

# Einführung in Data Science und maschinelles Lernen

**BEHANDLUNG VON  
FEHLENDEN WERTEN**

- Quiz
- Verschiedene Layer zur Implementierung eines NN
- Behandlung fehlender Werte
- (Support-Vektor-Maschinen)

# QUIZ



<https://forms.office.com/e/mZpJhXW90w>

# BATCH UND EPOCHE

## Batch

- **Eine Teilmenge 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

# DROPOUT LAYER ALS FORM DER REGULARISIERUNG

- Setzt bei jedem Iterationsschritt die Aktivierungen des vorherigen Layers mit der gewählten Wahrscheinlichkeit auf 0.
- Hilft Overfitting zu vermeiden.
- Integriert Redundanz in das Netz.
- Wird nur angewendet im Training, für die Inferenz werden keine Werte verworfen.

# NEURONALES NETZ MIT DROPOUT LAYER

```
model = Sequential([
    InputLayer(input_shape=(len(r.training_features.keys()), )),
    BatchNormalization(),
    Dense(10, activation='relu'),
    Dropout(.2),
    Dense(4, activation='relu'),
    Dense(1)
])
```

R code - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

missing-data.R neural-net-estimation-with-dropout.Rmd

16 **### Definition des Neuronalen Netzes**  
17 **```{python}**  
18 **# Import needed Python libraries and functions**  
19 **import numpy as np**  
20 **import tensorflow as tf**  
21 **from tensorflow.keras.models import sequential**  
22 **from tensorflow.keras.layers import InputLayer, Dense, BatchNormalization, Dropout**  
23 **from tensorflow.keras.optimizers import Adam**  
24  
25 **# The argument "input\_shape" for the definition of the input layer must include the number input variables (features) used for the model. To automatically calculate this number, we use the function 'r.training\_features.keys()', which returns the list of variable names of the dataframe 'training\_features'. Then, the function 'len()' returns the length of this list of variable names (i.e. the number of variables in the input).**  
26  
27 **model = sequential([**  
28  **InputLayer(input\_shape=(len(r.training\_features.keys()), ), ),**  
29  **BatchNormalization(),**  
30  **Dense(10, activation='relu'),**  
31  **Dropout(.2),**  
32  **Dense(4, activation='relu'),**  
33  **Dense(1)**  
34 **])**  
35  
36 **# Ausgabe einer Zusammenfassung zur Form des Modells, das geschaezt wird (nicht notwendig)**  
37 **model.summary()**  
38  
39 **...**

Model: "sequential\_1"

Layer (type)	output Shape	Param #
batch_normalization_1 (Batch Normalization)	(None, 104)	416
dense_3 (Dense)	(None, 10)	1050
dropout (Dropout)	(None, 10)	0
dense_4 (Dense)	(None, 4)	44
dense_5 (Dense)	(None, 1)	5

Total params: 1,515  
Trainable params: 1,307  
Non-trainable params: 208

37:16 C Chunk 2 R Markdown

Console Terminal Jobs

Python 3.9.7 · C:/Users/Steffen/Nextcloud/00\_Arbeitsdateien\_und\_Vertraege/99\_opencampus/06\_Kurse/12\_Data Science/Session 8/code/ ↵

```
>>>
>>>
>>> # Definition der Kosten-(Loss-)Funktion und der optimierungsfunction mit seinen hyperparametern
>>> model.compile(loss="mse", optimizer=Adam(learning_rate=0.001))
...
```

Environment

Python Main

Data

his... <t... R [R...]

Values

Adam <clas...  
Batch... <clas...  
Dense <clas...

Files Plot

Python: class D

>>> laye  
>>> data  
>>> prin  
[[0. 1.]  
[2. 3.]  
[4. 5.]  
[6. 7.]  
[8. 9.]  
>>> outp  
>>> prin  
tf.Tenso  
[[ 0.  
[ 2.5  
[ 5.  
[ 7.5  
[10.  
  
Args:  
rate:  
noise\_  
bina  
For  
(ba  
you  
you  
seed:  
  
Call arg  
inputs  
traini  
trai

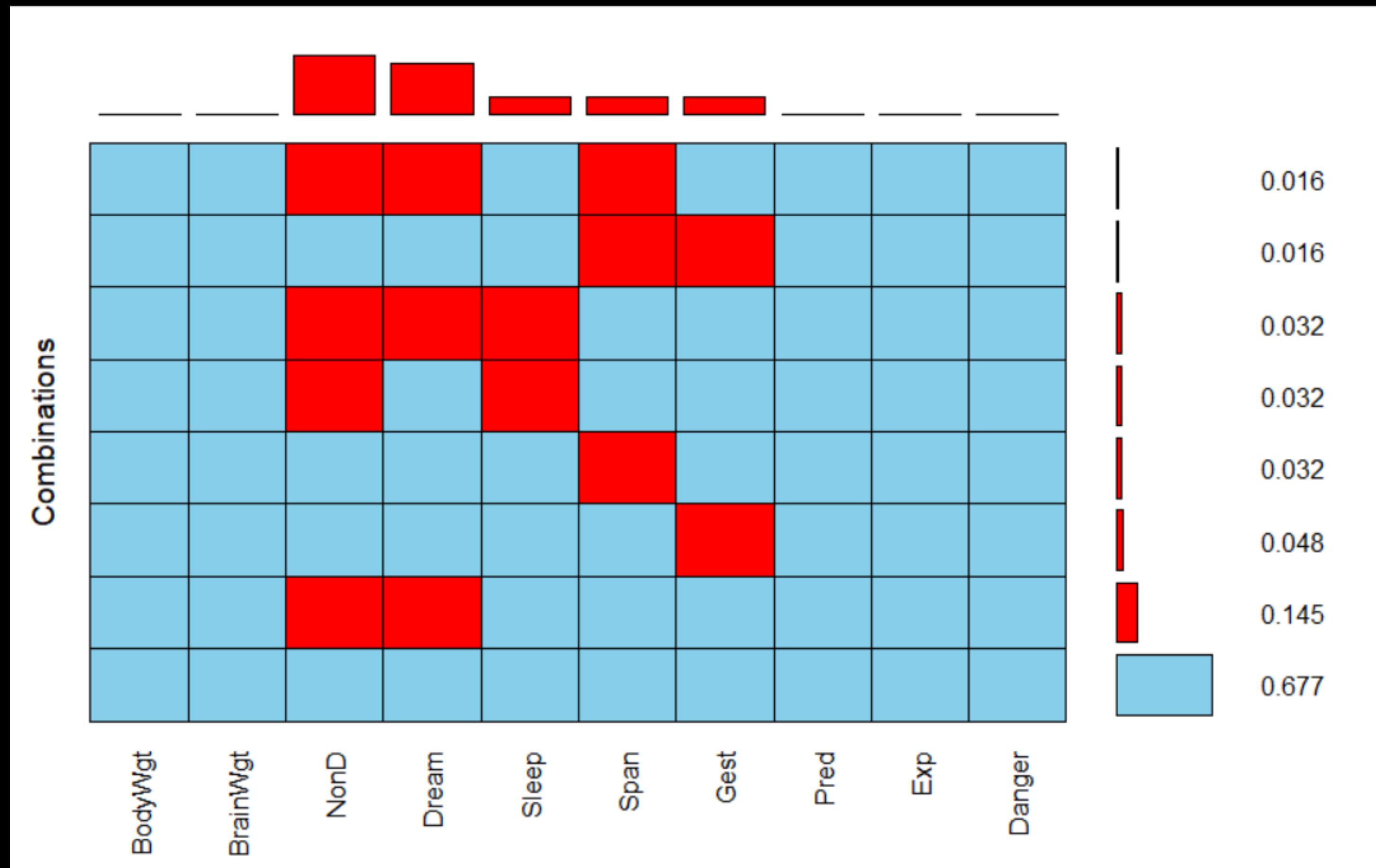
# **GRÜNDE FÜR FEHLENDE WERTE**

- **Nicht gegebene Antworten in Umfragen**
- **Zusammenführung von Daten aus verschiedenen Quellen mit unterschiedlichen Variablen**
- **Technische Probleme in der Datenerhebung oder Aufzeichnung**
- **...**

# TYPEN VON FEHLENDEN WERTEN

- Missing Completely at Random (MCAR)
- Missing at Random (MAR)
- Missing not at Random (MNAR)

# VISUALISIERUNG FEHLENDER WERTE



**VIM package**  
**(Visualization and  
Imputation of  
Missing Values)**

`aggr(dataset,  
combined=TRUE,  
numbers=TRUE)`

Kowarik, A., & Templ, M. (2016). Imputation with the R Package VIM. *Journal of Statistical Software*, 74, 1–16.

# BEHANDLUNG VON FEHLENDEN WERTE

- **Zugehörige Fälle löschen (listwise deletion)**
- **Einfache „Spender“-basierte Imputation (donor-based)**
  - **Mittelwertschätzung (bzw. Median oder Mode)**
  - **nach „Ähnlichkeit“ (Hot-Deck Imputation)**
  - **durch minimalen Abstand (k Nearest Neighbors)**
- **Einfache modellbasierte Imputation**
  - **Iterative Regression**
- **Multiple Imputation**

# HOT-DECK IMPUTATION

## Nach Domänen

The diagram illustrates the domain-based hot-deck imputation process for the 'PhysActive' dataset. It shows two versions of the dataset: the original with missing values and the imputed version.

**Original Data:**

PhysActive	Weight
TRUE	51
TRUE	63
FALSE	98
TRUE	NA
FALSE	81
FALSE	88

A red box highlights the missing value 'NA' in the fourth row. A red arrow points from this row to the bottom section.

**Imputed Data:**

PhysActive	Weight
TRUE	51
TRUE	63
FALSE	98
TRUE	NA
FALSE	81
FALSE	88

A green arrow points from the bottom section back up to the fourth row of the original data, indicating that the missing value was replaced by the value from the same domain (TRUE).

## Nach Korrelation

The diagram illustrates the correlation-based hot-deck imputation process for the 'Height' and 'Weight' datasets. It shows two versions of each dataset: the original with missing values and the imputed version.

**Height Dataset:**

**Original Data:**

Height	Weight
150	51
161	63
189	98
155	NA
182	81
184	88

A red box highlights the missing value 'NA' in the fourth row. A red arrow points from this row to the bottom section.

**Imputed Data:**

Height	Weight
150	51
155	NA
161	63
182	81
184	88
189	98

A green arrow points from the bottom section back up to the fourth row of the original data, indicating that the missing value was replaced by the value from the same domain (Height).

**Weight Dataset:**

**Original Data:**

Height	Weight
150	51
161	63
189	98
155	NA
182	81
184	88

A red box highlights the missing value 'NA' in the fourth row. A red arrow points from this row to the bottom section.

**Imputed Data:**

Height	Weight
150	51
155	NA
161	63
182	81
184	88
189	98

A green arrow points from the bottom section back up to the fourth row of the original data, indicating that the missing value was replaced by the value from the same domain (Weight).

# K-NEAREST NEIGHBORS (KNN)

**Suche nach den k Fällen mit minimalem Abstand**

- je nach Variablentyp unterschiedliche Abstandsmessung
- Zusammenführung der Abstände über eine Summenfunktion

**Verschiedene Vorgehensweisen zur Berechnung des Imputationswerts:**

- Der Wert mit minimalem Abstand wird genommen (1NN)
- Zufällige Ziehung aus den k Fällen
- Berechnung aus den k-Fällen über den (gewichteten Mittelwert)

# ITERATIVE REGRESSION

## 1) Vorhersage fehlender Werte in A

A	B	C	D
5	34	NA	1
1	22	NA	4
NA	65	55	2
4	87	27	2
NA	23	10	1

# ITERATIVE REGRESSION

## 1) Vorhersage fehlender Werte in A

A	B	C	D
5	34	NA	1
1	22	NA	4
5	65	55	2
4	87	27	2
2	23	10	1

# ITERATIVE REGRESSION

2) Vorhersage Fehlender Werte in C mit den imputierten  
werten von A

A	B	C	D
5	34	NA	1
1	22	NA	4
5	65	55	2
4	87	27	2
2	23	10	1

# ITERATIVE REGRESSION

2) Vorhersage Fehlender Werte in C mit den imputierten  
werten von A

A	B	C	D
5	34	32	1
1	22	16	4
5	65	55	2
4	87	27	2
2	23	10	1

# ITERATIVE REGRESSION

3) Vorhersage fehlender Werte in A mit den imputierten Werten von C

A	B	C	D
5	34	32	1
1	22	16	4
5	65	55	2
4	87	27	2
2	23	10	1

# ITERATIVE REGRESSION

3) Vorhersage fehlender Werte in A mit den imputierten Werten von C

A	B	C	D
5	34	32	1
1	22	16	4
NA	65	55	2
4	87	27	2
NA	23	10	1

→ Wiederholung bis keine Änderung mehr eintritt

# ITERATIVE REGRESSION

- 1) Gehe schrittweise durch alle Variablen des Datensatz
  - 2) Stelle dabei für jede Variable eine Regressionsmodell basierend auf allen anderen Variablen auf
  - 3) Berechne für alle fehlenden Werte eine Vorhersage
- 
- Jetzt wiederhole Schritt 1) bis 3) erneut und schätze die fehlenden Werte erneut - dieses Mal mit den bereits imputierten fehlenden Werten.
  - Wiederhole dies, bis sich die imputierten Werte nicht mehr ändern.

# IMPUTATION EXAMPLES

```
1 ---  
2 title: "Missing Values"  
3 output: html_notebook  
4 ---  
5  
6 ``{r}  
7 # Prepare Environment  
8 library(readr)  
9 library(VIM)  
10 library(dplyr)  
11 library(ggplot2)  
12  
13 options(datatable.use.index = TRUE)  
14  
15 ``  
16  
17 ``{r}  
18 # VIM Aggregation Plot  
19 sleep %>%  
20   aggr(combined=TRUE, numbers=TRUE)  
21  
22 ``  
23  
24  
25 ``{r}  
26 # Listwise deletion  
27 sleep_deletion <- na.omit(sleep)  
28  
29 sleep_deletion %>%  
30   aggr(combined=TRUE, numbers=TRUE)  
31  
32 ``  
33  
34  
35 ``{r}  
36 # VIM Hot-Deck Imputation  
37  
38 sleep_hotdeck1 <- sleep %>%  
39   hotdeck()  
40 sleep_hotdeck1 %>%  
41   aggr(combined=TRUE, numbers=TRUE)  
42 ggplot(sleep_hotdeck1) +  
43   geom_point(aes(x=sleep, y=Dream, color=Sleep_imn))
```

# BREAKOUT

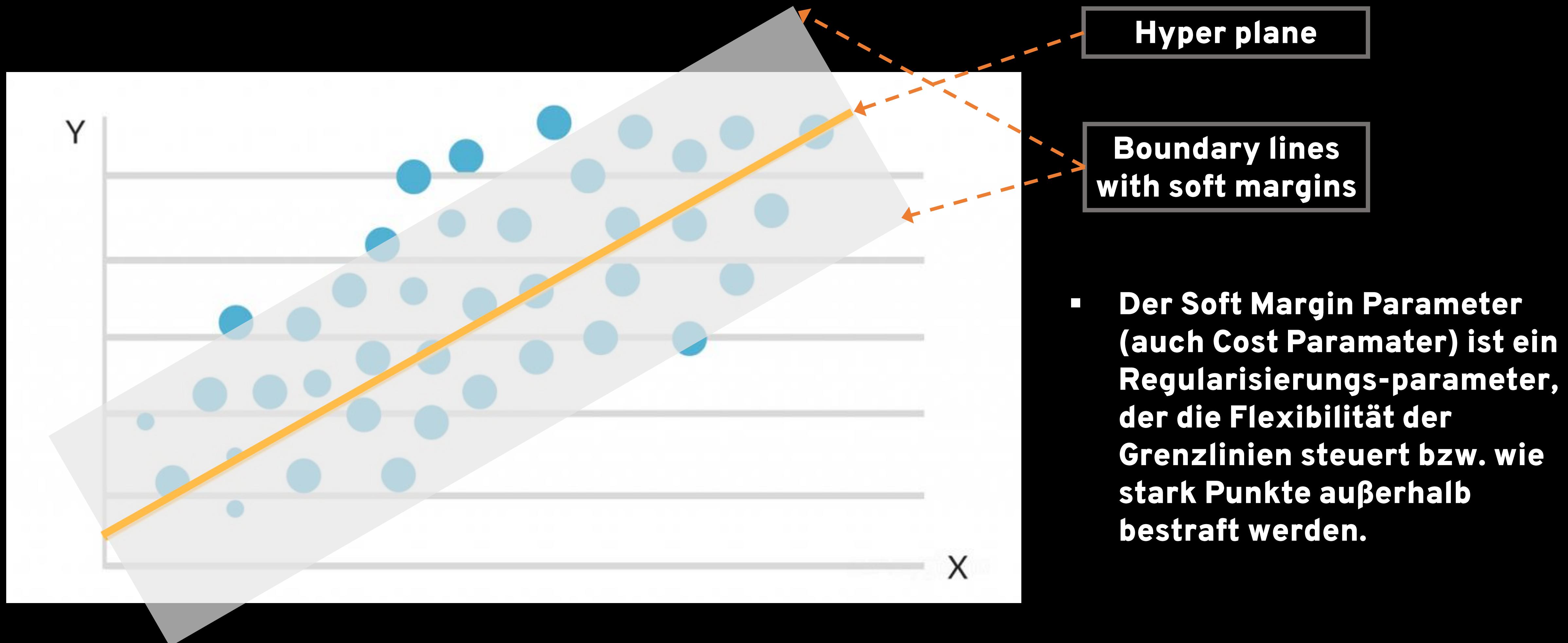
- Nutzt das VIM package, um Eure fehlenden Werte darzustellen
- Führt eine erste Imputation durch

# AUFGABEN

- Wählt ein (bzw. verschiedene) Verfahren, um die fehlenden Werte in Eurem Datensatz zu ersetzen.
- Schaut [dieses Video](#) (5 Minuten) zu Zeitreihenanalysen.
- Teilt Euch die Aufgaben im Team gut auf:  
Wer arbeitet an der Datenoptimierung, wer an der Modelloptimierung?

# SUPPORT VECTOR MACHINE (SVM)

## [SUPPORT VECTOR REGRESSION (SVR)]



# SUPPORT VECTOR MACHINES

## Lernressourcen

- Blog:  
<https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2>
- Video:  
<https://www.youtube.com/watch?v=efR1C6CvhmE>

# **OPTIMIERUNG EINER SVM IN R**

**R-Package „e1071“**

**svm()**

**Optimierung eines SVM Modells**

**tune()**

**Optimierung mehrerer SVM Modelle mit gleichzeitiger Optimierung des C- und Gamma-Parameters**

**predict()**

**Vorhersage auf Basis eines optimierten Modells**

# BEISPIELCODE IN R

```
1 ---  
2 title: "Support Vector Machine"  
3 output: html_notebook  
4 ---  
5  
6  
7 ## Imports   
22 ## Splitting Training and Test Data   
43 ## Data Preparation   
51 ## Training the SVM  
52  
53 ````{r}  
54 # Optimization of an SVM with standard hyper parameters  
55 # Typically NOT used; Instead, the function svm_tune() is used in order to also get a model with optimized hyper parameters  
56 model_svm <- svm(price ~ bathrooms, train_dataset)  
57 ````  
58  
59 ````{r}  
60 # Optimization of various SVM using systematically varied hyper parameters (typically called 'grid search' approach) and  
# cross validation  
61 # the resulting object includes the optimal model in the element named 'best.model'  
62 svm_tune <- tune(svm, price ~ bedrooms + bathrooms + sqft_living + zipcode, data=train_dataset,  
63 ranges = list(epsilon = seq(0.2,1,0.1), cost = 2^(2:3)))  
64 ````  
65  
66  
67 ## Checking the Prediction Quality 
```

# ZUSAMMENFASSUNG SVM

- Populärer, weil einfach zu optimierender Lernalgorithmus, der schnell gute Ergebnisse bringt.
- Geeignet zur Klassifikation und zur Regression.