

30.11.21

# Einführung in Data Science und maschinelles Lernen

## NEURONALE NETZE

- **Wiederholung Neuronale Netze (NN)**
- **Modellgütekriterien**
- **Hyperparameter in NN**
- **Frameworks zur Implementierung von NN**
- **Implementierung eines NN mit TensorFlow und Python**



# CODING. WATERKANT 2022

**7. bis 10. Juni auf dem Waterkant Festival Gelände**

## AI and Art

(Vladimir Alexeev, OpenAI Ambassador; Michael Haas, Freelance Digital Artist)

## KI.EL Racing Challenge

(Jake Petersen, opencampus.sh)

## Detection of Mobile Genetic Elements

(Yiqing Wang and Dustin Hanke, University of Kiel)

## Identifying Correct Arguments in Open Text Answers

(Christian Mayer and Georg Gorshid, University of Mannheim)

## Identifying Geo-Locations in Historical Documents

(Simon Van der Wulp and Juan Veliz, north.io)

## Predicting Stock Price Dynamics

(Jonas Mielck, University of Kiel and stackOcean GmbH)

## Automated Feedback for Students' Argumentative Essays

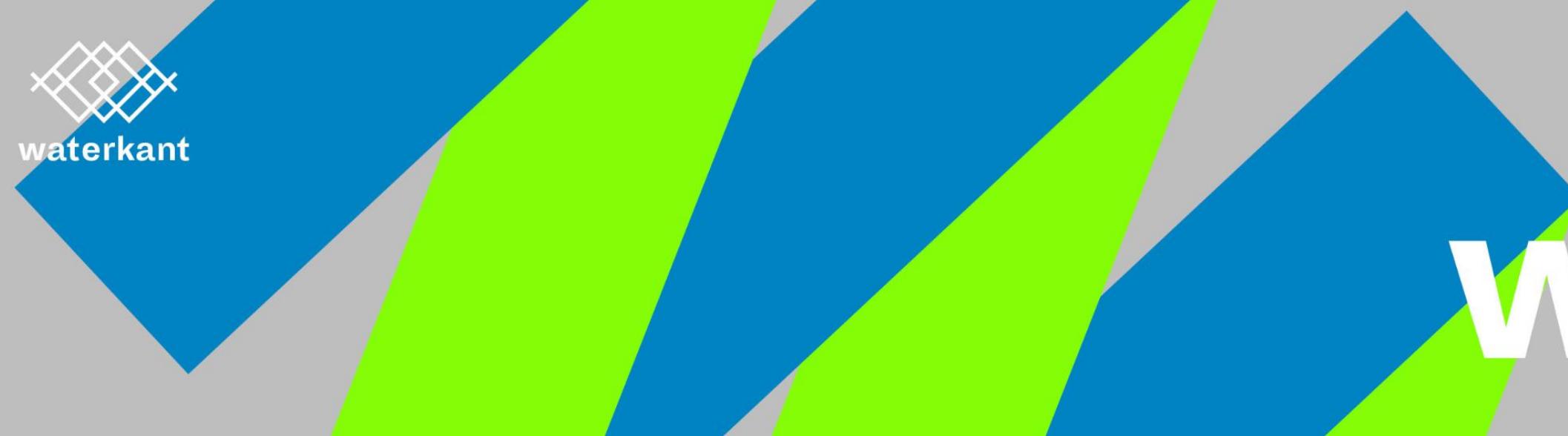
(Thorben Jansen and Nils-Jonathan Schaller, IPN Kiel)

## Enabling Researchers to Automatically Code their Text Responses from Assessments

(Fabian Zehner und Nico Andersen, DIPF Frankfurt)

## Improving Lung Imaging with AI

(Claus Glüer, Intelligent Imaging Lab at the University of Kiel)



# CODING. WATERKANT 2022

## **7. Juni auf dem Waterkant Festival Gelände**

09:00 Welcome to Coding.Waterkant

09:15 Project Pitches

10:00 Common Breakfast

11:00 Introduction workshops on the projects  
that are open for participation

12:00 Meet the Expert: Q&A with Ralf Krestel  
from the ZBW/University of Kiel

13:00 Lunch Break

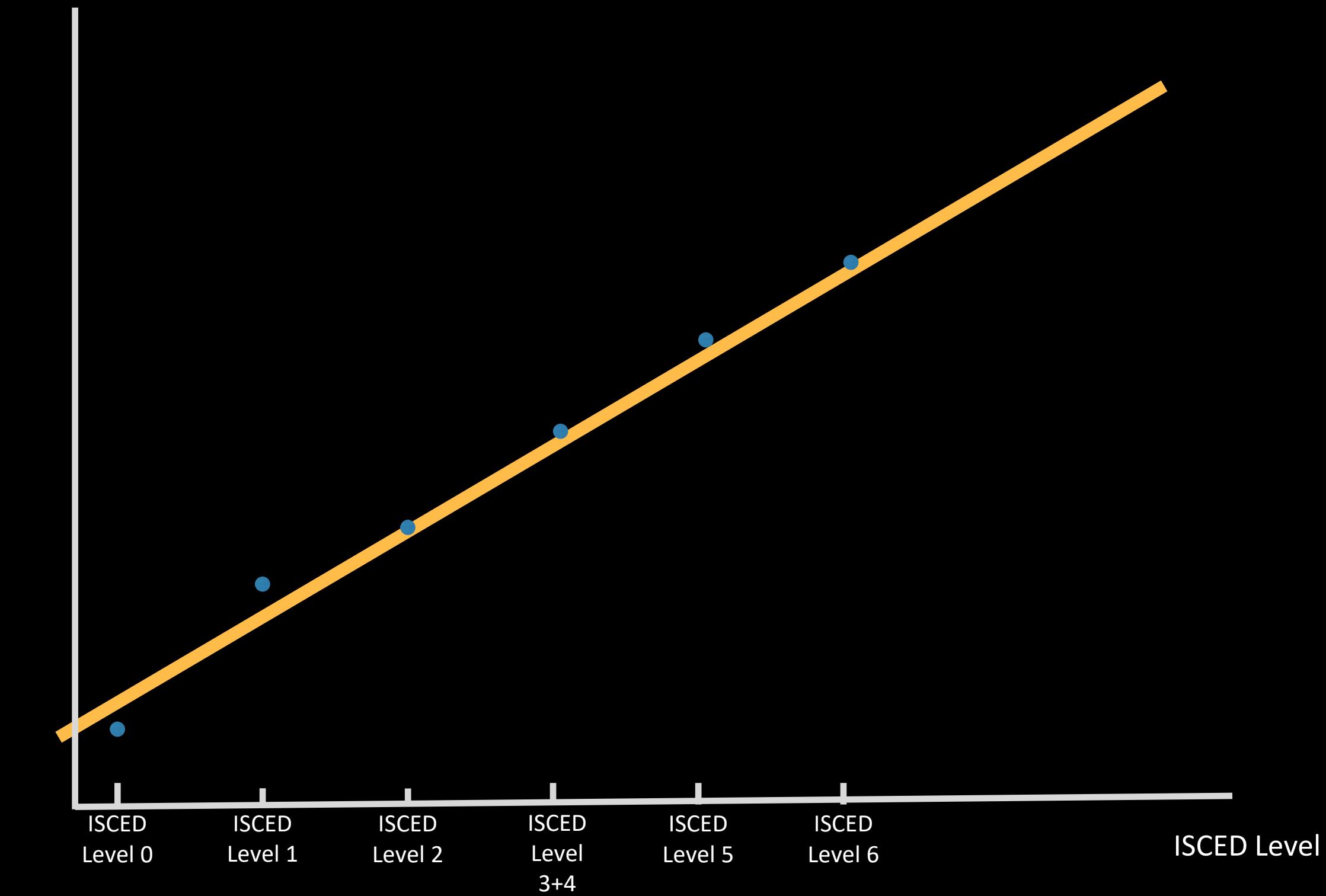
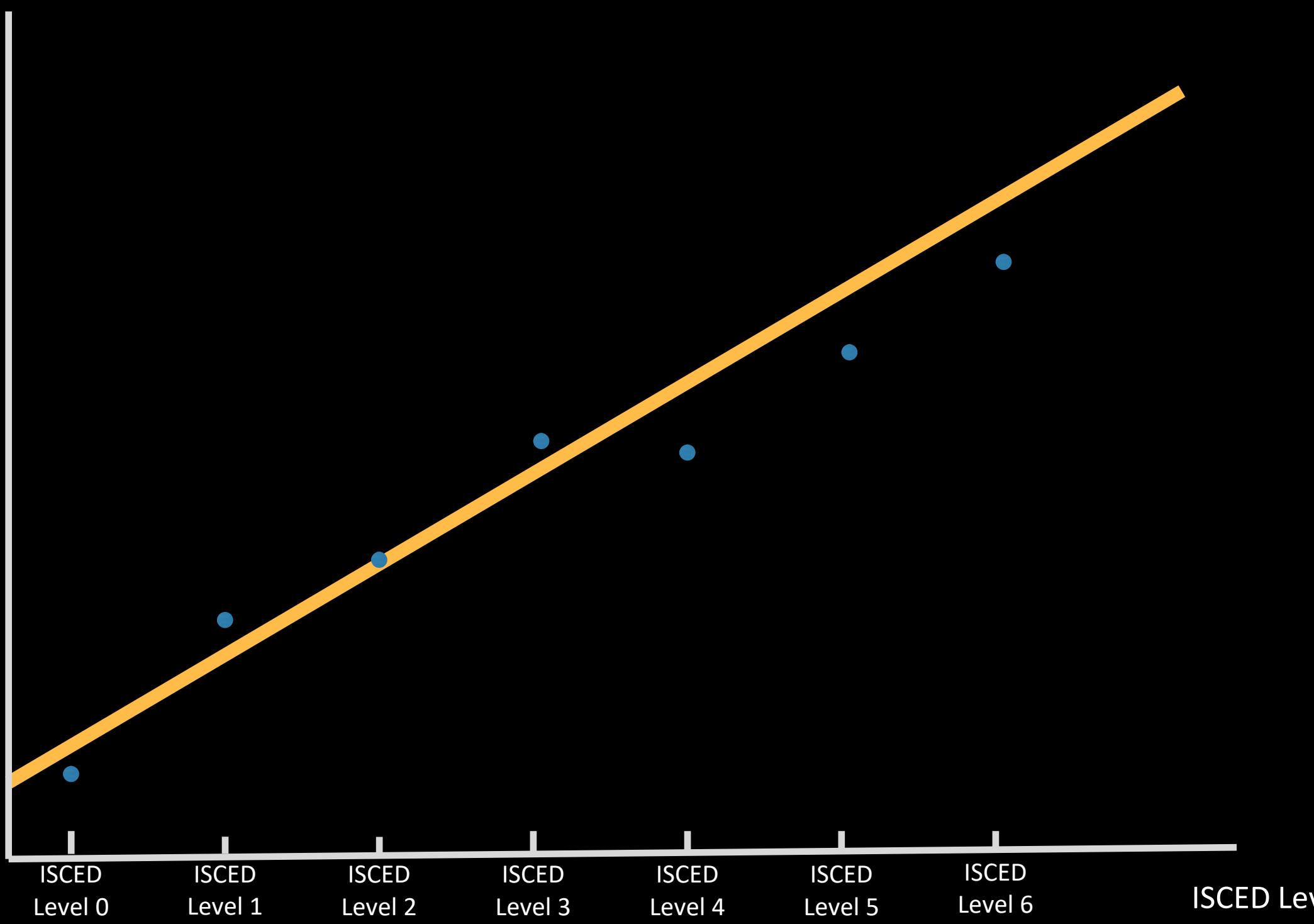
15:00 Thies Schönfeldt (meteolytix GmbH): The  
Do's and Don'ts in Time Series Prediction

16:00 Jan Deller (LMU  
München/opencampus.sh): Introduction to  
TensorBoard

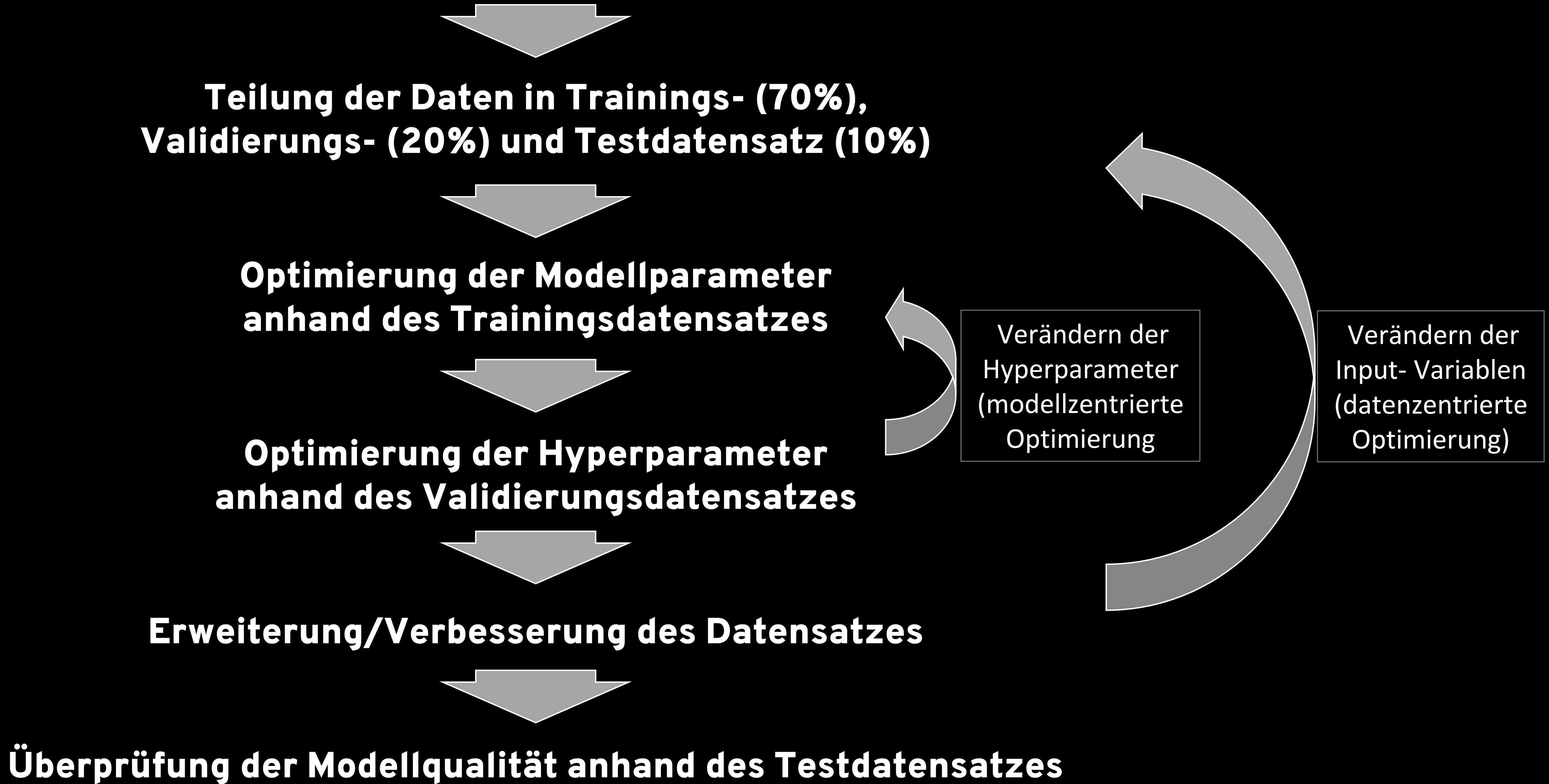
**18:00 Kursstart: Zeitreihenanalysen**

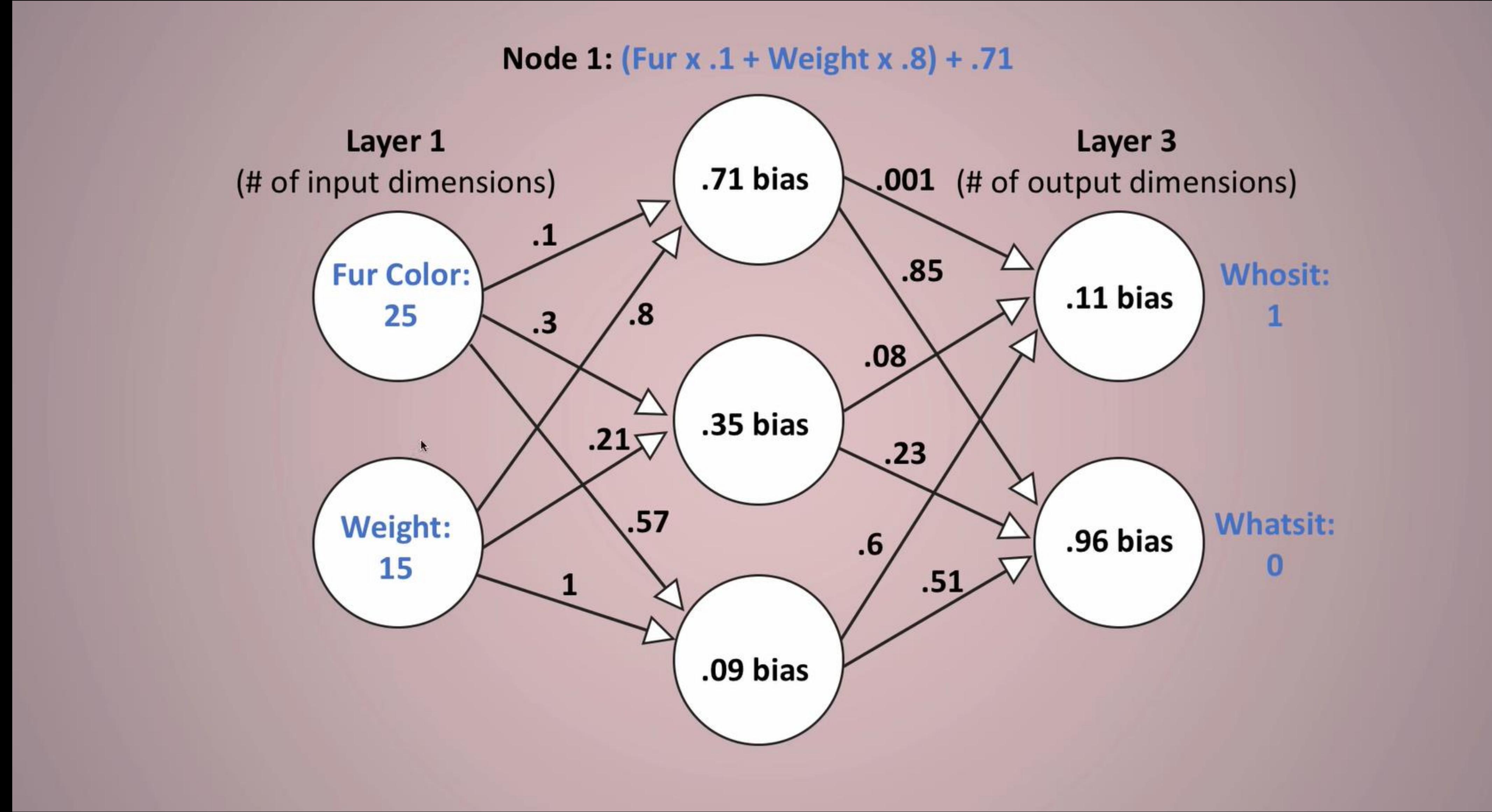
19:30 BBQ

# RELEVANZ VON KATEGORISIERUNGEN



## Wahl eines Prognosemodells





# NEURONALE NETZE

## Input Layer

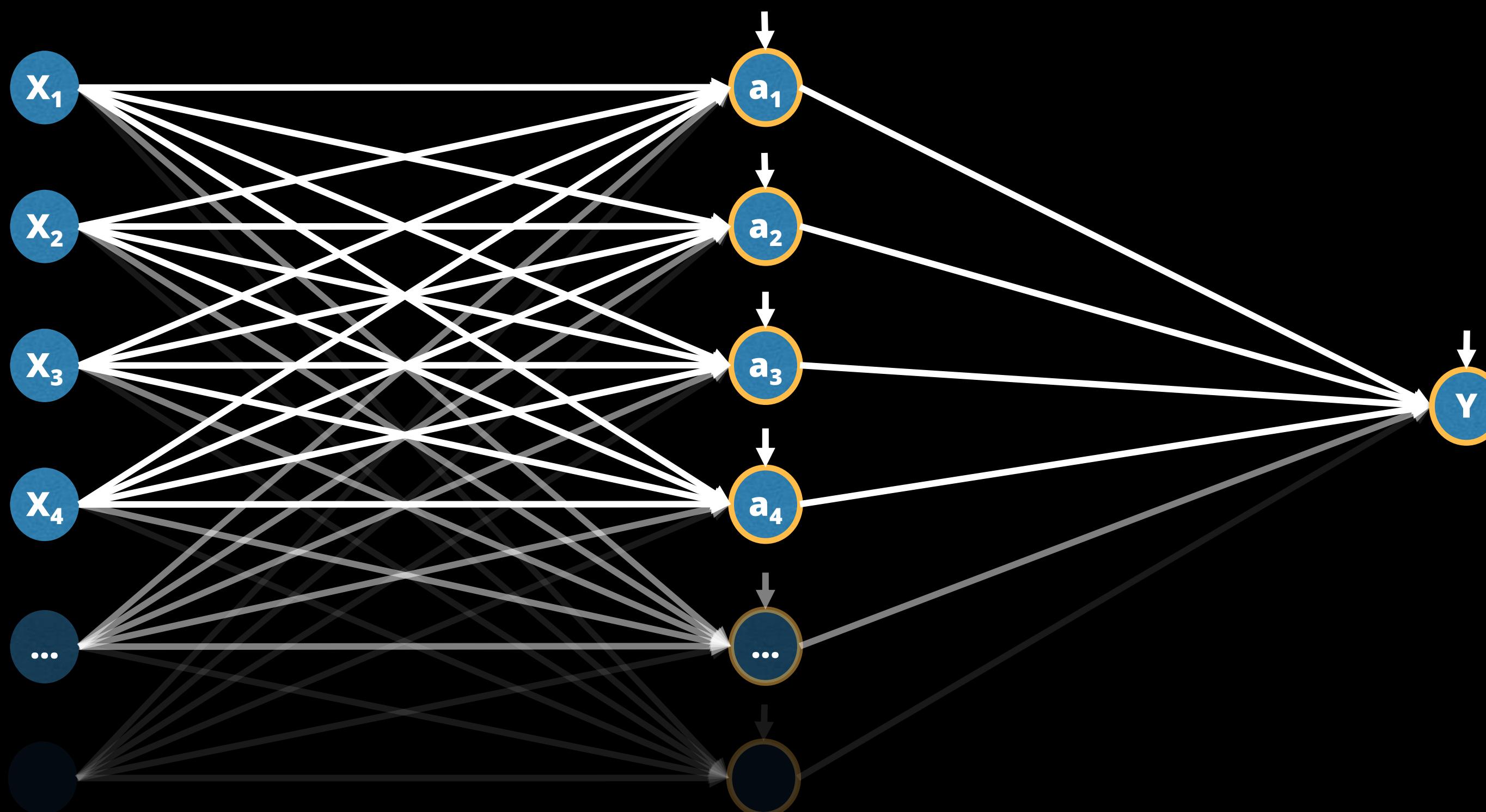
Besteht aus Input-Variablen/Features/Dimensionen

## 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.



# WICHTIGE KONZEPTE

## Forward Propagation:

- **Berechnung der Vorhersage basierend auf den aktuellen Parametern und den definierten Aktivierungsfunktionen**

## Backward Propagation:

- **Berechnung des Schätzfehlers mit Hilfe der Kostenfunktion**
- **Berechnung neuer, verbesserter Parameter mit Hilfe der Optimierungsfunktion**

# MODELLGÜTEKRITERIEN

**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 )**

**$r^2 = 1 - rse$**

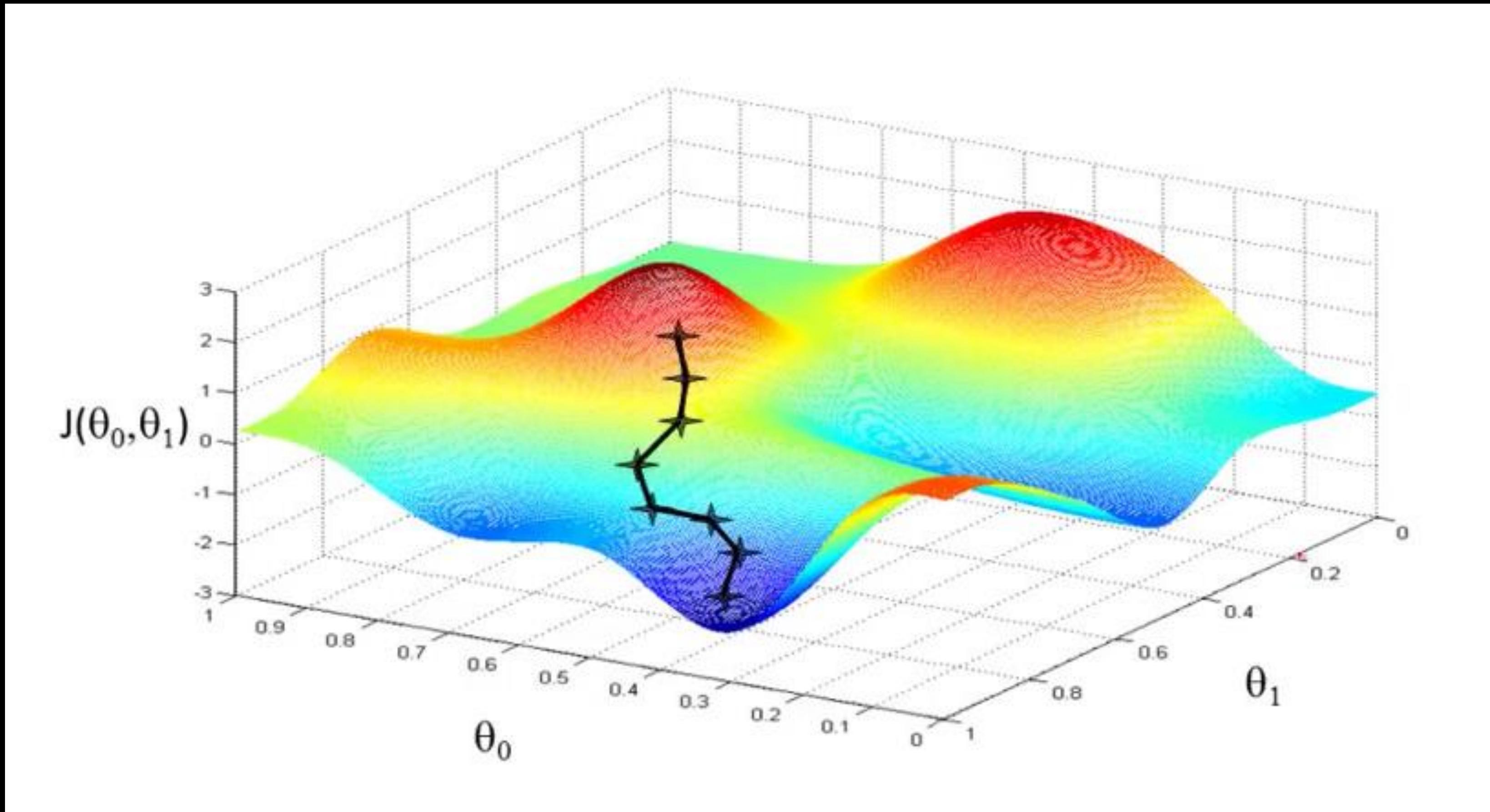
Video (3 Minuten) mit Erklärung und Darstellung der Kriterien:

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

# HYPERPARAMATER IN NEURONALEN NETZEN

- **Wahl der Architektur:**
  - Anzahl der Hidden Layer des Netzes
  - Typen der Hidden Layer
  - Anzahl der Neuronen je Hidden Layer
  - Wahl der Aktivierungsfunktionen
- **Wahl der Kostenfunktion („Loss Function“)**
- **Wahl der Optimierungsfunktion („Optimizer“)**
- **Wahl der Parameter des Optimizers**

# PARAMETER VON OPTIMIZERN



- **Schrittgröße für die Annäherung an das Kosten-Minimum („Learning Rate“)**
- **Trägheit bei Richtungsänderungen („Momentum“)**

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

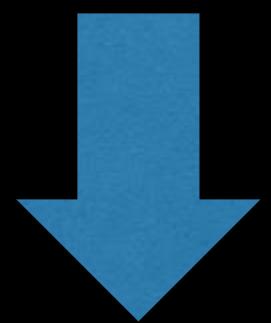
# PARAMETER DES OPTIMIZERS „ADAM“

- Lernparameter/ Schrittweite der Optimierung :  
**alpha (learning rate)**
- Trägheit der Optimierung:  
**beta1 and beta2 (momentum)**

# R VS. PYTHON

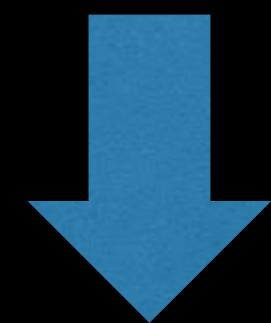
***Statistische Verfahren  
außerhalb des ML***

**Mathematik/ Statistik und  
Disziplinen mit  
Anwendungsbereichen von  
Statistik (Ökonometrie,  
Psychometrie, Biometrie, ...)**



***Statistische Verfahren des ML***

**Angewandte Informatik und  
freie Wirtschaft mit  
Anwendungsbereichen von ML**



# INSTALLATION VON PYTHON

```
1  ---
2  title: "R Notebook"
3  output: html_notebook
4  ---
5
6  ### Installation von Python und der für TensorFlow benötigten Pakete (nur einmalig notwendig)
7
8  ```{r}
9  install.packages("reticulate")
10 library(reticulate)
11
12 # Installation von miniconda (falls nicht vorhanden)
13 install_miniconda(update=TRUE)
14
15
16 # Anlegen einer speziellen Python Umgebung
17 conda_create("r-reticulate", python_version = "3.8" )
18
19 # Installieren der Pakete in der angelegten Umgebung
20 conda_install("r-reticulate", "pandas")
21 conda_install("r-reticulate", "numpy")
22 conda_install("r-reticulate", "tensorflow")
23 conda_install("r-reticulate", "h5py")
24
25 # Verwenden der speziellen Python Umgebung die zuvor erstellt wurde
26 use_condaenv("r-reticulate")
27
28 ````
```

# VERWENDUNG VON PYTHON IN RSTUDIO

```
31
32  ````{python}
33  import sys
34  import tensorflow
35
36  # Angabe der installierten Python- und TensorFlow-Versionen
37  print("Python Version: " + sys.version+"\nTensorFlow Version: "+tensorflow.__version__)
38
39  ````

40
41  ````{r}
42  # Import Libraries
43  library(reticulate)
44
45
46  # Importing Data
47  data <- mtcars
48
49  ````

50
51
52  ````{python}
53  mpg = r.data['mpg']
54
55  ````

56
57
58  ````{r}
59  table(py$mpg)
60
61  ````

62
```

Siehe auch: [https://rstudio.github.io/reticulate/articles/r\\_markdown.html](https://rstudio.github.io/reticulate/articles/r_markdown.html)

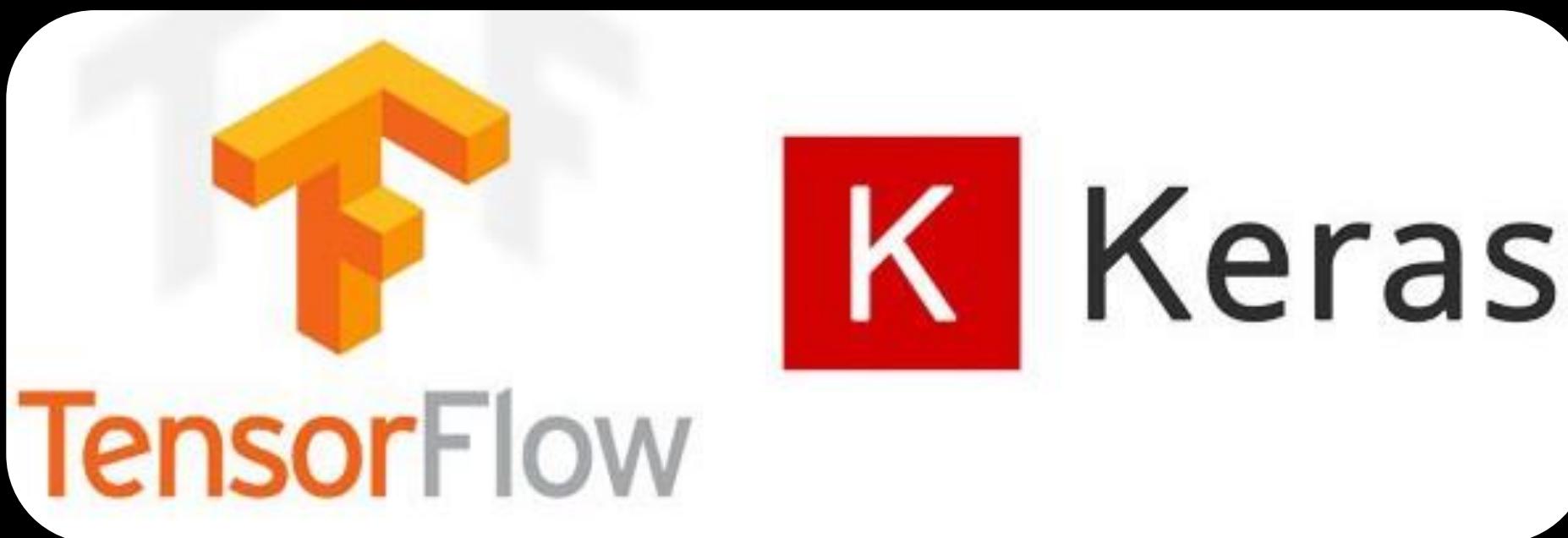
# DATENAUFBEREITUNG FÜR TENSORFLOW

```
1
2 ######
3 ## Preparation of the Environment #####
4
5
6 ## Data Import #####
7
8 ## Data Preparation #####
9
10
11 # Recoding of the variables into one-hot encoded (dummy) variables
12 dummy_list <- c("view", "condition")
13 house_pricing_dummy = dummy_cols(house_pricing, dummy_list)
14
15
16 # Definition of lists for each one-hot encoded variable (just to make the handling easier)
17 condition_dummies = c('condition_1', 'condition_2', 'condition_3', 'condition_4', 'condition_5')
18 view_dummies = c('view_0', 'view_1', 'view_2', 'view_3', 'view_4')
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40 ## Selection of the Feature Variables and the Label variable #####
41
42
43
44
45
46
47
48
49
50 ## Selection of Training, validation and Test Data #####
51
52
53
54
55
56
57
```

# BREAKOUT

- **Schaut Euch zusammen den Code zur Datenaufbereitung an und geht ihn Schritt für Schritt einmal durch.**
- **Diskutiert, was in den einzelnen Schritten getan wird!**





- **Feb 2017:** **TensorFlow 1.0 (Estimator API)**
- **Nov 2017:** **TensorFlow 1.4 (Estimator API, Keras API)**
- **Jan 2019:** **TensorFlow 2.0 (Estimator API, Keras API)**

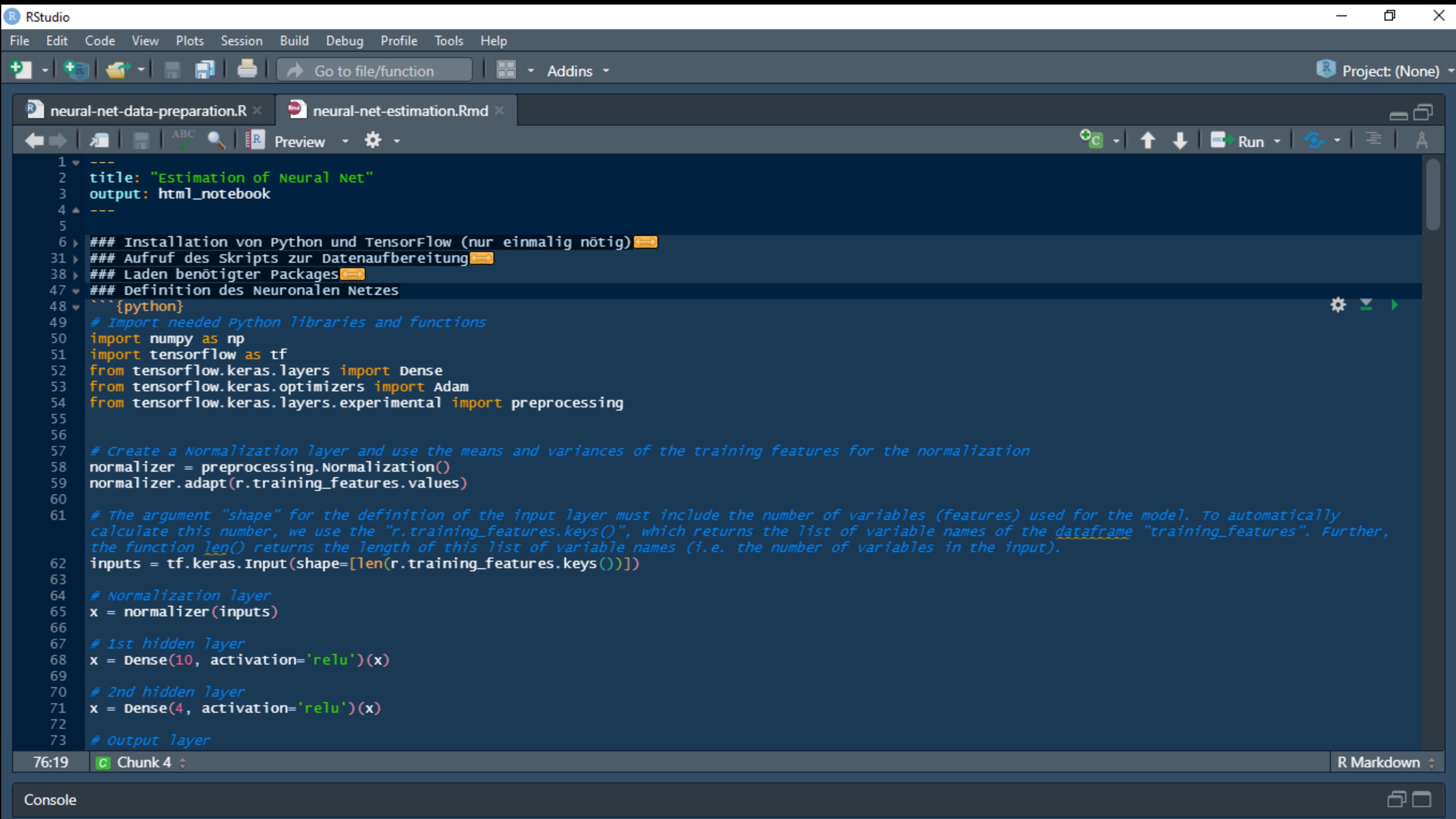
# NUTZUNG VON KERAS IN R

**Keras ist eine Schnittstelle (API/ ein Funktionswrapper) zur vereinfachenden Nutzung von TensorFlow.**

**Prinzipiell zwei Varianten :**

- **Nutzung des Packages „keras“  
(vgl. <https://keras.rstudio.com/>)**
- **Nutzung von Keras in Python und die Integration von Python über das Paket „reticulate“**

# DEFINITION EINES NEURONALEN NETZES



The screenshot shows the RStudio interface with two files open: "neural-net-data-preparation.R" and "neural-net-estimation.Rmd". The "neural-net-estimation.Rmd" file is the active tab, displaying R Markdown code. The code defines a neural network for estimation, including data preparation, model definition, and training layers. It uses Python libraries like numpy and tensorflow, and R packages like keras and optimizers.

```
1 ---  
2 title: "Estimation of Neural Net"  
3 output: html_notebook  
4 ---  
5  
6 ### Installation von Python und TensorFlow (nur einmalig nötig)  
31 ### Aufruf des Skripts zur Datenaufbereitung  
38 ### Laden benötigter Packages  
47 ### Definition des Neuronalen Netzes  
48 ^{python}  
49 # Import needed Python libraries and functions  
50 import numpy as np  
51 import tensorflow as tf  
52 from tensorflow.keras.layers import Dense  
53 from tensorflow.keras.optimizers import Adam  
54 from tensorflow.keras.layers.experimental import preprocessing  
55  
56  
57 # Create a Normalization layer and use the means and variances of the training features for the normalization  
58 normalizer = preprocessing.Normalization()  
59 normalizer.adapt(r.training_features.values)  
60  
61 # The argument "shape" for the definition of the input layer must include the number of variables (features) used for the model. To automatically calculate this number, we use the "r.training_features.keys()", which returns the list of variable names of the dataframe "training_features". Further, the function len() returns the length of this list of variable names (i.e. the number of variables in the input).  
62 inputs = tf.keras.Input(shape=[len(r.training_features.keys())])  
63  
64 # Normalization layer  
65 x = normalizer(inputs)  
66  
67 # 1st hidden layer  
68 x = Dense(10, activation='relu')(x)  
69  
70 # 2nd hidden layer  
71 x = Dense(4, activation='relu')(x)  
72  
73 # output layer
```

# BATCH UND EPOCHE

## Batch

- **Eine Menge von Fällen, die genutzt wird, um die Gewichte des Modells zu optimieren.**
- **Optimierungsschritte werden anhand der Batches durchgeführt.**

## Epoche

- **Optimierung des Modells anhand aller Fälle bzw. aller Batches zu einem Trainingsdatensatz**

**Je nach Modell kann es notwendig sein, es über mehrere hundert oder tausend Epochen zu optimieren.**

# NORMALISIERUNG

## Definition:

- **Abziehen des Mittelwerts und Teilen durch die Standardabweichung.**

**Erleichtert es dem neuronalen Netz, die benötigten Parameter zu erlernen, indem alle Werte in einem ähnlichen Wertebereich liegen.**

# BATCH-NORMALISIERUNG

- Durchführung der Normalisierung auf Batch-Ebene

## Zusätzliche Optimierung:

- exakt gleiche Mittelwerte und Standardabweichungen sind nicht notwendigerweise optimal im Sinne der Modellierung  
→ Einfügen der Normalisierungsparameter als trainierbare Parameter

# AUFGABEN

- Untersucht alle Eure Modellvariablen auf die Existenz von fehlenden und unplausiblen Werten
- Trainiert ein erstes neuronales Netz für Euren Datensatz (Löscht dazu zunächst alle Zeilen mit fehlenden Werten)
- Wenn ihr etwas mehr über Python lernen wollt, könnt Ihr diese Einführung auf Kaggle nutzen.