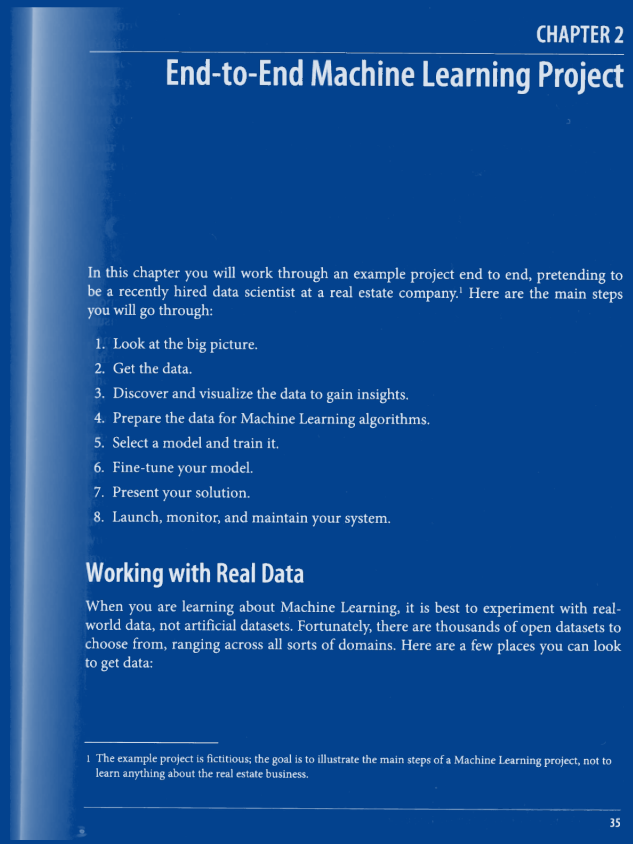
AARHUS UNIVERSITY

LESSON 3: End-to-end ML

CARSTEN EIE FRIGAARD

SPRING 2023



CHAPTER 2

End-to-End Machine Learning Project

In this chapter you will work through an example project end to end, pretending to be a recently hired data scientist at a real estate company.[1] Here are the main steps you will go through:

1. Look at the big picture.
2. Get the data.
3. Discover and visualize the data to gain insights.
4. Prepare the data for Machine Learning algorithms.
5. Select a model and train it.
6. Fine-tune your model.
7. Present your solution.
8. Launch, monitor, and maintain your system.

Working with Real Data

When you are learning about Machine Learning, it is best to experiment with real-world data, not artificial datasets. Fortunately, there are thousands of open datasets to choose from, ranging across all sorts of domains. Here are a few places you can look to get data:

1 The example project is fictitious; the goal is to illustrate the main steps of a Machine Learning project, not to learn anything about the real estate business.

35

Agenda

End-to-end Machine Learning

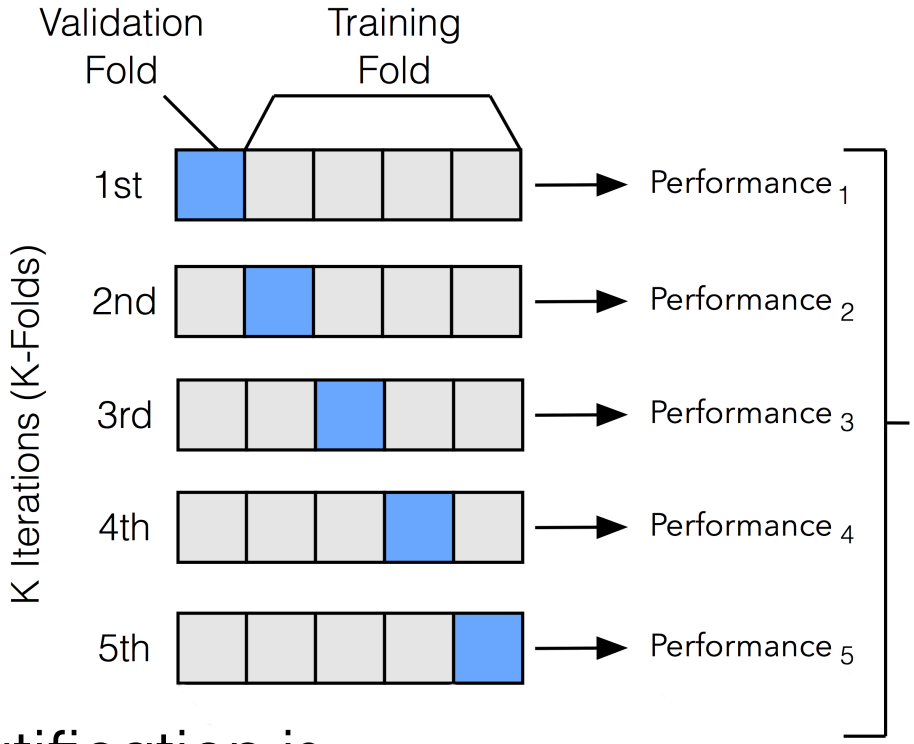
1. Admin
2. Algorithm and Model Selection,
   - model hyperparameters
   - k-fold cross validation,
   - ekstra materiale: `L03/Extra/k-fold_demo.ipynb`
3. Generel repetition af § 2,
   - Opgave: `L03/supergruppe_diskussion.ipynb`
4. End-to-end repetition,
   - via 'The Map'

ALGORITHM SELECTION AND MODEL SELECTION

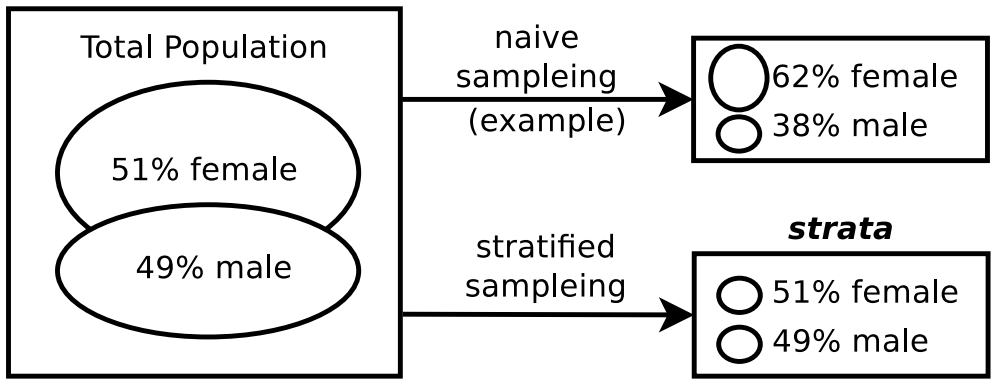
And k-fold CV

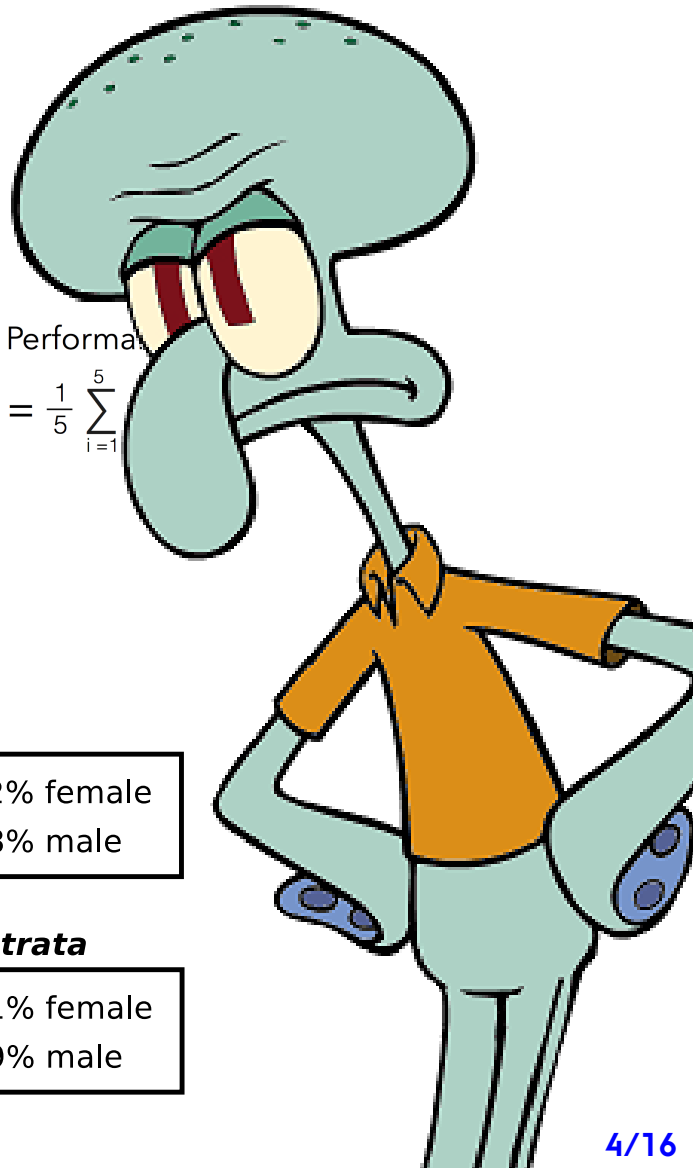
Forbered data: cross-validation, stratification

K-fold cross-validation (CV)...



and stratification is...

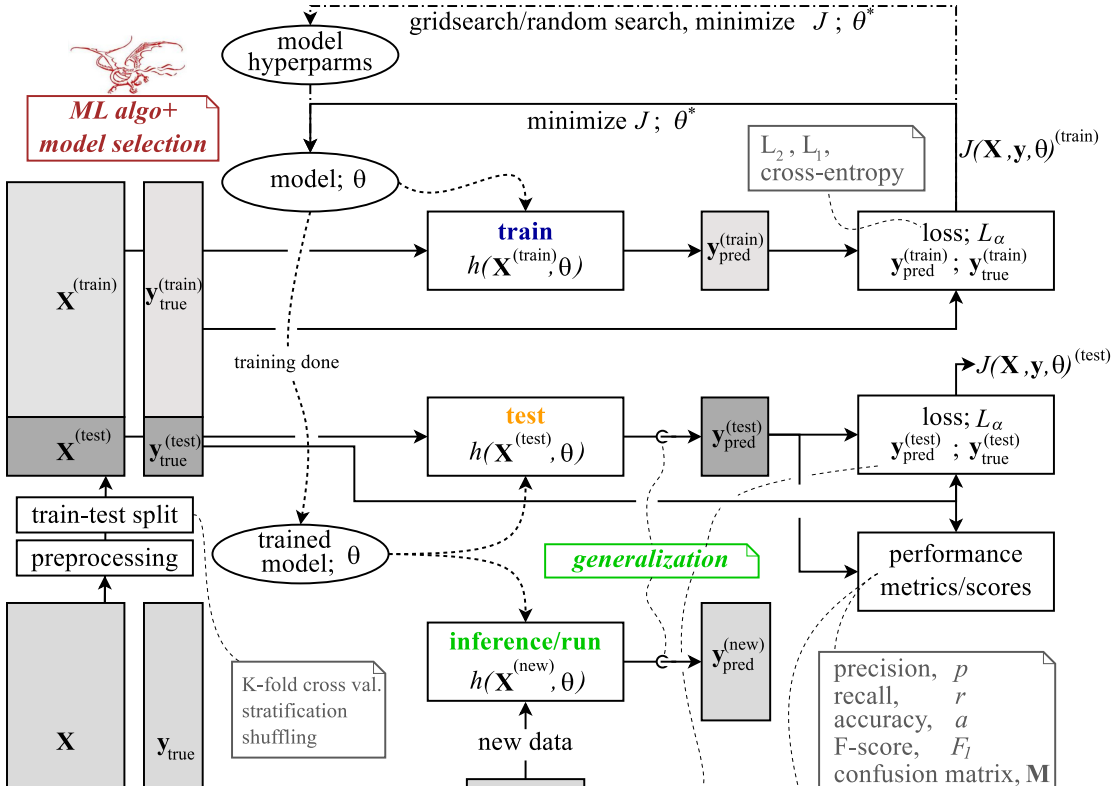


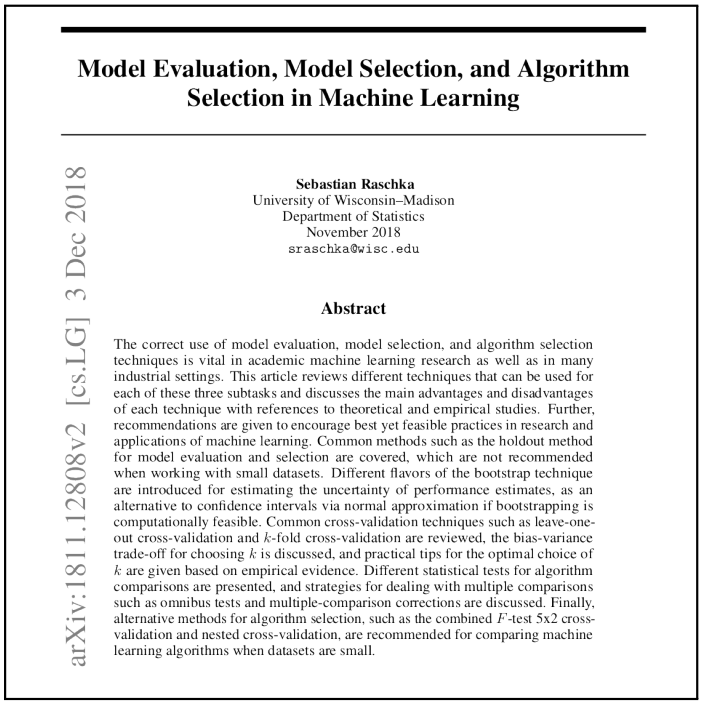


ML Algorithm Selection and Model Selection

Manually Choosing an Algorithm and Tuning a Model..

- algorithm selection
  (choose a $h()$).
- model selection
  (set hyperparameters on $h()$),
- model evaluation (train, test),
- re-iteration and re-selection!





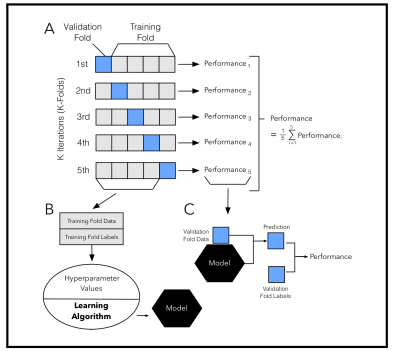
Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning

Sebastian Raschka
University of Wisconsin–Madison
Department of Statistics
November 2018
sraschka@wisc.edu

Abstract

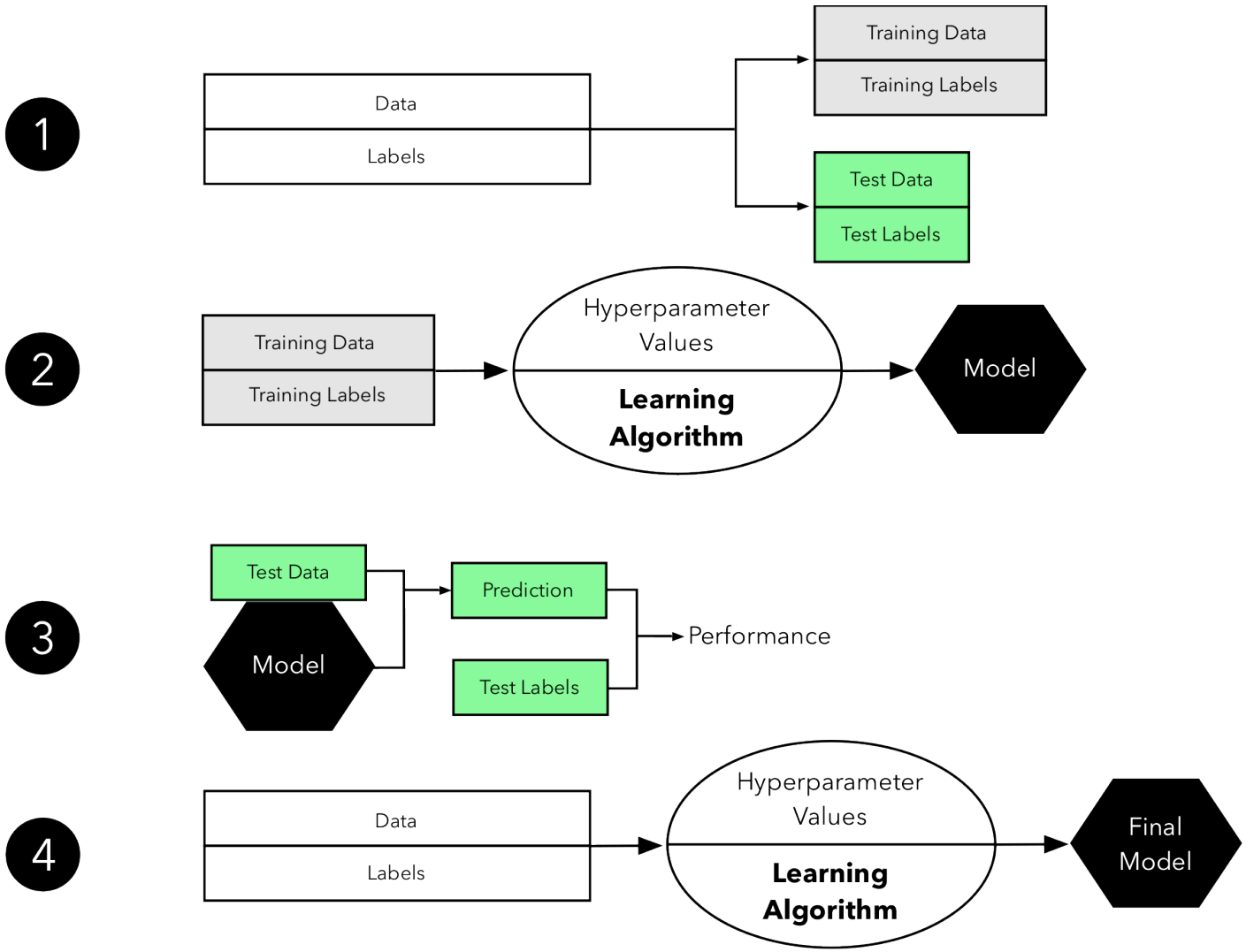
The correct use of model evaluation, model selection, and algorithm selection techniques is vital in academic machine learning research as well as in many industrial settings. This article reviews different techniques that can be used for each of these three subtasks and discusses the main advantages and disadvantages of each technique with references to theoretical and empirical studies. Further, recommendations are given to encourage best yet feasible practices in research and applications of machine learning. Common methods such as the holdout method for model evaluation and selection are covered, which are not recommended when working with small datasets. Different flavors of the bootstrap technique are introduced for estimating the uncertainty of performance estimates, as an alternative to confidence intervals via normal approximation if bootstrapping is computationally feasible. Common cross-validation techniques such as leave-one-out cross-validation and k-fold cross-validation are reviewed, the bias-variance trade-off for choosing k is discussed, and practical tips for the optimal choice of k are given based on empirical evidence. Different statistical tests for algorithm comparisons are presented, and strategies for dealing with multiple comparisons such as omnibus tests and multiple-comparison corrections are discussed. Finally, alternative methods for algorithm selection, such as the combined F-test 5x2 cross-validation and nested cross-validation, are recommended for comparing machine learning algorithms when datasets are small.

"Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning", Sebastian Raschka, 2018.



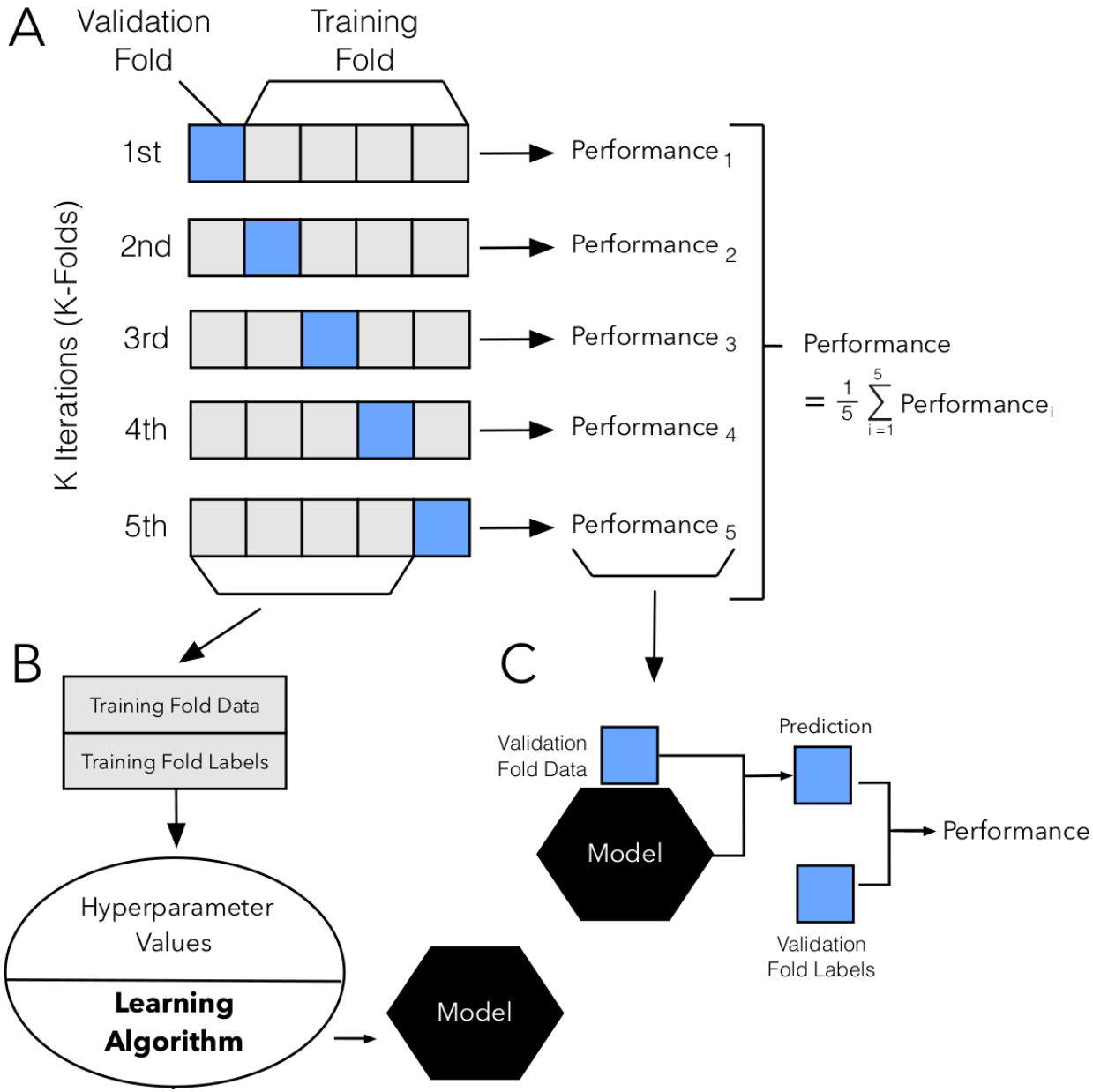
Model Evaluation

Simple Holdout Method (Train-Test Split)..

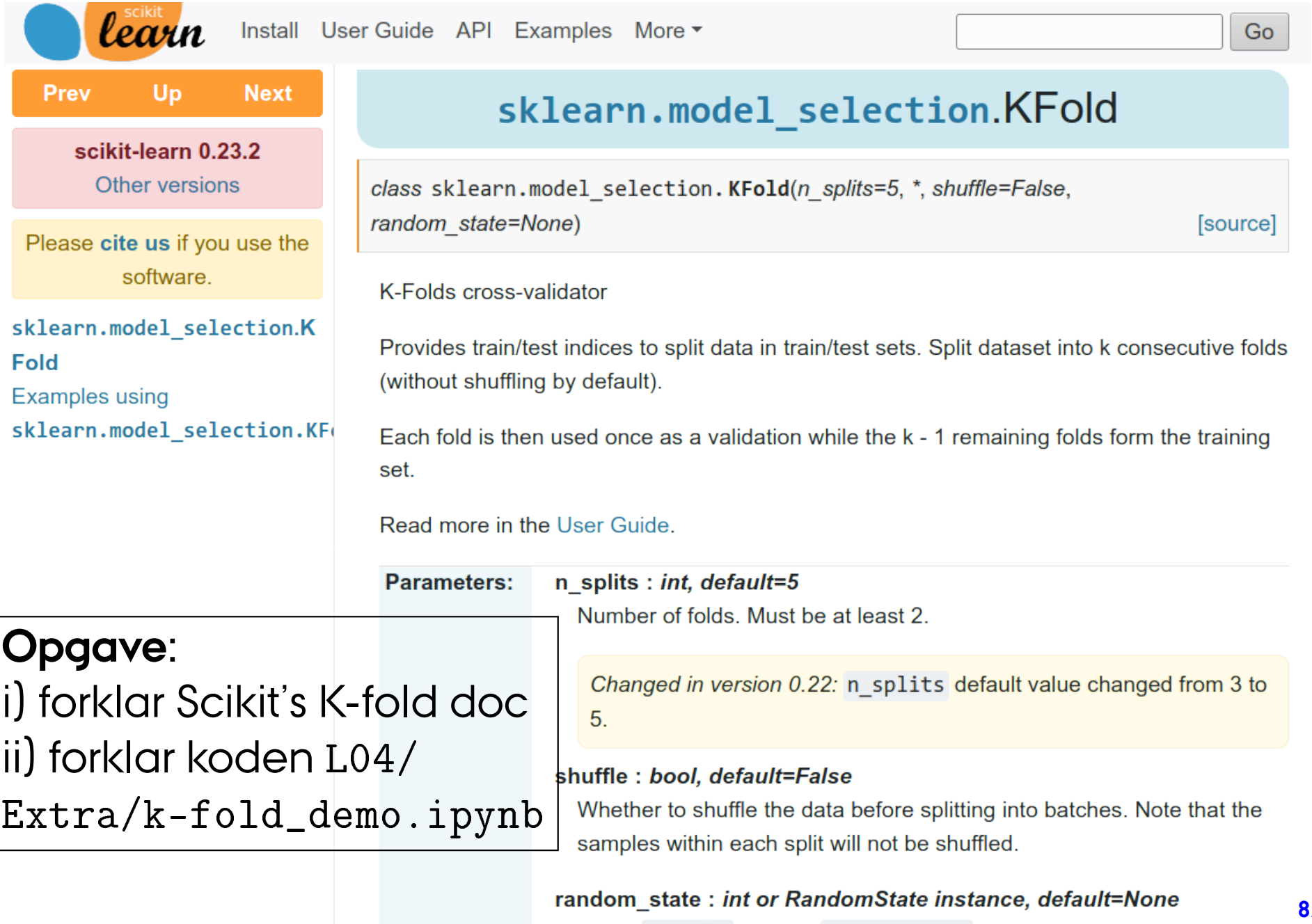


Model Evaluation

k-fold Cross-Validation Procedure, for *k*=5..



Scikit-learn K-fold Demo..



Opgave:
i) forklar Scikit's K-fold doc
ii) forklar koden `L04/Extra/k-fold_demo.ipynb`

Code Review: k-fold Cross Validation

p.89, [HOML]

Measuring Accuracy Using Cross-Validation

A good way to evaluate a model is to use cross-validation, just as you did in Chapter 2.

Implementing Cross-Validation

Occasionally you will need more control over the cross-validation process than what Scikit-Learn provides off-the-shelf. In these cases, you can implement cross-validation yourself; it is actually fairly straightforward. The following code does roughly the same thing as Scikit-Learn's `cross_val_score()` function, and prints the same result:

```
from sklearn.model_selection import StratifiedKFold
from sklearn.base import clone

skfolds = StratifiedKFold(n_splits=3, random_state=42)

for train_index, test_index in skfolds.split(X_train, y_train_5):
    clone_clf = clone(sgd_clf)
    X_train_folds = X_train[train_index]
    y_train_folds = y_train_5[train_index]
    X_test_fold = X_train[test_index]
    y_test_fold = y_train_5[test_index]

    clone_clf.fit(X_train_folds, y_train_folds)
    y_pred = clone_clf.predict(X_test_fold)
    n_correct = sum(y_pred == y_test_fold)
    print(n_correct / len(y_pred))  # prints 0.9502, 0.96565 and 0.96495
```

The `StratifiedKFold` class performs stratified sampling (as explained in Chapter 2) to produce folds that contain a representative ratio of each class. At each iteration the code creates a clone of the classifier, trains that clone on the training folds, and makes predictions on the test fold. Then it counts the number of correct predictions and outputs the ratio of correct predictions.

Code Review: k-fold Cross Validation

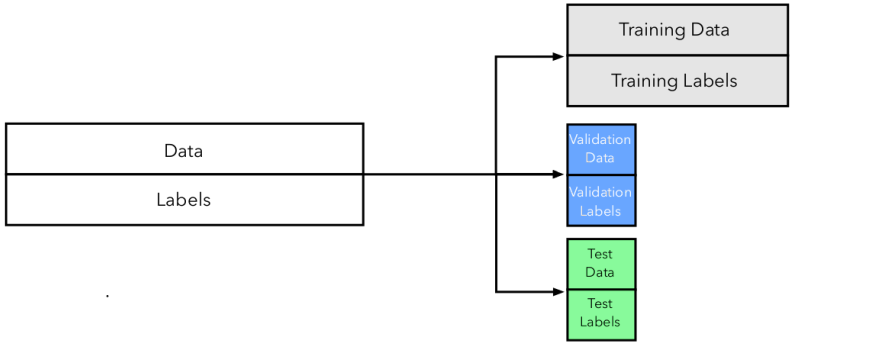
p.89, [HOML], with some code cleanup..

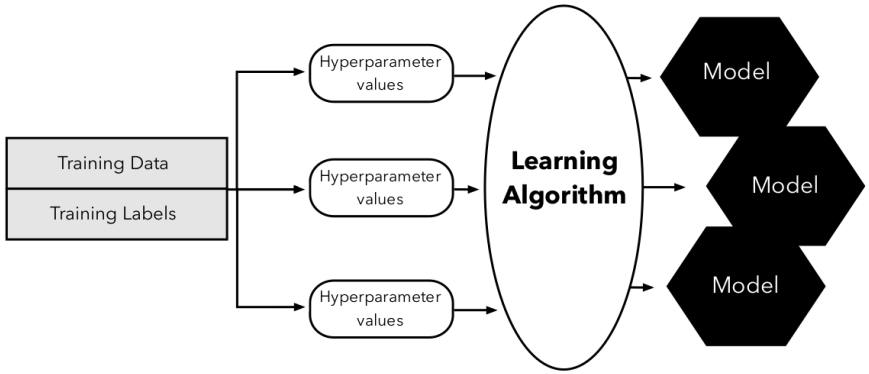
```
1   from sklearn.model_selection import KFold [..]
2
3   def MyKFoldSplit(clf, X, y, kfolds=3, debug=True):
4       def PrintVarInfo(varname, var): [..]
5
6       skfolds = KFold
           (n_splits=kfolds, random_state=42, shuffle=True)
7
8       for train_index, val_index in skfolds.split(X, y):
9           clone_clf = clone(clf)
10
11          X_train_folds = X[train_index]
12          y_train_folds = y[train_index]
13          X_val_fold    = X[val_index]
14          y_val_fold    = y[val_index]
15
16          clone_clf.fit(X_train_folds, y_train_folds)
17          y_pred = clone_clf.predict(X_val_fold)
18          CalcScores(i, y_val_fold, y_pred)
19          i += 1
20
21  print("K-fold demo..")
22  MyKFoldSplit(sgd_clf,
        X_train, y_train_5, 3)
```

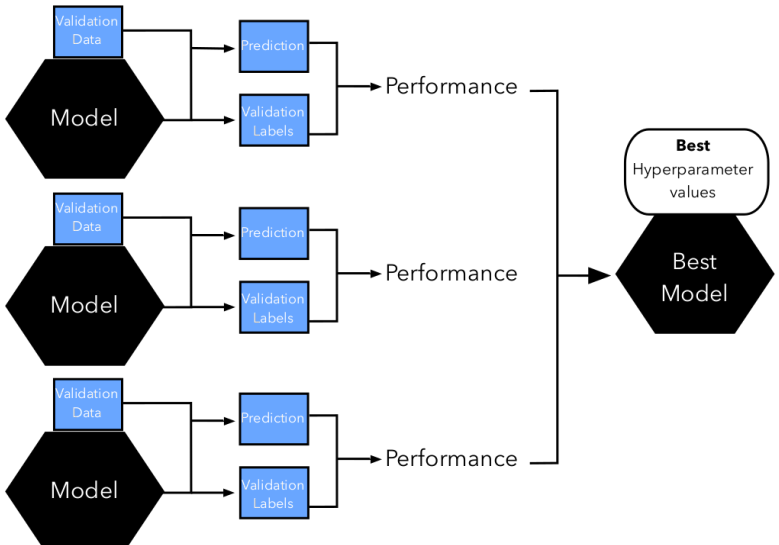
```
58 print("OK")

K-fold demo..
MyKFoldSplit(clf, X, y, kfolds=3)..
  type(X)           =<class 'numpy.ndarray'>,  X.shape           =(60000,
  type(y)           =<class 'numpy.ndarray'>,  y.shape           =(60000,
  type(train_index) =<class 'numpy.ndarray'>,  train_index.shape =(40000,
  type(val_index)   =<class 'numpy.ndarray'>,  val_index.shape   =(20000,
FOLD  0: accuracy=0.97, precision=0.94, recall=0.70, F1=0.80
FOLD  1: accuracy=0.95, precision=0.67, recall=0.89, F1=0.76
FOLD  2: accuracy=0.97, precision=0.89, recall=0.73, F1=0.80
```

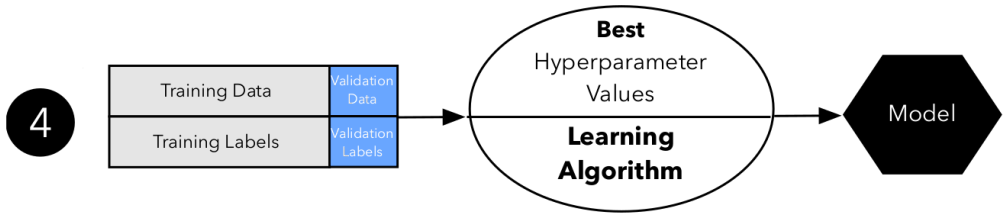
Model Evaluation and Selection

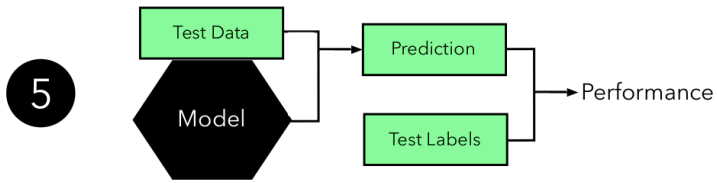
Three-way Holdout for Hyperparameter Tuning (Train-Validate-Test Split)...

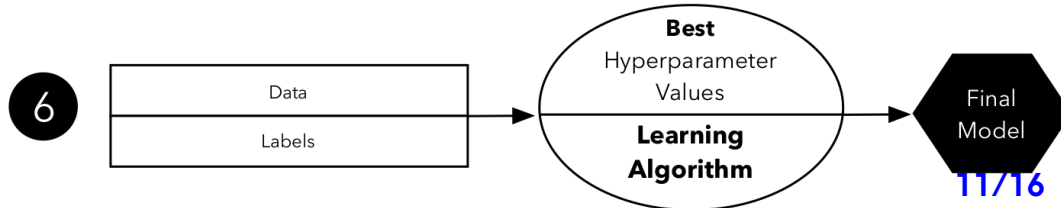






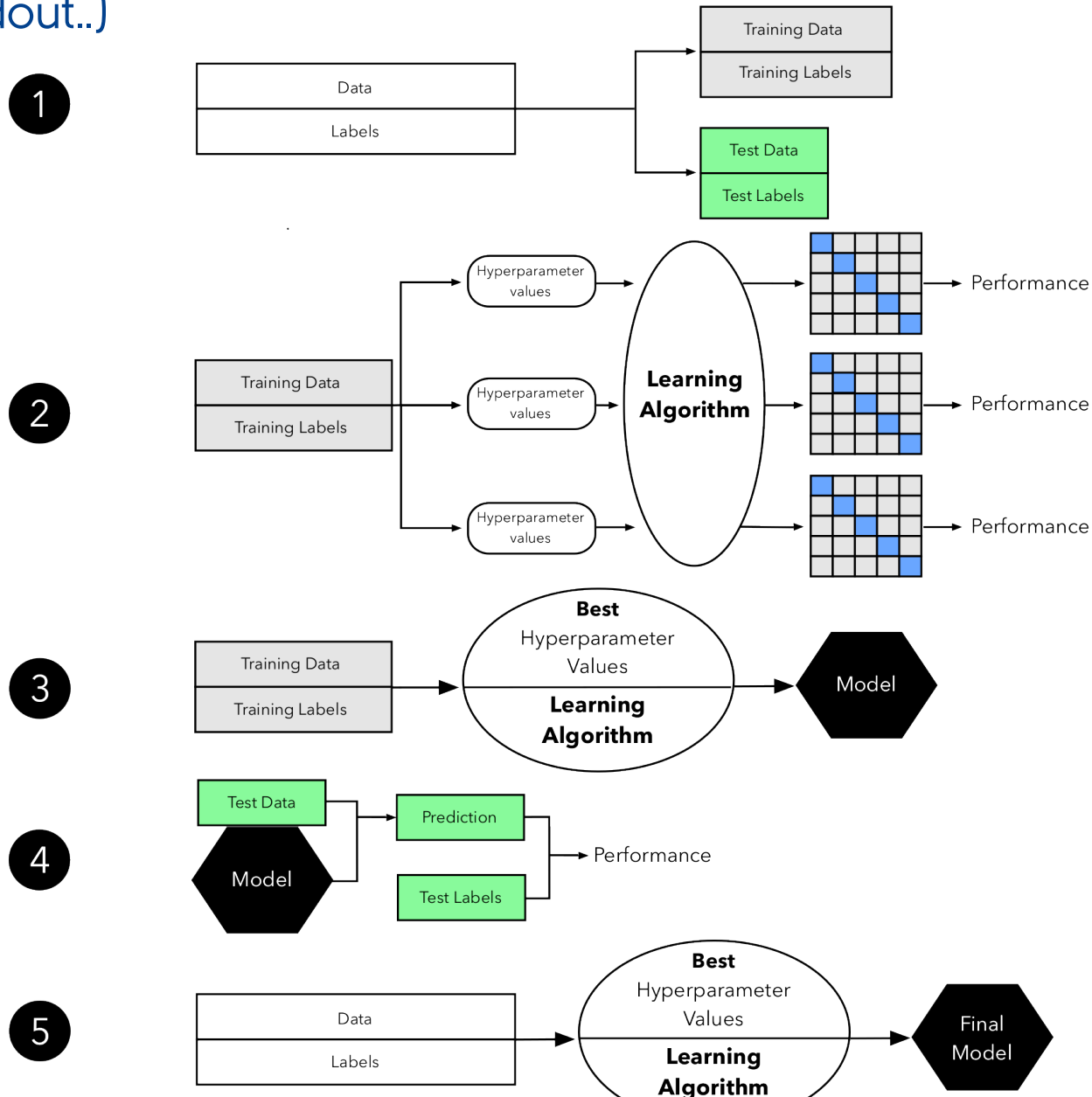






Model Evaluation and Selection

k-fold Cross-Validation for Hyperparameter Tuning (Somewhat Similar to Treeway Holdout..)



CHAPTER 2

End-to-End Machine Learning Project

Super-group Discussion

Present your oral resume for the [SG]

Supergruppe diskussion

§ 2 "End-to-End Machine Learning Project" [HOML]

Genlæs kapitel § 2 (eksklusiv "*Create the Workspace*" og "*Download the Data*"), og forbered mundtlig præsentation.

Forberedelse inden lektionen

Een eller flere af gruppe medlemmer forbereder et mundtligt resume af § 2:

- i skal kunne give et kort mundtligt resume af hele § 2 til en anden gruppe (på nær, som nævnt, Create the Workspace og Download the Data),
- resume holdes til koncept-plan, dvs. prøv at genfortælle, hvad de overordnede linier i kaptilerne i [HOML].

Lav et kort skriftlig resume af de enkelte underafsnit, ca. 5 til 20 liners tekst, se "TODO"-template herunder (MUST, til O1 aflevering).

Kapitler (incl. underkapitler):

- *Look at the Big Picture*
- *Get the Data (eksklusiv Create the Workspace og Download the Data),*
- *Discover and Visualize the Data to Gain Insights,*
- *Prepare the Data for Machine Learning Algorithms,*
- *Select and Train a Model,*
- *Fine-Tune Your Model,*
- *Launch, Monitor, and Maintain Your System,*
- *Try It Out!.*

På klassen

Supergruppe [SG] resume af § 2 End-to-End, ca. 30 til 45 min.

- en supergruppe [SG], sammensættes af to grupper [G], on-the-fly på klassen,
- hver gruppe [G] forbereder og giver en anden gruppe [G] et mundtligt resume af § 2 til en anden gruppe,
- tid: ca. 30 min- sammenlagt, den ene grupper genfortæller første haldel af § 2 i ca. 15 min., hvorefter den anden gruppe genfortæller resten i ca. 15 min.

End-to-end ML

'The Map'

ML algo+ model selection

model hyperparms

gridsearch/random search, minimize J ; θ^*

minimize J ; θ^*

model; θ

L_2 , L_1, cross-entropy

$J(\mathbf{X}, \mathbf{y}, \theta)^{(\text{train})}$

$\mathbf{X}^{(\text{train})}$

$\mathbf{y}^{(\text{train})}_{\text{true}}$

train $h(\mathbf{X}^{(\text{train})}, \theta)$

$\mathbf{y}^{(\text{train})}_{\text{pred}}$

loss; L_α $\mathbf{y}^{(\text{train})}_{\text{pred}}$; $\mathbf{y}^{(\text{train})}_{\text{true}}$

training done

$J(\mathbf{X}, \mathbf{y}, \theta)^{(\text{test})}$

$\mathbf{X}^{(\text{test})}$

$\mathbf{y}^{(\text{test})}_{\text{true}}$

test $h(\mathbf{X}^{(\text{test})}, \theta)$

$\mathbf{y}^{(\text{test})}_{\text{pred}}$

loss; L_α $\mathbf{y}^{(\text{test})}_{\text{pred}}$; $\mathbf{y}^{(\text{test})}_{\text{true}}$

train-test split

preprocessing

trained model; θ

generalization

performance metrics/scores

inference/run $h(\mathbf{X}^{(\text{new})}, \theta)$

$\mathbf{y}^{(\text{new})}_{\text{pred}}$

new data

$\mathbf{X}^{(\text{new})}$

$\mathbf{X}$

$\mathbf{y}_{\text{true}}$

K-fold cross val.
stratification
shuffling

precision, p
recall, r
accuracy, a
F-score, F_1
confusion matrix, $\mathbf{M}$

GD, BGD, epochs
early-stopping
generalization gap

R^2;
Pearson's r

ML Supervised Learning, Train/Test