

ML PRIMER FOR SOFTWARE SPECIALISTS

03. September 2024

\Nee/MG x adnexo

Martin Stypinski

- Partner @ VeeMG GmbH
 - Software & AI Consulting
- Dean of Studies :
 - CAS Machine Learning 4 Software Engineers
- Hobbies: Stupid Ideas & Cycling



Christian Fässler

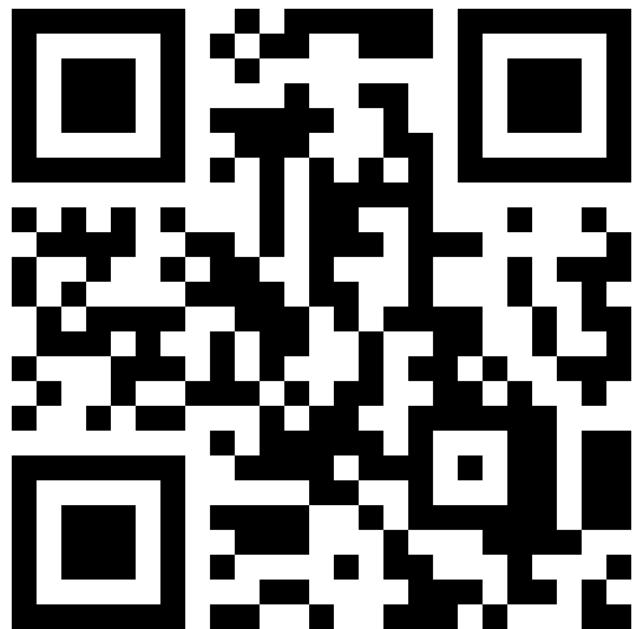
- Co-Founder @ adnexo GmbH
 - Software & IoT Consulting
 - Sensor / Timeseries Data
 - Pythonista
- Teacher :
 - CAS Machine Learning 4 Software Engineers
- Hobbies: Motivate Martin's Stupid Ideas



Let's connect

Martin Stypinski

veemg.com



Christian Fässler

www.adnexo.ch



AI Projekte / Consulting

- Potentielle AI Projekte Identifizieren
- Scoping Effective PoCs
- Evaluating AI Business Cases
- Training:
• <https://aikurs.ch>



Time Table - Vormittag

- 09:00 – 11:30: Einführung in ML
 - Was ist AI, ML, DL
 - Verstehen von Features, Bias Variance Trade-off, IID assumption
- 11:00 – 12:30: Klassische Methoden & Intro SciKit Learn
 - Klassifizierung Decision Trees & KNN
 - Lab Session

Time Table - Nachmittag

- 13:30 – 15:00: Deep Learning
 - Einführung in Deep Learning
 - Deep Learning Lab
- 15:30 – 17:00: MLOps & Deployment
 - 15:30 – 16:15: MLOps Allgemein & Deployment
 - 16:15 – 17:00 FastAPI Lab
 - Raum für Individuelle Fragen / Projekte
- Goal for today: You train and deploy your first model!

Ziele für heute

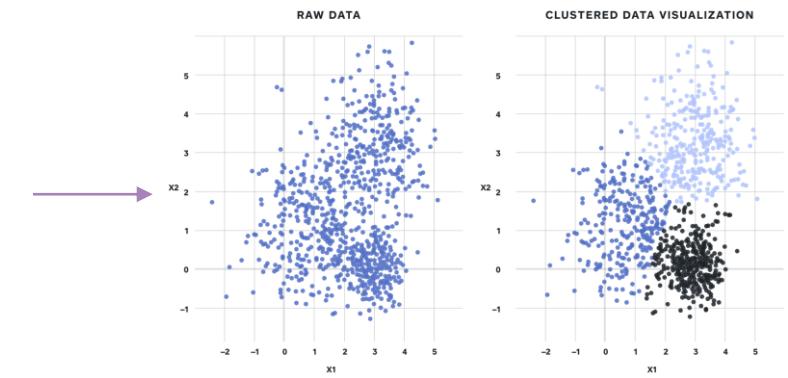
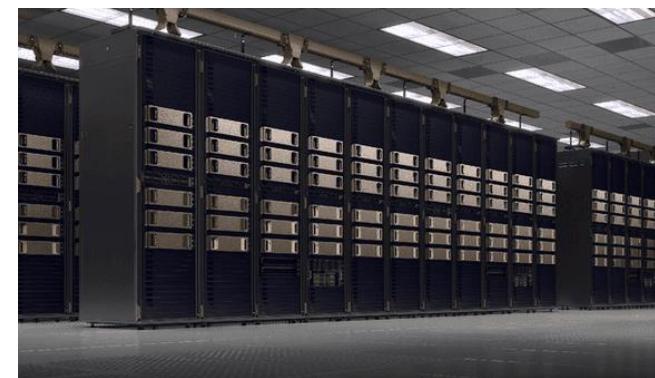
1. Supervised & Un-Supervised Learning verstehen
2. Was brauche ich für ein erfolgreiches ML Projekt?
3. Hands-on ausprobieren & experimentieren
4. Eigene Ideen / Projekte besprechen
5. Spass!

Block I

- Das Phänomen «Big Data»
- Data Mining
- Unterschied: Data Mining - Machine Learning
- AI, ML, DL – 3 Begriffe, aber was bedeuten sie?
- Beispiele moderner Machine Learning Projekte

Data Mining

- Automatisches Extrahieren von nützlichem Wissen aus grossen Datenbeständen.
- In der Regel, um die menschliche Entscheidungsfindung zu unterstützen.



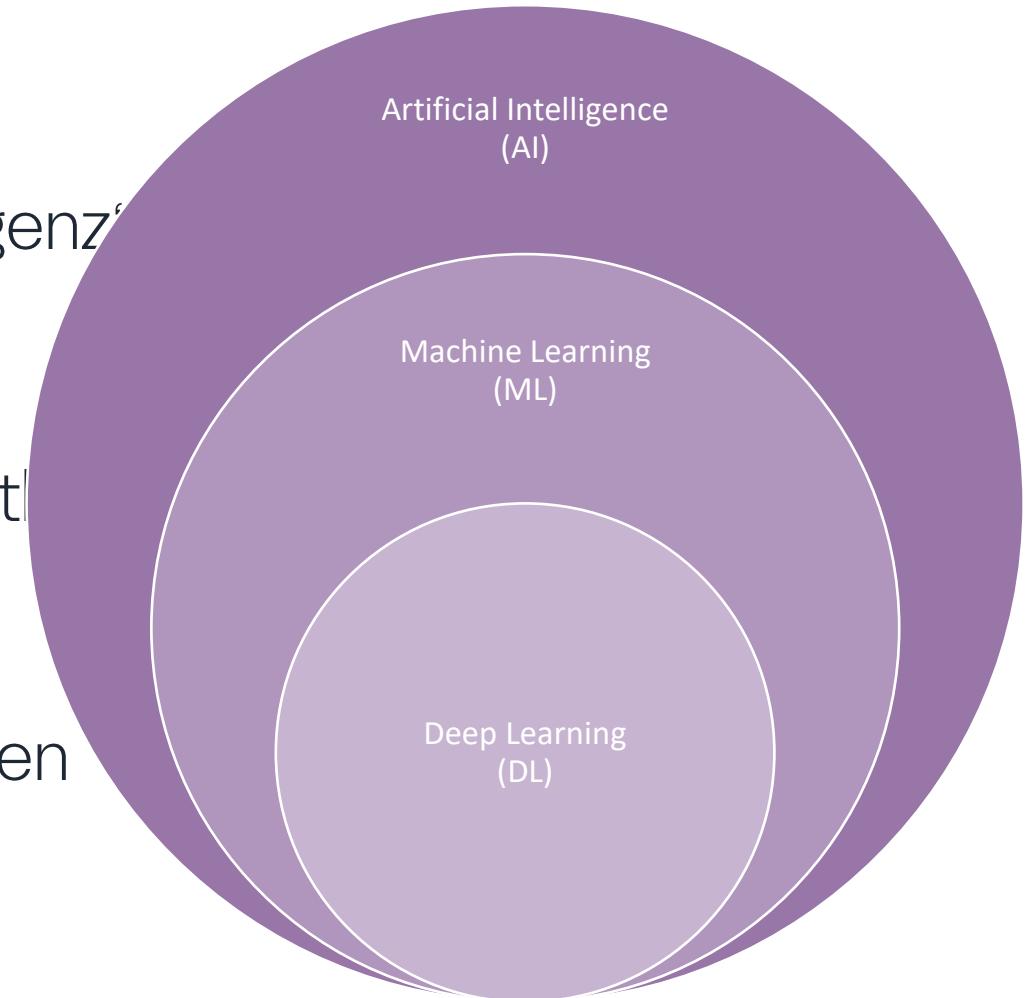
Machine Learning

- Einsatz von Computern zur **automatischen Erkennung von Mustern** in Daten und um **Vorhersagen** oder **Entscheidungen** zu treffen.
- Nützlich wenn:
 - Automatisation von menschlichen Prozessen
 - «Beyond Human Scope» - Schonmal 1TB Daten durchsucht 😲



Was ist AI, ML & DL?

- Artificial Intelligence (AI)
 - Intelligente Agenten, Autonome ‚Intelligenz‘
 - Deterministisch
- Machine Learning (ML)
 - Sammlung von mathematischen Algorithmen, die von Daten ‚lernen‘.
- Deep Learning (DL)
 - Machine Learning mit neuronalen Netzen
 - Deterministisch, aber *Fuzzy*



Beispiele für AI, ML, DL

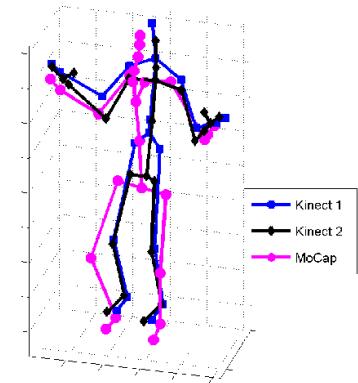
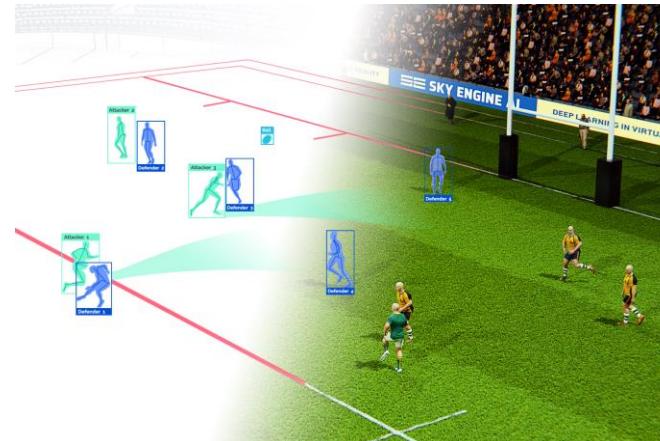
- Use Cases für AI:
 - Agenten Systeme (Google Maps, Board Games)
 - CSP Probleme (Stundenplan)
- Use Cases für ML:
 - Regressionsprobleme (Forecasting)
 - Random Forest (Kredit-Gewährleistung)
- Use Cases für DL:
 - Semantische Bilderkennung
 - Verarbeitung von Sprache (Übersetzung)
 - Reinforcement Learning (Computerspiel)
 - LLM / GenAI (Falcon-44b, GPT-4, Stable Diffusion, etc)

Muffin oder Chihuahua?

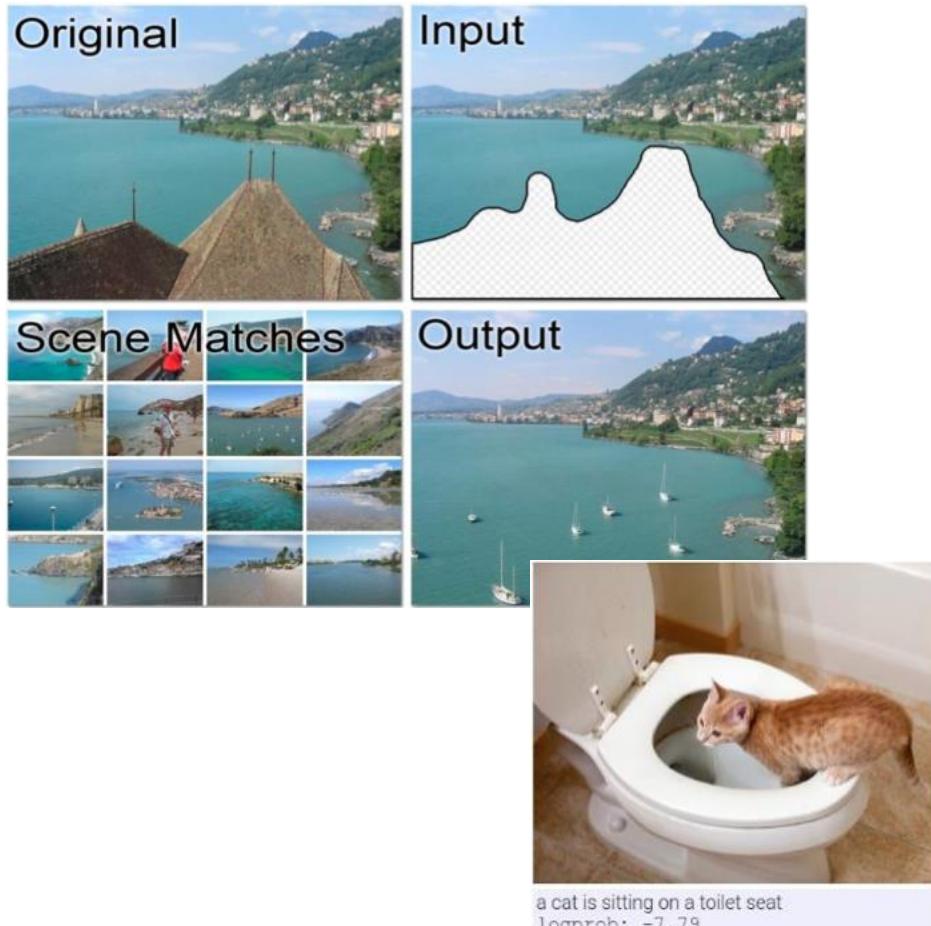


Mehr ML Use Cases

- Motion Capture
- Quality Control (Video)
- Movement Analysis (Hockey, Soccer)



Mehr Beispiele

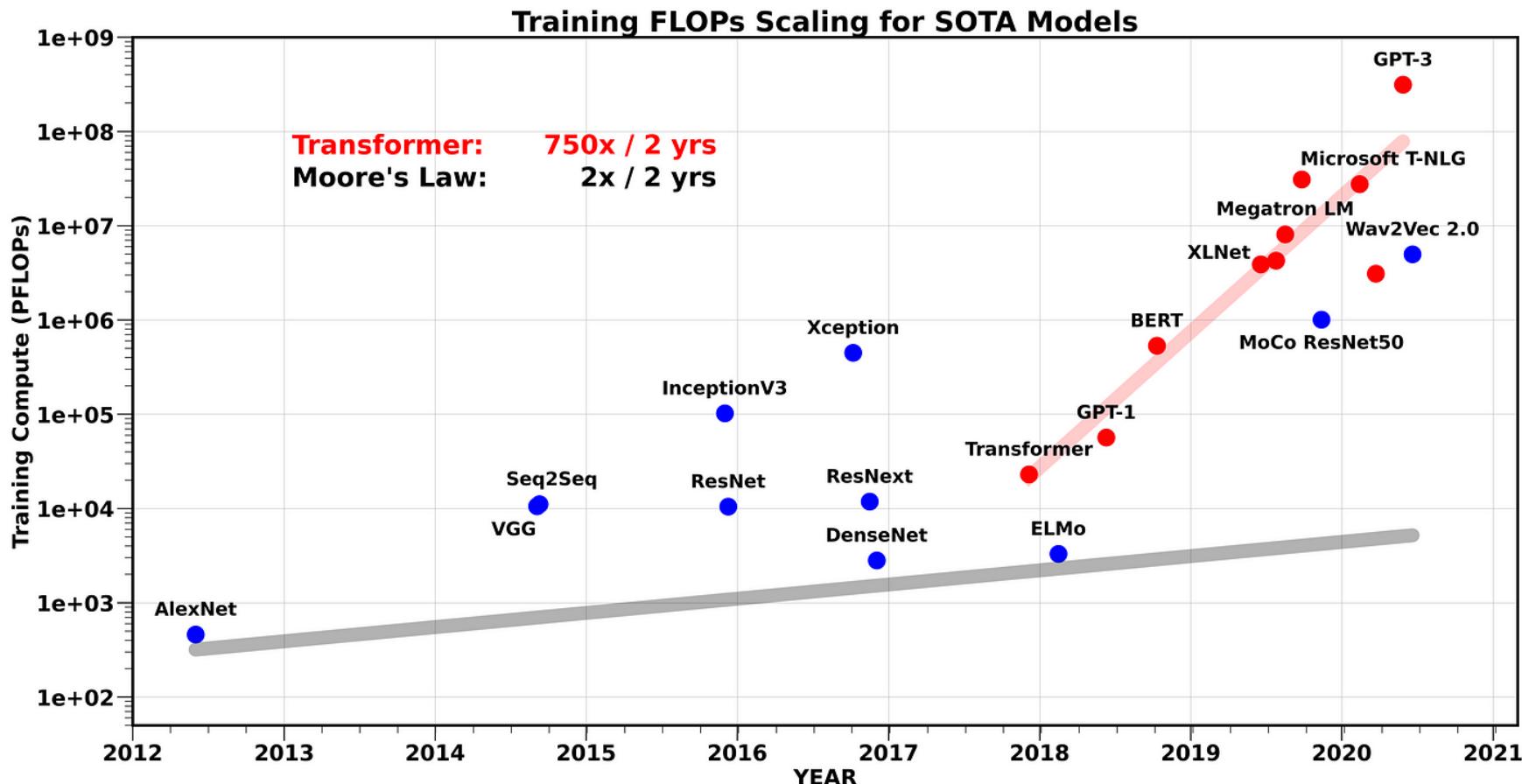


Treiber für den Deep Learning Hype

- 2006: IBM Blue Gene L
≈300TFLOPS
- 2022: Nvidia RTX4090 > 100 TFLOPS



Model Grösse & Computing Power



Zusammenfassung

- Mit etwas Statistik und einer Menge an Daten und Rechenpower – kann man einiges machen!
- Good times ahead:
 - Massive Fortschritte in Spracherkennung, NLP und Computer Vision
 - Extremer Innovationspace in kleinem Zeithorizont (3 – 5 Jahre)
 - NeurIPS Konferenz (unter 10min ausverkauft!)
 - Riesige Blase in AI Investments («AI» ist eigentlich ML, aber Marketing macht besser)
- Daher ist es wichtig, unsere **Grenzen** zu kennen:
 - “The combination of some data and an aching desire for an answer does not ensure that a reasonable answer can be extracted from a given body of data.” – John Tukey
 - Die meisten angewandten ML Projekte haben eine schlechte Generalisierung!

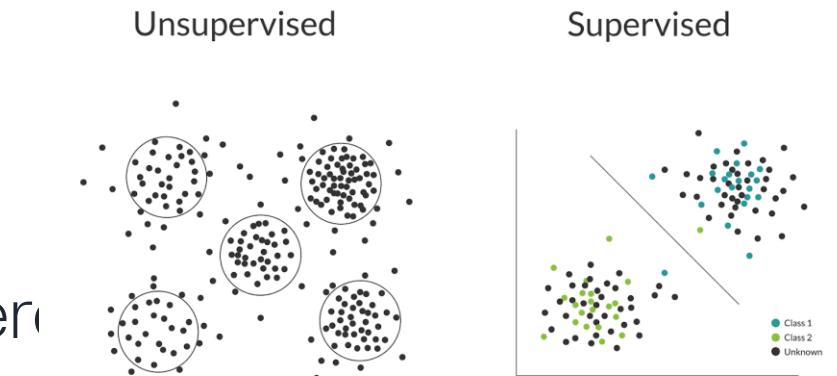
Block II

- Supervised, Unsupervised Learning
- Loss, Quality, Accuracy
- What are Labels?
- Training, Validation, Testing

- Datasets
- Bias – Variance Trade-off
- IID – Independent, Identically Distributed (IID)

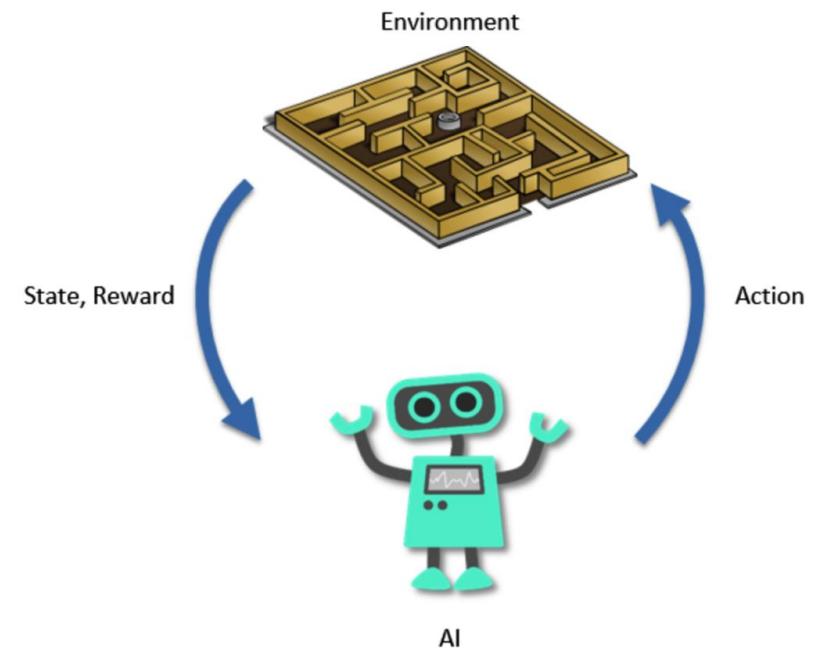
Supervised vs Unsupervised Learning

- Supervised Learning:
 - Vorhersage aufgrund von Daten treffen
 - Training Daten mit entsprechenden Labels
 - Model-Genauigkeit kann direkt bestimmt werden
- Unsupervised Learning:
 - «Verständnis» für Daten
 - Suche nach Struktur oder Mustern – Zusammenhängen
 - Nicht spezifisch für ein Label (Supervised)
 - Braucht entsprechend keine Labels
 - Evaluation oft indirekt oder nur qualitativ



Reinforcement Learning

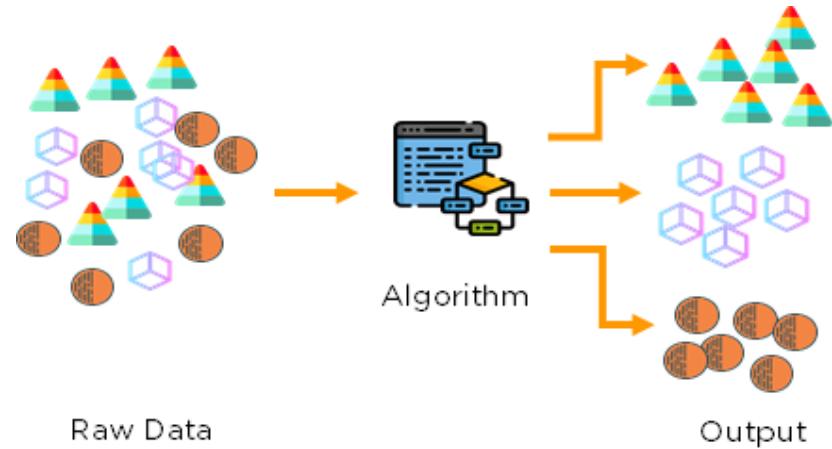
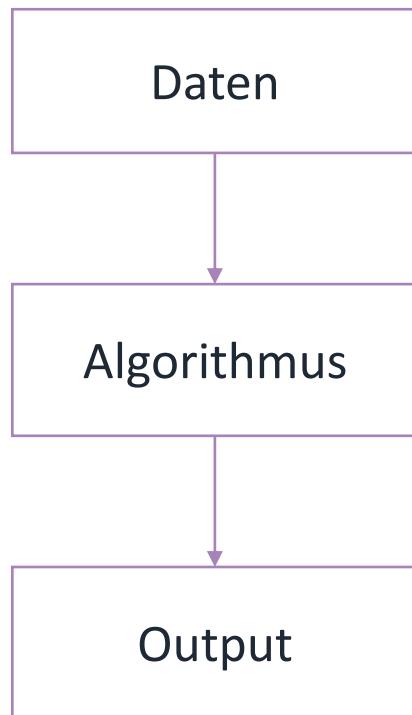
- Reinforcement Learning:
 - Zielgerichtetes Lernen
 - Belohnung bei Verbesserung des Resultats
 - Spiele, Regelungstechnik
- Example: Gran Turismo Sophy
<https://www.youtube.com/watch?v=AmtzhWilq0I>



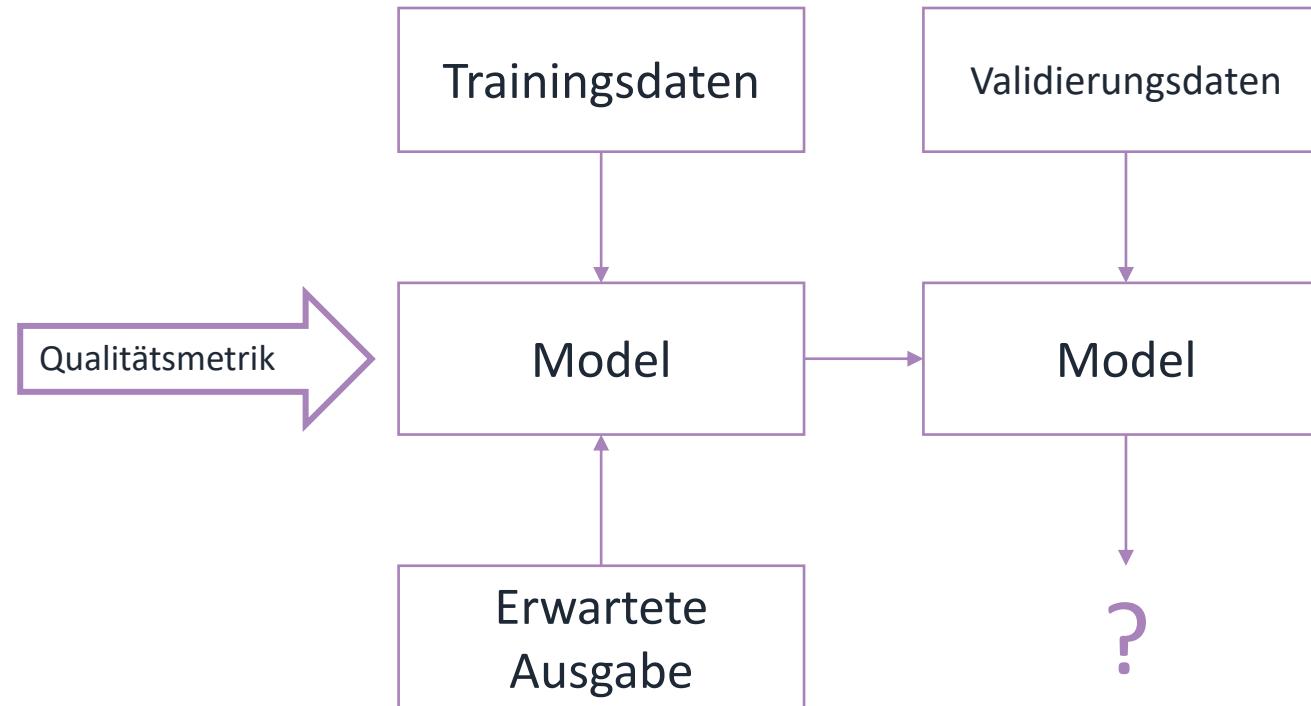
Supervised and Unsupervised, that's it?

- Semi-Supervised:
 - Wird oft gebraucht um supervised Algorithmen besser zu machen
 - Realität: Viele Daten, wenige haben Labels
 - Bsp: Network Intrusion Detection
- Vorteile:
 - Weniger Experten zur Erstellung des Datasets
 - Deutlich mehr Möglichkeiten als Unsupervised Learning

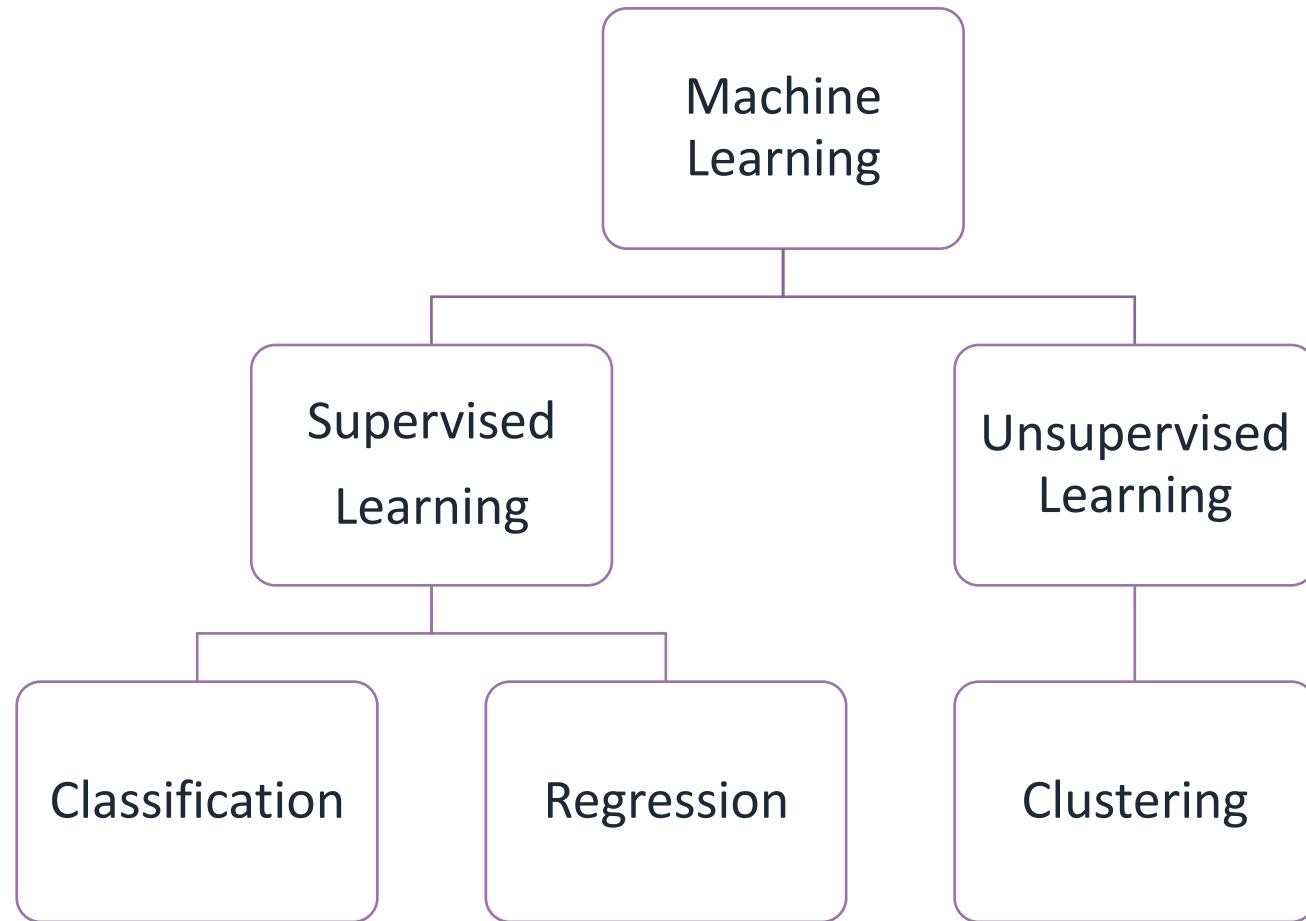
Unsupervised Modelle



Supervised Modelle



Methoden und Eigenschaften



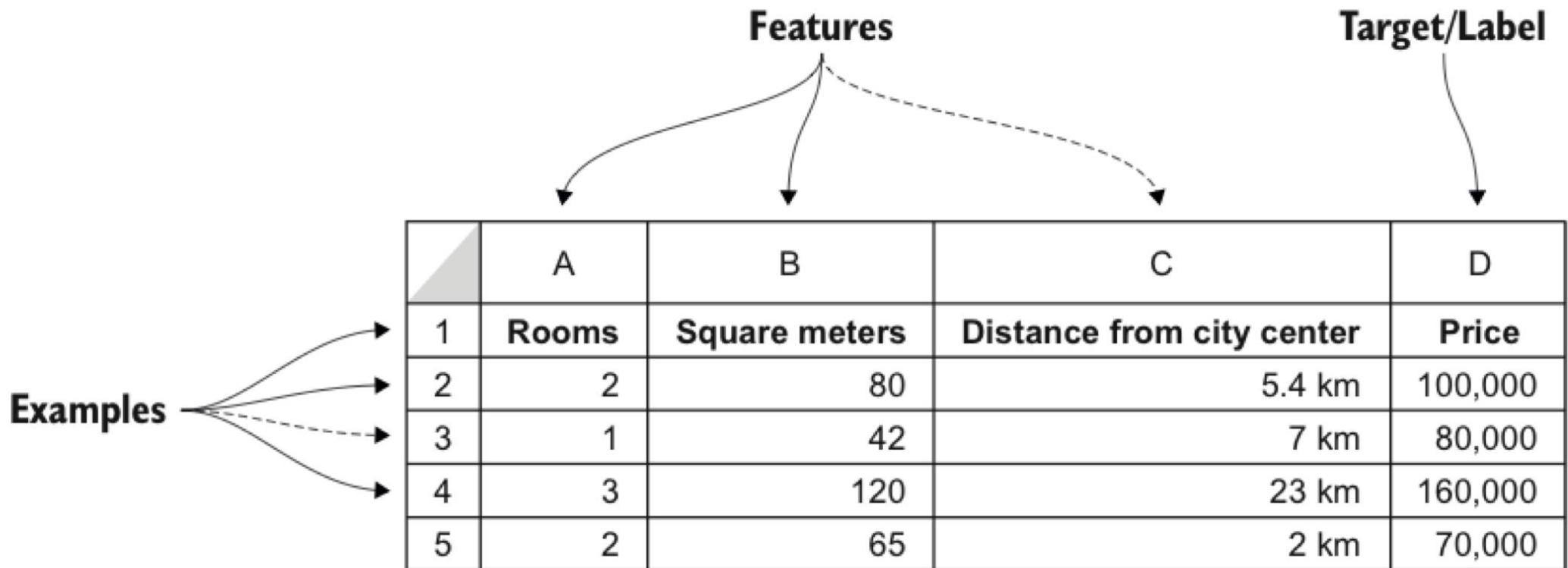
Methoden im Überblick

- Supervised:
 - Regression (isch das würkli ML?)
 - SVM (Support Vector Machine)
 - Decision Trees
 - Neuronale Netzwerke: Problem -> Label
- Unsupervised:
 - Clustering
 - Neuronale Netzwerke Problem -> Problem (Transformer)
- Semi-Supervised:
 - Neuronale Netzwerke: Problem -> Problem & Problem -> Label

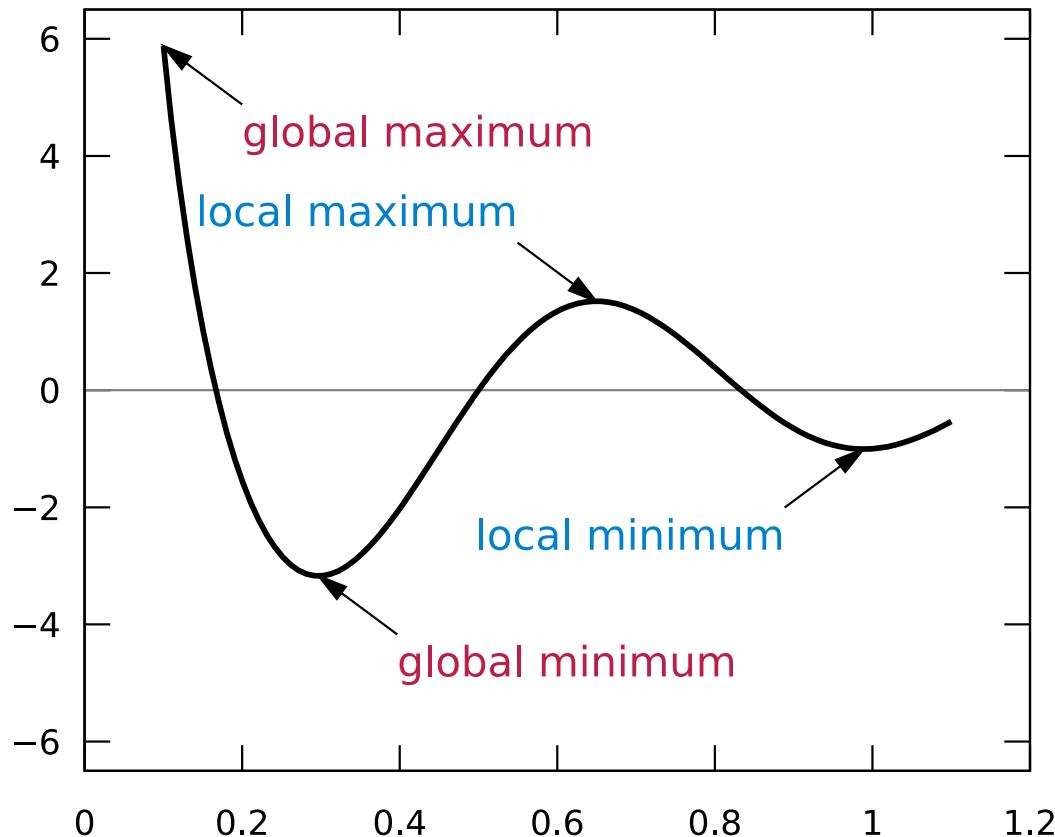
Was sind eigentlich diese Features?

- Ein Feature ist quasi ein ‘einzelner Datenpunkt’: **X, y**
 - Oft wird in Python X und y angetroffen, das kommt aus der mathematischen Beschreibung
- Numerische Werte können numerisch Abgebildet werden: 27 ist 27!
- Kategorische Werte über Enums und entsprechende Nummer:
 - Auto: 1, Motorrad 2
- Teilweise abstrakte Abbildung nötig:
 - Cat is at home
 - Dog is at home
 - Wird über 2 verschiedene Vektoren representiert

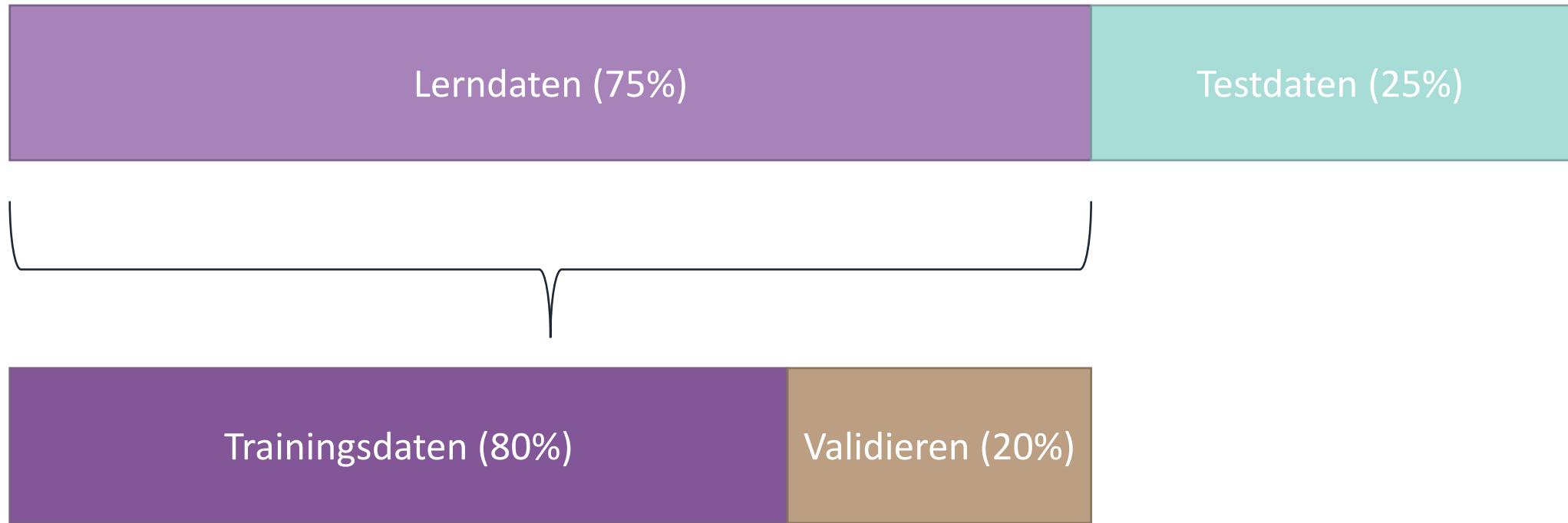
Features, Examples, Labels



Supervised Learning: Optimierungsproblem!



Datensplit: Wie machen wir QC?



Training, Validation & Testing

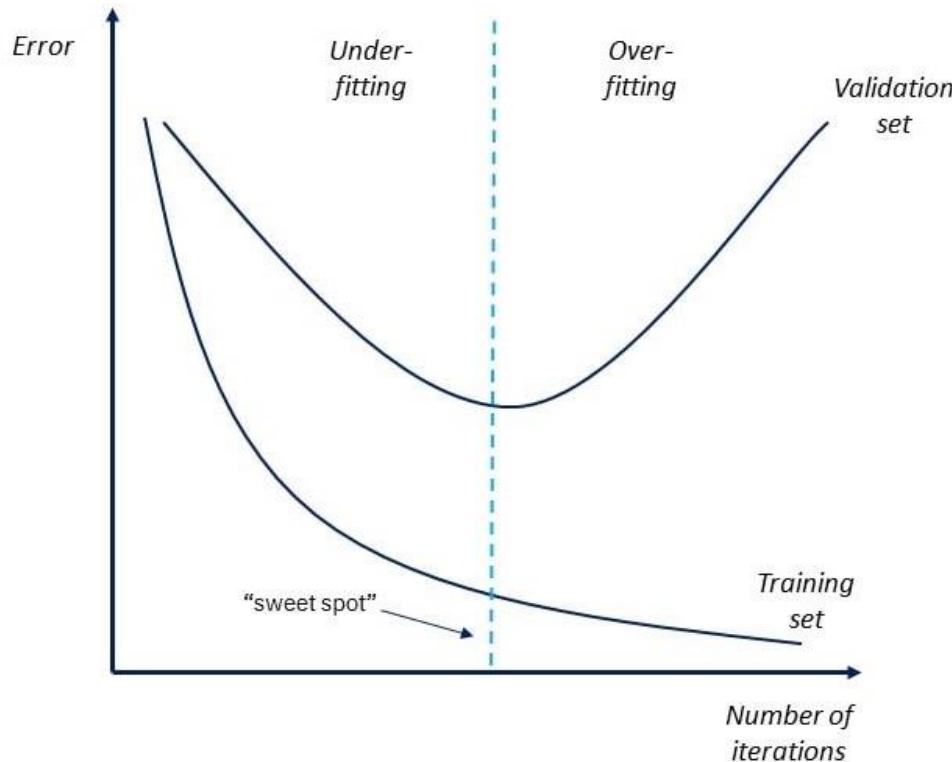
- Trainingsdaten (Training Set):
 - Werden gebraucht um das Model zu **trainieren!**
 - Diese Daten können beliebig manipuliert werden
- Validierungsdaten (Validation Set):
 - Validierung des trainierten Models
 - Überprüfung der Hyperparameter
 - Wie gut hat der Schüler gelernt?
- Testdaten (Test Set):
 - Unabhängiger Datensatz, der ausschliesslich der Qualitätsüberprüfung dient.
 - Daten Verteilung muss Analog zu Training Set & Validation Set sein! (10 Früchte, 3 Äpfel)

Hyperparameter?

- Hyperparameter sind Steuerungsmöglichkeiten, welche Einfluss auf das Model nehmen:
 - Welches Model trainiere ich? (Regression, Forest, SVM, etc?)
 - Hyperparameter werden nicht zur Trainingszeit gelernt, Sie bestimmen die gestalt des Models
 - Funktionsfitting: Nehme ich 2. oder 3. oder 4. Grad? (x^2 , x^3 , x^4 , etc.)
- Hyperparameter-Tuning wird oft auf Training und Validation-Set gemacht:
 - Grid Walk: Alle Möglichkeiten durchprobieren um bestes Parameterset zu finden.
 - Bias: wer zu oft das Validation-Set anschaut, weiss zuviel.

Overfitting & Underfitting

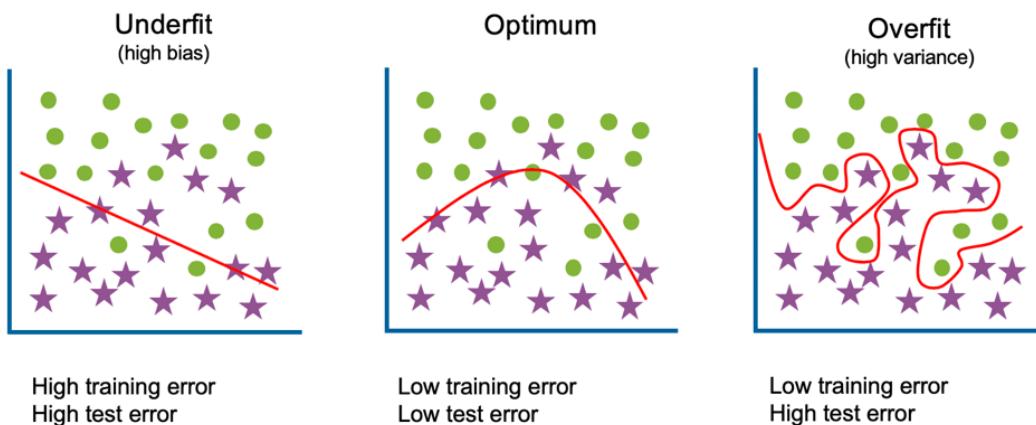
\Vee/MG



- Underfitting:
 - Training & Validation 'in-sync' aber schlecht!
 - Problem zu 'schwierig' für Model
 - Anspruchsvolleres Model
 - Mehr 'Trainingszeit'
 - Mehr Features, Mehr Daten
- Overfitting:
 - Training gut – Validation schlecht
 - Mehr Daten
 - Zu komplexes Model (weniger Parameter)

Bias Variance – Trade-off

\Vee/MG



- Underfit
 - Zu “simplistisch”
 - Was wenn wir viel ‘Noise’ haben?
- Optimum:
 - Das suchen wir!
- Overfit:
 - Schlechte Generalisierung

Fragen

Warum nimmt der Training-Fehler (training error) monoton ab?
(bei steigender Model-komplexität)

Fragen

Warum nimmt der Training-Fehler (training error) monoton ab?
(bei steigender Model-komplexität)

- Das grössere Model, ermöglicht einen grösseren Suchraum
- Mehr Parameter ermöglichen 'mehr zu lernen'

Fragen

Warum nimmt der Test Fehler (test error) zu beginn ab?

Fragen

Warum nimmt der Test Fehler (test error) zu beginn ab?

- So lange das Model under-fitted kann mehr gelernt werden, Training und Test error nehmen ab.
- Höhere Model-komplexität ermöglicht mehr zu lernen

Fragen

Warum nimmt der Test Fehler an einem gewissen Punkt wieder zu?

Fragen

Warum nimmt der Test Fehler an einem gewissen Punkt wieder zu?

- Over-fitting
- Das Model lernt «Noise», nicht aber das «Signal»

Fragen

Weshalb ist der Bias-Variance trade off so wichtig?

Fragen

Weshalb ist der Bias-Variance trade off so wichtig?

- Die Balance zwischen Under-fitting und Over-fitting ist das »beste« Modell
- Wenn die Trainingsresultate zu gut sind, ist vermutlich etwas faul

Recap: Supervised Learning Notation

- X : Training Data
 - y : Training Label
 - \hat{y} : Predicted Label
-
- \tilde{X} : Test Data
 - \tilde{y} : Test Labels

Recap: Supervised Learning Notation

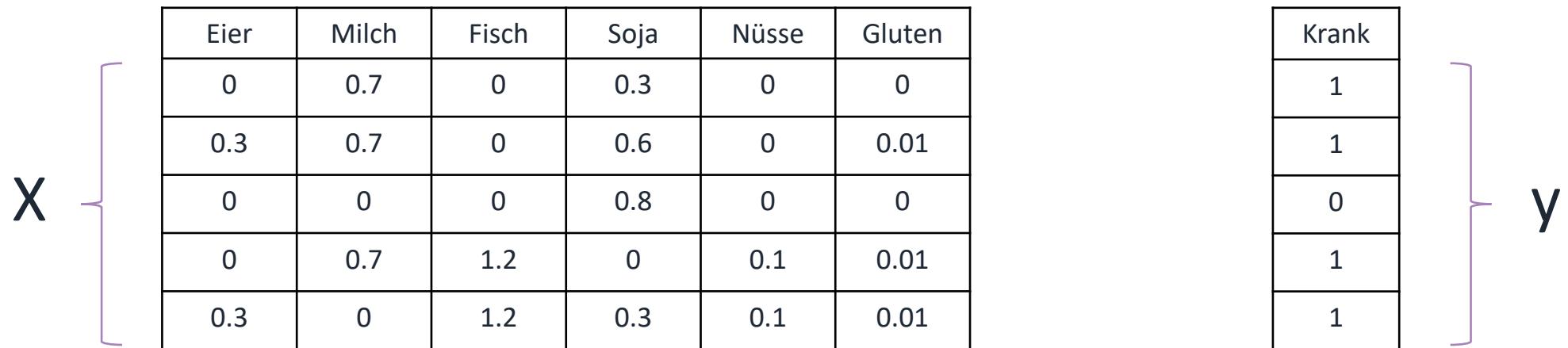


Diagram illustrating Supervised Learning Notation:

Feature Matrix X (left):

	Eier	Milch	Fisch	Soja	Nüsse	Gluten
0	0	0.7	0	0.3	0	0
0.3	0.3	0.7	0	0.6	0	0.01
0	0	0	0	0.8	0	0
0	0	0.7	1.2	0	0.1	0.01
0.3	0.3	0	1.2	0.3	0.1	0.01

Label vector y (right):

Krank
1
1
0
1
1

- Feature Matrix X : Spalte ist Feature, Zeile ist Datensample für Kategorie
 - X_{ij} entspricht Menge an Essen j an Tag i
 - X_i entspricht der Nahrung an Tag i
- Label y : Enthält ob "Krank" oder "nicht Krank"

Können wir überhaupt lernen?

- Sagt unser Training Error etwas über den Test Error aus?
 - NEIN: Testdaten können möglicherweise nichts mit Trainingsdaten zutun haben
 - Einnahme von Medikamenten wurde nicht dokumentiert (Laktose-Tabletten)

X			y	X̃			ỹ
Eier	0	0.7	1	0	0.3	0.7	0
	0.3	0.7	1	0	0	0	0
	0	0	0	0	0	0	0
Milch	0.7	0	1	0	0.3	0.7	0
	0	0	1	0	0	0	0
	0	0	0	0	0	0	0
Fisch	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0

- Um zu lernen, müssen wir eine Annahme treffen:
 - Trainingsdaten und Testdaten müssen im Zusammenhang stehen
 - Die einfachste Annahme: **Independent and identically distributed (IID)**

Dataset Splitting:

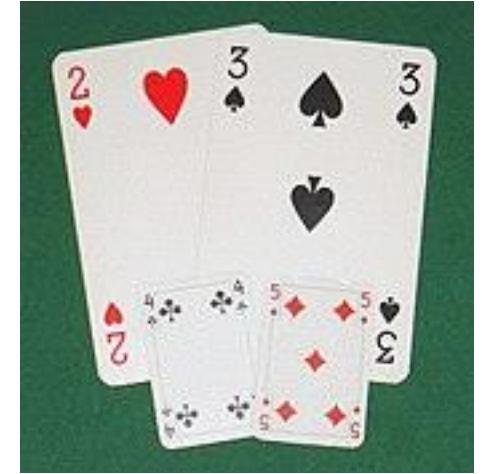
- Grundsätzlich muss Training-Set, Validation-Set, Test-Set gleiche **Verteilung** haben!
- Aus 100 Features sind: x Training, y Validation, z Test
 - x, y, z sind ‘in sich’ gleich!
- Gleichzeitig bedenken wir aber:
 - Haben wir ‘ungleichmässige’ Verteilungen?
 - Sind wir zeitliche gebunden? (Börsenkurse, Fabrik)

IID Assumption

- Independent and identically distributed (IID):
 - Unabhängig und identisch verteilte Zufallsvariablen
- IID Assumption ist erfüllt wenn:
 - Alle Beispiele haben / kommen von der gleichen Verteilung
 - Die Beispiele sind unabhängig (Reihenfolge spielt keine Rolle)
- Beispiel:
 - Trainingset X und Testset \tilde{X} enthält blaue und rote Kugeln
 - Wahrscheinlichkeit P_{blau} ist in Menge X und \tilde{X} : gleich

IID Assumption: Kartenspiel

- Französisches Blatt:
 - 4 Farben (Kreuz, Pik, Herz, Karo)
 - 9 Wertigkeiten (6-10, Bube, Dame, König, Ass)
- Ist IID Assumption erfüllt?
 - Ziehen einer Karte (oberste), zurücklegen, mischen, wiederholen
 - Ziehen einer Karte, zurücklegen, wiederholen
 - Ziehen einer Karte, *nicht* zurücklegen, mischen, wiederholen



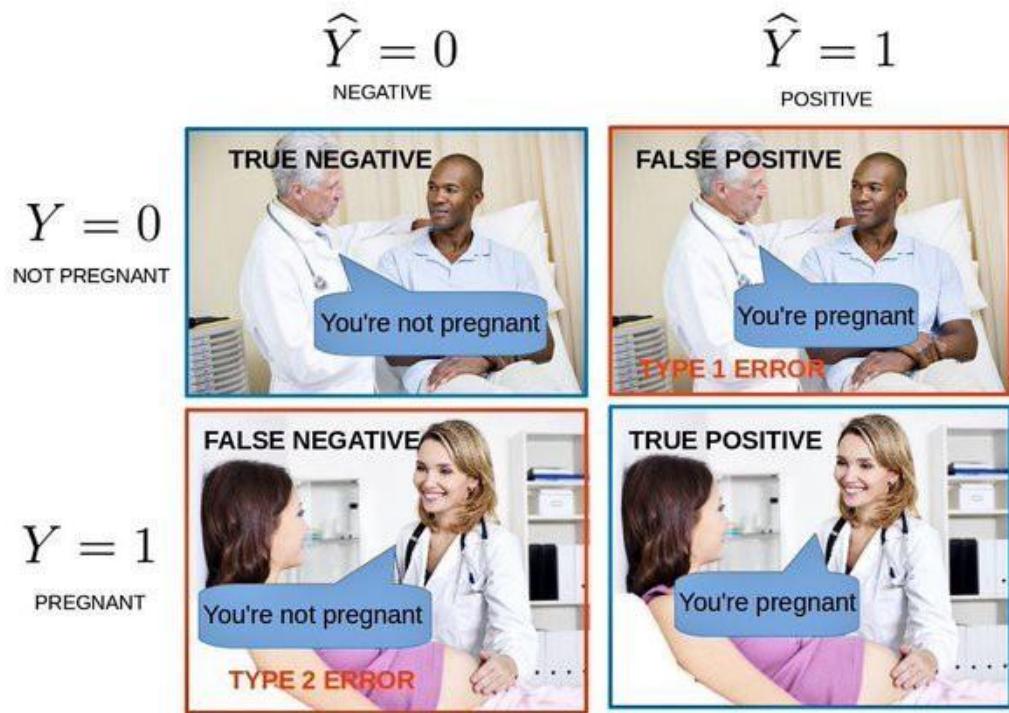
Optimisation Bias

- Nehmen wir einen multiple-choice Test (a,b,c,d) mit 10 Fragen:
 - Zufällig ausgefüllt: 25% richtig
 - Zufällig 2x ausgefüllt und den Besseren gezogen: 33% richtig
- Skalieren wir das Experiment auf 10 Tests: 47%
 - Bei 100: 62%
 - Bei 1000: 73%
- Wenn man viele Versuche hat, dann hat man die Möglichkeit es gut zu tun.
- Bester Score mit *Random.seed(42)*!

Metriken

- Confusion Matrix für 2-Klassen Klassifikation
- Metriken: Accuracy, Recall, Precision, F1-Score
- Unterschied zwischen Recall und Precision
- Decision Threshold: ROC Kurve und AOC Fläche

Confusion Matrix

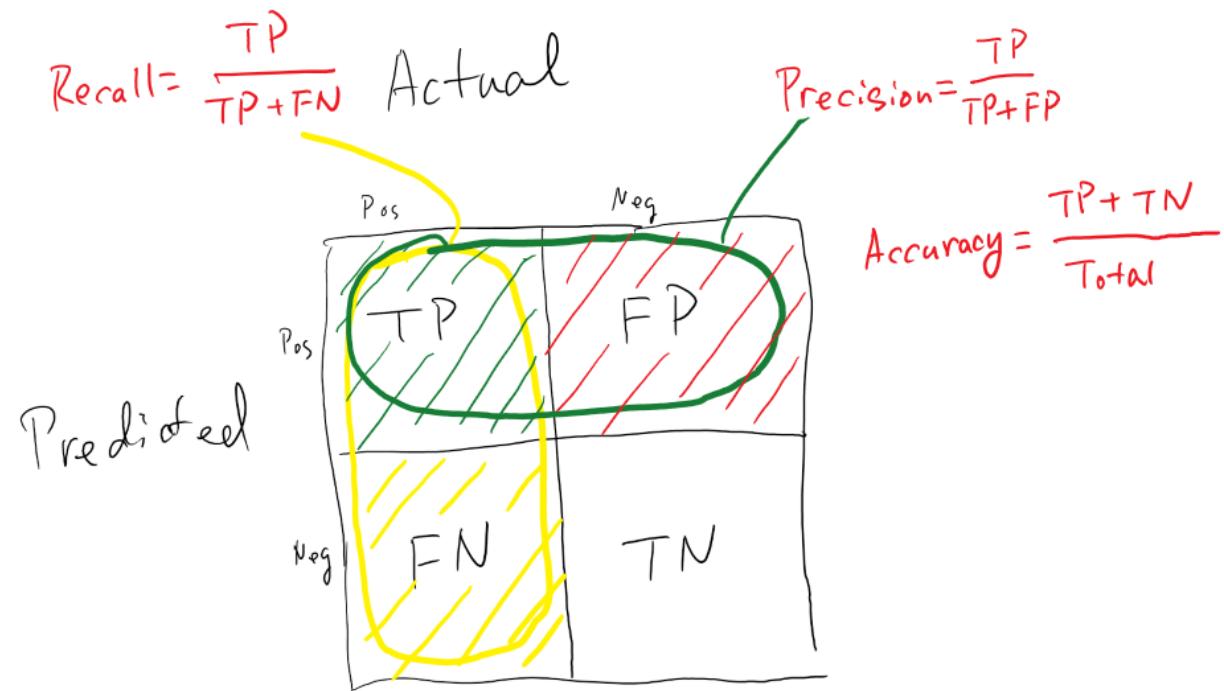


		Echte Werte (Actual Values)	
		1	0
Vorhergesagte Werte (Predicted Values)	1	True Positive (TP)	False Positive (FP): Type 1 Error
	0	False Negatives (FN): Type 2 Error	True Negatives (TN)

Metriken

- $Accuracy = \frac{Correct}{Total} = \frac{TP+TN}{TP+TN+FP+FN}$
- Recall: 'Wieviele relevante Daten habe ich erwischt?'
 - $Recall = \frac{TP}{TP+FN}$
- Precision: 'Wieviele ausgewählte Daten sind relevant?'
 - $Precision = \frac{TP}{TP+FP}$
- $F1 - Score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} = 2 * \frac{Precision*Recall}{Precision+Recall}$

Precision und Recall



Case 1: Covid

- Kosten für FN > Kosten für FP
- Labels:
 - Covid-Positive: 1
 - Gesund: 0

		Echte Werte (Actual Values)	
		1 (hat Covid)	0 (gesund)
Vorhergesagte Werte (Predicted Values)	1	True Positive (TP)	False Positive (FP): Type 1 Error
	0	False Negatives (FN): Type 2 Error	True Negatives (TN)

Hat Covid, aber Diagnose sagt gesund!

Case 1: Covid

- TP sind Covid Patienten mit Covid Diagnose
- TN sind gesunde Patienten mit gesunder Diagnose
- FP: Schade 10 Tage Quarantäne für nichts!
- FN: Potentielles *Risiko*, da Ansteckungsmöglichkeiten!
- FN ist für das System teurer als FP: *Recall*

Case 2: Spam

- Kosten für FN < Kosten für FP
- Labels:
 - Spam: 1
 - Kein Spam: 0

		Echte Werte (Actual Values)	
		1 (Spam)	0 (not Spam)
Vorhergesagte Werte (Predicted Values)	1	True Positive (TP)	False Positive (FP): Type 1 Error
	0	False Negatives (FN): Type 2 Error	True Negatives (TN)

Das Jobangebot von Google

Case 2: Spam

- TP sind Spam Mails, welche als Spam Mails erkannt werden
- TN sind legitime Mails, welche als solche erkannt werden
- FP: Das Job Angebot von Google
- FN: Der Persische-Prinz und sein Millionen-Erbe
- FP hat weittragendere Konsequenzen als FN: *Precision*

Übung: Kreditvergabe

- Zeit 10min
- Überlegen Sie mit Ihrem Partner das Szenario Bankkredit
 - Label 1: Schlechter Kredit (wird nicht bezahlt)
 - Label 0: Guter Kredit (wird bezahlt)

Nachtrag: F1-Score vs Accuracy

- Accuracy: Fokus auf TP und TN
- F1: Fokus auf FP and FN
- Accuracy ist ‘intuitiv’, F1 ist etwas ‘künstlicher’
- Accuracy funktioniert gut, wenn Klassen in balance
- F1-Score funktioniert aber auch, wenn Klassen disbalance besteht
- In der Realität ist F1-score nützlicher als Accuracy.
 - *Ich oute mich hier: primäres Augenmerk auf Accuracy und auf Confusion Matrix.*

Label Prediction: Probabilistisch

- Labels werden in allen stochastischen ML Modellen probabilistisch vorhergesagt:
 - Die Abbildung kann somit $[0;1]$ umfassen
 - Je näher an 0 desto eher ist es 0.
 - Je näher an 1 desto eher ist es 1.
- **Gutes Model:** Scharfe Trennung
- Threshold-Tuning möglich

Warum Neuronale Netzwerke?

- Popularität in Hype-Wellen:
 - Zur Zeit: Eines der heissten Themen in der Wissenschaft
- Aktuelles Hoch getrieben durch unbeschreibliche Performance in:
 - Computer Vision Tasks (Bild Erkennung, Semantische Segmentierung)
 - Language Recognition (Speech 2 Text)
 - Language Models (Sprach Verständnis, Interpretation)

Warum Neuronale Netzwerke II?

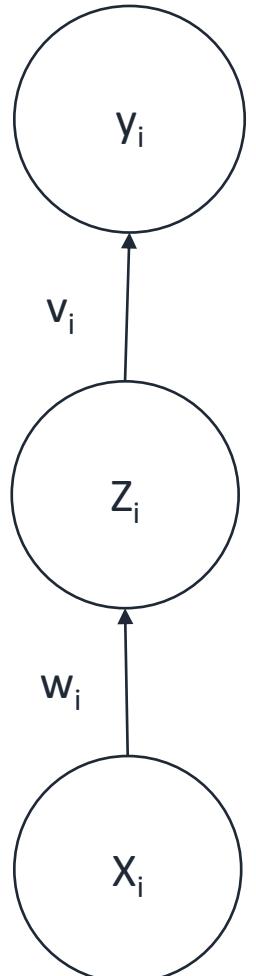
- Haupt-Treiber für den Erfolg:
 - Grosse qualitativ hochwertige Datensets: 100TB+
 - Sehr tiefe Netze (Deep Learning)
 - Unglaubliche Computerleistung zu 'keinem' Geld.
- Einige Tweaks an der Struktur der Netzwerke
 - CNN (Convolution)
 - LSTM (Long—Short Term Memory)
- Mathematik praktisch unverändert seit 1960...

Neural Networks: Motivation

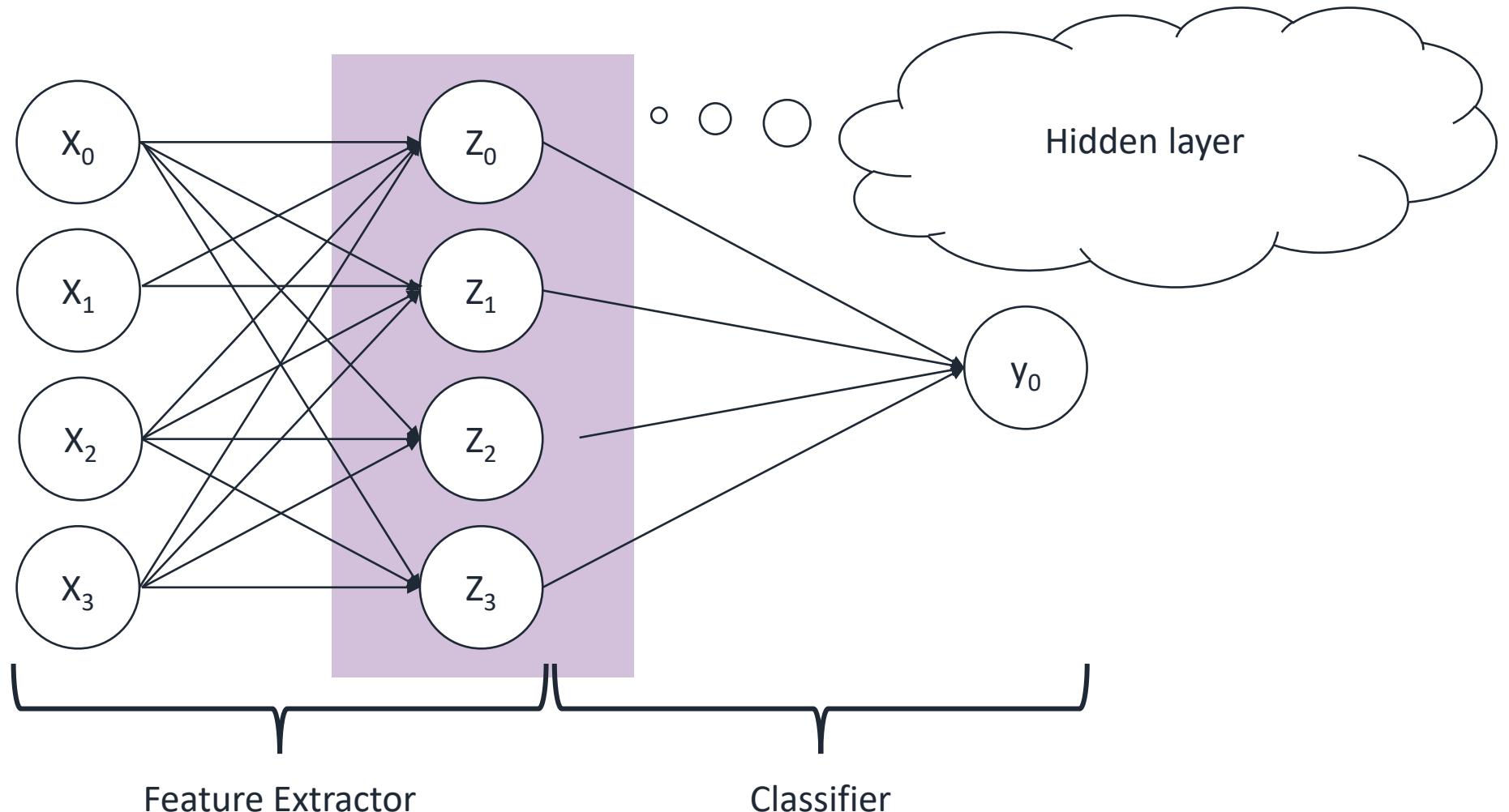
- Viele Domänen brauchen eine nicht lineare Transformation der Features
 - Aber manchmal ist es offensichtlich
 - Bsp. SciKit learn
- Neuronale Netzwerke versuchen diese Transformation zu lernen
 - Optimierung der ‘Feature’ für bestes Resultat
- Wir fangen mit einem hidden-layer an und gehen dann Deep...

Supervised Learning Roadmap

- **Level 1:** «Direct» Supervised Learning
 - Wir lernen Parameter w basierend auf x_i und y_i .
- **Level 2:** Basis Transformation
 - Wir lernen Parameter v basierend auf Basis Z_i und y_i . ($x_i \rightarrow Z_i$ existiert)
- **Level 3:** PCA, t-SNE, Latent-Factor Models (Manifolds)
 - Wir können Parameter w von Basis Z_i aufgrund von X_i erlernen
 - Wir können Parameter v für $Z_i \rightarrow y_i$ lernen. (Classification)
- **Level 5:** Neuronale Netzwerke
 - w und v werden direkt basierend auf X_i und y_i gelernt
 - Basis Z_i wird impliziert gelernt, welche gut geeignet für Supervised Learning Classification ist.

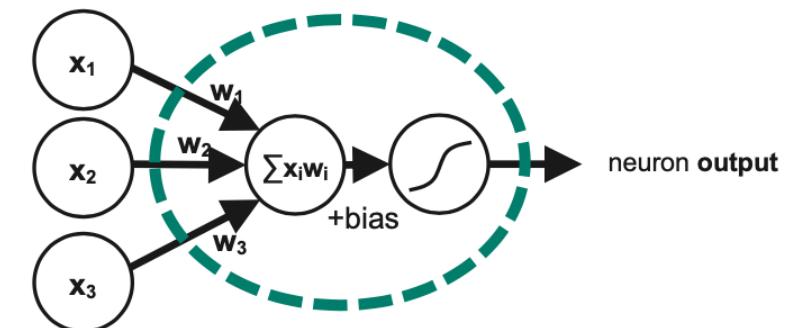


Feature Extractor & Classifier

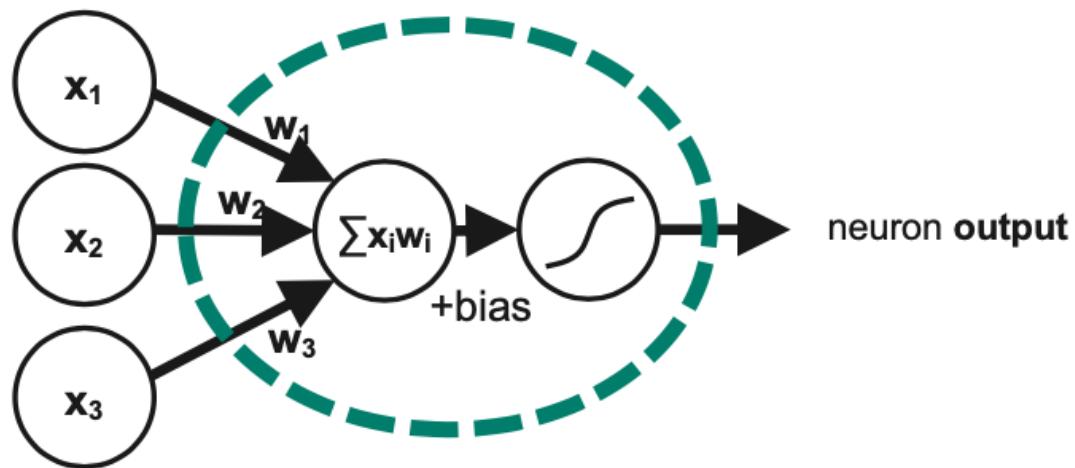


Neuronales Netz als Funktion

- Das Neuron erhält Input $[x_1, x_2, \dots]$
- Jedes Neuron besitzt eigene Eingangsgewichte W , einen Bias b .
- Output entspricht der Summe der gewichteten Inputs



Neuronales Netz als Funktion II



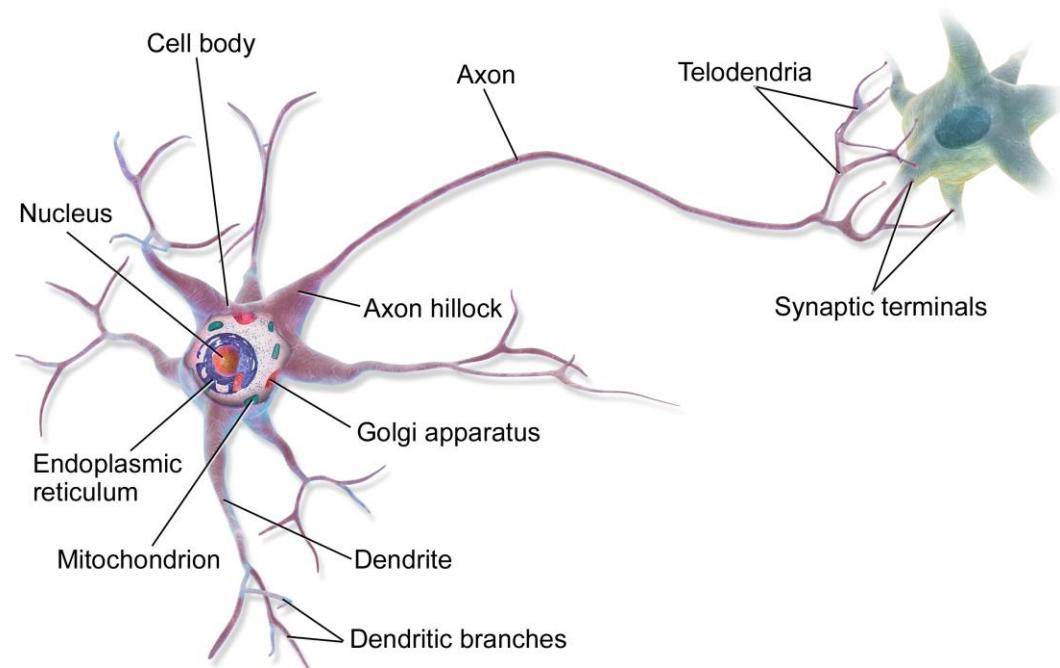
$$\sigma \left(\sum_i w_i f_i + b \right)$$

Aktivierungsfunktion?

- Avoid Linear Combinations:
 - $A^*x + b^*x = (a+b)^*x$

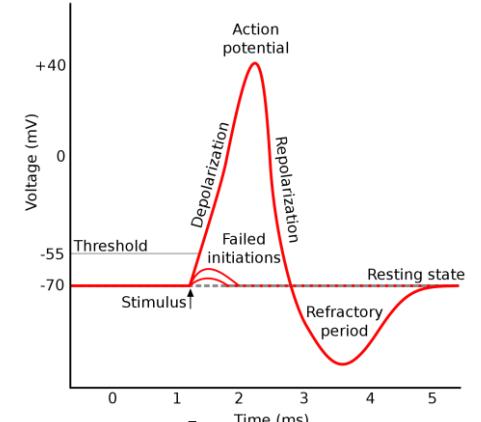
Biologische Sicht

\Vee/MG



<https://en.wikipedia.org/wiki/Neuron>

- Neuron hat 'Dendriten' als Inputs
- Neuron hat 'Axon' als einzelnen Output
- Mit dem richtigen Input, wird ein Output getriggert

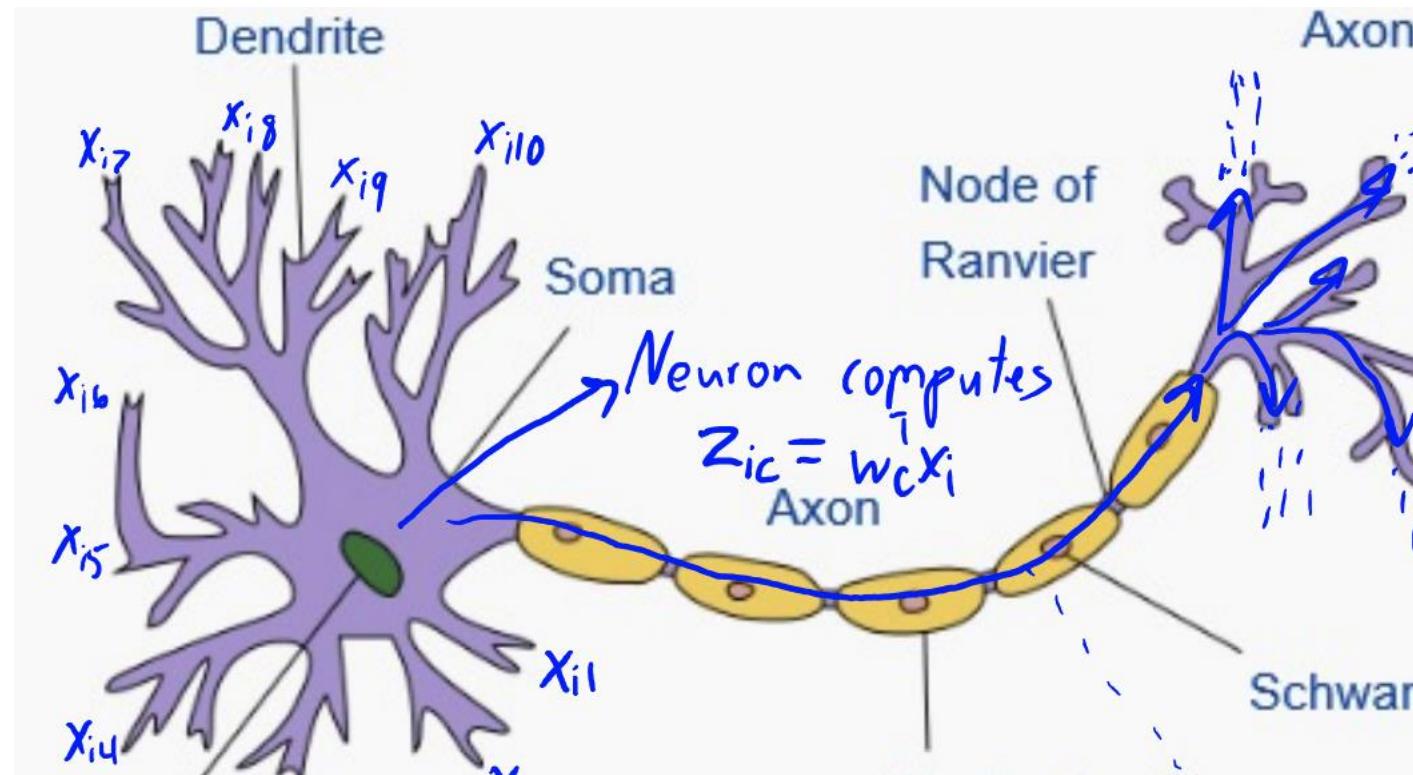


\Vee/MG

Martin Stypinski & Christian Fässler @ CH Open 2024

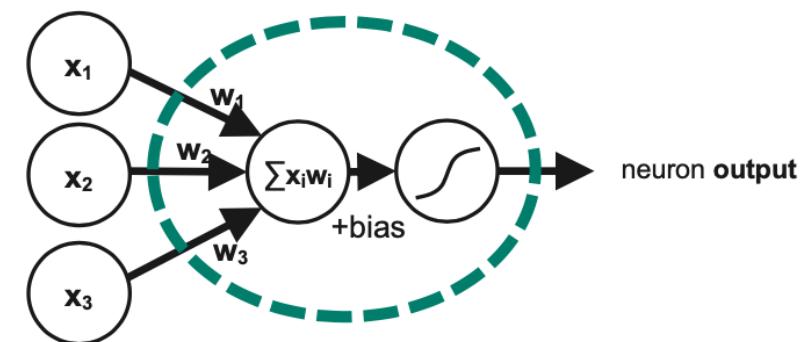
adnexo

Biologische Sicht II



Single Layer Perceptron

- Fähigkeit linear separierbare Muster zu lernen
- Hohe Input Dimensionalität -> lineare Separierbarkeit, aber im n-dimensionalen Raum
- Terminologie:
 - Perceptron ist ein 'Einzelnes Neuron'
 - Perceptron ist aber auch der Algorithmus zum trainieren eines binären Klassifizierers (Classifier-part)



- Layer können geschichtet (stacked) werden. Diese Architektur nennt man Multi Layer Perceptron MLP.
- Lasagne war eines der ersten Deep Learning Frameworks! ☺

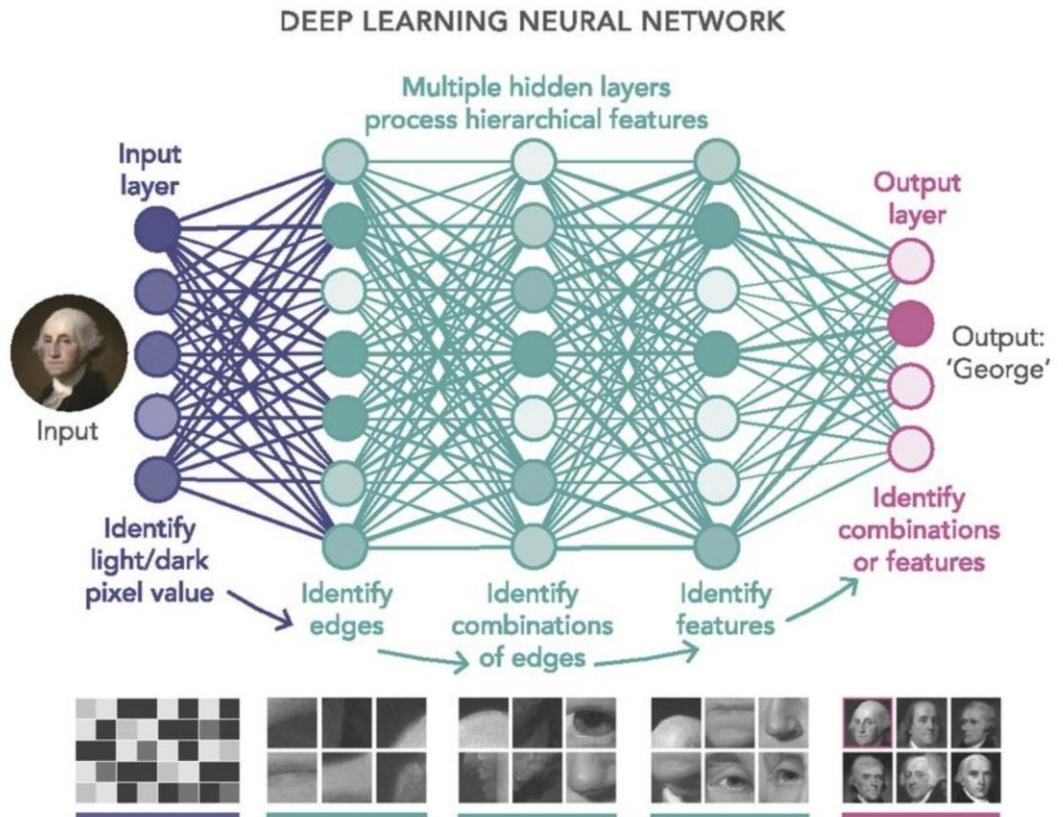
Stacking Layers -> Deep Learning

- Der Output des jeweiligen Layers, entspricht dem gewichteten Input des nächsten.
 - Komplexe Features kombinieren und aggregieren Informationen durch die Netztiefe
 - Alle Parameter werden mithilfe des Backpropagation Algorithmus erlernt. (Vorlesung Oliver am Donnerstag).
- Aktuelle KI ist angetrieben durch tiefe Modelle, mit Miliarden von Parametern (auch GPTv4!)
- Die freien Parameter werden durch **UNMENGEN** an Daten ‘trainiert’.

Visualisierung: Tiefe Neuronale Netzwerke

\Vee/MG

- Tiefe neuronale Netzwerke werden auch Deep Neural Networks genannt
- ANNs lernen Features zu erkennen und kombinieren in tieferen Schichten diese zu komplexeren Mustern
- Das Resultat ist nichts anderes als eine komplexe mathematische Funktion welche X_i auf y_i abbildet, bestehend aus vielen einfachen Komponenten.



<https://www.positronic.ai/consulting/deep-learning/>

Kleines Gedankenexperiment

- Schauen Sie sich die vorherige Folie an:
 - Wie würden Sie das Netz implementieren?
 - Besprechen Sie die Konzepte mit der Person neben ihnen.
- Implementieren Sie nicht!

Künstliche NN vs Reale NN

- Künstliche NN:
 - X_i ist Feature Darstellung der Welt
 - Z_i ist interne Darstellung der Welt
 - Y_i ist Output / Klassifikation o. Regression
- Reale NN:
 - Timing / Zeit-diskrete Aktion sind wichtig (Kontinuierliches Signal)
 - Hirn ist 'höchst organisiert' (Bsp. Strukturen für diverse Tasks)
 - Verbindungen -> Struktur Änderungen
 - Unterschiedliche Neurotransmitter (Elektrochemisch, Hormonell, etc.)

Agenda

- Einführung klassische ML Methoden
- Terminology
 - Models, Data, Cost Function
- Beispiele
 - Decision Tree
 - KNN
- Lab: Get your hands dirty

Was ist ein Modell

- Approximation einer Funktion und Optimierung
- Annahme
 - Es existiert eine Funktion / Beziehung / Set von Regeln
 - Um Daten zu interpretieren
 - $g(x)$ ist die «gegebene» Funktion
- Approximation
 - $f(\hat{x}, p)$

Model vs Algorithm

- Ein **Algorithmus** wird verwendet um ein **Model** zu trainieren
- Ein **Algorithmus** ist eine Funktion
 - Wir füttern dem Algorithmus Daten und er Produziert ein Modell
 - **Model** = **Algorithm**(Data)
- «Models are the specific representations learned from data»
 - Beispiel
 - $f(x) = ax+b$ ist ein **Algorithmus**
 - $f(x) = 4x + 2$ ist ein **Modell**

Äpfel & Birnen Klassifizierung

\Vee/MG

- Ziel: Algorithmus designen, um Äpfel und Birnen zu unterscheiden basierend auf Breite und Höhe
- Fragestellung:
Bei gegebener Breite und Höhe, was ist die Obst Klasse?
- Wie würden Sie das als Programmierer machen?

Width	Height	Fruit
7.1	7.3	Apple
7.9	7.5	Apple
7.4	7.0	Apple
8.2	7.3	Apple
7.6	6.9	Apple
7.8	8.0	Apple
7.0	7.5	Pear
7.1	7.9	Pear
6.8	8.0	Pear
6.6	7.7	Pear
7.3	8.2	Pear
7.2	7.9	Pear

Decision Tree

\Vee/MG

- Einfache if else Struktur

```
def get_fruit_class(width, height):  
    if width < 7.35:  
        if height < 7.4:  
            return 'Apple'  
        return 'Pear'  
    return 'Apple'
```

Width	Height	Fruit
7.1	7.3	Apple
7.9	7.5	Apple
7.4	7.0	Apple
8.2	7.3	Apple
7.6	6.9	Apple
7.8	8.0	Apple
7.0	7.5	Pear
7.1	7.9	Pear
6.8	8.0	Pear
6.6	7.7	Pear
7.3	8.2	Pear
7.2	7.9	Pear

\Vee/MG

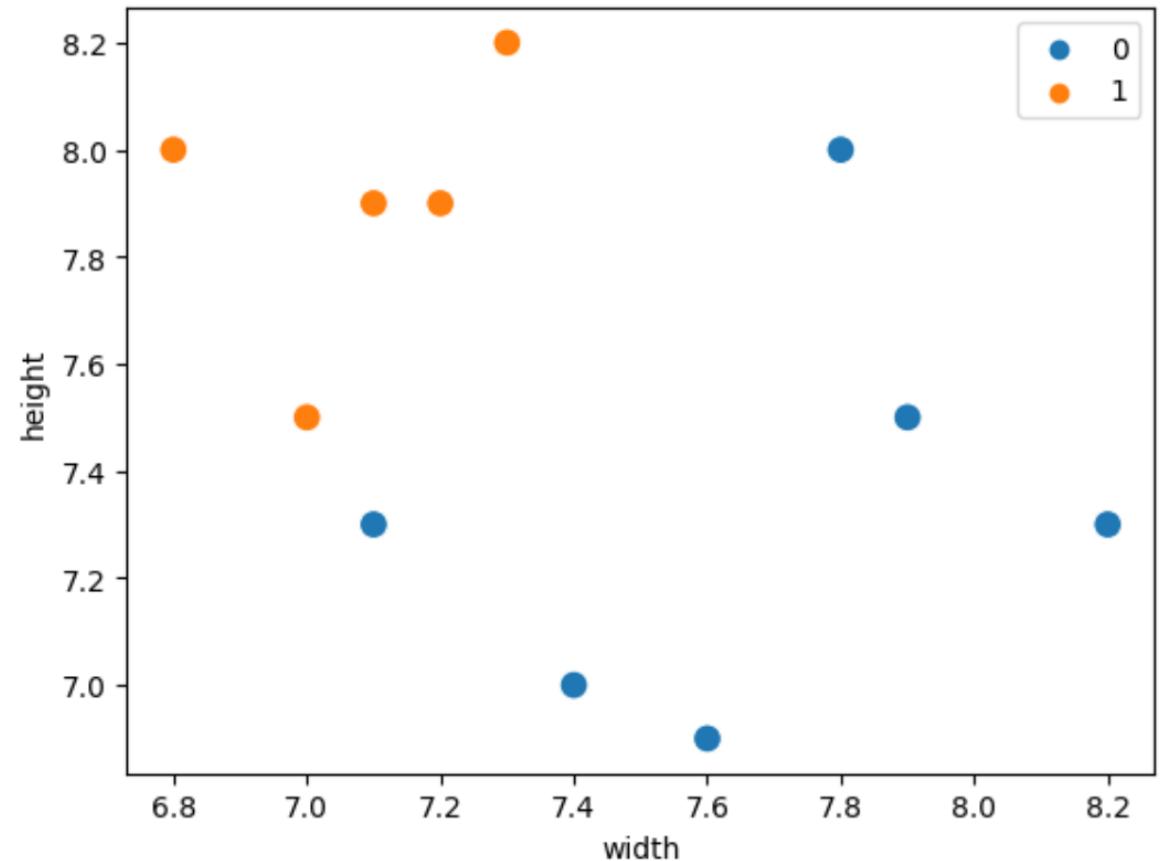
Martin Stypinski & Christian Fässler @ CH Open 2024

adnexo

Decision Tree Idee

\Vee/MG

- Fläche rekursiv aufteilen in «subsplits»
- Splitting abbrechen, sobald alle Elemente dieselbe Klasse haben «homogeneous splits»
- Ein «homogeneous split»: Hat nur Elemente von einer Klasse



\Vee/MG

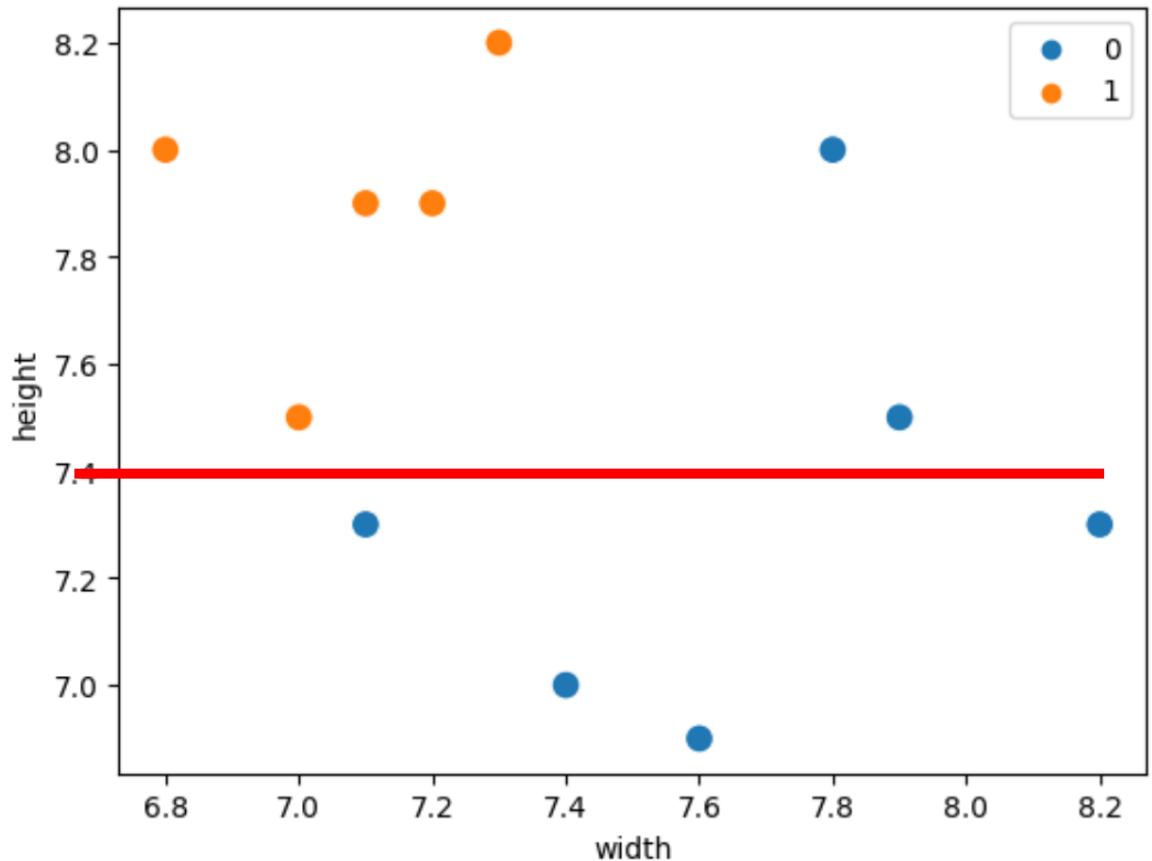
Martin Stypinski & Christian Fässler @ CH Open 2024

adnexo

Decision Tree

\Vee/MG

- Nur horizontale und vertikale Linien für Splits verwenden
- Was würden Sie als Programmierer hier machen mit if / else
→ Bruteforce



\Vee/MG

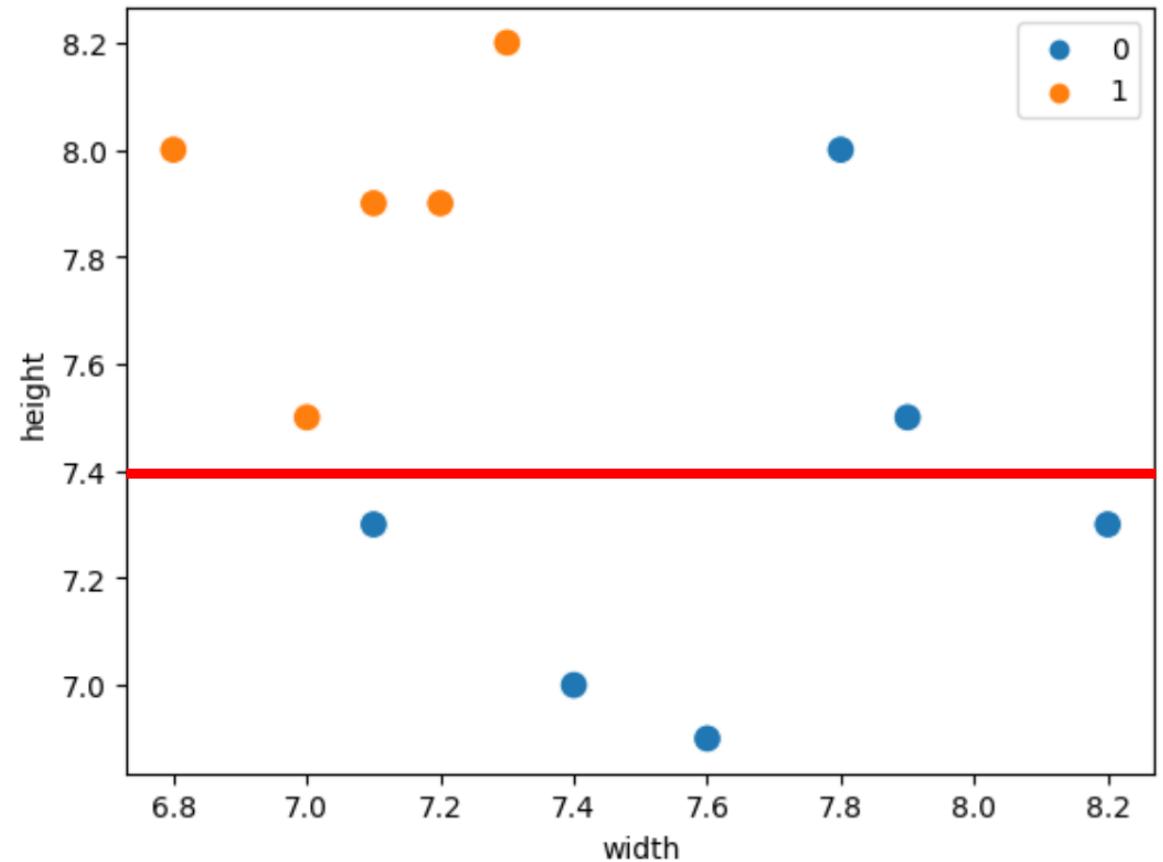
Martin Stypinski & Christian Fässler @ CH Open 2024

adnexo

Decision Tree

\Vee/MG

- Was ist ein guter Split?
- Wir brauchen ein Qualitätsmass, um unsere Splits zu beurteilen
 - Ideen?
 - GINI
 - Entropy



\Vee/MG

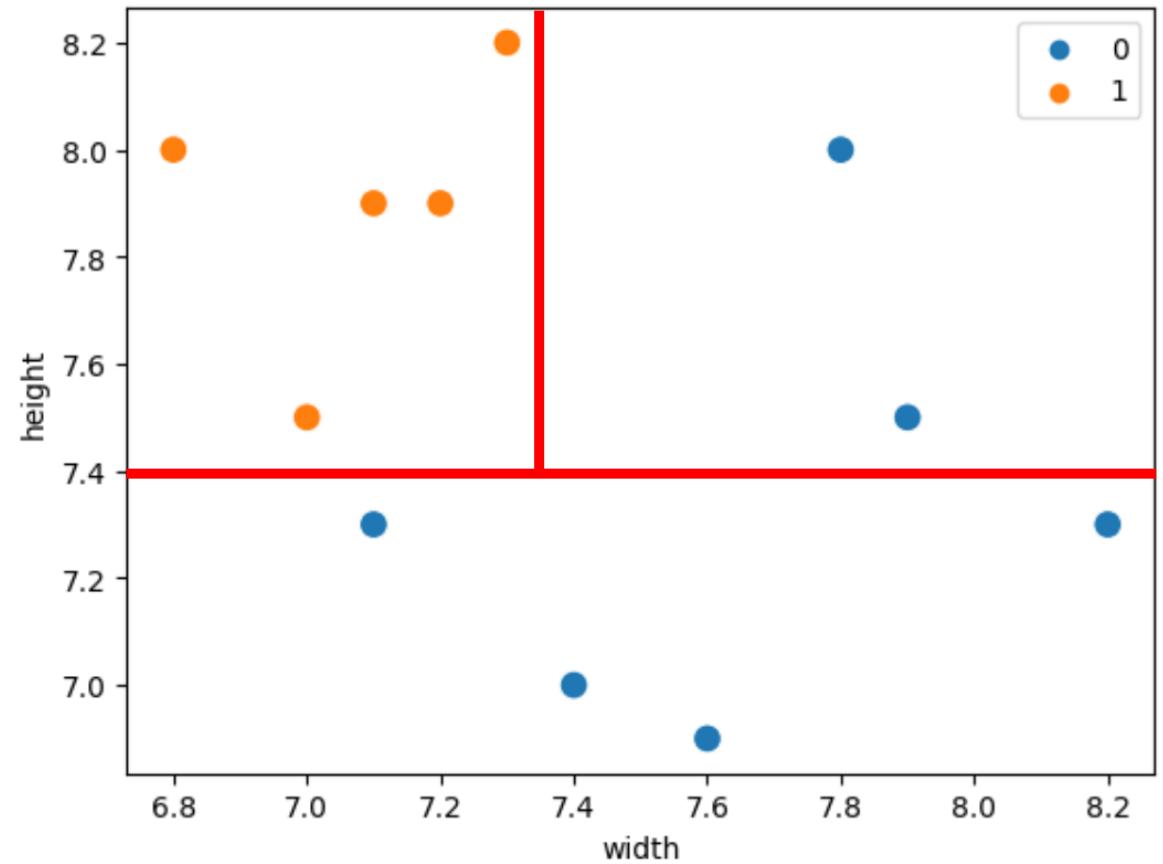
Martin Stypinski & Christian Fässler @ CH Open 2024

adnexo

Decision Tree

\Vee/MG

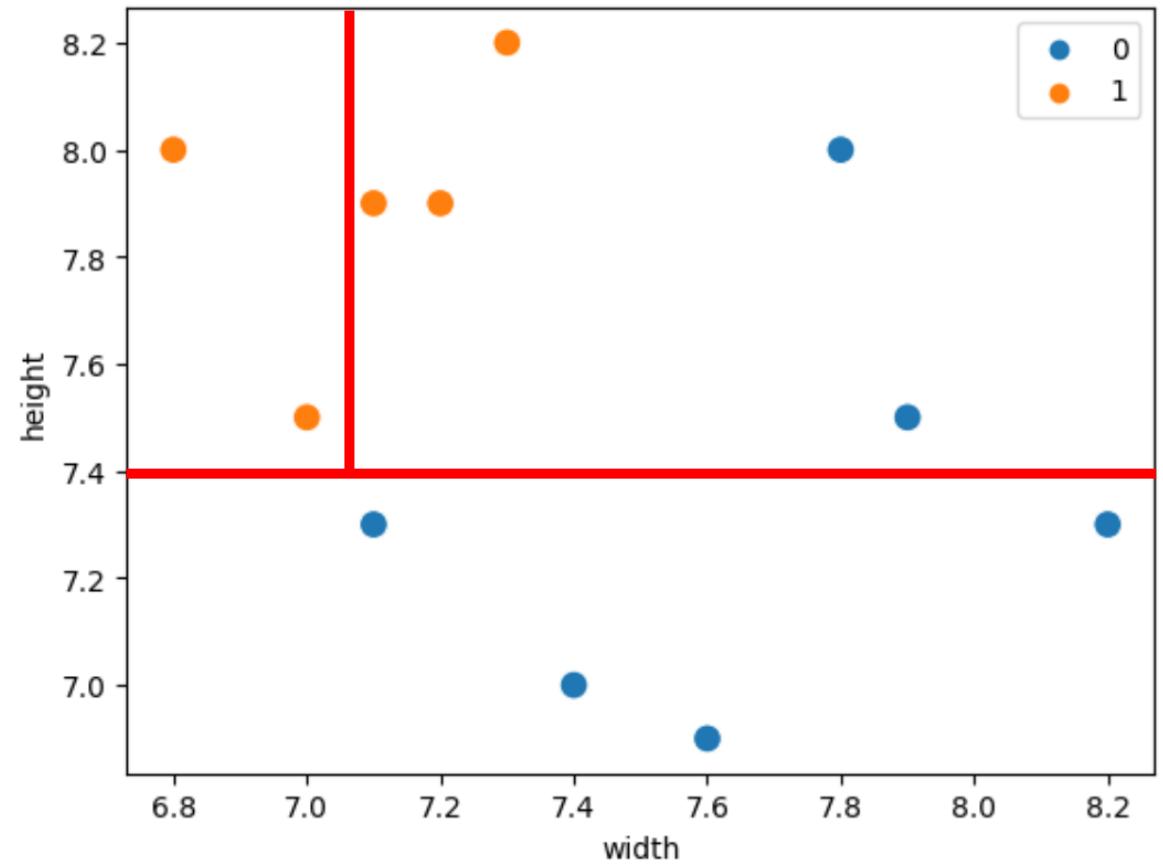
- Rekursives Splitten



Decision Tree

\Vee/MG

- Schlechter Split



\Vee/MG

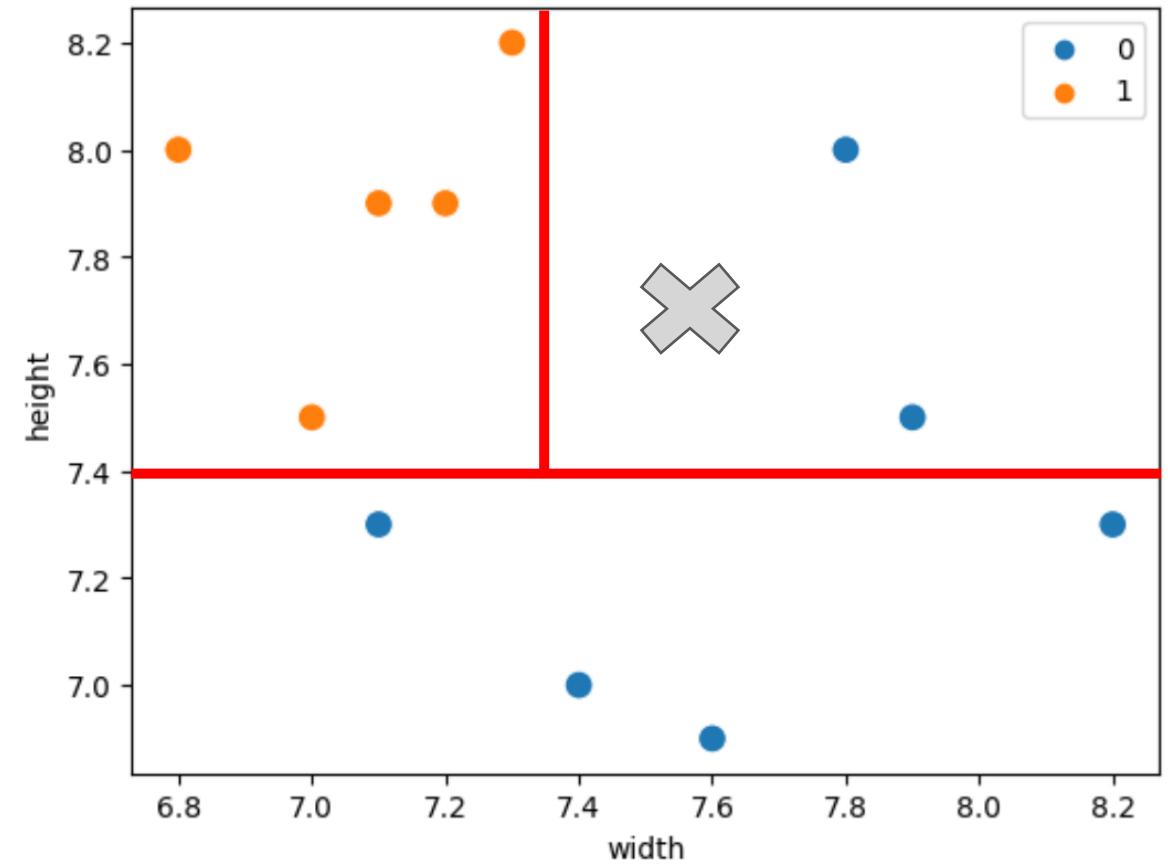
Martin Stypinski & Christian Fässler @ CH Open 2024

adnexo

Decision Tree

\vee/MG

- Wie können wir nun von neuen Datenpunkten die Klasse bestimmen?



\vee/MG

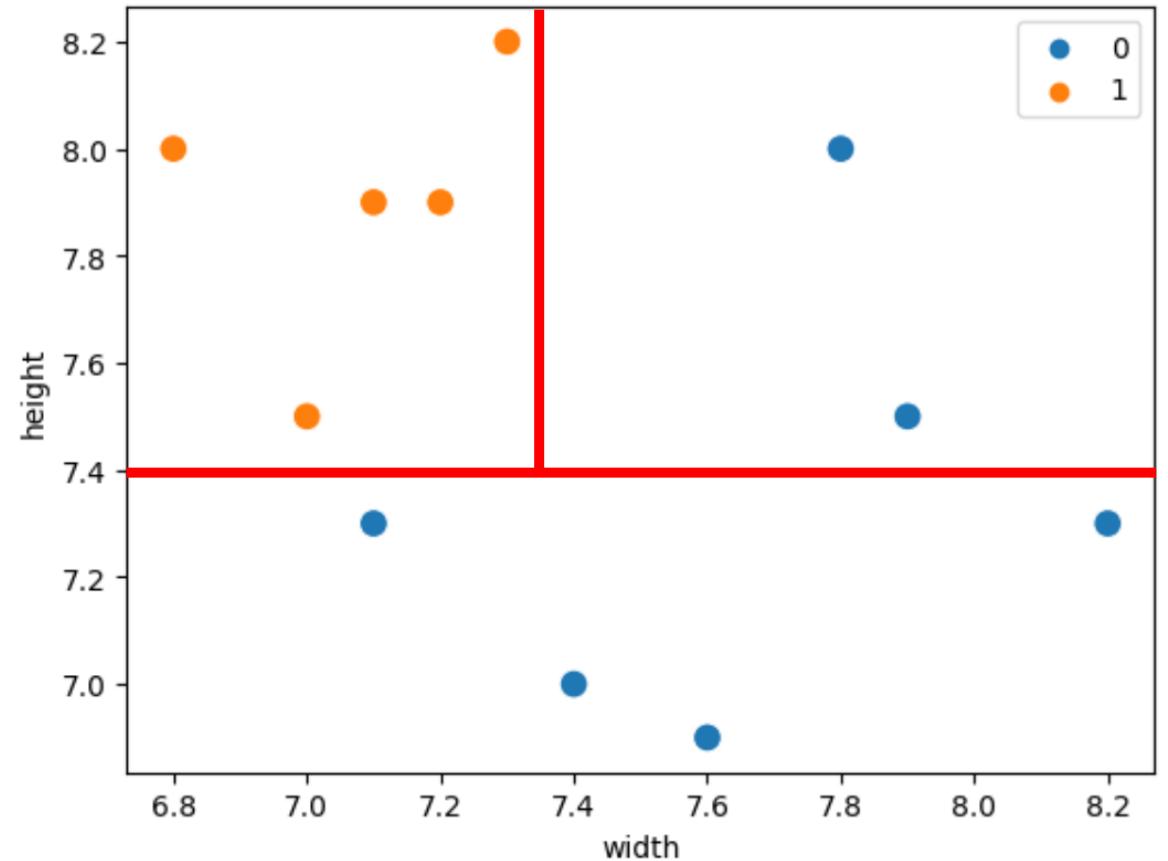
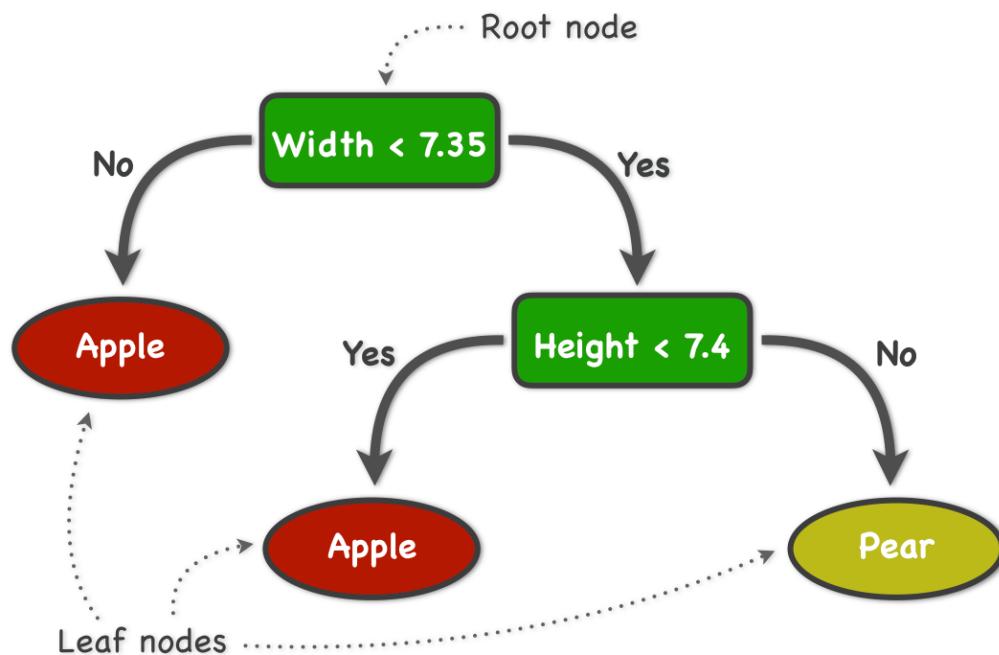
Martin Stypinski & Christian Fässler @ CH Open 2024

adnexo

It's a tree!

\Vee/MG

- Erlaubt «reasoning»



Was haben wir bis jetzt

- Algorithmus Decision Tree
- Training Data angewandt auf den Algorithmus ergibt uns ein trainiertes Modell
 - *Diese parametrisierte Instanz des Algorithmus hat Wissen aus den Daten*
- Ein Model das wir für Vorhersagen verwenden können
 - *Für neue (ungesehene Datenpunkte) können wir die Klasse vorhersagen*

Wie gut ist unser Modell?

- **Score** mit Test Daten
 - Testen mit Daten, für welche wir das «richtige» Label wissen
 - ABER das Modell nicht trainiert wurde
- **Accuracy** = % der richtigen Vorhersagen
- Nicht immer ganz so einfach
 - False positives vs False negatives → Business impact

Das Spitzli-Bier

Naturtrütes Bierprodukt aus dem
Berner Oberland, traditionell
von Hand gebraut.

Vor jedem Bierabend trinken Sie
ein helles, frisches Naturbier aus
dem Berner Oberland.

Spitzli geht immer.

Dinosaurier haben nie bier getrunken
und sind ausgestorben...
Mach nicht denselben fehler!

Les dinosaures ne buvaient pas de bières
et n'existent plus...
Ne faites pas la même erreur!

Hello World!

\Vee/MG

```
# Choose the algorithm
classifier = DecisionTreeClassifier()

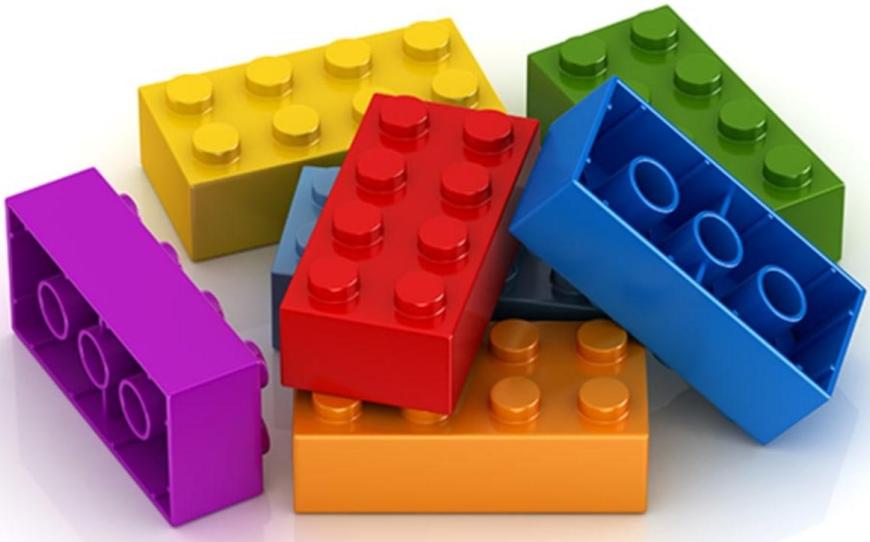
# Do the training
model = classifier.fit(training_input, y=training_correct_output)

# Use trained model to predict on unseen data
prediction = model.predict(new_unseen_data)
```

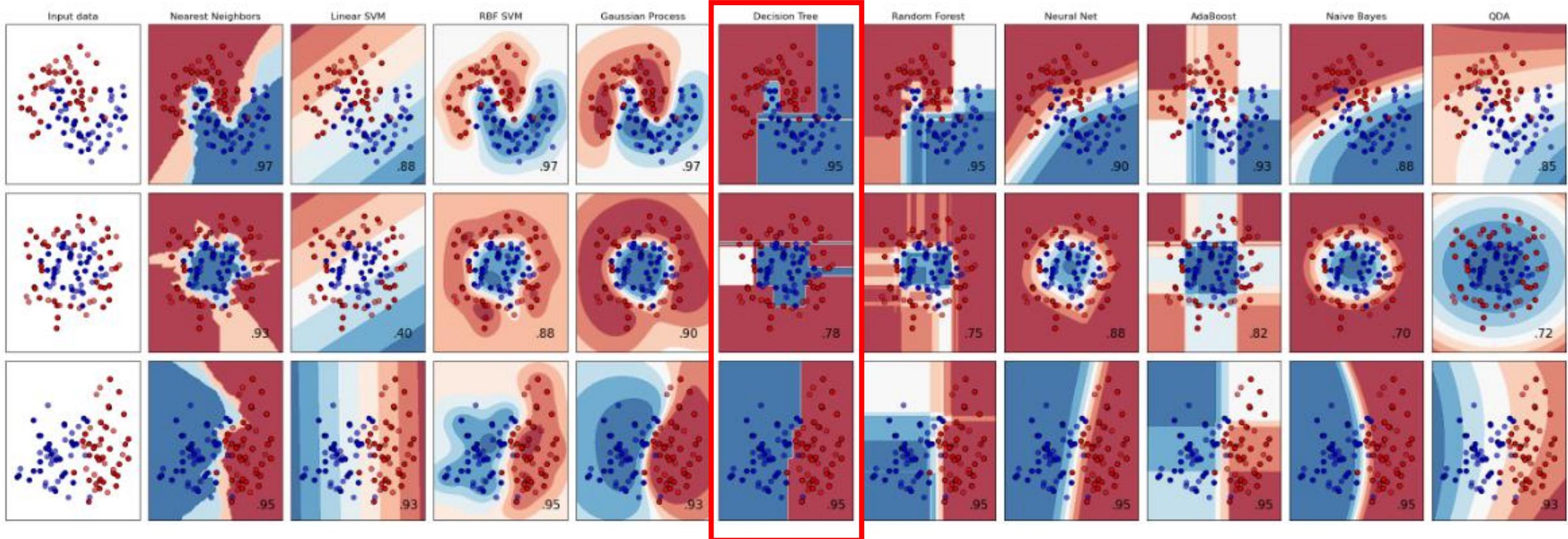
Demo

\Vee/MG

- Heutzutage wie LEGO bauen
 - Yeah... fast 😊
- Tooling
 - Python
 - Jupyter Notebooks
 - Scikit-Learn

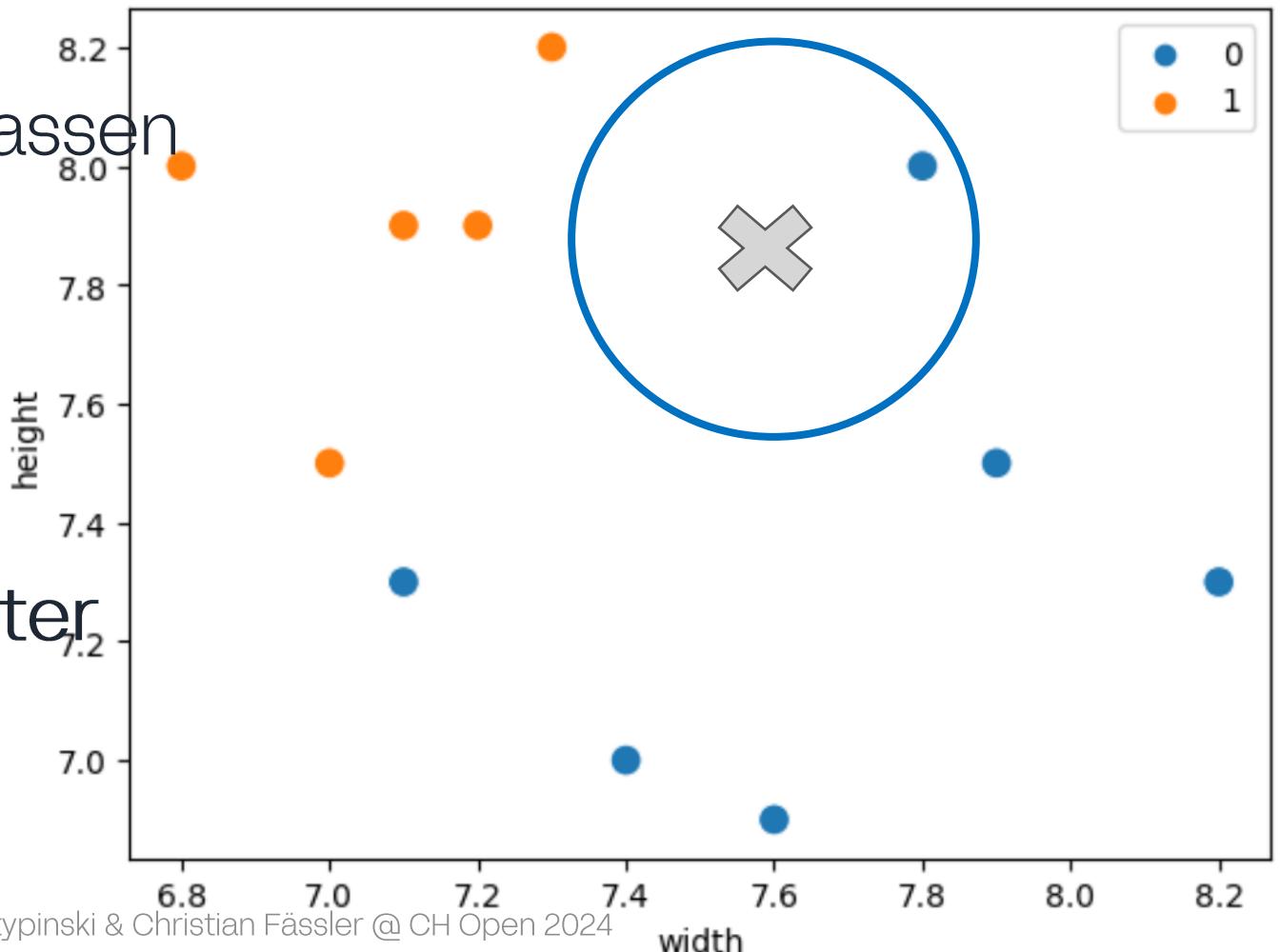


Classical ML Algorithms



- https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

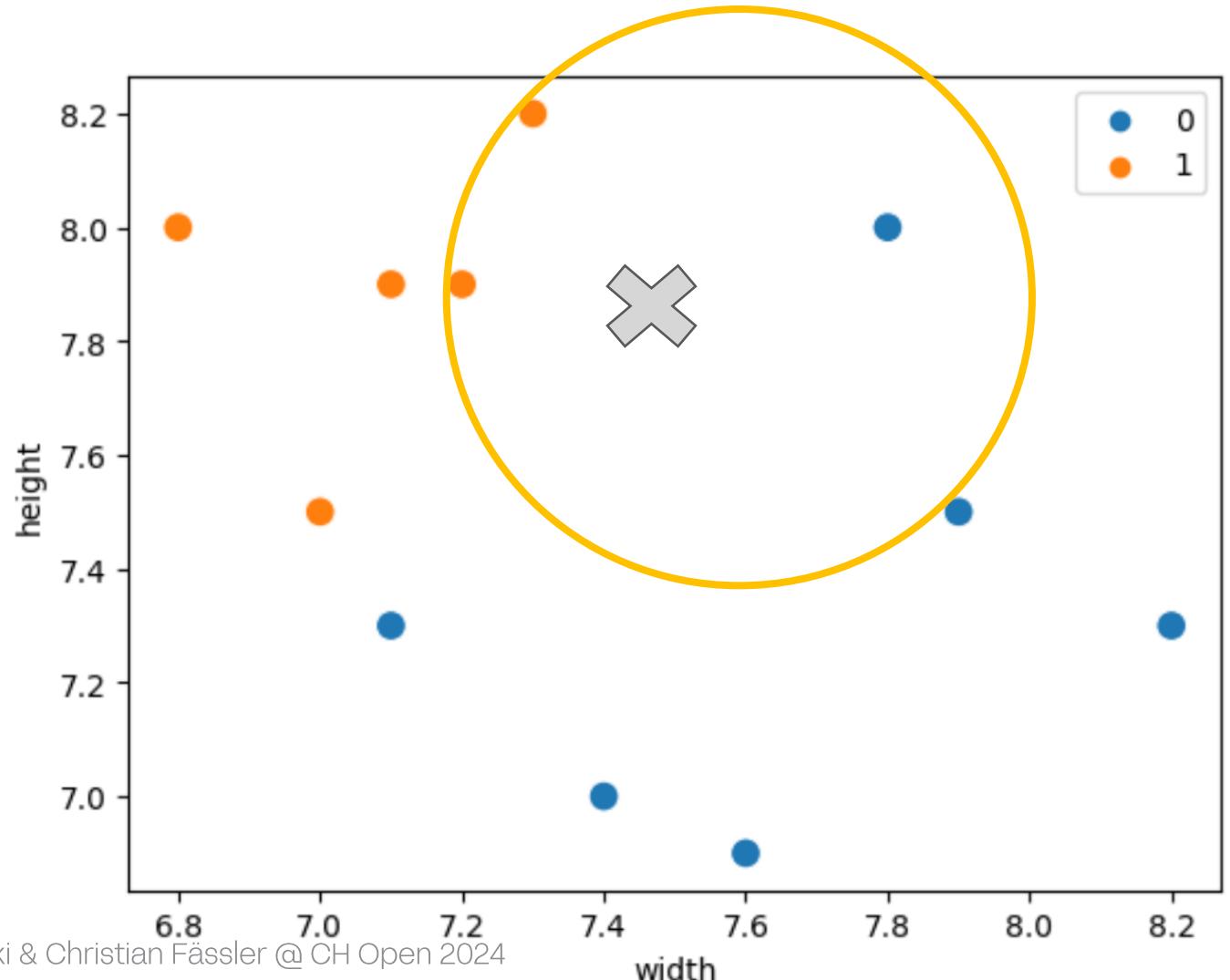
- K Nearest Neighbours
- Vergleichen mit den Klassen der N Nachbarn
- K=1 Apfel
- K = 2?
- K ist ein Hyperparameter



KNN

\Vee/MG

- K Nearest Neighbours
 - K=3 Birne
- K = Hyperparameter
- Wir möchten den Besten Wert finden



Hyperparameter K

- K = Hyperparameter
- Wir möchten den Besten Wert finden – aber wie?
- Suchen!
 - Wir machen eine Rastersuche und versuchen alle Werte durch
 - Basically we do a search and try out all values and check
 - Darum braucht man so viele NVIDIA Chips ☺
- Beispiel
 - 50'000 Katzenbilder, 50'000 Hundebilder, jedes 1MB
 - 100GB Daten laden k Mal trainieren (z.B k=3,5,7,9,11)

LAB 1

- Setup
 - Jupyter Notebooks <https://jupyter.org/install>
 - Use Google Colab <https://colab.research.google.com/>
- Notebook herunterladen und laufen lassen
- Einfache Aufgaben im Notebook lösen
- <https://github.com/cfaessler/mlcon>

MLOps

- Deployment
 - Wie uns was deployen wir?
 - Was sind die beweglichen Teile?
- Lifecycle
 - Wie und was verändert sich?
- Integration
 - Wie integriere ich meine Modelle / Daten in Software?
- Monitoring
 - Model Performance

Deployment

- Trainierte Modelle können serialisiert werden

```
from joblib import dump, load
# Serialize
dump(model, 'trained-model.joblib')
# Deserialize
loaded_model = load('trained-model.joblib')
output = loaded_model.predict([[50, 50, 100]])
```

Integration: API erstellen für Vorhersagen

- FastAPI mit gunicorn

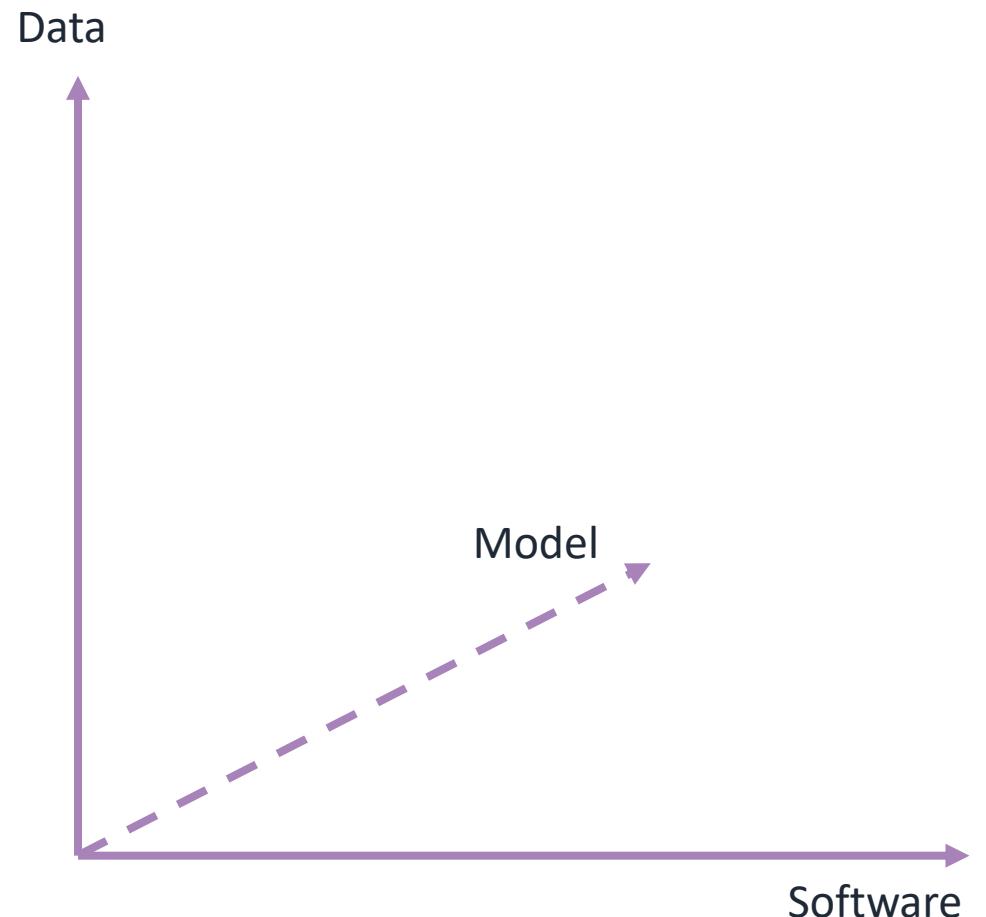
```
from fastapi import FastAPI
from joblib import load
from sklearn.tree import DecisionTreeClassifier

app = FastAPI()
loaded_model = load('./trained-apple-pears.joblib')

@app.get("/predict/")
def read_item(width: float, height: float):
    output = loaded_model.predict([[width, height]])
    if output[0] == 0:
        return "Apple"
    else:
        return "Pear"
```

Was sollten wir tracken

- Code Base der Software
 - Source Code
 - Dependencies
- Data
 - RAW Data
 - Preprocessed Data
- Models
 - Architecture / trained Parameters
 - Hyperparameters
 - Experiments



Wie geht deployment

Train

- Train the model
- Serialize the trained model

Deploy

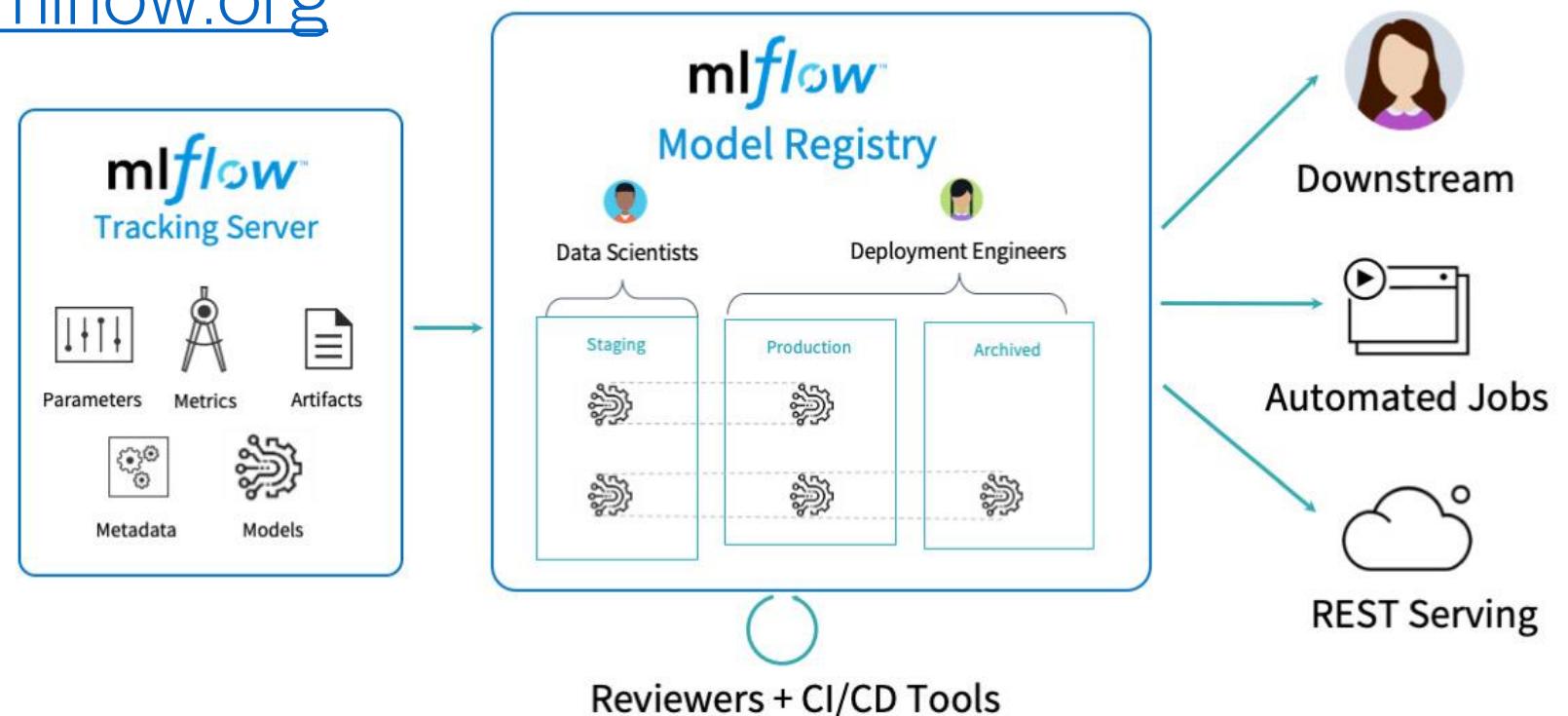
- Alles in ein Dockerfile «einbacken»
 - Python
 - Serialized model
 - requirements.txt
- Build docker container
- Create REST API mit Modell für «inference» (Vorhersagen)
- API deployen

Experiment Tracking

- Was haben wir versucht und hat zu welchem Ergebnis geführt
- Systematischer Prozess zur Findung des Modells und dessen Validierung
 - Training
 - Evaluation
 - Hyperparameters anpassen und Effekte beobachten
- Messen von «Qualität und Performance»
- Reproduzierbarkeit

Tools für Experiment-Tracking

- Weights and Biases <https://wandb.ai/site>
- MLFlow <https://mlflow.org>



Nächste Schritte

- 1. Daten handling mit Pandas
 - Daten filtern, Aggregieren, Laden und Speichern
 - https://pandas.pydata.org/docs/getting_started/index.html
- 2. Daten visualisieren mit Seaborn
 - <https://seaborn.pydata.org/tutorial.html>
- 3. Beschäftigung mit Algorithmen / NN
 - Sklearn <https://scikit-learn.org/>
 - Pytorch <https://pytorch.org/get-started/locally/>