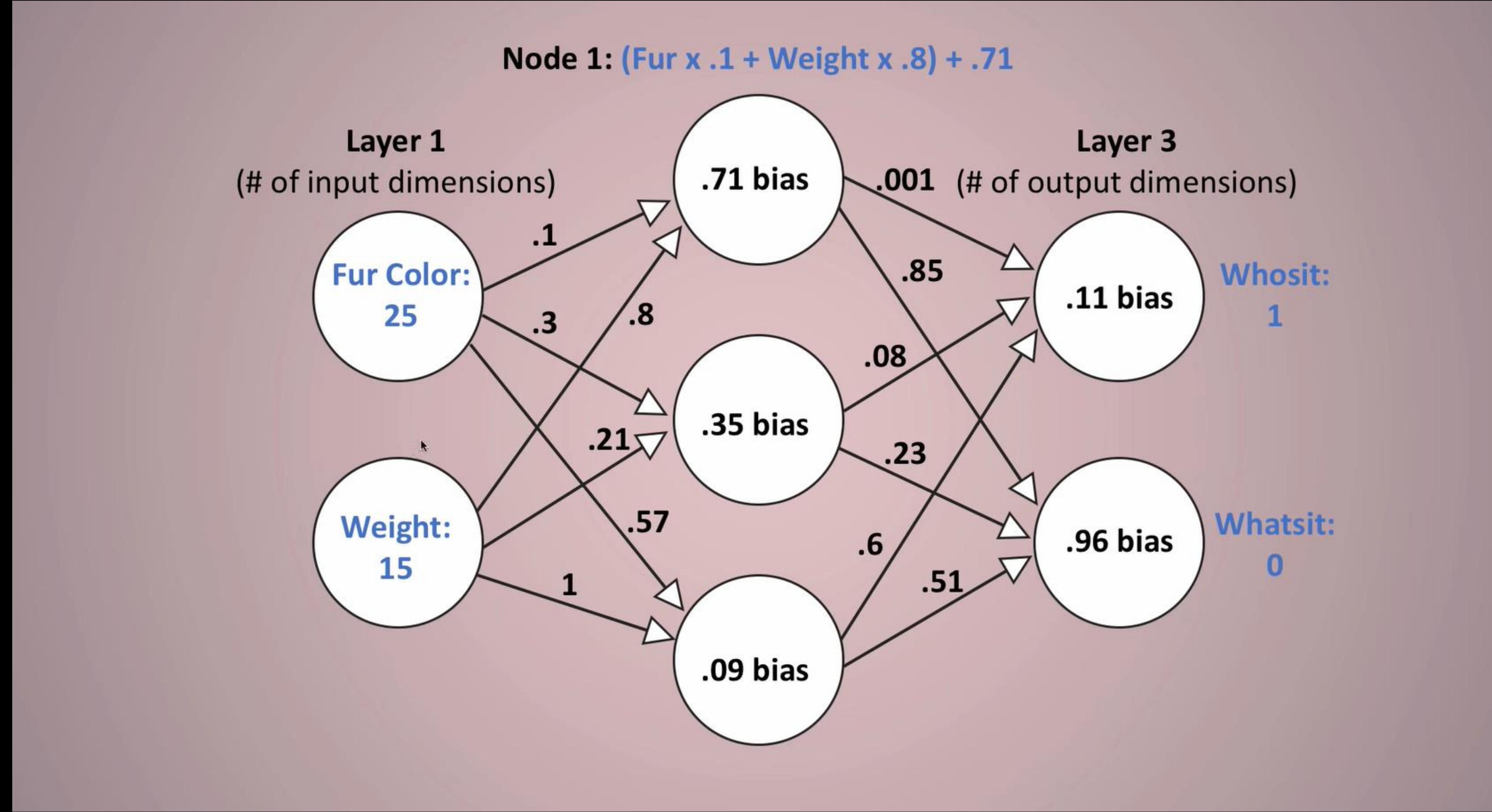


Einführung in Data Science und maschinelles Lernen

NEURONALE NETZE

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

QUIZ



Learning Rate:

How much should this step outcome affect our weights and biases?

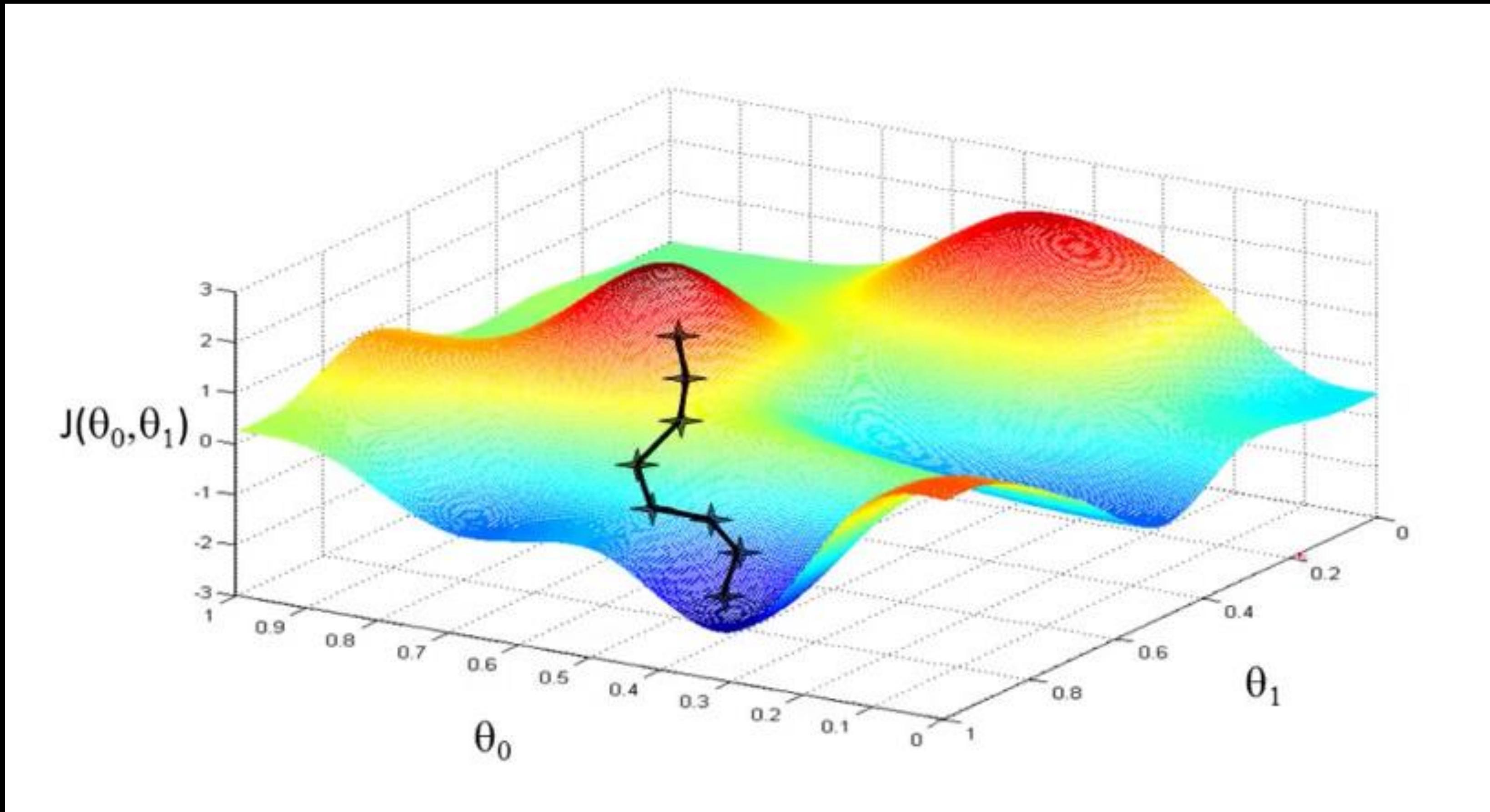
Momentum:

How much should past outcomes affect our weights and biases?

change = (learningRate * delta * value) + (momentum * pastChange)



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“

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

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 Learning Rate des Optimizers**

DATENAUFBEREITUNG

Bei jeder Modellierung müssen die Daten folgende Eigenschaften haben:

- 1. Es darf keine fehlenden Werte geben.**
- 2. Alle Werte müssen Zahlen sein.**
- 3. Kategoriale Variablen sind dummy-kodiert (one-hot-encoded).**

ONE-HOT ENCODING

id	color
1	red
2	blue
3	green
4	blue



id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

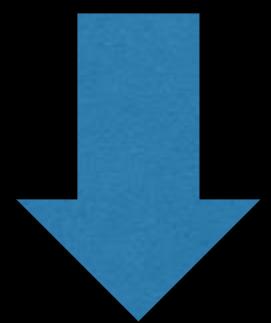
DATENAUFBEREITUNG FÜR DAS TRAINING DES NEURONALEN NETZES

```
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)
```

URSPRÜNGE VON R VS. PYTHON

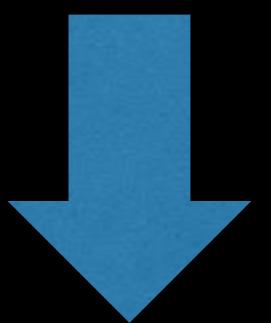
***Statistische Verfahren
außerhalb des ML***

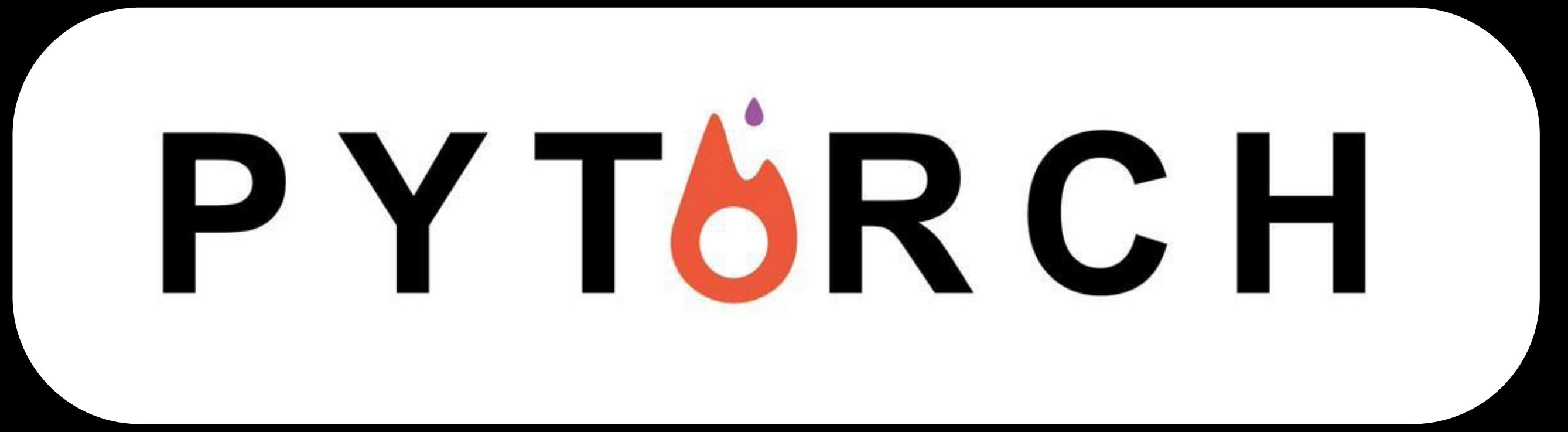
**Mathematik/ Statistik und
Disziplinen mit
Anwendungsbereichen von
Statistik (Ökonometrie,
Psychometrie, Biometrie, ...)**

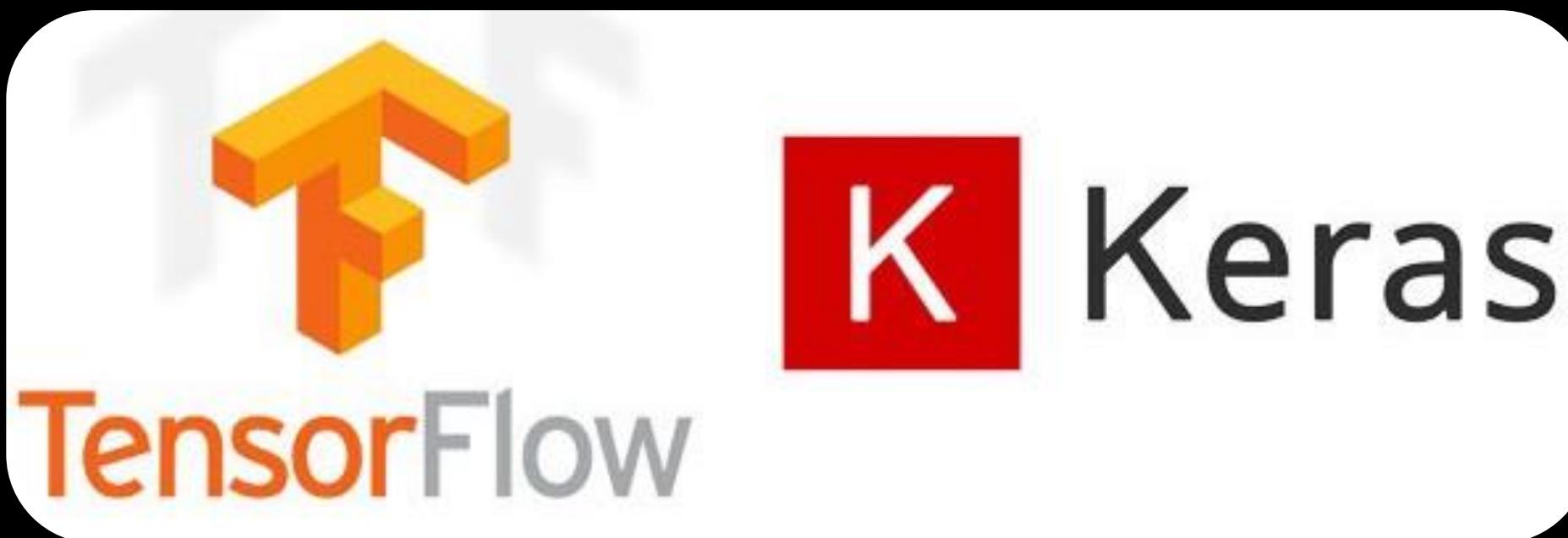


Statistische Verfahren des ML

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







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)

**Dez 2023: Keras 3.0
(unterstützt TensorFlow, Jax und PyTorch)**

NUTZUNG VON KERAS IN RSTUDIO

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

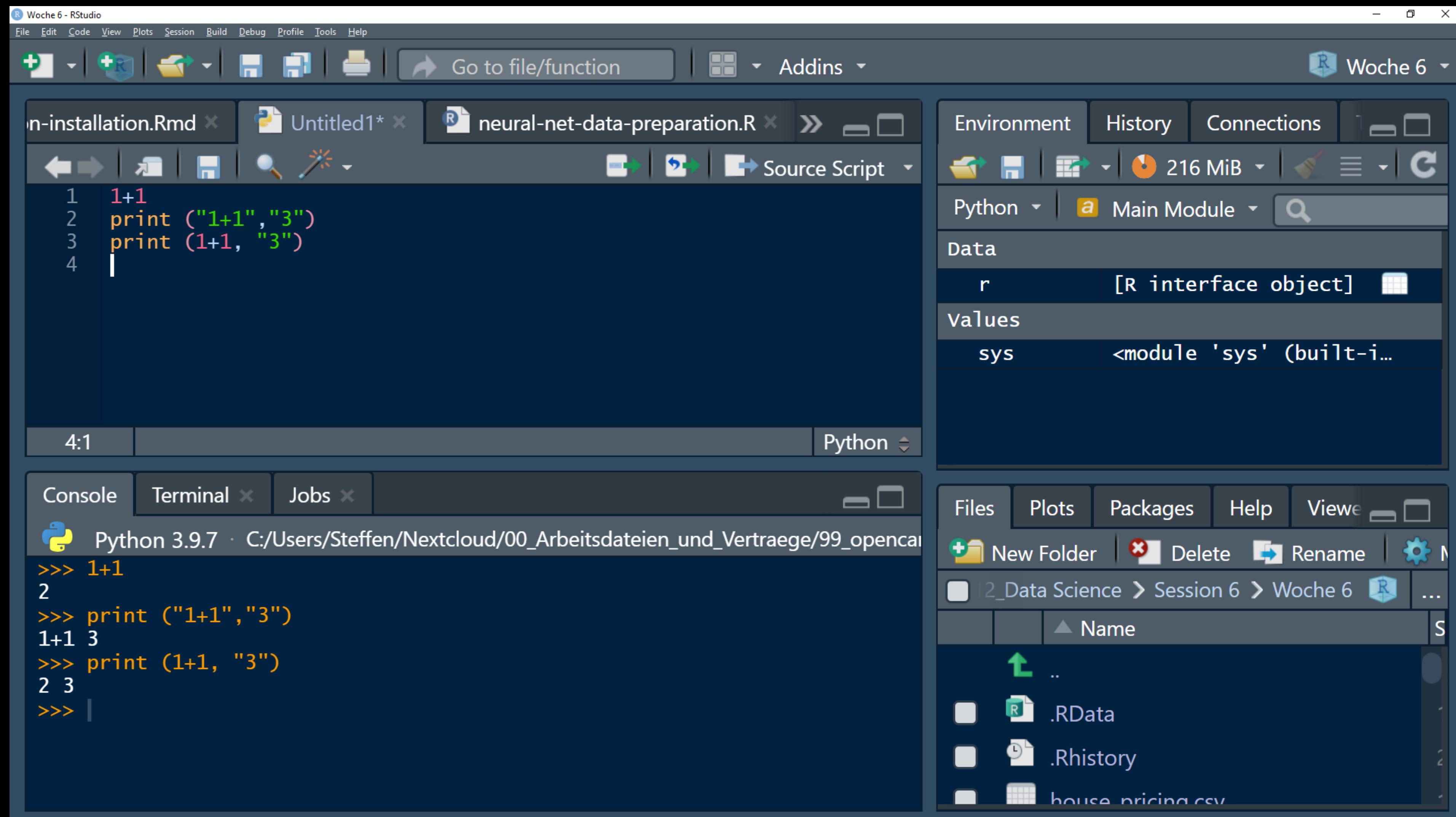
Prinzipiell zwei Varianten :

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

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



VERWENDUNG VON PYTHON IN KOMBINATION MIT R

```
31
32  ````{python}
33  import sys
34  import tensorflow
35
36  # Augabe 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
```

GOOGLE COLAB

- **Cloud-basierte Plattform**
- **Kostenloser Zugang zu GPUs**
- **Vorinstallierte Bibliotheken**
- **Kollaborativ und teilbar**
- **Integration mit Google Drive**

GOOGLE DRIVE

- Jeglicher Speicher in Colab ist temporär
- Direkter Zugriff aus Colab ermöglicht permanente Datenverwaltung
- Automatische Synchronisation und Backups

DEFINITION UND OPTIMIERUNG EINES NEURONALEN NETZES IN COLAB

The screenshot shows a Google Colab notebook interface. The title bar reads 'neural-net-estimation.ipynb'. The menu bar includes 'Datei', 'Bearbeiten', 'Anzeige', 'Einfügen', 'Laufzeit', 'Tools', 'Hilfe', and 'Alle Änderungen wurden gespeichert'. The toolbar on the left contains icons for file operations like new file, new folder, search, and refresh. The main area displays a section titled 'Mounting Google Drive and Loading Data'. It starts with a note: 'First, let's mount the Google Drive to access the preprocessed data. Ensure that your data files are stored in the correct path within your Google Drive.' Below this, cell [55] contains the Python code for mounting Google Drive:

```
[55]: # Mounting your Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

The output of this cell indicates that the drive is already mounted at /content/drive:

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Cell [56] changes the working directory:

```
[56]: # Change working directory
import os
os.chdir("/content/drive/MyDrive/Colab Notebooks/Introductory Course")
```

Cell [57] imports pandas and defines paths for training and validation data:

```
[57]: import pandas as pd

# Import your training and validation data
# Replace 'path_to_data' with the actual subfolder of your data in Google Drive
data_path = './csv_data'
training_features = pd.read_csv(f'{data_path}/training_features.csv')
training_labels = pd.read_csv(f'{data_path}/training_labels.csv')
validation_features = pd.read_csv(f'{data_path}/validation_features.csv')
validation_labels = pd.read_csv(f'{data_path}/validation_labels.csv')
test_features = pd.read_csv(f'{data_path}/test features.csv')
```

The status bar at the bottom shows '0 s' and 'Abgeschlossen um 00:20'.

BREAKOUT

- Ergänzt Eure Datenaufbereitung um die im [**Beispielskript**](#) durchgeföhrten Schritte:
 - 1) One-Hot-Kodierung kategorialer Variablen
 - 2) Entfernen von Fällen mit Missing Values
 - 3) Export der Trainings-, Validierungs- und Testdaten als CSV
- Öffnet das [**Jupyter-Notebook**](#) zur Schätzung des Neuronalen Netzes in Colab und speichert eine Kopie in Eurem Google Drive.
- Passt das Notebook an, um das neuronale Netz für Eure Daten zu optimieren.

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.
- Dieses Video (7 Minuten) als zusätzliche Einführung in die Definition von neuronalen Netzen schauen.