V12 Klassifikation (Machine Learning 001)

- Entscheidungsbäume (decision tree): ID3-Algorithmus
- Support Vector-Maschinen (SVM)
- Bewertung von Klassifizierungs-Ergebnissen

Grobe Unterteilung:

- Methoden für unsupervised classification von nicht-gelabelten Daten -> vor allem Clustering, siehe V3 – Folie 27, V8 – Folie 30 und k-means Clustering
- Methode für supervised classification von gelabelten Daten (HEUTE)

Ereignismenge: Beispiel

Gegeben sei eine Menge an Beobachtungen, z.B. ob ich in der Vergangenheit bei bestimmten äußeren Bedingungen zum Tennisspielen oder Golfspielen gegangen bin.

Wetter	Temperatur	Luftfeuchtigkeit	Wind	Gespielt?
Sonnig	Heiß	Hoch	Schwach	Nein
Sonnig	Heiß	Hoch	Stark	Nein
Bewölkt	Heiß	Hoch	Schwach	Ja
Regen	Mild	Hoch	Schwach	Ja
Regen	Kühl	Normal	Schwach	Ja
Regen	Kühl	Normal	Stark	Nein
Bewölkt	Kühl	Normal	Stark	Ja
Sonnig	Mild	Hoch	Schwach	Nein
Sonnig	Kühl	Normal	Schwach	Ja
Regen	Mild	Normal	Schwach	Ja
Sonnig	Mild	Normal	Stark	Ja
Bewölkt	Mild	Hoch	Stark	Ja
Bewölkt	Heiß	Normal	Schwach	Ja
Regen	Mild	Hoch	Stark	Nein

Angelehnt an:

http://www.coli.uni-saarland.de/courses/mathe3/SS12/Vorlesungen/decisiontree.pdf

Entscheidungsbäume: Prinzip

Wir möchten nun einen Entscheidungsbaum konstruieren, mit dem man diese Entscheidungen formalisieren kann.

D.h. wenn es heiß und feucht ist, dann spiele ich nicht

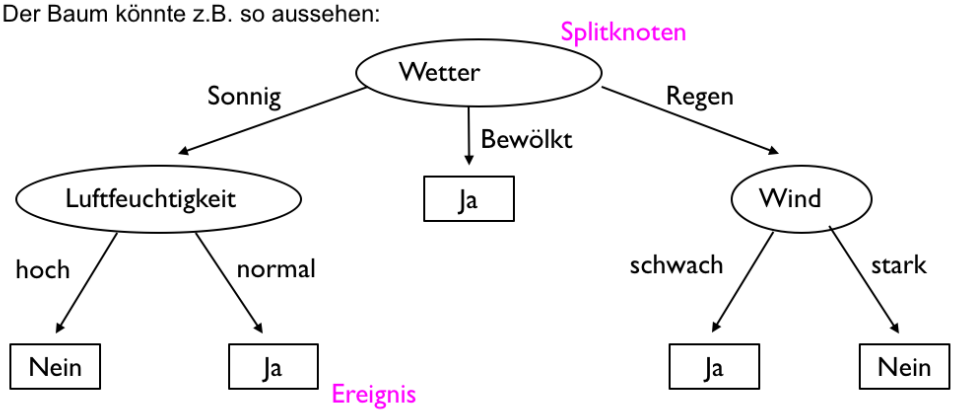
Wenn es kühl ist und starker Wind weht, dann spiele ich nicht.

Wenn es mild ist, die Sonne scheint und der Wind schwach ist, dann spiele ich etc.

Wie baut man den besten Baum?

Entscheidungsbäume

Der Baum könnte z.B. so aussehen:



- Beschreibt dieser Baum korrekt die existierenden Beobachtungen?
- Gibt es vielleicht einen besseren bzw. einfacheren Baum? Zum Beispiel haben wir die Temperatur hier ja gar nicht berücksichtigt.

Konstruiere Split-Knoten

Der von Russ Quinlan entwickelte **ID3-Algorithmus**

ist ein sogenannter *greedy* (gieriger) Algorithmus.

Quinlan, J. R. 1986. Induction of Decision Trees.

Mach. Learn. 1, 1 (Mar. 1986), 81–106

Russ Quinlan
https://www.rulequest.com/Personal/

So ein Algorithmus trifft jeweils die lokal bestmögliche Entscheidung.

Wenn eine Entscheidung getroffen wurde, wird diese später nicht mehr revidiert.

Man packt alle Beobachtungen in einen Wurzelknoten und sucht nun das beste Kriterium (d.h. den obersten Splitknoten), die Beobachtungen aufzuteilen.

Informationsgewinn

Ziel: Wir möchten die Entropie in den verbleibenden Klassen möglichst reduzieren.

Für jedes Attribut betrachten wir die Differenz zwischen der vorhandenen Entropie und der verbleibenden Entropie nach Einführung eines Splitknotens bzgl. dieses Attributs:

$$Zugewinn(S, A) = H(S) - \sum_{v=1}^{k} \frac{|S_v|}{|S|} H(S_v)$$

S: Datensatz

A: Attribut

H(S) Entropie im Datensatz oder Untermenge davon

S_v Untermenge von S, für die A den Wert *v* hat

|S| Mächtigkeit von S

$|S_v|$ Mächtigkeit von S_v

Entropie im Datensatz

Zur Berechnung der Entropie benutzen wir die übliche Shannon-Entropie

$$H(S) = -\sum_{c=1}^{|C|} p_c \log_2 p_c$$

S: Datensatz

|C| Anzahl an Kategorien (in diesem Beispiel: 2 oder 3)

p_c Anteil der Instanzen, die Kategorie *c* angehören

Die Entropie ist am höchsten, wenn die Häufigkeit p_c in allen Kategorien gleich ist.

Die Entropie ist gleich Null, wenn die Häufigkeit in einer Kategorie gleich 1 ist. Warum?

Beispiel für den Wetterdatensatz

Wir berechnen jetzt, welches Attribut an der Wurzel (d.h. dem obersten Splitknoten) in unserem Beispielproblem den größten Informationsgewinn bringen würde:

$$Zugewinn(S, A) = H(S) - \sum_{v=1}^{k} \frac{|S_v|}{|S|} H(S_v)$$

Die Gesamtentropie H(S) ist – mit 14 Instanzen (Ereignissen), davon 5 x Nein, 9 x Ja -

$$H(S) = -\frac{9}{14} log_2 \frac{9}{14} - \frac{5}{14} log_2 \frac{5}{14} = 0.940286$$

Für das Attribut Wind = „schwach“ gab es 8 Instanzen - 6 mal wurde gespielt, 2 mal nicht

$$H(S_{schwach}) = -\frac{6}{8} log_2 \frac{6}{8} - \frac{2}{8} log_2 \frac{2}{8} = 0.8112781$$

Für das Attribut Wind = „stark“ gab es 6 Instanzen - 3 mal wurde gespielt, 3 mal nicht

$$H(S_{stark}) = -\frac{3}{6} log_2 \frac{3}{6} - \frac{3}{6} log_2 \frac{3}{6} = 1$$

Beispiel für den Wetterdatensatz

$$Gain(S, A) = H(S) - \sum_{v=1}^{k} \frac{|S_v|}{|S|} H(S_v)$$

Aus der Gesamtentropie und der Entropie für die beiden Ereignisse Wind = schwach / stark ergibt sich

$$Gain(S, Wind) = H(S) - \frac{|S_v|}{|S|} H(S_v) - \frac{|S_v|}{|S|} H(S_v) = 0.940286 - \frac{8}{14} \cdot 0.8112781 - \frac{6}{14} \cdot 1 = 0.04812709$$

Auf ähnliche Weise bekommt man

Gain(S, Wetter) = 0.247

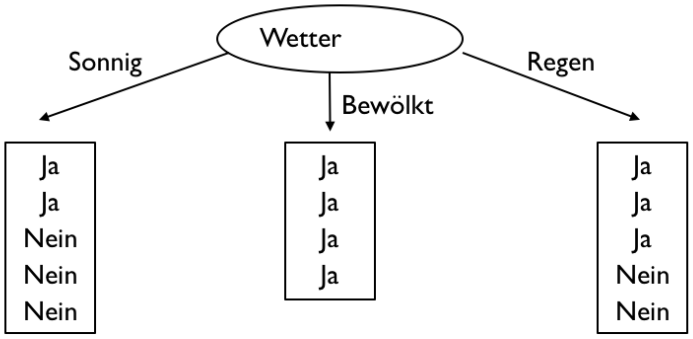
Gain(S, Temperatur) = 0.029

Gain(S, Luftfeuchtigkeit) = 0.152

„Wetter“ bringt also den größten Informationsgewinn und wird als Baumwurzel gewählt.

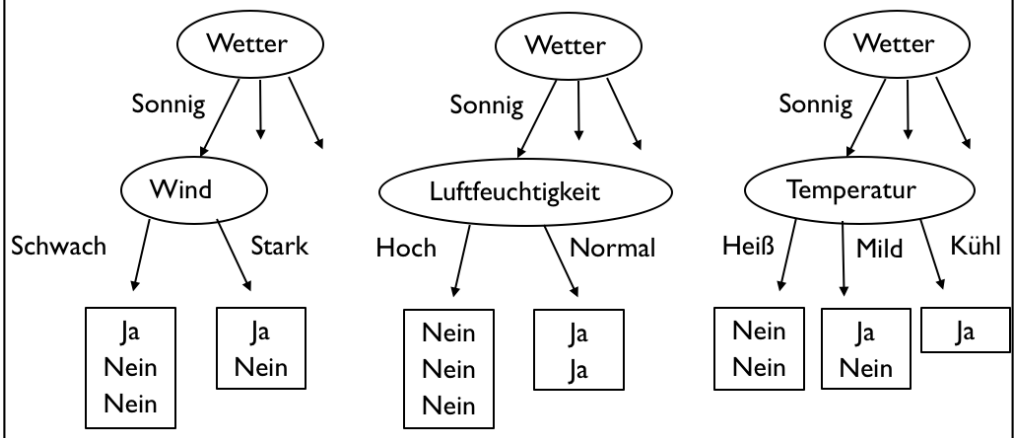
Entscheidungsbäume

Der Baum nach Einführung des Splitknotens „Wetter“ sieht folgendermaßen aus:



Im nächsten Schritt muss man für jeden der 3 Attributwerte, z.B. „sonnig“, dasselbe Verfahren für die Untermenge an Ereignissen rekursiv noch einmal anwenden.

Entscheidungsbaum Wetter - Rekursion



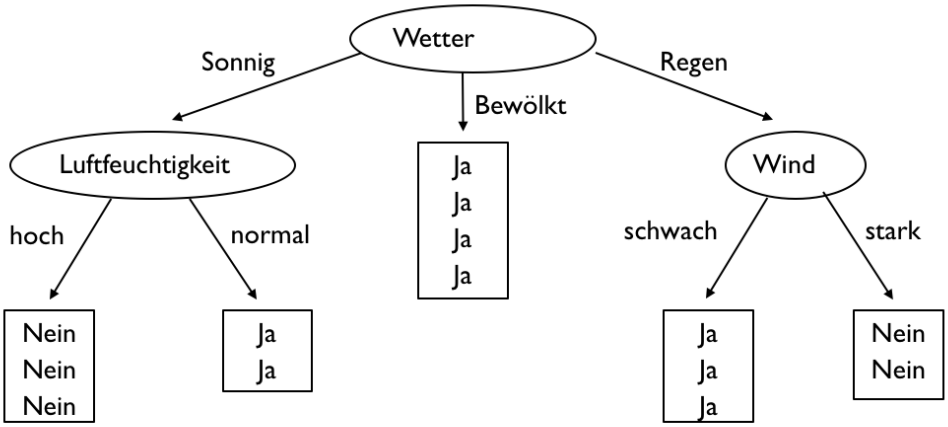
Gain(S, Wind) = 0.020

Gain(S, Temperatur) = 0.571

Gain(S, Luftfeuchtigkeit) = 0.971

Endergebnis – ID3-Algorithmus

Damit ergibt sich folgender Baum:



- In diesem Baum werden alle Instanzen unter einem Blatt jeweils gleich klassifiziert (Entropie = 0)
- Das Attribut „Temperatur“ wurde gar nicht benutzt.

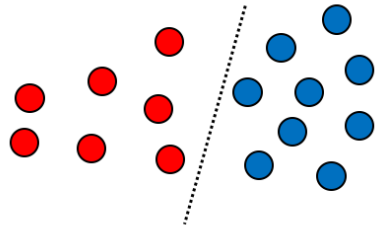
Eigenschaften des ID3-Algorithmus

Der Algorithmus ist „greedy“, d.h. unvollständig. Es werden nicht alle möglichen Bäume betrachtet.

Es ist im Prinzip möglich, dass ein „besserer“ (d.h. kleinerer) Baum nicht gefunden wird. Dieser könnte als erste Wurzel ein Attribut benutzen, das anfangs nicht den höchsten Informationsgewinn bietet.

Support Vektor Maschinen

Gegeben sei wiederum eine gelabelte Datenwolke (rot/blau)



Wir möchten gerne eine Trennlinie konstruieren, die wir verwenden können, um weitere Datenpunkte in rot/blau zu klassifizieren.

Wir könnten „einfach“ eine Gerade dazwischen legen, etwa die gestrichelte Linie.

Präsentation angelehnt an https://www.tu-chemnitz.de/urz/ittime/documents/Vortrag_Jens_Poenisch_SVM.pdf

SVM – Konstruktion der Trennebene

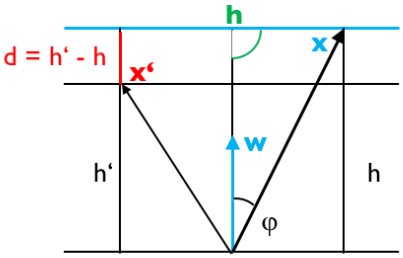
Bezeichnungen:

$\mathbf{x} = (x_1, \ldots, x_n)^T$ Vektor mit *n* Komponenten

$\langle \mathbf{x}, \mathbf{y} \rangle = x_1 y_1 + \cdots + x_n y_n$ Skalarprodukt der Vektoren **x** und **y**

$\|\mathbf{x}\| = \sqrt{x_1^2 + \cdots + x_n^2}$ Euklidische Norm (= Länge) des Vektors **x**

Abstand eines Punktes von der Ebene



w ist ein Vektor senkrecht zur blauen Ebene,

x ist der Vektor zu einem beliebigen Punkt auf der blauen Ebene,

φ ist der Winkel zwischen **w** und **x**.

Abstand der Ebene vom Ursprung $h = \|\mathbf{x}\| cos\varphi$

Skalarprodukt $\langle \mathbf{w}, \mathbf{x} \rangle = \|\mathbf{w}\| \cdot \|\mathbf{x}\| \cdot cos\varphi$

Damit gilt $h = \frac{\langle \mathbf{w},\mathbf{x} \rangle}{\|\mathbf{w}\|}$

Wir setzen $b := -h \cdot \|\mathbf{w}\|$

$h = \frac{\langle \mathbf{w},\mathbf{x} \rangle}{\|\mathbf{w}\|}$ formen wir um in

$\frac{\langle \mathbf{w},\mathbf{x} \rangle}{\|\mathbf{w}\|} - h = 0$ bzw. in die Ebenengleichung $\frac{\langle \mathbf{w},\mathbf{x} \rangle + b}{\|\mathbf{w}\|} = 0$

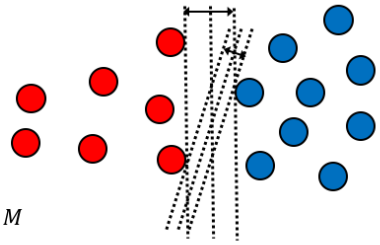
Der Punkt x‘ liegt auf einer Parallelebene mit Abstand $h' = \frac{\langle \mathbf{w},x' \rangle}{\|\mathbf{w}\|}$ vom Ursprung, also gilt für den Abstand von der Ebene

$$d = h' - h = h' + \frac{b}{\|\mathbf{w}\|} = \frac{\langle \mathbf{w},\mathbf{x}' \rangle + b}{\|\mathbf{w}\|}$$

SVM – Grundidee

Ziel: lege die Hyperebene (in zwei Dimensionen ist dies einfach eine Gerade) so, dass der kleinste Abstand eines Punktes zur Ebene möglichst groß ist und auf der „richtigen“ Seite/Klasse liegt.

$$t\left(\frac{\langle \mathbf{w}, \mathbf{x}' \rangle + b}{\|\mathbf{w}\|}\right) \rightarrow \max, \mathrm{t} \in \{-1, +1\}$$



Man kann dies umformen in

$$\arg \min_{\mathbf{w}, b} \frac{1}{2}\|\mathbf{w}\|^2, \text{ wobei } t_m(\langle \mathbf{w}, \mathbf{x}_m \rangle + b) \geq 1, m = 1, \ldots, M$$

- Man möchte einen möglichst breiten «Streifen» unterhalb und oberhalb der Ebene erhalten, der frei von Trainingspunkten ist.

- Die Punkte auf dem «Rand» heißen **Supportvektoren**.

- Neue Punkte werden durch diese Trennebene hoffentlich «richtig» klassifiziert.

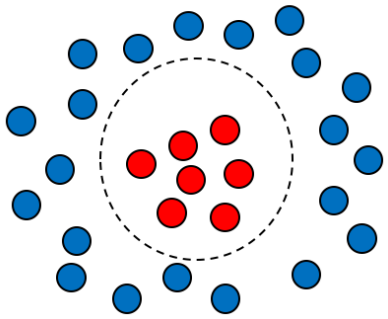
- Wähle die Ebene so, dass der kleinste Abstand eines Punktes zur ihr maximiert wird.

Problem 1: keine lineare Trennung möglich

Es könnten jedoch auch Fälle auftreten, wie der rechts gezeigte, bei dem sich die beiden Gruppen von Datenpunkten offensichtlich nicht durch eine Gerade trennen lassen.

In diesem Fall wäre ein Kreis um die roten Punkte am besten geeignet.

Allgemein überprüft man meist, ob sich die Datenpunkte nach der Transformation mit einer Kernelfunktion (typisch: radialer Gauss-Kernel, polynomisch, sigmoidal) besser in 2 Klassen separieren lassen.

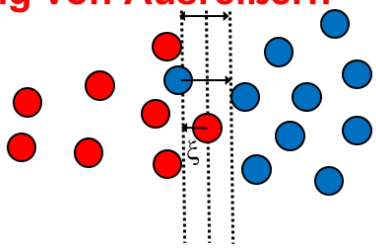


Gauß-Kernel

$$k(\mathbf{x}_i, \mathbf{x}_k) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_k\|^2}$$

Problem 2: Behandlung von Ausreißern

Durch «Ausreißer» liegen einzelne Datenpunkte in der «falschen» Klasse, es ist keine saubere Trennung möglich.



Erlaube einzelne Ausreißer durch eine „Slackvariable“ ξ. Man addiert einen Korrekturwert auf die eigentlich verletzte Randbedingung, um diese zu erfüllen.

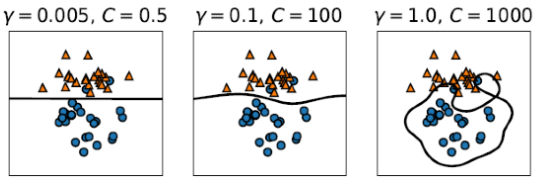
Die Summe der Korrekturen soll möglichst klein sein.

$$\arg\min_{\mathbf{w}, b, \xi_1, \ldots, \xi_M} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{m=1}^{M} \xi_m \right\}$$

wobei $t_m(\langle \mathbf{w}, \mathbf{x}_m \rangle + b) + \xi_m \geq 1, \xi_m \geq 0, m = 1, \ldots, M$

Beim Trainieren der SVM variiert man den „Regularisierungsparameter“ C, also den Einfluss von ξ.

Je größer C, desto komplexer das Modell (höhere «Bestrafung» der Ausreißer).



https://www.tu-chemnitz.de/urz/ittime/documents/Vortrag_Jens_Poenisch_SVM.pdf

Strategie bei Machine Learning

(1) Eigenschaft festlegen, die vorhergesagt werden soll

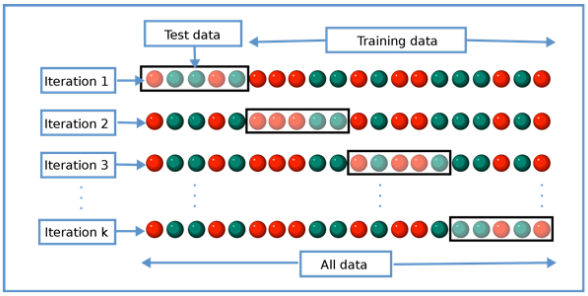
(2) Auswahl eines geeigneten **Datensatzes** (Gold-Standard), der eine ausreichende Anzahl an positiv gelabelten als auch an negativ gelabelten Datenpunkten enthält. Idealerweise sind beide Klassen (positiv|negativ) gleich groß. In V13 besprechen wir, was man tun kann, falls dies nicht der Fall ist (oversampling bzw. undersampling).

(3) Splitte Datensatz in einen größeren Anteil fürs **Trainieren** des Klassifikators und einen kleineren Anteil zur Bestimmung der Genauigkeit des trainierten Klassifikators (**Testen**) auf. Neben einer festen Auftrennung kann man auch abwechselnd verschiedene Anteile fürs Trainieren und Testen verwenden (Cross-Validation oder Leave One out).

(4) Trainiere Klassifikator auf Trainingsdatensatz, z.B. Optimierung von SVM-Parametern.

(5) Evaluiere Performanz des trainierten Klassifikators auf dem Test-Datensatz. Diese Daten wurden während dem Training nicht verwendet. Damit kann man also testen, wie hoch die Vorhersagegenauigkeit des Klassifikators für neue Datenpunkte ist.

Cross Validation und Leave One Out



Bei der *k*-fachen Kreuzvalidierung führt man *k* Iterationen des Trainings durch, bei der abwechselnd verschiedene Auftrennungen in Training / Test-Daten verwendet werden.

Danach bestimmt man die mittlere Genauigkeit der Vorhersage.

www.wikipedia.org

Eine Variante davon ist (**leave one out**), wobei man stets nur einen Datenpunkt fürs Testen verwendet und auf allen anderen Datenpunkten trainiert.

Dies verwendet man vor allem, wenn der Datensatz recht klein ist. Allerdings muss man dann *n* Iterationen des Trainings durchführen.

Bewertung von Klassifikationsergebnissen

TP: true positive - positiv gelabelter Datenpunkt: wurde korrekt als positiv vorhergesagt

FP: false positive - negativ gelabelter Datenpunkt; wurde falsch als positiv vorhergesagt

TN: true negative - negativ gelabelter Datenpunkt: wurde korrekt als negativ vorhergesagt

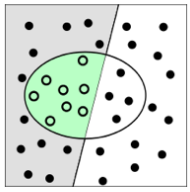
FN: false negative - positiv gelabelter Datenpunkt; wurde falsch als negativ vorhergesagt

Sensitivität (richtig-positiv-Rate):

Wahrscheinlichkeit, mit der ein positives Objekt korrekt als positiv klassifiziert wird.

$$P(positive\ Vorhersage|tatsächlich\ positiv) = \frac{TP}{TP + FN}$$

Dabei sind TP und FN die Anzahl an true positives und false negatives



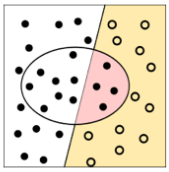
www.wikipedia.org

Bewertung von Klassifikationsergebnissen

Spezifität (richtig-negativ-Rate):

Wahrscheinlichkeit, mit der ein negatives Objekt korrekt als negativ klassifiziert wird. .

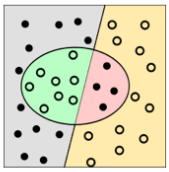
$$P(negative\ Vorhersage|tatsächlich\ negativ) = \frac{TN}{TN + FP}$$



Korrektklassifikationsrate (englisch: **accuracy**):

Anteil an Objekten, die korrekt klassifiziert werden.

$$P(richtig\ klassifiziert) = \frac{TN + TP}{TP + FP + TN + FN}$$



Es gibt auch kombinierte Maße wie den F-Wert

(auch als F1-Wert bezeichnet).

www.wikipedia.org

ROC-Kurve, AUC-ROC

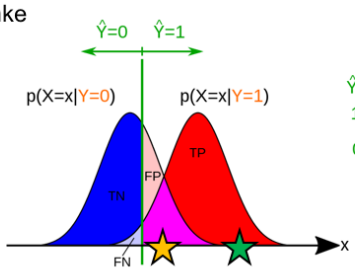
Oft verwendet man eine Parameterschranke um positive und negative Vorhersagen voneinander zu trennen.

Dann stellt sich jedoch die Frage, bei welchem Parameterwert man die besten Vorhersagen erhält.

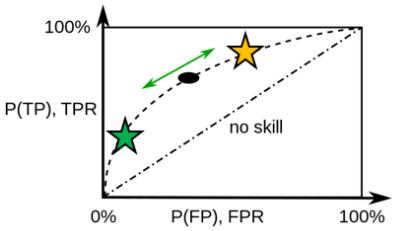
Dazu kann man die ROC-Kurve (receiver operator characteristics) verwenden, bei der man die Sensitivität (Richtig-Positiv-Rate) auf der y-Achse gegen die Falsch-Positiv-Rate auf der x-Achse aufträgt.

Zur Bewertung des Klassifikators benutzt man oft die Fläche unter der ROC-Kurve (Area Under Curve): AUC-ROC.

Eine perfekte Vorhersage ergibt AUC = 1.



Ŷ \ Y	1	0
1	TP	FP
0	FN	TN



www.wikipedia.org

In diesem Beispiel erhält man für einen hohen positiven Wert (grüner Stern) der Parameterschranke x nur TPs und fast keine FPs. Mit einem tieferen Wert (oranger Stern) findet man zwar einen größeren Anteil der TPs, aber nun auch FPs.