

Grundlagen des maschinellen Lernens

Preprocessing

Prof. Dr. Volker Gruhn

1. Preprocessing
 - 1.1.Fehlende Werte
 - 1.2.Standardisierung & Normalisierung
 - 1.3.Ausblick: Nicht-lineare Transformation
 - 1.4.Encoding kategorischer Werte
 - 1.5.Encoding von Uhrzeiten, Koordinaten,...
2. Evaluierung
 - 2.1.Accuracy
 - 2.2.Precision
 - 2.3.Recall
 - 2.4.F1-Score
 - 2.5.ROC-Kurve und AUC-Score
 - 2.6.Mean Average Error
 - 2.7.Mean Average Percentage Error

- Matrix $X = (x^{(1)}, \dots, x^{(m)})^T = (x_{ij})_{i=1, \dots, m; j=1, \dots, n} \in \mathbb{R}^{m \times n}$ sei der aus erhobenen Daten in einen geeigneten Feature Space transformierte Datensatz
- $x^{(1)}, \dots, x^{(m)}$ bezeichne die Samples
- y_1, \dots, y_m bezeichne die Labels
- $(x^{(1)}, y_1), \dots, (x^{(m)}, y_m)$ seien Datenpunkte
- x_1, \dots, x_n bezeichne die Featurevektoren über alle Samples

Preprocessing

Umgang mit fehlenden Werten

- (Fast) alle realen Datensätze weisen fehlende Werte auf, die die meisten ML Algorithmen nicht verarbeiten können
- Einige Optionen für den Umgang mit fehlenden Werten
 1. Entfernen von Samples (Zeilen von X)
 2. Entfernen von Features (Spalten von X)
 3. Setzen der fehlenden Werte auf einen bestimmten Wert
 4. Zusätzliches Feature (Flag) einfügen

- **Lösungsansatz 1: Samples mit fehlenden Werten entfernen**
 - Datensatz wird verkleinert!
 - Daher nur empfehlenswert, wenn nur ein kleiner Teilbereich aller Samples betroffen ist
 - Weitere Probleme können auftreten
 - z.B. kann die Verteilung der eigentlichen Population verfälscht werden (z.B. wenn Daten immer zu einer bestimmten Personengruppe fehlen)

		Features			
Samples					

- **Lösungsansatz 2: Features mit fehlenden Werten entfernen**
 - Menge der Samples wird nicht verringert
 - Dafür besteht die Gefahr, dass relevante Features vollständig entfernt werden und das mögliche Ergebnis verschlechtern
 - Bei sehr vielen fehlenden Werten meistens sinnvoll

Features

Samples

-
- The figure displays a 10x4 grid representing feature importance for 10 samples across 4 features. The y-axis is labeled 'Samples' and the x-axis is labeled 'Features'. The grid is mostly light gray, indicating low importance, with two green cells at (Sample 4, Feature 3) and (Sample 9, Feature 3), indicating high importance.
- | Samples | Feature 1 | Feature 2 | Feature 3 | Feature 4 |
|---------|-----------|-----------|-----------|-----------|
| 1 | Low | Low | Low | Low |
| 2 | Low | Low | Low | Low |
| 3 | Low | Low | Low | Low |
| 4 | Low | Low | High | Low |
| 5 | Low | Low | Low | Low |
| 6 | Low | Low | Low | Low |
| 7 | Low | Low | Low | Low |
| 8 | Low | Low | Low | Low |
| 9 | Low | Low | High | Low |
| 10 | Low | Low | Low | Low |

- **Lösungsansatz 4: Zusätzliches Feature als Markierung für fehlende Werte**

- Datensatz bleibt vollständig erhalten
- Auch möglich, wenn kein gutes Verfahren für einen Defaultwert bekannt ist
 - Fehlende Werte werden durch beliebigen Defaultwerte ersetzt
 - Zusätzlich wird der Ersatz markiert
- Der Umgang mit fehlenden Werten wird somit in den eigentlichen Lernprozess integriert
- Tendenziell eher für komplexere Verfahren geeignet.
- Nicht immer sinnvoll. Bedarf gute Kenntnisse der eingesetzten Modelle.

Features				
Samples				0
				0
				0
				1
				0
				0
				0
				0
				1
				0

Preprocessing

Standardisierung & Normalisierung

- Oft stark unterschiedliche Skalen numerischer Features
 - Beispiel Hauspreise: mittleres Einkommen $\in [6, 39320]$, Zimmeranzahl $\in [0, 15]$
- Die meisten ML Algorithmen weisen keine gute Performance bei Daten auf, welche diese Eigenschaft besitzen
 - Bspw. MSE als Kostenfunktion: Features mit einem großen Wertebereich fließen in die Berechnung überproportional ein
→ Skalierung von Features notwendig, sodass Features in gleichem Maße zum Ergebnis beitragen
- Hier betrachten wir zwei gängige Methoden zur Skalierung
 - Min-Max Skalierung, auch Normalisierung genannt
 - Standardisierung

- $[w_{\min}, w_{\max}]$ sei das Intervall, in dem alle Werte des Feature liegen

- $w \in [w_{\min}, w_{\max}]$ sei ein Wert des Feature

- Skalierung von w : $\hat{w} = \frac{w - w_{\min}}{w_{\max} - w_{\min}} \in [0, 1]$

$$\hat{w} = 0 \iff w = w_{\min}$$

$$\hat{w} = 1 \iff w = w_{\max}$$

$$\hat{w} \in (0,1) \quad \forall w \neq w_{\min}, w \neq w_{\max}$$

→ Nach Min-Max Skalierung liegen alle Werte numerischer Features im Intervall $[0, 1]$

- w_1, \dots, w_m seien die Werte des Feature

- Berechnung des Durchschnitts über alle Werte des Feature: $\bar{w} = \frac{1}{m} \sum_{i=1}^m w_i$

- Berechnung der Standardabweichung σ über alle Werte des Feature: $\sigma = \sqrt{\frac{1}{m} \sum_{i=1}^m (w_i - \bar{w})^2}$

- Skalierung von w_i , $i \in \{1, \dots, m\}$: $\hat{w}_i = \frac{w_i - \bar{w}}{\sigma}$

Für den skalierten Featurevektor $\hat{w} = (\hat{w}_1, \dots, \hat{w}_m)^T$ gilt: $\bar{\hat{w}} = \frac{1}{m} \sum_{i=1}^m \hat{w}_i = 0$, $\sigma = \sqrt{\frac{1}{m} \sum_{i=1}^m (\hat{w}_i - \bar{\hat{w}})^2} = 1$

→ Nach Standardisierung haben alle numerischen Features einen Durchschnitt von 0 und eine Standardabweichung von 1, Standardisierung legt somit die Verteilung fest

Hinweis: in der Literatur ist auch eine Skalierung mit Varianz = σ^2 zu finden, für den Durchschnitt und die Standardabweichung des skalierten Featurevektors gilt dasselbe

Min-Max Skalierung

- Alle Werte numerischer Features liegen im Intervall $[0, 1]$
- Wegen Festlegung des Wertebereichs anfällig für Outlier
- Kein Festlegen einer Verteilung
- Nützlich bei Algorithmen, die keine Verteilung annehmen/unterstellen

Standardisierung

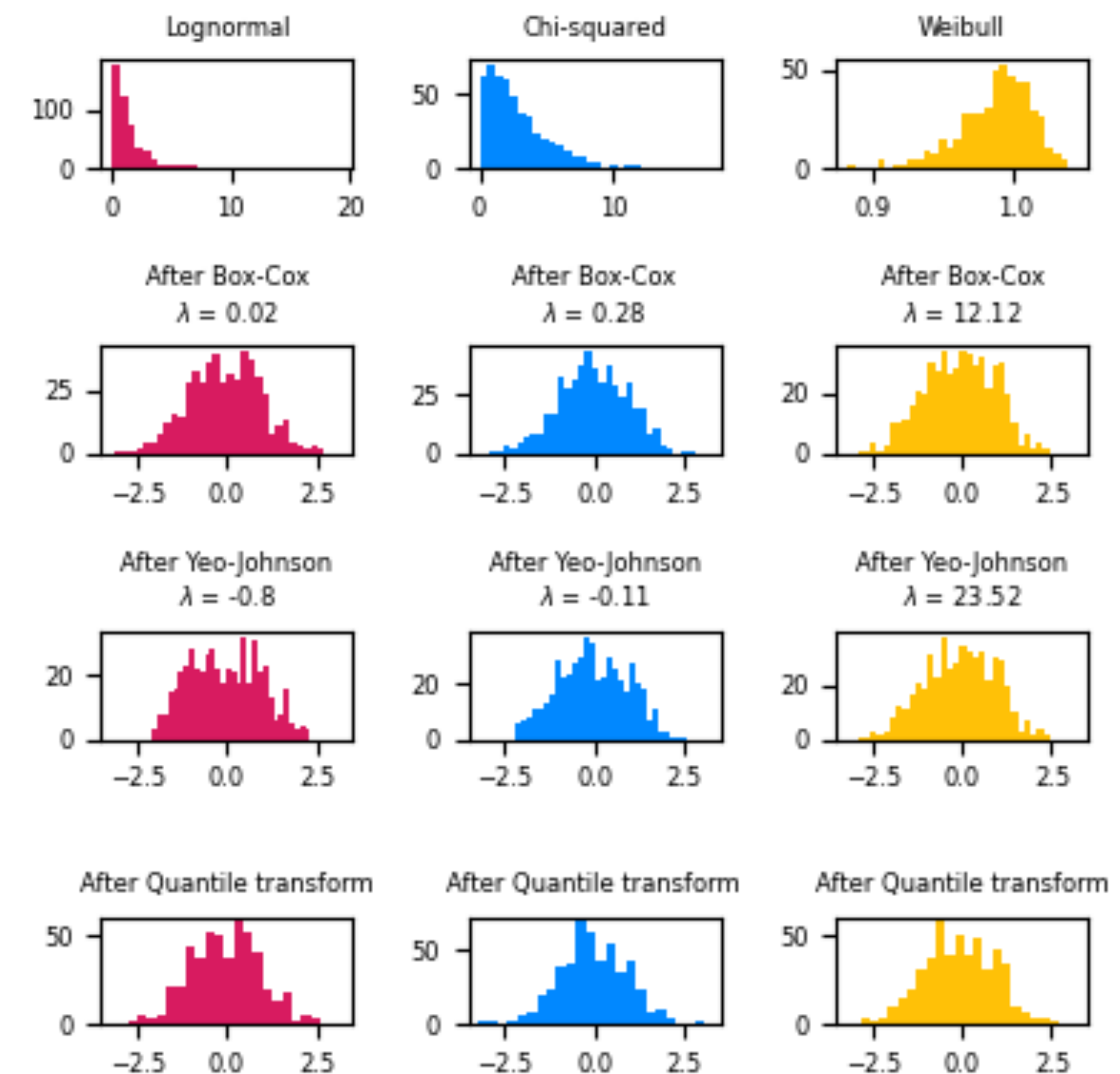
- Standardisierung legt die Verteilung fest: alle numerischen Features haben einen Durchschnitt von 0 und eine Standardabweichung von 1
- Es wird kein Wertebereich für Werte numerischer Features festgelegt
- Viel weniger anfällig für Outlier als Min-Max Skalierung

Welche Art der Skalierung verwendet werden soll, hängt von den Daten, dem Problem und dem Algorithmus ab!

Preprocessing

Ausblick: Nicht-lineare Transformationen

- Häufig setzten Verfahren des maschinellen Lernens zusätzlich voraus, dass Daten einer bestimmten Verteilung folgen.
 - Üblich: Gaußverteilung
- Verteilungen lassen sich mit der Hilfe bestimmter nicht-linearer Transformationen „verschieben“.
- Eine Möglichkeit: Power Transform mit zwei Methoden:
 - Yeo-Johnson: positive und negative Werte
 - Box-Cox: nur strikt positive Werte!



Sollte diese Problem irgendwann Auftreten, hier dran denken ;)

Preprocessing

Encoding kategorischer Werte

- Wie können kategoriale Features verarbeitet werden?
- Wie schon bekannt können die meisten Algorithmen nur mit numerischen Daten umgehen
 - Die ursprünglich erhobenen Daten müssen in einen geeigneten Feature Space transformiert werden
 - Nach Transformation entsteht ein neuer Datensatz X , welcher für das weitere Training verwendet wird
- Hier betrachten wir zwei Möglichkeiten der Variablentransformation: Label Encoding und One-Hot Encoding
- k bezeichne die Anzahl der Ausprägungen eines Feature



→
Extraktion
von
Features

Durchmesser	Farbe	Gewicht	Label
18 cm	rot	190gr	Apfel
16 cm	Grün	250 gr	Birne
...

Obststück 1 = (18cm, Rot, 190gr, Apfel)^T

Obststück 2 = (16cm, Grün, 250gr, Birne)^T

...

Vorgehen:

- Die Ausprägungen eines kategorialen Features (hier Farbe und Label) werden alphabetisch geordnet, der ersten Ausprägung wird eine 0 zugeordnet, der zweiten eine 1 usw.
- „Apfel“ wird demnach durch 0 ersetzt, „Birne“ durch 1
- „Grün“ wird durch 0 ersetzt, „rot“ durch 1

Label Encoding für kategoriale Features



Extraktion
von
Features

Durchmesser	Farbe	Gewicht	Label
18 cm	rot	190gr	Apfel
16 cm	Grün	250 gr	Birne
...

Obststück 1 = (18cm, Rot, 190gr, Apfel)^T

Obststück 2 = (16cm, Grün, 250gr, Birne)^T

...

Transformation mittels Label Encoding

X =

Durchmesser	Farbe	Gewicht	Label
18	1	190gr	0
16	0	250 gr	1
...

$x^{(1)} = (18, 0, 190, 0)^T$

$x^{(2)} = (16, 1, 250, 1)^T$

...

Label Encoding für kategoriale Features

Kleiner Ausschnitt aus dem Adult Data Set (UCI Machine Learning Repository):

Age	Marital status	Workclass	Income
25	Never-married	Private	<=50K
28	Married-civ-spouse	Local-gov	>50K
37	Widowed	Private	<=50K

Person 1 = $(25, \text{Male}, \text{Never-married}, \text{Private}, \leq 50K)^T$

Person 2 = $(28, \text{Male}, \text{Married-civ-spouse}, \text{Local-gov}, > 50K)^T$

Person 3 = $(37, \text{Female}, \text{Widowed}, \text{Private}, \leq 50K)^T$

↓
Transformation mittels Label Encoding

$X =$

Age	Marital status	Workclass	Income
25	1	1	0
28	0	0	1
37	2	1	0

$x^{(1)} = (25, 1, 1, 0)^T$

$x^{(2)} = (28, 0, 0, 1)^T$

$x^{(3)} = (37, 2, 1, 0)^T$

Label Encoding für kategoriale Features

Kleiner Ausschnitt aus dem Adult Data Set (UCI Machine Learning Repository):

Age	Marital status	Workclass	Income
25	Never-married	Private	<=50K
28	Married-civ-spouse	Local-gov	>50K
37	Widowed	Private	<=50K

Person 1 = (25, Male, Never-married, Private, <=50K)^T

Person 2 = (28, Male, Married-civ-spouse, Local-gov, >50K)^T

Person 3 = (37, Female, Widowed, Private, <=50K)^T

Transformation mittels Label Encoding

$X =$

Age	Marital status
25	1
28	0
37	2

Was ist das Problem hier?

- Label Encoding impliziert bei mehr als zwei Ausprägungen eine metrische Skalierung des Features
 - Metrische Skalierung = Es lassen sich Abstände zwischen den Ausprägungen messen
 - Abstand zwischen „Never-married“ und „Married-civ-spouse“ ist im Beispiel 1
 - Abstand zwischen „Widowed“ und „Married-civ-spouse“ ist im Beispiel 2
- Diese Skalierung existiert in der Realität i.R. nicht!
 - Es wird eine Information kodiert, welche falsch ist
 - Da viele ML-Verfahren über Distanzen arbeiten, entsteht hierdurch ein Problem für die weitere Verarbeitung
- **Lösung: One-Hot Encoding**



Kleiner Ausschnitt aus dem Adult Data Set (UCI Machine Learning Repository):

Age	Marital status	Workclass	Income
25	Never-married	Private	<=50K
28	Married-civ-spouse	Local-gov	>50K
37	Widowed	Private	<=50K

Person 1 = (25, Male, Never-married, Private, <=50K)^T

Person 2 = (28, Male, Married-civ-spouse, Local-gov, >50K)^T

Person 3 = (37, Female, Widowed, Private, <=50K)^T

Vorgehen:

- Für jede Ausprägung eines Features wird ein eigenes Feature erstellt
 - Dieses ist 1 wenn die Ausprägung vorhanden war, sonst 0
- In diesem Fall wird der Datensatz um die Features „Never-married“, „Married-civ-spouse“ und „Widowed“ ergänzt
- Die resultierenden Features sind binär
- Das ursprüngliche Feature kann entfernt werden, es ist bereits über die neuen Features abgebildet

One-Hot Encoding für kategoriale Features

Kleiner Ausschnitt aus dem Adult Data Set (UCI Machine Learning Repository):

Age	Marital status	Workclass	Income
25	Never-married	Private	<=50K
28	Married-civ-spouse	Local-gov	>50K
37	Widowed	Private	<=50K

Person 1 = (25, Male, Never-married, Private, <=50K)^T

Person 2 = (28, Male, Married-civ-spouse, Local-gov, >50K)^T

Person 3 = (37, Female, Widowed, Private, <=50K)^T

Transformation mittels One-Hot Encoding

Age	Never-married?	Married-civ-spouse?	Widowed?	Private?	Local-gov?	<=50K?	>50K?
25	1	0	0	1	0	1	0
28	0	1	0	0	1	0	1
37	0	0	1	1	0	1	0


$x^{(1)} = (25, 1, 1, 0, 0, 1, 0, 1, 0)^T$

$x^{(2)} = (28, 1, 0, 1, 0, 0, 1, 0, 1)^T$

$x^{(3)} = (37, 0, 0, 0, 1, 1, 0, 1, 0)^T$

X =

Label Encoding für kategoriale Features

- Zuordnung von Zahlen 0 bis $k-1$ zu Ausprägungen in alphabetischer Reihenfolge: Zuordnung von 0 der ersten Ausprägung, von 1 der zweiten usw.
- Problem: bei $k > 2$ Ordnung der Ausprägungen und Abstände zwischen Ausprägungen 
→ Bei mehr als zwei Ausprägungen ist Label Encoding i.N. nicht geeignet
- Anzahl der Features (Spalten) im transformierten Datensatz X = Anzahl ursprünglicher Features

One-Hot Encoding für kategoriale Features

- $k=2$ Ausprägungen a_1, a_2 : eine Spalte mit $x_{ij} = 1$, falls der ursprüngliche Eintrag a_1 lautet, 0 sonst
- $k > 2$ Ausprägungen: eine Spalte pro Ausprägung, wobei in jeder Spalte $x_{ij} = 1$ für genau ein $i \in \{1, \dots, m\}$ gilt, 0 sonst
→ Aus einer entstehen k Spalten
- Anzahl der Features (Spalten) im transformierten Datensatz X kann (sehr) hoch werden

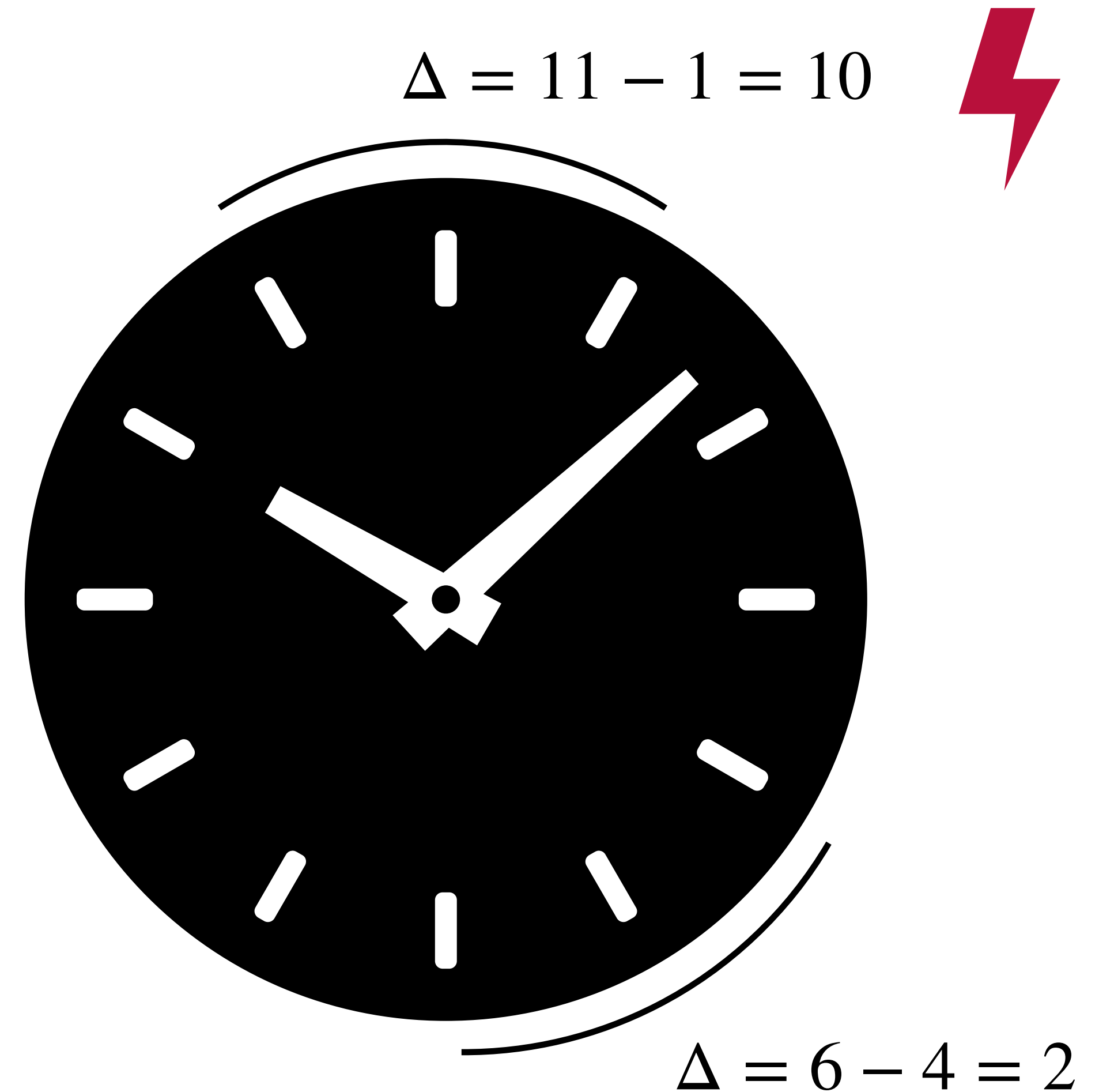
Kategoriales Feature mit $k > 2$ Ausprägungen \Rightarrow One-Hot Encoding

Kategoriales Feature haben $k=2$ Ausprägungen \Rightarrow Label Encoding

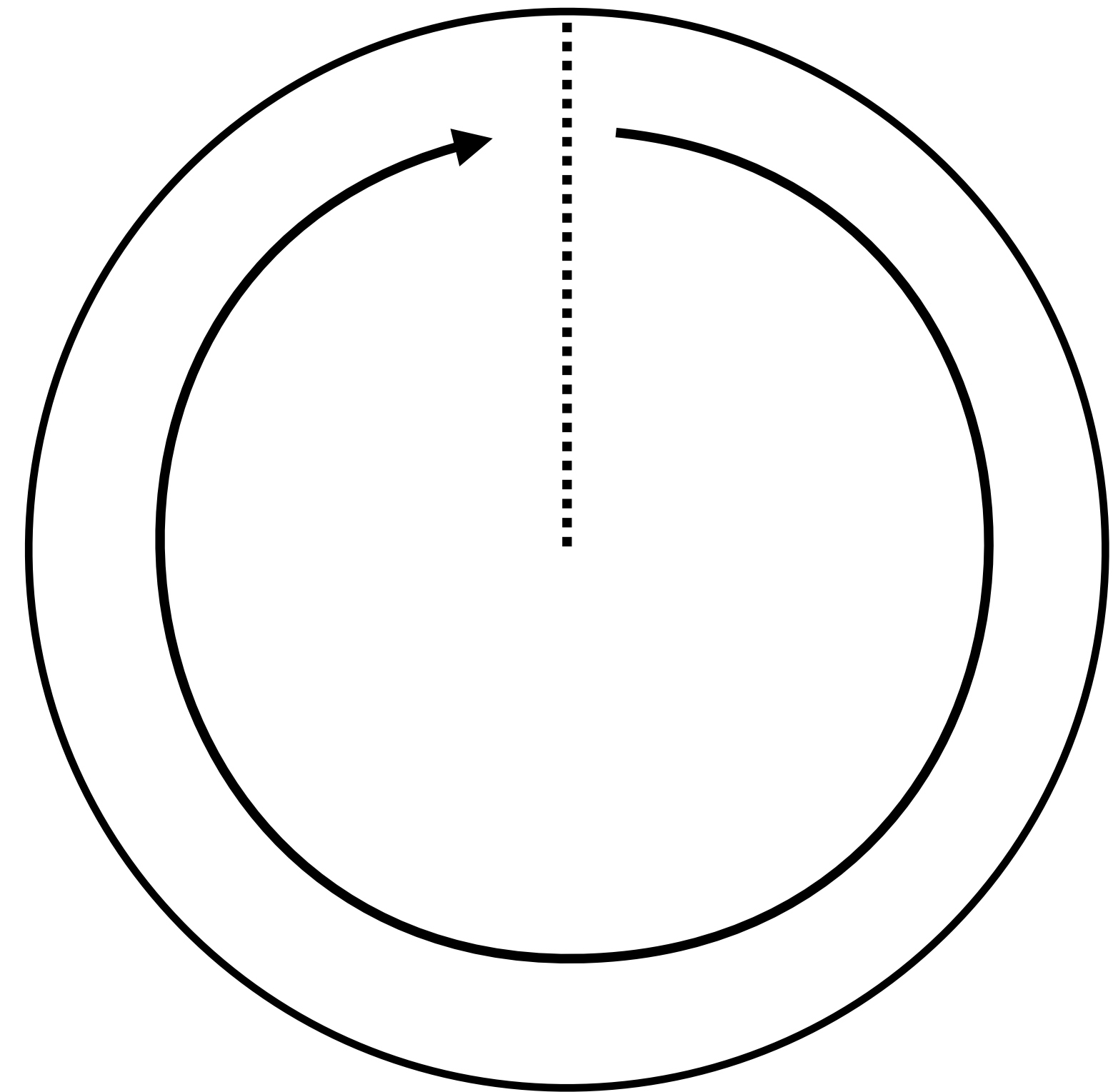
Preprocessing

Encoding von Uhrzeiten, Koordinaten,...

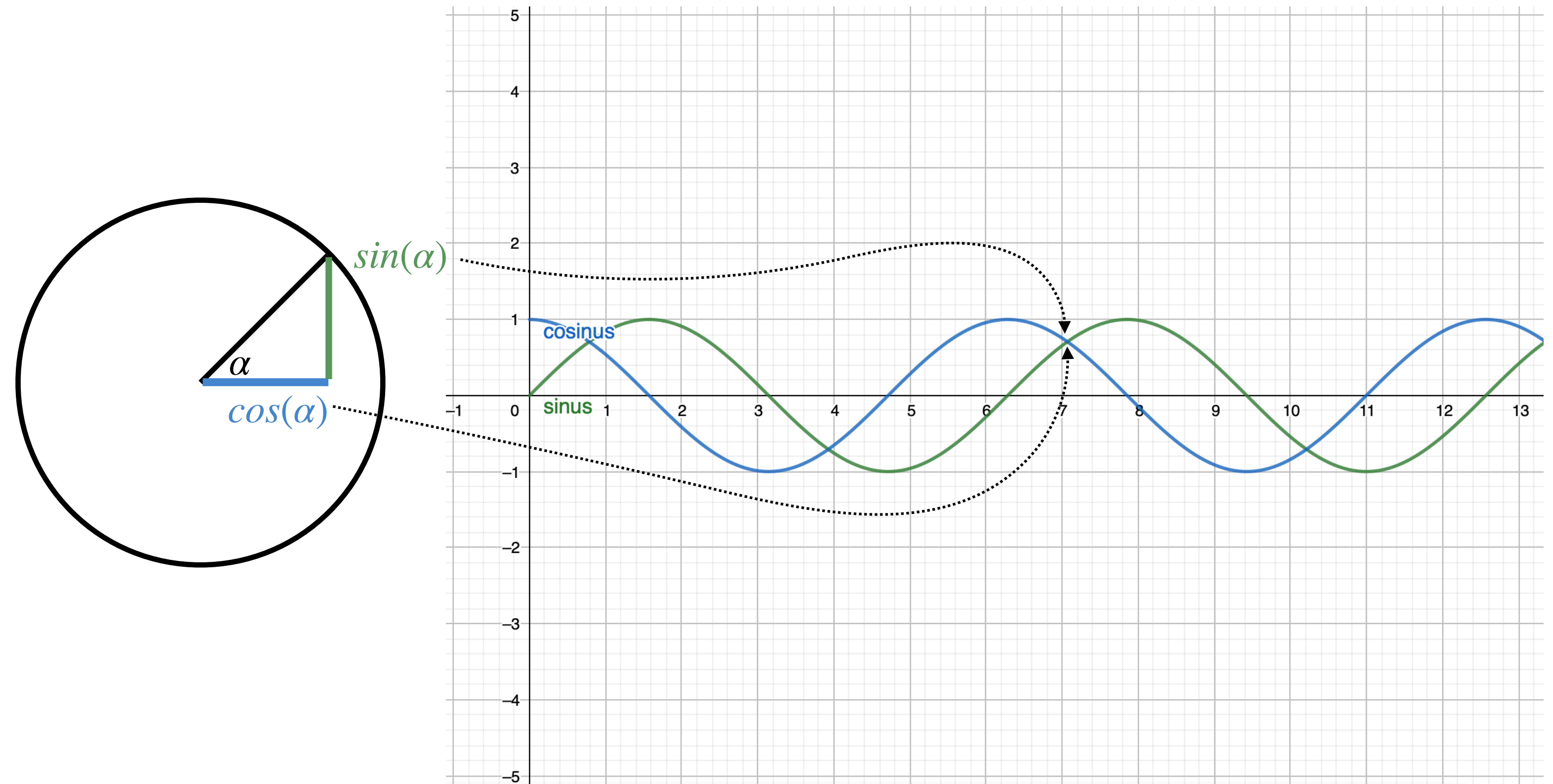
- Uhrzeiten, Monatsangaben, etc. sind eine besondere Herausforderungen für die meisten ML-Verfahren
- Problem:
 - Es gibt immer einen Episodenwechsel, welche über einfache Distanzmetriken nicht abgebildet wird
 - Fehlerfunktionen können dadurch unverhältnismäßig groß ausfallen
- Beispiel:
 - Label=6 Uhr, Vorhersage=4 Uhr
 - Distanz= 2
 - Label=11 Uhr, Vorhersage=1 Uhr
 - Vermeintliche Distanz=10
 - In der Folge findet eine übergroße Anpassung des Modells statt
 - Der Lernerfolg wird i.R. gemindert oder verhindert



- Weitere Bereiche in denen dieses Problem auftritt:
 - Koordinaten
 - Winkelangaben
 - z.B. Stellung von Windrädern, Reifen
 - Angaben von Wochentagen
 - Weitere Saisonale und zyklische Werte
 - Saisonangaben
 - Tageszeit
 - Umlaufbahnen
 - ...



- Lösung: Umwandlung in Kreiskoordinaten
- Angaben wie z.B. eine Uhrzeit sind im Prinzip eine Winkelangabe α in einem Einheitskreis.
- Jeder Winkel α kann auch als Koordinate des Punktes im Einheitskreis angegeben werden
 - Hier gilt: $x = \cos(\alpha)$ und $y = \sin(\alpha)$
- Sinus und Cosinus sind fortlaufend und stetig definiert
 - Durch die Stetigkeit der Funktion gibt es nun keinen „Bruch“ mehr in der Kodierung der Zeitangabe, Koordinate, ...



Beispiel 1: Distanz zwischen 11 Uhr und 1 Uhr

Encoding für 11 Uhr:

$$x_{11} = \cos(11\frac{2\pi}{12}) = 0.866 \text{ und } y_{11} = \sin(11\frac{2\pi}{12}) = -0.5$$

Encoding für 1 Uhr:

$$x_1 = \cos(\frac{2\pi}{12}) = 0.866 \text{ und } y_1 = \sin(\frac{2\pi}{12}) = 0.5$$

Nun kann z.B. der MSE bestimmt werden:

$$L = \frac{(.866 - .866)^2 + (-.5 - .5)^2}{2} = .5$$

Beispiel 2: Distanz zwischen 6 Uhr und 4 Uhr

Encoding für 6 Uhr:

$$x_{11} = \cos(6\frac{2\pi}{12}) = -1 \text{ und } y_{11} = \sin(6\frac{2\pi}{12}) = 0$$

Encoding für 4 Uhr:

$$x_1 = \cos(4\frac{2\pi}{12}) = -0.5 \text{ und } y_1 = \sin(4\frac{2\pi}{12}) = 0.866$$

Nun kann z.B. der MSE bestimmt werden:

$$L = \frac{(-1 + .5)^2 + (0 - .866)^2}{2} = .5$$

Gleicher zeitlicher Abstand = Gleiche Distanz



Evaluierung

Klassifikation

- Accuracy ist einer der am meisten verwendeten Metriken für Klassifikationsprobleme
- Zielfrage: Mit welcher Wahrscheinlichkeit ist das Ergebnis korrekt?
 - Korrekt sind zwei Fälle: TP und TN
 - Falsch sind zwei Fälle: FN und FP

$$\text{Accuracy} = \frac{\sum TP + \sum TN}{\sum TP + \sum TN + \sum FP + \sum FN}$$

		Label	
		True	False
Model	True	True positive (TP)	False positive (FP)
	False	False negative (FN)	True negative (TN)

- $$\text{Accuracy} = \frac{\sum TP + \sum TN}{\sum TP + \sum TN + \sum FP + \sum FN}$$
- Achtung bei Datensets mit ungleicher Verteilung der verschiedenen Klassen
- Beispiel:
 - 1 von 100 getesteten Personen ist Corona-positiv
 - Das Modell soll den Status anhand der Haarfarbe vorhersagen
 - Erzielbares Ergebnis, wenn immer „negativ“ vorhergesagt wird: 99% Accuracy
- Die Angabe & Interpretation der Accuracy benötigt immer auch einen Blick auf die eigentlichen Daten

		Label	
		True	False
Model	True	True positive (TP)	False positive (FP)
	False	False negative (FN)	True negative (TN)

- Die Precision gibt an mit welcher Wahrscheinlichkeit ein als positiv vorhergesagtes Ergebnis auch wirklich positiv ist.

- $$\text{Precision} = \frac{\sum TP}{\sum TP + FP}$$

- Eine niedrige Precision bedeutet, dass viele False Positives zu erwarten sind.
- Fachliche Beurteilung notwendig:
 - Suchmaschine: Teilweise falsche Ergebnisse können akzeptabel sein
 - Erkennung von Betrugsfällen: Sehr niedrige Precision möglicherweise unakzeptabel, da Kosten für die Nachverfolgung irgendwann den Nutzen übersteigen

		Label	
		True	False
Model	True	True positive (TP)	False positive (FP)
	False	False negative (FN)	True negative (TN)

- Der Recall gibt an mit welcher Wahrscheinlichkeit ein echt-positiver Wert auch tatsächlich als positiv erkannt wird.

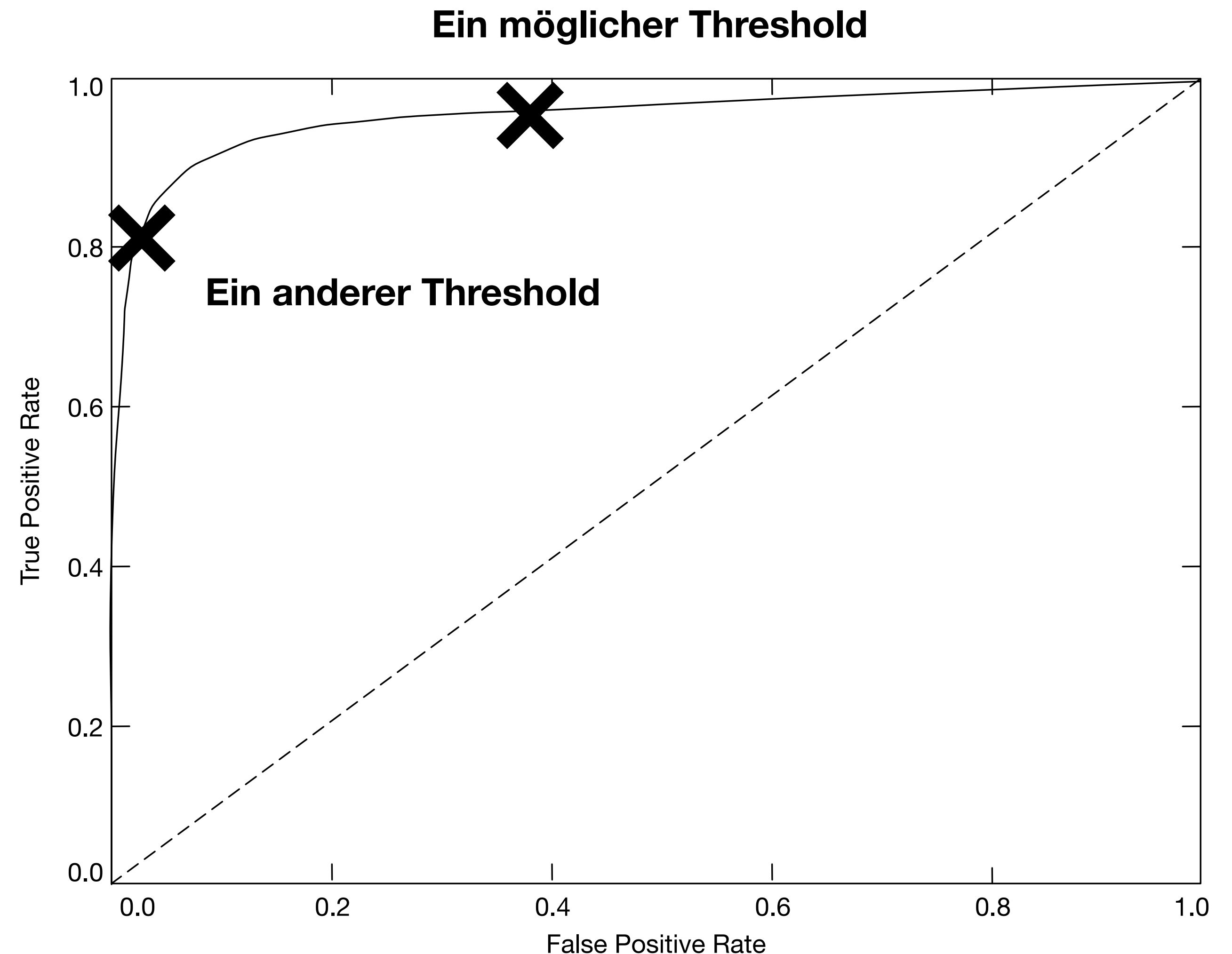
- $$\text{Recall} = \frac{\sum TP}{\sum TP + FN}$$

- Ein niedriger Recall bedeutet, dass viele Fälle nicht erkannt werden
- Fachliche Beurteilung notwendig.
 - Precision & Recall sind häufig eine Abwägung, welche vom konkreten Anwendungsfall abhängig sind.

		Label	
		True	False
Model	True	True positive (TP)	False positive (FP)
	False	False negative (FN)	True negative (TN)

- Precision & Recall sollten immer in Kombination betrachtet werden
- Der F1-Score kombiniert beider Metriken und ist der harmonische Mittelwert beider Werte
- $$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$
- Der harmonische Mittelwert ist weniger anfällig für Extremwerte als der arithmetische Mittelwert
 - Der F1-Score berücksichtigt dadurch False Positives und False Negatives stärker als die Accuracy
- Durch die Berücksichtigung von Precision & Recall zu jeweils gleichen Teilen eignet sich der F1-Score auch für unbalancierte Datensätze
 - Accuracy verliert hier zunehmend an Aussagekraft (siehe Beispiel zur Folie „Accuracy“)

- In binären Klassifikationsproblemen hängt das Verhältnis von True Positives und False Positives von der Wahl des Thresholds ab
- Üblich ist ein Threshold von 0.5, jeder andere Wert ist aber auch möglich!
- Die ROC-Kurve bildet den Verlauf bei beliebigen verschiedenen Thresholds ab
- Es gibt dabei immer einen Threshold mit keinen False Positives und einen Threshold mit keinen True Positives!
- Eine zufallsbasiertes Modell erzeugt dabei die gestrichelte Linie.
- Im Idealfall steigt die Kurve bereits zu Beginn möglichst steil an.
- Basierend auf der ROC-Kurve kann die Area under the Curve (AUC) gemessen werden. Bei einem Zufallsmodell ist diese 0.5 (gestrichelte Linie). Im Idealfall ist diese möglichst nahe bei 1.



Evaluierung

Regression

- Für das Training von Modelle haben wir bereits MSE als Fehlerfunktion kennengelernt
- Für die Auswertung ist der MSE oft schwierig, da der konkrete Wert durch das Quadrat der Residuen schwer interpretierbar ist
- Besser geeignet ist der Mean Absolute Error (MAE)

$$\bullet MAE = \frac{\sum_{i=1}^n |f(x) - y)_i|}{n}$$

- Aussage: Wieviel weicht das Modell im Durchschnitt ab

- Eine Erweiterung des MAE ist der Mean Average Percentage Error (MAPE)

- $$MAPE = \frac{100}{n} \sum_{i=1}^n \left| \frac{f(x) - y)_i}{y} \right|$$

- Der MAPE gewichtet den Fehler prozentual zum tatsächlichen Wert
- Der MAPE kann in besonderen Anwendungsfällen sinnvoll sein. Z.B.:
 - Distanzmessung für autonome Fahrzeuge: Bei sehr nahen Distanzen ist ein Fehler dramatisch, bei größeren Distanzen immer mehr tolerierbar.
- Es gilt wie immer: Anwendungsfall betrachten & gute Kenntnisse, Verständnis der Metriken wird benötigt