Machine Learning

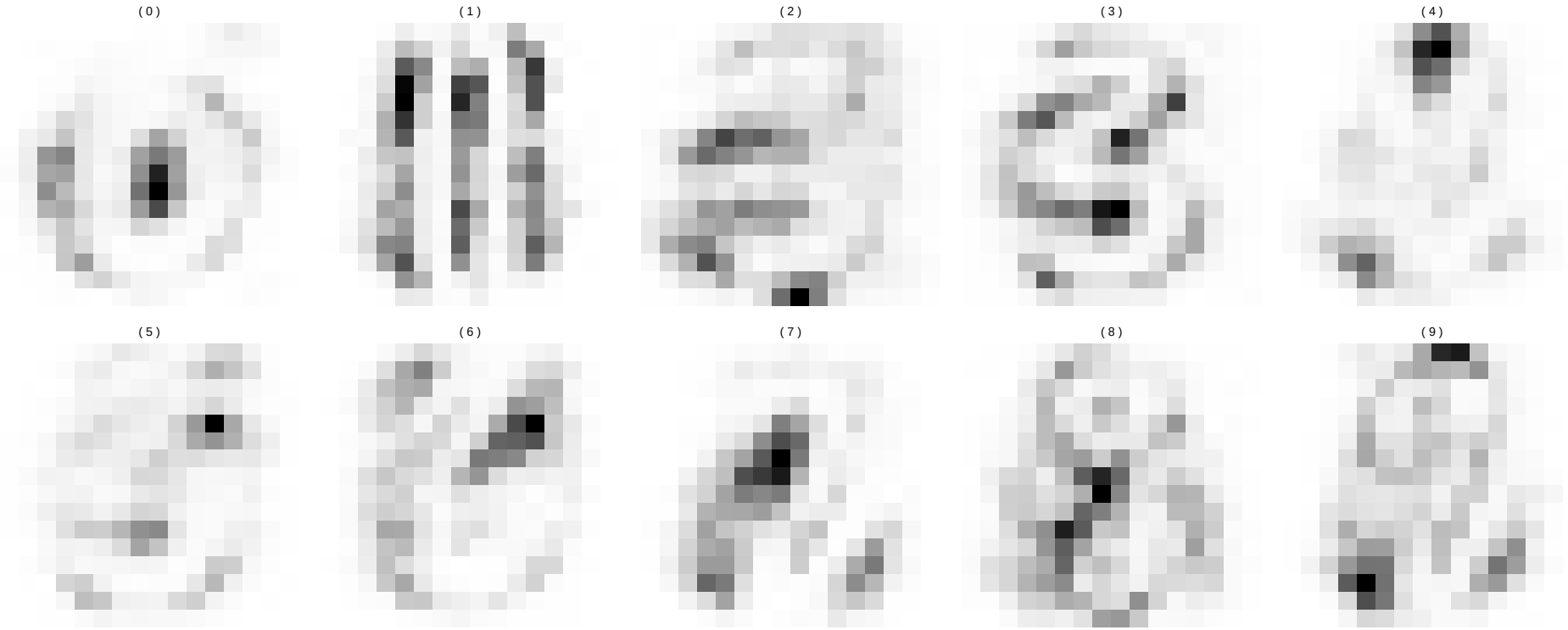
Random forest

Sebastian Engelke

Master in Business Analytics

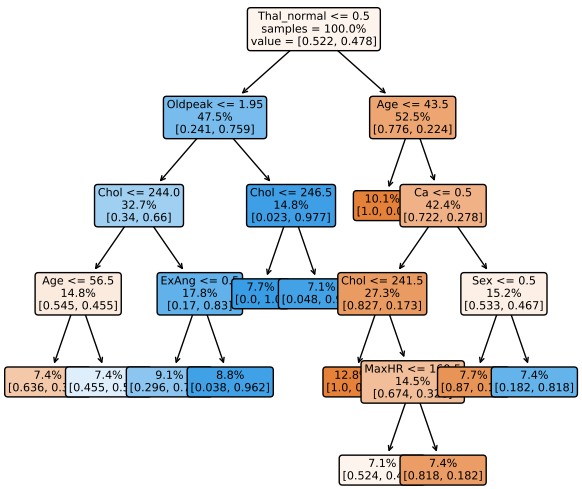
UNIVERSITÉ
DE GENÈVE

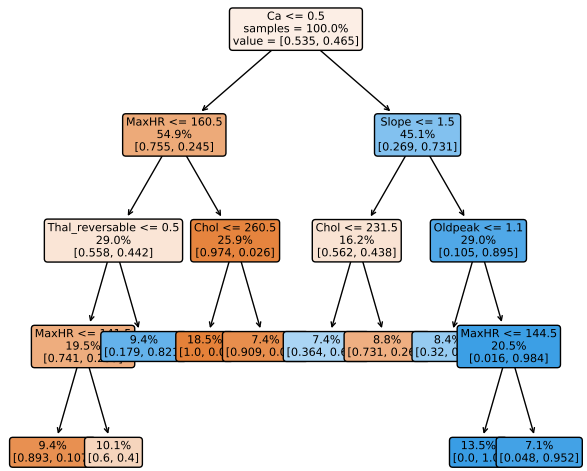
Digit classification: variable importance

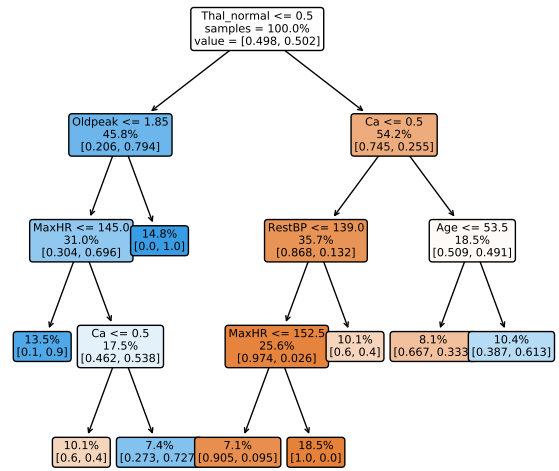


Bagging: correlated trees

- In bagging, the trees grown with each bootstrap data set `share many common` training observations.
- The greedy algorithm thus chooses often the `same predictors` for `early splits`!
- This means that the trees are very similar and the predictions $\widehat{f}^{*b}(x_0)$ for a new input x_0 are very correlated for all $b = 1, \ldots, B$.







Random Forest

- `Random Forest` extends the concept of bagging.
- The idea is to `decorrelate` the trees from the bootstrap data sets by introducting constraints in the growing process.

Random Forest algorithm:

(1) Bootstrap the data B times.

(2) When growing each tree on the bootstrap samples, for each split `randomly choose` m `of the` p `predictors` to be used for the split. This subset of variables changes for each split.

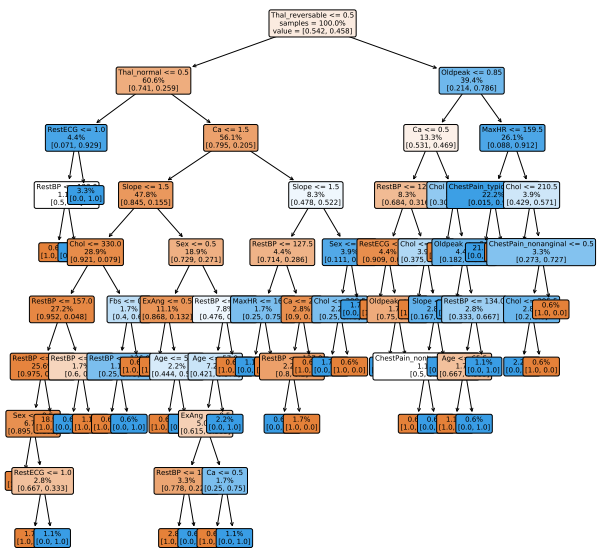
(3) Grow `full unpruned trees` (pure terminal nodes).

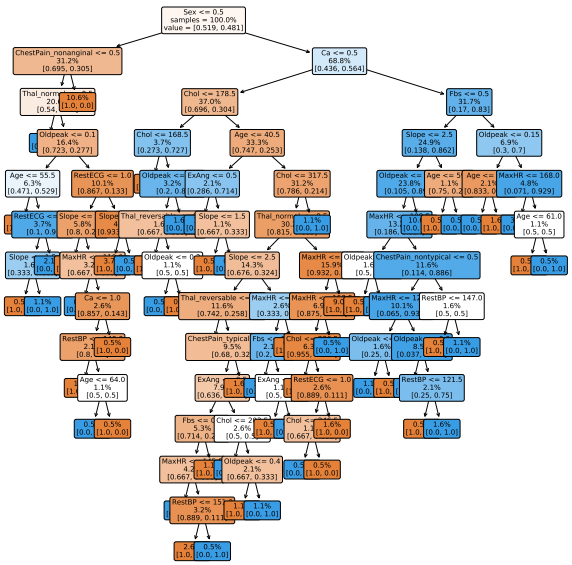
(4) For prediction (as for bagging):
   - `regression`: average the predictions from the B trees;
   - `classification`: majority vote from the B trees.

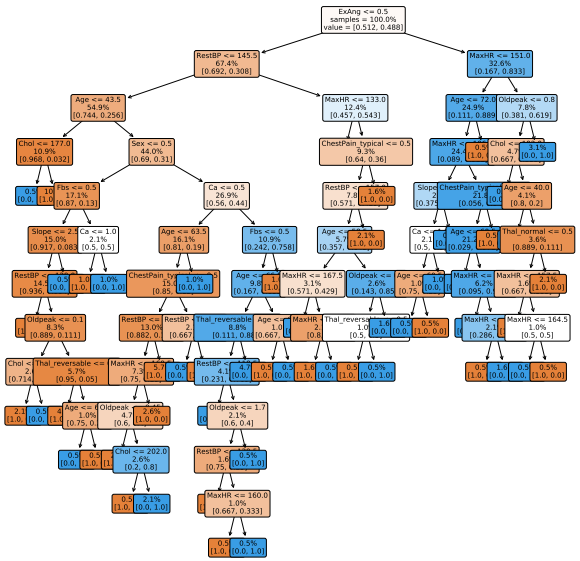
Random Forest

Intuition: selecting a random subset of m variables for each split `avoids too similar trees`.

Choice of m: $m \approx \sqrt{p}$ for classification, and $m \approx p/3$ for regression typically works well in practice.

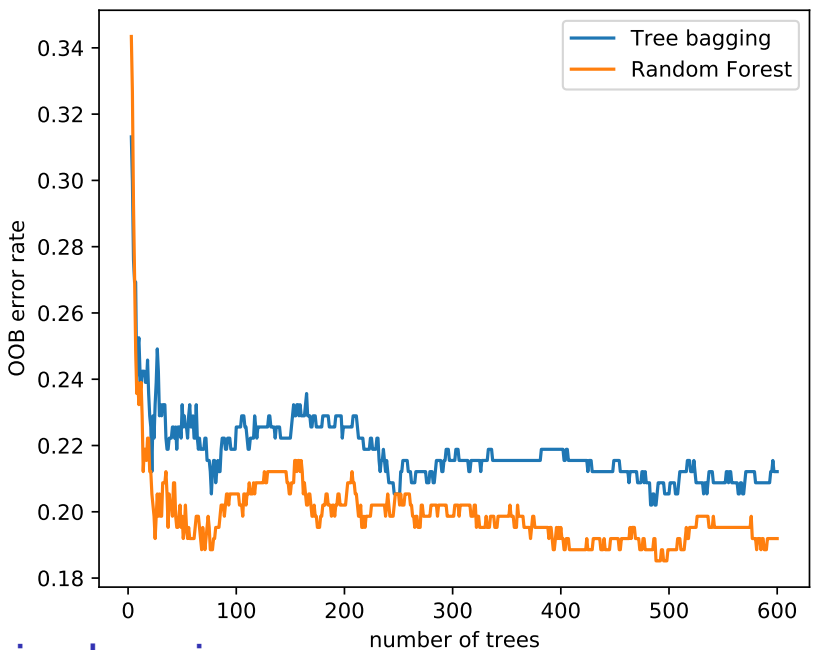


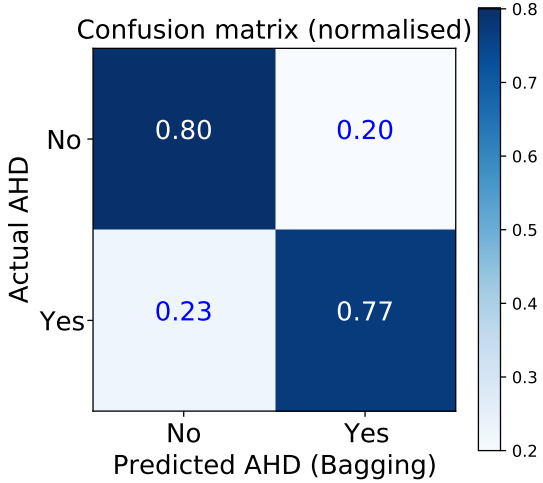


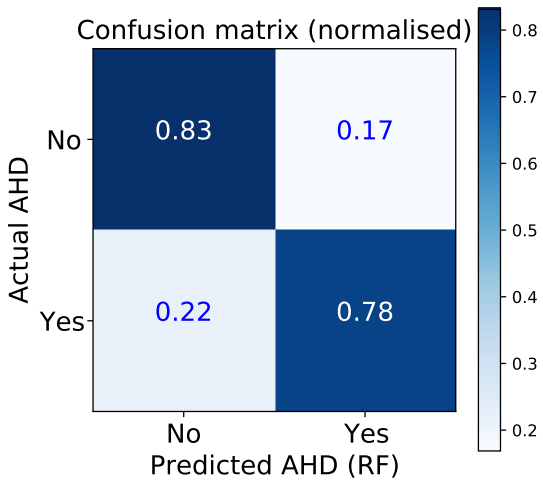


Heart data: bagging and random forest in python

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=500, max_features="sqrt", oob_score=True,
                             warm_start=True, random_state=1)
bagging = RandomForestClassifier(n_estimators=500, max_features=None, oob_score=True,
                             warm_start=True, random_state=1)
oob_rfc, oob_bag = list(), list()
for i in range(3,601):
    rfc.set_params(n_estimators=i)
    rfc.fit(X, y)
    oob_rfc.append(1 - rfc.oob_score_)
    bagging.set_params(n_estimators=i)
    bagging.fit(X, y)
    oob_bag.append(1 - bagging.oob_score_)
plt.plot(np.arange(3,601), oob_bag, label="Tree bagging")
plt.plot(np.arange(3,601), oob_rfc, label="Random Forest")
```

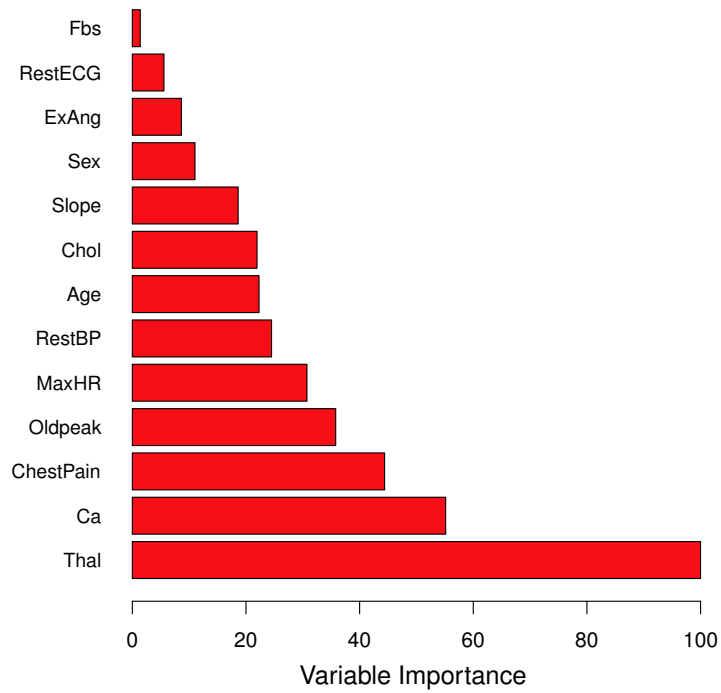






Variable importance

- Bagging and random forest have good predictive power, but we can no longer plot a single tree for interpretation.
- However, we can define an overall summary of variable importance.
- For regression, we record the total amount that the RSS is decreased due to splits over a given predictor X_j, $j = 1, \ldots, p$.
- This can be done by randomly permuting this given predictor in the OOB observations and then measure the increase in prediction error.
- Similarly, classification, we record the total amount that the Gini index/deviance is decreased due to splits over a given predictor.



Random forests and nearest-neighbors

- Random forests can be seen as an adaptive/weighted nearest-neighbors method, where "close" observations are assigned different weights.
- Recall that the trees in random forests are fully grown and unpruned.
- Hence for each tree the prediction of y_0 for a new input x_0 is equal to a response y_i corresponding to one training observation x_i, as in one-NN.
- The "neighbor" x_i might be obtained from a particular path in the feature space and might not correspond to the nearest observation in terms of Euclidean distance.
- Each tree in the forest will predict the value of y_0 as a y_i. The final random forest prediction (for regression) will be of the form

$$\widehat{y}_0 = \sum_{i=1}^{n} w_i y_i,$$

where the weights $w_i \geq 0$ are the fractions of trees where x_i was in the terminal node of x_0.

$\Rightarrow$ Random forests can thus be seen as a smoothing method with complex notion of neighbors.

Pros and cons of random forests

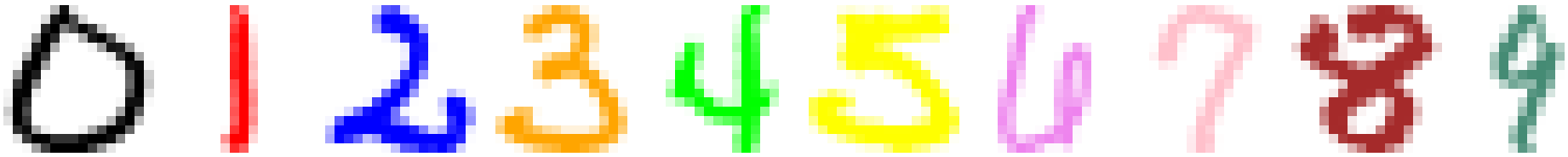
Pros:

- Great predictive performance.
- Almost no tuning needed.
- Out-of-bag error estimate; no need for CV.
- Variable importance measure available.
- Easy to fit using existing functions in R and quite fast.

Cons:

- By bagging trees we lose the interpretability of a single tree.

Digit classification with random forest



- We can update our results on the digit classification.

- linreg: direct linear regression;
- LDA1: LDA based on the values x_i;
- LDA2: LDA based on x_i and x_i^2;
- QDA: different cov. matrix for each class;
- log: logistic regression (multinomial).
- log-ridge: log. reg. with ridge penality.
- log-lasso: log. reg. with lasso penality.
- RF: random forest.

	Training error	Test error
linreg	7.6%	13.1%
LDA1	6.2%	11.5%
LDA2	3.9%	10.2%
QDA	1.8%	13.5%
log	0.01%	11.1%
log-ridge	4.1%	9.0%
log-lasso	2.7%	8.8%
RF	0%	5.9%

Digit classification: variable importance