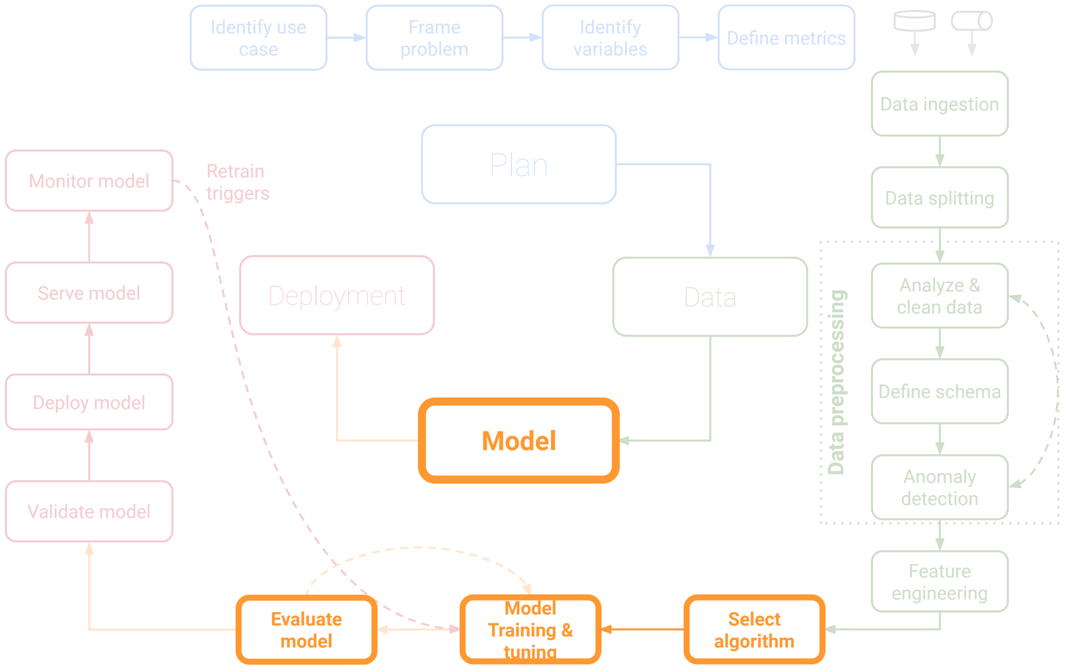
# Models

Fundamentals

Prof. Dr. Jan Kirenz
HdM Stuttgart

# Data Science Lifecycle

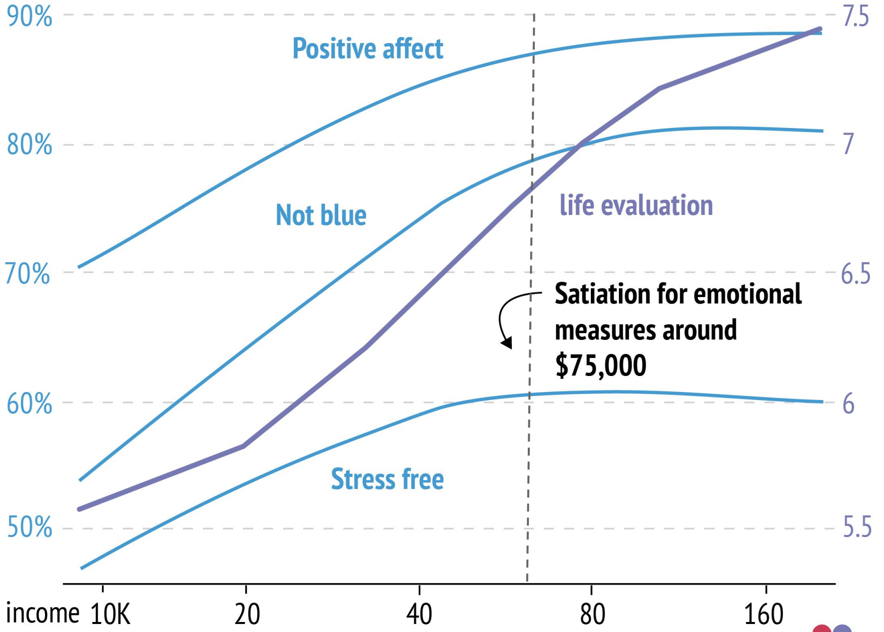
Plan | Data | **Model** | Deployment



**How do we select, train, tune (optimize) and evaluate models?**

Prof. Dr. Jan Kirenz

# Data: Money & Happiness



*Note: X axis scale is not linear.*
*Source: Kahneman and Deaton (2010)*

CNBC

*Widely regarded as one of the world's most influential living psychologist, **Daniel Kahneman** won the Nobel in Economics for his pioneering work in behavioral economics:*

*„Below an income of … $60,000 a year, people are unhappy, and they get progressively unhappier the poorer they get. Above that, we get an absolutely flat line. … Money does not buy you experiential happiness, but lack of money certainly buys you misery."*

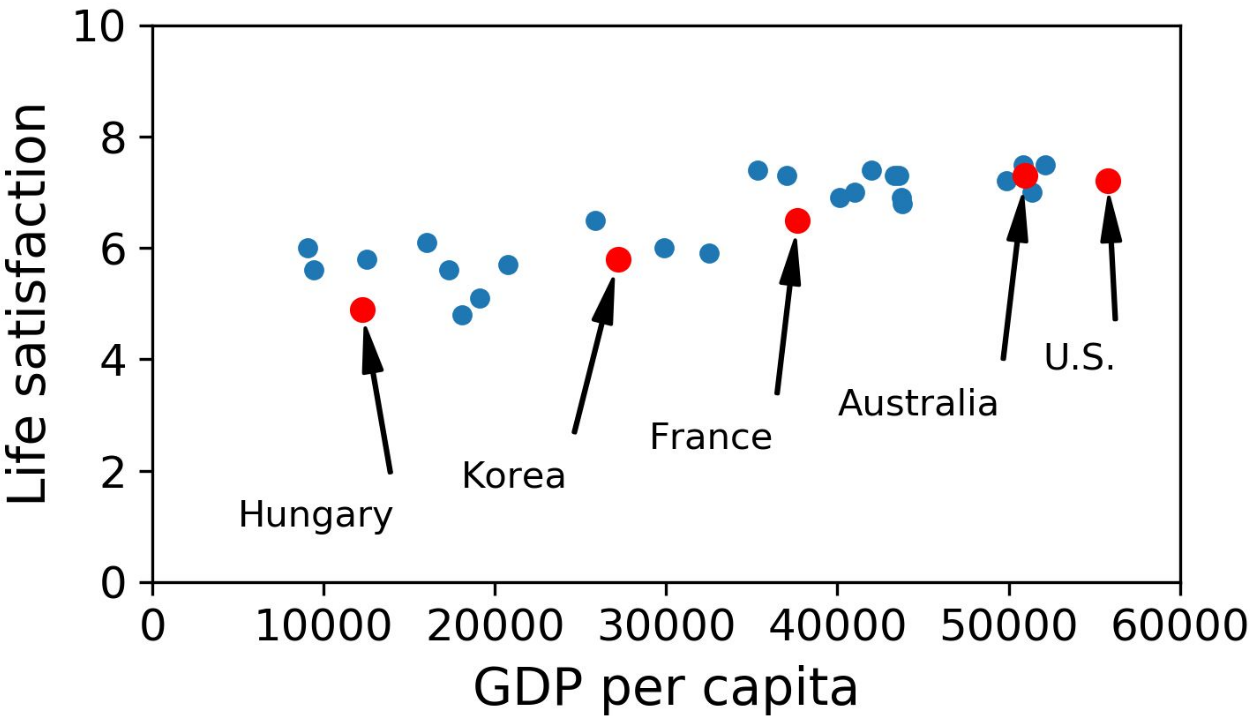
*Watch TED-talk:* TED

# Money can buy happiness, but only to a point

# Model can refer to a

1. Type of model (we usually call this “algorithm”)
    - e.g., Linear Regression
2. Fully specified model architecture
    - e.g., Linear Regression with one input and one output.
3. Final trained model
    - e.g. Linear Regression with one input and one output, using $\boldsymbol{\theta}_0$= 4.85 $\boldsymbol{\theta}_1$= 4.91 $\times 10^{-5}$

# Exploratory data analysis (EDA)

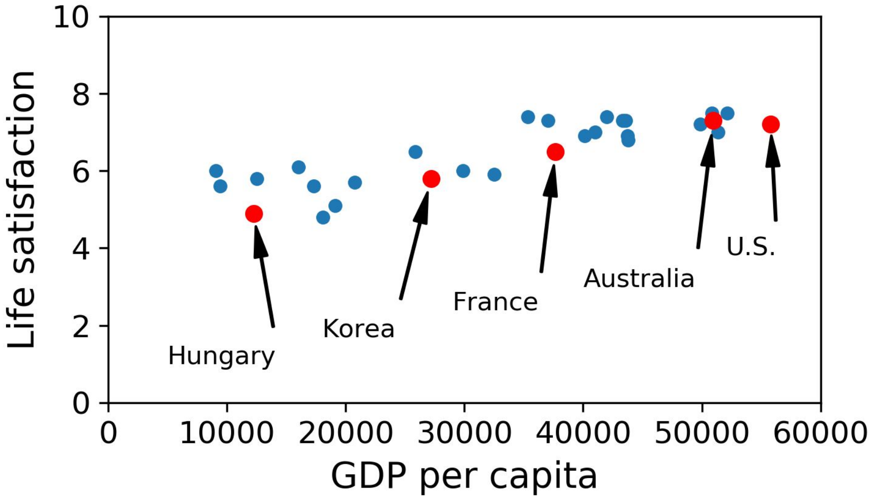


# Do you see a trend?

1. Data exploration gives indication for a trend that is:
   a. Positive or negative?
   b. Linear or non-linear?

   *… but be careful, the data is noisy (i.e., partly random)*

2. Model (type) selection:
   a. Regression or classification?
   b. Linear or non-linear?



# A simple linear regression model

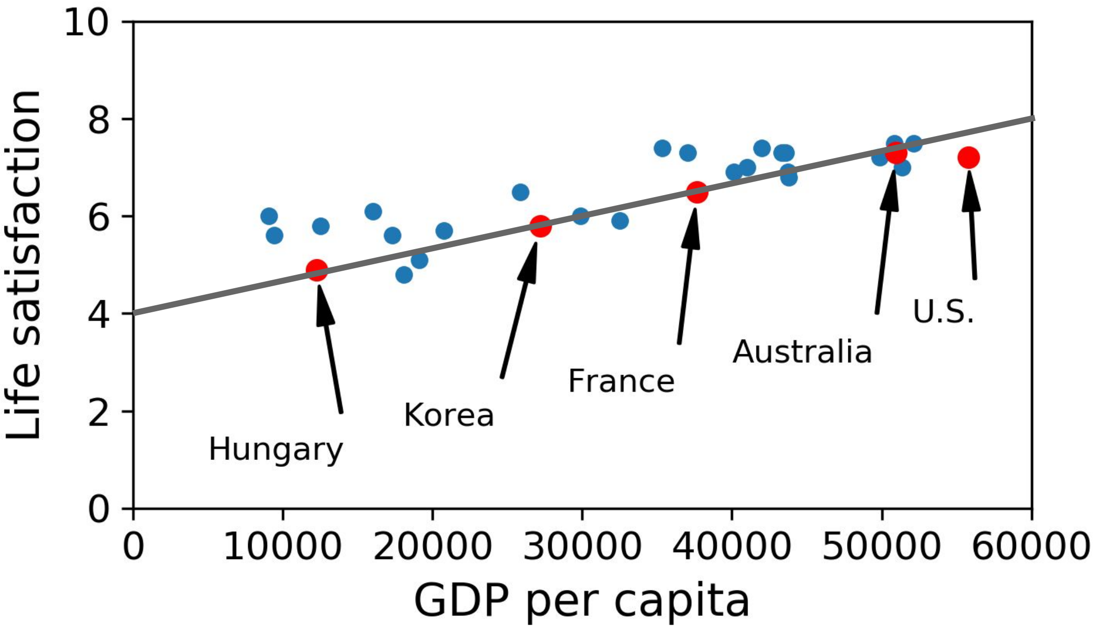
$$\hat{y}_i = \theta_0 + \theta_1 \times x_1$$

- $\hat{y}_i$ is the predicted output (life satisfaction).
- $\theta_0$ is the bias (the y-intercept).
- $\theta_1$ is the slope of our feature 1 (in machine learning often called weight of the feature)
- $x_1$ is our feature GDP (a known input).

All the same, just different notations:

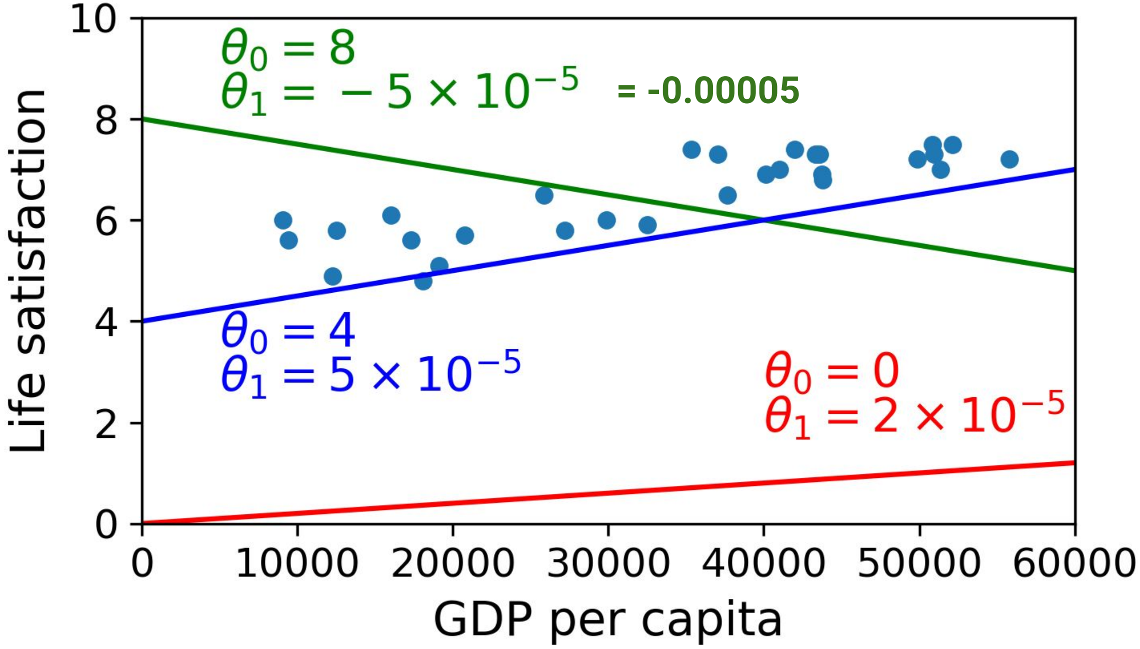
$$\hat{y}_i = b_0 + b_1 \times x_1$$

$$\hat{y}_i = w_0 + w_1 \times x_1$$



# A few possible linear models with different parameters ($\theta$)

$10^{-5} = 0.00001$

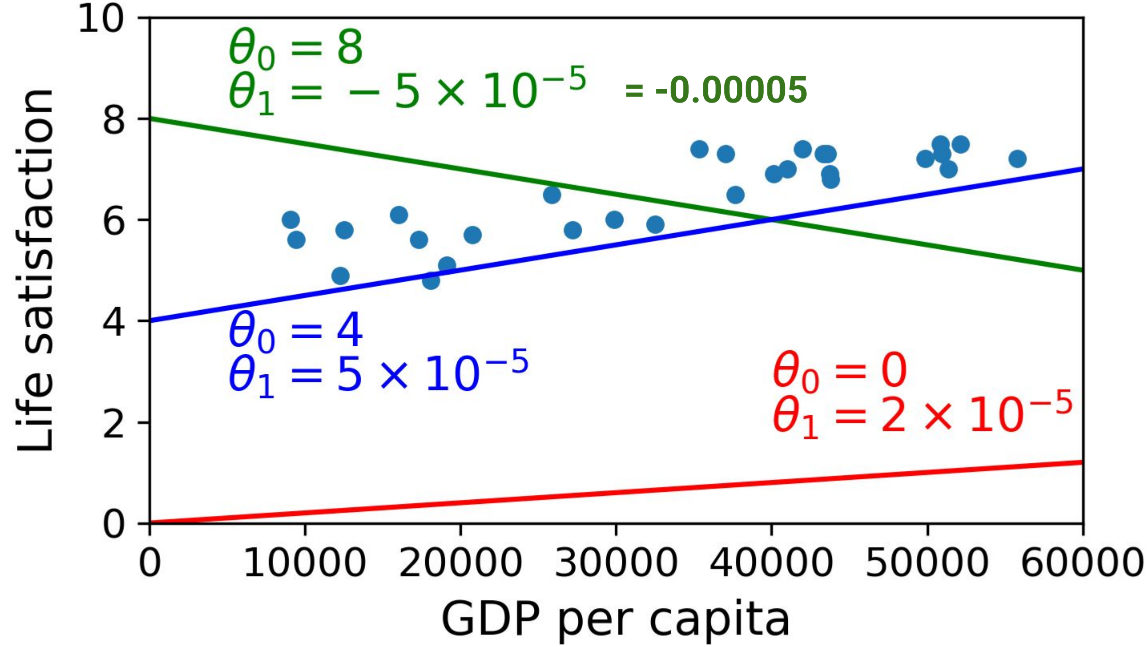


# Model **selection** includes

1. Choosing the type of algorithm
   - *e.g., Linear Regression*
2. Fully specifying its architecture
   - *e.g., Linear Regression with one input and one output.*
3. Fitting (training) the model to find the model parameters that will make it best fit the training data
   - e.g. Linear Regression with one input and one output, using $\boldsymbol{\theta}_0$= 4.85 $\boldsymbol{\theta}_1$= 4.91 × $10^{-5}$

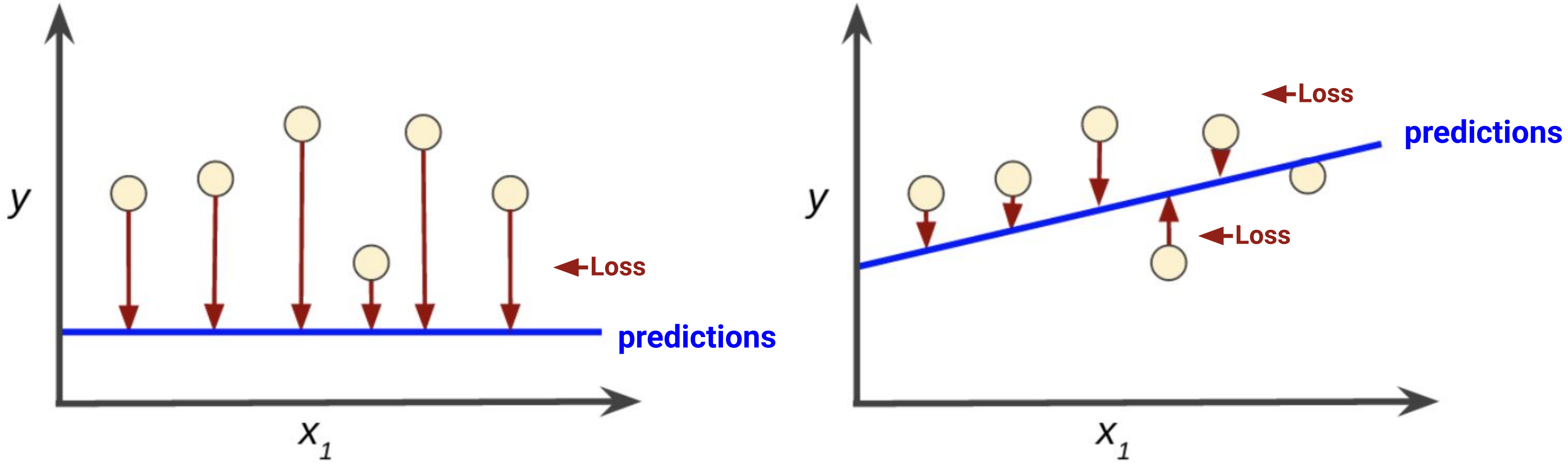
# How to select the **best fitting model**?

$10^{-5} = 0.00001$



**= -0.00005**

# How to select the **best fitting model**?



- The arrows represent residuals (loss).
- The **blue lines** represent predictions.

Source: Google Developers https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss

# We calculate the **squared loss**

The squared loss for a **single observation** (example) is as follows

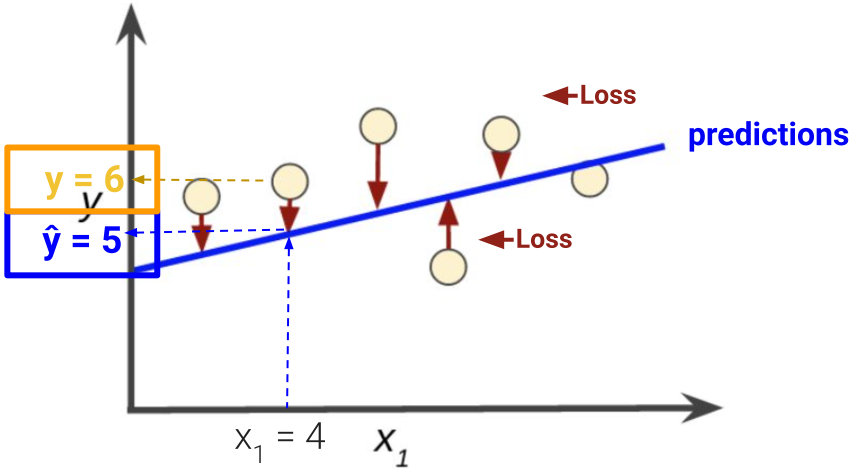
= the square of the difference between the **true outcome (label)** and the **prediction**

= (**observation** - **prediction**(x))$^2$

= (**y** - **ŷ**)$^2$

= (**6** - **5**)$^2$

= 1



# We calculate the **squared loss**

The squared loss for a **single observation** (example) is as follows

= the square of the difference between the **true outcome (label)** and the **prediction**

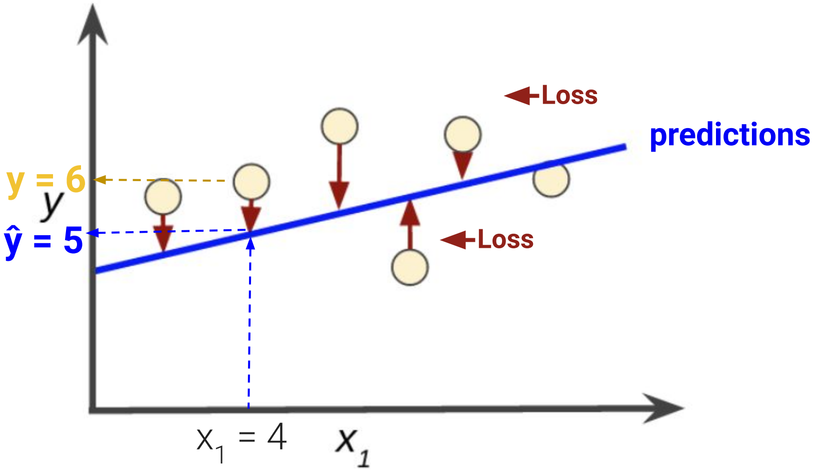
= (**observation** - **prediction**(x))$^2$

$$\boxed{= (y - \hat{y})^2}$$

= (**6** - **5**)$^2$

= 1

$$\boxed{(y_i - \hat{y}_i)^2}$$

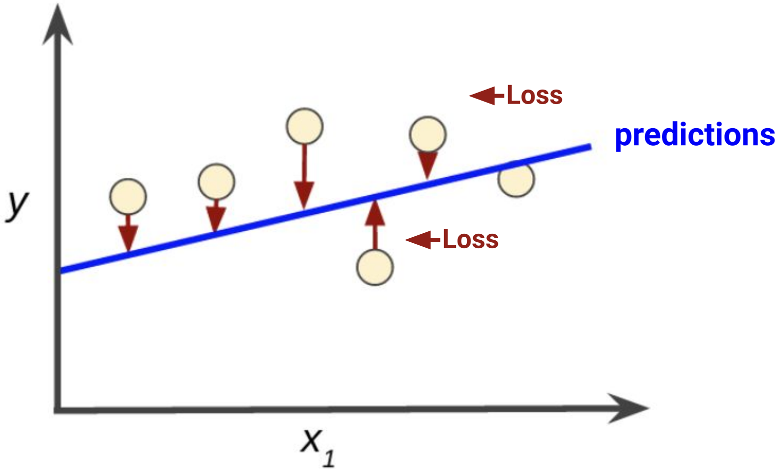


# **Mean squared error** (squared loss; $L_2$ loss)

Mean square error (MSE) is the **average squared loss per example** over the whole dataset.

To calculate MSE, sum up all the squared losses for individual examples and then divide by the number of examples:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\boxed{(y_i-\hat{y}_i)^2}$$

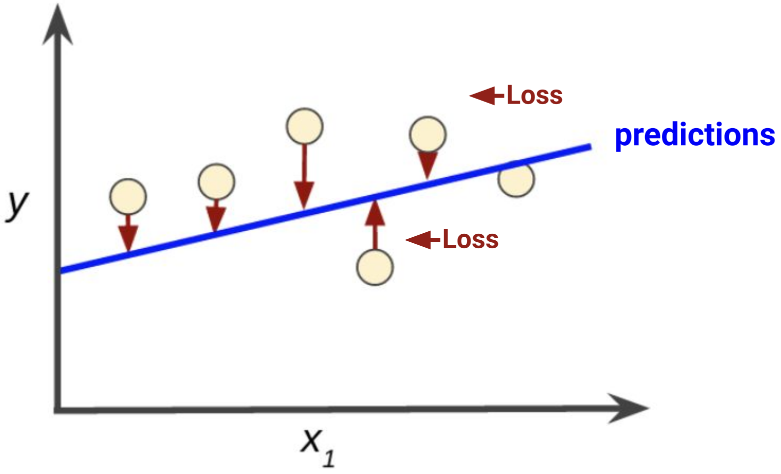


# **Mean squared error** (squared loss; $L_2$ loss)

Mean square error (MSE) is the average squared loss per example over the whole dataset.

To calculate MSE, **sum up** all the squared losses for **individual examples** and then divide by the number of examples:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

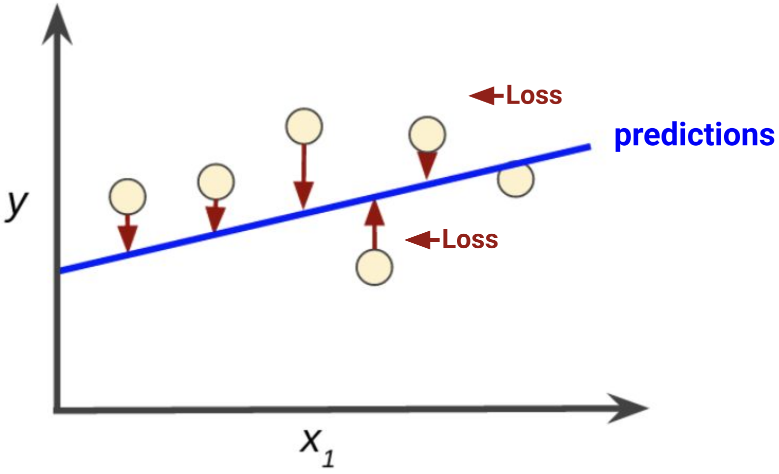


# **Mean squared error** (squared loss; $L_2$ loss)

Mean square error (MSE) is the average squared loss per example over the whole dataset.

To calculate MSE, **sum up** all the squared losses for **individual examples** and then **divide by** the **number** of **examples**:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

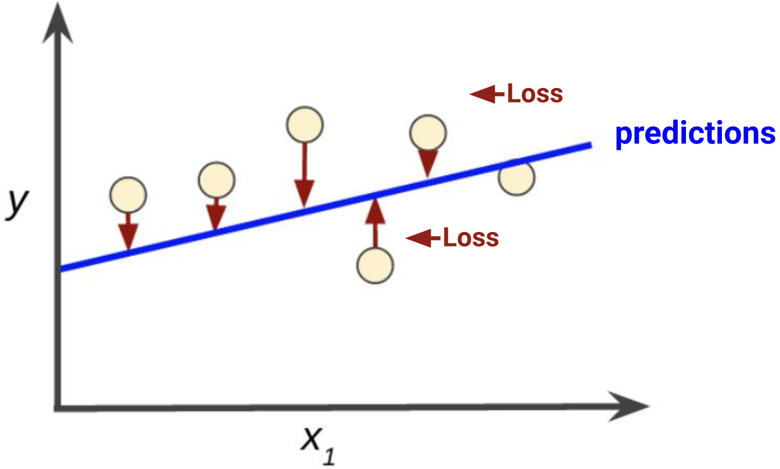


# **Mean squared error** (squared loss; $L_2$ loss)

Mean square error (MSE) is the average squared loss per example over the whole dataset.

To calculate MSE, sum up all the squared losses for individual examples and then divide by the number of examples:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$



# Does money make people happier?

- Get the data at GitHub:

GitHub

- Code in Colab:

colab

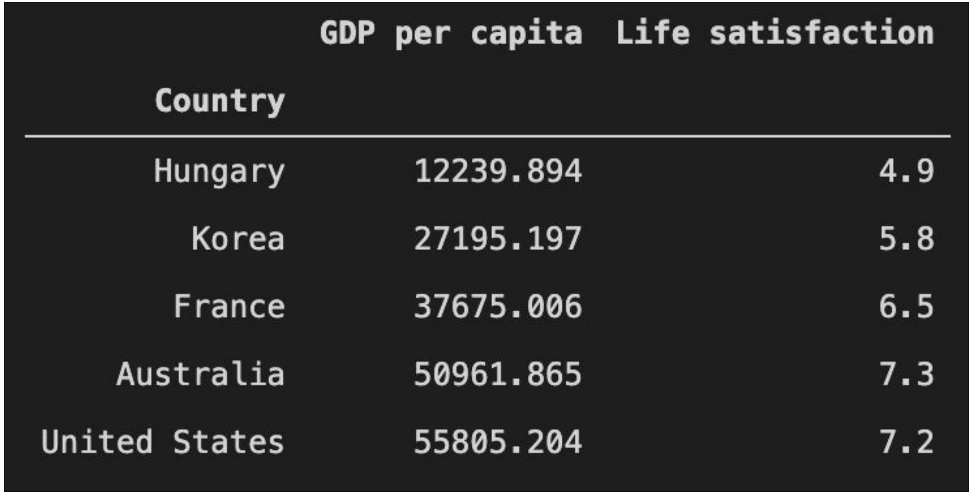
Raw data:

OECD Better Life Index data: **Life satisfaction**

💾 ORGANISATION FOR ECONOMIC CO-OPERATION AND DEVELOPMENT OECD.Stat

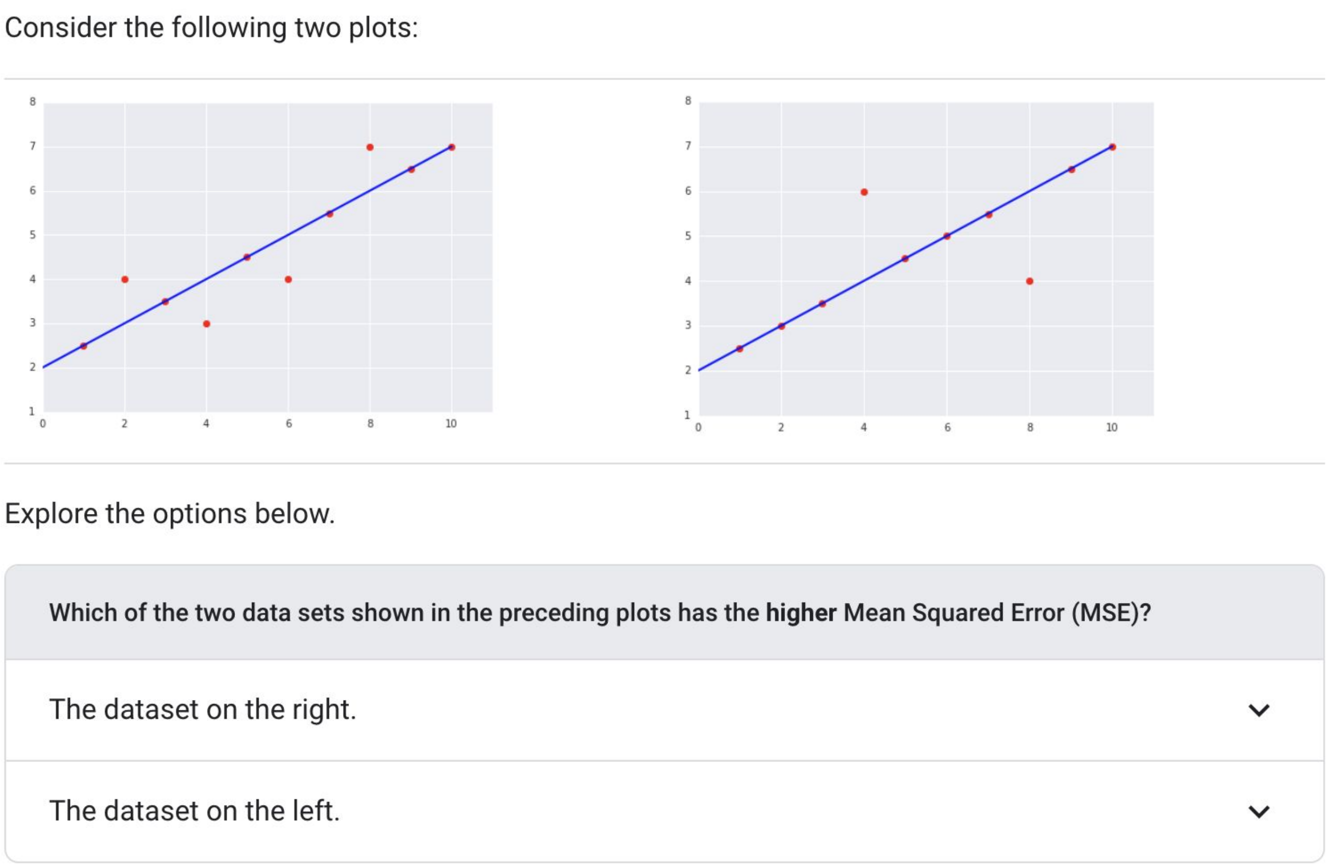
IMF: **Gross domestic product per capita**

💾 INTERNATIONAL MONETARY FUND



| | GDP per capita | Life satisfaction |
|---|---|---|
| Country | | |
| Hungary | 12239.894 | 4.9 |
| Korea | 27195.197 | 5.8 |
| France | 37675.006 | 6.5 |
| Australia | 50961.865 | 7.3 |
| United States | 55805.204 | 7.2 |

# Mean Squared Error



Consider the following two plots:

Explore the options below.

**Which of the two data sets shown in the preceding plots has the higher** Mean Squared Error (MSE)?

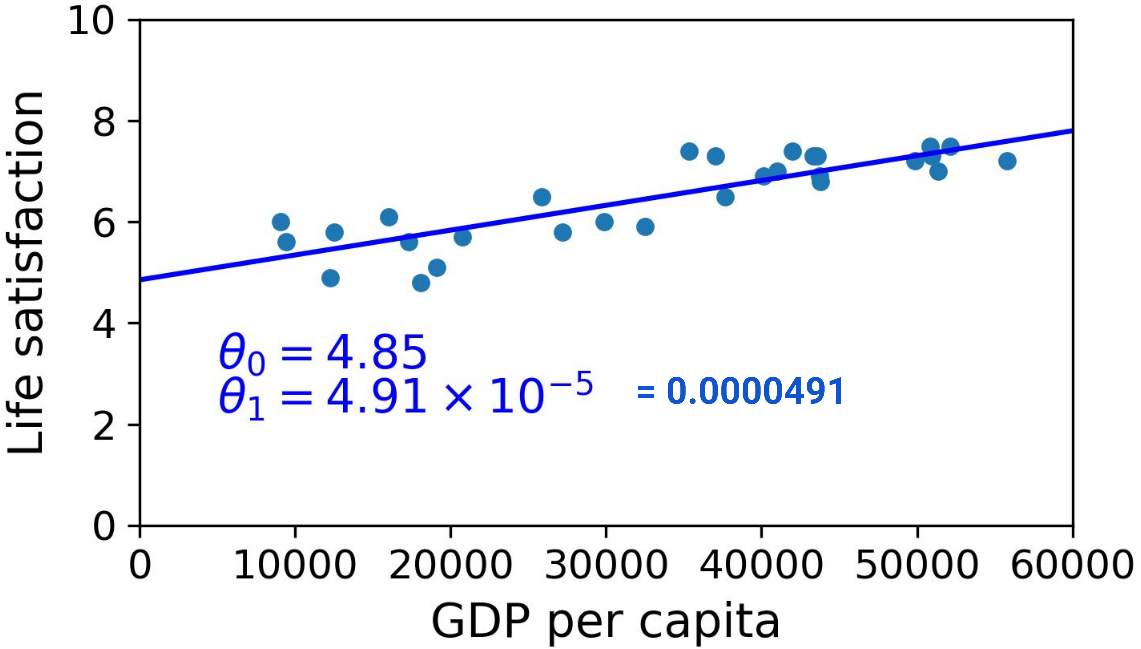
The dataset on the right.

The dataset on the left.

Source: Google Developers https://developers.google.com/machine-learning/crash-course/descending-into-ml/check-your-understanding

# Our **best fitting model**

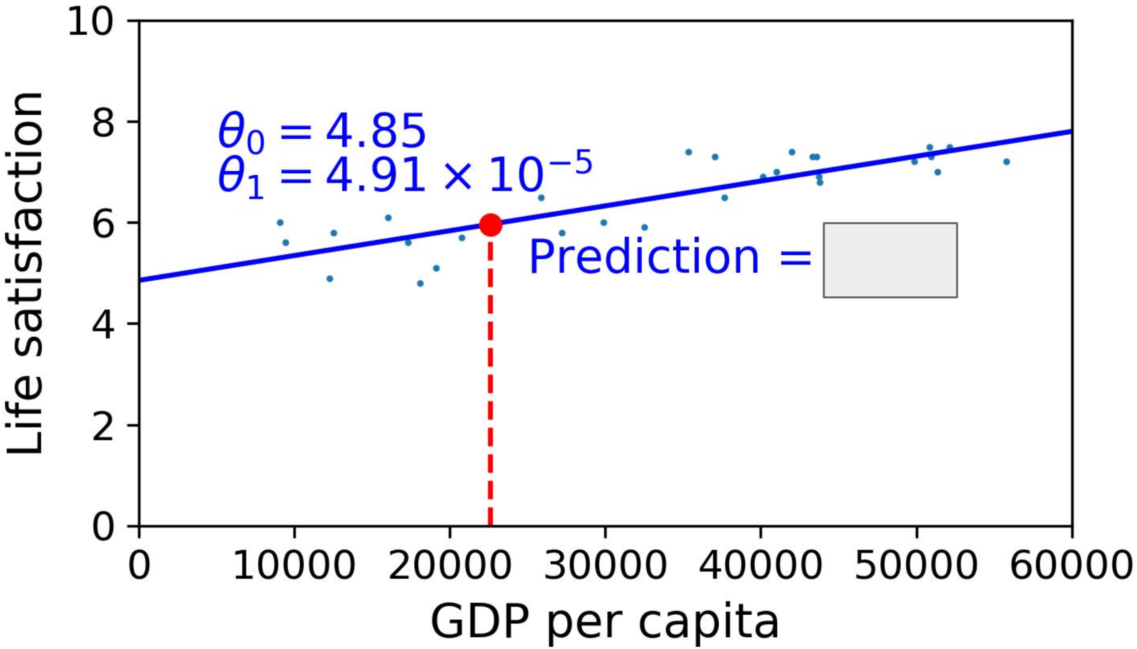
We used the **mean squared error** to select the best model.



**= 0.0000491**

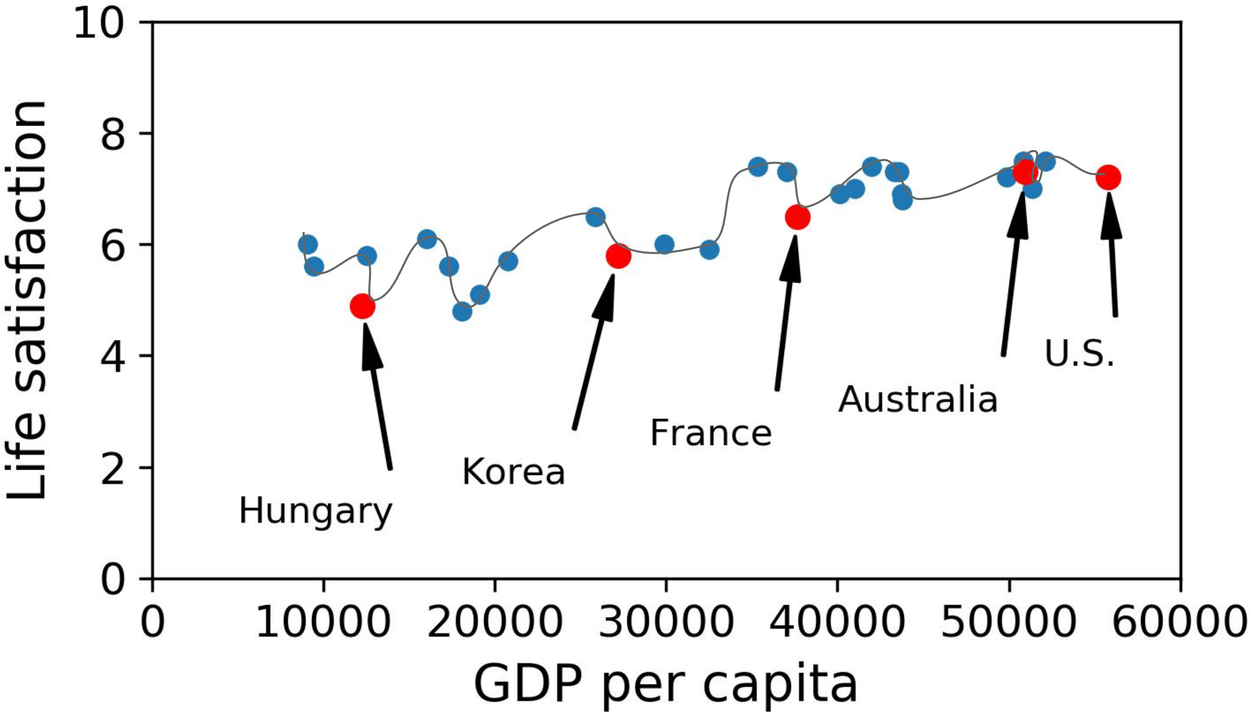
# Use your best fitting model to **predict** the life satisfaction of new data (Cypriots).

Cyprus's GDP per capita: $22587.



Prof. Dr. Jan Kirenz

# Can a model be too good? Yes, it can

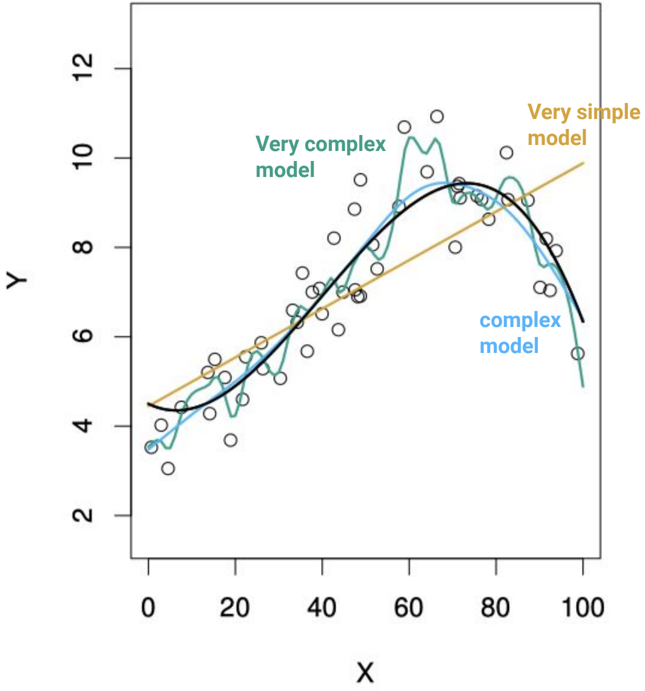


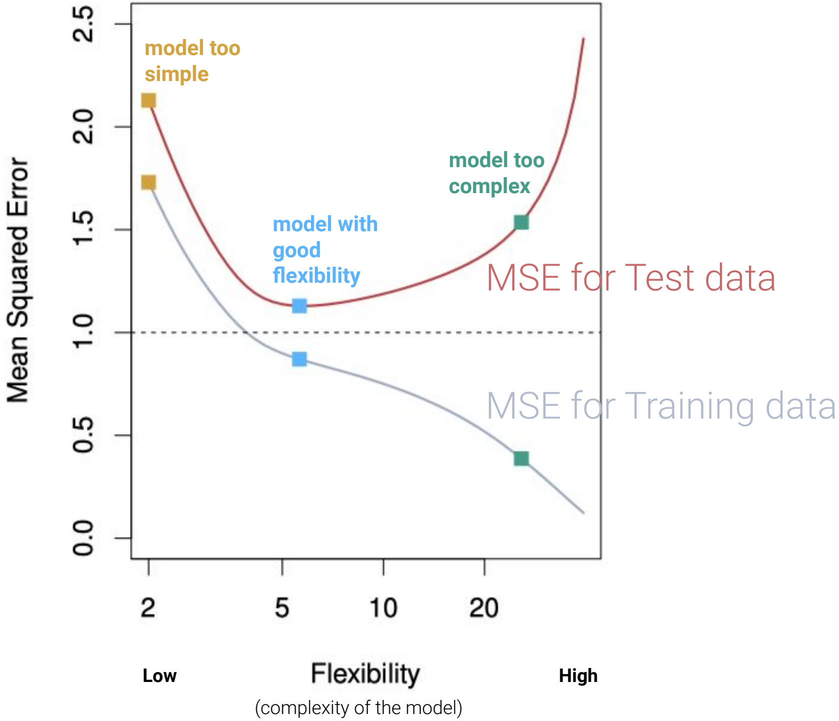
# **Bias-Variance** trade-off

Underfitting ← → Overfitting

High bias
Low variance

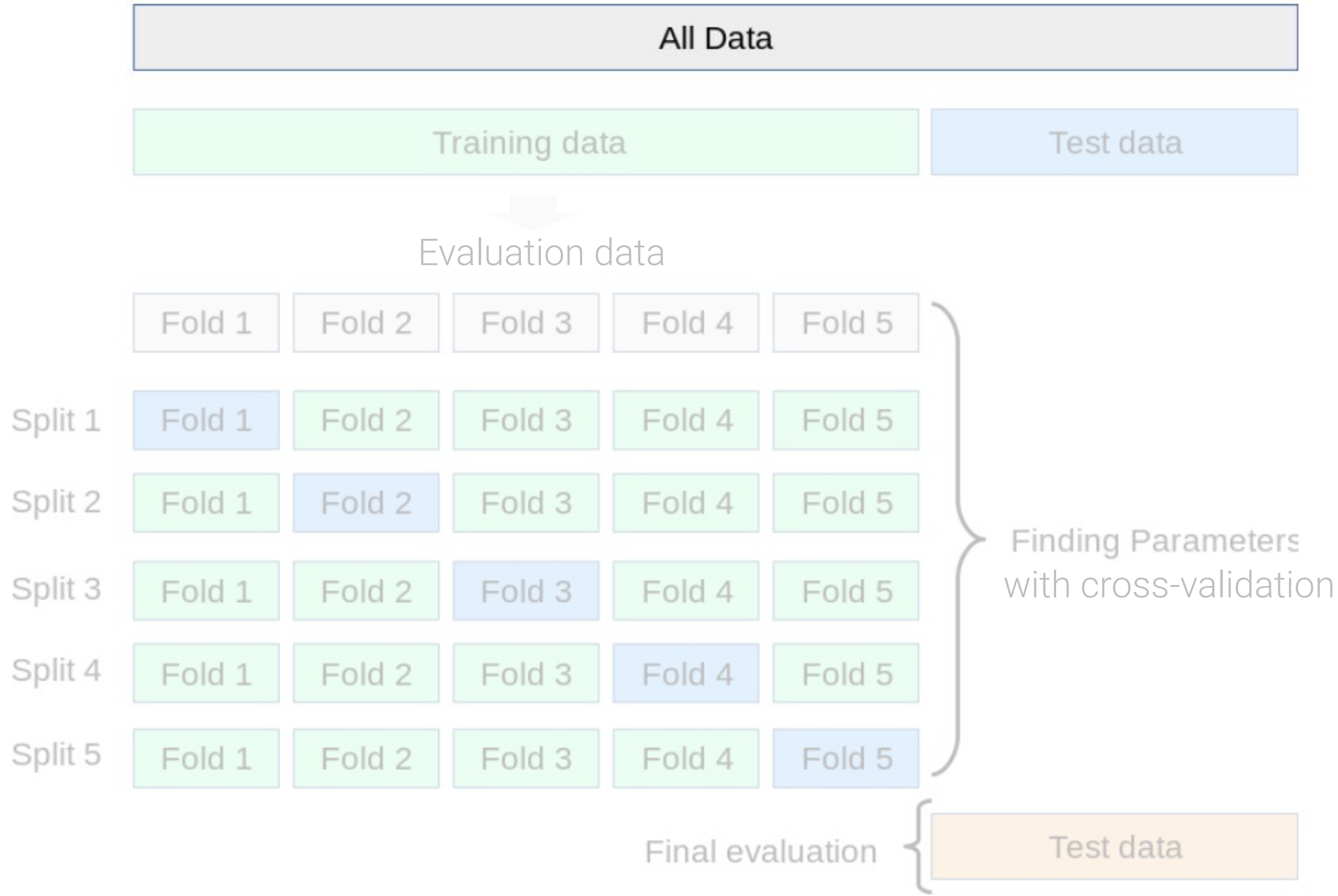
Low bias
High variance



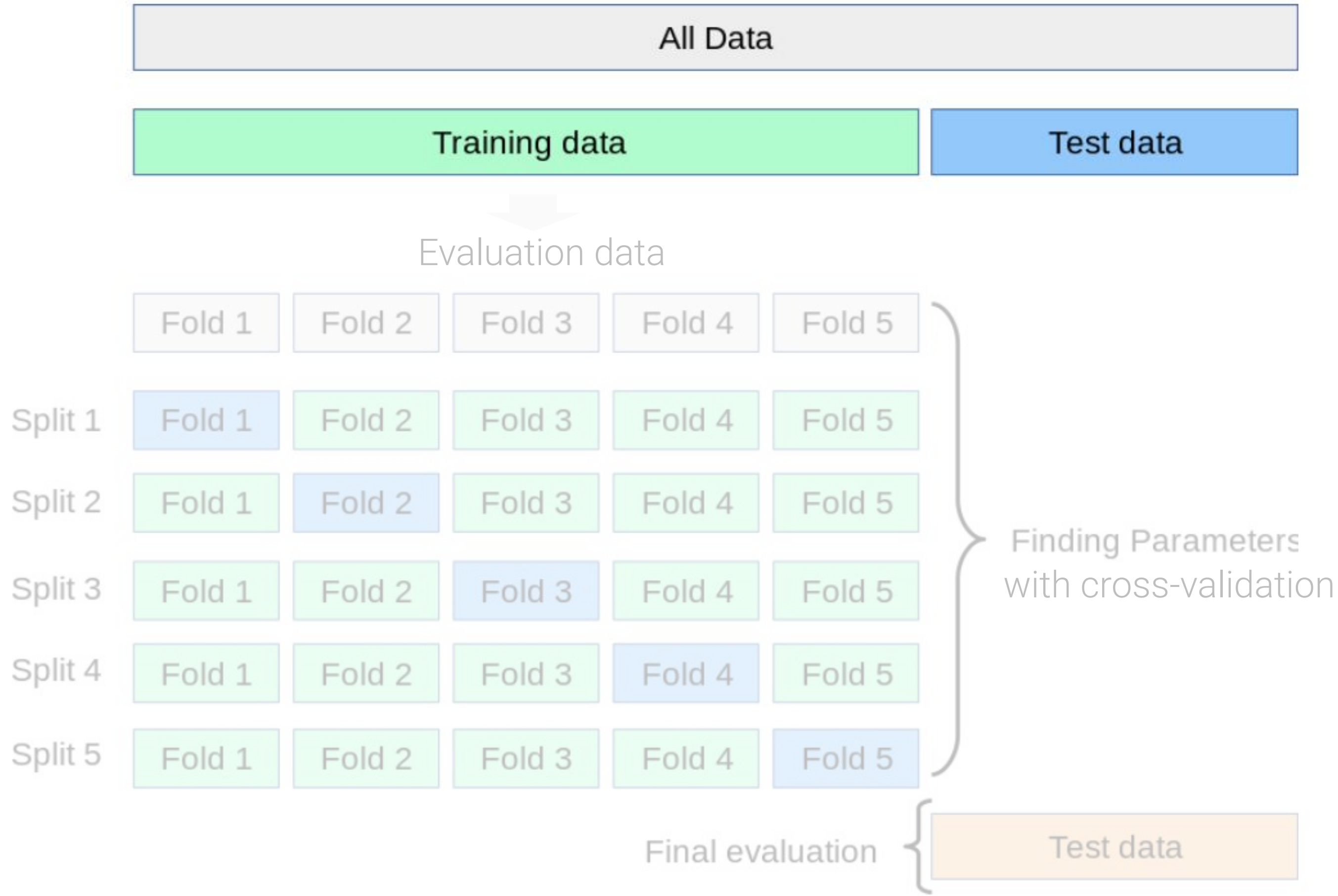


Source: James et. al. (2013)

# Training, evaluation and testing data



Source: scikit-learn [https://scikit-learn.org/stable/modules/cross_validation.html](https://scikit-learn.org/stable/modules/cross_validation.html)



All Data

Training data

Test data

Evaluation data

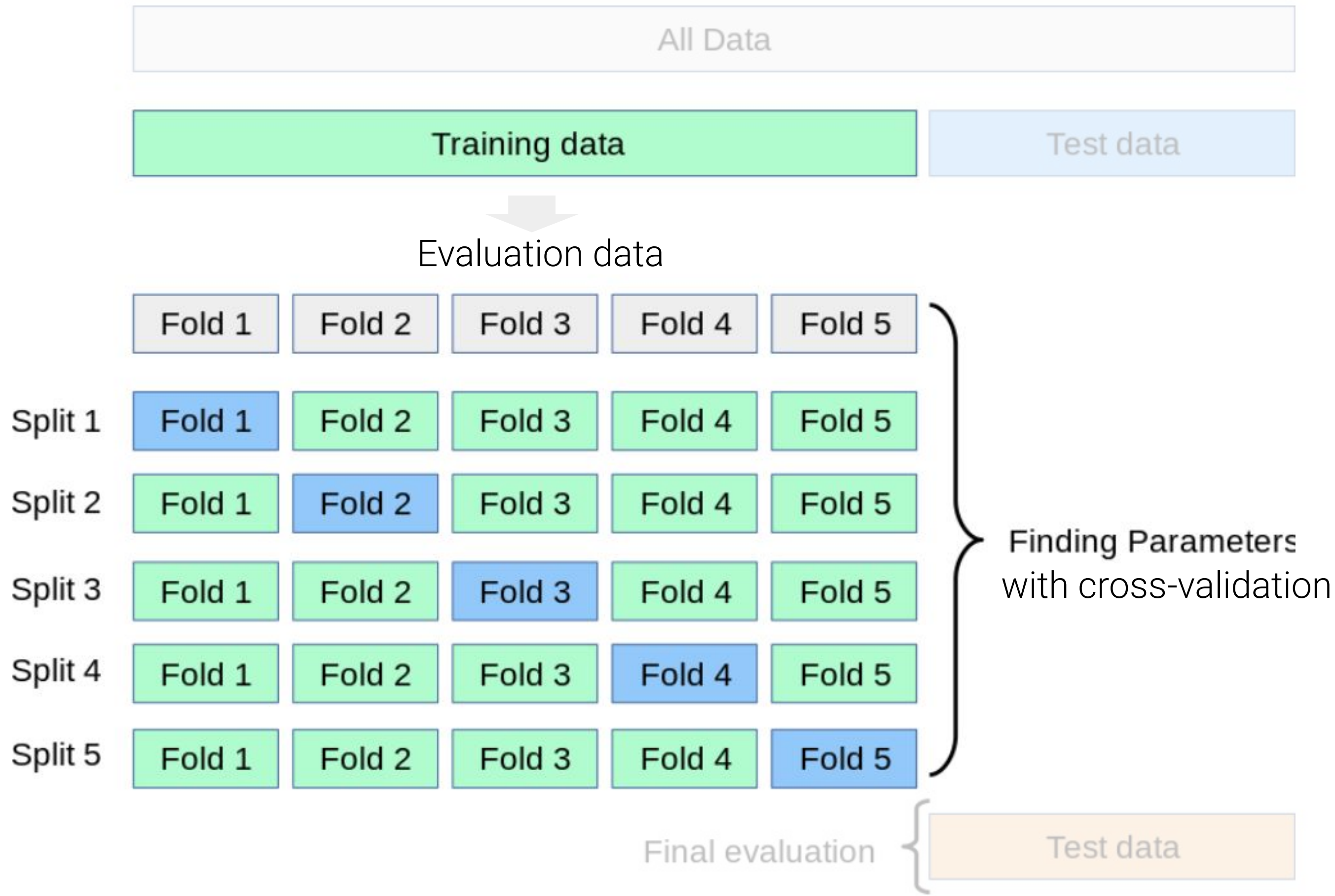
|  |  |  |  |  |  |
|---|---|---|---|---|---|
|  | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 1 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 2 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 3 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 4 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 5 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |

Finding Parameters with cross-validation

Final evaluation

Test data

Source: scikit-learn https://scikit-learn.org/stable/modules/cross_validation.html



All Data

| Training data | Test data |
|---|---|

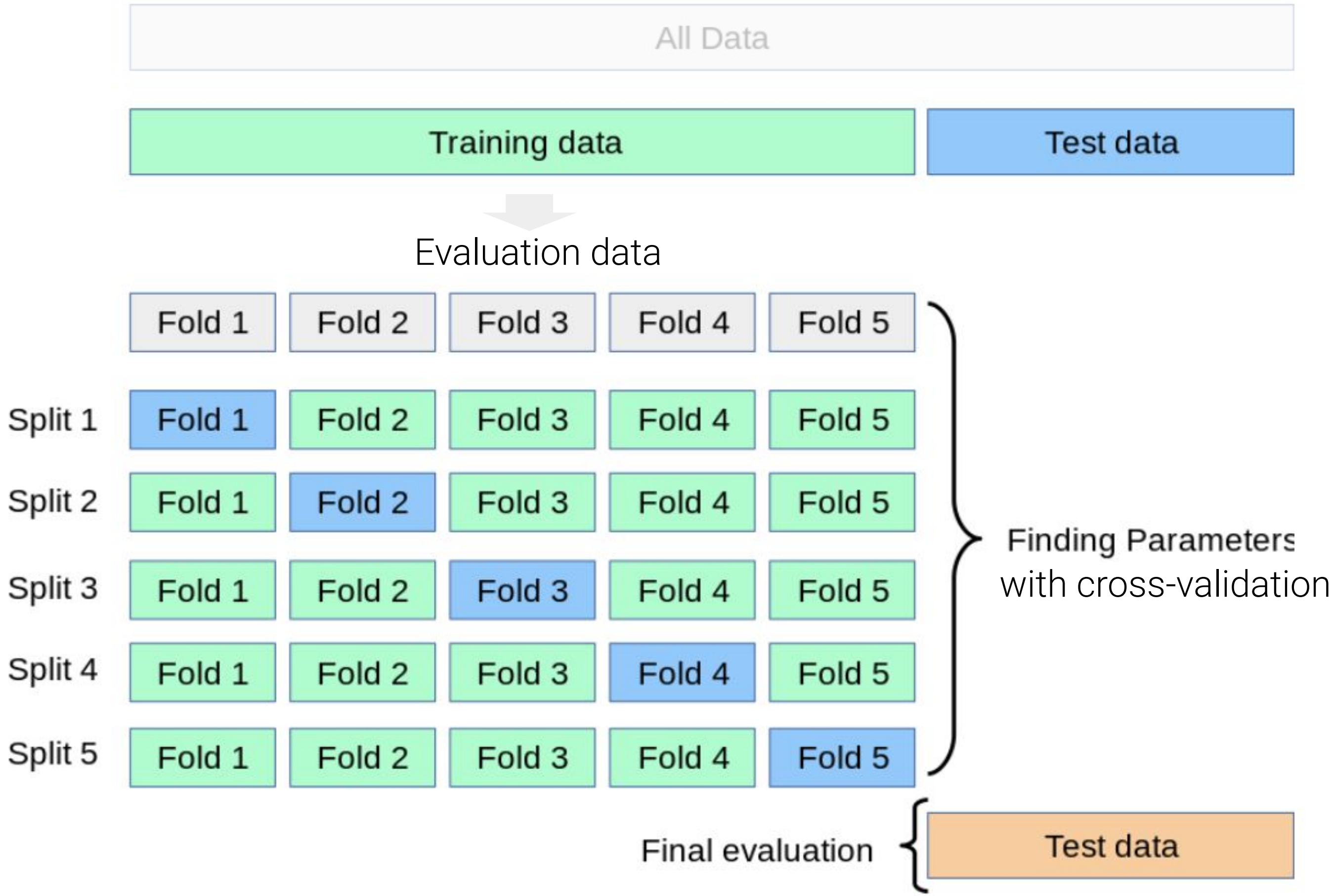
Evaluation data

| | | | | | |
|---|---|---|---|---|---|
| | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 1 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 2 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 3 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 4 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 5 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |

Finding Parameters
with cross-validation

Final evaluation | Test data

Source: scikit-learn https://scikit-learn.org/stable/modules/cross_validation.html



All Data

Training data

Test data

Evaluation data

|  | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
|---|---|---|---|---|---|
| Split 1 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 2 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 3 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 4 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 5 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |

Finding Parameters
with cross-validation

Final evaluation

Test data

Source: scikit-learn https://scikit-learn.org/stable/modules/cross_validation.html

# **Training** set, **evaluation** set and **test** set

### Training set

- Usually 80% of the data, but depends on sample size
- The error rate on training data is called the training error

### Evaluation set (holdout set; development set)

- We split the training set into smaller k sets (usually 5 or 10) to be able to find optimal parameters (**hyperparameters**)
- This approach is called **cross-validation**

### Test set

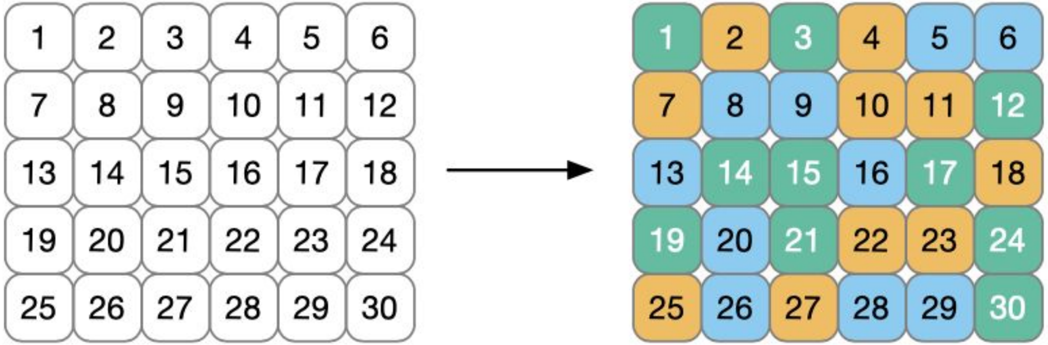
- Also called hold out data (usually 20%)
- The error rate on new cases is called the test error or generalization error (out-of-sample error)
- Low training error in combination with a high test error is a sign of overfitting

Source: Géron, A. (2019)

# Closer look at **cross-validation (k=3)**

- While there are a number of variations, the most common cross-validation method is k-fold cross-validation
- The data are randomly partitioned into k sets of roughly equal size (called the "**folds**").

**Step 1**: assign each observation to one of 3 folds



| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 |

| | | | | | |
|---|---|---|---|---|---|
| 🟩 1 | 🟧 2 | 🟩 3 | 🟧 4 | 🟦 5 | 🟦 6 |
| 🟧 7 | 🟦 8 | 🟦 9 | 🟧 10 | 🟧 11 | 🟩 12 |
| 🟦 13 | 🟩 14 | 🟩 15 | 🟦 16 | 🟩 17 | 🟧 18 |
| 🟩 19 | 🟦 20 | 🟩 21 | 🟧 22 | 🟧 23 | 🟩 24 |
| 🟧 25 | 🟦 26 | 🟧 27 | 🟦 28 | 🟦 29 | 🟩 30 |

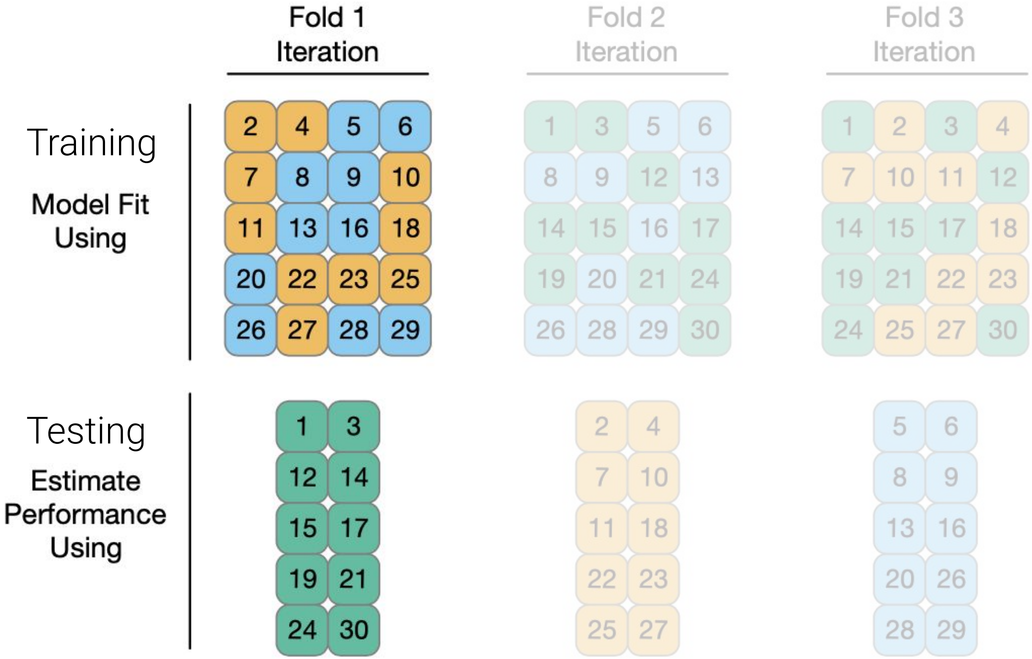
For illustration, k = 3 is shown for a data set of thirty training set points with random fold allocations.

Source: Kuhn & Silge (2020) https://www.tmwr.org/resampling.html

# Closer look at **cross-validation (k=3)**

- For each iteration, one fold is held out for assessment statistics and the remaining folds are substrate for the model.
- This process continues for each fold so that three models produce three sets of performance statistics.

**Step 2**: split the folds into training and test partitions



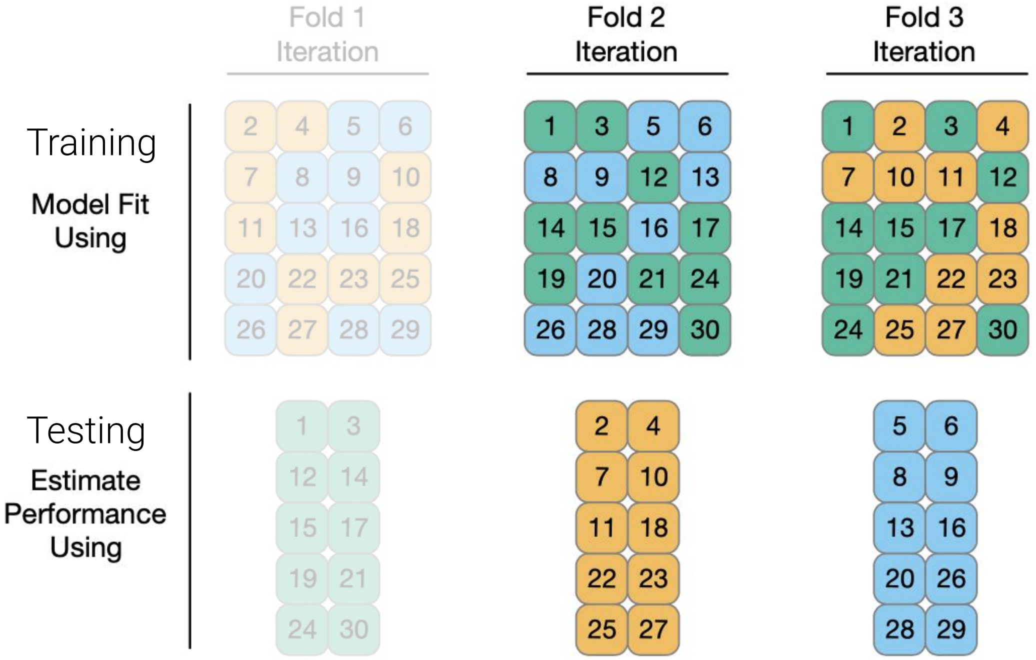
For 3-fold cross-validation, the three iterations of resampling are illustrated above.

Source: Kuhn & Silge (2020) https://www.tmwr.org/resampling.html

# Closer look at **cross-validation (k=3)**

- For each iteration, one fold is held out for assessment statistics and the remaining folds are substrate for the model.
- This process continues for each fold so that three models produce three sets of performance statistics.

**Step 2**: split the folds into training and test partitions



|  | Fold 1 Iteration | Fold 2 Iteration | Fold 3 Iteration |
|---|---|---|---|
| Training<br>Model Fit Using | 2 4 5 6<br>7 8 9 10<br>11 13 16 18<br>20 22 23 25<br>26 27 28 29 | 1 3 5 6<br>8 9 12 13<br>14 15 16 17<br>19 20 21 24<br>26 28 29 30 | 1 2 3 4<br>7 10 11 12<br>14 15 17 18<br>19 21 22 23<br>24 25 27 30 |
| Testing<br>Estimate Performance Using | 1 3<br>12 14<br>15 17<br>19 21<br>24 30 | 2 4<br>7 10<br>11 18<br>22 23<br>25 27 | 5 6<br>8 9<br>13 16<br>20 26<br>28 29 |

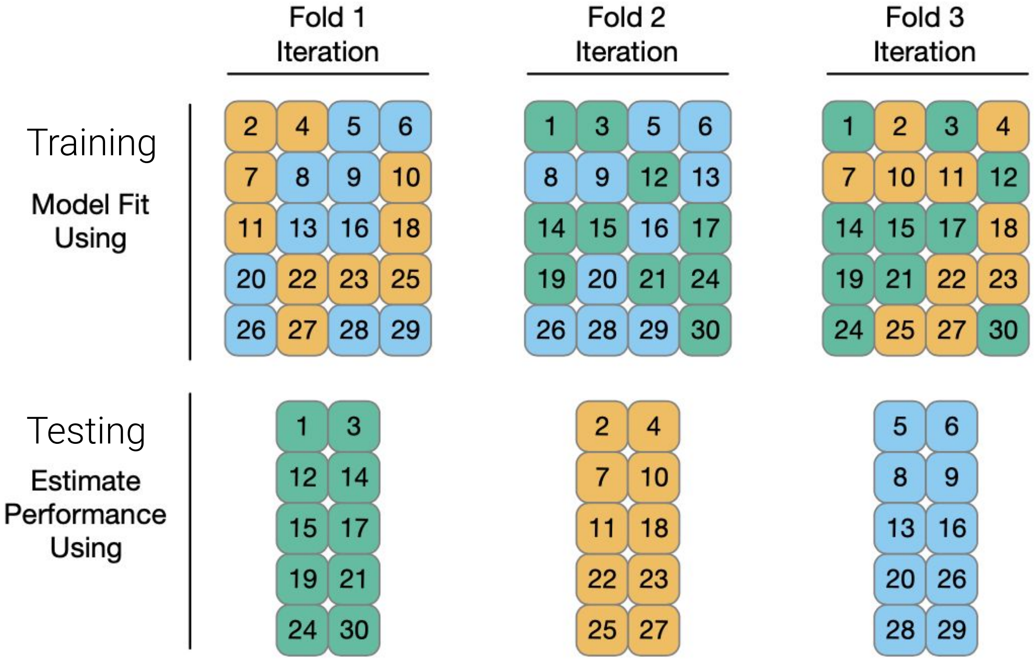
For 3-fold cross-validation, the three iterations of resampling are illustrated above.

Source: Kuhn & Silge (2020) https://www.tmwr.org/resampling.html

# Closer look at **cross-validation (k=3)**

- For each iteration, one fold is held out for assessment statistics and the remaining folds are substrate for the model.
- This process continues for each fold so that three models produce three sets of performance statistics.

**Step 2**: split the folds into training and test partitions



| | Fold 1 Iteration | Fold 2 Iteration | Fold 3 Iteration |
|---|---|---|---|
| Training<br>Model Fit Using | 2 4 5 6<br>7 8 9 10<br>11 13 16 18<br>20 22 23 25<br>26 27 28 29 | 1 3 5 6<br>8 9 12 13<br>14 15 16 17<br>19 20 21 24<br>26 28 29 30 | 1 2 3 4<br>7 10 11 12<br>14 15 17 18<br>19 21 22 23<br>24 25 27 30 |
| Testing<br>Estimate Performance Using | 1 3<br>12 14<br>15 17<br>19 21<br>24 30 | 2 4<br>7 10<br>11 18<br>22 23<br>25 27 | 5 6<br>8 9<br>13 16<br>20 26<br>28 29 |

For 3-fold cross-validation, the three iterations of resampling are illustrated above.

Source: Kuhn & Silge (2020) https://www.tmwr.org/resampling.html

# **k**-Fold-Cross-Validation

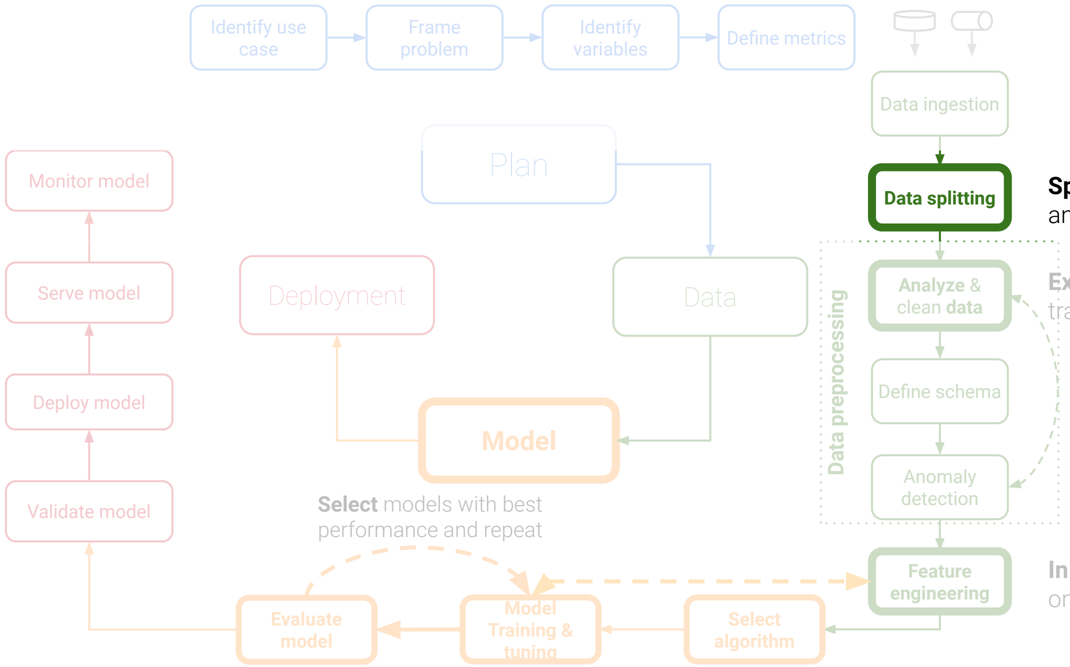
- Using k = 3 is a good choice to illustrate cross-validation but is a poor choice in practice.
- Values of **k** are most often **5** or **10**.
- We generally prefer 10-fold cross-validation as a default.

Source: Kuhn & Silge (2020) https://www.tmwr.org/resampling.html

# Procedure

# **Data Science Lifecycle** with focus on **modeling**

Plan | Data | **Model** | Deployment



**Split data** into training and test set

**Exploratory data analysis** on training set

**Select** models with best performance and repeat

**Initial feature engineering** on training set

**Evaluate final** best model on test set

**Evaluate** models on evaluation set

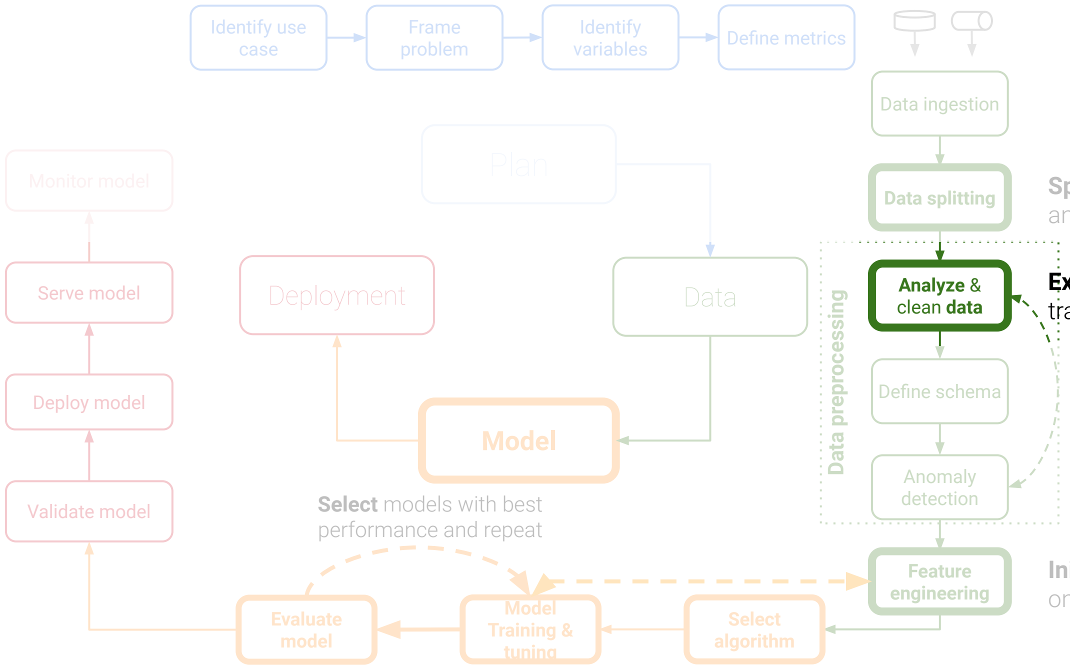
**Train** ( **tune**) models on evaluation set

**Select** a set of initial algorithms

Prof. Dr. Jan Kirenz

# **Data Science Lifecycle** with focus on **modeling**

Plan | Data | **Model** | Deployment



**Split data** into training and test set

**Exploratory data analysis** on training set

**Initial feature engineering** on training set

**Select** models with best performance and repeat

**Evaluate final** best model on test set

**Evaluate** models on evaluation set

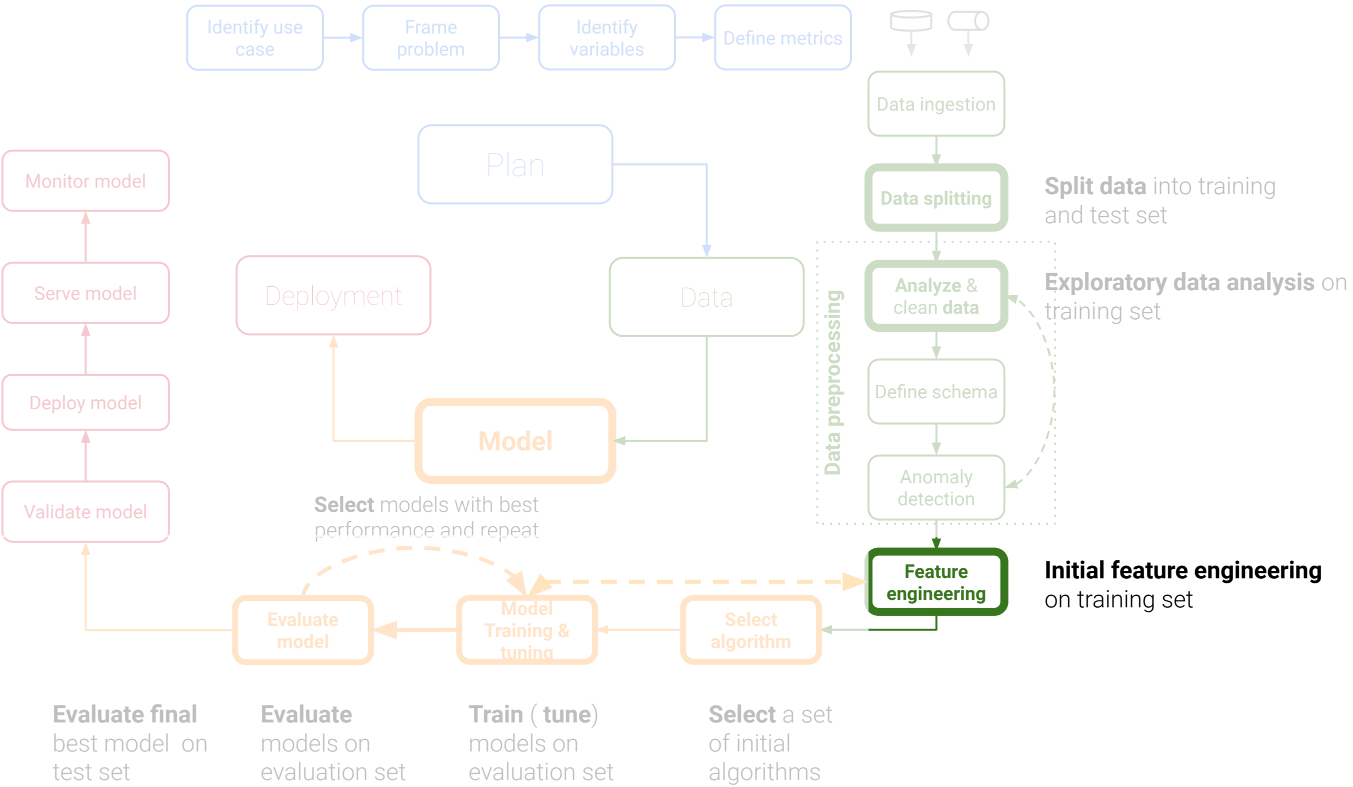
**Train** ( **tune**) models on evaluation set

**Select** a set of initial algorithms

Prof. Dr. Jan Kirenz

# **Data Science Lifecycle** with focus on **modeling**

Plan | Data | **Model** | Deployment

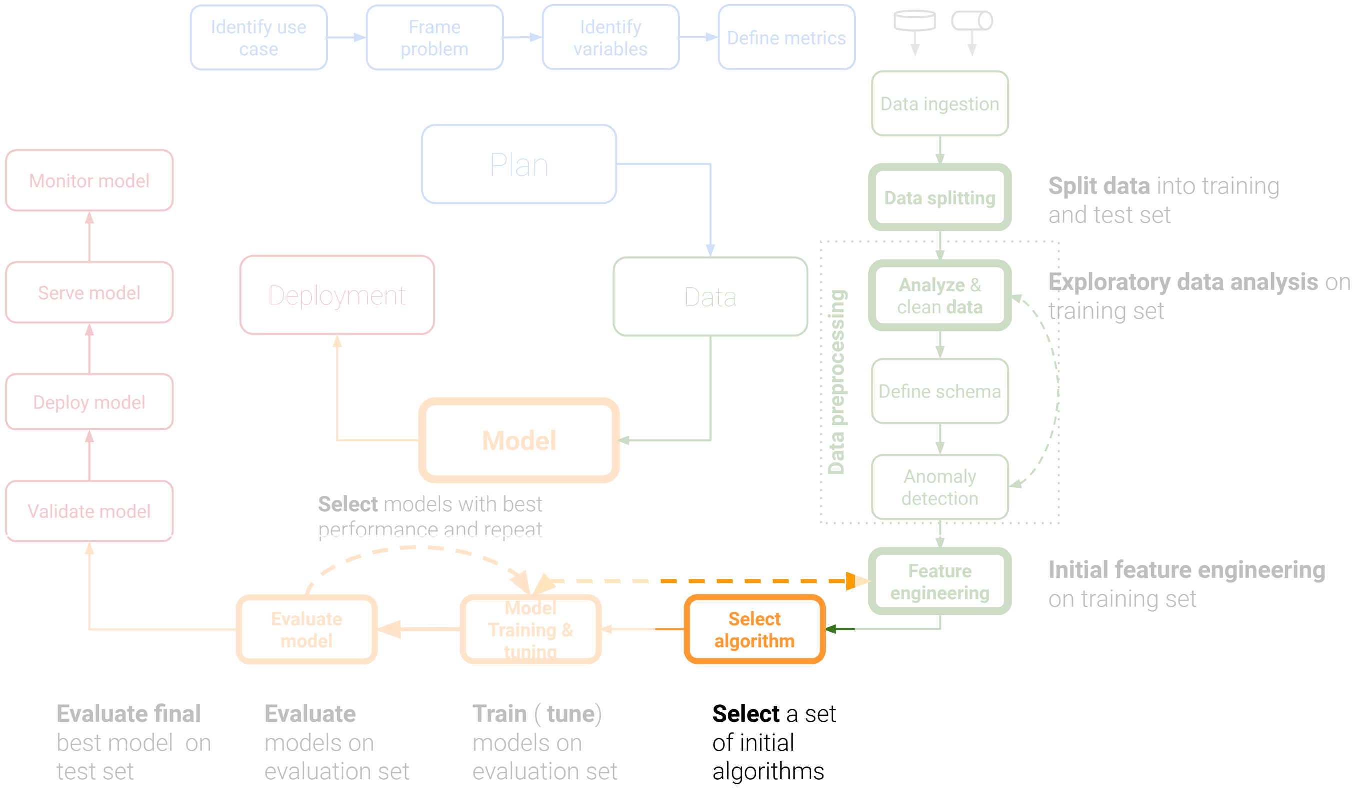


Identify use case
Frame problem
Identify variables
Define metrics
Data ingestion
Plan
Data splitting
Split data into training and test set
Monitor model
Serve model
Deployment
Data
Data preprocessing
Analyze & clean data
Exploratory data analysis on training set
Define schema
Deploy model
Model
Anomaly detection
Validate model
Select models with best performance and repeat
Feature engineering
Initial feature engineering on training set
Evaluate model
Model Training & tuning
Select algorithm
Evaluate final best model on test set
Evaluate models on evaluation set
Train ( tune) models on evaluation set
Select a set of initial algorithms

Prof. Dr. Jan Kirenz

# **Data Science Lifecycle** with focus on **modeling**

Plan | Data | **Model** | Deployment



**Split data** into training and test set

**Exploratory data analysis** on training set

**Select** models with best performance and repeat

**Initial feature engineering** on training set

**Evaluate final** best model on test set

**Evaluate** models on evaluation set

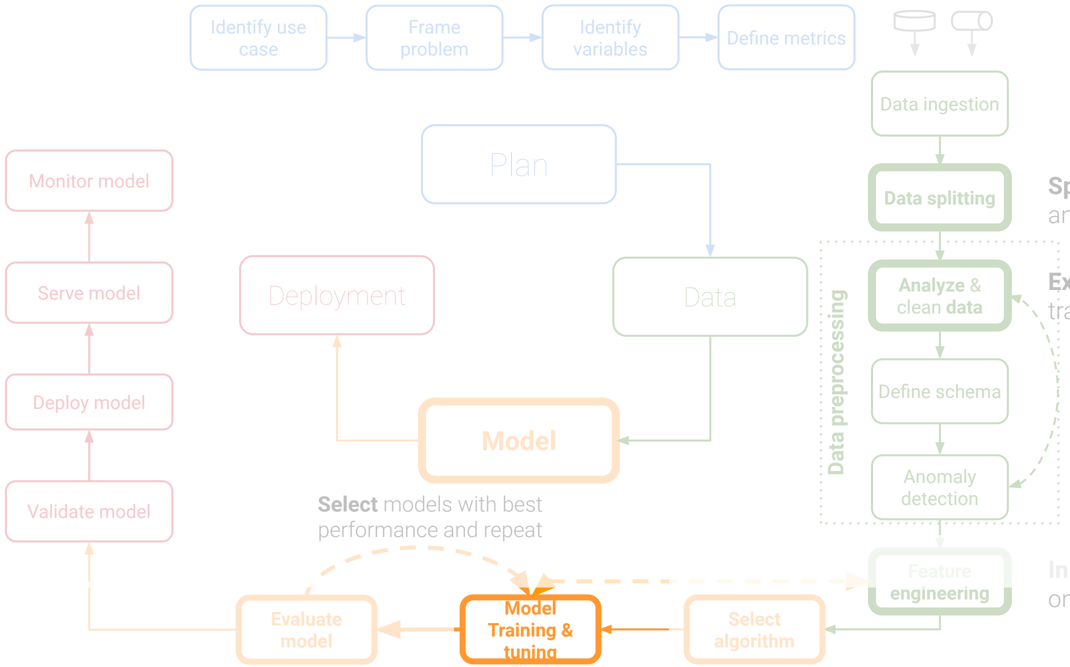
**Train** ( **tune**) models on evaluation set

**Select** a set of initial algorithms

Prof. Dr. Jan Kirenz

# **Data Science Lifecycle** with focus on **modeling**

Plan | Data | **Model** | Deployment



**Split data** into training and test set

**Exploratory data analysis** on training set

**Select** models with best performance and repeat

**Initial feature engineering** on training set

**Evaluate final** best model on test set

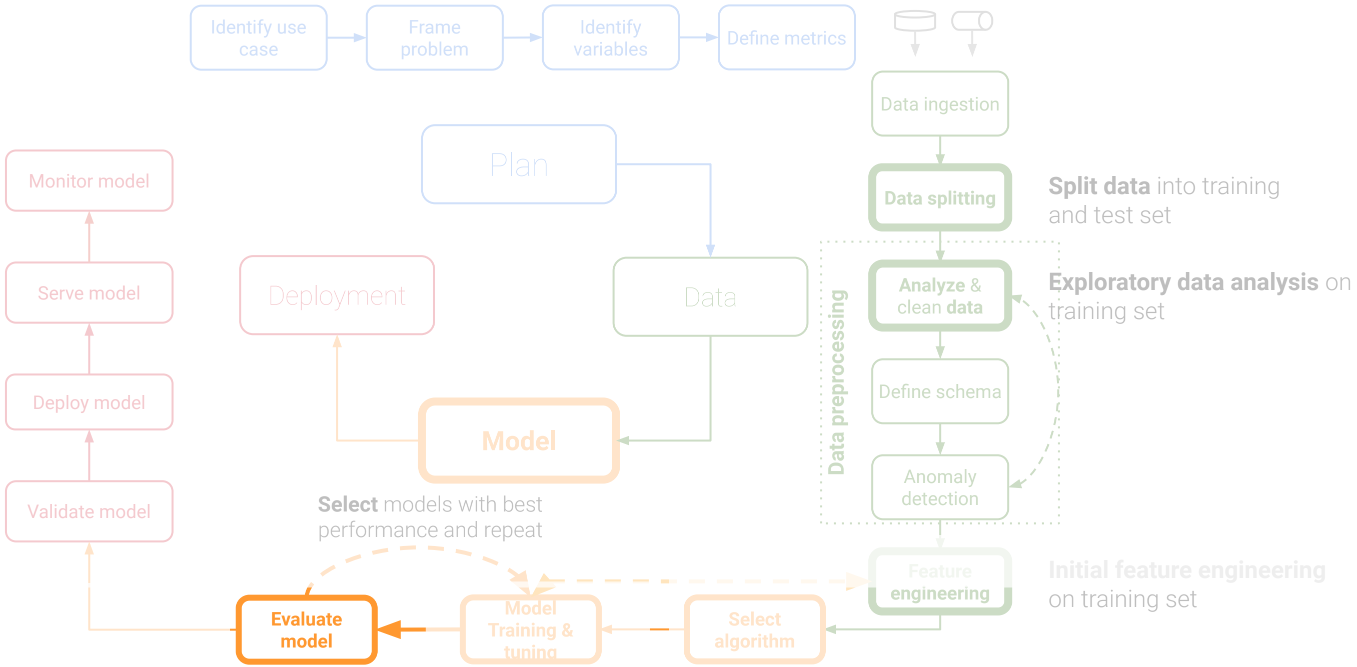
**Evaluate** models on evaluation set

**Train** ( tune) models on evaluation set

**Select** a set of initial algorithms

# **Data Science Lifecycle** with focus on **modeling**

Plan | Data | **Model** | Deployment



**Split data** into training and test set

**Exploratory data analysis** on training set

**Select** models with best performance and repeat

**Initial feature engineering** on training set

**Evaluate final** best model on test set

**Evaluate** models on evaluation set

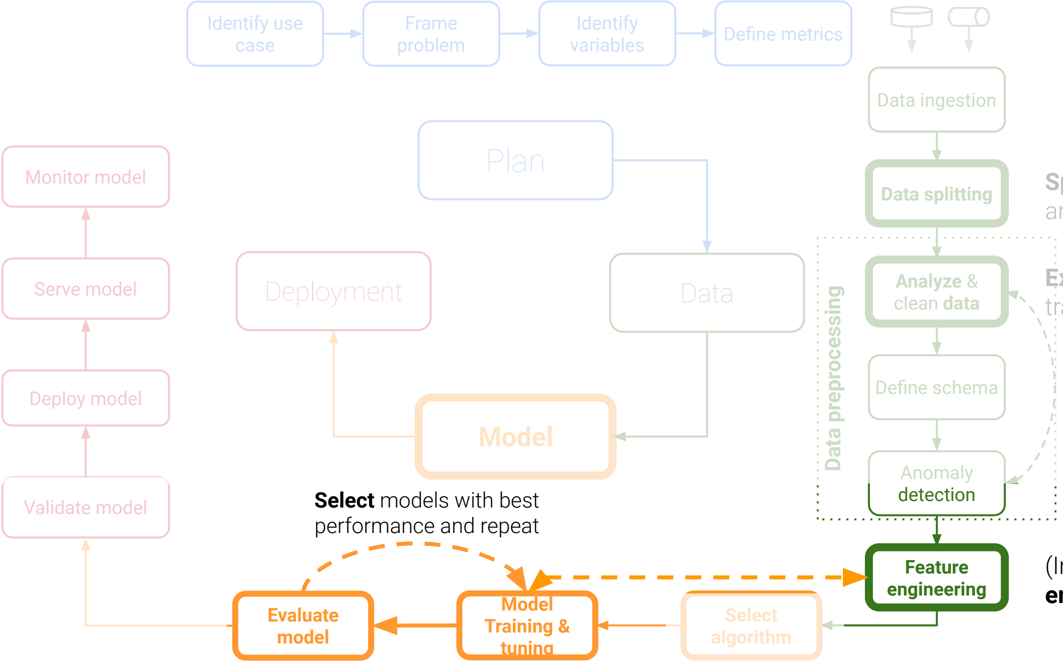
**Train** ( tune) models on evaluation set

**Select** a set of initial algorithms

Prof. Dr. Jan Kirenz

# **Data Science Lifecycle** with focus on **modeling**

Plan | Data | **Model** | Deployment



**Select** models with best performance and repeat

**Split data** into training and test set

**Exploratory data analysis** on training set

(Initial) **feature engineering** on training set

**Evaluate final** best model on test set

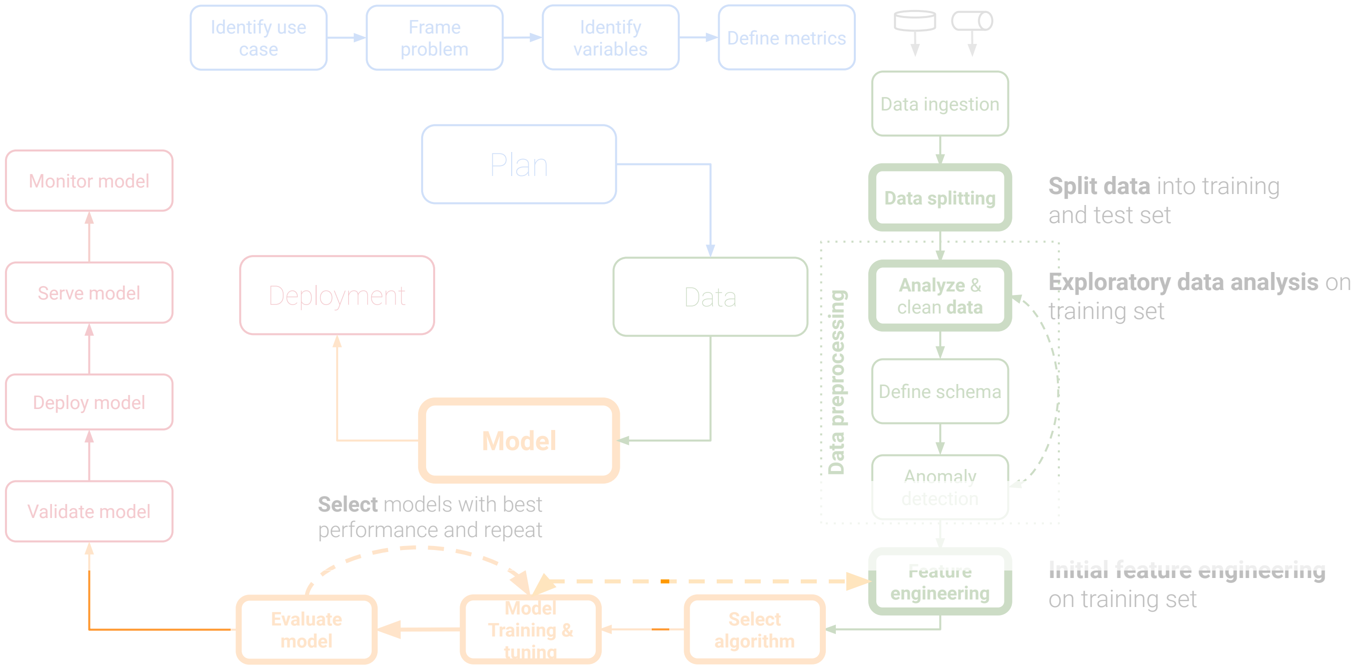
**Evaluate** models on evaluation set

**Train** ( **tune**) models on evaluation set

**Select** a set of initial algorithms

# **Data Science Lifecycle** with focus on **modeling**

Plan | Data | **Model** | Deployment



**Split data** into training and test set

**Exploratory data analysis** on training set

**Select** models with best performance and repeat

**Initial feature engineering** on training set

**Evaluate final** best model on test set

**Evaluate** models on evaluation set

**Train** ( **tune**) models on evaluation set

**Select** a set of initial algorithms

Prof. Dr. Jan Kirenz

# Modeling procedere in summary

1. **Split data** into training and test set
2. Create **evaluation set** from training data (cross-validation)
3. **Train** a set of initial models using cross-validation (without extensive tuning)
4. **Select** the models with the best performance on the validation set
5. **Optimize** your models (hyperparameter tuning) on the validation set
6. Train the best model one more time on the **full training set**
7. Evaluate the final model on the **test set**
8. Do not further improve the model (this is your final result!)

Source: Géron (2019)

# Literature

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

Kuhn, M., & Silge, J. (2020). Tidy Modeling with R.