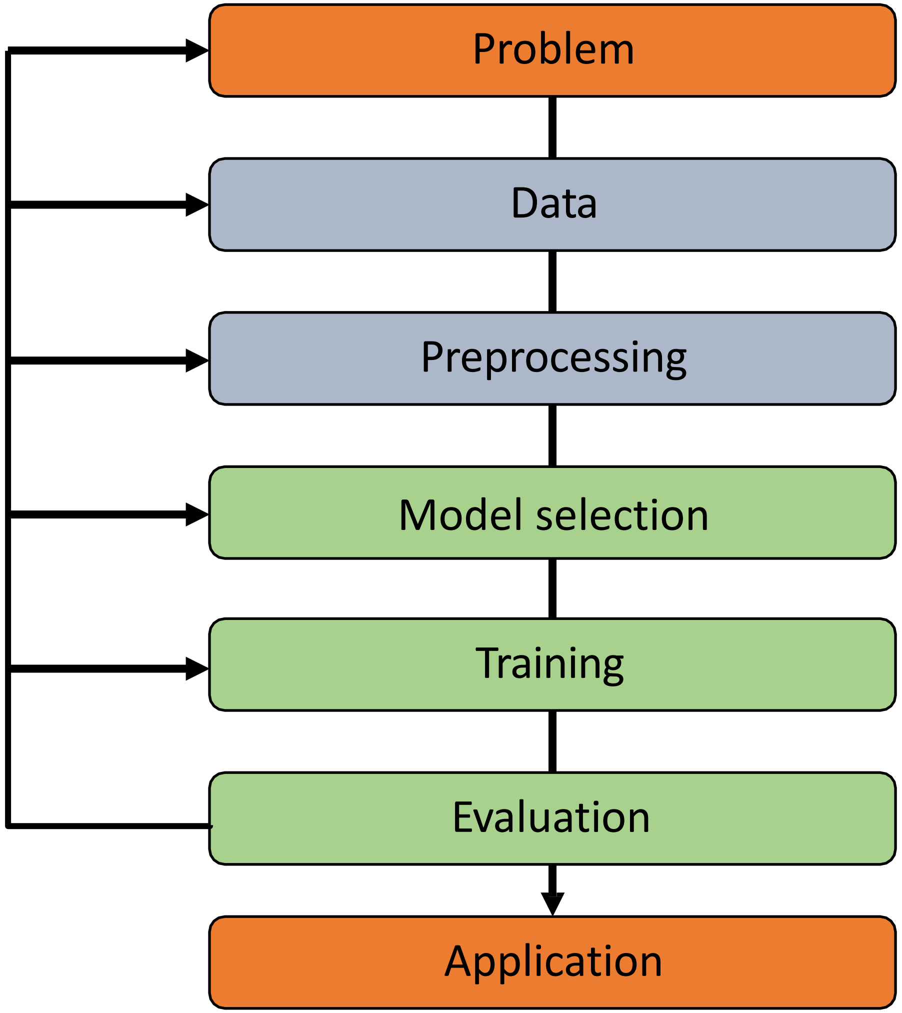
Introduction to Machine Learning

Machine Learning Workflow

Machine Learning Workflow



What problem do we want to solve?
What do we want to predict?

What information do we need for this?
How much data do we need?
How do we get this data?

How do we get this data into the right form for our ML model?

Which algorithm / method is suitable for our data?

How do we make the algorithm learn using our data?

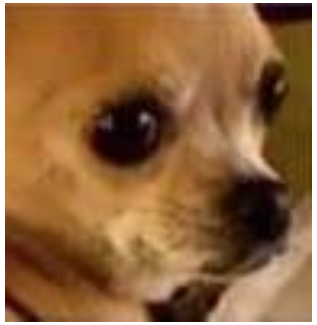
How do we assess whether our model/predictions are adequate?

Machine Learning Workflow

Problem

What problem do we want to solve?
What do we want to predict?

Our algorithm should be able to distinguish dogs from muffins



vs

Machine Learning Workflow

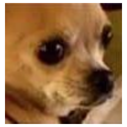
Problem

Data

Feature Engineering

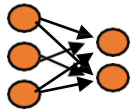
What information in the image, and in what form, can we use to train our algorithm?

Picture



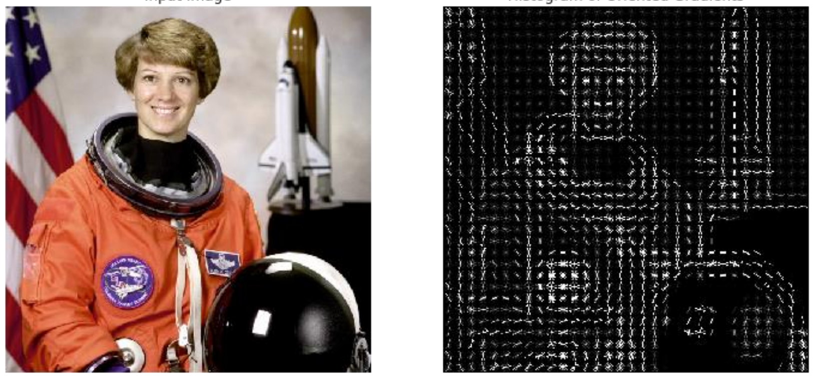
Feature engineering

Algorithm



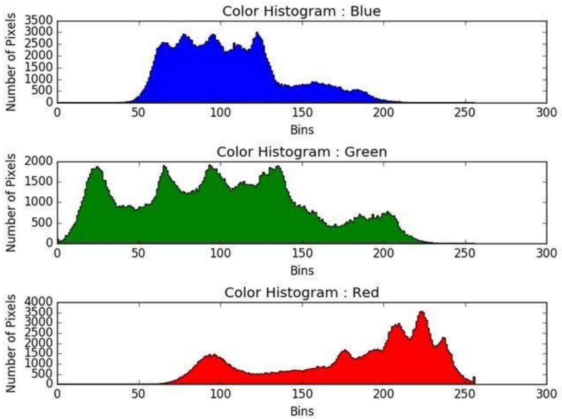
e.g.

HOG (Histogram of Oriented Gradients)



https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

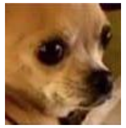
Color Histogram



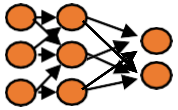
J. -S. Li, I. -H. Liu, C. -J. Tsai, Z. -Y. Su, C. -F. Li and C. -G. Liu, "Secure Content-Based Image Retrieval in the Cloud With Key Confidentiality," in IEEE Access, vol. 8, 2020

Spoiler: In Deep Learning, we skip feature engineering!

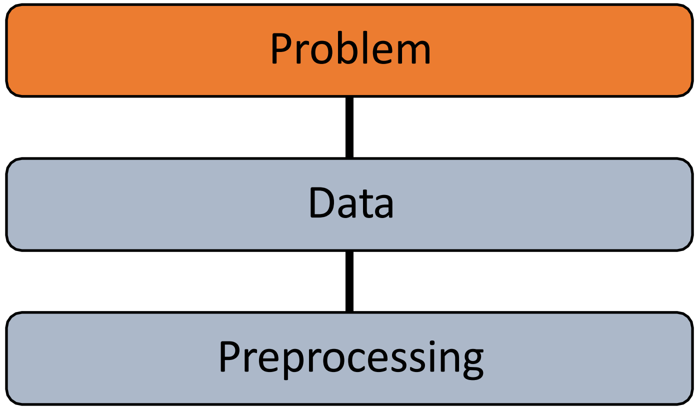
Picture



Neural Net



Machine Learning Workflow



How do we get this data into the right form for our ML model?

Each column contains a feature

Class labels

Each line contains information about **one** image

HOG_Pixel_1	HOG_Pixel_2	…	Red_Bin1	Red_Bin2	…
…	…	…	…	…	…
…	…	…	…	…	…
…	…	…	…	…	…
…	…	…	…	…	…

Label
Dog
Muffin
Muffin
Dog

PREPROCESSING

Problem

Data

Preprocessing

Scaling / Standardization

Problem: Features are often available in a wide variety of forms e.g. scales from 1 to 10 vs. from - ∞ to + ∞.

However: Many algorithms require standardized feature vectors

Features

HOG_Pixel_1	HOG_Pixel_2	…	Red_Bin1	Red_Bin2	…
-1	…	…	400	389	…
1	…	…	3765	3400	…
.5	…	…	520	502	…
-.3	…	…	27	16	…

Class Labels

Label
Dog
Muffin
Muffin
Dog

Each line contains information about **one** image

PREPROCESSING

Problem

Data

Preprocessing

Dimensionality Reduction

- Often we have a lot of features
- Many features contain redundant information

Features

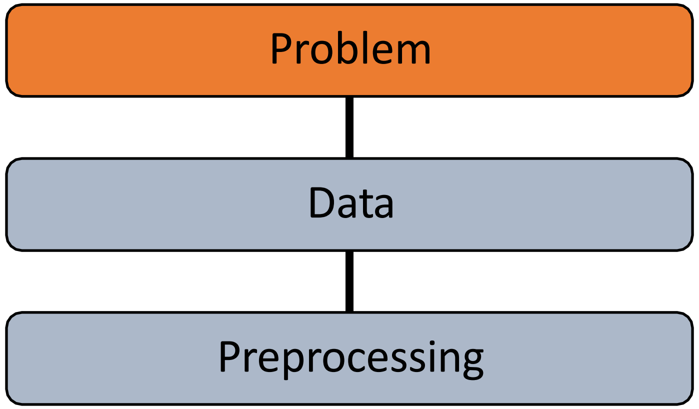
HOG_Pixel_1	HOG_Pixel_2	…	Red_Bin1	Red_Bin2	…
-1	…	…	- .45	…	…
1	…	…	.97	…	…
.5	…	…	-.47	…	…
-.3	…	…	-.99	…	…

Class Labels

Label
Dog
Muffin
Muffin
Dog

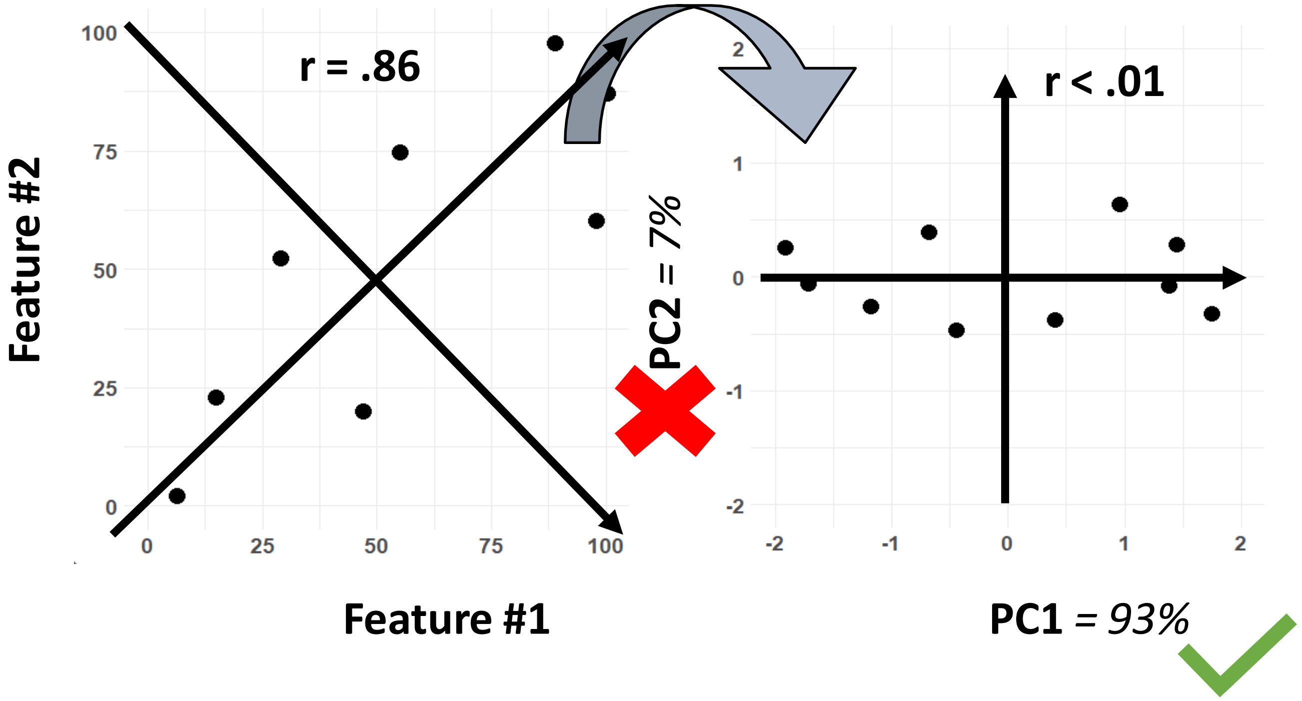
Each line contains information about **one** image

PREPROCESSING



Dimensionality Reduction

Principle Component Analysis



Machine Learning Workflow

Problem

Data

Preprocessing

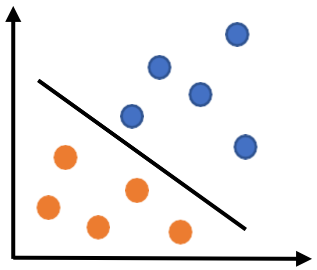
Model selection

Supervised Learning

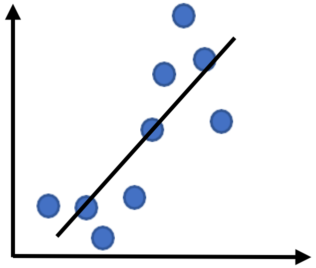
→ We know the labels of our data

We know the group membership of our data and want the algorithm to learn this membership

Classification



Regression



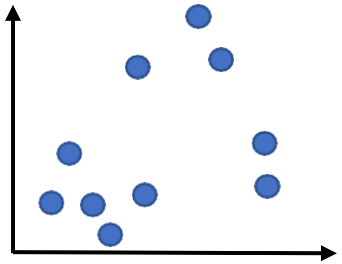
We know the true values of the variable to be predicted in our data and want the algorithm to learn to predict them

Unsupervised Learning

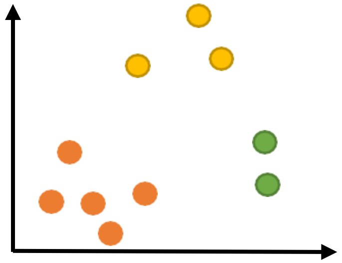
→ We don’t know the labels of our data

We do not know the group membership of our data and would like the algorithm to independently find clusters in our data

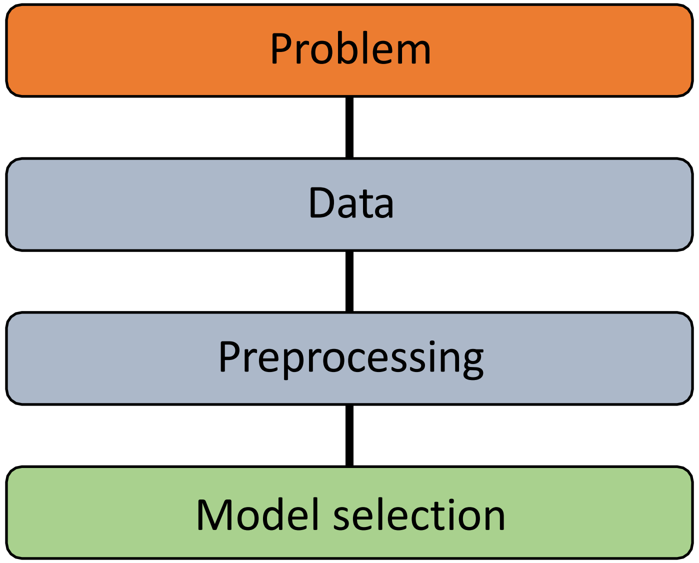
Unlabeled data



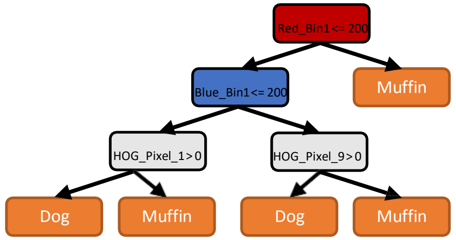
Clustered data



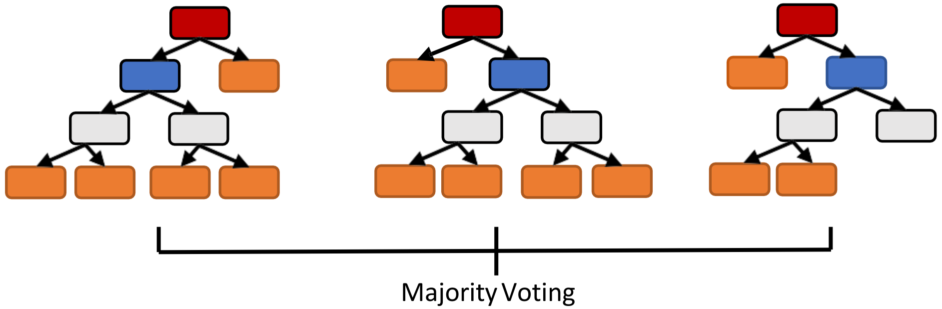
Machine Learning Workflow



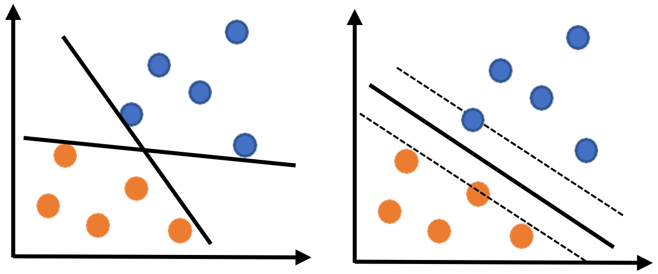
Decision Tree



Random Forest



Support Vector Machines (SVM)



And so on…

Machine Learning Workflow

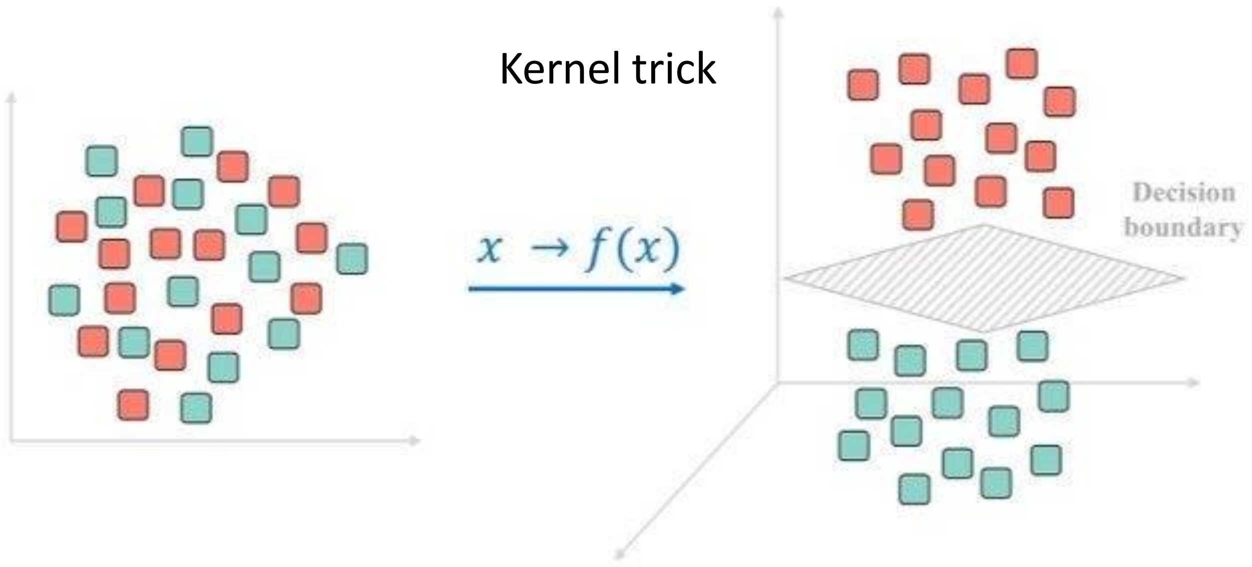
Problem

Data

Preprocessing

Model selection

Support Vector Machines



Sheykhmousa, M., Mahdianpari, M., Ghanbari, H., Mohammadimanesh, F., Ghamisi, P., & Homayouni, S. (2020). Support vector machine versus random forest for remote sensing image classification: A meta-analysis and systematic review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *13*, 6308-6325.

Machine Learning Workflow

Problem

Data

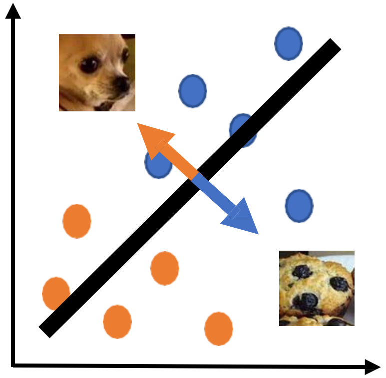
Preprocessing

Model selection

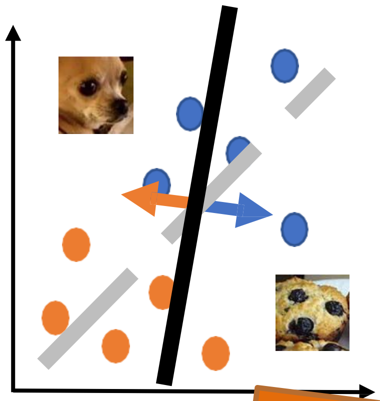
Training

How does learning work?

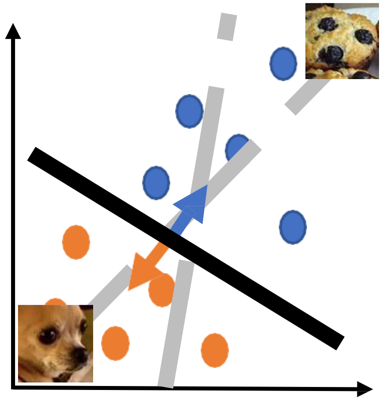
1. We initialize our SVM with a random hyperplane



2. After each learning phase the algorithm adjusts the hyperplane…



3. …until it has found the optimal hyperplane



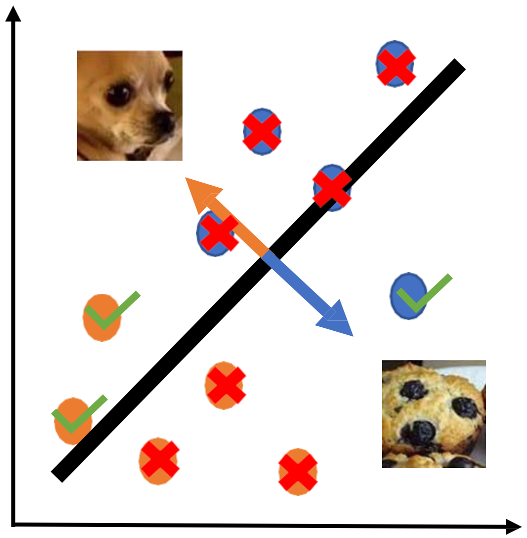
But how does the algorithm do it? How does it find the right hyperplane?

Machine Learning Workflow

- Problem
- Data
- Preprocessing
- Model selection
- Training

Back to the beginning:

1. We initialize our SVM with a random hyperplane



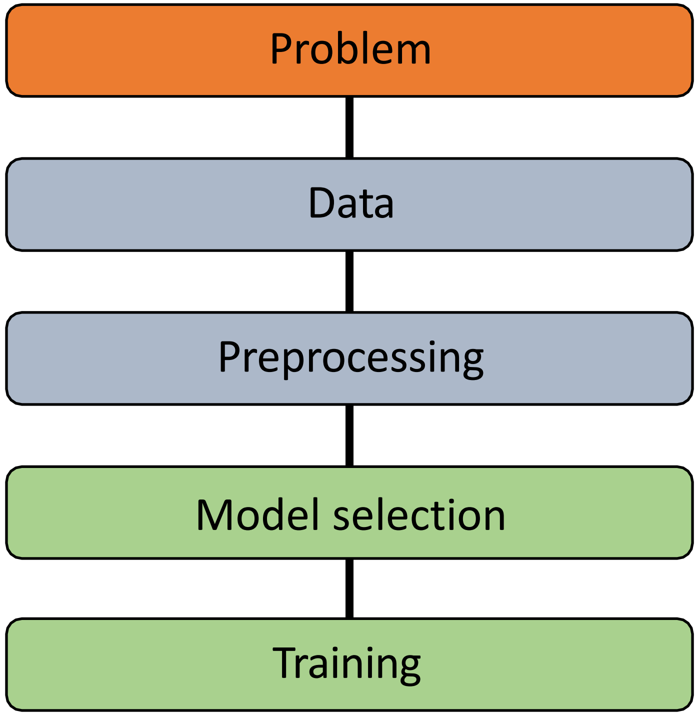
Idea:
Algorith should learn how to distinguish right and wrong classifications!

How do we quantify good or bad predictions of the classifier?

→ **True class sign**

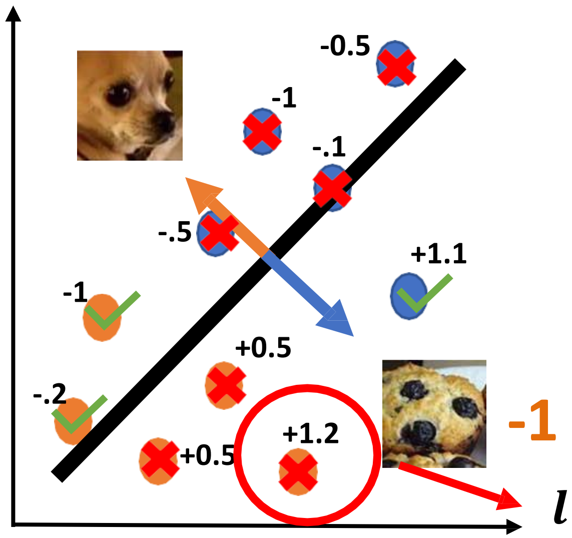
→ **Distance from hyperplane**

Machine Learning Workflow



Back to the beginning:

1. We initialize our SVM with a random hyperplane



This point is on the **wrong side** and **far from** the dividing line. That is why its loss function is **large**.

We need a mathematical function with will measure our "satisfaction" with the classification of the algorithm.

→ **Loss function**

$$l(y) = max(0; 1 - t * y)$$

y = Prediction (distance)

t = true class sign: -1 or +1

$$l(1.2) = max(0; 1 - (-1) * 1.2)$$

$$l(1.2) = max(0; 2.2)$$

$$l(1.2) = 2.2$$

Machine Learning Workflow

Problem

Data

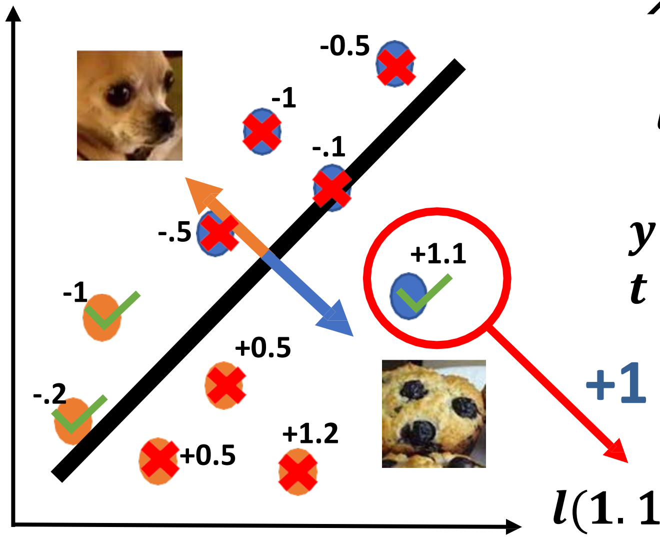
Preprocessing

Model selection

Training

Back to the beginning:

1. We initialize our SVM with a random hyperplane



This point is on **the right side** and **far from** the dividing line. Therefore, its loss is zero, that is, **minimal**.

We need a mathematical function with will measure our "satisfaction" with the classification of the algorithm.

→ **Loss function**

$$l(y) = max(0; 1 - t * y)$$

y = Prediction (distance)

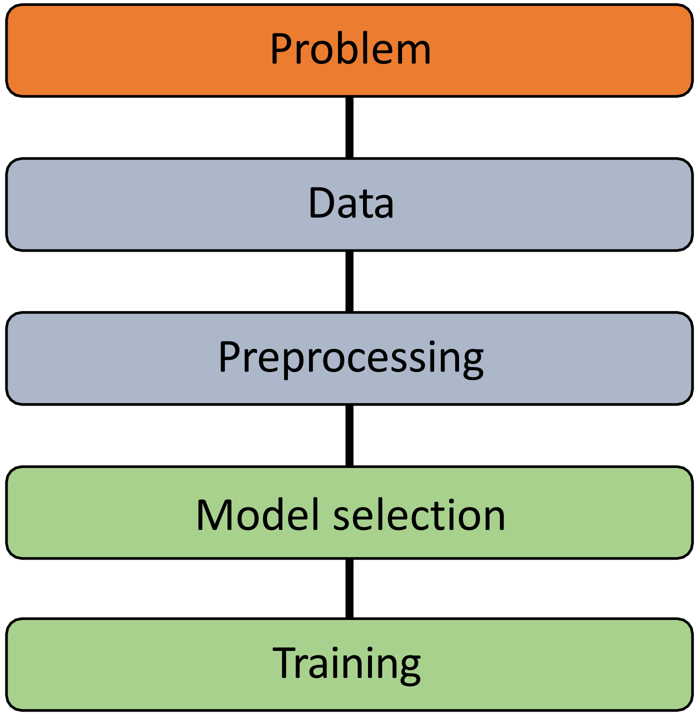
t = true class sign: -1 or +1

$$l(1.1) = max(0; 1 - (+1) * 1.1)$$

$$l(1.1) = max(0; -0.1)$$

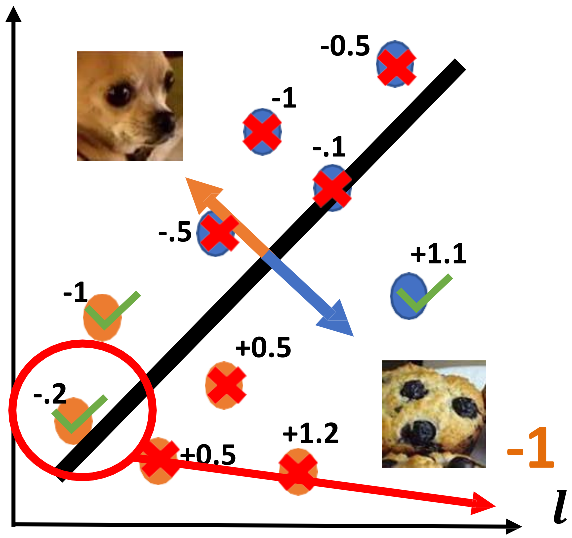
$$l(1.1) = 0$$

Machine Learning Workflow



Back to the beginning:

1. We initialize our SVM with a random hyperplane



This point is on **the right side**, but **very close** to the dividing line. Therefore, its loss is **low, but not minimal.**

We need a mathematical function with will measure our "satisfaction" with the classification of the algorithm.

→ **Loss function**

$$\boldsymbol{l(y) = max(0; 1 - t * y)}$$

y = Prediction (distance)

t = true class sign: -1 or +1

$$\boldsymbol{l(-.2) = max(0; 1 - (-1) * -0.2)}$$

$$\boldsymbol{l(-.2) = max(0; 0.8)}$$

$$\boldsymbol{l(-.2) = 0.8}$$

Machine Learning Workflow

Problem

Data

Preprocessing

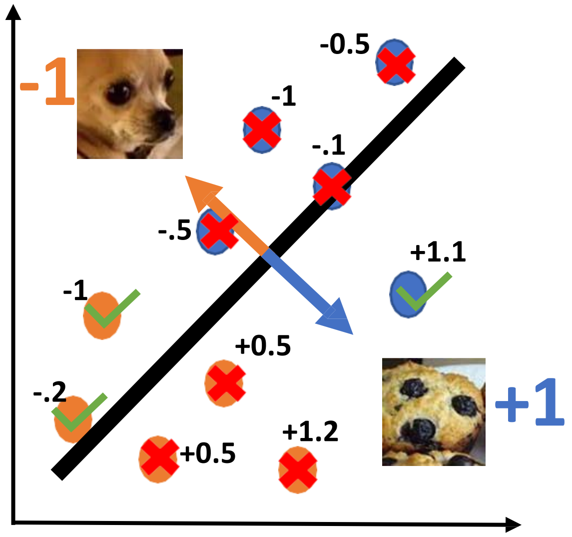
Model selection

Training

Idea:
We look for a rotated hyperplane that minimizes our loss (optimization)!

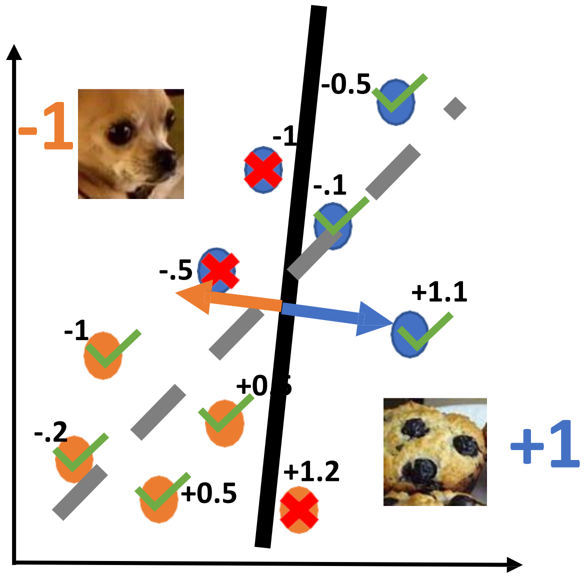
1. We initialize our SVM with a random hyperplane

2. After each le… …se the algorithm adjusts the hyperplane…



Mean loss after 1. rotation:

$l = 1.21$

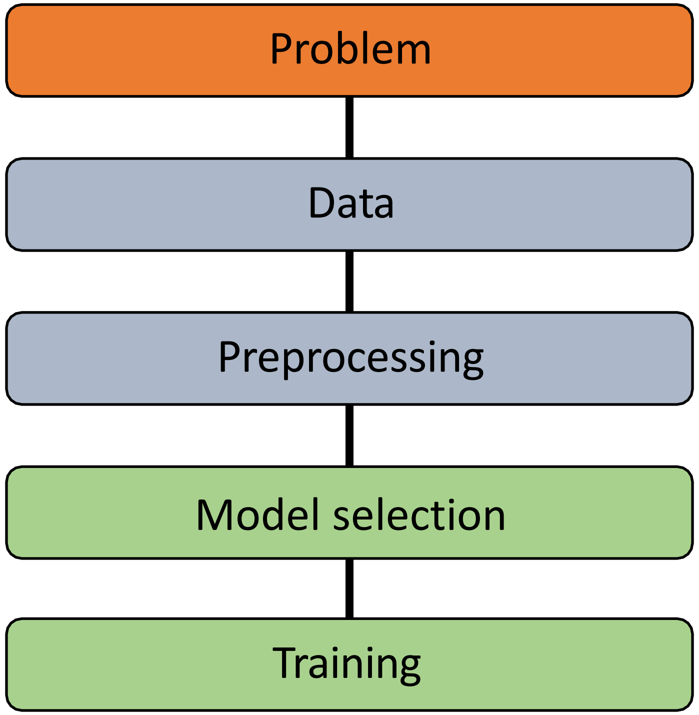


Mean loss after 2. rotation:

$l = 0.98$

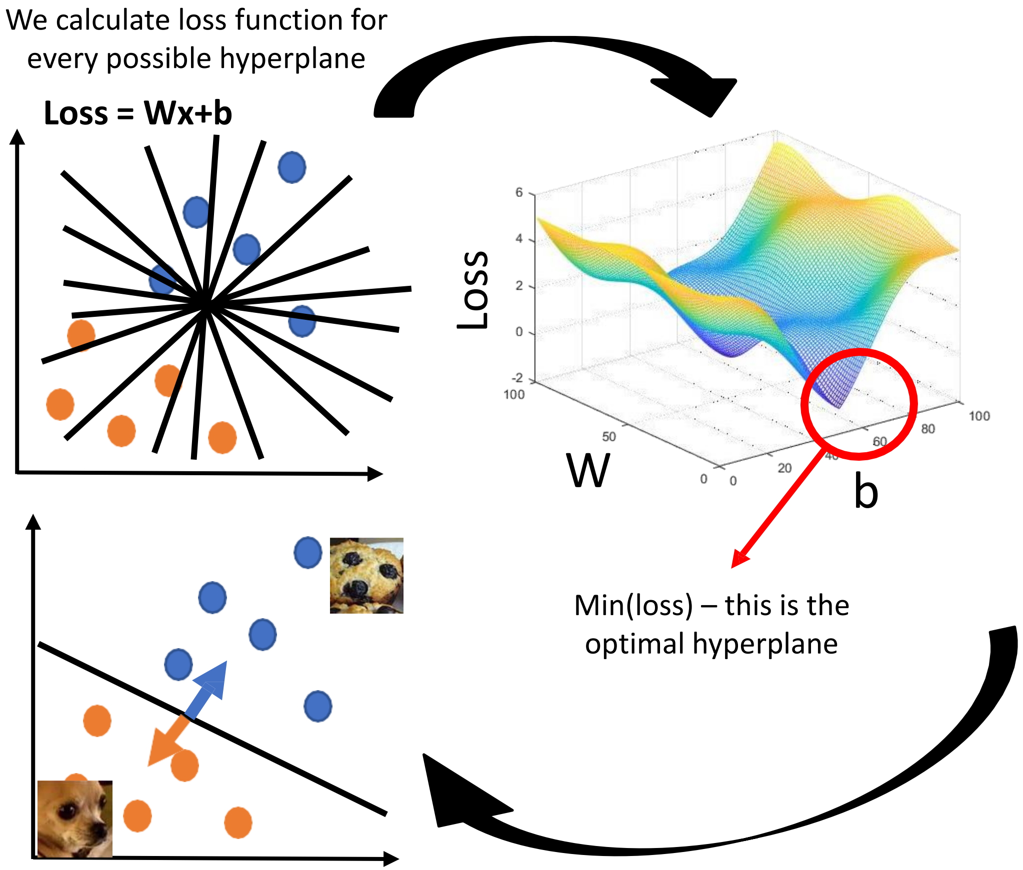
The rotation of the hyperplane has reduced our loss!

Machine Learning Workflow

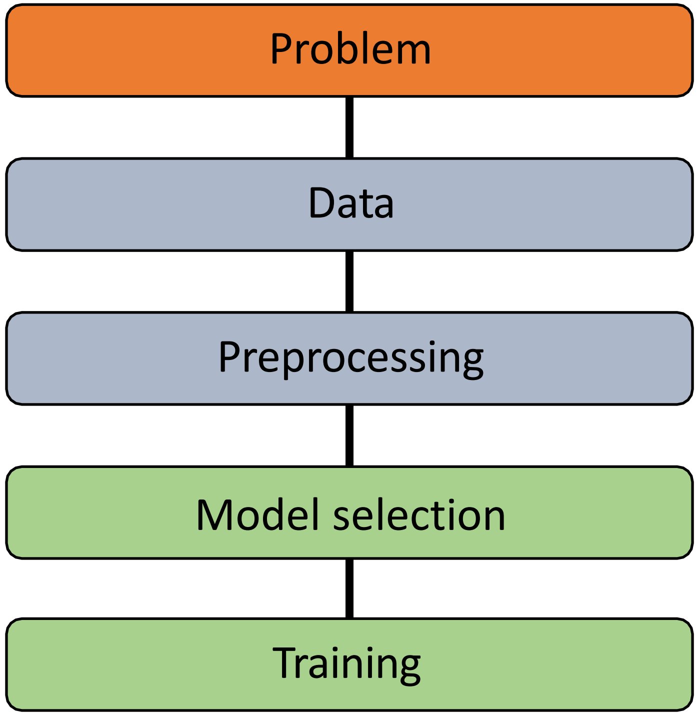


Exhaustive Search

We try all possible hyperplanes and calculate loss. Then we choose the hyperplane that minimizes loss function.

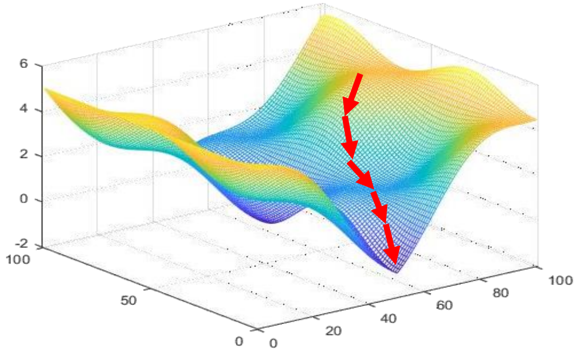


Machine Learning Workflow



Gradient Descent

Iterative optimization method, with which we search for the minimum of the loss function. After each learning epoch we choose the next step (i.e. parameter combination) which leads us "steepest" downwards (instead of testing all combinations).

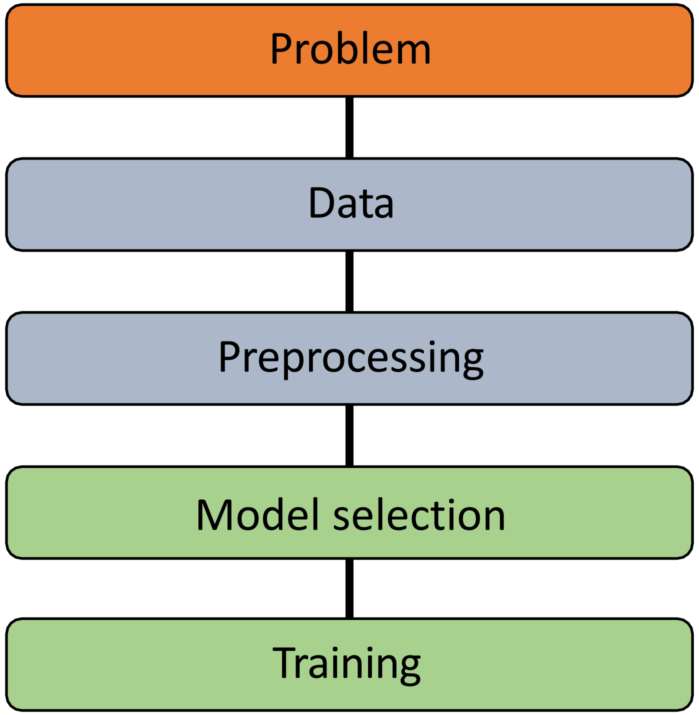


Metaphor

We are standing on a mountain, and suddenly heavy fog sets in. We would like to get back to the valley as quickly as possible, but we can only see a few meters away. Which way do we choose?

→ **We look at our feet and always choose the direction in which the terrain slope is steepest as our next step**

Machine Learning Workflow



We quantify the goodness of classification with a:

Loss function e.g. Hinge loss

We search for the optimal parameter combination of our algorithm (which minimizes the loss function), using a:

Optimization procedure e.g. Gradient Descent

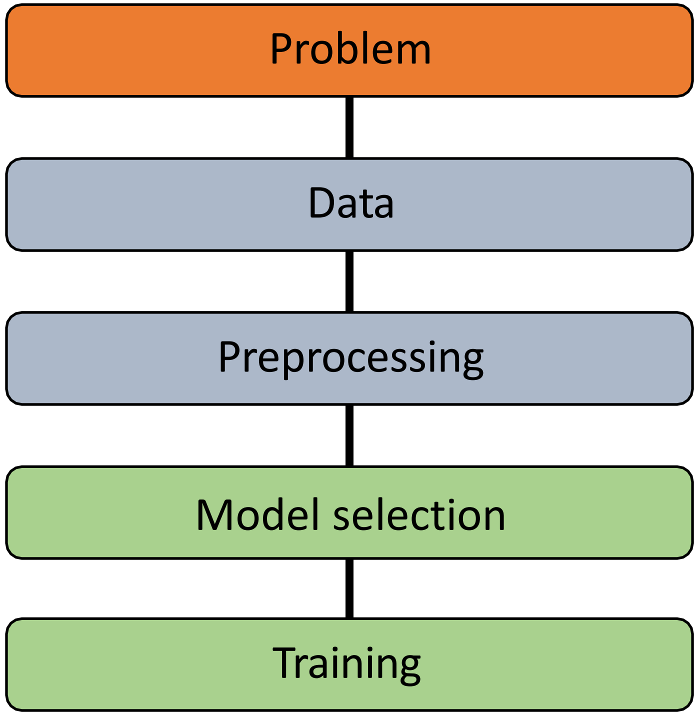
Each optimization step is thereby an:

Iteration (epoch)

The step size with which we change the parameters per iteration is the:

Lerning rate

Machine Learning Workflow



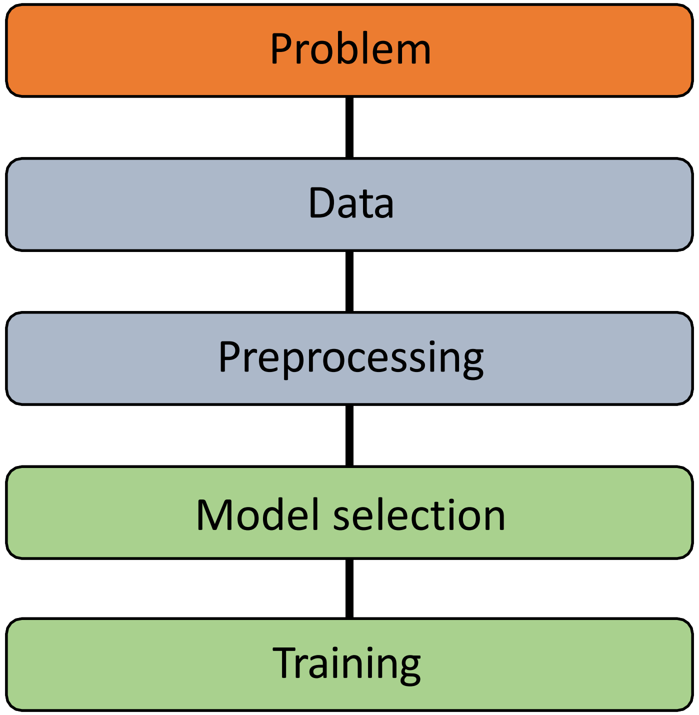
Gradient Descent Algorithm

```
epochs = 1000
x = random.choice(…) #starting location
learning_rate = 0.01

for i in range(epochs):
    gradient = derivative(x)
    x = x – learning_rate×gradient #updating rule
```

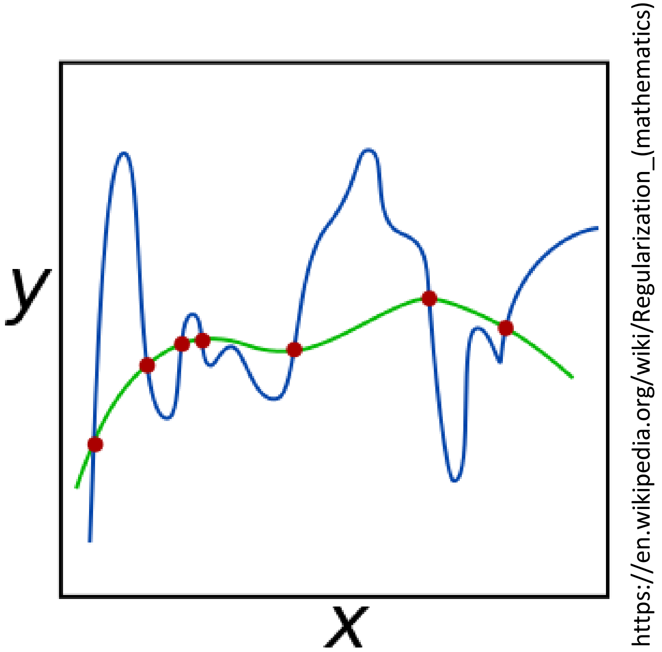
https://www.i-am.ai/gradient-descent.html

Overfitting



Regularization

Problem: The more complex a model (e.g. due to more parameters), the better it can adapt to the training data and minimize the error there.



Machine Learning Workflow

Problem

Data

Preprocessing

Model selection

Training

Regularization

Idea: We add a **regularization term** to our loss function, which penalizes the complexity of our model

$$L(f) + \boldsymbol{\lambda}\, \boldsymbol{R}(\boldsymbol{f})$$

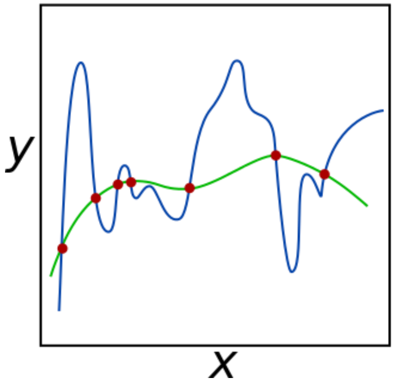
$\boldsymbol{\lambda}$ = Strength of regularization

e.g.

$L1 = \dots + \lambda \sum |\beta_j|$ — LASSO regression: "keep number of coefficients small„

$L2 = \dots + \lambda \sum {\beta_j}^2$ — RIDGE regression: "keep coefficients small"

Elastic net — Combination of L1 + L2

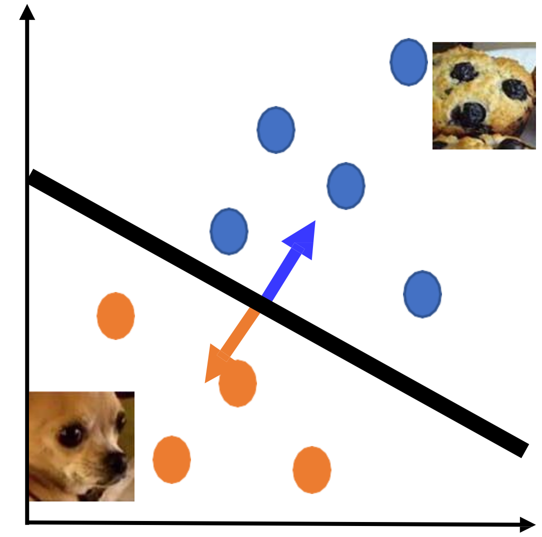


https://en.wikipedia.org/wiki/Regularization_(mathematics)

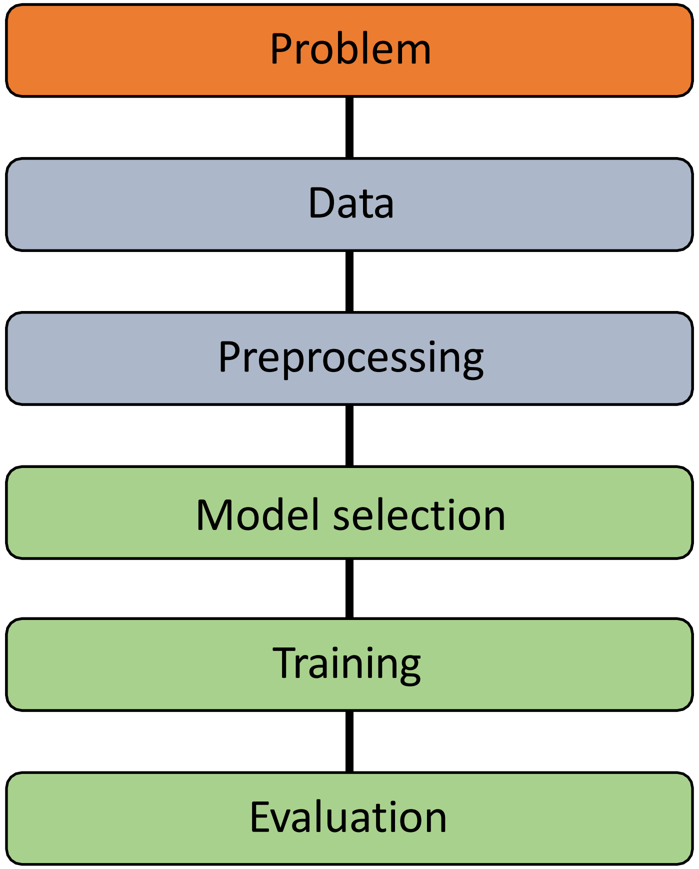
Machine Learning Workflow

- Problem
- Data
- Preprocessing
- Model selection
- Training
- Evaluation

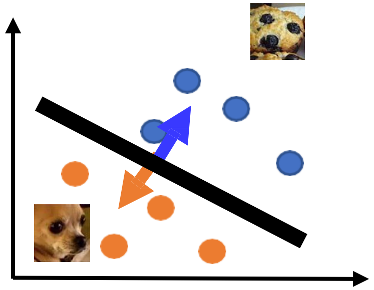
How do we assess whether our model/predictions are adequate?



Machine Learning Workflow



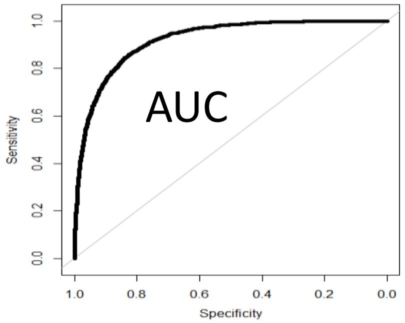
How do we assess whether our model/predictions are adequate?



Classification:

	Predicted	
Actual	True Positive (Hit)	False Negative (Miss)
	False Positive (False Alarm)	True Negative (Correct Rejection)

ROC



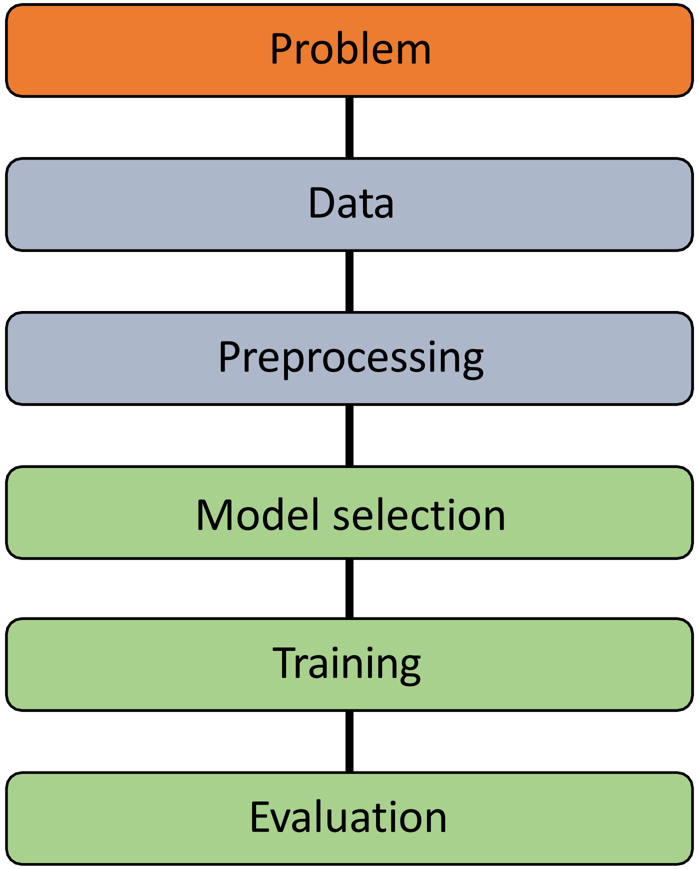
$$Accuracy = \frac{Hit+CR}{Hit+CR+Miss+FA}$$

$$Sensitivity = \frac{Hit}{Hit+Miss}$$

$$Precision = \frac{Hit}{Hit+FA}$$

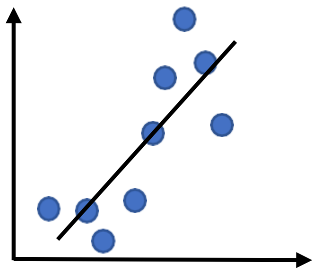
$$Specificity = \frac{CR}{CR+FA}$$

Machine Learning Workflow



How do we assess whether our model/predictions are adequate?

Regression



Regression:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2}$$

$$MSE = \frac{1}{N}\sum(y_i - \hat{y}_i)^2$$

Machine Learning Workflow

Problem

Data

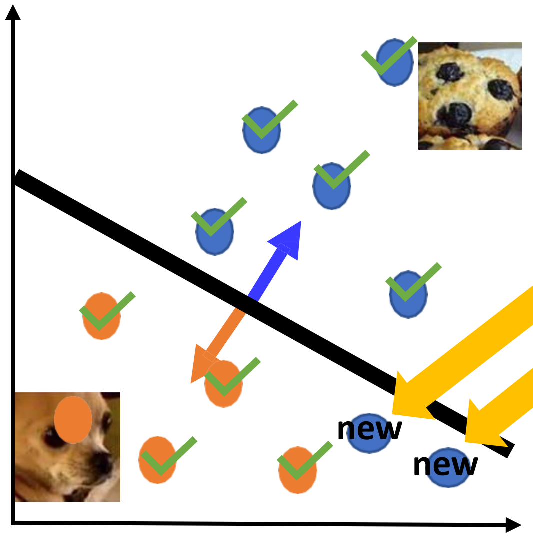
Preprocessing

Model selection

Training

Evaluation

How do we assess whether our model/predictions are adequate?



Danger of overfitting!
occurs when our model fits exactly against its training data, but fails to classify new observations

100% Accuracy!

Machine Learning W

Question:
What if we are dissatisfied with our model's performance?

Problem

Data

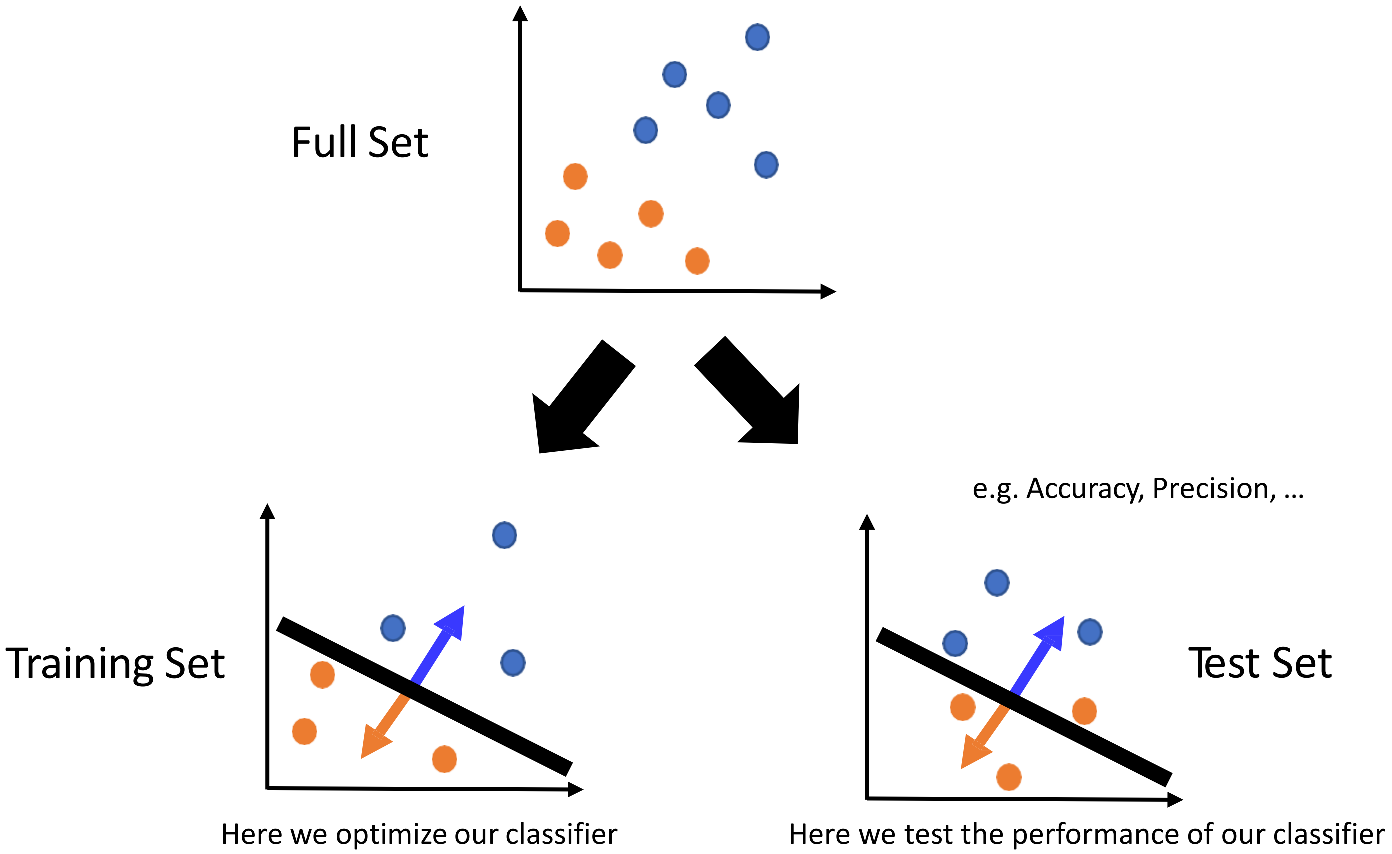
Preprocessing

Model selection

Training

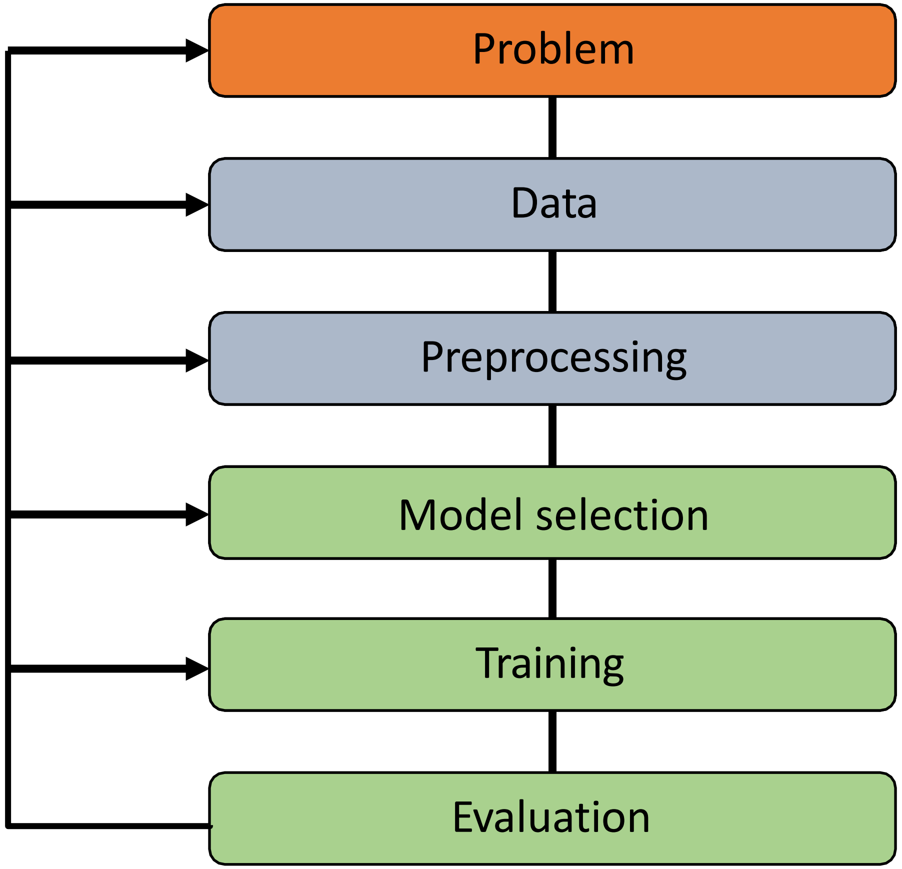
Evaluation

Train / Test Set Split



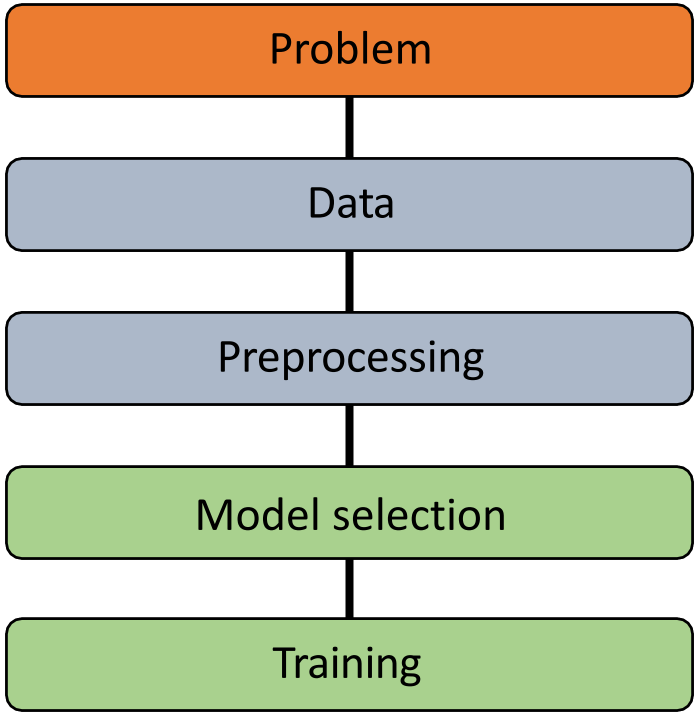
Machine Learning V

Question:
What if we are dissatisfied with our model's performance?



- Do we need to rethink / adjust the question?
- Do we need other data / features?
- Do we need to preprocess the data differently?
- Do we need another classifier?
- Can we train the classifier in any other way?
  - → **Different loss function?**
  - → **Regularization?**

Machine Learning Workflow



Hyperparameter
Parameters that control learning behavior of the algorithm

For Support Vector Machines, e.g.

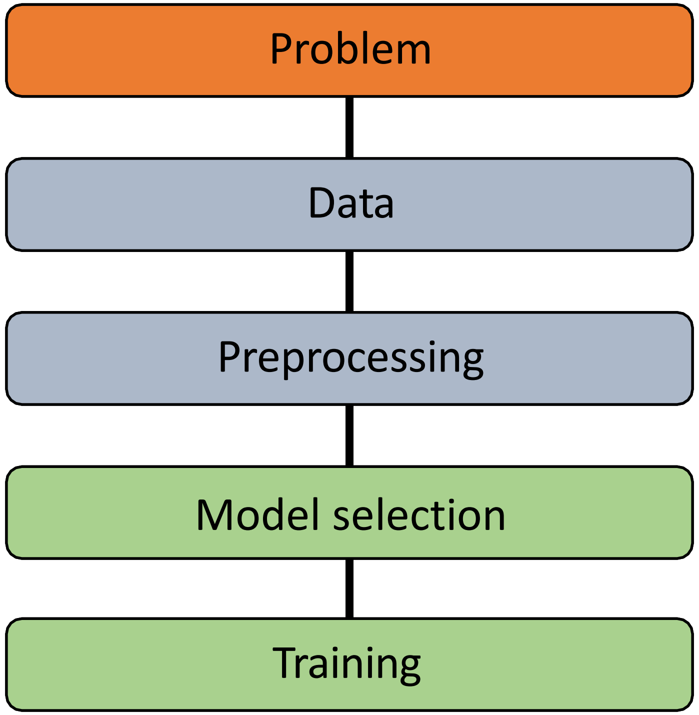
- **Kernel-Function** (e.g. ‚rbf‘, ‚poly‘, ‚linear‘)
- **λ (Penalty Parameter)**

Hyperparameter Tuning
Procedure to find out best combination of hyperparameters

e.g. Grid Search

		Kernel		
λ		linear	rbf	poly
	0.1	0.1 + linear	0.1 + rbf	0.1 + poly
	1	1 + linear	1 + rbf	1 + poly
	10	10 + linear	10 + rbf	10 + poly

Machine Learning Workflow



Hyperparameter

Parameters that control learning behavior of the algorithm

For Support Vector Machines, e.g.

- **Kernel-Function** (e.g. ‚rbf', ‚poly', ‚linear')
- **λ (Penalty Parameter)**

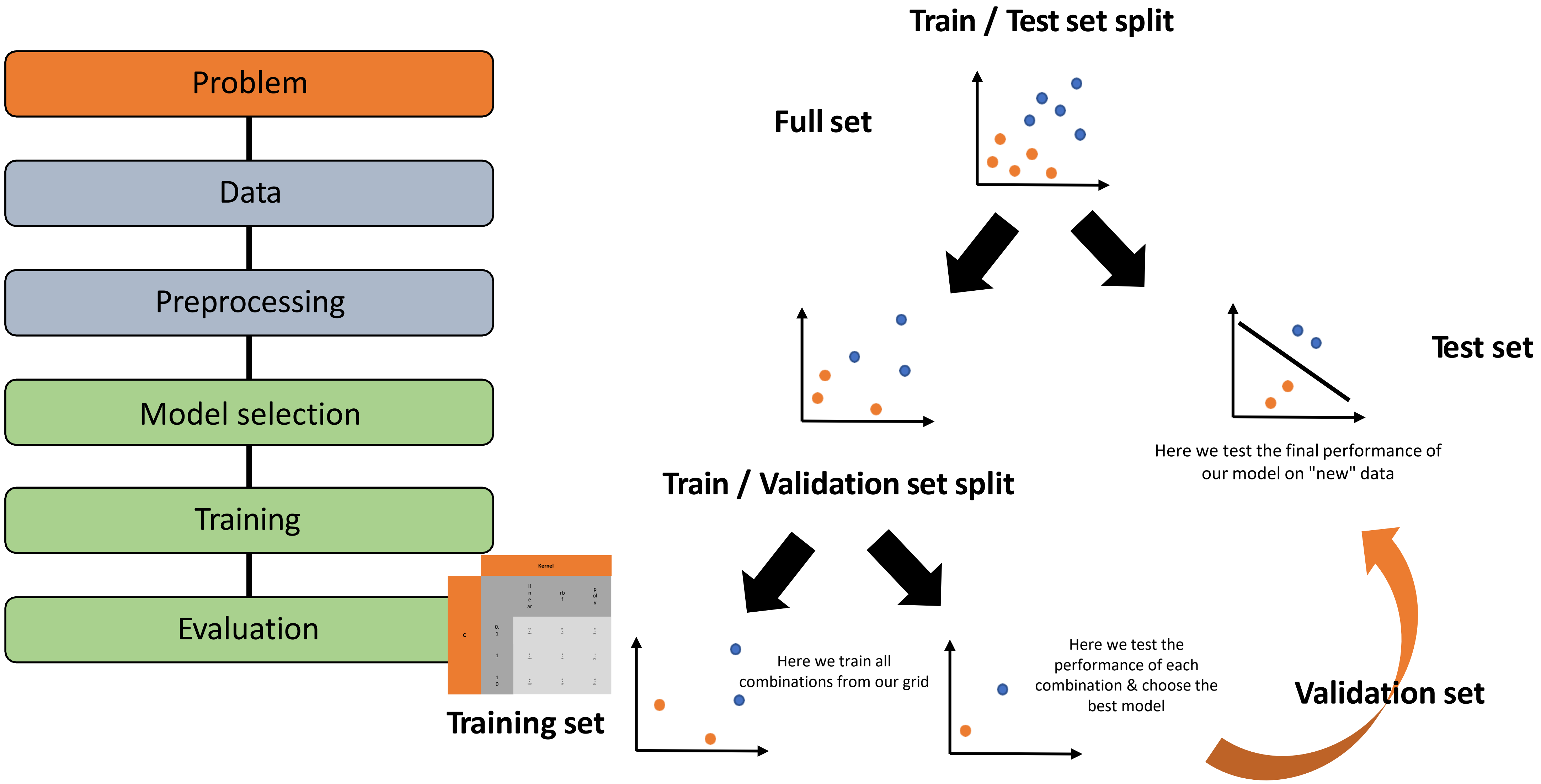
Hyperparameter Tuning

Procedure to find out best combination of hyperparameters

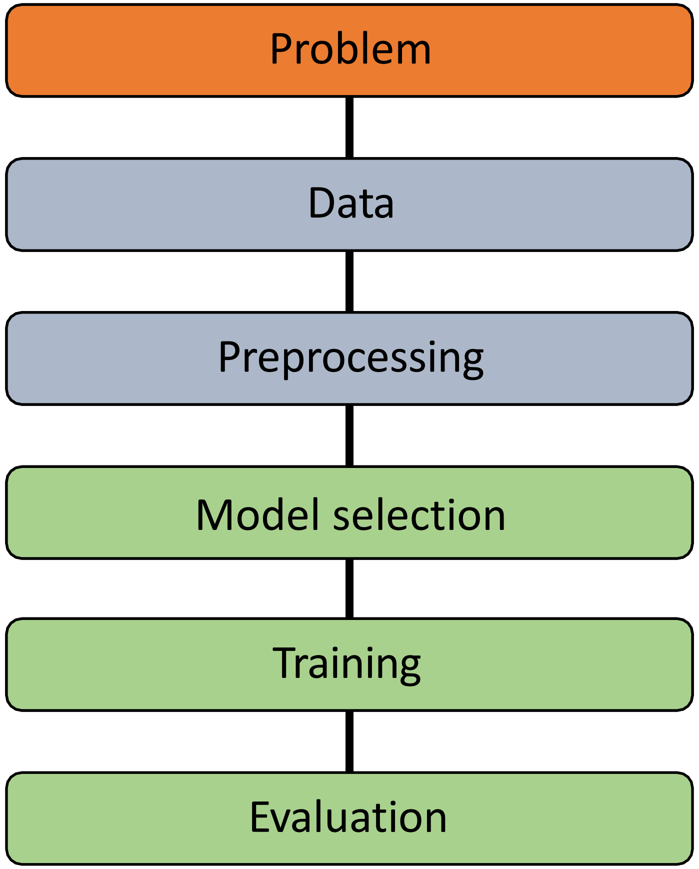
		Kernel		
		linear	rbf	poly
λ	0.1	70%	72%	68%
	1	72%	57%	65%
	10	74%	59%	67%

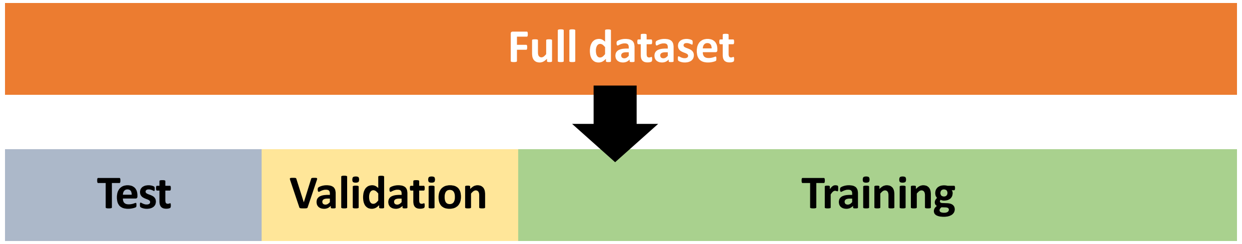
Question:
Where do we test the performance of the grid search? Test or train set?

Nested Cross Validation



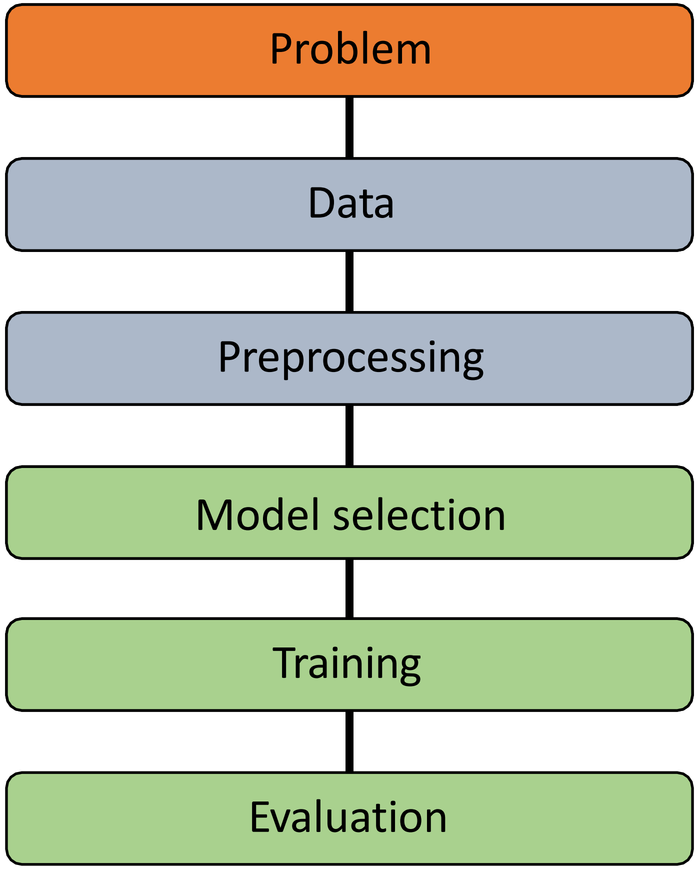
Nested Cross Validation

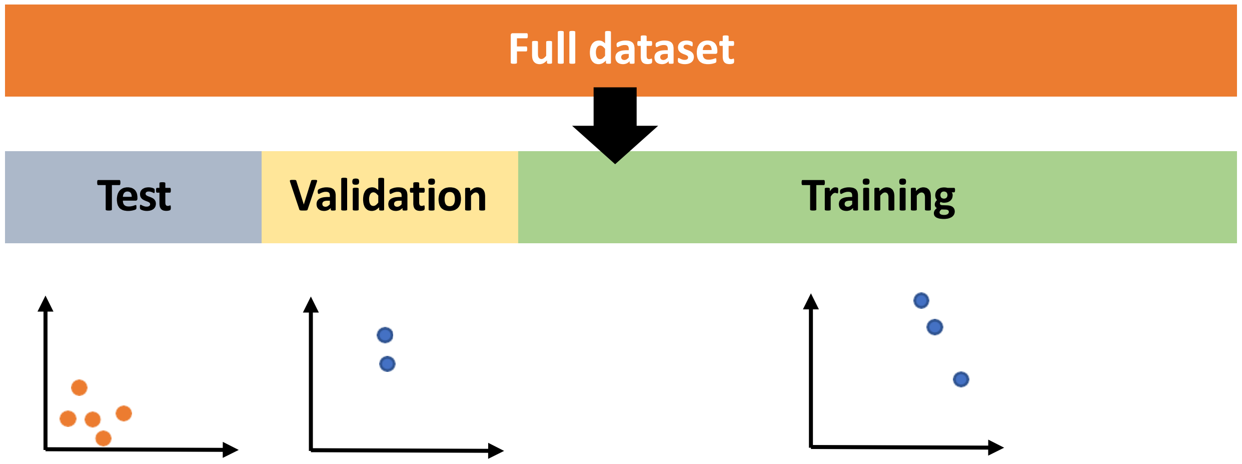




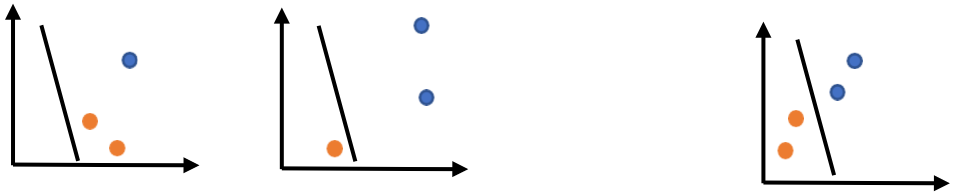
Question:
What problems might arise with this approach?

Nested Cross Validation



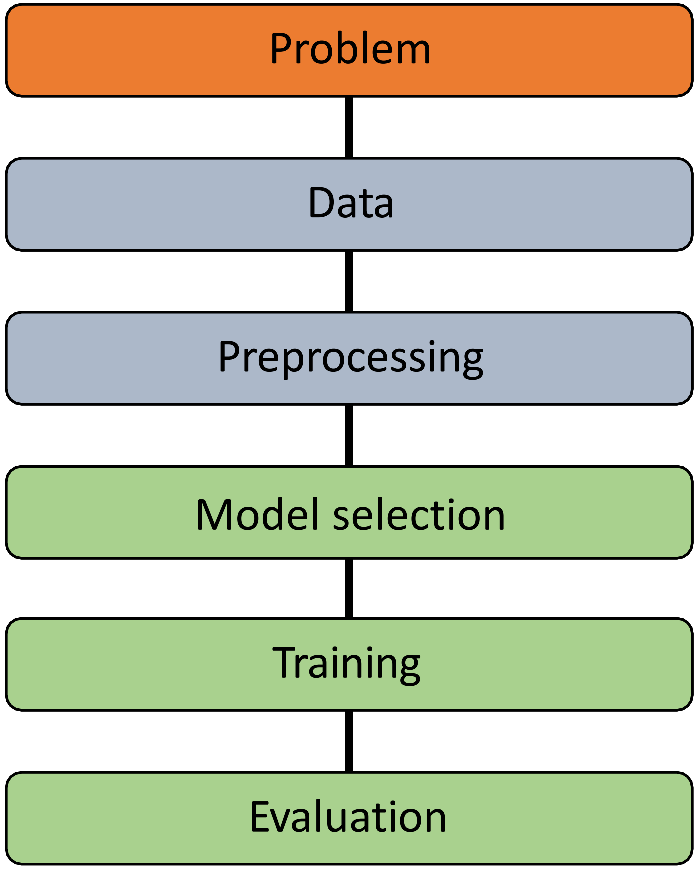


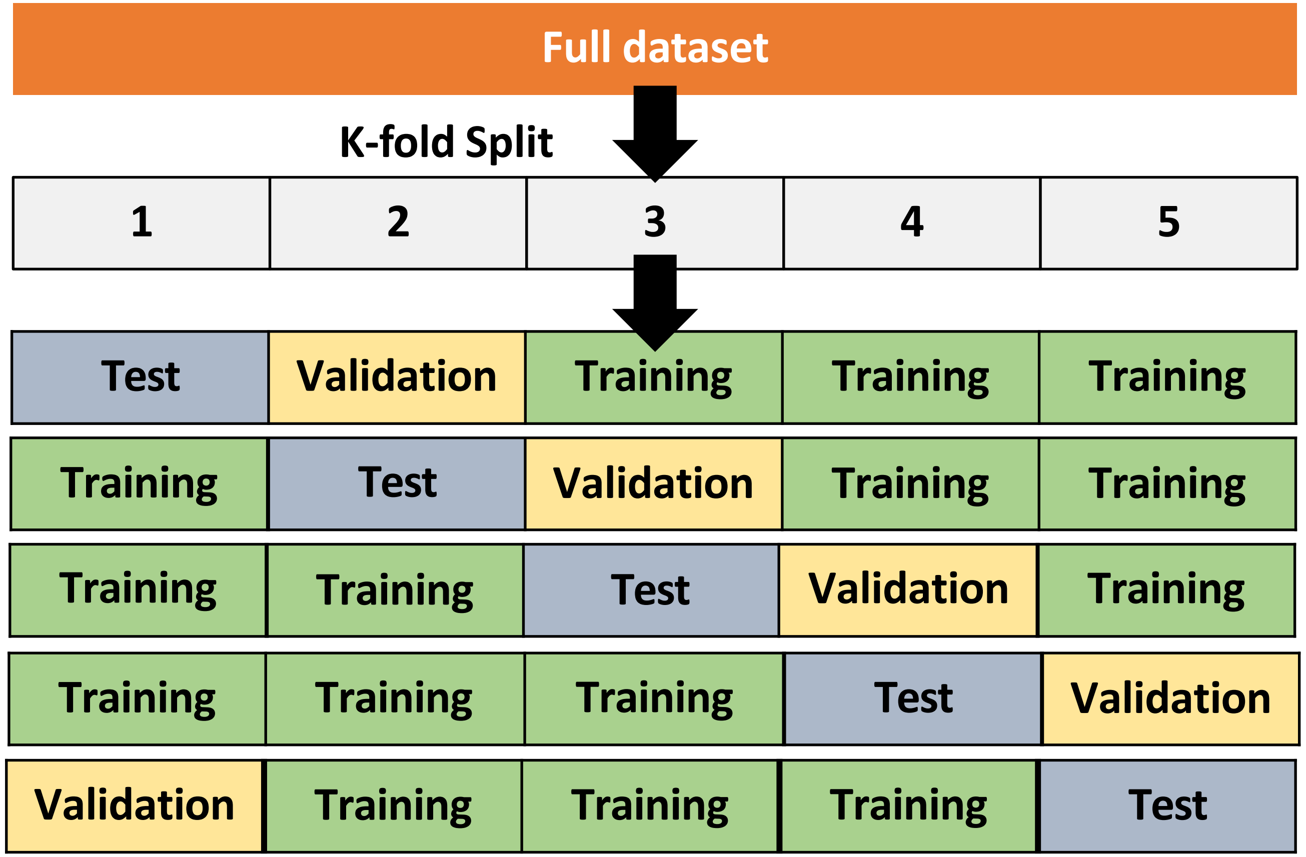
Solution: no random assignment, but attention to the distribution of classes in the complete data set.



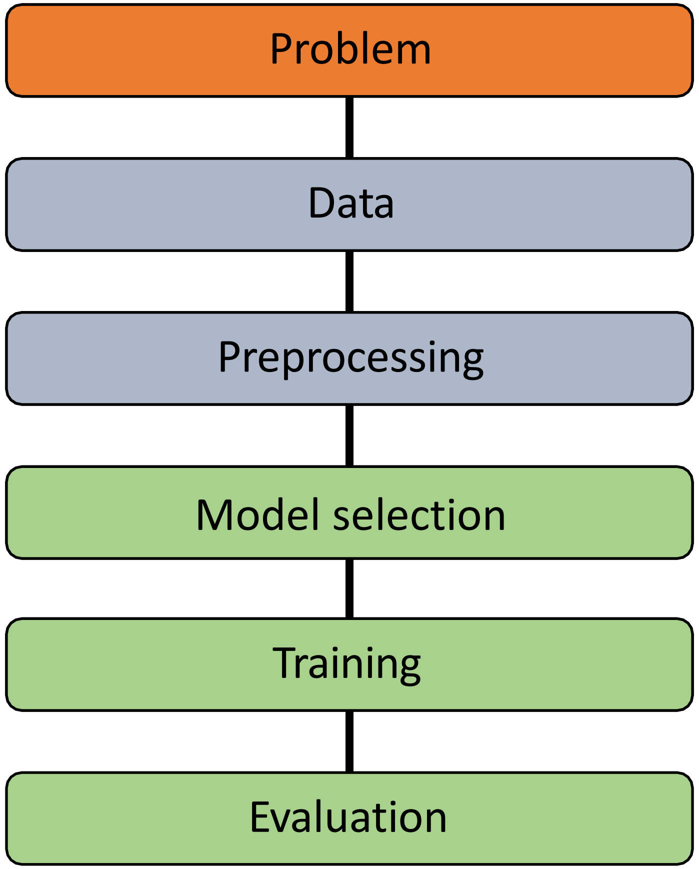
Solution: we iterate through several different splits in training/validation/testing

Nested Cross Validation

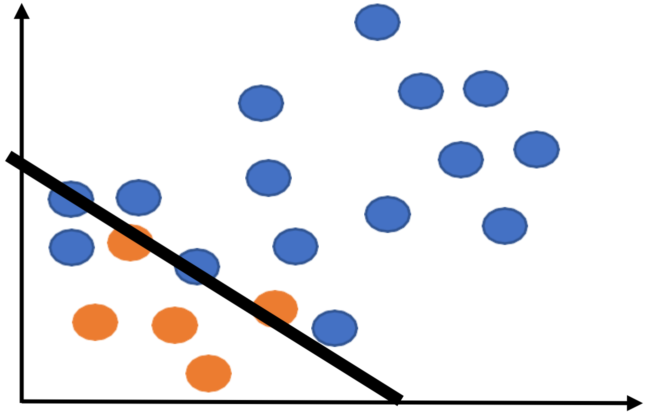




EVALUATION

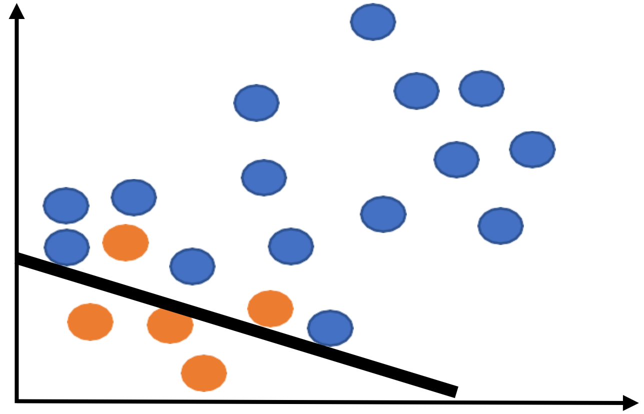


Problem: Imbalanced Datasets



$$Accuracy = \frac{17}{20} = 85\%$$

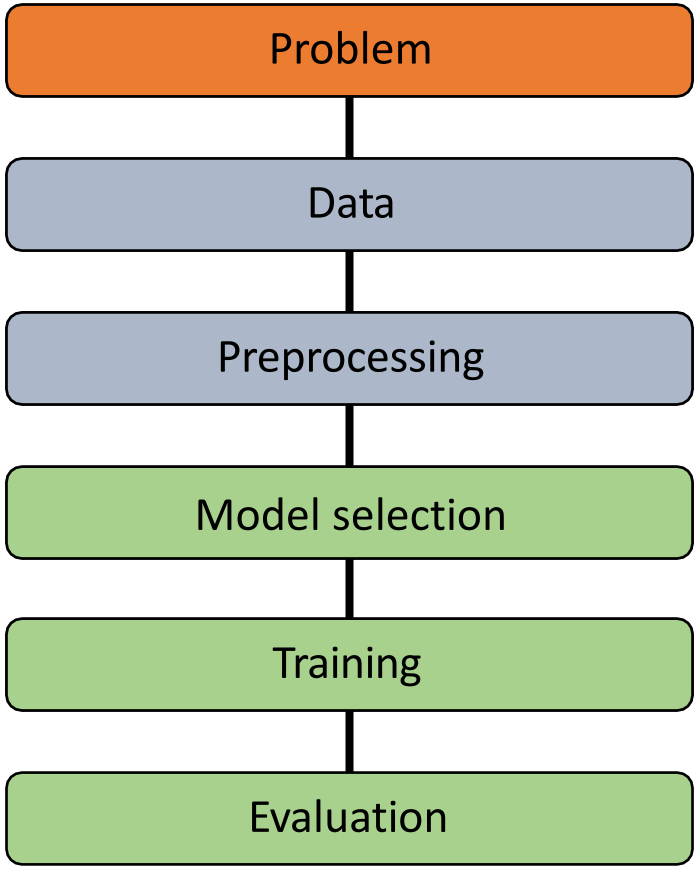
$$Balanced\ Accuracy = \frac{\frac{4}{5} + \frac{13}{15}}{2} = 83.3\%$$



$$Accuracy = \frac{18}{20} = 90\%$$

$$Balanced\ Accuracy = \frac{\frac{3}{5} + \frac{15}{15}}{2} = 80\%$$

SUMMARY



- Supervised vs Unsupervised? Regression? Classification?
- Which features are usefull? (feature selection)
- Bring features to correct form: scaling, dimensionality reduction (feature engineering)
- e.g. SVM, random frest, decision tree, …
- Nested Cross Validation, HPO, Grid Search
- Loss function, Gradient Descent, Learning Rate, Regularization
- e.g. AUC, (Balanced) Accuracy, Sensitivity, Specificity, …