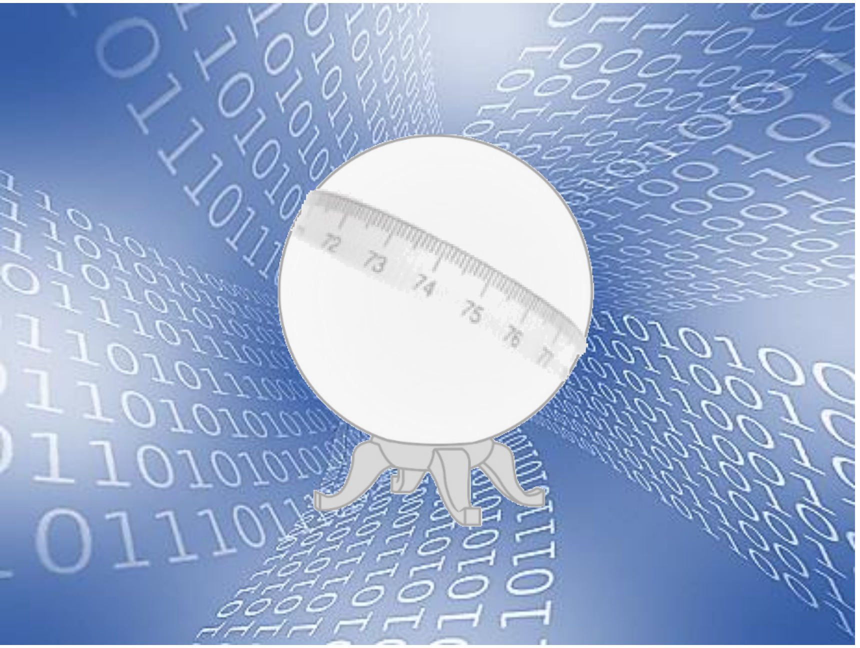
Prof. Dr. Bastian Amberg
School of Business & Economics
Department of Information Systems



Business Intelligence

12 How to avoid overfitting?

Prof. Dr. Bastian Amberg
(summer term 2024)
26.6.2024

Schedule

		Wed., 10:00-12:00		Fr., 14:00-16:00 (Start at 14:30)		Self-study		
	W1	17.4.	(Meta-)Introduction	19.4.		Python-Basics	Chap. 1	
	W2	24.4.	Data Warehouse – Overview & OLAP	26.4.	*[Blockveranstaltung SE Prof. Gersch]*		Chap. 2	
Basics	W3	1.5.		3.5.			Chap. 3	
	W4	8.5.	Data Warehouse Modeling I & II	10.5.	Data Mining Introduction			
	W5	15.5.	CRISP-DM, Project understanding	17.5.	Python-Basics-Online Exercise	Python-Analytics	Chap. 1	
	W6	22.5.	Data Understanding, Data Visualization I	24.5.	*No lectures, but bonus tasks* *1.) Co-Create your exam*		Chap. 2	
	W7	29.5.	Data Visualization II	31.5.	*2.) Earn bonus points for the exam*			
Main Part	W8	5.6.	Data Preparation	7.6.	Predictive Modeling I (10:00 -12:00)	BI-Project	Start	
	W9	12.6.	Predictive Modeling II	14.6.	Python-Analytics-Online Exercise		|	
	W10	19.6.	*Guest Lecture Dr. Ionescu*	21.6.	Fitting a Model		|	Case Study
	W11	26.6.	How to avoid overfitting	28.6.	What is a good Model?		|	
	W12	3.7.	Project status update Evidence and Probabilities	5.7.	Similarity (and Clusters) From Machine to Deep Learning I		|	
Deep-ening	W13	10.7.		12.7.	From Machine to Deep Learning II		|	
	W14	17.7.	Project presentation	19.7.	Project presentation		End	
					Klausur 1.Termin, 31.7.'24 *Klausur 2.Termin, 2.10.'24*	Projektbericht		

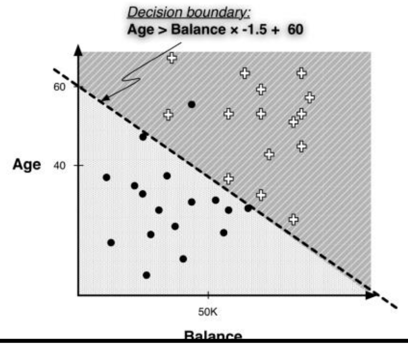
Ref.

Last Lesson

Predictive Modeling   Classification via Mathematical Functions – Fitting a Model to Data

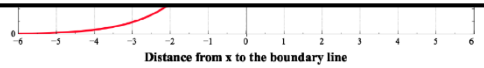
- We **specify the structure of the model**, but leave certain numeric parameters unspecified
- Data Mining calculates the best parameter values given a particular set of training data
- The form of the model and the attributes is specified
- The goal of DM is to tune the parameters so that the model fits the data as good as possible (**parameter learning**)

Instance-space view
linear classifier function



Kahoot-Fragen
www.kahoot.it
(über Smartphone oder Laptop)
PIN folgt

Diese Folie ist nach der Vorlesung vollständig sichtbar.



The identified maximum likelihood model "on average" gives the highest probabilities to the positive examples and the lowest probabilities to the negative examples.

Now: Tree-induction vs. logistic regression

Ref.

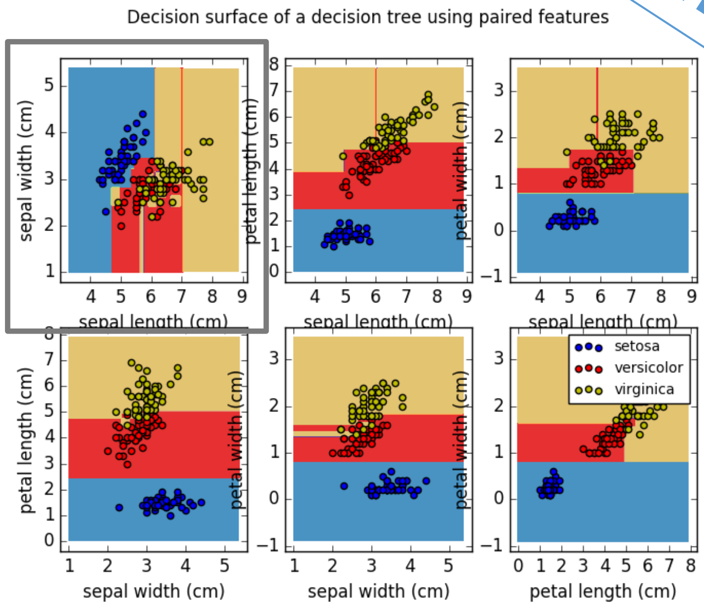
Tree induction vs. linear classifier (in general)

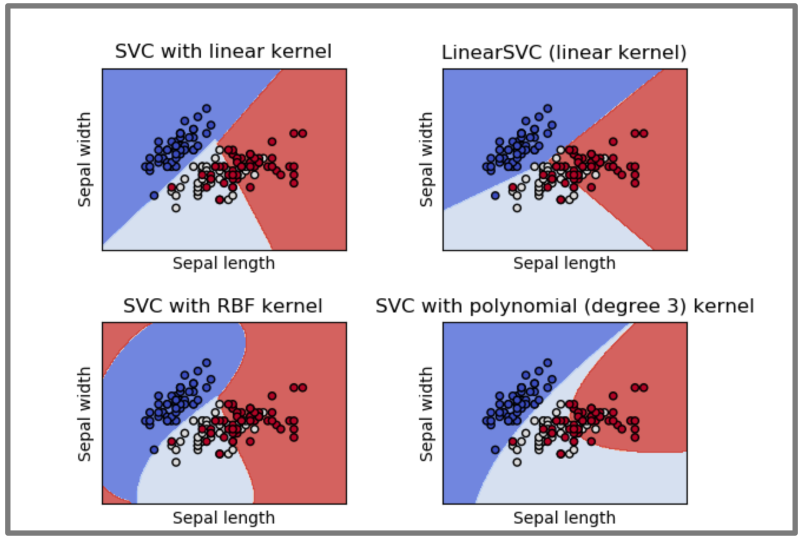
Wdh.

Important differences between trees and linear classifiers

- A classification tree uses decision boundaries that are **perpendicular** to the instance-space axes.
- The linear classifier can use **decision boundaries of any direction** or orientation
- A classification tree is a **"piecewise" classifier** that segments the instance space recursively → cut in arbitrarily small regions possible.
- The linear classifier places **a single decision** surface through the entire space.

Which of these characteristics are a better match to a given data set?





Ref.

https://scikit-learn.org

Tree induction vs. logistic regression

Example: Breast Cancer Dataset

Wdh.

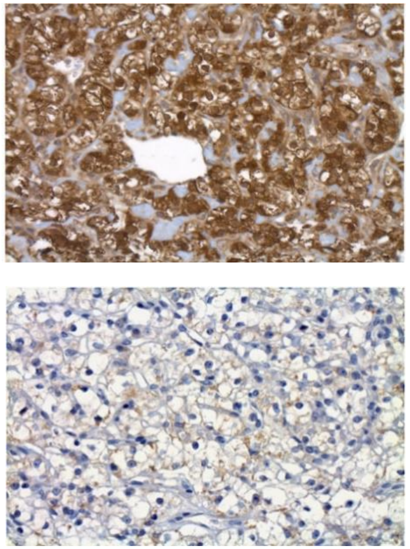
Consider the background of the stakeholders

- A decision tree may be considerably **more understandable** to someone without a strong background in statistics
- Data Mining team does not have the ultimate say how models are used or implemented!

Example: Wisconsin Breast Cancer Dataset

http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

- Each record describes characteristics of a cell nuclei image, which has been labeled as either “benign” or “malignant” (**cancerous**)
- Ten fundamental characteristics were extracted and summarized in a mean (_mean), standard error (_SE) and mean of the three largest values (_worst)
  → 30 measured attributes
- 357 benign images and 212 malignant images



From Mu et al. (2011) doi:10.1038/ncomms1332

Ref.

Tree induction vs. logistic regression

Example: Breast Cancer Dataset

Results of logistic regression

Weights of linear model

Ordered from highest to lowest

Performance: only six mistakes on the entire dataset, accuracy 98.9%

Attribute	Weight
Smoothness_worst	22.30
Concave_mean	19.47
Concave_worst	11.68
Symmetry_worst	4.99
Concavity_worst	2.86
Concavity_mean	2.34
Radius_worst	0.25
Texture_worst	0.13
Area_SE	0.06
Texture_mean	0.03
Texture_SE	-0.29
Compactness_mean	-7.10
Compactness_SE	-27.87
w_0 (intercept)	-17.70

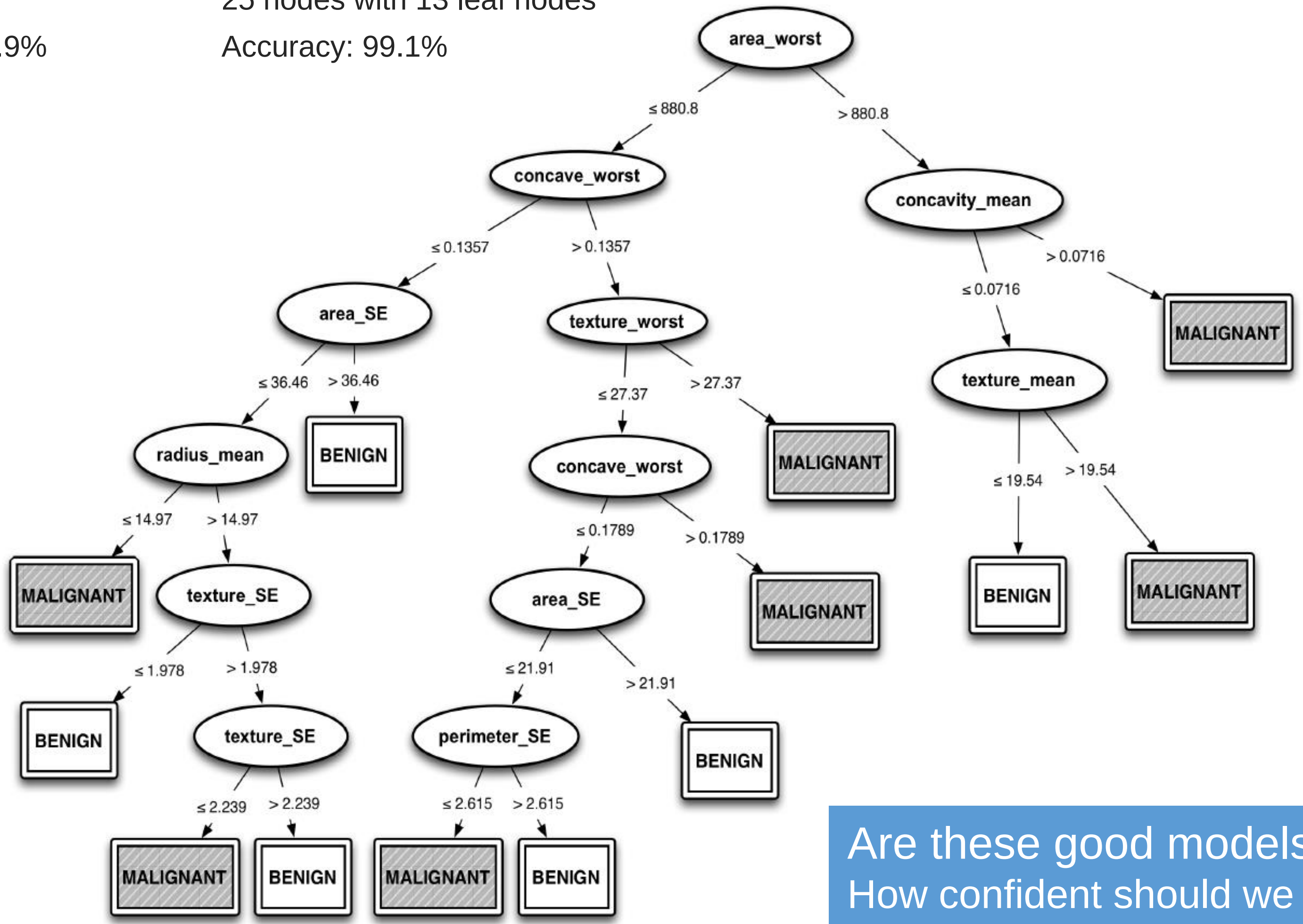
Comparison with classification tree from the same dataset

Wdh.

Weka's J48 implementation

25 nodes with 13 leaf nodes

Accuracy: 99.1%



Are these good models?
How confident should we be in this evaluation?

Agenda

How to avoid overfitting?

(1) Generalization and Overfitting

(2) From holdout evaluation to cross-validation

(3) Learning curves

(4) Overfitting avoidance and complexity control

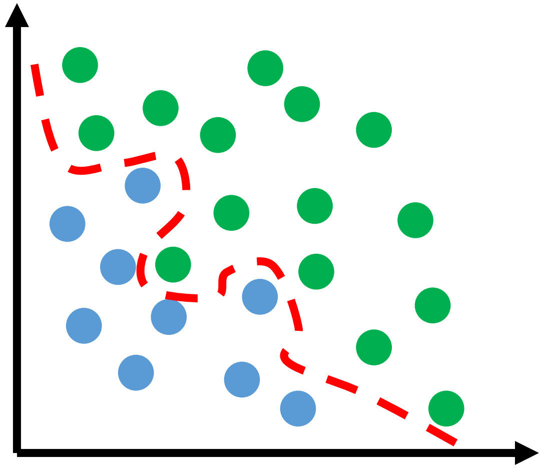
Ref.

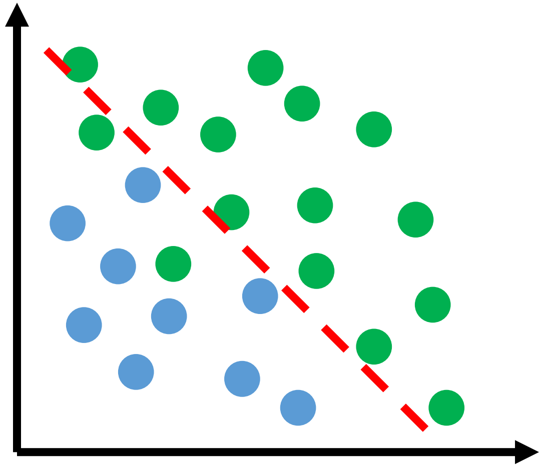
How to avoid overfitting?

Introduction

Fundamental trade-off in DM between **overfitting** and **generalization**

If we allow ourselves enough flexibility in searching, we *will* find patterns
Unfortunately, these patterns may be just occurences by chance





Overfitting: finding chance occurences in data that *look like* interesting patterns, but which do *not generalize*

We are interested in patterns that **generalize**, i.e., that predict well for instances that we have not yet observed

Ref.

Generalization

Counterexample: The table model

We can **always** build a perfect model!

- Store the feature vector for each customer who churned (*table model*)
- Look the customer up when determining the likelihood of churning

College	Income	Long Calls per Month	Average Call Duration	…	Leave?
Yes	3,000	23	28	…	Yes
Yes	1,424	2	120	…	No
No	6,201	42	70	…	No
…	…	…	…	…	…
No	543.12	20	140		Yes

Example: Churn data set

- Historical data on customers who have stayed with the company, and customers who have departed within six months of contract expiration
- Task:
  build a model to distinguish customers who are likely to churn based on some features
- We test the model based on historical data, and the model is **100% accurate**, identifying correctly all the churners as well as the non-churners

Ref.

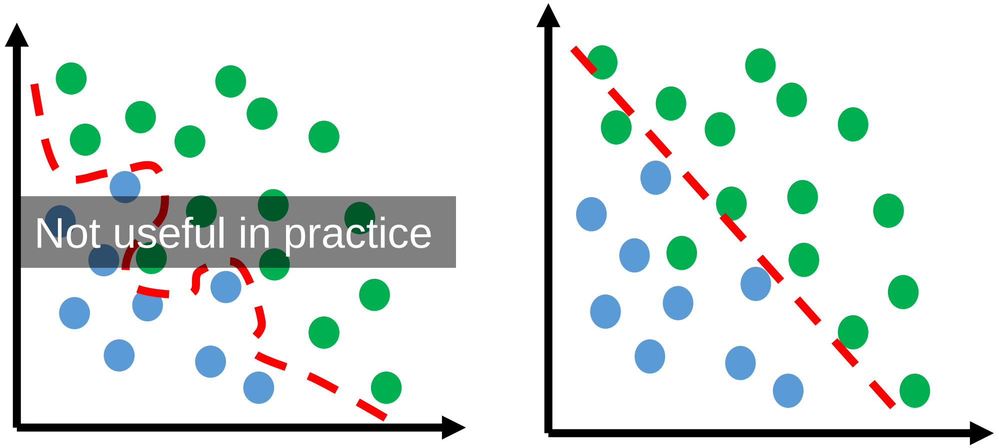
Generalization

A table model memorizes the training data and performs **no generalization**

- Useless in practice! Previously unseen customer's would all end up with “0% likelihood of churning”

Generalization is the property of a model or modeling process whereby the model applies to data **that were not used to build the model**

- If models do not generalize at all, they fit perfectly to the training data → they overfit.

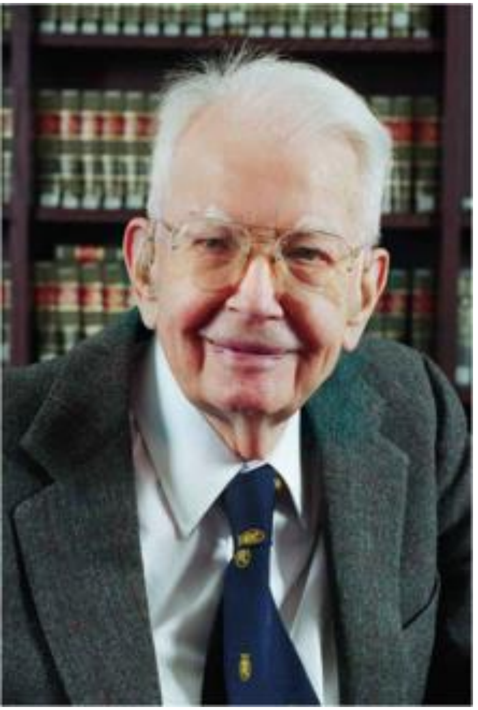


Our imperative:
DM needs to create models that **generalize** beyond training data

Ref.

Overfitting

„If you torture the data long enough, it will confess.“ (Ronald Coase)



Overfitting is the tendency of DM procedures to tailor models to the training data, at the expense of generalization to previously unseen data points.

All DM procedures tend to overfitting

Trade-off between **model complexity** and the possibility of overfitting

- Recognize overfitting and manage complexity in a principled way

Complexity:
Number of decisions to be made by the model, i.e., amount of parameters to be estimated;

Examples?

Ref.

Holdout data

Data for testing; data for training

Evaluation on training data provides no assessment of how well the model generalizes to unseen cases

> Rational:
> „Hold out" some data for which we know the value of the target variable, but which will not be used to build the model → „lab test"

Predict the values of the **„holdout data"** (aka **„test set"**) with the model and **compare** them with the hidden true values → generalization performance

> There is likely to be a difference between the model's accuracy („in-sample" accuracy) and the model's generalization accuracy

Real world data
A B A A A B A A B B

Training Data | Hold out data
A B A A A B | A A B B

Predicted Value | A *B* B B
Hold out value | A *A* B B

Where have we already done this?

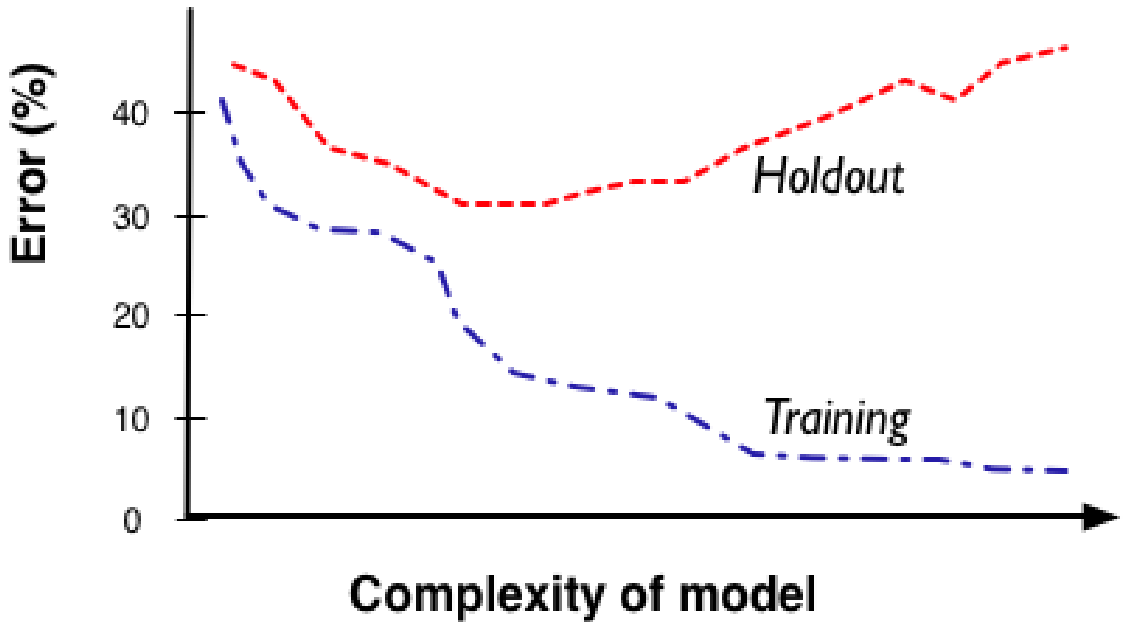
Ref.

Fitting graph

Error / Complexity

A fitting graph shows the **accuracy of a model** as a function of complexity

(accuracy = 1 – error)



Generally, there will be more overfitting as one allows the model to be more complex

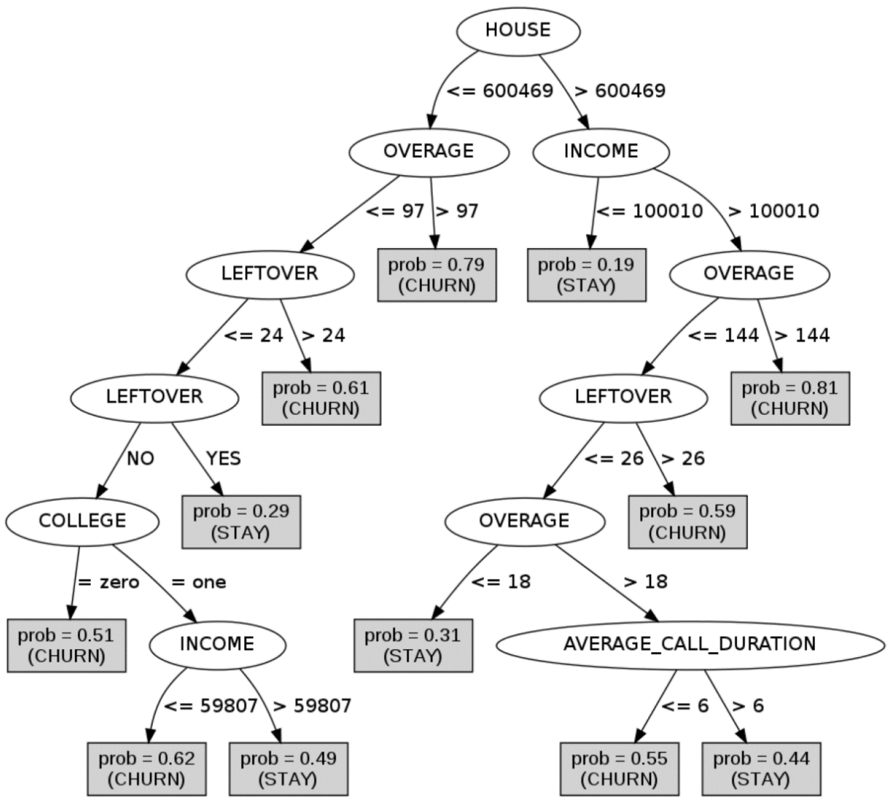
Ref.

Overfitting in tree induction (1/2)

Recall **tree induction**:
find important, predictive individual attributes recursively to smaller and smaller data subsets

- Eventually, the subsets will be pure – we have found the leaves of our decision tree
- The accuracy of this tree will be perfect!
- This is the same as the table model, i.e., an **extreme example of overfitting**
- This tree should be slightly better than the lookup table, because every previously unseen instance also will arrive at some classification rather than just failing to match

Useful for comparison of how well the accuracy on the training data tends to correspond to the accuracy on test data



prob = estimated probabilities of churning

Ref.

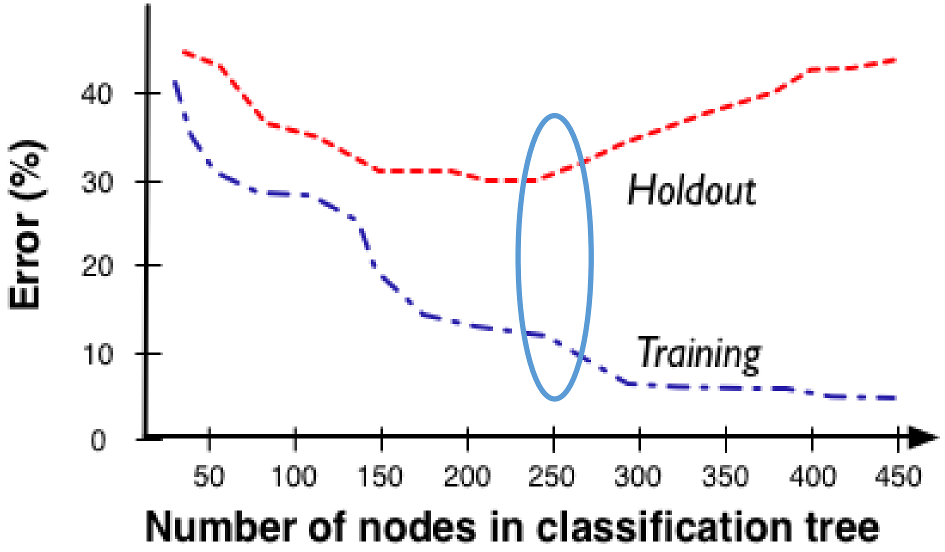
Overfitting in tree induction (2/2)

Generally:

A procedure that grows trees until the leaves are pure tends to overfit

If allowed to grow without bound, decision trees can fit any data to arbitrary precision

The **complexity of a tree** lies in the number of nodes



Ref.

Overfitting in parametric learning (1/2)

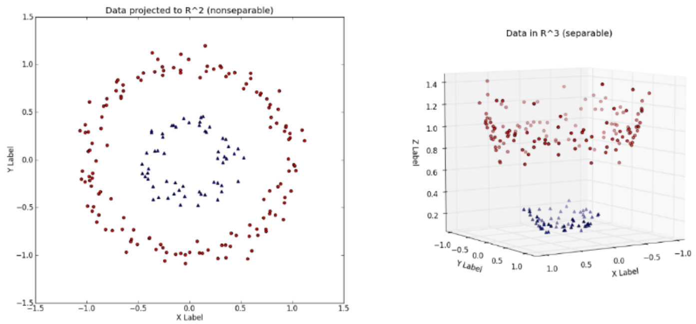
Next Lesson

There are different ways to allow more or less complexity in mathematical functions

- Add more variables (more attributes):

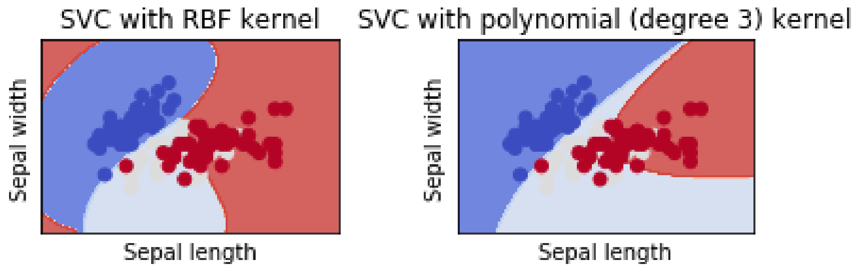
$$f(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_3x_3$$
$$f(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5$$

- Add attributes that are non-linear, i.e., $x_5 = x_2/x_3$ or $x_4 = x_1^2$ (see kernel trick from SVM)



As you **increase the dimensionality**, you can perfectly fit larger and larger sets of arbitrary points

- Often, modelers carefully prune the attributes in order to avoid overfitting → manual selection
- Automatic feature selection



Ref.

Overfitting in parametric learning (2/2)

Freie Universität Berlin

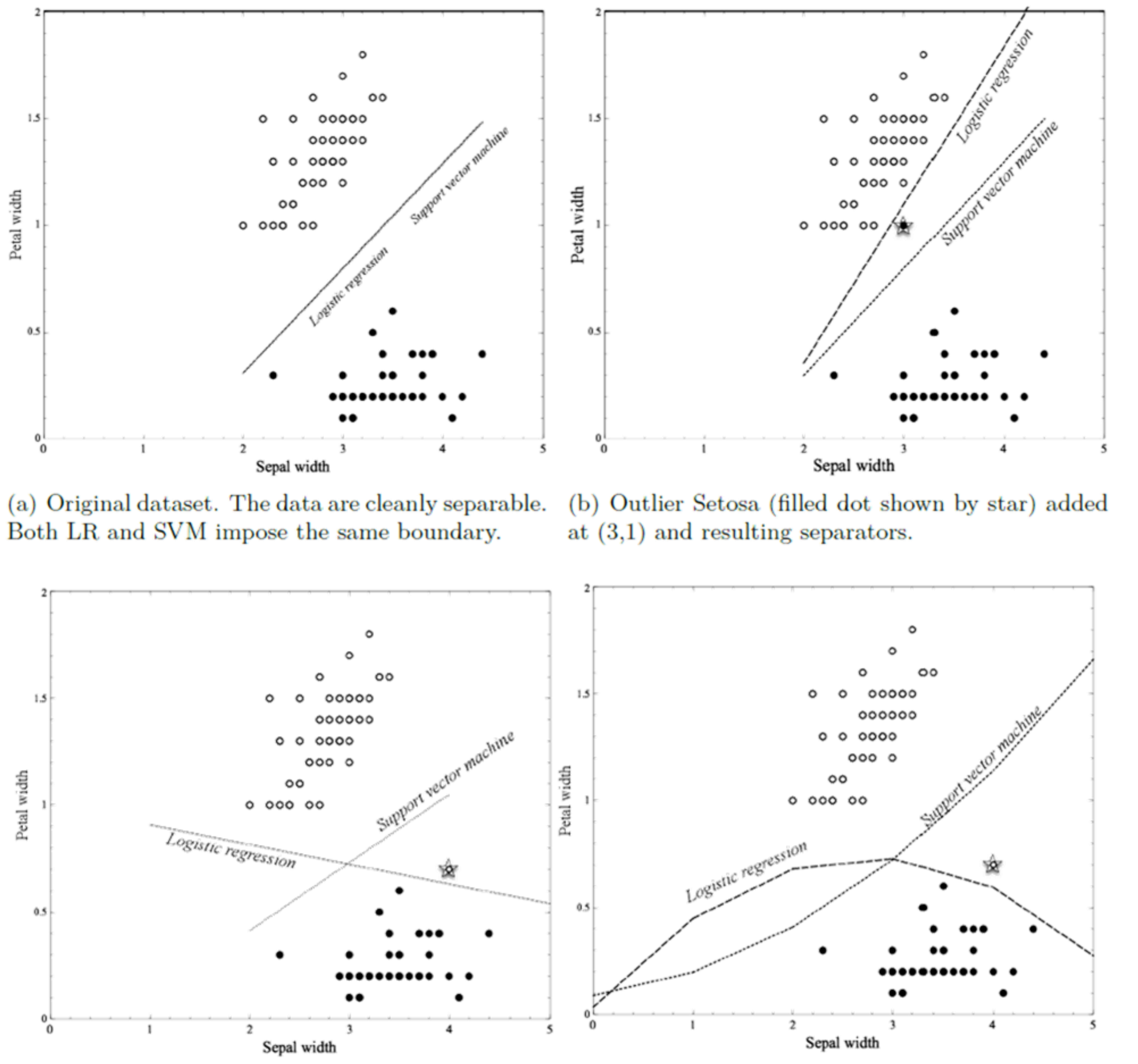
Iris Example

a) Original Iris data set
both logistic regression and support vector machines place separating boundaries in the middle

b) A single new example has been added (3,1) ●
logistics regression still separates the groups perfectly, while the SVM line barely moves at all

c) A different outlier has been added (4,0.7) ○
again, SVM only moves very little – logistic regression appears to be overfitting considerably

d) Add a squared term of the sepal width
More flexibility in fitting the data – seperating line becomes a parabola



(a) Original dataset. The data are cleanly separable. Both LR and SVM impose the same boundary.

(b) Outlier Setosa (filled dot shown by star) added at (3,1) and resulting separators.

(c) Outlier Versicolor (circle with star) added at (4,0.7) and resulting separators.

(d) Same data as in (c) but with squared Sepal width term added.

Ref.

Agenda

Freie Universität Berlin

(1) Generalization and Overfitting

(2) From holdout evaluation to cross-validation

(3) Learning curves

(4) Overfitting avoidance and complexity control

Ref.

Holdout training and testing

Cross-validation is a more sophisticated training and testing procedure

- Not only a simple estimate of the generalization performance, but also some statistics on the estimated performance (mean, variance, …)
- How does the performance vary across data sets?

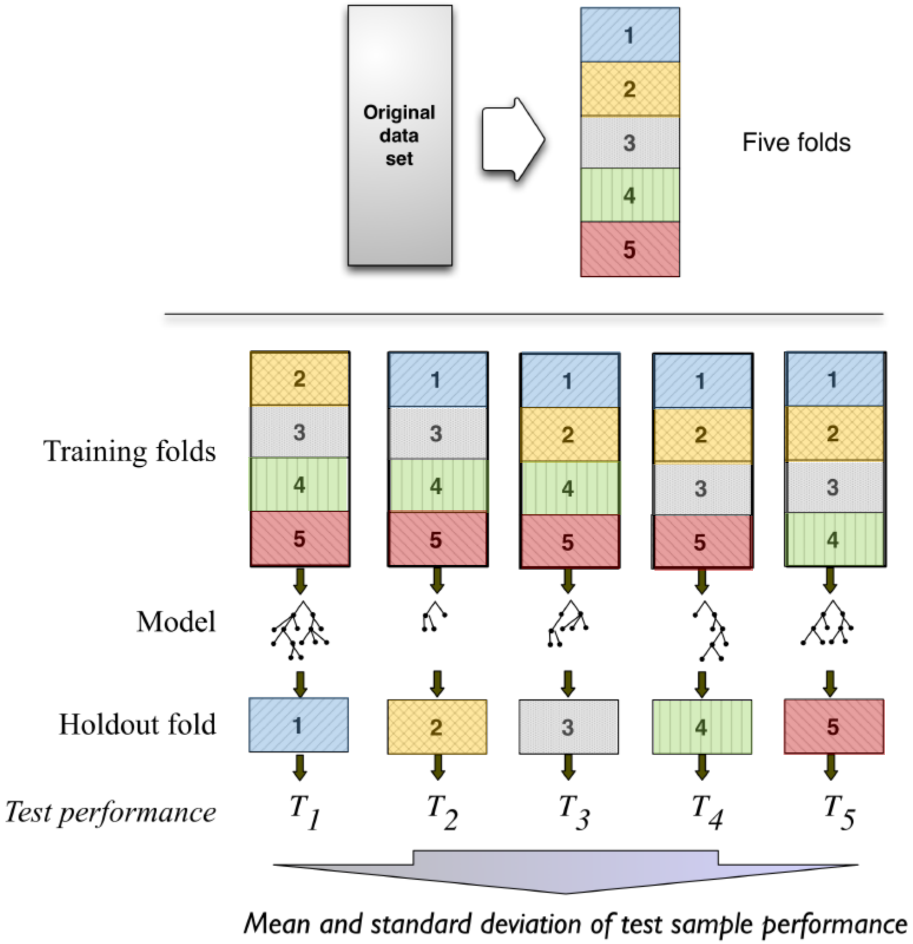
→ assessing confidence in the performance estimate

Cross-validation computes its estimates over all the data by **performing multiple splits** and systematically swapping out samples for testing

- Split a data set into k (equal sized) partitions called **folds** ($k = 5$ or 10)
- Iterate training and testing k times
- In each iteration, a different fold is chosen as the test data. The other $k - 1$ folds are combined to form the training data.

Illustration of cross-validation

- Every example will have been used only once for testing but $k - 1$ times for training
- Compute average and standard deviation from k folds



Ref.

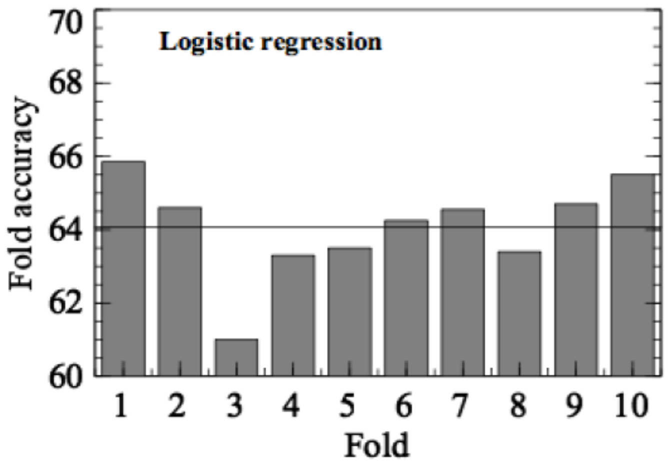
Cross-validation for the churn data set

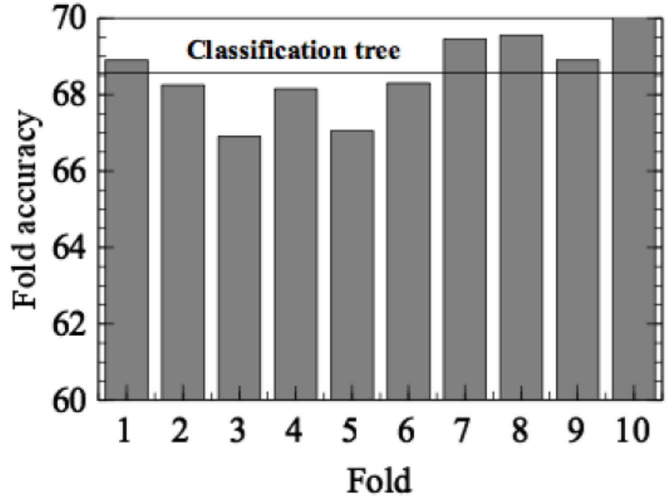
Recall churn dataset with an accuracy of 73%

Cross-validation:
the dataset was first shuffled,
then divided into ten partitions

Classification trees:
avg accuracy is 68.6% (std 1.1)

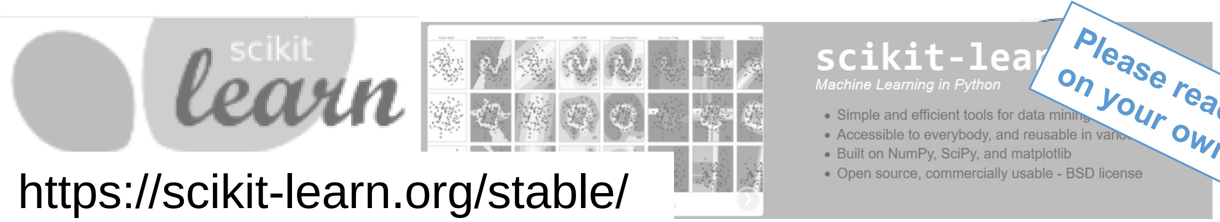
Logistic regression models:
avg accuracy is 64.1% (std 1.3)





Ref.

Cross-validation in Python



https://scikit-learn.org/stable/

Please read on your own

Example

python™

Python sklearn offers multiple ways for validation, but also for cross-validation.

In its easiest form, you train a model:

```
clf = svm.SVC(kernel='linear', C=1)
```

Then, you test it.

Either with a *simple scoring method*

```
scores = clf.score(iris.data,
iris.target)
```

```
print("Accuracy: %0.2f"
% (scores))
```

```
e.g. Accuracy: 0.96
```

Or with *cross-validation* (X times, with varying splits)

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(clf, iris.data, iris.target, cv=5)
```

```
print("Accuracy: %0.2f (+/- %0.2f)"
% (scores.mean(), scores.std() ))
```

```
e.g. Accuracy: 0.96 (+/- 0.03)
```

Ref. SKLearn (Documentation)

Agenda

(1) Generalization and Overfitting

(2) From holdout evaluation to cross-validation

(3) Learning curves

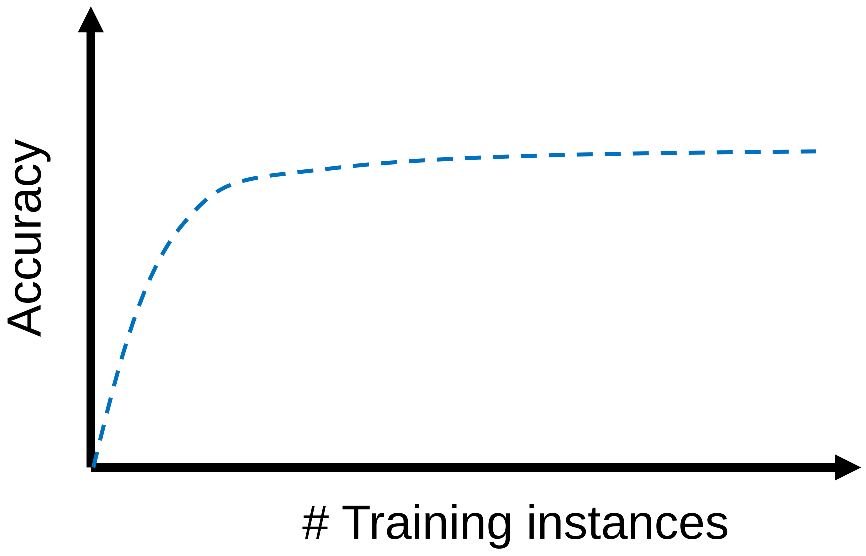
(4) Overfitting avoidance and complexity control

Ref.

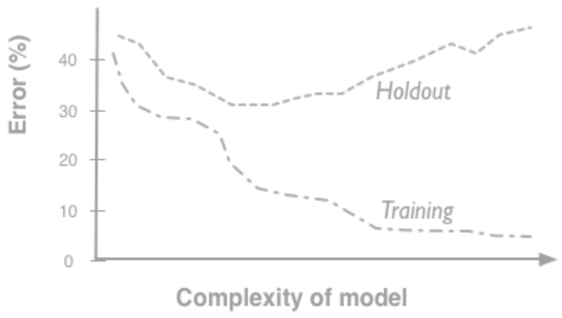
Learning curves (1/2)

Learning curve (accuracy to Instances)
= a plot of the **generalization performance** (testing data) against the amount of training data

- Reflects that generalization performance improves as more training data are available
- Characteristic **shape**: steep initially, but then marginal advantage of more data decreases



Fitting curve (accuracy to features)
= shows the performance on the training and the testing data against model complexity (for a fixed amount of training data)



Ref.

Learning curves (2/2)

Example for the churn data set

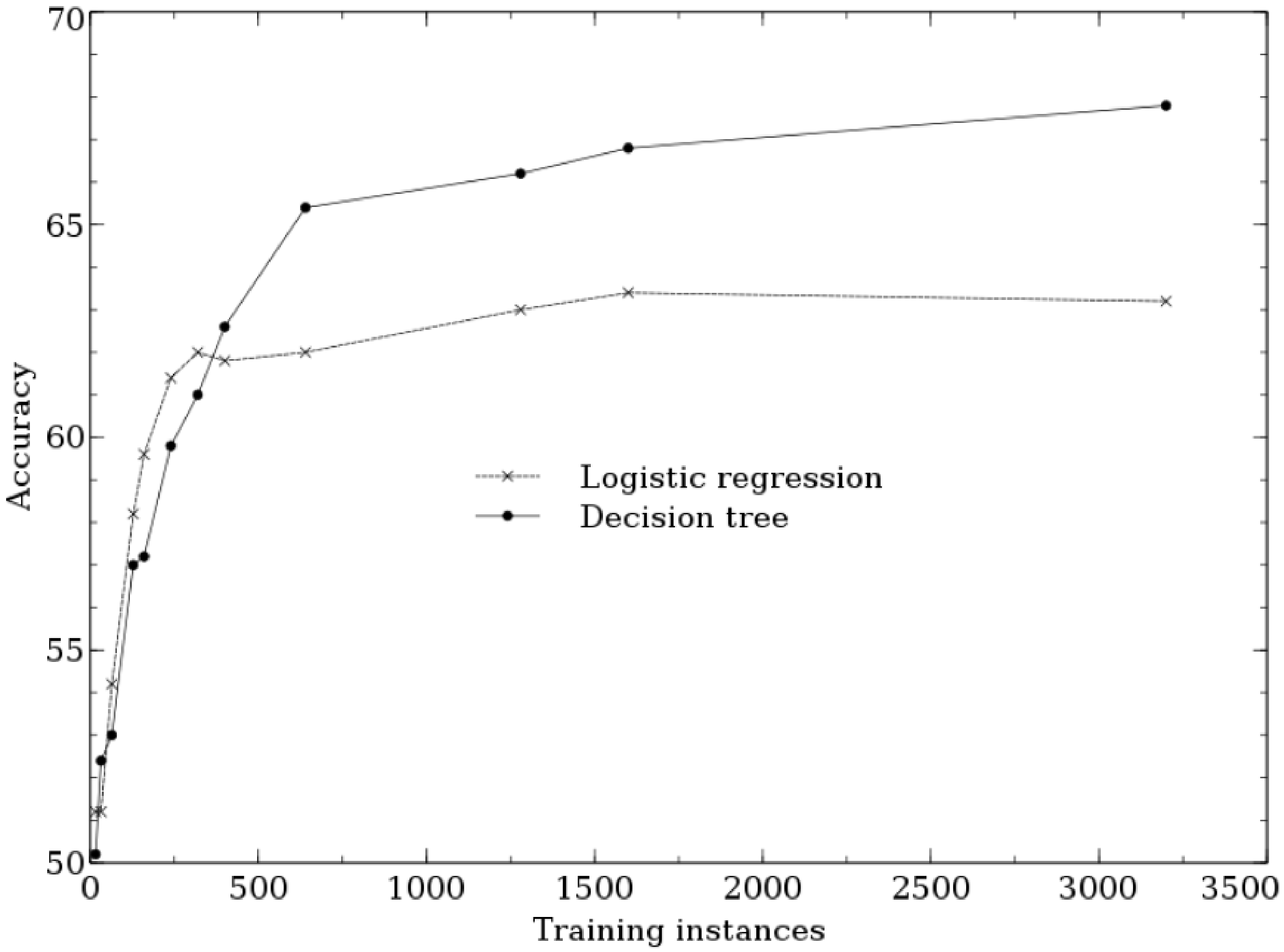
Same data – different generalization performance:

- For **smaller** training-set sizes, logistic regression yields better generalization accuracy
- For **larger** training-set sizes, tree induction soon is more accurate

Classification trees are a **more flexible model** representation than linear logistic regression

- Smaller data: tree induction will tend to overfit more
- Flexibility of tree induction for larger training sets

Learning curve may give recommendations on **how much to** invest in training data



Ref.

Avoiding overfitting for tree induction

Tree induction will likely result in large, overly complex trees that overfit the data

- **Stop growing** the tree before it gets too complex

Simplest method to limit tree size: specify a **minimum number of instances** that must be present in a leaf

Automatically grow the tree branches that have a lot of data and cut short branches that have less data

What **threshold** should we use?

1. Experience
2. Conduct a hypothesis test at every leaf to determine whether the observed difference in information gain could have been due to chance (e.g., p-value below 5%) Accept split, if it was likely not due to chance

Ref.

- **Prune back** a tree that is too large (reduce its size)

Prune an overly large tree = cut off leaves and branches and replace them with leaves

Estimate whether replacing a set of leaves or a branch with a leaf would reduce accuracy

If not, then prune

Continue process iteratively, until any removal or replacement would reduce accuracy

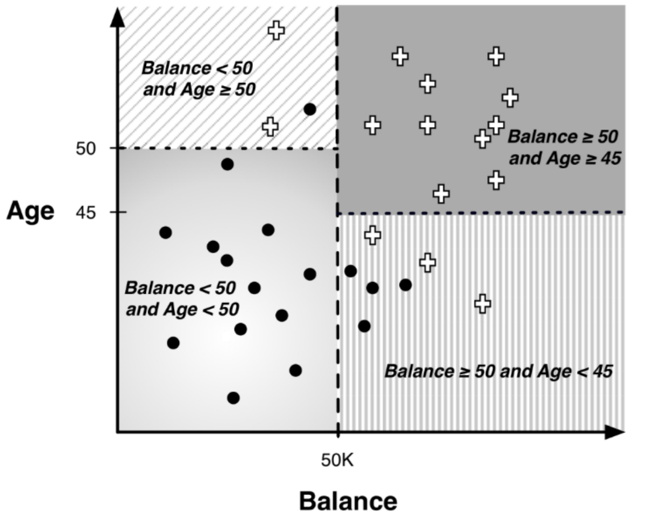
- **Build trees with all sorts of different complexities** and **estimate their generalization performance**

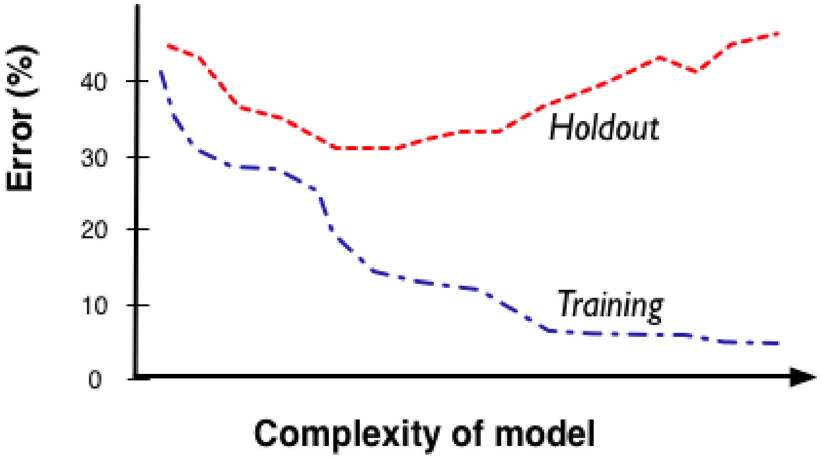
Pick the one that is estimated to be the best

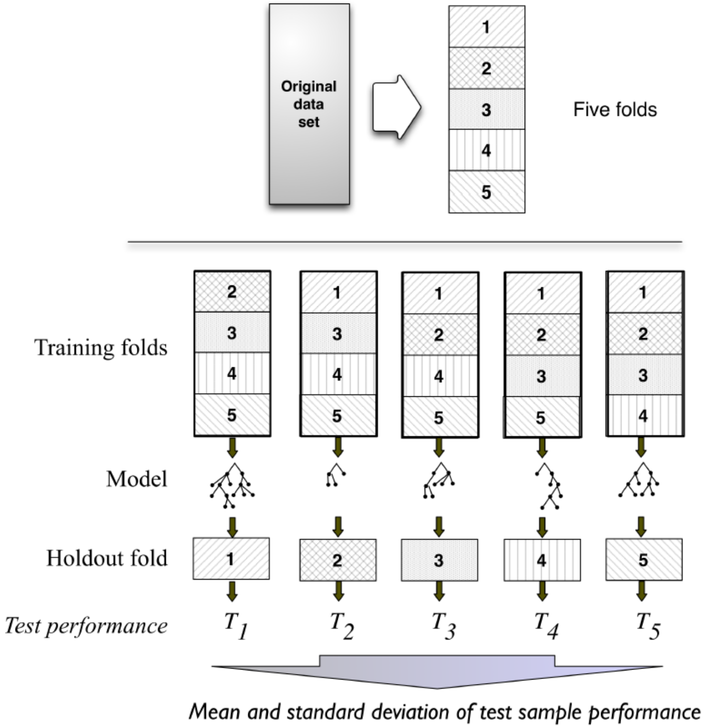
Excursus: Random Forest, e.g., kaggle tutorial python

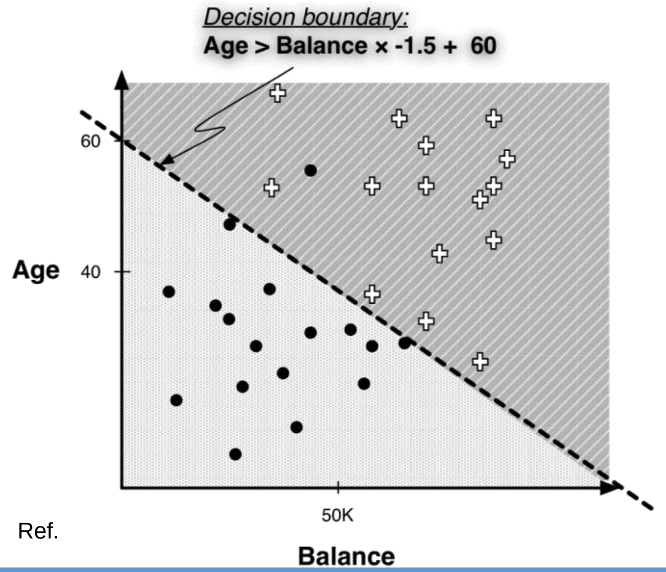
Summary – Exercise

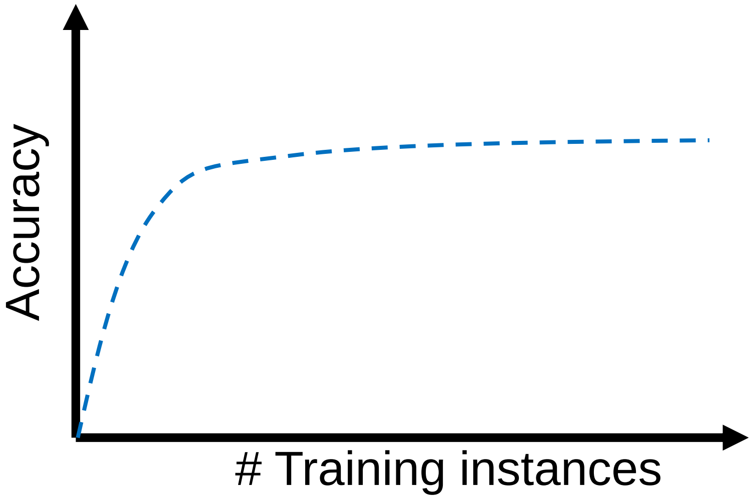
Explain what you see.











10 Min.

Ref.

Fragen?

- ✓ Tree induction vs. linear classifier

- ✓ Generalization and Overfitting
- ✓ From holdout evaluation to cross-validation
- ✓ Fitting graphs
- ✓ Learning curves
- ✓ Overfitting avoidance and complexity control

Todos for this week

Further information about overfitting and avoiding overfitting

- Please read: Why is overfitting bad? - Example
  Slides 30-31

- Please read: A general method for avoiding overfitting
  Kursmaterial > Readings & Übungen
  (short version on slides 32-33)

Ref.

Recommended reading

How to avoid Overfitting

Provost, F., Fawcett, T.	Data Science for Business Chapter 5
Berthold et al.	Several subchapters

Ref.

Why is overfitting bad?

Please read on your own

Why is overfitting causing a **model to become worse**?

- As a model gets more complex, it is allowed to pick up harmful „spurious“ correlations
- These correlations do not represent characteristics of the population in general
- They may become harmful when they produce **incorrect** generalizations in the model

➢ Example: a simple two-class problem

Ref.

Why is overfitting bad? - Example

Please read on your own

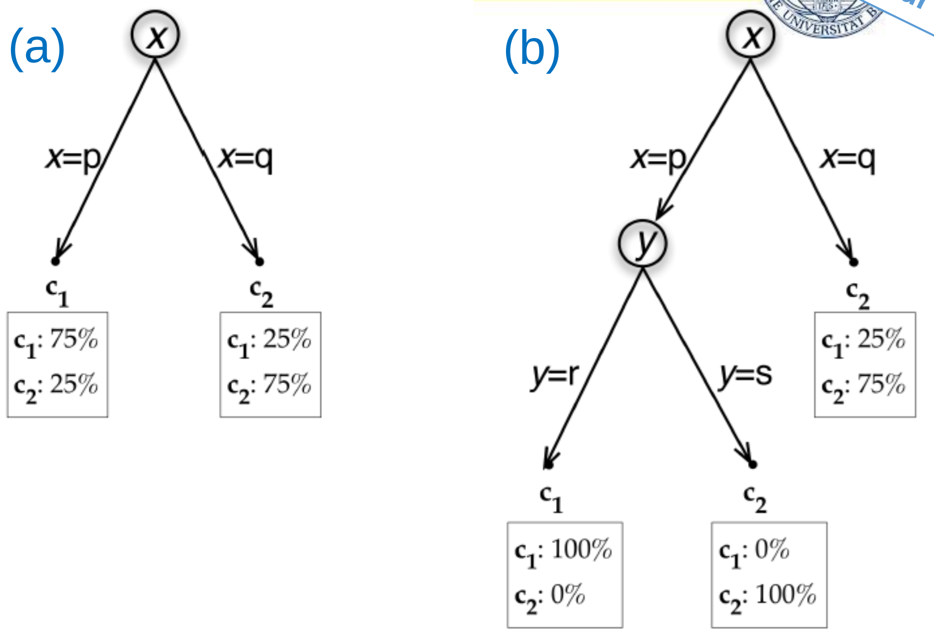
Instance	x	y	Class
1	p	r	c_1
2	p	r	c_1
3	p	r	c_1
4	q	s	c_1
5	p	s	c_2
6	q	r	c_2
7	q	s	c_2
8	q	r	c_2

Classes c_1 and c_2, attributes x and y

An evenly balanced population of examples

x has two values, p and q, and y has two values, r and s

- $x = p$ occurs 75% of the time in class c_1 examples and in 25% of c_2 examples
  ➔ x provides some prediction of that class
- Both of y's values occur in both classes nearly equally
  ➔ y has little predictive value
- The instances in the domain are difficult to separate, with only x providing some predictive leverage (75% accuracy)



- Small training set of examples

  A tree learner would split on x and produce a tree (a) with error 25%

  In this particular dataset, y's values of r and s are not evenly split between the classes, so y seems to provide some predictness

  Tree induction would achieve information gain by splitting on y's values and create tree (b)

- Tree (b) performs better than (a)

  Because $y = r$ purely by chance correlates with class c_1 in this data sample

  The extra branch in (b) is not extraneous, it is **harmful**!
  The spurios $y = s$ branch predicts c_2, which is wrong.

This phenomenon is not particular to decision trees

It is also not because of atypical training data

There is **no general analytic** way to avoid overfitting

A general method for avoiding overfitting

Nested holdout testing

Freie Universität

Please read on your own

How to estimate the generalization performance of models with different complexities?

Test data should be strictly independent of model building.

Real world data
A B A A A B A A B B A A

(Training | Testing) | Validation
(A B A A | A B A A) | B B A A

Nested holdout testing

1. Split the training data set into a training subset and a testing subset
2. Build models on the training subset (**sub-training**) and pick the best model based on the testing subset (**validation**)
3. Validation set is separate from final test set

Ref.

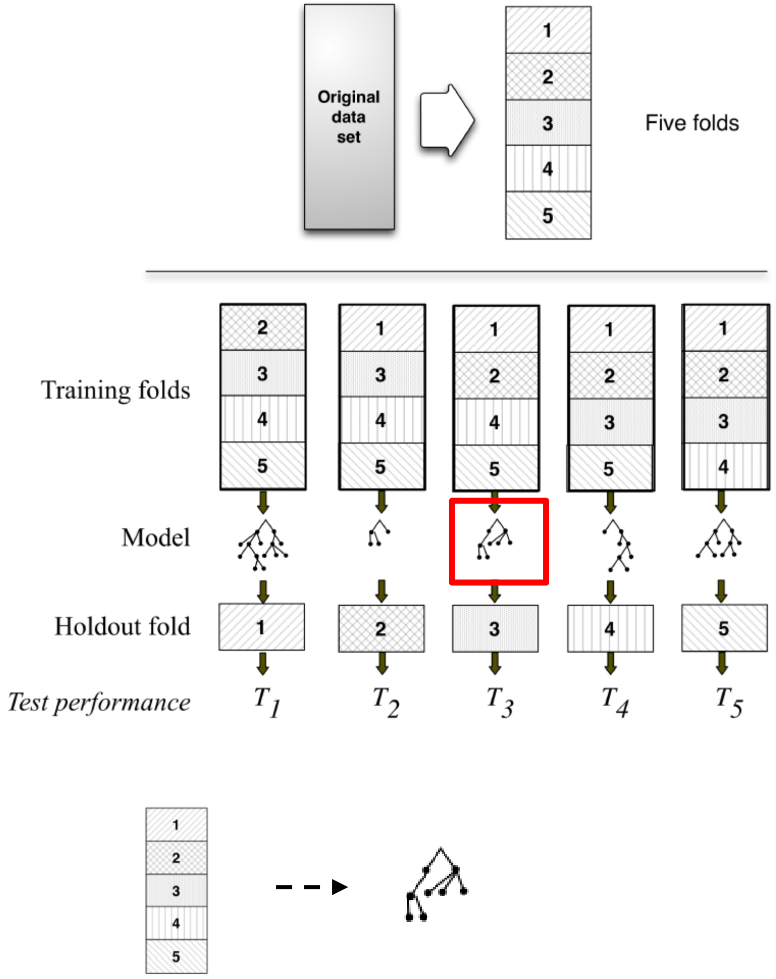
A general method for avoiding overfitting

Please read on your own

1. Use the **sub-training/validation split** to pick the best complexity without tainting the set
2. Build a model of this best complexity on the **entire training set**

Example for classification trees

- Induce trees of many complexities from sub-training set
- Estimate generalization performance from validation set (e.g., the best model has a complexity of 122 nodes)
- Estimate the actual generalization performance on final holdout set
- For the given complexity, induce a new tree with 122 nodes from the original training set



Ref.