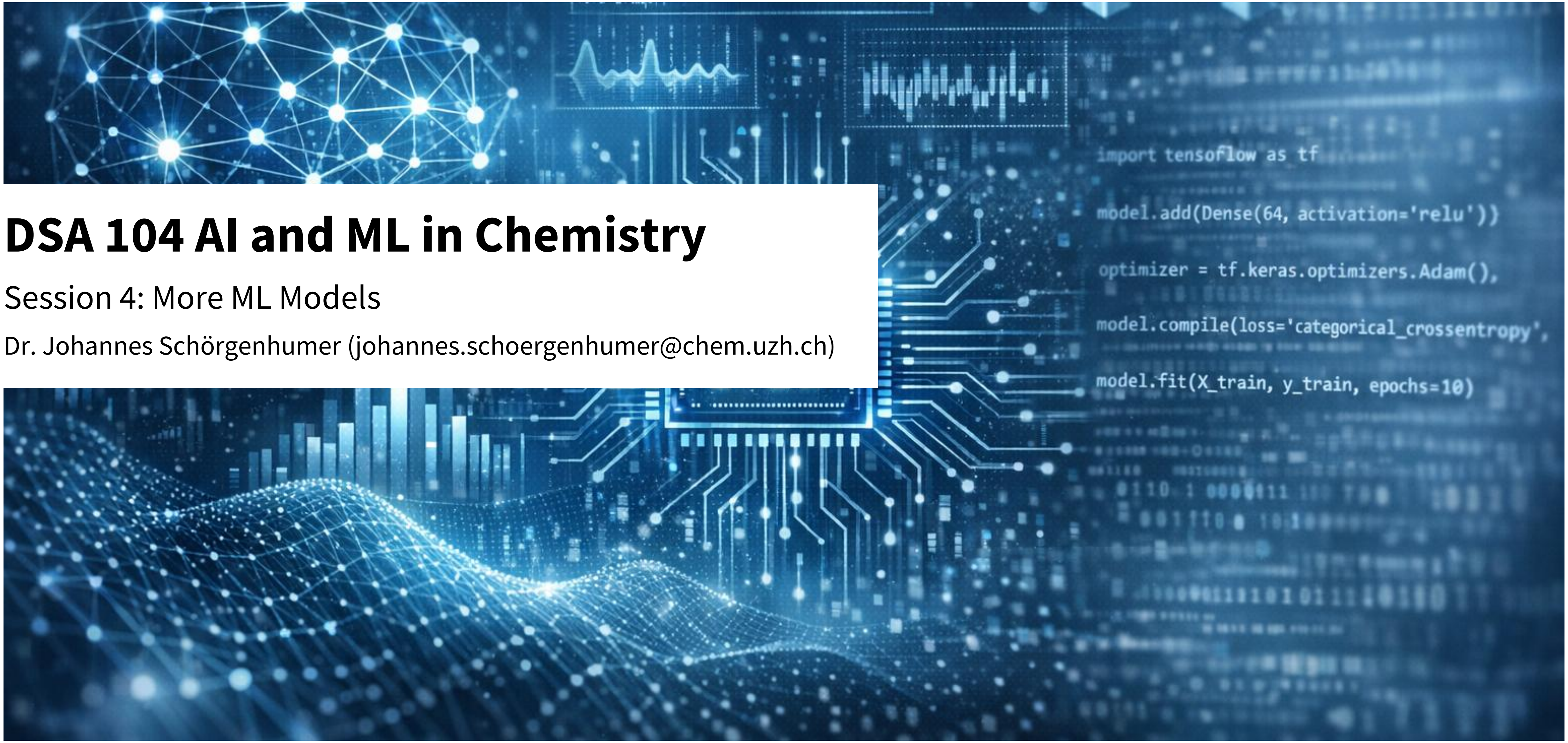
Universität Zürich UZH | Department of Chemistry


import tensoflow as tf
model.add(Dense(64, activation='relu'))
optimizer = tf.keras.optimizers.Adam(),
model.compile(loss='categorical_crossentropy',
model.fit(X_train, y_train, epochs=10)

DSA 104 AI and ML in Chemistry

Session 4: More ML Models

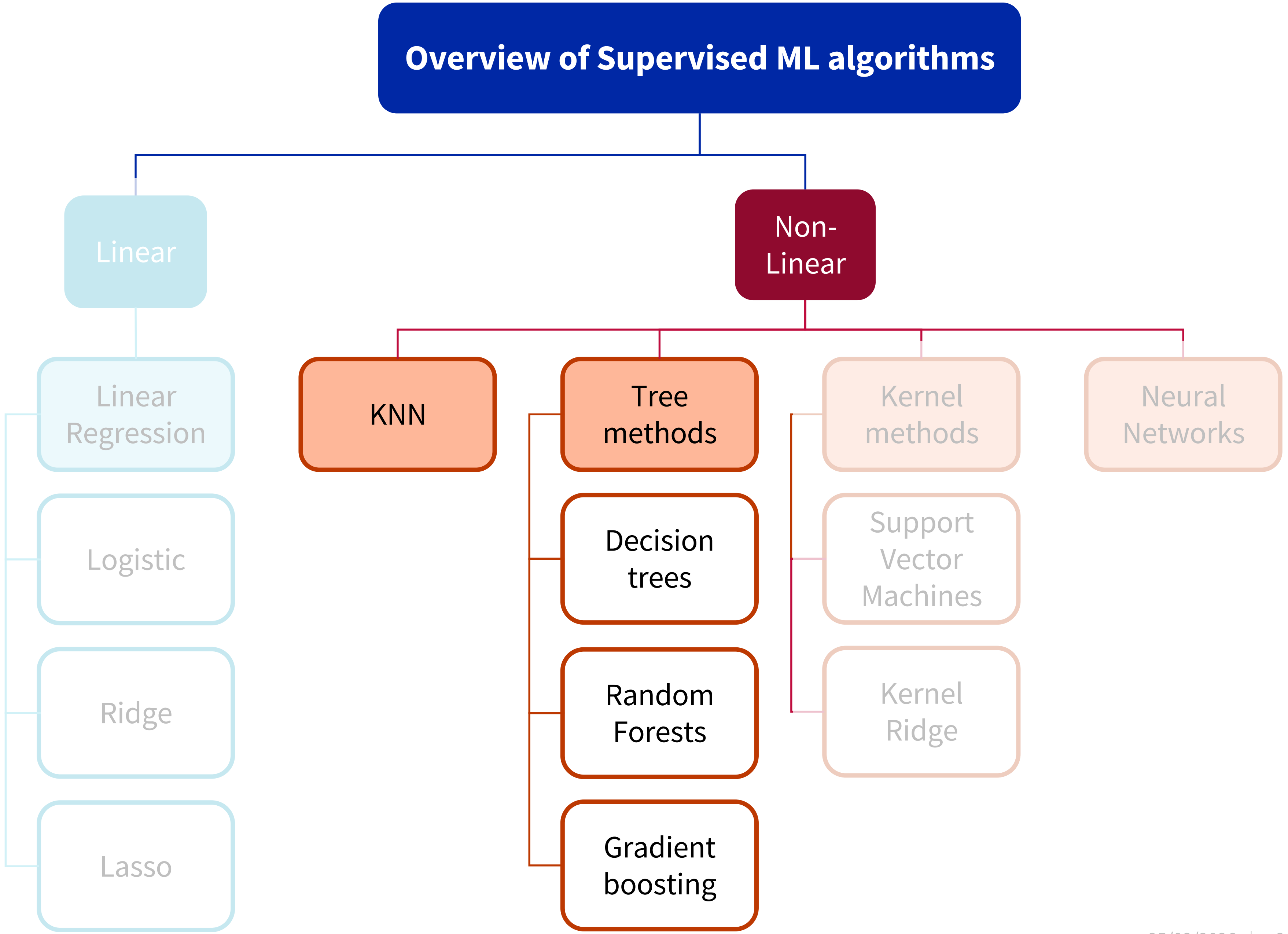
Dr. Johannes Schörgenhumer (johannes.schoergenhumer@chem.uzh.ch)

Lesson outline

Wrap up KNN

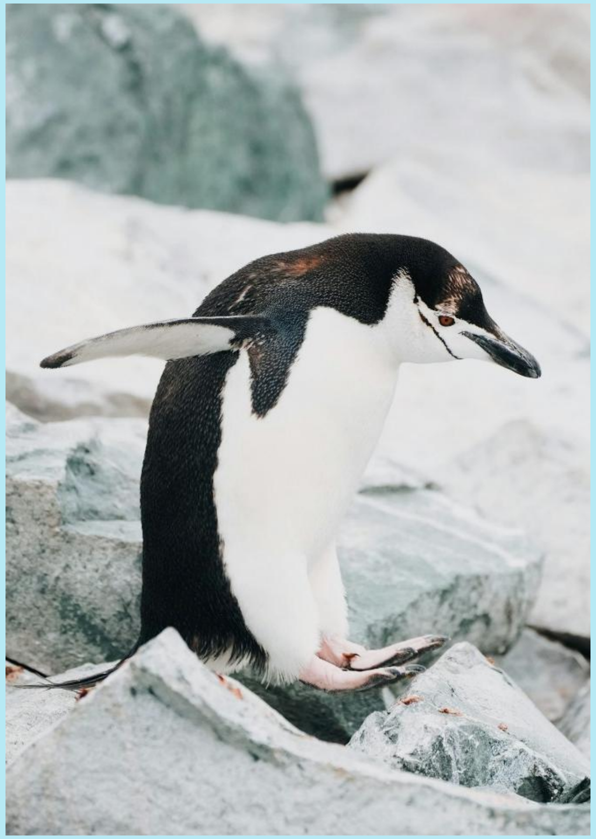
GridSearchCV

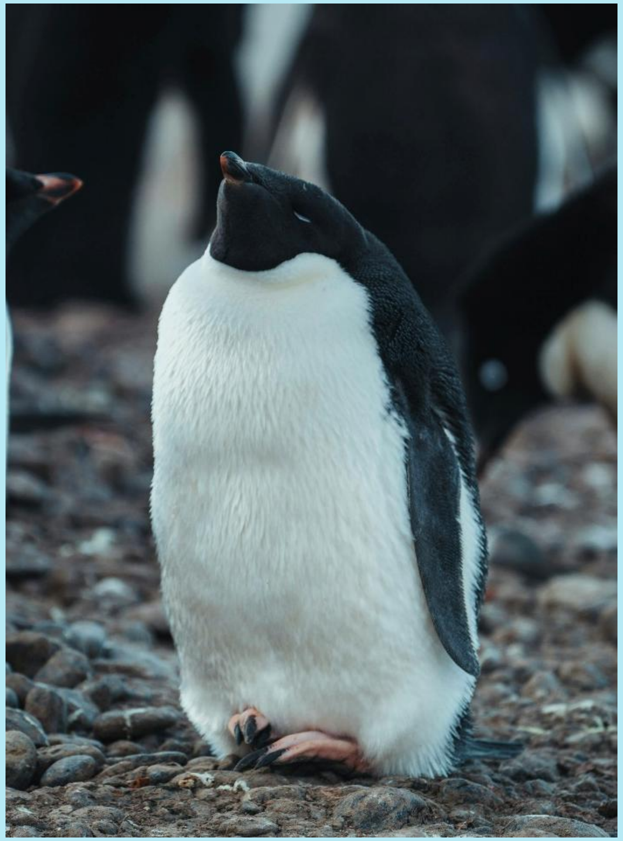
Tree methods:
Decision trees
Random forests
Gradient Boosting

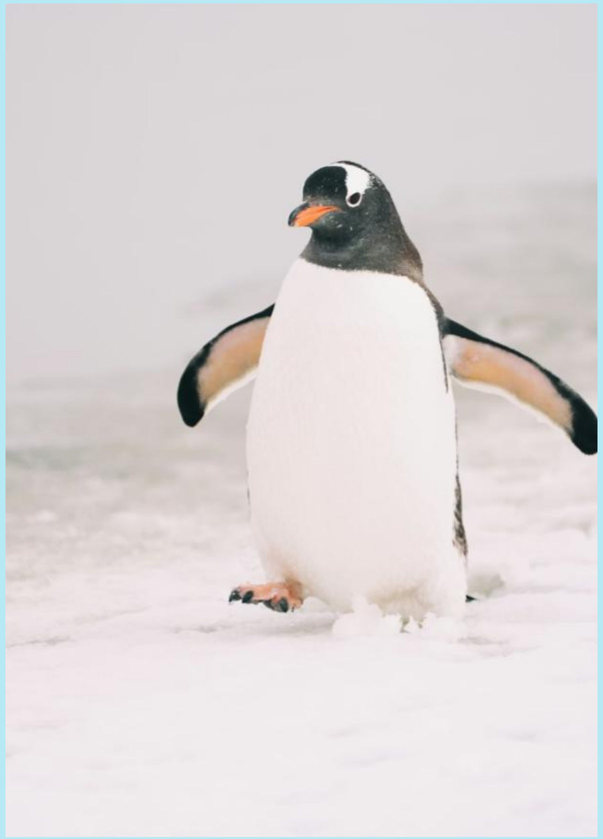


Classification model KNN

- Explore the impact of different numbers for K (*KNN-penguins-decisionboundaries*)
- Use a KNN Classifier to predict different penguin species for unknown data! (*KNN-penguins*)


Chinstrap


Adélie


Gentoo

Your penguin?

?

Useful Tools: Cross-validation and GridSearch

Cross-validation (CV) = model evaluation to deliver more **stable and reliable estimation** of the generalisation:

— A single train/test split might not represent the true data distribution

— Performance may vary due to randomness

— Small datasets are particularly sensitive

— Working principle – e.g. K-fold CV:

  — Split the dataset into **K equal-sized folds (=subsets)**

  — Train the model on K-1 folds, test on the remaining one

  — Permutate and average the K-results

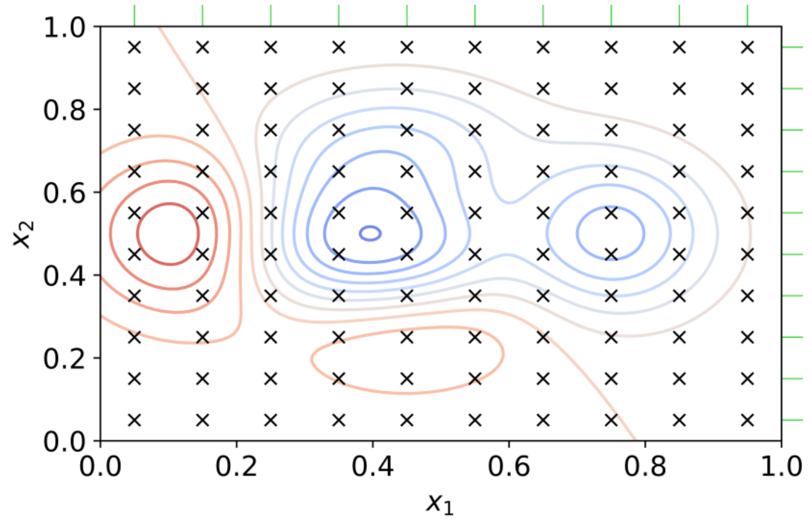
— Best practice: keep another independent subset for final model evaluation!

Train	Train	Train	Train	Test
Train	Train	Train	Test	Train
Train	Train	Test	Train	Train
Train	Test	Train	Train	Train
Test	Train	Train	Train	Train

GridSearch = Systematic scan of hyperparameters:

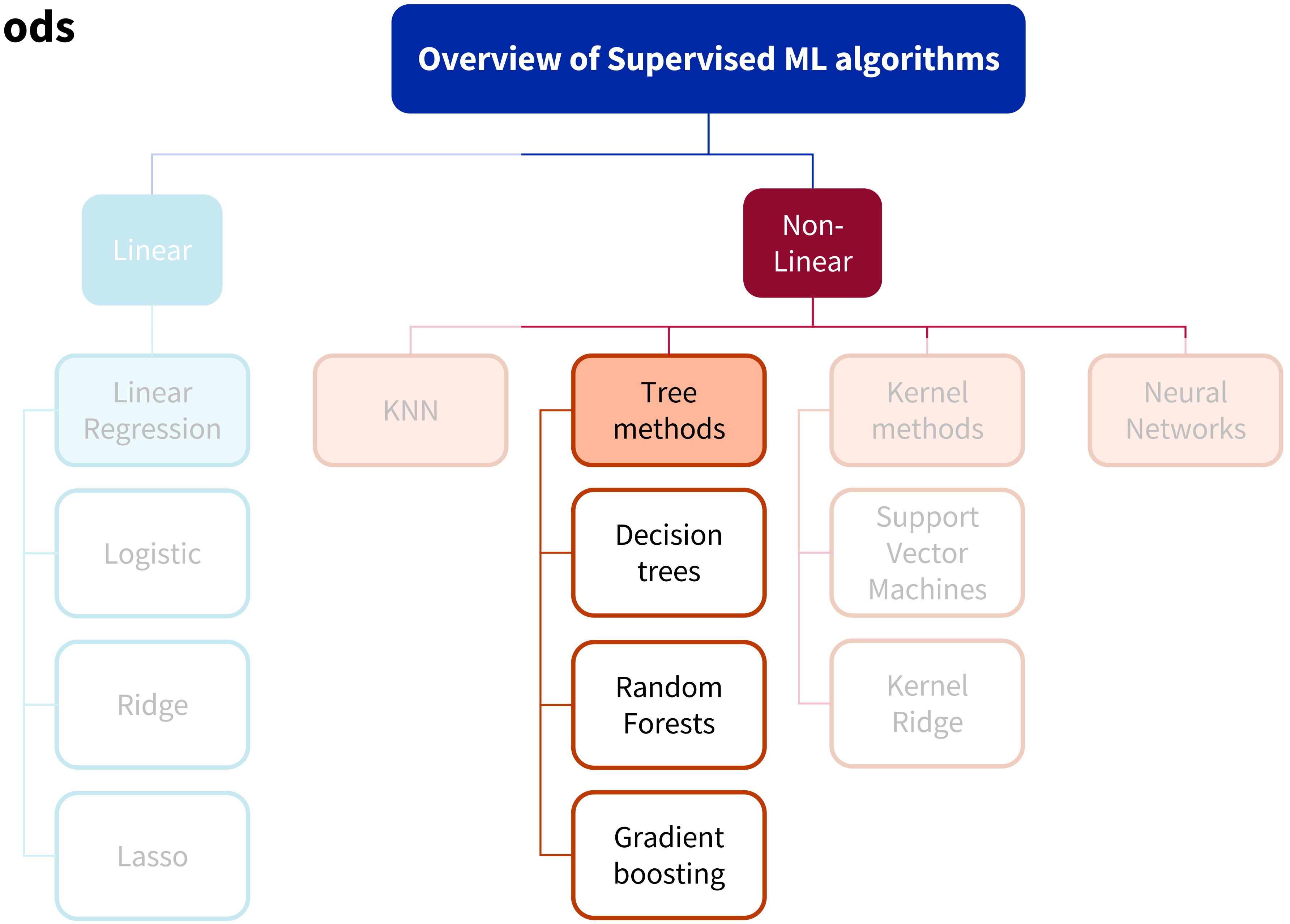
— Traditional mode for hyperparameter optimisation

— Curse of dimensionality: restrict and discretise hyperparameter space



Alexander Elvers – own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=84255403

Tree methods



Decision trees

Core idea:

— Reach predictions by “asking questions successively”.

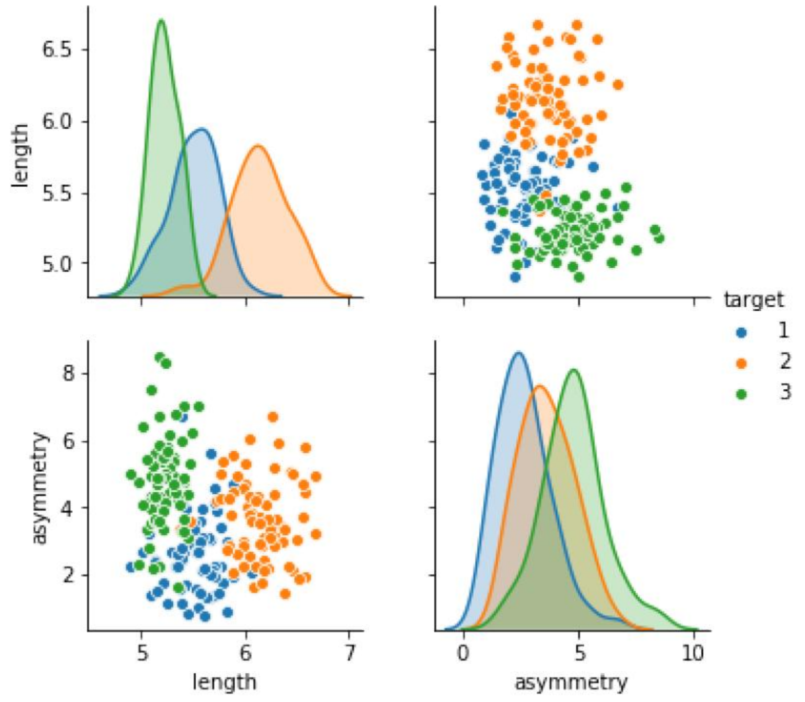
— Each question **splits** the data into different **branches**

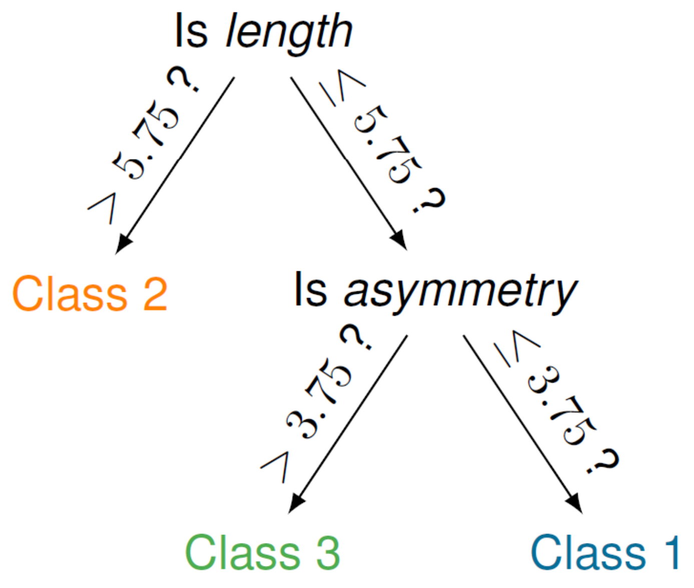
— **Non-leaf** nodes correspond to **questions**

— Each **leaf** corresponds to a final **prediction**

— Data is learned hierarchically

— Used for Classification And Regression (e.g. CART algorithm in scikit)





Picture from A. Schörgenhumer, *Hands-on AI I*, Lecture materials, **2023**, JKU Linz.

Decision trees: Working principle

— **Start** with the full dataset, pick a feature and threshold that best splits the data (the **root node**) to reduce:

  — impurity (for classification) or

  — error (for regression)

— **Recursively** repeat the process for each subset, creating **branches**

— **Stop** when:

  — maximum depth is reached

  — further splits don't improve performance

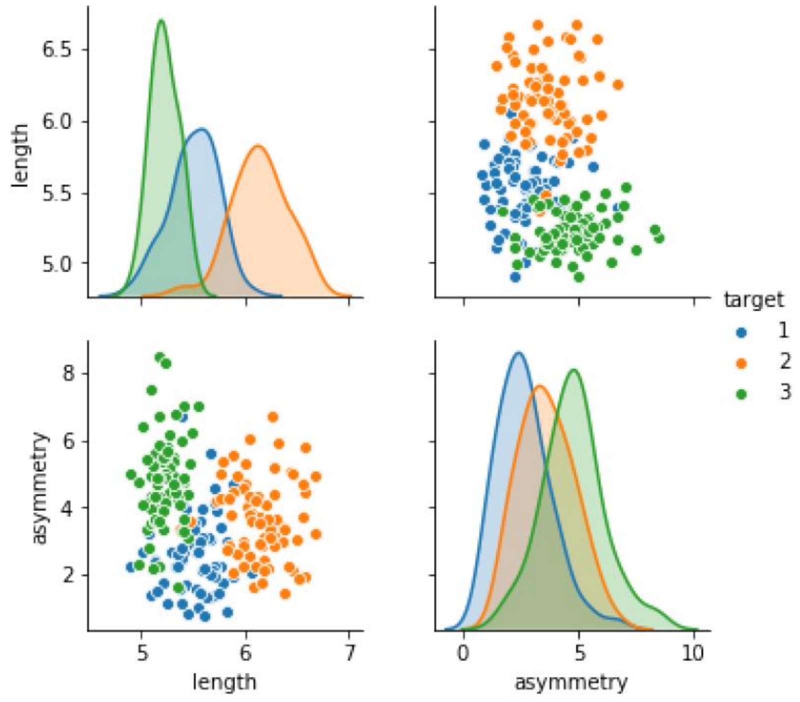
  — nodes are too small

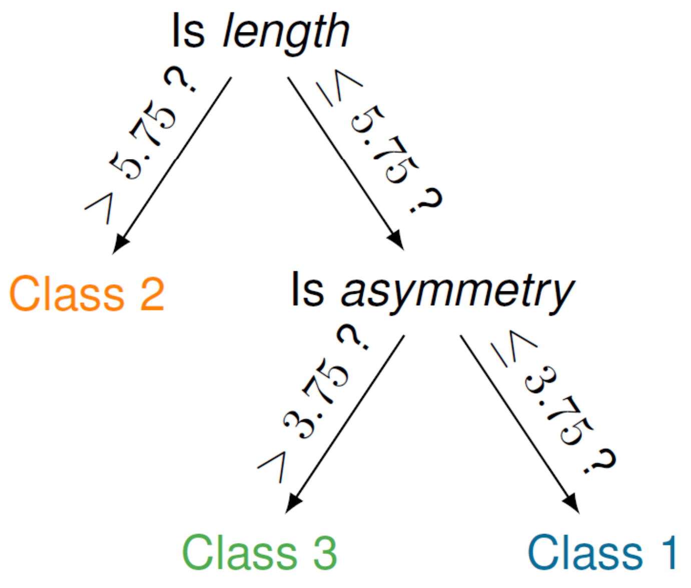
Pruning:

— Reduces size of trees to reduce overfitting e.g. by

  — Pre-pruning, e.g. stopping early

  — Post-pruning, e.g. removing "inefficient" branches





Picture from A. Schörgenhumer, *Hands-on AI I*, Lecture materials, **2023**, JKU Linz.

Decision trees: Metrics

Impurity measures (classification) or variance/error measures (for regression) evaluate how "good" a split is. E.g. for classification:

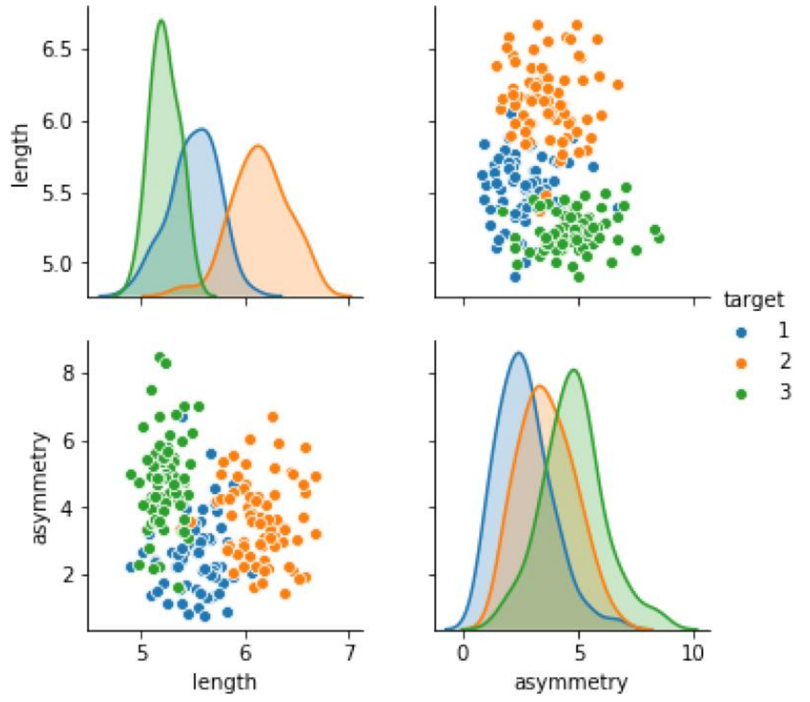
— Gini impurity = probability of mislabelling randomly according to class distribution in dataset, e.g. for K classes:

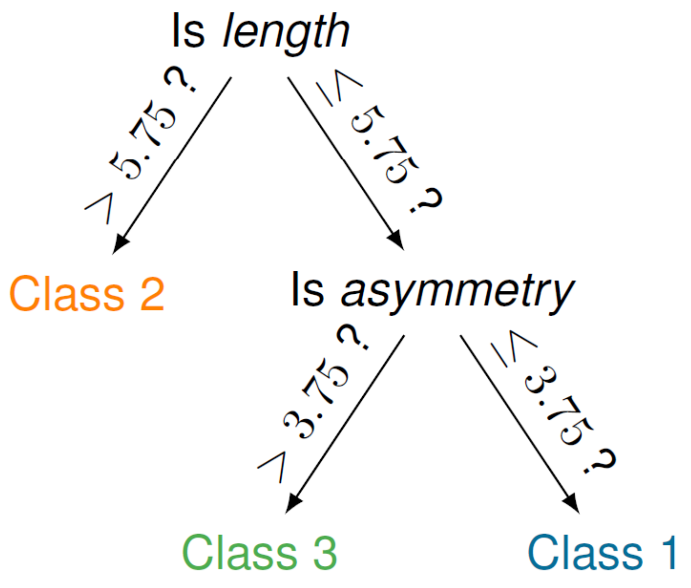
$$G = I_G(p) = 1 - \sum_{1=1}^{K} p_i^2$$

— Information gain / entropy:

$$H = I_E(p) = -\sum_{1=1}^{K} p_i \log_2 p_i$$

— Work as "loss functions" for each split and will be optimised in tree-building (class concentration in child nodes maximised)





Picture from A. Schörgenhumer, *Hands-on AI I*, Lecture materials, **2023**, JKU Linz.

Decision trees

Strengths

— Easy to understand and **interpret**

— Work with numerical and categorical data

— Requires little preprocessing (no scaling needed)

— Capture nonlinear relationships

Weaknesses:

— **Unstable**: small data changes can lead to very different trees

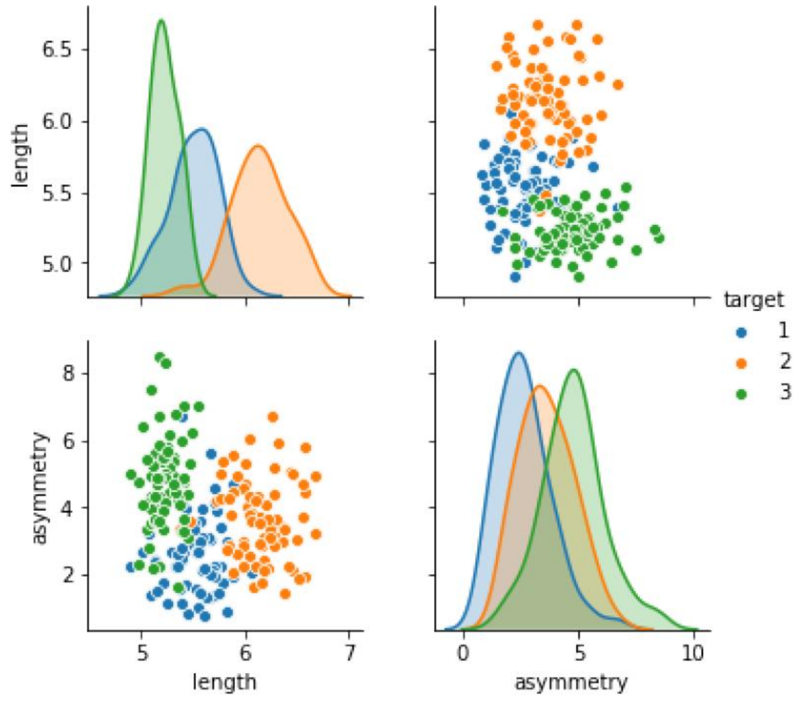
— Prone to **overfitting** (need **pruning**!)

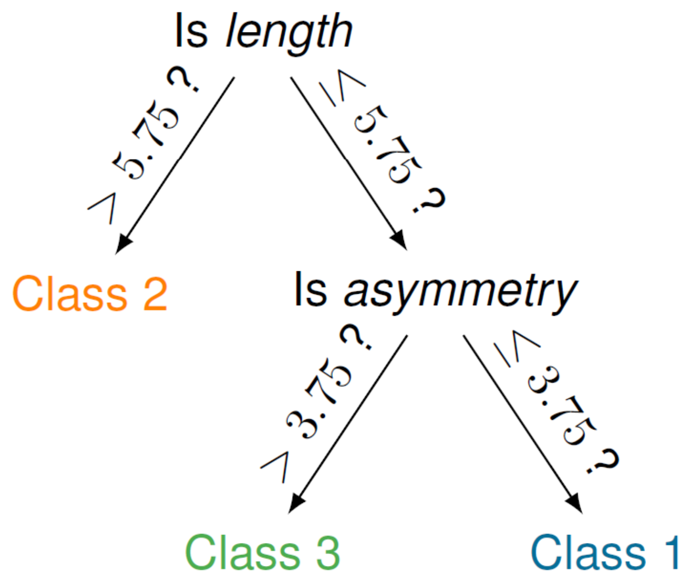
— Often less accurate than ensemble methods

Applications:

— Exploratory analysis, quick baseline models (when **interpretability** matters)

— More important as basis for: **Random forests** and **gradient boosting** methods





Picture from A. Schörgenhumer, *Hands-on AI I*, Lecture materials, **2023**, JKU Linz.

Building a tree-based model for activities

1) Work together in small groups and (randomly) pick **three** of these **features**:

Temperature
Precipitation
Snow
Wind
Time of the day
Exhaustion level
Hunger

2) Build a decision tree with the three features choosing simple splits (e.g. low, medium, high). Consider the following **labels**:

Winter sports
Climbing
Reading
BBQ in the park
Sleep

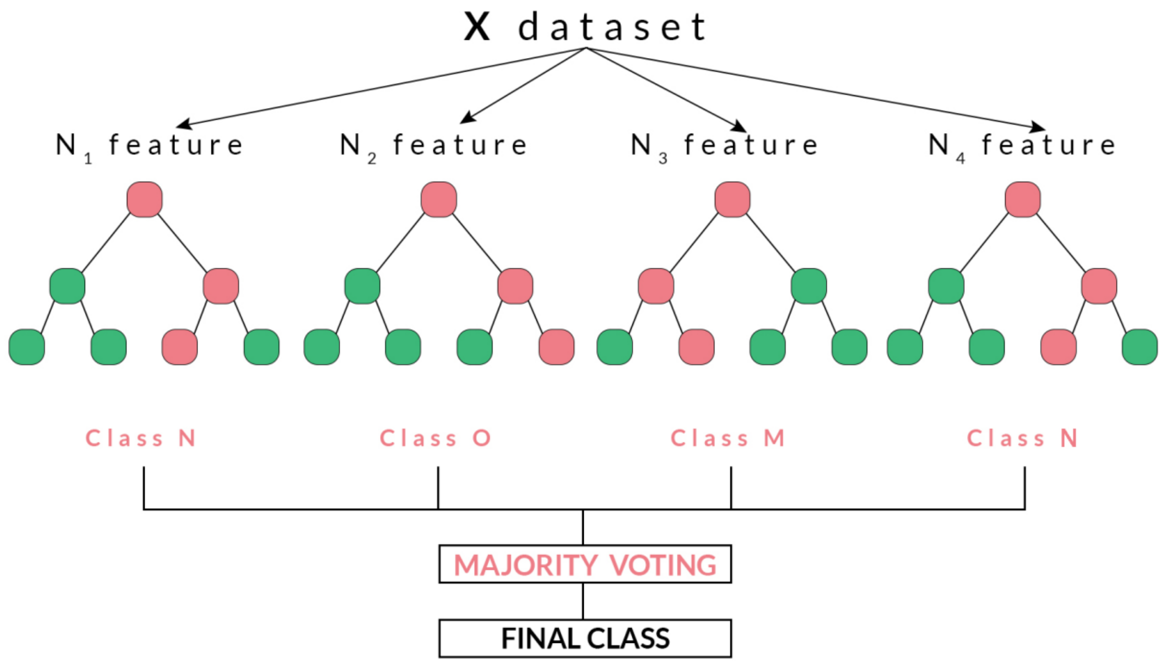
PS: You can use one feature more than one time in your tree!

Random Forests (RF)

Decision trees prone to overfitting! Thus:

- Build **ensemble model** from several decision trees!
- Final decision is either
  - A **majority vote** (Classification) or
  - An **average of all tree outputs** (Regression)

- **Two sources of randomness** introduced:
  - Bootstrap sampling (Bagging): Each tree trained on **random subset** of training data
  - Decorrelating trees: At each split, trees pick from **random subsets of features** (no “same splits”)



Picture taken from: https://blog.quantinsti.com

Random Forests: Pros and Cons

Advantages

- Works with both classification & regression
- Easy to parallelise (trees train independently)
- **Reduced overfitting** (averaging many diverse trees reduces variance)
- High accuracy well out-of-the-box (**strong baseline performance**)
- Handles well:
    - nonlinear relationships
    - feature interactions
    - high-dimensional data
    - missing data (to some extent)
- **Robust** to noise and outliers (one noisy sample affects only a few trees)

Disadvantages

- **Less interpretable** than a single decision tree
- Computationally **more expensive**
- Large ensembles can require a lot of memory
- Slower inference than simple models

Random Forest: Applications

Not ideal for:

— Very high-dimensional sparse data (e.g., text → use linear models)

— Problems where interpretability is crucial

Otherwise well suited if you need:

— A strong baseline model

— Good performance without complex tuning

— A robust algorithm for noisy or tabular data

— Feature importance insights

— Something still fairly simple (e.g. as compared to gradient boosted trees)

Many applications in real-world data!

Random Forests: Breast Cancer Classifications.

- Use a RandomForestClassifier to distinguish benign and malignant tumors
- GridSearchCV to find best hyperparameters

Gradient boosting (GB): gradient-boosted trees

Gradient boosting: general approach, applies to different models. E.g. XGBoos or LightGBM use decision trees!

Another ensemble model, but:

— Trees built **sequentially**

— Each new tree tries to correct mistakes (**residual errors**, i.e. loss function) of the previous one

Key idea:

— Start with a simple prediction (e.g. the mean of the target).

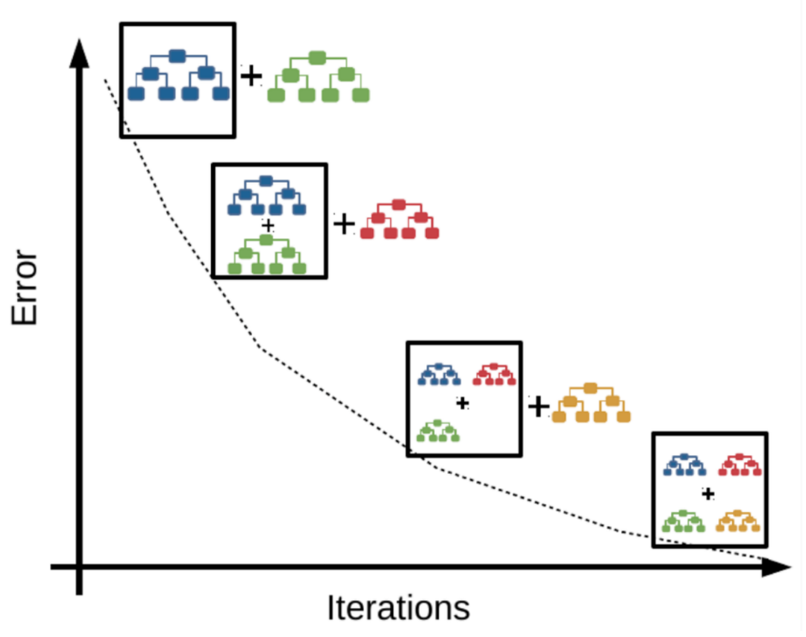
— Compute the residuals = what the model got wrong

— Train a small tree to predict those residuals

— Add that tree to the model (scaled by a learning rate)

— Repeat many times

Model **transforms weak learners** (decision trees) **into a strong learner** by gradually reducing the remaining error!



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

with $\hat{y} = F(x)$

$$-\frac{\partial L_{\mathrm{MSE}}}{\partial F(x_i)} = \frac{2}{n}\left(y_i - F(x_i)\right) = \frac{2}{n}h_m(x_i)$$

With h_m as new estimator (tree) to be fitted to residual $y_i - F(x_i)$. (where $1<m<M$ in M stages of the algorithm)

Picture adapted from: https://blog.mlreview.com/

GB vs. RF

Gradient-boosted trees

— Many shallow trees

— Trained one after another

— Each tree fixes previous error

Random Forest

— Many deep trees

— Trained independently

— Averaged together

GB, e.g. XGBoost, therefore usually **more accurate but more sensitive** to hyperparameters and noise.

Crucial: regularisation to reduce overfitting and enhance generalisation.

— Regularisation parameters include, e.g.

  — Natural parameters, e.g. Number of gradient boosting iterations M, or tree depth

  — Shrinkage: $F_m(x) = F_{m-1}(x) + \nu \cdot \gamma_m h_m(x)$, with γ_m as weight of tree $h_m(x)$, and **regularisation parameter $\boldsymbol{\nu}$** for learning rate (from 0 to 1), i.e. learning scaling

  — Complexity penalty, e.g. restrict number of leaves (post-pruning)

Tree ensemble models in review

- Get together in small groups
- Do a quick search and find an applications of RF and GB in literature of your domain. Check:
  - Which models were used?
  - What were they used for?
  - Was there any modification done?
  - How did they compare to others?
- Report a quick summary to the other groups