AI & Robotics

Data & Optimization

AI & ROBOTICS LAB

Goals

The **junior-colleague**

- can explain the link between data & optimization
- can describe the reasons for understanding your data
- can explain how to prepare and clean a dataset for training a Machine Learning model (categorical variables, dates, missing values)
- can explain one-hot encoding
- can explain the difference between parameters and hyperparameters
- can list and describe 3 ways to tune the hyperparameters of a Machine Learning model
- can list and describe 3 hyperparameters of a Random Forest
- can explain how to acquire the out-of-bag score for a Random Forest and why it's useful
- can explain why feature importance is important in the context of Machine Learning
- can explain the concept of Occam's Razor in the context of Machine Learning
- can describe ways to speed up the training process of a Random Forest

Intro

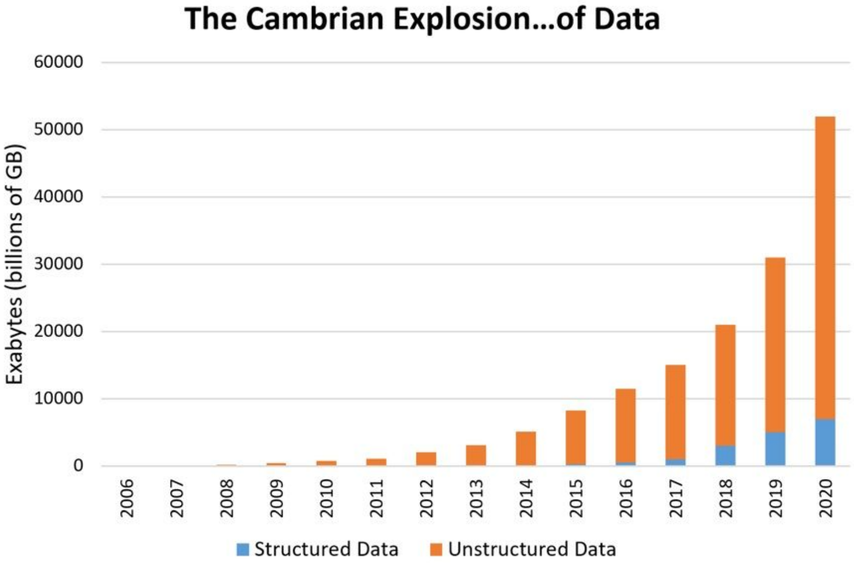
- Data & Optimization are inextricably linked
- Understanding your data gives you insights into optimization

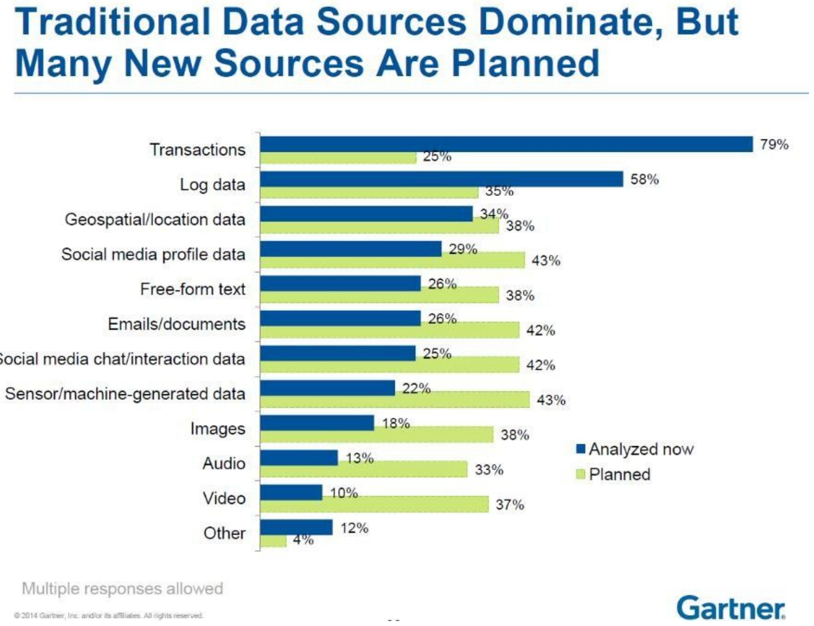
Data

https://xkcd.com/1838/

Data

… is the key ingredient for all Machine Learning problems!

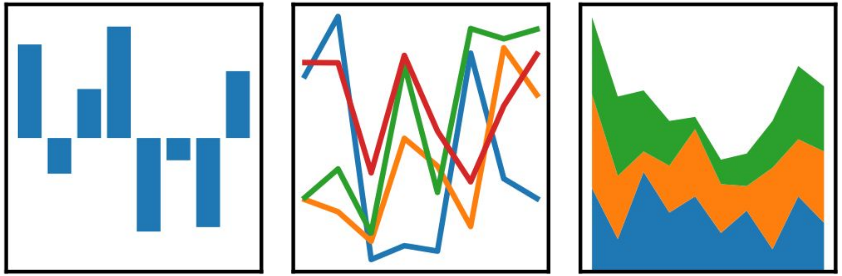




Processing data

Pandas

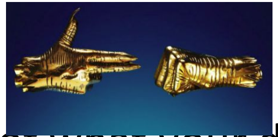
- Python library for data manipulation and analysis
- Highly optimized for performance
- Seamless integration with *numpy*



Understanding data

- Important to look at your data
    - Understand how it’s stored
    - Understand the format and types of values

  => But! Don’t get biased



- Important to get an idea of what your data is about

  => find correlations

  => look for ways to optimize your model

- Important to note what metric is being used for a project

Cleaning and preparing data

Categorical variables

- Categorical variables usually get stored as strings in pandas

  => inefficient

  => not numerical (impossible to process for most ML algorithms)
- Convert to “categories”
  - Will be treated as numerical data internally
  - Optional ordering

=> Make sure you use the same procedure for your different datasets, otherwise the categories won’t match

Cleaning and preparing data

	A	B	C	D	E	F	G
1	Plot: 3						
2	Date collected	Species	Sex	Weight	Month	Day	Year
3	1/8	PF	M	7	=MONTH(A3)	=DAY(A3)	=YEAR(A3)
4	2/18	OT	M	24	2	18	2015
5	2/19	OT	F	23	2	19	2015
6	3/11	NA	M	232	3	11	2015
7	3/11	OT	F	22	3	11	2015
8	3/11	OT	M	26	3	11	2015
9	3/11	PF	M	8	3	11	2015
10	4/8	NA	F		4	8	2015
11	5/6				5	6	2015
12	5/18	NA	F	182	5	18	2015
13	6/9	OT	F	29	6	9	2015
14	7/8	NA	F	115	7	8	2015
15	7/8	NA	M	190	7	8	2015

“Dates”

- Contain lots of information
  - Day, Month, Year, DayOfWeek
  - Seasonal variation
  - Temporal continuity
- Separate data type in pandas

Cleaning and preparing data

Handling missing values

- Missing categorical values get a numeric code
- Missing continuous values get assigned the median (or other measure) for that feature
- Optionally create extra columns (features) for missing categorical data

=> Make sure you use the same procedure for your different datasets, otherwise the categories won't match

Cleaning and preparing data

One-hot encoding

- Necessary for some ML algorithms
- Random Forests don’t require One-hot encoding

Color
Red
Red
Yellow
Green
Yellow

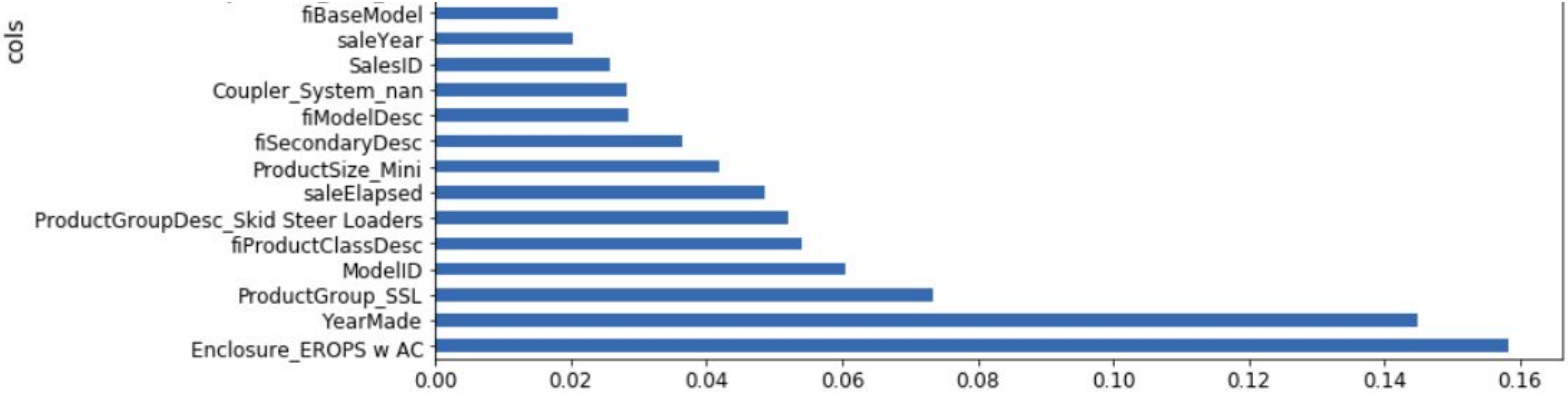
Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Cleaning and preparing data

One-hot encoding

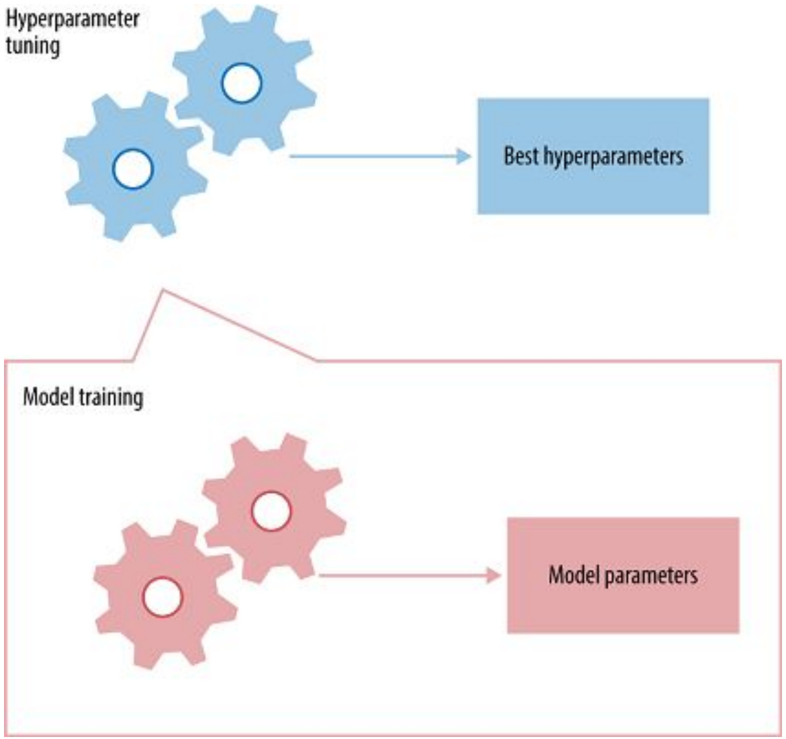
Random Forests don't require One-hot encoding

=> but, can be useful for new insights into your data



Optimization

Optimization: parameters vs. hyperparameters



- Parameters
  - Learned during training
- Hyperparameters
  - Set by the data scientist before training
  - **Validation set!!**

Optimization: hyperparameter tuning

How?

- Trial and error:
  => Use metrics and graphs to estimate good hyperparameters
  => Gives you good insights into the data
- Grid search
  => Just try every possible combination of parameters and values
  => Guaranteed to find the optimal hyperparameters
- Random search
  => Try random parameters and values
  => Not guaranteed to find the optimal hyperparameters

Depending on the size of the problem, tuning can require ***<u>A LOT</u>*** of computational resources

Optimization: Random Forests

- Scikit-learn hyperparameters
    - n_estimators
    - max_depth
    - min_samples_split
    - min_samples_leaf
    - min_weight_fraction_leaf
    - max_features
    - max_leaf_nodes
    - min_impurity_decrease
    - min_impurity_split
    - …
- Check the documentation for details

Optimization: Random Forests

n_estimators

- Number of trees
- Incrementally increase the amount of trees
- Use graphs + R^2 to see at which point it stops helping
- Use fewer trees when experimenting

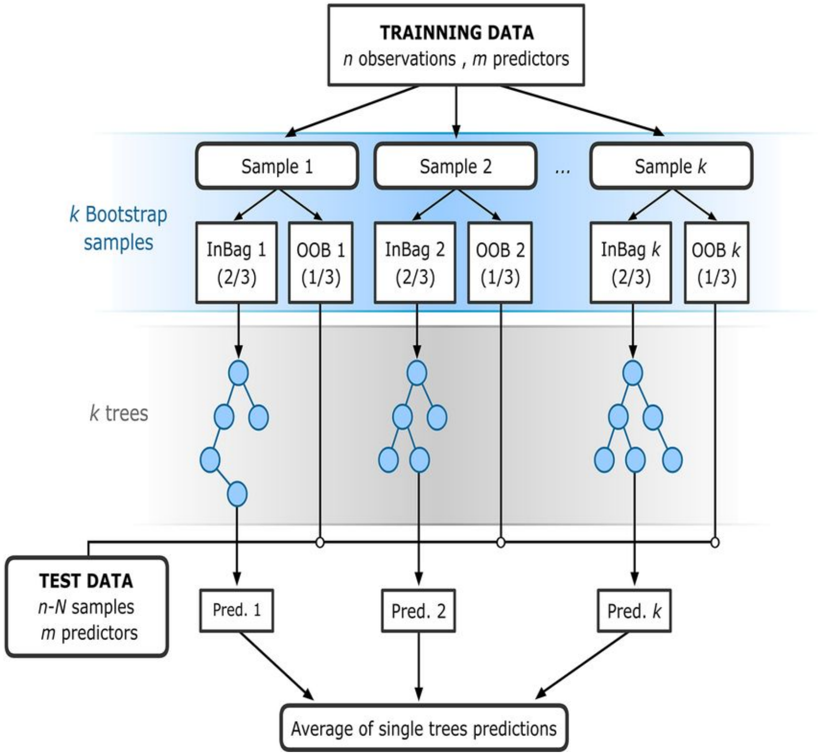
min_samples_leaf

- The minimum amount of samples in each leaf node
- Reduces the chance of overfitting

max_features

- The number of features to choose a splitpoint from at each node in a tree
- Increases randomness

Optimization: Random Forests - Out of bag

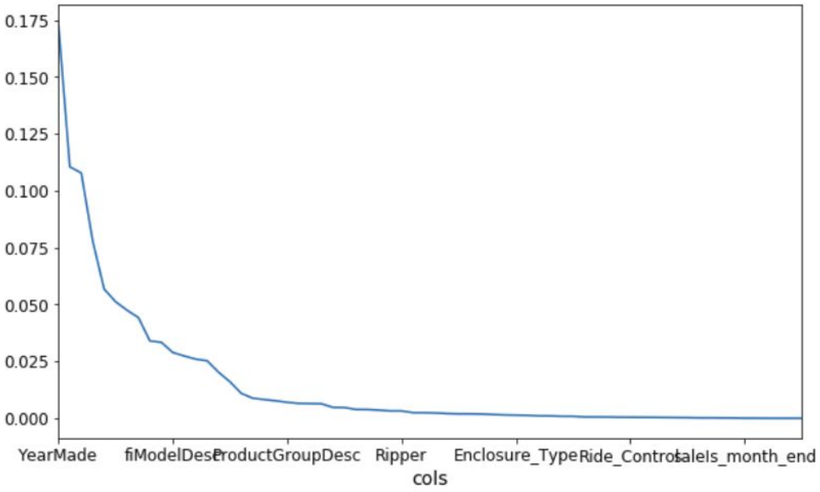


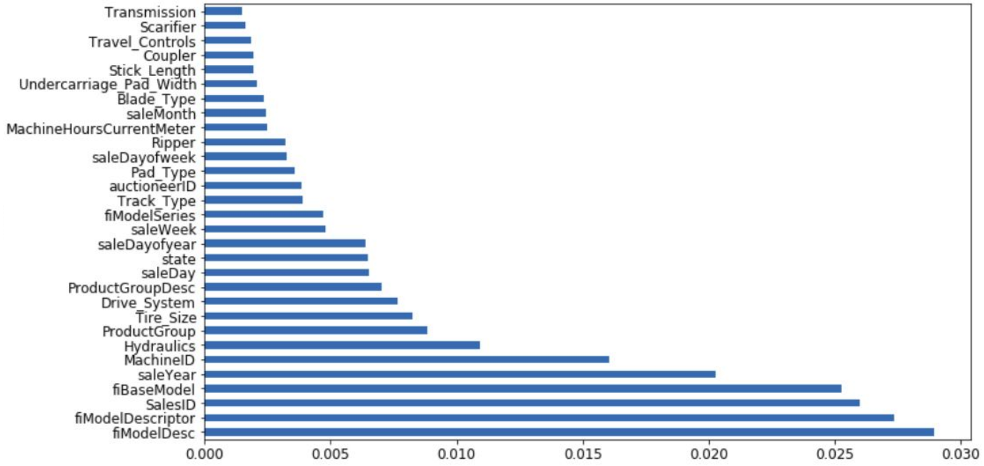
- What if our dataset is too small?
- For every tree: use the rows not used to train the model as a validation set

  => different validation set for each tree
- Average over all trees where that row was not used for training
- The more trees you have, the higher the chance of each sample appearing in valid set of multiple trees
- Less useful than a true validation set
- Is statistically significant

  => good overfitting check

Optimization: Feature Importance





Optimization: Feature Importance

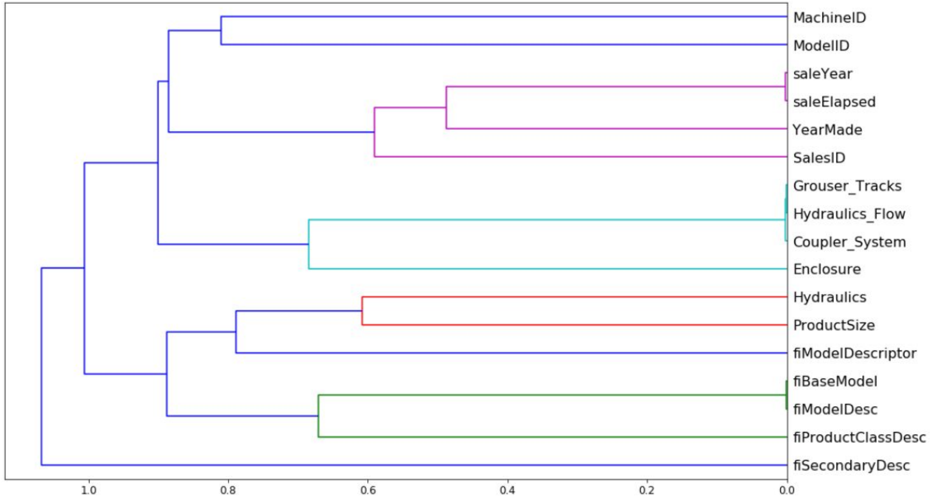
- One of the most important optimization techniques
- Remove less important features, so the important ones stand out more, because there may have been collinearity
- How to calculate feature importance: Shuffle the rows of that feature, compare the R^2 of this model to the original model

- Remarks:
  - Shuffle method can be misleading if 2 attributes are closely related (correlated)

    => Checking for feature similarity can help
  - The model indicates Feature x is important, yet the client says that it doesn't make sense

    => Client may not understand his/her own data (data leakage)

Optimization: Feature similarity

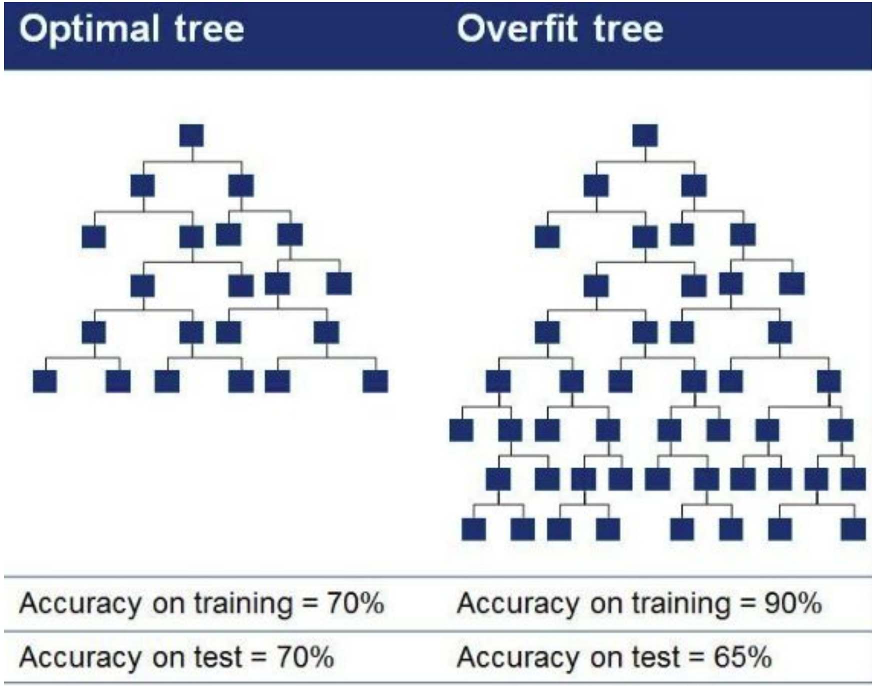


1. Visualize potential correlations (i.e. hierarchical clustering graph)
2. Check how the model behaves when removing correlated features one at a time
3. Remove the features that have a negligible impact

How to measure similarity or correlation?

=> Correlation coefficient (R^2) / Spearman's R (Doesn't really matter what, as long as it measures correlation)

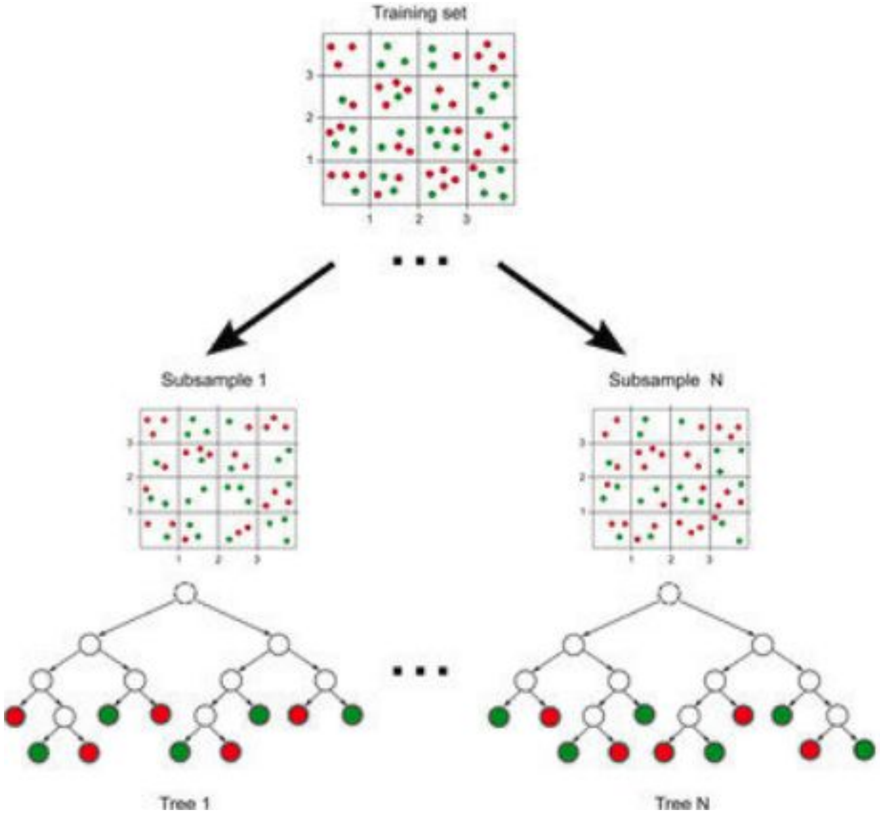
Optimization: Occam's Razor



- Simple models >>> complex models
- Similar principle in science: Occam's Razor
  - "Simpler solutions are more likely to be correct than complex ones"
  - "Select the solution with the fewest assumptions"
- Decision trees & Random Forests

  => more complex trees have higher probability of overfitting the data set

Optimization: training speedup



Subsampling

Pick a different random subset of samples for each tree

=> given enough trees, the model can still see all the data

Optimization: closing remarks

- If something goes wrong in Machine Learning
  - No error message
  - Your model is just slightly less good
- Domain knowledge can help but not necessarily