Healthcare Data Analytics

Decision Tress

Dr. Michael Strobel

09.05.2022

Inhalt

Letzte Woche

- Polynomielle Regression
- Overfitting
- Underfitting
- Regularisierung

Diese Woche

- Wiederholung Regularisierung
- Decision Trees

Wiederholung Definition: L_2 Regularisierung

Erinnerung: Minimierungsproblem

$$\hat{\alpha} = \min_{\alpha \in \mathbb{R}^n} \|X\alpha - \hat{y}\|_2^2$$

Definition: L_2 Minimierungsproblem

Sei $\lambda \geq 0$, dann definieren wir das L_2 *regulierte Minimierungsproblem*

$$\hat{\alpha} = \min_{\alpha \in \mathbb{R}^n} \|X\alpha - \hat{y}\|_2^2 + \lambda \|\alpha\|_2^2$$

Weitere Namen sind auch *Tikhonov-* oder *Ridge-* Regularisierung.

Geschlossene Lösung: L_2 Minimierungsproblem

Sei $\lambda \geq 0$ und gegeben L_2 *regulierte Minimierungsproblem*

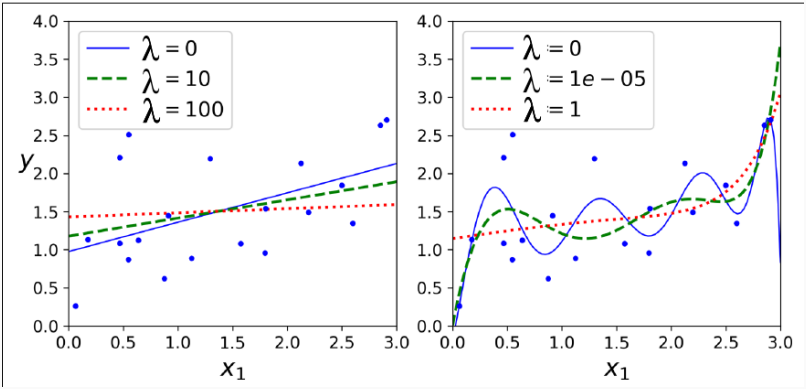
$$\hat{\alpha} = \min_{\alpha \in \mathbb{R}^n} \|X\alpha - \hat{y}\|_2^2 + \lambda \|\alpha\|_2^2$$

Dann ist die geschlossene optimale Lösung des Minimierungsproblems gegeben durch

$$\hat{\alpha} = (X^T X + \mathrm{Id}\,\lambda)^{-1} X^T \hat{y}$$

Visualisierung: L_2 Minimierungsproblem

Visualisierung der L_2 Regularisierung mit verschiedenen Parametern



Definition: L_1 Minimierungsproblem

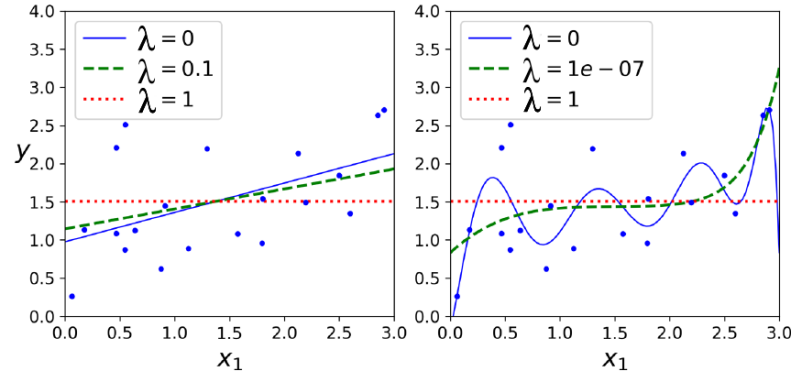
Sei $\lambda \geq 0$, dann definieren wir das L_1 *regulierte Minimierungsproblem*

$$\hat{\alpha} = \min_{\alpha \in \mathbb{R}^n} \|X\alpha - \hat{y}\|_2^2 + \lambda \|\alpha\|_1$$

Ein weiterer gebräuchlicher Name ist *Lasso* (Least Absolute Shrinkage and Selection Operator Regression).

Visualisierung: L_1 Minimierungsproblem

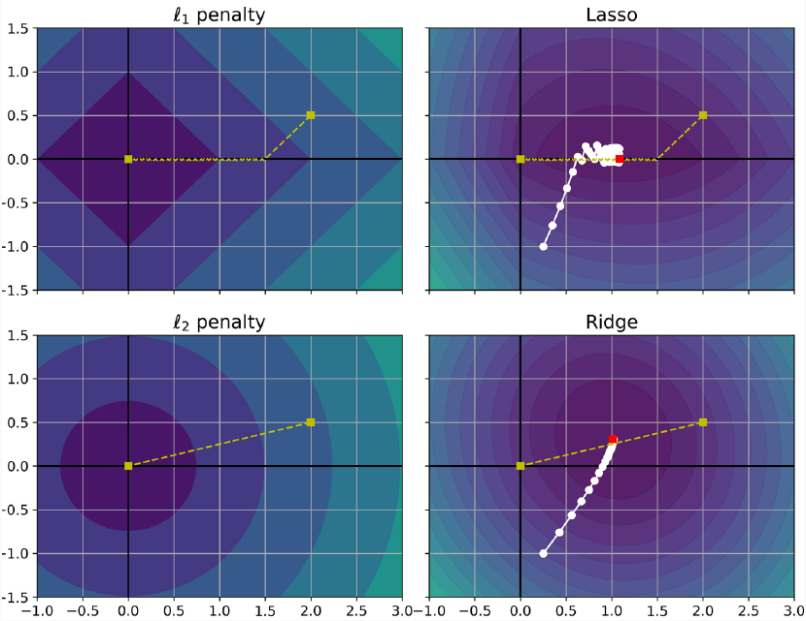
Visualisierung der L_1 Regularisierung mit verschiedenen Parametern



L_1 vs L_2 Regularisierung

Generell neigt die L_1 Regularisierung dazu möglichst viele Koeffizienten auf 0 zu drücken und führt damit zu einem *dünnbesetzten (engl. sparse)* Modell. Die L_2 Regularisierung wiederum neigt dazu die Koeffizienten stärker gleichmäßig zu minimieren was zu mehr gleichverteilten Koeffizienten führt.

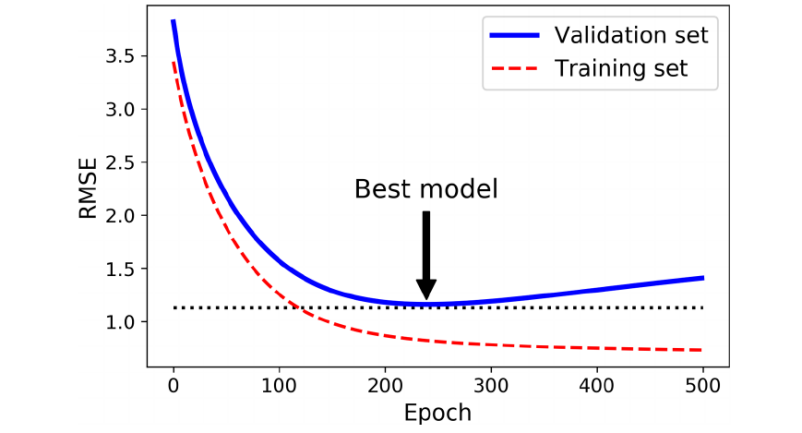
Visualisierung: L_1 vs L_2 Regularisierung



Géron, Aurélien. "Hands-on machine learning with scikit-learn and tensorflow"

Early Stopping

Ein weiterer Weg das Modell zu regularisieren ist *early stopping*. Hierbei stoppt man den Gradientenabstieg sobald der Fehler in der **Testmenge** nicht mehr sinkt beziehungsweise sogar steigt.

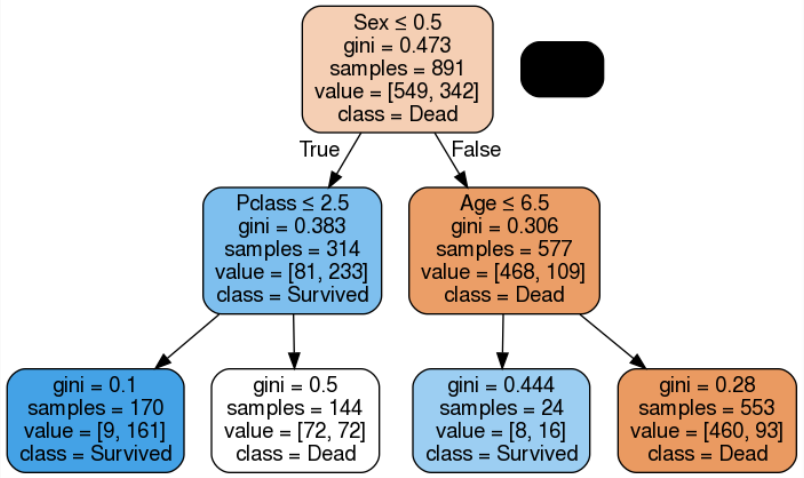


Géron, Aurélien. "Hands-on machine learning with scikit-learn and tensorflow"

Decision Trees

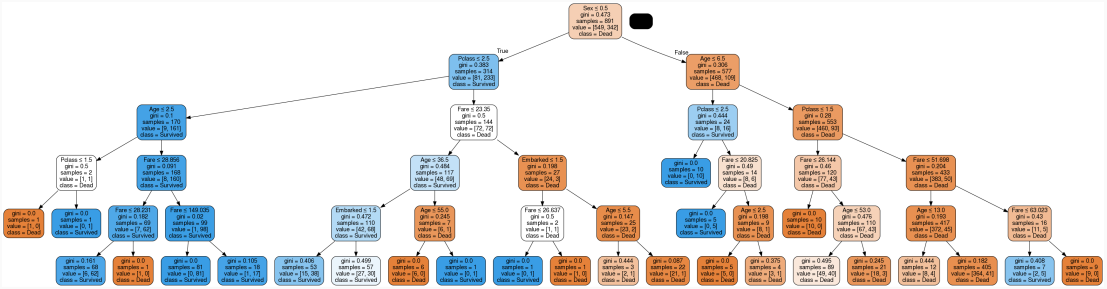
- Bis jetzt haben wir Machine Learning Methoden nur angewandt
- Jetzt gehen wir zumindest für eine Methoden ins Detail
- Eine sehr bekannte Methode sind *Decision Trees*
- Decision Trees eignen sich für Regression und auch für Mehrklassen-Klassifikation

Decision Tree auf dem Titanic Dataset



Decision Tree der Tiefe 2

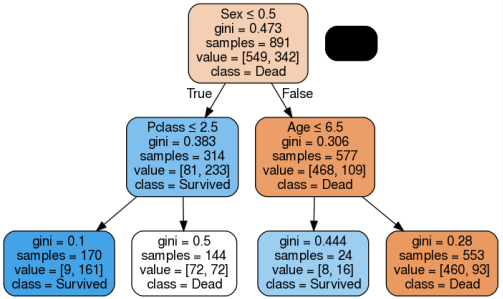
Decision Trees - Visualisierung 2



Decision Tree der Tiefe 5

Decision Trees - Wie liest man einen Decision Tree

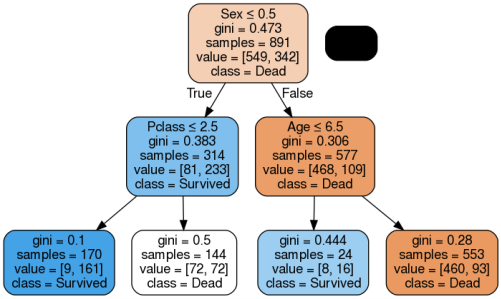
- Man starte an der Wurzel
- Die erste Zeile zeigt die **Ja oder Nein Frage**
  - links der der Unterbaum der die Frage mit **ja** beantwortet
  - rechts der der Unterbaum der die Frage mit **nein** beantwortet
- Die Blätter enthalten die jeweilige Klasse



Decision Tree - weitere Daten

Der Decision Tree enthält weitere Informationen

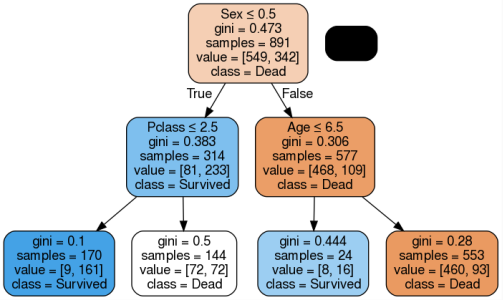
- wie viele Testdaten wurden dem Blatt zugeordnet (samples)
- welcher Klasse wurden diese zugeordnet (value)
- mit diesen Daten können Wahrscheinlichkeiten berechnet werden (klassisches Urnenmodell)



Wie beurteilt man ob der Baum gute Entscheidungen trifft?

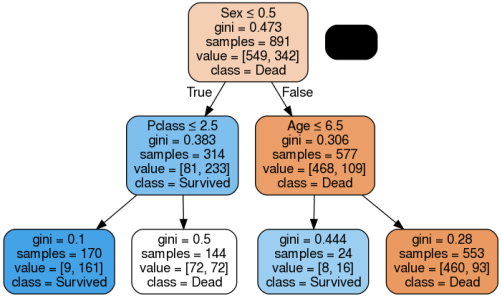
Gini Impurity

- Über die Anzahl aller Testdaten und Samples im Knoten können wir eine Klassen-Wahrscheinlichkeit $p_{i,k}$ berechnen, genauer $p_{i,k}$ ist das Verhältnis der Klasse k unter allen Traningsdaten im Knoten i
- Wir definieren die *Gini Impurity* im Knoten i als $G_i := 1 - \sum_{k=1}^{n} p_{i,k}^2$
- $G = 0$ heißt, dass alle Testdaten in die gleiche Kategorie fallen, also *pure* sind.
- $G = 0.5$ heißt, dass die Angabe essentiell geraten it.



Gini Impurity - Beispiel

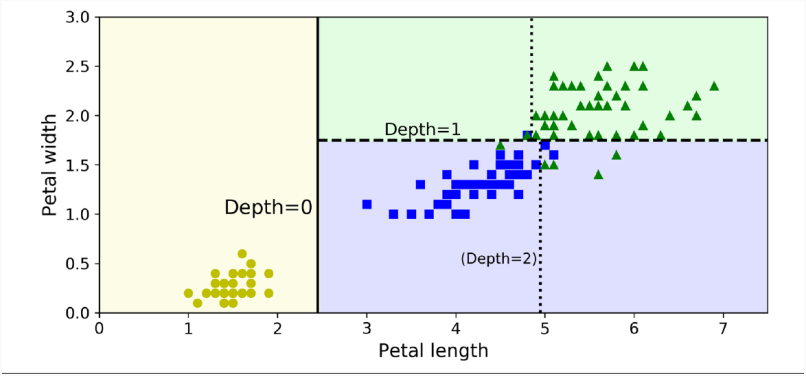
- Beispiel am Blatt links unten: $G = 1 - (\frac{9}{170})^2 - (\frac{161}{170})^2 = 0.1$
- Ein Wert von 0.1 ist gut, der Decision Tree ist sich sicher
- Der Nachbarknoten hat einen Gini Wert von 0.5, d.h. er rät
- Das finden des optimalen Baumes ist NP Schwer, es gibt polynomielle Algorithmen die gute Lösungen finden



Entscheidungsgrenzen

Da der Baum Stückweise lineare Funktionen repräsentieren können diese gut visualisiert werden

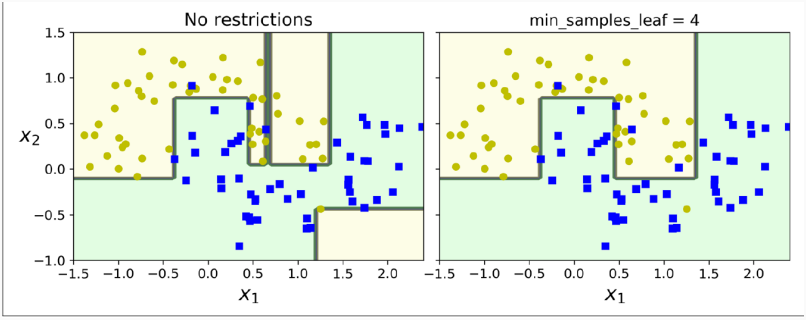
Hier ein Beispiel mit drei Klassen: Iris Dataset



Géron, Aurélien. “Hands-on machine learning with scikit-learn and tensorflow”

Decision Trees – Overfitting

Decision Tree neigen zu extremen Overfitting wenn der Parameterraum nicht begrenzt wird



Géron, Aurélien. "Hands-on machine learning with scikit-learn and tensorflow"

Decision Trees – Regularisierung

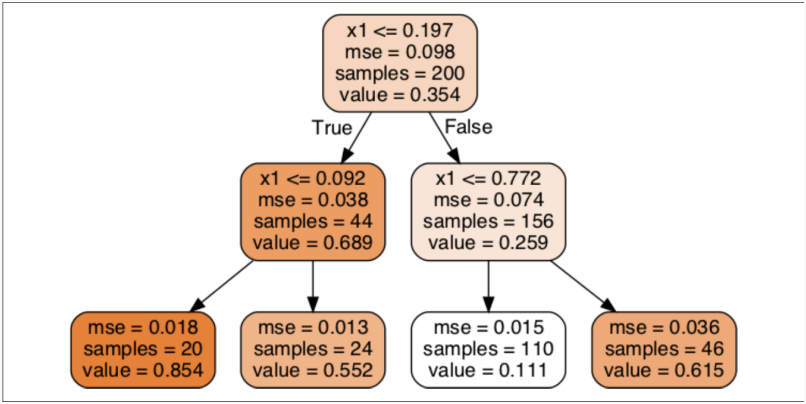
Decision Tree können über verschiedene Parameter regularisiert werden

- Maximale Tiefe des Baums (max_depth)
- Minimale Anzahl der Beobachtungseinheiten um einen Knoten aufzuteilen (min_samples_split)
- Minimale Anzahl der Beobachtungseinheiten in einem Blatt (min_samples_leaf)
- Maximale Anzahl der Features (max_features)
- ...

Der Decision Tree kann auch zur Regression verwendet werden

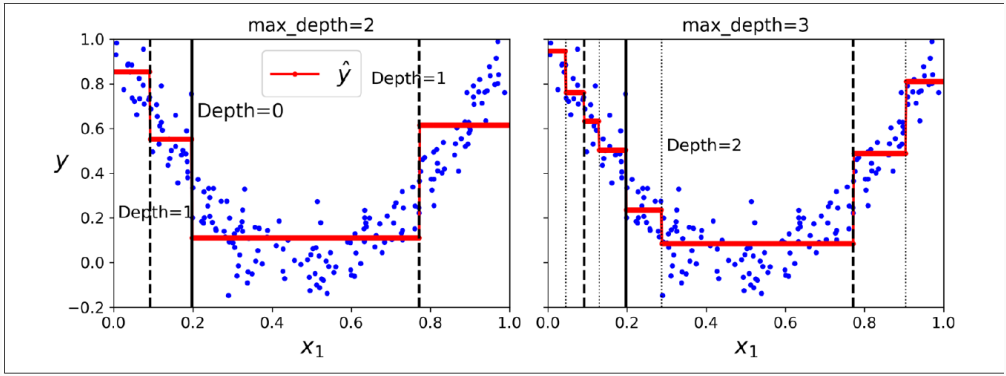
Der Tree zur Regression ähnelt dem der Klassifikation: statt einer Klasse wird ein **Wert** zugeordnet.

Beispiel: Regression auf $f(x) = x^2$ mit etwas Rauschen



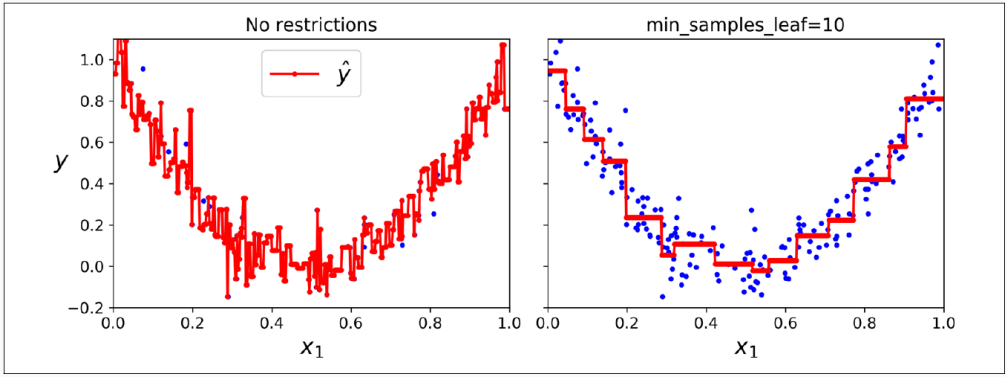
Géron, Aurélien. "Hands-on machine learning with scikit-learn and tensorflow"

Decision Trees – Regression Visualisierung



Géron, Aurélien. "Hands-on machine learning with scikit-learn and tensorflow"

Decision Trees neigen zu extremen Overfitting



Géron, Aurélien. "Hands-on machine learning with scikit-learn and tensorflow"

Decision Trees – Eigenschaften

Decision Trees haben einige interessante Eigenschaften:

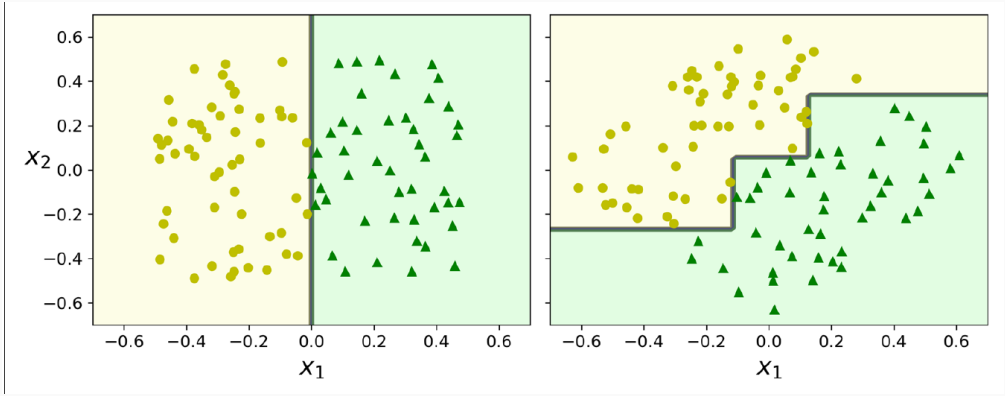
- Features müssen vorher nicht skaliert oder normalisiert werden
- Sie haben eine extreme Anpassungsfähigkeit an die Daten
- Sie können direkt Mehrklassen-Klassifikation leisten

Nachteile sind z.B.:

- Neigen zu starkem Overfitting wenn sie nicht regularisiert werden
- Sind nicht invariant unter Rotation (nächste Folie)

Decision Trees – Instabilität

Das Training von Decision Trees ist nicht *rotationsinvariant* d.h. eine Rotation der Trainingsdaten führt nicht zur gleichen Rotation der Entscheidungsgrenzen



Géron, Aurélien. "Hands-on machine learning with scikit-learn and tensorflow"

Referenzen

- Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media.