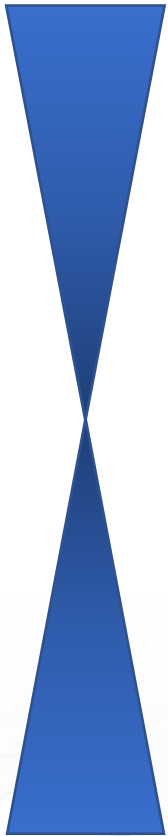




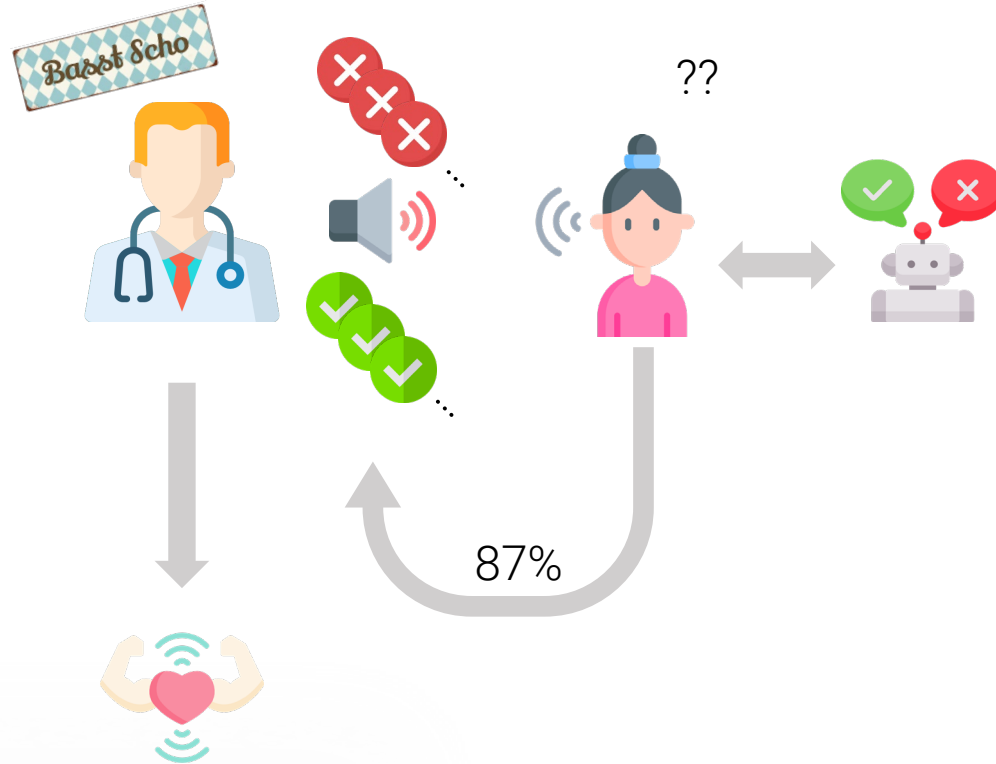
Evaluation

Verrauschen von Daten, Confusion Matrix, ROC Curves

Agenda

- 
1. Signal Detection Theory
 2. Confusion Matrix
 3. Receiver-Operator-Characteristic (ROC Curves)
 4. Weitere Maße aus der Confusion Matrix:
 1. F1-Score
 2. ...
 5. Verrauschen von Daten
 6. SHAP (SHapley Additive exPlanations) → Towards XAI

Stellen Sie sich vor...



So what?

In diesem Beispiel sind Sie der Klassifikator! Alles, was wir uns nun über dieses Beispiel herleiten, können Sie eins-zu-eins auf ein Klassifikationsmodell übertragen!

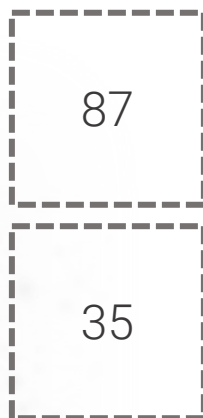
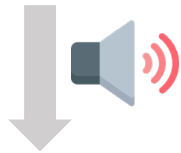
- ...Sie gingen zum Ohrenarzt und wollen Ihre Hörfähigkeit überprüfen lassen
- Hierzu präsentiert Ihnen der Arzt ein – für Sie gerade noch hörbaren – Sound
- Und zwar durchlaufen Sie insgesamt 200 Durchläufe dieses Experiments – jedoch 100 mal mit und 100 mal ohne diesen Sound
- Aus diesen 100 Signaldurchläufen erkennen Sie 87 mal den leisen Sound
- Ihr Arzt bescheinigt Ihnen, dass Sie eine 87%-ige „Hörfähigkeit“ haben und schickt Sie wieder los

Sind Sie mit Ihrem Arzt zufrieden?



Was denken Sie?
Was hat Ihr Arzt
grundlegend
falsch gemacht?

Confusion Matrix: towards Signalentdeckungstheorie



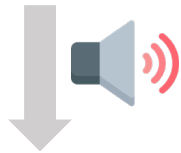
- Was Ihr Arzt getan hat war ein grober Fehler – wissenschaftlich gesehen
 - Er hat nur die sog. *True Positives* zur Bewertung Ihrer Hörfähigkeit herangezogen und daraus eine Genauigkeit berechnet
 - Stellen Sie sich vor Sie würden in den 200 Durchgängen einfach *jedes Mal* „ja – Sound gehört“ sagen. Was passiert?
 - Genau – Sie hätten eine 100%-ige „Hörfähigkeit“ bzw. würden zu 100% True Positives liefern
 - Hierbei werden jedoch die Durchgänge vernachlässigt, bei denen Sie fälschlicherweise „ja – Sound gehört“ sagen
→ die sog. **False Positives**
- Erst durch die Betrachtung dieser beiden Maße – zur gleichen Zeit – lässt sich die Güte eines Klassifikators wirklich beurteilen!



Was denken Sie?
Was fehlt jetzt noch?

Confusion Matrix: towards Signalentdeckungstheorie

Actual Condition positive (P)	True Positives (TP)	False Negatives (FN)
Actual Condition Negative (N)	False Positives (FP)	True Negatives (TN)



100	87	
100	35	

- Wenn wir nun noch die Durchgänge ohne Sound betrachten, dann fehlen uns noch die Durchgänge, bei denen wir richtigerweise (**True Negatives**) und fälschlicherweise (**False Negatives**) „nein – kein Sound“ gesagt haben
- Die Anzahl der FN ergeben sich aus der Anzahl der gesamten *Signaldurchgänge* (P), die TN aus der Anzahl der gesamten *Rauschdurchgänge* (N)
→ zu dieser Anordnung sagt man **Confusion Matrix**
- In unserem Fall haben wir die gleiche Anzahl an P und N – das muss aber nicht sein! Daher sind die TP, FN, FP und TN nur relativ zu den P und N interpretierbar
- Wir landen bei **Rates** (die zwei wichtigsten):
 - **True Positive Rate (TPR):**

$$\text{TPR} = \frac{\text{TP}}{P} = 1 - \text{FNR}$$

- **False Positive Rate (FPR):**

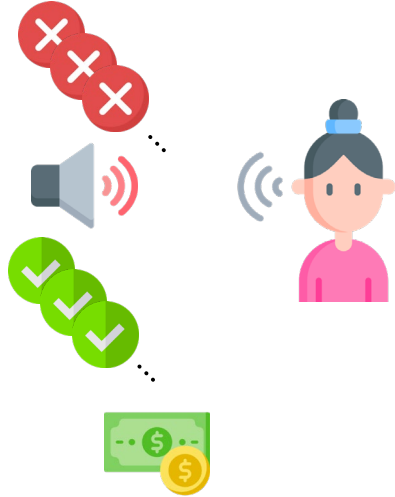
$$\text{FPR} = \frac{\text{FP}}{N} = 1 - \text{TNR}$$



Was denken Sie?

Was muss ich in
unserem konkreten
Beispiel in die
Felder eintragen?

Stellen Sie sich wieder vor...



98	2
83	17

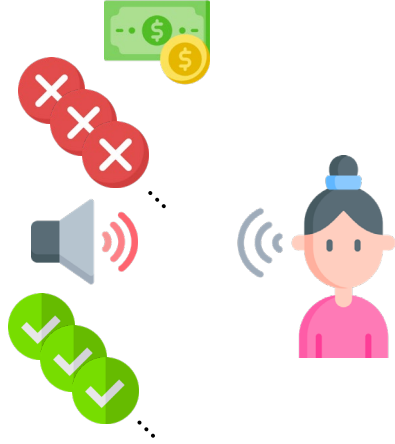
- ...Sie saßen wieder bei Ihrem Arzt. Jetzt ist Ihr Arzt auch noch korrupt und gibt Ihnen Geld für bestimmte **Antwortkategorien**
- Er meint nun, dass er das mit der Confusion Matrix verstanden hätte und will Sie nun so beeinflussen, dass Sie ein gutes Hörvermögen im Test zeigen
- Bei diesem Treffen erhalten Sie pro True Positive eine Belohnung von 5EUR
- Was passiert: Sie werden *liberaler* in Ihren Antworten. Daher werden Sie mehr True Positives, aber auch mehr False Positives produzieren

Geht der Plan Ihres Arztes auf? Hat er die Confusion Matrix wirklich verstanden?



Was denken Sie?
Wie wird sich unsere Confusion Matrix verändern?

Stellen Sie sich wieder vor...



33	67
16	84

- ...so, das hat also nicht geklappt, denkt sich der Arzt. Also versucht er es in die andere Richtung – weil wenn es so nicht geht, dann muss es doch anders gehen
- Also: Sie bekommen nun für jedes False Positive eine Rechnung von 5EUR durch Ihren Arzt gestellt. Wie wird sich nun Ihr Verhalten ändern?
- Genau, Sie werden *konservativer*. Sie werden versuchen False Positives zu vermeiden
- Die Confusion Matrix verändert sich wieder – aber nun in eine andere Richtung

Hat Ihr Arzt es diesmal geschafft Sie bzw. Ihre Hörfähigkeit zu beeinflussen?

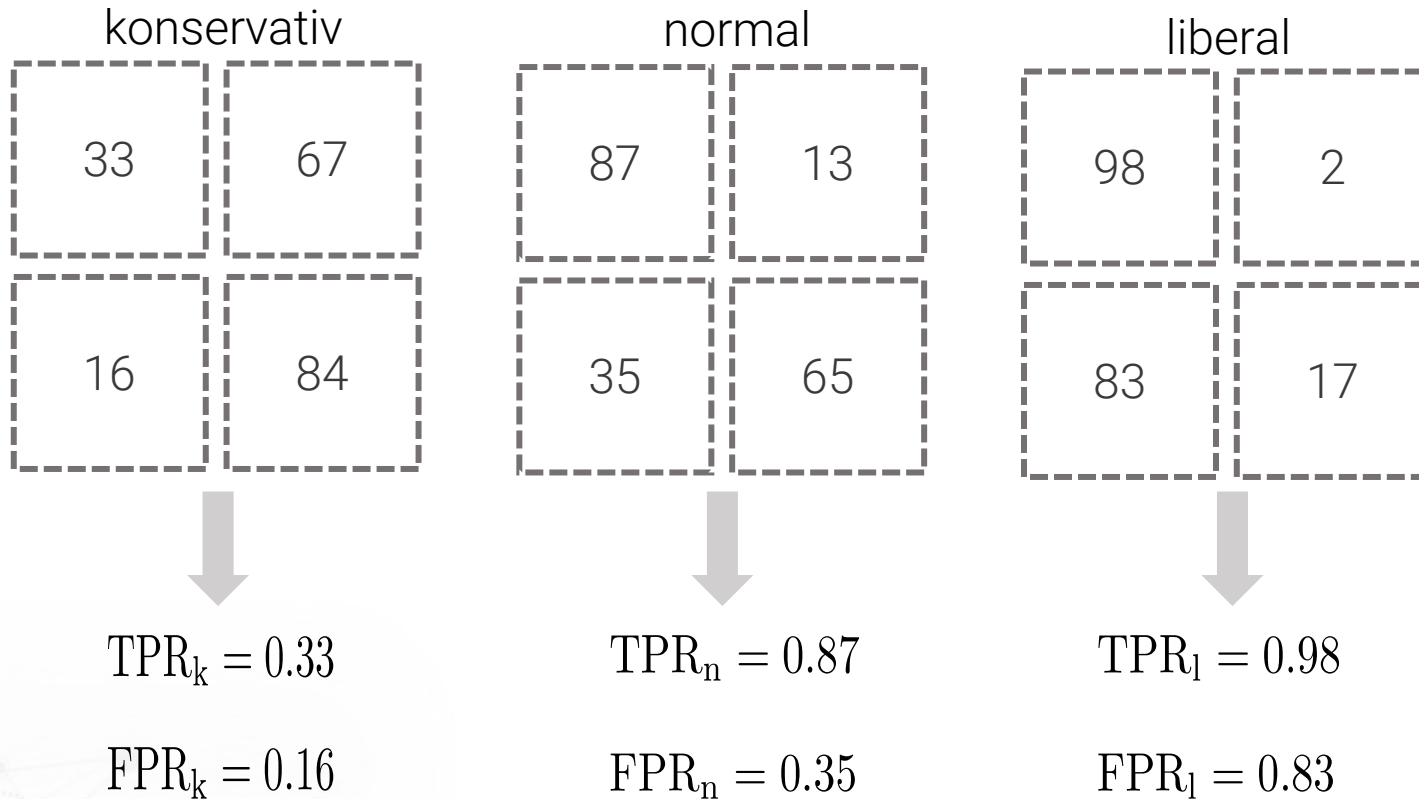


Towards Receiver Operator Characteristic



Was denken Sie?

Was tun wir jetzt mit diesen Werten?



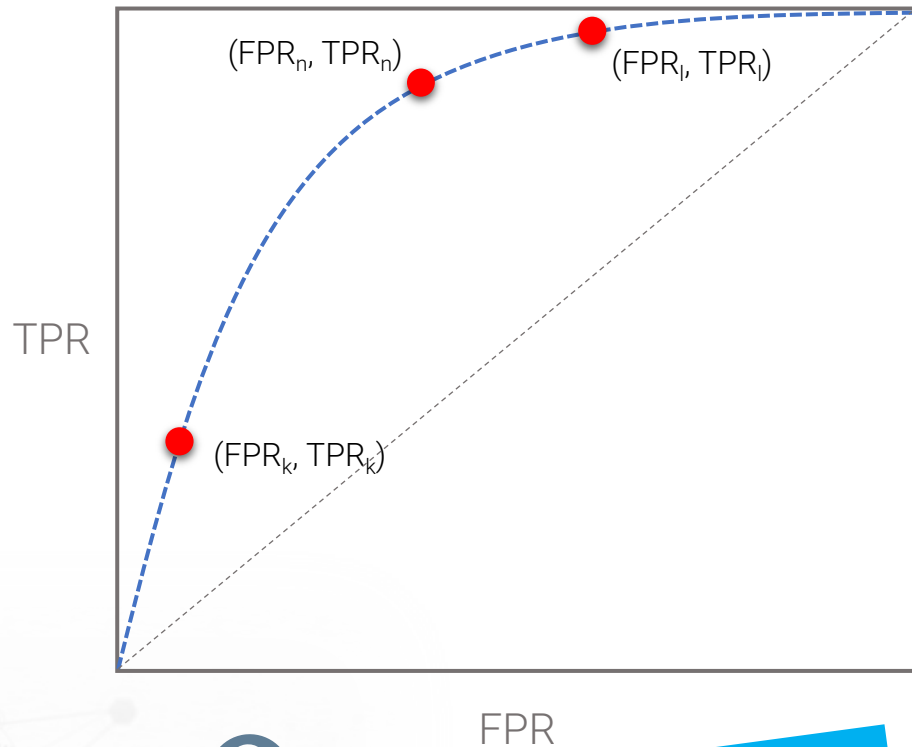
- Lassen Sie uns mal diese drei Fälle gegenüberstellen – den konservativen, normalen und liberalen – und die Rates ausrechnen
- Was haben wir getan: irgendwie haben wir ein *Kriterium* variiert
- Nun tragen wir die TPR und FPR für die jeweiligen Kriterien in ein zweidimensionales Koordinatensystem auf

0

So what?

Zu diesem Phänomen sagt man auch *Sensitivity-Specificity-Trade-off*, da man zur TPR auch *Sensitivity* und zur True Negative Rate *Specificity* sagen kann

ROC Curves



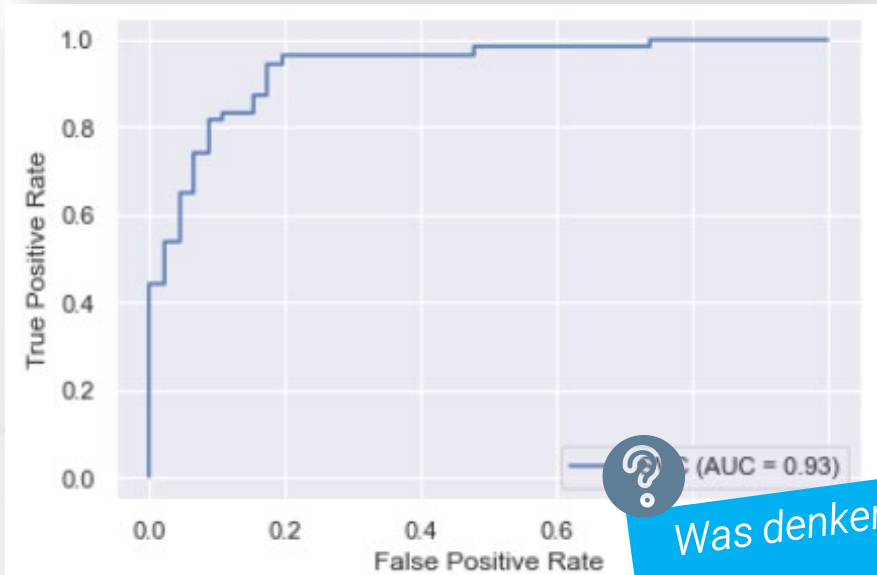
- Wie bauen wir dieses Koordinatensystem auf: Abszisse repräsentiert die FPR, die Ordinate die TPR
- Dann tragen wir unsere Wertepaare der Rates auf
- Vorher haben wir schon indirekt gesehen, dass die Veränderung unseres Kriteriums nicht wirklich mit der Unterscheidungsfähigkeit bzw. Klassifikationsgenauigkeit zu tun hat
- Daher schlussfolgern wir: alle Punkte charakterisieren einen Klassifikator (uns bzw. eine Person im konkreten Fall), der natürlich nur eine bestimmte Klassifikationsgenauigkeit aufweisen kann
- Die Punkte liegen also auf einer **Kurve**, die diesen Klassifikator beschreibt
→ Die Receiver-Operator-Characteristic (ROC) Curve
- Der **ideale Klassifikator** steigt bei $FPR = 0$ sofort auf $TPR = 1$ und bleibt konstant
- Ein Klassifikator, der nichts unterscheiden kann entspricht der Winkelhalbierenden



Was denken Sie?
Beim Patienten war es klar: wie kommen wir aber nun bei einem ML Modell zu einer Kurve?

ROC Curves in sklearn

```
1 # Train some classifier
2 from sklearn.svm import SVC
3 from sklearn.metrics import roc_curve, plot_roc_curve
4 model = SVC(kernel='linear').fit(X_train, y_train)
5 y_score = model.decision_function(X_test)
6
7 # Compute ROC
8 fpr, tpr, thresholds = roc_curve(y_test, y_score)
9 plot_roc_curve(model, X_test, y_test);
```



Was denken Sie?
Was ist hier genau
der `y_score`?

- Lassen Sie uns hierzu vorab einmal eine ROC Kurve mittels `sklearn` malen
- **Nachdem** wir einen Klassifikator trainiert haben, haben wir die Möglichkeit die entsprechende Funktion zu nutzen
`sklearn.metrics.roc_curve(y_true, y_score)`
- Sie sehen schon: wir benötigen zum einen die wahren Labels des Testdatensatzes `y_true` und `y_score`
→ Ausgabe der Entscheidungsfunktion oder auch von `.predict_proba()` – wir brauchen etwas, das mit einem Schwellwert (unserem Kriterium) versehen werden kann
- Wenn wir in die Doku schauen: „*this implementation is restricted to the binary classification task.*“ → auf Multi-Class kommen wir später zu sprechen
- Wir haben auch eine Convenience-Funktion, die uns gleich die ROC-Werte berechnet und plottet

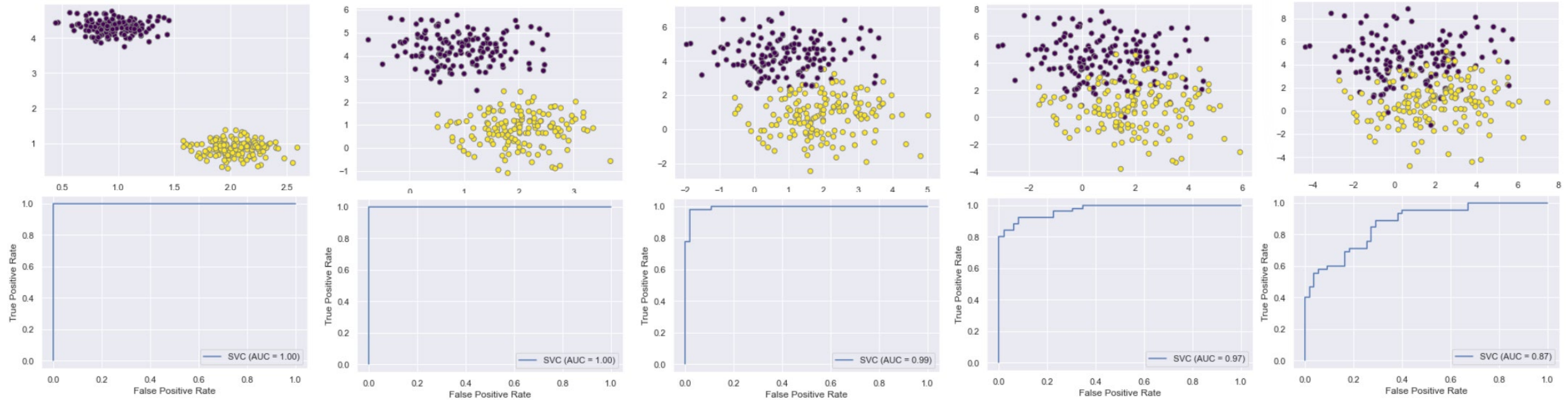
`plot_roc_curve(model, X, y)`

0

So what?

Wir haben also auch bei ML Modellen die Möglichkeit einen *Threshold* zu variieren, um ROC Kurven zu erzeugen

ROC Curves in sklearn: a G'fühl griang



- Wir variieren mal den Datensatz und führen eine zunehmende Streuung ein – damit eine zunehmende Überlappung der Klassen
- Dadurch wird es für unseren Klassifikator immer schwieriger genaue Vorhersagen zu treffen → dies sehen wir an der Veränderung der ROC Kurven



Was denken Sie?

Können Sie schon erraten was – abgeleitet von der ROC Kurve – ein Gütekriterium bzw. –maß für einen Klassifikator sein könnte?

Decision Threshold

T

X



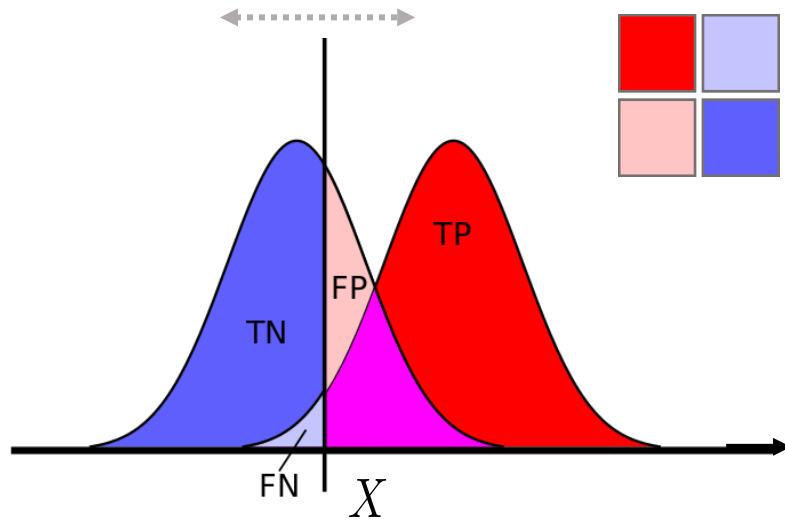
Was denken Sie?

Wo in dieser Abbildung verstecken sich nun die TPR, FPR, FNR und TNR?

- Um diese ROC Kurven zu erhalten, muss ein Kriterium variiert werden, das unseren Klassifikator (bzw. unseren Patienten) dazu veranlasst unterschiedlich konservativ bzw. liberal zu antworten
- Im Machine Learning bezeichnet man dieses Kriterium als sog. *Decision Threshold* bzw. *Discrimination Threshold*
- Um diese zu verstehen, müssen wir tiefer in die zugrundeliegende Theorie zur ROC abtauchen
- Lassen Sie uns mal den Klassifikator als etwas betrachten, das die **Wahrscheinlichkeit** abschätzt, ob ein Datenpunkt eher zur einen oder anderen Klasse gehört
→ dies kann durch eine Zufallsvariable X repräsentiert werden
- Die Zugehörigkeit zu einer Klasse wird dann mittels eines Schwellwerts bestimmt: $X > T \rightarrow$ Klasse „positiv“
- Im Umkehrschluss gibt es für X sowohl eine Wahrscheinlichkeitsdichte $f_1(x)$ für Datenpunkte, die zu Klasse „positiv“ gehören, als auch $f_0(x)$ für Klasse „negativ“



Decision Threshold



- Genau – als Fläche unter den Graphen bzw. mittels Integralschreibweise
- Da ja für die Prediction „positiv“ $X > T$ gilt, können wir Folgendes ausdrücken:

$$TPR = \int_T^{\infty} f_1(x) dx$$

$$FPR = \int_T^{\infty} f_0(x) dx$$

- Die anderen Rates ergeben sich dann über die Differenzen von 1
- Indem nun der Schwellwert von minus unendlich bis plus unendlich „abgefahren“ wird, ergeben sich die ROC Kurven
→ Jede Einstellung des Schwellwerts (Decision Threshold) ergibt einen Punkt im TPR-FPR-Koordinatensystem – die ROC Kurve



Was denken Sie?
Wie komme ich jetzt
letztendlich auf ROC Kurven?



0

So what?

Wenn man nun annehmen würde,
dass die Wahrscheinlichkeitsdichten
Gauß'sche sind, dann könnte man
sogar ROC Kurven **analytisch** herleiten!

Decision Threshold: Use Cases



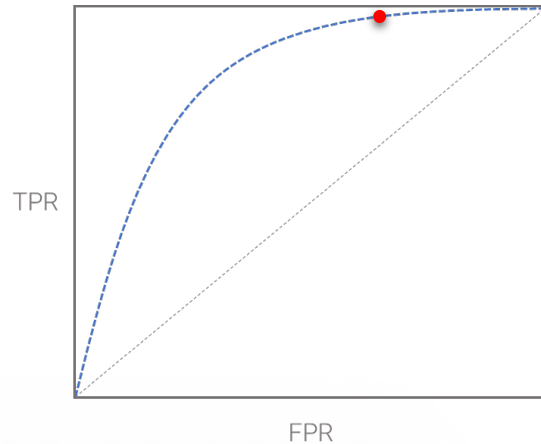
Was denken Sie?
Können Sie sich vorstellen wo für Sie
in der Praxis dieser Decision
Threshold von Interesse sein könnte?



Klassifikator zur Detektion von Krankheiten

Anhand von Bildgebungsdaten soll mittels eines ML-Modells klassifiziert werden, ob der Patient eine bestimmte Krankheit hat

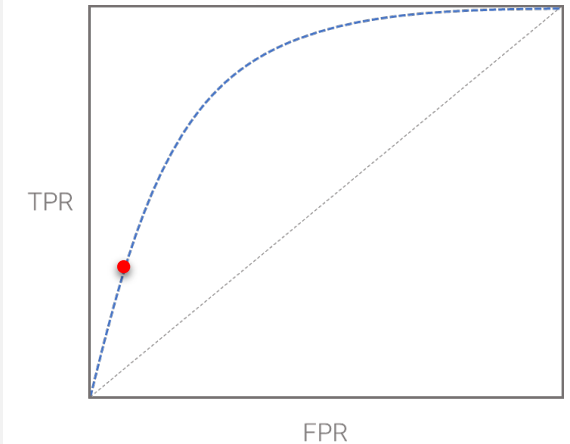
→ Wohin verschieben sie den Decision Threshold?



Predictive Maintenance von Kugellagern

Anhand von Vibrationsdaten soll vorausgesagt werden, wann bzw. ob ein Lager beschädigt sein wird bzw. ist. Jeder Alarm wird an den Operator gemeldet

→ Wohin verschieben sie den Decision Threshold?



So what?

Mit dem Decision Threshold können Sie Ihr Modell „fine-tunen“ und an die Anforderungen des zugrundeliegenden Use Cases anpassen.

Beispiel: Cheap and Expensive Bearings



Stellen Sie sich vor Sie würden in einem Maschinenbauunternehmen arbeiten. An allen Ecken und Enden sind Kugellager verschiedenster Arten verbaut - teure große, billige kleine. Sie haben mit Ihrem Datenanalyseteam ein Machine Learning Modell (`bearing_model.pkl`) erzeugt, das die Klassifikation von Kugellager aller Arten in "gut" und "beschädigt" auf zufriedenstellende Weise durchführen kann. Nun will Ihr Vorgesetzter aber einerseits vermeiden, dass Sie die teuren Kugellager **zu häufig fälschlicherweise**, die billigen **zu selten zum richtigen Zeitpunkt** auswechseln. Für Ihren Vorgesetzten bedeutet Ersteres `fpr~0.023` , und Letzteres `tpr~0.97` . Bauen Sie Ihrem Vorgesetzten hierzu eine Funktion, die Ihnen ein trainiertes Modell `model` und einen Testdatensatz `Xtest` aufnehmen kann. Die Funktion soll die angepassten Vorhersagen `ypredicted_new` zurückgeben und in einer ROC-Kurve markieren, an welchem Punkt es sich mit seiner neuen Einstellung befindet. Welche Eingabeargumente benötigen wir noch, um die unterschiedliche Behandlung von `fpr` und `tpr` zu berücksichtigen?

Confusion Matrix in sklearn

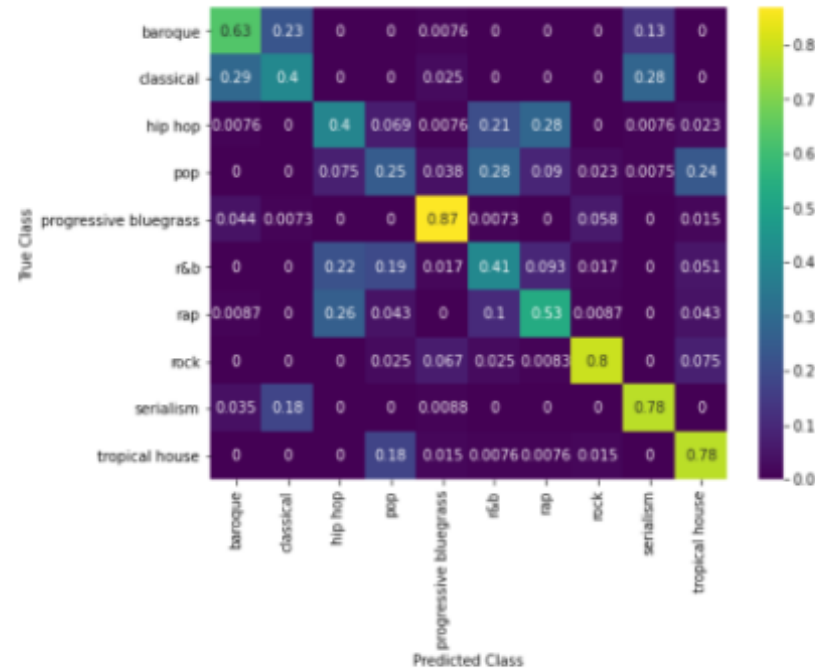


- „Vor lauter lass mi aa no mit“ → wir schauen uns noch Confusion Matrizen in sklearn an

`confusion_matrix(ytest, ypredicted, normalize)`

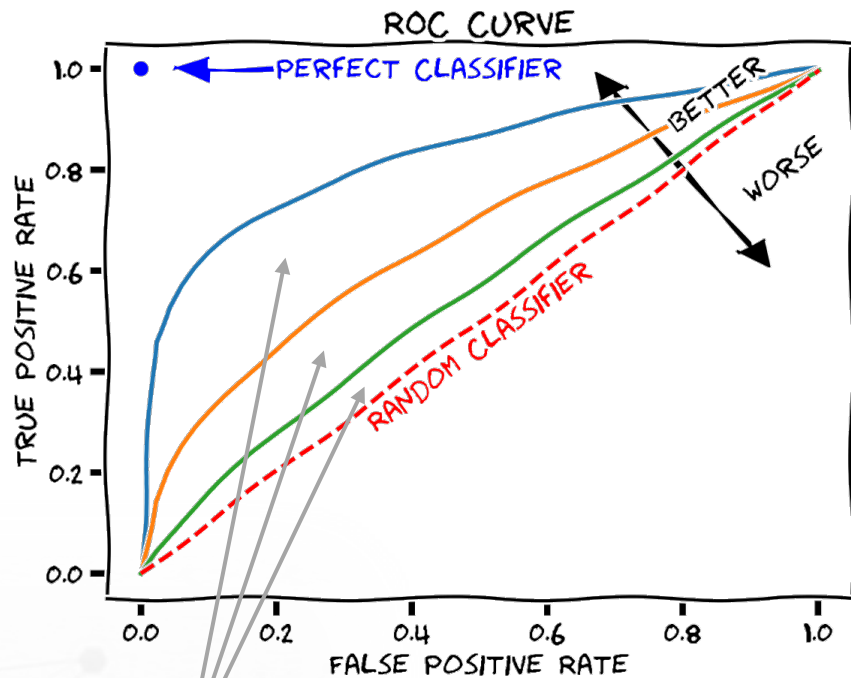
- `normalize=True` bedeutet, dass die Einträge auf das Intervall `[0; 1]` transformiert werden
- Am besten visualisiert man sich die Confusion Matrix mit einer Heatmap – z.B. `sns.heatmap`

Beispiel: *What* confuses our models?



Wir laden uns die Modelle verschiedener vergangener Aufgaben und lassen uns die Confusion Matrix mittels einer `sns.heatmap` visualisieren. Welche Klassen können unsere Modelle am besten, welche am wenigsten auseinanderhalten?

Area-Under-the-Curve: AUC



Immer größer werdende AUC

- In einer vorhergehenden Folie hatten wir schon gesehen, dass ein geeignetes Maß zur Bewertung einer ROC-Kurve die Fläche darunter ist → die Area-Under-the-Curve (AUC)
- Auch hierfür gibt es in `sklearn` eine implementierte Funktion
`roc_auc_score(y_true, y_score)`
- Wobei auch hier – wie bei der `roc_curve`-Funktion – `y_score` dem Output der `.decision_function()`- oder `.predict_proba()`-Methode entspricht
- Auch die `plot_roc_curve()`-Funktion gibt uns im Plot die AUC an

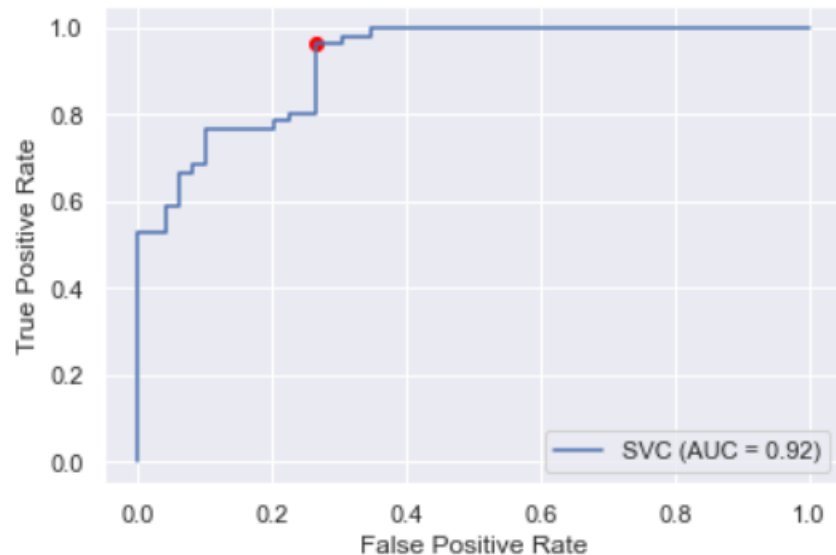
Optimal Decision Threshold



Was denken Sie?

Wie würden Sie einen optimalen Threshold bestimmen?

```
1 # Get TPR and FPR
2 fpr, tpr, thresholds = roc_curve(y_test, y_score)
3 plot_roc_curve(model, X_test, y_test);
4
5 # Calculate Youden's J
6 J = tpr - fpr
7 best_threshold_index = np.argmax(J)
8 best_tpr = tpr[best_threshold_index]
9 best_fpr = fpr[best_threshold_index]
10
11 # Show
12 plt.scatter(best_fpr, best_tpr, c='red');
```



- Eine pragmatische Möglichkeit einen optimalen Wert des Decision Threshold zu wählen ist den Punkt zu wählen, an dem $J = TPR - FPR$ maximal wird – der sog. *Youden-Index*
- Dadurch liegt ein einfaches Maß vor, um einen Decision Threshold zu wählen, an dem eine optimale Balance zwischen True und False Positives vorliegt

Weitere Maße aus der Confusion Matrix

Sources: [20][21][22][23][24][25][26][27] view · talk · edit

		Predicted condition			
		Positive (PP)	Negative (PN)	Informedness, bookmaker informedness (BM) $= \text{TPR} + \text{TNR} - 1$	Prevalence threshold (PT) $= \frac{\sqrt{\text{TPR} \times \text{FPR}} - \text{FPR}}{\text{TPR} - \text{FPR}}$
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{\text{TP}}{\text{P}} = 1 - \text{FNR}$	False negative rate (FNR), miss rate $= \frac{\text{FN}}{\text{P}} = 1 - \text{TPR}$
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection	False positive rate (FPR), probability of false alarm, fall-out $= \frac{\text{FP}}{\text{N}} = 1 - \text{TNR}$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{\text{TN}}{\text{N}} = 1 - \text{FPR}$
	Prevalence $= \frac{\text{P}}{\text{P} + \text{N}}$	Positive predictive value (PPV), precision $= \frac{\text{TP}}{\text{PP}} = 1 - \text{FDR}$	False omission rate (FOR) $= \frac{\text{FN}}{\text{PN}} = 1 - \text{NPV}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Negative likelihood ratio (LR-) $= \frac{\text{FNR}}{\text{TNR}}$
	Accuracy (ACC) $= \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}}$	False discovery rate (FDR) $= \frac{\text{FP}}{\text{PP}} = 1 - \text{PPV}$	Negative predictive value (NPV) $= \frac{\text{TN}}{\text{PN}} = 1 - \text{FOR}$	Markedness (MK), deltaP (Δp) $= \text{PPV} + \text{NPV} - 1$	Diagnostic odds ratio (DOR) $= \frac{\text{LR}+}{\text{LR}-}$
	Balanced accuracy (BA) $= \frac{\text{TPR} + \text{TNR}}{2}$	F ₁ score $= \frac{2\text{PPV} \times \text{TPR}}{\text{PPV} + \text{TPR}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$	Fowlkes–Mallows index (FM) $= \sqrt{\text{PPV} \times \text{TPR}}$	Matthews correlation coefficient (MCC) $= \sqrt{\text{TPR} \times \text{TNR} \times \text{PPV} \times \text{NPV}} - \sqrt{\text{FNR} \times \text{FPR} \times \text{FOR} \times \text{FDR}}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{\text{TP}}{\text{TP} + \text{FN} + \text{FP}}$

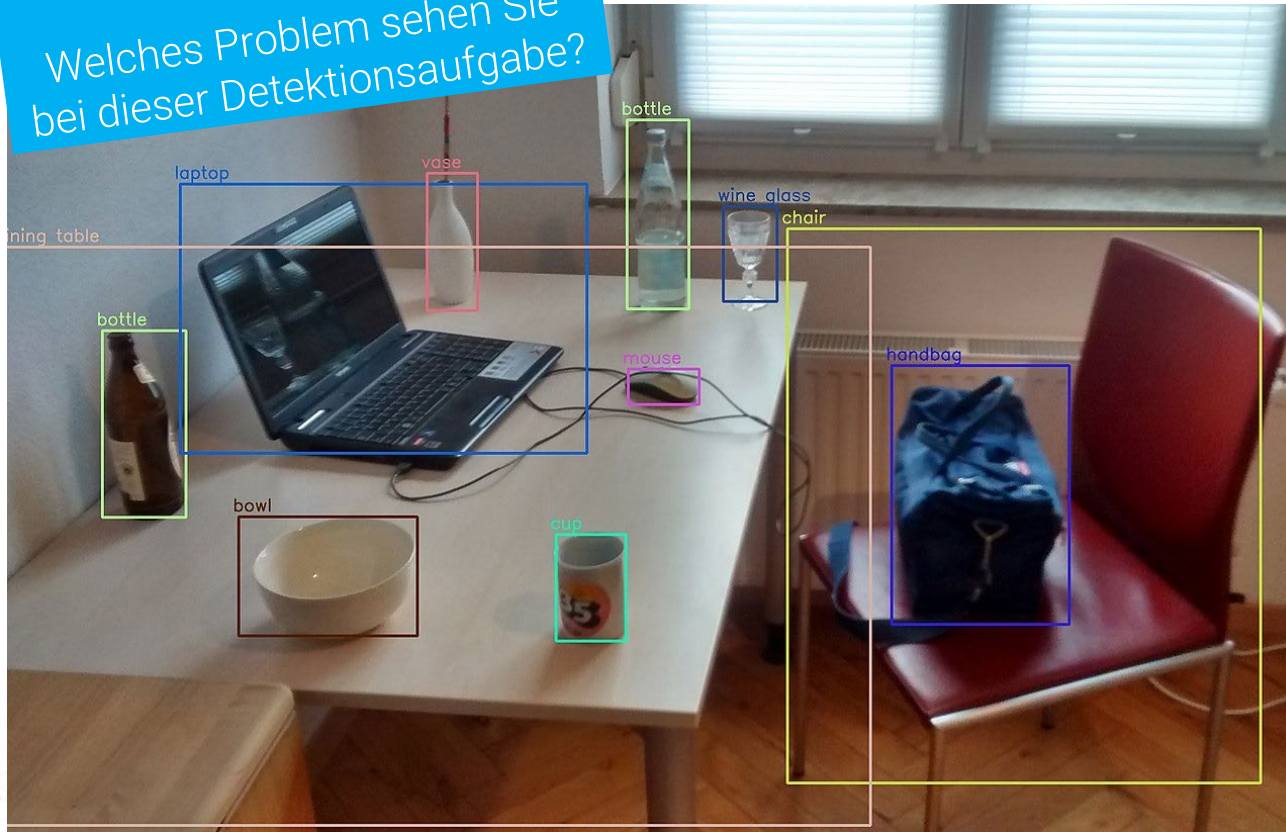
https://en.wikipedia.org/wiki/Confusion_matrix

Precision Recall Curve:



Was denken Sie?

Welches Problem sehen Sie bei dieser Detektionsaufgabe?



Was denken Sie?

Wie könnte diese Alternative aussehen?

- In der Realität kann es Ihnen passieren, dass z.B. True Negatives nicht verfügbar sind (s. Use Case Object Detection)
- In solchen Fällen müssen Sie sich eine Alternative überlegen
→ Anstatt der FPR brauchen wir ein anderes Maß
- Man nutzt hierzu die sog. *Precision* bzw. den *Positive Predictive Value*

$$PPV = \frac{TP}{TP + FP}$$

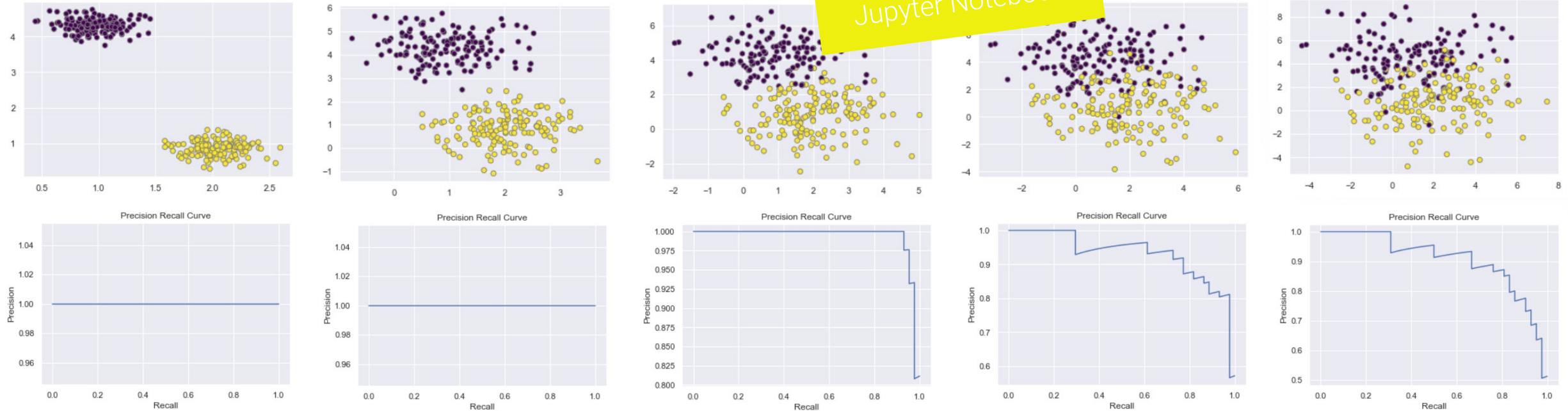
→ Im Nenner sind alle *Positive Calls* repräsentiert – es liegt also ein Maß vor, das angibt „welcher Anteil meiner ‚Entdeckungen‘ war richtig?“

- Zur TPR sagt man auch *Recall* - und jetzt wissen Sie schon was man tut...

Precision Recall Curve in sklearn



Demo
Ausführung in einem
Jupyter Notebook



- In sklearn haben wir wieder eine entsprechende Funktion `precision_recall_curve(y_true, y_score)`, die uns entsprechend die `precisions`, `recalls` und `thresholds` zurückgibt
- Wir sehen:
 - auch hier spiegelt eine große Fläche unter der Kurve einen guten Klassifikator wider
 - die besseren Klassifikatoren streben in die Ecke rechts oben

F1-Score



- Es gibt eine Statistik, die uns Precision und Recall im harmonischen Mittel zusammenfasst – der F1-Score:

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

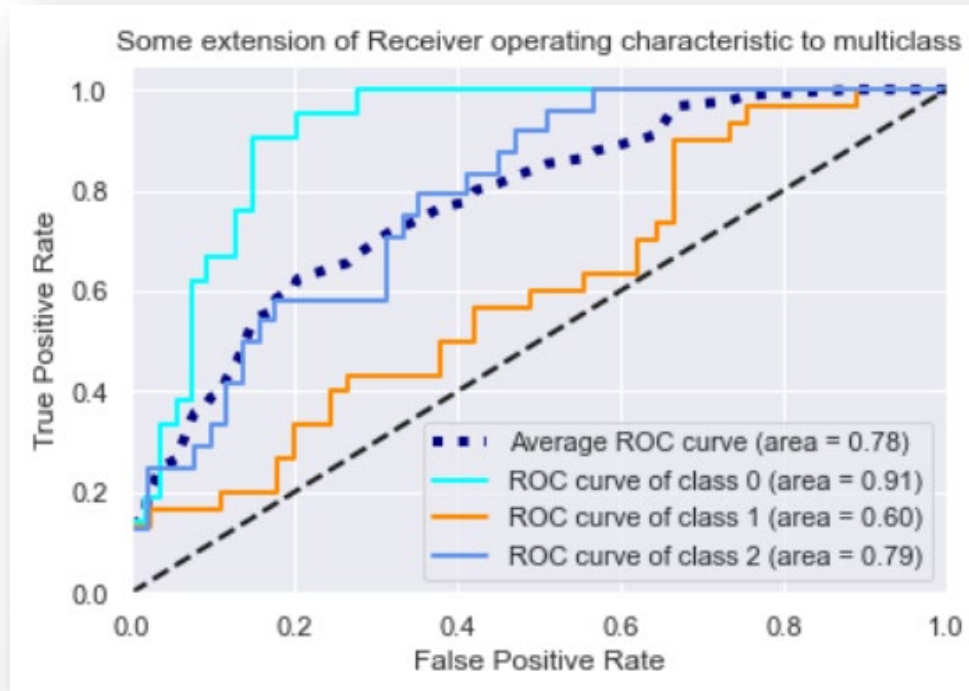
- Dadurch liegt uns ein Maß vor, um die Qualität eines Klassifikator mittels *eines* Skalars zu bewerten
- Der Wertebereich liegt zwischen 0 (schlecht) und 1 (gut)
- In `sklearn` zu finden unter `sklearn.metrics.f1_score`

Multi-Class ROC Curves



Was denken Sie?

Was tun wir nun bei mehr als zwei Klassen?



- Wir tun das gleiche wie bei der SVM: wir transformieren das Problem in ein binäres
- Entweder erzeugen wir uns für jede Klassenpaarung eine ROC-Kurve („One-versus-one“)
- Oder für jede Klasse im Vergleich zu allen anderen („One-versus-rest“)
- Daraus kann dann eine mittlere ROC-Kurve erzeugt werden
- Entsprechend können dann die AUCs berechnet werden
- Die `sklearn` Funktion `roc_curve` kann nur mit binären, die Funktion `roc_auc_score` auch mit multi-class Problemen umgehen