Random Forests

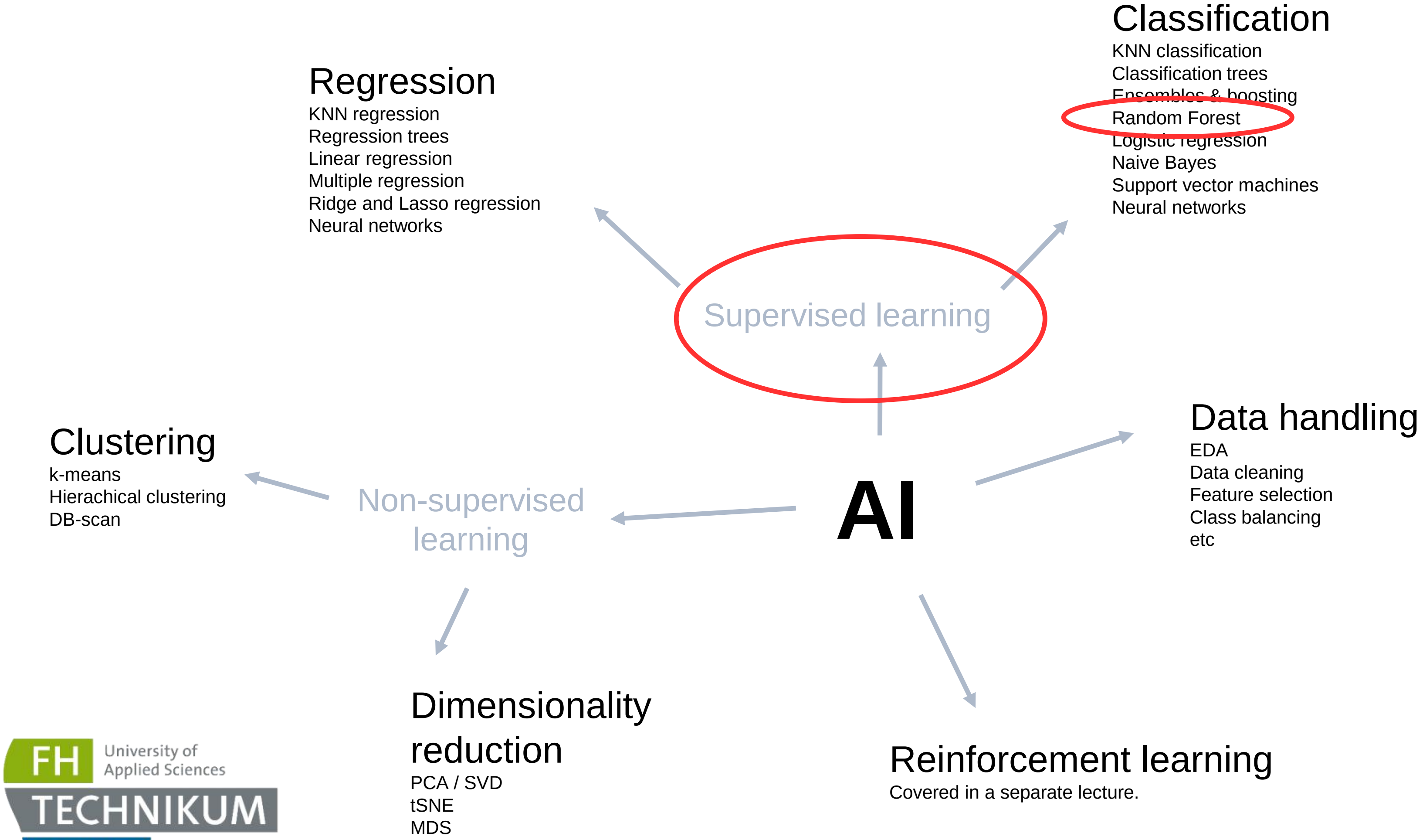
ML2: AI Concepts and Algorithms (SS2025)

Faculty of Computer Science and Applied Mathematics

University of Applied Sciences Technikum Wien

Lecturer: Rosana de Oliveira Gomes

Authors: S. Lackner, B. Knapp, S. Rezagholi, R.O. Gomes



AI

Supervised learning

Regression

KNN regression
Regression trees
Linear regression
Multiple regression
Ridge and Lasso regression
Neural networks

Classification

KNN classification
Classification trees
Ensembles & boosting
Random Forest
Logistic regression
Naive Bayes
Support vector machines
Neural networks

Non-supervised learning

Clustering

k-means
Hierachical clustering
DB-scan

Dimensionality reduction

PCA / SVD
tSNE
MDS

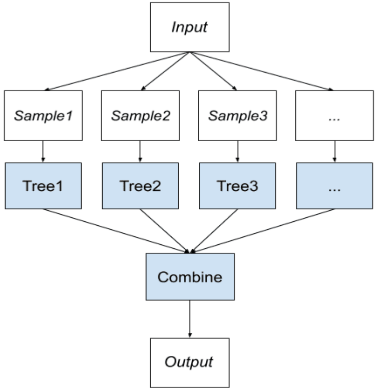
Data handling

EDA
Data cleaning
Feature selection
Class balancing
etc

Reinforcement learning

Covered in a separate lecture.

Ensemble Learning: Recap



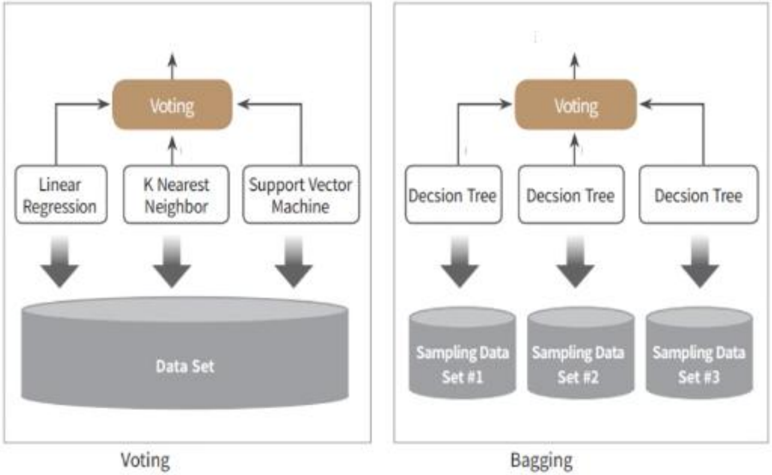
Most common non-DL models used in ML.

In ensemble learning several **base learners are combined** to a single model: *"Wisdom of the crowd".*

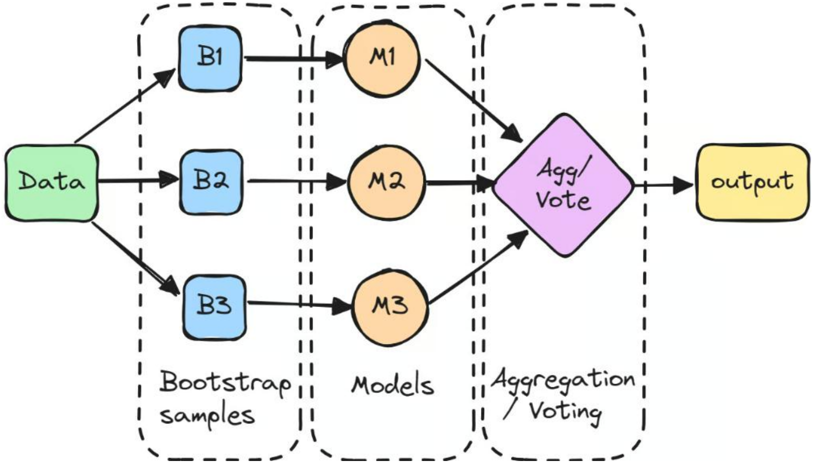
From last time:

Majority Voting combines predictions from multiple model. The final output is the most common class (for classification)

Bagging trains multiple independent models on different bootstrap samples of data. Reduces variance.



Bagging: Recap



- Bootstrap samples from training data

- ML model for each sample

- Aggregation vote to decide which prediction to use

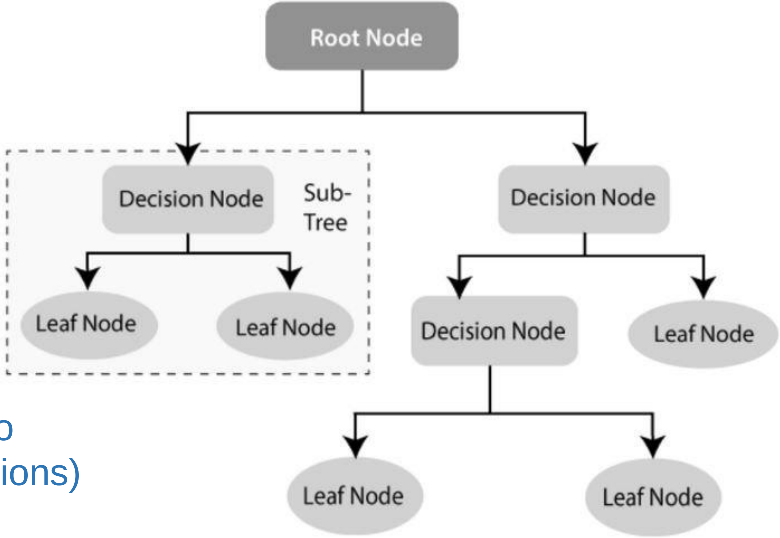
  majority voting (classification)

  average (regression)

Bootstraping: sampling random patches with or without replacement. Reduces variance of a model.

Multiple learners: diversity in learning decreases bias.

Decision Trees: Recap



Components:

Root Node: top node of the tree.
Decision Nodes: split data based on feature thresholds criteria.
Leaf Nodes: endpoints that assign a prediction.
Branches (Edges): paths connecting nodes, representing outcomes of decisions.

A **high impurity** means the samples in the node belong to multiple classes — we want to reduce this! (impurity functions)

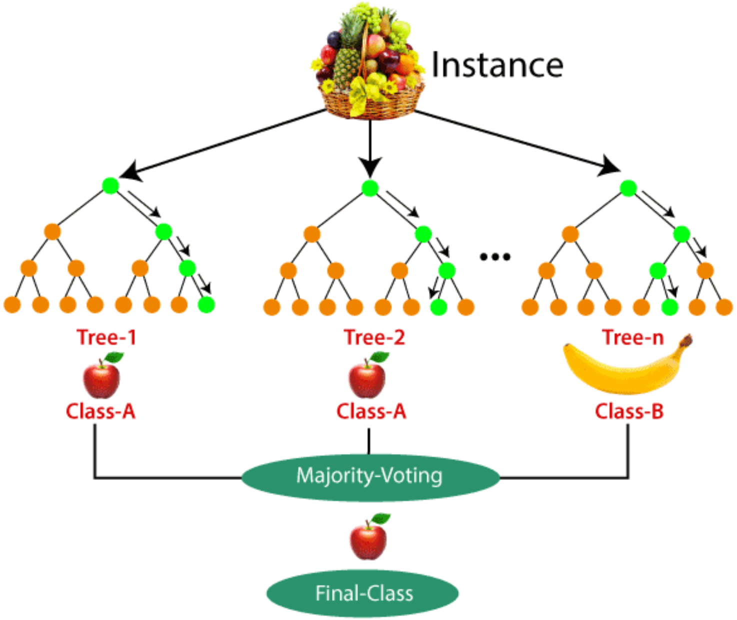
Recursive Partitioning Algorithm:

1. Evaluate all possible feature-threshold pairs
2. Choose the pair that minimizes impurity after the split
3. Split the data accordingly until it reaches a **stopping condition.**

Parameters:

Tree depth, minimal # of observations per node, impurity decrease.

Random Forest: Idea



1998, Random Subspace Method (Tim Kan Ho): ensemble-based decision trees with **random feature selection** per tree.

2001, Random Forest Method (Leo Breiman): A bagging-based ensemble of decision trees with **random feature selection** at each tree split. (Trademark random forest)

https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

Random Forest

Random feature selection ("modern"):

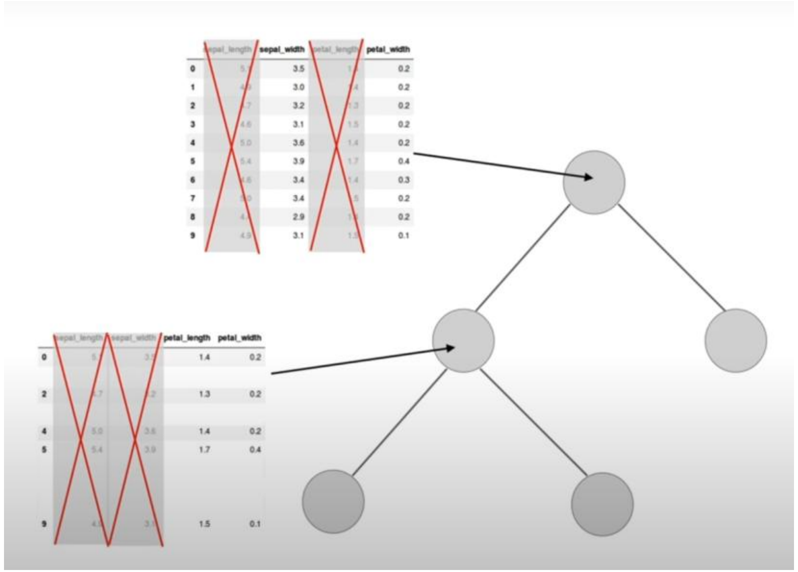
For one decision tree, at each split (tree node), the model can only see a subset of the features.

[Breiman L., "Random Forests", Machine Learning 45. 1 (2001) 5-32](#)

Number of features per per node → hyperparameter

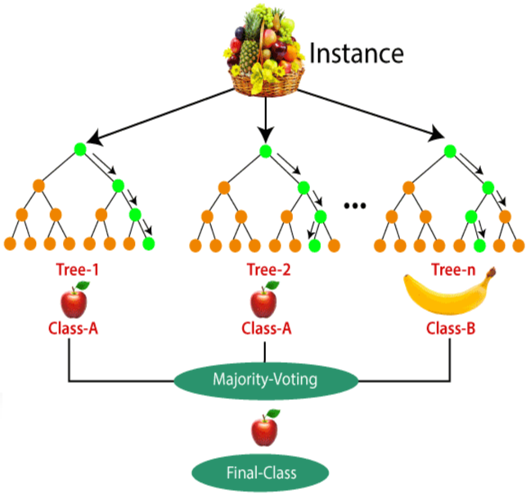
Usually:
#features ~ $\log_2 m + 1$
m= # of input features

E.g:
m=16,
#features = 5
16 input features
5 features /node



	~~sepal_length~~	sepal_width	~~petal_length~~	petal_width
0	~~5.1~~	3.5	~~1.4~~	0.2
1	~~4.9~~	3.0	~~1.4~~	0.2
2	~~4.7~~	3.2	~~1.3~~	0.2
3	~~4.6~~	3.1	~~1.5~~	0.2
4	~~5.0~~	3.6	~~1.4~~	0.2
5	~~5.4~~	3.9	~~1.7~~	0.4
6	~~4.6~~	3.4	~~1.4~~	0.3
7	~~5.0~~	3.4	~~1.5~~	0.2
8	~~4.4~~	2.9	~~1.4~~	0.2
9	~~4.9~~	3.1	~~1.5~~	0.1

	~~sepal_length~~	~~sepal_width~~	petal_length	petal_width
0	~~5.1~~	~~3.5~~	1.4	0.2
2	~~4.7~~	~~3.2~~	1.3	0.2
4	~~5.0~~	~~3.6~~	1.4	0.2
5	~~5.4~~	~~3.9~~	1.7	0.4
9	~~4.9~~	~~3.1~~	1.5	0.1



Note: Decision trees with bagging will always have THE SAME subset of feature at each split!

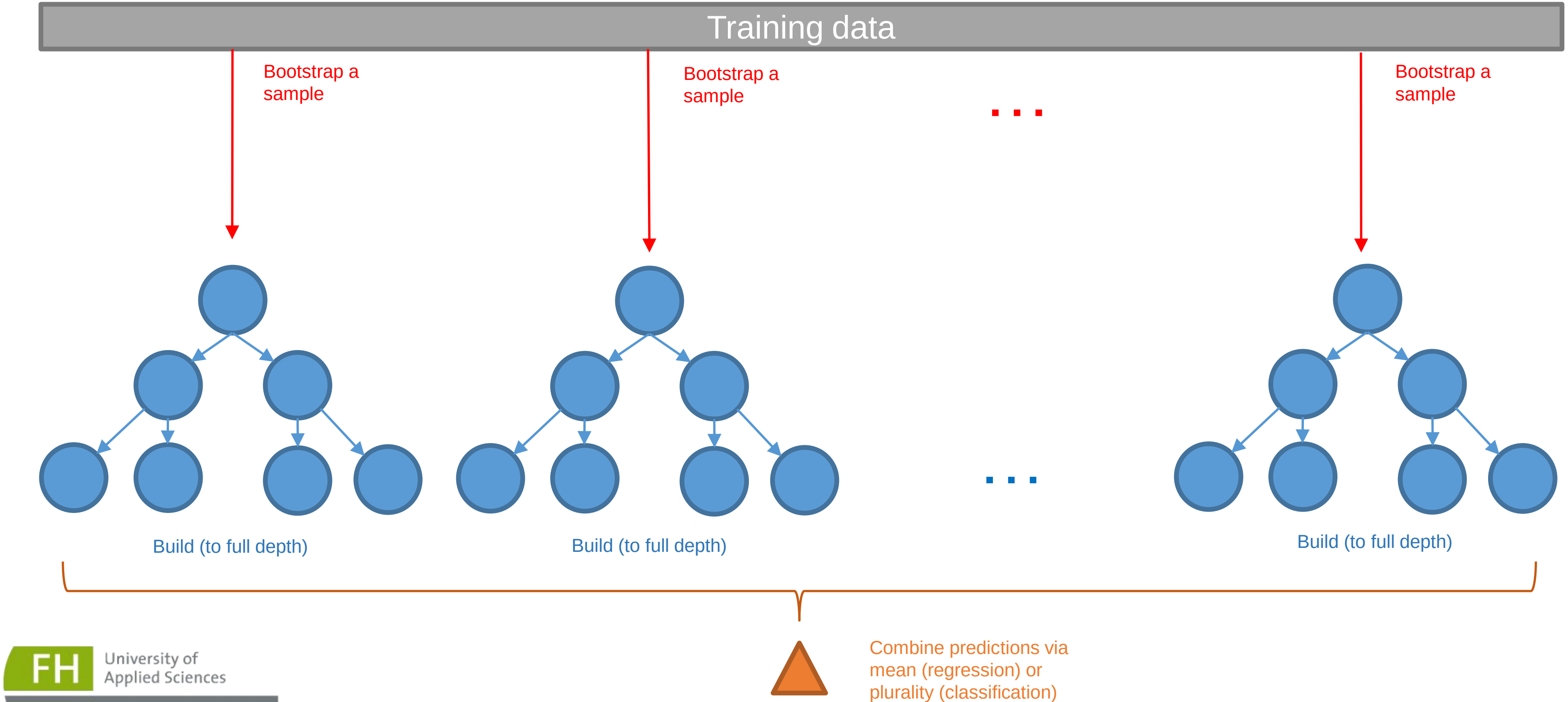
FH University of Applied Sciences TECHNIKUM WIEN

Random Forest

The ensemble method

- **Base learners:** Unrestricted or minimally restricted **decision trees**.
- **Bagging**: Each decision tree is trained on a bootstrapped subset of the observations.
- **Random subspaces**: At each node during tree-building, the splitting criterion evaluates only a randomized subset of the available features (generalization of random patches).
- Prediction: **Plurality voting** (classification) or mean of predictions (regression).

An Ensemble of Bagged Decision Trees



Training data

Bootstrap a sample

Bootstrap a sample

Bootstrap a sample

. . .

. . .

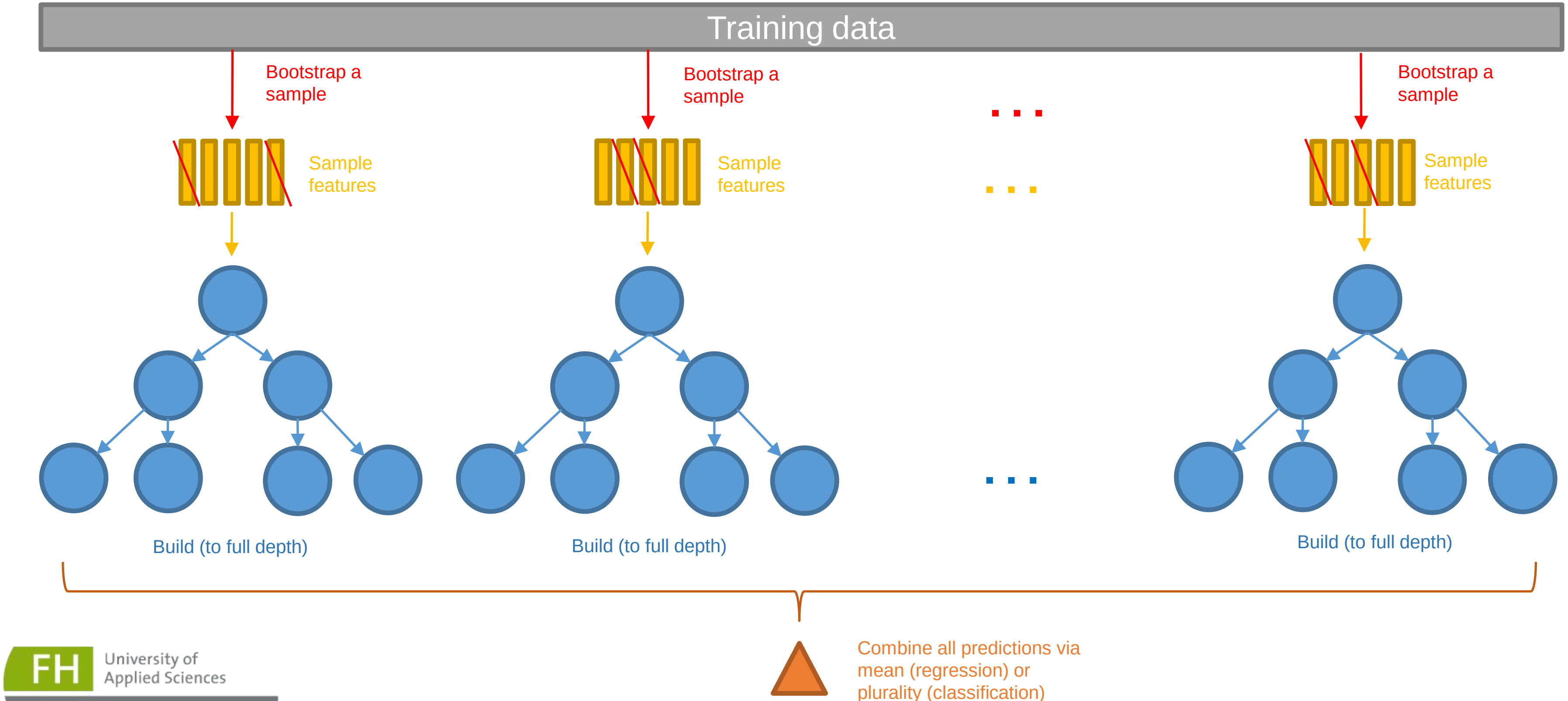
Build (to full depth)

Build (to full depth)

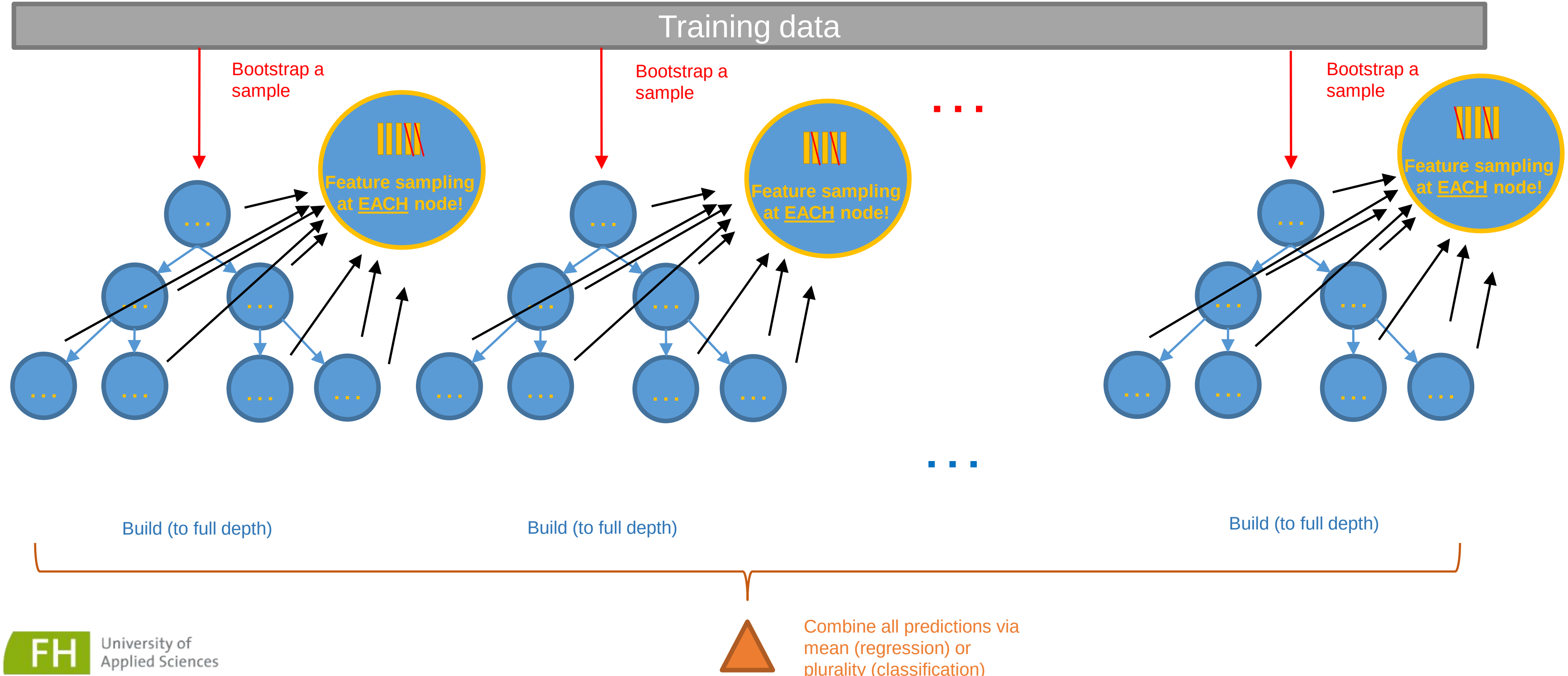
Build (to full depth)

Combine predictions via mean (regression) or plurality (classification)

An Ensemble of Random Patches Decision Trees



A Random Forest



Training data

Bootstrap a sample

Feature sampling at EACH node!

Bootstrap a sample

Feature sampling at EACH node!

. . .

Bootstrap a sample

Feature sampling at EACH node!

. . .

Build (to full depth)

Build (to full depth)

Build (to full depth)

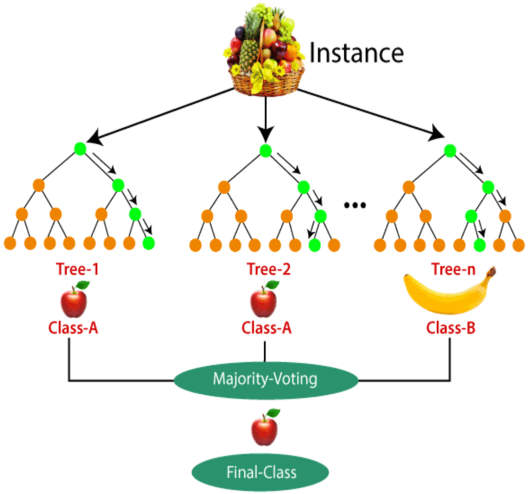
Combine all predictions via mean (regression) or plurality (classification)

Random Forest: Interpretation

Ensemble of Trees: similarity among trees (models) reduces *strength* of ensemble model. Trees that are too different might not contain a good description of data.

Tradeoff between similarity of trees and quality of each individual tree.

Bootstrap Sampling: Each tree sees a slightly different training set.
Random Feature Subsets: Reduces correlation between trees.
Voting/Averaging: Aggregates diverse predictions for robustness.



<u>Out-of-Bag (OOB) samples</u>: *B*ecause each decision tree in RF is trained on a **bootstrap sample**, **about 1/3 of the data points are left out** of each tree's training set, providing a natural **cross-validation** method to test model performance.

FH University of Applied Sciences TECHNIKUM WIEN

Bagging: Recap

Bagging and pasting:

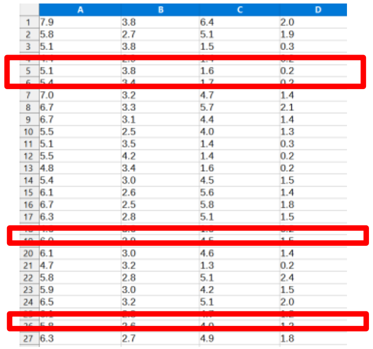
- Resampling from data (observations).

Random Subspaces:

- Resampling features (columns).
- Sometimes called “feature bagging”.

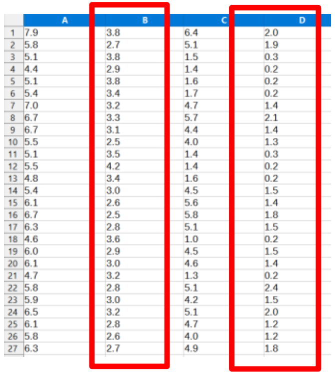
Random Patches:

- Resampling from data and features (observations and features).
- Sometimes called “random subspace plus bagging”.



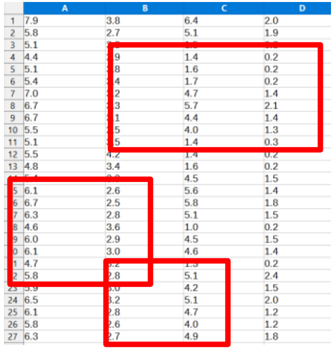
	A	B	C	D
1	7.9	3.8	6.4	2.0
2	5.8	2.7	5.1	1.9
3	5.1	3.8	1.5	0.3
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
5	5.1	3.8	1.6	0.2
6	5.4	3.4	1.7	0.2
7	7.0	3.2	4.7	1.4
8	6.7	3.3	5.7	2.1
9	6.7	3.1	4.4	1.4
10	5.5	2.5	4.0	1.3
11	5.1	3.5	1.4	0.3
12	5.5	4.2	1.4	0.2
13	4.8	3.4	1.6	0.2
14	5.4	3.0	4.5	1.5
15	6.1	2.6	5.6	1.4
16	6.7	2.5	5.8	1.8
17	6.3	2.8	5.1	1.5
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	6.0	2.9	4.5	1.5
20	6.1	3.0	4.6	1.4
21	4.7	3.2	1.3	0.2
22	5.8	2.8	5.1	2.4
23	5.9	3.0	4.2	1.5
24	6.5	3.2	5.1	2.0
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	5.8	2.6	4.0	1.2
27	6.3	2.7	4.9	1.8

Bagging



	A	B	C	D
1	7.9	3.8	6.4	2.0
2	5.8	2.7	5.1	1.9
3	5.1	3.8	1.5	0.3
4	4.4	2.9	1.4	0.2
5	5.1	3.8	1.6	0.2
6	5.4	3.4	1.7	0.2
7	7.0	3.2	4.7	1.4
8	6.7	3.3	5.7	2.1
9	6.7	3.1	4.4	1.4
10	5.5	2.5	4.0	1.3
11	5.1	3.5	1.4	0.3
12	5.5	4.2	1.4	0.2
13	4.8	3.4	1.6	0.2
14	5.4	3.0	4.5	1.5
15	6.1	2.6	5.6	1.4
16	6.7	2.5	5.8	1.8
17	6.3	2.8	5.1	1.5
18	4.6	3.6	1.0	0.2
19	6.0	2.9	4.5	1.5
20	6.1	3.0	4.6	1.4
21	4.7	3.2	1.3	0.2
22	5.8	2.8	5.1	2.4
23	5.9	3.0	4.2	1.5
24	6.5	3.2	5.1	2.0
25	6.1	2.8	4.7	1.2
26	5.8	2.6	4.0	1.2
27	6.3	2.7	4.9	1.8

Random subspace



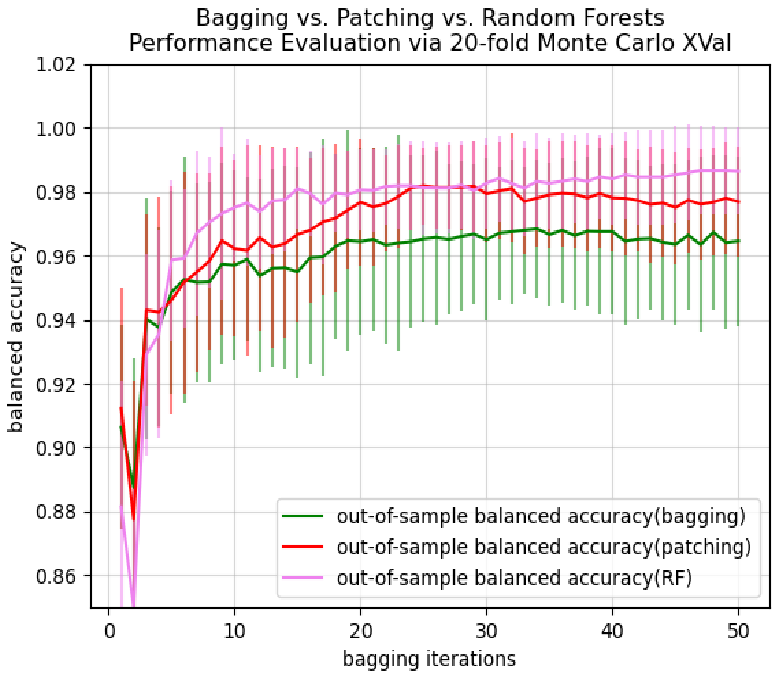
	A	B	C	D
1	7.9	3.8	6.4	2.0
2	5.8	2.7	5.1	1.9
3	5.1	[illegible]	[illegible]	[illegible]
4	4.4	9	1.4	0.2
5	5.1	8	1.6	0.2
6	5.4	4	1.7	0.2
7	7.0	2	4.7	1.4
8	6.7	3	5.7	2.1
9	6.7	1	4.4	1.4
10	5.5	5	4.0	1.3
11	5.1	5	1.4	0.3
12	5.5	4.2	1.4	0.2
13	4.8	3.4	1.6	0.2
[illegible]	[illegible]	[illegible]	4.5	1.5
5	6.1	2.6	5.6	1.4
6	6.7	2.5	5.8	1.8
7	6.3	2.8	5.1	1.5
8	4.6	3.6	1.0	0.2
9	6.0	2.9	4.5	1.5
0	6.1	3.0	4.6	1.4
1	4.7	[illegible]	[illegible]	0.2
2	5.8	2.8	5.1	2.4
[illegible]	[illegible]	[illegible]	4.2	1.5
24	6.5	3.2	5.1	2.0
25	6.1	2.8	4.7	1.2
26	5.8	2.6	4.0	1.2
27	6.3	2.7	4.9	1.8

Random patch

These illustrations are very simplified.
Random choice is key in resampling.

FH University of Applied Sciences
TECHNIKUM
WIEN

Random Forests vs. Random Patches DTs vs. Bagging DTs



- **Random forests gain most** from large ensemble sizes.
- Reason: The **increased base learner diversity** due to more randomization (feature sampling at every split).

Bagging DTs:

- better trees,
- more similar trees

Random Forest

- weaker trees,
- more diverse trees

Quiz

1. What is the main purpose of using a random subset of features at each split in Random Forest?

a) To reduce training time
b) To increase tree depth
c) To decorrelate trees
d) To reduce bias

2. In Random Forest regression, which of the following statements is true regarding out-of-bag (OOB) error?

a) It is computed using a separate test set
b) It underestimates the generalization error
c) It provides a cross-validated estimate of model performance
d) It is not suitable for high-dimensional datasets

Random Forests: Hyperparameters (in SciKit-Learn)

- **Sklearn.ensemble.RandomForestClassifier** implements the classical RF.
- **n_estimators**: Number of trees in ensemble. Performance gains level off. To find the best setting tuning is needed. More trees increase prediction time.
- **max_features**: Number of features to be sampled at each split. **sqrt** and **log2** are common rules of thumb.
- **Hyperparameters inherited from decision trees**:
  Tree depth, tree size, minimal number of observations per node.
- **Hyperparameter for class weighting**: This is convenient to handle imbalanced datasets.

sklearn.ensemble.RandomForestClassifier — scikit-learn 0.24.2 documentation

Random Forests: Pros & Cons

- Random forests are **easy to train: work well without needing much hyperparameter tuning.**
- Scales well for **moderately large data** and **large amounts of features.**
- **Good baseline model:** competitive performance and stable results.
- Handles **missing values.**
- **Feature importance** can be investigated.
- Random forests have been **combined with neural models and deep learning** (e.g. Kontschieder et al. 2016).

- Can be **slow during test-time** (prediction) if many trees are used.
- Can be **hard to interpret.**
- **Not ideal for extrapolation** of data (unlike linear) regression, which uses existing observations to estimate values beyond the observation range).
- Most applications of random forest are to classification problems. **Not good with sparse data**.
- Unproductive splits may occur.

FH University of Applied Sciences TECHNIKUM WIEN

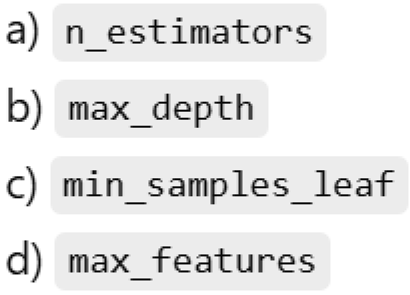
Takeaway

- Random forests are ensembles of decision trees combining bagging and random selection of features (at each tree node) to construct an ensemble of decision trees with lower variance than the constituent trees.
- Random forests perform very well, especially for classical tabular data.
- Often a good baseline model.

Next: Boosting

Quiz

3. In a Random Forest classifier, which parameter most directly controls the diversity of the trees?



a) `n_estimators`

b) `max_depth`

c) `min_samples_leaf`

d) `max_features`

4. Suppose you are working with a healthcare dataset to predict disease presence based on 50 biomarkers. Why might Random Forest be a better choice than Logistic Regression?

a) It is more interpretable
b) It naturally handles feature interactions and non-linearity
c) It guarantees a higher accuracy
d) It doesn’t require labeled data

Assignment: Random Forests

a) Explain RFs by pseudocode and/or visualizations.

– Add RFs to the comparison table of the previous assignment and compare it to other ensemble methods (if you did not do the previous assignment you are allowed to copy the table of the previous assignment from a fellow student).

– Use self-made images or even hand drawings (of which you take a photo). Use self-written explanations. Do not copy from the lecture slides or the internet (neither text nor images).

b) Implement a performance evaluation between kNN, trees, bagged trees, patched trees, and random forests.

You can use as many libraries as you want. No need for self-implementation this time.

Run the implementation on 5 datasets of your choice.

For each method and dataset, show the accuracy (mean and standard deviation) over 10 splits into training and test data. Do this in an interpretable way, e.g. using barplots bar*(x_pos, your_acc_mean_values, yerr=your_ standard_deviation) in 6*5=30 bars*.

References

- Géron A. (2017): Hands-On Machine Learning with Scikit-Learn & Tensorflow. – O‘Reilly.
- James G., Witten D., Hastie T., Tibshirani R. (2017): An introduction to Statistical Learning. – Springer.
- Kuhn M., Johnson K. (2016): Applied Predictive Modeling. – Springer.
- Berk R. (2016): Statistical Learning from a Regression Perspective. – Springer.