TAG 1

Einführung, Regression, Gradient Descent, Feature Preprocessing & Engineering

n|w

INTRO

ÜBER MICH

- Benjamin Meyer (Beni)
- Bachelor in Computer Science an der FHNW
- 1 Jahr Wissenschaftlicher Assistent an der FHNW
- Masterstudent Universität Basel
- keeValue AG: Machine Learning mit kleinen und mittel grossen Datensätzen
- ZHAW im CAI (Center of Artificial Intelligence): In der "Computer Vision, Perception and Cognition Group", mehrheitlich Deep Learning

4 TAGE MACHINE LEARNING

Tag 1	Einführung, Regression, Gradient Descent, Feature Preprocessing & Engineering
Tag 2	Klassifikation, weitere Algorithmen
Tag 3	Clustering, Dimensionality Reduction
Tag 4	Review, Neural Network, Diskussion

PROGRAMMIEREN! PROGRAMMIEREN! PROGRAMMIEREN!

Code-Beispiele	Fertiges minimales Beispiel
Übungen	Gelerntes an einfachem Problemfall selbst anzuwenden
ML-Lab	Gelerntes an echtem Problemfall selbst anwenden

PROGRAMMIEREN! PROGRAMMIEREN! PROGRAMMIEREN!

ÜBER DIE SLIDES

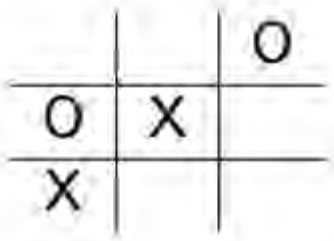
Slide behandelt fortgeschrittenes Thema. Sollte (!) nicht an der Prüfung kommen.

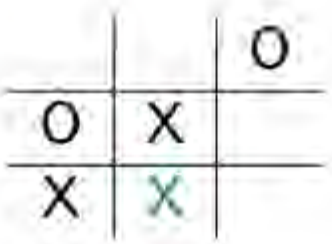
Slide verweist auf Code-Beispiel

https://github.com/benikm91/cas_m
learning-slides

WAS IST MACHINE LEARNING

PROBLEMFÄLLE





- f ist eine TicTacToe-AI die perfekt spielt

Ist das möglich?

PROBLEMFÄLLE

f

- f ist eine Chess-AI die perfekt spielt

approx f

- approx f ist eine Chess-AI die sehr gut spielt

PROBLEMFÄLLE

f

- f ist eine AI die Häuserpreise perfekt vorhersagt

Fläche: 500m²

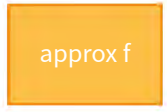
Baujahr: 1998

approx f

100'000$

- approx f ist eine AI die Häuserpreise gut vorhersagt

WAS IST MACHINE LEARNING



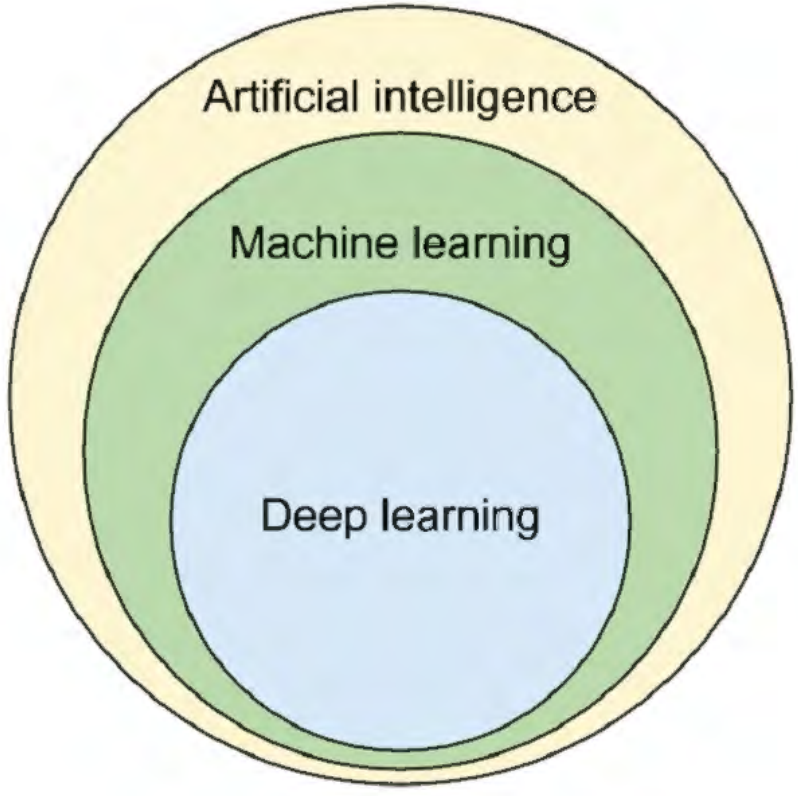
nennen wir ein Modell

Wenn ein Modell für uns Menschen intelligent erscheint, dann sprechen wir von Künstlicher Intelligenz.

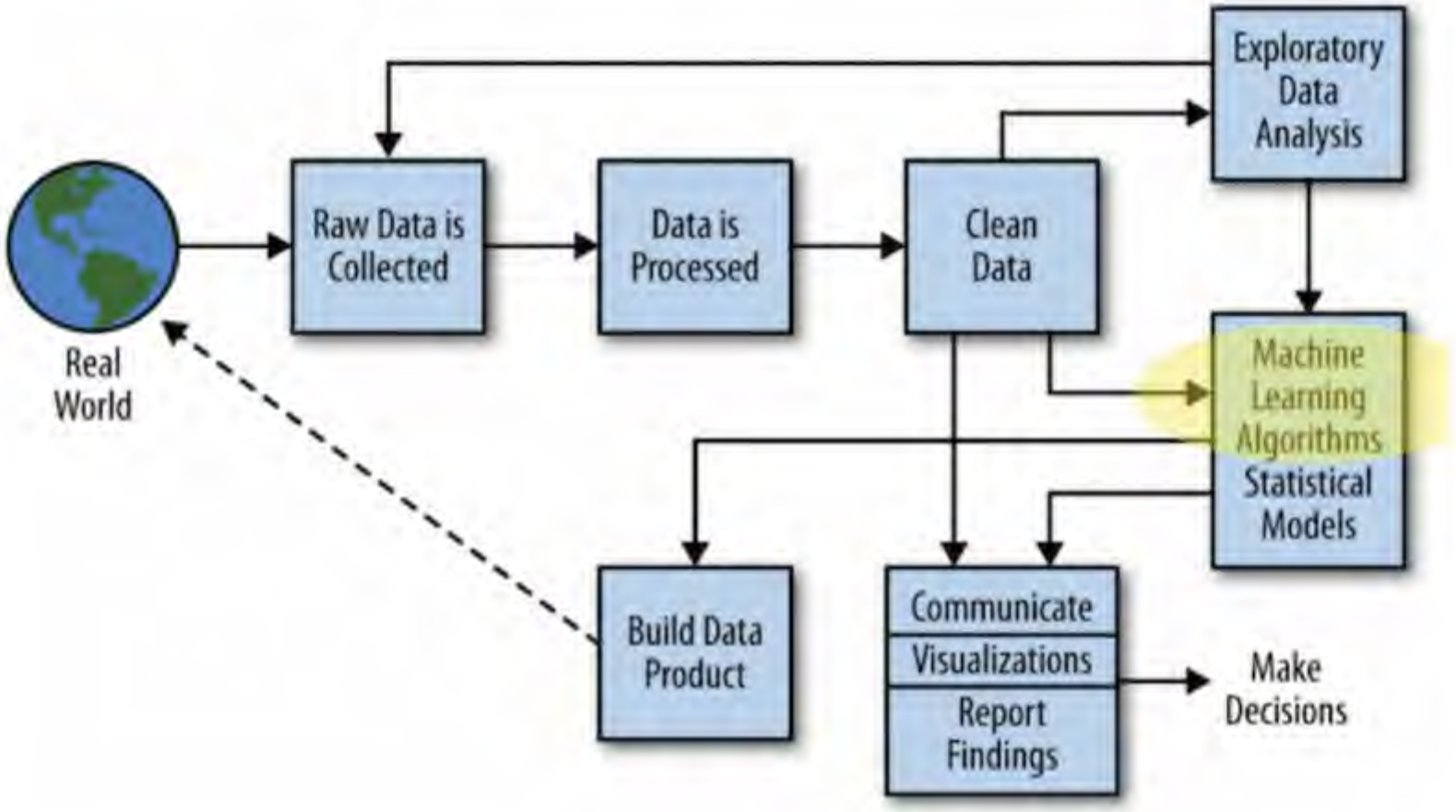
Ein Modell kann programmiert sein (von Experten) oder gelernt (aus Daten).

Ist ein Modell teilweise gelernt sprechen wir von Machine Learning!

WAS IST MACHINE LEARNING

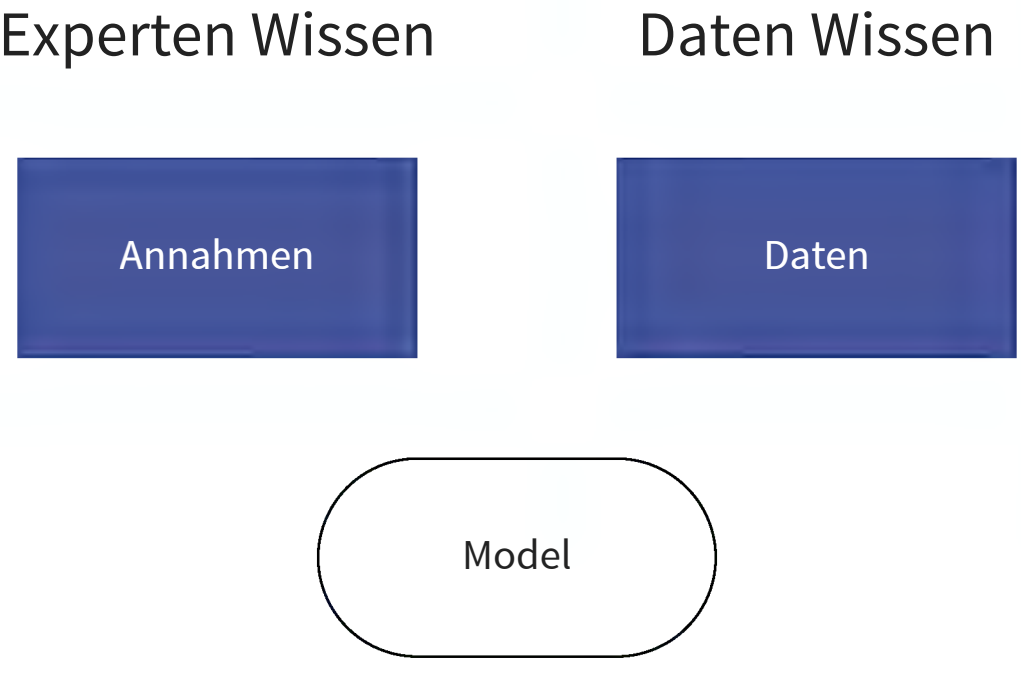


WO IST MACHINE LEARNING



WAS IST "LEARNING" - WISSENS ANSICHT

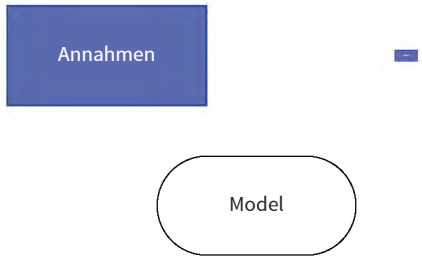
Wissen kann von Experten oder aus Daten kommen



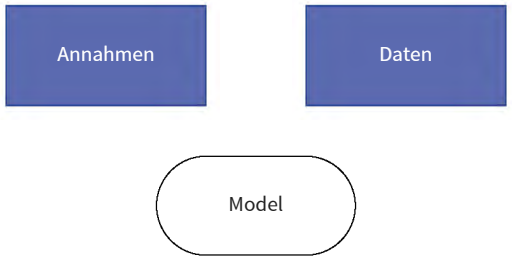
WAS IST "LEARNING" - TRADE-OFF

Woher das Wissen kommt ist ein Trade-Off.

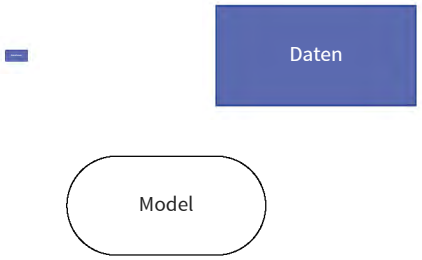
Strong assumptions, Few data



Moderate assumptions, Some data



Weak assumptions, A lot data



WAS IST "LEARNING"

- Wann wird gelernt
  - Offline Learning
  - (Online Learning)
- Wie wird gelernt
  - Supervised Learning
  - Unsupervised Learning
  - (Reinforcement Learning)

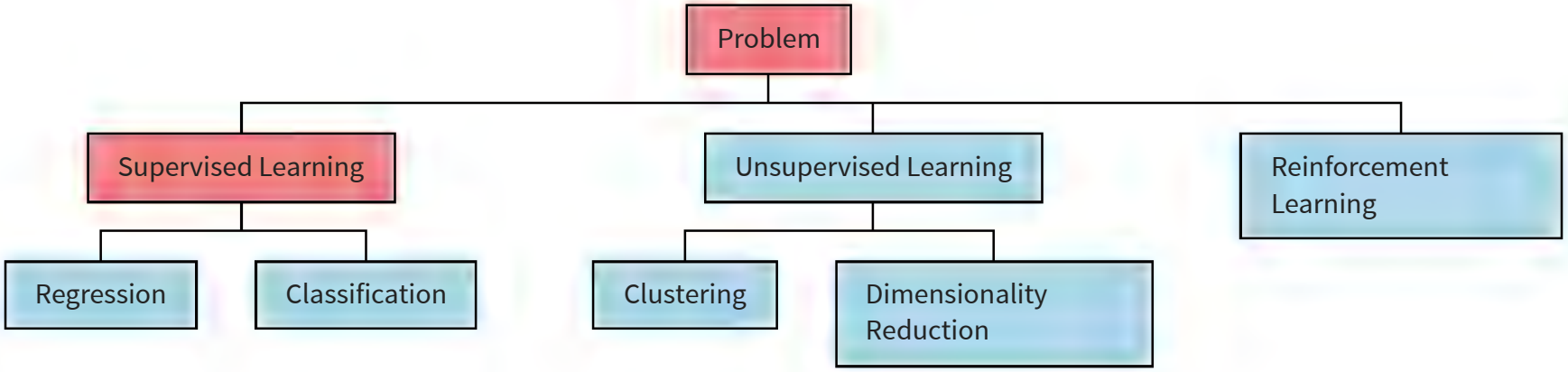
WANN WIRD GELERNT

- Offline Learning
  - Es gibt eine Lernphase, während dieser wird von Daten gelernt.
  - Und es gibt eine Anwendungsphase, während dieser wird nichts mehr gelernt.
- Online Learning
  - Während der Anwendung wird gelernt

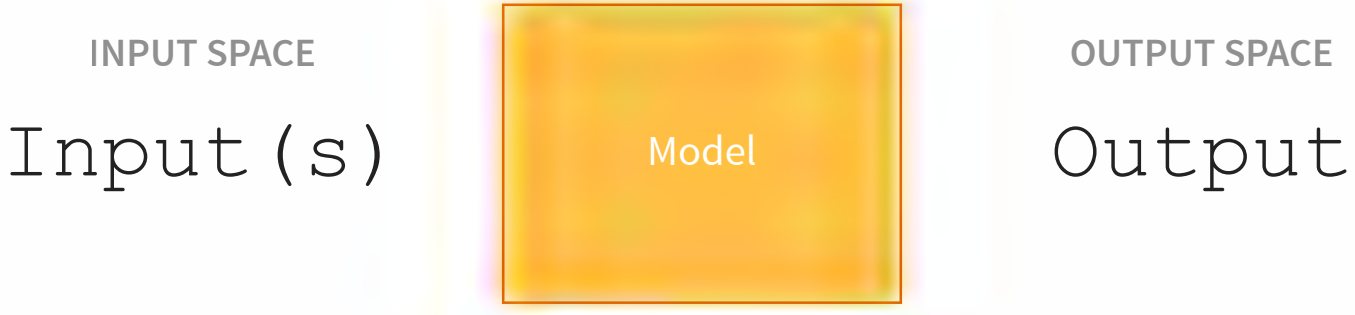
Im Kurs nutzen wir nur Offline Learning!

MACHINE LEARNING IM CAS

SUPERVISED LEARNING



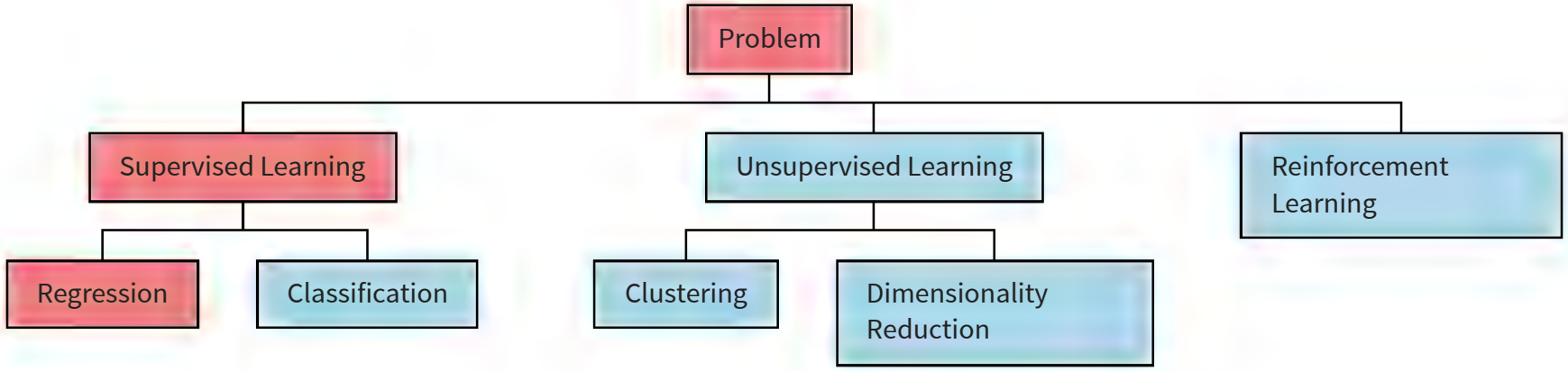
SUPERVISED LEARNING



Daten unterteilt in Input und dazugehörende Output
(unabhängige und abhängige Variablen)

Vorhergehende Problemfälle waren alle Supervised Learning!

REGRESSION


Problem
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Regression
Classification
Clustering
Dimensionality Reduction

REGRESSION

Der Output ist eine kontinuierliche Grösse.

- Immobilien Verkaufspreis abschätzen
- Gewicht eines Fisches vorhersagen

REGRESSION - BEISPIEL - GEWICHT EINES FISCHES

- Input(s): `width (cm)`; Output: `weight (g)`

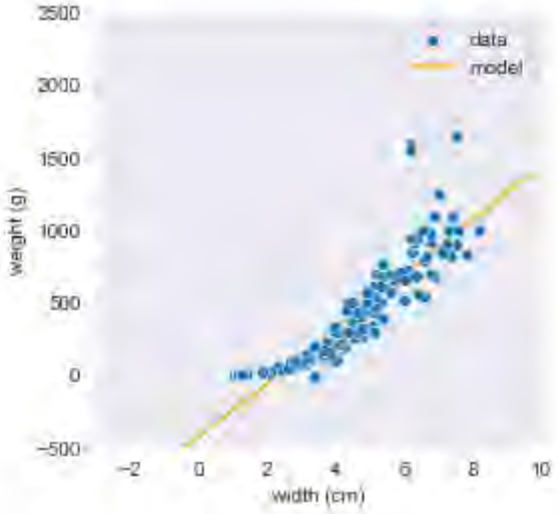
INPUT SPACE

`width: 5cm`

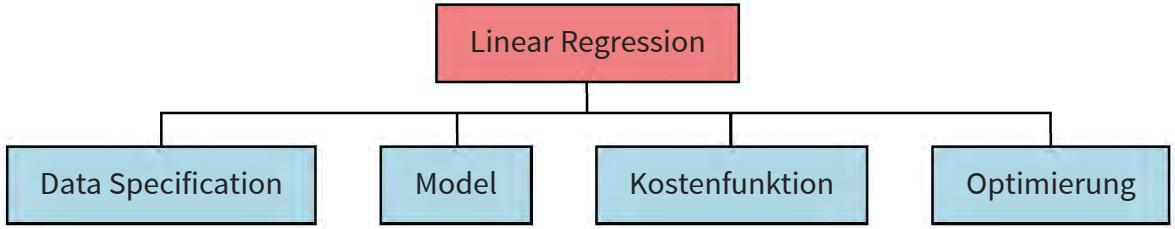
Modell

OUTPUT SPACE

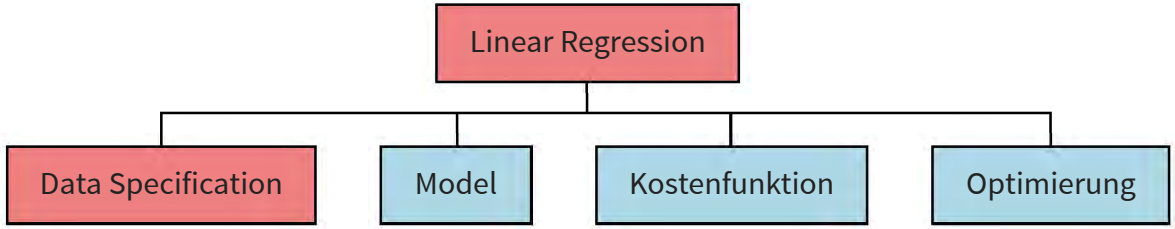
`weight: 508g`



LINEAR REGRESSION



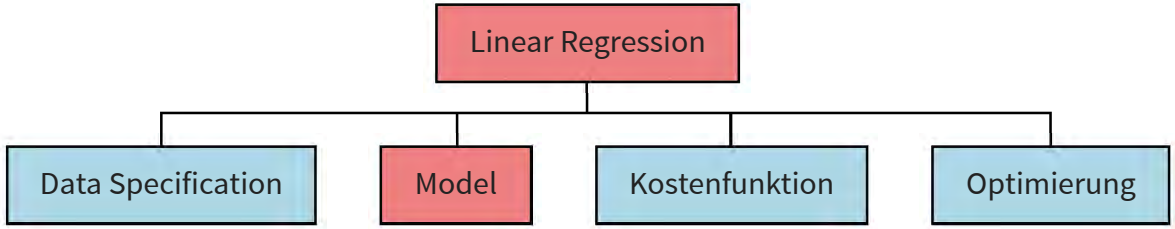
LINEAR REGRESSION



LINEAR REGRESSION - DATA SPECIFICATION

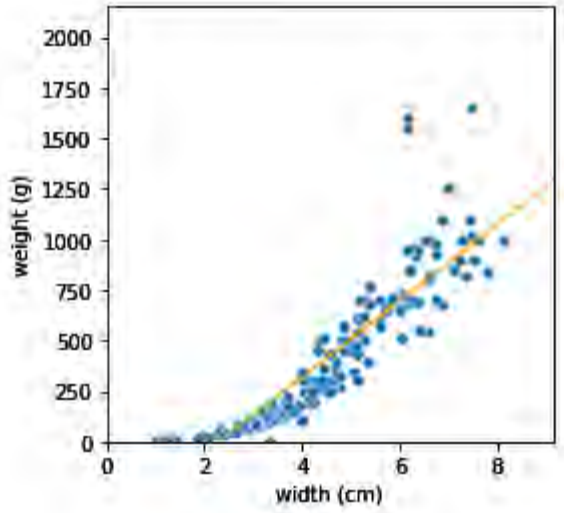
- Was ist die kontinuierliche Ziel-Variable, z.B. `weight (g)`
- Welche Features wählen wir, z.B. um einen Fisch zu repräsentieren (`width (cm), ...`)
- Kategorische Features müssen encoded werden.
- Wenn Regularisiert: Numerische Features müssen standardisiert werden.

LINEAR REGRESSION



LINEAR REGRESSION - INTUITION

Wir möchten das Gewicht (weight) anhand eines einzigen Features, der Breite (width) vorhersagen.



Dazu verwenden wir das folgende Lineare Modell

$$\text{weight} = \beta_0 + \beta_1 * \text{width}$$

INPUT SPACE

`width: 5cm`

$\text{weight} = \beta_0 + \beta_1 * \text{width}$

OUTPUT SPACE

`weight: 508g`

LINEAR REGRESSION - CODE

Teil 1 in linear_regression.ipynb

LINEAR REGRESSION - THEORY - VERALLGEMEINERN

- Wir möchten das lineare Modell verallgemeinern
    1. Unabhängig vom Problem
    2. Mehrere Features
    3. Bezug zu echten Beobachtungen

LINEAR REGRESSION - THEORY - VERALLGEMEINERN

Wir möchten das lineare Modell verallgemeinern

1. Unabhängig vom Problem
2. Mehrere Features
3. Bezug zu echten Beobachtungen

$$\text{weight} = \beta_0 + \beta_1 * \text{width}$$

$$\hat{y} = \beta_0 + \beta_1 * x_1$$

LINEAR REGRESSION - THEORY - VERALLGEMEINERN

- Wir möchten das lineare Modell verallgemeinern
  1. Unabhängig vom Problem
  2. Mehrere Features
  3. Bezug zu echten Beobachtungen

$$\hat{y} = \beta_0 + \beta_1 * x_1$$

$$\hat{y} = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2$$

$$\hat{y} = \beta_0 + \beta_1 * x_1 + \ldots + \beta_p * x_p$$

Output ist eine Gewichtete Summe der Features

LINEAR REGRESSION - THEORY - VERALLGEMEINERN

Wir möchten das lineare Modell verallgemeinern

- 1. Unabhängig vom Problem
  2. Mehrere Features
  3. Bezug zu echten Beobachtungen

$$\hat{y} = \beta_0 + \beta_1 * x_1 + \ldots + \beta_p * x_p$$

$$\hat{y}^{(i)} = \beta_0 + \beta_1 * x_1^{(i)} + \ldots + \beta_p * x_p^{(i)}$$

$$y^{(i)} \approx \hat{y}^{(i)} = \beta_0 + \beta_1 * x_1^{(i)} + \ldots + \beta_p * x_p^{(i)}$$

$$y^{(i)} = \hat{y}^{(i)} + \epsilon^{(i)} = \beta_0 + \beta_1 * x_1^{(i)} + \ldots + \beta_p * x_p^{(i)} + \epsilon^{(i)}$$

"GANZ SIMPLE MATHEMATIK"

$$\hat{y} = x_1 \beta_1 + x_2 \beta_2 + \cdots + x_p \beta_p + \beta_0$$

$$\hat{y} = \sum_{i=1}^{M} x_i \beta_i + \beta_0$$

$$\hat{y} = \vec{x} \vec{\beta} + \beta_0$$

$$\hat{y} = x\beta + \beta_0$$

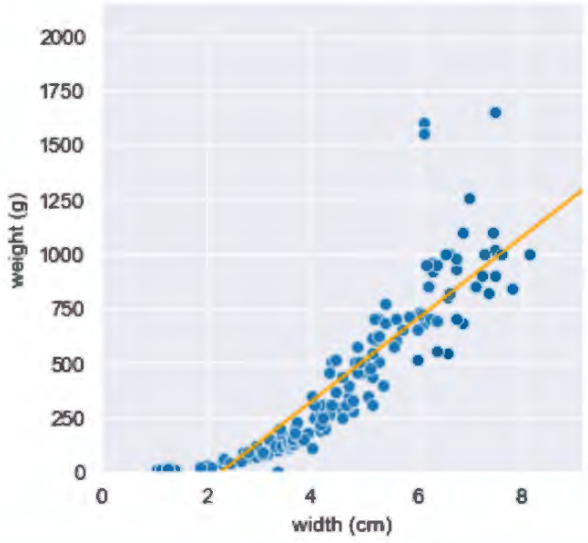
$$\hat{y} = x\beta$$

Nur Notation: Alle bedeutet das Gleiche!

LINEAR REGRESSION - MEHRERE FEATURES

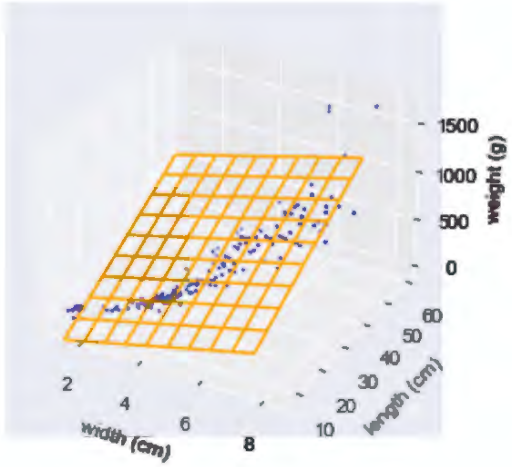
1 FEATURE

$$\hat{y} = \beta_0 + \beta_1 * x_1$$



2 FEATURE

$$\hat{y} = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2$$



3 FEATURE

$$\hat{y} = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \beta_3 * x_3$$

???

Visualisierung gut für Intuition.
Mathematik verallgemeinert auf höhere Dimensionen.

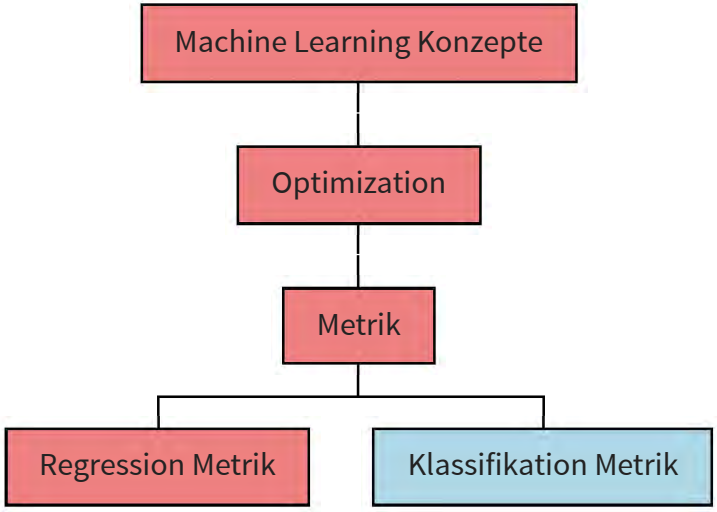
QUESTIONS

QUESTIONS

1. Was ist das lineare Modell?

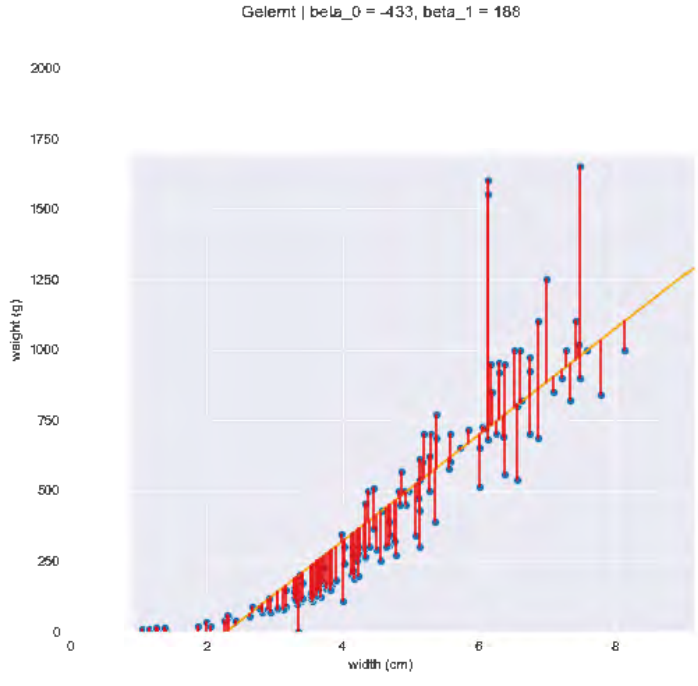
2. Ist das lineare Modell immer eine Gerade?

REGRESSION METRIK



REGRESSION - FEHLER EINES MODELLS MESSEN?

- Residuals messen Fehler pro Datenpunkt



$$\text{Residual} = y - \hat{y}$$

- Residuals zu einer Zahl zusammenfassen => Metrik

REGRESSION - METRIK

- Wie gewichten wir die Fehler pro Datenpunkt?
  - Sind alle Fehler gleich zu bewerten?
  - Sind gröbere Fehler mehr zu bestrafen?
  - Sind Fehler Absolut oder Prozentual zu bestrafen?
- Die Wahl der Metrik gibt vor wie Fehler zu gewichten sind.

Welche Metrik geeignet ist, ist Problemabhängig!

REGRESSION - METRIK - BEISPIELE

- MAE: Mean Absolute Error

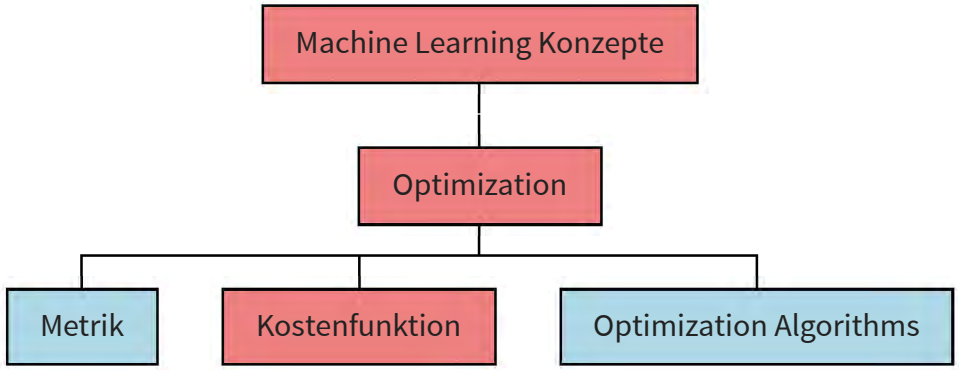
$$MAE(\vec{y}, \vec{\hat{y}}) = \frac{1}{N} \sum_{i=1}^{N} |y^{(i)} - \hat{y}^{(i)}|$$

- MSE: Mean Squared Error

$$MSE(\vec{y}, \vec{\hat{y}}) = \frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)})^2$$

MSE bestraft im Vergleich zu MAE Fehler grösser 1 stärker und Fehler kleiner 1 schwächer.

KOSTENFUNKTION



KOSTENFUNKTION

$$J(\vec{\beta}) = MSE(\vec{y}, \hat{\vec{y}})$$

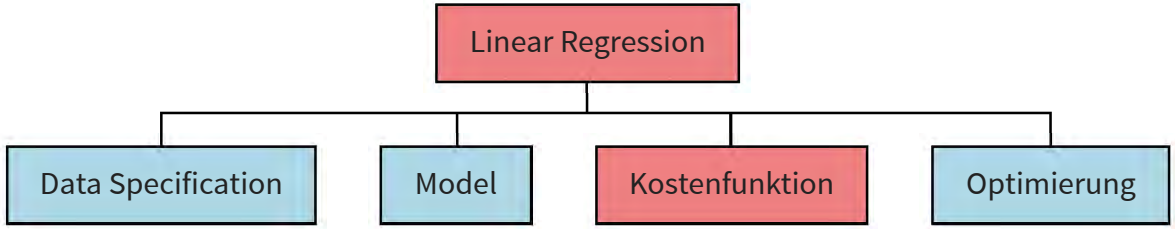
$$= \frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - (\beta_0 + \beta_1 * x_1^{(i)} + \ldots + \beta_p * x_p^{(i)}))^2$$

Beispiel (Lineare Regression mit 2 Parameter):

$$J(\beta_0, \beta_1) = \frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - (\beta_0 + \beta_1 x_1^{(i)}))^2$$

Synonym: Kostenfunktion = Loss f. = Objective f.

LINEAR REGRESSION



LINEAR REGRESSION - KOSTENFUNKTION

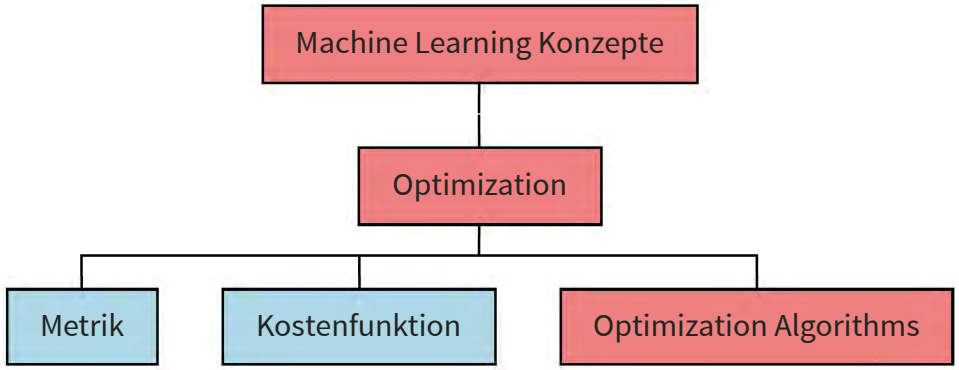
$$J(\vec{\beta}) = MSE(\vec{y}, \hat{\vec{y}})$$

Im `sklearn` wird `LinearRegression` nach der MSE-Kostenfunktion optimiert.
Es wären andere Kostenfunktionen möglich.

LINEAR REGRESSION - CODE

Teil 2 in in linear_regression.ipynb

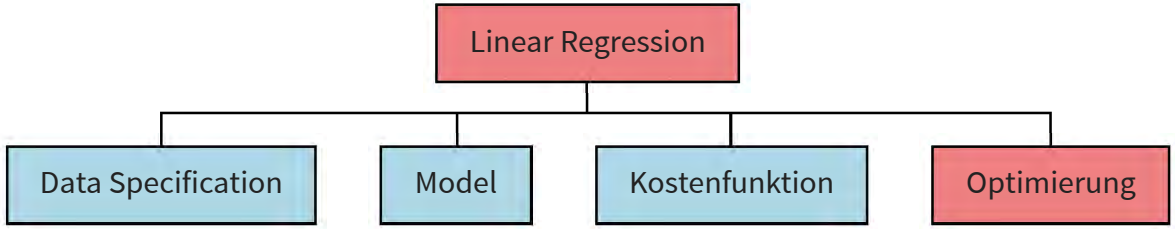
OPTIMIZATION ALGORITHMS



OPTIMIERUNG

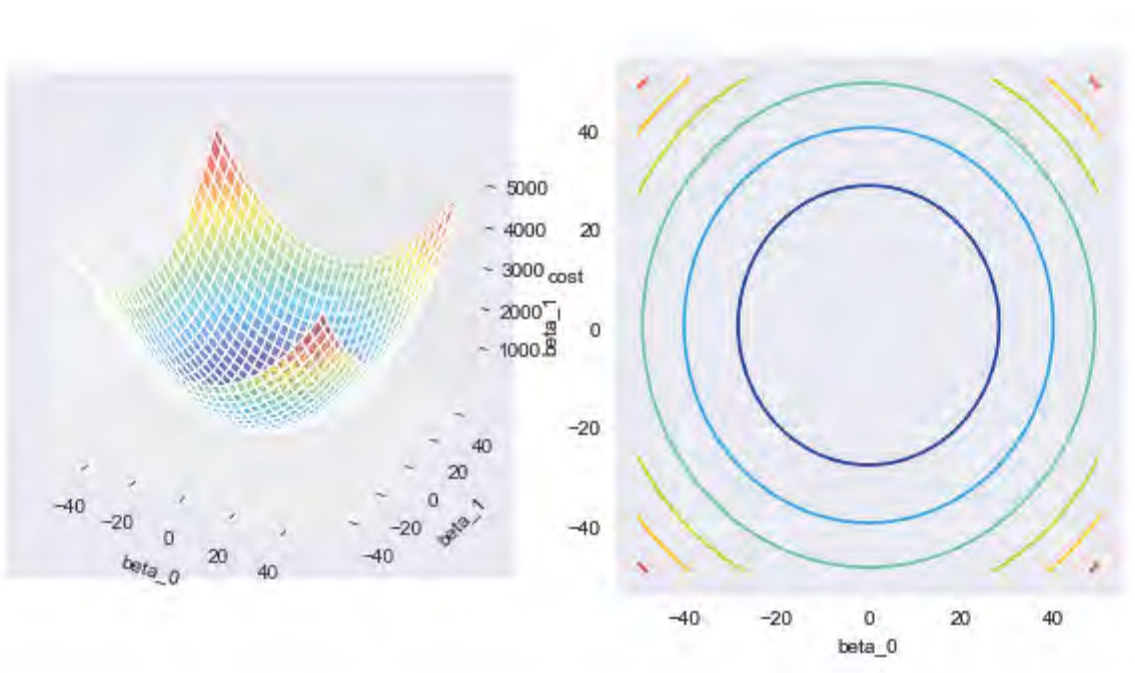
- Optimierung ist der Mechanismus, wie wir die lernbaren Parameter eines Models für eine Kostenfunktion aus Daten lernen.
- Unterschiedliche Optimierungs-Algorithmen existieren
- Unterschiedliche Garantien
  - Performanz: Wie schnell
  - (Generalisierung: Qualität)

LINEAR REGRESSION



LINEAR REGRESSION - KOSTENFUNKTION - VISUELL

Folgende Funktion $J(\beta_0, \beta_1)$ visualisiert für einen bestimmten Datensatz:

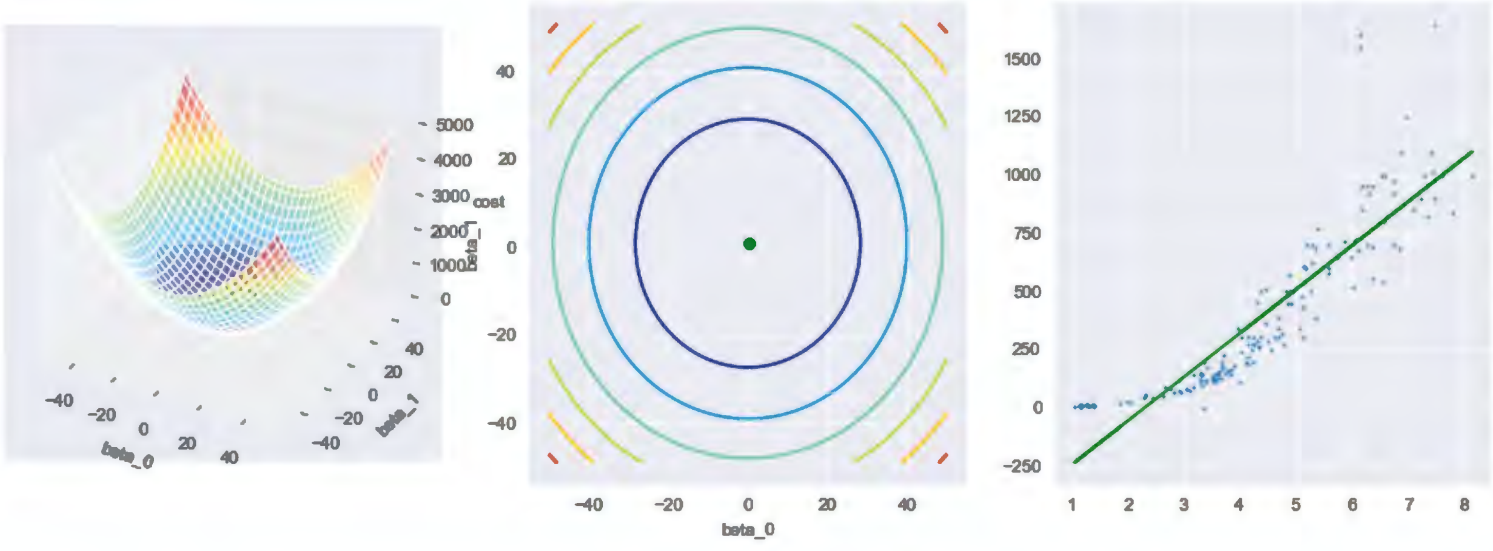


Schüssel nennt man Convex Kostenfunktion

LINEAR REGRESSION - OPTIMIERUNG - ANALYTISCH

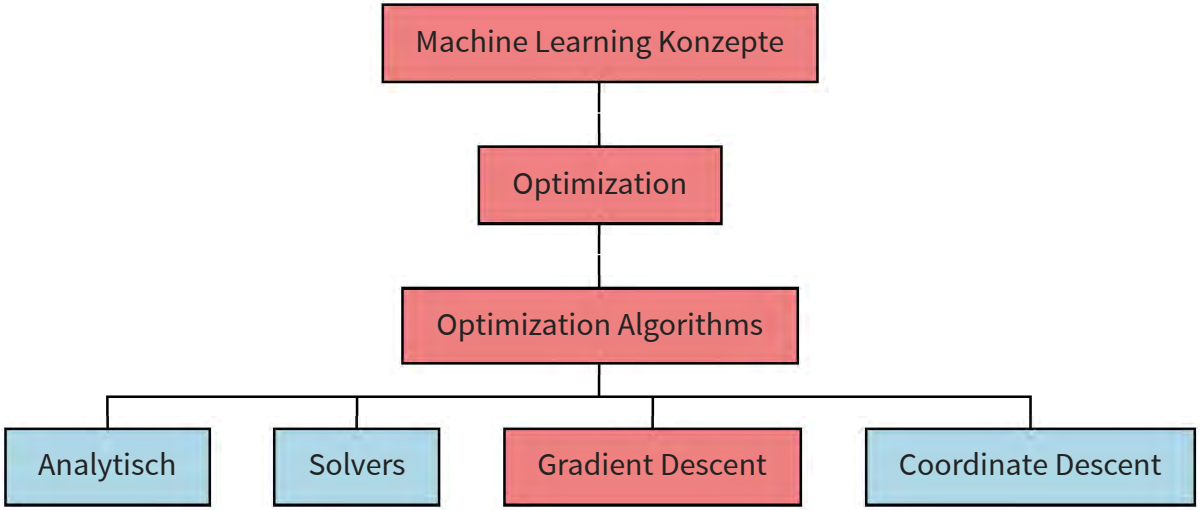
Idee: Wir lösen folgende Gleichung:

$$\frac{\partial J(\vec{\beta})}{\partial \vec{\beta}} \stackrel{!}{=} 0$$



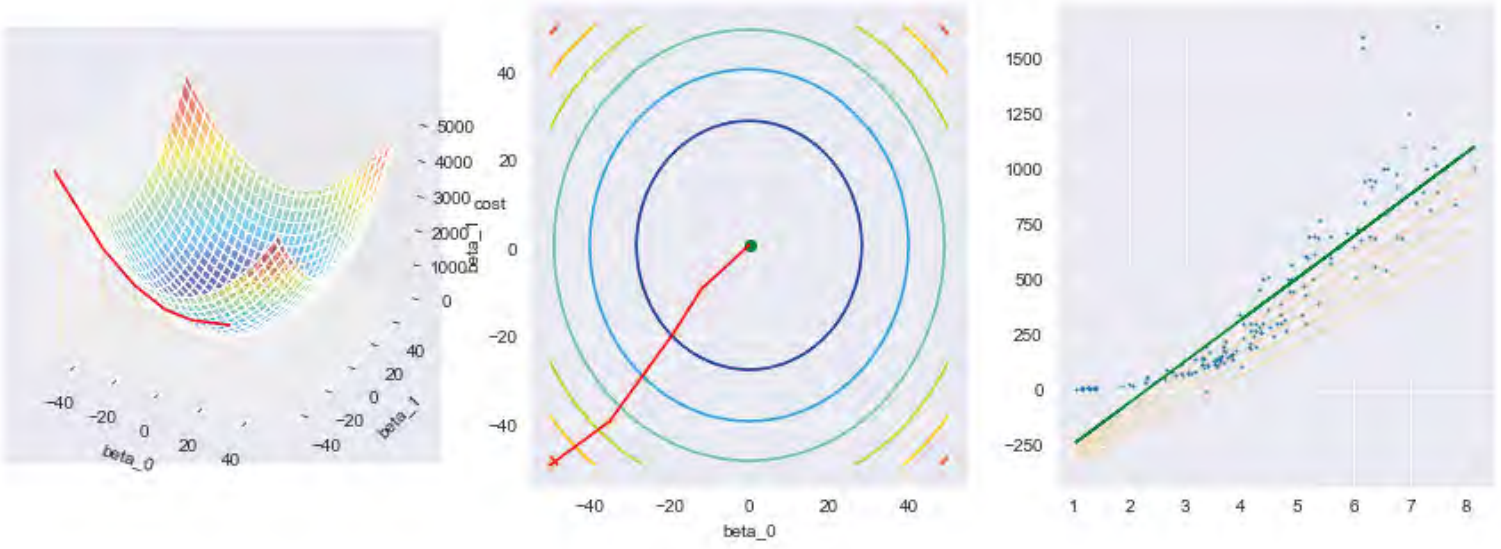
Nur bei bestimmten Funktionen möglich

GRADIENT DESCENT



LINEAR REGRESSION - OPTIMIERUNG - GRADIENT DESCENT

Alternative Methode zur Analytischen Methode zum setzen der lernbaren Parameter (β)



LINEAR REGRESSION - OPTIMIERUNG - GRADIENT DESCENT

Wie finden wir die Richtung herunter?

Richtung entspricht der Steigung der Kostenfunktion!

Wie finden wir die Steigung einer Funktion?

Analysis => Die Ableitung der Kostenfunktion gibt uns die Steigung/Richtung.

$$\nabla J(\beta) = \left(\frac{\partial J(\beta)}{\partial \beta_0}, \frac{\partial J(\beta)}{\partial \beta_1}, \ \dots \ , \frac{\partial J(\beta)}{\partial \beta_p}\right)$$

OPTIMIERUNG - GRADIENT DESCENT

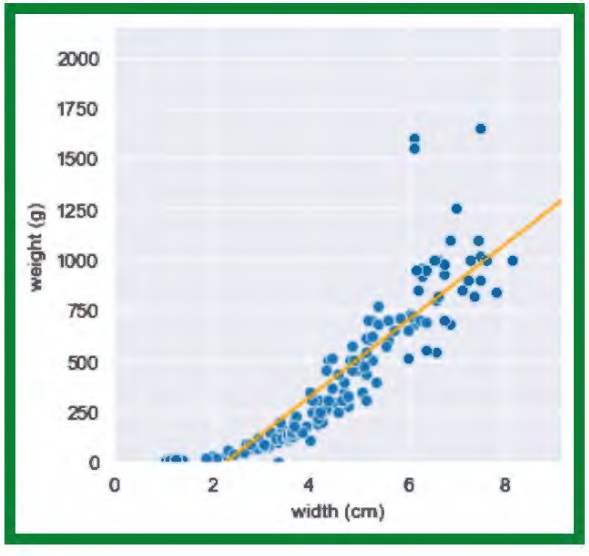
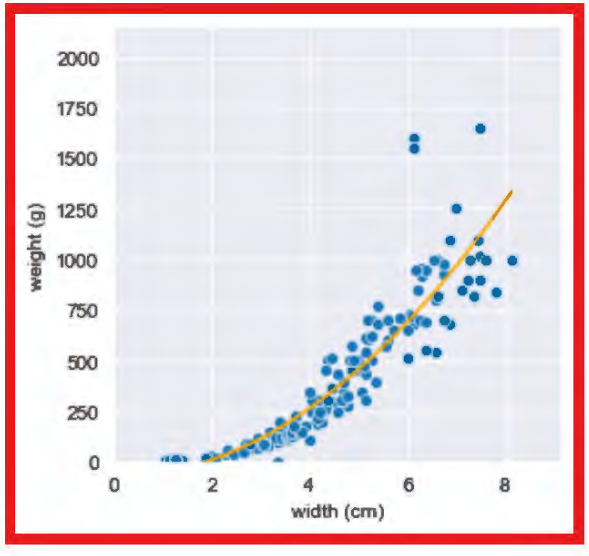
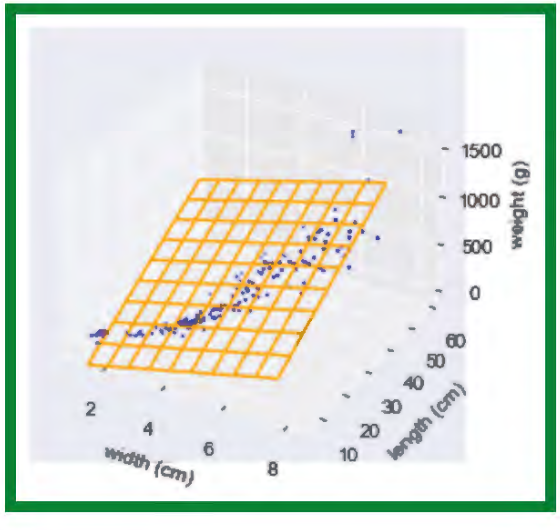
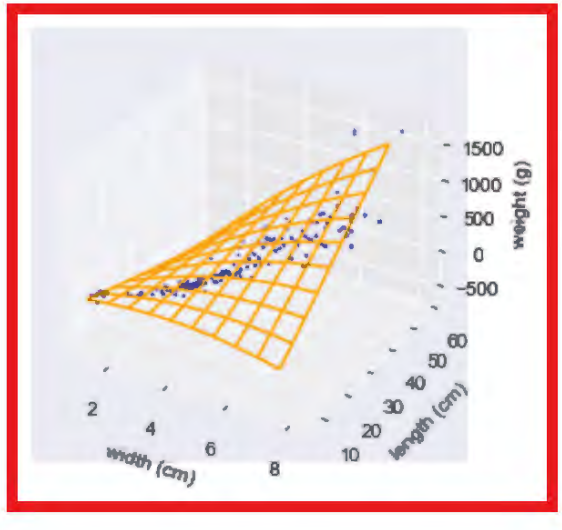
Gradient Descent-Algorithmus funktioniert wie folgt:

1. Initialisiere $\vec{\beta}^{(0)}$ mit zufälligen Werten.
2. Aktualisiere: $\vec{\beta}^{(k)} = \vec{\beta}^{(k-1)} - \eta \cdot \nabla J(\vec{\beta}^{(k-1)})$
3. Wiederhole 2. bis eine Abbruchbedingung erreicht.

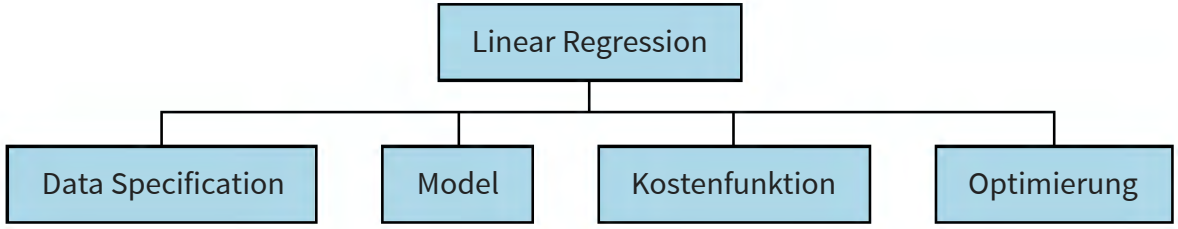
Auch bei Non Convex Kostenfunktion möglich

In der Praxis wird die Analytische Methode für Lineare Regression eingesetzt (schneller)

LINEAR REGRESSION - LIMITS

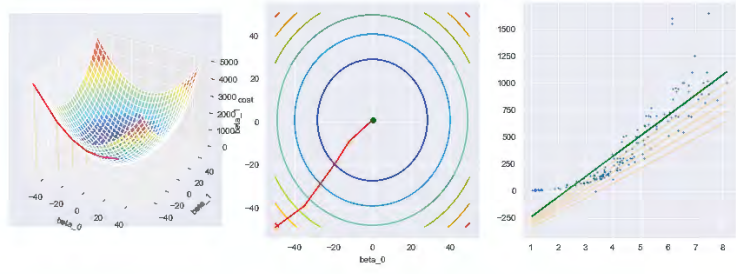
	POSSIBLE	IMPOSSIBLE (IN FEATURE SPACE)
1 FEATURE		
2 FEATURES		

LINEAR REGRESSION



$$y^{(i)} = \hat{y}^{(i)} + \epsilon^{(i)} = \beta_0 + \beta_1 * x_1^{(i)} + \ldots + \beta_p * x_p^{(i)} + \epsilon^{(i)}$$

$$MSE(\vec{y}, \vec{\hat{y}}) = \frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)})^2$$



QUESTIONS

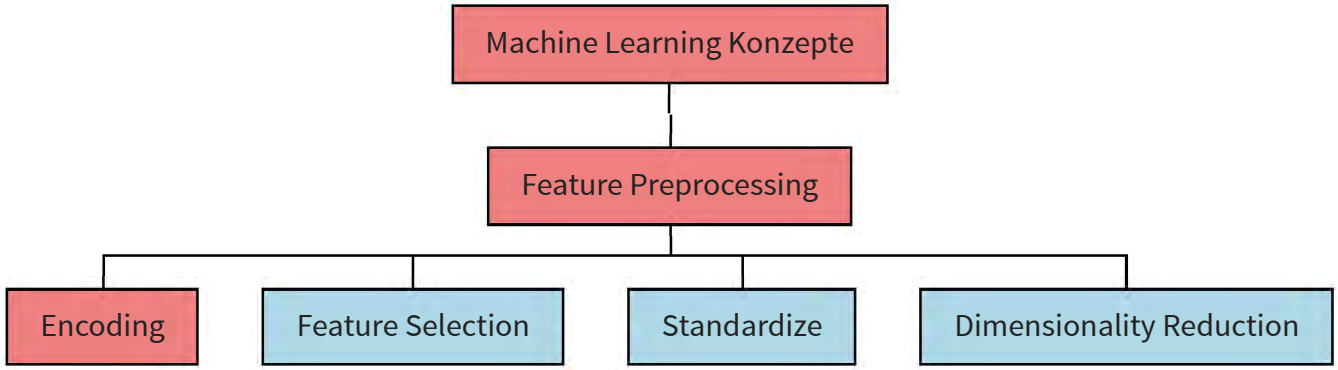
QUESTIONS

1. Was für Annahmen trifft die Lineare Regression?
2. Was sind lernbare Parameter?
3. Wie kann ich die Lineare Regression optimieren?

ÜBUNGSZEIT (30 MINUTEN)

`exercise/introduction.ipynb`

ENCODING


Machine Learning Konzepte
Feature Preprocessing
Encoding
Feature Selection
Standardize
Dimensionality Reduction

ENCODING

- Informationen in anderer Form (meistens Machinen freundlicher) zu repräsentieren.
- Oft notwendig damit Machine Learning Algorithmus mit Text sinnvoll umgehen kann.
- Kann mit oder ohne Informationsverlust passieren.

ENCODING FÜR KATEGORISCHE FEATURES

- Kategorische Features = ["Roach", "Pike", ...]
- Es gibt:
  - Ordinal-Encoding
  - One-Hot-Encoding

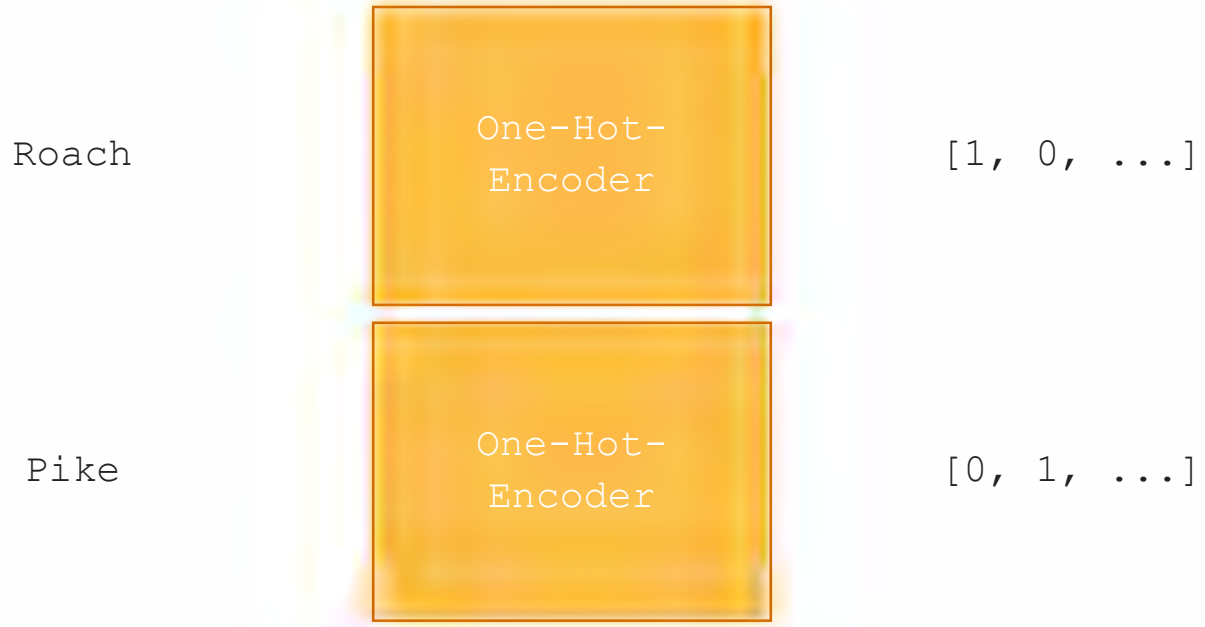
ORDINAL-ENCODING

Mappt ein Kategorisches Feature mit `n` einzigartigen Werten zu aufsteigende Zahlen (`0` bis `n-1`)

Macht aus `1` Feature, `1` Feature. Das neue Feature (Kategorien) hat eine Ordnung!

ONE-HOT-ENCODING

Mappt ein Kategorisches Feature mit `n` einzigartigen Werten zu `n`-dimensionalen Vektoren



Macht aus `1` Feature, `n` Features. Die neuen Features (Kategorien) haben keine Ordnung!

ENCODING

- Meistens wird One-Hot-Encoding eingesetzt, da es keine Ordnung zwischen den Werten annimmt.
- Bei wenig Daten und dem richtigen Feature (z.B. Kleidergrössen "S", "M", "L") könnte Ordinal-Encoding besser sein.
- Man kann auch eigene Encodings konstruieren (z.B. durch Domänen-Wissen oder gelernt aus Daten).

QUESTIONS

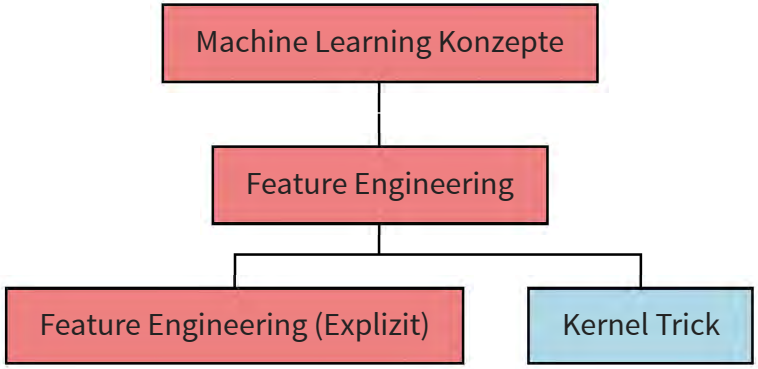
QUESTIONS

1. Was ist Ordinal-Encoding?
2. Was ist One-Hot-Encoding?
3. Wann One-Hot-Encoding vs. Ordinal-Encoding?

LINEAR REGRESSION - CODE

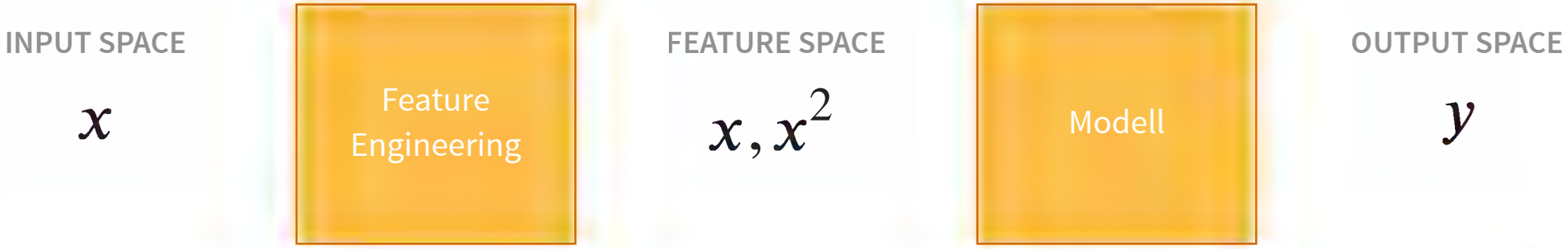
linear_regression_categorical_feature.i

FEATURE ENGINEERING (EXPLIZIT)



FEATURE ENGINEERING

- Idee: Mache aus bestehende Features neue Features, um dem Modell zu helfen
- Standard, z.B. Polynomielle Features
- Kreatives, z.B. durchschnittlicher Quadratmeter-Preis im Umkreis der Immobilie



Domänen-Wissen ins Modell einfliessen zur Verbesserung.
In der Praxis oft entscheidend für gute Performanz des Modells!

QUESTIONS

QUESTIONS

1. Was ist Feature Engineering
2. Wozu Feature Engineering

POLYNOMIELLE REGRESSION - INTUITION

INPUT SPACE
2 FEATURES

x_1 , x_2

Feature Engineering (Polynome)

FEATURE SPACE
5 FEATURES

$x_1 , x_2 , {x_1}^2 , {x_2}^2 , x_1 x_2$

Model (Lineare Regression)

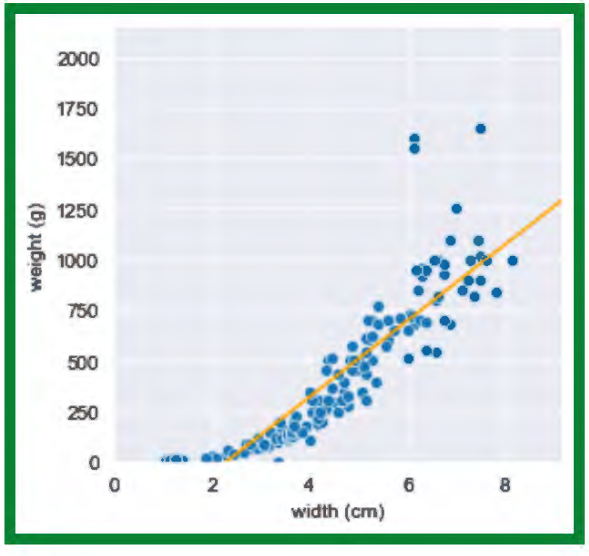
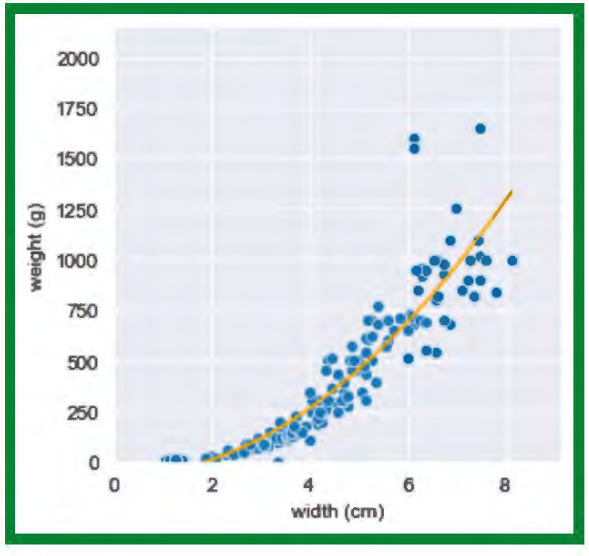
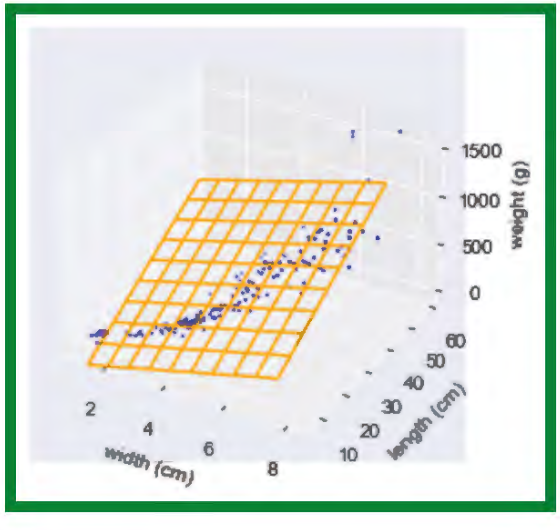
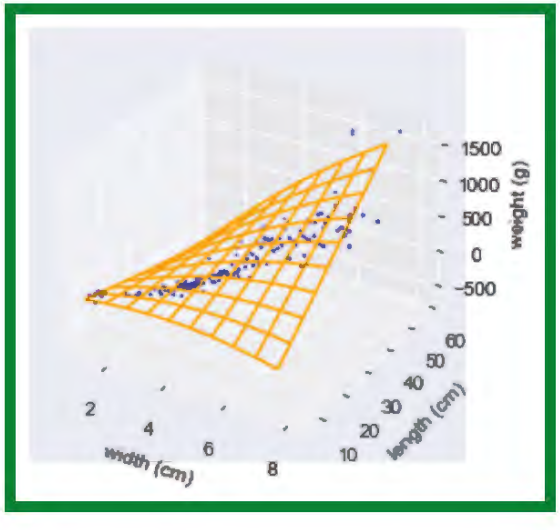
OUTPUT SPACE

y

POLYNOMIELLE REGRESSION - CODE

poly_regression.ipynb

POLYNOMIELLE REGRESSION

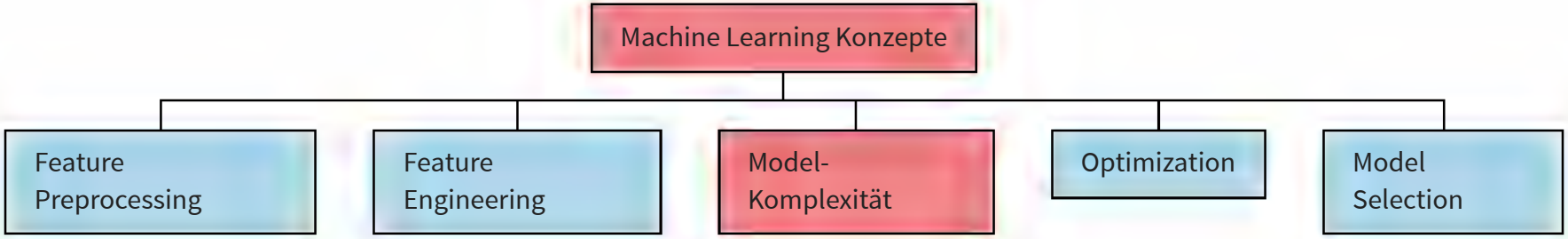
	POSSIBLE	POSSIBLE (IN INPUT SPACE)
1 FEATURE		
2 FEATURES		

~~Ist die Polynomielle Regression immer besser als die Lineare Regression?~~

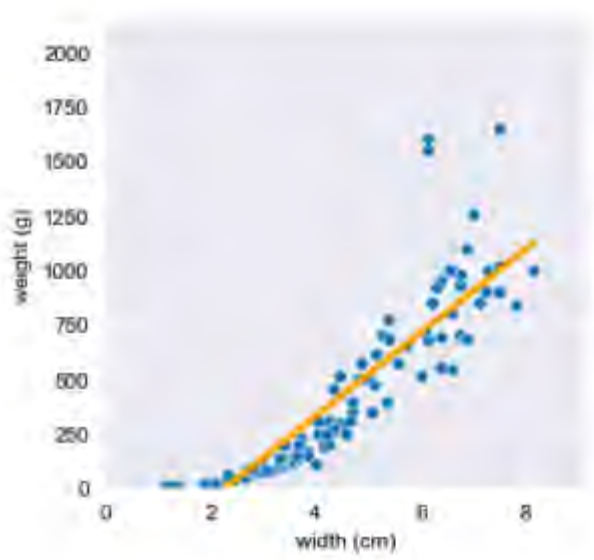
Ja auf den Trainings-Daten, aber Nein (nicht immer) auf neuen, ungesehenen Daten!

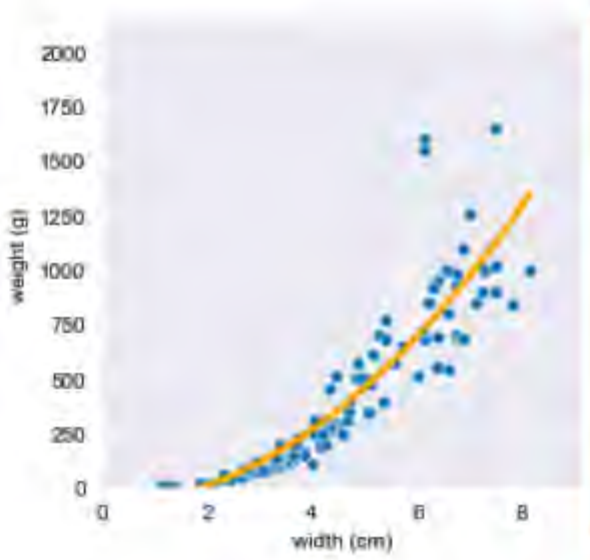
Und Performanz auf neuen, ungesehenen Daten ist das Entscheidende!

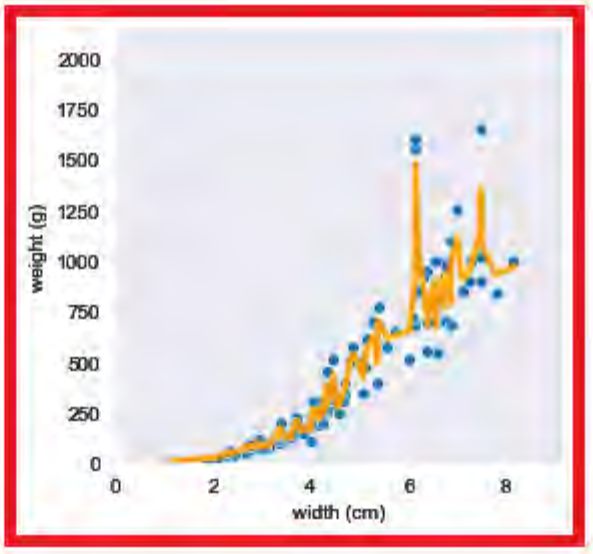
MODEL-KOMPLEXITÄT



MODEL-KOMPLEXITÄT







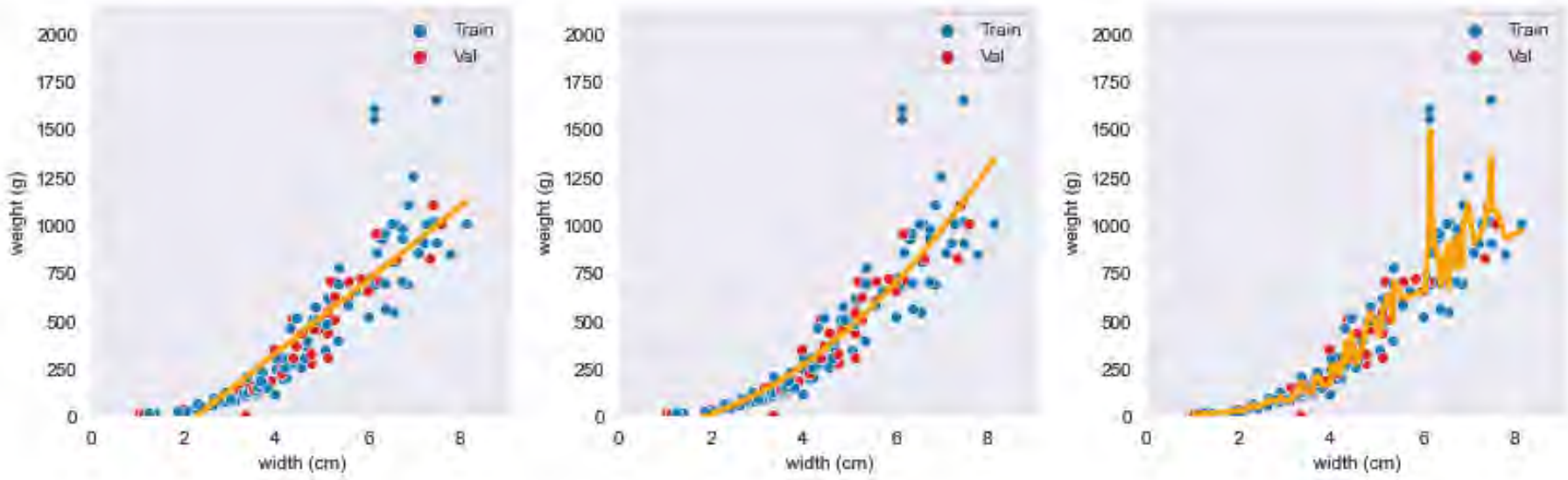
MSE Linear Regression = 33174

MSE Poly. Regression = 27875

MSE RandomForest = 4336

Was fällt auf? Was ist falsch?

MODEL-KOMPLEXITÄT



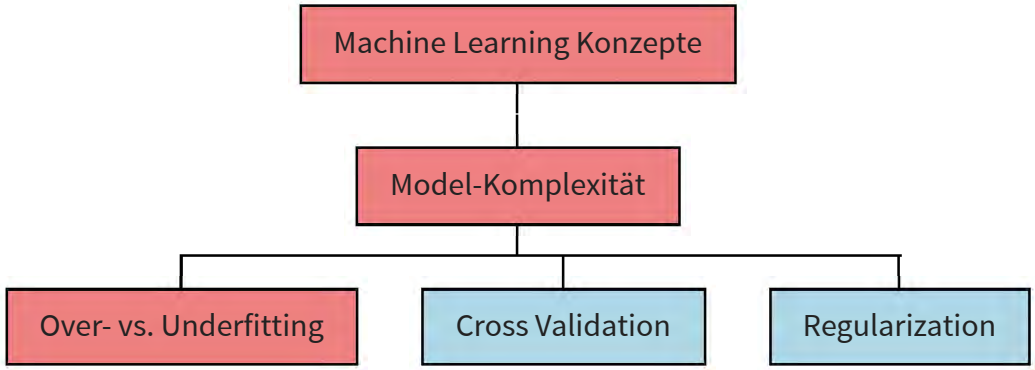
MSE Linear Regression = 14667

MSE Poly. Regression = 9127

MSE RandomForest = 14824

Modell-Performanz nicht (nur) auf Trainings-Daten, sondern auf neuen, ungesehenen Daten evaluieren!

OVER- VS. UNDERFITTING

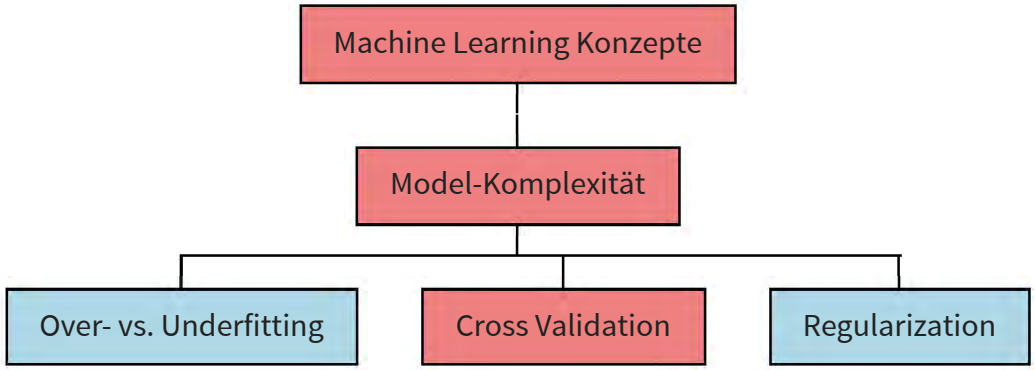


UNDERFITTING VS. OVERFITTING

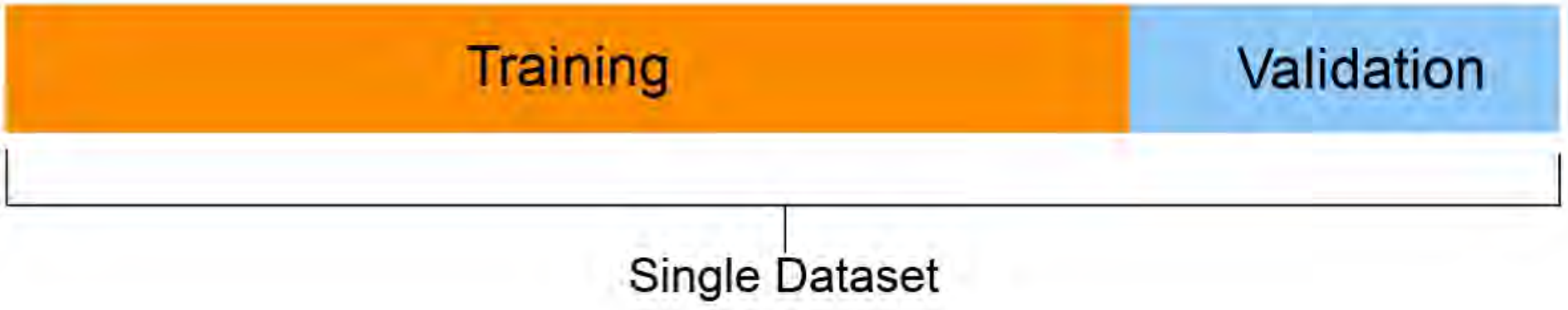
Overfitting:	Wir sind gut auf den Trainings-Daten, aber schlecht auf neuen Daten!
Underfitting:	Wir sind okay auf den Trainings-Daten und okay auf neuen Daten!
Genau richtig:	Wir sind gut auf den Trainings-Daten und gut auf neuen Daten!

UNDERFITTING RICHTIG OVERFITTING

CROSS VALIDATION



HOLD-OUT CROSS VALIDATION



z.B. Unterricht (Training) und
Prüfungsfragen (Validation)

GEFAHREN BEI HOLD-OUT CROSS VALIDATION

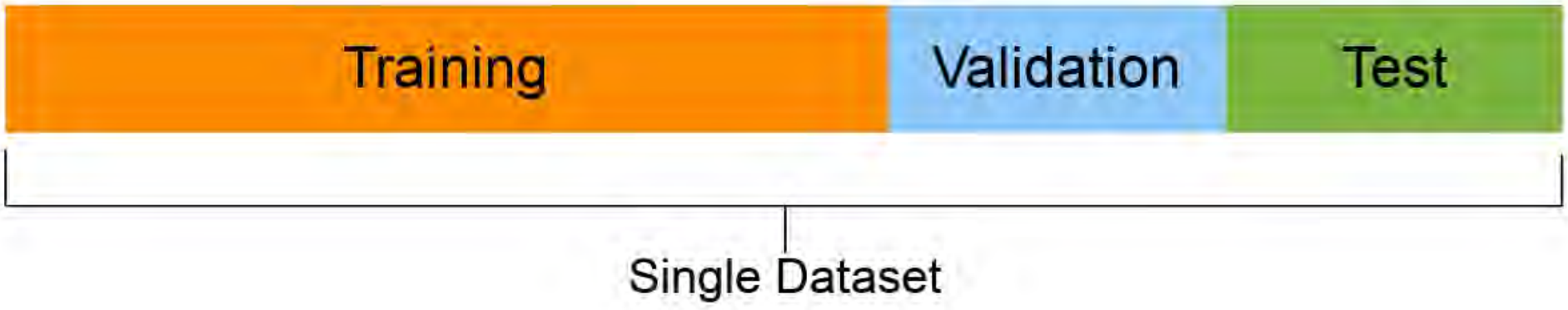
- Falsches Modell wird gewählt
  - Ein Modell ist nur durch Zufall das beste auf einem (zu) kleinen Validation-Set.
  - Je grösser das Validation-Set desto unwahrscheinlicher
- Evaluation auf finalem Modell ist zu optimistisch
  - Weil das Validation-Set die Model-Selection beeinflusste, ist das Modell auf Validation-Set (allenfalls) zu optimistisch ("peeking").
  - Je grössere das Validation-Set desto weniger "zu optimistisch"

Gedankenexperiment: Wir trainieren 1'000'000 verschiedene Modelle und wählen anhand von einem kleinen Validation-Sets (100 samples)

LÖSUNGEN BEI HOLD-OUT CROSS VALIDATION

- "zu optimistisch" feststellen
  => Extra Hold-out Data-Set (Train/Val/Test-Split)
- Grösseres Validation-Set "herbeizaubern"
  => k-Fold Cross Validation

HOLD-OUT CROSS VALIDATION MIT TEST-SET



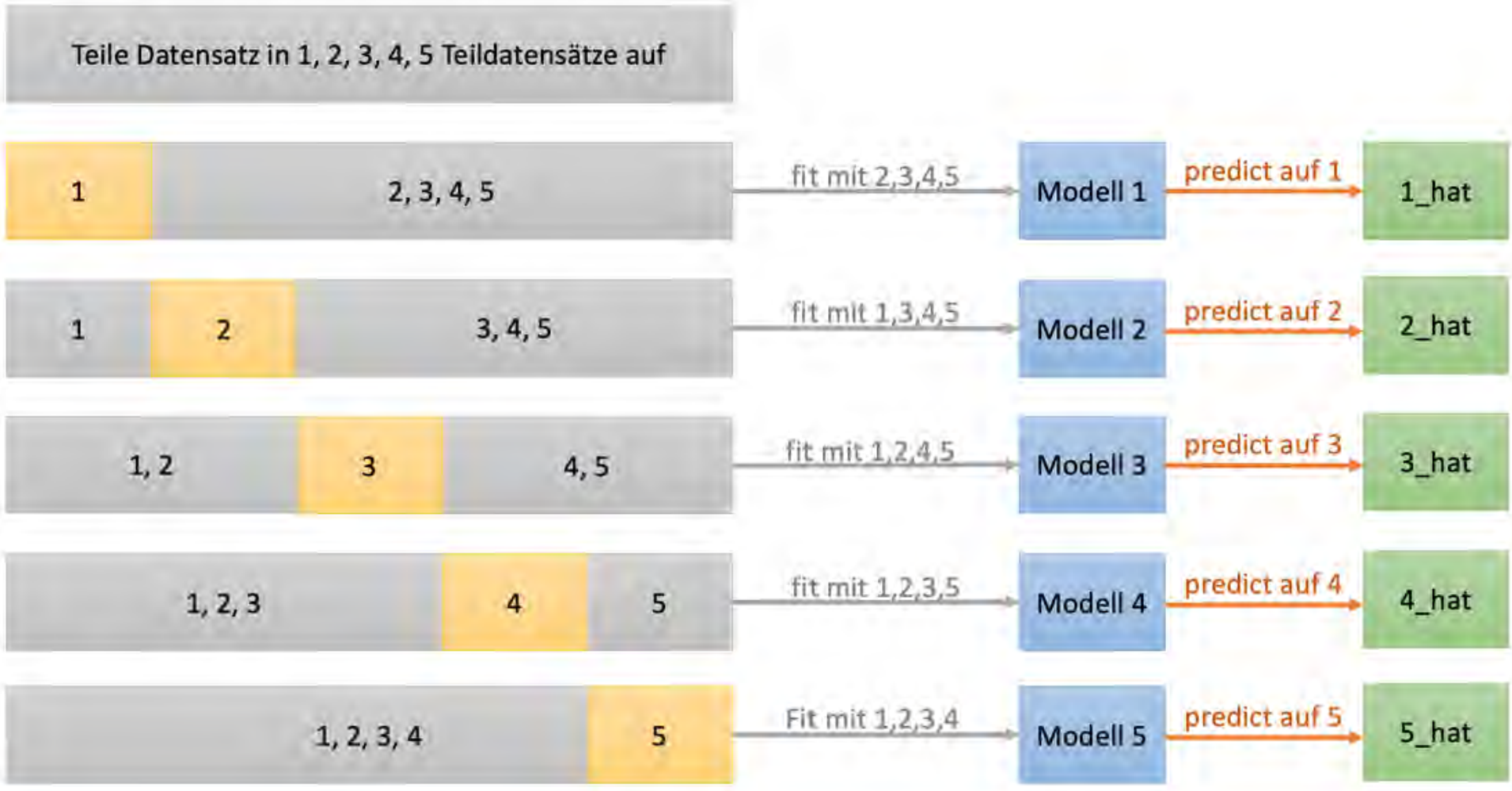
Test-Set darf keinen Einfluss haben (no peeking)!

z.B. Unterricht (Training), Beispiel-Prüfungsfragen (Validation) und echte Prüfungsfragen (Test)

K-FOLD CROSS VALIDATION

- Ein Verfahren um alle Daten als Validation-Set zu verwenden, das Validation-Set wird grösser!
- Benötigt mehr Rechenzeit (bei wenig Daten oft kein Problem)
- In der Praxis sehr oft verwendet
- Üblicherweise wird 5-fold oder 10-fold cross validation gemacht

K-FOLD CROSS VALIDATION

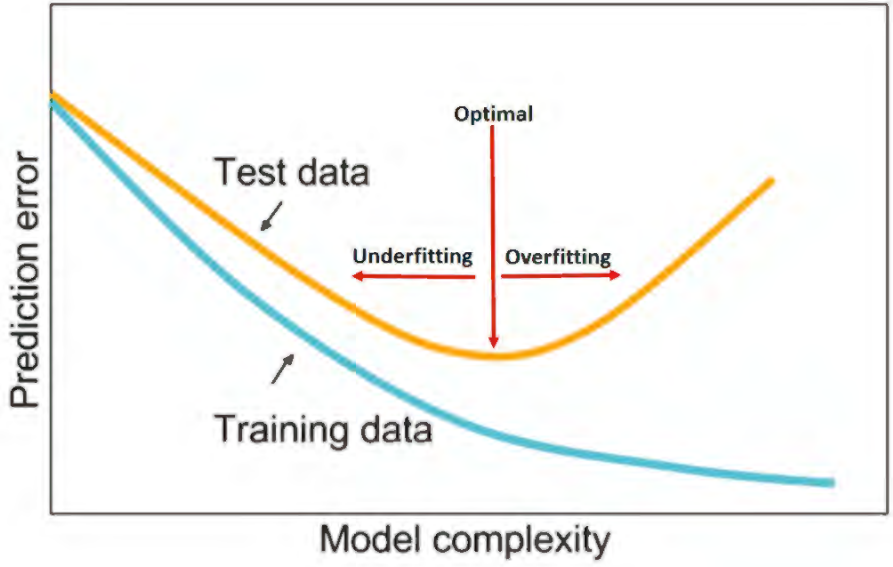


Hier 5-Fold Cross Validation

CROSS VALIDATION

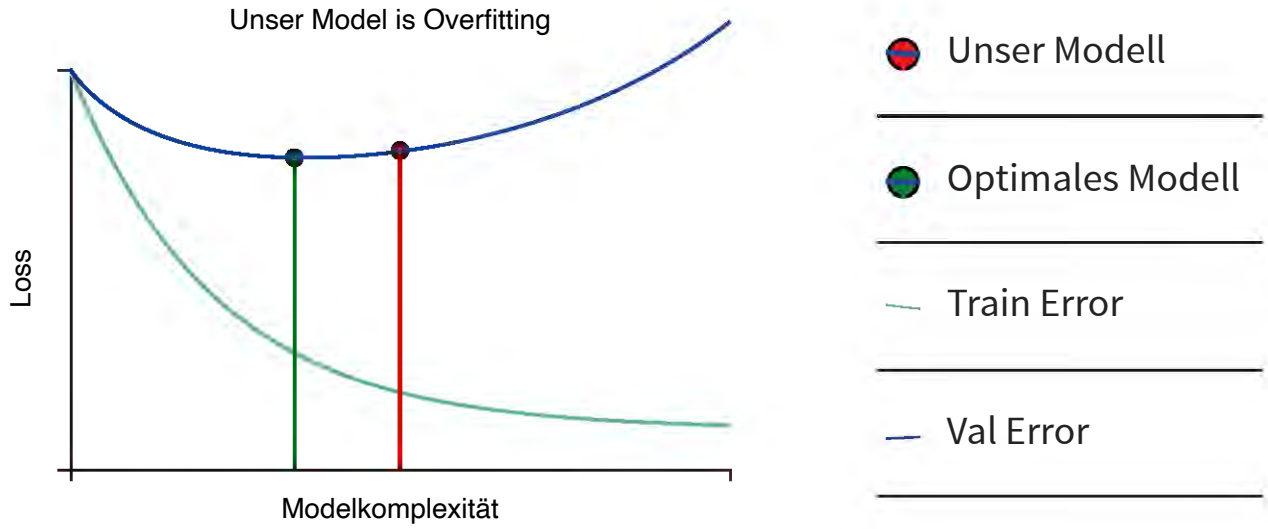
`cross_validation.ipynb`

MODEL-KOMPLEXITÄT UND PERFORMANZ - BILD



Vereinfachung der Realität! Nur Grundidee.

MODEL-KOMPLEXITÄT UND PERFORMANZ - TOOL



DATEN

- Mehr Samples
- Data Augmentation

FEATUERS

- Feature Selection
- Feature Engineering

MODEL KOMPLEXITÄT

- Representation Capacity: Linear Regression Polynomielle Regression Random Forest Big Neural Network
- Regularisierung: - +
- Domänen-Wissen ausprogrammieren (spezifisches Feature Encoding, Feature Preprocessing, Weight-Sharing, ...)

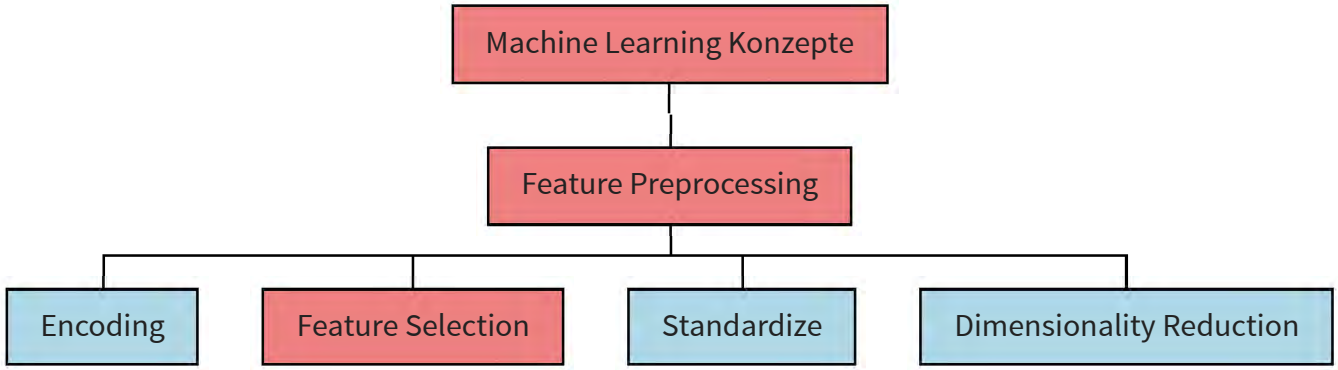
Vereinfachung der Realität! Nur Grundidee.

QUESTIONS

QUESTIONS

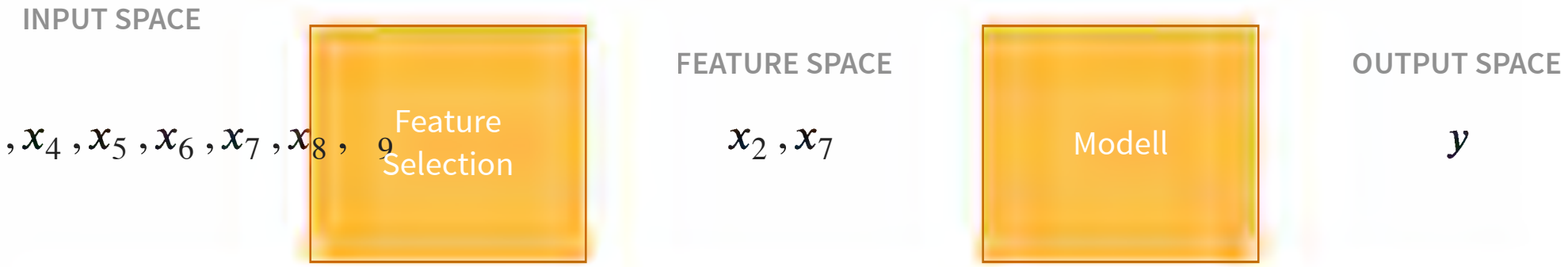
1. Was ist Underfitting?
2. Was ist Overfitting?
3. Wie können wir Overfitting feststellen?
4. Was ist hold-out Cross Validation?
5. Was ist k-fold Cross Validation?
6. Was ist die Modell-Kompexität?

FEATURE SELECTION



FEATURE SELECTION

- Nehme eine Teilmenge der verfügbaren Features (anstatt alle verfügbaren Features).
- Entferne Features, die mit dem Problem wenig oder nichts zu tun haben.

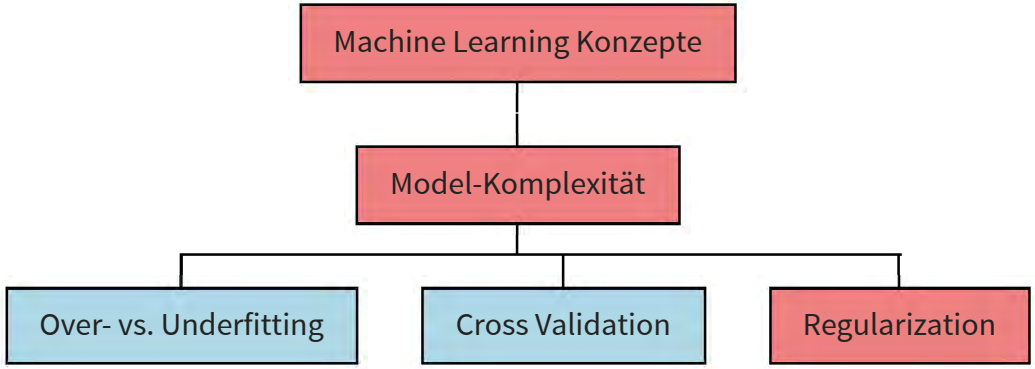


Weniger Features => Bessere Lern-Ausgangslage
=> Bessere Interpretation

FEATURE SELECTION - ANSÄTZE

- Manuell: Filtern der unwichtigen Features durch Domänen-Wissen (mit Annahmen).
  - z.B. Ausschliessen von Hausfarbe für den Rückbaukosten
- Automatisch: Filtern der unwichtigen Features durch Zusammenhänge in Daten (aus Daten).
  - z.B. Ausschliessen von Features die keine oder wenig Korrelation mit Zielvariable haben.
  - [Additional Resources]: sklearn User Guide
    Empfehlung: Grundsätzlich Methoden von "Recursive feature elimination" nicht "Univariate feature selection" verwenden. Da Korrelationen zwischen den Features einen Einfluss hat.

REGULARIZATION



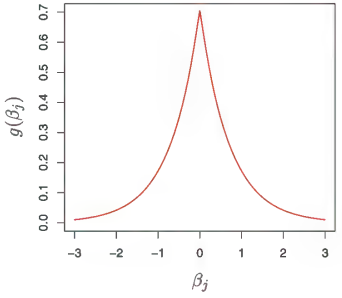
REGULARIZATION

- Wir treffe zusätzliche Annahme über lernbare Parameter (β)
- Übliche Annahme:
  - Grosse Betas sind unwahrscheinlicher
  - Betas sind nahe 0
- Beispiel:
  - L1-Regularization (Absolut Wert - "Hoch 1")
  - L2-Regularization (Quadrieren - Hoch 2)

Daten müssen Standard Skaliert werden!

L1-REGULARIZATION

- Wir nehmen an jeder Parameter β ist so verteilt:



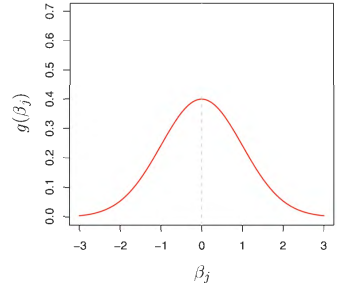
- Dies erreichen wir mit der Anpassung der Kostenfunktion J:

$$J_{Regulasierung}(\vec{\beta}) = J(\vec{\beta}) + \lambda \sum_{j=1}^{p} |\beta_j|$$

Gut für automatische Feature-Selection!

L2-REGULARIZATION

- Wir nehmen an jeder Parameter β ist so verteilt:



- Dies erreichen wir mit der Anpassung der Kostenfunktion J:

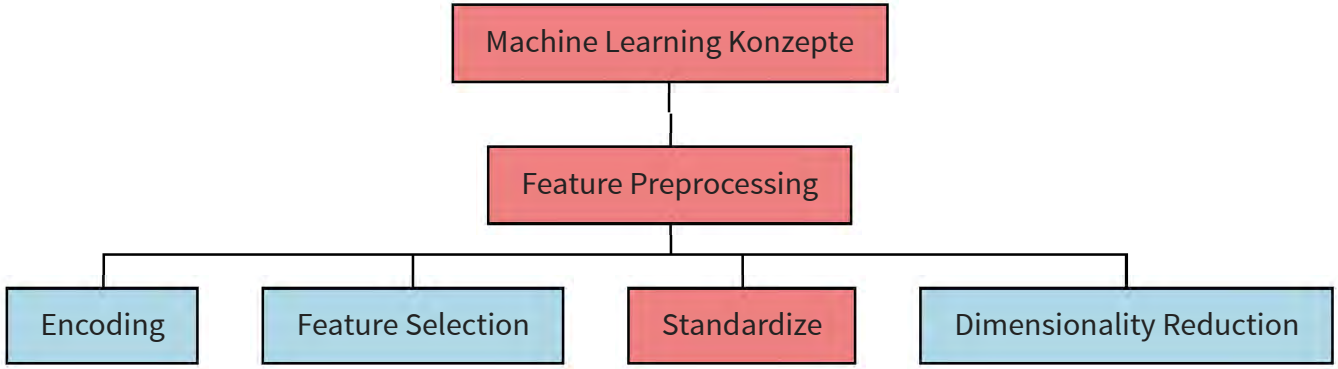
$$J_{Regulasierung}(\vec{\beta}) = J(\vec{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2$$

LINEAR REGRESSION MIT REGULARIZATION

`Lasso` = Linear Regression mit L1-Regularization

`Ridge` = Linear Regression mit L2-Regularization

STANDARDIZE



STANDARD SCALER

- "Nur" ein notwendiger Preprocessing Schritt, damit manche Modelle "richtig" funktionieren.
- Entfernt Einheit (z.B. cm) der Messung und schiebt Verteilung zum Nullpunkt.
- Wird separat pro Feature angewandt, z.B. wird width (cm) transformiert zu width (std)
- Es gibt auch andere Skalierungen (Scalers), meistens wird aber der StandardScaler verwendet.

STANDARD SCALER - PREPROCESSING

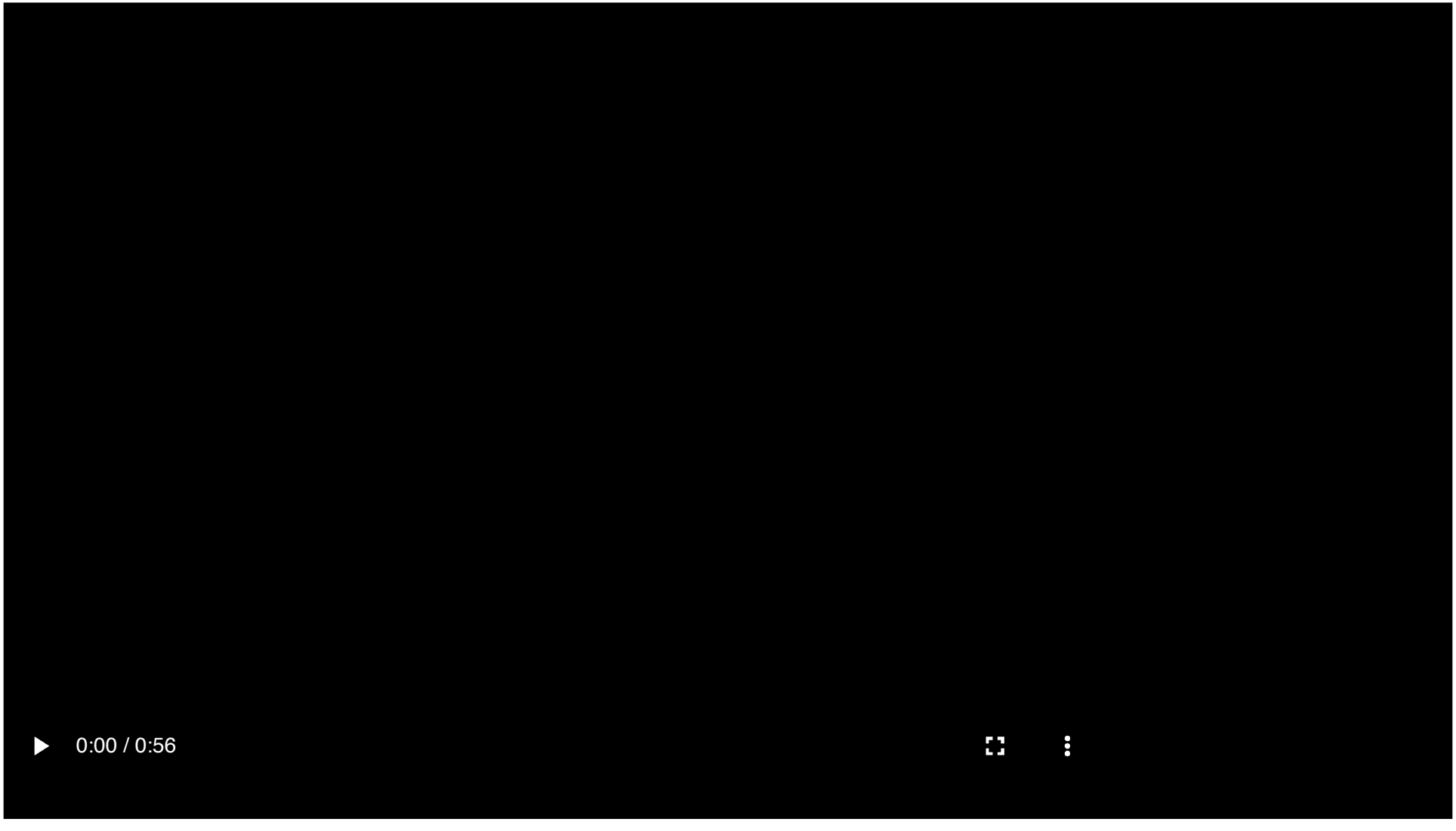
BERECHNUNG PRO FEATURE

1. Minus den Durchschnitt
2. Durch die Standardabweichung

INTUITION

1. Um Nullpunkt verschieben
2. Einheit der Messung entfernen

STANDARD SCALER - INTUITION



WANN?

1. Lernbare Parameter vergleichbar sein müssen

   z.B. Regularization, Gradient Descent

2. Distanzen im Input-Space eine Rolle spielen

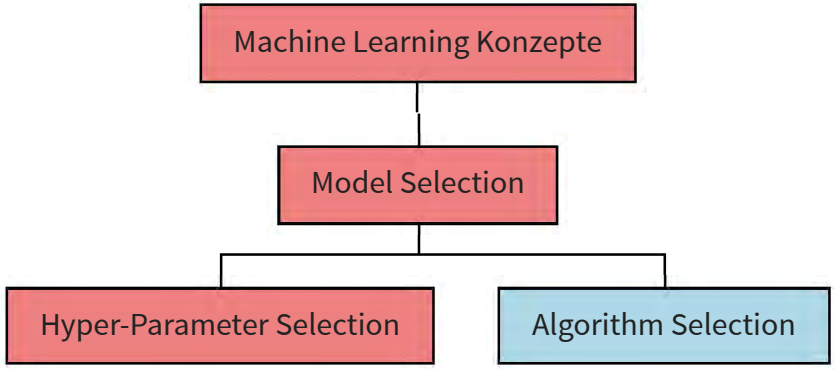
   z.B. k-nearest-neighbors, Support Vector Machine, PCA

QUESTIONS

QUESTIONS

1. Was ist Regularisierung?
2. Wozu Regularisierung?
3. Was ist Standardisieren?
4. Wozu Standardisieren?

HYPER-PARAMETER SELECTION



HYPER-PARAMETER

- Hyper-Parameter sind Parameter, die nicht gelernt werden: "konfiguriert" das Modell
- Beispiele:
  - Regularisierungsstärke: $J_{Regulasierung}(\vec{\beta}) = J(\vec{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2$
  - Anzahl Bäume (Random Forest)
  - Anzahl Cluster (k-Means)
  - Polynomgrad (Feature Engineering)
  - Welche Features (Feature Selection)
  - Netzwerk Architektur (Neural Network)
  - ...

HYPER-PARAMETER SELECTION

Wie können wir einen Hyper-Parameter wählen?

- Manuell: Mit Erfahrung und Theorie bestmöglich setzen.
  - Schwierig, braucht viel Erfahrung
  - Fehler anfällig
- Suchen: Verschiedene Werte ausprobieren, besten (auf ungesehenen Daten) merken.
  - Rechenintensiv, vor allem bei rechenintensiven Modellen (z.B. Neural Networks)

HYPER-PARAMETER SELECTION - SUCHEN

GridSearch	Probiere eine Liste von Werten für beispielsweise die Regularisierungsstärke durch, z.B. 0.1, 1.0 und 10.0
RandomizedSearch	Probiere zufällige Werte für beispielsweise die Regularisierungsstärke innerhalb eines Bereiches aus, z.B. 10 zufällige Werte innerhalb [0.1, 10.0]

HYPER-PARAMETER SELECTION - CODE

`hyper_parameter_regularization.ipynb`

QUESTIONS

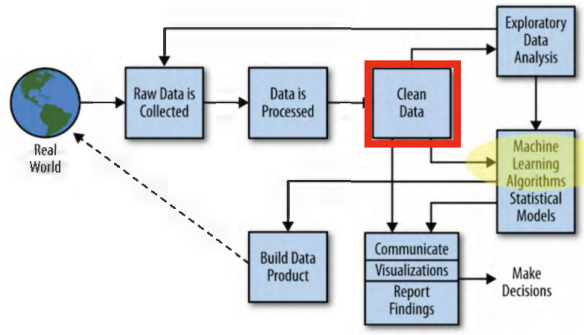
DATA SCIENCE TIPP - WIE EIN PROBLEM ANGEHEN?

- Verstehe das Problem (z.B. Regression, Klassifikation, Zeitreihe, Spezialfälle)
  - Einlesen in jeweiliges Gebiet: Wir haben hier nicht alles behandelt!
- Verstehe das Ziel, wie bestmögliche Genauigkeit (z.B. Noise im Markt nicht vorhersagbar).
  - Siehe Metrikwahl
- Verstehe und evaluiere die verfügbaren Daten
  - Siehe Garbage-in-Garbage-Out und genug Daten
- Evaluiere die Datenlandschaft (z.B. crawlbare Daten einer öffentlichen API)
- Rede mit Domänen Experten
  - Welche Informationen sind wichtig (für Feature Selection)
  - Was sind bekannte Zusammenhänge (für Sicherung der Datenqualität und Feature Engineering)
- Konsolidiere die Literatur oder andere Data Science Experten
  - Welche Modelle haben bei ähnlichen Problemen gut funktioniert?
  - Welche Feature Engineerings, Datenquellen haben geholfen?

TODO

DATA SCIENCE PITFALL - "GARBAGE IN, GARBAGE OUT"

- Datenqualität ist wichtig für Machine Learning: Sind Trainings-Daten schlecht (Garbage in), wird auch das Modell schlecht sein (Garbage out)
  - Manche Metriken sind anfällig auf Outliers (z.B. Mean Squared Error), dann kann bereits ein fehlerhafter Wert (Fehler bei Eingabe/Crawlen) grossen Einfluss haben.



- Überprüfe immer zuerst die Datenqualität:
  - Daten verstehen
  - Daten vereinheitlichen (z.B. Zeitzonen aus Zeit rechnen)
  - Inkonsistenz finden (z.B. Anzahl Räume, Wohnfläche widersprüchlich)
  - Auffällige Werte finden und erklären (oder korrigieren/ausschliessen)
  - Daten visualisieren (Verteilungen, Pairplot) und analysieren
  - Gefundene Muster mit Domänen-Experte gegenchecken
  - Bekannte Zusammenhänge in Daten wiederfinden bzw. gegenprüfen

---

[Alternative Quelle]: Section 2.1 in Paper "How to avoid machine learning pitfalls: a guide for academic researchers"

ÜBUNGSZEIT (60 MINUTEN)

`exercise/linear_regression.ipynb`