HOGESCHOOL UTRECHT

Fundamentals of Machine Learning
Week 5: machine learning
Decision trees and Random Forest

Jonas Moons

Check-in

Feedback assignment #3 (LR)

- Note that many steps (splitting data, linear prediction) are 1 line in sklearn (see example Notebooks)
  - Though if you program it yourself for practice, it's fine (but do both to compare)
- It was multiple linear regression, so multiple coefficients
- Keep on using Markdown (headers, introduction, interpretation, steps)

Final assignment tips #1

- You can still submit for feedback!
- Good idea to gather and clean the data straight away
- Consider working in parallel (more fun / insightful):
  - Make a small csv file (+/- 20 rows) with dummy data or hand copied
  - Build your model and clean data in parallel

Final assignment tips #2

- Really visualize your final data. Make a mock file in Excel so you know what you are working towards
- Unit of analysis = one row. Be as precise as possible
  - All rows have to be the same!
  - "When": is this date, year, hour, etc.?
- Preferably use variables that can be used in a ‘real’ scenario
  - E.g. movie rating from duration and actors, but not from critics rating (not there when movie is made)
- From some variables you can extract multiple variables
  - Date -> day of week, working day (y/n), month

Topics

- *k*-nearest neighbor
- Decision tree
- Random Forest

k-nearest neighbor algorithm

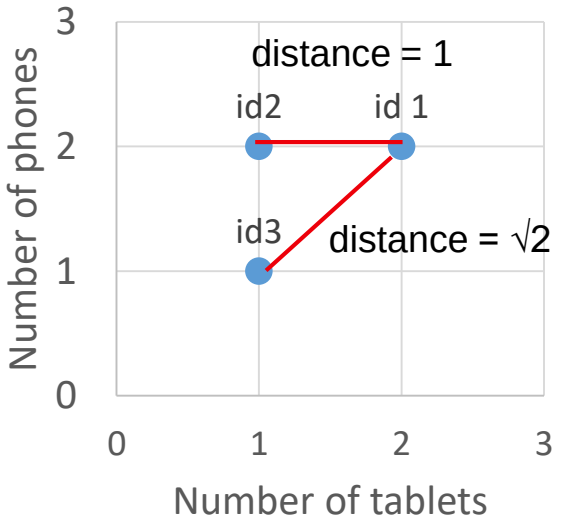
- *k*-nearest neighbor is one of the simplest algorithms in machine learning

- From the data set, pick the *k* nearest neighbors (*k* = 3, 5, 7, etc.) of the individual you want to predict for
  - Classification: pick the most frequent answer (e.g., Ronaldo or Messi)
  - Regression: take the mean of the neighbors (e.g., apps downloaded last year)

Distance

- How do you calculate a ‘distance’ between individuals?
- Take ‘Euclidean distance’
  - A ‘distance’ between two individuals can be calculated for any number of dimensions/variables
- Treat all variables the same
  - *Normalize* all variables so that they are all on the same scale (mean = 0, sd = 1)

Id	nr phones	nr tablets	Apps donwloaded
1	2	2	**50**
2	1	2	**42**
3	1	1	**23**



Evaluation of classification

Confusion matrix:

	Predicted: Not spam	Predicted: Spam	Total
Actual: Not spam	20	10	**30**
Actual: Spam	40	60	**100**
Total	**60**	**70**	**130**

What proportion is correctly predicted?

$$accuracy = \frac{20 + 60}{20 + 40 + 10 + 60} = \frac{80}{130} = 0.62$$

How much of the predicted 'spam' is actually spam?

$$precision\ (spam) = \frac{60}{60 + 10} = \frac{60}{70} = 0.86$$

How much of the real spam is predicted as spam?

$$recall\ (spam) = \frac{60}{40 + 60} = \frac{60}{100} = 0.60$$

Exercise 1: *k*-NN

We are going to predict survivors on the *Titanic*. This is a classic data set in machine learning. So there are plenty of ideas on the web. For an explanation of the variables, see here.

See the Notebook *k-nn* in the *examples* folder for example code (and/or the *cookbook* Notebook). Find the data set in the *exercise* folder (Week 5).

1. Inspect the data set. There are 5 columns which contain 'easy to work with' categorical or numerical (independent) variables. Which are they? Select these.
2. Then, get rid of the rows with empty cells. Why should we do this *after* step 1?
3. Create dummy variables for the categorical variable and add **one** of them (remember that you shouldn't add "redundant" variables in terms of information)
4. Split the data into a training and test set.
5. Train a kNN-algorithm on the data, with $k = 3$
   - Calculate accuracy, precision and recall for survival **on the test set**
6. Try out different settings for k. Which k works best?

Parameter setting

- Every machine learning algorithm has *parameters*
- For instance, with KNN the most important parameter is *k*: the number of neighbors
- You can try different parameter settings to optimize your model (use the test set to evaluate)
- One way to do so is a *grid search*
  - Make a function of your model building, fitting and evaluation
  - Try out with for loops for all combinations of parameters
  - However, leads to overfitting. More detailed discussion here.

Topics

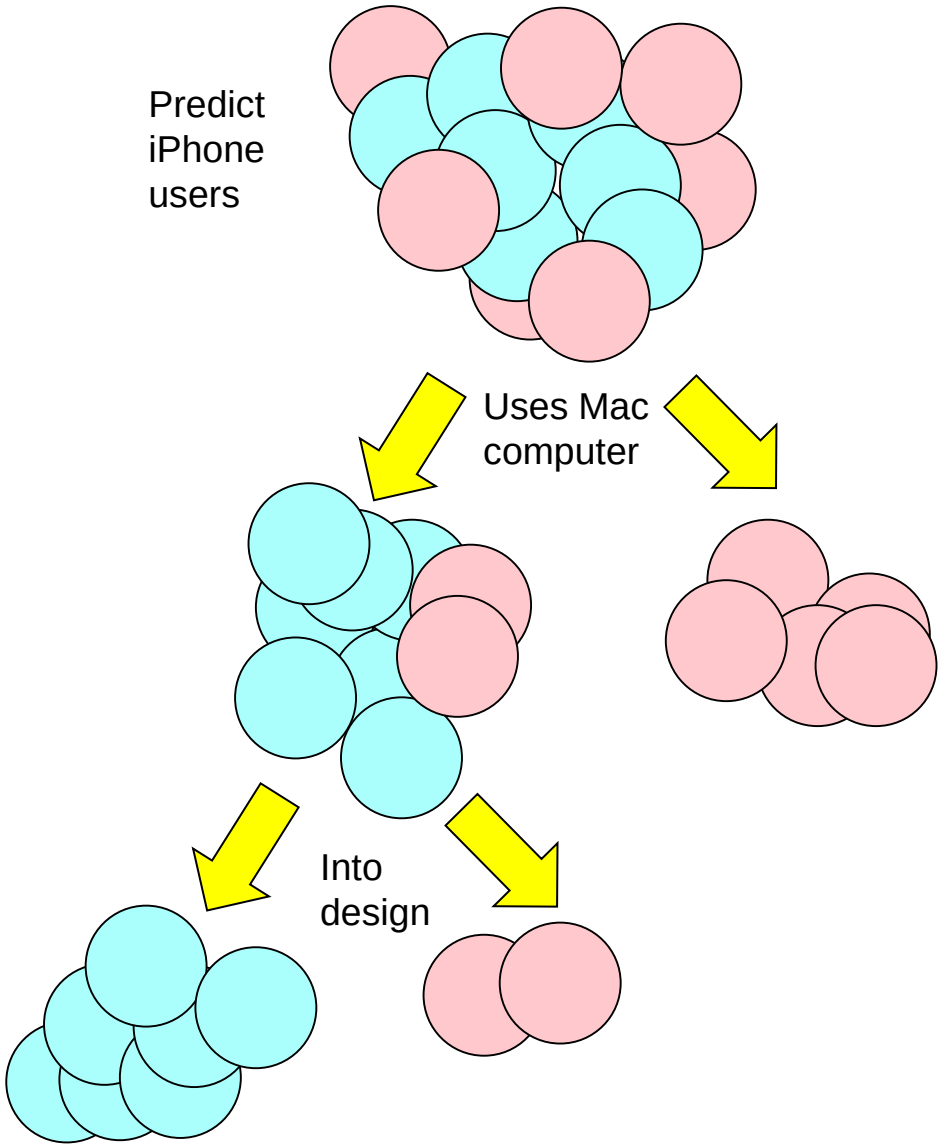
- *k*-nearest neighbor
- Decision tree
- Random Forest

Decision tree: intuition

- I need a volunteer…

Decision tree algorithm

- Choose a Y variable to predict
- At each node of the tree calculate the X variable & cut-off point that produces the optimal split for the Y variable
- Optimal: large variance (difference) between branches, small variance (similarity) within branch
- Stop when you have ‘pure’ leaves



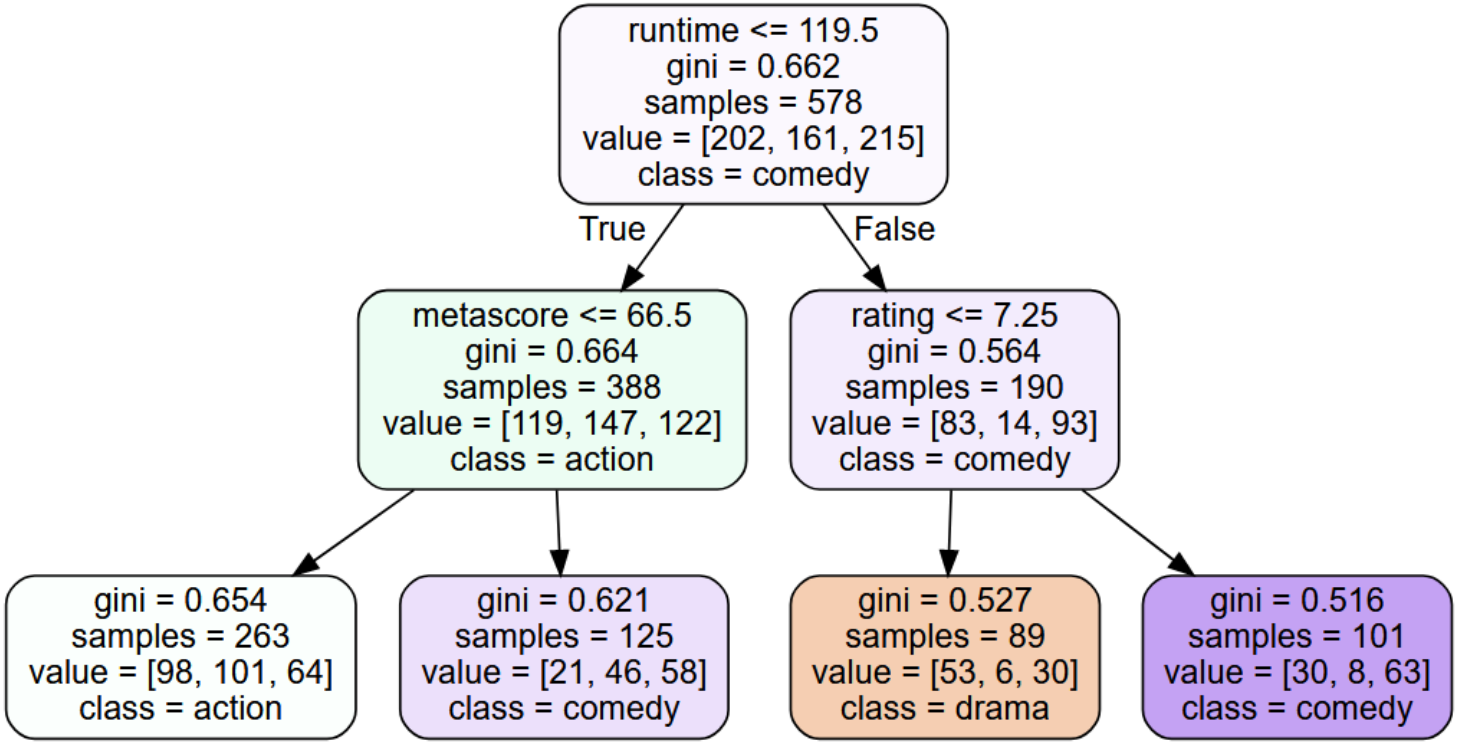
Pros and cons of decision tree

- Advantages
  - Easy to interpret
  - Not a ‘black box’ algorithm
  - When ’pruned’, easy to communicate to other professionals, e.g. clinicians

- Disadvantages
  - Overfits easily
  - Also not a very strong algorithm in terms of prediction

Example

See Notebook *decision_tree* in Examples folder if you want to use this algorithm



value = [drama, action, comedy]

Topics

- *k*-nearest neighbor
- Decision tree
- Random Forest

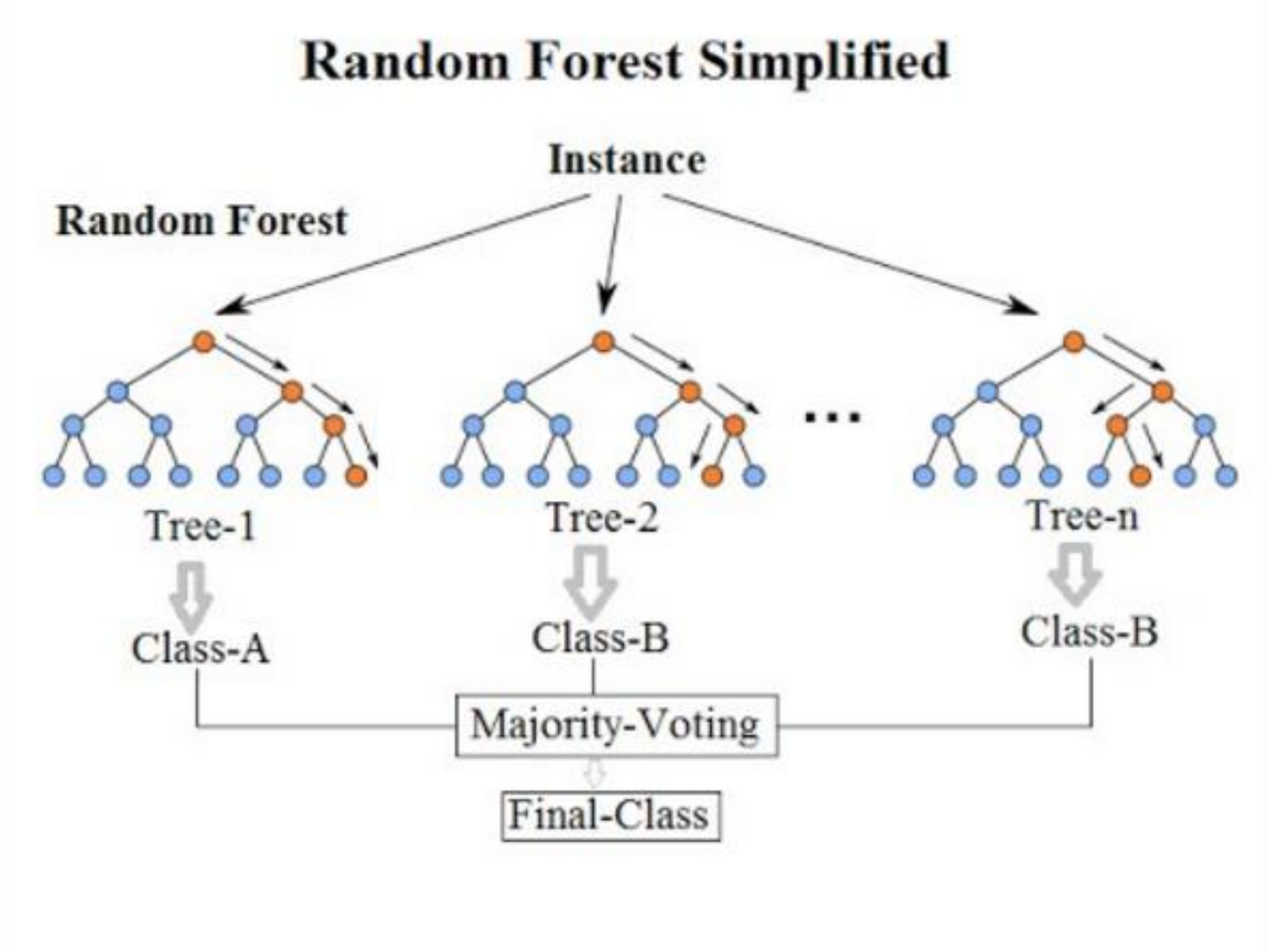
Random Forest

- Random Forest is an extension of decision tree (Ho, 1995; Breiman, 2001)

- Intuition:
  - Grow many different trees
  - Train each on a different sample of the data
  - Let them vote on the case to be predicted

- Random Forest improves accuracy (at the cost of interpretability)

Random Forest



Parameters to tweak

- *n_estimators*: the number of trees grown
- *max_features*: the number of variables that are considered at each split of the tree

Exercise 2

For fun, we're going to train the Random Forest algorithm on a bigger data set (66 MB). The data set includes 285,000 cases of credit card transactions, and 31 (secret) variables. Your task is to detect fraud.

This time, there is no example Notebook. The challenge is to create a Notebook from scratch. Here is the documentation of the object. Remember how to import an object from sk-learn?

1. Split the data into a training and a test set
2. Fit the algorithm using the train data (all variables) to predict *Class* (may take a few seconds)
3. Calculate the precision and recall on the test set for fraud and not-fraud.
4. Do you feel the precision and recall of your model for fraud are acceptable in this situation (assessing transaction fraud risk)? Why (not)? Which is more acceptable to be worse, precision or recall? Why?
5. Experiment with different parameter settings (especially *n_estimators* and *max_features*) and test them against a test set.

Image credit

- Random Forest by Venkata Jagannath (CC-BY-SA)