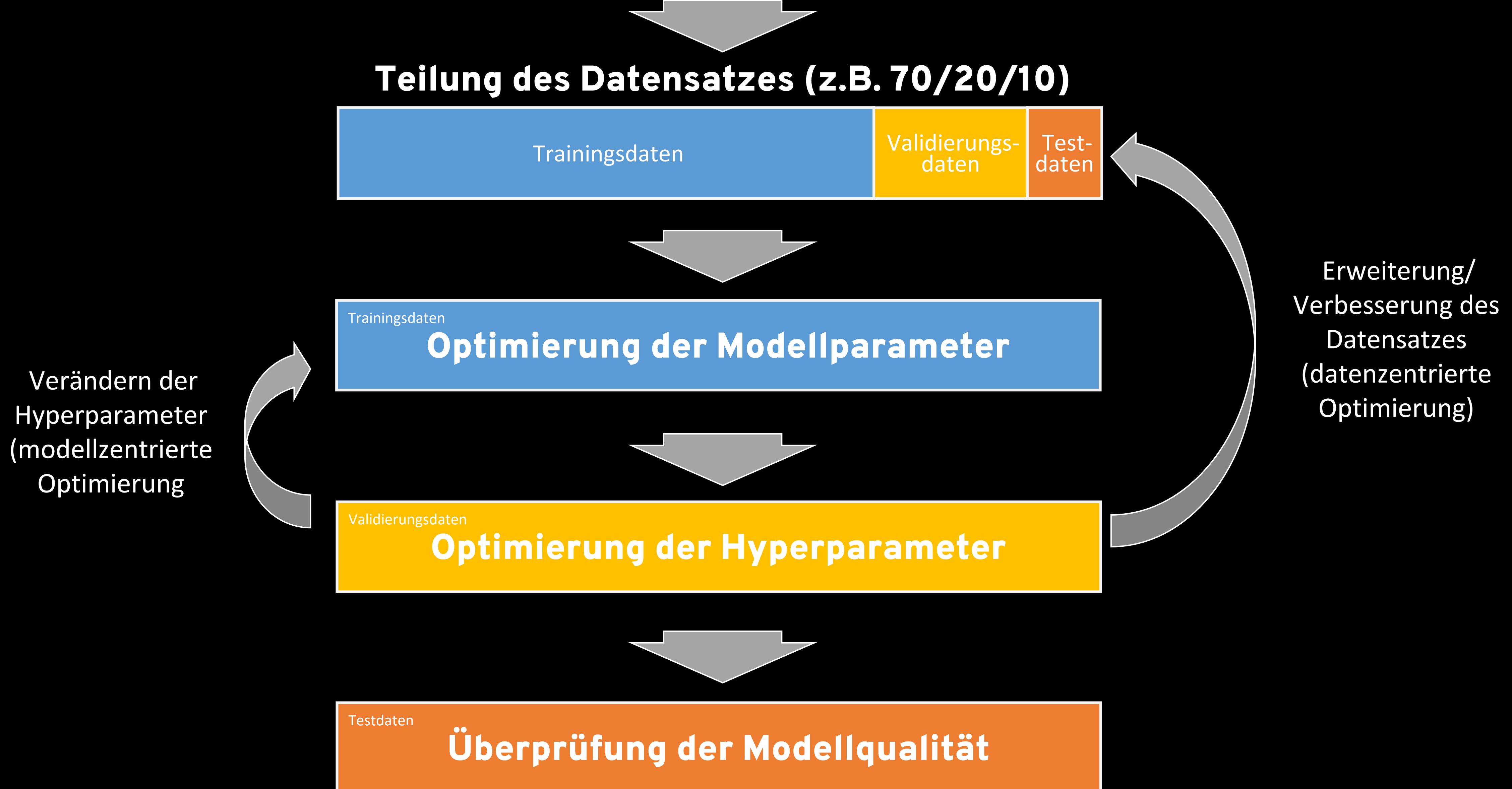


Einführung in Data Science und maschinelles Lernen

OVERFITTING UND MODELL-EVALUATION

- **Wiederholung wichtiger Begriffe des ML**
- **Interaktionseffekte**
- **Overfitting**
- **Regularisierung**
- **Modellgütekriterien**
- **Einführung in neuronale Netze**

Wahl eines Prognosemodells



BREAKOUT

Ihr wollt Euer Modell nutzen, um eine Vorhersage für den Zeitraum vom 01.08.2018 bis zum 31.07.2019 erstellen.

**Wie muss Euer Datensatz für diese Vorhersage aussehen
– welche Variablen muss er enthalten?**

OPTIMIERUNG DER MODELLPARAMETER

Forward Propagation

- Berechnung des vorhergesagten Wertes auf Basis der aktuellen Modellgleichung
- Berechnung der Kosten bzw. des Loss

Backward Propagation

- Berechnung des Gradienten (d.h. aller partiellen Ableitungen), um die Richtung des Minimums zu bestimmen.
- Anpassung der Modellparameter im Ausmaß der definierten Lernrate:

$$\text{Neuer Wert} = \text{Alter Wert} - \text{Lernrate} \times \text{Partieller Gradient}$$

MODELLPARAMETER VS. HYPERPARAMETER

Modellparameter

- **Teile des Modells, die sich während des Trainingsprozesses automatisch anpassen, wie Gewichte und Biases**

Hyperparameter

- **Einstellungen, die du vor dem Training festlegst, wie etwa die Lernrate oder die Anzahl der Modell-Parameter.**

FEATURES AND LABELS

Machine Learning

Feature

Label

Statistik

Unabhängige Variable

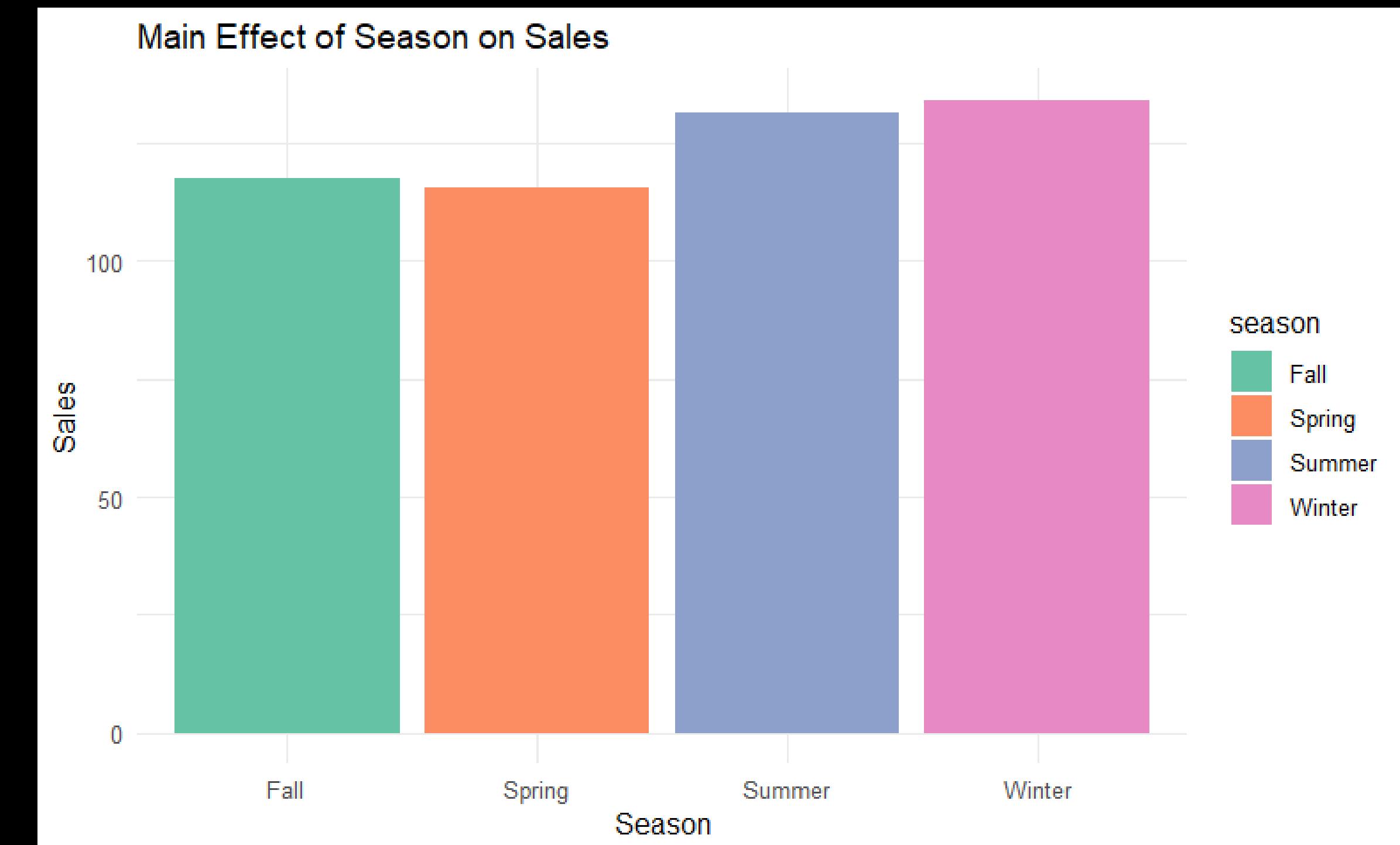
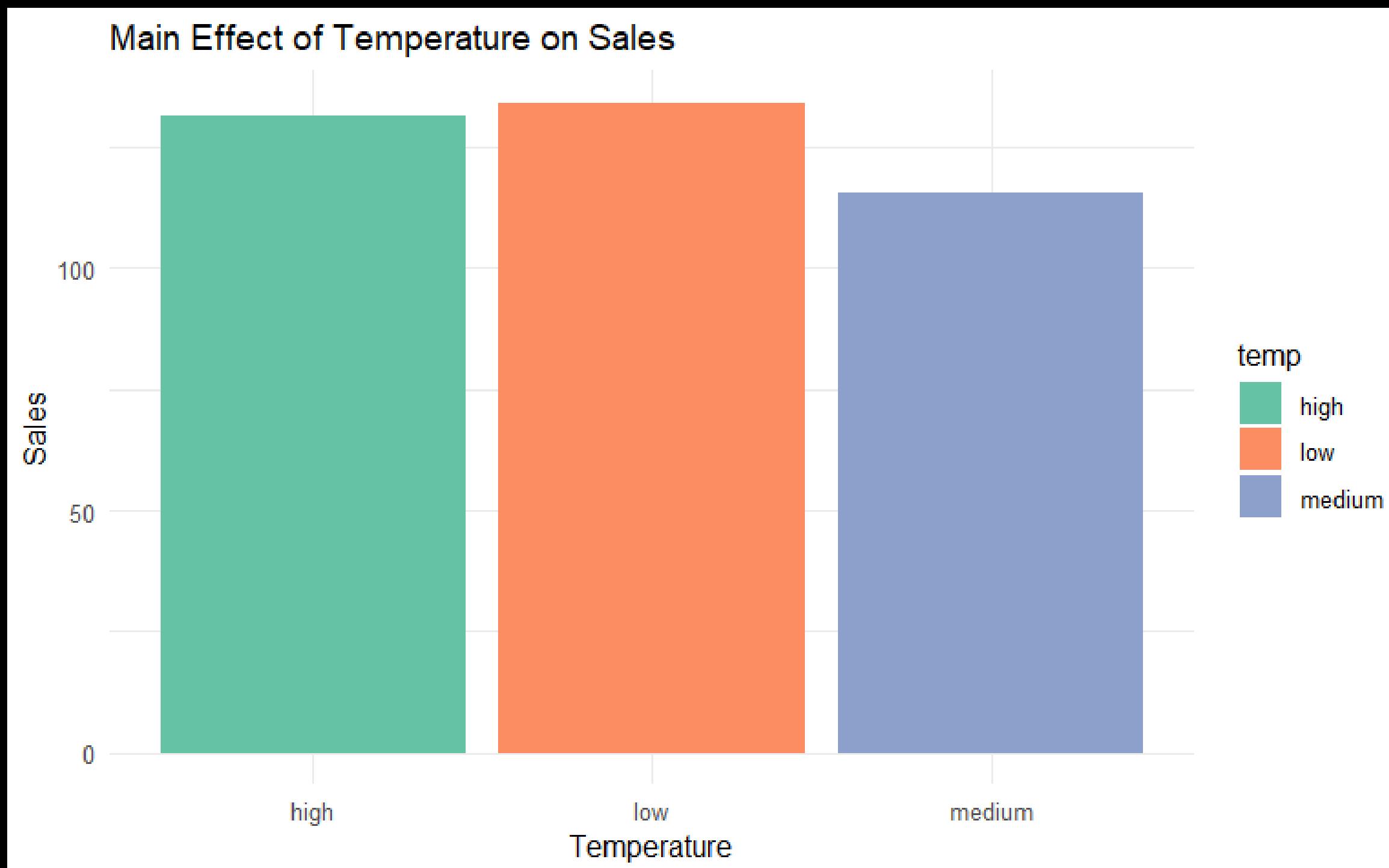
Abhängige Variable

Beschreibung

**Werte, die zur Vorhersage
genutzt werden**

Vorherzusagende Werte

HAUPTEFFEKTE



```
```{r}
Linear model without interaction
model <- lm(sales ~ season + temp, data = df)
summary(model)
```

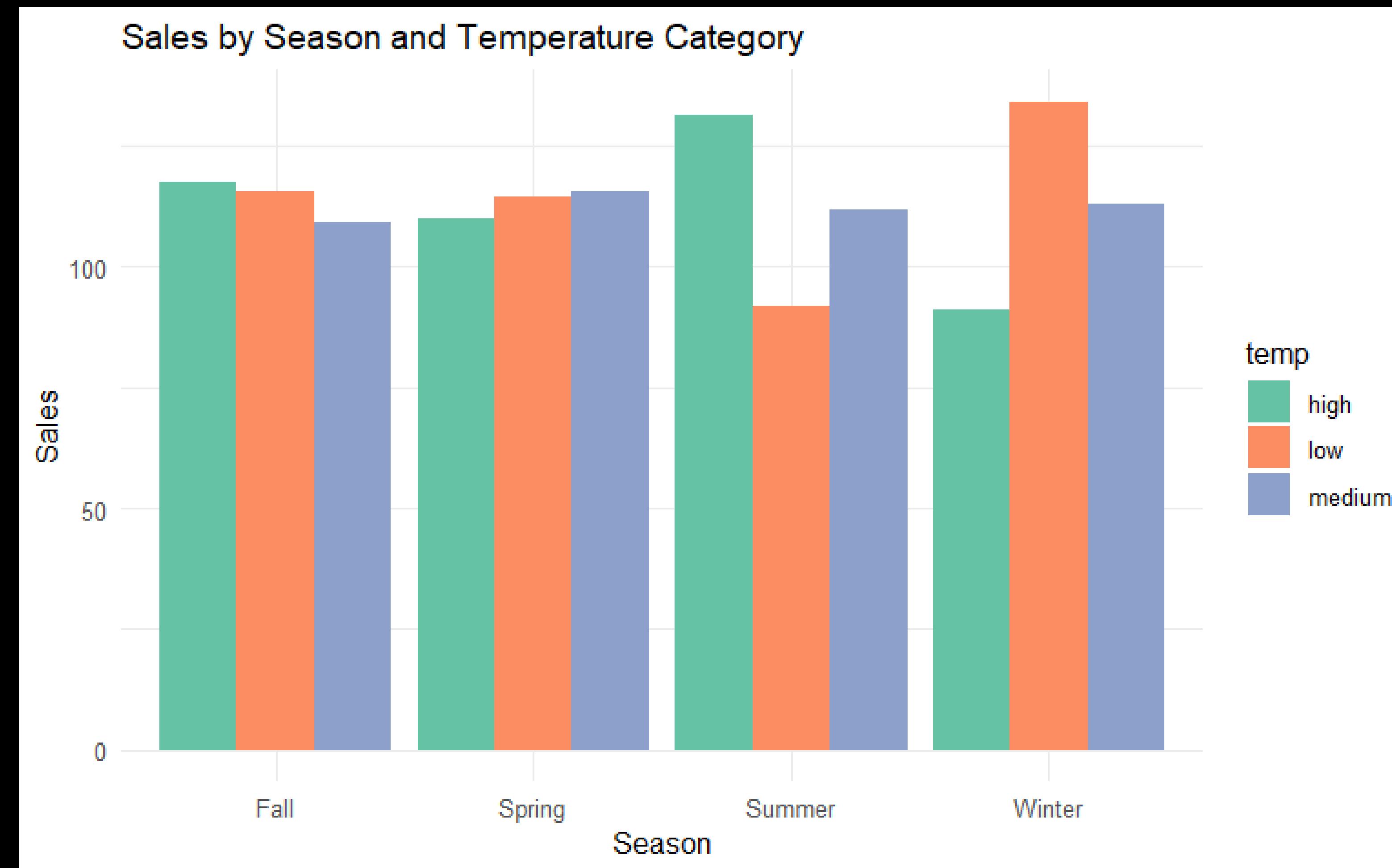
```
```
Call:
lm(formula = sales ~ season + temp, data = df)

Residuals:
    Min      1Q  Median      3Q     Max 
-32.547 -6.223 -0.069  5.012 35.928 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 100.57903   1.02571  98.058   <2e-16 ***
seasonSpring  0.02688   1.13877   0.024    0.981    
seasonSummer -1.43607   1.14434  -1.255    0.210    
seasonWinter -1.23872   1.15230  -1.075    0.283    
templow       -1.23032   0.98948  -1.243    0.214    
tempmedium    -0.37754   0.99002  -0.381    0.703    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 12.66 on 994 degrees of freedom
Multiple R-squared:  0.004561, Adjusted R-squared:  -0.0004467 
F-statistic: 0.9108 on 5 and 994 DF,  p-value: 0.4732
```

INTERAKTIONSEFFEKT



```
```{r}
Linear model with interaction
model <- lm(sales ~ season + temp + season * temp, data = df)
summary(model)

```
Call:
lm(formula = sales ~ season + temp + season * temp, data = df)

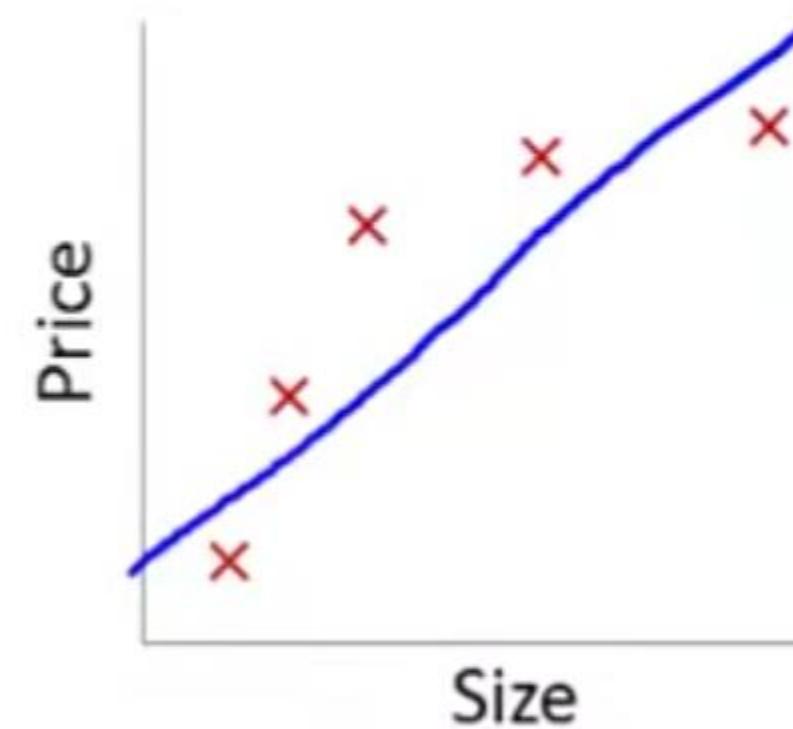
Residuals:
    Min      1Q  Median      3Q     Max 
-15.8497 -3.2951  0.1132  3.2120 16.9311 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 100.7683   0.6074 165.892 <2e-16 ***  
seasonSpring -0.6269   0.8220 -0.763  0.4458    
seasonSummer  20.2747   0.8364  24.240 <2e-16 ***  
seasonWinter -20.7398   0.8155 -25.433 <2e-16 ***  
tempLow       -0.6637   0.8314 -0.798  0.4249    
tempMedium    -1.3939   0.8113 -1.718  0.0861 .    
seasonSpring:tempLow 1.2527   1.1304  1.108  0.2681    
seasonSummer:tempLow -39.6853   1.1276 -35.194 <2e-16 ***  
seasonWinter:tempLow  40.5964   1.1580  35.058 <2e-16 ***  
seasonSpring:tempMedium 0.4956   1.1230  0.441  0.6591    
seasonSummer:tempMedium -19.9748   1.1487 -17.389 <2e-16 ***  
seasonWinter:tempMedium  21.3007   1.1123  19.150 <2e-16 ***  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

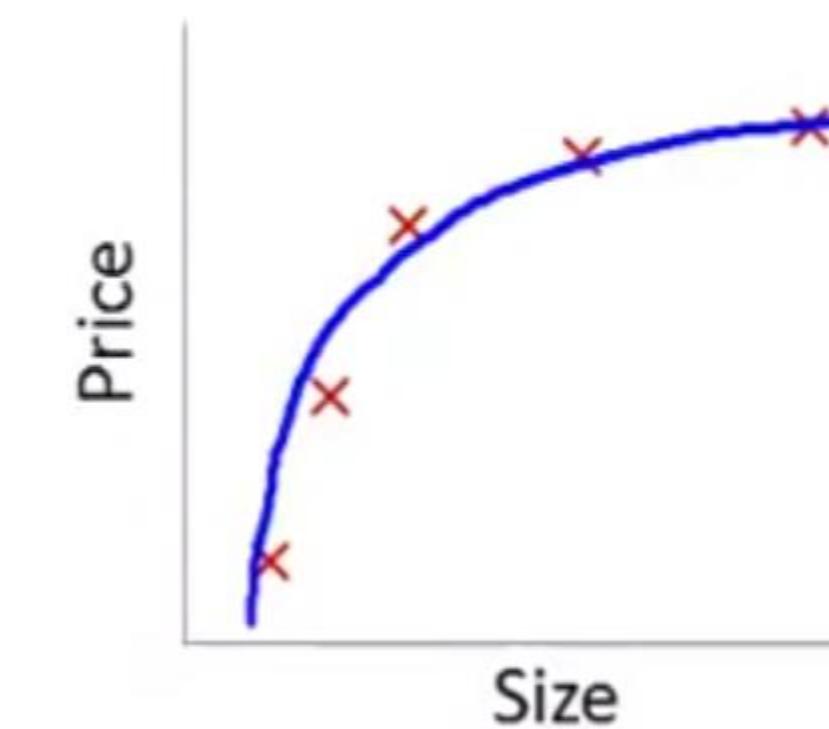
Residual standard error: 5.046 on 988 degrees of freedom
Multiple R-squared:  0.8428,    Adjusted R-squared:  0.8411 
F-statistic: 481.6 on 11 and 988 DF,  p-value: < 2.2e-16
```

OVERFITTING

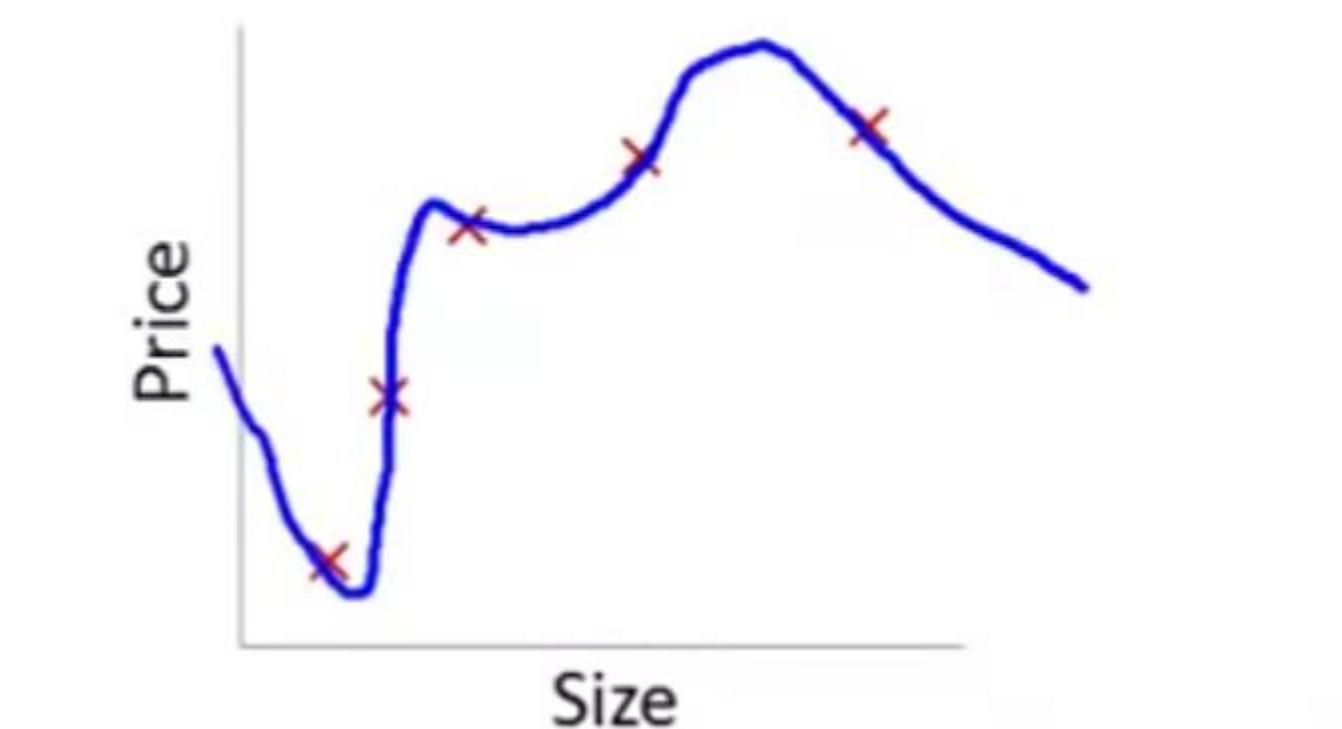
Example: Linear regression (housing prices)



$\rightarrow \theta_0 + \theta_1 x$
"Underfit" "High bias"



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$
"Just right"



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$
"Overfit" "High variance"

Overfitting: If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fail to generalize to new examples (predict prices on new examples).

STRATEGIEN ZUR VERMEIDUNG VON OVERFITTING

Options:

1. Reduce number of features.
 - — Manually select which features to keep.
 - — Model selection algorithm

2. Regularization.
 - — Keep all the features, but reduce magnitude/values of parameters θ_j .
 - Works well when we have a lot of features, each of which contributes a bit to predicting y .

REGULARISIERUNG

„Bestrafen“ der Verwendung von Variableninformation im Rahmen der Kostenfunktion

Lineares Modell mit mehreren Variablen x_1, x_2 und vielen möglichen weiteren:

$$h_x(\theta_0, \theta_1, \theta_2, \dots) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots$$

(mit $\theta_0, \theta_1, \theta_2, \dots$ als den zu schätzenden Modellparametern)

Kostenfunktion mit Regularisierung:

$$J_x(\theta_0, \theta_1, \theta_2, \dots) = \frac{1}{m} \left[\sum_m (h_x(\theta_0, \theta_1, \theta_2, \dots) - y)^2 + \lambda(|\theta_0| + |\theta_1| + |\theta_2| + \dots) \right]$$

(mit λ als Regularisierungsparameter)

BEISPIEL ZU OVERFITTING

```
1 * ---
2   title: "Linear Regression"
3   output: html_notebook
4 *
5
6 * ````{r}
7 # Importing Function Packages
8 library(dplyr)
9 library(readr)
10 library(lubridate)
11 library(broom)
12 library(Metrics)
13 ```
14
15
16 * ````{r}
17 # Importing Training and Test Data
18 house_pricing_train <- read_csv("./house_pricing_data/house_pricing_train.csv")
19 house_pricing_test <- read_csv("./house_pricing_data/house_pricing_test.csv")
20 ```
21
22
23 * ````{r}
24 # Estimating (Training) Models
25 mod1 <- lm(price ~ bathrooms, house_pricing_train)
26 mod2 <- lm(price ~ as.factor(bathrooms), house_pricing_train)
27 mod3 <- lm(price ~ as.factor(bathrooms) + as.factor(zipcode), house_pricing_train)
28 mod4 <- lm(price ~ as.factor(bathrooms) + as.factor(zipcode) + condition, house_pricing_train)
29 mod5 <- lm(price ~ as.factor(bathrooms) + as.factor(zipcode) + as.factor(condition), house_pricing_train)
30 mod6 <- lm(price ~ as.factor(bathrooms) + as.factor(zipcode) + as.factor(condition) + sqft_living15, house_pricing_train)
31 mod7 <- lm(price ~ as.factor(bathrooms) + as.factor(zipcode) + as.factor(condition) + sqft_living15 + floors + view + grade +
32   as.factor(zipcode)*as.factor(bathrooms), house_pricing_train)
33
34
35 * ````{r}
36 summary(mod1)
```

GENERALIZED LINEAR MODEL

```
```{r}
library(glmnet)

Reformat features and label from tibble to matrices as expected by glmnet
train_features_matrix <- as.matrix(train_features)
train_label_matrix <- as.matrix(train_label)

Calibration of linear regressions with regularisation
mod1 <- glmnet(x=train_features_matrix, y=train_label_matrix, lambda=.001)
mod1

Calibration of linear regressions with regularisation
mod2 <- glmnet(x=train_features_matrix, y=train_label_matrix, lambda=1000)
mod2
```

```
```
```

BATCHES, STEPS UND EPOCHE

Batch

- **Die Gesamtmenge an Trainingsdaten wird in separate Teilgruppen mit gleicher Größe eingeteilt.**
- **Standardgröße eines Batchs ist 32.**

Step

- **Die Backward-Propagation mit einem Batch (alle Gewichte werden einmal optimiert)**

Epoche

- **Optimierung des Modells mit allen Batches des Trainingsdatensatzes**
- **Je nach Modell genügen sehr wenige Epochen oder sind mehrere hundert oder tausend Epochen notwendig zur Optimierung.**

MODELLGÜTEKRITERIEN FÜR REGRESSIONSAUFGABEN

errors: **forecast - actual** (auch: residuals)

mae: **mean(abs(errors))**

mape: **mean(abs(errors/actual))**

mse: **mean(errors^2)**

rmse: **sqrt(mean(errors^2))**

rse: **sum(errors^2) / sum((actual-mean(actual))^2)**

$r^2 = 1 - rse$

Video (3 Minuten) mit Erklärung und Darstellung der Kriterien:

<https://www.coursera.org/lecture/machine-learning-with-python/evaluation-metrics-in-regression-models-5SxtZ>

MODELLGÜTEKRITERIEN FÜR KLASSEIFIKATIONSAUFGABEN

$$\text{Accuracy} = \frac{\text{Anzahl richtiger Vorhersagen}}{\text{Gesamtzahl der Vorhersagen}}$$

$$\text{Precision} = \frac{\text{Anzahl richtiger positiver Vorhersagen}}{\text{Gesamtzahl der Vorhersagen}}$$

$$\text{Recall} = \frac{\text{Anzahl richtiger positiver Vorhersagen}}{\text{Anzahl tatsächlich positiver Fälle}}$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

| | | Tatsächliche Beobachtungen | |
|-------------|----------------|----------------------------|----------------|
| | | <i>positiv</i> | <i>negativ</i> |
| Vorhersagen | <i>positiv</i> | true | false |
| | <i>negativ</i> | false | true |

Blog mit genauerer Erklärung der Metriken:

<https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec>

TESTEN DER VORHERSAGEN AUF KAGGLE

The screenshot shows the Kaggle interface for a competition titled "Bakery Sales Prediction".

Left Sidebar:

- Search bar: Search
- User profile: STEFFEN · COMMUNITY PREDICTION COMPETITION · PRIVATE · A MONTH TO GO
- Navigation menu:
 - Create
 - Home
 - Competitions
 - Datasets
 - Models
 - Code
 - Discussions
 - Learn
 - More
- Your Work: Bakery Sales Predict...
- VIEWED:
 - Bakery Sales Predict...
 - Switching from a Go...
 - Kaggle Community ...
 - Datasets
 - Competitions
- View Active Events

Top Right Actions:

- Submit Prediction
- ...

Section Headers:

- Bakery Sales Prediction**
- Dataset Description**
- File Descriptions and Data Field Information**

Dataset Details:

- Files**: 5 files
- Size**: 477.34 kB
- Type**: CSV
- License**: Subject to Competition Rules

Description:

In this competition, you will predict sales for six product categories of a bakery branch located somewhere in Kiel. The training data includes dates, Product categories, and sales numbers. Additional files include supplementary information that may be useful in building your models.

File Descriptions and Data Field Information:

train.csv

The training data, comprising daily time series data with the sales (`Umsatz`) for each product category (`Warengruppe`).

- `id` is a unique identifier for the recorded sales.
- `Datum` gives the date of the recorded sales.
- `Warengruppe` identifies the product categories according to following scheme:
 - 1 : Bread
 - 2 : Rolls

BREAKOUT

- **Schaut Euch die Kaggle-Seite genauer an.**
- **Versucht eine Vorhersage für den Test-Zeitraum zu erstellen und hochzuladen.**

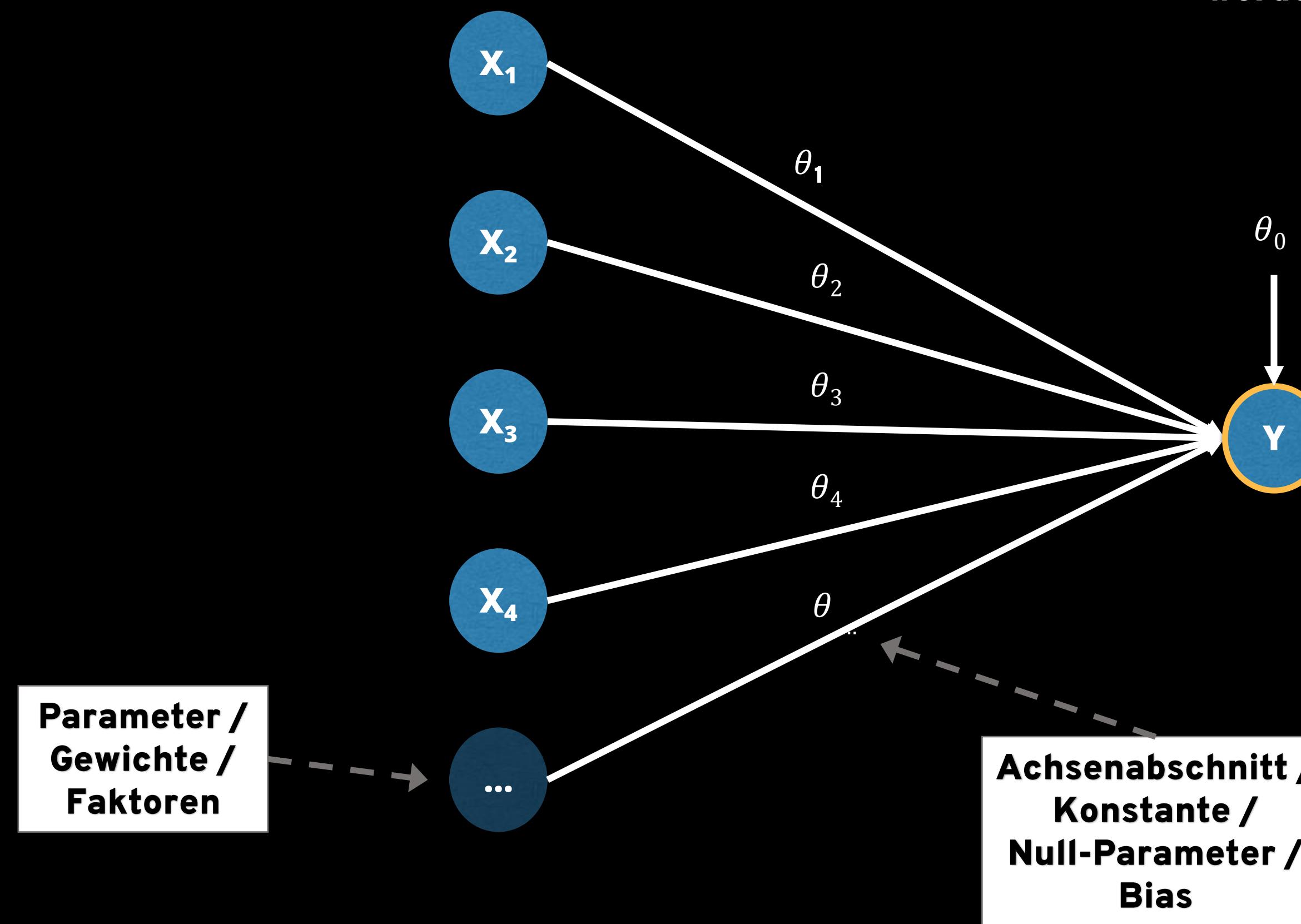
ZUSAMMENFASSUNG LINEARE REGRESSION

Input Layer

Elemente sind die Input-Variablen; auch genannt: Input-Features oder Input-Dimensionen.

Output Layer

Nutzt eine „*Aktivierungsfunktion*“ (hier lineare Funktion) mit der die Parameter θ der eingehenden Schicht zusammengefasst werden.



EIGENSCHAFTEN DES LINEAREN MODELLS

- **Sowohl ohne als auch mit Regularisierung ist die Optimierung der Parameter für das lineare Modell sehr leicht und schnell möglich.**
→ Für einfache Modelle ist es einfach optimierte Parameter zu erhalten.
- **Einfacher zu optimierende Modelle haben in der Regel stärkere Annahmen über die Zusammenhänge der Variablen (hier lineare Zusammenhänge).**
→ Eine optimale Kodierung/Kategorisierung der Variablen entsprechend der Annahmen ist umso wichtiger.
→ Ggf. können die tatsächlichen Zusammenhänge nicht modelliert werden.

EIGENSCHAFTEN DES NEURONALEN NETZES

- **Definition von zusätzlichen „Hidden Layern“ zwischen Input und Output Layer.**
 - **Statistik: Definition von latenten Variablen mit in der Regel unbekannter Bedeutung**
- **Nutzung nicht linearer Aktivierungsfunktionen.**
 - **Erlaubt Modellierung von Interaktionseffekten**
 - **Erlaubt Modellierung von nicht-linearen Effekten**

NEURONALE NETZE

Input Layer

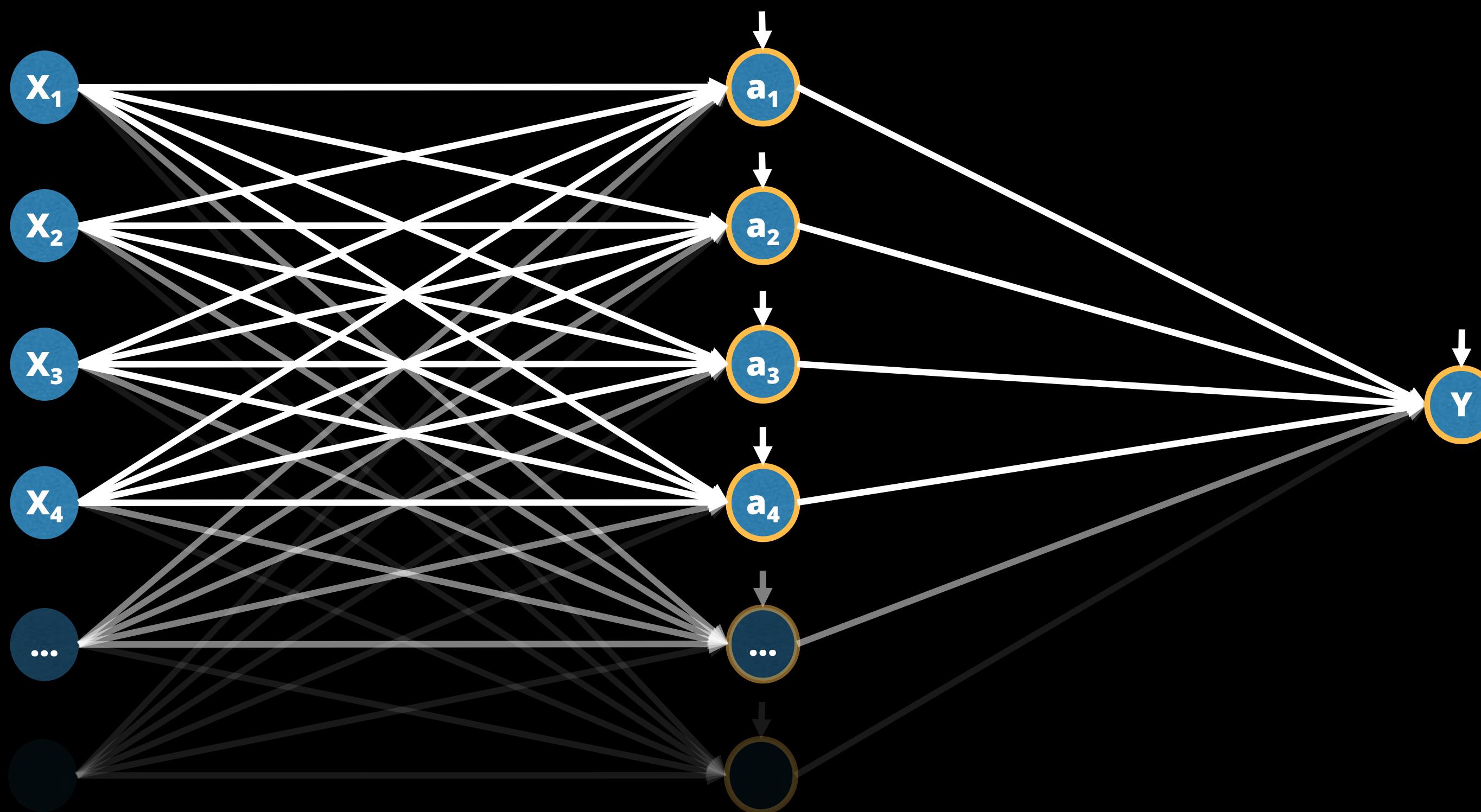
Besteht aus Input-Variablen/Features/Dimensionen

Hidden Layer

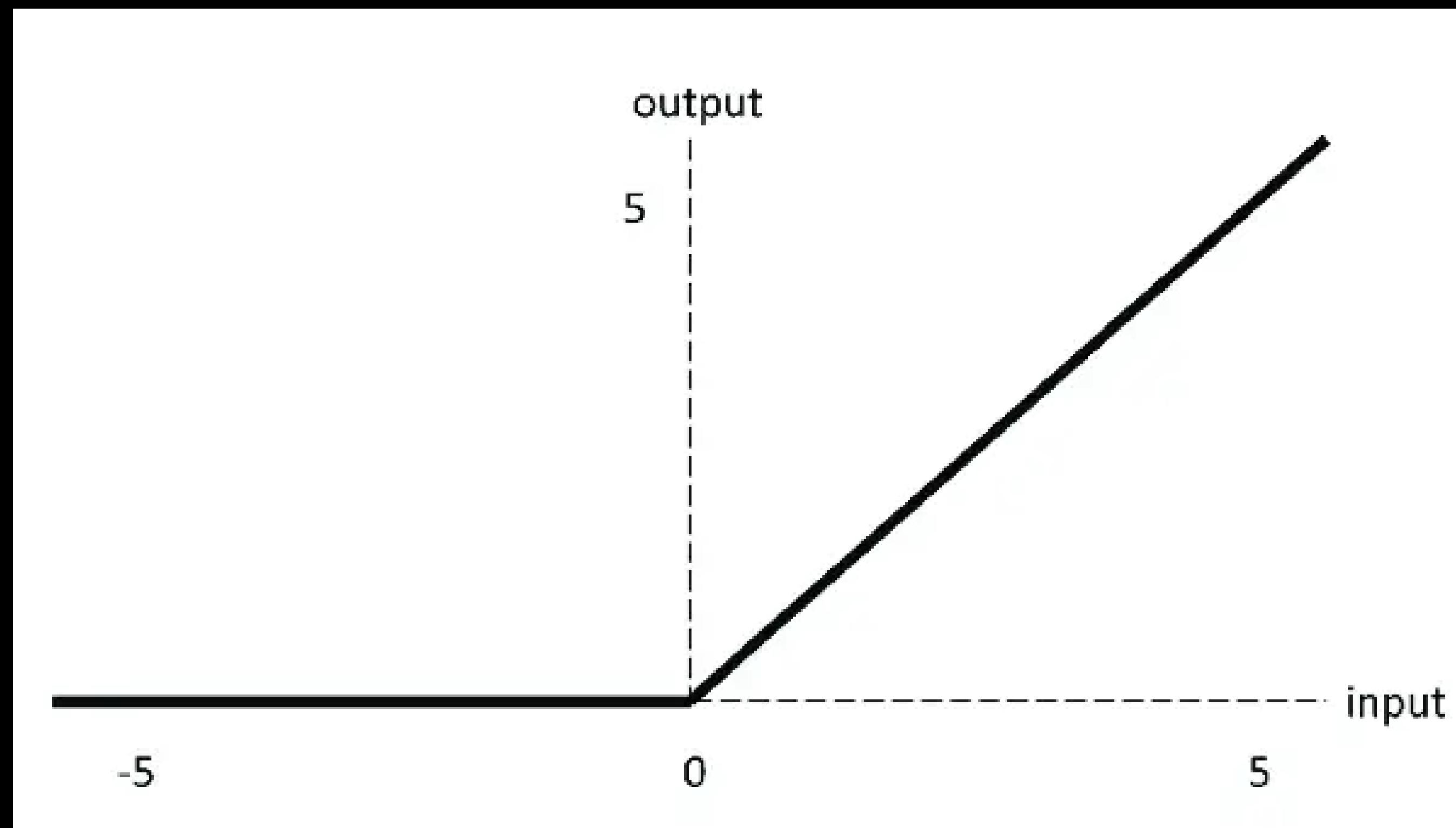
Fasst mit Hilfe von Aktivierungsfunktionen und geschätzten Gewichten die Werte der vorherigen Schicht in jeweils einem Neuron zusammen.

Output Layer

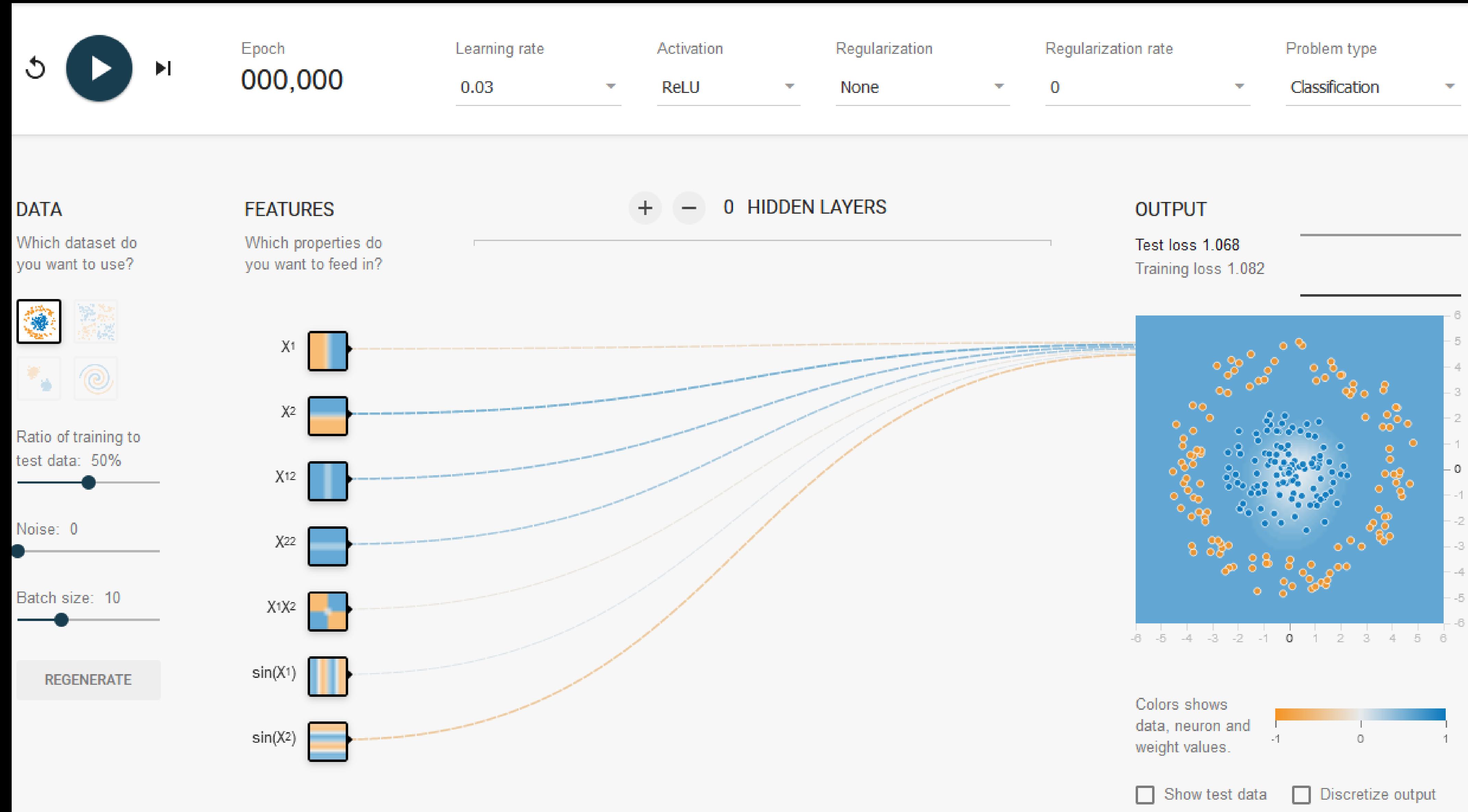
Fasst ebenfalls mit Hilfe von Aktivierungsfunktion und geschätzten Gewichten die Werte der vorherigen Schicht zusammen.



DIE NICHT LINEARE AKTIVIERUNGSFUNKTION RELU



$$f(x) = \max(0, x)$$



BREAKOUT

Ruft folgendes Tool auf: <https://playground.tensorflow.org/>

- 1) **Definiert ein lineares Modell (keine Hidden Layer)**
 - **Welche der 4 Datensätze könnt Ihr mit dem linearen Modell erfolgreich klassifizieren?**
 - **Inwieweit könnt Ihr die Ergebnisse hinsichtlich der verwendeten Features (Variablen) interpretieren?**

- 2) **Definiert zwei Hidden Layer und probiert die Anzahlen der Neuronen so zu ändern, dass Ihr den spiralförmigen Datensatz vorhersagen könnt.**
 - **Welche Verteilungen könnt Ihr erfolgreich vorhersagen?**
 - **Inwieweit könnt Ihr die Ergebnisse hinsichtlich der verwendeten Features interpretieren?**

HYPERPARAMETER

- **Wahl des Modells bzw. der Modellarchitektur**
- **Wahl der Aktivierungsfunktionen**
- **Wahl der Kostenfunktion**
 - MAE, MSE, ...
- **Wahl der Optimierungsfunktion**
 - Größe der Lernrate
- **Wahl der Batch-Größe**

Je nach Modellarchitektur und gewählten Komponenten zahlreiche weitere...

AUFGABEN

- Datensatz weiter um zusätzliche Variablen ergänzen, die für die Schätzung des Umsatzes relevant sein könnten.
- Die Vorhersagegüte Eures linearen Modells [hier](#) auf Kaggle testen.
- [Dieses Video](#) (12 Minuten) zur Einführung in Neuronale Netze an anschauen.
- Mich zu Eurem Repo einladen oder, wenn es öffentlich ist, mir den Link zu Eurem Repo in einer privaten Nachricht schicken.
- Lest die [Hello Python Einführung](#) auf Kaggle und macht die [zugehörige Übung](#).
- Nutzt [diesen Link](#), um ein Google Colab Notebook zu öffnen. (Falls Ihr keinen Google-Account habt, müsst Ihr dazu ggf. zunächst einen anlegen.) Auf der sich öffnenden Seite klickt dann „Abbrechen“ und lest Euch das dargestellte Notebook durch.