

# TECHNICAL APPLICATIONS AND DATA MANAGEMENT. SS 2023.

## VORLESUNG 6 UND 7.

09.05.2023

MÜNCHEN

STUDIENGANG  
SUSTAINABILITY  
MANAGEMENT



## AGENDA

### 1. Überwachtes Lernen:

- Workflow
- „Simple“ Lernverfahren
- Kombinierte Lernverfahren / Ensemble Learning
- Metriken für überwachtes Lernen

### 2. Case Study

## GEPLANTE ROADMAP DER VORLESUNG.

ROADMAP	WAS HABEN WIR VOR?
Vorlesung 1	Übersicht und Einführung
Vorlesung 2	Einführung Data Science und Data Science Workflow, Detaillierung Data Engineering
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 und Vorlesung 7	Überwachtes Lernen
	Vertiefung überwachtes Lernen anhand Case Study
Vorlesung 8 und Vorlesung 9	Neuronale Netze und Convolutional Neural Networks (CNN)
	Vertiefung CNN anhand Case Study, Aufgabenstellung AI
Vorlesung 10	Rekurrente Neuronale Netze
Vorlesung 11	Generative AI
Vorlesung 12	Ausblick
Vorlesung 13	„Fragestunde“

Folien der bisherigen Vorlesung verfügbar unter [Link](#)

# 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

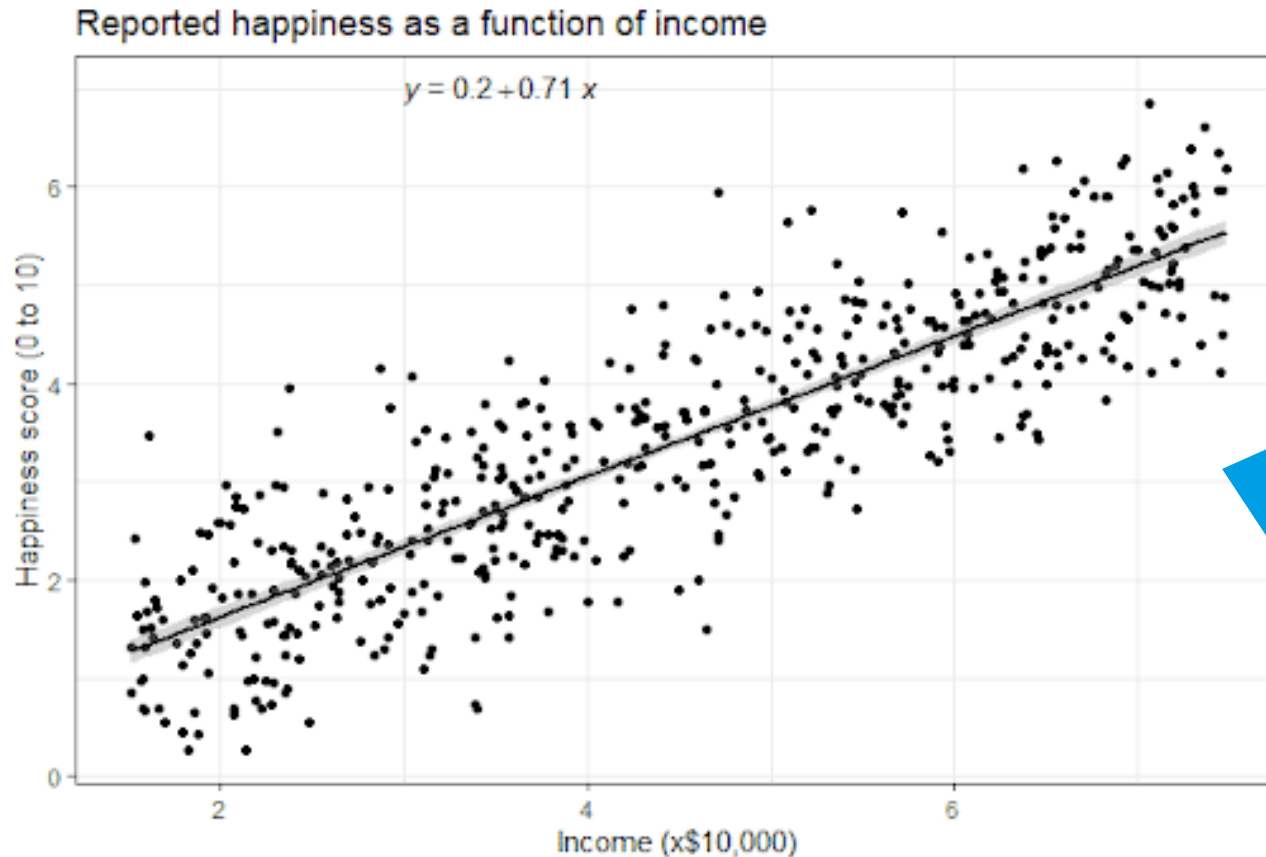
**Aber:**

- Hoher Aufwand für das Bestimmen der Zielwerte (= Label). Dies ist beim unüberwachten 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.
- ...

## BEISPIEL SUPERVISED LEARNING REGRESSION: LINEARE REGRESSION.



### Veranschaulicht:

- Wir wollen eine Gerade/ n-dimensionale Ebene lernen, die bestehende Datenpunkte möglichst genau abbildet, um so weitere Punkte vorherzusagen.
- Um die bestehenden Punkte möglichst genau abzubilden, wird mittels iterativen Prozess der Abstand aller Datenpunkte zur Gerade minimiert.
- Da der Abstand in beide Richtungen gehen kann, wird der Abstand quadriert. Deshalb heißt das auch „Methode der kleinsten Quadrate“ (Gauss, 1795)



# SUPERVISED LEARNING: KLASSIFIKATION

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

**Ermöglicht:** Klassifizieren, d.h. Einordnen eines Wertes in eine endliche Gruppe bzw. Prädiktion eines endlichen Wertes.

**Aber:**

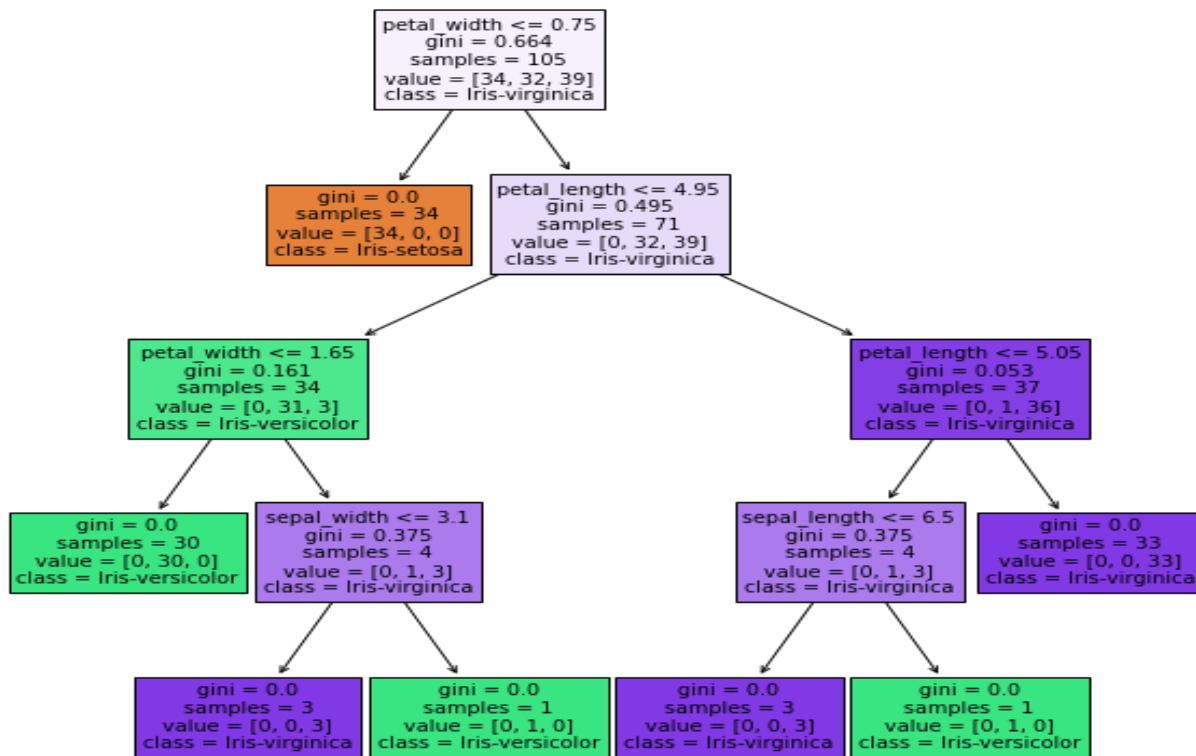
- Hoher Aufwand für das Bestimmen der Zielwerte (Labeln). Dies ist beim unüberwachten Lernen nicht vorhanden.
- Genauigkeit hängt von der Größe der Gruppe ab → für sehr große Gruppen haben wir ja quasi eine Regression.

**Anwendungsbeispiele:**

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



# BEISPIEL SUPERVISED LEARNING KLASSIFIKATION: DECISION TREE.



## Veranschaulicht:

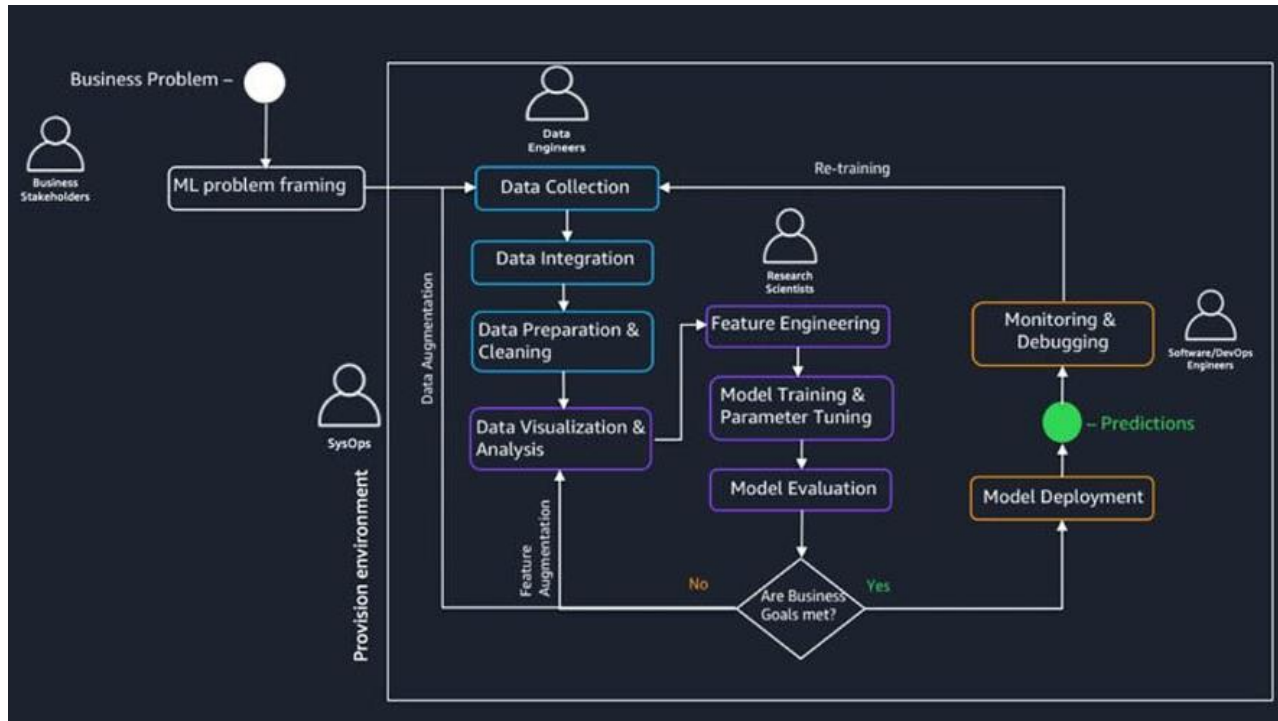
- Features sollen Datensatz möglichst gut in verschiedene Zielklassen einteilen (meist: 2 Klassen).
- Je genauer/ „reiner“ ein Feature das kann, desto besser (niedriger Gini Impurity Wert).
- Je höher der Impurity-Wert, desto mehr falsche Einteilungen (Fehlklassifikationen).
- Der Algorithmus lernt während des Trainings, welche/s Feature das am besten kann.
- Je besser ein Feature, desto weiter oben im Baum.

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.



## 2. GENERISCHER ABLAUF ÜBERWACHTES LERNEN.

# GENERISCHER ABLAUF SUPERVISED LEARNING. ÜBERSICHT.



1. Customer/ Business problem statement und Metriken definieren
2. Daten organisieren, aufbereiten, säubern, ggf. neue Features.
3. Machine Learning Vorbereitung: Daten aufteilen in Trainings- und Testmenge (und ggf. Validierungsmenge). Algorithmus und Kostenfunktion wählen und Modellparameter initialisieren.
4. ML Training: schrittweise Optimierung Modellparameter bis Modell möglichst gute Performance für Trainingsmenge hat.
5. Modell(-güte) validieren mit Testmenge, dessen Elemente Modell nicht kannte.  
Falls Modellgüte i.O., weiter zu Schritt 7. Sonst zu Schritt 5.
6. Deployment: Modell einsetzen im „Live“-Betrieb inkl. kontinuierliches Überprüfen Modellgüte und ggf. Aktualisierung.

Ziel: Lernen eines möglichst genauen Modells  $H(\text{Input}) = \text{Zielgröße}^1$

# **FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY.**

## **SCHRITT 1: BUSINESS/ CUSTOMER PROBLEM STATEMENT.**

- Allgemein:
  - Welches Business/ Customer problem wollen Sie mit AI lösen?
  - Woran erkennen Sie, ob und wie gut Sie das Problem lösen?
  - Woher bekommen Sie die Daten?

→ Dies kann bspw. mit einer Business Canvas oder Ansatz „Working backwards from Customer“ erfolgen.
  
- In unserem Fall:
  - Business Problem: Automatisierung Kreditvergabe inkl. Vermeiden von Kreditausfall.
  - Kundenproblem: langfristiger Prozess Kreditvergabe.
  - Problem wird gut gelöst, wenn schnell eine Aussage über Kreditwürdigkeit erfolgt bei Minimierung Kreditausfälle.
  - Woher kommen Daten: Beispielhafter Datensatz.

# FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY.

## SCHRITT 2.1: DATEN ORGANISIEREN.

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urban	Y
6	LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y
7	LP001014	Male	Yes	3+	Graduate	No	3036	2504.0	158.0	360.0	0.0	Semiurban	N
8	LP001018	Male	Yes	2	Graduate	No	4006	1526.0	168.0	360.0	1.0	Urban	Y
9	LP001020	Male	Yes	1	Graduate	No	12841	10968.0	349.0	360.0	1.0	Semiurban	N
10	LP001024	Male	Yes	2	Graduate	No	3200	700.0	70.0	360.0	1.0	Urban	Y
11	LP001027	Male	Yes	2	Graduate	NaN	2500	1840.0	109.0	360.0	1.0	Urban	Y
12	LP001028	Male	Yes	2	Graduate	No	3073	8106.0	200.0	360.0	1.0	Urban	Y
13	LP001029	Male	No	0	Graduate	No	1853	2840.0	114.0	360.0	1.0	Rural	N
14	LP001030	Male	Yes	2	Graduate	No	1299	1086.0	17.0	120.0	1.0	Urban	Y
15	LP001032	Male	No	0	Graduate	No	4950	0.0	125.0	360.0	1.0	Urban	Y
16	LP001034	Male	No	1	Not Graduate	No	3596	0.0	100.0	240.0	NaN	Urban	Y
17	LP001036	Female	No	0	Graduate	No	3510	0.0	76.0	360.0	0.0	Urban	N
18	LP001038	Male	Yes	0	Not Graduate	No	4887	0.0	133.0	360.0	1.0	Rural	N
19	LP001041	Male	Yes	0	Graduate	NaN	2600	3500.0	115.0	NaN	1.0	Urban	Y

# FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY.

## SCHRITT 2.2: DATEN AUFBEREITEN.

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
1	1	1	1	0	0	4583	1508.0	128.0	360.0	1.0	0	0
2	1	1	0	0	1	3000	0.0	66.0	360.0	1.0	2	1
3	1	1	0	1	0	2583	2358.0	120.0	360.0	1.0	2	1
4	1	0	0	0	0	6000	0.0	141.0	360.0	1.0	2	1
5	1	1	2	0	1	5417	4196.0	267.0	360.0	1.0	2	1
6	1	1	0	1	0	2333	1516.0	95.0	360.0	1.0	2	1
7	1	1	3	0	0	3036	2504.0	158.0	360.0	0.0	1	0
8	1	1	2	0	0	4006	1526.0	168.0	360.0	1.0	2	1
9	1	1	1	0	0	12841	10968.0	349.0	360.0	1.0	1	0
10	1	1	2	0	0	3200	700.0	70.0	360.0	1.0	2	1
12	1	1	2	0	0	3073	8106.0	200.0	360.0	1.0	2	1
13	1	0	0	0	0	1853	2840.0	114.0	360.0	1.0	0	0
14	1	1	2	0	0	1299	1086.0	17.0	120.0	1.0	2	1
15	1	0	0	0	0	4950	0.0	125.0	360.0	1.0	2	1
17	0	0	0	0	0	3510	0.0	76.0	360.0	0.0	2	0
18	1	1	0	1	0	4887	0.0	133.0	360.0	1.0	0	0
20	1	1	0	1	0	7660	0.0	104.0	360.0	0.0	2	0
21	1	1	1	0	0	5955	5625.0	315.0	360.0	1.0	2	1
22	1	1	0	1	0	2600	1911.0	116.0	360.0	0.0	1	0
25	1	1	0	0	1	9560	0.0	191.0	360.0	1.0	1	1

Eingesetzte Data Engineering Operationen:

- Löschen Spalte Loan\_id
- Label Encoding: jeder Wert eines Features erhält eine eindeutige Zahl

**Saubere, viele Daten sind der wichtigste Faktor um ein möglichst gutes Modell zu erhalten!**

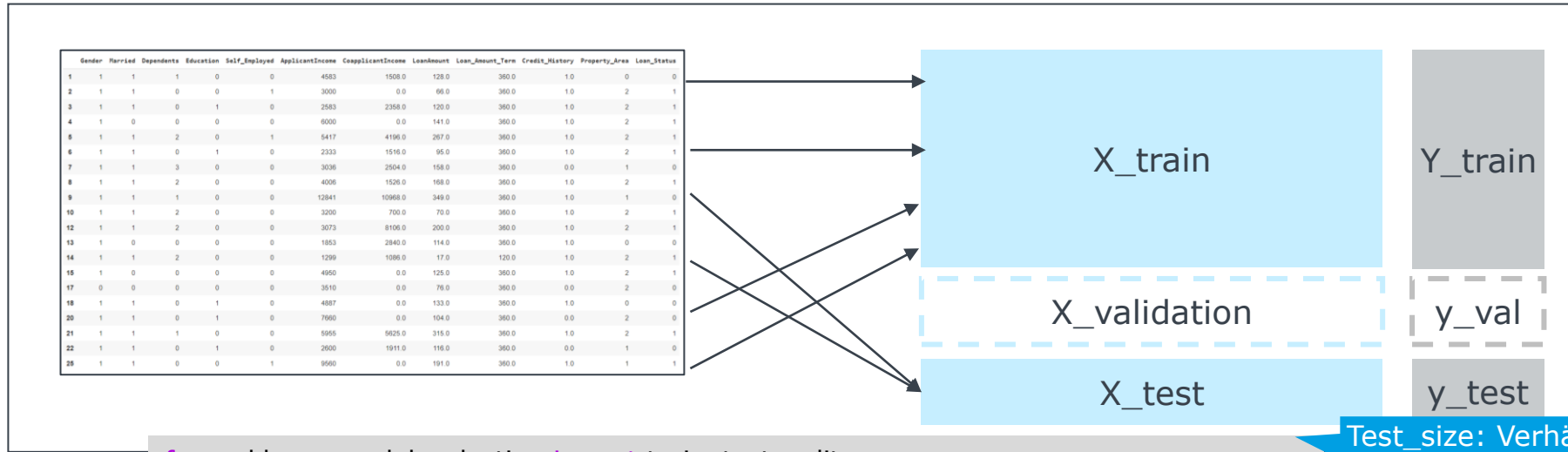
Ziel der Datenaufbereitung ist das Verbessern der Datenqualität um somit bessere Ergebnisse beim Machine Learning zu erhalten. Dazu gehört z.B. das Beseitigen von Fehlern, Datentypumwandlung, Erstellen neuer Features und Bestimmen Zielklassen („labeln“).

# FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY.

## SCHRITT 2.3: DATENAUFTEILUNG.

Matrix mit Features/ Eigenschaften

Vektor mit den Zielwerten



```
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%.

Eselsbrücke anhand Prüfungsvorbereitung (z.B. Führerschein):

- Zur Prüfungsvorbereitung lernen Sie anhand Prüfungsbögen und vergleichen ihre Antworten mit vorgegebenen Musterantworten.
- Sie sehen ihre Fehler und üben an den Prüfungsbögen solange, bis Sie die Prüfungsbögen „gut genug“ können.
- Bei der realen Prüfung kennen Sie die Antworten nicht und sehen dann, wie gut Sie den Stoff können.



# FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY.

## SCHRITT 3: MODELL WÄHLEN UND INITIALISIEREN.

### 1. Wahl Machine Learning-Modell abhängig von:

- Datentyp: Zahlen, Bilder, Sprache, .... → tabellarische Daten
- Verfügbaren Ressourcen: Rechen-Power, Zeit, Kosten, ...
- Datenmenge: viele oder wenig Daten vorhanden? → wenige Daten (Testdatensatz)
- Qualität Daten: fehlende Werte/ Ausreißer stören manche Lernverfahren. → einige fehlende Werte
- Anforderungen an Modellgüte → sehr hoch
- Komplexität Algorithmus: komplexe Algorithmen neigen zu hoher Streuung, simple zu hoher Verzerrung.

Zur Veranschaulichung nehmen wir  
Decision Tree.

### 2. Initialisierung Parameter

- Hyperparameter, d.h. Parameter für Trainingsprozess: bspw. „maximale Tiefe Baum“
- Modellparameter: werden gelernt.

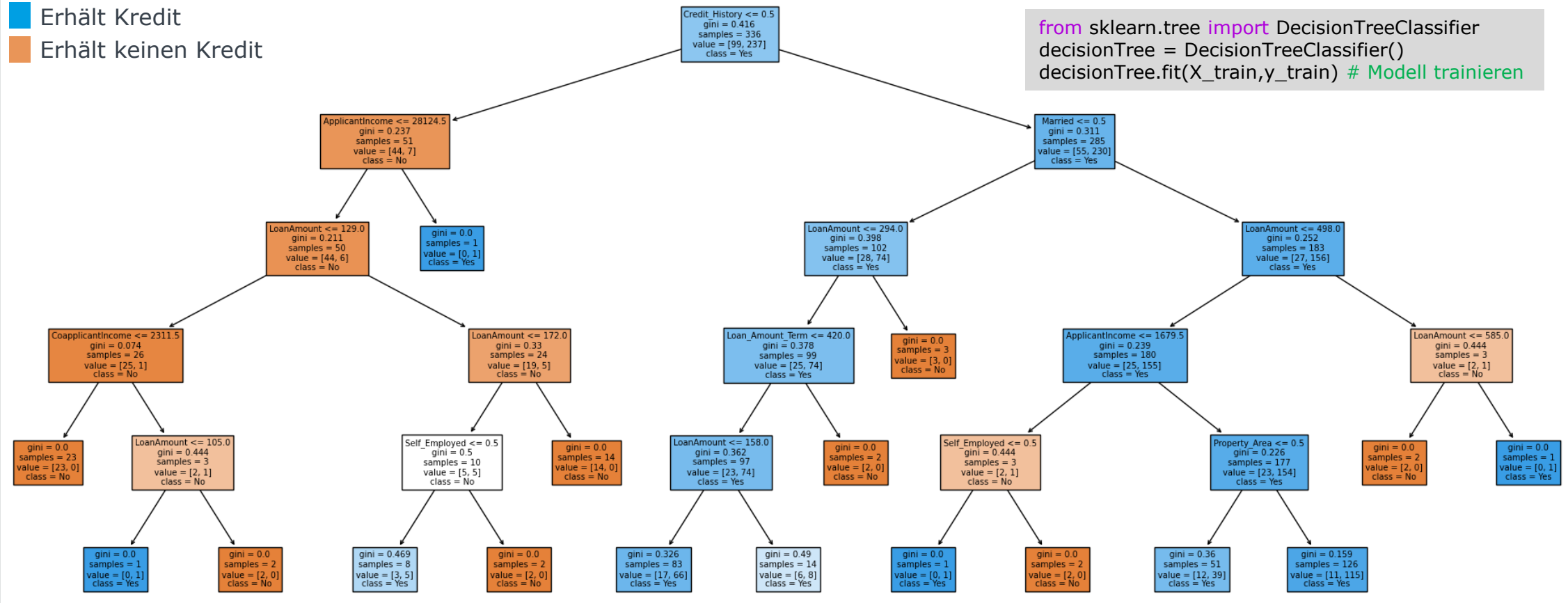
### 3. Wahl Loss-Funktion: Bewertungsfunktion f. Genauigkeit Modell → Anzahl Fehlklassifikationen.

Die Auswahl Machine Learning Verfahren ist abhängig von diesen Fragen und ist erfahrungsbasierend.  
Es gibt für die meisten Datentypen jedoch Standardverfahren, die auch für die Parameter gute Standardwerte treffen.

# FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY. SCHRITT 4: TRAINING

■ Erhält Kredit  
■ Erhält keinen Kredit

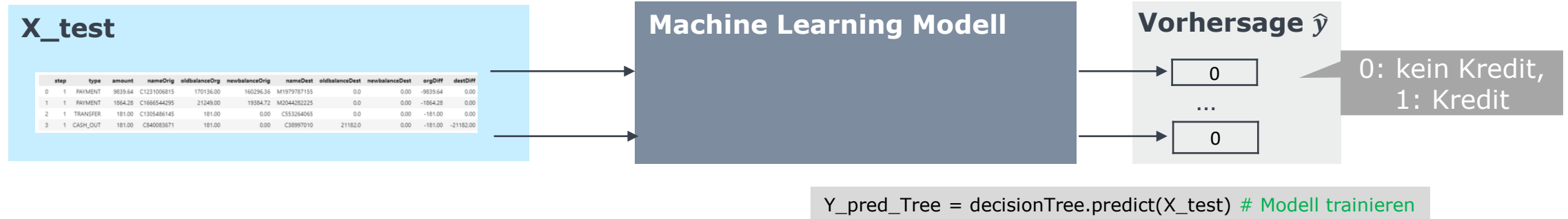
```
from sklearn.tree import DecisionTreeClassifier
decisionTree = DecisionTreeClassifier()
decisionTree.fit(X_train,y_train) # Modell trainieren
```



Klassifikation: wiederholtes Aufteilen Datenset in zwei Untermengen anhand des Features mit höchstem Informationsgehalt.

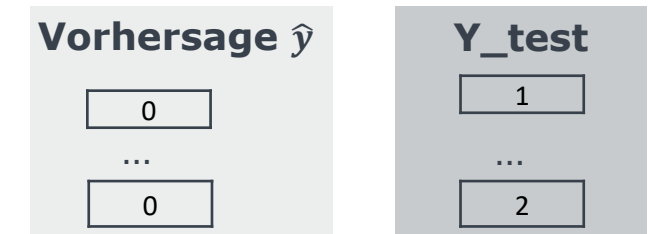
# FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY. SCHRITT 5: MODELL TESTEN.

**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 realer zu vorhergesagtem Wert
- **Klassifikationsmetriken:** Confusion Matrix → nächste Seite
- **Regressionsmetriken:** MSE, RSME, ...



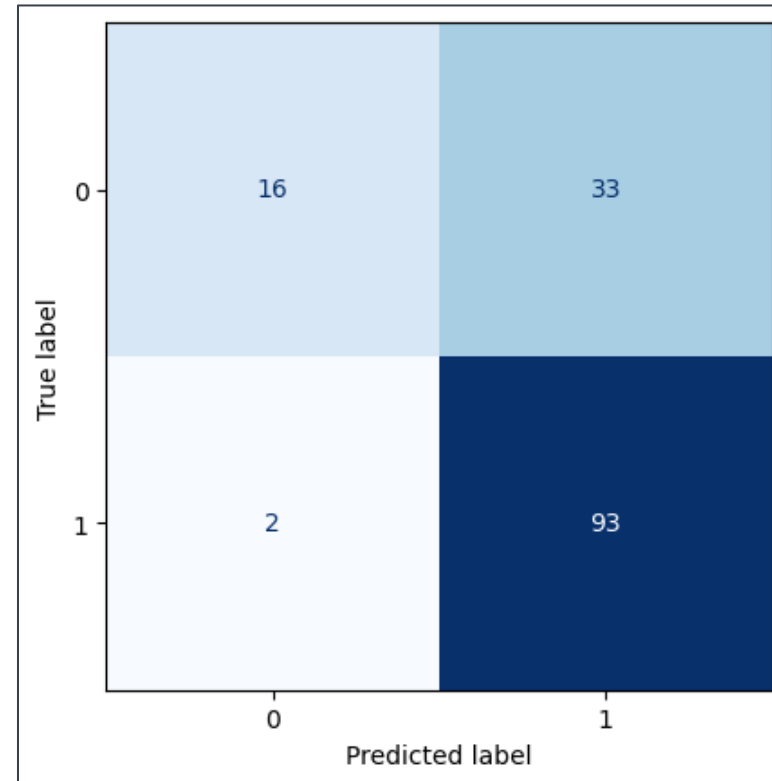
Eselsbrücke anhand Prüfungsvorbereitung (z.B. Führerschein):

- Zur Prüfungsvorbereitung lernen Sie anhand Prüfungsbögen und vergleichen ihre Antworten mit vorgegebenen.
- Sie machen das solange, bis Sie die Prüfungsbögen „gut genug“ können.
- Bei der realen Prüfung kennen Sie die Antworten nicht und sehen dann, wie gut Sie den Stoff können.

## FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY. SCHRITT 5: MODELL TESTEN: ERGEBNISSE.

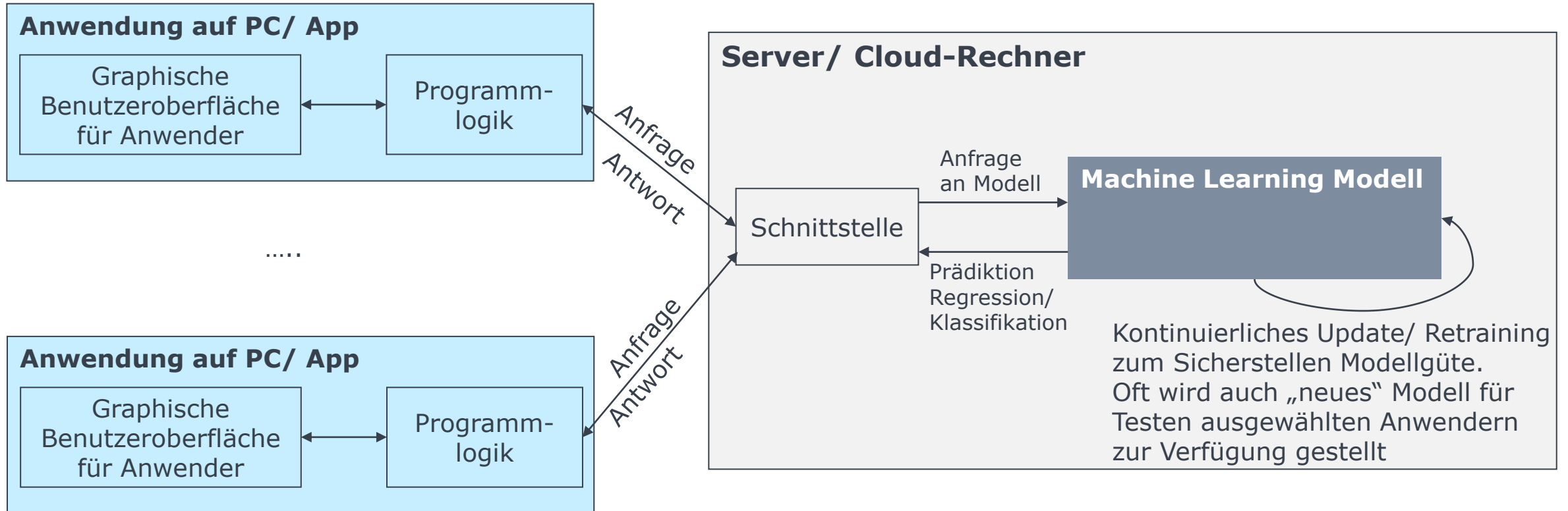
$$\frac{\text{Anzahl richtige Vorhersagen}}{\text{Anzahl aller Vorhersagen}} = 74\%$$

```
from sklearn.metrics import accuracy_score  
accuracy_score(y_pred_DecTree, y_test)
```

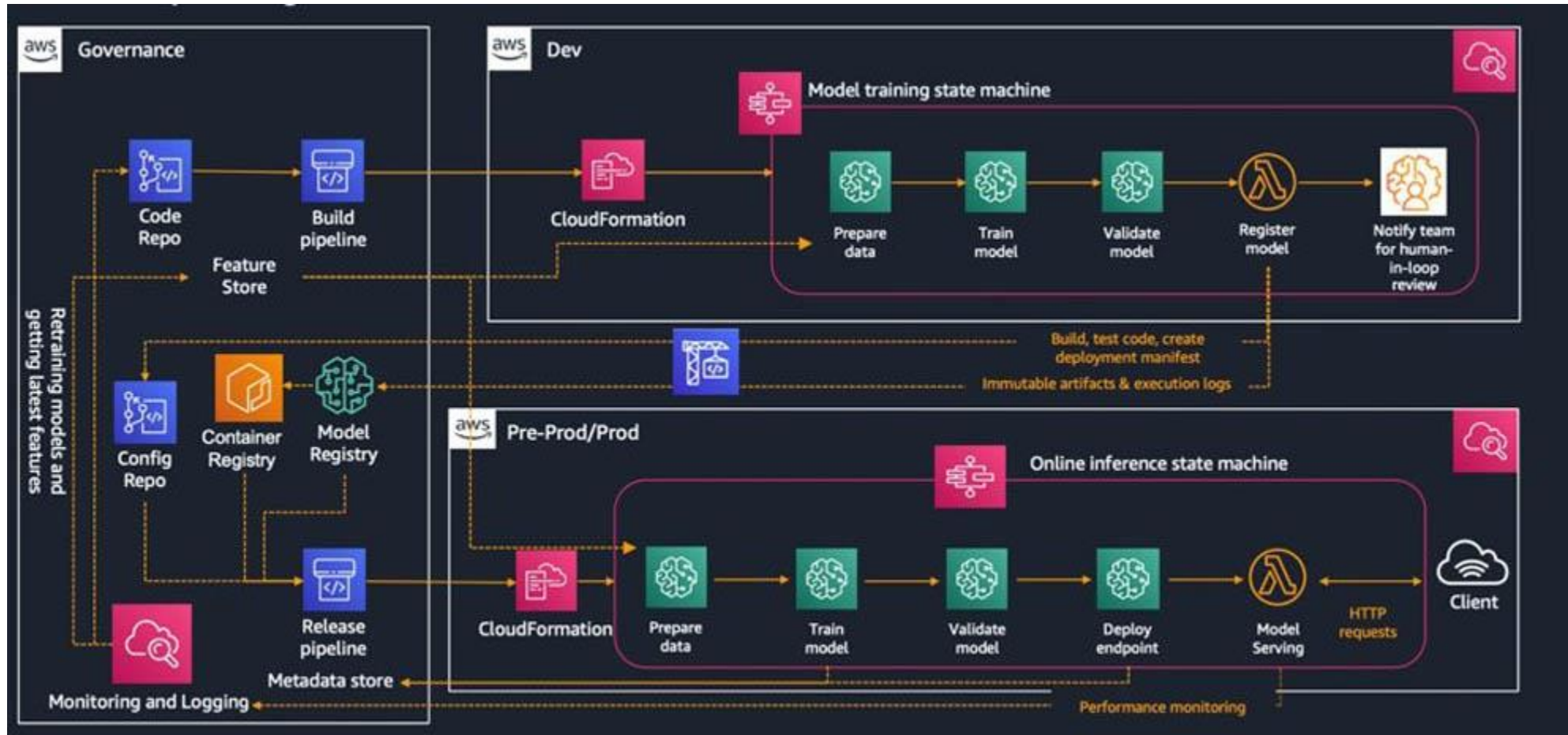


```
from sklearn.metrics import ConfusionMatrixDisplay  
disp = ConfusionMatrixDisplay.from_estimator(clf, X_test, y_test, display_labels=clf.classes_,)
```

## FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY. SCHRITT 6: MODELL EINSETZEN (DEPLOYMENT).



## FALLBEISPIEL MACHINE LEARNING: LOAN ELIGIBILITY. SCHRITT 6: MODEL AT SCALE.





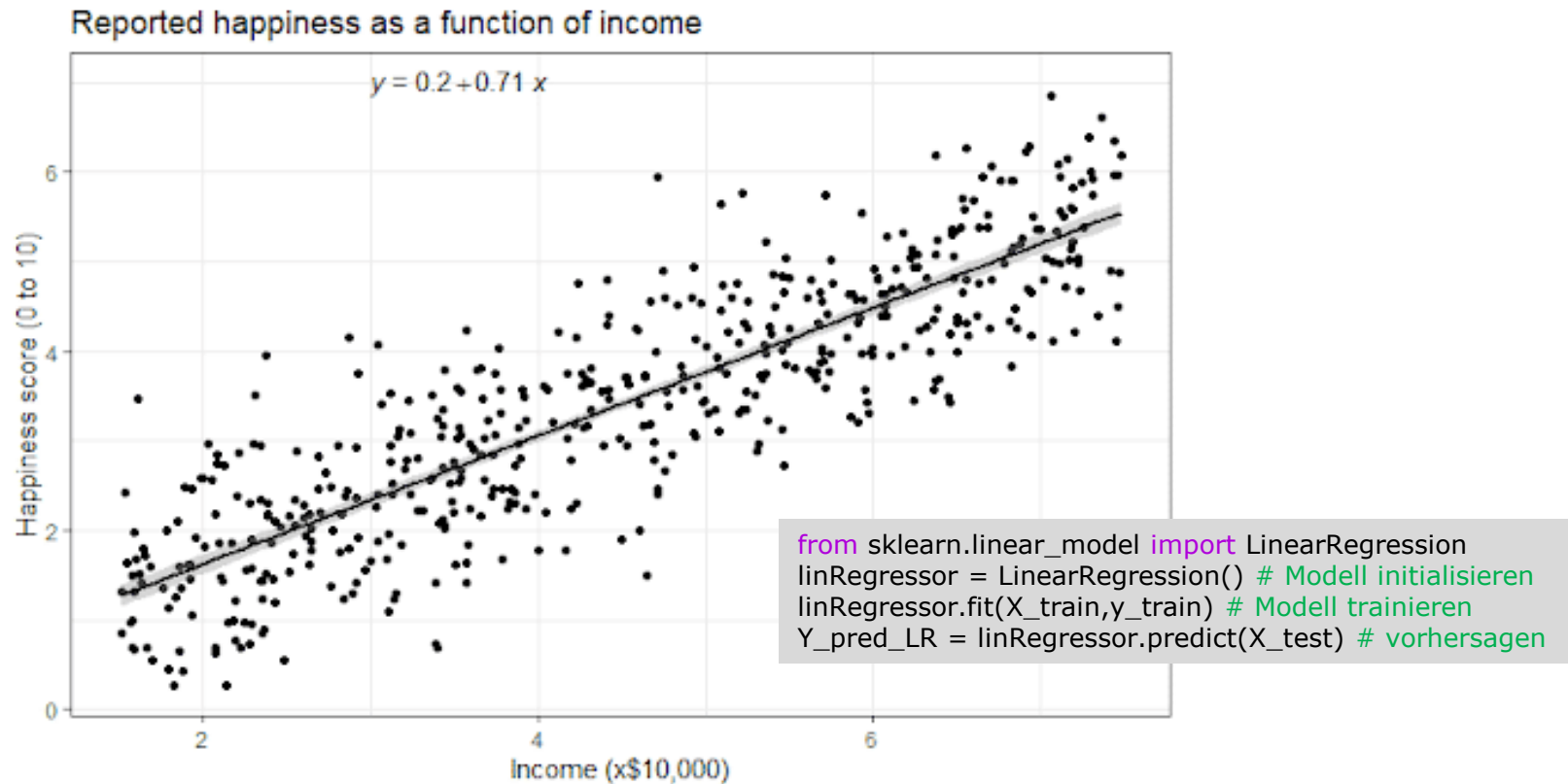
## 3. DETAILLIERUNG AUSGEWÄHLTER ALGORITHMEN





## 3.1 „SIMPLE“ MACHINE LEARNING ALGORITHMEN

# BEISPIEL SUPERVISED LEARNING REGRESSION: LINEARE REGRESSION.



## Vorteile:

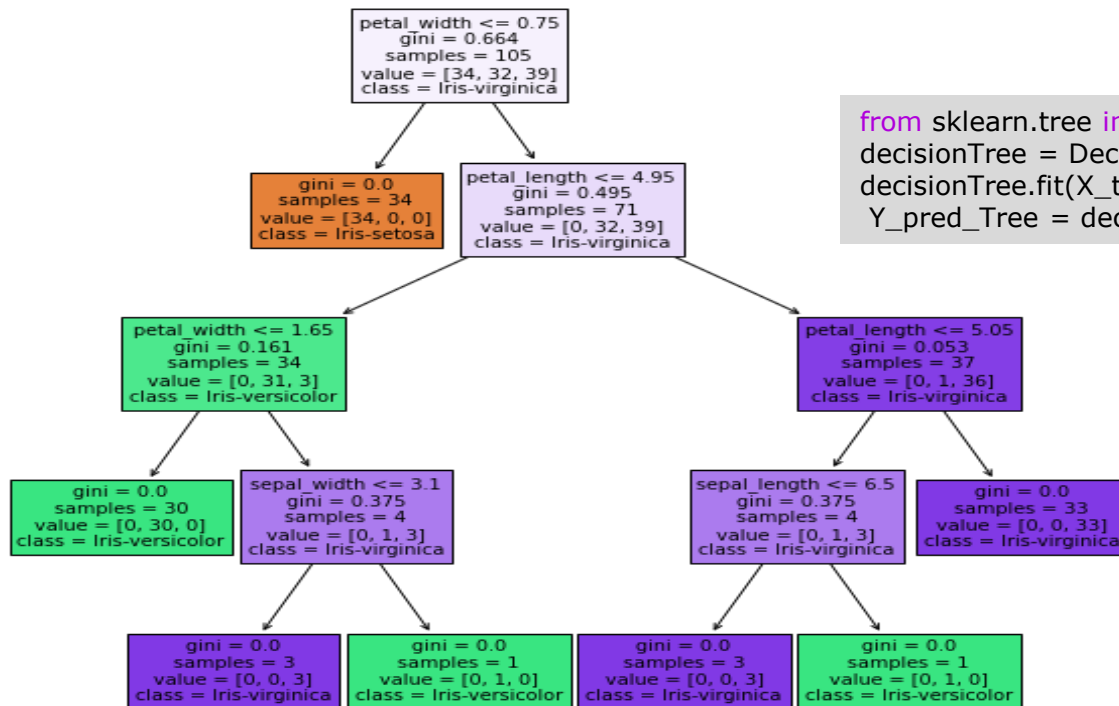
- Leicht verständlicher Algorithmus und Ergebnisse.
- Leicht implementierbar.
- Einsetzbar für Regression und Klassifikation.

## 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.

# BEISPIEL SUPERVISED LEARNING KLASSIFIKATION: DECISION TREE.



```
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.



## 3.2 ENSEMBLE LEARNING

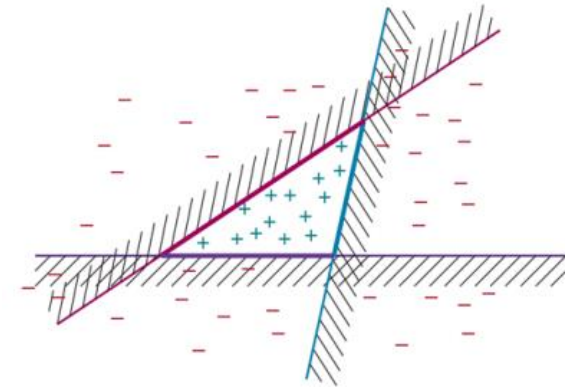
# SUPERVISED LEARNING: ENSEMBLE LEARNING

**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.

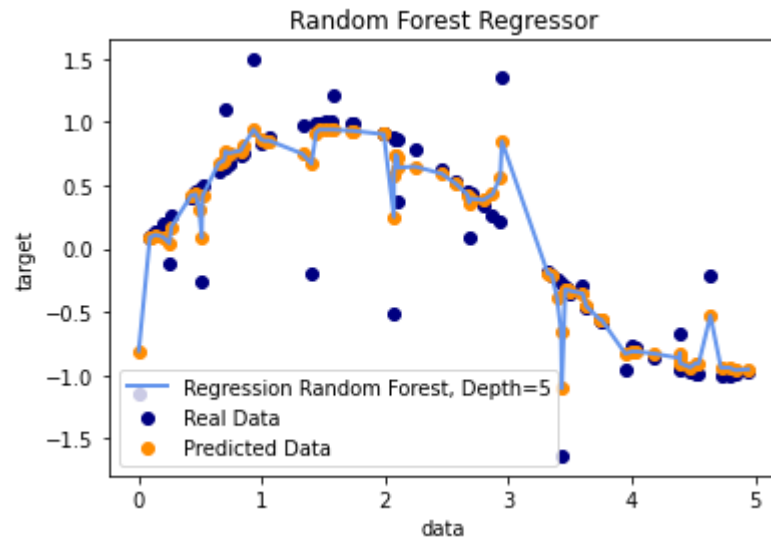


**Häufigst eingesetzte Verfahren:**

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

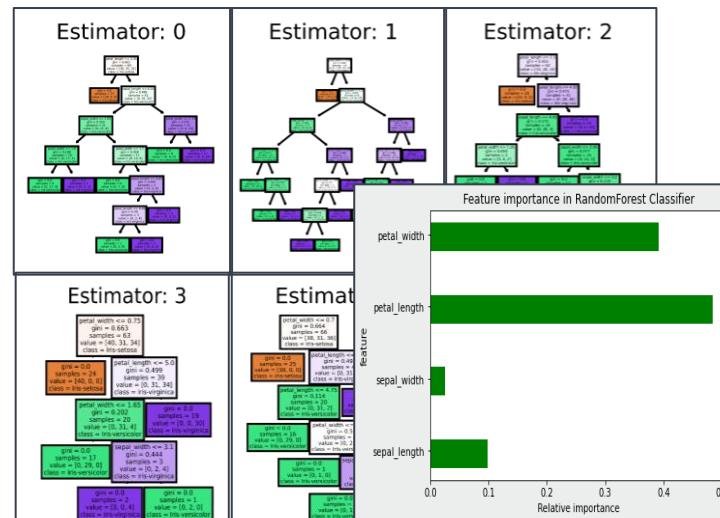
# SUPERVISED LEARNING: RANDOM FOREST.

## 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 Entscheidungsbäume (Reduziert Overfitting durch Gewichtung mehrerer Entscheidungs-).

## 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.

# SUPERVISED LEARNING: XGBOOST.

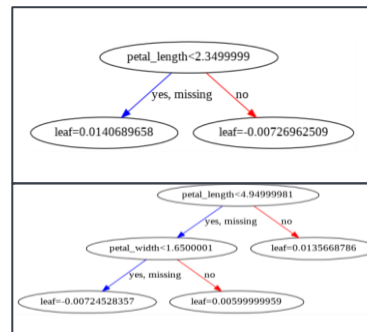
## Regression kontinuierliche Variable



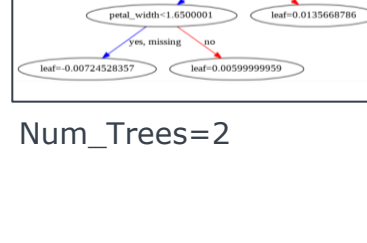
```
import xgboost as xgb
XGB_Regressor = xgboost.XGBRegressor()
XGB_Regressor.fit(X_train,y_train) # Modell trainieren
Y_pred_XGBR = XGB_Regressor.predict(X_test)
```

## Für diskrete, mehrere Klassen

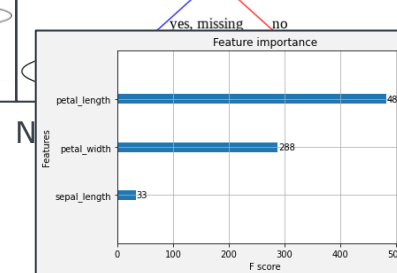
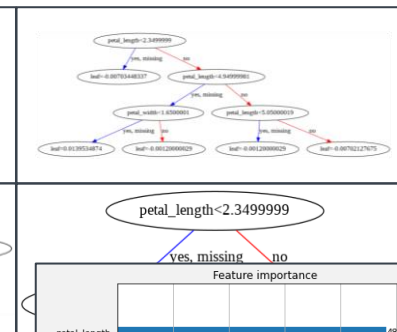
Num\_Trees=0



Num\_Trees=2



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, ...).





## 4. 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	<b>1.0</b>	0.79
Entscheidungsbaum	0.76	<b>0.93</b>	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	<b>0.93</b>	0.98	<b>0.82</b>
XGBoost	<b>0.99</b>	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



## 5. MODELLGÜTE

# 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

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

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

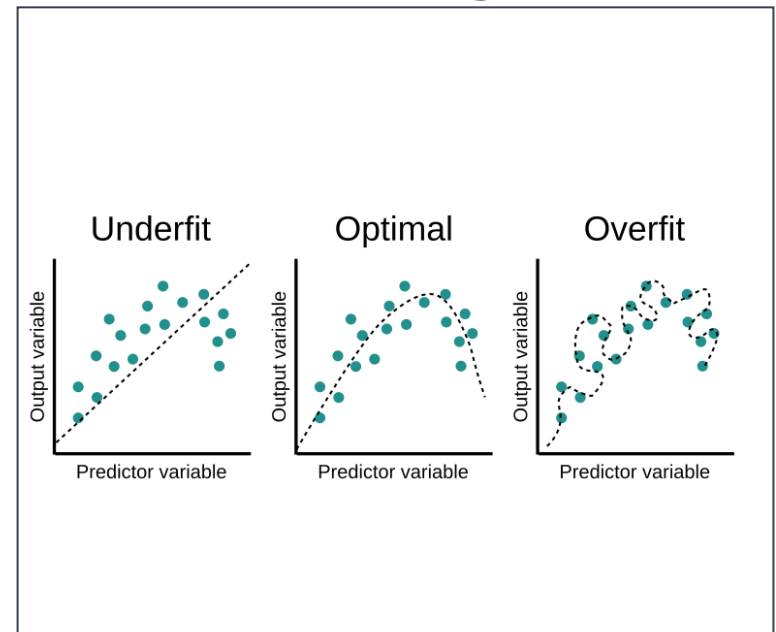
Daraus abgeleitete Metriken:

- F1 Score
- Precision
- Recall
- ROC/ AUC

## Regressionsmetriken

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

## Over- und Underfitting



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

## DETAILLIERUNG CONFUSION MATRIX AN BEISPIEL.



Welche False-Klasse ist schlimmer?

## 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.

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

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

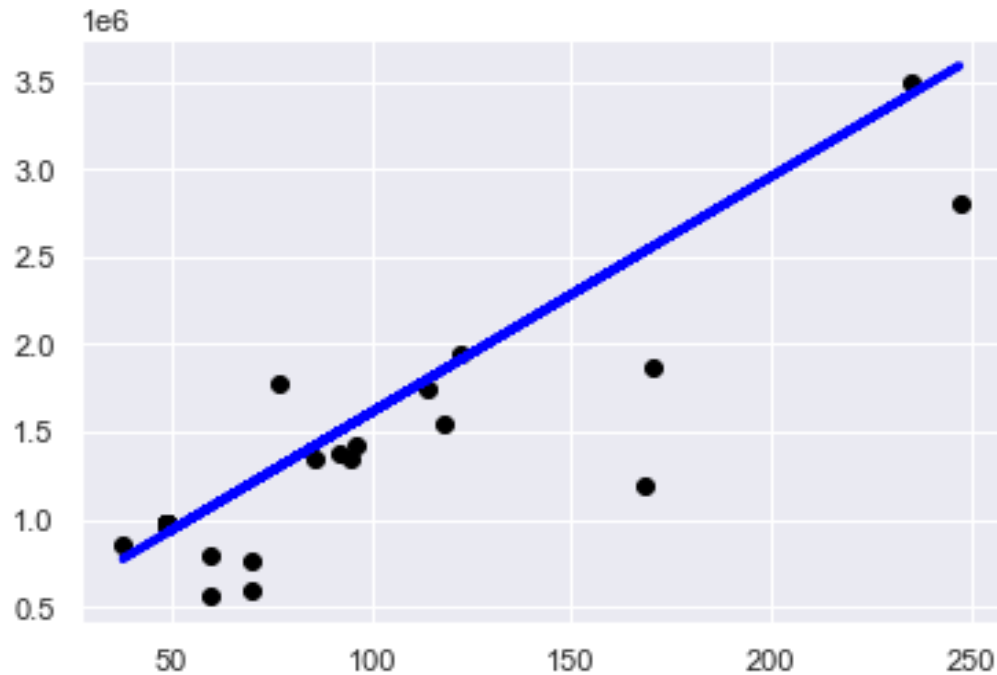
$$\sqrt{MSE}$$

- **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.

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



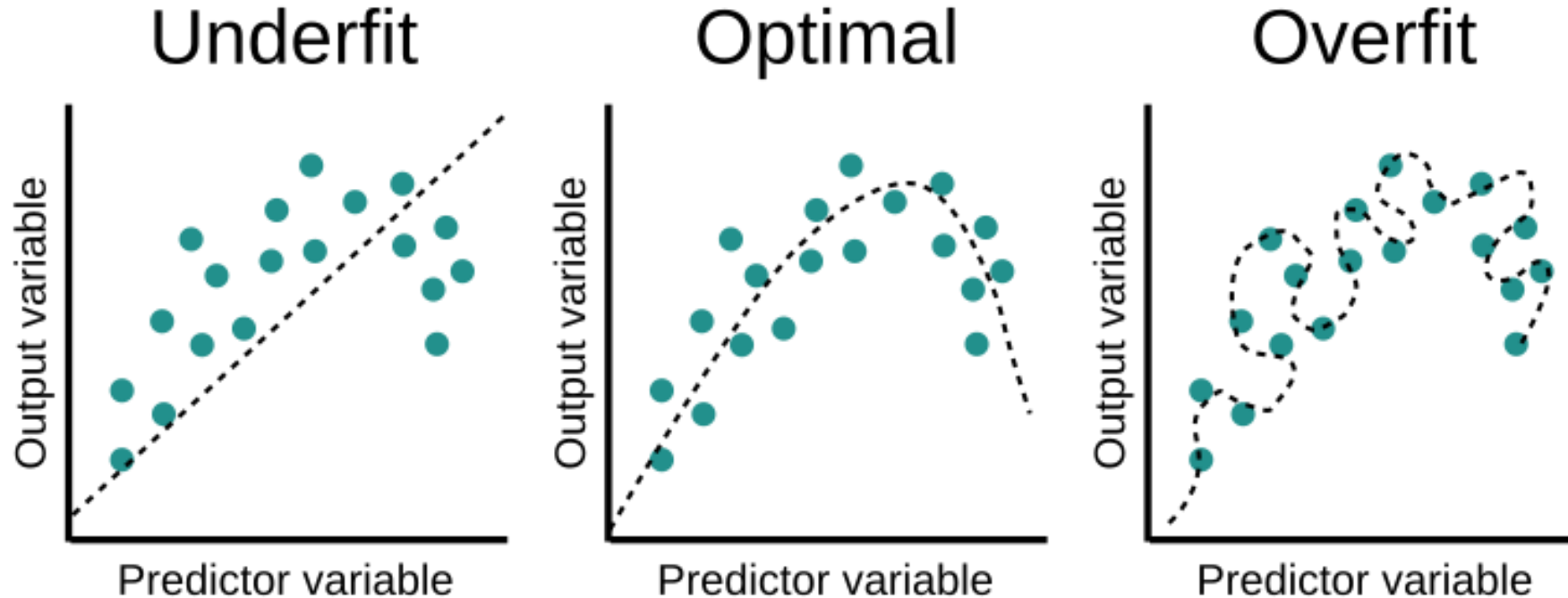
## DETAILLIERUNG METRIKEN: BEISPIEL REGRESSIONSMETRIKEN.



- 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).

## 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!



## 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](#))

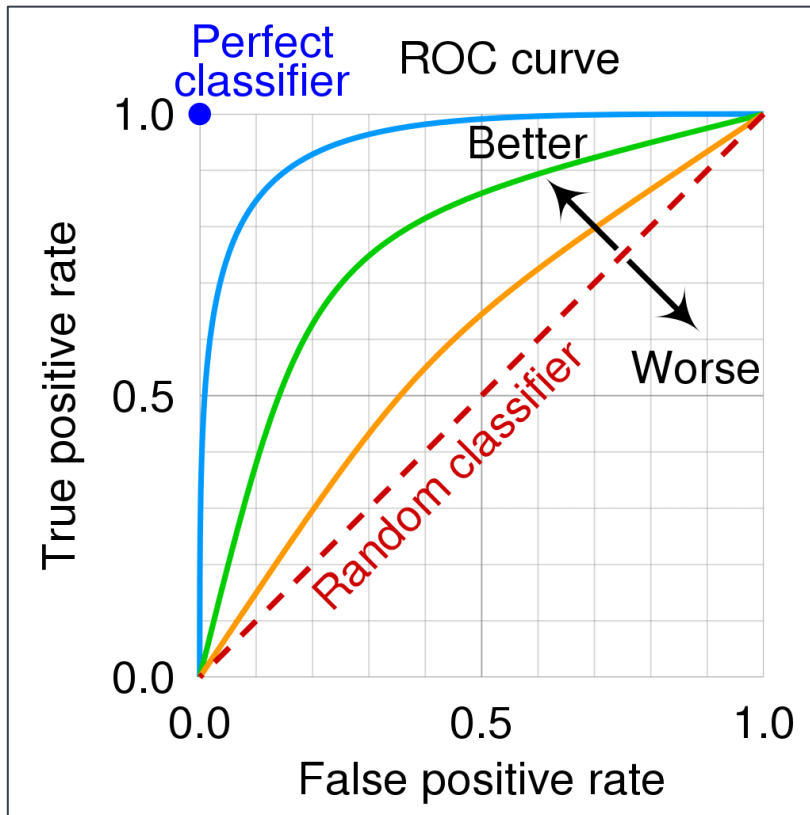
### Links:

- Workflow Fraud detection: [Link](#), [Artikel](#)
- Visualisierung Decision Tree: [Link](#), [Link](#)



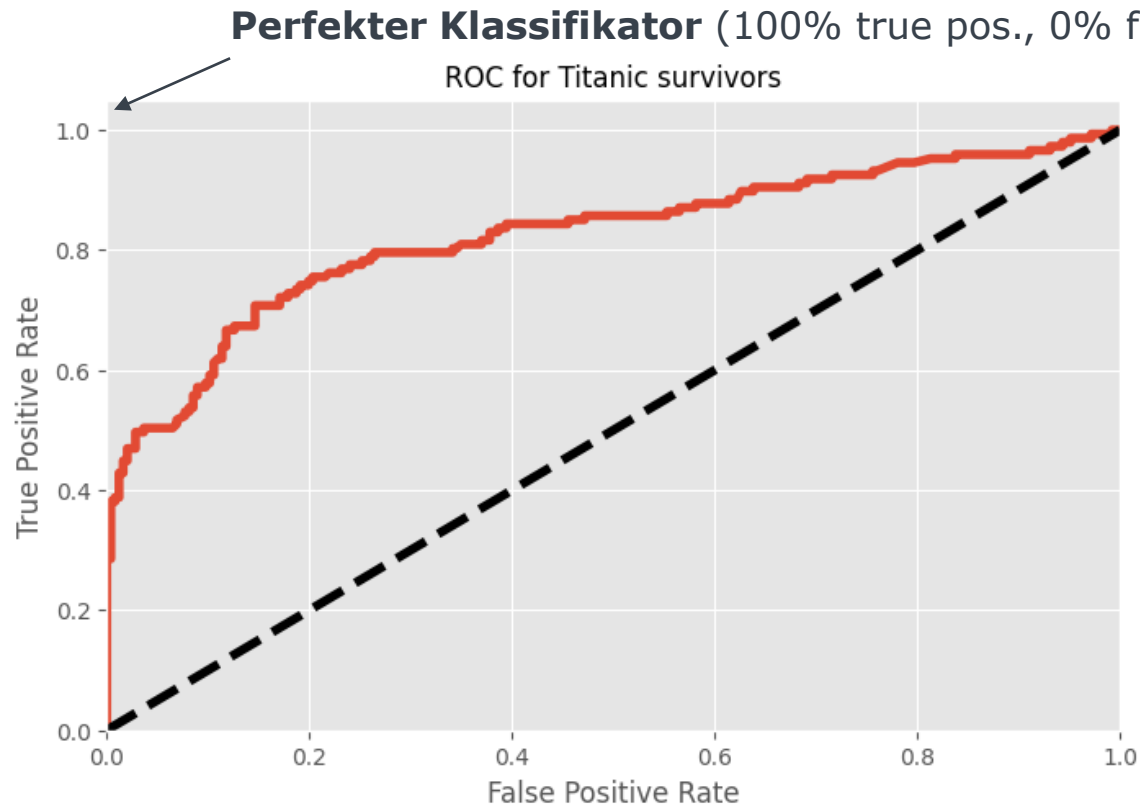
# BACKUP

## 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.

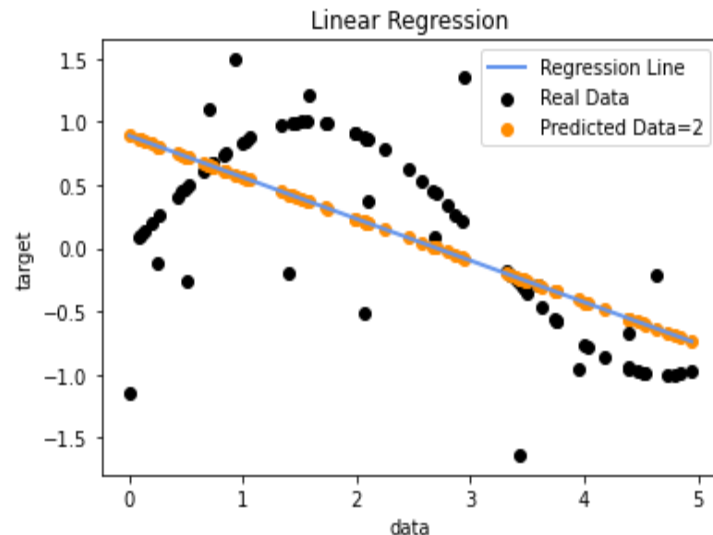
## DETAILLIERUNG KLASSIFIZIERUNGSMETRIKEN. ROC/ AUC.



- **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.

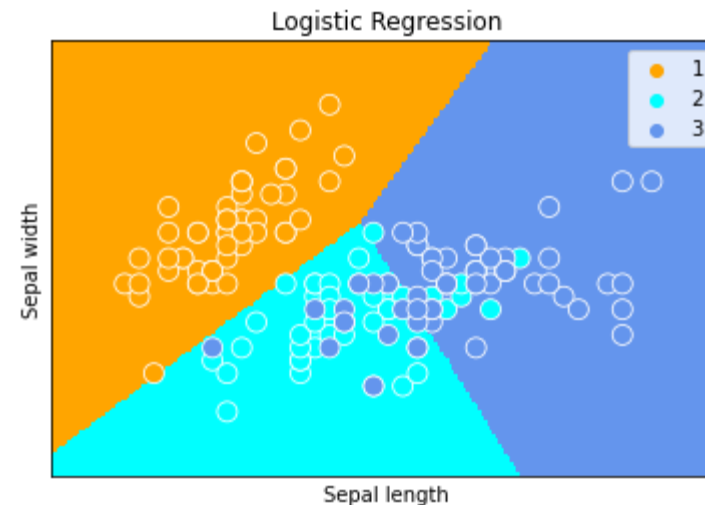
# SUPERVISED LEARNING: LINEARE/ LOGISTISCHE REGRESSION.

## 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.

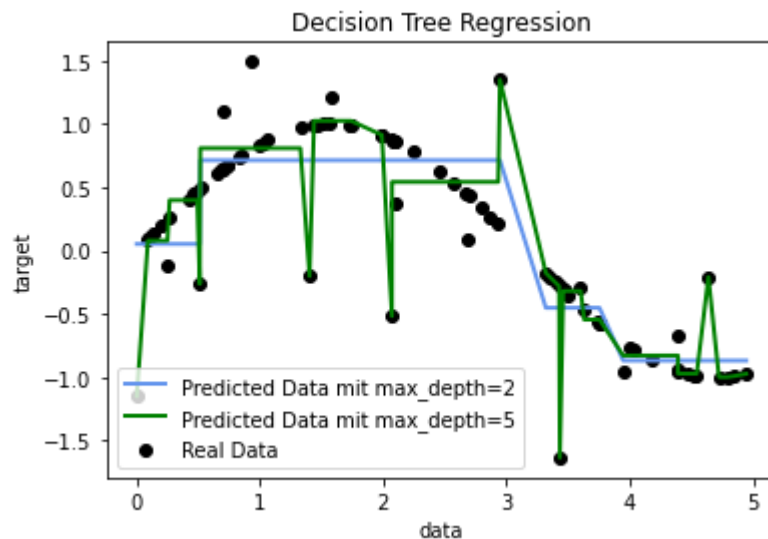
### Nachteile:

- 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.

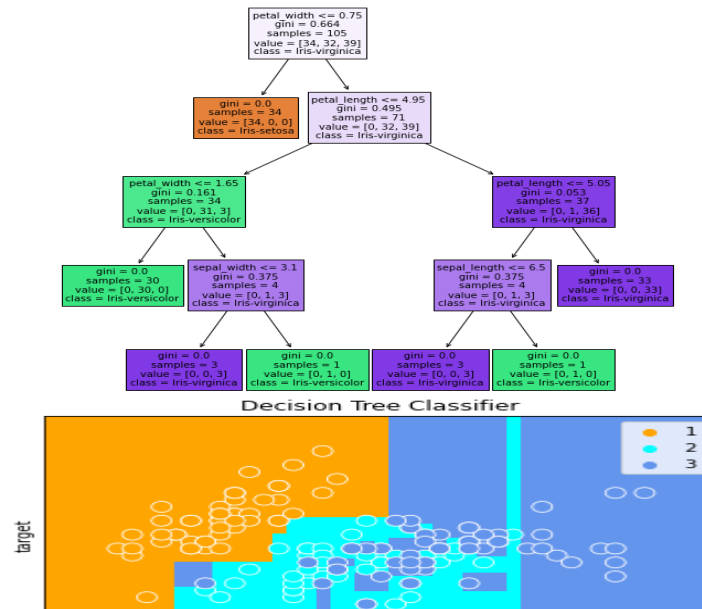
# SUPERVISED LEARNING: DECISION TREE.

## 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.

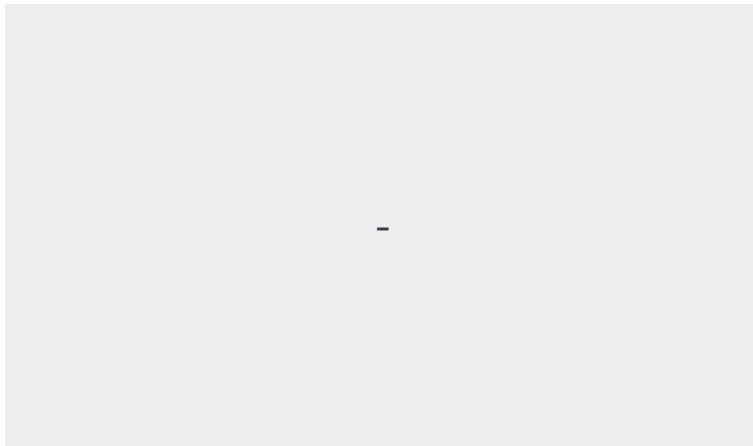
## Nachteile:

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

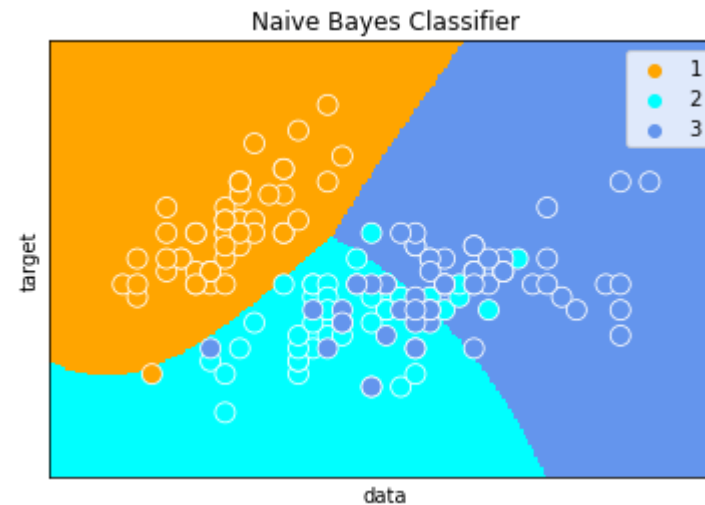
Klassifikation: wiederholtes Aufteilen Datenset in zwei Untermengen anhand des Features mit höchstem Informationsgehalt.  
Regression: erfolgt durch Bestimmen des Durchschnitts über die y-Werte des finalen Endknoten.

# 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.

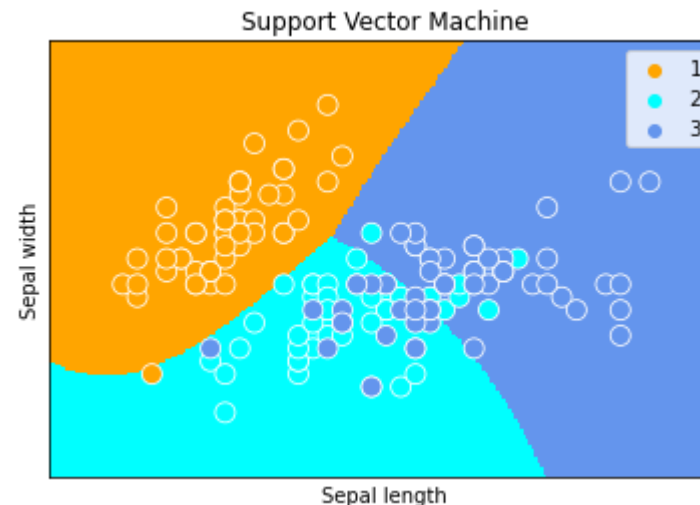
# SUPERVISED LEARNING: SUPPORT VECTOR MACHINE.

## 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:

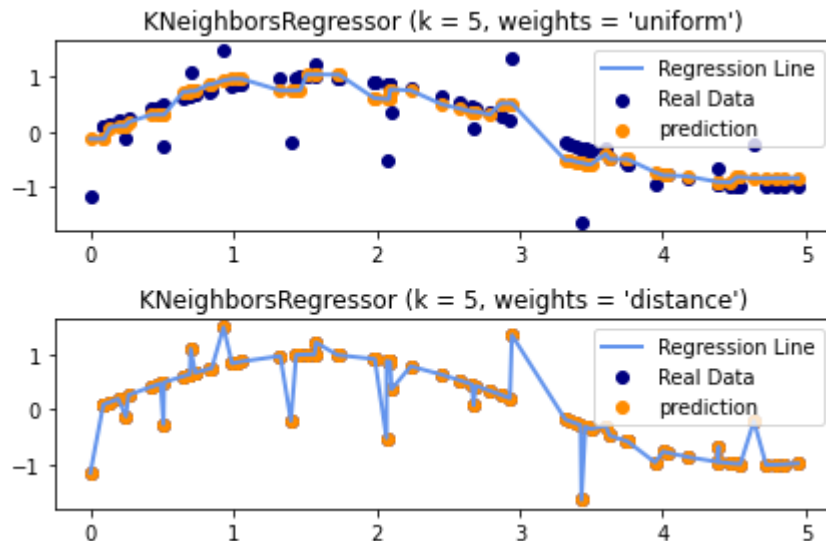
- Viele Hyperparameter, deshalb Tuning notwendig.
- Längere Trainingszeit bei großen Datenmengen.
- Erklärbarkeit Ergebnisse schwierig (Hyperebene n-dimensionaler Raum).

Klassifikation: Algorithmus bestimmt „Hyperebene“<sup>1</sup>, 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

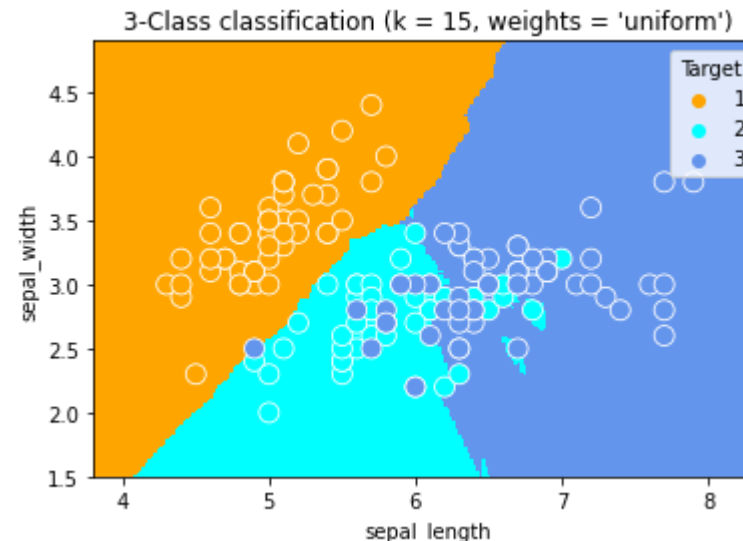
# SUPERVISED LEARNING: K NEAREST NEIGHBOURS.

## 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.





# 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.

