

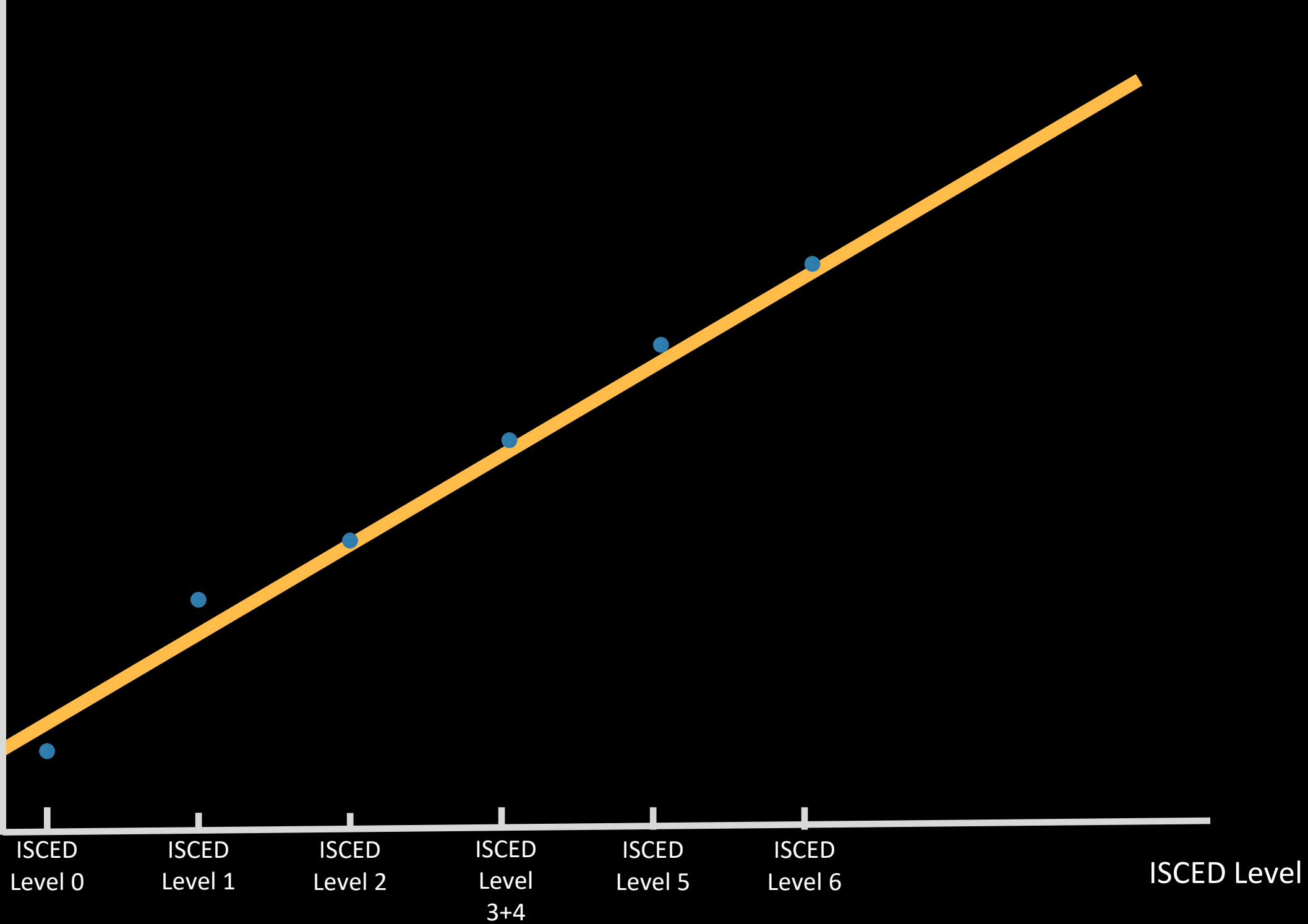
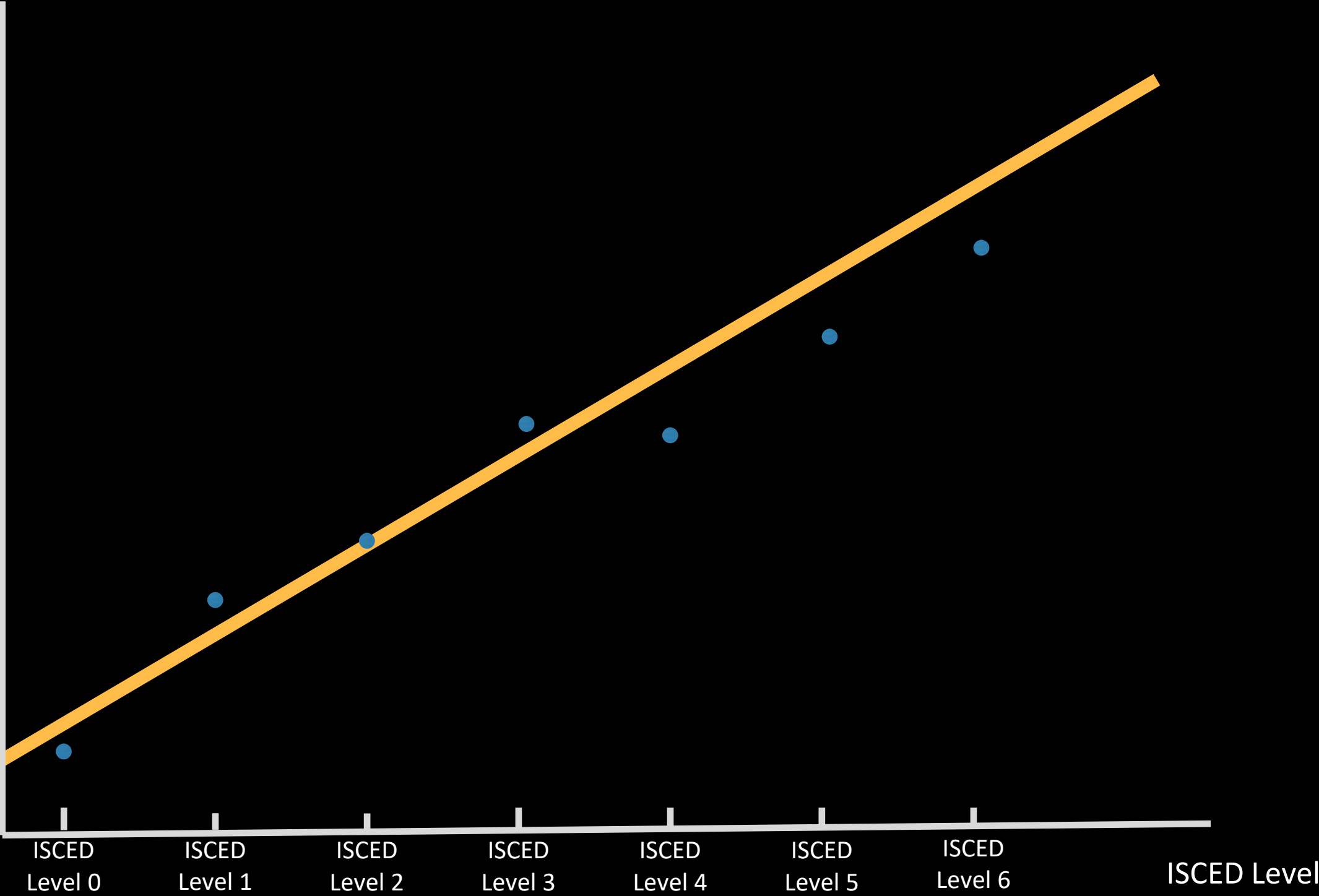
30.11.21

Einführung in Data Science und maschinelles Lernen

NEURONALE NETZE

- **Modellgütekriterien**
- **Aufbau Neuronaler Netze (NN)**
- **Hyperparameter in NN**
- **Frameworks zur Implementierung von NN**
- **Implementierung eines NN mit TensorFlow und Python**
- **Zwischen-Feedback**

RELEVANZ VON KATEGORISIERUNGEN



MODELLGÜTEKRITERIEN

errors: **forecast – actual** **(eigentlich: residuals)**

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

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

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

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

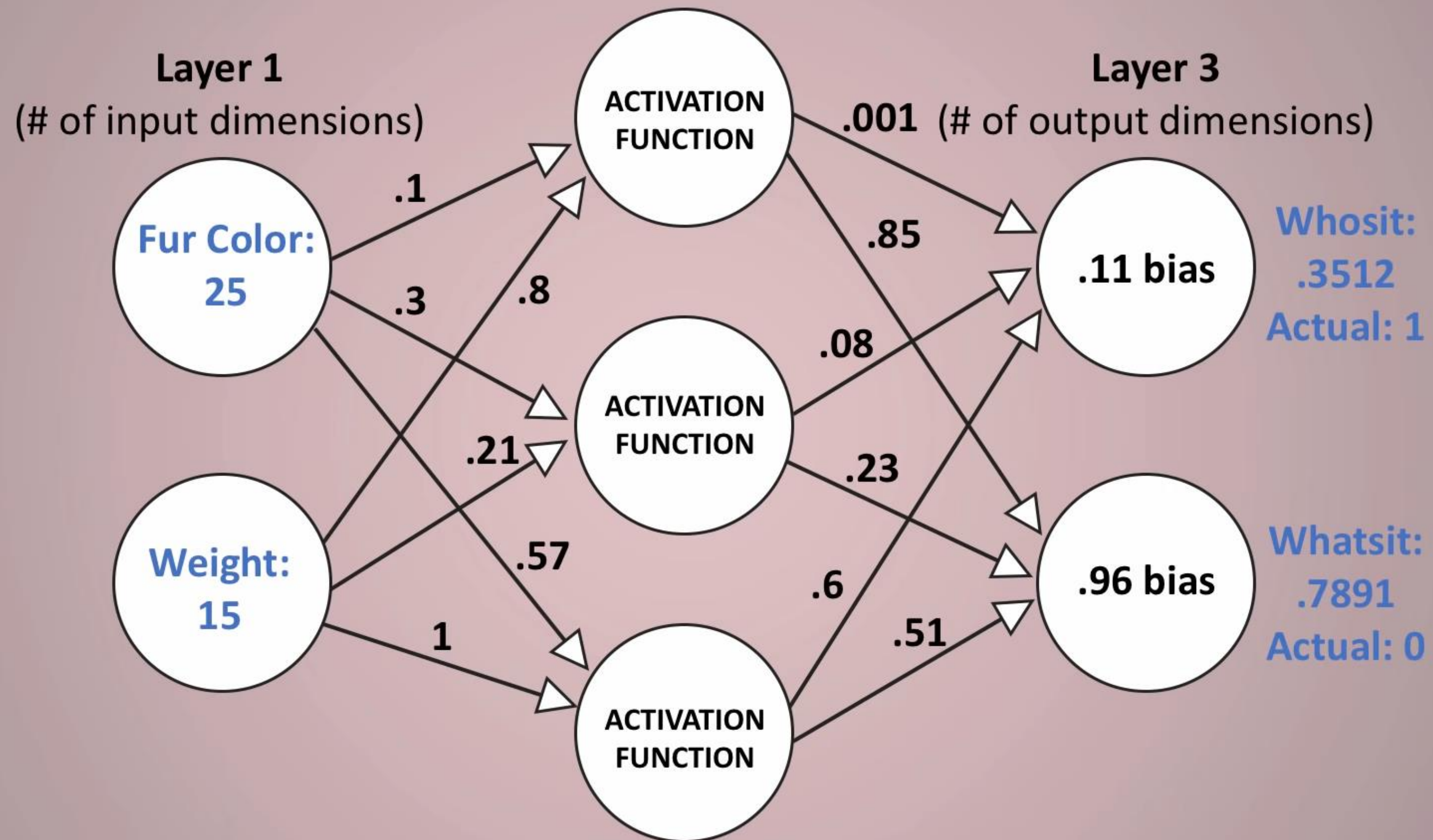
rse: **sum(errors^2) / sum((actual-mean(actual))^2)**

$r^2 =$ **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>

HOW DID WE DO?



WICHTIGE KONZEPTE

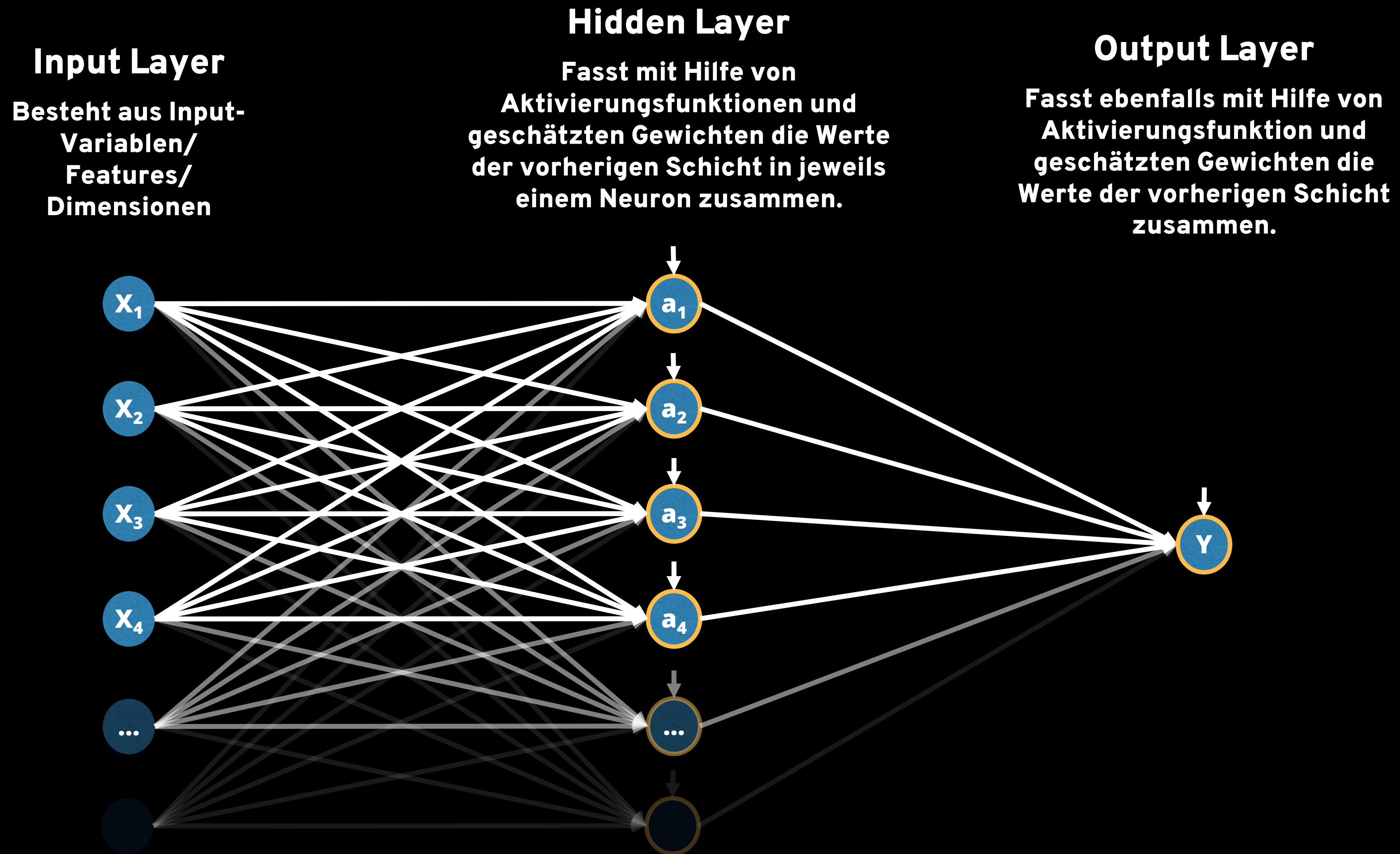
Forward Propagation:

- **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**

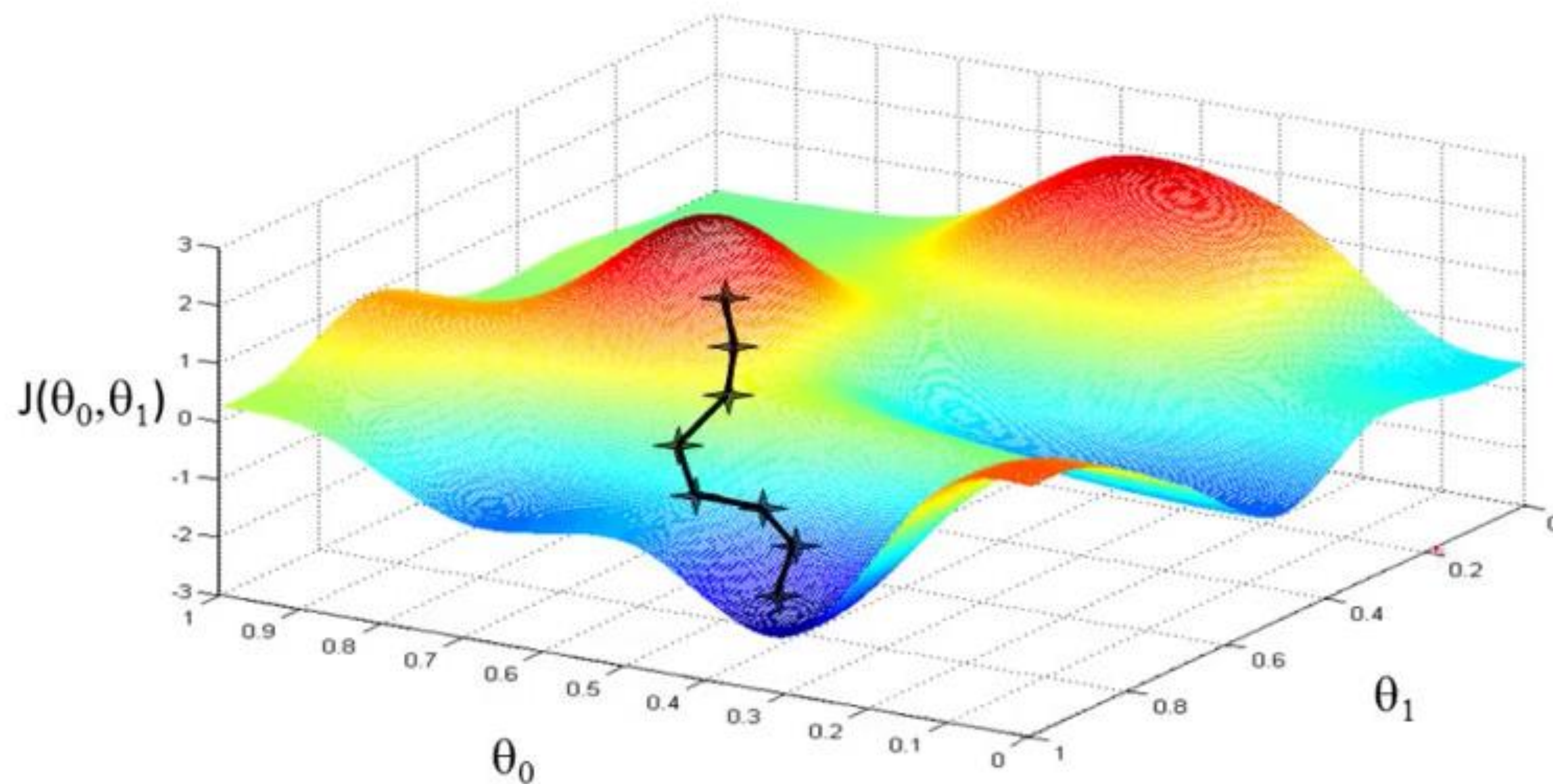
NEURONALE NETZE



HYPERPARAMETER 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“)**

PARAMETER DES OPTIMIZERS „ADAM“

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

ZWISCHEN- FEEDBACK

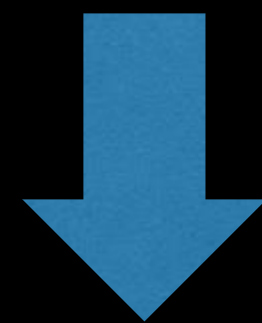


<https://forms.office.com/r/eEN2SSVjHQ>

R VS. PYTHON

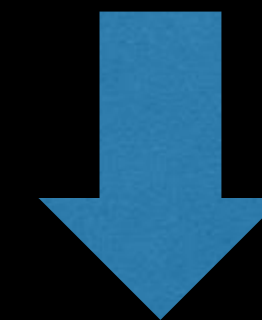
*Statistische Verfahren
außerhalb des ML*

**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 nötig)
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 # Ausgabe 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
```

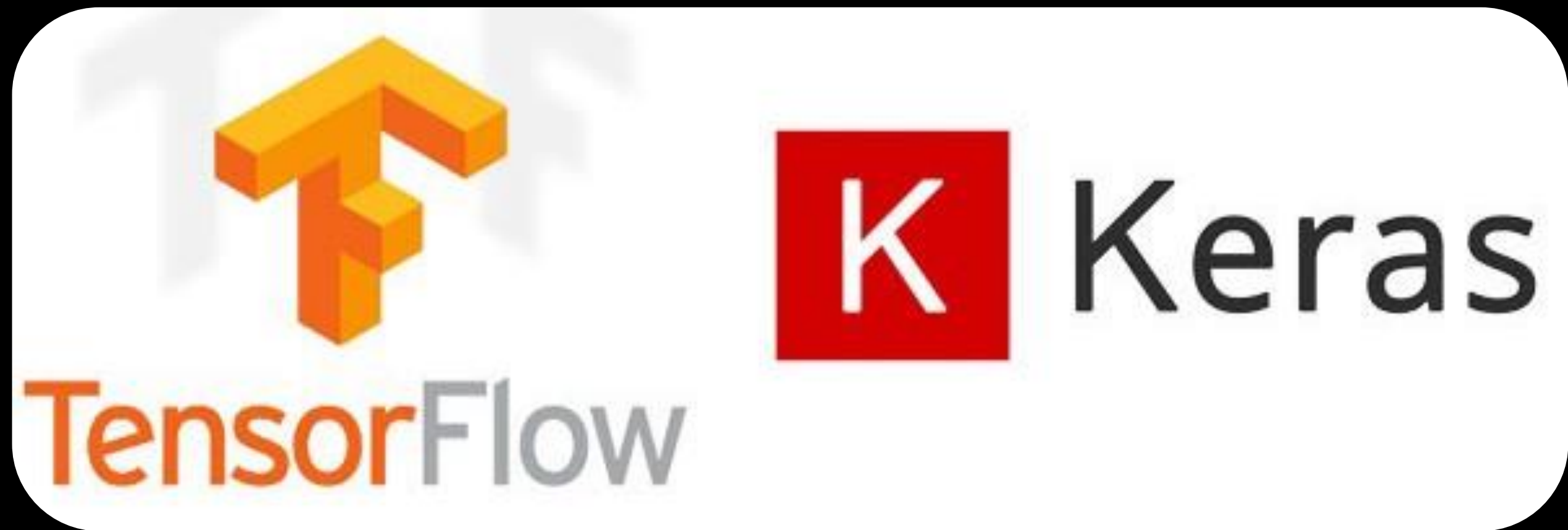
Siehe auch: https://rstudio.github.io/reticulate/articles/r_markdown.html

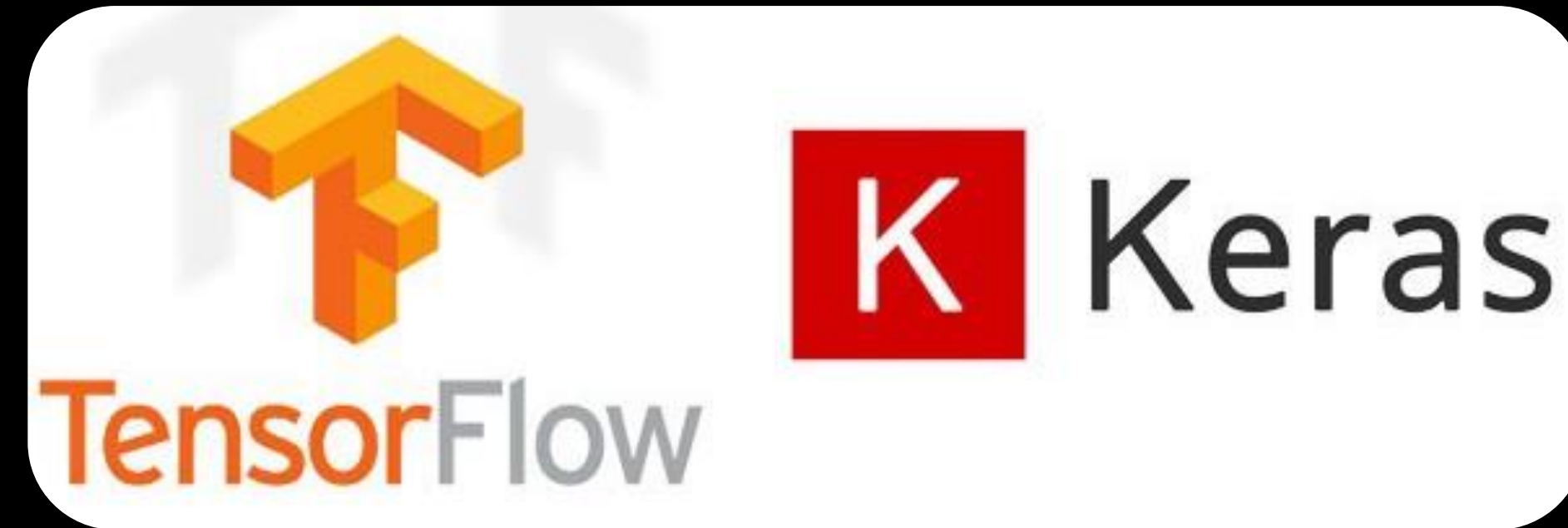
DATENAUFBEREITUNG FÜR TENSORFLOW

```
1
2 ▼ #####
3 ▶ ### Preparation of the Environment
20 ▼ #####
21 ▶ ### Data Import
27 ▼ #####
28 ▼ ### Data Preparation ###
29
30 # Recoding of the variables into one-hot encoded (dummy) variables
31 dummy_list <- c("view", "condition")
32 house_pricing_dummy = dummy_cols(house_pricing, dummy_list)
33
34 # Definition of lists for each one-hot encoded variable (just to make the handling easier)
35 condition_dummies = c('condition_1', 'condition_2', 'condition_3', 'condition_4', 'condition_5')
36 view_dummies = c('view_0', 'view_1', 'view_2', 'view_3', 'view_4')
37
38 |
39 ▼ #####
40 ▼ ### Selection of the Feature Variables and the Label Variable ###
41
42 # Selection of the features (the independent variables used to predict the dependent)
43 #features <- c('sqft_lot', 'waterfront', 'grade', 'bathrooms', view_dummies, condition_dummies)
44 features <- c('sqft_lot', 'waterfront', 'grade', 'bathrooms', condition_dummies, view_dummies)
45 # Selection of the label (the dependent variable)
46 labels <- 'price'
47
48
49 ▼ #####
50 ▼ ### Selection of Training, Validation and Test Data ###
51
52 # Look at the data
53 str(house_pricing_dummy)
54
55 # Setting the random counter to a fixed value, so the random initialization stays the same (the ra
56 set.seed(1)
57
```

BREAKOUT

- **Schaut Euch zusammen den Code zur Datenaufbereitung an und geht ihn Schritt für Schritt einmal durch.**
- **Schreibt Euch Fragen auf!**





- **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)**

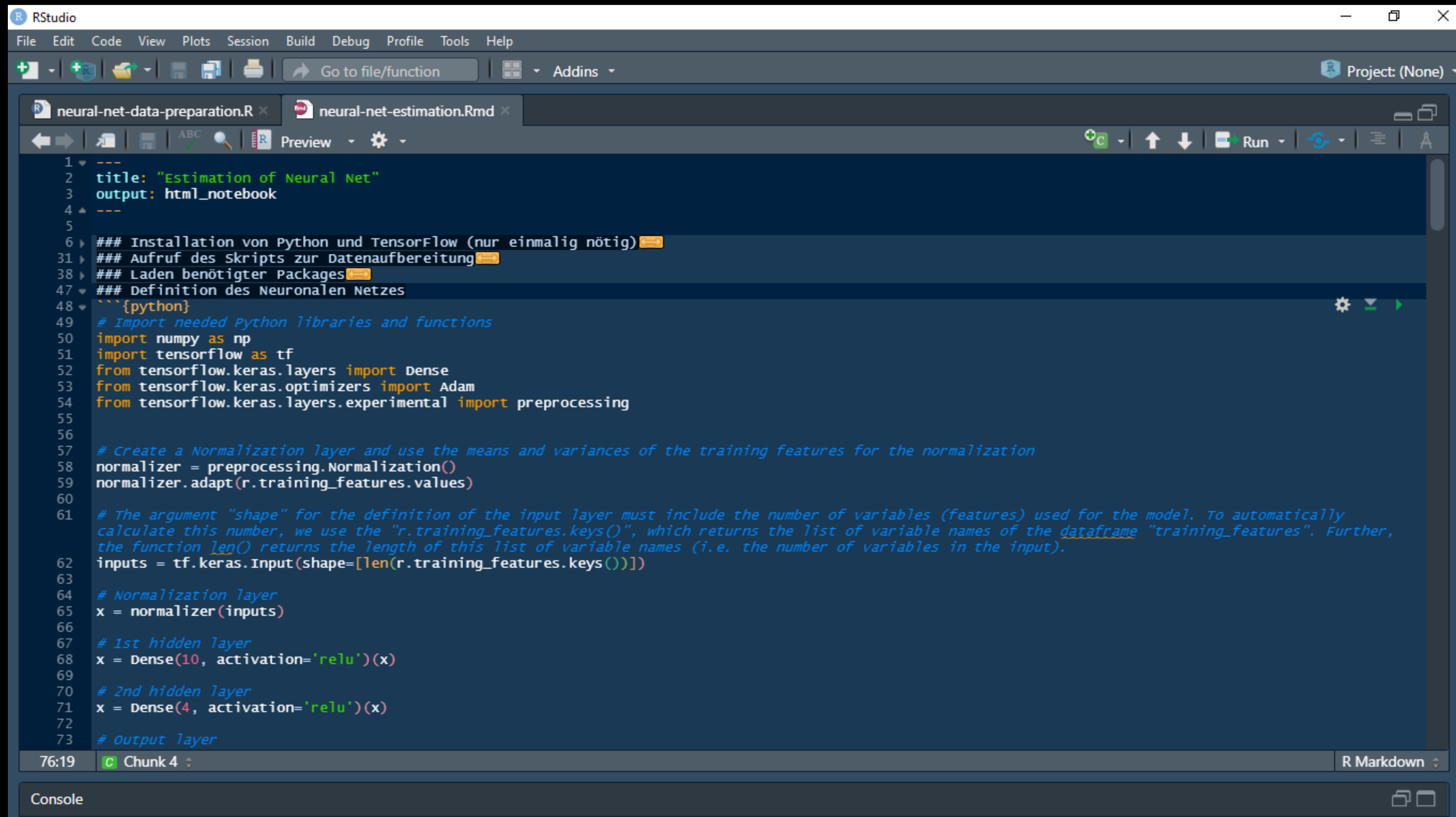
NUTZUNG VON KERAS IN R

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 open files: `neural-net-data-preparation.R` and `neural-net-estimation.Rmd`. The `neural-net-estimation.Rmd` file is active, displaying R Markdown code for defining a neural network. The code includes a title, output format, and several sections for installation, package loading, and network definition. The network definition uses TensorFlow and Keras to create a normalization layer, two hidden layers, and an output layer.

```
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
62 # calculate this number, we use the "r.training_features.keys()", which returns the list of variable names of the dataframe "training_features". Further,
63 # the function len() returns the length of this list of variable names (i.e. the number of variables in the input).
64 inputs = tf.keras.Input(shape=[len(r.training_features.keys())])
65
66 # Normalization layer
67 x = normalizer(inputs)
68
69 # 1st hidden layer
70 x = Dense(10, activation='relu')(x)
71
72 # 2nd hidden layer
73 x = Dense(4, activation='relu')(x)
74
75 # Output layer
```

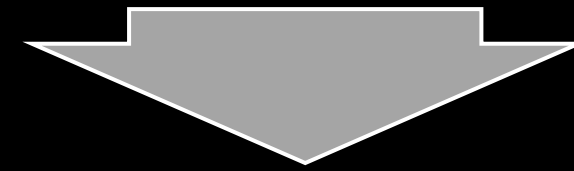
76:19 | C Chunk 4 | R Markdown

Console

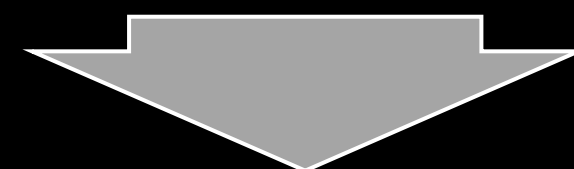
AUFGABEN

- **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.**

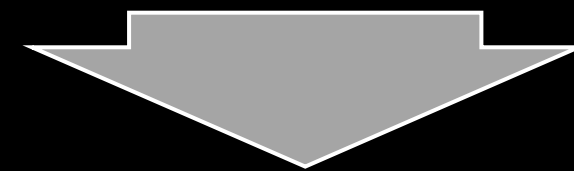
Wahl eines Prognosemodells



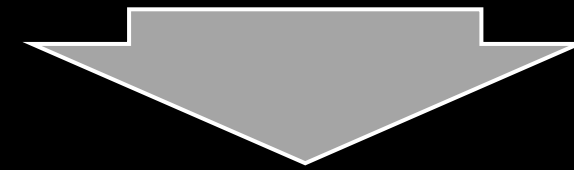
**Teilung der Daten in Trainings- (70%),
Validierungs- (20%) und Testdatensatz (10%)**



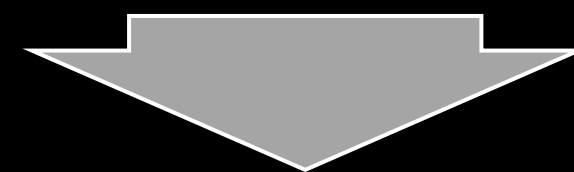
**Optimierung der Modellparameter
anhand des Trainingsdatensatzes**



**Optimierung der Hyperparameter
anhand des Validierungsdatensatzes**



Erweiterung/Verbesserung des Datensatzes



Überprüfung der Modellqualität anhand des Testdatensatzes



Verändern der
Hyperparameter
(modellzentrierte
Optimierung)



Verändern der
Input- Variablen
(datenzentrierte
Optimierung)