TECHNICAL APPLICATIONS AND DATA MANAGEMENT. WS 2022.

VORLESUNG 6 UND 7.

25.10.2022

MÜNCHEN

STUDIENGANG SUSTAINABILITY MANAGEMENT & LEADERSHIP SOWIE MEDIEN & KOMMUNIKATIO

AGENDA

1. Überwachtes Lernen:
   - Workflow
   - „Simple“ Algorithmen
   - Kombinierte Algorithmen/ Ensemble Learning
   - Metriken für überwachtes Lernen
2. Case Study

ROADMAP VORLESUNG.

ROADMAP	WAS HABEN WIR VOR?
Vorlesung 1	Workflow Data Management, Datentypen und Datenqualität
Vorlesung 2	Einführung Data Science und Data Science Workflow, Grundlagen Data Management
Vorlesung 3 und Vorlesung 4	Deskriptive und explorative Datenanalyse und Vertiefung anhand Case Study
	Vertiefung Datenanalyse anhand Case Study
Vorlesung 5	Aufgabenstellung Data Science, Übersicht und Einführung Machine Learning, unüberwachtes Lernen
Vorlesung 6	Überwachtes Lernen
Vorlesung 7	Vertiefung überwachtes Lernen anhand Case Study
Vorlesung 8	Neuronale Netze und Convolutional Neural Networks (CNN)
Vorlesung 9	Vertiefung CNN anhand Case Study, Aufgabenstellung AI
Vorlesung 10	Schulterblick 1
Vorlesung 11	Übersicht Rekurrente Neuronale Netze
Vorlesung 12	Schulterblick 2
Vorlesung 13	Ausblick zukünftige AI-Themen, „Fragestunde“
Vorlesung 14	Präsentation Ergebnisse

Zusammenlegen!

1. ÜBERWACHTES LERNEN

SUPERVISED LEARNING: ÜBERSICHT.

Ziel: Wir suchen ein Modell um für beliebige **Eingaben $X_{i...n}$** eine **Zielgröße Y** möglichst genau zu bestimmen oder anzunähern.

Ermöglicht:

- **Regression**: Vorhersage eines kontinuierlichen Wertes Y (bspw. natürliche oder reelle Zahl wie Aktienkurs).
- **Klassifikation**: Vorhersage eines diskreten Wertes Y (bspw. wahr/ falsch, Tierart, Personenentdeckung, ….).

Was schauen wir uns?

- Gesamter Workflow: von Datenaufbereitung, Trainieren, Testen bis hin zur Anwendung.
- Ausgewählte „simple" Algorithmen für Regression und Klassifikation (weitere Algorithmen im Backup).
- Kombinierte Algorithmen (Ensemble Learning).
- Metriken: wie gut ist das Modell?.

SUPERVISED LEARNING: REGRESSION.

Ziel: Vorhersage eines Wertes aus einer großen Wertemenge (kontinuierliche Variable).

Ermöglicht: Prädiktion eines Wertes

Aber:

- Hoher Aufwand für das Bestimmen der Zielwerte (= Label). Dies ist beim unüberwachtem Lernen nicht vorhanden.
- Ist ungenauer als Klassifikation (wir versuchen einen genauen Wert aus einer großen Wertemenge vorherzusagen).
- Oft wird diese zusätzliche "Genauigkeit" nicht benötigt; z.B. reicht im Aktienhandel oft Prädiktion, ob Kursziel überschritten.

Anwendungsbeispiele:

- Aktienkursvorhersage: Prädiktion eines genauen Kurswertes.
- Biomonitoring: Prädiktion Cholesterin-/ Insulinspiegel.
- …

SUPERVISED LEARNING: KLASSIFIKATION

Ziel: Vorhersage eines Wertes aus einer kleineren, abzählbaren Menge (diskrete Variable).

Ermöglicht:

- Klassifizieren: Einordnen eines Wertes in eine endliche Gruppe bzw. Prädiktion eines endlichen Wertes.

Aber:

- Hoher Aufwand für das Bestimmen der Zielwerte (Labeln). Dies ist beim unüberwachtem Lernen nicht vorhanden.
- Genauigkeit hängt von der Größe der Gruppe ab → für sehr große Gruppen haben wir ja quasi eine Regression.

Anwendungsbeispiele:

- Online-Werbung: klickt der User die Werbung an oder nicht an?
- Bilderkennung: Erkennen eines bestimmten Tieres, Schriftzeichen, Bildinhalte bewerten, …
- Übersetzung: Deutsch zu Spanisch, …
- …

1 Han et al., 2011: Data Mining. Concepts and Techniques

2. GENERISCHER ABLAUF ÜBERWACHTES LERNEN

GENERISCHER ABLAUF SUPERVISED LEARNING. ÜBERSICHT.

1. Daten organisieren und hochladen.
2. Daten aufbereiten/ Data cleaning.

Das haben wir uns schon in vorigen Vorlesungen angesehen

3. Daten aufteilen in Trainings- und Testmenge (sowie ggf. Validierungsmenge).
4. Vorbereitung: Machine Learning-Algorithmus und Kostenfunktion wählen sowie Modellparameter initialisieren.
5. Training: schrittweise Optimierung Modellparameter bis das Modell möglichst gute Performance für die Trainingsmenge hat.
6. Modell(-güte) validieren per Testmenge, dessen Elemente Modell nicht kannte. Falls Modellgüte i.O., weiter zu Schritt 7.

   Sonst zu Schritt 5.
7. Deployment: Modell einsetzen im „Live"-Betrieb inkl. kontinuierliches Überprüfen der Modellgüte und ggf. Aktualisierung.

Ziel: Lernen eines möglichst genauen Modells H(Input) = Zielgröße[1]

1 Mathematisch gesagt: gegeben y=f(x) suchen wir eine Funktion h, die f(x) approximiert.

VERANSCHAULICHUNG DES GENERISCHEN ABLAUFS DES MACHINE LEARNINGS ANHAND DES IRIS-DATENSATZ.

Fisher, „ The use of multiple measurements in taxonomic problems” (1936).

- Datenset enthält 50 Samples à 4 Features zu jeder der 3 Irissorten:
  - Sepal length (Kelchblattlänge)
  - Sepal width (Kelchblattweite)
  - Petal length (Blütenblattlänge)
  - Petal Width (Blütenblattweite)
- Datensatz wird sehr häufig als Testdatensatz für Klassifikation verwendet.
- Aber: kleiner Datensatz

Zielgrösse: Klasse der Iris

Iris Setosa
(Klasse 0)

Iris versicolor
(Klasse 1)

Iris virginica
(Klasse 2)

Aufgabenstellung: Bestimmen Klasse der Iris anhand Messungen der 4 Features

Quelle Bilder: Wikipedia-Artikel, verfügbar unter Link

GENERISCHER ABLAUF SUPERVISED LEARNING. SCHRITT 1 UND SCHRITT 2: DATEN ORGANISIEREN UND AUFBEREITEN.

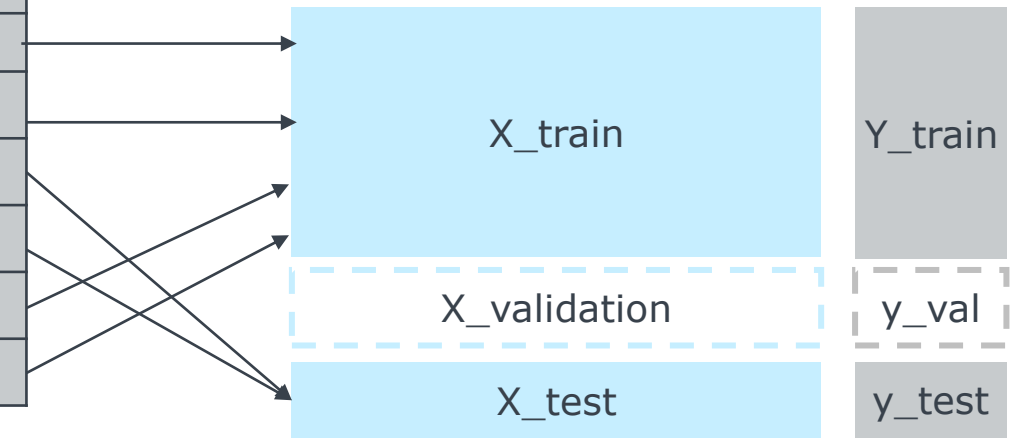
Hatten wir bereits in der 2. Vorlesung

GENERISCHER ABLAUF SUPERVISED LEARNING. SCHRITT 3: DATENAUFTEILUNG.

Matrix mit den Features/ Eigenschaften

Vektor mit den Zielwerten

Kelchblatt-länge	Kelchblatt-weite	Blütenblatt-länge	Blütenblatt-weite	Klasse
51	35	14	2	0
49	30	14	2	1
47	32	13	2	1
46	31	15	2	2
50	36	14	2	0
54	39	17	4	1




```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

Test_size: Verhältnis Trainings-/u Testmenge
Random_state = ermöglicht reproduzierbare zufällige Verteilung.

- Aufgabe **Trainingsset**: Lernen/ Optimieren der Parameter des Machine Learning Modells.
- Aufgabe **Testset**: Evaluation, wie gut die Parameter und somit das Modell sind.
- **Zufällige Aufteilung** in Trainings- und Testmenge stellt ähnliche statistische Eigenschaften sicher (sortierte Ausgangsdaten!)
- Oft wird **Validierungsset** zur Optimierung der Parameter des Trainingsprozesses (**Hyperparameter**) gebildet.
- Verhältnis Trainings- zu Testmenge: oft 70% zu 30%. Bei Einsatz Validierungsmenge: oft 60%-20%-20%.

GENERISCHER ABLAUF SUPERVISED LEARNING. SCHRITT 4: MODELL WÄHLEN UND INITIALISIEREN.

1. **Wahl Machine Learning-Modell** abhängig von:
   - Datentyp: Zahlen, Bilder, Sprache, ….
   - Verfügbaren Ressourcen: Rechen-Power, Zeit, Kosten, …
   - Datenmenge: viele oder wenig Daten vorhanden?
   - Qualität Daten: fehlende Werte/ Ausreißer stören manche Lernverfahren.
   - Anforderungen an Modellgüte.
   - Komplexität Algorithmus: komplexe Algorithmen neigen zu hoher Streuung, simple zu hoher Verzerrung. Bias-/ Variance Tradeoff

2. **Initialisierung Parameter**
   - Hyperparameter, d.h. Parameter für Trainingsprozess (bspw. Lernrate): erst Einsatz Standardwerte, dann Tuning (oft erfahrungsbasiert oder ausprobieren)
   - Modellparameter (Weights w_i und Bias b): Initialisierung mit zufälligen Werten (für schnelleres Lernen).

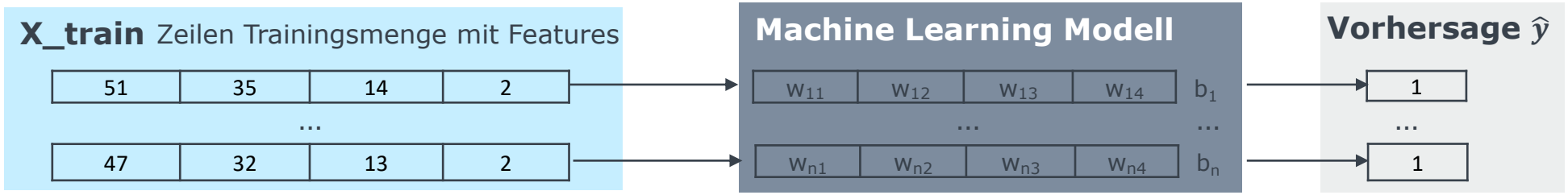
3. **Wahl Loss-Funktion:** Bewertungsfunktion für Genauigkeit des Machine Learning Modells (bspw. MSE[1], Cross-Entropy[2], …)

Auswahl ist erfahrungsbasierend. Die meisten Machine Learning Bibliotheken treffen aber standardmäßig gute Vorauswahlen.

Quelle: 1 Mean Squared Error
2 Rubinstein, R.Y. and Kroese, D.P., The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning, 2004.

GENERISCHER ABLAUF SUPERVISED LEARNING. SCHRITT 5: TRAINING

1. Feed Forward: Vorhersage Wert je **Sample** (individuelle Zeile Datensatz)



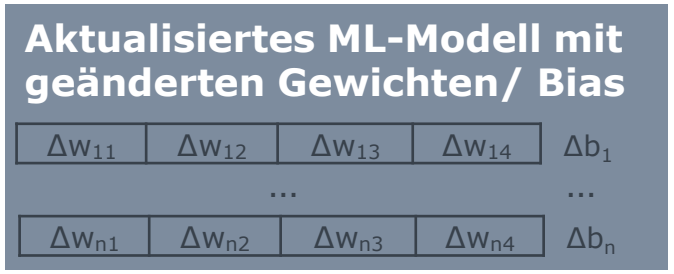
2. Berechnen Modellgüte und Fehler des Modells

- **Loss Function:** Berechnen Delta zwischen vorhergesagtem und realem Wert je Sample.
- **Cost Function:** Durchschnitt der Losses für jedes Sample.

3. Minimierung Fehler und Anpassen Modellparameter:

- Aufstellen Gleichung für Minimierung Cost Function (erste Ableitung gleich 0 setzen).
- Lösen Gleichung für die Modellparameter.
- Aktualisieren Modellparameter (Δw_{ij} und Δb_i), um so im nächsten Trainingslauf näher am realen Y-Wert zu landen.

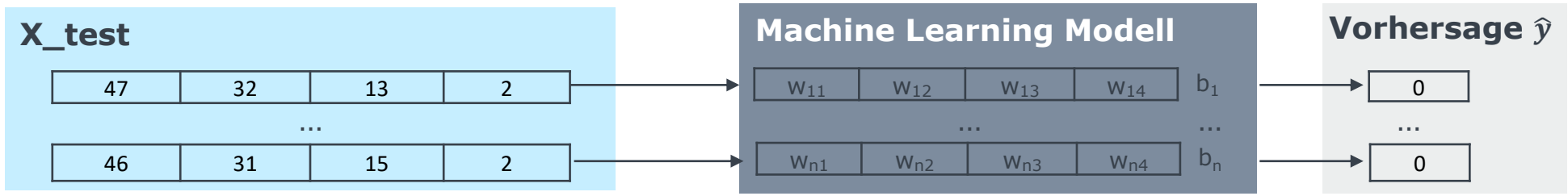
→ Einsatz iterativer Algorithmus bei hochdimensionalen Daten (bspw. Gradient Descent).



Durchführen Schritte 1-3 inkl. Update Modellparameter für gesamtes Trainingsset heißt **Epoch**, für Teile Trainingsset **Batch.** Anzahl Durchläufe je Batch/ Epoch erfahrungsabhängig inkl. Abbruchkriterien (ausreichende oder stagnierende Güte)

GENERISCHER ABLAUF SUPERVISED LEARNING. SCHRITT 6: MODELL TESTEN.

1. Feed Forward: Vorhersage Wert je **Sample**, d.h. eine individuelle Zeile des Datensatzes



2. Berechnen Modellgüte und Fehler des Modells

- **Validation Accuracy:** Berechnen Delta zwischen realem und vorhergesagtem Wert
- **Klassifikationsmetriken:** Confusion Matrix
- **Regressionsmetriken**: MSE, RSME, …

GENERISCHER ABLAUF SUPERVISED LEARNING. SCHRITT 6: MODELL TESTEN – WIE BEWERTET MAN DIE GÜTE EINES MODELLS?

Klassifizierungsmetriken

Confusion Matrix

		Vorhergesagt	
		Ja	**Nein**
Tatsächlich	**Ja**	True Positive	False Negative
	Nein	False Positive	True Negative

```
from sklearn.metrics import confusion_matrix
confusion_matrix(real_y, predicted_y)
```

Accuracy

$$\frac{Anzahl\ richtige\ Vorhersagen}{Anzahl\ aller\ Vorhersagen}$$

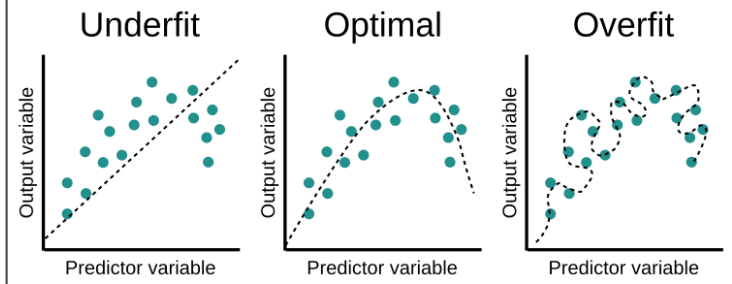
```
from sklearn.metrics import accuracy_score
accuracy_score(predicted_y, real_y)
```

Daraus abgeleitete Metriken:
- F1 Score
- Precision
- Recall
- ROC/ AUC

Regressionsmetriken

- **MAE (Mean absolute error)**
- **MSE**
- **RSME**
- **R-squared**
- Adjusted R squared
- Explained Variance

Over- und Underfitting



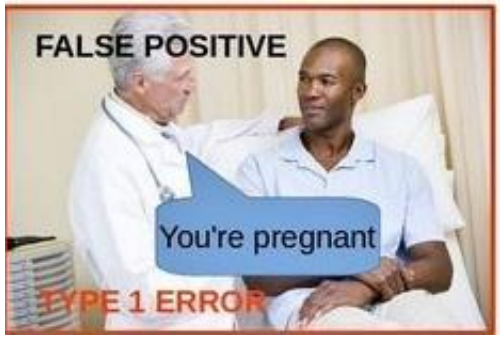
Quelle Bilder: Link

DETAILLIERUNG KLASSIFIZIERUNGSMETRIKEN: CONFUSION MATRIX.

		Vorhergesagt	
		Ja	**Nein**
Tatsächlich	**Ja**	True Positive (TP)	False Negative (FN)
	Nein	False Positive (FP)	True Negative (TN)

- **Recall** = wie viele Elemente wurden korrekt vorhergesagt $:= (\frac{\text{True positive}}{\text{True Positive+False negative}})$
- **Specifity** = wie viele negative Element korrekt als negativ vorhergesagt wurden $:= (\frac{\text{True negative}}{\text{True negative +False positive}})$
- **Precision** = wie viele der als "wahr" vorhergesagten Elementen waren wirklich wahr $:= (\frac{\text{True positive}}{\text{True Positive+False positive}})$
- **Accuracy** = die Anzahl korrekt vorhergesagter Elemente geteilt durch die Gesamtzahl $:= (\frac{\text{True positive+True negative}}{\text{Gesamtzahl}})$
- **F1-Score** = je höher der F1-Score, desto besser kann das Modell vorhersagen $:= (\frac{2 * \text{Recall} * \text{Precision}}{\text{Recall+Precision}})$

DETAILLIERUNG CONFUSION MATRIX AN BEISPIEL.

Tatsächlich \ Vorhersage	schwanger	nicht schwanger
schwanger		
nicht schwanger		

Welche False-Klasse ist schlimmer?

Quelle: Link

DETAILLIERUNG REGRESSIONSMETRIKEN.

- **MAE (Mean absolute error):** gemittelte Abweichung des prädizierten vom realen Wert. Gut für Feintuning.
  → je kleiner, desto besser das Modell.

$$\frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

- **MSE (**Mean Squared Error): gibt an, wie weit die Prognosewerte um den erwarteten/ realen Wert streut. Durch Quadrierung werden starke Abweichungen besonders "bestraft".
  → Je größer, desto schlechter das Modell.

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2$$

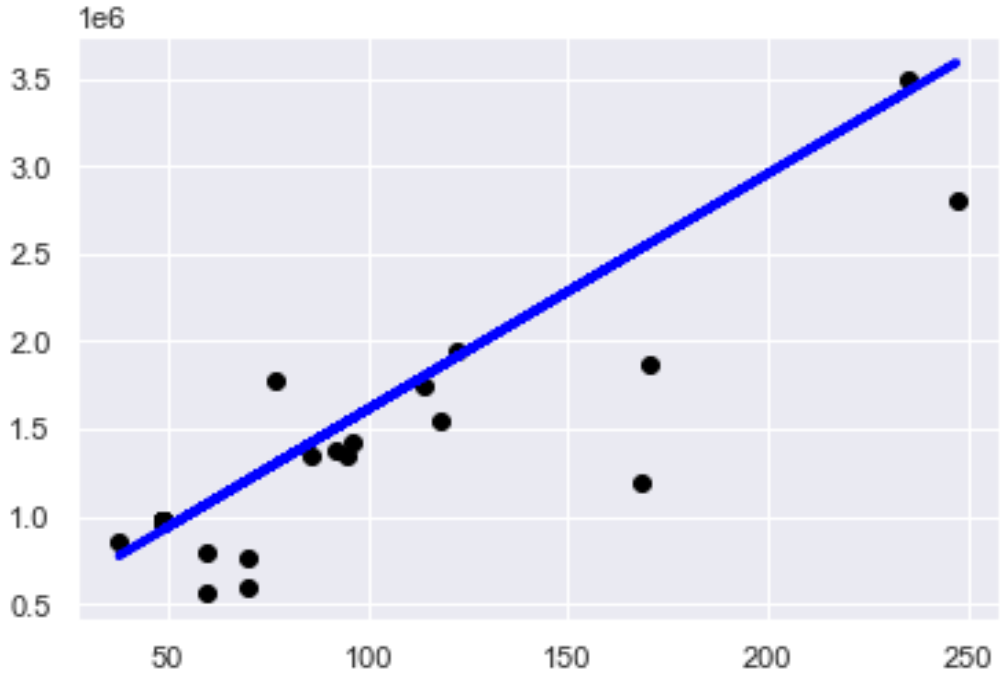
- **RSME** (Rooted Mean Squared Error): Wurzel aus Mean Squared Error.
  →Je größer, desto schlechter Modell.

$$\sqrt{MSE}$$

- **R-squared =** gibt an, wie gut das Modell die Daten erklären kann; d.h. wie gut das Modell die Y-Werte abdeckt. Wert ist zwischen 0 und 1
  → Wert von 1 bedeutet perfekte Abdeckung, jeder reale Wert wird durch prädizierten Wert abgedeckt.

$$\frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y}_i)^2}{\sum_{i=1}^{n}(yi - \overline{y_i})^2}$$

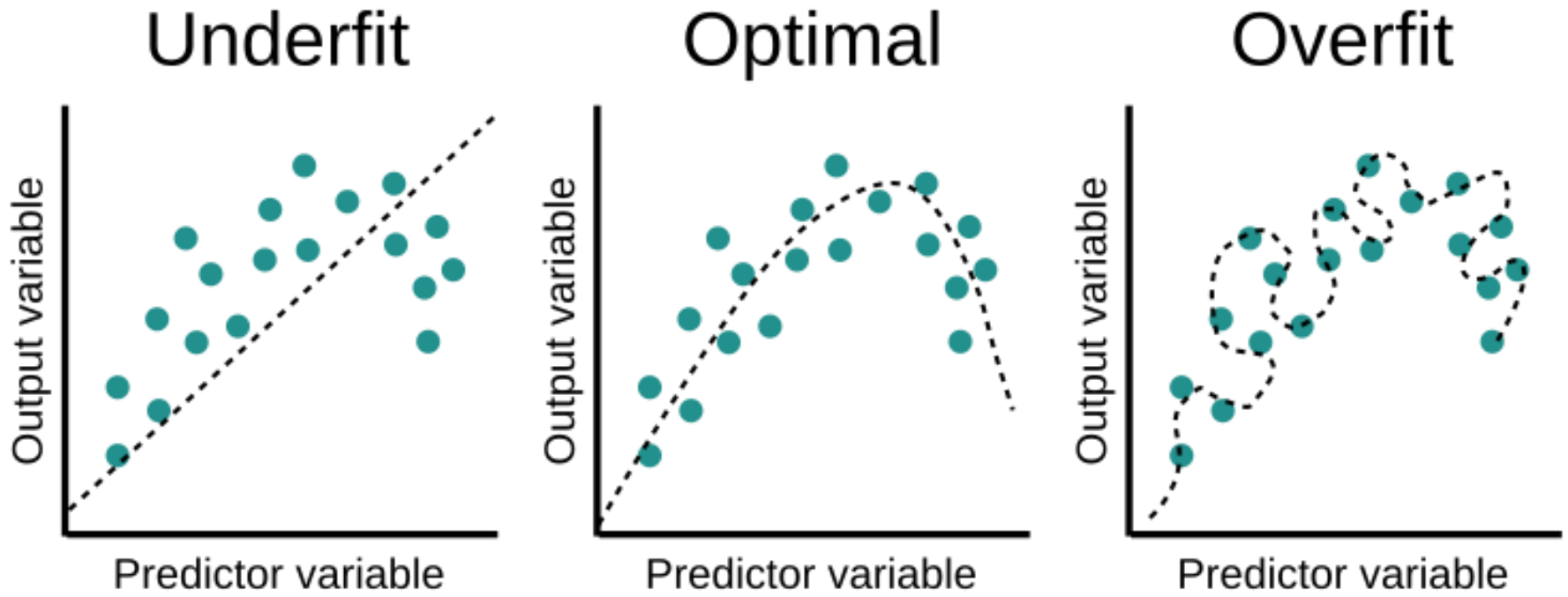
DETAILLIERUNG METRIKEN: BEISPIEL REGRESSIONSMETRIKEN.



- Wenig oder viele Ausreißer?
- MSE: hoch oder niedrig?
- RMSE: hoch oder niedrig?
- R^2: hoch oder niedrig?

Oder ganz einfach gefragt: wie gut ist das Modell?

OVER- UND UNDERFITTING MODEL.



Underfitting
Hoher bias/ Verzerrung: wichtige Abhängigkeiten werden nicht erkannt, da Modell zu simpel.
→ Erkennbar an: schlechter Accuracy Trainings- und Testdaten.

Overfitting
Hohe Varianz: nicht relevante Eigenheiten Trainingsdaten werden gelernt („Auswendiglernen"), Modell zu komplex
→ Erkennbar an: Trainings-Accuracy steigt und ist höher als die Test-Accuracy, Test Accuracy stagniert.

Eselsbrücke anhand Prüfungsvorbereitung:
- Lernten Sie zu wenig, dann haben Sie sowohl bei Übungsprüfungen als auch bei realer Prüfung viele Fehler (Sie wissen nicht, was wichtig ist).
- Lernten Sie sehr viel anhand von Übungsprüfungen, dann hilft das nur, falls die der realen Prüfung ähneln (Sie lernten nicht, was dran kam).

Quelle Bilder: Link

WORAN ERKENNE ICH, WAS DENN OPTIMAL IST?

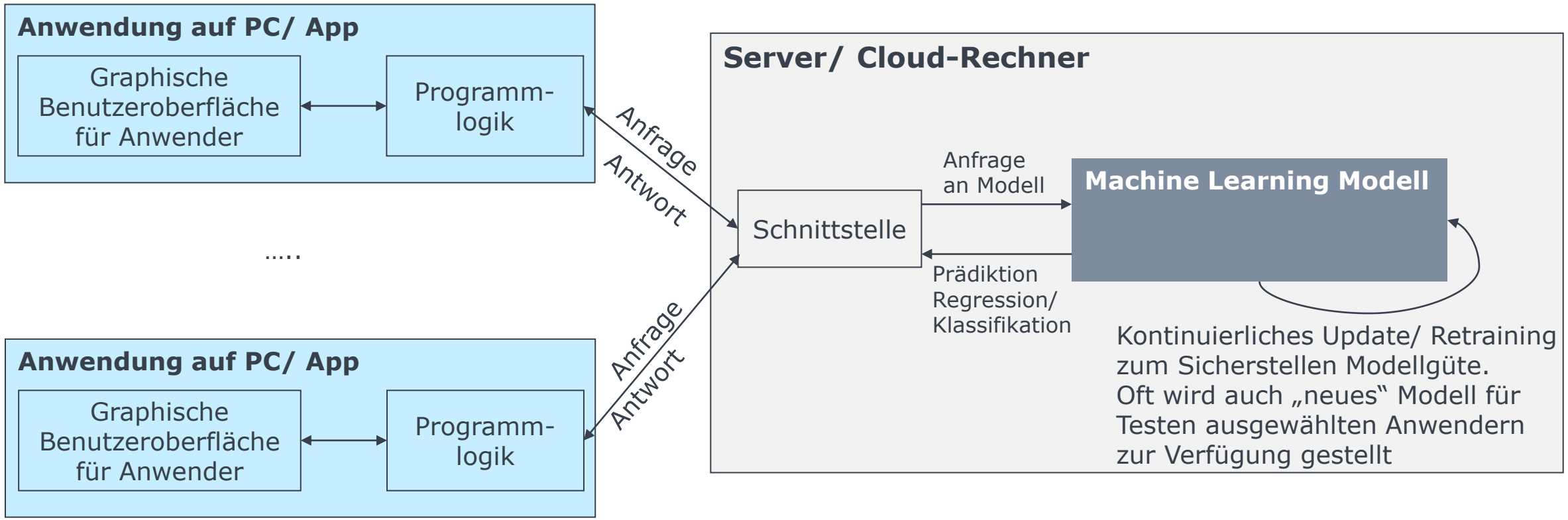
Bau einer Baseline und Vergleich mit Model Performance:

- Wie genau ist ein ganz simples Verfahren? (Bspw. Raten bei 2 Zielklassen ist 50% Chance).
- Wie genau ist ein ganz simples ML-Verfahren (bspw. lineare Regression?)
- Wie genau ist ein Mensch (Human baseline, bspw. Arzt bei Diagnose)

„Optimales" Model: Modell und Hyperparameter verbessern, bis bessere Performance als Mensch und kein Overfitting!

1 Ameisen: Building Machine Learning Powered Applications: Going from Idea to Product, O'Reilly-Verlag 2018.

GENERISCHER ABLAUF SUPERVISED LEARNING. SCHRITT 7: MODELL EINSETZEN (DEPLOYMENT).



3. DETAILLIERUNG AUSGEWÄHLTER ALGORITHMEN

DETAILLIERUNG ALGORITHMEN SUPERVISED LEARNING.

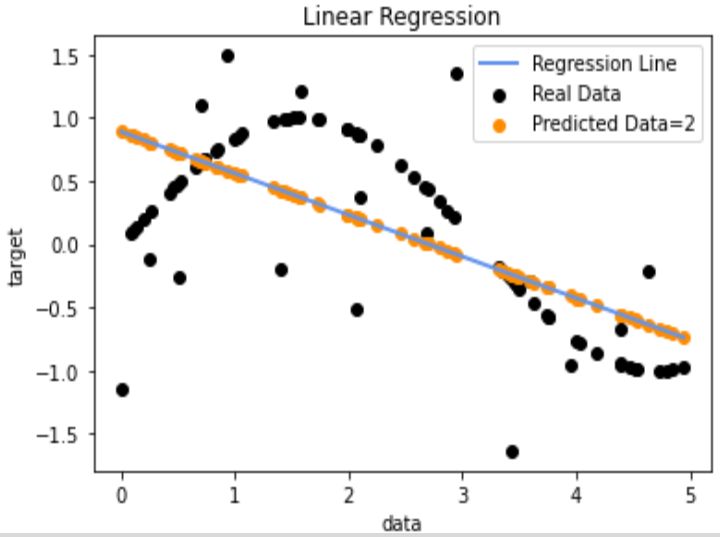
Auf den folgenden Folien werden geläufige Algorithmen an zwei Beispielen veranschaulicht.

- **Regression**: Vorhersage der Werte einer Sinuskurve. Dabei ist Sinuskurve an bestimmten Stellen manuell geändert, um die Vorhersage zu erschweren.
- **Klassifikation**: anhand Iris-Datensatz. Da dieser Datensatz 4 Features/ Dimensionen hat, läßt er sich schwer graphisch darstellen. Zur Veranschaulichung wurde der Datensatz deshalb auf die 2 Sepal-Werte reduziert.

Eine Übersicht bekannter Algorithmen finden Sie bspw. hier: Link

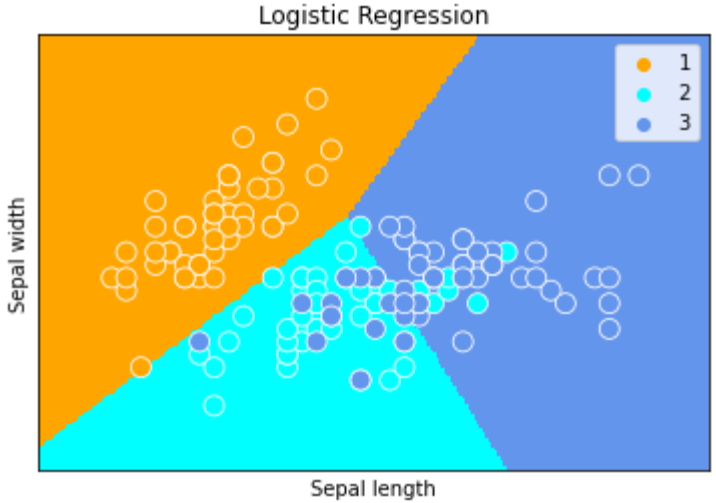
SUPERVISED LEARNING: LINEARE/ LOGISTISCHE REGRESSION.

Regression kontinuierliche Variable




```
from sklearn.linear_model import LinearRegression
linRegressor = LinearRegression() # Modell initialisieren
linRegressor.fit(X_train,y_train) # Modell trainieren
Y_pred_LR = linRegressor.predict(X_test) # vorhersagen
```

Klassifizierung diskrete Variable




```
from sklearn.linear_model import LogisticRegression
logRegressor = LogisticRegression() # Modell init.
logRegressor.fit(X_train,y_train) # Modell trainieren
Y_pred_LR = logRegressor.predict(X_test) # vorhersagen
```

Vorteile:

- Leicht verständlicher Algorithmus und Ergebnisse.
- Leicht implementierbar.

Nachteile:

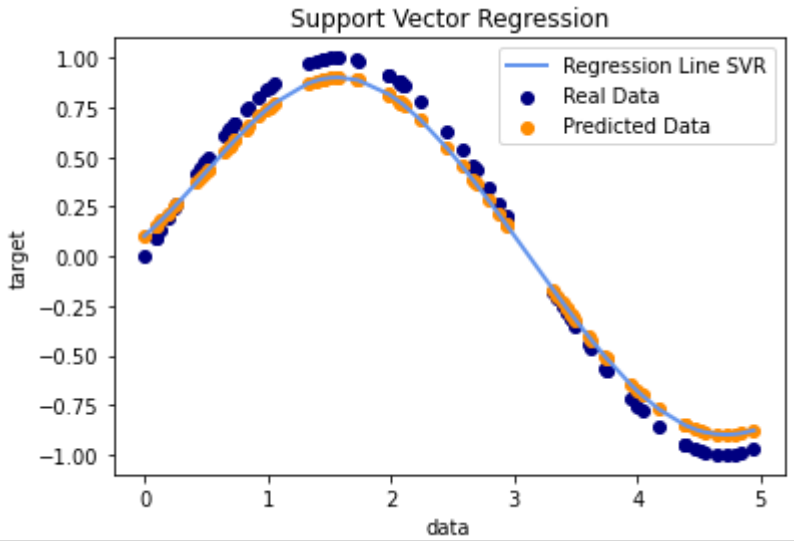
- Probleme mit Ausreißern.
- Benötigt Zusammenhang/ Abhängigkeit zwischen Variablen.
- Probleme mit komplexen/ hochdimensionalen Daten.

Lineare Regression versucht, die Zielvariable durch ein lineares Modell (Gerade) zu beschreiben.
Die Gerade/n ist/sind dabei so konstruiert, daß sie möglichst geringen Abstand von den Punkten des Datensatzes hat.

Quellen: SK-Learn Dokumentation: Link, Link

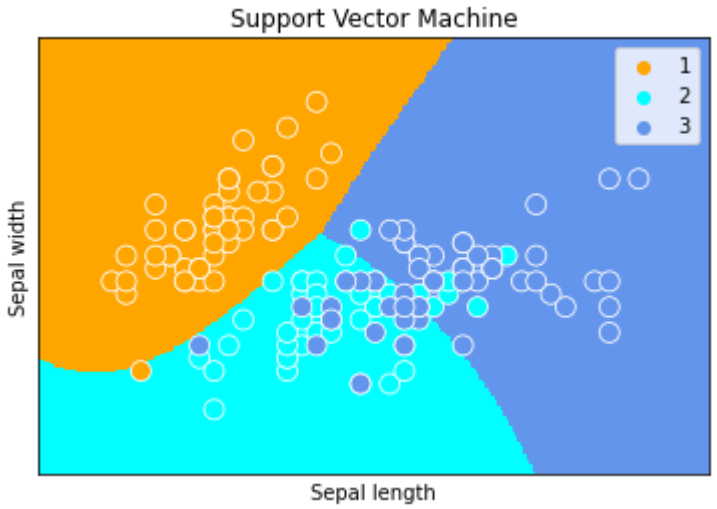
SUPERVISED LEARNING: SUPPORT VECTOR MACHINE.

Regression kontinuierliche Variable




```
from sklearn import svm
SVR_basic = svm.SVR() # Modell initialisieren
SVR_basic.fit(X_train,y_train) # Modell trainieren
Y_pred_SVR = SVR_basic.predict(X_test) # vorhersagen
```

Klassifizierung diskrete Variable




```
from sklearn import svm
SVC_basic = svm.SVC() # Modell initialisieren
SVC_basic.fit(X_train,y_train) # Modell trainieren
Y_pred_SVC = SVC_basic.predict(X_test) # vorhersagen
```

Vorteile:

- Robust auch bei hochdimensionalen Features.
- Einsetzbar für Klassifikation und Regression.
- Einsetzbar auch für Klassifikation Bilder.

Nachteile:

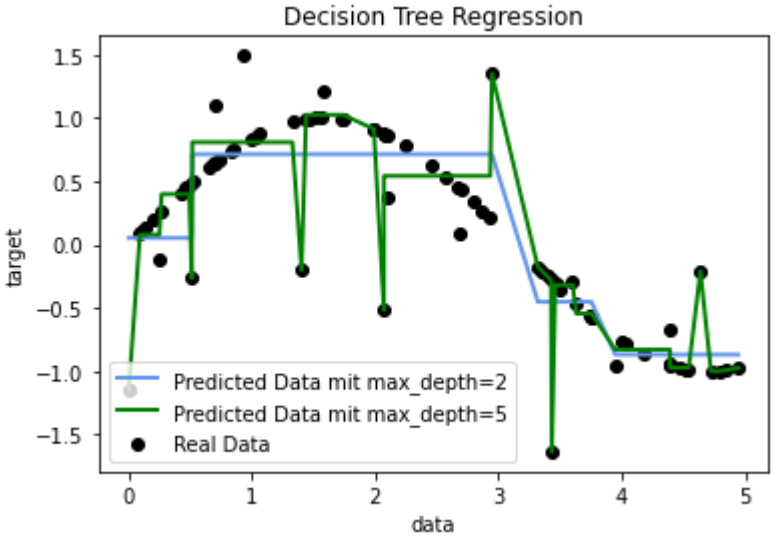
- Viele Hyperparameter, deshalb Tuning notwendig.
- Längere Trainingszeit bei großen Datenmengen.
- Erklärbarkeit Ergebnisse schwierig (Hyperebene n-dimensionaler Raum).

Klassifikation: Algorithmus bestimmt „Hyperebene“[1], die Datensatz am besten in die gewünschte Anzahl Klassen einteilt. Dabei wird die Hyperebene so gelegt, daß die Grenze zwischen den Klassen möglichst breit ist.
Regression: Algorithmus bestimmt „Hyperebene“ und nimmt Punkt, der möglichst nah an der Hyperebene liegt

Quelle: Vapnik und Chervonenkis, Theory of Pattern Recognition, 1974.
1 Hyperebene: Ebene mit um 1 geringeren Dimension als Datensatz (Bspw. Gerade für 2 Features oder flaches Rechteck im Dreidimensionalen).

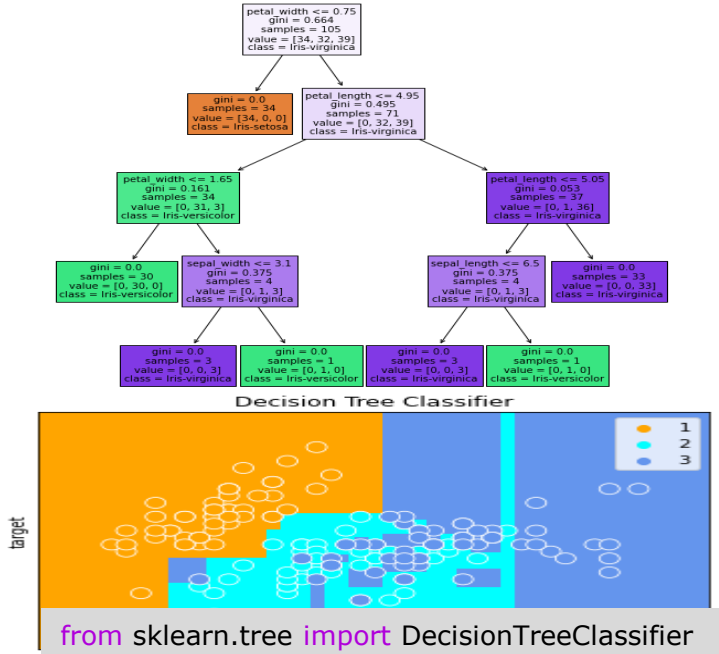
SUPERVISED LEARNING: DECISION TREE.

Regression kontinuierliche Variable




```
from sklearn.tree import DecisionTreeRegressor
TreeRegr = DecisionTreeRegressor(max_depth=5)
TreeRegr.fit(X_train,y_train) # Modell trainieren
Y_pred_Tree = TreeRegr.predict(X_test)
```

Klassifizierung diskrete Variable




```
from sklearn.tree import DecisionTreeClassifier
decisionTree = DecisionTreeClassifier()
decisionTree.fit(X_train,y_train) # Modell trainieren
 Y_pred_Tree = decisionTree.predict(X_test)
```

Vorteile:

- Einfach verständliche Ergebnisse.
- Einsetzbar für Regression und Klassifikation.
- Weniger Aufwand Datenaufbereitung (keine Skalierung/ Normierung).
- Robust bei fehlenden Daten.
- Robuster auch bei vieldimensionalen Features.

Nachteile:

- Nicht stabil gegenüber Updates (dann muß erneut trainiert werden).
- Schlechte Übertragbarkeit auf ähnliche Daten.
  Höherer Zeit-/ Ressourcenbedarf für Training.

Klassifikation: wiederholtes Aufteilen Datenset in zwei Untermengen anhand des Features mit höchstem Informationsgehalt.
Regression: erfolgt durch Bestimmen des Durchschnitts über die y-Werte des finalen Endknoten.

Quellen: SK-Learn Dokumentation: Link
Breiman, Friedman, Olshen, Stone: Classification and regression trees, 1984.

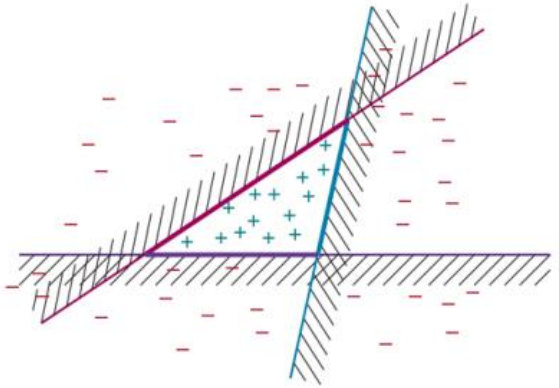
4. ENSEMBLE LEARNING

SUPERVISED LEARNING: ENSEMBLE LEARNING

Ziel: Verbesserte Performance durch die Kombination einzelner Lernverfahren (und Ausgleichen deren Schwächen).

Ermöglicht:

- Höherere Genauigkeit (Reduktion Streuung und Verzerrung).
- Mehr Anwendungsfelder können abgedeckt werden.



Aber: Höherer Ressourcenbedarf für Training.

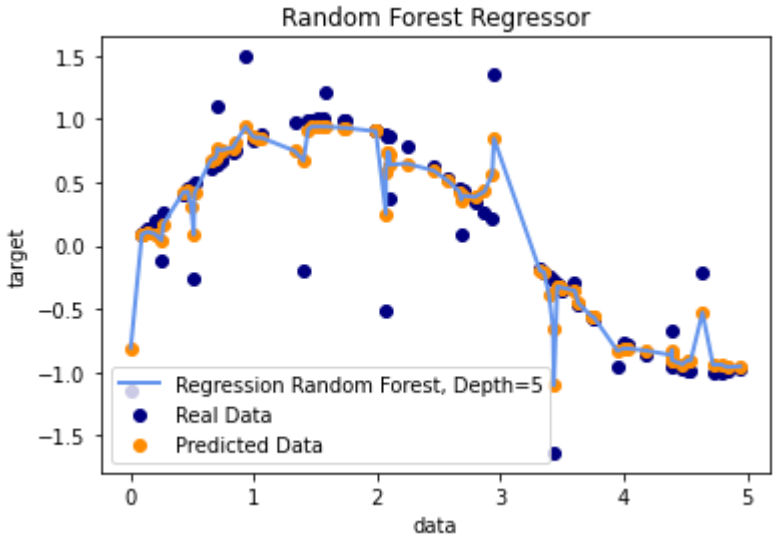
Häufigst eingesetzte Verfahren:

- Bagging: Kombination verschiedener Lernverfahren und **gleiche** Gewichtung (bspw. Durchschnitt) → Random Forest.
- Boosting: Kombination "schwacher" Lernverfahren per **performance-abhängiger Gewichtung** → XGBoost.

Quelle: Russell, Norvig: Artificial Intelligence: a modern approach, Kapitel 19.8

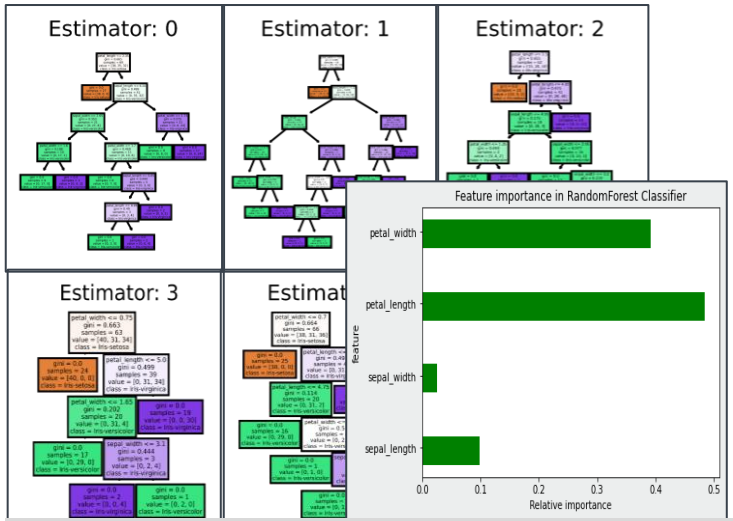
SUPERVISED LEARNING: RANDOM FOREST.

Regression kontinuierliche Variable




```
from sklearn.ensemble import RandomForestRegressor
RFRegressor = RandomForestRegressor ()
RFRegressor.fit(X_train,y_train) # Modell trainieren
Y_pred_RFR = RFRegressor.predict(X_test)
```

Für diskrete, mehrere Klassen




```
from sklearn.ensemble import RandomForestClassifier
RandomForest_clf = RandomForestClassifier()
RandomForest.fit(X_train,y_train) # Modell trainieren
Y_pred_RFC = RandomForest.predict(X_test)
```

Vorteile:

- Parallelisierbarkeit Training.
- Feature Importance: Algorithmus liefert Aussage, was die wichtigsten Features für die Vorhersage sind.
- Einsetzbar für hochdimensionale Daten.
- Höhere Genauigkeit als Entscheidungs-bäume (Reduziert Overfitting durch Gewichtung mehrerer Entscheidungsb.).

Nachteile:

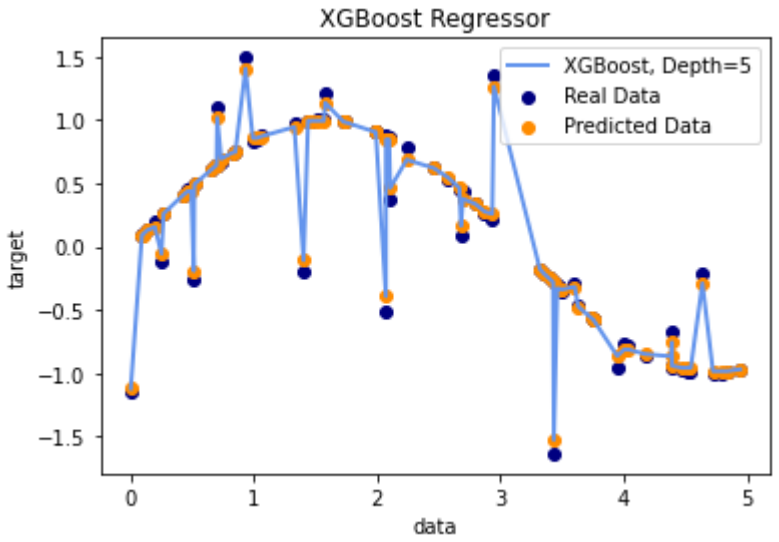
- Komplexer als Entscheidungsbäume.
- Nicht mehr so leicht interpretierbar.
- Schlechtere Performance bei Regression.
- Schlechtere Laufzeit-Performance.

Einsatz verschiedener Entscheidungsbäume und danach Kombination der Entscheidungsbäum.
Dabei wird jeder Entscheidungsbaum gleich gewichtet, d. h. hat gleichen Einfluß auf das Ergebnis.

Quellen: Ho: "Random Decision Forests", 1995.
Breiman, L: "Random Forests". Machine Learning. 45 (1): 5–32, 2001.
SK-Learn Dokumentation: Link

SUPERVISED LEARNING: XGBOOST.

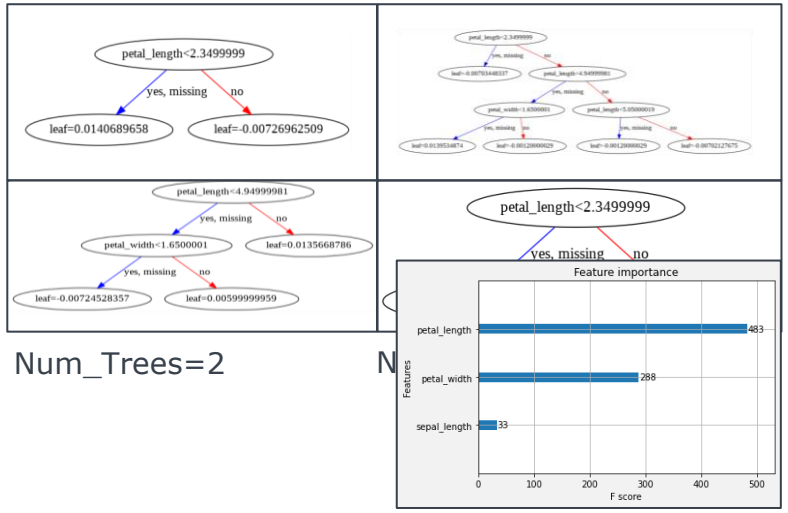
Regression kontinuierliche Variable




```
import xgboost as xgb
XGB_Regressor = xgboost.XGBRegressor()
XGB_Regressor.fit(X_train,y_train) # Modell trainieren
Y_pred_XGBR = XGB_Regressor.predict(X_test)
```

Für diskrete, mehrere Klassen

Num_Trees=0 Num_Trees=1




```
import xgboost as xgb
xgb_clf = xgb.XGBClassifier()
xgb_clf.fit(X_train,y_train) # Modell trainieren
Y_pred_XGB_CLF = xgb_clf.predict(X_test)
```

Vorteile:

- Hervorragende Performance für strukturierte Daten.
- Sehr hohe Genauigkeit.
- Parallelisierbarkeit Training.
- Feature Importance (was sind die wichtigsten Features für Vorhersage?).

Nachteile:

- Erklärbarkeit Algorithmus.
- Arbeitet nur mit numerischen Werten (Kategorische Variablen müssen in 1-dimensionale Features je Wert umgewandelt werden).
- Hyperparameter-Tuning notwendig für Vermeiden Overfitting.

Einsatz mehrerer Modelle mit Ziel, durch Hinzunahme zusätzlicher Modelle die Fehlklassifikationen der bisherigen Modelle zu reduzieren (Start mit Modell 1, dann trainiere Modell 2 auf Fehler von Modell 1, …).

Quellen: Chen, Guestrin: „XGBoost: A Scalable Tree Boosting System", 2016.

5. VERGLEICH/ GEGENÜBERSTELLUNG VERFAHREN

GEGENÜBERSTELLUNG ERGEBNISSE ALGORITHMEN AM BEISPIEL VON DREI DATENSÄTZEN.

Kein Tuning Parameter eingesetzt, Source Code liegt im Github.

ALGORITHMUS	REGRESSION MODIFIZIERTE SINUS-KURVE	IRIS CLASSIFICATION AUF 2 SEPAL-FEATURES (SEHR KLEINE DATENMENGE)	IRIS CLASSIFICATION AUF 4 FEATURES (KLEINE DATENMENGE)	TITANIC CLASSIFICATION (KLEINE DATENMENGE)
Lineare Regression	0.43	0.82	0.98	0.80
Naive Bayes	0.43	0.78	**1.0**	0.79
Entscheidungsbaum	0.76	**0.93**	0.98	0.79
K-Nearest Neighbours	0.80	0.83	0.98	0.81
Support Vector Machine/ Regressor	0.77	0.82	0.98	0.79
Random Forest	0.91	**0.93**	0.98	**0.82**
XGBoost	**0.99**	0.82	0.98	0.79

Nächste Woche(n): Verfahren für große/ sehr große Datenmengen

Man weiß sehr häufig nicht im Voraus, welche Algorithmen am besten für einen Datensatz performen.
Deshalb Empfehlung: verschiedene Algorithmen ausprobieren und deren Parameter tunen

6. CASE STUDY

CASE STUDY IN GRUPPENARBEIT

1. Laden Sie in Google Colab eines der Übungsaufgaben-Notebooks für Vorlesung 6 aus folgendem Verzeichnis: Link
2. Wenden Sie den gelernten Workflow des Supervised learnings auf den Datensatz an.
   1. Probieren Sie verschiedene Verteilungen Trainings- und Testmengen aus. Wie ändern sich die Ergebnisse?
   2. Wählen Sie 2 verschiedene „simple" Algorithmen des überwachten Lernens an.
      - Welcher Algorithmus liefert die höchste Genauigkeit?
      - Wie ändern sich die Ergebnisse, falls Sie die Hyperparameter des Algorithmus ändern? Die Hyperparameter finden Sie auf der jeweiligen Seite des Algorithmus (SciKit: Link).
   3. Wenden Sie XGBoost und RandomForest an.
      - Wie ändern sich die Ergebnisse, falls Sie die Hyperparameter des Algorithmus ändern? Die Hyperparameter finden Sie auf der jeweiligen Seite des Algorithmus (RandomForest: Link, XGBoost: Link).
      - Gibt es Unterschiede bzgl. Genauigkeit?
      - Wie sind die Unterschiede zu den „simplen" Algorithmen?
3. Stellen Sie Ihre Ergebnisse, Hypothesen und Plots vor.

ZUSAMMENFASSUNG

ZUSAMMENFASSUNG SUPERVISED LEARNING.

Vorteile:

- Wir erhalten „bewertete" Ergebnisse.
- Eigenständiges Handeln des Algorithmus möglich.

Nachteile:

- Bestimmen Zielvariable (Labeln) wichtig und aufwendig.
- Datenqualität sehr wichtig.
- Kein Algorithmus perfekt, Tuning notwendig (v.a. Hyperparameter).

LITERATUR UND WEITERE QUELLEN (AUSZUG).

Künstliche Intelligenz:

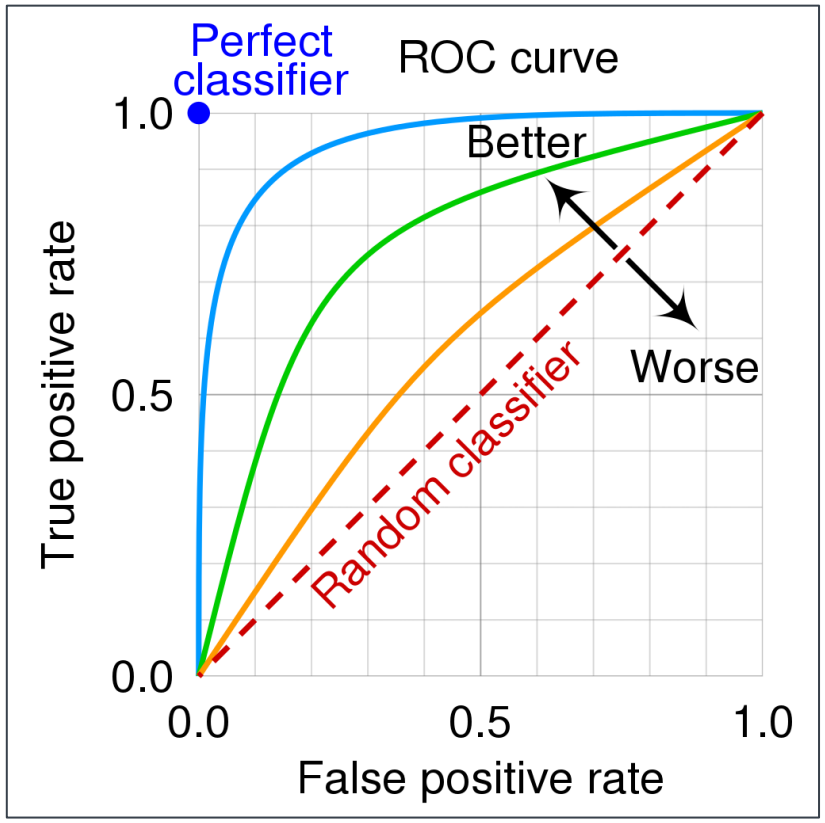
- Gröner, Heinecke: Kollege KI
- Burkov: The Hundred-Page Machine Learning Book, online verfügbar unter Link
- Nielsen: Neural Networks and Deep Learning, online verfügbar unter Link
- Russel, Norvig: Artificial Intelligence – a modern approach
- Produktentwicklung mit AI:
  - Ameisen: Building Machine Learning Powered Applications: Going from Idea to Product
  - Ng: Machine Learning Yearning, online verfügbar unter Link

Kostenfreie Online-Kurse (bei Interesse):

- Python-Kurse
  - Python for Everybody (Link)
  - Udacity Python Course (Link)
  - **Coursera Course Deep Learning** (Link)
  - FAST AI (Link)

BACKUP

DETAILLIERUNG KLASSIFIZIERUNGSMETRIKEN. ROC/ AUC.



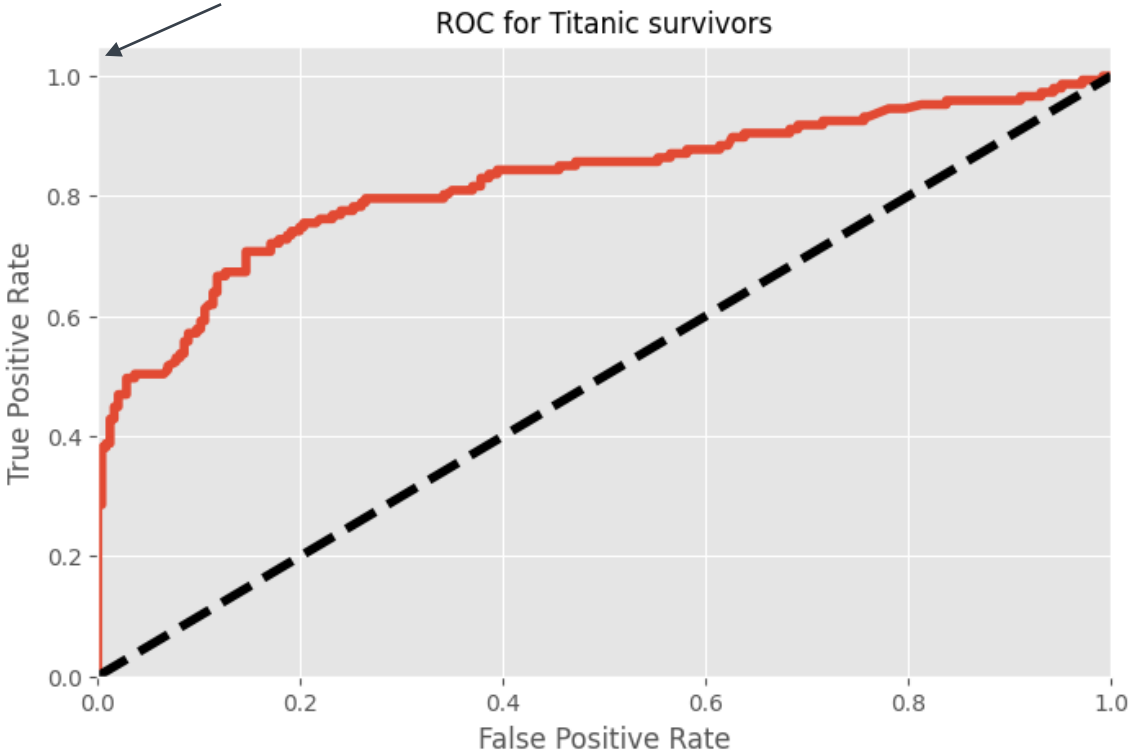
- **ROC**(Receiver-Operating-Characteristics): Darstellung des prozentualen Verhältnisses zwischen den richtig (true positive) sowie den falsch als wahr vorhergesagten (false positive) Werten.

- **AUC**(Area under curve): Flächenintegral der ROC-Kurve.
  Je größer die AUC-Fläche, desto besser ist der Klassifikator.

Quelle Abbildung: Link

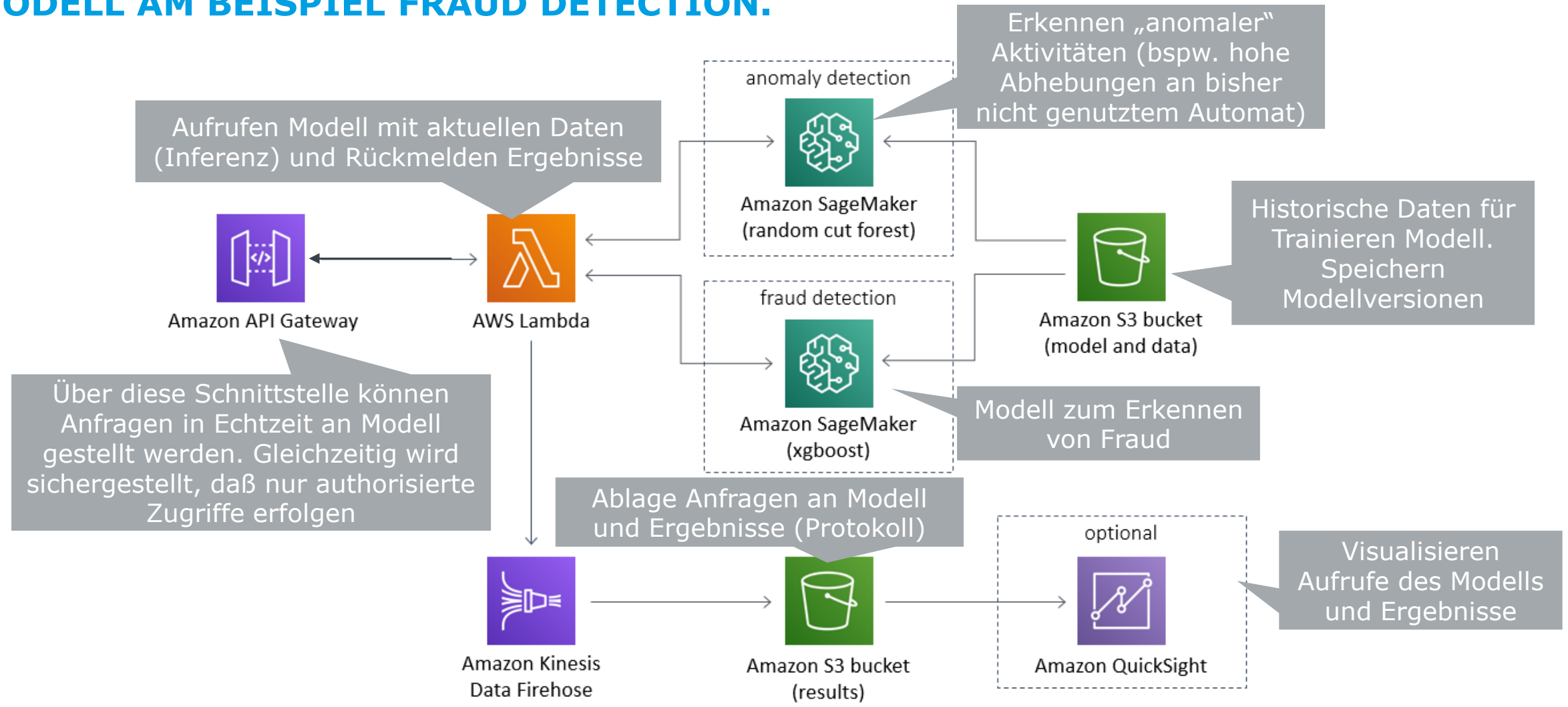
DETAILLIERUNG KLASSIFIZIERUNGSMETRIKEN. ROC/ AUC.

Perfekter Klassifikator (100% true pos., 0% false pos.)



- **ROC**(Receiver-Operating-Characteristics): Darstellung prozentuales Verhältnis zwischen den richtig und falsch als wahr prädizierten Werten.
- **AUC**(Area under ROC Curve): Integral der ROC-Kurve. Je höher AUC-Fläche, desto besser der Klassifikator.

DETAILLIERUNG DEPLOYMENT MACHINE LEARNING MODELL AM BEISPIEL FRAUD DETECTION.



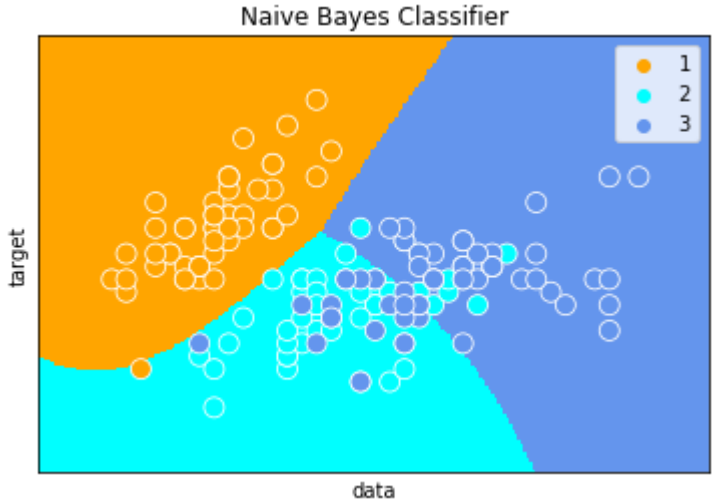
Quelle: Amazon Web Services Referenzarchitekturen: Link

SUPERVISED LEARNING: NAIVE BAYES.

Regression kontinuierliche Variable

-

Klassifizierung diskrete Variable




```
from sklearn.naive_bayes import  GaussianNB
gnb = GaussianNB()
gnb.fit(X_train,y_train) # Modell trainieren
Y_pred_GB = gnb.predict(X_test)
```

Vorteile:

- Kann gut mit hochdimensionalen Daten umgehen.
- Wenig Trainingsdaten notwendig.
- Schnell berechenbarer Algorithmus.
- Oft bessere Genauigkeit als kompliziertere Modelle (aber nur falls Variablen statistisch unabhängig).

Nachteile:

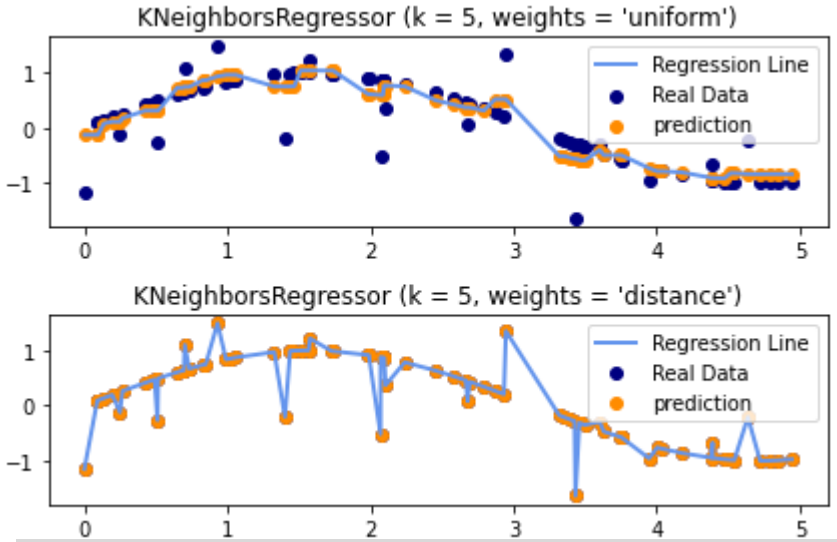
- Annahme bedingte Unabhängigkeit nicht immer gegeben.
- Bei großen Datenmengen liefern andere Algorithmen bessere Ergebnisqualität (wegen Bayes-Regel für Updates).

Nutzt Bayes-Theorem für das Berechnen der Zugehörigkeit eines Features zu einer Klasse und ordnet das Feature der Klasse zu, für die die berechnete Wahrscheinlichkeit am höchsten ist.

Quellen: SK-Learn Dokumentation: Link

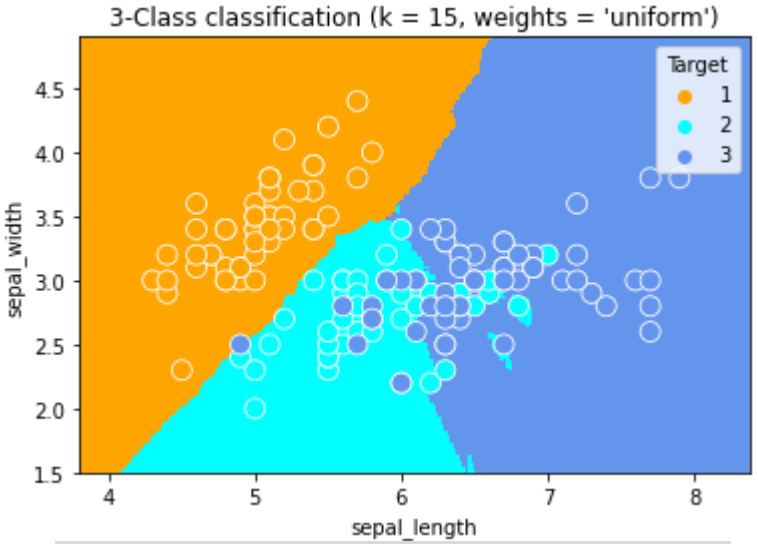
SUPERVISED LEARNING: K NEAREST NEIGHBOURS.

Regression kontinuierliche Variable




```
from sklearn import neighbors
knn = neighbors.KNeighborsRegressor(weights="distance")
knn.fit(X_train,y_train) # Modell trainieren
Y_pred_knn = knn.predict(X_test)
```

Klassifizierung diskrete Variable




```
from sklearn import neighbors
knn = neighbors.KNeighborsClassifier()
knn.fit(X_train,y_train) # Modell trainieren
Y_pred_knn = knn.predict(X_test)
```

Vorteile:

- Einfach verständlich.
- Einsetzbar für Regression und Klassifikation.
- Ohne Labels und ohne Training einsetzbar.
- Nur 1 Hyperparameter.

Nachteile:

- Lernt nicht aus Trainingsdaten.
- Hoher Speicherbedarf (ganzer Datensatz wird in Speicher abgelegt).
- Ausreißer/ fehlende Daten haben großen Einfluß auf Algorithmus.
- Probleme mit hochdim. Daten.
- Selten eingesetzt für Regression.

Klassifikation eines Punktes erfolgt anhand Mehrheitsvotum seiner k-Nachbarn.
Regression erfolgt anhand Bilden des gewichteten Durchschnitt der Zielwerte der k-Nachbarn.

Quellen: SK-Learn Dokumentation: Link, Link
KNN: Fix, E., Hodges, J.L. Discriminatory analysis, nonparametric discrimination: Consistency properties. Technical Report 4, 1951.

Microsoft Azure Machine Learning Algorithm Cheat Sheet

This cheat sheet helps you choose the best machine learning algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the goal you want to achieve with your data.

What do you want to do?

Extract information from text

Text Analytics

Derives high-quality information from text

Answers questions like: What info is in this text?

- **Extract N-Gram Features from Text** ← Creates a dictionary of n-grams from a column of free text
- **Feature Hashing** ← Converts text data to integer encoded features using the Vowpal Wabbit library
- **Preprocess Text** ← Performs cleaning operations on text, like removal of stop-words, case normalization
- **Word2Vector** ← Converts words to values for use in NLP tasks, like recommender, named entity recognition, machine translation

Predict values

Regression

Makes forecasts by estimating the relationship between values

Answers questions like: How much or how many?

- **Fast Forest Quantile Regression** ← Predicts a distribution
- **Poisson Regression** ← Predicts event counts
- **Linear Regression** ← Fast training, linear model
- **Bayesian Linear Regression** ← Linear model, small data sets
- **Decision Forest Regression** ← Accurate, fast training times
- **Neural Network Regression** ← Accurate, long training times
- **Boosted Decision Tree Regression** ← Accurate, fast training times, large memory footprint

Generate recommendations

Recommenders

Predicts what someone will be interested in

Answers the question: What will they be interested in?

- **SVD Recommender** ← Collaborative filtering, better performance with lower cost by reducing dimensionality

Discover structure

Clustering

Separates similar data points into intuitive groups

Answers questions like: How is this organized?

- **K-Means** ← Unsupervised learning

Find unusual occurrences

Anomaly Detection

Identifies and predicts rare or unusual data points

Answers the question: Is this weird?

- **One Class SVM** ← Under 100 features, aggressive boundary
- **PCA-Based Anomaly Detection** ← Fast training times

Predict between several categories

Multiclass Classification

Answers complex questions with multiple possible answers

Answers questions like: Is this A or B or C or D?

- **Multiclass Logistic Regression** ← Fast training times, linear model
- **Multiclass Neural Network** ← Accuracy, long training times
- **Multiclass Decision Forest** ← Accuracy, fast training times
- **One-vs-All Multiclass** ← Depends on the two-class classifier
- **Multiclass Boosted Decision Tree** ← Non-parametric, fast training times and scalable

Predict between two categories

Two-Class Classification

Answers simple two-choice questions, like yes or no, true or false

Answers questions like: Is this A or B?

- **Two-Class Support Vector Machine** ← Under 100 features, linear model
- **Two-Class Averaged Perceptron** ← Fast training, linear model
- **Two-Class Decision Forest** ← Accurate, fast training
- **Two-Class Logistic Regression** ← Fast training, linear model
- **Two-Class Boosted Decision Tree** ← Accurate, fast training, large memory footprint
- **Two-Class Neural Network** ← Accurate, long training times

Classify images

Image Classification

Classifies images with popular networks

Answers questions like: What does this image represent?

- **DenseNet** ← High accuracy, better efficiency

Share this poster: aka.ms/mlcheatsheet

Microsoft