Parameters vs hyperparameters

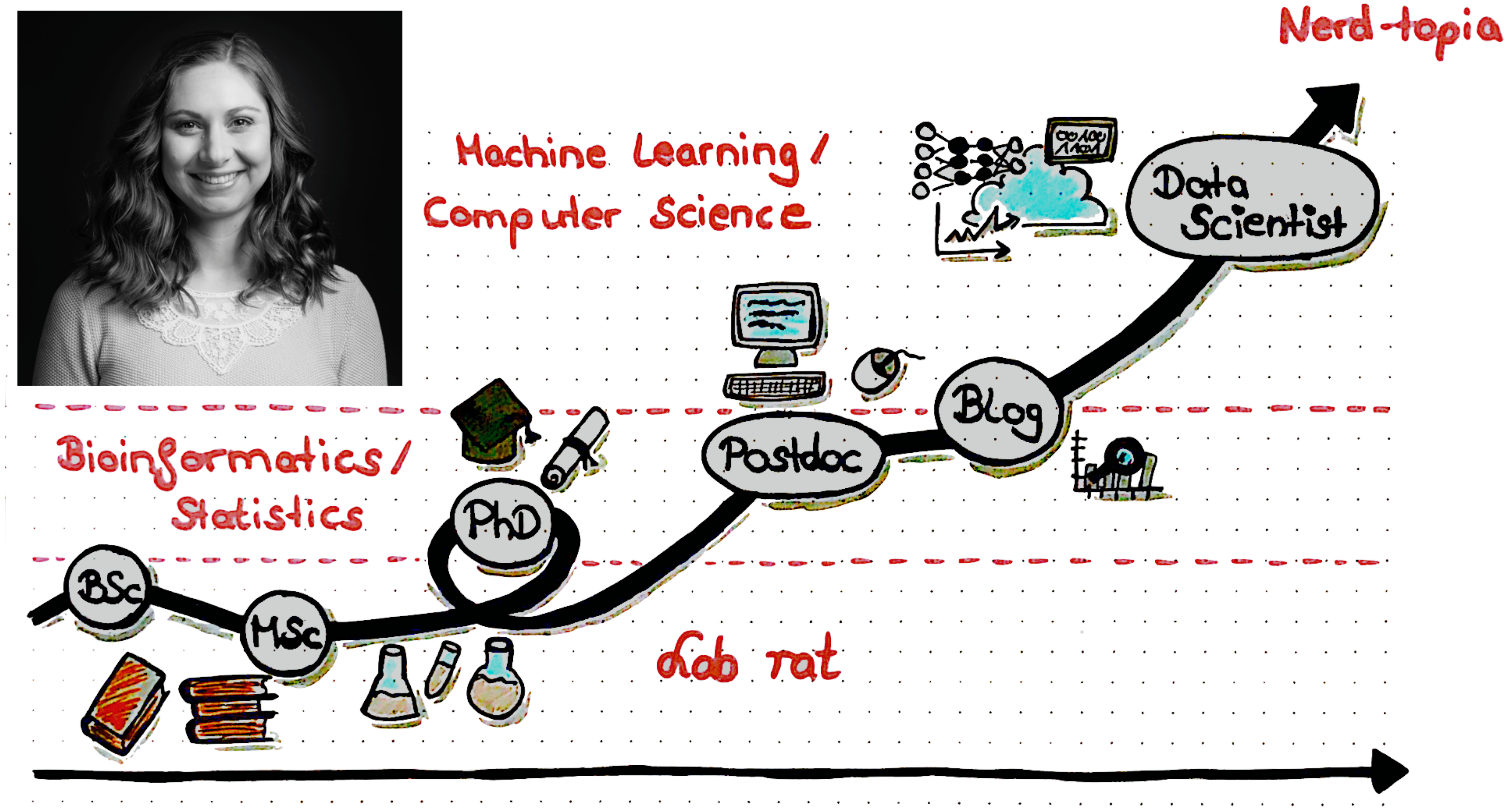
HYPERPARAMETER TUNING IN R

Dr. Shirin Elsinghorst
Senior Data Scientist

datacamp

About me

www.shirin-glander.de



"Hyper"parameters vs model parameters

- Let's look at an example dataset:

```
head(breast_cancer_data)
```

```
# A tibble: 6 x 11
  diagnosis concavity_mean symmetry_mean fractal_dimension perimeter_se smoothness_se
  <chr>              <dbl>         <dbl>             <dbl>        <dbl>         <dbl>
1 M                 0.300          0.242            0.0787         8.59       0.00640
2 M                 0.0869         0.181            0.0567         3.40       0.00522
3 M                 0.197          0.207            0.0600         4.58       0.00615
4 M                 0.241          0.260            0.0974         3.44       0.00911
```

- And build a simple **linear model**.

Let's start simple: Model parameters in a linear model

```
# Create linear model
linear_model <- lm(perimeter_worst ~ fractal_dimension_mean, data = breast_cancer_data)
```

```
# Get coefficients
linear_model$coefficients
```

```
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)              167.60      25.91   6.469  3.9e-09 ***
fractal_dimension_mean  -926.39     392.86  -2.358   0.0204 *
```

Let's start simple: Model parameters in a linear model

- Model **parameters** are being fit (i.e. found) during training.
- They are the **result** of model fitting or training.
- In a linear model, we want to find the **coefficients.**

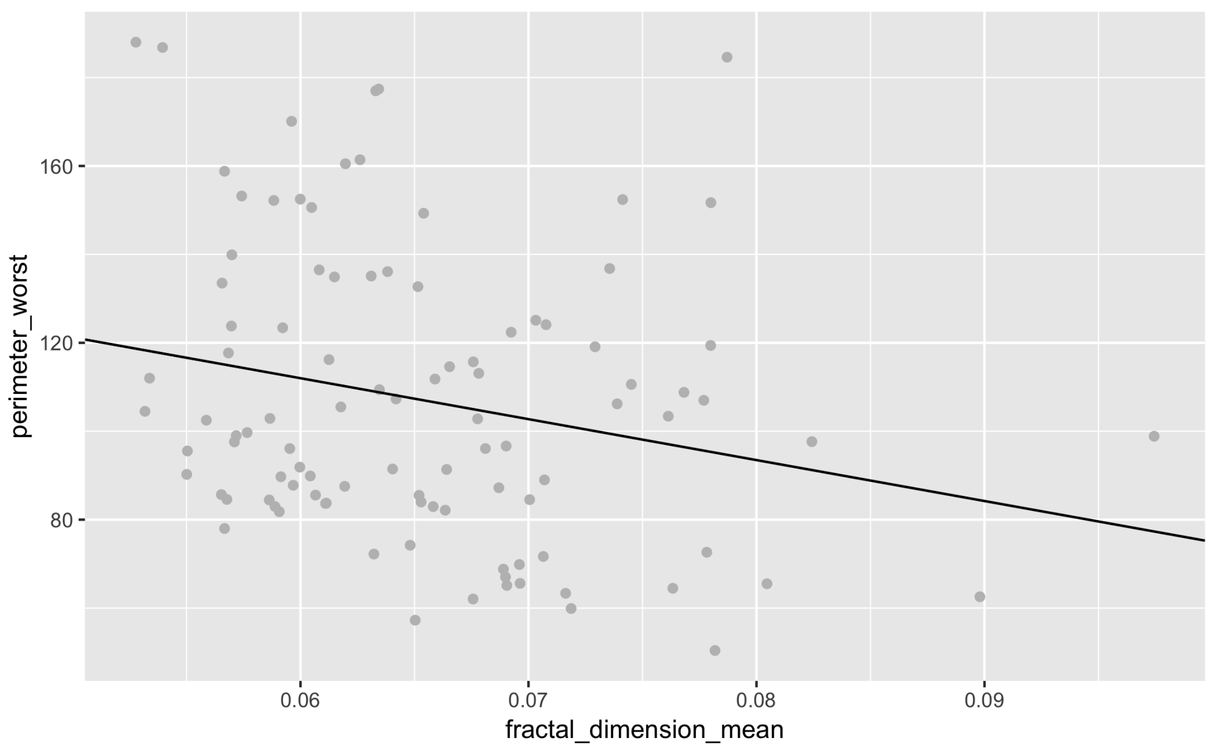
```
linear_model$coefficients
```

```
(Intercept) fractal_dimension_mean
   167.5972              -926.3866
```

- We can think of them as the **slope** and the **y-intercept** of our model.

Coefficients in a linear model

```
ggp <- ggplot(data = breast_cancer_data,
              aes(x = fractal_dimension_mean, y = perimeter_worst)) +
        geom_point(color = "grey")
ggp + geom_abline(slope = linear_model$coefficients[2],
                  intercept = linear_model$coefficients[1])
```



Model parameters vs hyperparameters in a linear model

- *Remember*: model **parameters** are being fit (i.e. found) during training; they are the **result** of model fitting or training.
- **Hyperparameters** are being set before training.
- They specify **HOW** the training is supposed to happen.

```
args(lm)
help(lm)
?lm

linear_model <- lm(perimeter_worst ~ fractal_dimension_mean,
                   data = breast_cancer_data,
                   method = "qr")
```

Parameters vs hyperparameters in machine learning

In our **linear model**:

- Coefficients were **found** during fitting.
- `method` was an option to set **before** fitting.

In **machine learning** we might have:

- Weights and biases of neural nets that are optimized during training => model **parameters.**
- Options like learning rate, weight decay and number of trees in a Random Forest model that can be tweaked => **hyperparameters.**

Why tune hyperparameters?

- Fantasy football players ~ **Hyperparameters**
- Football players' positions ~ Hyperparameter **values**
- Finding the best combination of players and positions ~ Finding the best **combination** of hyperparameters

Let's practice!

HYPERPARAMETER TUNING IN R

Recap of machine learning basics

HYPERPARAMETER TUNING IN R

Dr. Shirin Elsinghorst
Senior Data Scientist

datacamp

Machine learning with caret - splitting data

```
# Load caret and set seed
library(caret)
set.seed(42)

# Create partition index
index <- createDataPartition(breast_cancer_data$diagnosis, p = .70,
                             list = FALSE)
# Subset `breast_cancer_data` with index
bc_train_data <- breast_cancer_data[index, ]
bc_test_data  <- breast_cancer_data[-index, ]
```

- Training set with enough **power.**
- **Representative** test set.

Train a machine learning model with caret

- Set up **cross-validation:**

```
library(caret)
library(tictoc)
fitControl <- trainControl(method = "repeatedcv", number = 3, repeats = 5)
```

- **Train** a Random Forest model:

```
tic()
set.seed(42)
rf_model <- train(diagnosis ~ ., data = bc_train_data, method = "rf", trControl = fitControl,
                  verbose = FALSE)
toc()
```

```
1.431 sec elapsed
```

Automatic hyperparameter tuning in caret

```
Random Forest

...

Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.9006783  0.8015924
   6    0.9126645  0.8253289
  10    0.8999389  0.7999386

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 6.
```

Let's start modeling!

HYPERPARAMETER TUNING IN R

Hyperparameter tuning with caret

HYPERPARAMETER TUNING IN R

Dr. Shirin Elsinghorst
Senior Data Scientist

Automatic hyperparameter tuning in caret

```
Random Forest

...

Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.9006783  0.8015924
   6    0.9126645  0.8253289
  10    0.8999389  0.7999386

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 6.
```

Hyperparameters are specific to model algorithms

- `modelLookup(model)`
- https://topepo.github.io/caret/available-models.html

6 Available Models

The models below are available in `train`. The code behind these protocols can be obtained using the function `getModelInfo` or by going to the github repository.

Show 237 entries

Search:

Model	*method* Value	Type	Libraries	Tuning Parameters
AdaBoost Classification Trees	adaboost	Classification	fastAdaboost	nIter, method
AdaBoost.M1	AdaBoost.M1	Classification	adabag, plyr	mfinal, maxdepth, coeflearn
Adaptive Mixture Discriminant Analysis	amdai	Classification	adaptDA	model
Adaptive-Network-Based				

Hyperparameters in Support Vector Machines (SVM)

```
fitControl <- trainControl(method = "repeatedcv", number = 3, repeats = 5)

tic()
svm_model <- train(diagnosis ~ .,
                   data = bc_train_data,
                   method = "svmPoly",
                   trControl = fitControl,
                   verbose= FALSE)
toc()
```

```
3.836 sec elapsed
```

Hyperparameters in Support Vector Machines (SVM)

```
svm_model
```

```
Support Vector Machines with Polynomial Kernel
...

Resampling results across tuning parameters:

  degree  scale  C     Accuracy   Kappa
  1       0.100  1.00  0.9104803  0.8211459

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were degree = 1, scale = 0.1 and C = 1.
```

Defining hyperparameters for automatic tuning

- **tuneLength**

```
tic()
set.seed(42)
svm_model_2 <- train(diagnosis ~ .,
                     data = bc_train_data,
                     method = "svmPoly",
                     trControl = fitControl,
                     verbose = FALSE,
                     tuneLength = 5)
toc()
```

```
7.458 sec elapsed


Accuracy was used to select the optimal model using the largest value.
The final values used for the model were degree = 1, scale = 1 and C = 1.
```

Manual hyperparameter tuning in caret

- **tuneGrid + expand.grid**

```
hyperparams <- expand.grid(degree = 4, scale = 1, C = 1)
tic()
set.seed(42)
svm_model_3 <- train(diagnosis ~ .,
                     data = bc_train_data,
                     method = "svmPoly",
                     trControl = fitControl,
                     tuneGrid = hyperparams,
                     verbose = FALSE)
toc()
```

```
0.691 sec elapsed
```

Manual hyperparameter tuning in caret

```
svm_model_3
```

```
Support Vector Machines with Polynomial Kernel


...


  Accuracy   Kappa
  0.7772947  0.554812

Tuning parameter 'degree' was held constant at a value of 4
Tuning parameter 'scale' was held constant at a value of 1
Tuning parameter 'C' was
 held constant at a value of 1
```

It's your turn!

HYPERPARAMETER TUNING IN R