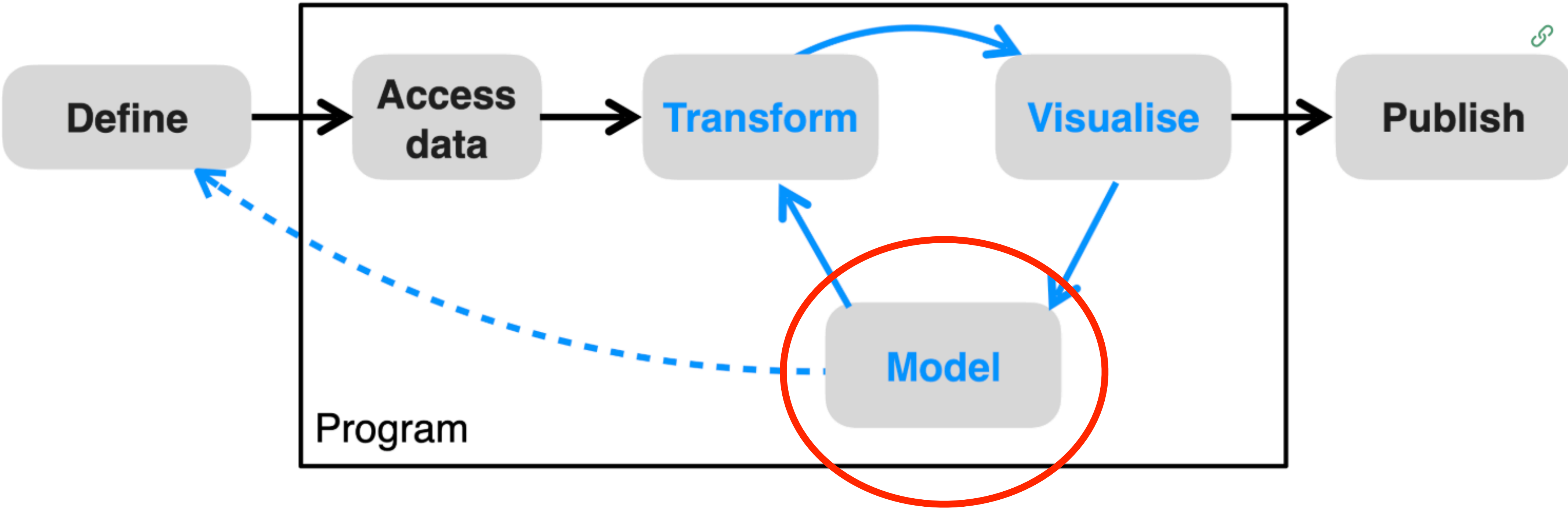
Applied Geodata Science I

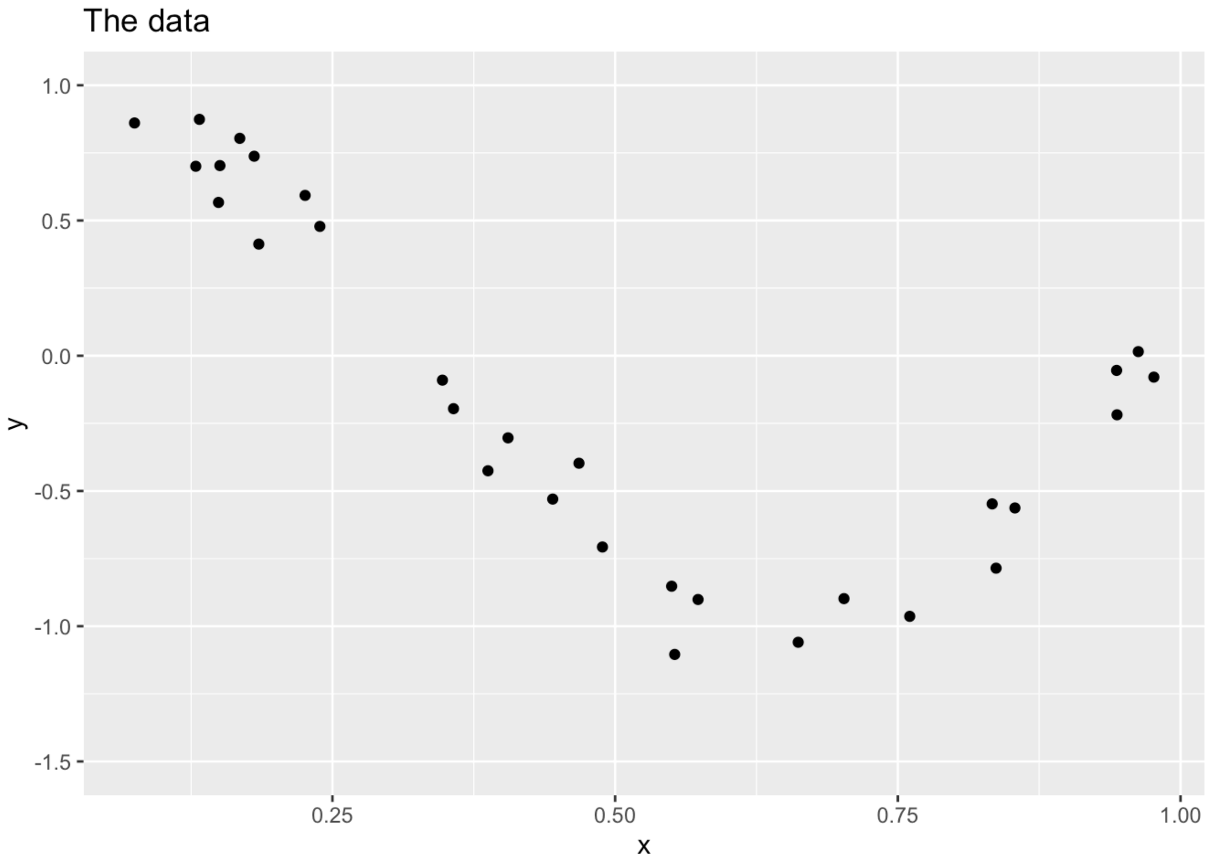
Session 10

Prof. Dr. Benjamin Stocker
28.04.2025

The data science workflow



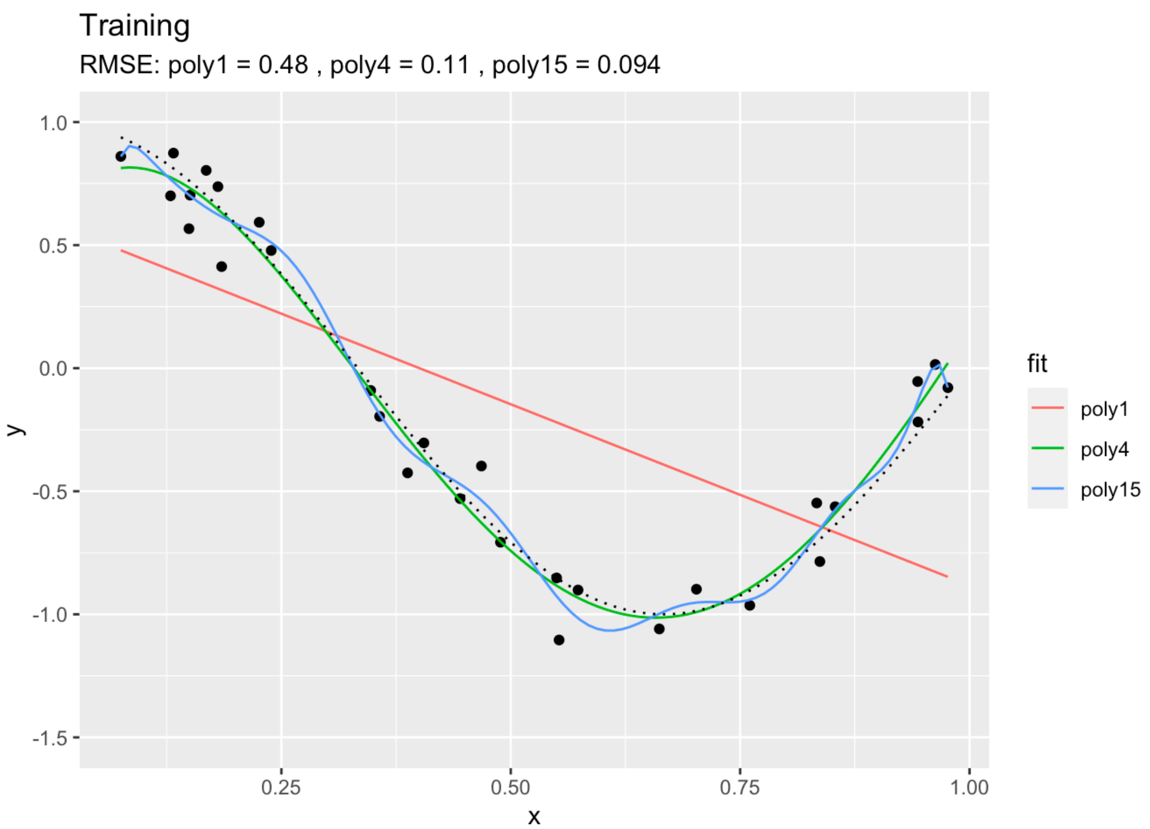
Example data



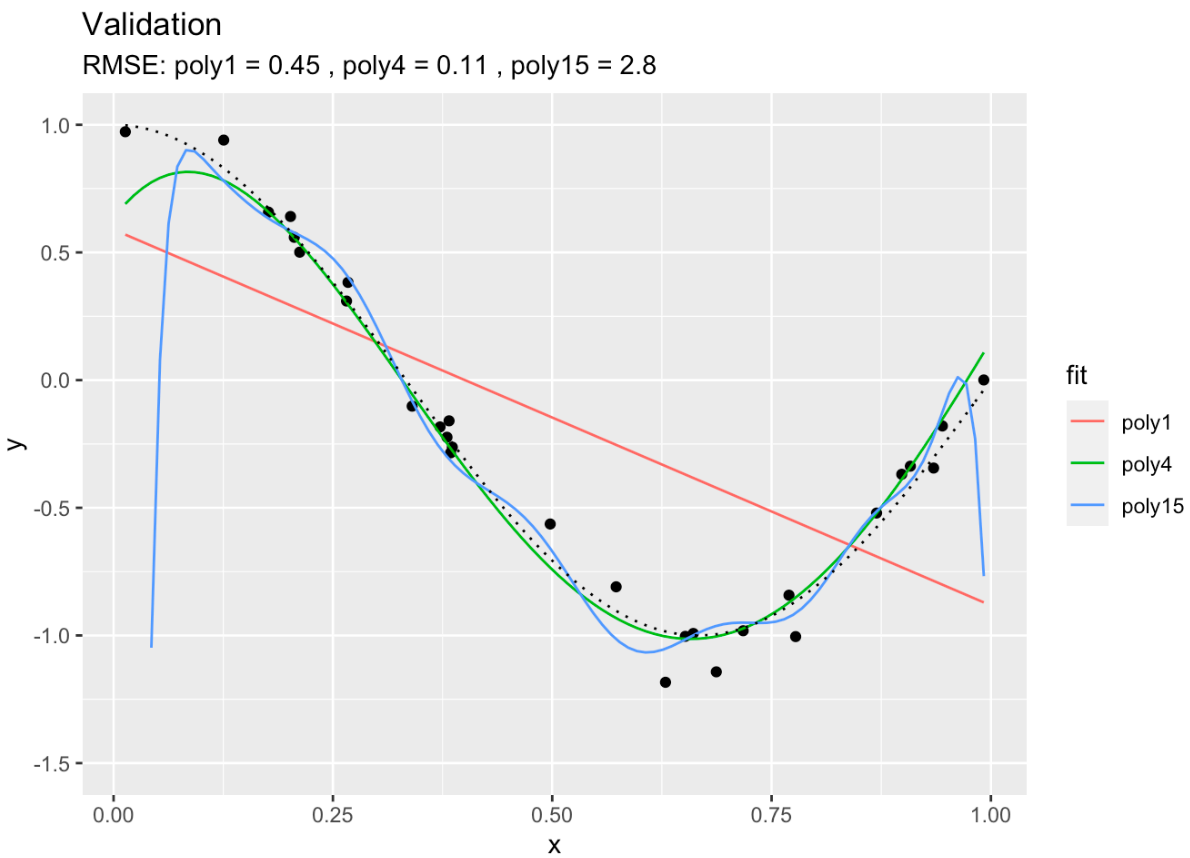
Model complexity

$$y = \sum_{n=0}^{N} a_n x^n$$

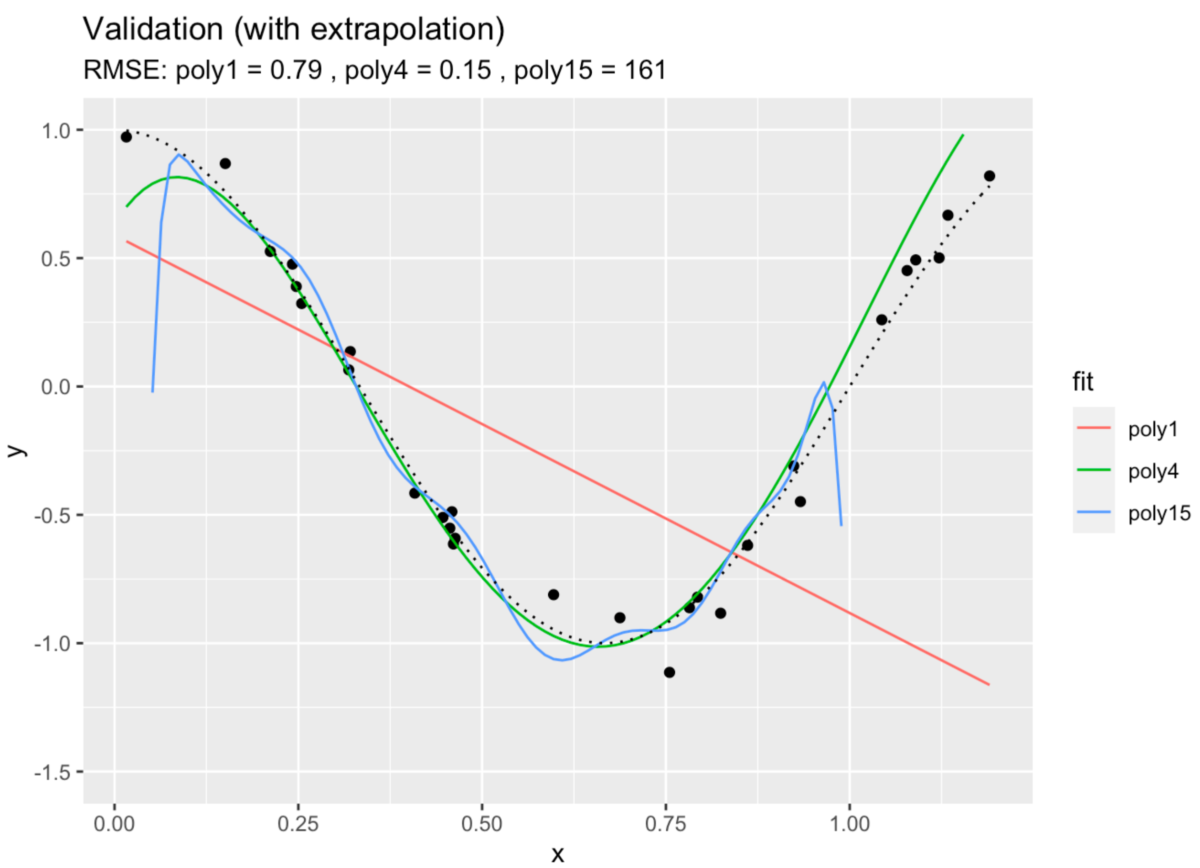
Training


Training
RMSE: poly1 = 0.48 , poly4 = 0.11 , poly15 = 0.094
1.0
0.5
0.0
-0.5
-1.0
-1.5
y
0.25
0.50
0.75
1.00
x
fit
poly1
poly4
poly15

Validation


Validation
RMSE: poly1 = 0.45 , poly4 = 0.11 , poly15 = 2.8
1.0
0.5
0.0
-0.5
-1.0
-1.5
y
0.00
0.25
0.50
0.75
1.00
x
fit
poly1
poly4
poly15

Validation (with extrapolation)



Supervised machine learning

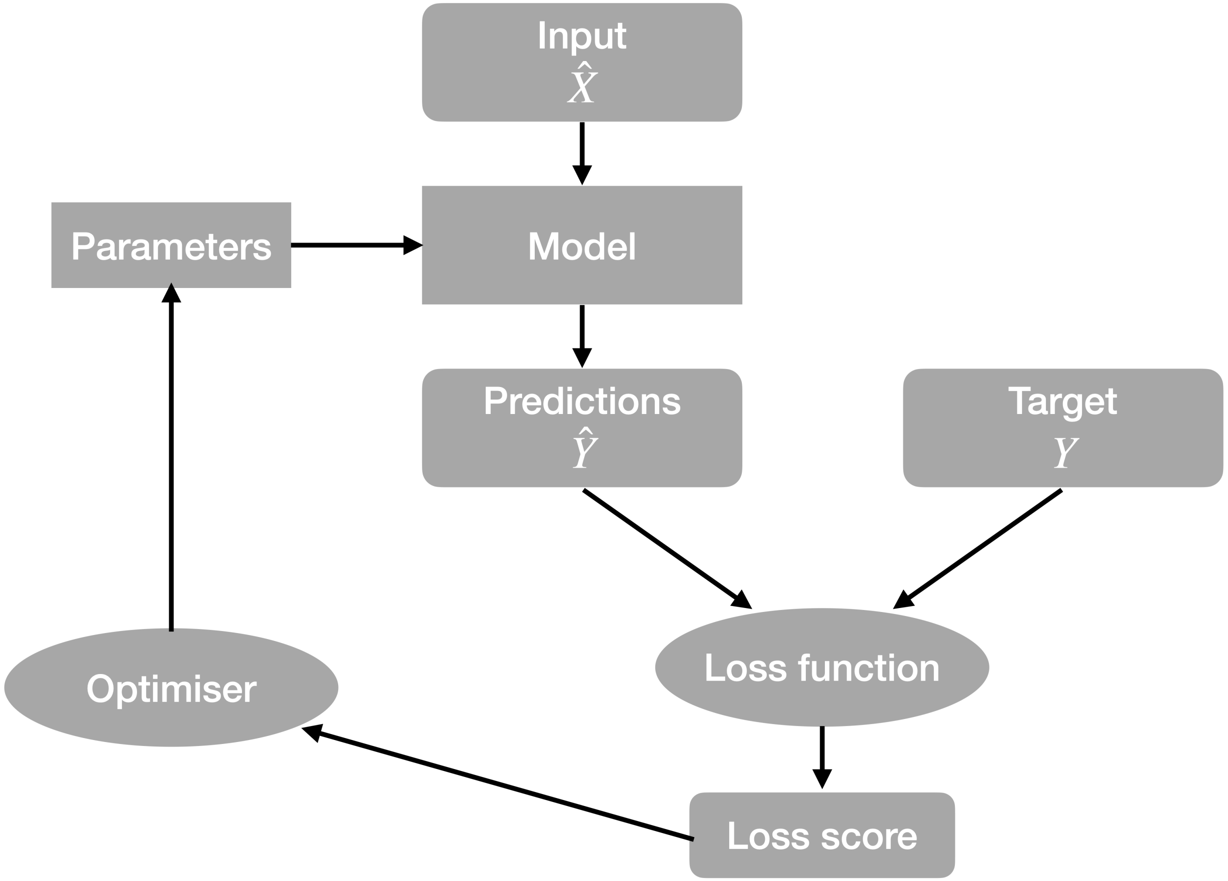
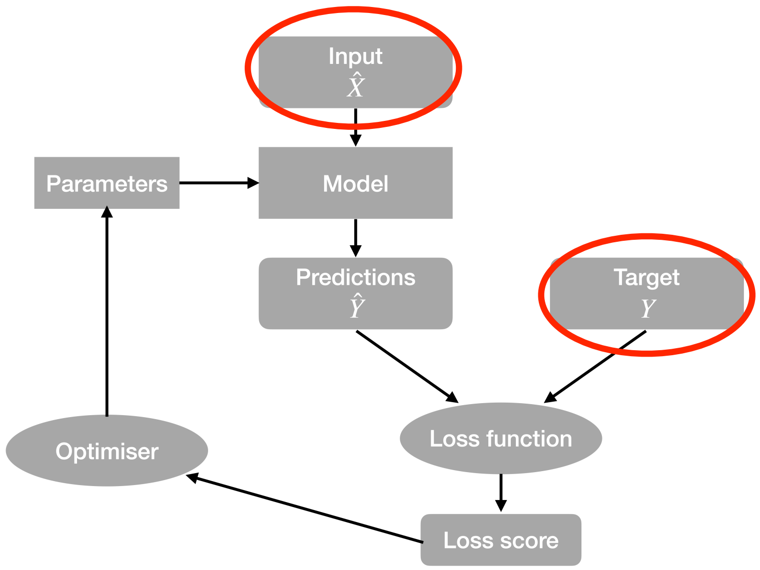
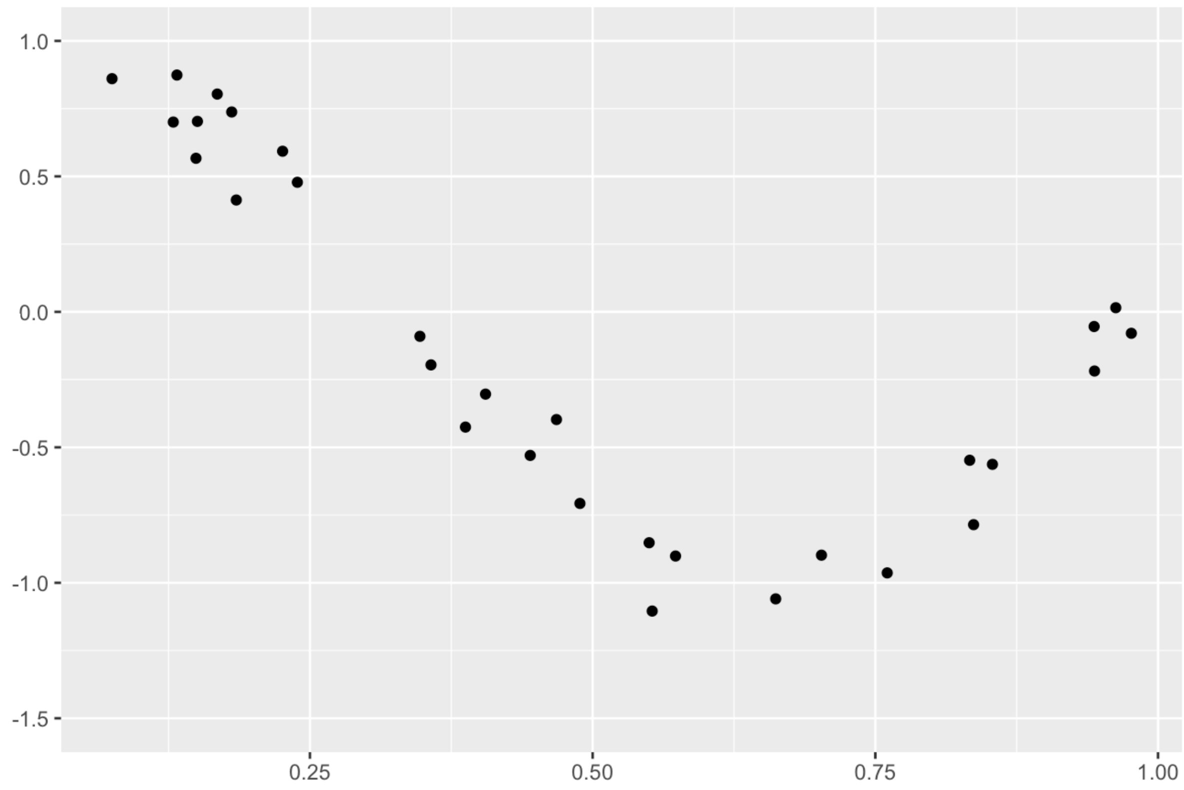
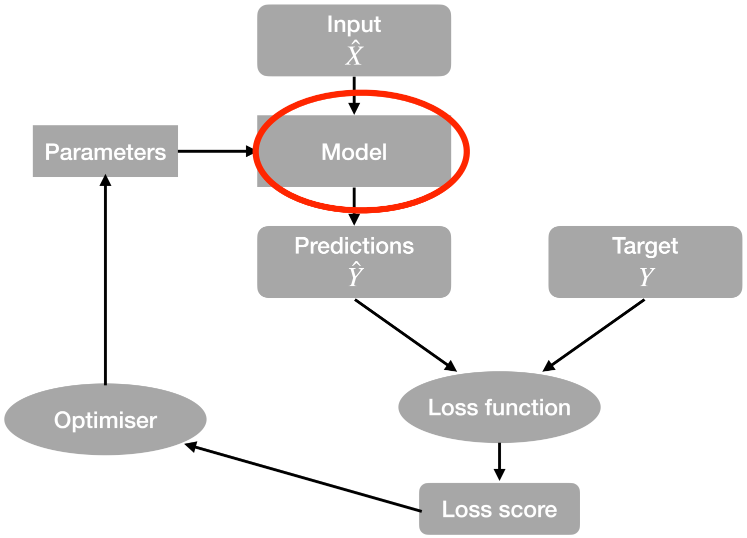


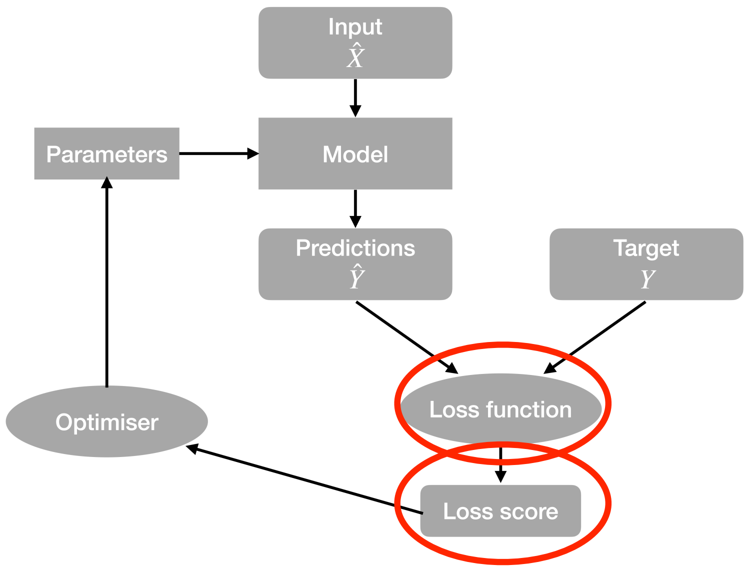
Figure adapted from Chollet & Allaire (2018)


Input
$\hat{X}$
Parameters
Model
Predictions
$\hat{Y}$
Target
Y
Optimiser
Loss function
Loss score

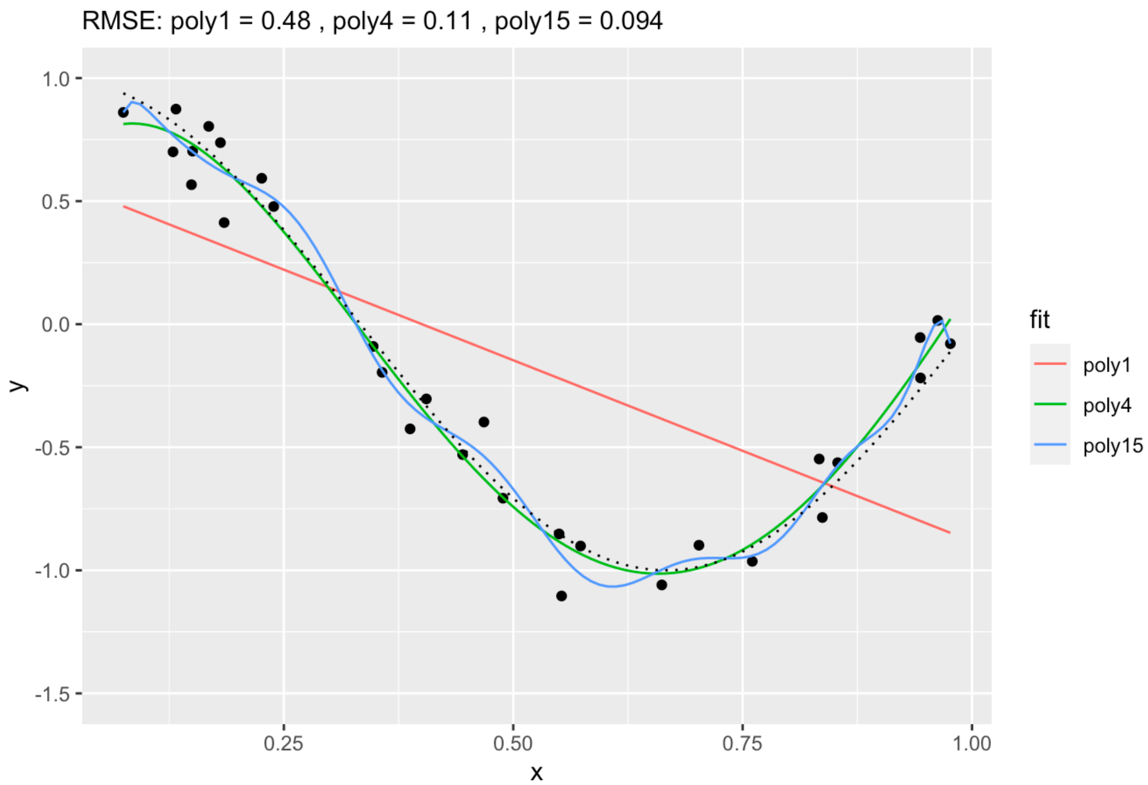

1.0
0.5
0.0
-0.5
-1.0
-1.5
0.25
0.50
0.75
1.00



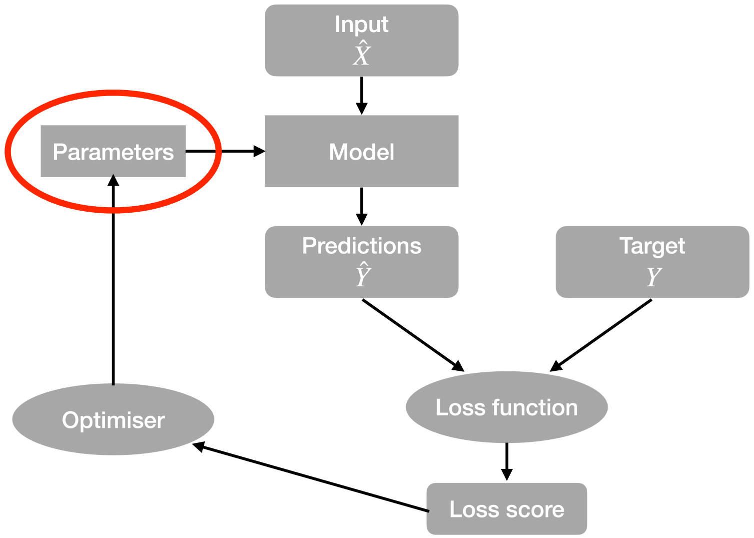
$$y = \sum_{n=0}^{N} a_n x^n$$



Input
$\hat{X}$
Parameters
Model
Predictions
$\hat{Y}$
Target
Y
Optimiser
Loss function
Loss score



RMSE: poly1 = 0.48 , poly4 = 0.11 , poly15 = 0.094
1.0
0.5
0.0
-0.5
-1.0
-1.5
y
0.25
0.50
0.75
1.00
x
fit
poly1
poly4
poly15




```
> polyfit_4

Call:
lm(formula = y ~ poly(x, 4), data = df_train)

Coefficients:
(Intercept)  poly(x, 4)1  poly(x, 4)2  poly(x, 4)3  poly(x, 4)4
    -0.1310      -2.3504       2.4388       0.7023      -0.3058
```

Data and the modelling task

Data reading

```
daily_fluxes <- read_csv("./data/FLX_CH-Dav_FLUXNET2015_FULLSET_DD_1997-2014_1-3.c

  # select only the variables we are interested in
  dplyr::select(TIMESTAMP,
                GPP_NT_VUT_REF,    # the target
                ends_with("_QC"),  # quality control info
                ends_with("_F"),   # includes all all meteorological covariates
                -contains("JSB")   # weird useless variable
                ) |>

  # convert to a nice date object
  dplyr::mutate(TIMESTAMP = lubridate::ymd(TIMESTAMP)) |>

  # set all -9999 to NA
  mutate(across(where(is.numeric), ~na_if(., -9999))) |>

  # retain only data based on >=80% good-quality measurements
  # overwrite bad data with NA (not dropping rows)
  dplyr::mutate(GPP_NT_VUT_REF = ifelse(NEE_VUT_REF_QC < 0.8, NA, GPP_NT_VUT_REF),
                TA_F           = ifelse(TA_F_QC        < 0.8, NA, TA_F),
                SW_IN_F        = ifelse(SW_IN_F_QC     < 0.8, NA, SW_IN_F),
                LW_IN_F        = ifelse(LW_IN_F_QC     < 0.8, NA, LW_IN_F),
                VPD_F          = ifelse(VPD_F_QC       < 0.8, NA, VPD_F),
                PA_F           = ifelse(PA_F_QC        < 0.8, NA, PA_F),
                P_F            = ifelse(P_F_QC         < 0.8, NA, P_F),
                WS_F           = ifelse(WS_F_QC        < 0.8, NA, WS_F)) |>

  # drop QC variables (no longer needed)
  dplyr::select(-ends_with("_QC"))
```

Model formulation

Base-R:

```r
lm(GPP_NT_VUT_REF ~ SW_IN_F + VPD_F + TA_F, data = daily_fluxes)
```

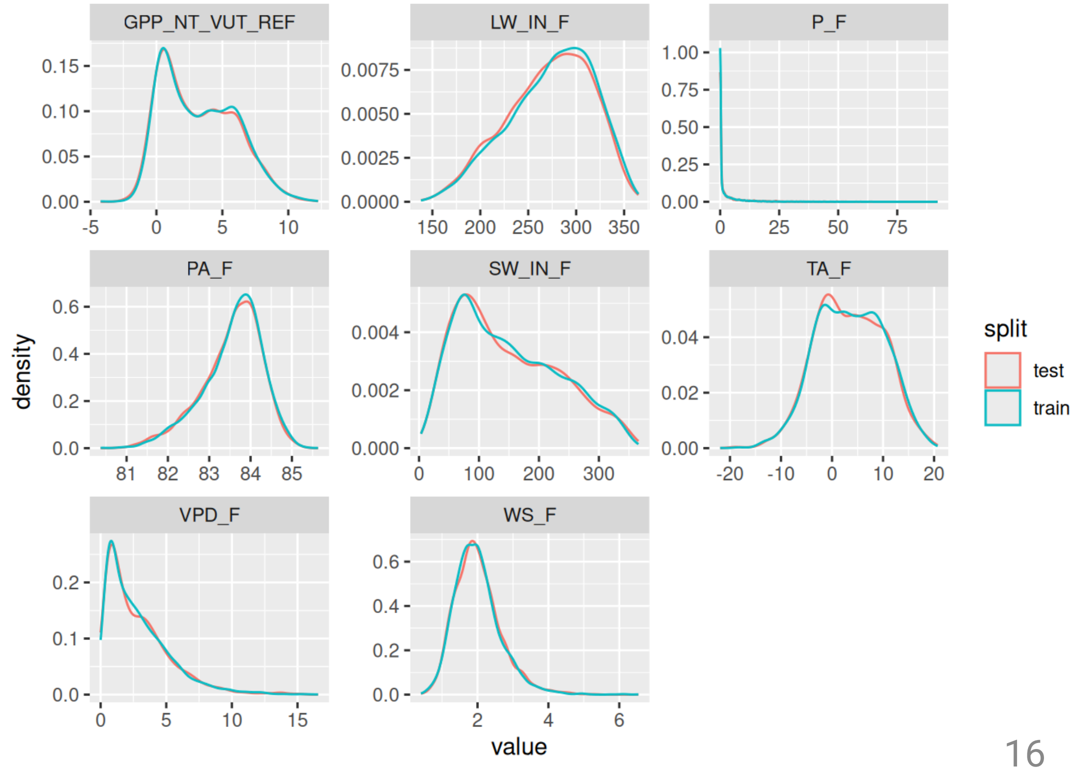
Caret:

```r
caret::train(
  form = GPP_NT_VUT_REF ~ SW_IN_F + VPD_F + TA_F,
  data = daily_fluxes |> drop_na(),  # drop missing values
  trControl = caret::trainControl(method = "none"),  # no resampling
  method = "lm"
)
```

```r
caret::train(
  form = GPP_NT_VUT_REF ~ SW_IN_F + VPD_F + TA_F,
  data = daily_fluxes |> drop_na(),
  trControl = caret::trainControl(method = "none"),
  method = "knn"
)
```

Data splitting

```r
set.seed(123)  # for reproducibility
split <- rsample::initial_split(daily_fluxes, prop = 0.7, strata = "VPD_F")
daily_fluxes_train <- rsample::training(split)
daily_fluxes_test <- rsample::testing(split)
```

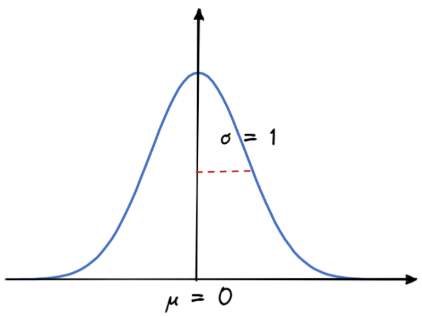


Pre-processing

- Data transformation to improve model training
- Must consider data splitting: Do not use test data for determining data transformation parameters! Danger of *data leakage*.
- Therefore, pre-processing is to be specified as part of the model training (And not applied to the data before model training!)

Example:

- Standardisation




```
pp <- recipes::recipe(GPP_NT_VUT_REF ~ SW_IN_F + VPD_F + TA_F, data = daily_fluxes_
  recipes::step_center(recipes::all_numeric(), -recipes::all_outcomes()) |>
  recipes::step_scale(recipes::all_numeric(), -recipes::all_outcomes())
```

```
caret::train(
  pp,
  data = daily_fluxes_train,
  method = "knn",
  trControl = caret::trainControl(method = "none")
)
```

https://algodaily.com/lessons/standardization-and-normalization

Pre-processing: imputation

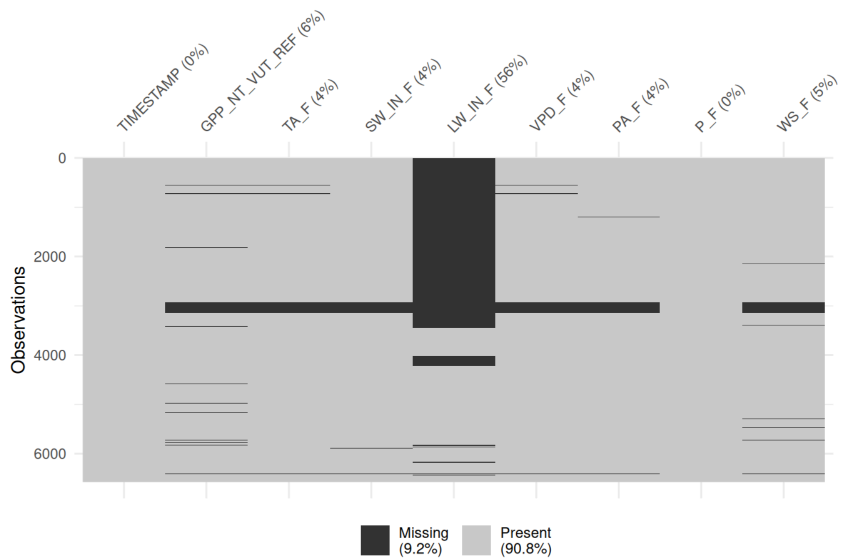
- Imputation = filling missing data
- Can improve model training by avoiding loss of data due to NA removal (removing entire rows if algorithm doesn't allow NA).
- Must be specified as part of the model training.
- Always look at pattern of missing data.

Example:

- Median imputation

```
pp |>
  step_impute_median(all_predictors())
```

```
visdat::vis_miss(
  daily_fluxes,
  cluster = FALSE,
  warn_large_data = FALSE
  )
```



Pre-processing: imputation: one-hot encoding

- Makes categorical data numeric.
- Numeric required by certain algorithms.

```
# A tibble: 4 × 2
     id color
  <int> <chr>
1     1 red
2     2 red
3     3 green
4     4 blue
```

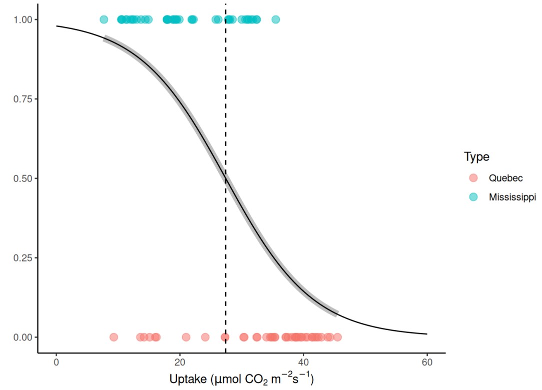
```
  id colorblue colorgreen colorred
1  1         0          0        1
2  2         0          0        1
3  3         0          1        0
4  4         1          0        0
```

```
recipe(GPP_NT_VUT_REF ~ ., data = daily_fluxes) |>
  step_dummy(all_nominal(), one_hot = TRUE)
```

Target engineering

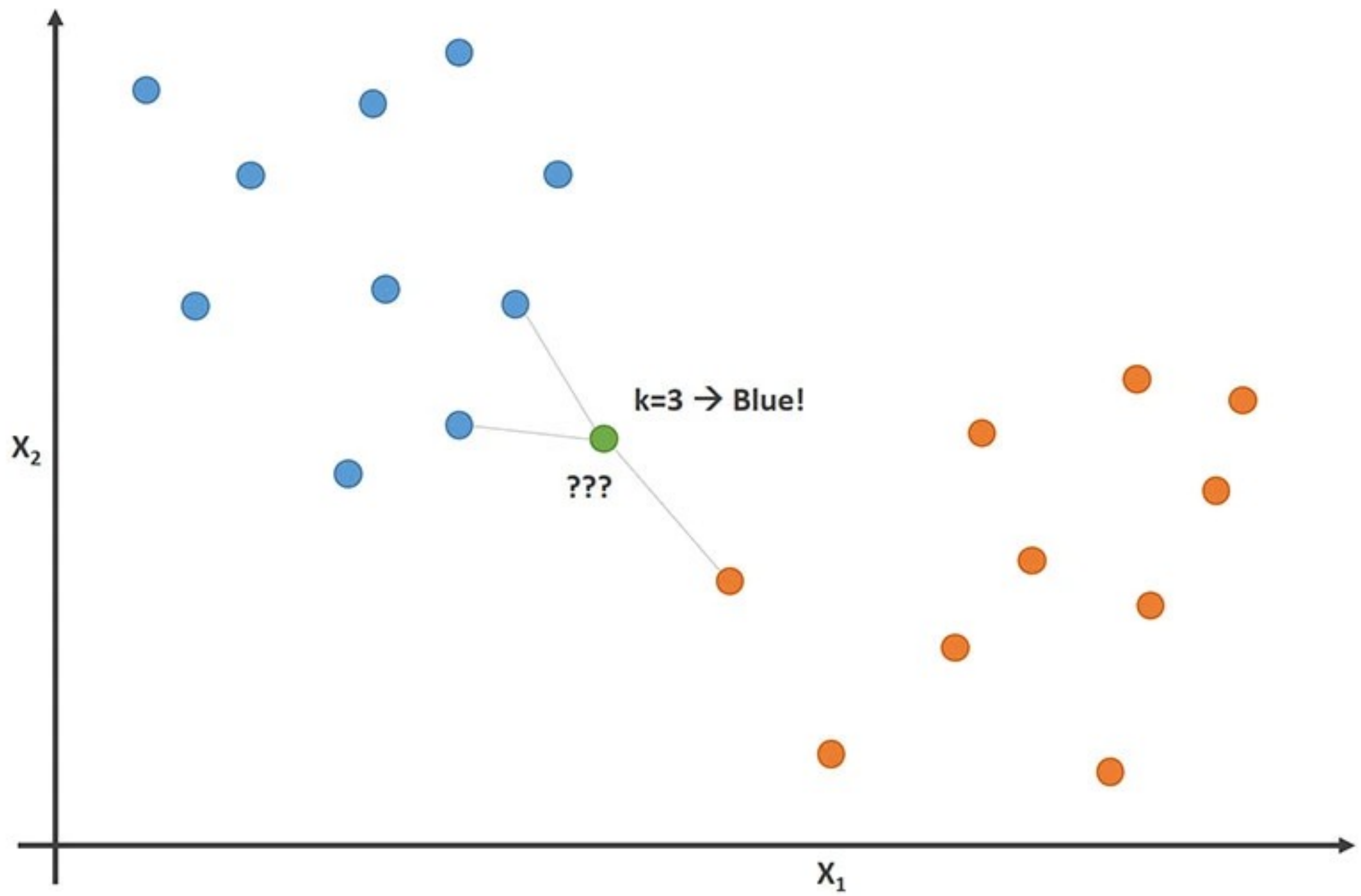
- Can improve model predictions when the target variable has a skewed distribution or when it is bounded between 0 and 1.
- Can be critical if model assumes a certain distribution of prediction errors.

```
recipes::recipe(WS_F ~ ., data = daily_fluxes) |>   # it's of course non-sense to r
  recipes::step_log(all_outcomes())
```



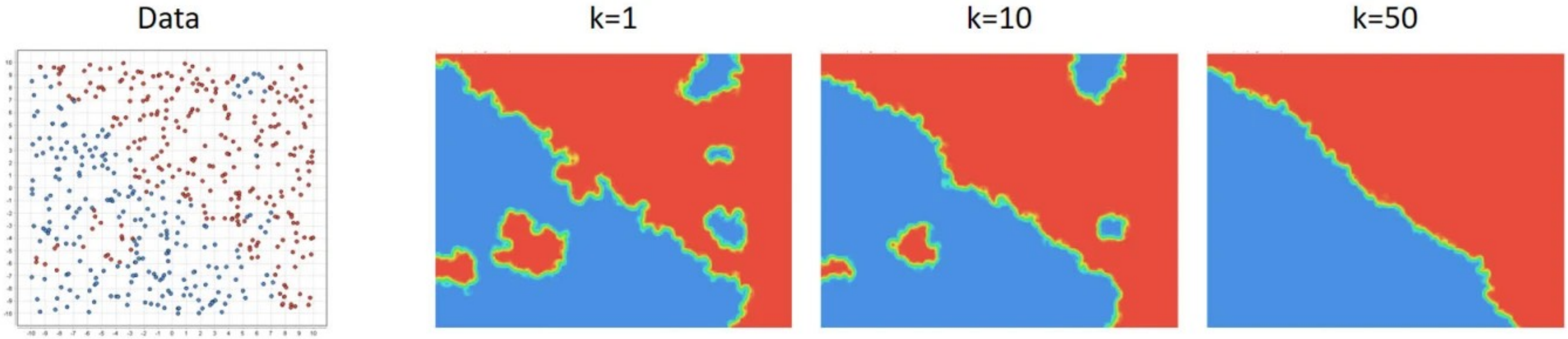
$$\mathrm{logit}(z) = \frac{\exp(z)}{1+\exp(z)}.$$

k-Nearest Neighbours



https://rapidminer.com/blog/k-nearest-neighbors-laziest-machine-learning-technique/

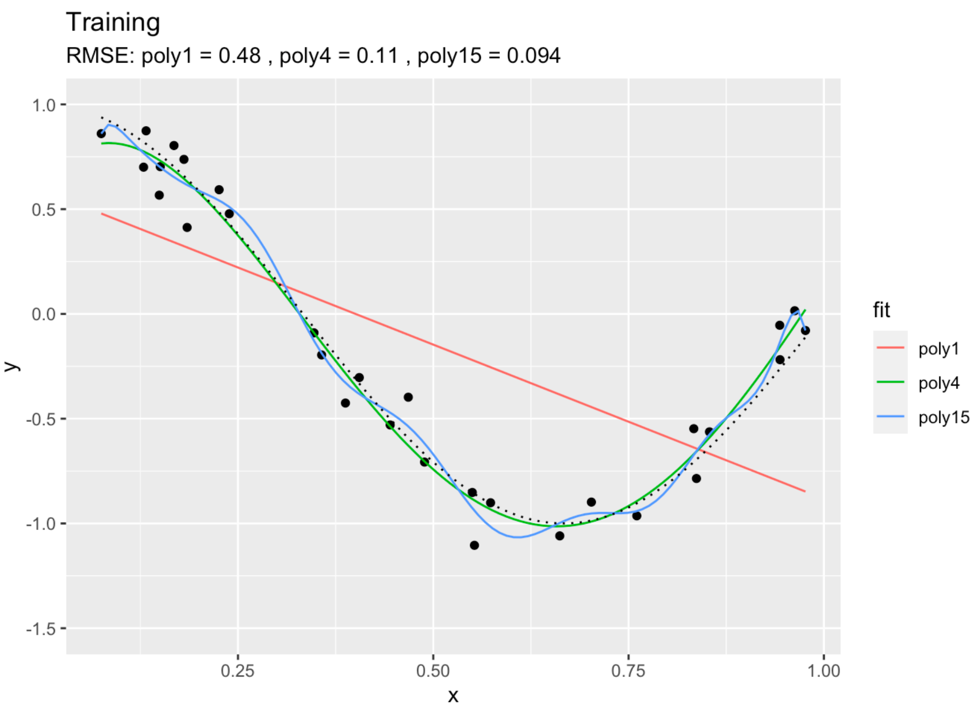
k-Nearest Neighbours

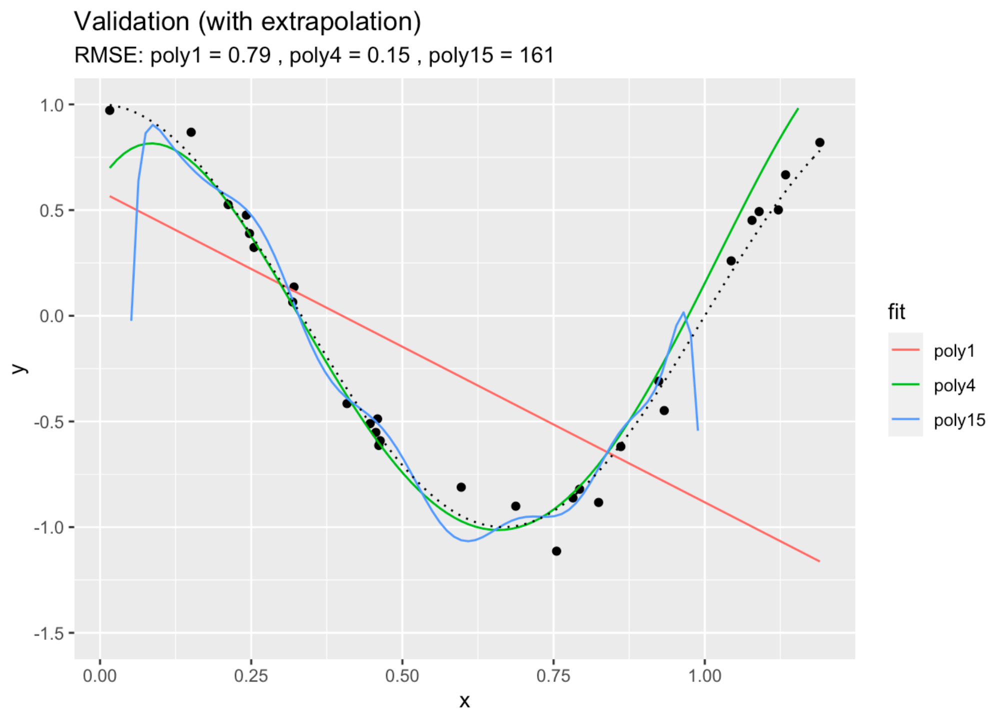


https://rapidminer.com/blog/k-nearest-neighbors-laziest-machine-learning-technique/

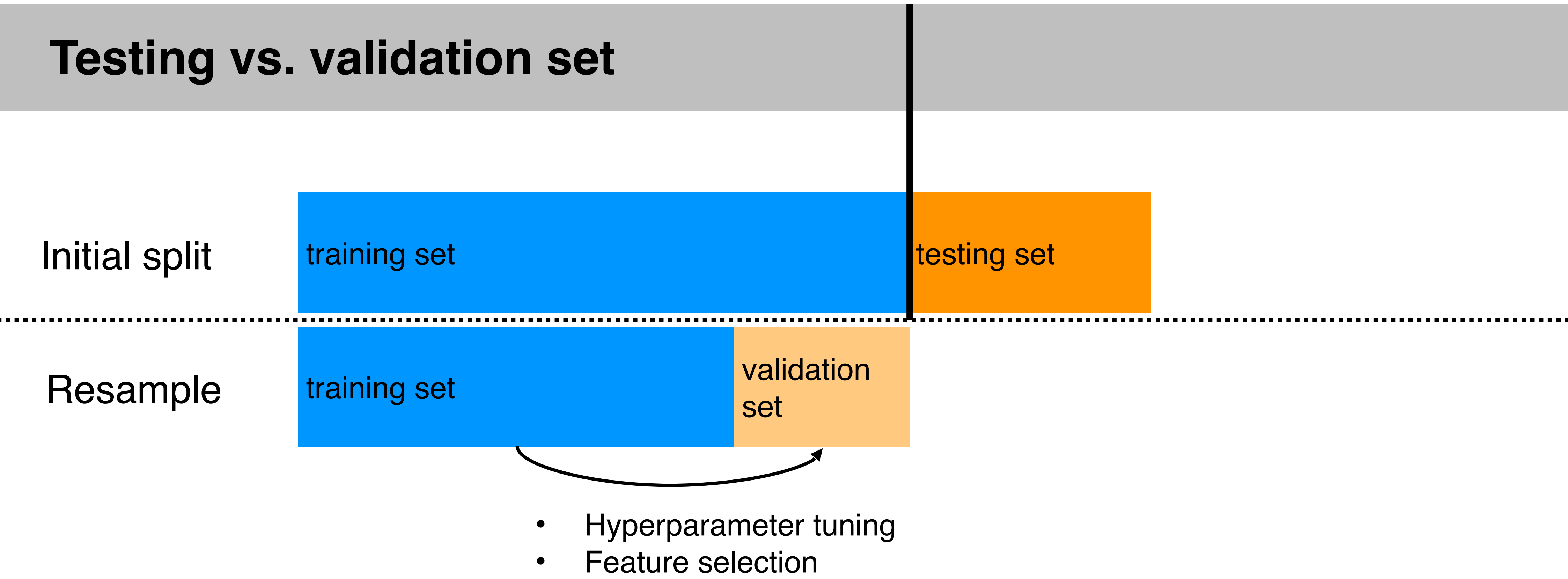
Course evaluation

Validation (with extrapolation)

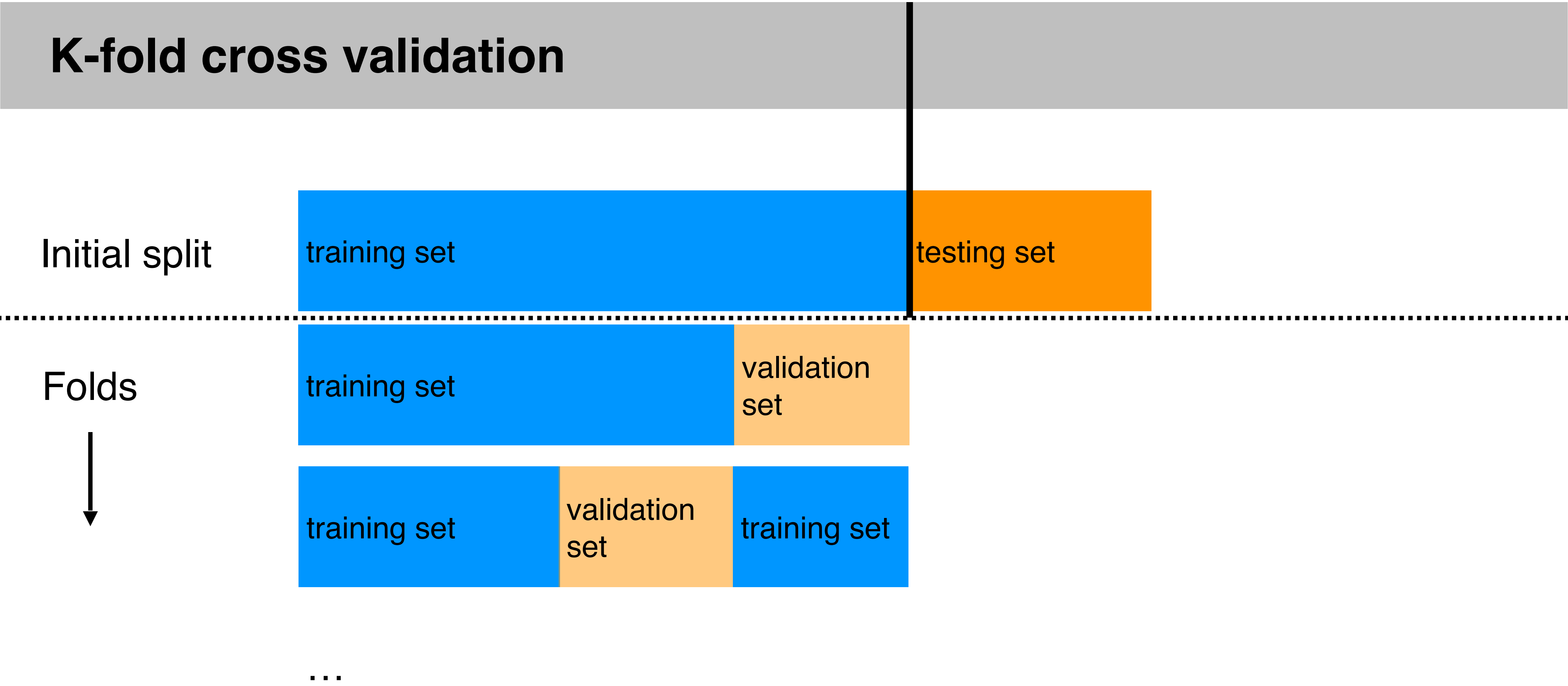




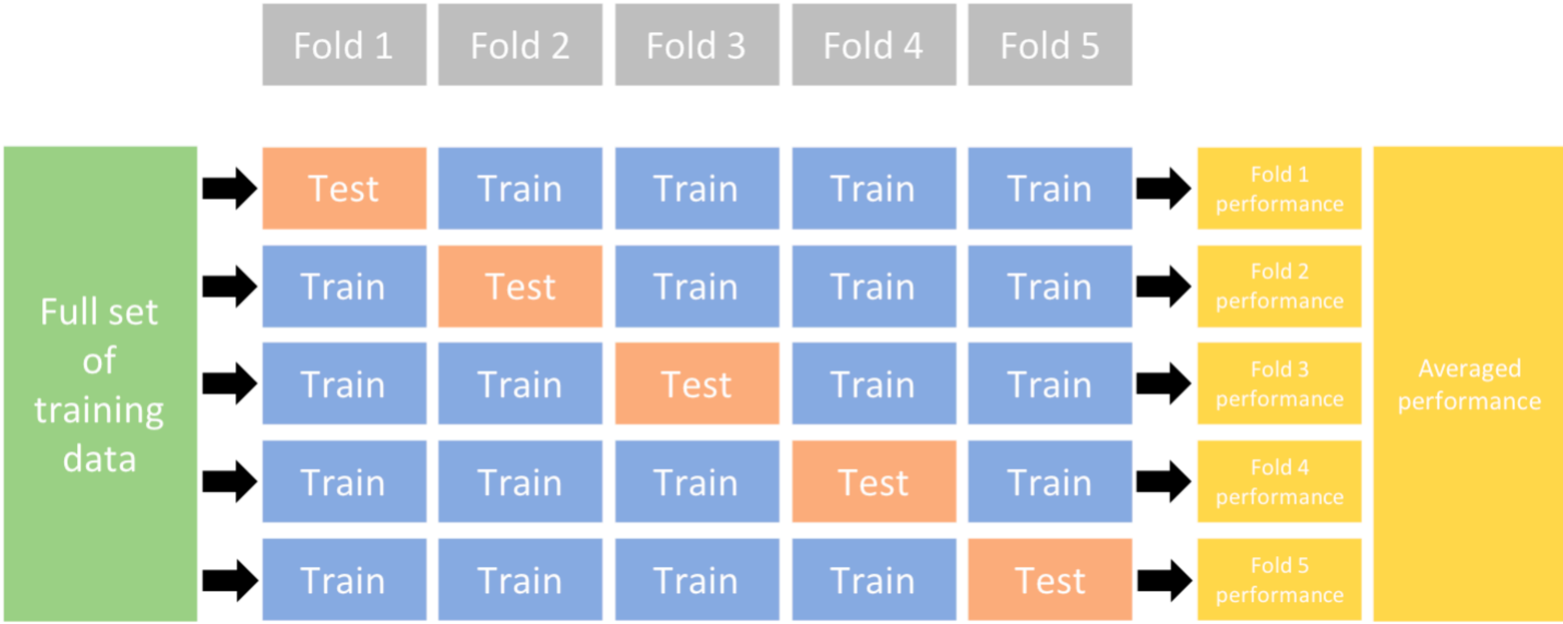
Testing vs. validation set



K-fold cross validation



Resampling



Boehmke & Greenwell (2019) *Hands On Machine Learning in R*

Workflow of model training and testing

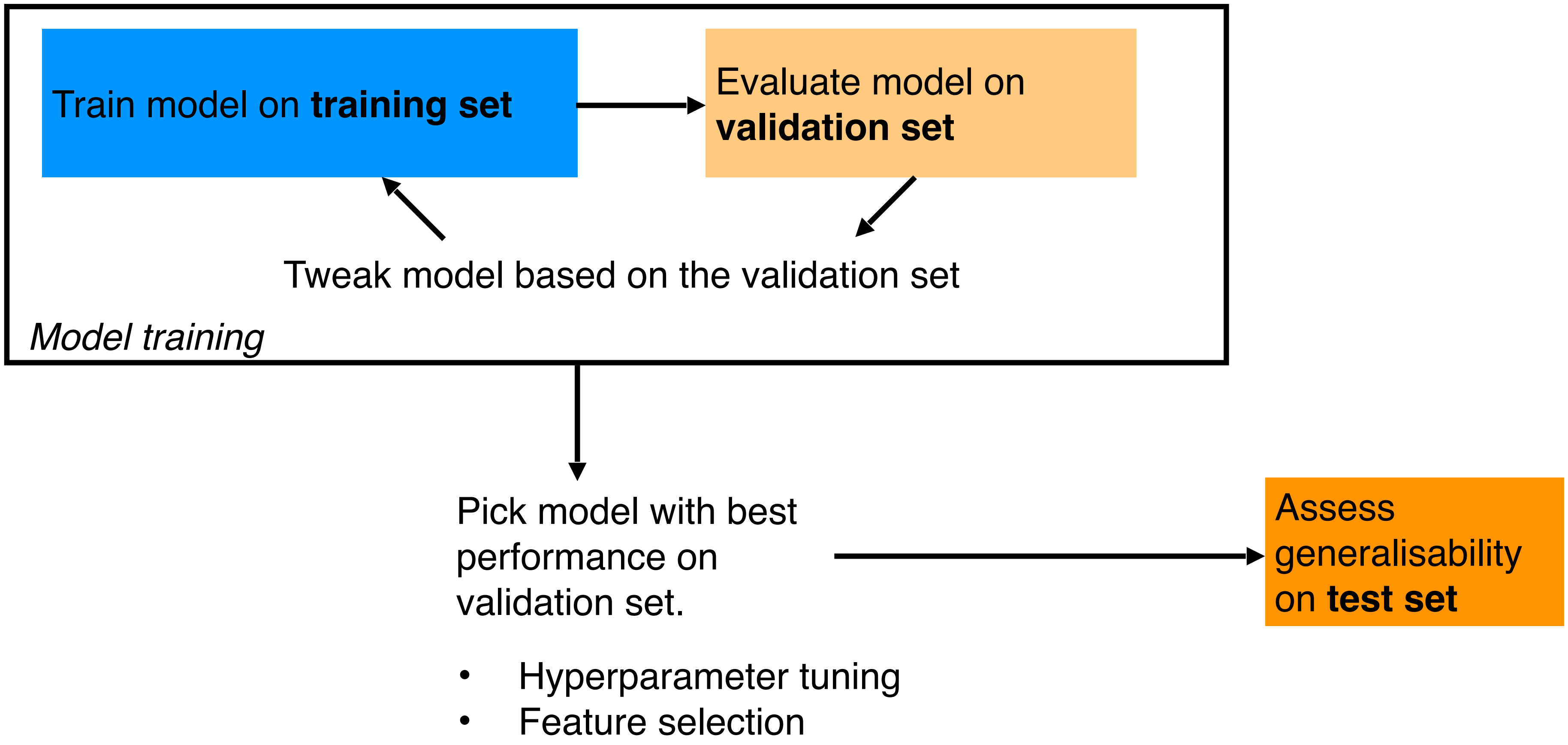


Figure adopted form [Google Machine Learning Crash Course]