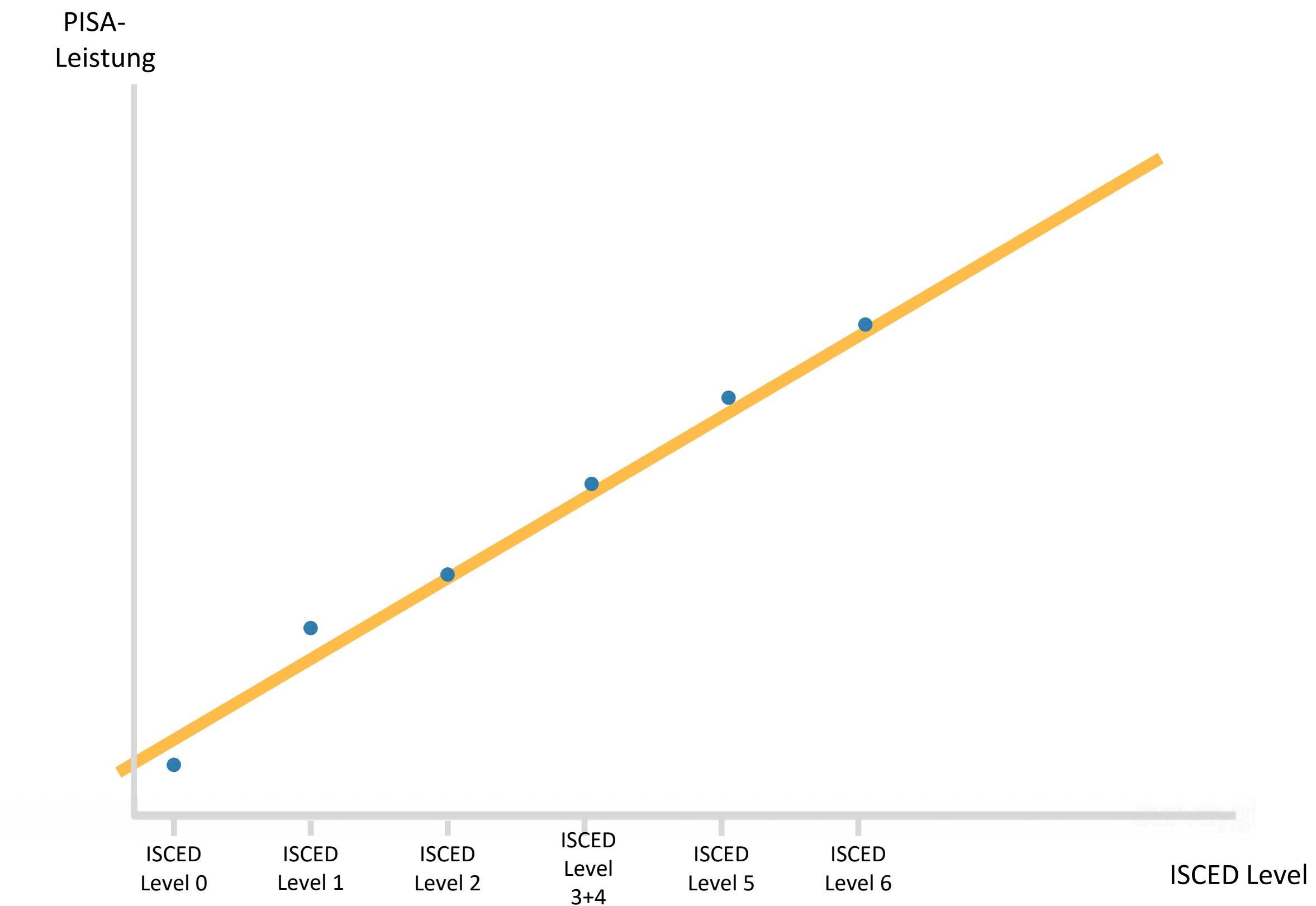
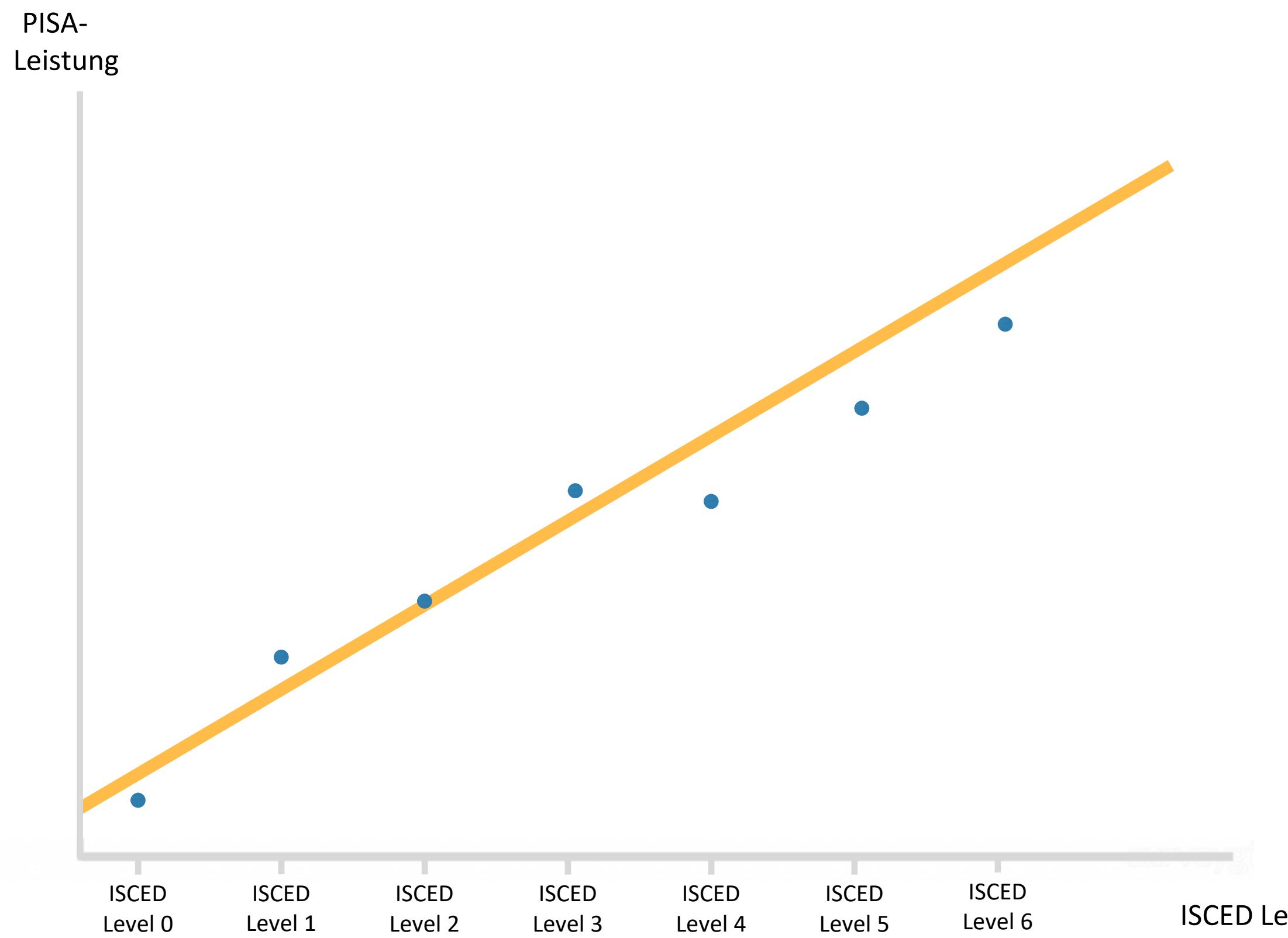


*Einführung in Data
Science und maschinelles
Lernen mit R*

Support Vektor Maschinen

- **Wiederholung**
- **Definition der Support Vektor Maschine (SVM)**
- **Kreuzvalidierung**
- **Implementierung von SVMs in R**
- **Modellgütekriterien**
- **Einstieg in neuronale Netze**
- **Integration von Python Code in RStudio**

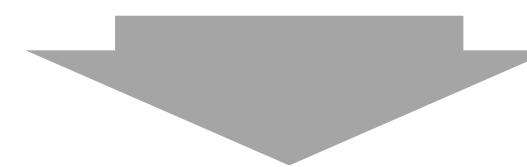
RELEVANZ VON KATEGORISIERUNGEN



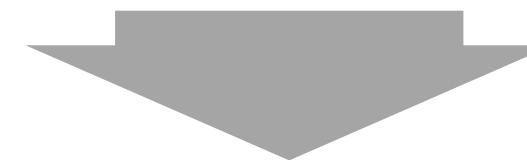
Wahl eines Prognosemodells



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



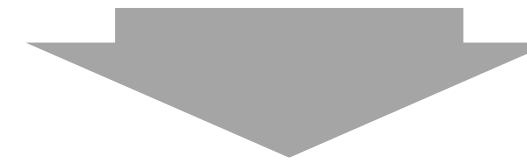
Optimierung der Modellparameter anhand des Trainingsdatensatzes



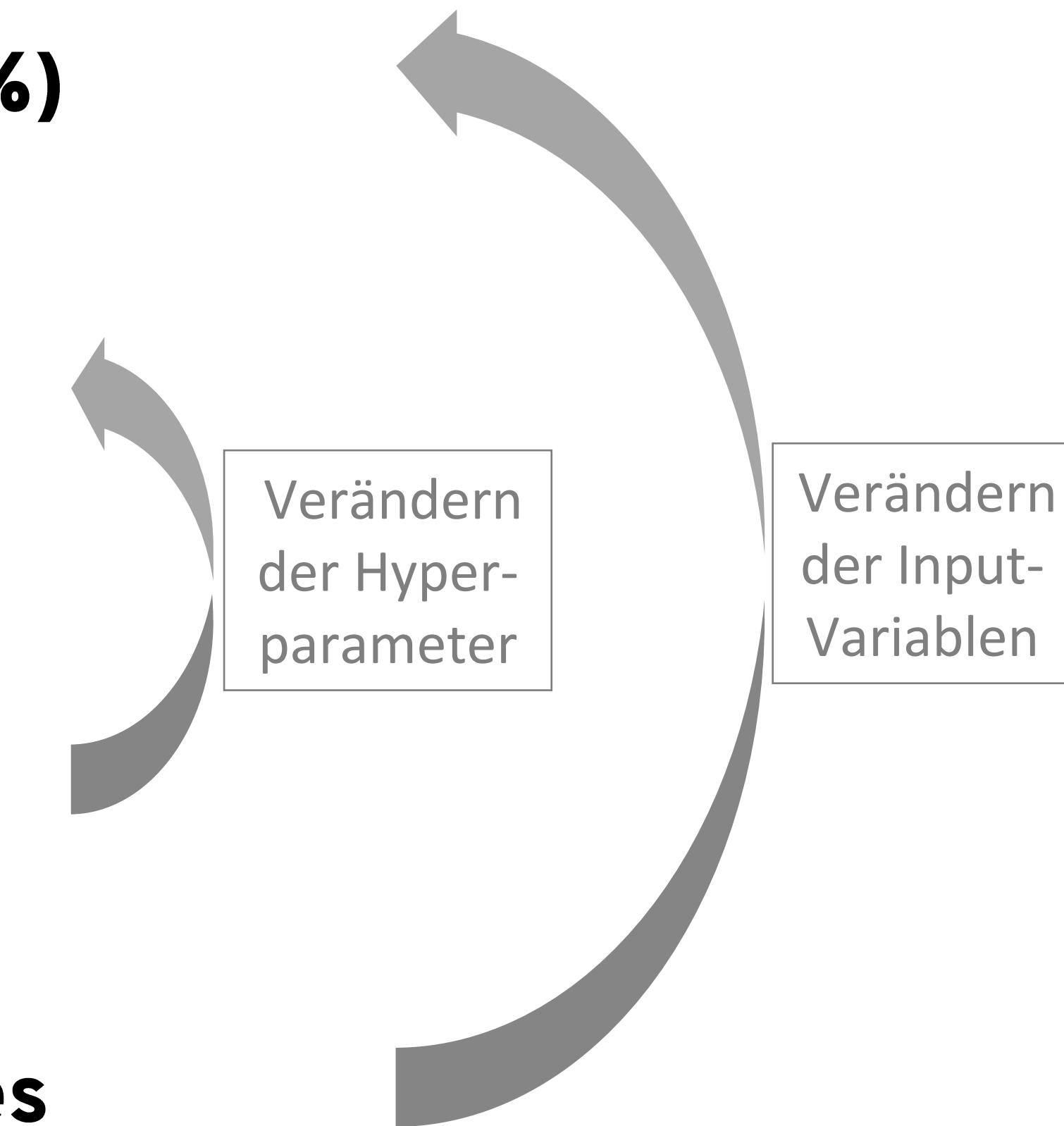
Optimierung der Hyperparameter anhand des Validierungsdatensatzes



Erweiterung/Verbesserung des Datensatzes



Überprüfung der Modellqualität anhand des Testdatensatzes



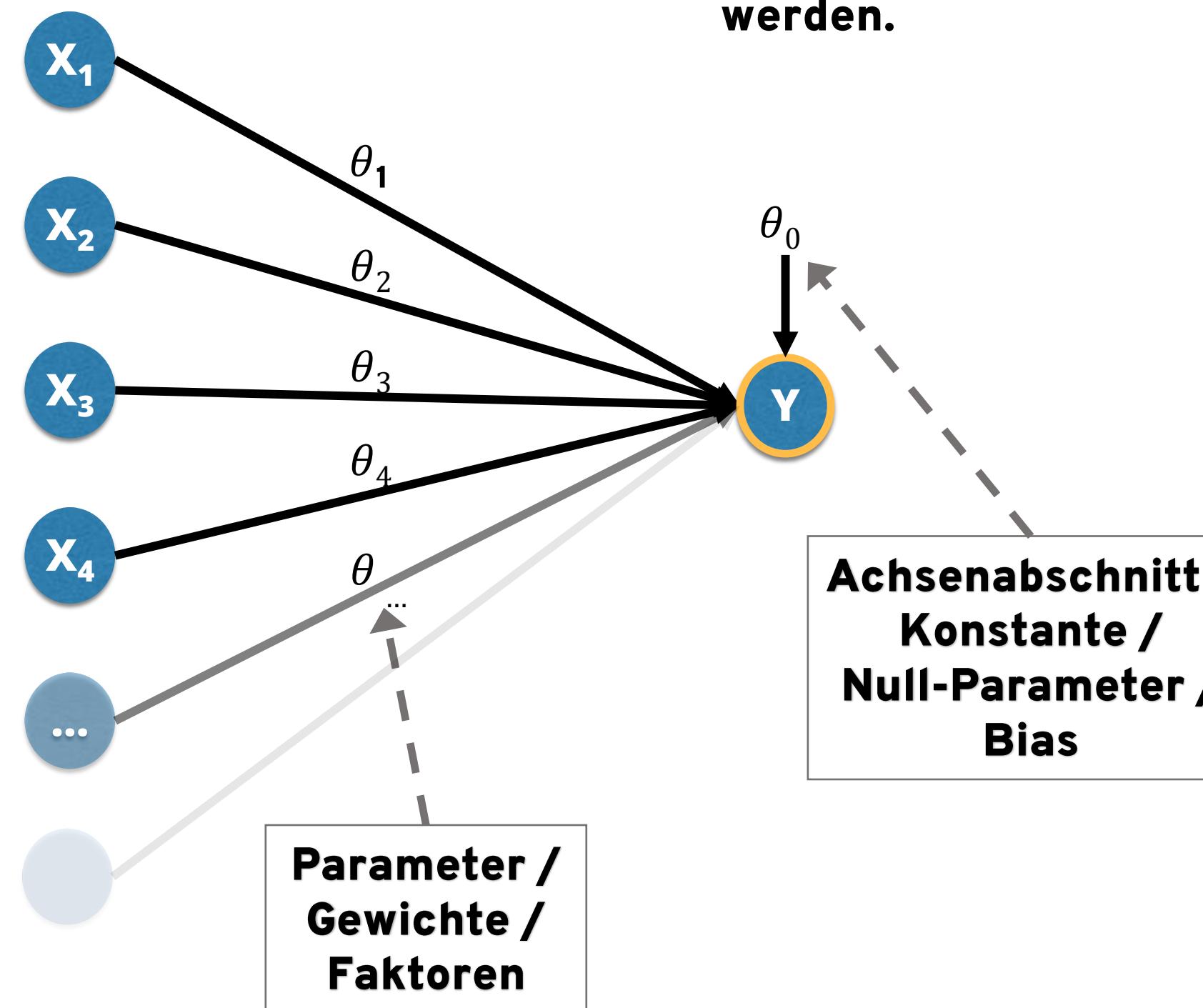
WICHTIGE KONZEPTE

- **Aktivierungsfunktion („Vorhersagefunktion“)**
- **Kostenfunktion**
 - **Regularisierung
(Bestrafung der Verwendung von Variablen/ großen Parametern)**
- **Optimierungsfunktion
(zur Minimierung der Kostenfunktion)**
 - **Lernrate (Eigenschaft der Optimierungsfunktion)**

ZUSAMMENFASSUNG LINEARE REGRESSION

Input Layer

Elemente sind die Input-Variablen; auch genannt: Input-Features oder Input-Dimensionen.



Output Layer

Nutzt eine „Aktivierungsfunktion“ (hier lineare Funktion) mit den Parametern θ der eingehenden Schicht zusammengefasst werden.

- Berechnung der Vorhersage (Prediction): Forward Propagation
- Berechnung der Abweichung der Vorhersage von den tatsächlichen Werten

BEISPIEL OVERFITTING

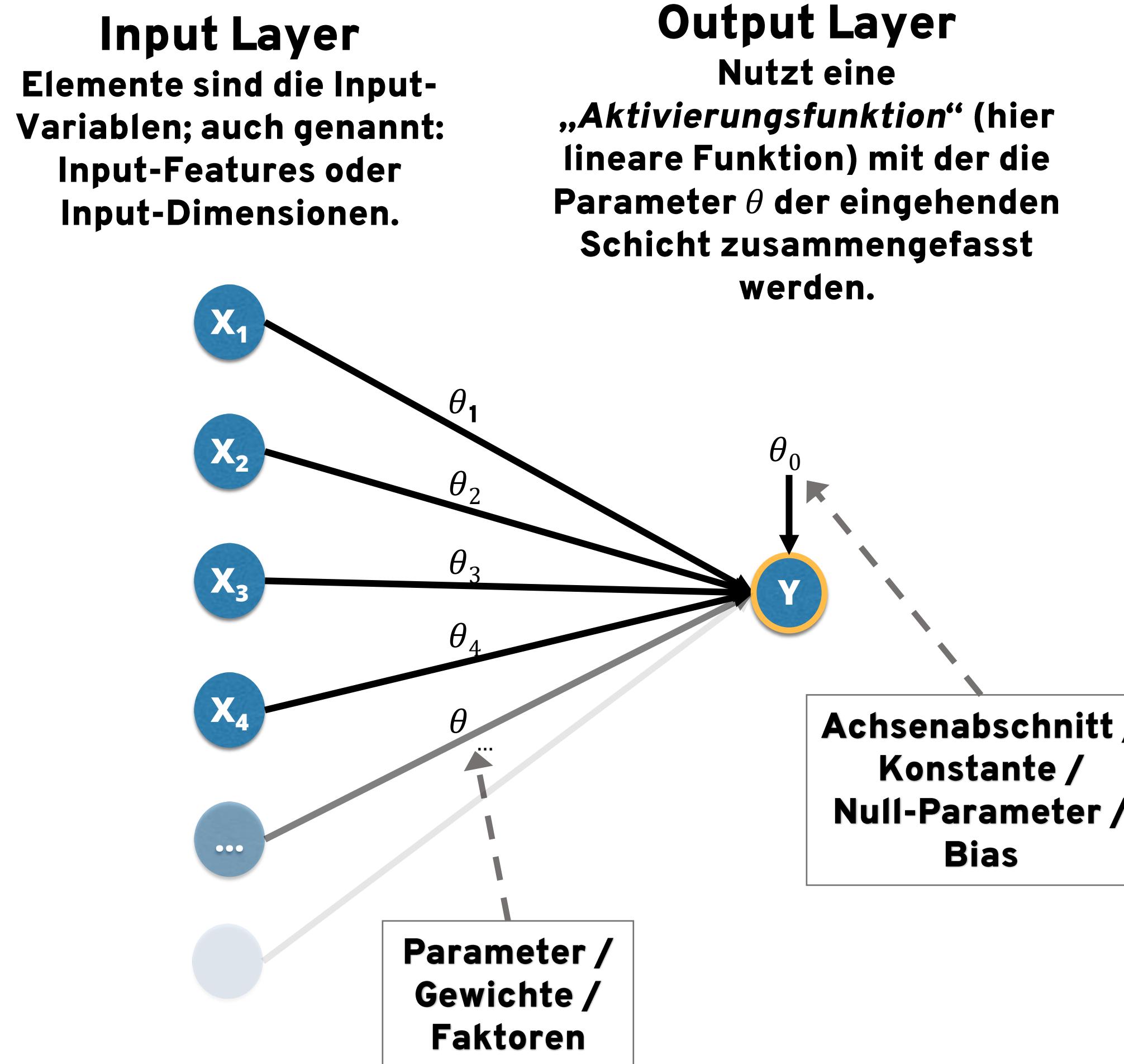
```
# Model Prediction Quality for the Training Data Using t  
rbind(mape(house_pricing_train$price, predict(mod1)),  
       mape(house_pricing_train$price, predict(mod2)),  
       mape(house_pricing_train$price, predict(mod3)),  
       mape(house_pricing_train$price, predict(mod4)),  
       mape(house_pricing_train$price, predict(mod5)),  
       mape(house_pricing_train$price, predict(mod6)),  
       mape(house_pricing_train$price, predict(mod7)))
```

```
[,1]  
[1,] 0.4328406  
[2,] 0.4146068  
[3,] 0.2446909  
[4,] 0.2422199  
[5,] 0.2423780  
[6,] 0.2190553  
[7,] 0.1781694
```

```
# Model Prediction Quality for the (Unknown) Test Data Us  
rbind(mape(house_pricing_test$price, predict(mod1, newdat  
       mape(house_pricing_test$price, predict(mod2, newdat  
       mape(house_pricing_test$price, predict(mod3, newdat  
       mape(house_pricing_test$price, predict(mod4, newdat  
       mape(house_pricing_test$price, predict(mod5, newdat  
       mape(house_pricing_test$price, predict(mod6, newdat  
       mape(house_pricing_test$price, predict(mod7, newdat
```

```
[,1]  
[1,] 0.4323069  
[2,] 0.4164473  
[3,] 0.2473555  
[4,] 0.2449223  
[5,] 0.2450962  
[6,] 0.2230855  
[7,] 0.2112316
```

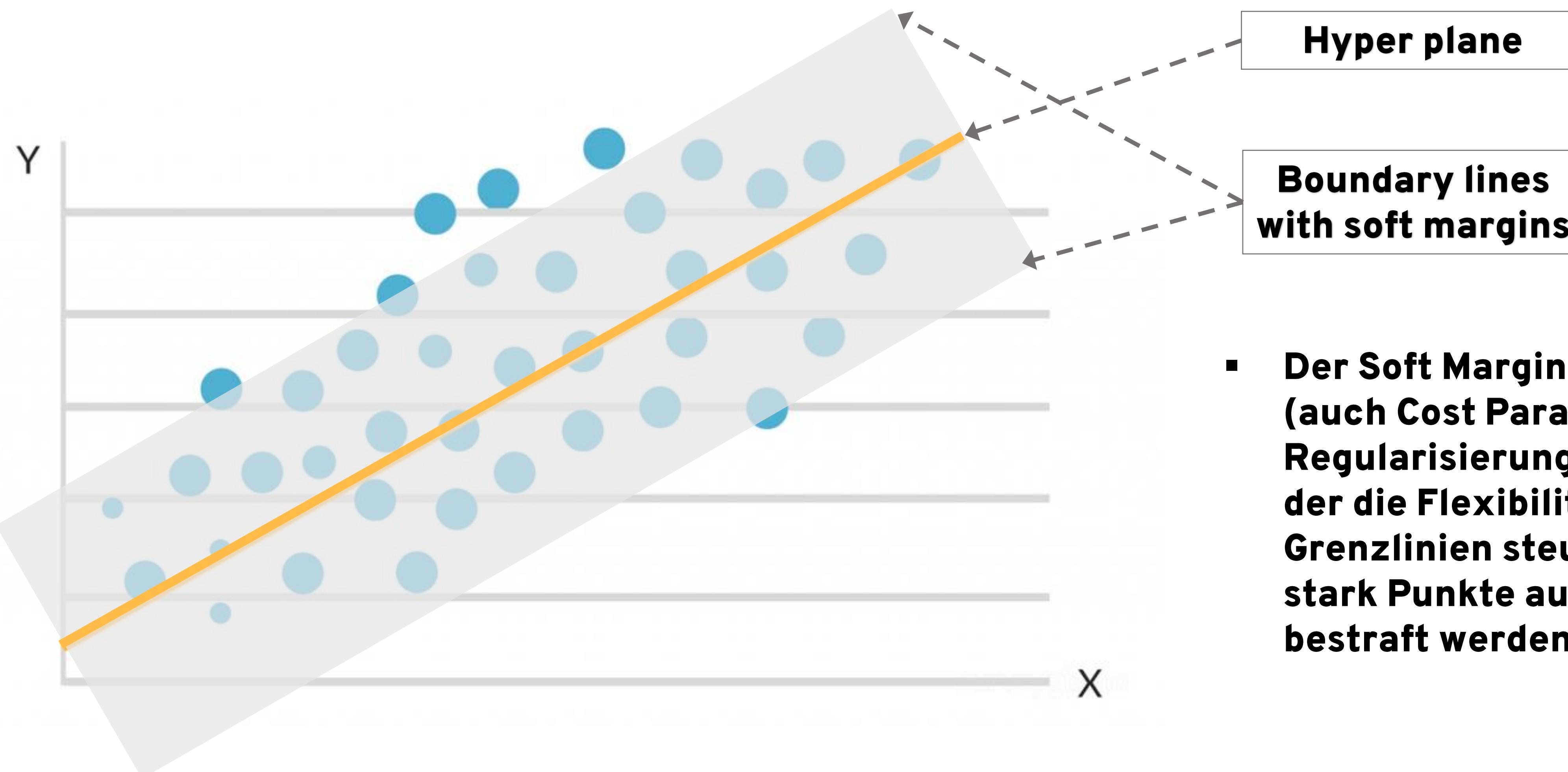
ZUSAMMENFASSUNG LINEARE REGRESSION



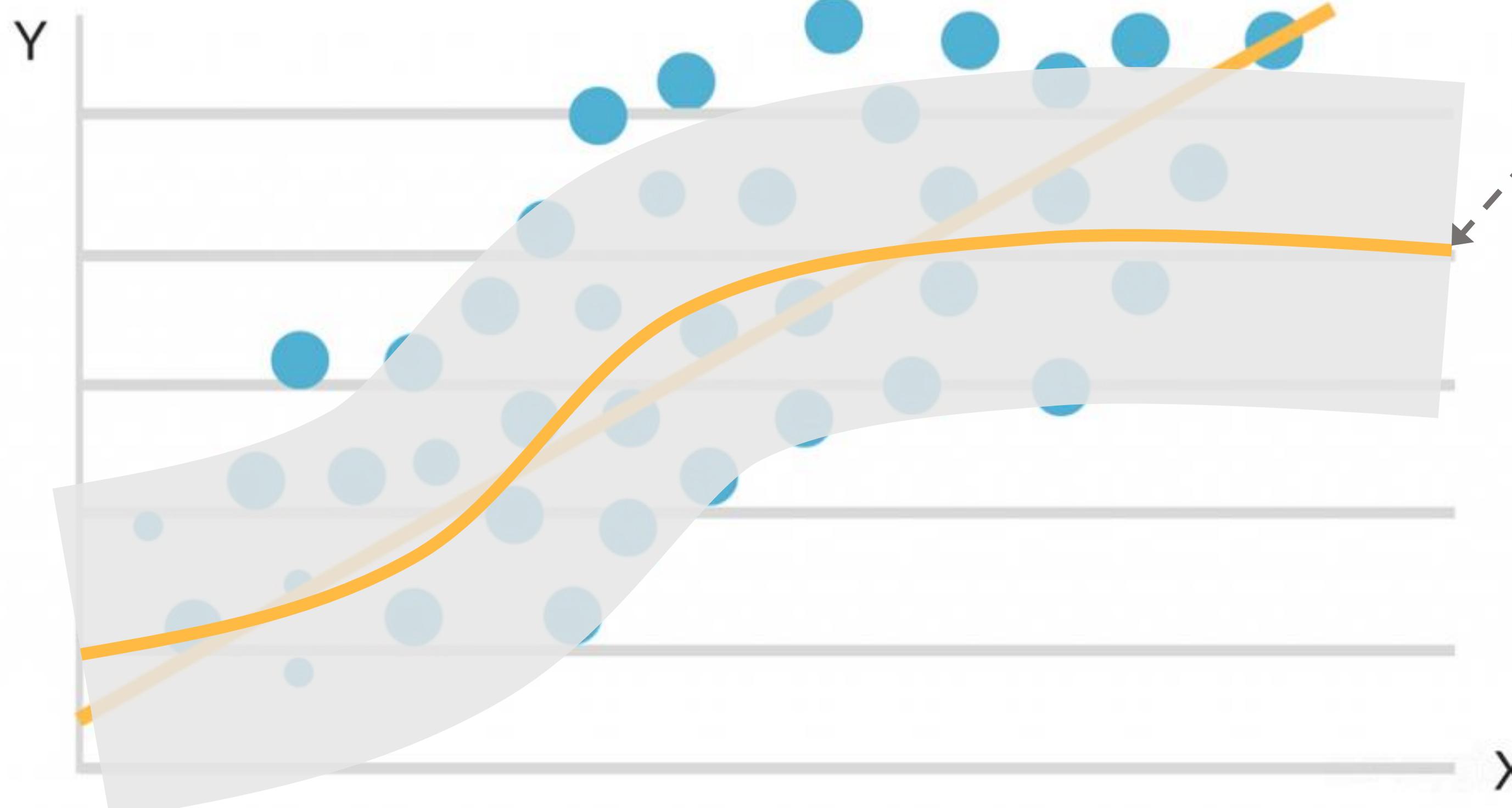
- Ziel ist, anhand des Trainingsdatensatzes die Parameter θ der Aktivierungsfunktion für eine bestmögliche Vorhersage des Testdatensatzes zu optimieren.
- Die Optimierung mit Regularisierung erlaubt, viele Variablen in das Modell eingehen zu lassen und über einen Regularisierungs- (oder Shrinkage-) Parameter den Umfang des Einsatzes der Variablen zu kontrollieren, um so Over-/Underfitting zu kontrollieren.

SUPPORT VECTOR MACHINE (SVM)

[SUPPORT VECTOR REGRESSION (SVR)]



KERNEL TRICK



Flexibilisierung der Hyperebene (bzw. der Stützvektoren) durch Abbildung in einen höherdimensionalen Raum

- Die Funktion zur Abbildung wird Kernel-Funktion genannt
- Die Flexibilität der Kernel-Funktion wird durch einen weiteren Regularisierungsparameter gesteuert.

Blog zur Einführung in SVM:

<https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2>

HYPERPARAMETER VON SVM

C-Parameter (Cost-Parameter / Soft-Margin-Parameter)

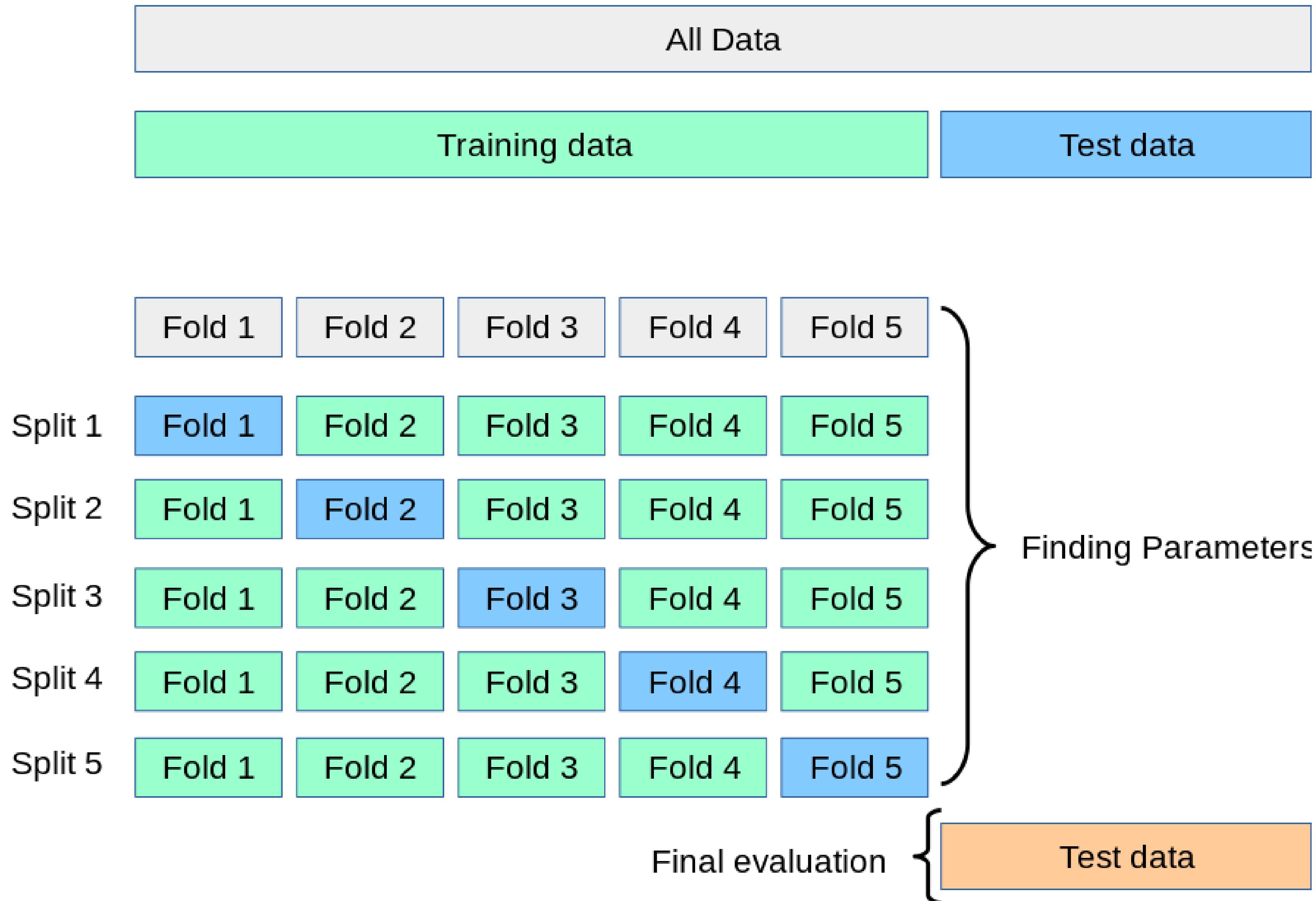
- Regularisierungsparameter der Kostenfunktion, der die Flexibilität der Grenzen kontrolliert
- Kleines C macht die Grenzen flexibel → hohe Varianz / niedriger Bias
- Großes C macht die Grenzen starr → niedrige Varianz / hoher Bias
- Entspricht $1/\lambda$

Kernel-Parameter

- Kontrolle der Flexibilität der Kernel-Funktion
- Großes Epsilon erlaubt hohe Flexibilität → hohe Varianz / niedriger Bias
- Kleines Epsilon erlaubt geringe Flexibilität → niedrige Varianz / hoher Bias

N-FACHE KREUZVALIDIERUNG

- **Zufällige Aufteilung des Datensatzes in n gleich große Teildatensätze.**
- **Optimierung von n Modellen, wobei in jeder Modelloptimierung eine anderer der n Teildatensätze als Validierungsdatensatz genutzt wird und der Rest jeweils als Trainingsdatensatz.**
- **Die Modellgüte wird anhand des Mittelwerts der Vorhersageabweichungen der n Modelle für die Testdatensätze beurteilt.**



Cross-validation: Evaluating estimator performance. (n.d.). Retrieved December 15, 2020, from Scikit-learn 0.23.2 documentation website: https://scikit-learn.org/stable/modules/cross_validation.html

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 auf Basis einer Kreuzvalidierung (Cross Validation)

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 
```

BREAKOUT

- **Optimiert eine SVM für Euren Datensatz**

ZUSAMMENFASSUNG SVM

- Sehr populärer, weil einfach zu optimierender Lernalgorithmus, der schnell gute Ergebnisse bringt.
- Geeignet zur Klassifikation und zur Regression.
- Im Fall der Klassifikation ist anstelle des Epsilon-Parameters ein vergleichbarer Gamma-Parameter zu definieren.

MODELLGÜTEKRITERIEN

errors: **forecast - actual**

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))**

$r^2 = 1 - rse$

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

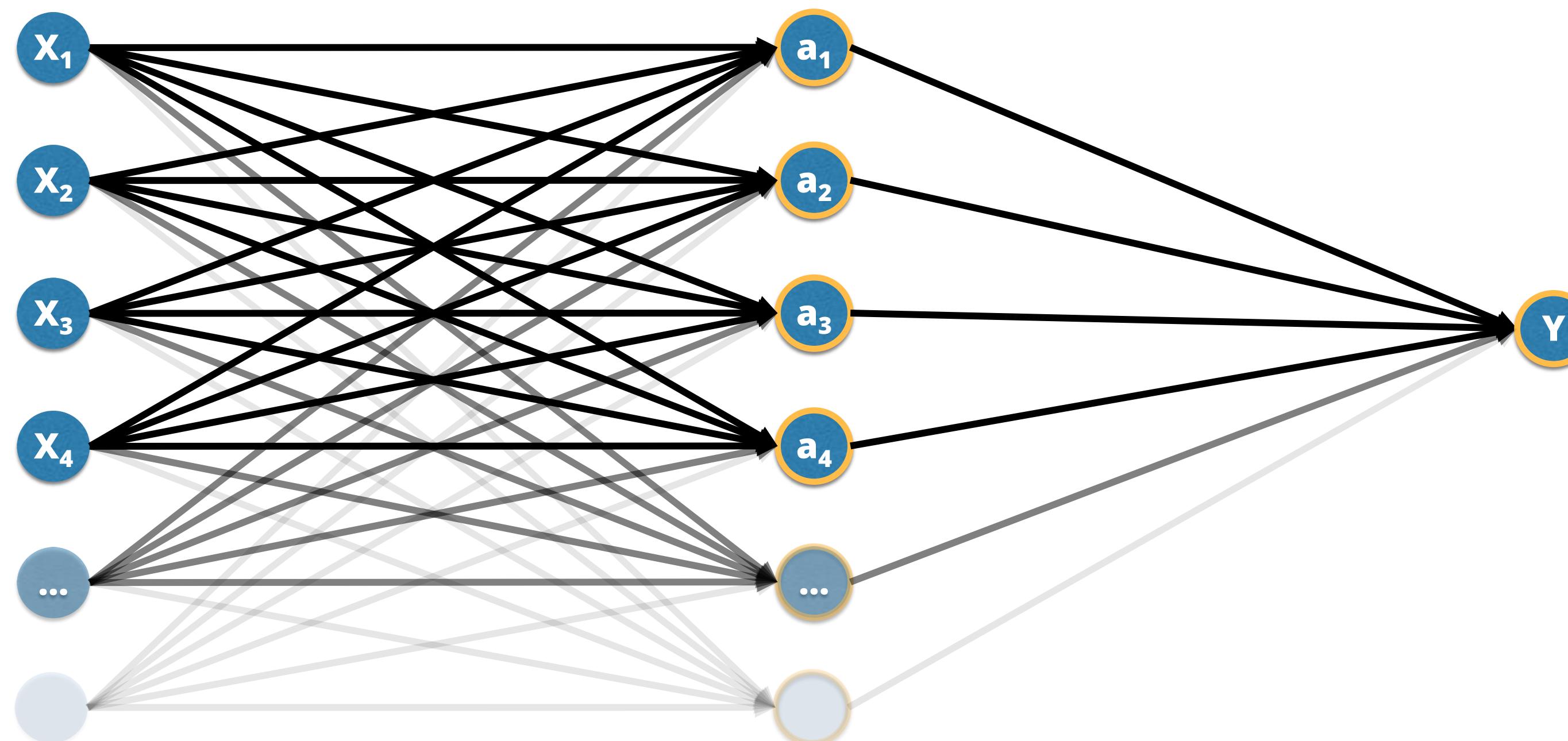
<https://www.coursera.org/lecture/machine-learning-with-python/evaluation-metrics-in-regression-models-5SxtZ>

NEURONALE NETZE

Input Layer
Besteht aus
Input-Variablen/
Features/
Dimensionen

Hidden Layer
Fasst mit Hilfe einer
Aktivierungsfunktion und der
geschätzten Gewichte die Werte
der eingehenden Schicht
zusammen in jeweils einem
Neuron zusammen.

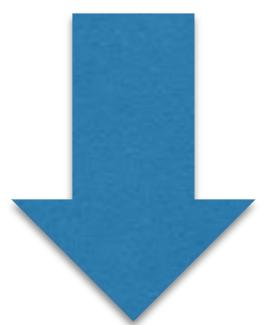
Output Layer
Fasst ebenfalls mit Hilfe einer
Aktivierungsfunktion und
geschätzter Gewichte die Werte
der eingehenden Schicht
zusammen.



R VS. PYTHON

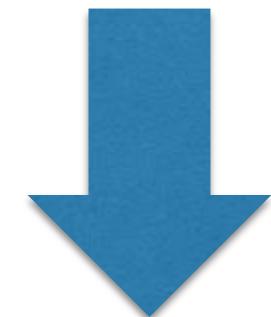
***Statistische Verfahren
außerhalb des ML***

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



Statistische Verfahren zum 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

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
```

AUFGABEN

- **Modellvariablen weiter optimieren!**
- **SVM für Euren Datensatz optimieren und eine Vorhersage für den 06.06.2019 erstellen.**
- **Schaut Euch sehr genau [dieses Video](#) zur Einführung in Neuronale Netze an (12 Minuten)**
- **Installation von Python (bzw. Miniconda) und den zugehörigen Paketen für R**