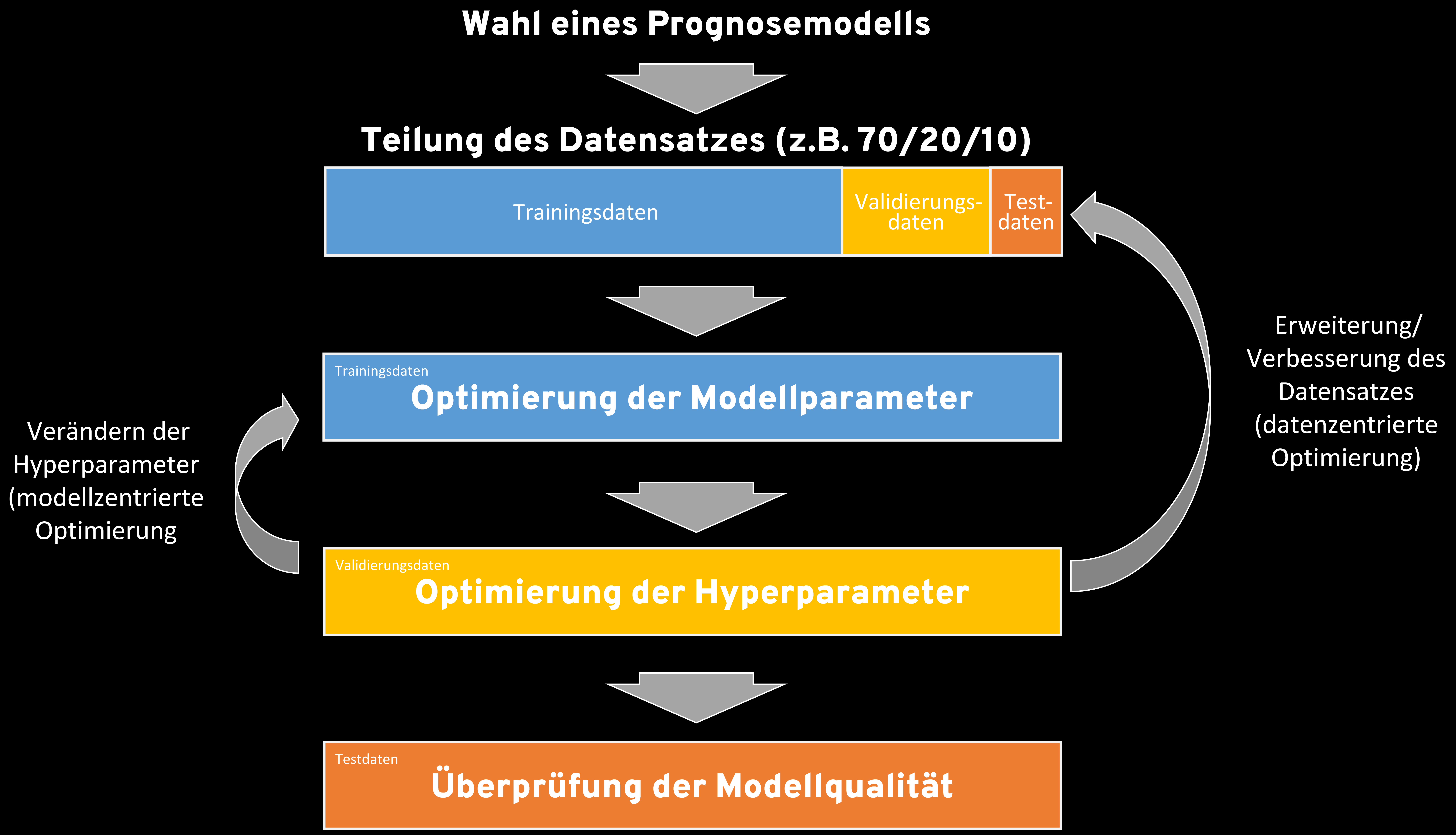
16.05.23

OPENCAMPUS.sh

Einführung in Data Science und maschinelles Lernen

EINFÜHRUNG IN MASCHINELLES LERNEN

- Vorgehen zum Trainieren von maschinellen Lernalgorithmen
- Definition der linearen Regression
- Kostenfunktionen
- Optimierungsfunktionen


Wahl eines Prognosemodells
Teilung des Datensatzes (z.B. 70/20/10)
Trainingsdaten
Validierungs-daten
Test-daten
Erweiterung/ Verbesserung des Datensatzes (datenzentrierte Optimierung)
Trainingsdaten
Optimierung der Modellparameter
Verändern der Hyperparameter (modellzentrierte Optimierung
Validierungsdaten
Optimierung der Hyperparameter
Testdaten
Überprüfung der Modellqualität

TEILUNG EINES DATENSATZES (OHNE ZEITREIHEN!)

(1) Mischung der Reihenfolge (Randomisierung)

(2) Definition der Größen von Trainings-, Validierungs- und Testdatensatz in Prozent

(3) Teilung des Datensatzes

TEILUNG EINES DATENSATZES MIT ZEITREIHEN

(1) Ordnen der Zeitreihe

(2) Definition der Zeitreihenfenster für Validierungs- und Testdatensatz

(3) Teilung des Datensatzes

PROGNOSEMODELLE

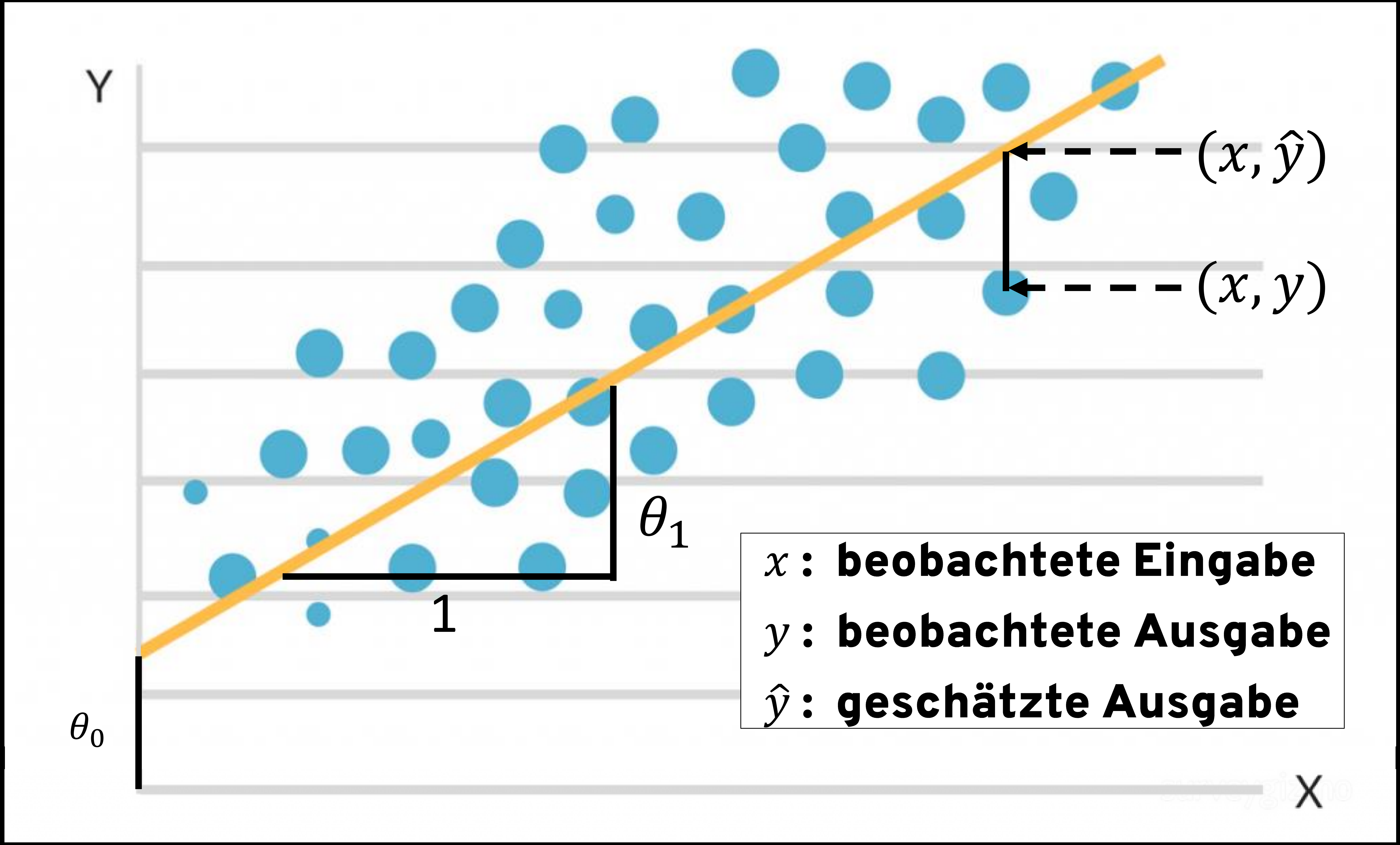
In diesem Kurs behandelte:

- Lineares Modell
- Neuronales Netz
- (Support Vektor Maschine)

LINEARES MODELL

$$\hat{y} = \theta_0 + \theta_1 x$$

$$= h_x\left(\theta_0, \theta_1\right)$$



KOSTENFUNKTION

Zur Berechnung der Funktion mit den optimalen Parametern θ_0 und θ_1:

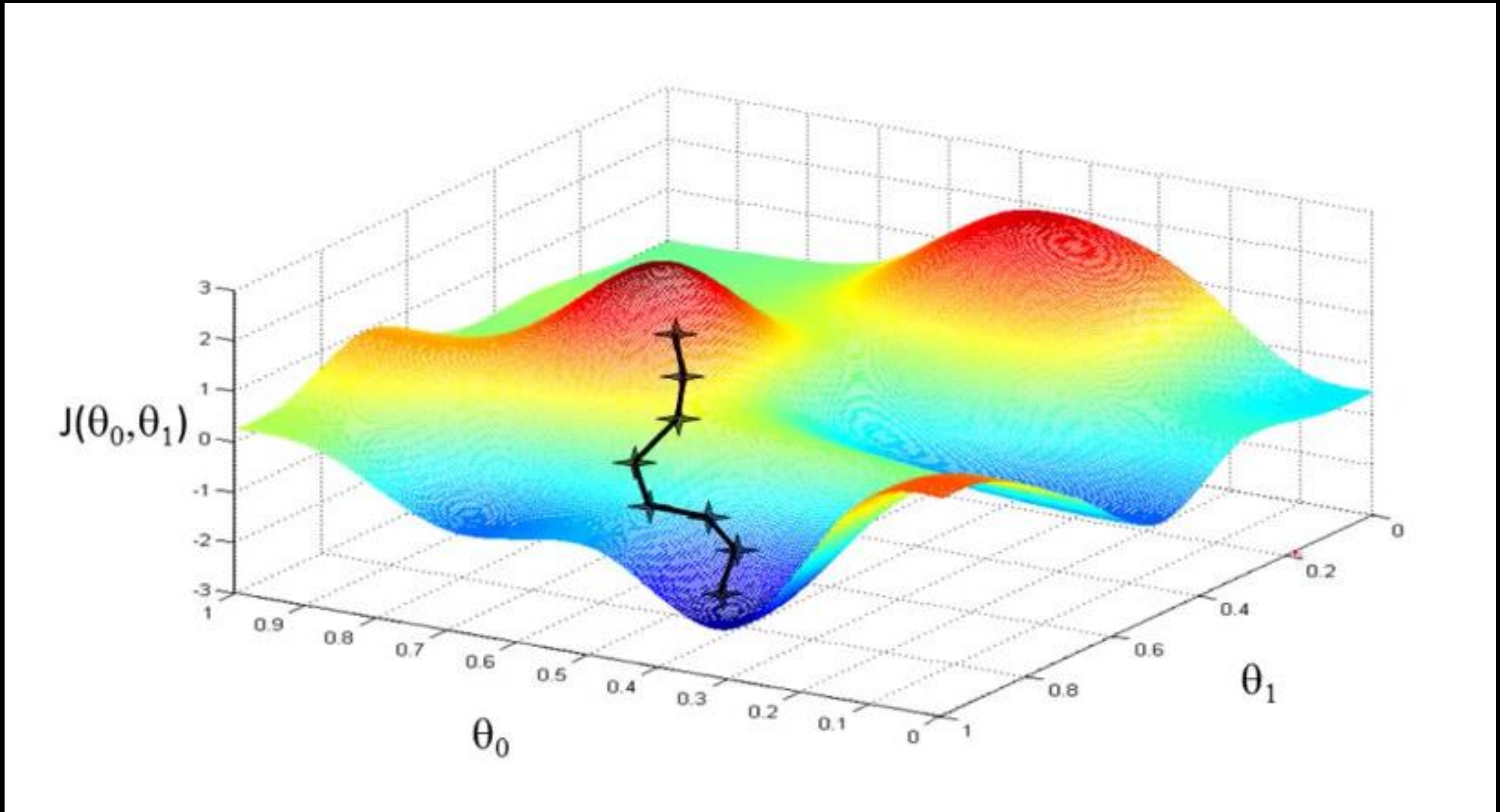
$$J_x(\theta_0, \theta_1) = \frac{1}{m}\sum |h_x(\theta_0, \theta_1) - y|$$

Mean Absolute Error (MAE)

$$J_x(\theta_0, \theta_1) = \frac{1}{m}\sum (h_x(\theta_0, \theta_1) - y)^2$$

Mean Squared Error (MSE)

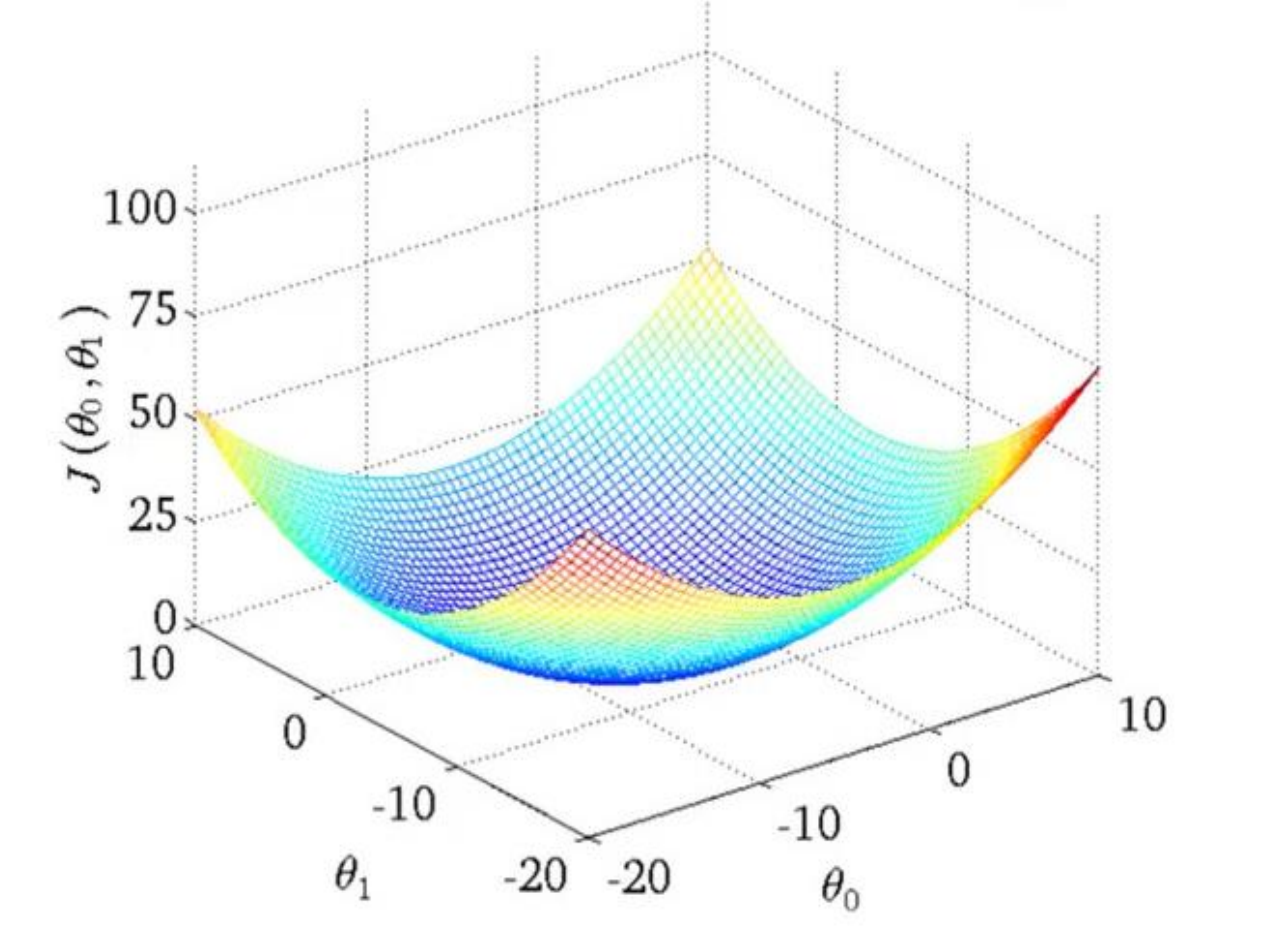
OPTIMIERUNGSFUNKTION



Quelle: https://www.coursera.org/learn/machine-learning

- **Iteratives Verfahren (Gradient Descent), um das Minimum der Kostenfunktion zu finden.**
- **Schrittgröße zur Annäherung wird durch die Lernrate („Learning Parameter") kontrolliert**

MINIMIERUNG BEI LINEAREN MODELLEN



Quelle: https://www.coursera.org/learn/machine-learning

- **Für lineare Modelle ist die Kostenfunktion konvex und besitzt keine lokalen Minima.**
- **Hier werden häufig auch andere statistische Verfahren als Gradient Descent genutzt.**

(Insbesondere wenn das Modell nur wenige Variablen umfasst.)

BEISPIEL EINES LINEAREN MODELLS

```
library(readr)
house_pricing <- read_csv("https://raw.githubusercontent.com/
                          opencampus-sh/einfuehrung-in-data-
                          science-und-ml/main/house_pricing_data/
                          house_pricing_train.csv")

mod <- lm(price ~ sqft_lot15 + as.factor(condition), house_pricing)
summary(mod)
```

ERGEBNIS DES LINEAREN MODELLS

```
Call:
lm(formula = price ~ sqft_lot15 + as.factor(condition), data = house_pricing)

Residuals:
    Min      1Q  Median      3Q     Max 
-795388 -214555  -85989  101761 7183941 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)           3.261e+05  7.049e+04   4.625 3.77e-06 ***
sqft_lot15            1.138e+00  1.035e-01  11.002  < 2e-16 ***
as.factor(condition)2 -2.305e+04  7.690e+04  -0.300 0.764379    
as.factor(condition)3  2.031e+05  7.057e+04   2.878 0.004008 ** 
as.factor(condition)4  1.800e+05  7.070e+04   2.546 0.010915 *  
as.factor(condition)5  2.762e+05  7.118e+04   3.880 0.000105 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 366300 on 17284 degrees of freedom
Multiple R-squared:  0.01409,	Adjusted R-squared:  0.0138 
F-statistic: 49.39 on 5 and 17284 DF,  p-value: < 2.2e-16
```

KENNWERTE DER REGRESSION

p-Wert

- Signifikanzwert
- Wahrscheinlichkeit, dass der zugehörige Wert in der Regression gleich null ist.
- Werte unter 0,05 (entspricht 5%) werden üblicherweise als signifikant betrachtet

(Adjustiertes) R^2

- Kennzahl zur Beurteilung der Güte einer Regression
- Wert zwischen 0 und 1, der der dem Anteil der erklärten Variation entspricht (1 entspricht 100%):

$$R^2 = \frac{\text{Erklärte Varianz}}{\text{Gesamtvarianz}}$$

- Das adjustierte R^2 bestraft das Hinzufügen zusätzlicher Variablen/Parameter

DataCamp Tutorial zur Linearen Modellierung in R :
https://www.datacamp.com/community/tutorials/linear-regression-R

BEISPIEL DATENSATZTEILUNG (KEINE ZEITREIHE)

```
#### Teilen des Datensatzen in Trainings- Validierungs- und Testdatensatz
```{r}
# Load the dplyr library
library(dplyr)

# Set a random seed for reproducibility
set.seed(42)
# Shuffle the data
data_shuffled <- data %>% sample_frac(1)

# Calculate the number of rows for each dataset
n_total <- nrow(data)
n_train <- floor(0.7 * n_total)
n_validation <- floor(0.20 * n_total)

# Split the data into training, validation, and test datasets
train_data <- data_shuffled %>% slice(1:n_train)
validation_data <- data_shuffled %>% slice((n_train + 1):(n_train + n_validation))
test_data <- data_shuffled %>% slice((n_train + n_validation + 1):n_total)

# Check the dimensions of the datasets
cat("Training dataset dimensions:", dim(train_data), "\n")
cat("Validation dataset dimensions:", dim(validation_data), "\n")
cat("Test dataset dimensions:", dim(test_data), "\n")

```
```

BEISPIEL LINEARE MODELLIERUNG UND VORHERSAGE

````
### Beispiel einer einfachen linearen Regression
```{r}
  mod <- lm(price ~ sqft_lot15 + as.factor(condition), train_data)
  summary(mod)

```

### Nutzung des resultierenden Modells für eine Vohersage
```{r}
# Make predictions using the test data
predicted_values <- predict(mod, newdata = validation_data)

# Compare the predicted values with the actual values
comparison <- data.frame(Actual = test_data$price, Predicted = predicted_values)

# Calculate the mean squared error (RMSE)
rmse <- sqrt(mean((comparison$Actual - comparison$Predicted)^2))

# Display the comparison and RMSE
head(comparison)
cat("Root Mean Squared Error (RMSE):", rmse, "\n")

```
````

BREAKOUT

- Splittet Euren Datensatz in Trainings, Validierungs- und Testdatensatz

- Definiert eine lineare Modellgleichung und führt mit Eurem Trainingsdatensatz eine lineare Regression durch. Versucht das adjustierte R^2 des linearen Modells zu maximieren.

- Nutzt das lineare Modell, um eine Vorhersage für den Validierungsdatensatz zu erstellen.

ABRUFEN DES MAPE FÜR DEN TESTDATENSATZ

```r
library(dplyr)
library(readr)
library(httr)

# Dataframe for request must include columns `Datum`, `Warengruppe` und `Umsatz`
predictions <- read_csv("prediction_template.csv")

# name must not be provided; however, each team must upload at least one not
# anonymous prediction
name <- "Gruppe X"

# Execution of the request
r <- POST("https://bakery-sales-mape-tolicqztoq-ey.a.run.app/",
          body = list(name=name, predictions=predictions),
          encode = "json")
# Output of MAPE in Percent
content(r, "parsed", "application/json")
```

AUFGABEN

- Datensatz weiter um zusätzliche Variablen ergänzen, die für die Schätzung des Umsatzes relevant sein könnten.
- Einmal dieses R-Script durchlaufen lassen, um Python (bzw. Miniconda) mit verschiedenen zusätzlichen Funktionspaketen zu installieren.
- Euren Datensatz teilen in einen Trainingsdatensatz vom 01.07.2013 bis 31.03.2019, einem Validierungsdatensatz vom 01.04. bis 08.06.2019 und einem Testdatensatz vom 09.06. bis 30.07.2019
- Eine lineare Modellgleichung aufstellen, die das adjustierte R^2 für Euren Trainingsdatensatz maximiert.
- Zum Thema Overfitting dieses Video (9 Minuten) anschauen.