

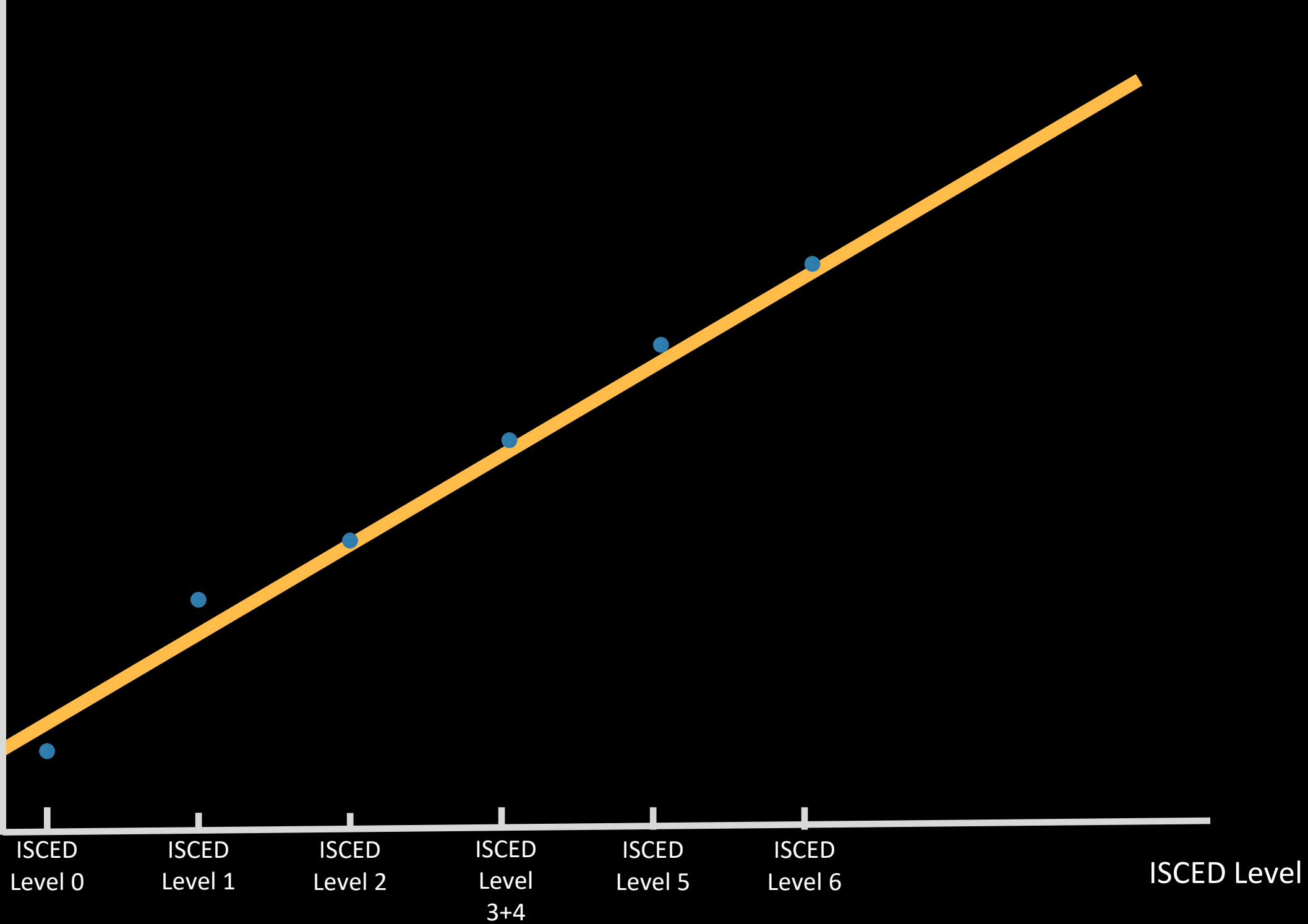
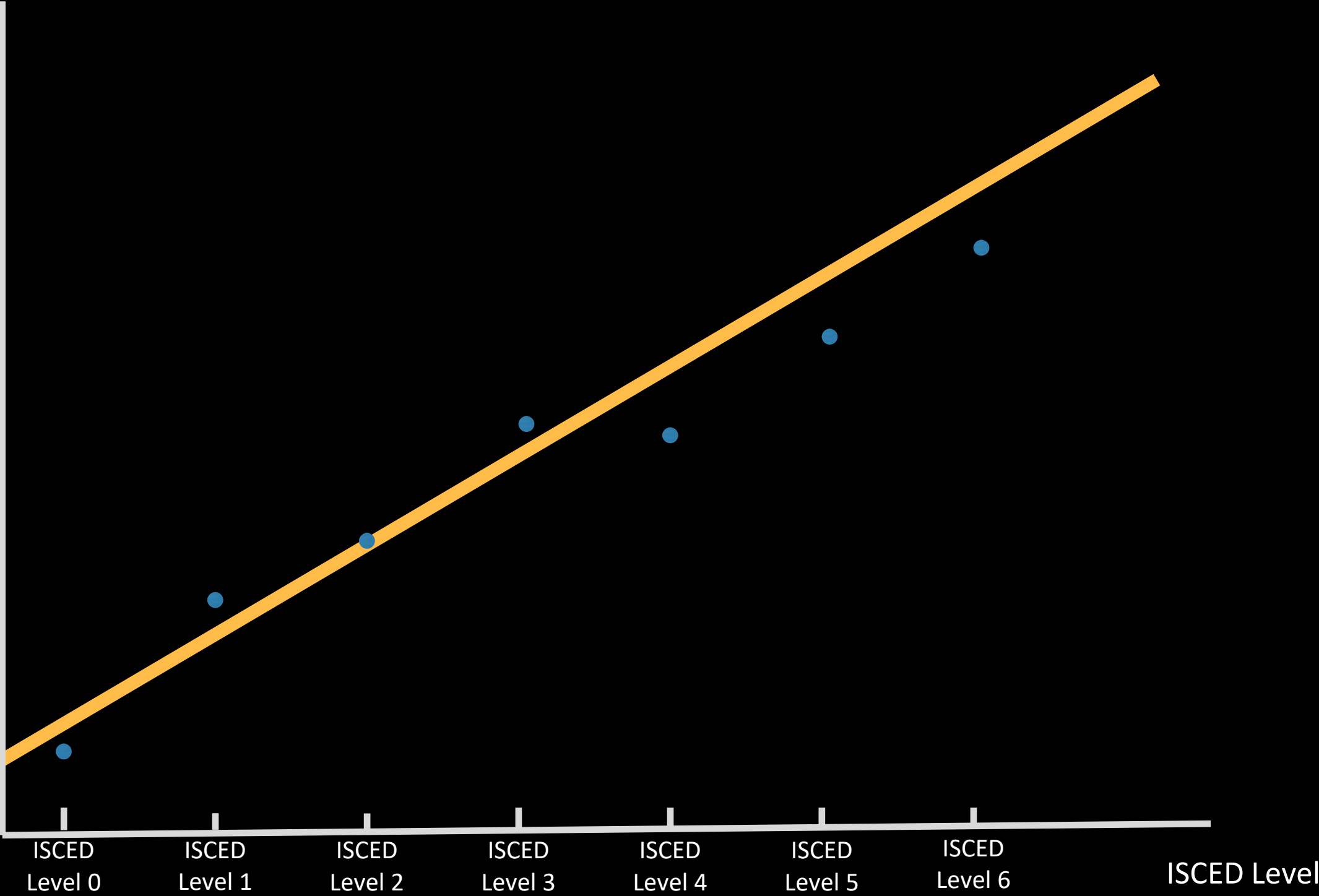
28.11.22

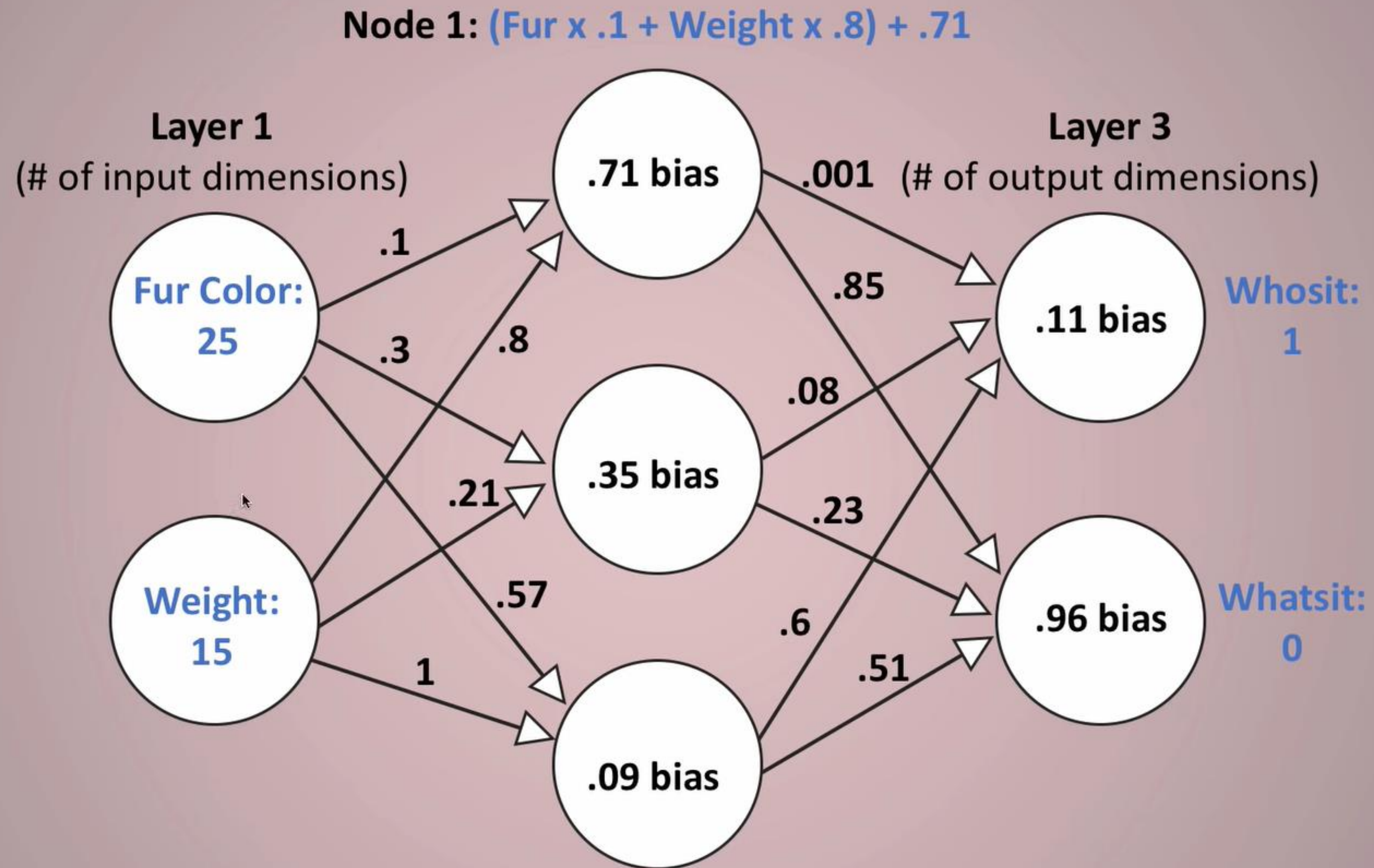
Einführung in Data Science und maschinelles Lernen

NEURONALE NETZE

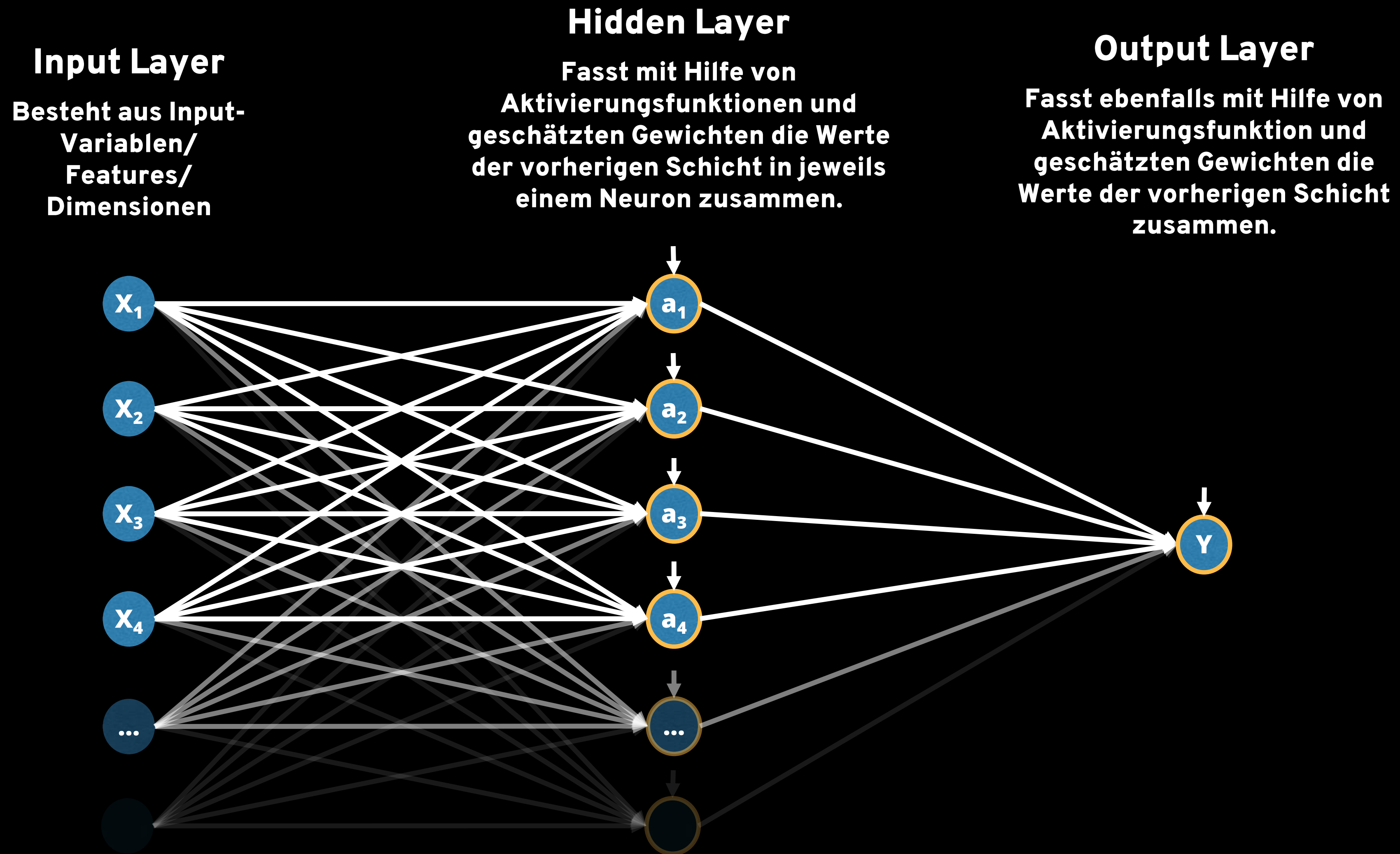
- **Neuronale Netze (NN)**
- **Hyperparameter in NN**
- **Frameworks zur Implementierung von NN**
- **Implementierung eines NN mit TensorFlow und Python**

RELEVANZ VON KATEGORISIERUNGEN





NEURONALE NETZE



WICHTIGE KONZEPTE

Forward Propagation:

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

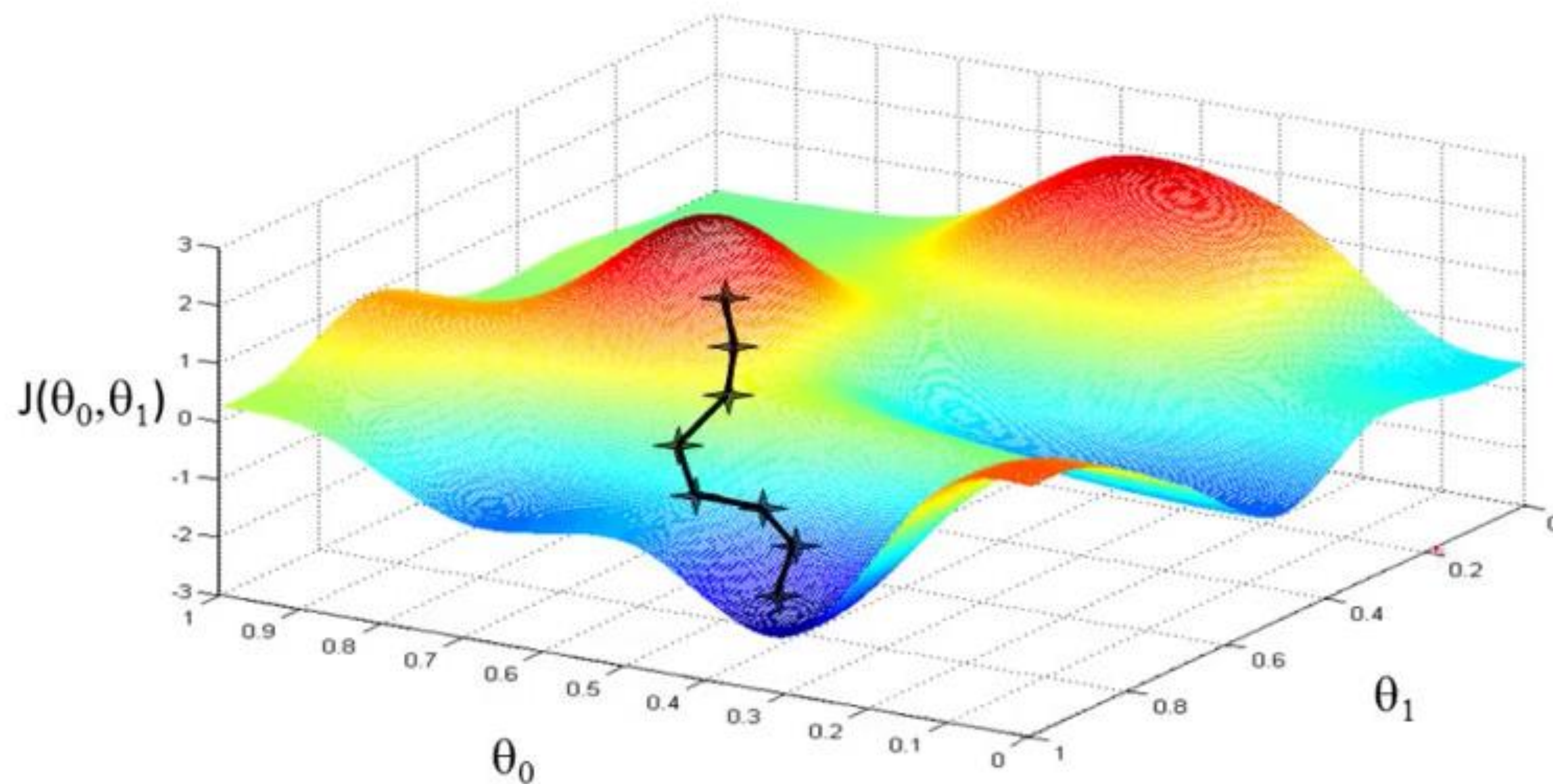
Backward Propagation:

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

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“)**

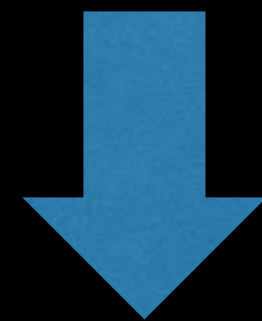
PARAMETER DES OPTIMIZERS „ADAM“

- **Initialer Lernparameter (Schrittweite) für die Optimierung :**
alpha (learning rate)
- **Anteil des aktuellen Gradienten in der Berechnung des nächsten Optimierungsschritts:**
beta1 and beta2 (decay rates)

URSPRÜNGE VON R VS. PYTHON

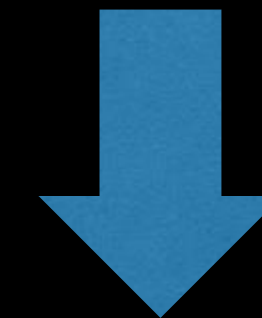
*Statistische Verfahren
außerhalb des ML*

**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

The screenshot displays the RStudio IDE interface with the following components:

- Source Editor:** Contains a file named `neural-net-data-preparation.R` with the following R code:

```
1 1+1
2 print("1+1","3")
3 print(1+1, "3")
4
```
- Environment Panel:** Shows the current session's environment. It includes a search bar, a memory usage indicator (216 MiB), and a list of objects:
 - `r`: [R interface object]
 - `sys`: <module 'sys' (built-i...
- Console:** Displays the output of the R code executed in the source editor, showing the results of the `print` statements:

```
>>> 1+1
2
>>> print("1+1","3")
1+1 3
>>> print(1+1, "3")
2 3
>>>
```
- Files Panel:** Shows the file explorer for the current project, displaying files such as `.RData`, `.Rhistory`, and `house_pricing.csv`.

VERWENDUNG VON PYTHON IN KOMBINATION MIT R

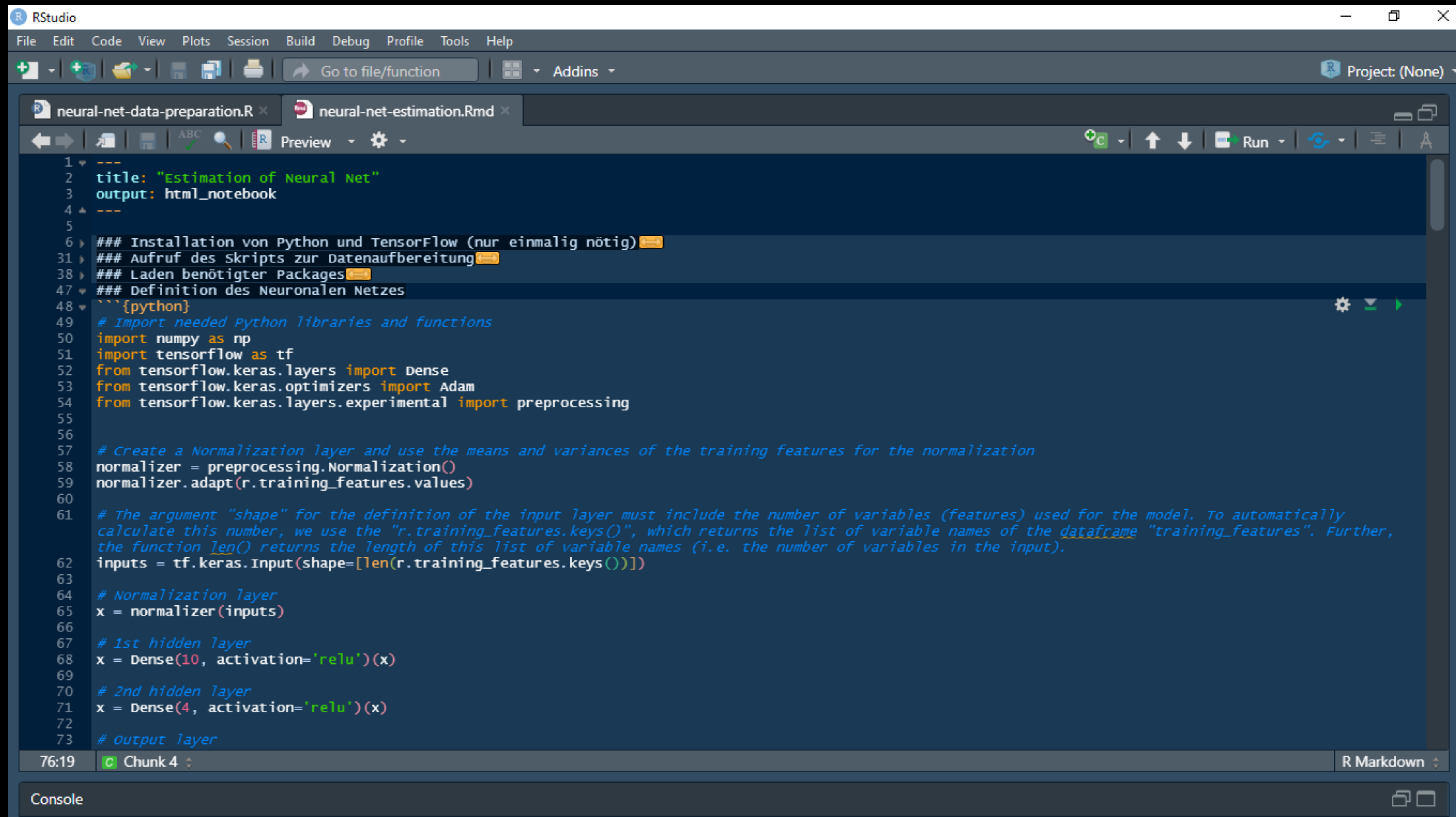
```
31
32 ▾ ```{python}
33   import sys
34   import tensorflow
35
36   # Ausgabe der installierten Python- und TensorFlow-Versionen
37   print("Python Version: " + sys.version+"\nTensorFlow Version: "+tensorflow.__version__)
38
39 ▲ ```
40
41 ▾ ```{r}
42   # Import Libraries
43   library(reticulate)
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 #####
22 ▾ #####
23 ▸ ### Data Import #####
32 ▾ #####
33 ▾ ### Data Preparation ###
34
35 # Preparation of independent variables ('features') by dummy coding the categorical variables
36 features <- as_tibble(model.matrix(price ~ as.factor(bathrooms) + as.factor(zipcode) + as.factor(condition) +
37 names(features)
38
39 # Construction of prepared data set
40 prepared_data <- tibble(label=data$price, features) %>% # inclusion of the dependent variable ('label')
41   filter(complete.cases(.)) # Handling of missing values (here: only keeping rows without missing values)
42
43
44
45
46 ▾ #####
47 ▾ ### Selection of Training, Validation and Test Data ###
48
49 # Set a random seed for reproducibility
50 set.seed(42)
51 # Shuffle the data
52 prepared_data_shuffled <- prepared_data %>% sample_frac(1)
53
54
55 # Calculate the number of rows for each dataset
56 n_total <- nrow(prepared_data_shuffled)
57 n_training <- floor(0.7 * n_total)
58 n_validation <- floor(0.20 * n_total)
59
60
61 # Split the features and labels for training, validation, and test
62 training_features <-
63   prepared_data_shuffled %>% select(-label) %>% slice(1:n_training)
64 validation_features <-
65   prepared_data_shuffled %>% select(-label) %>% slice((n_training + 1):(n_training + n_validation))
66 test_features <-
67   prepared_data_shuffled %>% select(-label) %>% slice((n_training + n_validation + 1):n_total)
```


DEFINITION EINES NEURONALEN NETZES



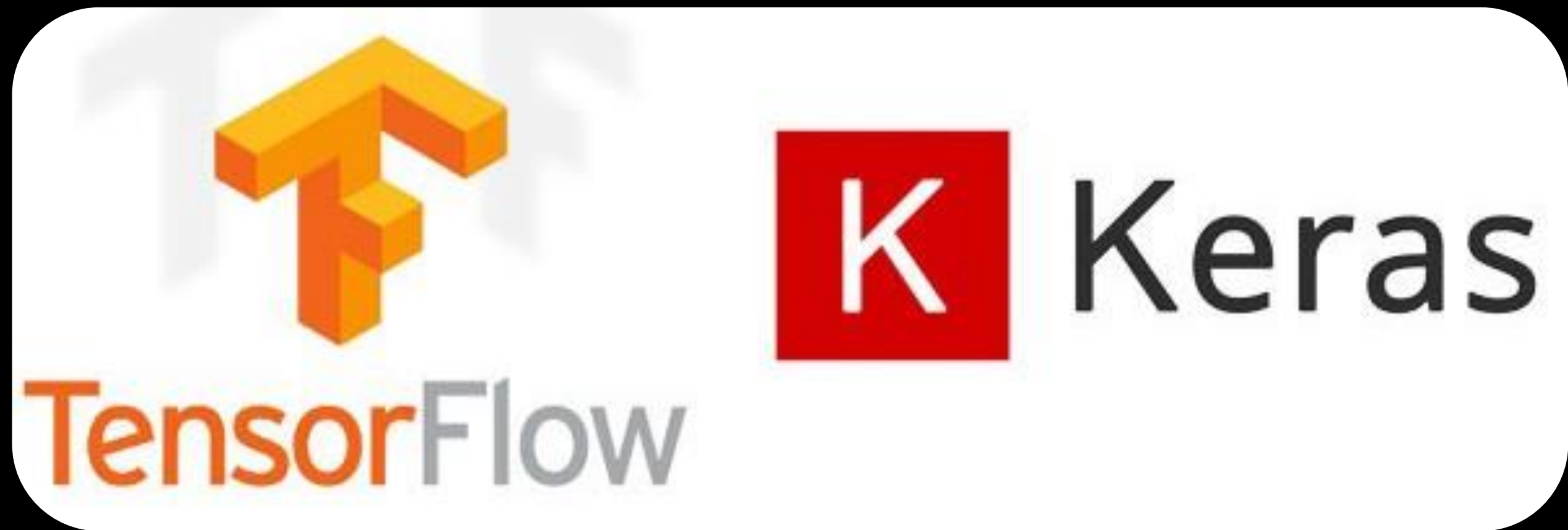
The screenshot shows the RStudio interface with two open files: `neural-net-data-preparation.R` and `neural-net-estimation.Rmd`. The `neural-net-estimation.Rmd` file is active, displaying R Markdown code for defining a neural network. The code includes a title, output format, and several sections for installation, package loading, and network definition. The network definition is written in Python code blocks within the R Markdown file.

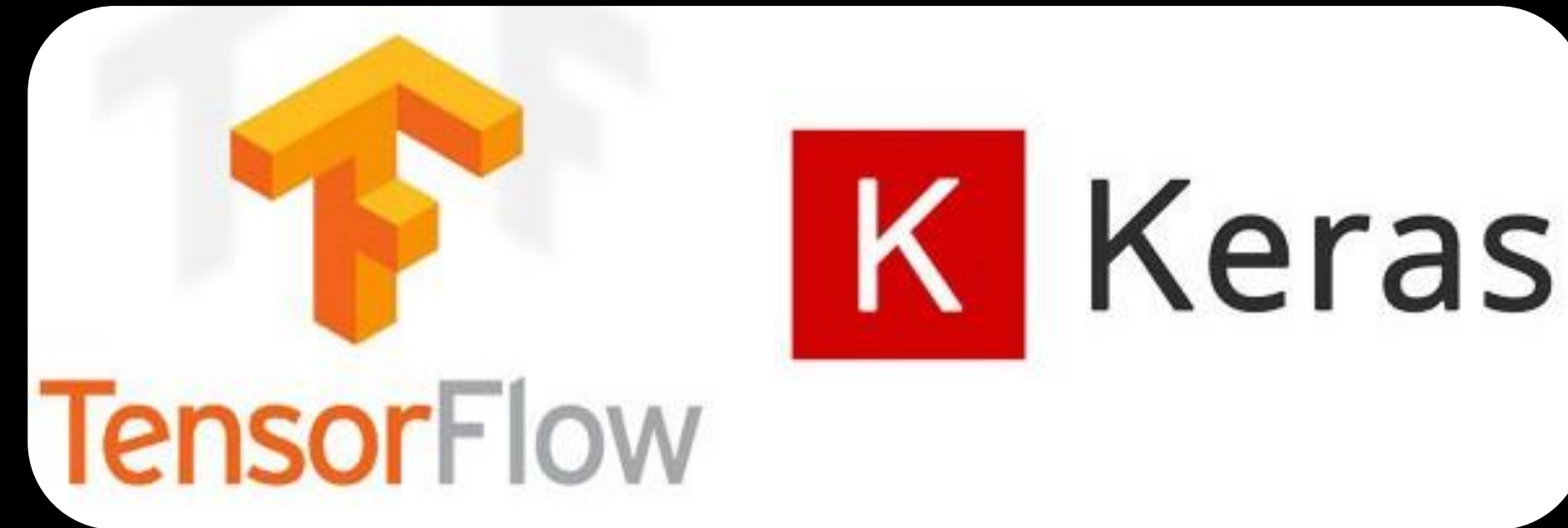
```
1 ---
2 title: "Estimation of Neural Net"
3 output: html_notebook
4 ---
5
6 ### Installation von Python und TensorFlow (nur einmalig nötig)
31 ### Aufruf des Skripts zur Datenaufbereitung
38 ### Laden benötigter Packages
47 ### Definition des Neuronalen Netzes
48 ```{python}
49 # Import needed Python 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
```

The status bar at the bottom indicates the current position is 76:19 in Chunk 4, and the file type is R Markdown.

BREAKOUT

- **Führt zunächst noch einmal die Abfrage des MAPE für Euer Lineares Model durch, falls es bisher noch nicht geklappt hat.**
- **Ergänzt Eure Datenaufbereitung um die Behandlung von fehlenden Werten; zunächst, in dem ihr alle Fälle entfernt, in denen welche vorkommen.**
- **Probiert die Schätzung des neuronalen Netzes anhand des gegebenen Beispielcodes.**





- **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“**

BATCH UND EPOCHE

Batch

- **Eine Menge von Fällen, die genutzt wird, um die Gewichte des Modells zu optimieren.**
- **Ein Optimierungsschritt wird auf Basis eines Batches durchgeführt.**

Epoche

- **Das Modell wurde einmal anhand aller Fälle bzw. aller zu einem Trainingsdatensatz existierenden Batches trainiert.**

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)**
- **[Dieses Video](#) (2 Minuten) zu Dropout-Layern schauen.**
- **Wenn ihr mehr über Python lernen wollt, könnt ihr, z.B., [diese Einführung](#) auf Kaggle nutzen.**