



Ostbayerische Technische Hochschule  
Amberg-Weiden

# Machine Learning

Prof. Dr. Fabian Brunner

<fa.brunner@oth-aw.de>

Amberg, 12. Dezember 2023

## Thema heute: Ensemble-Methoden

- Grundgedanke von Ensemble-Methoden
- Methoden zur Konstruktion von Ensembles
- Bagging
- Boosting
- AdaBoost
- RandomForest
- GradientBoosting
- Übung: Ensemble-Methoden in Scikit-learn

## Grundgedanke:

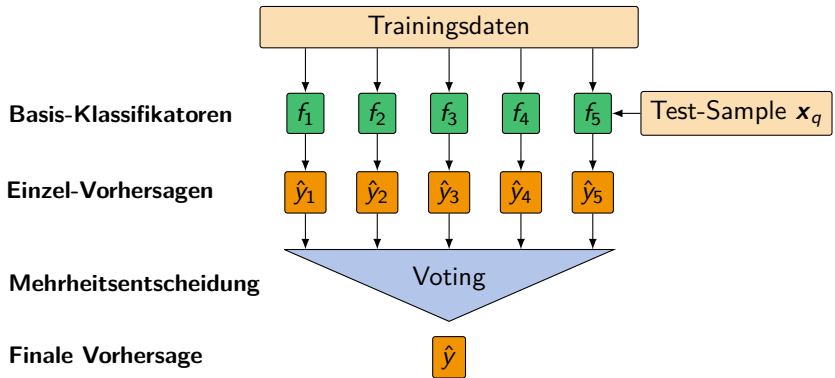
- Reduktion der Häufigkeit von Fehlurteilen durch Bilden einer „Jury von Experten“ und Abstimmung über die richtige Vorhersage
- Dazu: Kombination mehrerer (Basis-)Modelle zu einem „Ensemble“
- Prognostiziere ein neues Sample durch geeignete **Aggregation** der Vorhersagen der Basis-Modelle.
- Geeignet für Regression (Aggregation durch Mittelwertbildung) und Klassifikation (Aggregation durch Voting, s. gleich). Wir beschränken uns im Folgenden auf die Klassifikation.

## Mögliche Ziele:

- Genauigkeit erhöhen
- Verzerrung reduzieren
- Varianz reduzieren
- Effizienz steigern

## Idee:

- Trainiere mehrere separate Klassifikatoren
- Lasse die Mehrheit entscheiden, welcher Klasse ein gegebenes Sample zugeordnet wird.



## Binomialverteilung

- Zufallsexperiment mit zwei möglichen Ausgängen: 1 („Erfolg“) und 0 („Misserfolg“).
- Das Experiment wird  $n$ -mal unabhängig voneinander ausgeführt.
- Sei  $X$  die Zufallsvariable, die die Anzahl der Erfolge misst. Dann ist  $X$  binomialverteilt:

$$P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k} .$$

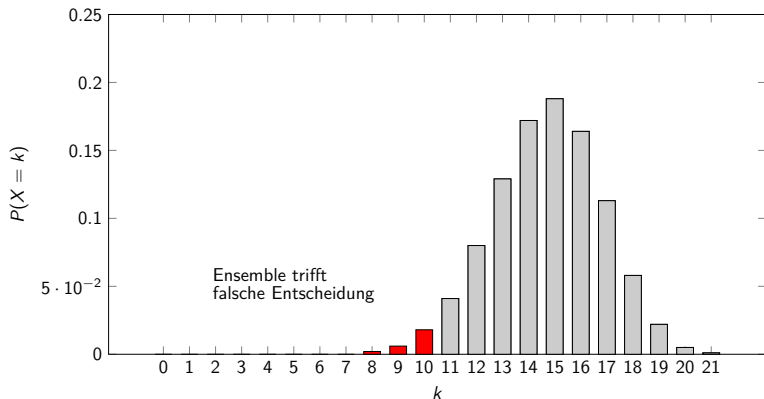
## Warum funktionieren Ensemble-Methoden?

- Betrachte Zusammenschluss von  $N = 21$  Klassifikatoren zu einem Ensemble, die per Mehrheitsentscheidung binär klassifizieren.
- Stark vereinfachende Annahme: Die Klassifikatoren sind unabhängig und jeder klassifiziert mit  $p = 0.7$  korrekt.
- Wahrscheinlichkeit, dass das Ensemble korrekt klassifiziert:

$$\sum_{i=11}^{21} \binom{21}{i} p^i (1 - p)^{21-i} = 0.97 .$$

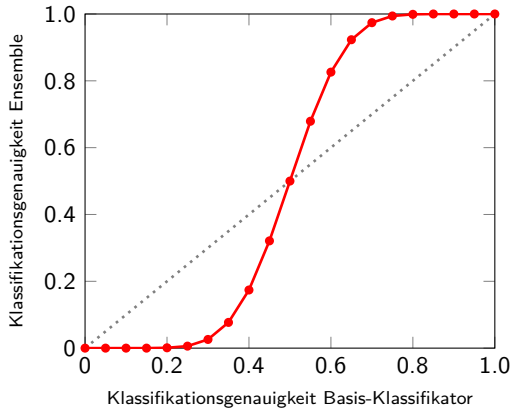
Einzelne Modelle sollten divers (unabhängig)

Sei  $X$  die Zufallsvariable, die angibt, wie viele der 21 Klassifikatoren richtig liegen. Unter der Annahme, dass sie unabhängig sind und eine Accuracy von  $p = 0.7$  haben, ist  $X$  binomialverteilt. Die folgende Grafik zeigt die Wahrscheinlichkeitsfunktion von  $X$ :



# Ensemble vs. Basis-Klassifikatoren

Solange der Basis-Klassifikator besser ist als „zufälliges Raten“ ( $p = 0.5$ ), ist das Ensemble dem Basis-Klassifikator überlegen:



## Hinreichend guter klassifikator nötig

Schlussfolgerung aus obiger Überlegung:

## Notwendige Bedingung für die Verbesserung der Gesamt-Fehlerrate

1. Alle Basis-Klassifikatoren müssen hinreichend genau sein („Accuracy“).
2. Die einzelnen Basis-Klassifikatoren müssen hinreichend unterschiedlich sein („Diversity“).

Bemerkungen:

- Es gibt verschiedene Maße, um Diversity und Accuracy zu messen (Beispiel Regression: Varianz und Mean Squared Error)
- Es besteht ein Trade-off zwischen Diversity und Accuracy.



- Verwendet man Basis-Klassifikatoren, die die Klassenzuordnung basierend auf einem kontinuierlichen Score treffen (z.B. Logistische Regression), kann man auch diesen für die Klassenzuordnung durch ein Ensemble verwenden:

$$\hat{y} = \arg \max_{j \in \{1, \dots, n\}} \sum_{i=1}^N w_i p_{i,j} ,$$

wobei  $p_{i,j}$  die Klassenwahrscheinlichkeit des  $i$ -ten Klassifikators für die Klasse  $j$  bezeichnet.

- Voraussetzung ist, dass die Scores der Basis-Klassifikatoren kalibriert sind.
- Beispiel: drei binäre Klassifikatoren liefern folgende Klassenwahrscheinlichkeiten für die positive Klasse:

$$p_1 = 0.45 , \quad p_2 = 0.45, \quad p_3 = 0.7 .$$

Welche Klasse würde ein Ensemble-Klassifikator basierend auf Majority-Voting prognostizieren? Welche ein Ensemble-Klassifikator, der auf Soft Voting mit  $w_i = \frac{1}{3}$  basiert?

## Statistische Varianz

- Der Raum möglicher Hypothesen ist zu groß, um anhand der begrenzten Trainingsdaten eine beste Hypothese zu bestimmen.
- Kombination mehrerer Hypothesen reduziert das Risiko, stark daneben zu liegen.

## Berechnungs-Varianz

- Lern-Algorithmen liefern nicht immer eine garantiert beste Hypothese aus einem Raum möglicher Hypothesen.
- Ein Optimierungsverfahren zum Modelltraining findet ggf. nur ein lokales Optimum, aber nicht das globale.
- Durch Kombination mehrerer Hypothesen kann das Risiko reduziert werden, das falsche Optimum gewählt zu haben.

## Darstellungsproblem

- Der Hypothesenraum enthält gar keine guten Approximationen an die „wahre“ Funktion  $f$ .
- Kombination mehrerer Hypothesen kann den Raum darstellbarer Hypothesen erweitern.

Um diverse Basis-Klassifikatoren zu erzeugen, existieren verschiedene Ansätze, die auf verschiedene Klassen von Ensemble-Methoden führen.

## Variieren der Trainingsdaten

- Unterschiedliche Gewichtung der Samples
- Subsampling (Modelltraining auf Teilmengen der Trainingsdatenmenge)

## Manipulieren der Input-Features

- Zufällige Auswahl einer Teilmenge von Features („random subspace“)
- Zufällige Projektion von Features („random feature extraction“)

## Manipulieren des Lernalgorithmus

- Zufällige Initialisierung von Modellparametern
- Unterschiedliche Startkonfigurationen

## Bagging=Bootstrap **A**ggregating

- Bagging ist eine Strategie zur Erzeugung eines Ensembles, welche auf der (zufälligen) Variierung der Trainingsdaten basiert.
- Die Klassifikatoren des Ensembles werden gewonnen, indem ein Lernverfahren auf unterschiedlichen Trainingsdatensätzen angewendet wird. Die Trainingsdatensätze werden durch wiederholtes **Bootstrapping** gebildet.
- Dieser Prozess kann gut parallelisiert werden.
- Ein neues, unbekanntes Sample wird durch Majority Voting klassifiziert.
- Bagging wird typischerweise eingesetzt, um Varianz zu reduzieren.

Bagging basiert auf der Bildung zufälliger Trainingsdatensätze durch wiederholtes **Bootstrapping**:

- Gewinnung mehrerer Samples aus der gegebenen Trainingsdatenmenge durch Ziehen mit Zurücklegen
- Jedes Sample hat die gleiche Größe wie die ursprüngliche Trainingsdatenmenge
- Ein Bootstrap-Sample enthält ca. 63% der Elemente der Trainingsdatenmenge (einige mehrfach, etwa 37% gar nicht).
- Begründung: ist  $m$  die Länge des Trainingsdatensatzes, so wird ein einzelnes Trainingsobjekt mit Wahrscheinlichkeit  $1 - \frac{1}{m}$  bei einmaligem Ziehen mit Zurücklegen nicht ausgewählt. Bei  $m$ -fachem Ziehen ist es also mit Wahrscheinlichkeit  $(1 - \frac{1}{m})^m$  nicht im Bootstrap-Sample vertreten. Es gilt  $\lim_{m \rightarrow \infty} (1 - \frac{1}{m})^m = e^{-1} \approx 0.368$ .

# Ensemble-Learning durch Bagging

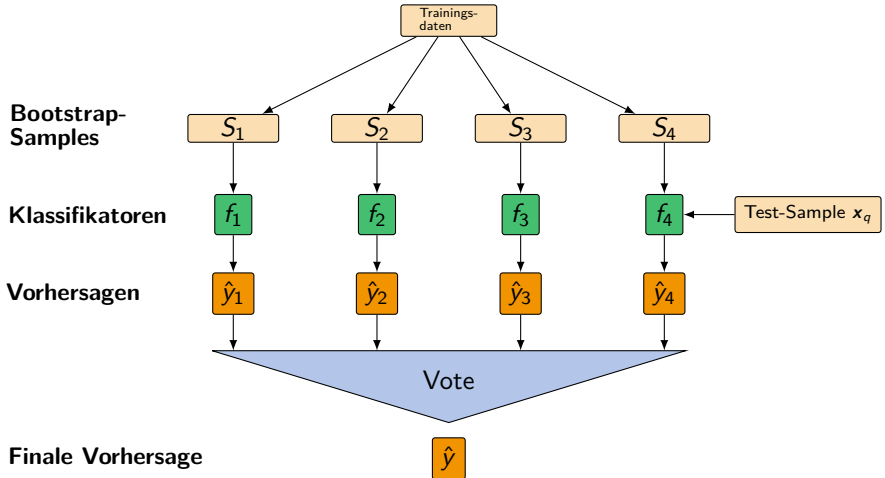
---

## Algorithm 1 Bagging

---

- 1: Sei  $N$  die Anzahl der Bootstrap Samples und  $m$  die Anzahl der Trainings-Samples
  - 2: **for**  $i = 1$  to  $N$  **do**
  - 3:   Ziehe ein Bootstrap-Sample  $S$  der Länge  $m$
  - 4:   Trainiere ein Modell  $f_i$  auf  $S$
  - 5: **end for**
  - 6:  $\hat{y} = \text{mode}\{f_1(\mathbf{x}_q), \dots, f_N(\mathbf{x}_q)\}$
-

# Prinzip beim Bagging



## Wie funktioniert Boosting?

- Ziel beim Boosting ist es, durch Kombination mehrerer „schwacher Lerner“ (z.B. Decision-Tree-Stumpf) einen „starken Lerner“ zu generieren.
- Es handelt sich um einen iterativen Prozess, bei dem die Klassifikatoren eines Ensembles sequenziell, d.h. abhängig vom Ergebnis des jeweils vorangegangenen Schritts, erzeugt werden.
- In jedem Schritt wird, basierend auf dem aktuellen Modell, ein schwacher Lerner trainiert und (additiv-gewichtet) dem Ensemble hinzugefügt.

## Unterschiede zwischen Bagging und Boosting:

- Bagging: Ein gegebenes Lernverfahren wird **unabhängig** auf verschiedenen Bootstrap-Samples trainiert.
- Boosting: mehrere Klassifikatoren werden **sequenziell** auf der gewichteten Trainingsdatenmenge trainiert.

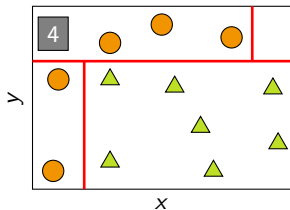
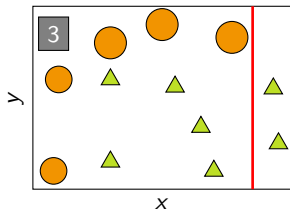
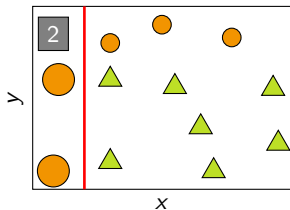
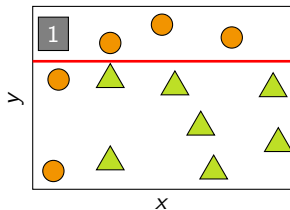


# AdaBoost-Verfahren

- Beim AdaBoost-Verfahren werden die Klassifikatoren des Ensembles erhalten, indem ein schwacher Lerner mehrfach auf den mit Gewichten versehenen Trainingsdatensatz angewendet wird.
- Die Gewichte des Trainingsdatensatzes werden in jedem Schritt basierend auf den Klassifikationsfehlern des vorherigen Schritts neu berechnet.
- Ist  $w(\mathbf{x}^{(i)})$  das Gewicht des  $i$ -ten Trainingsobjekts, dann bedeutet das, dass dieses  $w$ -mal zählt. Dies kann beispielsweise durch Resampling der Trainingsdatenmenge erreicht werden, oder indem der Beitrag des Objekts zu einer Fehlermetrik mit  $w(\mathbf{x}^{(i)})$  gewichtet wird.
- Objekte, die schwierig zu klassifizieren sind bzw. falsch klassifiziert wurden, erhalten im nächsten Schritt ein höheres Gewicht.
- AdaBoost ist typischerweise robust gegen Overfitting.

# Beispiel zum AdaBoost-Verfahren

- Basis-Klassifikator: Decision Tree mit einem Knoten.
- Die ersten drei Abbildungen zeigen die ersten drei Iterationen des AdaBoost-Verfahrens, das vierte den resultierenden Ensemble-Lerner.



# AdaBoost - Pseudocode

Pseudocode für ein binäres Klassifikationsproblem mit einem Trainingsdatensatz  $\mathbf{X} = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})\}$  und Labels  $y^{(i)} \in \{-1, 1\}$

---

## Algorithm 2 AdaBoost

---

- 1: Sei  $N$  die Anzahl der AdaBoost-Iterationen
- 2: Seien  $\mathbf{w}_1 = (w_1^{(1)}, \dots, w_1^{(m)})$  mit  $w_1^{(i)} = \frac{1}{m}$  die Initialgewichte
- 3: **for**  $k = 1$  to  $N$  **do**
- 4:   Trainiere einen schwachen Klassifikator  $f_k = f_k(\mathbf{X}, \mathbf{y}, \mathbf{w}_k)$
- 5:   Bestimme mit diesem die Labels auf  $\mathbf{X}$ :  $\hat{\mathbf{y}}_k := \text{predict}(f_k, \mathbf{X})$
- 6:   Berechne den gewichteten Fehler  $\varepsilon_k = \sum_{i=1}^m w_k^{(i)} \cdot \mathbf{1}_{\hat{y}_k^{(i)} \neq y^{(i)}}$
- 7:   Setze  $\alpha_k := \frac{1}{2} \log \left( \frac{1 - \varepsilon_k}{\varepsilon_k} \right)$
- 8:   Update der Gewichte:
$$w_{k+1}^{(i)} := w_k^{(i)} \cdot \begin{cases} e^{-\alpha_k} & \text{falls } \hat{y}_k^{(i)} = y^{(i)} \\ e^{\alpha_k} & \text{falls } \hat{y}_k^{(i)} \neq y^{(i)} \end{cases}$$
- 9:   Normierung der Gewichte  $w_{k+1}$ , sodass  $\sum_{i=1}^m w_{k+1}^{(i)} = 1$  gilt.
- 10: **end for**
- 11: Finale Prognose:  $\hat{y}(\mathbf{x}_q) = \text{sign} \left( \sum_{k=1}^N \alpha_k \cdot \text{predict}(f_k, \mathbf{x}_q) \right)$

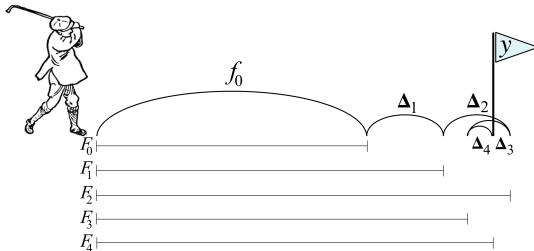
```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
import pandas as pd
import numpy as np

X, y = ... #some labeled training data and labels
clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1),
                        n_estimators=50)

clf.fit(X, y)
clf.predict(...)
```

## Gradient Boosting = Gradient Descent + Boosting

- Ziel ist es, ein Ensemble  $f(x) = \sum_{k=1}^N h_k(x)$  sequenziell zu bestimmen.
- Im Schritt  $k$  wird ein (schwaches) Lernverfahren  $h_k$  trainiert, das die Fehler des bisher gewonnenen Ensembles kompensieren soll. Dieses wird anschließend zum bisher berechneten Modell (gewichtet) addiert.
- Im Gegensatz zu AdaBoost werden also nicht die Gewichte der Trainingsdaten verändert, sondern Modelle auf den Residuen trainiert.



Quelle: <https://explained.ai/gradient-boosting/L2-loss.html>

Gegeben sei ein Trainingsdatensatz bestehend aus drei Objekten

$$(x^{(1)}, y^{(1)}) , (x^{(2)}, y^{(2)}) , (x^{(3)}, y^{(3)}) .$$

Sei  $f$  ein einfaches univariates Regressionsmodell, das auf den Trainingsdaten bestimmt wurde. Im ersten Schritt des Gradient Boosting wird ein Modell  $h$  bestimmt, sodass bestenfalls

$$f(x^{(1)}) + h(x^{(1)}) = y^{(1)} ,$$

$$f(x^{(2)}) + h(x^{(2)}) = y^{(2)} ,$$

$$f(x^{(3)}) + h(x^{(3)}) = y^{(3)}$$

gilt, bzw., äquivalent dazu,

$$h(x^{(1)}) = y^{(1)} - f(x^{(1)}) ,$$

$$h(x^{(2)}) = y^{(2)} - f(x^{(2)}) ,$$

$$h(x^{(3)}) = y^{(3)} - f(x^{(3)}) .$$

Das Modell  $h$  verwendet die Residuen als  $y$ -Werte. Es wird anschließend zu  $f$  addiert. Dieser Schritt wird beim Gradient Boosting mehrfach wiederholt.

- Ist  $f_k$  das im  $k$ -ten Iterationsschritt berechnete Modell, dann sucht man beim Gradient Boosting ein Regressionsmodell  $h_k$ , sodass bestenfalls

$$f_{k+1}(\mathbf{x}^{(i)}) := f_k(\mathbf{x}^{(i)}) + h_k(\mathbf{x}^{(i)}) = y^{(i)}, \quad i = 1, \dots, m,$$

bzw.

$$h_k(\mathbf{x}^{(i)}) = \underbrace{y^{(i)} - f_k(\mathbf{x}^{(i)})}_{=: r^{(i)}}.$$

- Dazu wird ein Regressionsmodell  $h_k$  auf dem **Residuum** trainiert, d.h. auf den Trainingsdaten  $(\mathbf{x}^{(1)}, r^{(1)}), \dots, (\mathbf{x}^{(m)}, r^{(m)})$ .
- Das Residuum  $r^{(j)}$  ist die negative partielle Ableitung des Mean Squared Errors

$$MSE(f_k) = \frac{1}{2} \sum_{i=1}^m (y^{(i)} - f_k(\mathbf{x}^{(i)}))^2$$

nach  $f_k(\mathbf{x}^{(j)})$ . In diesem Sinn lässt sich Gradient Boosting als Gradientenverfahren auffassen. Das Update  $f_{k+1} := f_k + h_k$  entspricht einem Schritt des Gradientenverfahrens.

- Gradient Boosting lässt sich als Gradientenverfahren in einem Funktionenraum interpretieren (die Unbekannten und die Updates sind Funktionen).
- Das oben beschriebene Vorgehen lässt sich auf andere Fehlermaße (zusammen mit dem jeweiligen Gradienten) übertragen, um andere Gradient Boosting-Verfahren zu konstruieren.
- Bei Klassifikationsproblemen wird Gradient Boosting häufig mit Decision Trees als schwachen Lernern verwendet, z.B. Decision-Trees mit nur einem Knoten.
- Um Overfitting zu vermeiden, sollte die Anzahl der Schritte begrenzt werden und Regularisierung angewendet werden.



## Eigenschaften von Random Forests

- Random Forests sind für Klassifikation und Regression gleichermaßen geeignet.
- Sie sind in der Lage, nichtlineare Beziehungen zu modellieren.
- Sie liefern oft schon bei kleinen Datensätze gute Ergebnisse und sind in der Lage, mit hochdimensionalen Daten und korrelierten Features umzugehen.
- Das Modelltraining ist parallelisierbar.
- Random Forests benötigen wenig Tuning.
- Sie liefern eine **Feature Importance** und können damit auch zur Feature Selection verwendet werden.



# Random Forest

## Random Forests als Ensemble Lerner

- Kombiniere mehrere Decision Trees zu einem „forest“.
- Aggregation der Ergebnisse z.B. durch Mittelung (Regression) oder Majority Voting (Klassifikation).
- Erzeuge die einzelnen Bäume durch
  1. Bootstrap Sampling zum Training eines einzelnen Baums
  2. Auswahl einer zufälligen Teilmenge der Features in jedem Knoten eines jeden Baums
  3. Verzicht auf Pruning
- Die resultierenden Bäume sind weniger stark korreliert als beim reinen Bagging.
- Das Modelltraining und die Klassifikation können parallel erfolgen.

- Random Forests liefern auch eine Gewichtung („importance“) der Features.
- Dabei wird jedem Feature  $X_j$  die mittlere Reduktion des Maßes für die Unreinheit (z.B. Entropie, Gini-Index) zugeordnet, die erzielt wurde, wenn  $X_j$  als Split-Variable verwendet wurde (gewichtet mit der Anzahl an Samples, die den Knoten erreicht haben):

## Mean Decrease in Impurity (MDI)

$$\text{Importance}(X_j) = \frac{1}{N_T} \sum_T \sum_{\substack{t \in T : \\ v(s_t) = X_j}} p(t) \Delta i(s_t, t) ,$$

- Notation:
  - $T$ : Entscheidungsbaum
  - $t \in T$ : Knoten des Baums  $T$
  - $N_T$ : Anzahl der Bäume des Random Forests
  - $\Delta i(\cdot)$ : Reduktion des Impurity Measures
  - $p(t)$ : Anteil der Daten, der den Knoten  $t$  erreicht
  - $v(s_t)$ : Variable, die im Split  $s_t$  verwendet wird ist.

# Vor- und Nachteile von Random Forest

## Vorteile

- Gute Vorhersagekraft
- Wenig Tuning erforderlich
- Robustheit gegen Overfitting
- keine Skalierung der Daten notwendig
- Feature-Importance wird mitgeliefert
- Parallelisierbarkeit

## Nachteile

- Höherer Rechenaufwand als bei Decision Trees
- Geringere Interpretierbarkeit

# Zusammenfassung

- Beim Ensemble-Lernen werden mehrere Basis-Klassifikatoren kombiniert, um die Qualität der Vorhersage zu verbessern.
- Um dieses Ziel zu erreichen, sollten die Basis-Klassifikatoren akkurat und divers sein.
- Diverse Basis-Klassifikationen können durch verschiedene Strategien gewonnen werden, z.B. zufällige Variation der Trainingsdaten (Bagging) oder der Modellparameter oder der Features.
- Bagging: Variation der Trainingsdaten durch Bootstrap-Sampling
- Beispiel für Bagging: Random Forest
- Boosting: Sequentielle Generierung und additive Kombination mehrerer schwacher Lerner
- Beispiele für Boosting: AdaBoost und Gradient Boosting