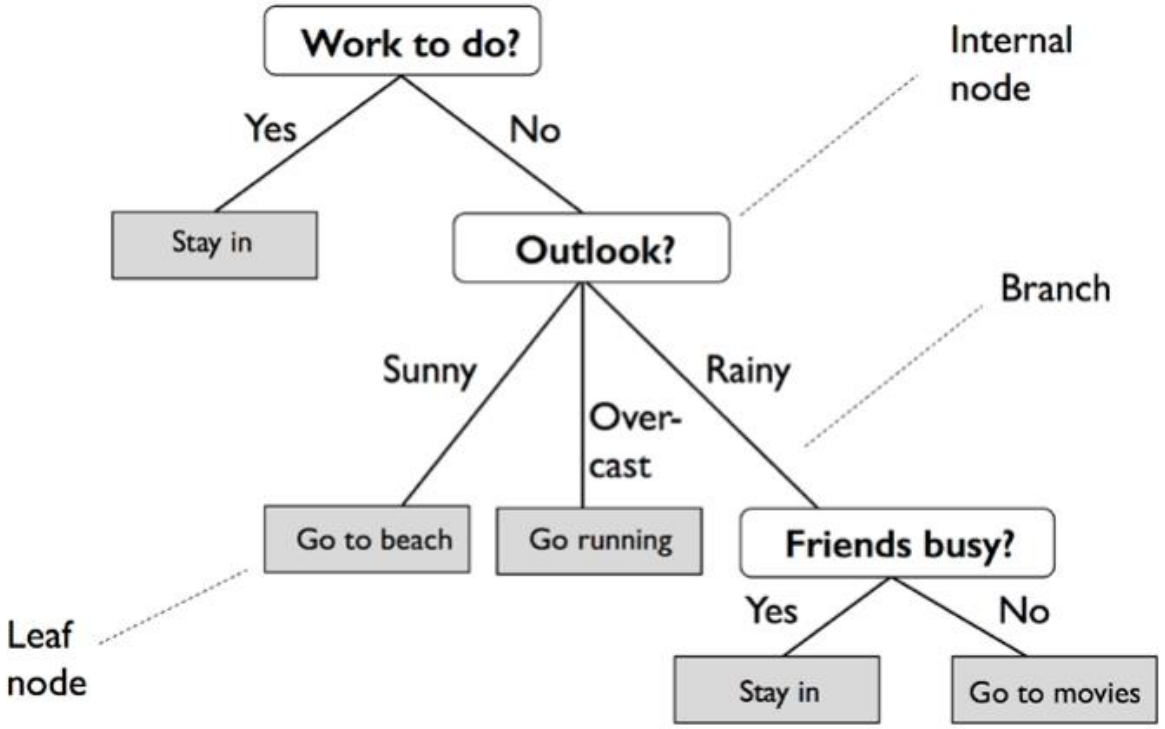
# Maschinelles Lernen

Vorlesung 2

# Lernen mit Entscheidungsbäumen

Sehr "sichere" Klassifikationsmodelle:

- Kleines Risiko an Overfitting
- Braucht kein Skalieren
- Robust gegenüber miteinander korrelierten Features
- Ergebnisse sind erklärbar




Quelle: Raschka (2019)

# Lernen mit Entscheidungsbäumen Informationsgewinn

**Definition:** Die Reinheit eines Knotens ist die Wahrscheinlichkeit dass, für einen gegebenen Knoten, alle zugehörigen Punkte zu einer Klasse gehören. Die Reinheit ist maximal, wenn in einem Knoten nur Punkte aus einer einzigen Klasse sind, und minimal wenn alle Klassen gleichmäßig vertreten sind.

Der Informationsgewinn ist der Unterschied zwischen der Reinheit des Vaterknoten und der beiden Kindknoten.

# Lernen mit Entscheidungsbäumen

Aufbauen der Bäume:

1. Split wo der größte Informationsgewinn ist (z.B. Falls a < 5, neuer Knoten links, sonst neuer Knoten rechts)
2. Wiederhole bis die maximale Tiefe des Baumes erreicht wird, oder bis alle Trainingsdaten im Knoten zu derselben Klasse gehören

Zu tiefe Bäume führen zu Overfitting: Hyperparameter für maximale Tiefe

# Lernen mit Entscheidungsbäumen

Verlustfunktion: man will den Informationsgewinn für jedes Split maximieren

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

IG – information gain
I(D) - Unreinheit eines Knotens
N – Anzahl der Punkte im Knoten

**Typische Definition für I:**
Die Summe der Wahrscheinlichkeiten, dass dass ein Punkt aus dem Knotenzu jeder der gegebenen Klassen gehöre

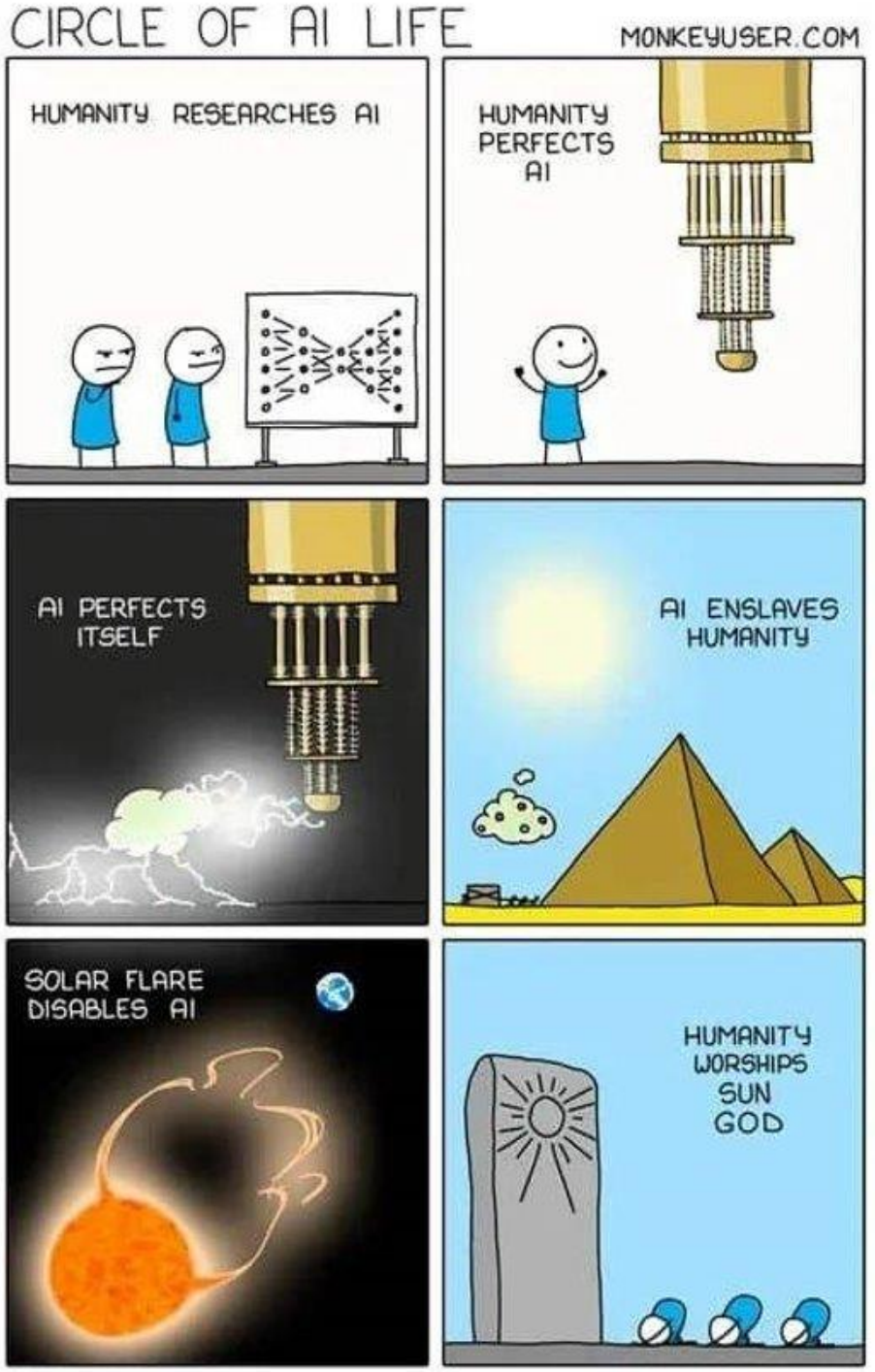
**Extreme Fälle:**

- Alle Punkte in einem Knoten gehören zu derselben Klasse
- Die Punkte sind gleichmäßig auf alle Klassen verteilt

# Lernen mit Entscheidungsbäumen Beispiel

Eine Telekomgesellschaft braucht eine Anwendung, die vorhersagen kann, welche Kunden ihr Abo im nächsten Monat kündigen, damit die Marketingdivision ihnen proaktiv Sonderangebote anbieten kann ("client churn").

- Was für Informationen könnte das Dataset enthalten?
- Welches ist die Zielvariable?
- Was für zusätzliche Features kann man bauen?
- Wie könnte man so ein AI langfristig einsetzen?
- Was bedeutet, für diesem Fall, Reinheit der Knoten, bzw. Informationsgewinn?



# Lernen mit Entscheidungsbäumen Beispiel

In Fällen wie diesem ist es sehr wichtig, dass man "feature / data leakage" vermeidet, d.h. dass man nicht Daten aus der Zukunft mitnimmt.

Beispiele von solchen Features für den Fall Customer Churn:

- Anrufe zum Kundendienst
- Verspätete Bezahlungen
- Ungewöhnliches Konsumverhalten

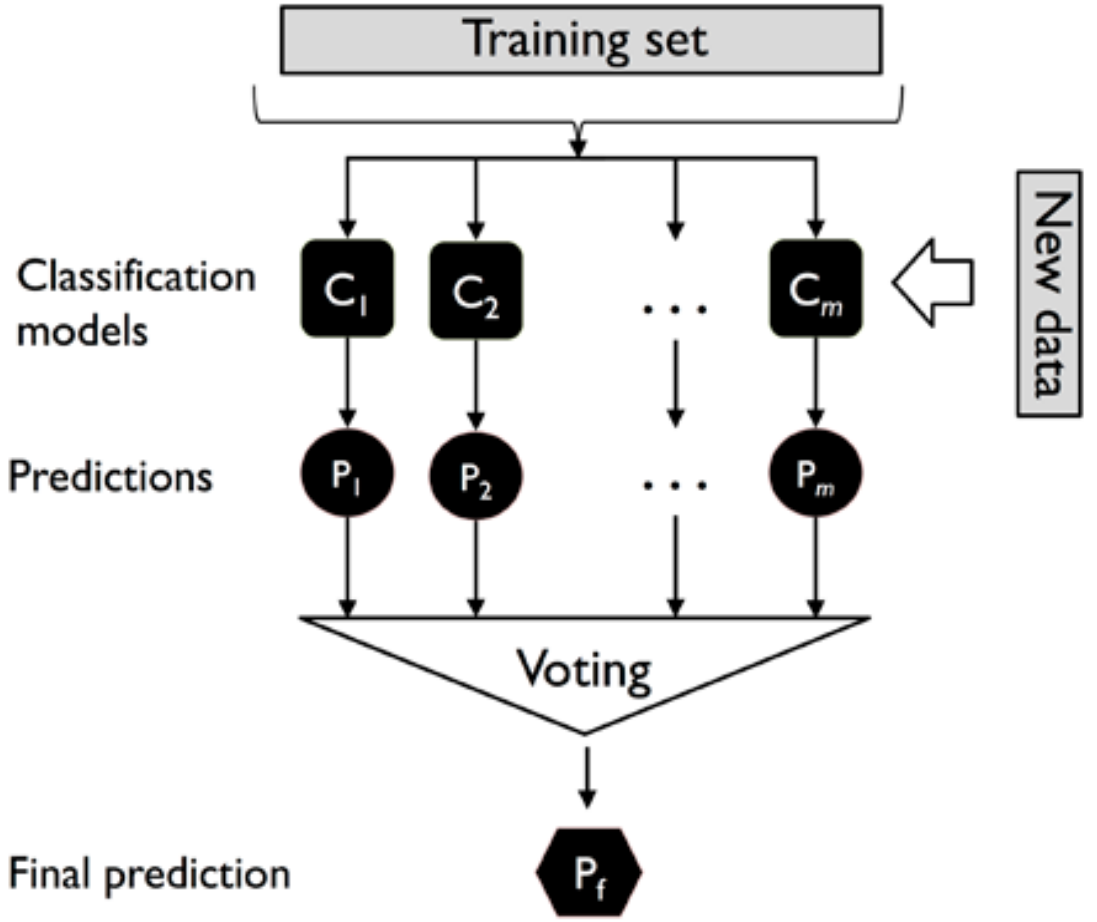
Lösung: Filtern nach Zeitstempel

# Lernen mit Entscheidungsbäumen Ensembles

- In der Praxis: Sammlungen von Bäumen

**Definition**: Ensemble: Kombinieren von verschiedenen Modellen in einem sogenannten Meta-Modell / Meta-Classifier

So werden oft bessere Ergebnisse erhalten

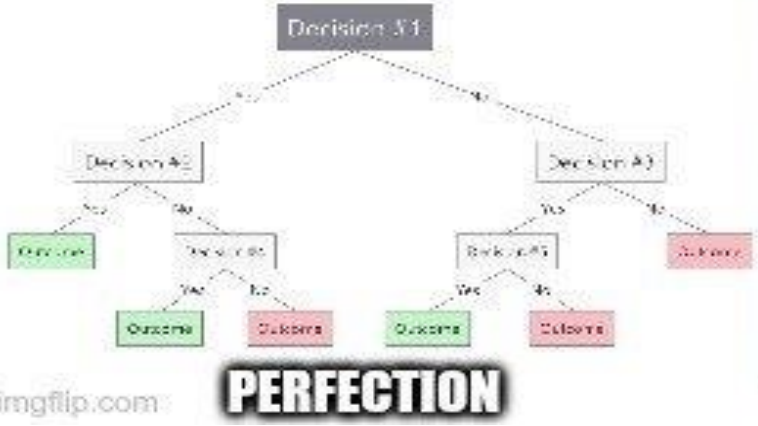



Quelle: Raschka (2019)

# Random Forest

- Ein "Wald" von Entscheidungsbäumen
- Die Bäume werden unabhängig voneinander gebaut
- Für jeden Baum benutzt man eine Untermenge der Daten, ein Punkt kann mehrmals gewählt werden ("bootstrapping")
- Ist eigentlich ein Ensemble von Entscheidungsbäumen
- Im klassischen Algorithmus: Endergebnis durch Mehrheitsstimme






# Ensemble-Methoden

Man kann alle Arten von Modellen zu einem Ensemble kombinieren

Ensembles sind oft aussagekräftiger als einzelne Modelle weil sich die Schwächen der individuellen Modelle ausgleichen

**Wichtige Annahme:** die Fehlerrate der einzelnen Modelle ist unabhängig voneinander und gleichmäßig verteilt. Falls diese Annahme nicht gilt (z.B. ein Modell hat ein systematisches Bias), dann sind die Ergebnisse nicht unbedingt vertrauenswert

# Ensemble-Methoden Metamodell

**Mehrheitsvotum**

Wenn die Mehrheit der Baseline-Modelle sagen, dass ein Punkt zu einer Klasse gehört

**Aggregierte Wahrscheinlichkeiten**

Für Modelle, die Wahrscheinlichkeiten zurückgeben, berechnet man ihren Mittelwert

**Meta-Classifier**

Main trainiert ein neues Modell mit den Ergebnissen der Modelle des Ensembles.

Am meisten in der Praxis benutzt: logistische Regression oder xgboost

# Ensemble-Methoden
# Bagging

**(von bootstrap aggregating)**

- Für jedes Modell nimmt man nur eine Untermenge aller Punkte zum Training, die Untermengen dürfen sich zum Teil überschneiden, z.B. jedes Modell benutzt 70% der Daten
- Man kann dasselbe Modell mehrmals nehmen

**Typische use cases:**

- Sehr viele Features, viele Trainingsdaten
- Instabile Daten, d.h. komplexe Verhältnisse zwischen den Features
- Sehr hohes Risiko an overfitting

# Boosted Trees
# (xgboost, lightgbm, catboost, adaboost)

- Ein einziger Baum hat eine schwache Vorhersagekraft
- Jeder Baum wird sukzessive aufgebaut (→ Definition des Begriffs Boosting)
- Man will von einem Baum zum nächsten die Fehlerrate minimieren → dies ist unsere neue Verlustfunktion
- Nach jedem Schritt enthalten die Knoten den Unreinheitsgrad
- Unterschied zwischen den Methoden: Aufbau der Verlustfunktion, genaue Implementierung der Optimumsuche

# Boosted Trees Hyperparameter (Auszug)

- `n_estimators`
- `learning_rate`
- `max_depth`
- `subsample`
- `colsample_bytree`
- `min_samples_leaf`

Die meisten Hyperparameter dienen in der Praxis dazu, Flexibilität für unbalancierte Datensätze antzbieten, und die Wahrscheinlichkeit an Overfitting zu verkleinern

Namen der Hyperparameter kann zwischen Implementierungen variieren

# Regularisierung

**Definitionen:**

Wenn man das Modell mehrmals trainiert, mit verschiedenen Untermengen des gesamten Datensatzes…

- Bias: wie weit weg ist man vom richtigen, erwarteten Wert?
- Variance: um wie viel variieren die Ergebnisse des Trainings?

Wie kann man mit diesen beiden Metriken Over- und Underfitting beschreiben?

In der Praxis: k fold cross validation

# Regularisierung

= Menge von Techniken, welche Over- und Underfitting minimieren

- In der Praxis: zusätzliches Glied bei der Verlustfunktion
- Kann bei allen Modellen eingeführt werden

$$L_{reg}(\theta) = L(\theta) + \lambda R(\theta)$$

$\lambda$: der Regularisierungsparameter, gibt die Größe der Funktion an

$R$: die Regularisierungsfunktion

# Regularisierung

Am meisten benutzt: *L1* und *L2* Regularisierung

Hyperparameter der Modelle

- L1: nützlich wenn man sehr viele Parameter hat und die meisten [fast] kein Einfluss auf die Prädiktion haben $\|\boldsymbol{w}\|_1 = \sum_{j=1}^{m} |w_j|$
- L2: in der Praxis am meisten benutzt, sehr große Parameter werden minimiert $\|\boldsymbol{w}\|_2^2 = \sum_{j=1}^{m} w_j^2$

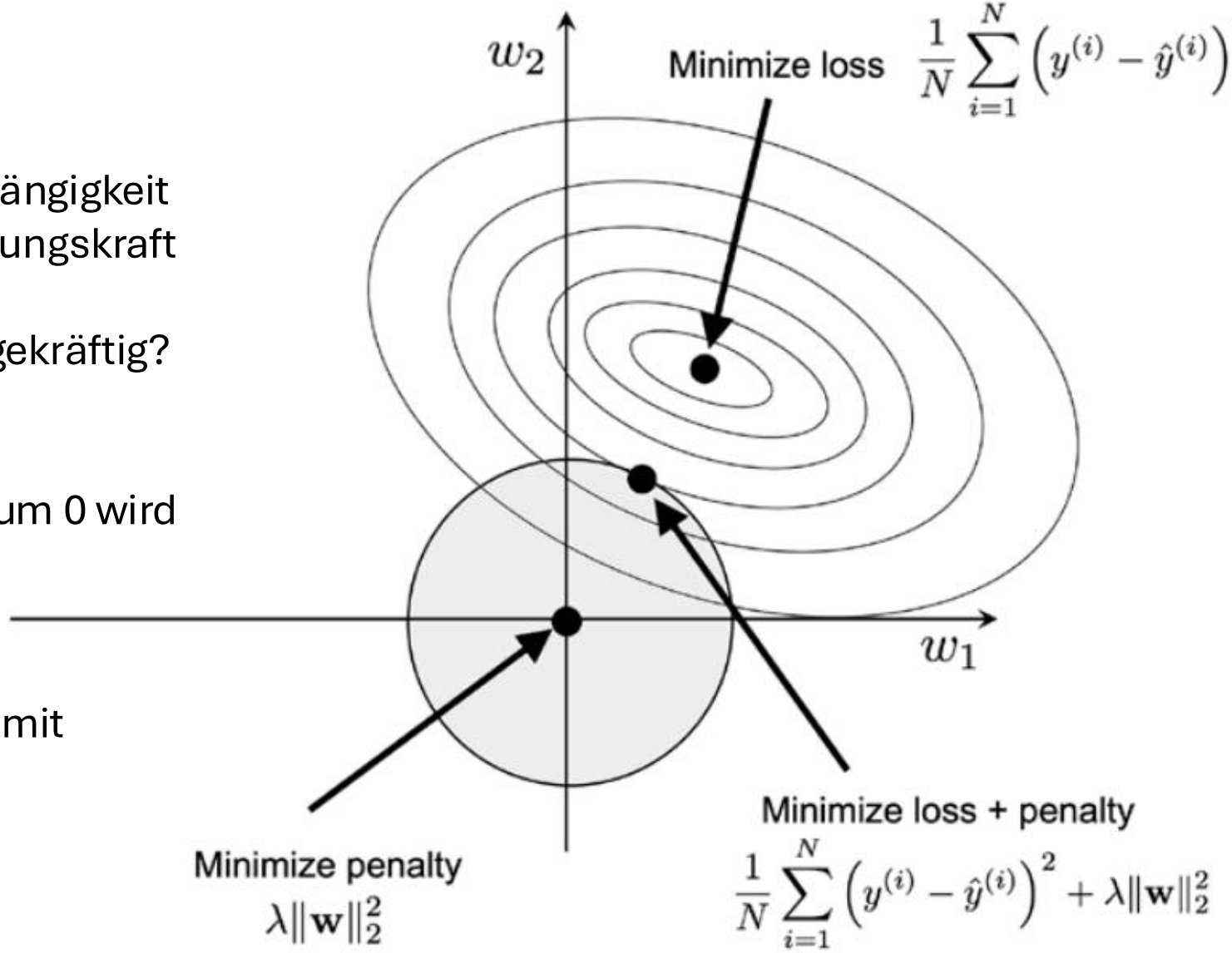
# Regularisierung

Geometrische Intuition: Weniger Abhängigkeit vom Datensatz, mehr Verallgemeinerungskraft

Ist der Datensatz vollständig? Aussagekräftig? Ohne systematischem Bias?

Großes λ: weniger Freiraum, der Ball um 0 wird kleiner

In sklearn: ridge_regression, bzw. lasso_regression: lineare Regression mit




Quelle: Raschka (2019)

# Lernen mit Entscheidungsbäumen
# Erklärbarkeit

Man kann berechnen, wie oben in den Bäumen nach welchen Features geteilt wird, und wie aussagekräftig ein Split ist

Intuitiv:

- Was bedeutet, dass ein Feature wichtiger als ein anderer ist?
- Was ändert sich, wenn man an Ensembles von Bäumen denkt?

# Bewertung der Modelle

Die Konfusionsmatrix

| Actual Class \ Predicted Class | Positive | Negative | |
|---|---|---|---|
| **Positive** | True Positive (TP) | False Negative (FN) **Type II Error** | **Sensitivity** $\frac{TP}{(TP+FN)}$ |
| **Negative** | False Positive (FP) **Type I Error** | True Negative (TN) | **Specificity** $\frac{TN}{(TN+FP)}$ |
| | **Precision** $\frac{TP}{(TP+FP)}$ | **Negative Predictive Value** $\frac{TN}{(TN+FN)}$ | **Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$ |


$$\text{Precision} = \frac{\text{5 true pos.}}{\text{8 total pos.}} \qquad \text{Recall} = \frac{\text{5 true pos.}}{\text{12 total dogs}}$$

$$\text{Prevalence} = \frac{\text{12 total dogs}}{\text{22 total images}}$$

$$\text{Accuracy} = \frac{\text{5 true pos. + 7 true neg.}}{\text{22 total images}}$$

# Bewertung der Modelle

Accuracy: nicht aussagekräftig.

Z.B. ein binäres Klassifikationsalgorithmus, in welchem eine Klasse in 98% der Fälle vorkommt "lernt" immer die größere Klasse vorherzusagen und hat so 98% accuracy

F1: das harmonische Mittel von precision und recall

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

- Alle besprochene Metriken: zwischen 0 und 1
- Zweck der Anpassung von Hyperparametern: bessere Metriken
- Für komplexe Aufgaben kann man eigene Metriken definieren

# Bewertung der Modelle

**Zum Nachdenken:**

1. Wenn man nicht sowohl precision, als auch recall optimieren kann, was ist wichtiger? Denke an konkrete Beispiele!

2. Wie könnte man diese Metriken und die Konfusionsmatrix auf Multiclass-Classifier verallgemeinern?

# Bewertung der Modelle

Die Holdout-Methode: Beim Training eines Modells benutzt man nur ein Teil vom gesamten Datensatz, typisch 70-90% → Trainingsdaten

Der Rest → Testdaten, werden für die Bewertung des trainierten Modells benutzt. Kann das Modell gut auf nicht bekannte Daten verallgemeinern?

Der Trainings- und Testdatensatz müssen die gleichen statistischen Eigenschaften haben

Stratification: behält für jedes Fold die Verteilung der Zielvariablen