

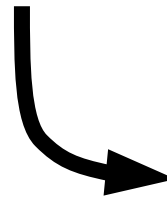
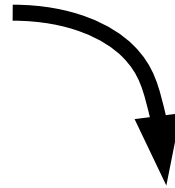


Machine Learning with Scikit-Learn

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

<http://bit.ly/sklodsc>

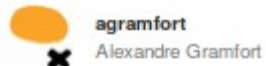
Me



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



GaelVaroquaux
Gael Varoquaux



glouppe
Gilles Louppe



jakevdp
Jake Vanderplas



jaquesgrobler
Jaques Grobler



jnothman



kastnerkyle
Kyle Kastner



bthirion
bthirion



chrisfilo
Chris Filo Gorgole...



cournape
David Cournapeau



duchesnay
Duchesnay



dwf
David Warde-Farley



fabianp
Fabian Pedregosa



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!

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          ]
```

<http://bit.ly/sklosdc>

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 from the first tutorial speaker, presenting me with a message, saying he's ill and won't be attending.

I know you have a lot of experience in this field, so I thought I would ask yourself to give me two presentations on the topic of machine learning. I think you could increase your audience by offering a free trial of your course, 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,

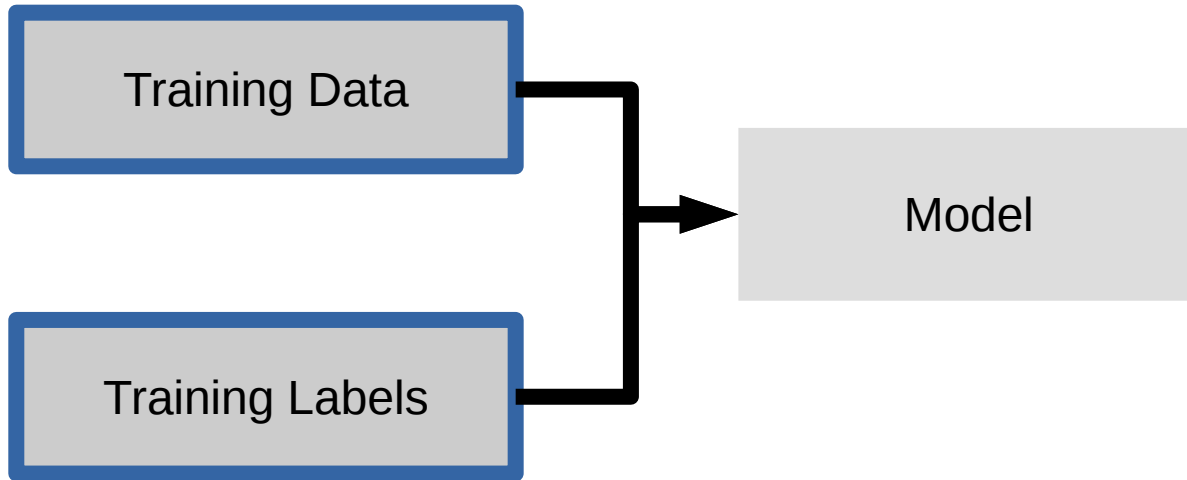
I am very interested in your Machine Learning background. I work for X Recruiting who have been engaged by Z, a worldwide leading supplier of Y. We are expanding the core engineering team and we are looking for really passionate engineers who can tell their own story and help millions of people.

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

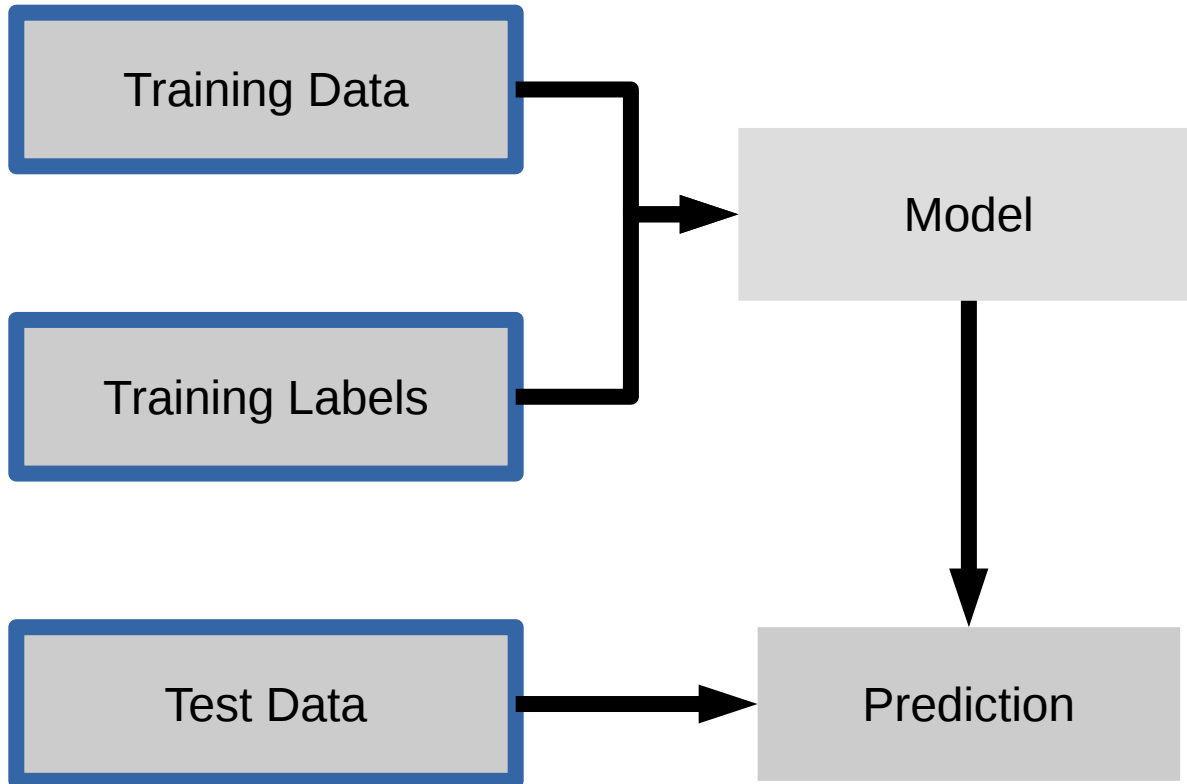
Thanks



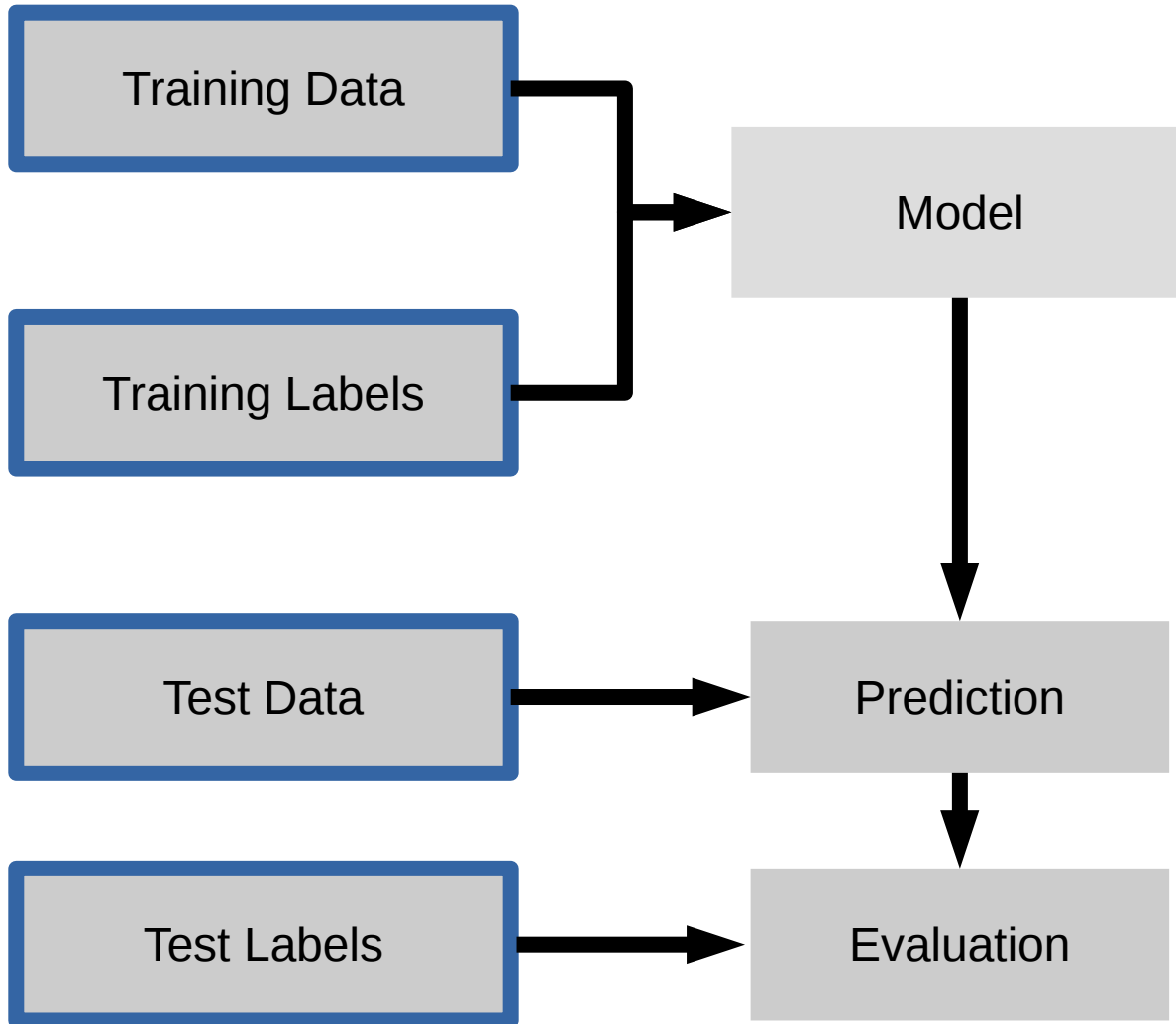
Supervised Machine Learning



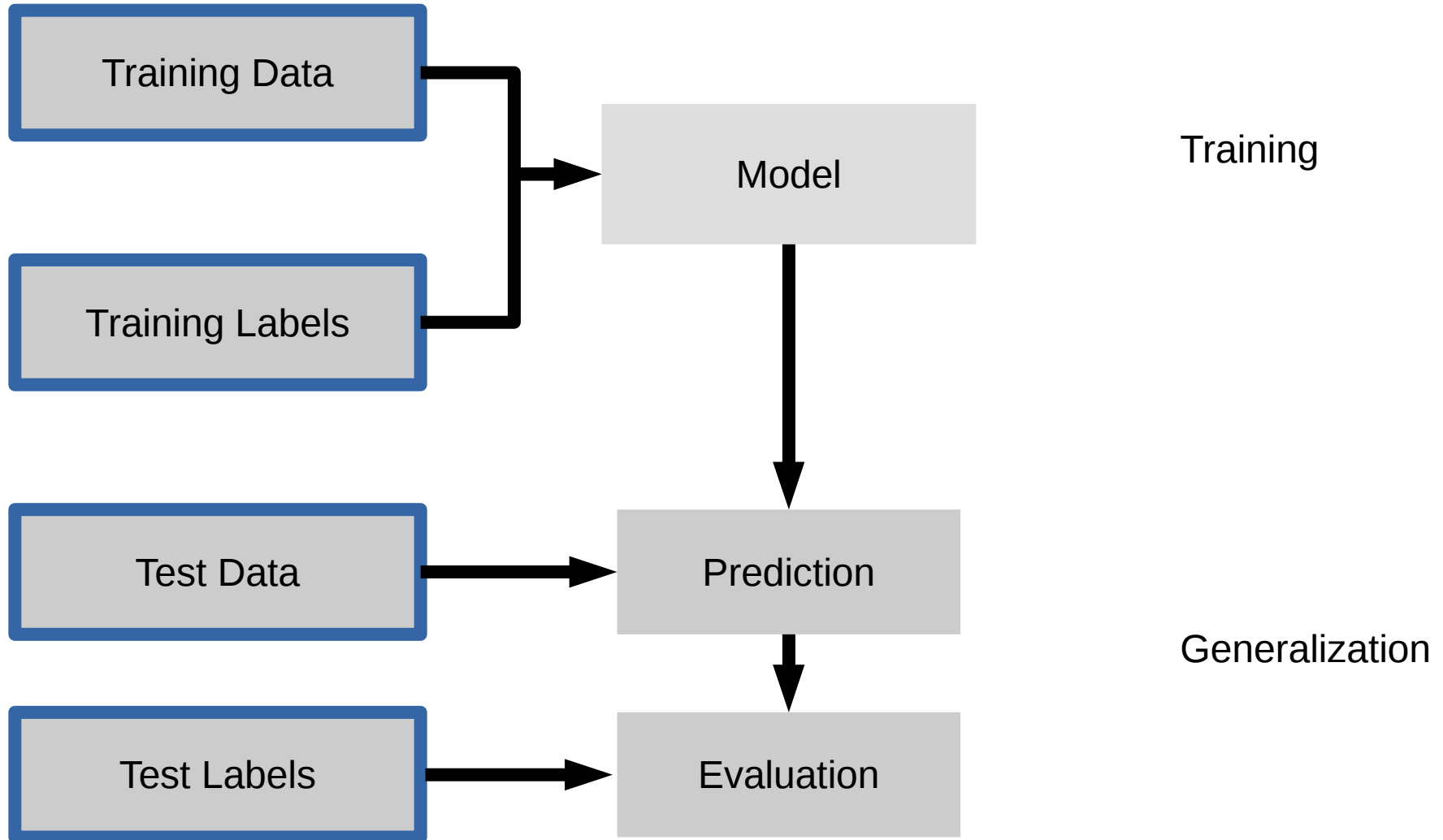
Supervised Machine Learning



Supervised Machine Learning



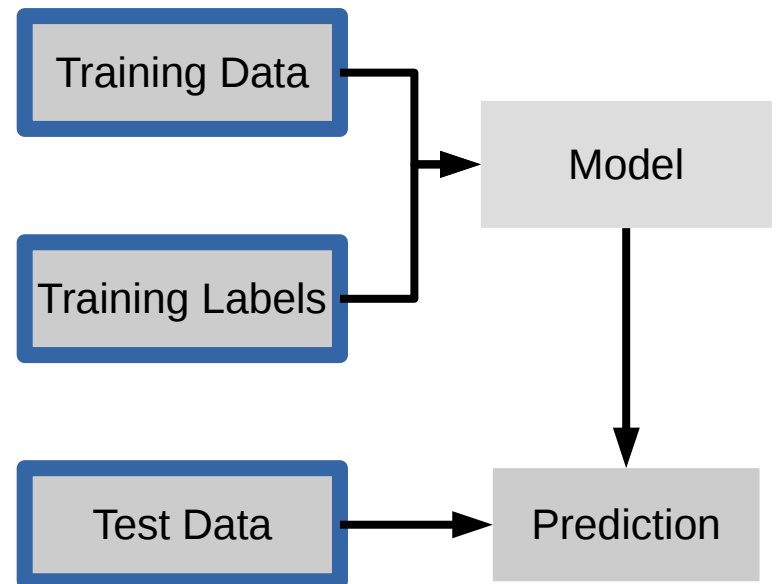
Supervised Machine Learning



```
clf = RandomForestClassifier()
```

```
clf.fit(X_train, y_train)
```

```
y_pred = clf.predict(X_test)
```

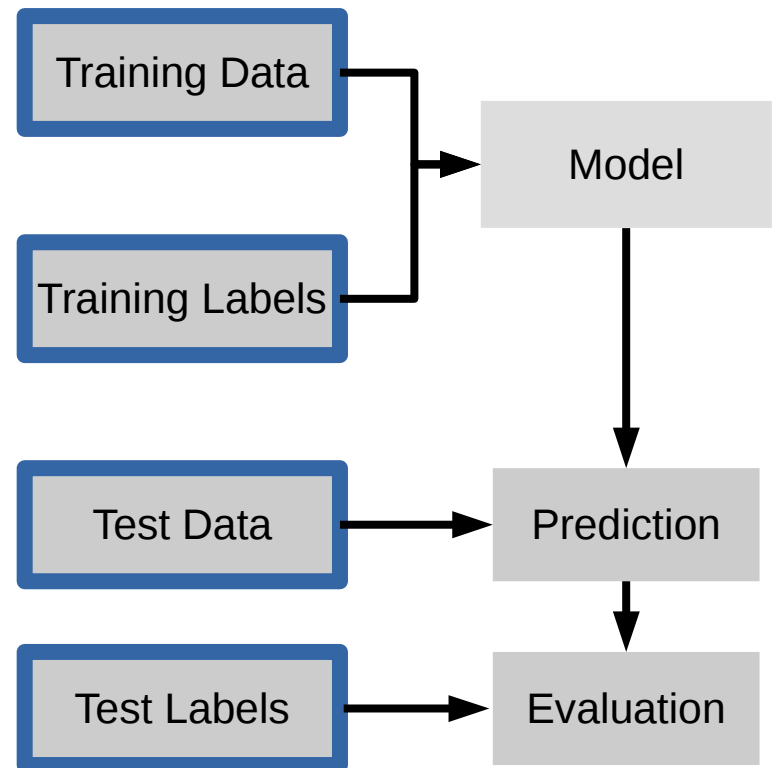


```
clf = RandomForestClassifier()
```

```
clf.fit(X_train, y_train)
```

```
y_pred = clf.predict(X_test)
```

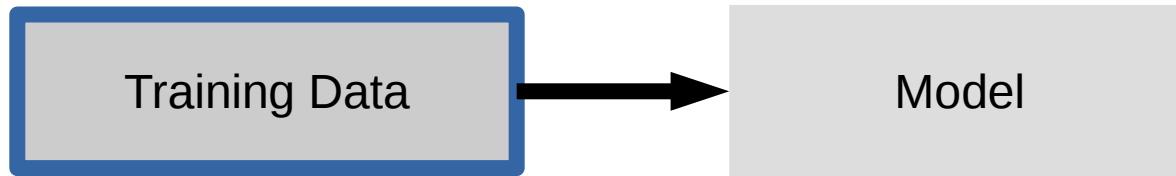
```
clf.score(X_test, y_test)
```



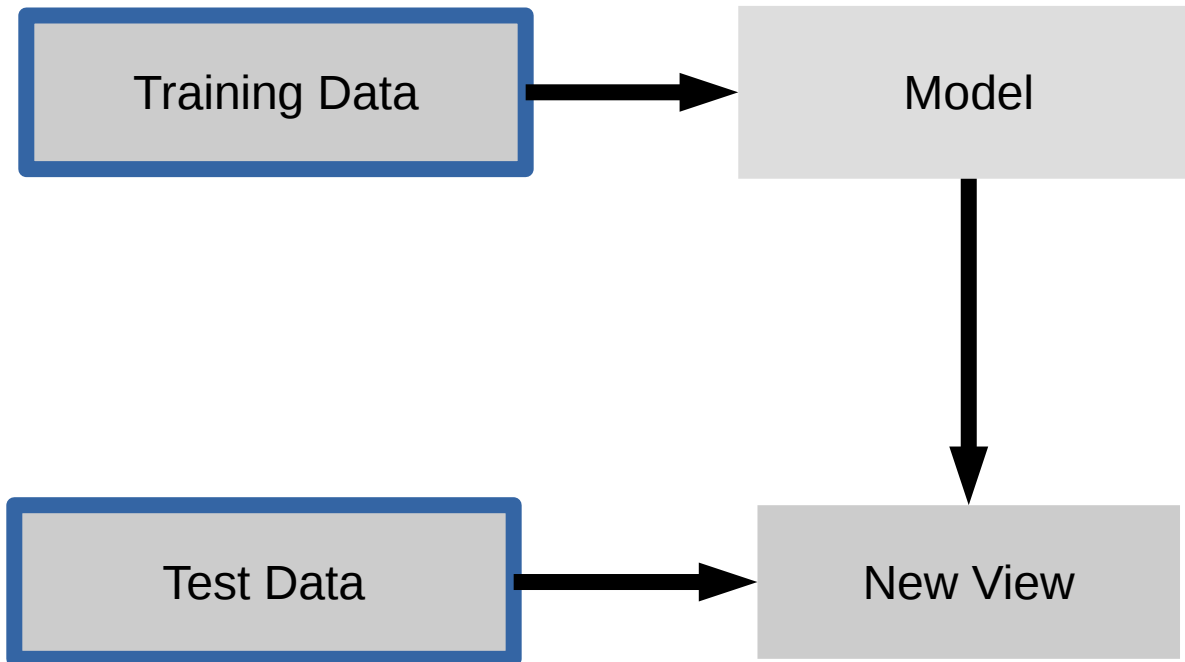
IPython Notebook:

Part 1 - Introduction to Scikit-learn

Unsupervised Machine Learning



Unsupervised Machine Learning

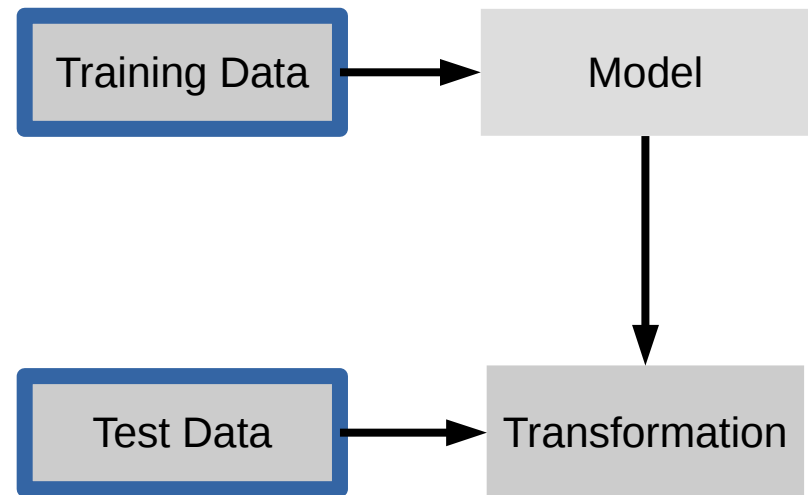


Unsupervised Transformations

```
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

```
In [2]: 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=None)

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)

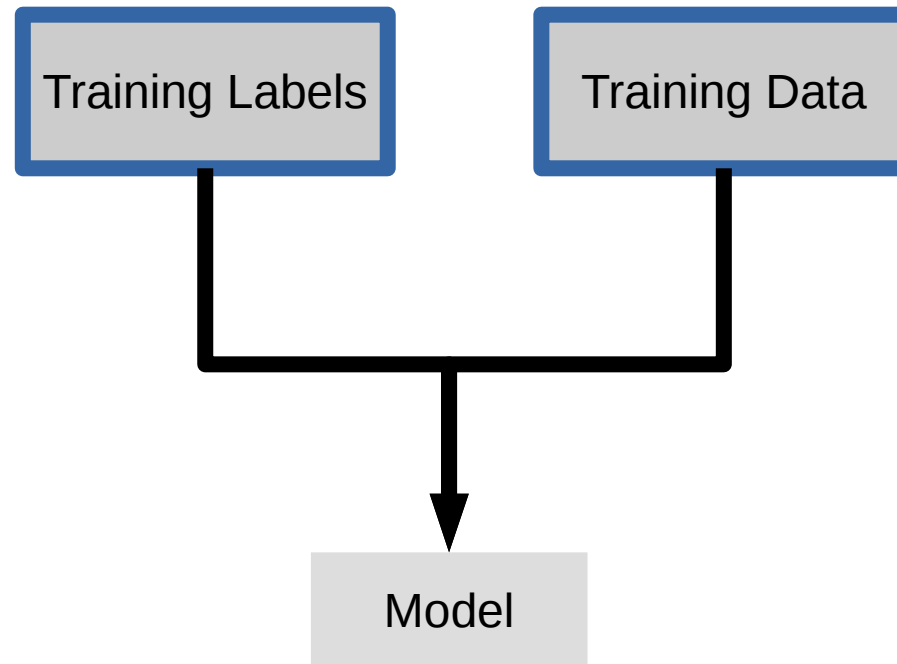
SVC(C=10,
gamma=0.001)

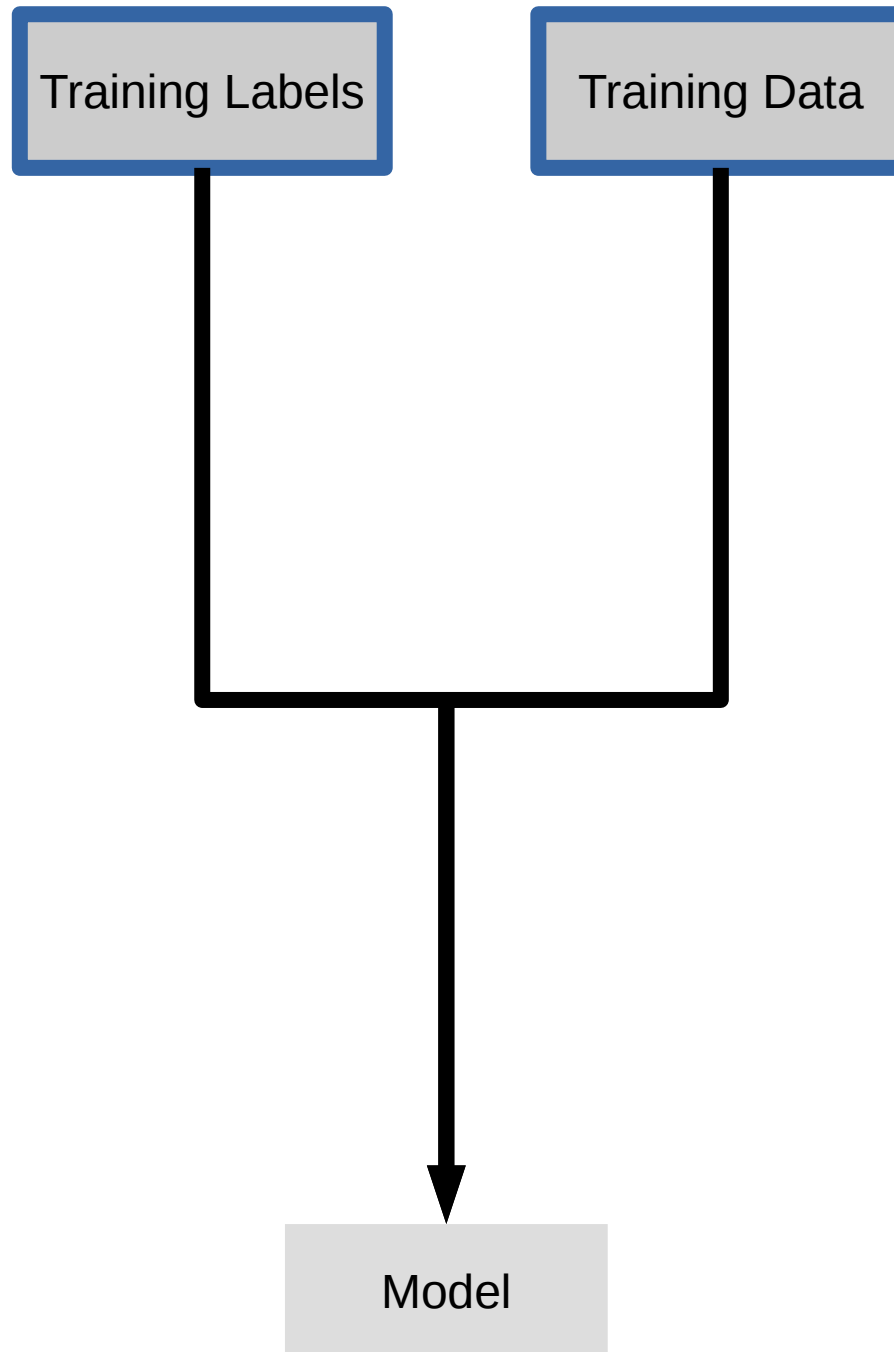
SVC(C=0.001, gamma=0.001)	SVC(C=0.01, gamma=0.001)	SVC(C=0.1, gamma=0.001)	SVC(C=1, gamma=0.001)	SVC(C=10, gamma=0.001)
SVC(C=0.001, gamma=0.01)	SVC(C=0.01, gamma=0.01)	SVC(C=0.1, gamma=0.01)	SVC(C=1, gamma=0.01)	SVC(C=10, gamma=0.01)

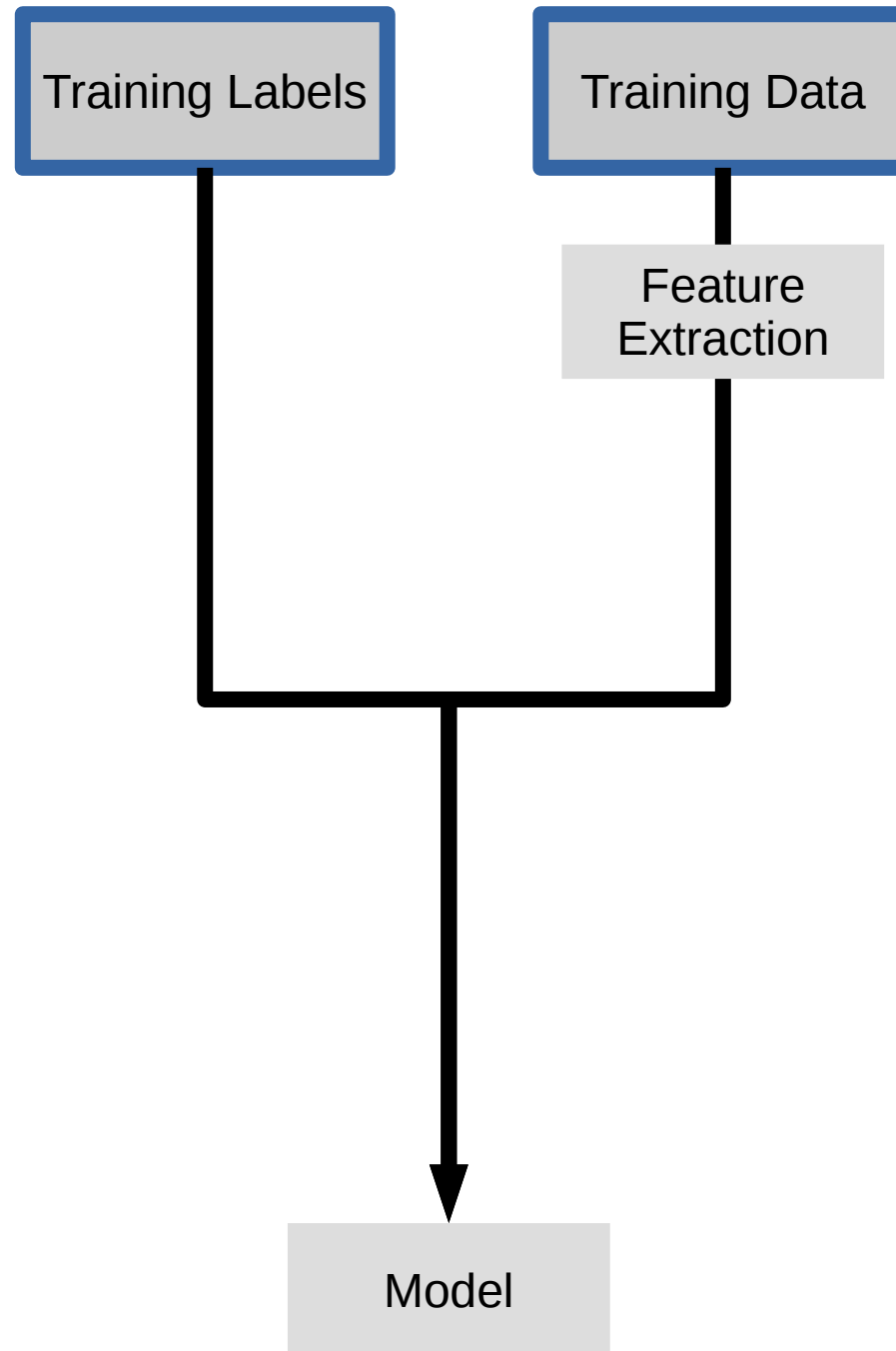
SVC(C=0.001, gamma=0.001)	SVC(C=0.01, gamma=0.001)	SVC(C=0.1, gamma=0.001)	SVC(C=1, gamma=0.001)	SVC(C=10, gamma=0.001)
SVC(C=0.001, gamma=0.01)	SVC(C=0.01, gamma=0.01)	SVC(C=0.1, gamma=0.01)	SVC(C=1, gamma=0.01)	SVC(C=10, gamma=0.01)
SVC(C=0.001, gamma=0.1)	SVC(C=0.01, gamma=0.1)	SVC(C=0.1, gamma=0.1)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

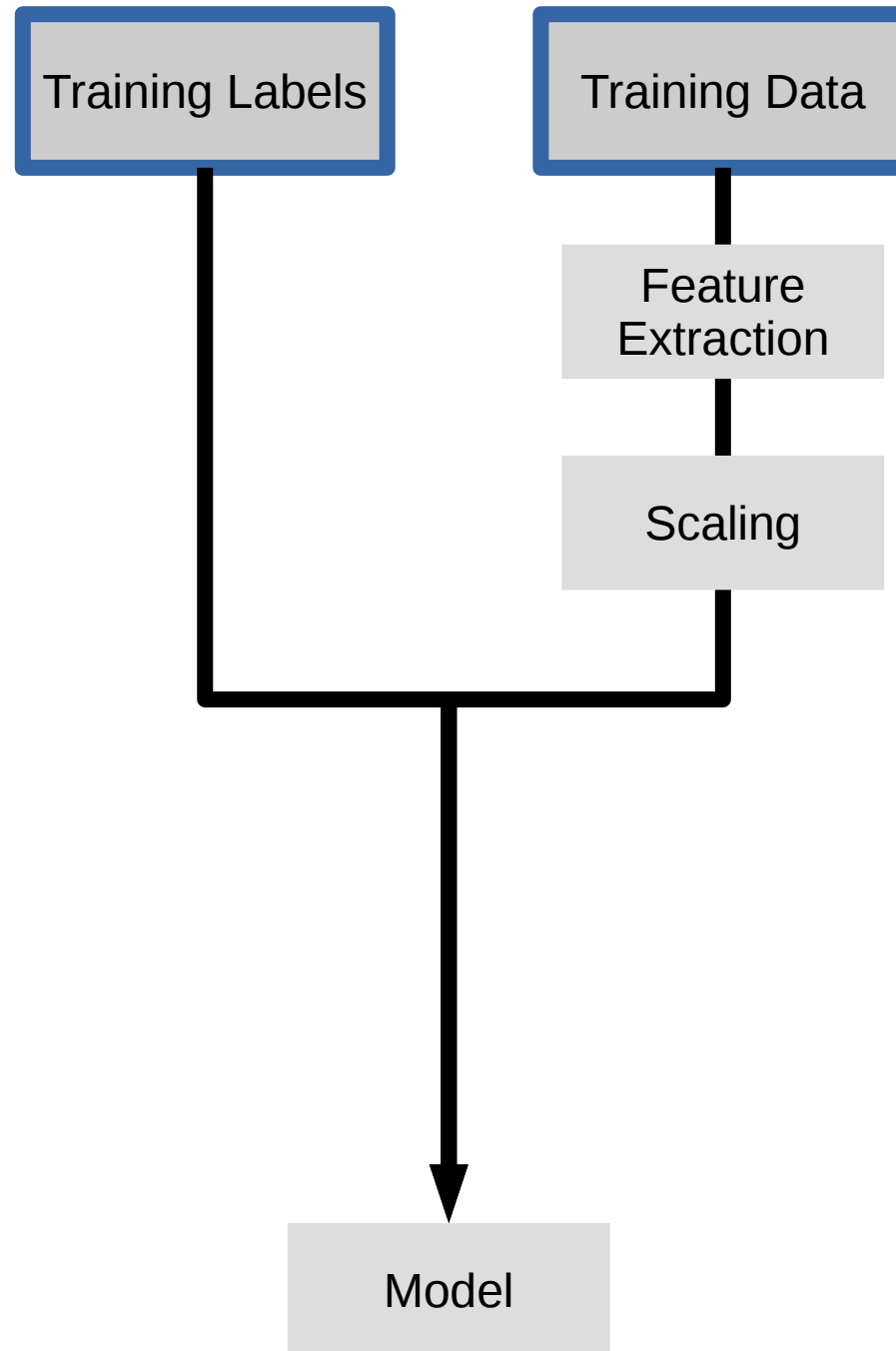
SVC(C=0.001, gamma=0.001)	SVC(C=0.01, gamma=0.001)	SVC(C=0.1, gamma=0.001)	SVC(C=1, gamma=0.001)	SVC(C=10, gamma=0.001)
SVC(C=0.001, gamma=0.01)	SVC(C=0.01, gamma=0.01)	SVC(C=0.1, gamma=0.01)	SVC(C=1, gamma=0.01)	SVC(C=10, gamma=0.01)
SVC(C=0.001, gamma=0.1)	SVC(C=0.01, gamma=0.1)	SVC(C=0.1, gamma=0.1)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

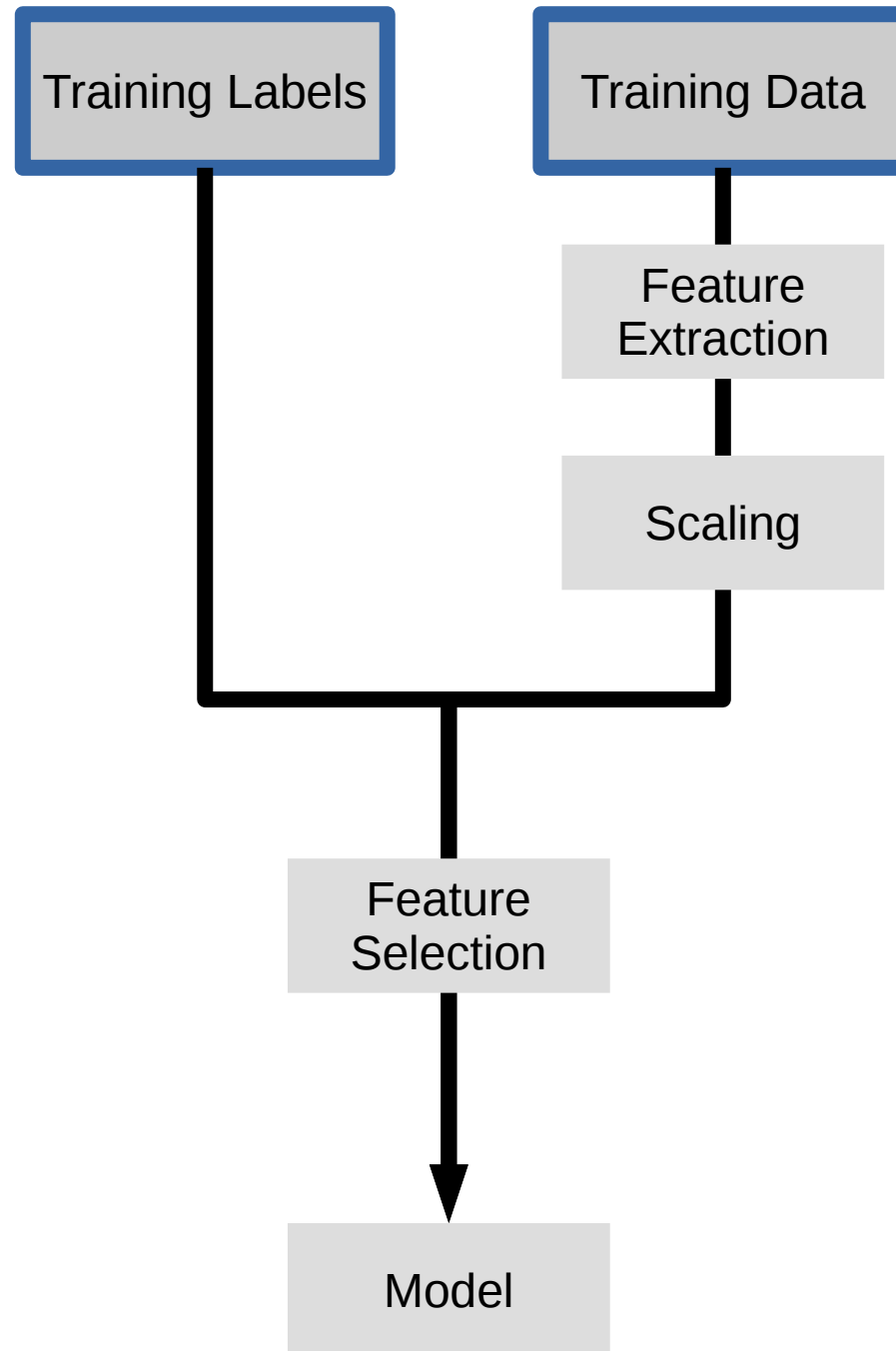
IPython Notebook: Part 4 – Grid Searches

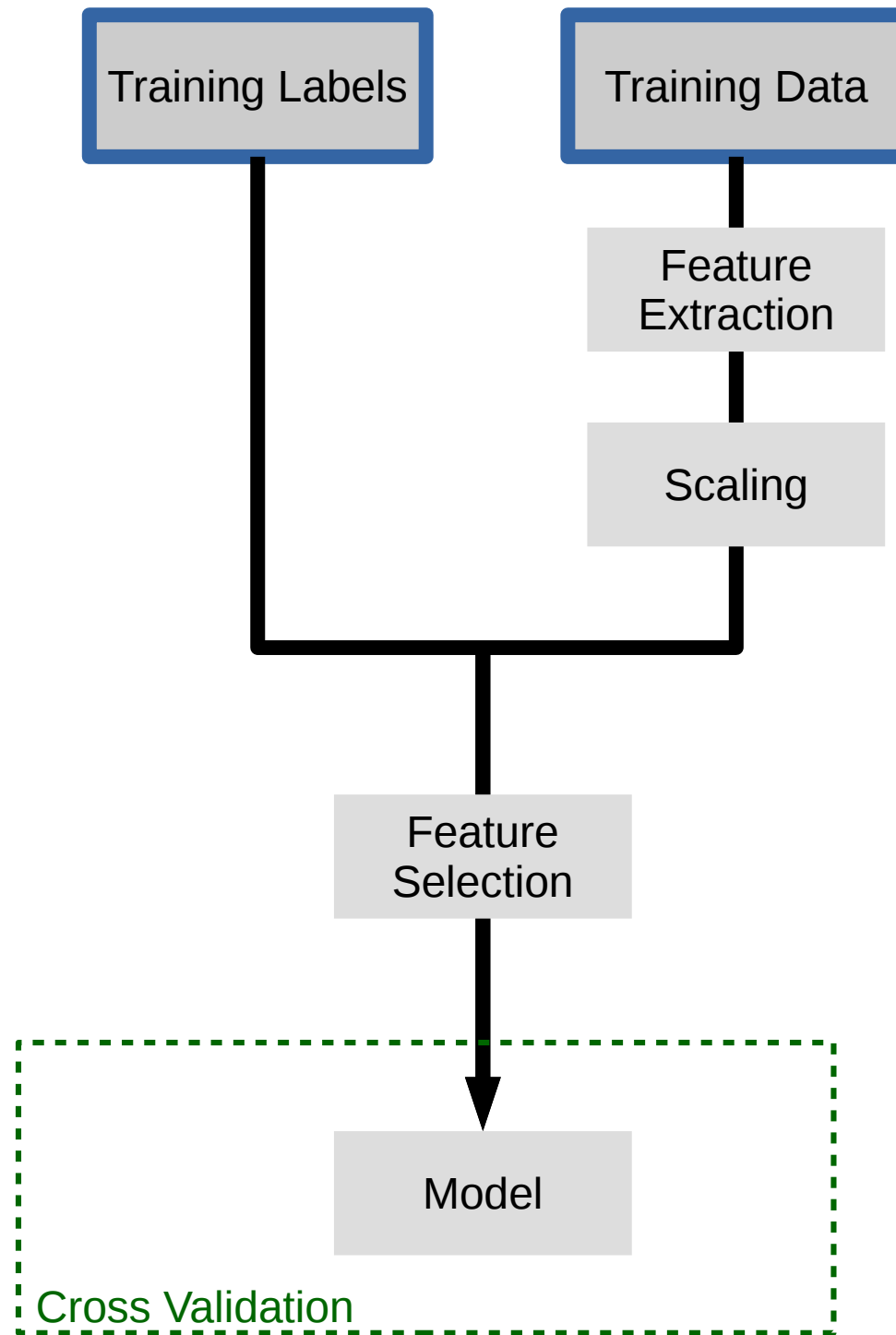


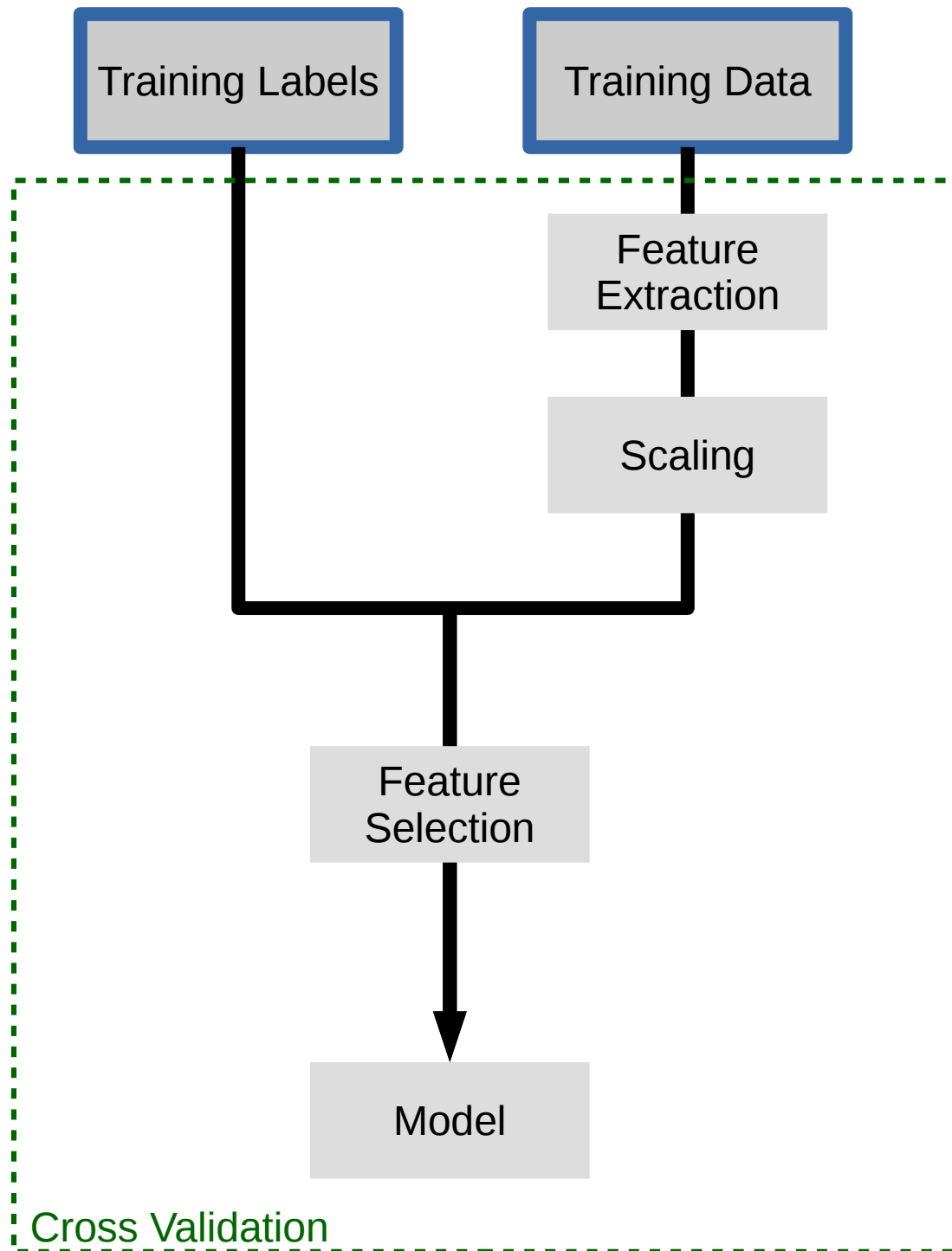












IPython Notebook:

Part 5 - Preprocessing and Pipelines

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

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

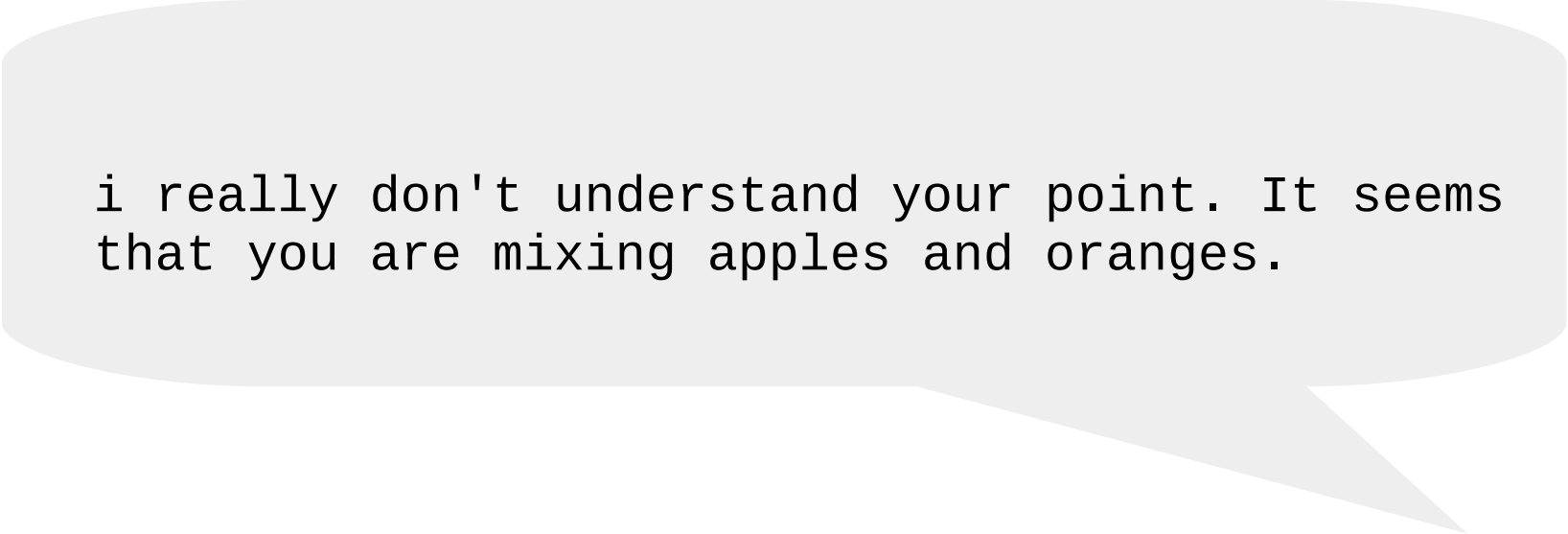
['this', 'is', 'how', 'you', 'get', 'ants']

Build a vocabulary
Sparse matrix encoding

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

Application: Insult detection

Application: Insult detection



i really don't understand your point. It seems that you are mixing apples and oranges.

Application: Insult detection

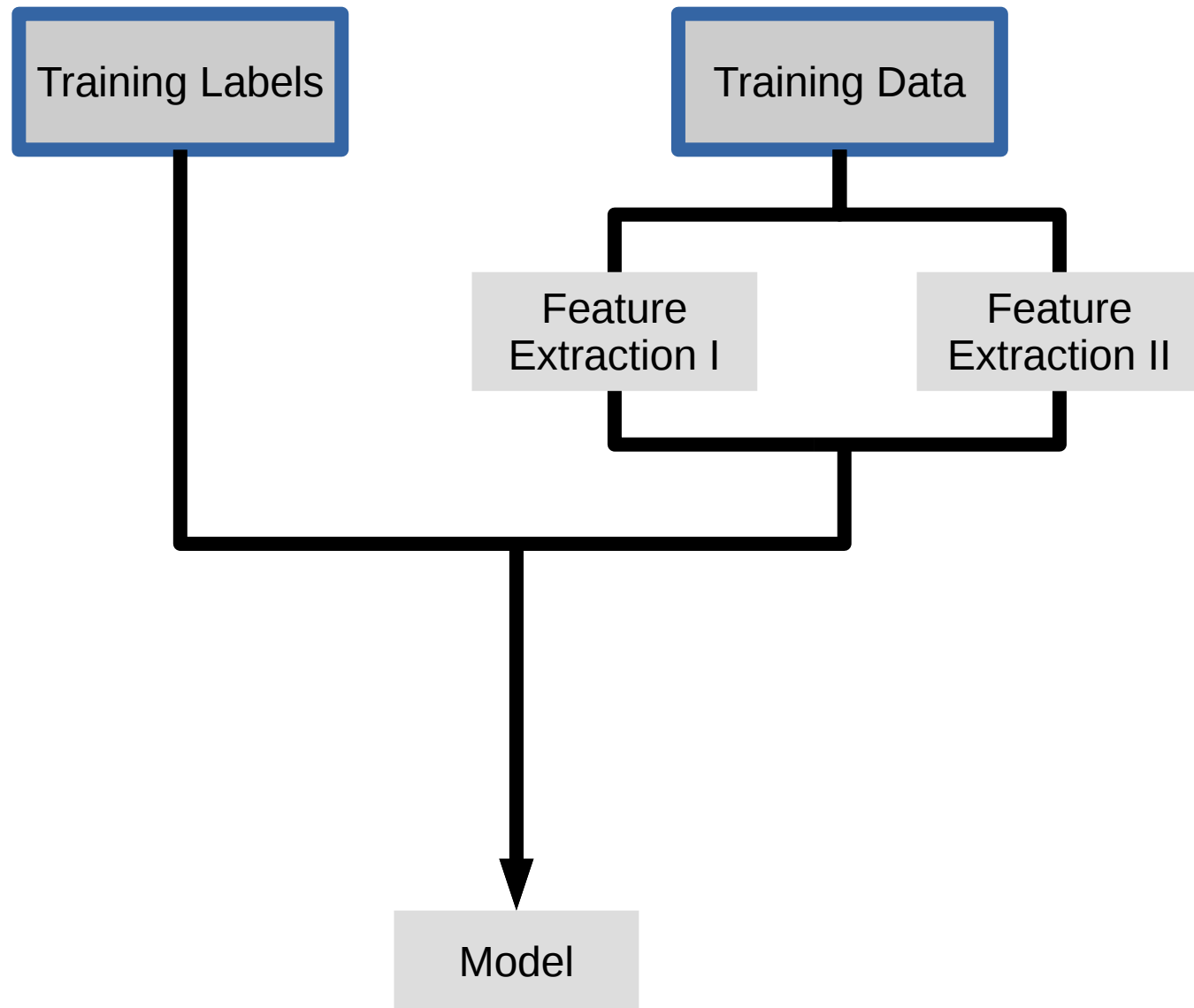
i really don't understand your point. It seems that you are mixing apples and oranges.

Clearly you're a fucktard.

IPython Notebook:

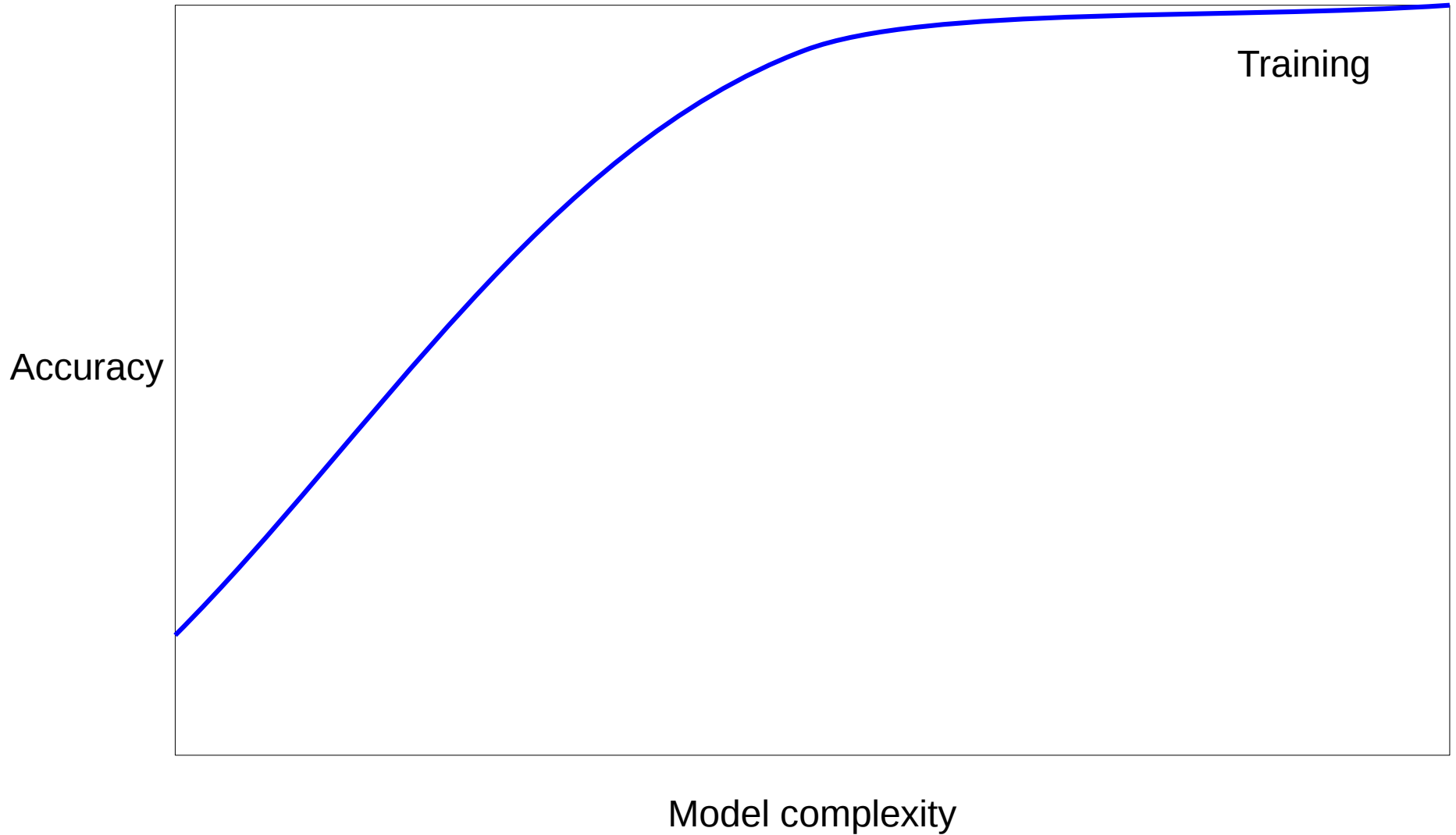
Part 6 - Working With Text Data

Feature Union

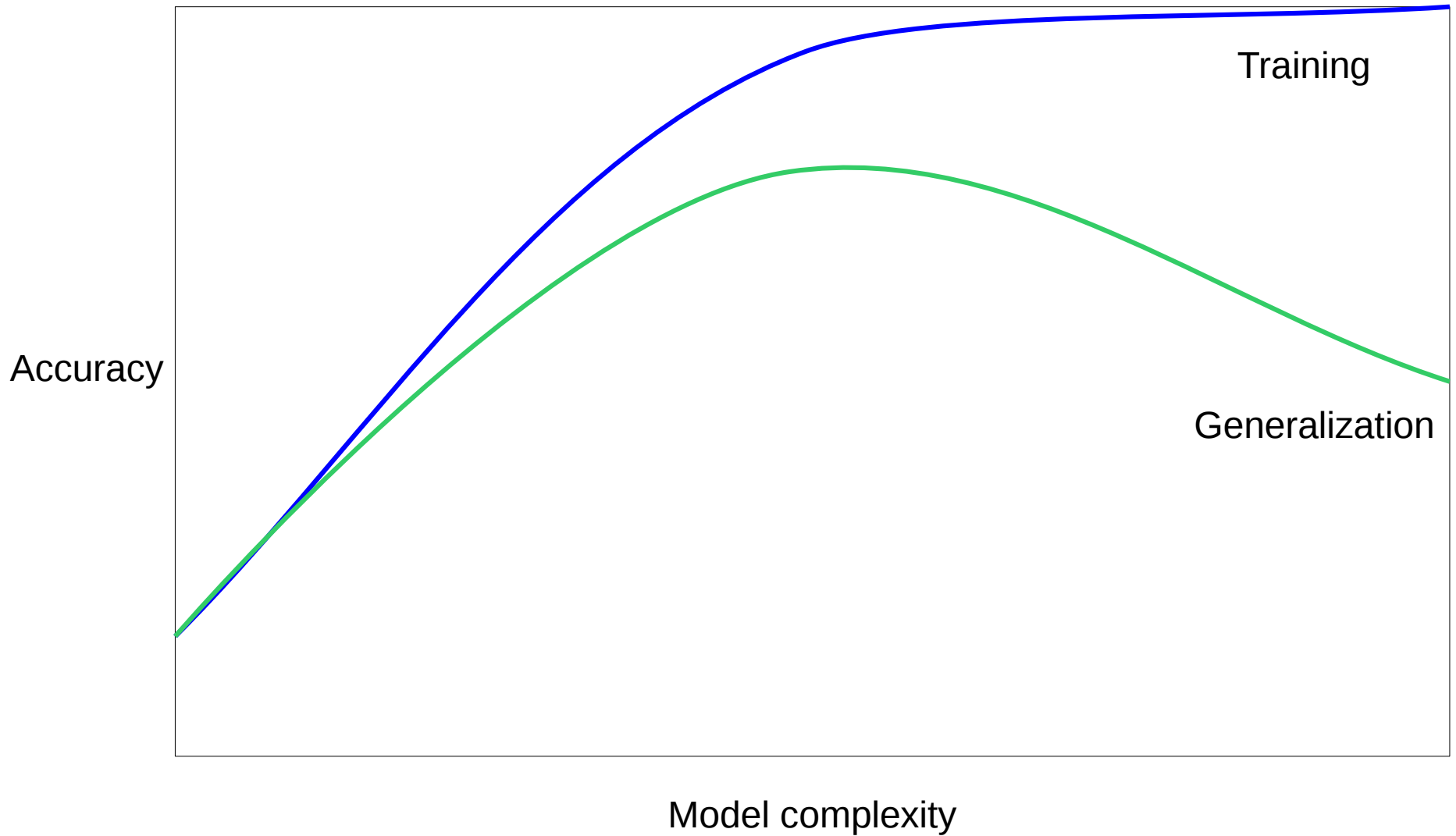


IPython Notebook: Part 7 – FeatureUnion

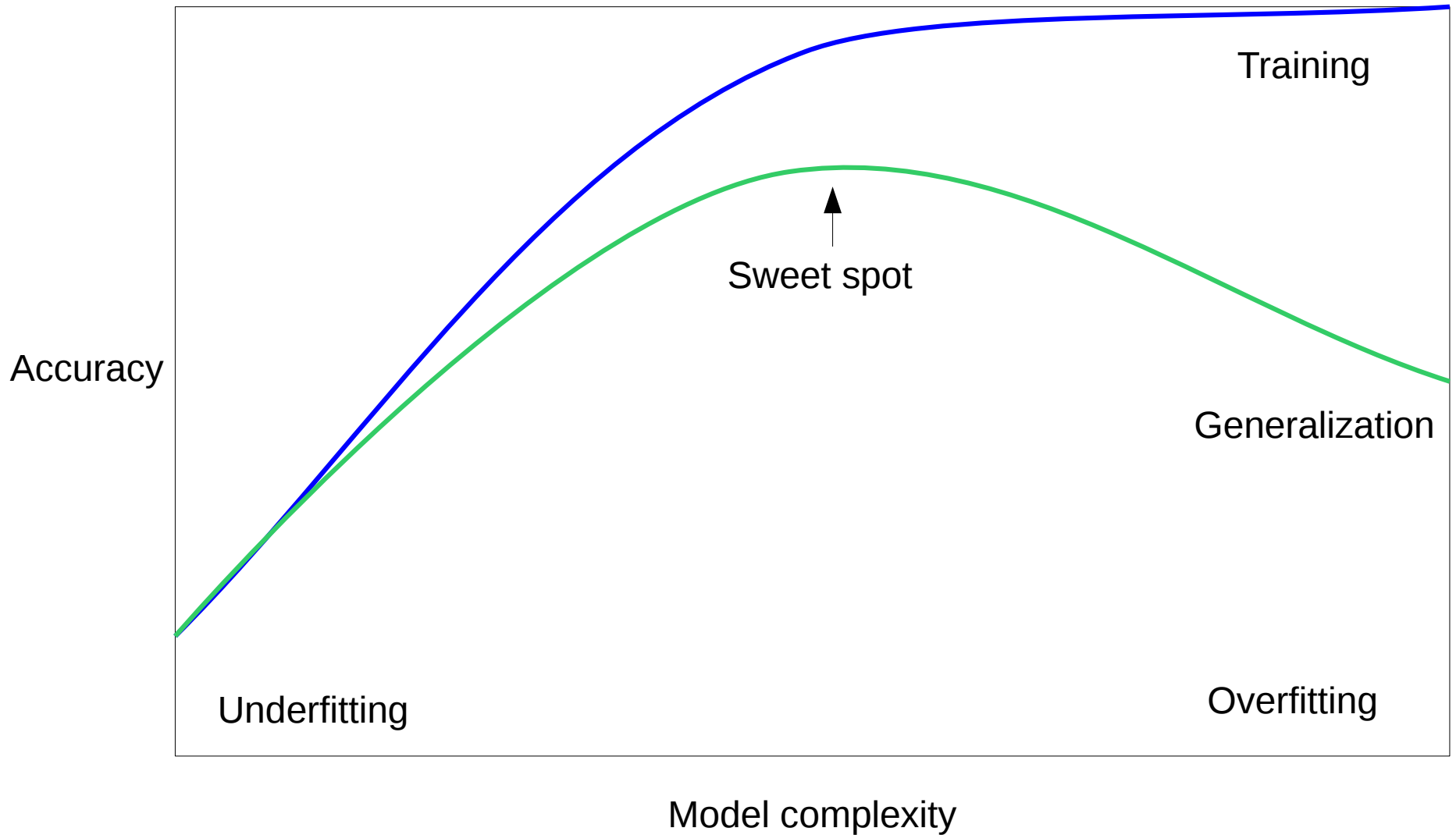
Overfitting and Underfitting



Overfitting and Underfitting



Overfitting and Underfitting

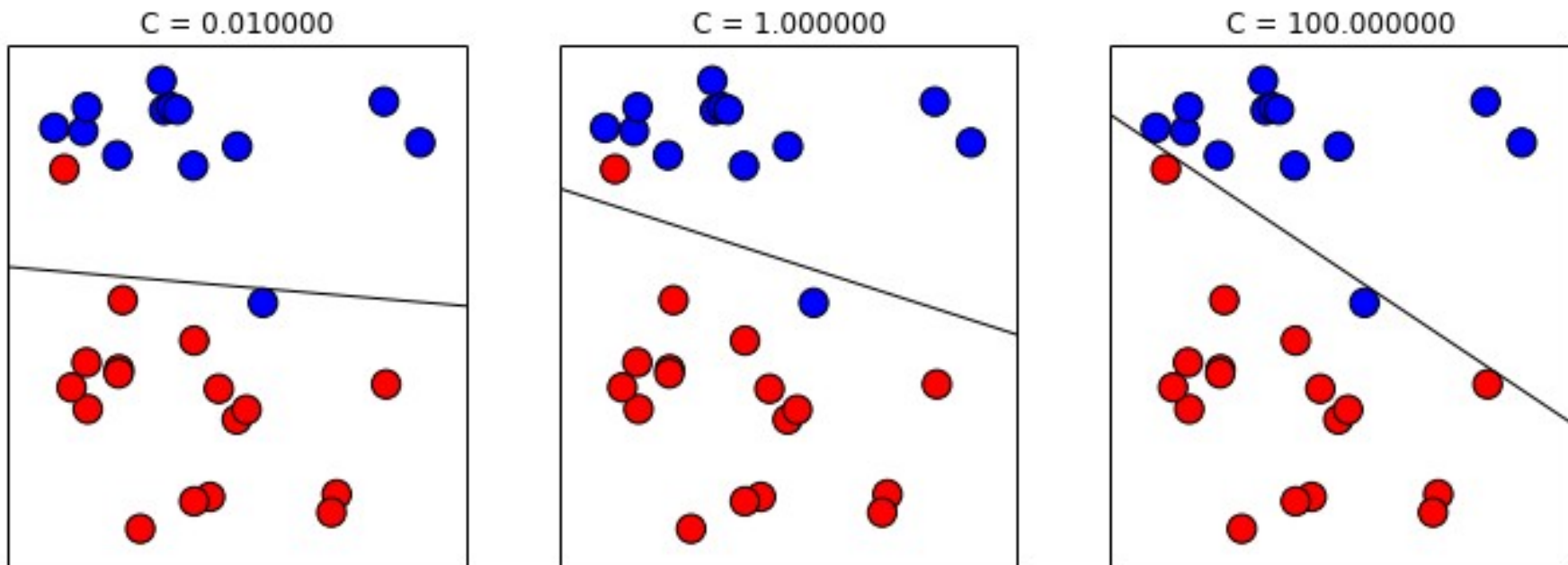


Linear SVM

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

Linear SVM

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



(RBF) Kernel SVM

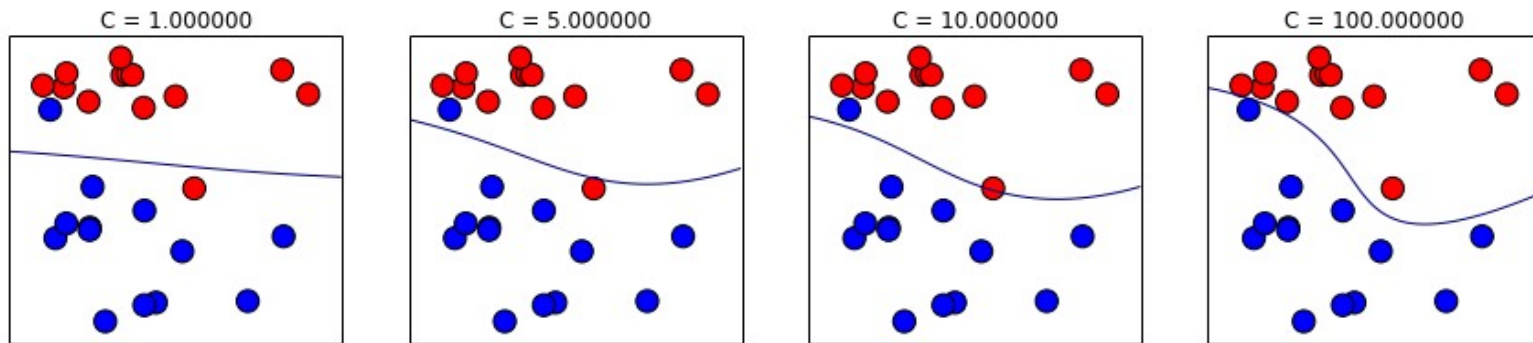
$$\hat{y} = \text{sign}(\alpha_0 + \sum_j \alpha_j y_j k(\mathbf{x}^{(j)}, \mathbf{x}))$$

(RBF) Kernel SVM

$$\hat{y} = \text{sign}(\alpha_0 + \sum_j \alpha_j y_j k(\mathbf{x}^{(j)}, \mathbf{x}))$$
$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$$

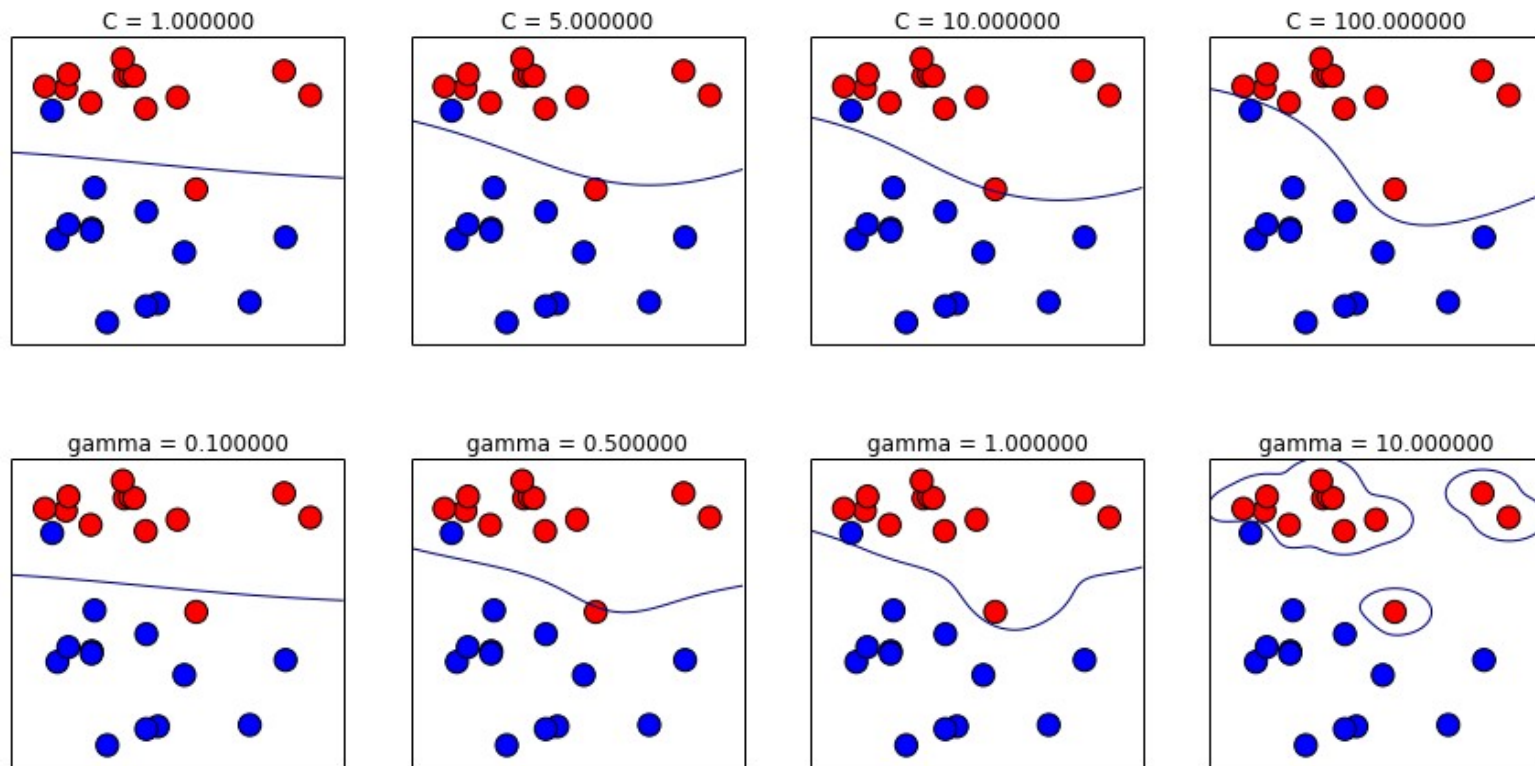
(RBF) Kernel SVM

$$\hat{y} = \text{sign}(\alpha_0 + \sum_j \alpha_j y_j k(\mathbf{x}^{(j)}, \mathbf{x}))$$
$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$$

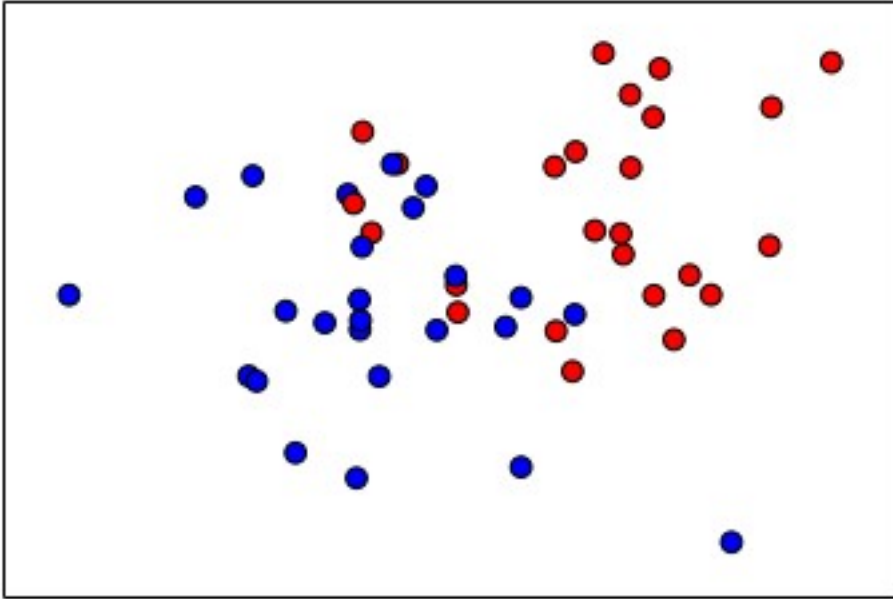


(RBF) Kernel SVM

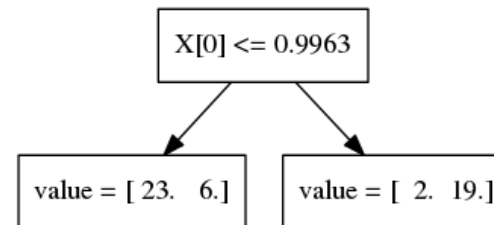
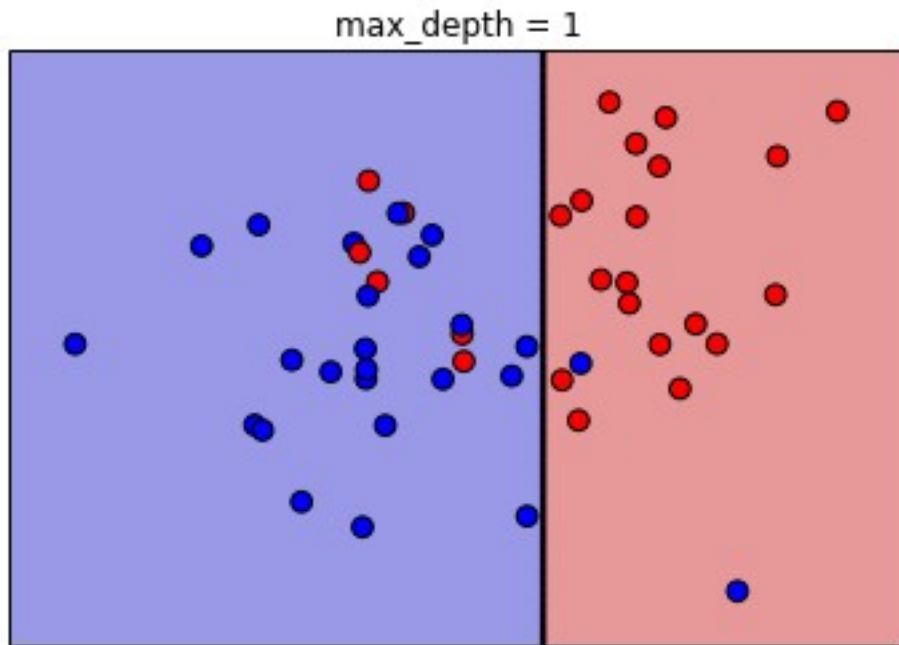
$$\hat{y} = \text{sign}(\alpha_0 + \sum_j \alpha_j y_j k(\mathbf{x}^{(j)}, \mathbf{x}))$$
$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$$



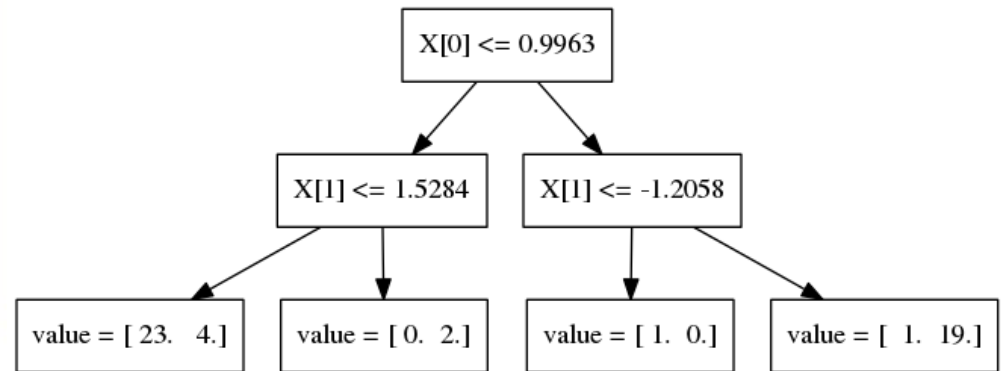
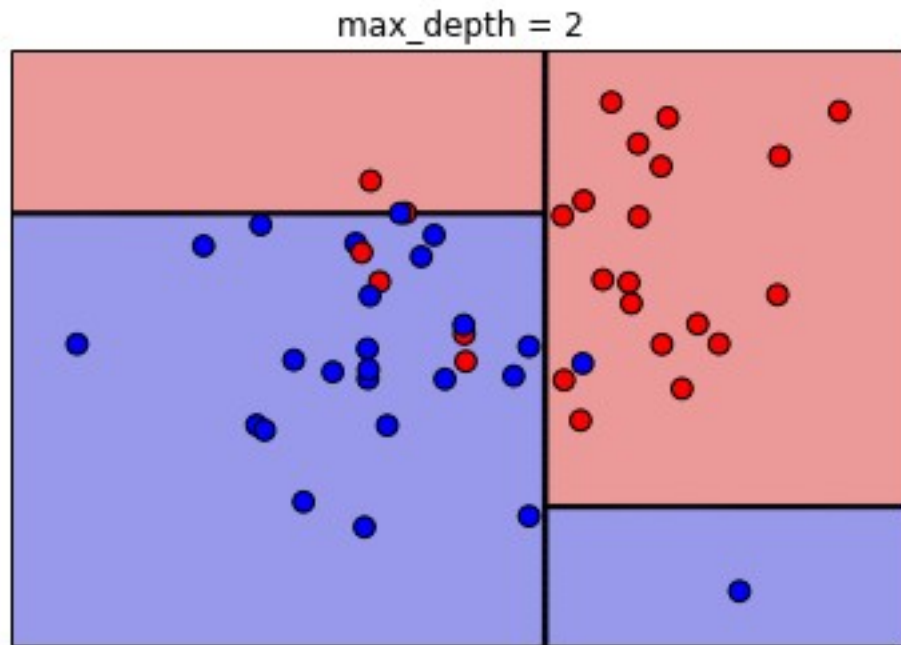
Decision Trees



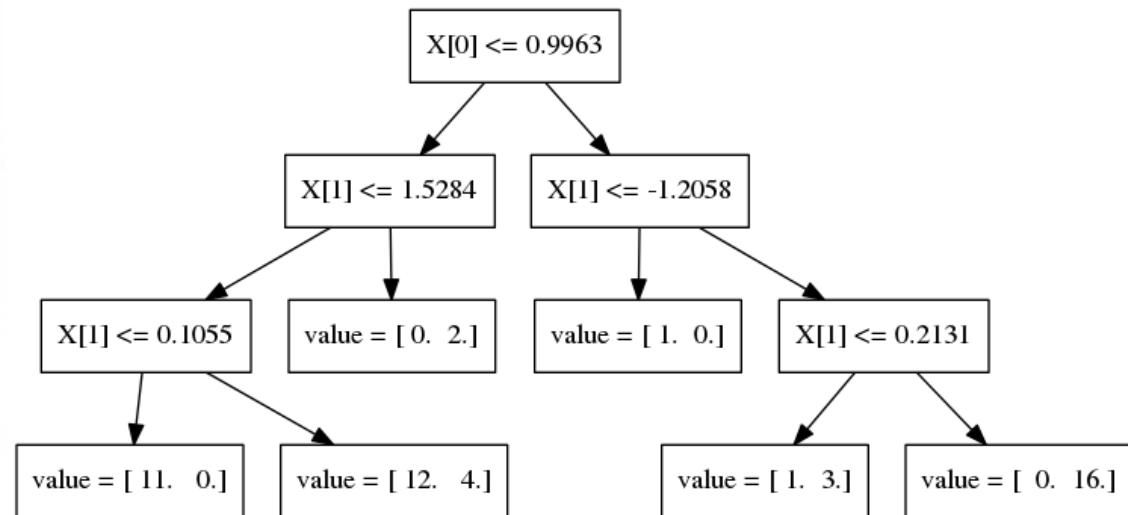
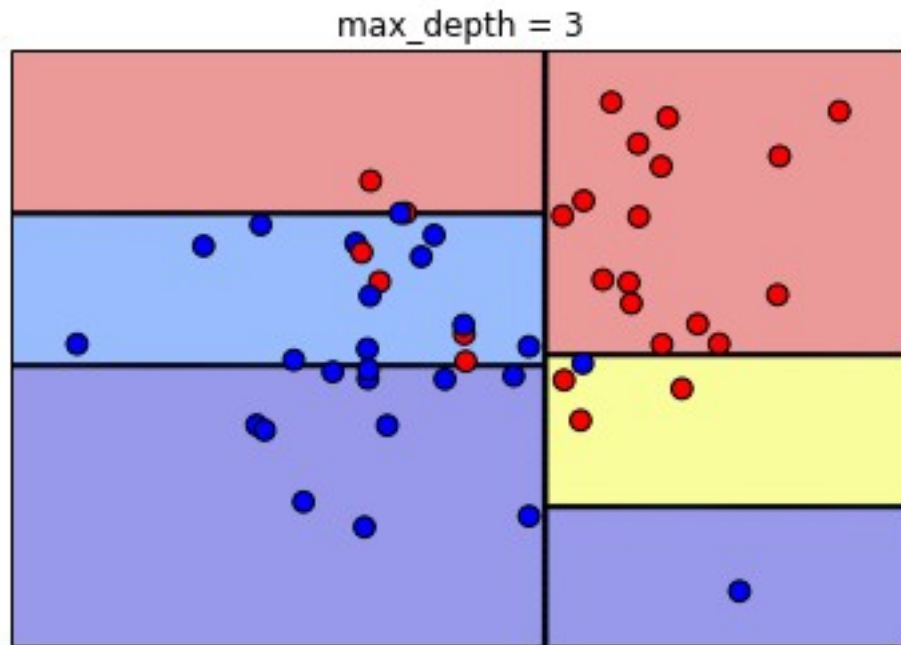
Decision Trees



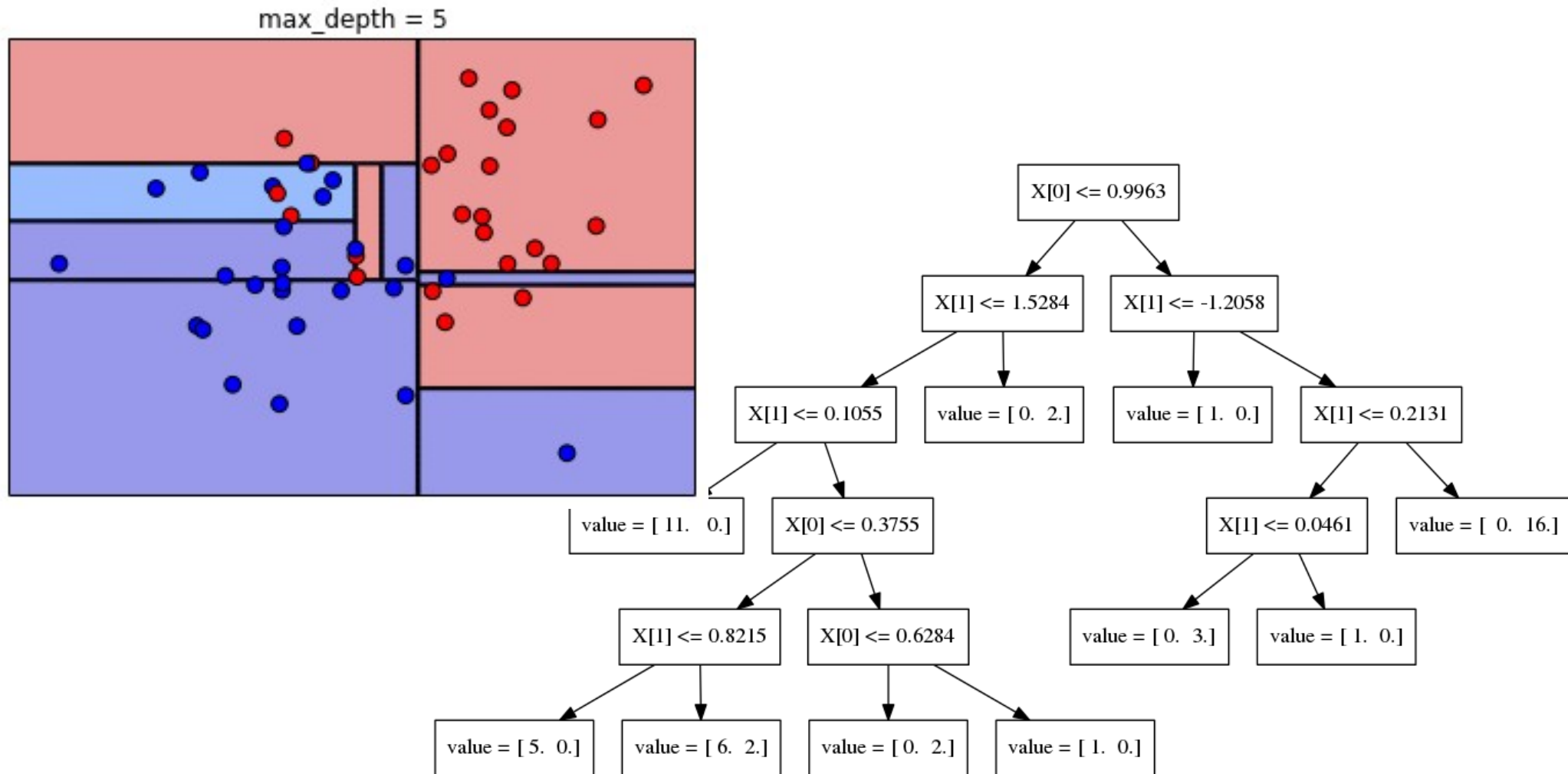
Decision Trees



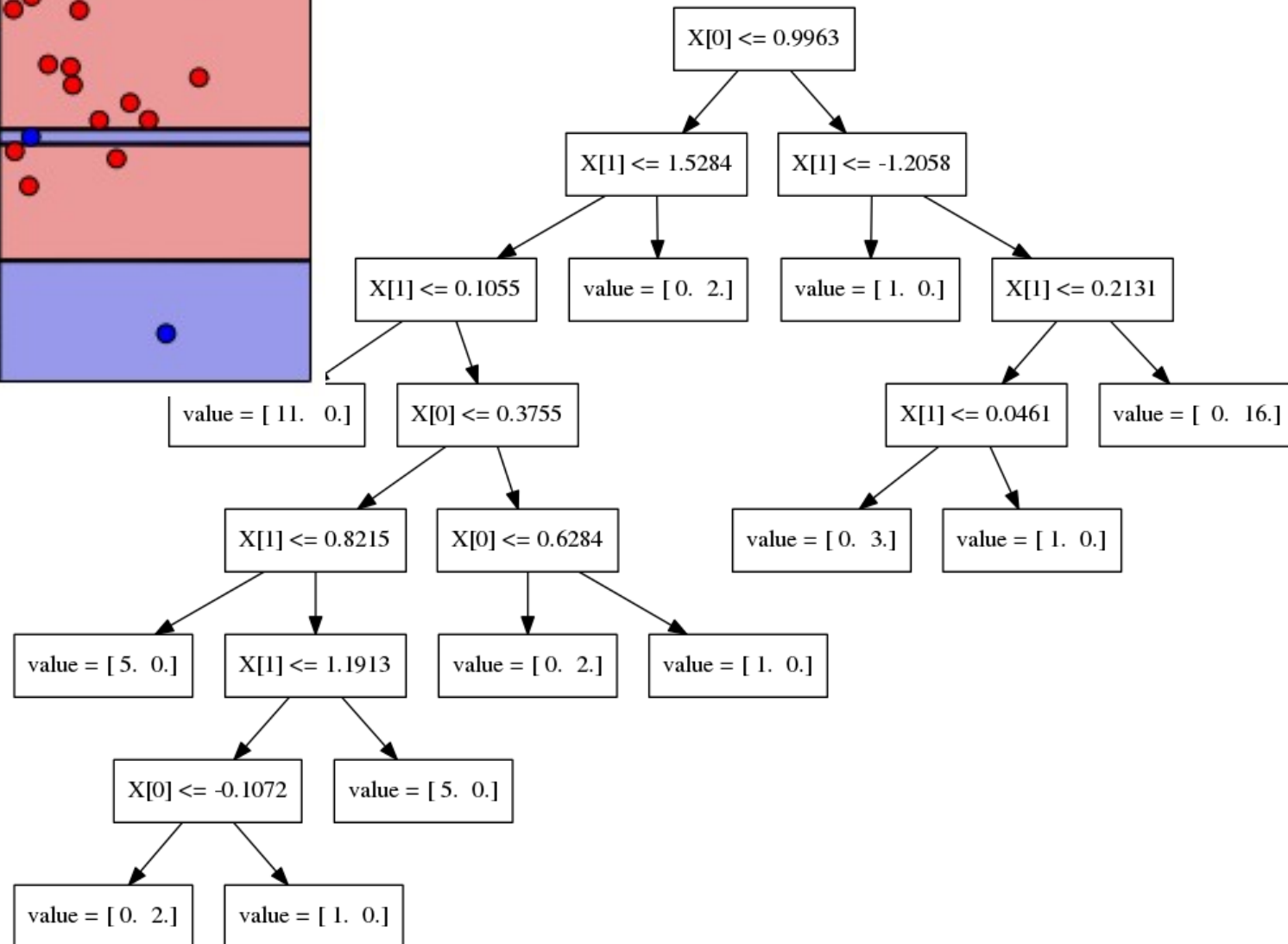
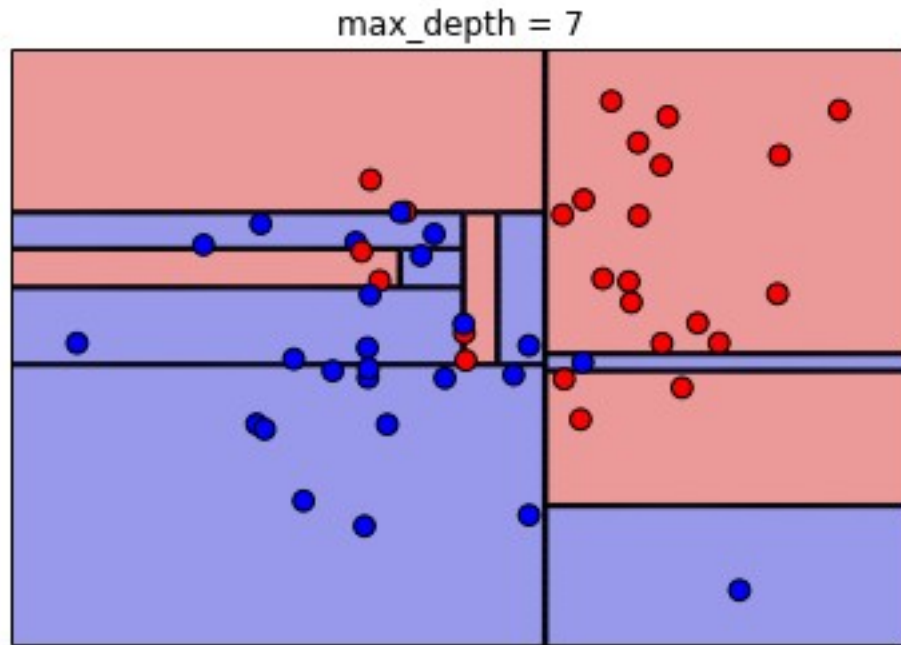
Decision Trees



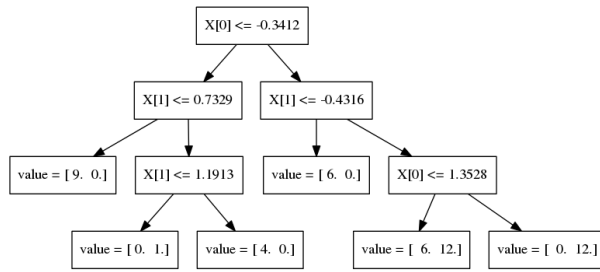
Decision Trees



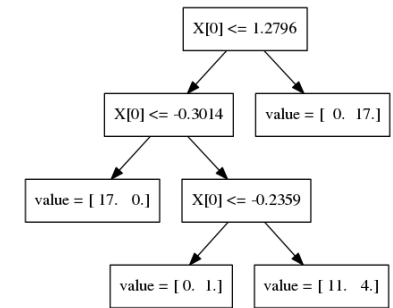
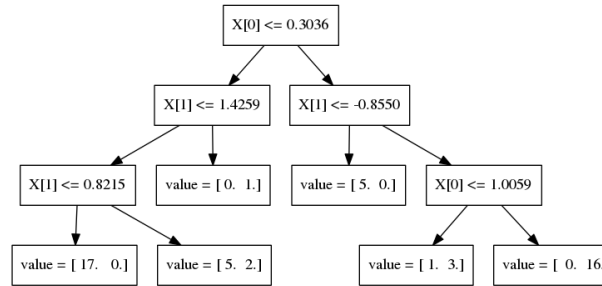
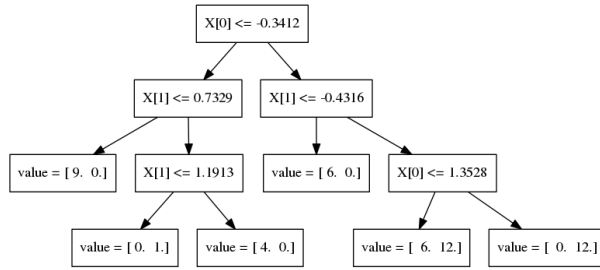
Decision Trees



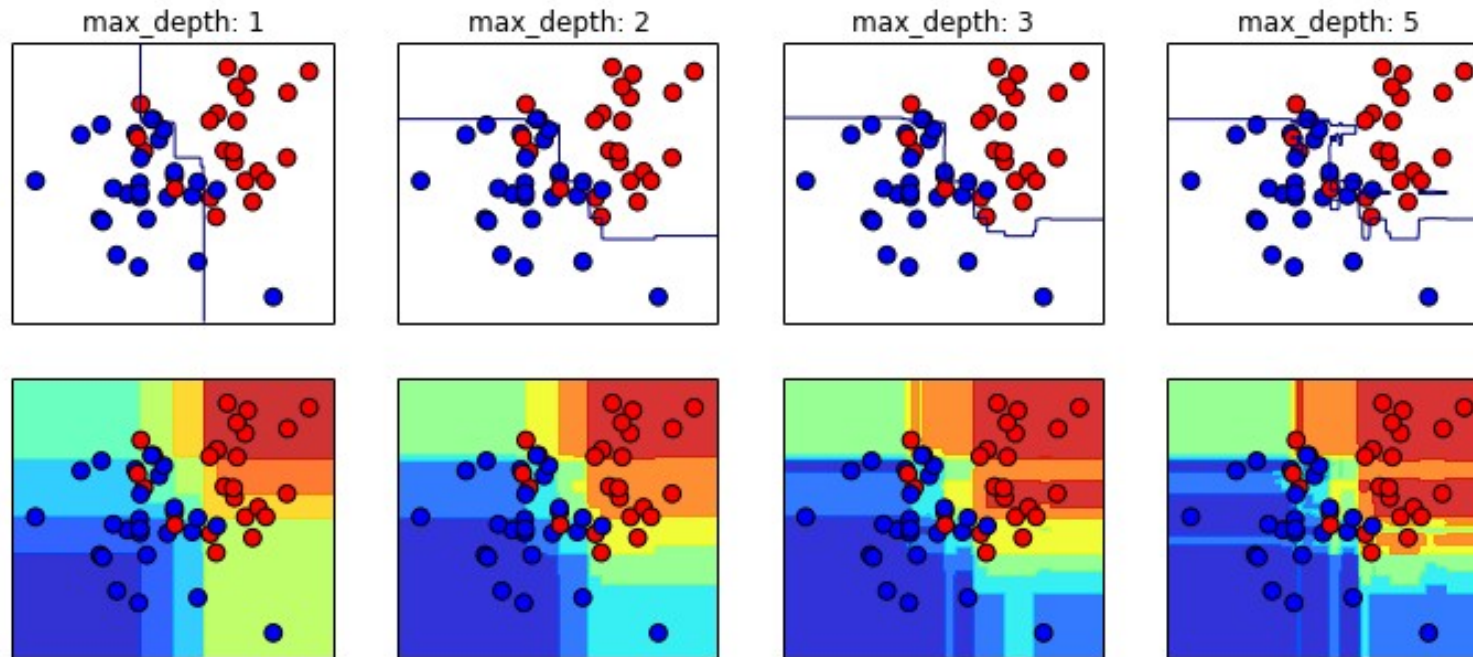
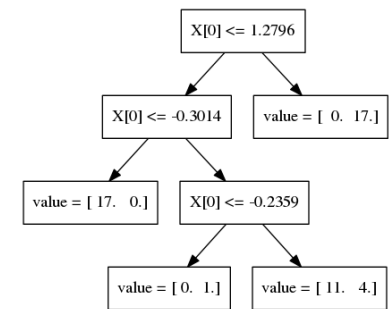
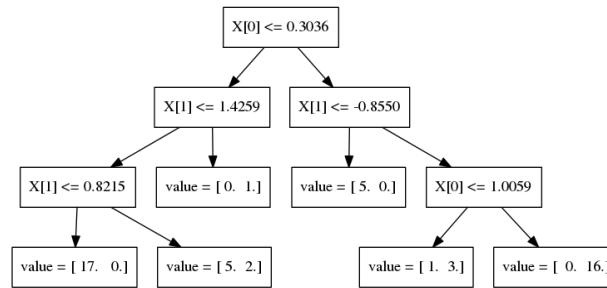
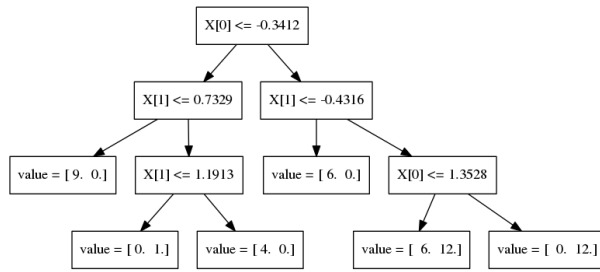
Random Forests



Random Forests

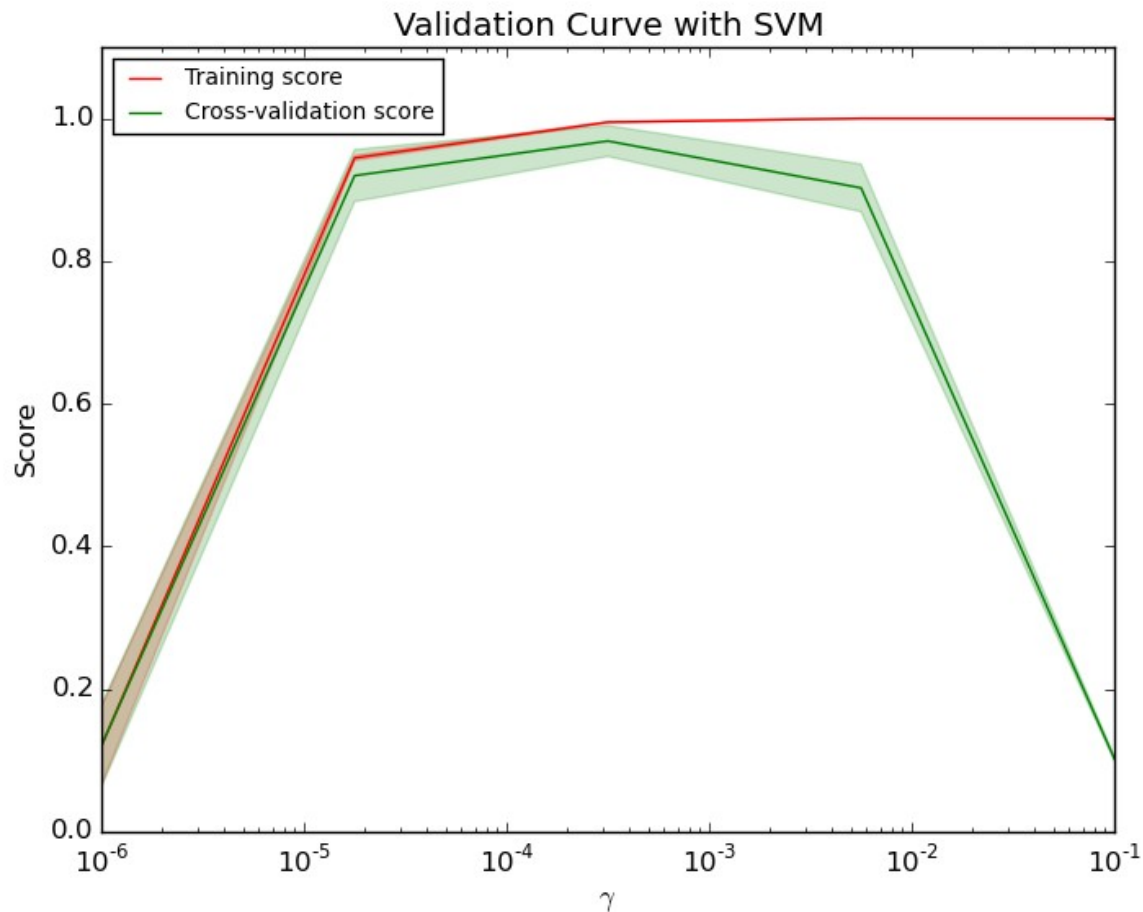


Random Forests



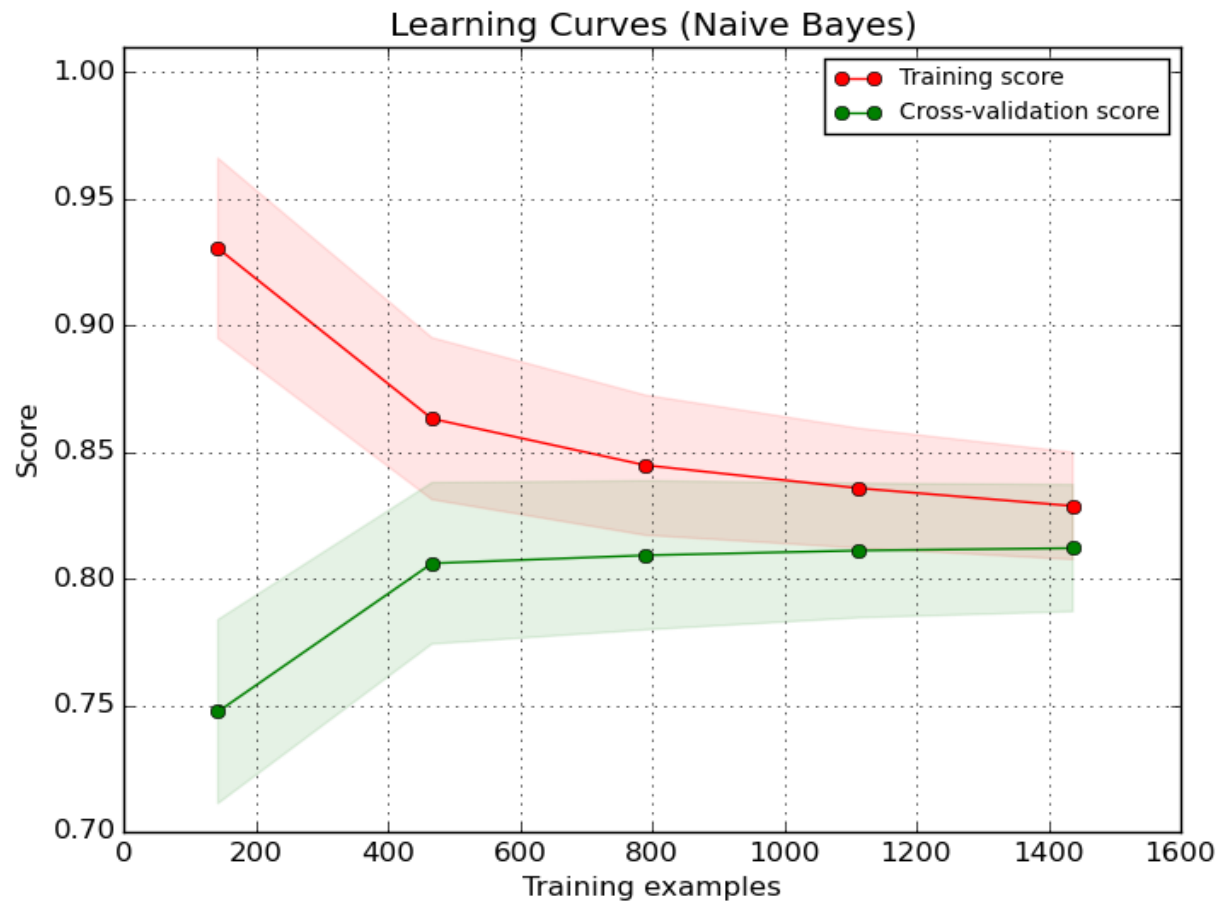
Validation Curves

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



Learning Curves

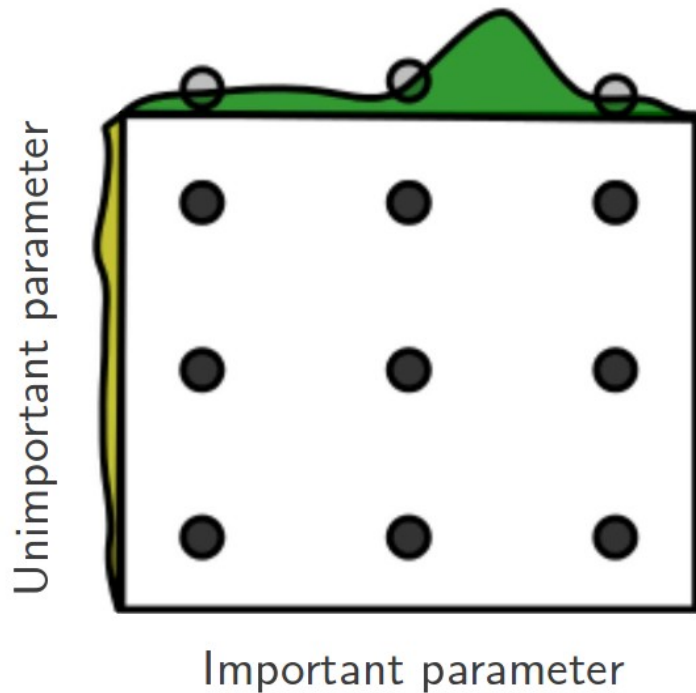
```
train_sizes, train_scores, test_scores = learning_curve(  
    estimator, X, y, train_sizes=train_sizes)
```



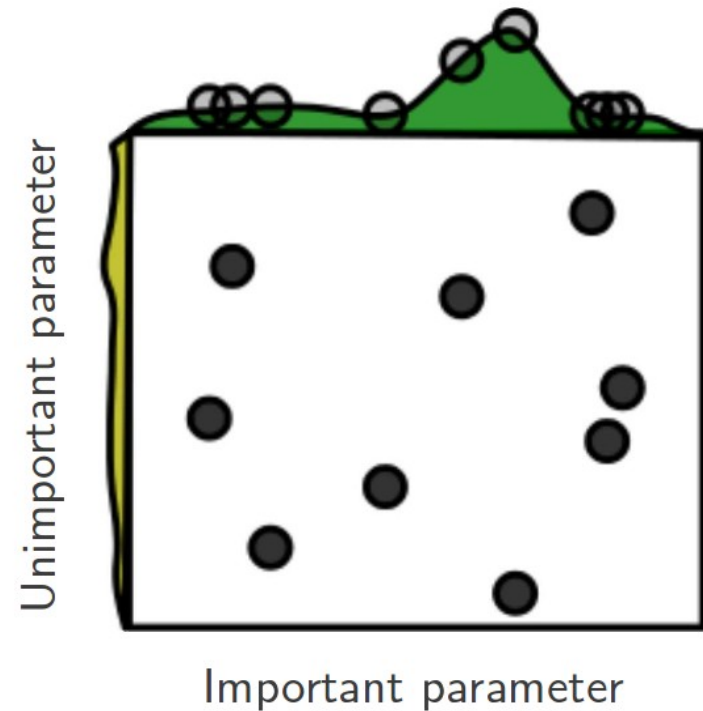
Randomized Parameter Search

Randomized Parameter Search

Grid Layout

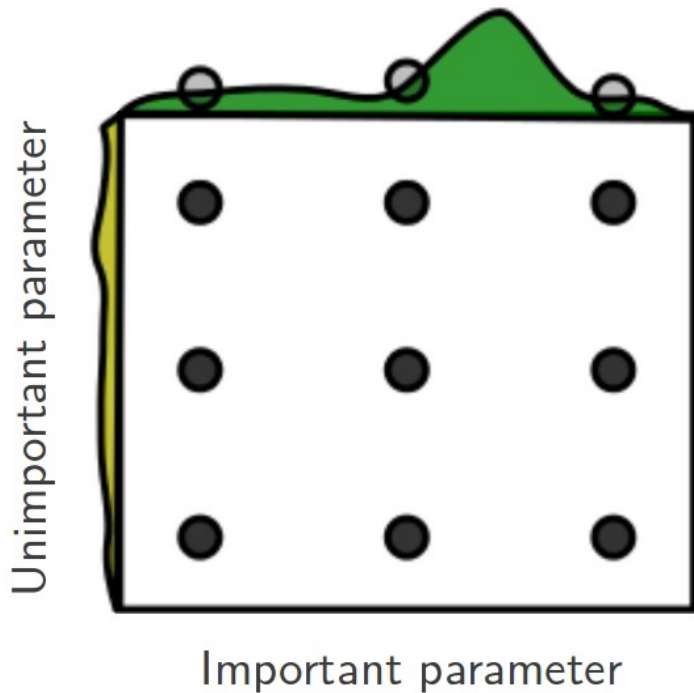


Random Layout

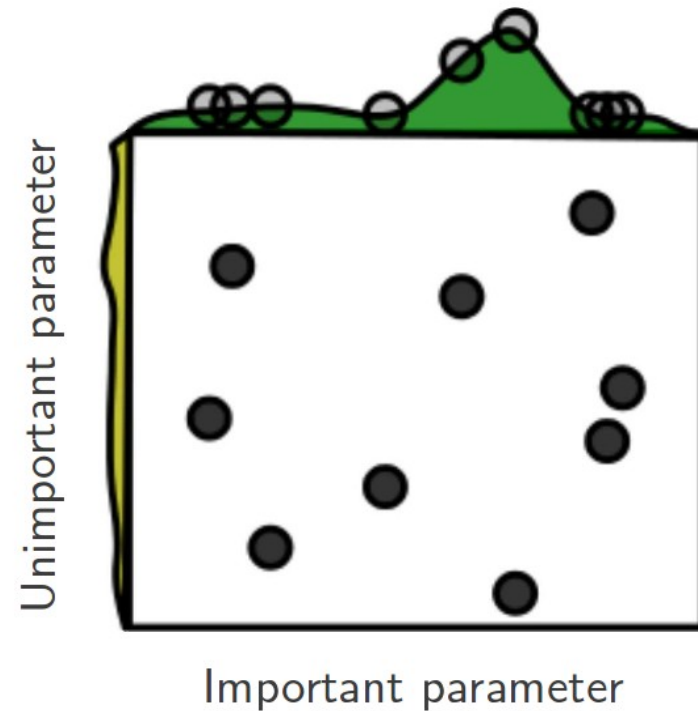


Randomized Parameter Search

Grid Layout



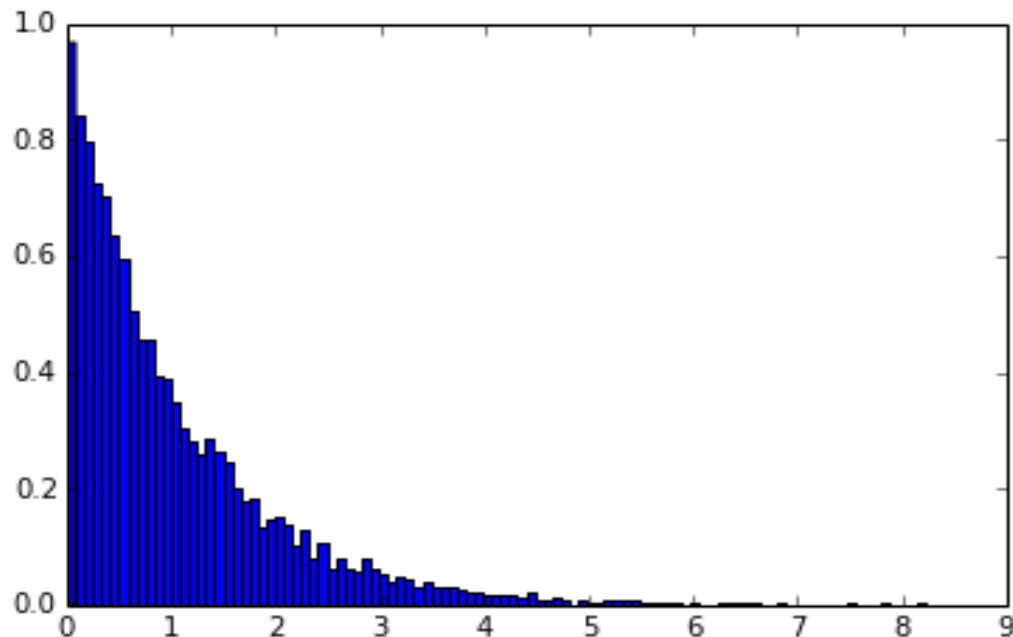
Random Layout



Step-size free for continuous parameters
Decouples runtime from search-space size
Robust against irrelevant parameters

Randomized Parameter Search

```
params = {'featureunion__countvectorizer-1__ngram_range':  
          [(1, 3), (1, 5), (2, 5)],  
          'featureunion__countvectorizer-2__ngram_range':  
          [(1, 1), (1, 2), (2, 2)],  
          'linearsvc__C': expon()}
```



```
rs = RandomizedSearchCV(text_pipe,  
                        param_distributions=param_distributins, n_iter=50)
```

Randomized Parameter Search

- Always use distributions for continuous variables.
- Don't use for low dimensional spaces.
- Future: Bayesian optimization based search.

IPython Notebook:

Part 8 – Randomized Parameter Search

Generalized Cross-Validation and Path Algorithms

IPython Notebook:

Part 9 – Generalized Cross-Validation

Linear Models	Feature Selection	Tree-Based models [possible]
LogisticRegressionCV [new]	RFECV	[DecisionTreeCV]
RidgeCV		[RandomForestClassifierCV]
RidgeClassifierCV		[GradientBoostingClassifierCV]
LarsCV		
ElasticNetCV		
...		

Scoring Functions

GridSeachCV
RandomizedSearchCV
cross_val_score
...CV

Default:
Accuracy (classification)
R2 (regression)

IPython Notebook:

Part 10 – Scoring Metrics

Scoring with imbalanced data

```
cross_val_score(SVC(), X_train, y_train)  
>>> array([ 0.9,  0.9,  0.9])
```


Scoring with imbalanced data

```
cross_val_score(SVC(), X_train, y_train)  
>>> array([ 0.9,  0.9,  0.9])
```

```
cross_val_score(DummyClassifier("most_frequent"), X_train, y_train)  
>>> array([ 0.9,  0.9,  0.9])
```

Scoring with imbalanced data

```
cross_val_score(SVC(), X_train, y_train)
>>> array([ 0.9,  0.9,  0.9])
```

```
cross_val_score(DummyClassifier("most_frequent"), X_train, y_train)
>>> array([ 0.9,  0.9,  0.9])
```

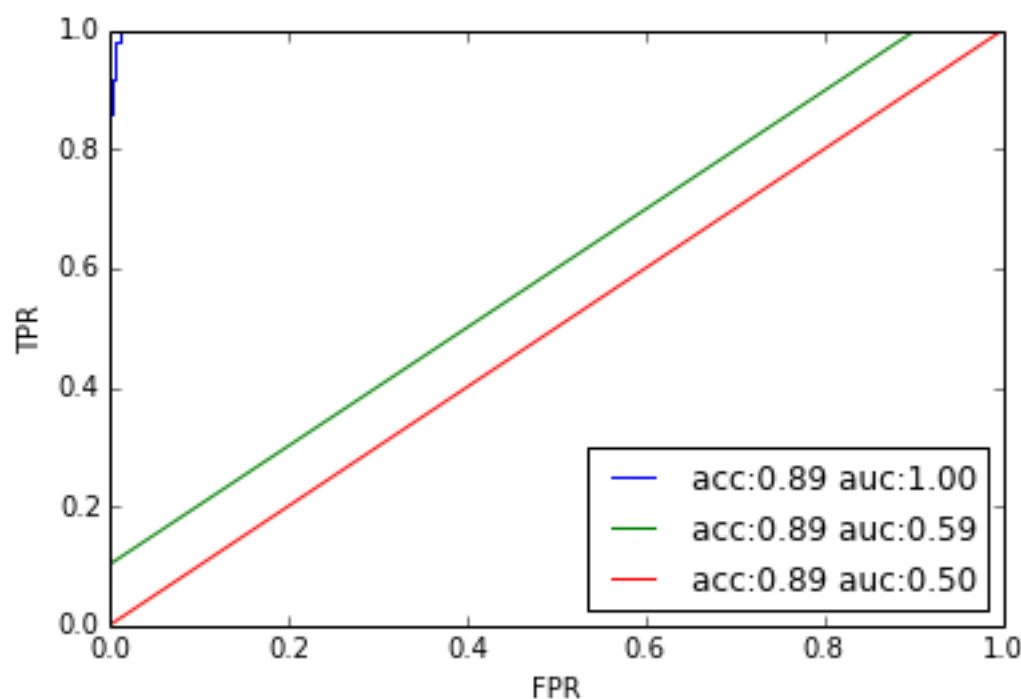
```
cross_val_score(SVC(), X_train, y_train, scoring="roc_auc")
array([ 0.99961591,  0.99983498,  0.99966247])
```

Scoring with imbalanced data

```
cross_val_score(SVC(), X_train, y_train)
>>> array([ 0.9,  0.9,  0.9])
```

```
cross_val_score(DummyClassifier("most_frequent"), X_train, y_train)
>>> array([ 0.9,  0.9,  0.9])
```

```
cross_val_score(SVC(), X_train, y_train, scoring="roc_auc")
array([ 0.99961591,  0.99983498,  0.99966247])
```



Available metrics

```
print(SCORERS.keys())  
  
>> ['adjusted_rand_score',  
     'f1',  
     'mean_absolute_error',  
     'r2',  
     'recall',  
     'median_absolute_error',  
     'precision',  
     'log_loss',  
     'mean_squared_error',  
     'roc_auc',  
     'average_precision',  
     'accuracy']
```

Defining your own scoring

```
def my_super_scoring(est, X, y):  
    return accuracy_scorer(est, X, y) - np.sum(est.coef_ != 0)
```

Out of Core Learning

Or: save ourself the effort

```
1 [|||||100.0%]
2 [|||||100.0%]
3 [|||||100.0%]
4 [|||||100.0%]
5 [|||||100.0%]
6 [|||||100.0%]
7 [|||||100.0%]
8 [|||||100.0%]
9 [|||||100.0%]
10 [|||||100.0%]
11 [|||||100.0%]
12 [|||||100.0%]
13 [|||||100.0%]
14 [|||||100.0%]
15 [|||||100.0%]
16 [|||||100.0%]
17 [|||||100.0%]
18 [|||||100.0%]
19 [|||||100.0%]
20 [|||||100.0%]
21 [|||||100.0%]
22 [|||||100.0%]
23 [|||||100.0%]
24 [|||||100.0%]
25 [|||||100.0%]
26 [|||||100.0%]
27 [|||||100.0%]
28 [|||||100.0%]
29 [|||||100.0%]
30 [|||||100.0%]
31 [|||||100.0%]
32 [|||||100.0%]
Mem[|||||23164/245759M]
Swp[|||||0/0MB]
```

Think twice!

- Old laptop: 4GB Ram
- 1073741824 float32
- Or 1mio data points with 1000 features
- EC2 : 256 GB Ram
- 68719476736 float32
- Or 68mio data points with 1000 features

Supported Algorithms

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

Out of Core Learning

```
sgd = SGDClassifier()

for i in range(9):
    X_batch, y_batch = cPickle.load(open("batch_%02d" % i))
    sgd.partial_fit(X_batch, y_batch, classes=range(10))
```

Possibly go over the data multiple times.

IPython Notebook:

Part 11 – Out Of Core Learning

Stateless Transformers

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

Bag Of Word Representations

CountVectorizer / TfidfVectorizer

"You better call Kenny Loggins"

tokenizer

['you', 'better', 'call', 'kenny', 'loggins']

Sparse matrix encoding

aardvak	better	call	you	zyxst
[0, ..., 0, 1, 0, ... , 0, 1 , 0, ..., 0, 1, 0,	0]			

Hashing Trick

HashingVectorizer

"You better call Kenny Loggins"

tokenizer

['you', 'better', 'call', 'kenny', 'loggins']

hashing

[hash('you'), hash('better'), hash('call'), hash('kenny'), hash('loggins')]
= [832412, 223788, 366226, 81185, 835749]

Sparse matrix encoding

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

IPython Notebook:

Part 12 – Out Of Core Learning for Text

Documentation of scikit-learn 0.16-git

Quick Start

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

Tutorials

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

Contributing

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

User Guide

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

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.

Flow Chart

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

Other Versions

- [scikit-learn 0.15 \(stable\)](#)
- [scikit-learn 0.16 \(development\)](#)
- [scikit-learn 0.14](#)
- [scikit-learn 0.13](#)
- [scikit-learn 0.12](#)
- Older versions 

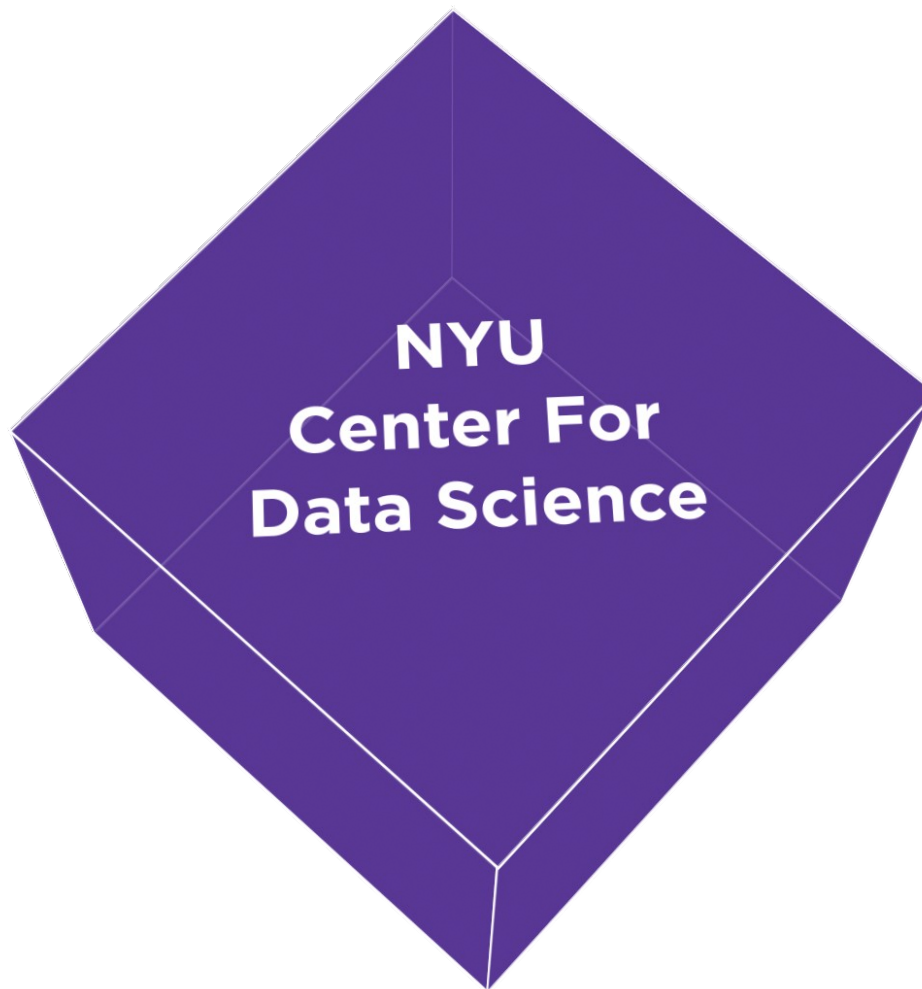
Additional Resources

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

FAQ

Frequently asked questions about the project and contributing.

CDS is hiring Research Engineers



Thank you for your attention.



@t3kcit



@amueller



importamueller@gmail.com



<http://amueller.github.io>