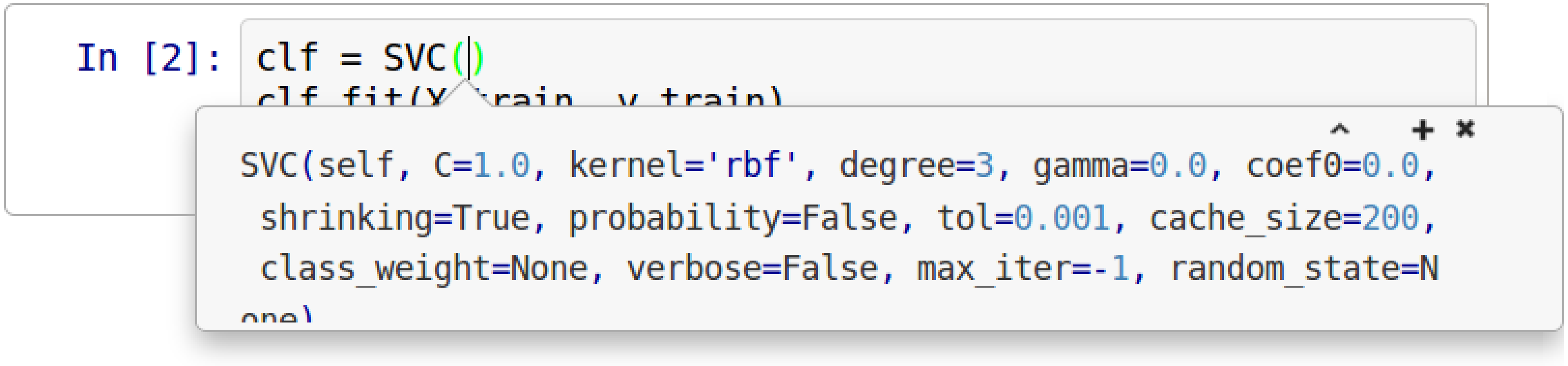
Training data | Test data

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

IPython Notebook:
Part 3 - Cross-validation

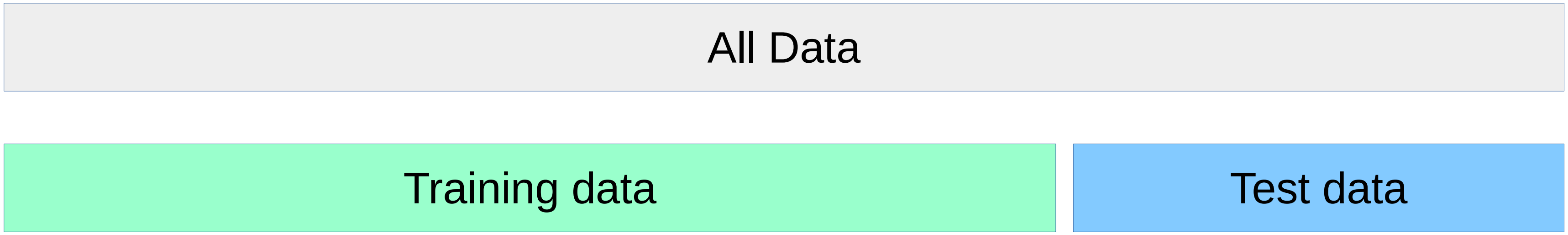


```
clf = SVC()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```



```
In [2]: clf = SVC()
        clf.fit(X_train, y_train)
```

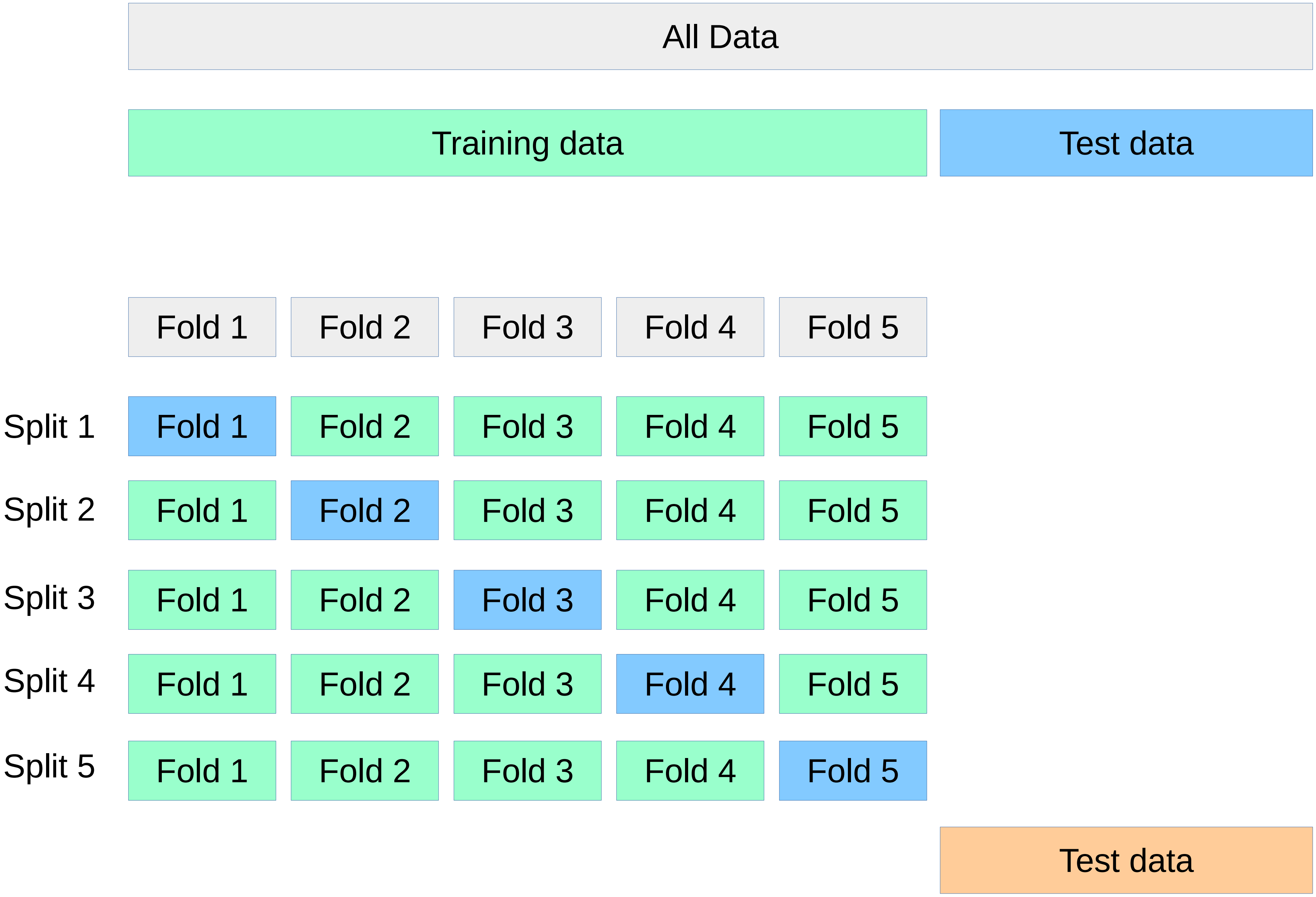
```
SVC(self, C=1.0, kernel='rbf', degree=3, gamma=0.0, coef0=0.0,
 shrinking=True, probability=False, tol=0.001, cache_size=200,
 class_weight=None, verbose=False, max_iter=-1, random_state=N
one)
```



All Data

Training data

Test data



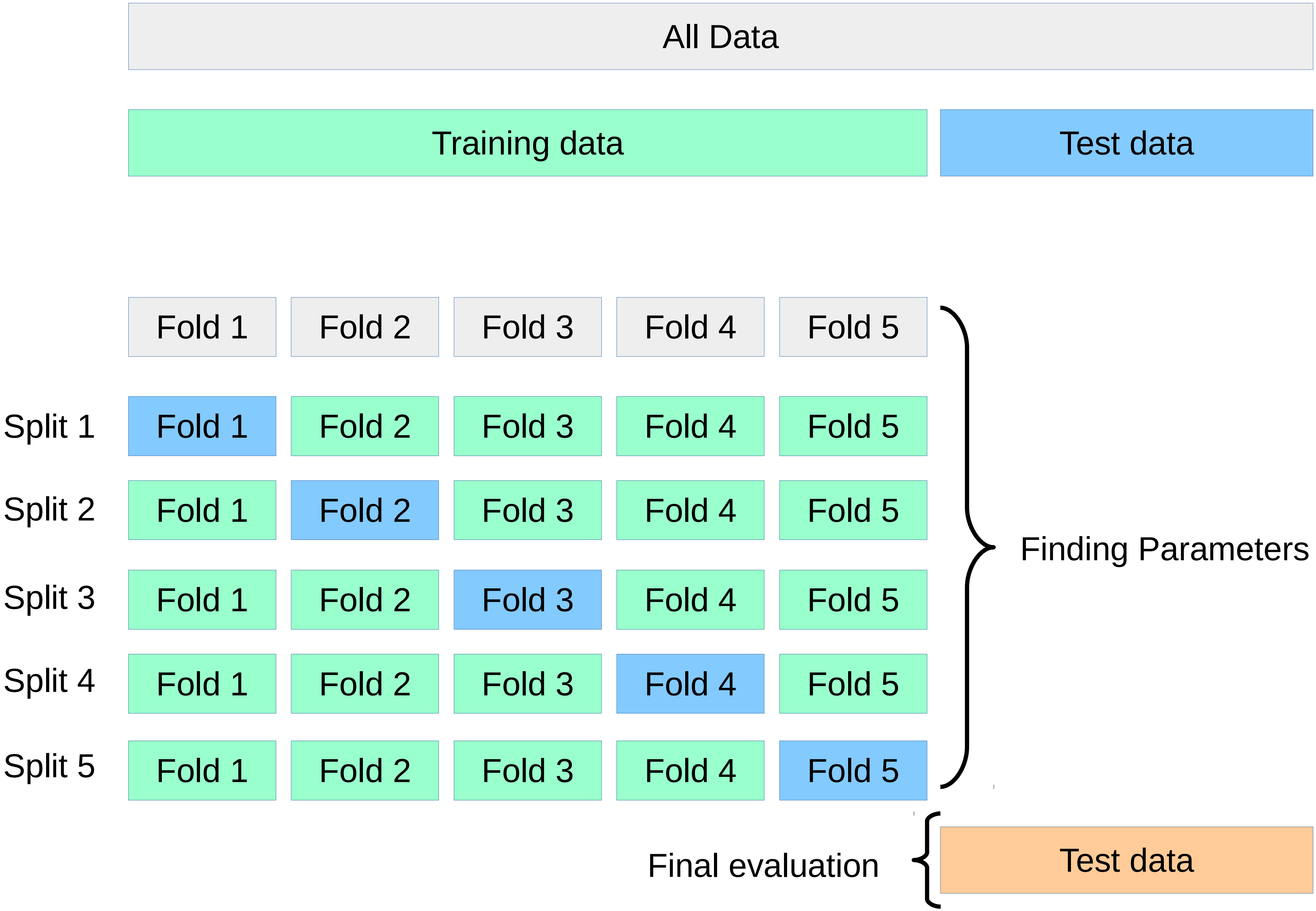
All Data

Training data

Test data

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

Test data



All Data

Training data | Test data

Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

Split 1 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

Split 2 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

Split 3 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

Split 4 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

Split 5 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

Finding Parameters

Final evaluation

Test data

SVC(C=0.001,
gamma=0.001)

SVC(C=0.001, gamma=0.001)

SVC(C=0.01, gamma=0.001)

SVC(C=0.1, gamma=0.001)

SVC(C=1, gamma=0.001)

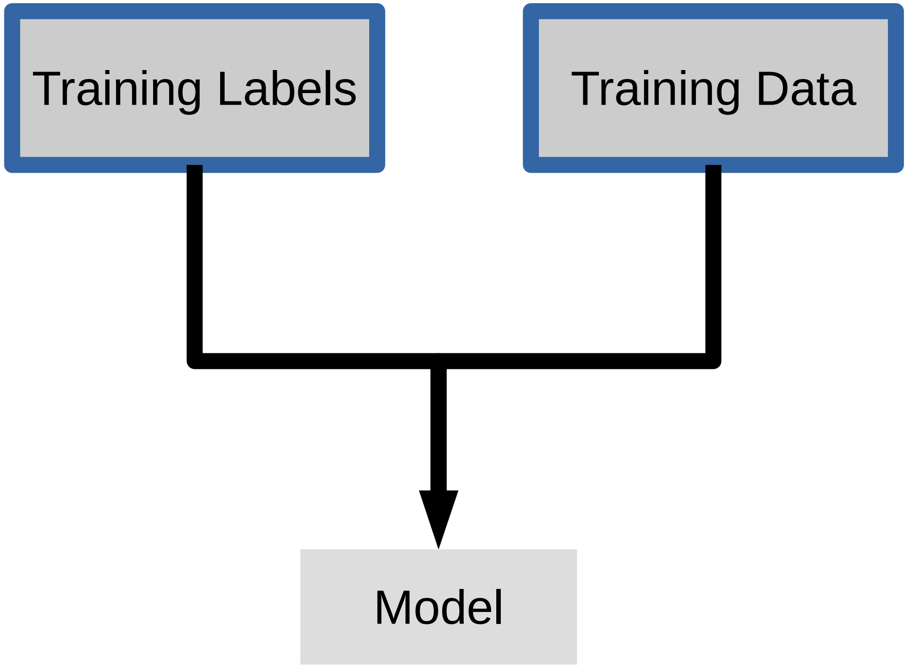
SVC(C=10, gamma=0.001)

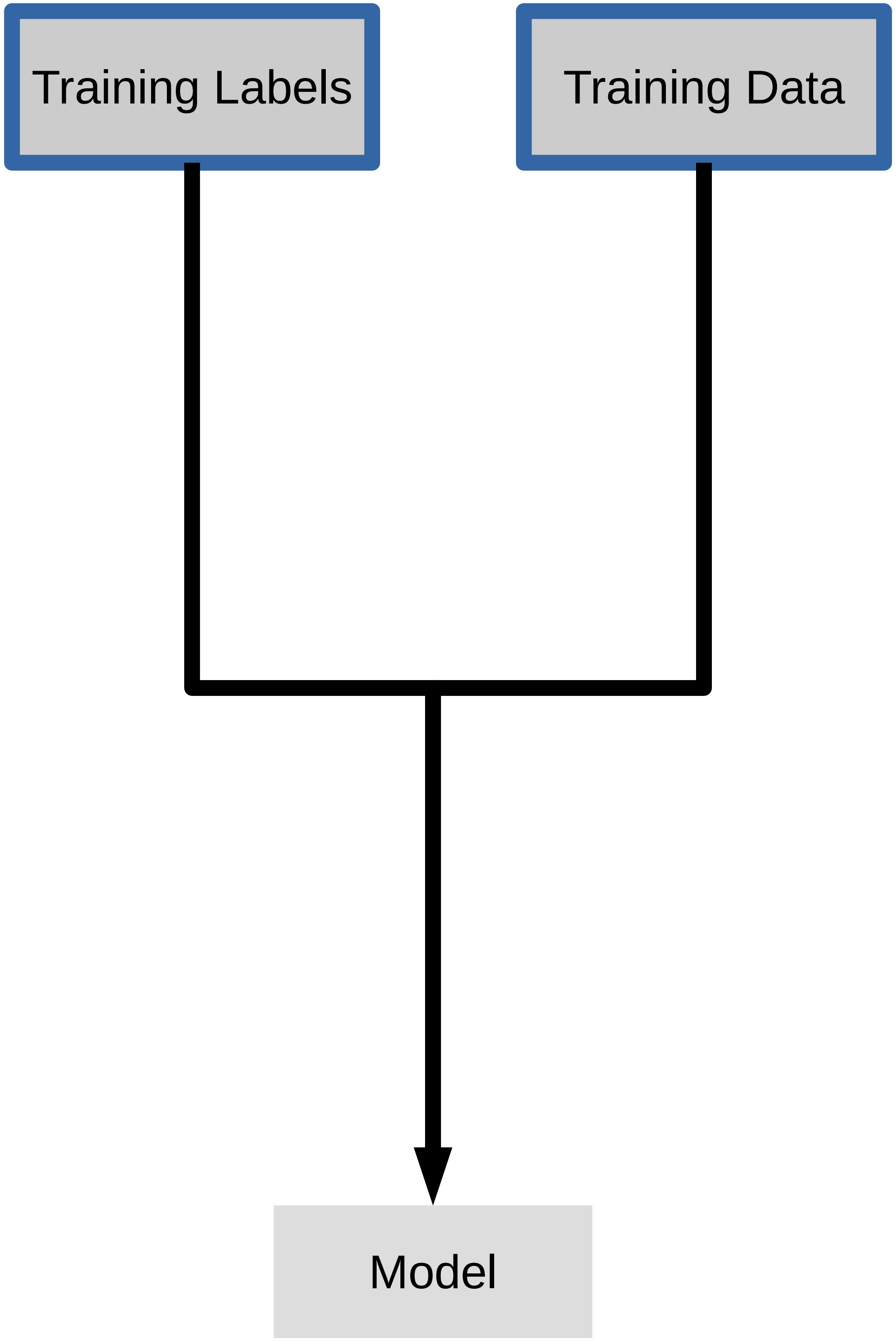
SVC(C=0.001, gamma=0.001)	SVC(C=0.01, gamma=0.001)	SVC(C=0.1, gamma=0.001)	SVC(C=1, gamma=0.001)	SVC(C=10, gamma=0.001)
SVC(C=0.001, gamma=0.01)	SVC(C=0.01, gamma=0.01)	SVC(C=0.1, gamma=0.01)	SVC(C=1, gamma=0.01)	SVC(C=10, gamma=0.01)

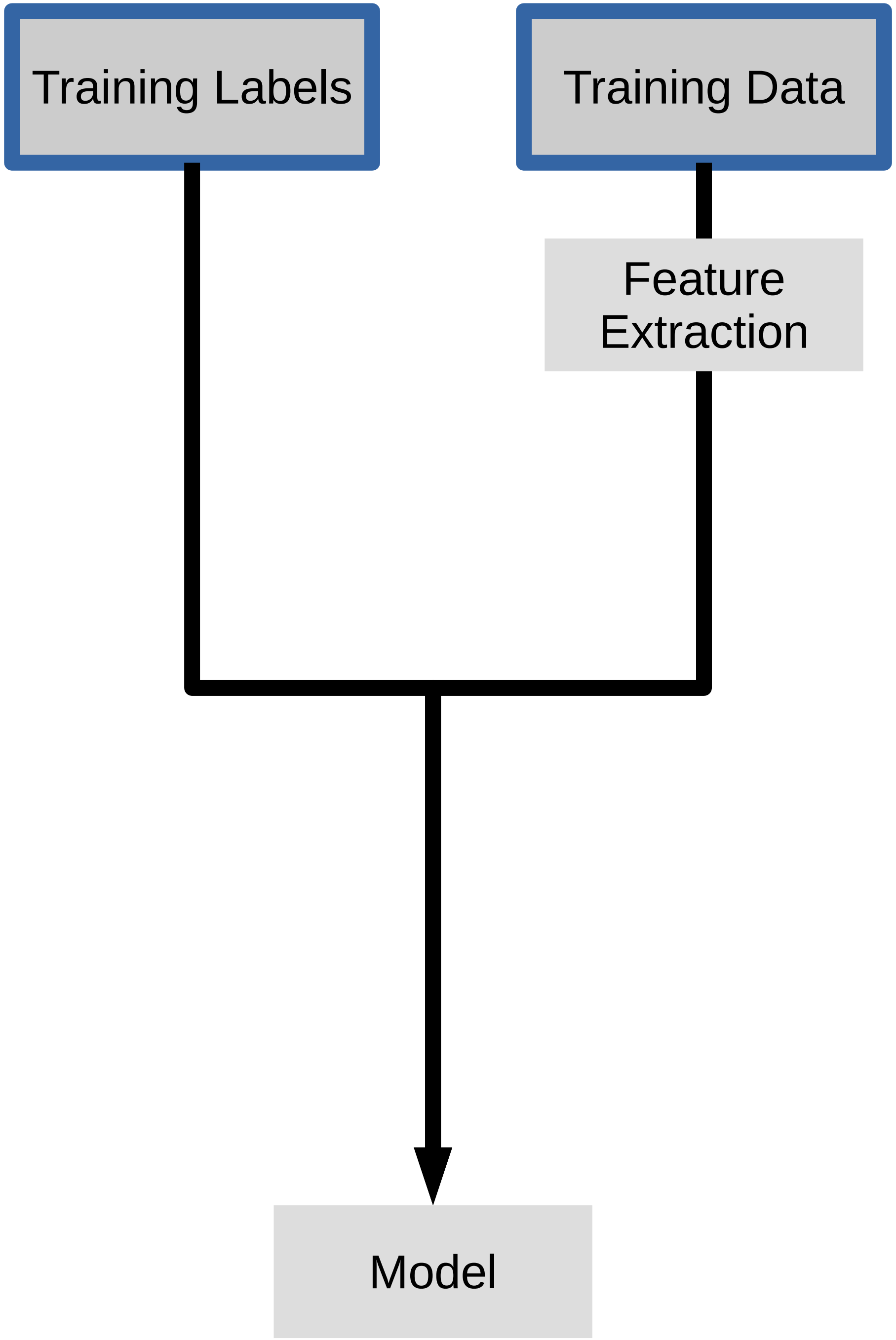
SVC(C=0.001, gamma=0.001)	SVC(C=0.01, gamma=0.001)	SVC(C=0.1, gamma=0.001)	SVC(C=1, gamma=0.001)	SVC(C=10, gamma=0.001)
SVC(C=0.001, gamma=0.01)	SVC(C=0.01, gamma=0.01)	SVC(C=0.1, gamma=0.01)	SVC(C=1, gamma=0.01)	SVC(C=10, gamma=0.01)
SVC(C=0.001, gamma=0.1)	SVC(C=0.01, gamma=0.1)	SVC(C=0.1, gamma=0.1)	SVC(C=1, gamma=0.1)	SVC(C=10, gamma=0.1)

SVC(C=0.001, gamma=0.001)	SVC(C=0.01, gamma=0.001)	SVC(C=0.1, gamma=0.001)	SVC(C=1, gamma=0.001)	SVC(C=10, gamma=0.001)
SVC(C=0.001, gamma=0.01)	SVC(C=0.01, gamma=0.01)	SVC(C=0.1, gamma=0.01)	SVC(C=1, gamma=0.01)	SVC(C=10, gamma=0.01)
SVC(C=0.001, gamma=0.1)	SVC(C=0.01, gamma=0.1)	SVC(C=0.1, gamma=0.1)	SVC(C=1, gamma=0.1)	SVC(C=10, gamma=0.1)
SVC(C=0.001, gamma=1)	SVC(C=0.01, gamma=1)	SVC(C=0.1, gamma=1)	SVC(C=1, gamma=1)	SVC(C=10, gamma=1)
SVC(C=0.001, gamma=10)	SVC(C=0.01, gamma=10)	SVC(C=0.1, gamma=10)	SVC(C=1, gamma=10)	SVC(C=10, gamma=10)

IPython Notebook:
Part 4 – Grid Searches


Training Labels
Training Data
Model


Training Labels
Training Data
Model

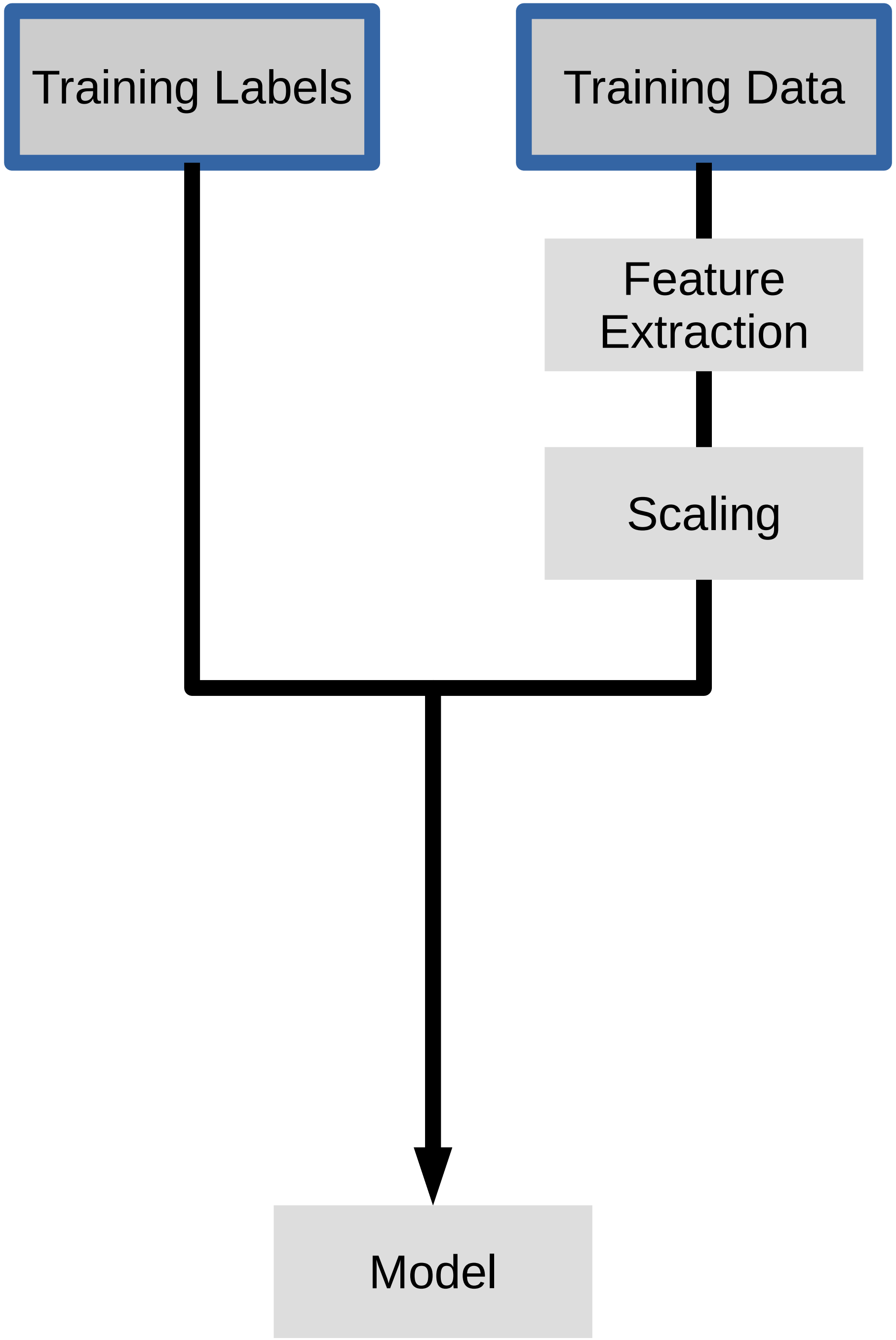


Training Labels

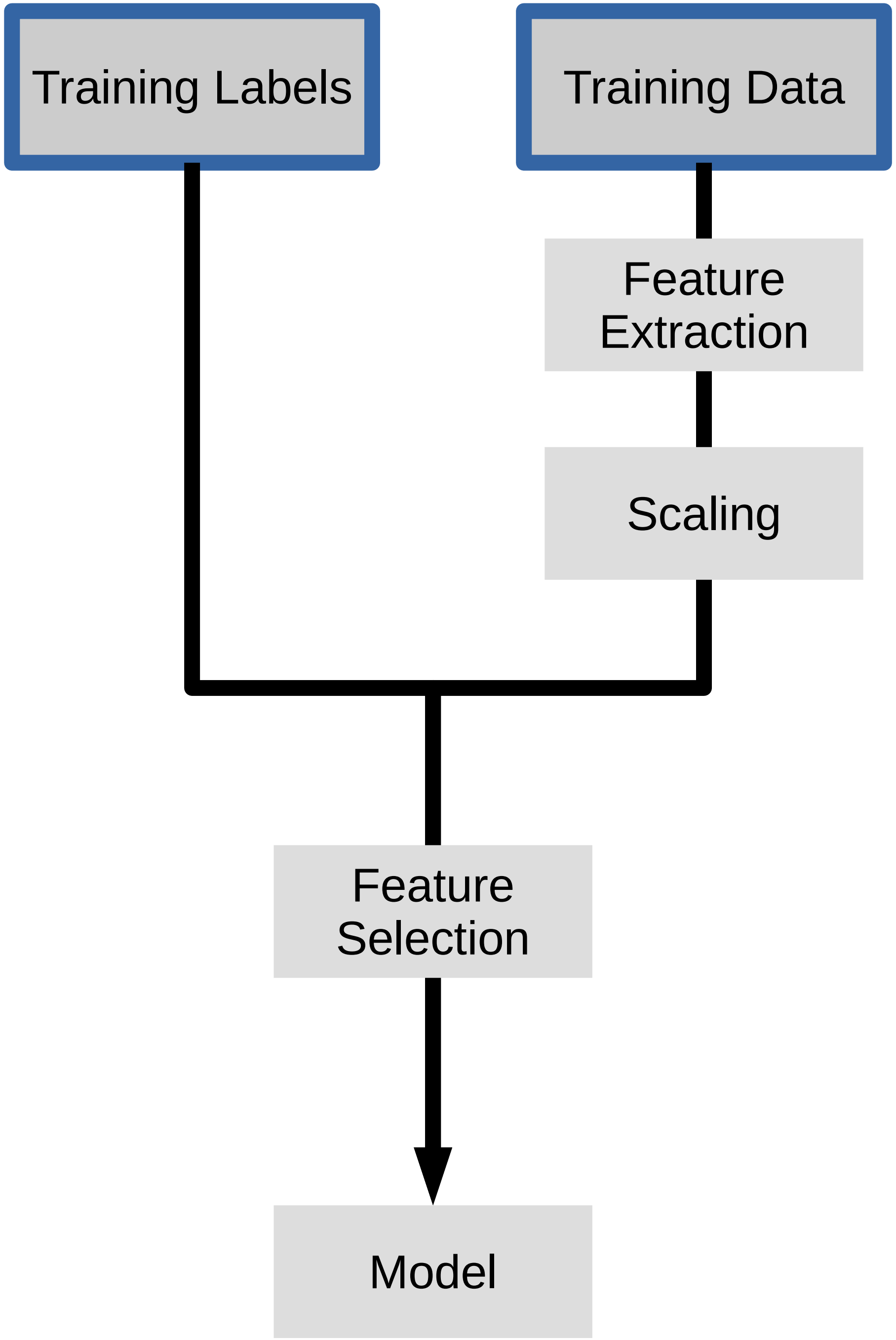
Training Data

Feature Extraction

Model



Training Labels
Training Data
Feature Extraction
Scaling
Model


Training Labels

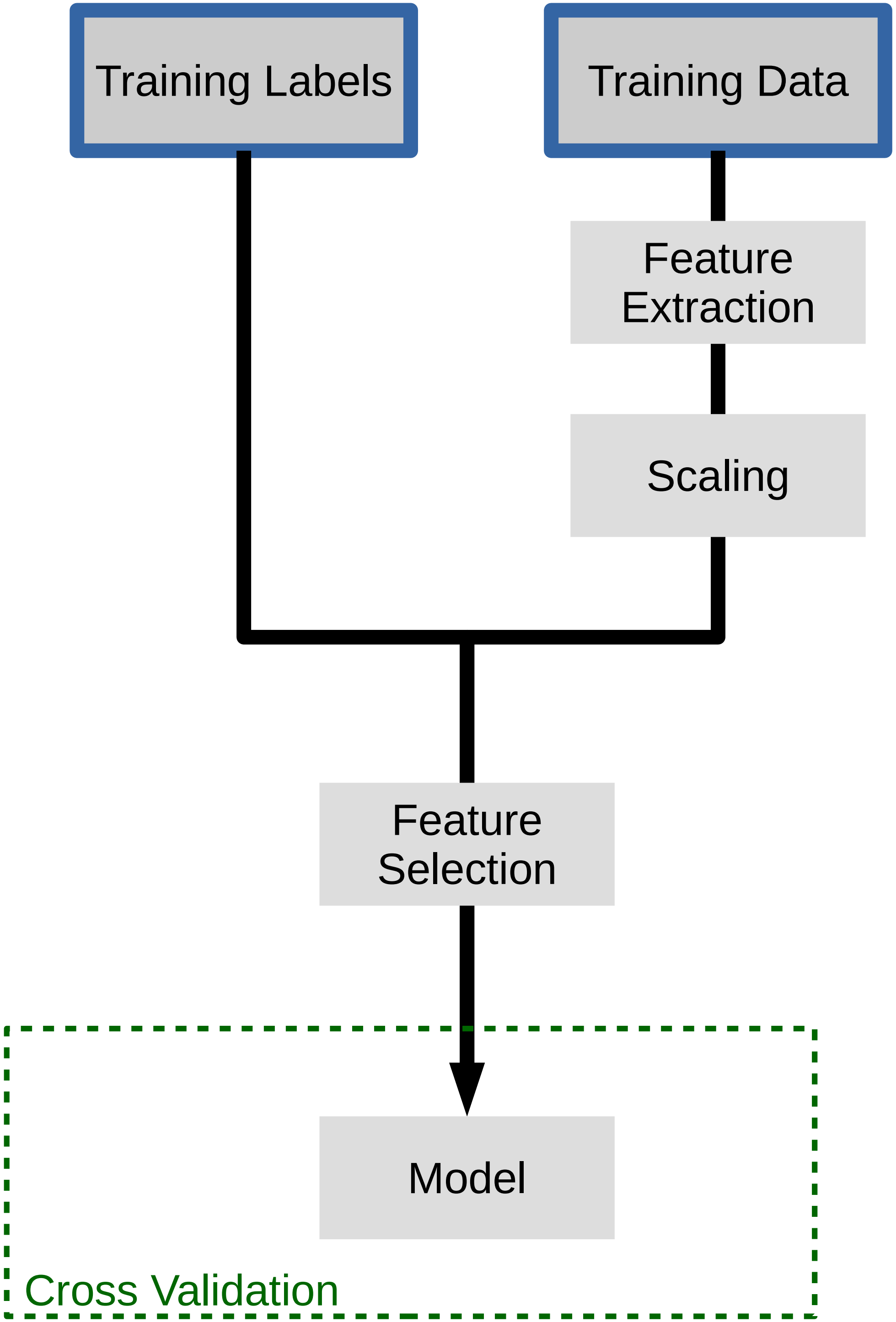
Training Data

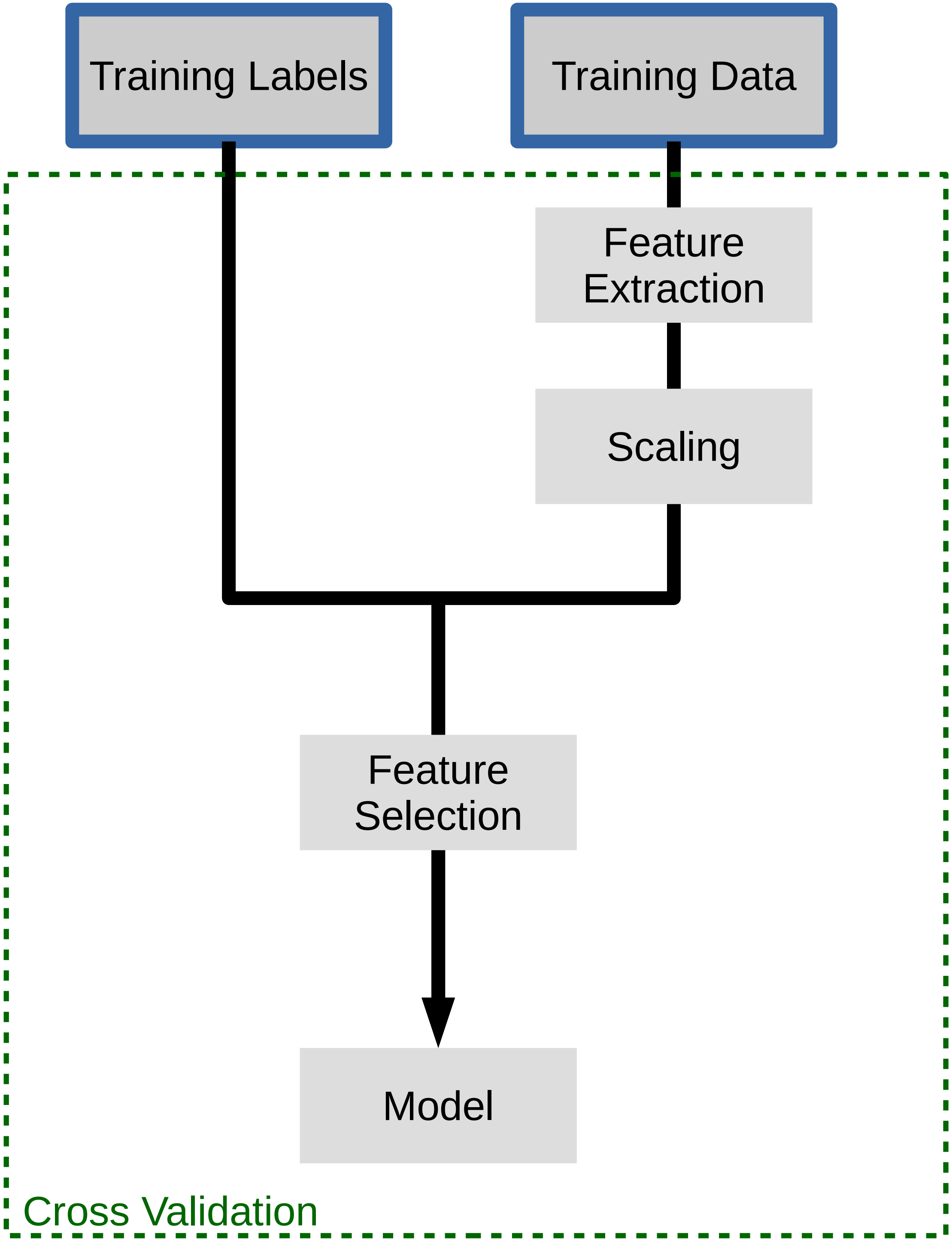
Feature Extraction

Scaling

Feature Selection

Model

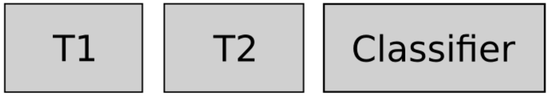

Training Labels
Training Data
Feature Extraction
Scaling
Feature Selection
Model
Cross Validation


Training Labels
Training Data
Feature Extraction
Scaling
Feature Selection
Model
Cross Validation

Pipelines

Pipelines

pipe = make_pipeline(T1(), T2(), Classifier())

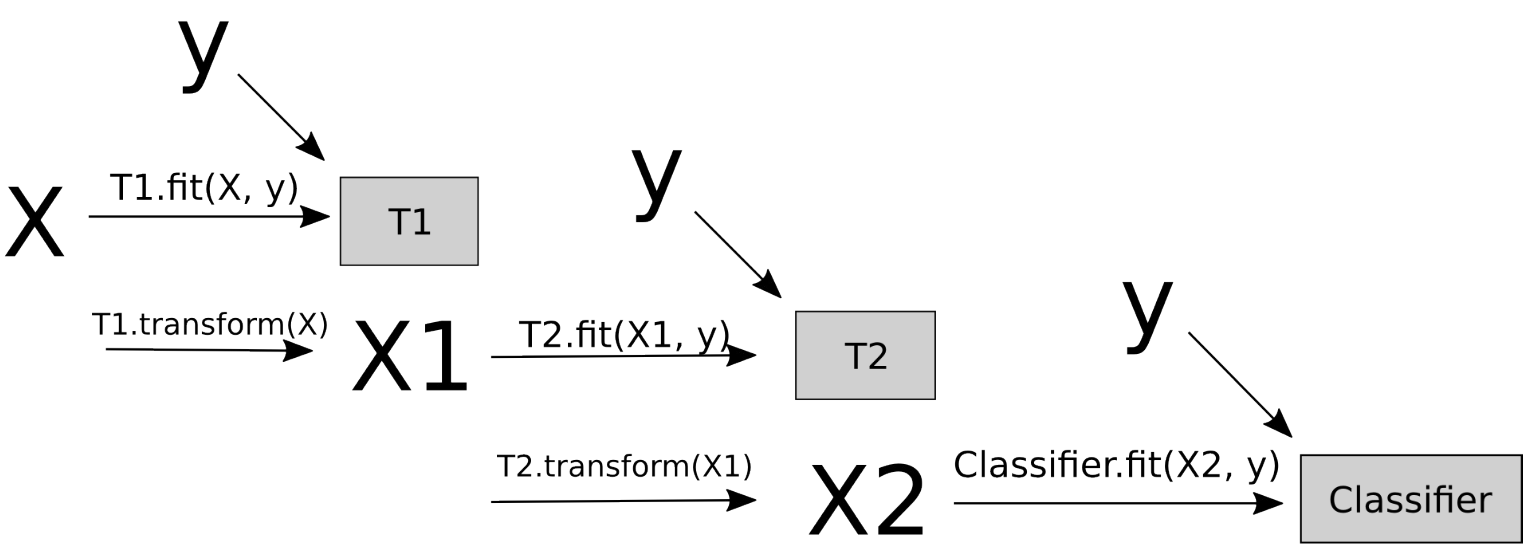


T1

T2

Classifier

pipe.fit(X, y)



y

X

T1.fit(X, y)

T1

y

T1.transform(X)

X1

T2.fit(X1, y)

T2

y

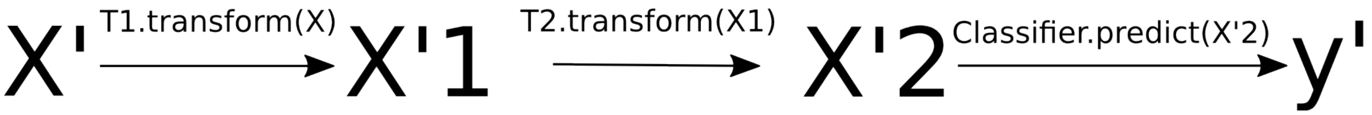
T2.transform(X1)

X2

Classifier.fit(X2, y)

Classifier

pipe.predict(X)



X'

T1.transform(X)

X'1

T2.transform(X1)

X'2

Classifier.predict(X'2)

y'

IPython Notebook:
Part 5 - Preprocessing and Pipelines

Do cross-validation over all steps jointly.
Keep a separate test set until the very end.

Sample application: Sentiment Analysis

IMDB Movie Reviews Data

Review:
One of the worst movies I've ever rented. Sorry it had one of my favorite actors on it (Travolta) in a nonsense role. In fact, anything made sense in this movie.

Who can say there was true love between Eddy and Maureen? Don't you remember the beginning of the movie ?

Is she so lovely? Ask her daughters. I don't think so.

Label: negative

Training data: 12500 positive, 12500 negative

Bag Of Word Representations

CountVectorizer / TfidfVectorizer

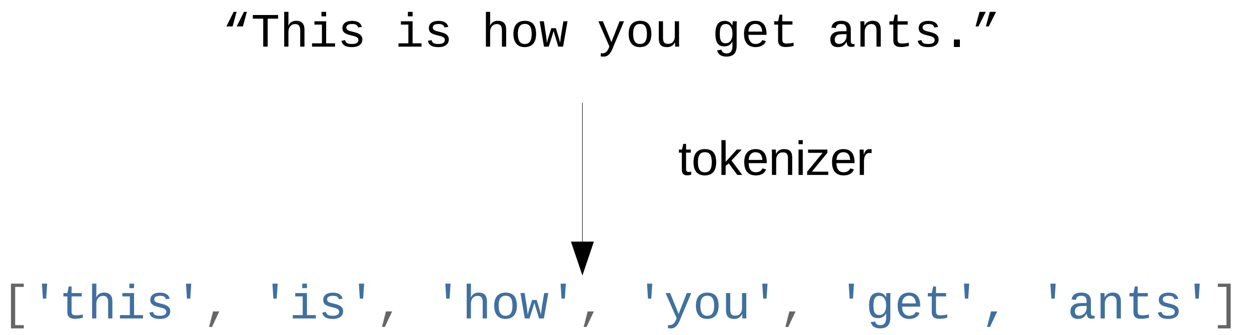
Bag Of Word Representations

CountVectorizer / TfidfVectorizer

“This is how you get ants.”

Bag Of Word Representations

`CountVectorizer / TfidfVectorizer`



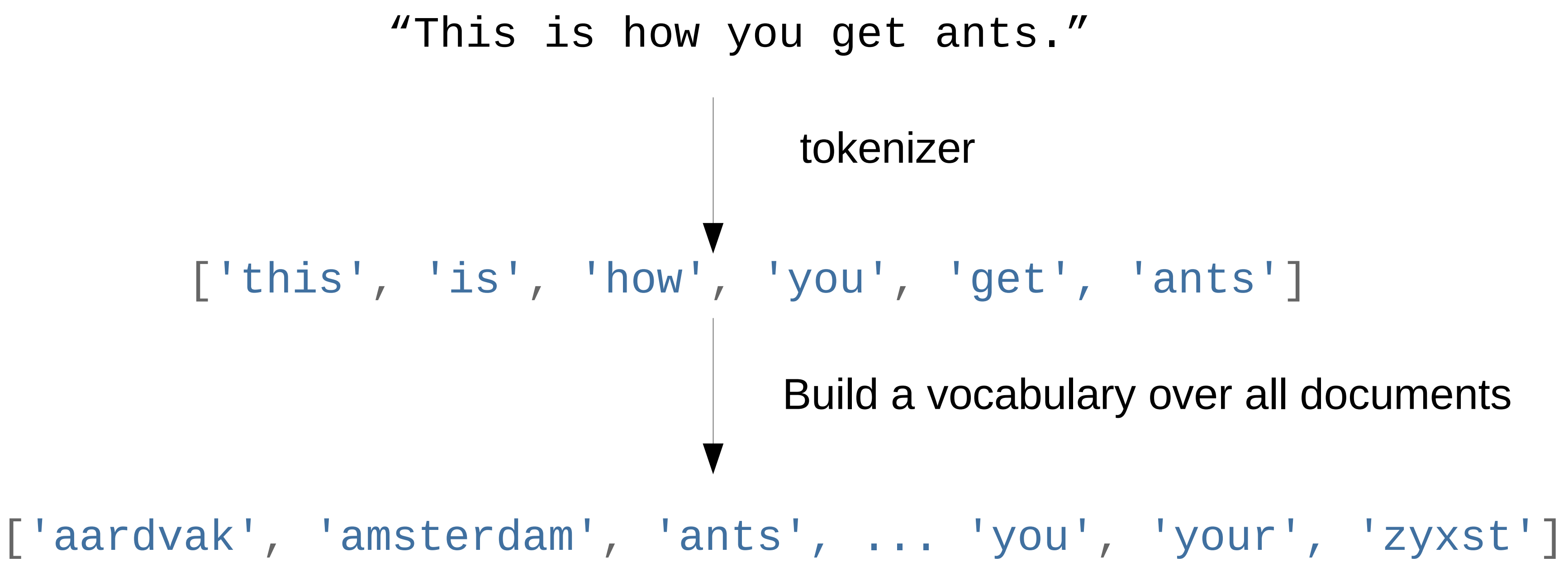
“This is how you get ants.”

tokenizer

```
['this', 'is', 'how', 'you', 'get', 'ants']
```

Bag Of Word Representations

`CountVectorizer / TfidfVectorizer`



“This is how you get ants.”

tokenizer

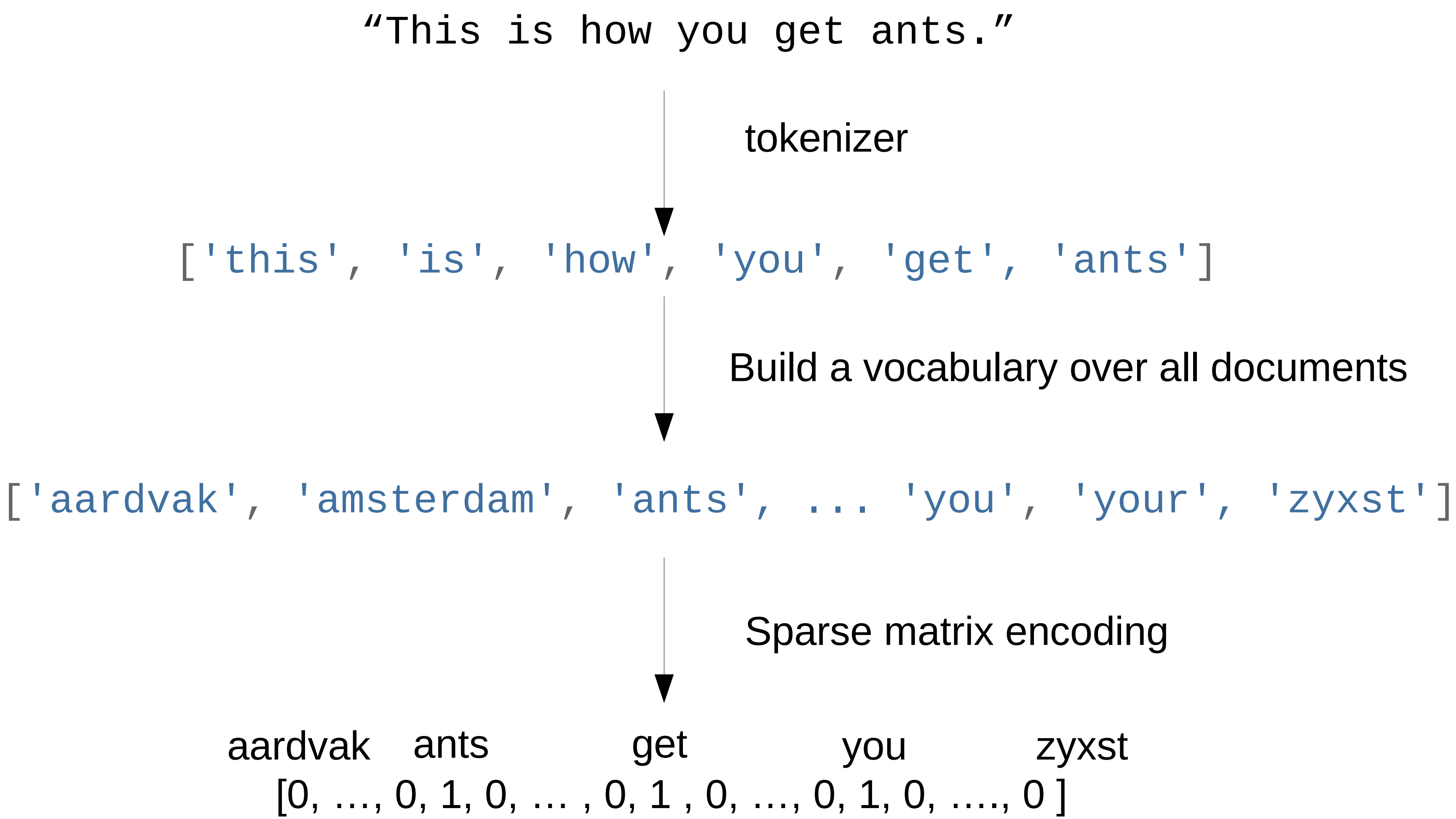
```
['this', 'is', 'how', 'you', 'get', 'ants']
```

Build a vocabulary over all documents

```
['aardvak', 'amsterdam', 'ants', ... 'you', 'your', 'zyxst']
```

Bag Of Word Representations

`CountVectorizer / TfidfVectorizer`



```
“This is how you get ants.”
```

tokenizer

```
['this', 'is', 'how', 'you', 'get', 'ants']
```

Build a vocabulary over all documents

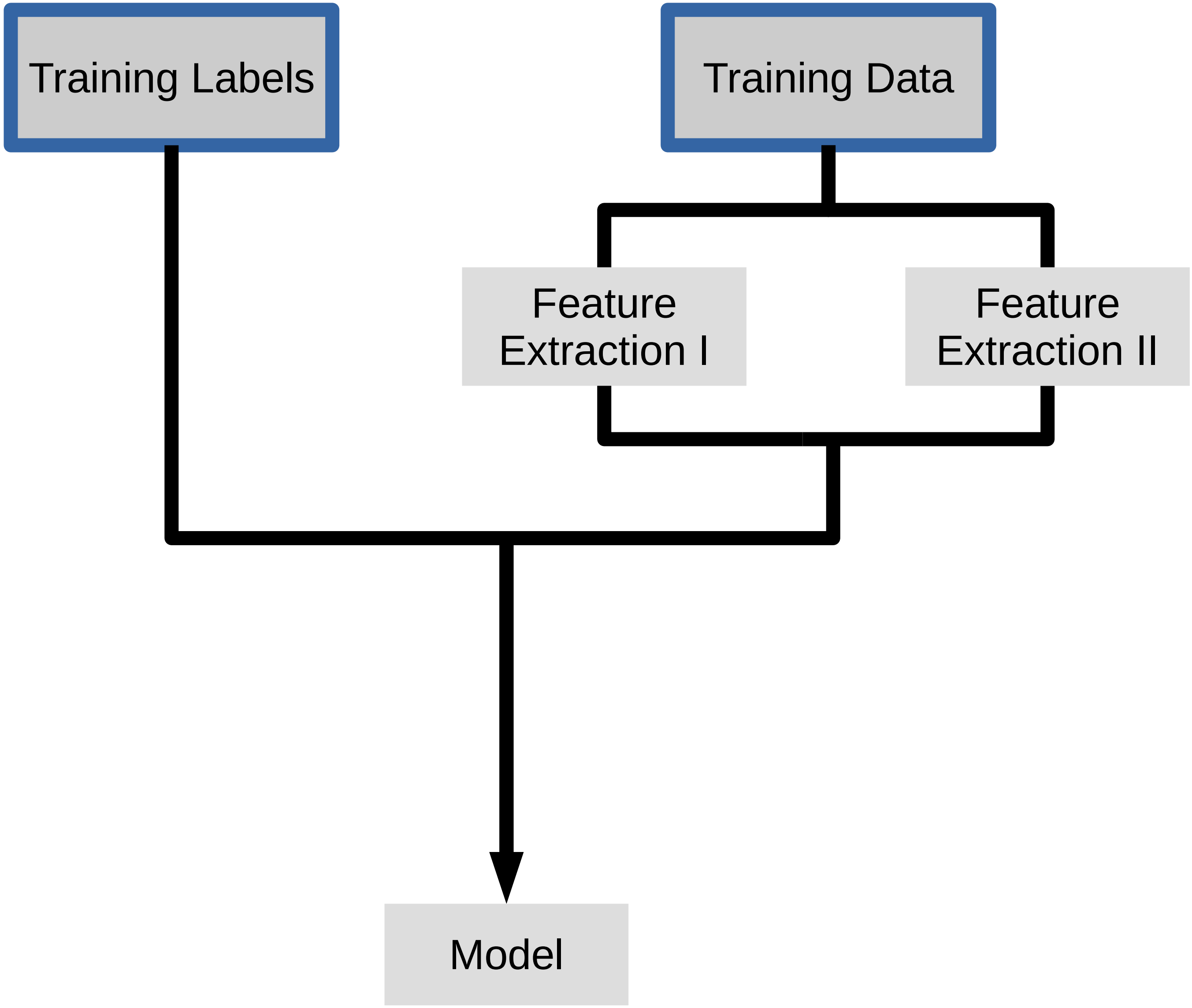
```
['aardvak', 'amsterdam', 'ants', ... 'you', 'your', 'zyxst']
```

Sparse matrix encoding

```
aardvak   ants          get          you          zyxst
   [0, …, 0, 1, 0, … , 0, 1 , 0, …, 0, 1, 0, …., 0 ]
```

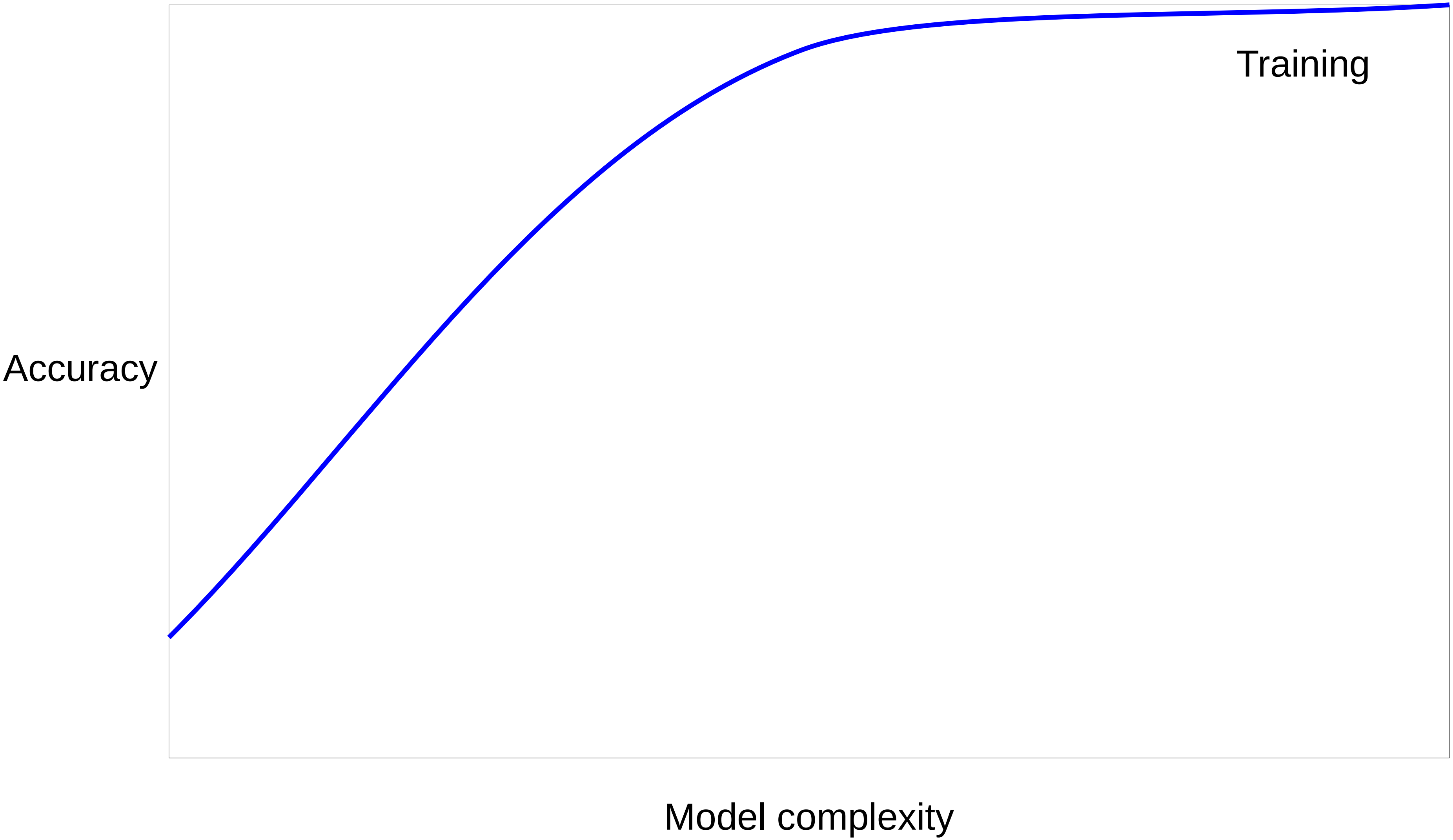
IPython Notebook:
Part 6 - Working With Text Data

Feature Union

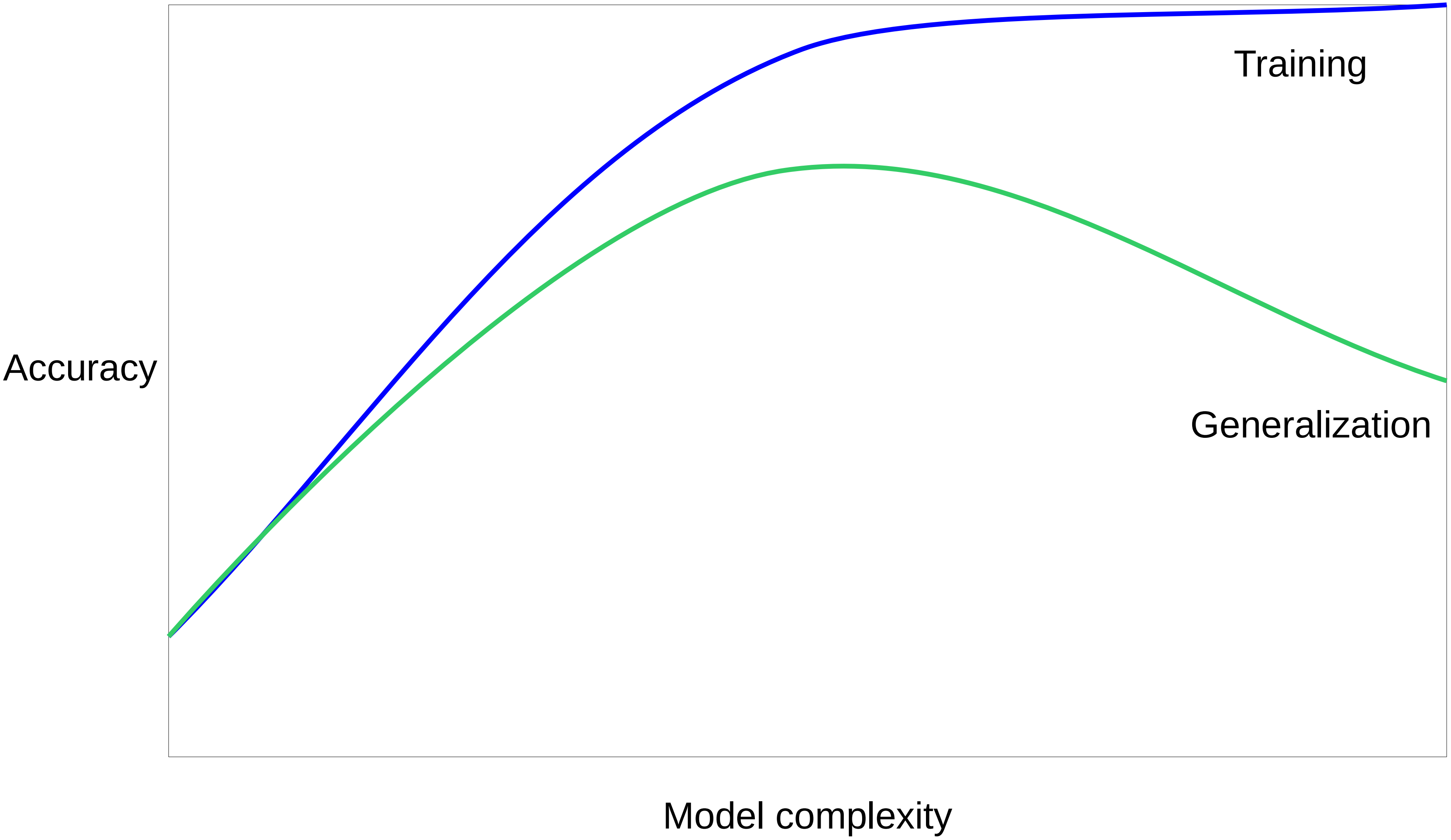


IPython Notebook:
Part 7 – FeatureUnion

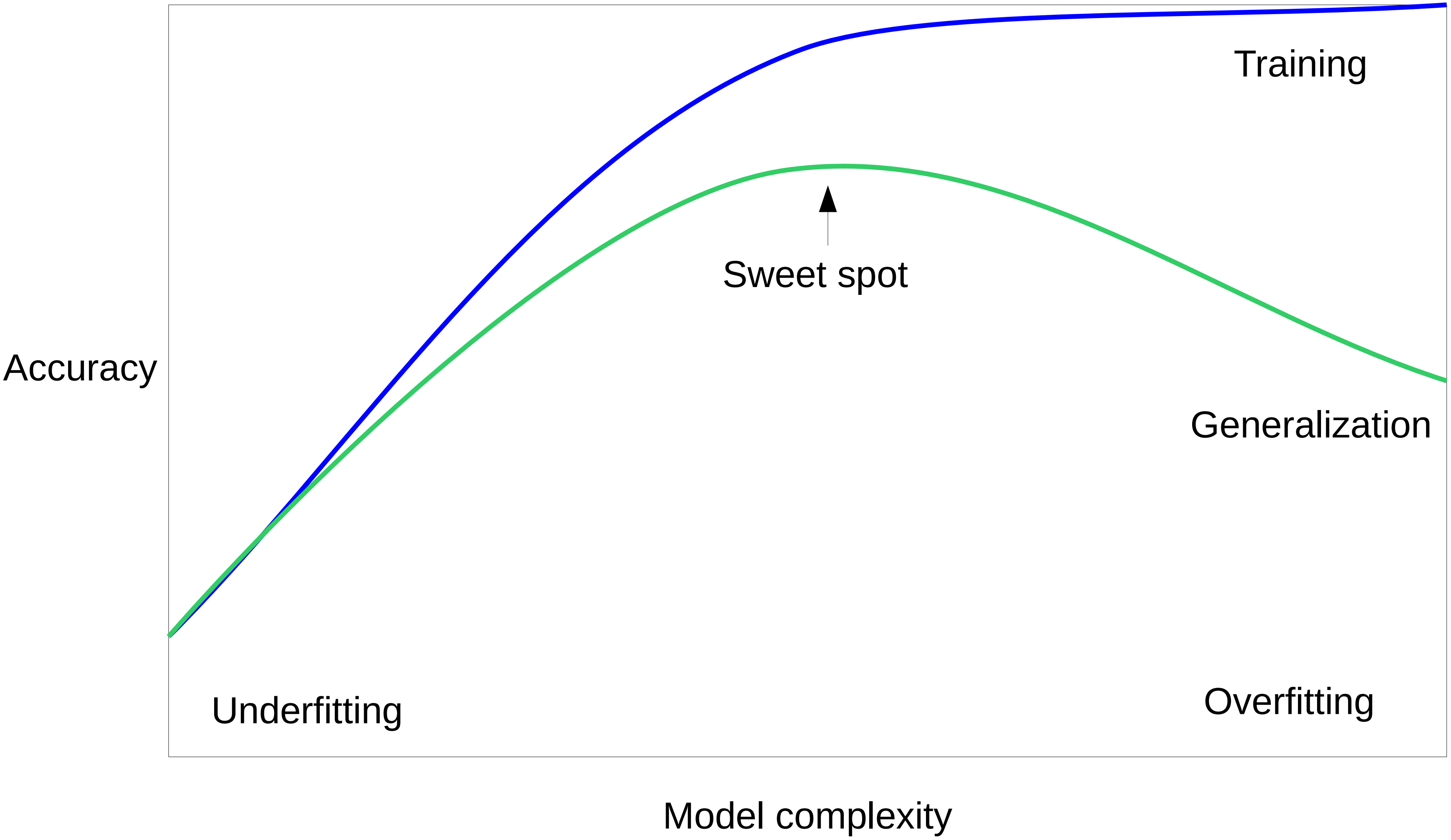
Overfitting and Underfitting



Overfitting and Underfitting



Overfitting and Underfitting

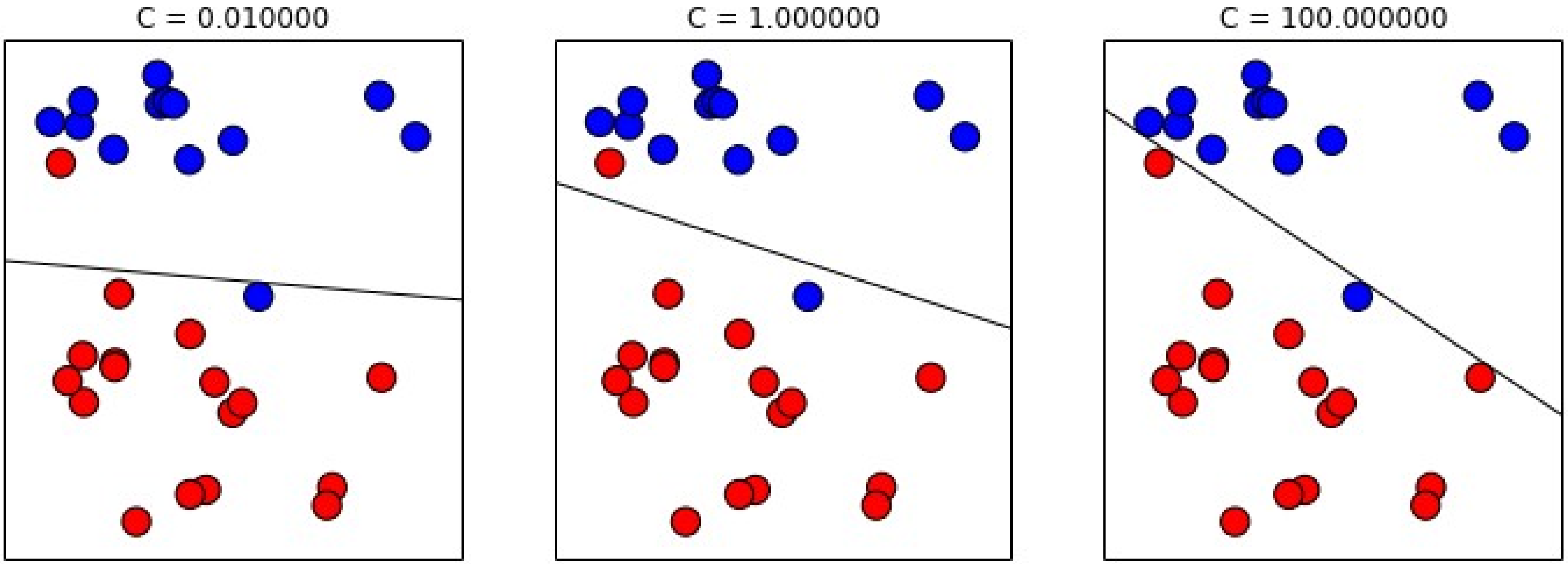


Linear SVM

$$\hat{y} = \mathrm{sign}(w_0 + \sum_i w_i x_i)$$

Linear SVM

$$\hat{y} = \mathrm{sign}(w_0 + \sum_i w_i x_i)$$



(RBF) Kernel SVM

$$\hat{y} = \text{sign}(\alpha_0 + \sum_j \alpha_j y_j k(\mathbf{x^{(j)}}, \mathbf{x}))$$

(RBF) Kernel SVM

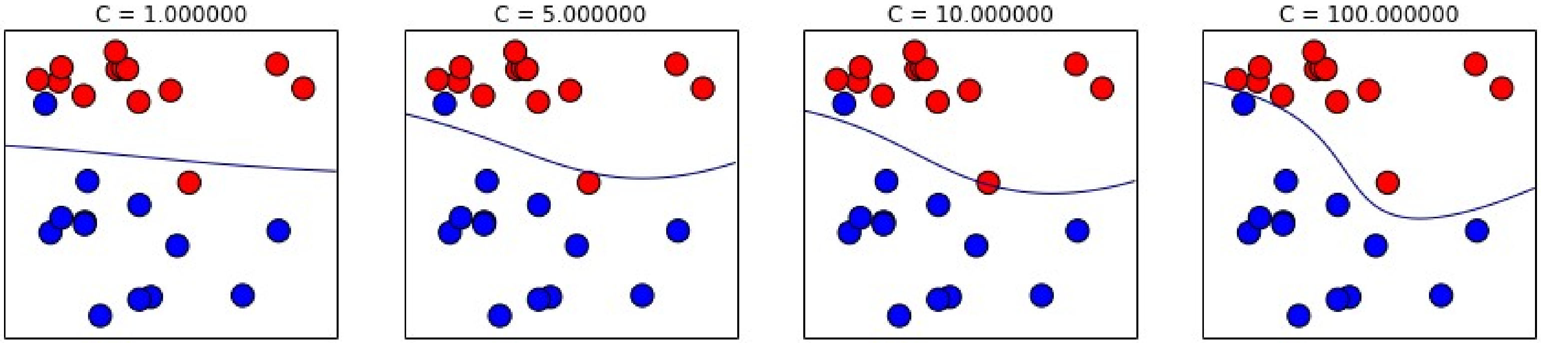
$$\hat{y} = \text{sign}(\alpha_0 + \sum_j \alpha_j y_j k(\mathbf{x}^{(\mathbf{j})}, \mathbf{x}))$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma||\mathbf{x} - \mathbf{x}'||^2)$$

(RBF) Kernel SVM

$$\hat{y} = \mathrm{sign}(\alpha_0 + \sum_j \alpha_j y_j k(\mathbf{x}^{(\mathbf{j})}, \mathbf{x}))$$

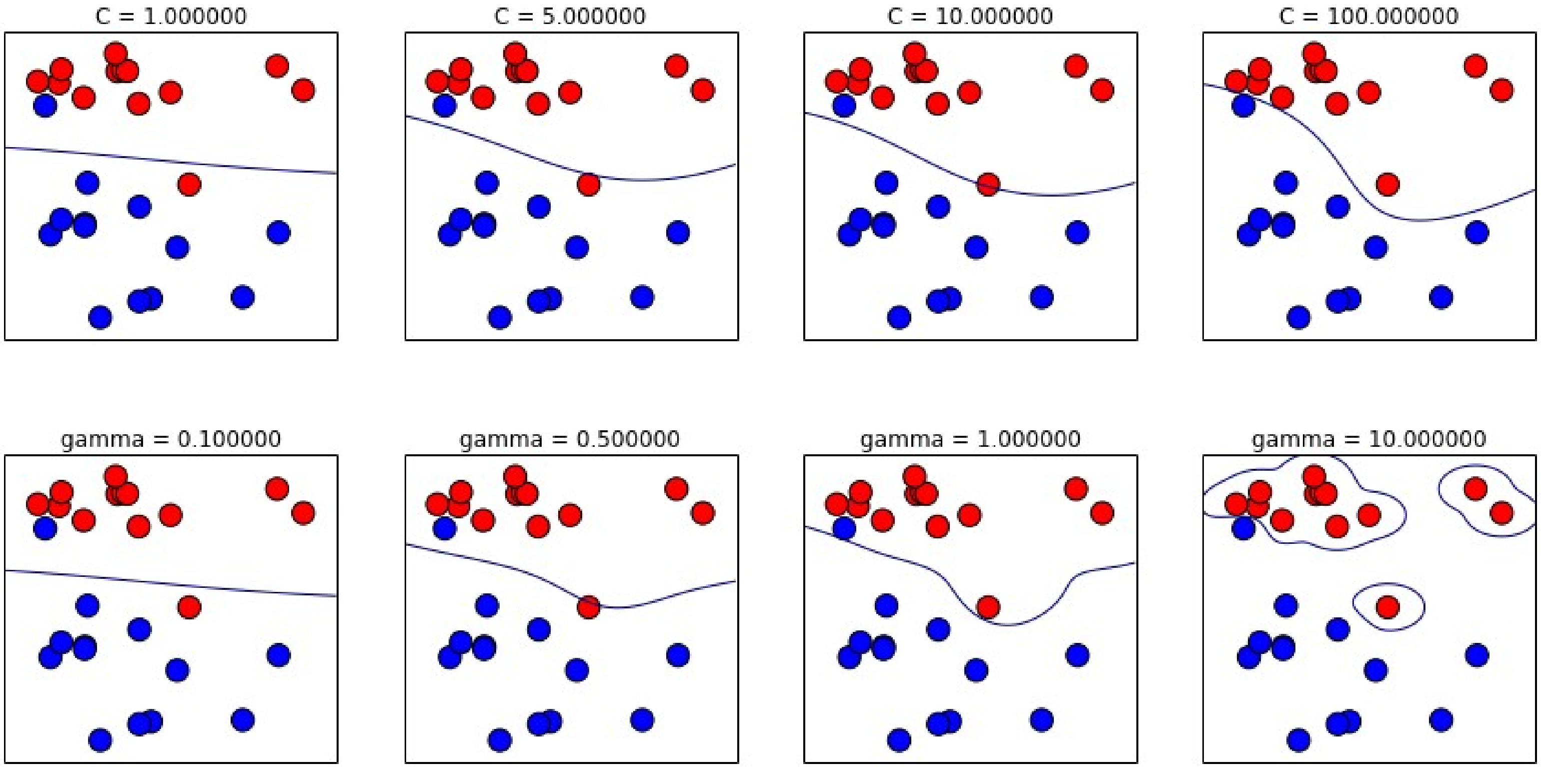
$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma ||\mathbf{x} - \mathbf{x}'||^2)$$



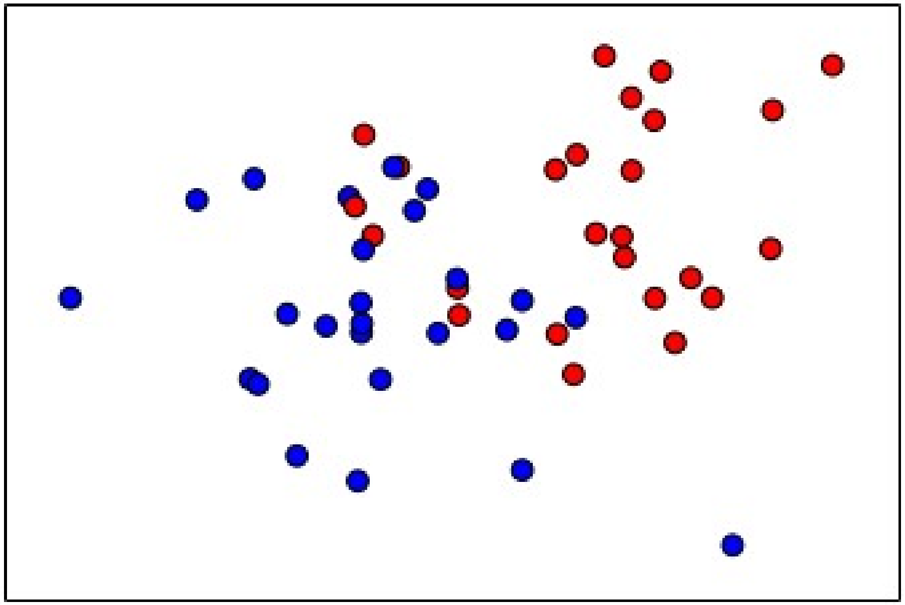
(RBF) Kernel SVM

$$\hat{y} = \mathrm{sign}(\alpha_0 + \sum_j \alpha_j y_j k(\mathbf{x^{(j)}}, \mathbf{x}))$$

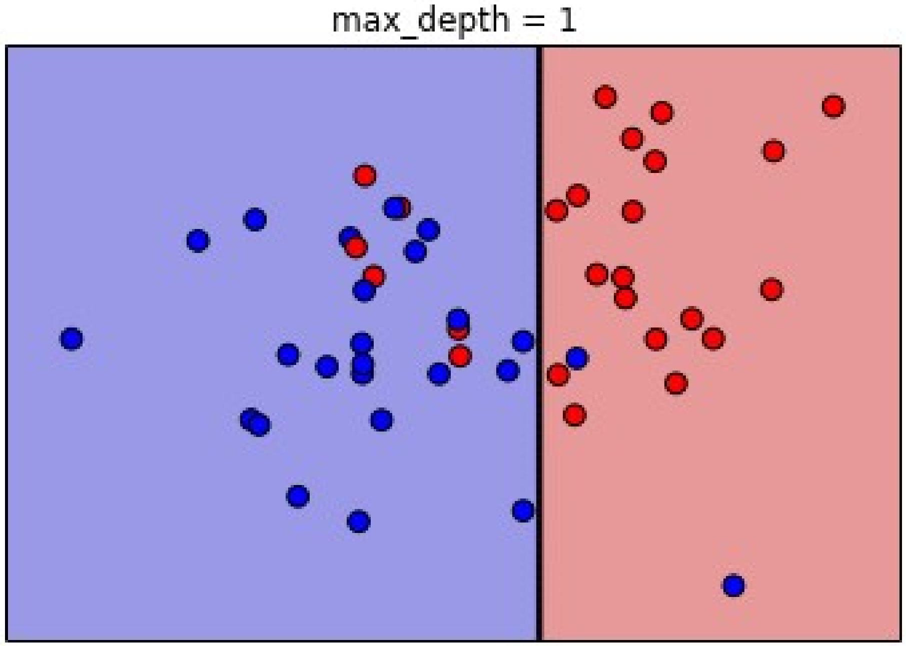
$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma ||\mathbf{x} - \mathbf{x}'||^2)$$

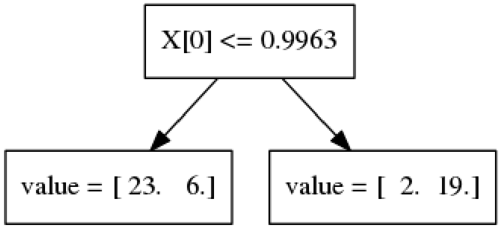


Decision Trees



Decision Trees

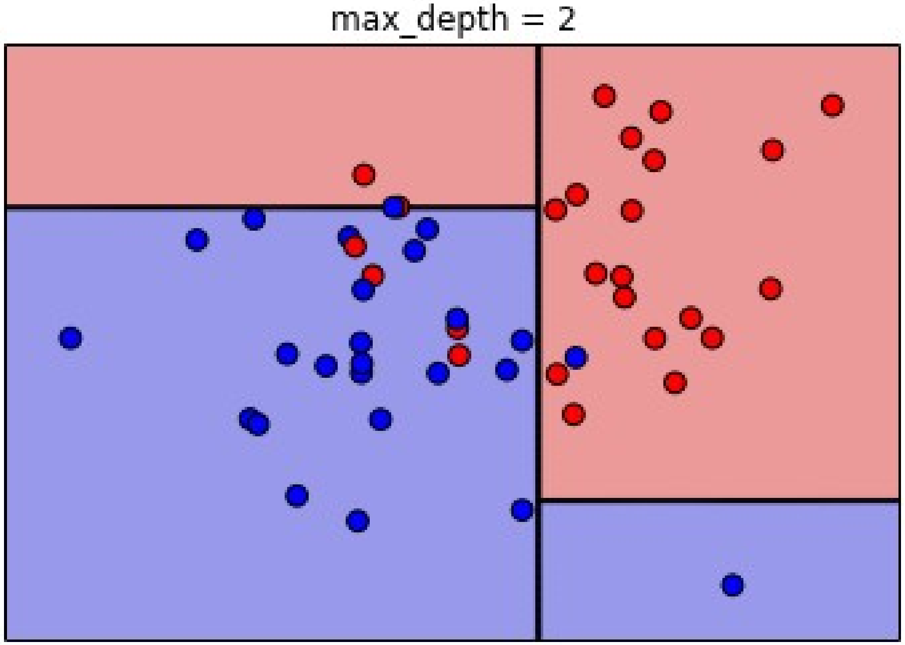


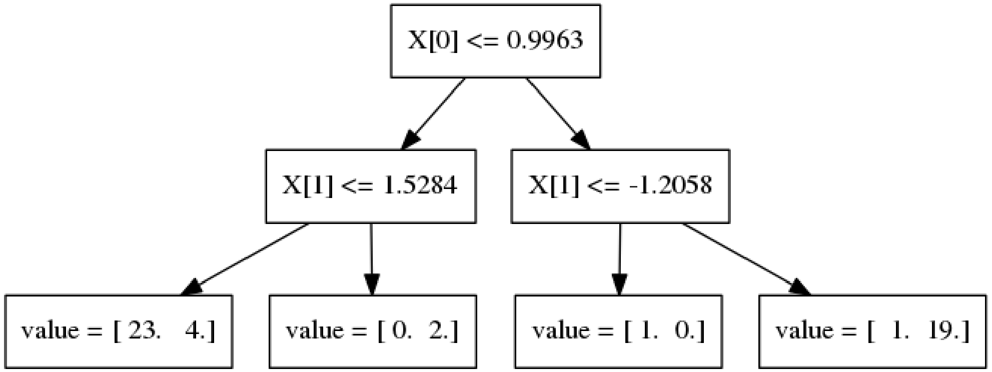

X[0] <= 0.9963

value = [23.   6.]

value = [2.   19.]

Decision Trees





X[0] <= 0.9963

X[1] <= 1.5284

X[1] <= -1.2058

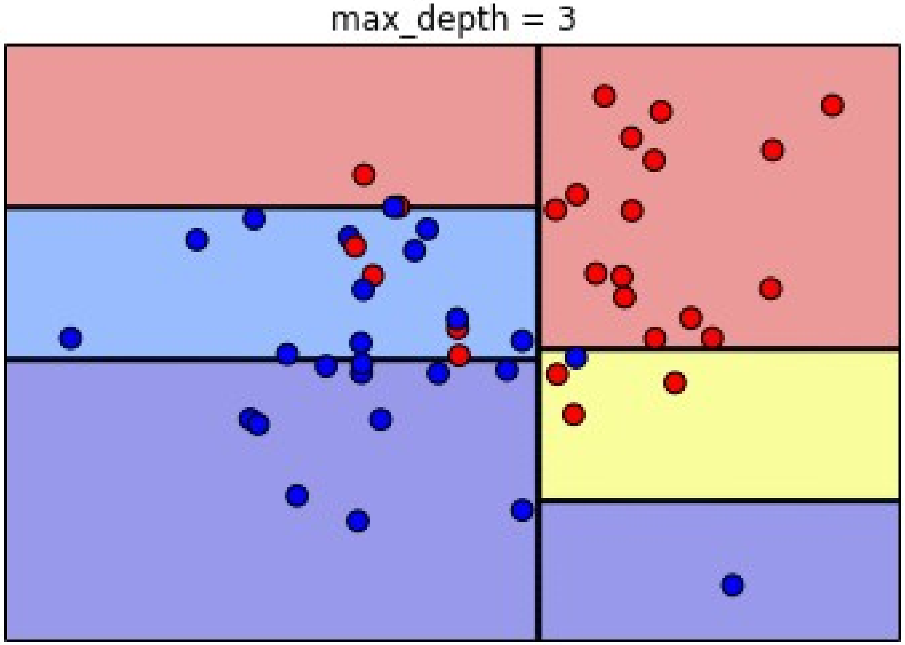
value = [23.   4.]

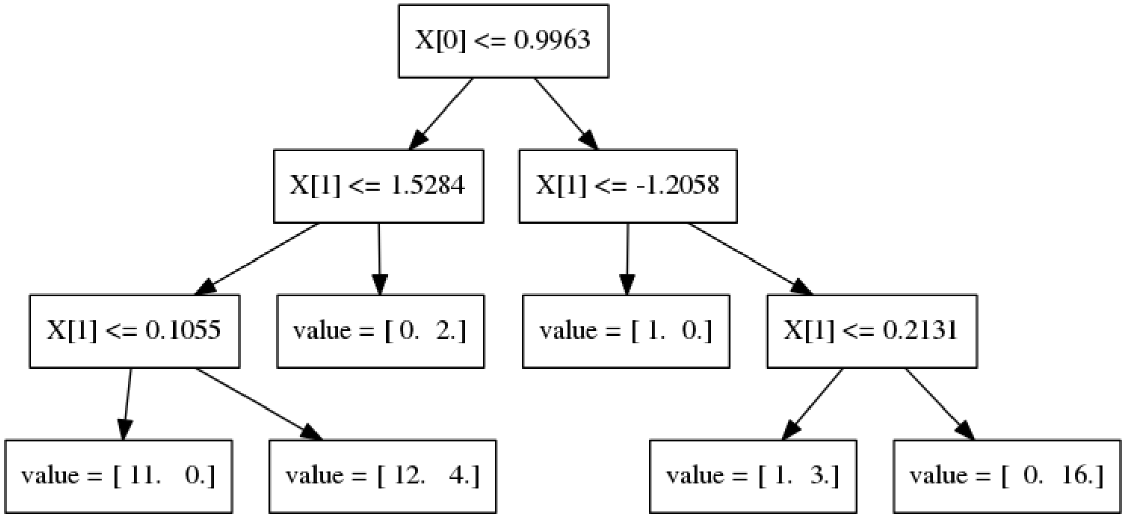
value = [0. 2.]

value = [1. 0.]

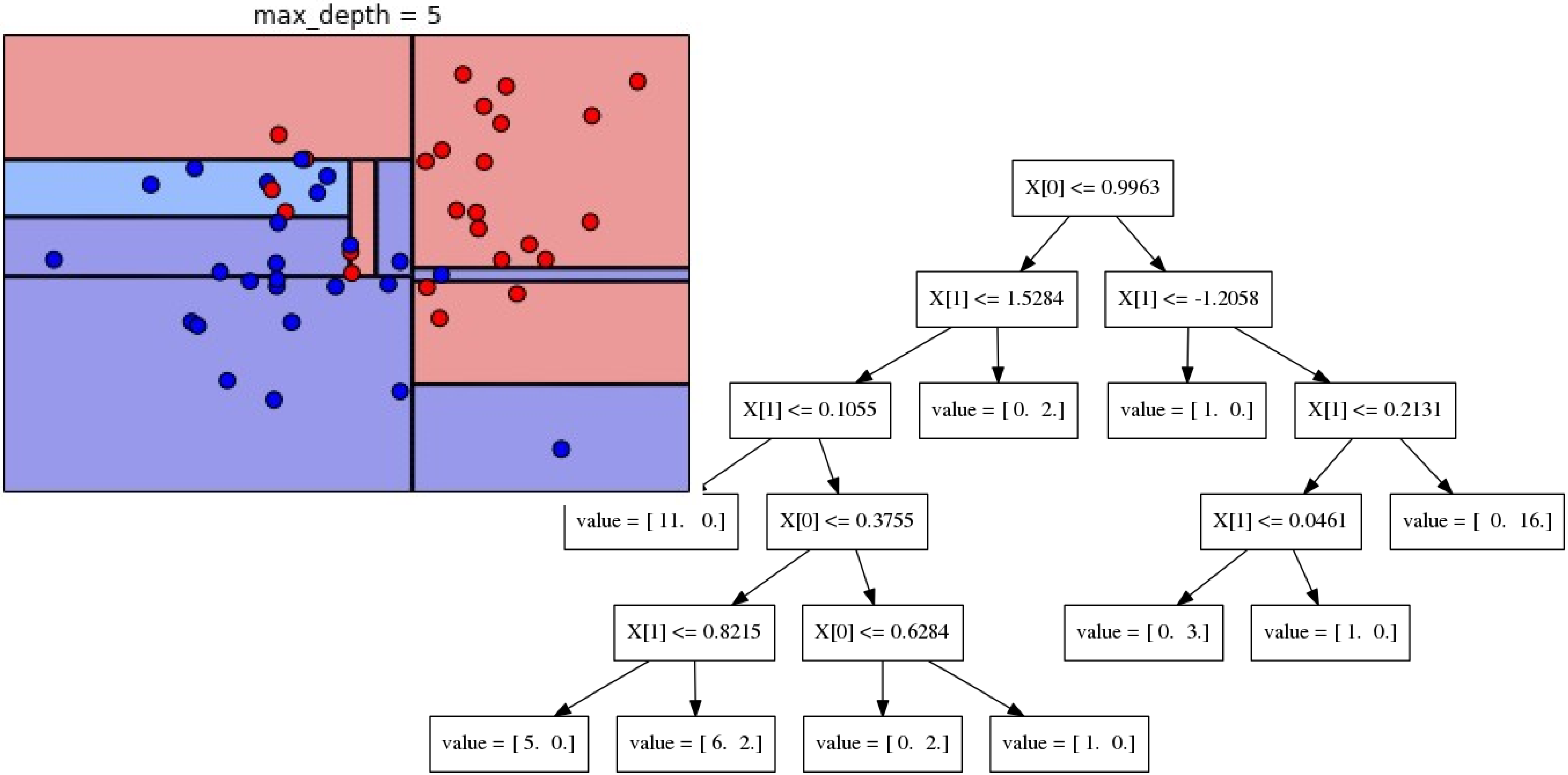
value = [1. 19.]

Decision Trees





Decision Trees



X[0] <= 0.9963

X[1] <= 1.5284

X[1] <= -1.2058

X[1] <= 0.1055

value = [0. 2.]

value = [1. 0.]

X[1] <= 0.2131

value = [11.   0.]

X[0] <= 0.3755

X[1] <= 0.0461

value = [0. 16.]

X[1] <= 0.8215

X[0] <= 0.6284

value = [0. 3.]

value = [1. 0.]

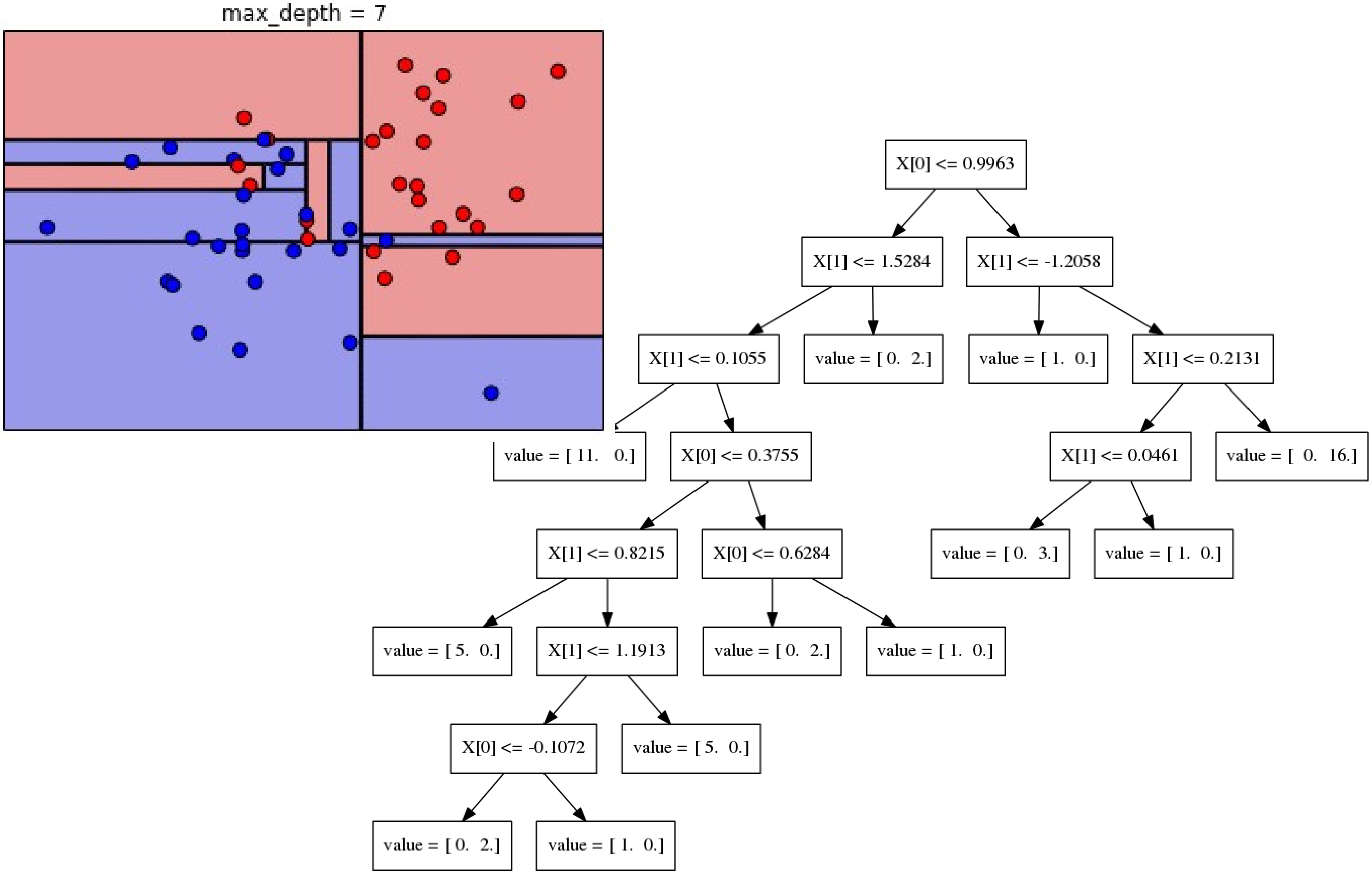
value = [5. 0.]

value = [6. 2.]

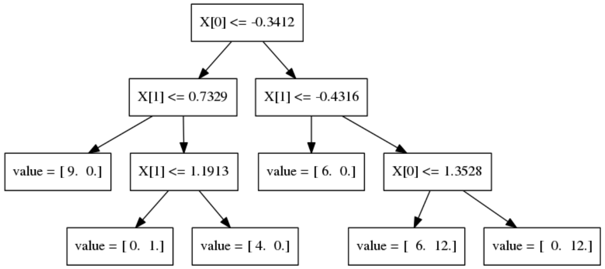
value = [0. 2.]

value = [1. 0.]

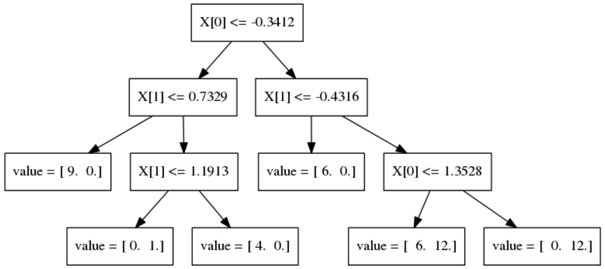
Decision Trees

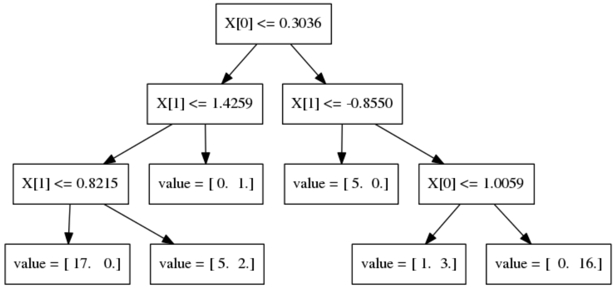


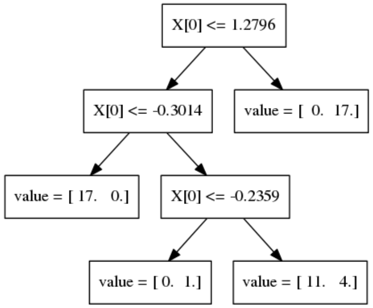
Random Forests



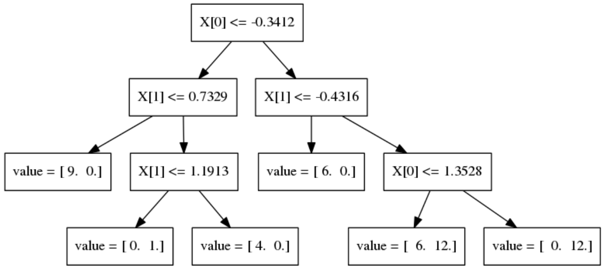
Random Forests

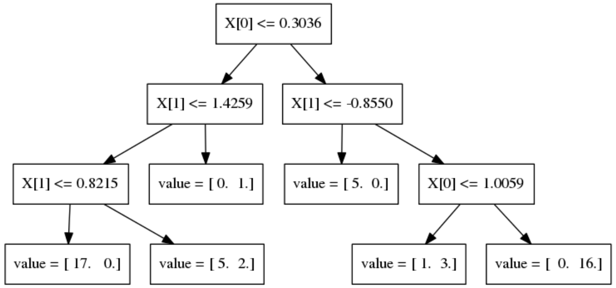


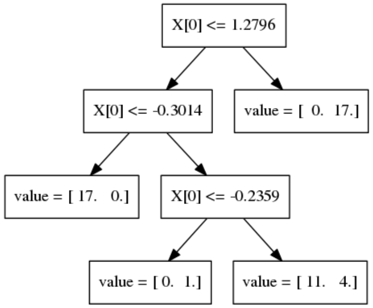


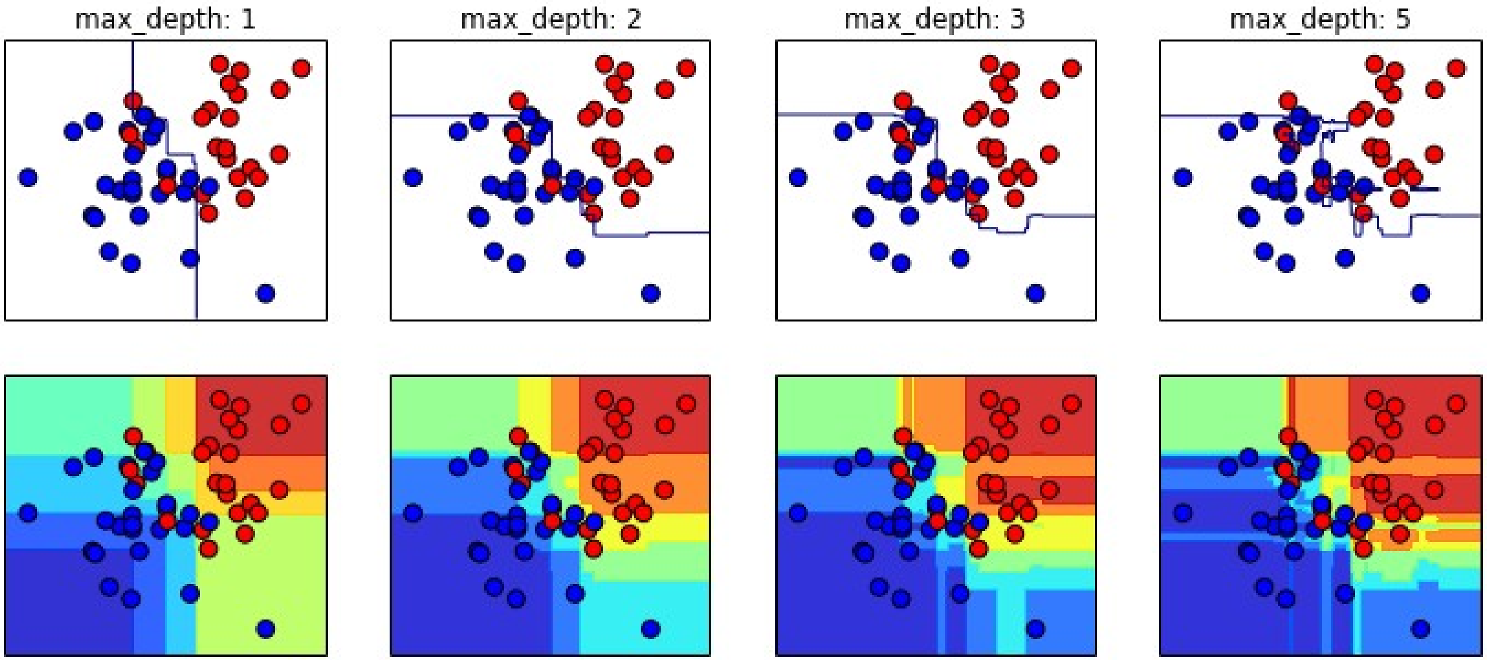


Random Forests



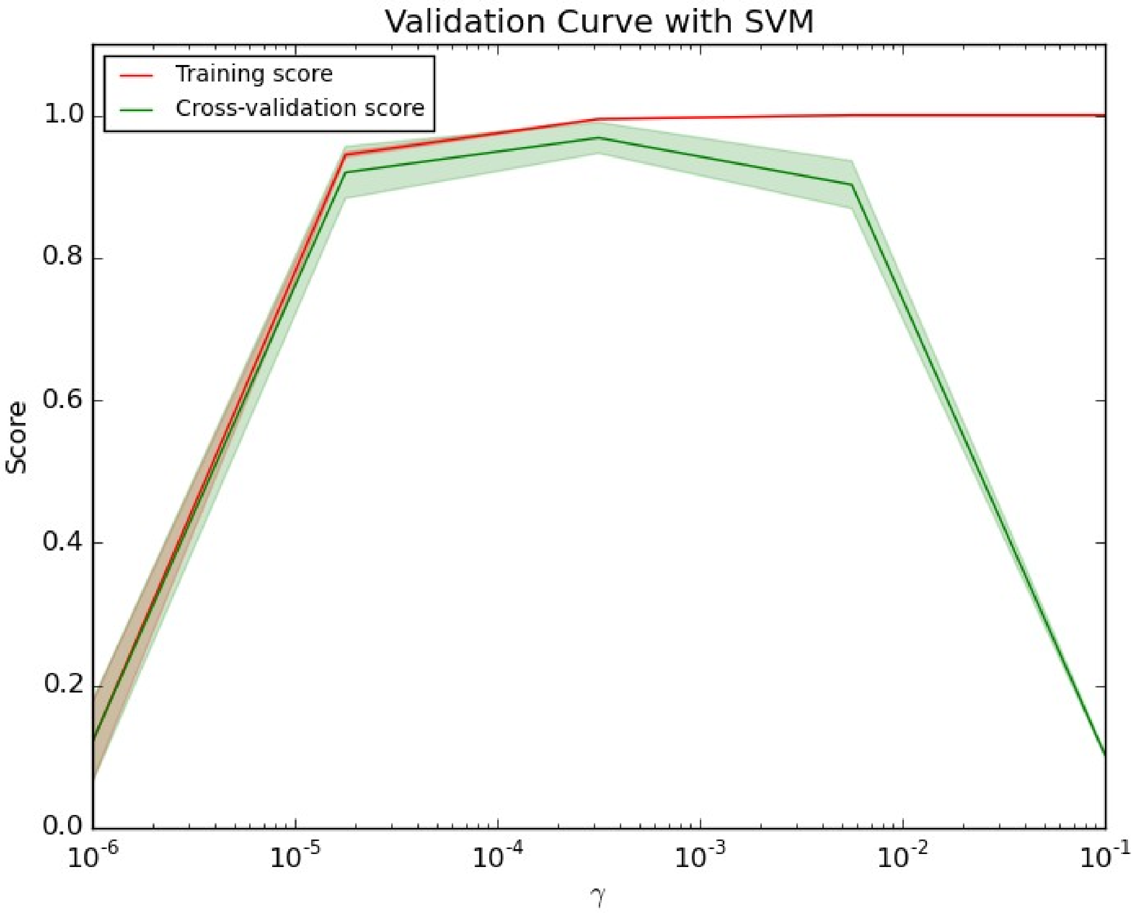






Validation Curves

```
train_scores, test_scores = validation_curve(SVC(), X, y,
    param_name="gamma", param_range=param_range)
```



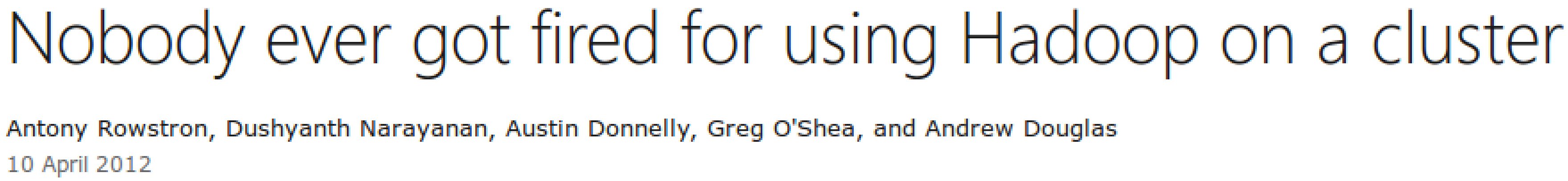
Scaling Up

Three regimes of data

- Fits in RAM
- Fits on a Hard Drive
- Doesn't fit on a single PC

Three regimes of data

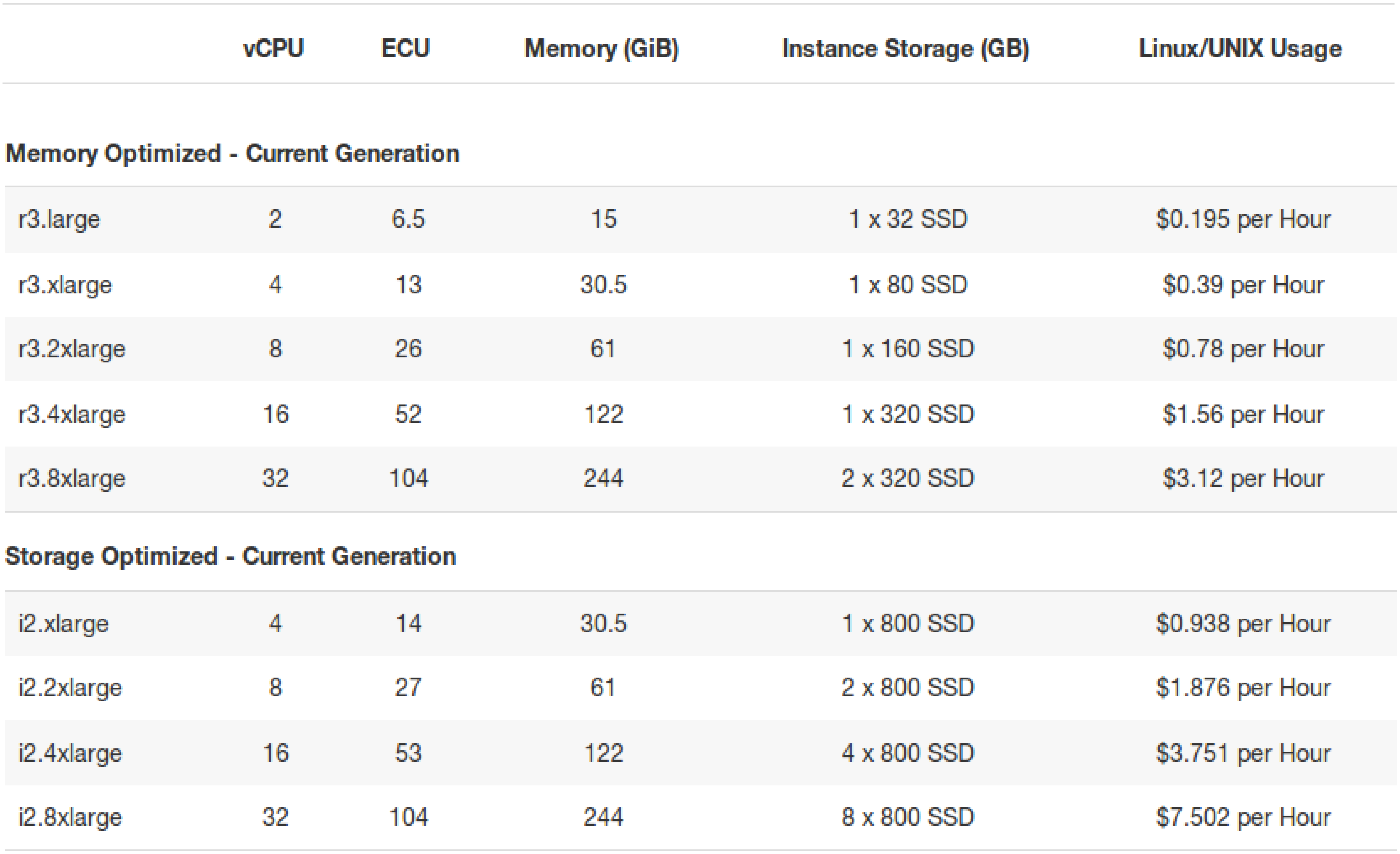
- Fits in RAM (up to 256 GB?)
- Fits on a Hard Drive (up to 6TB?)
- Doesn't fit on a single PC



Nobody ever got fired for using Hadoop on a cluster

Antony Rowstron, Dushyanth Narayanan, Austin Donnelly, Greg O'Shea, and Andrew Douglas

10 April 2012



	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
Memory Optimized - Current Generation					
r3.large	2	6.5	15	1 x 32 SSD	$0.195 per Hour
r3.xlarge	4	13	30.5	1 x 80 SSD	$0.39 per Hour
r3.2xlarge	8	26	61	1 x 160 SSD	$0.78 per Hour
r3.4xlarge	16	52	122	1 x 320 SSD	$1.56 per Hour
r3.8xlarge	32	104	244	2 x 320 SSD	$3.12 per Hour
Storage Optimized - Current Generation					
i2.xlarge	4	14	30.5	1 x 800 SSD	$0.938 per Hour
i2.2xlarge	8	27	61	2 x 800 SSD	$1.876 per Hour
i2.4xlarge	16	53	122	4 x 800 SSD	$3.751 per Hour
i2.8xlarge	32	104	244	8 x 800 SSD	$7.502 per Hour

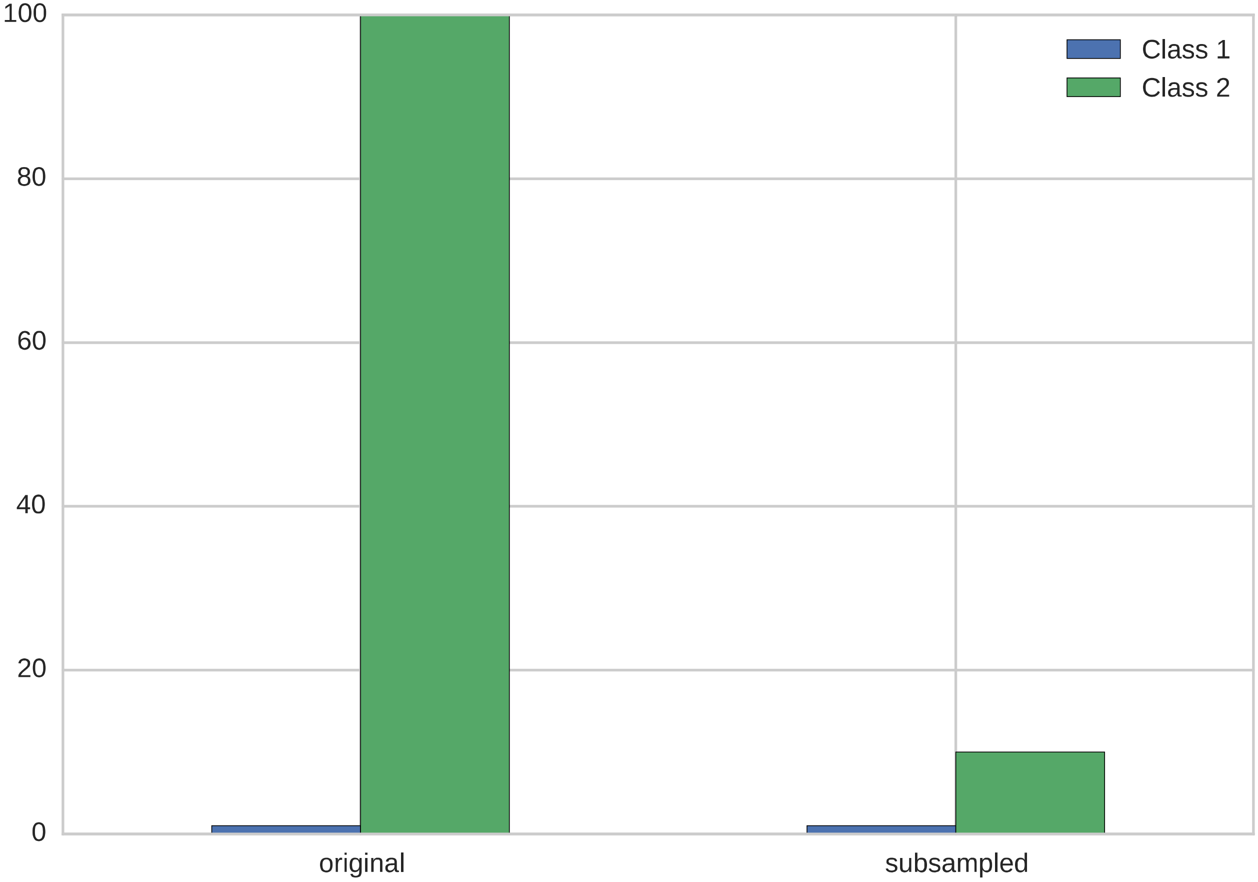
"256Gb ought to be enough for anybody."

- me

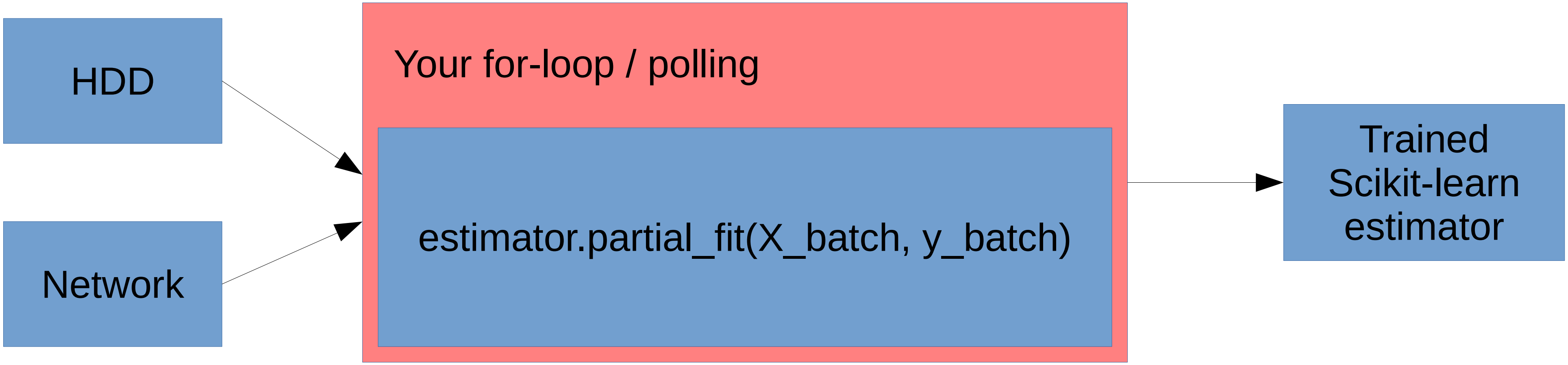
"256Gb ought to be enough for anybody."
- me

(for machine learning)

Subsample!



The scikit-learn way


HDD
Network
Your for-loop / polling
estimator.partial_fit(X_batch, y_batch)
Trained Scikit-learn estimator

Supported Algorithms

- All `SGDClassifier` derivatives
- Naive Bayes
- `MinibatchKMeans`
- `Birch`
- `IncrementalPCA`
- `MiniBatchDictionaryLearning`

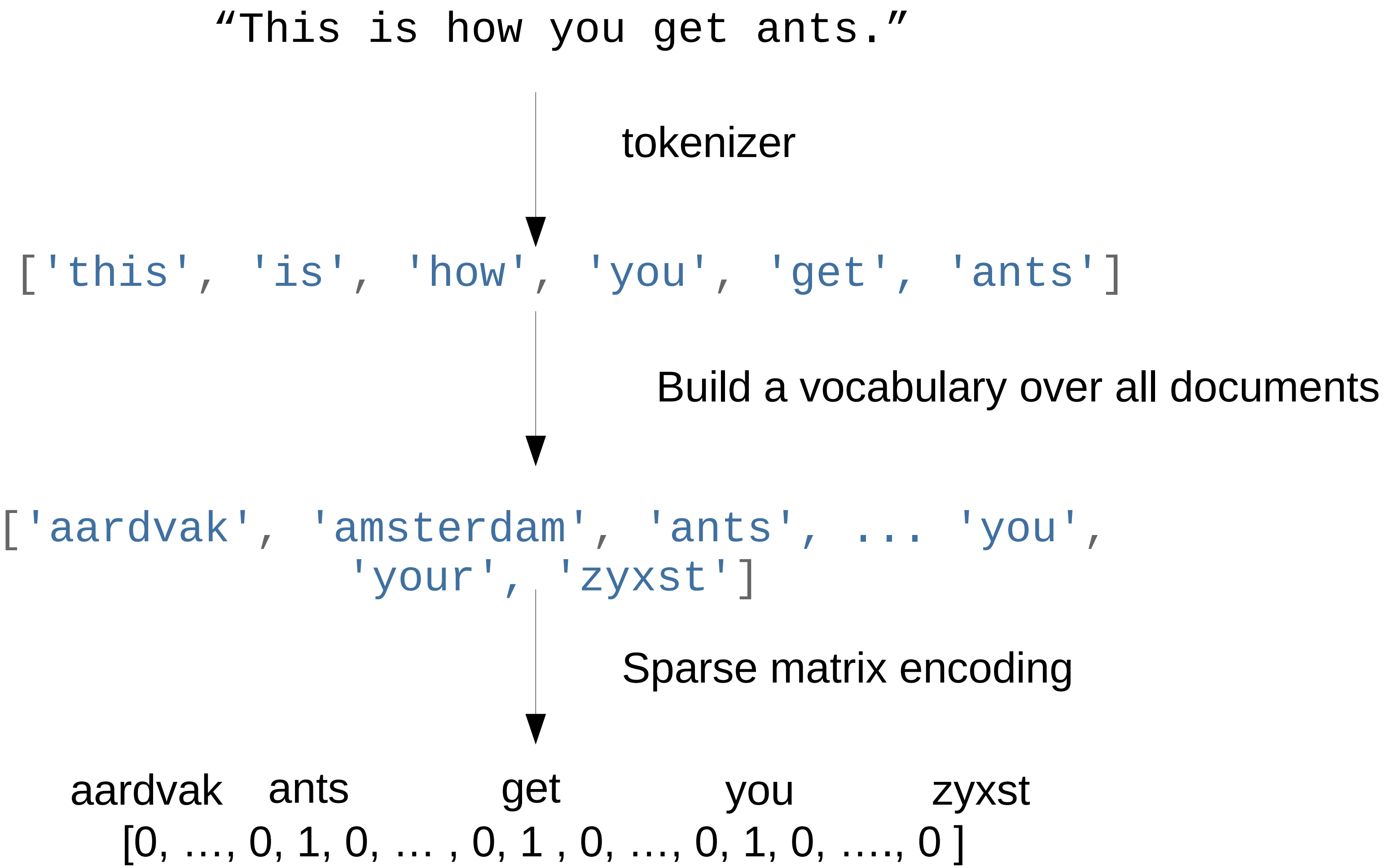
IPython Notebook:
Part 8 – Out Of Core Learning

Stateless Transformers

- `Normalizer`
- `HashingVectorizer`
- `RBFSampler` (and other kernel approx)

Bag Of Word Representations

`CountVectorizer / TfidfVectorizer`



`“This is how you get ants.”`

tokenizer

`['this', 'is', 'how', 'you', 'get', 'ants']`

Build a vocabulary over all documents

`['aardvak', 'amsterdam', 'ants', ... 'you', 'your', 'zyxst']`

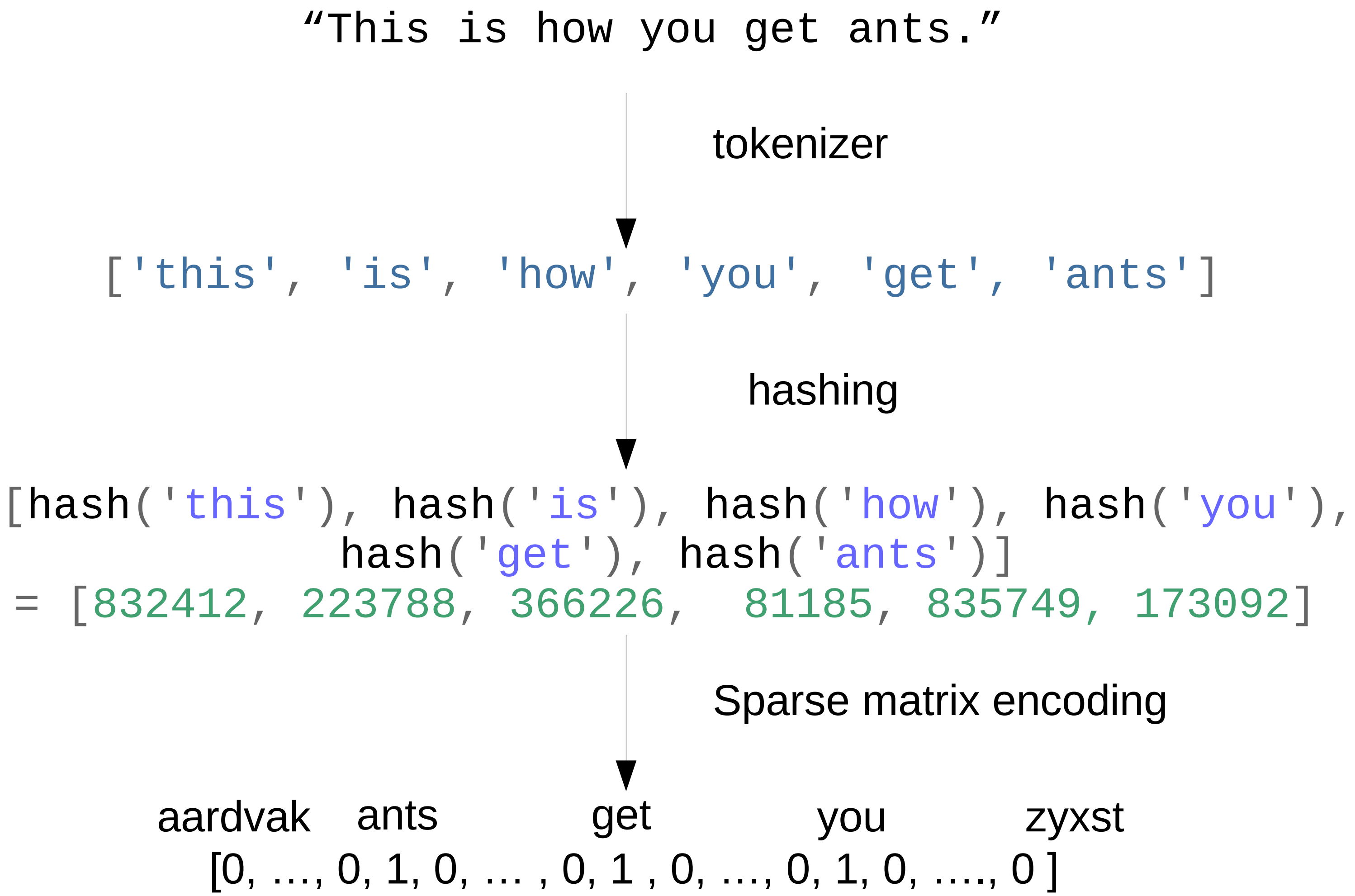
Sparse matrix encoding

aardvak ants get you zyxst

[0, …, 0, 1, 0, … , 0, 1 , 0, …, 0, 1, 0, …., 0]

Hashing Trick

`HashingVectorizer`



“This is how you get ants.”

tokenizer

```
['this', 'is', 'how', 'you', 'get', 'ants']
```

hashing

```
[hash('this'), hash('is'), hash('how'), hash('you'),
hash('get'), hash('ants')]
= [832412, 223788, 366226,  81185, 835749, 173092]
```

Sparse matrix encoding

aardvak ants get you zyxst

[0, …, 0, 1, 0, … , 0, 1 , 0, …, 0, 1, 0, …., 0]

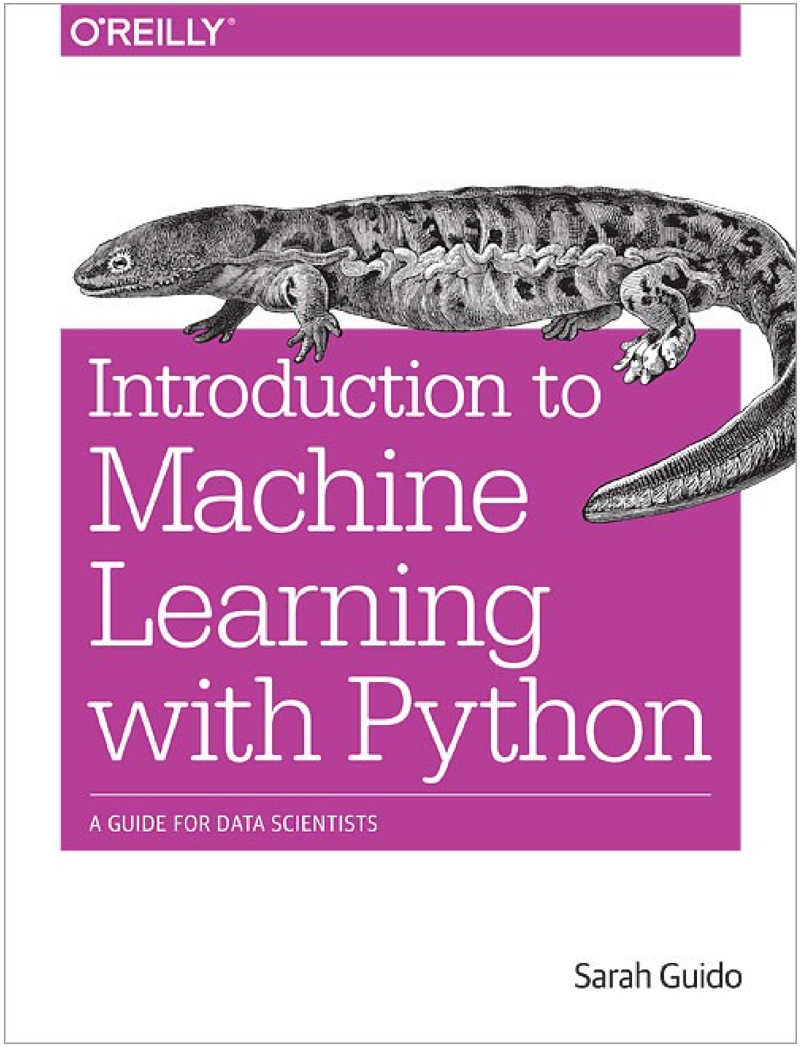
IPython Notebook:
Part 9 – Out Of Core Learning for Text

Video Series
Advanced Machine Learning with scikit-learn

50% Off Coupon Code: AUTHD

Video Series
Advanced Machine Learning with scikit-learn

50% Off Coupon Code: AUTHD


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