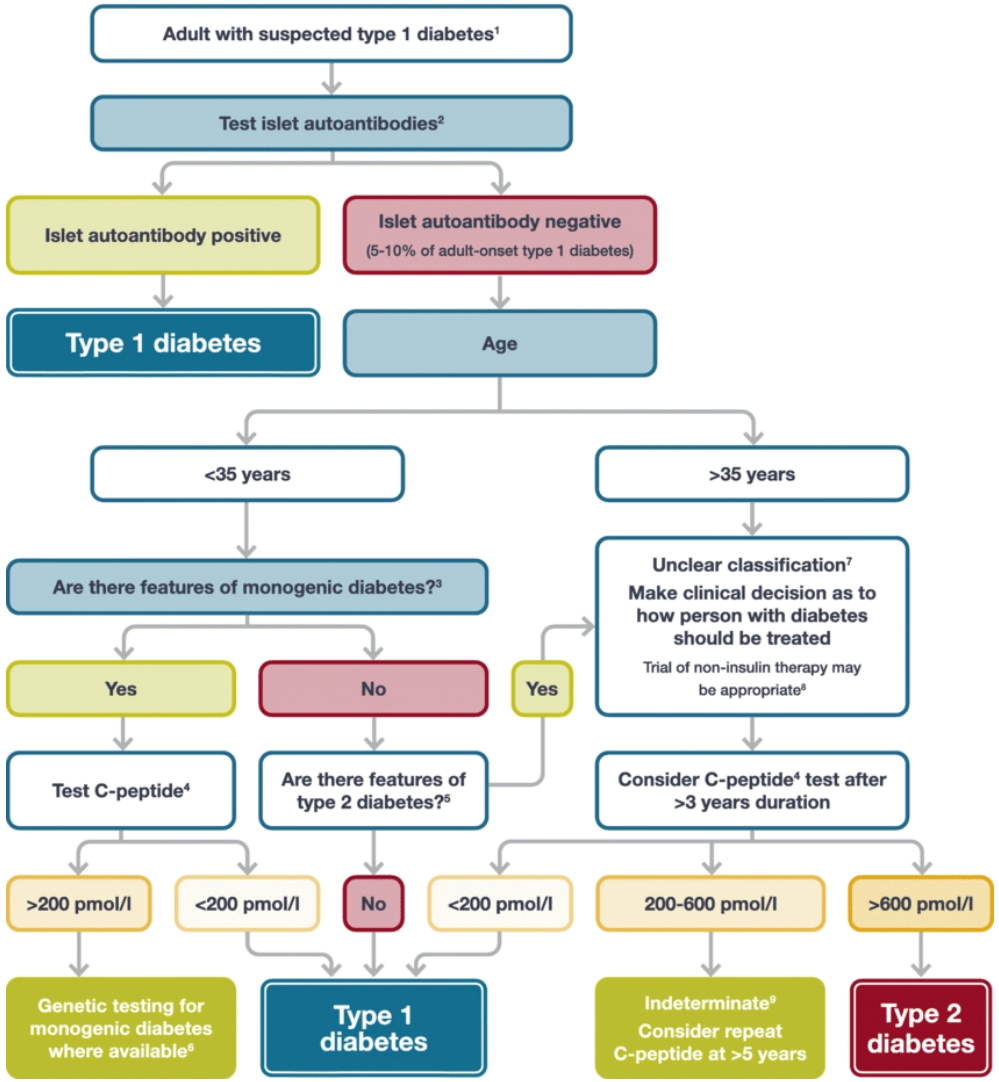
Non-parametric methods
Trees

Natasha Jaques

UNIVERSITY of WASHINGTON

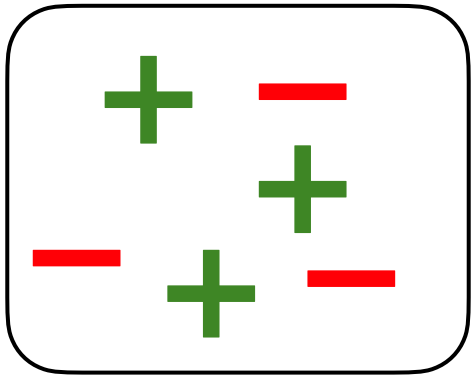
Flow chart for investigation of suspected type 1 diabetes in newly diagnosed adults, based on data from White European populations



Holt et al., 2021

Decision trees

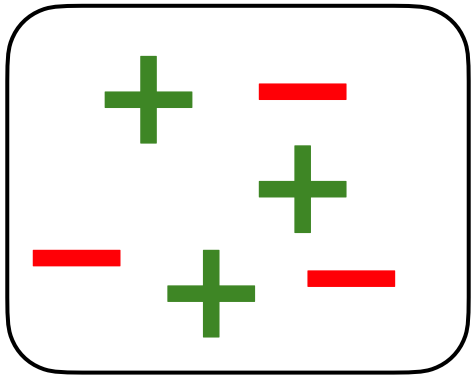
Want to classify spam from not spam using text-based features. What feature gives me the best split?



Full dataset: 50% spam, 50% not
Entropy: 1.0

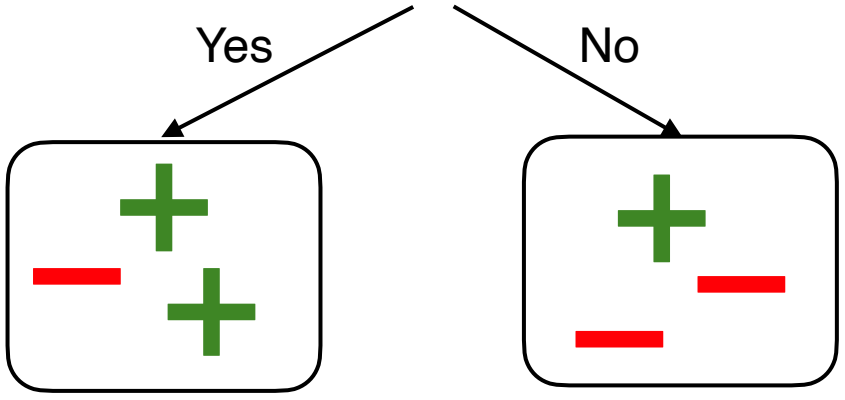
Decision trees

Want to classify spam from not spam using text-based features. What feature gives me the best split?



Full dataset: 50% spam, 50% not
Entropy: 1.0

Split on feature: contains “Limited offer”

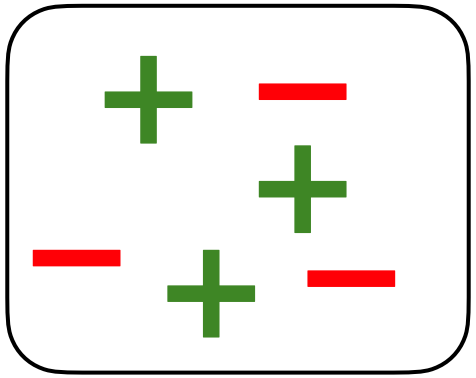


66% spam, 33% not
Entropy: 0.918

33% spam, 66% not
Entropy: 0.918

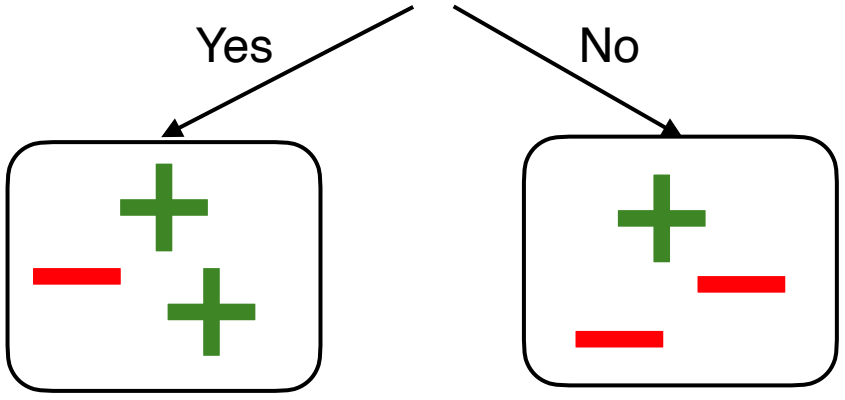
Decision trees

Want to classify spam from not spam using text-based features. What feature gives me the best split?



Full dataset: 50% spam, 50% not
Entropy: 1.0

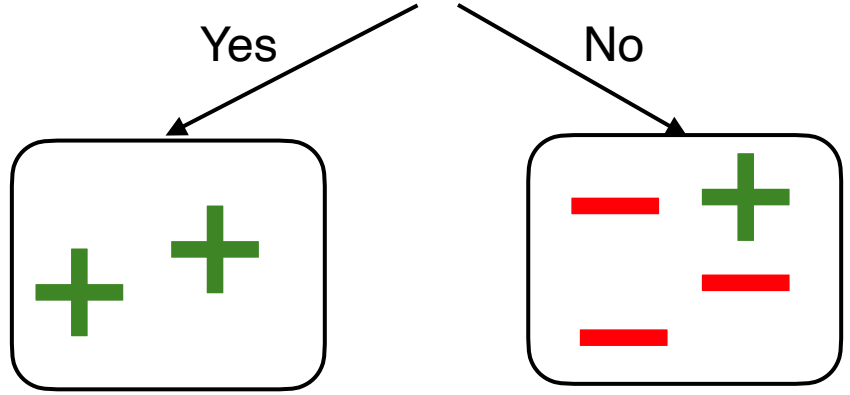
Split on feature: contains "Limited offer"



66% spam, 33% not
Entropy: 0.918

33% spam, 66% not
Entropy: 0.918

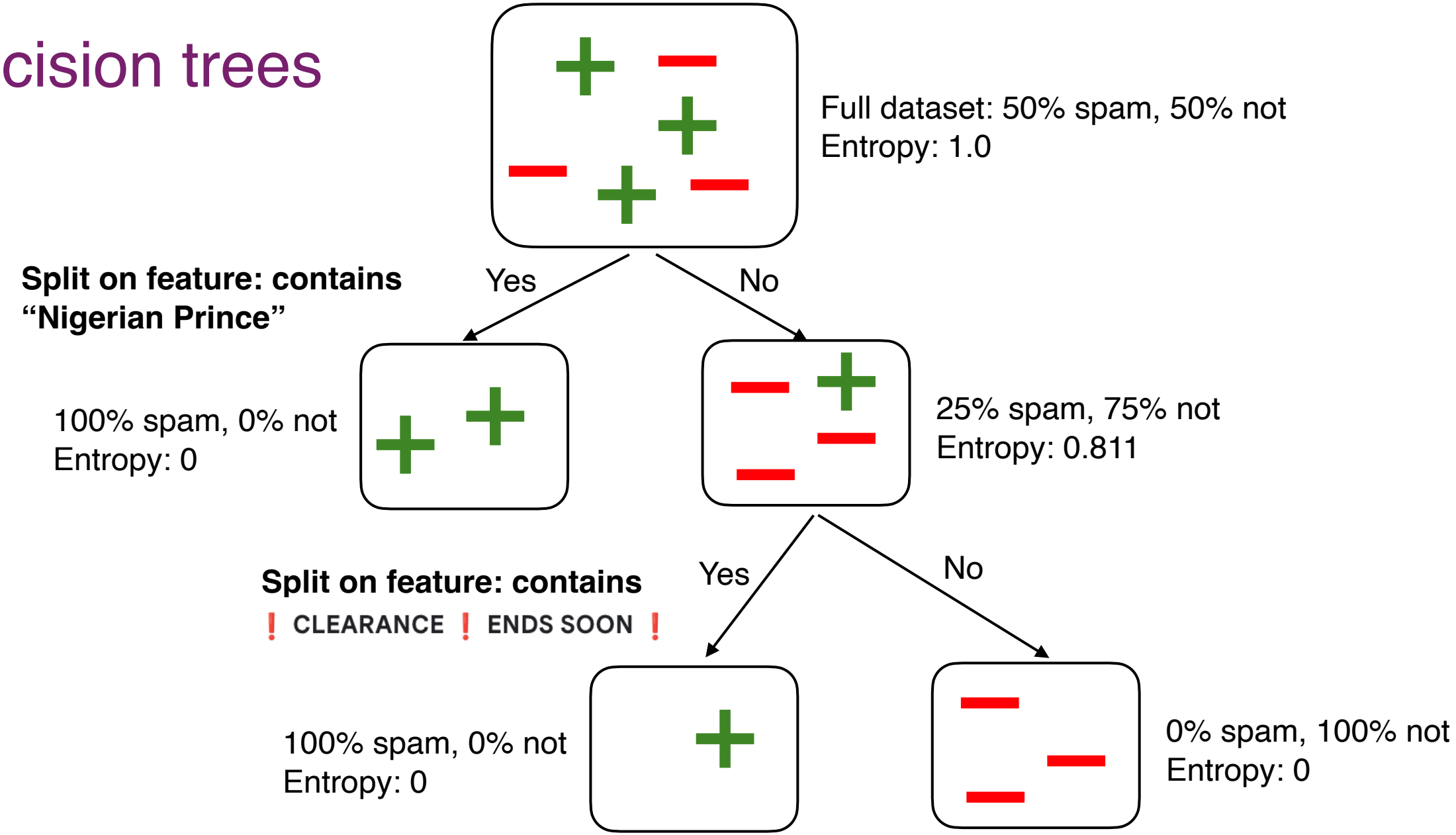
Split on feature: contains "Nigerian Prince"



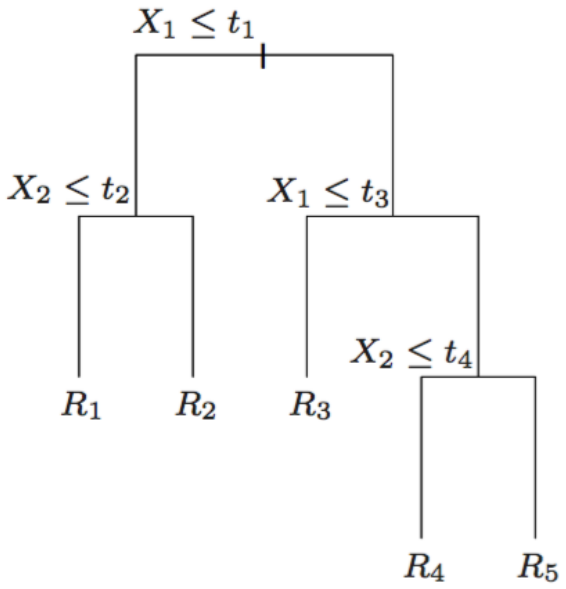
100% spam, 0% not
Entropy: 0

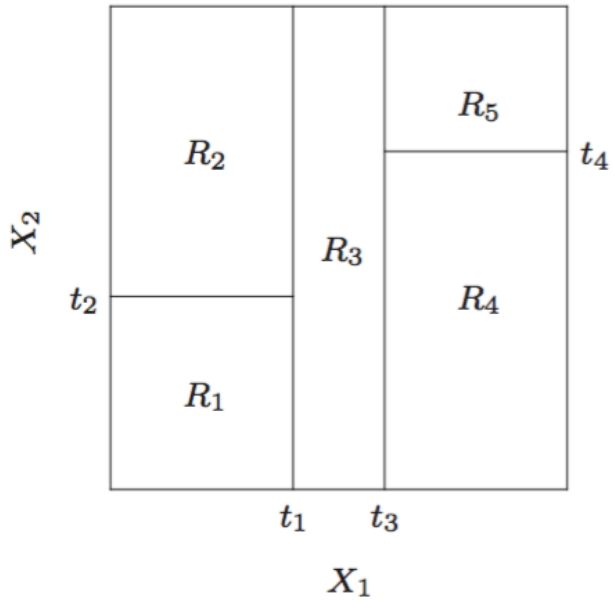
25% spam, 75% not
Entropy: 0.811

Decision trees

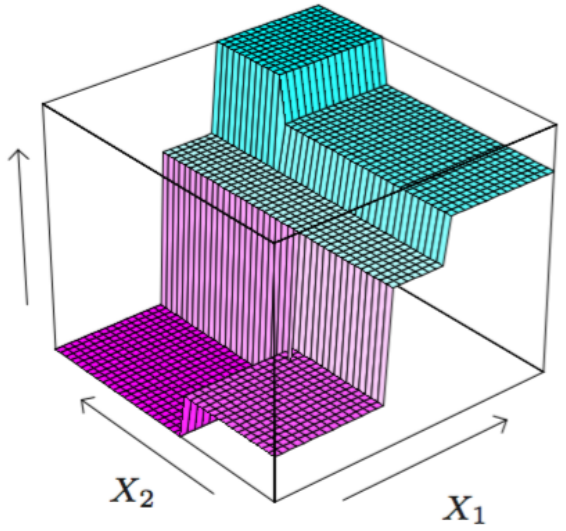


Regression trees





$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m).$$



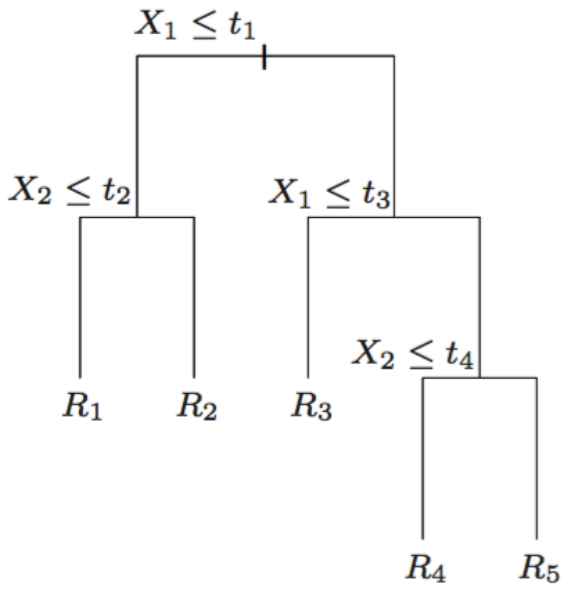
Generic algorithm for building trees

1. Start from empty decision tree
2. Recursively, for each node:
   - Iterate through all features and compute how good it'd be to split on each feature
   - Split on the "best" feature
3. Prune

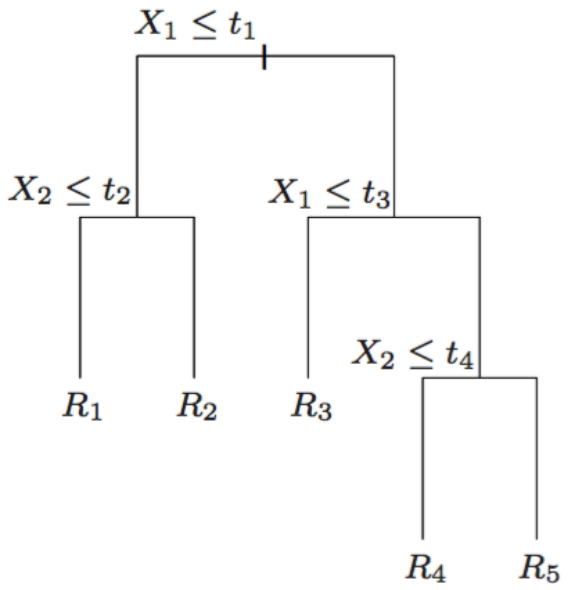
Design choices:

- Termination condition
- Tree complexity
- Splitting criterion
- Pruning

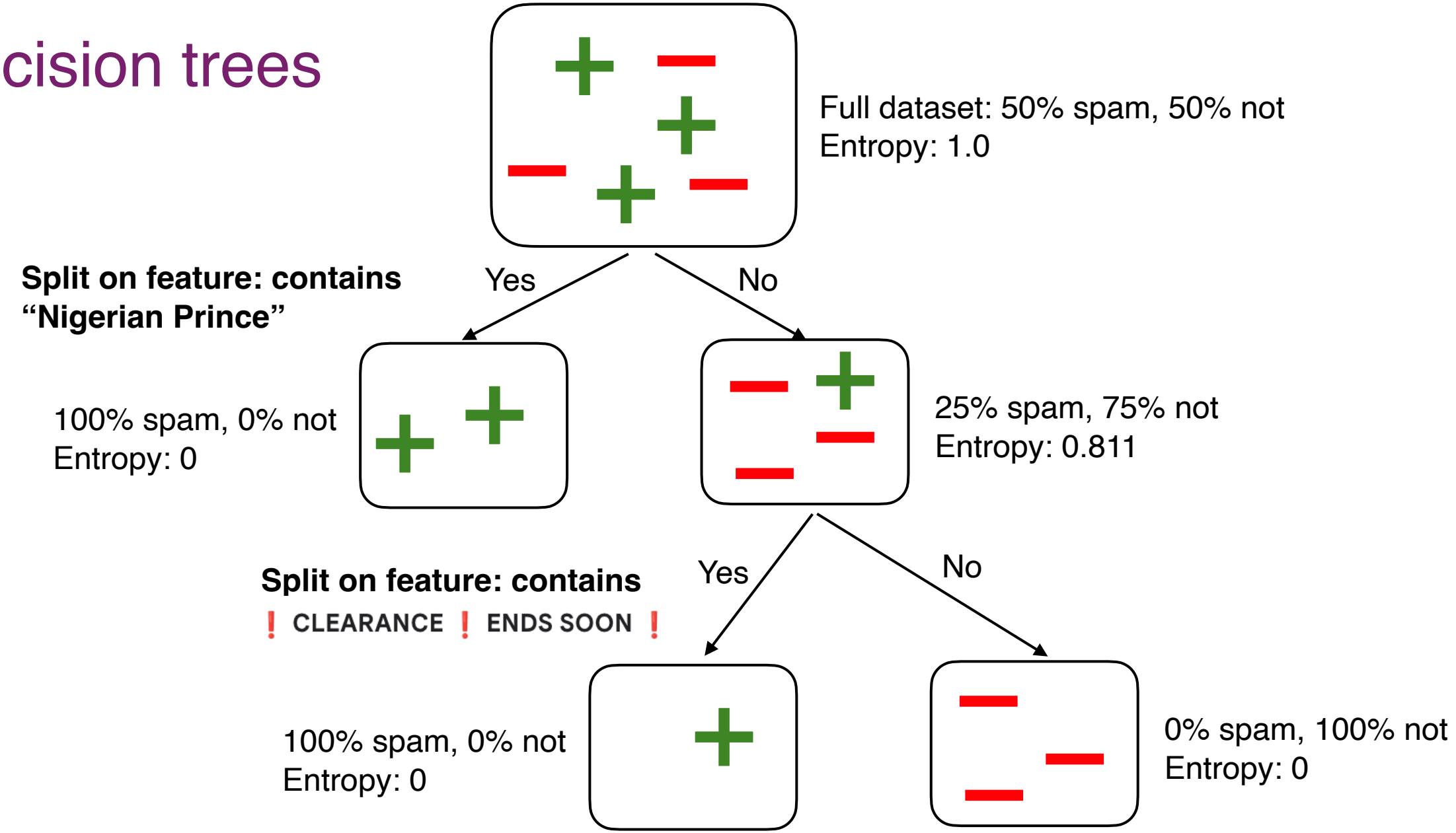
Splitting regression trees



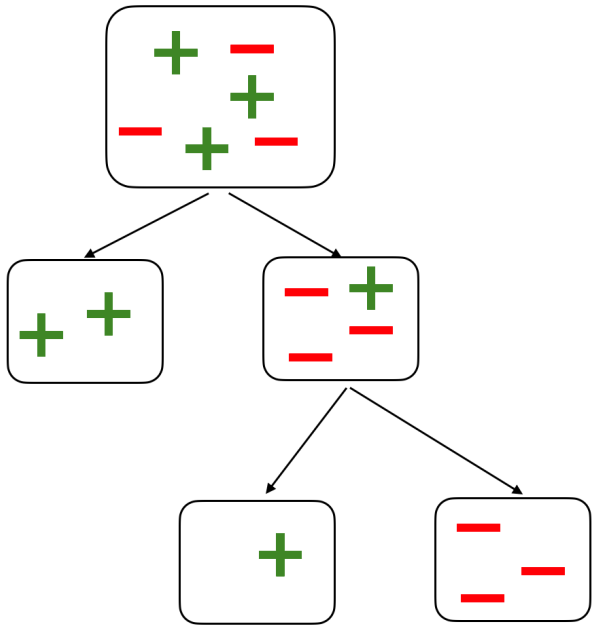
Splitting regression trees



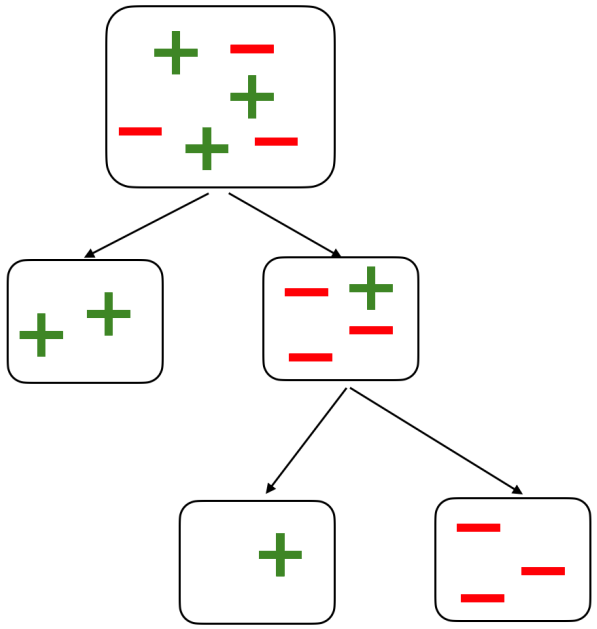
Decision trees



Splitting decision trees



Splitting decision trees



Interpreting trees

Trees are “easy” to interpret:

- You can explain how the classifier came to the conclusion it did

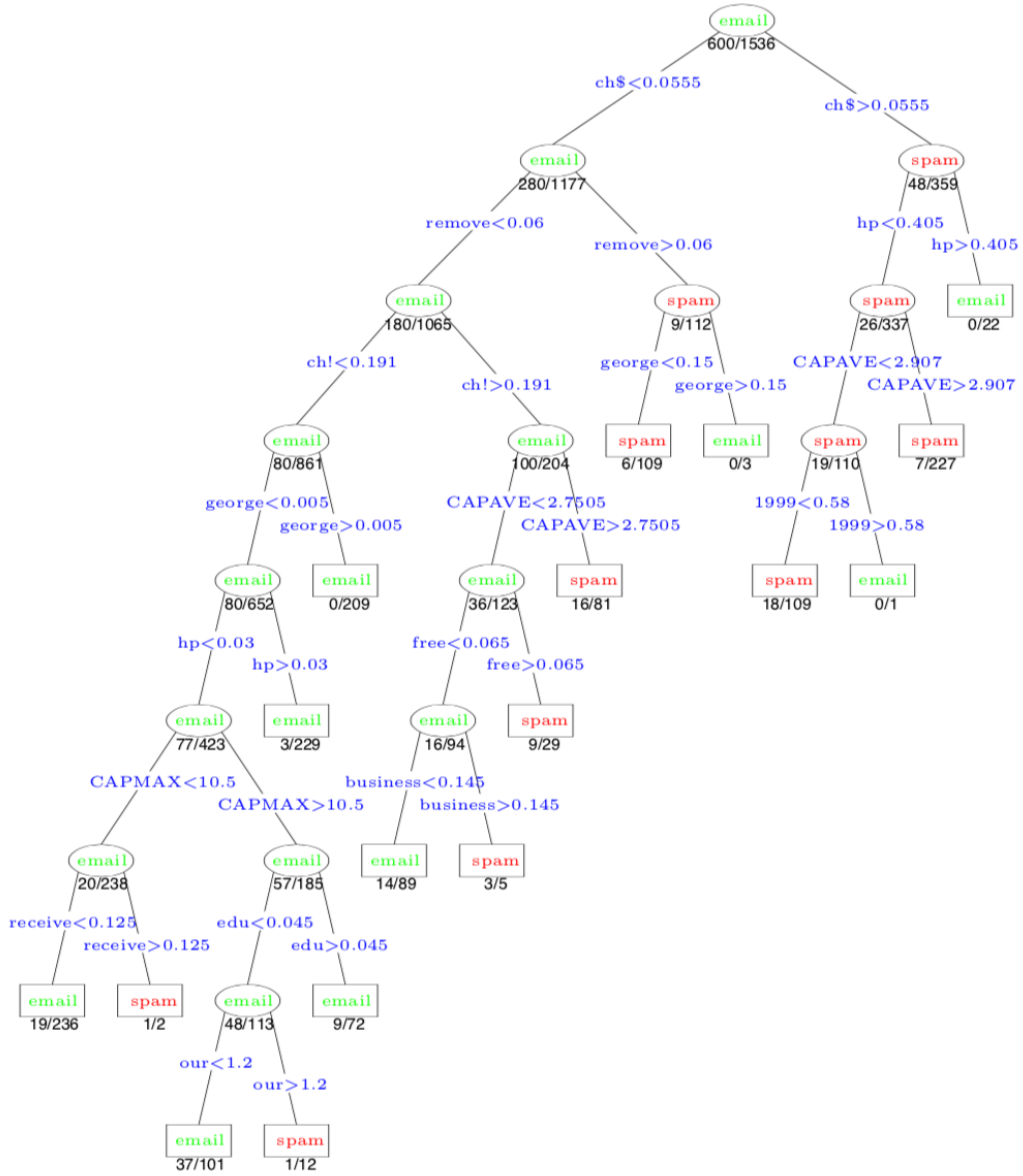
Interpreting trees

Trees are “easy” to interpret:

- You can explain how the classifier came to the conclusion it did

But they can be complex

- Small changes in data can result in large difference in trees



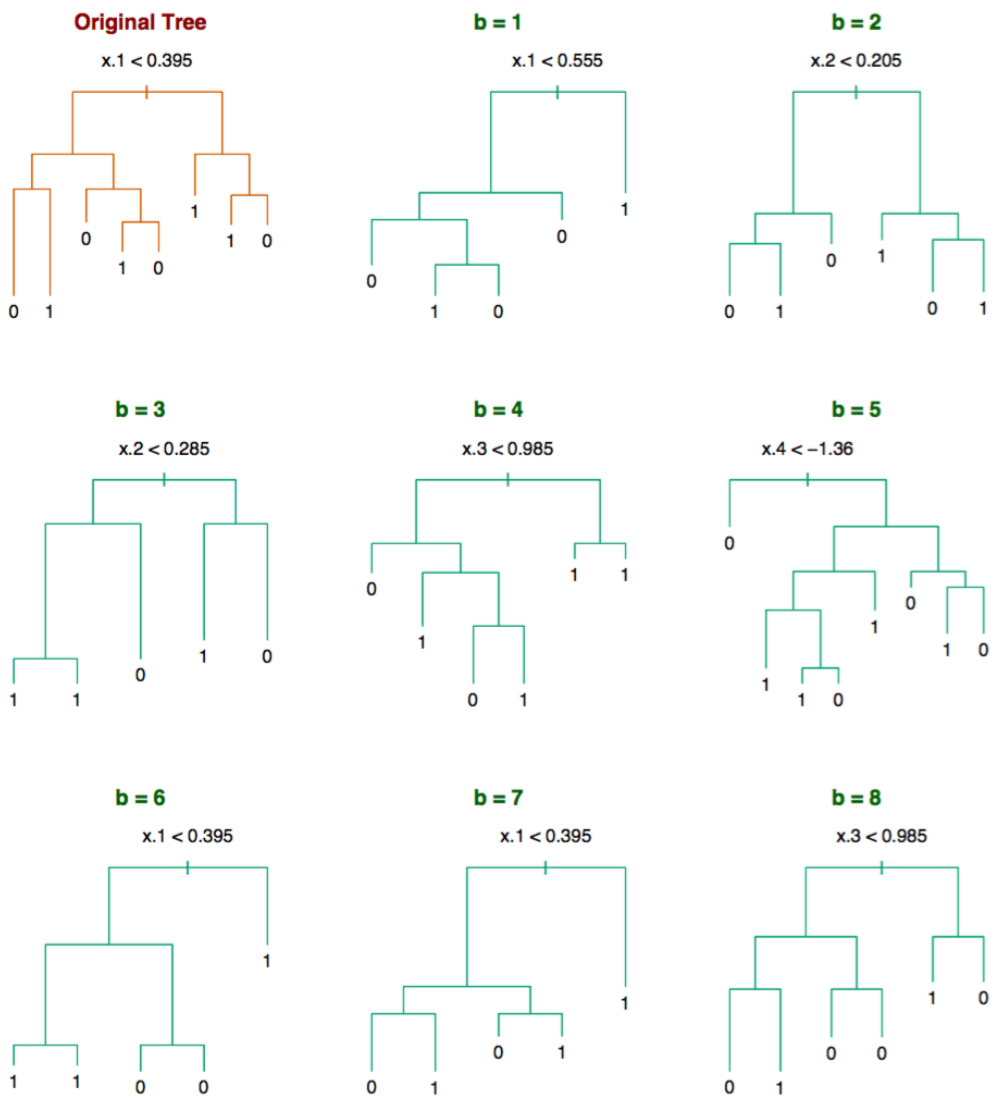
Summary so far

- Trees have bias, variance
- Deal with categorial variables well
- Intuitive, “interpretable”
- Good software exists
- Some theoretical guarantees

Random forests

- Forest = many trees
- Tree methods have low bias but high variance
- We can reduce variance by constructing many “lightly correlated” trees and averaging them

- Bagging: Bootstrap aggregating



Random forests

Algorithm 15.1 *Random Forest for Regression or Classification.*

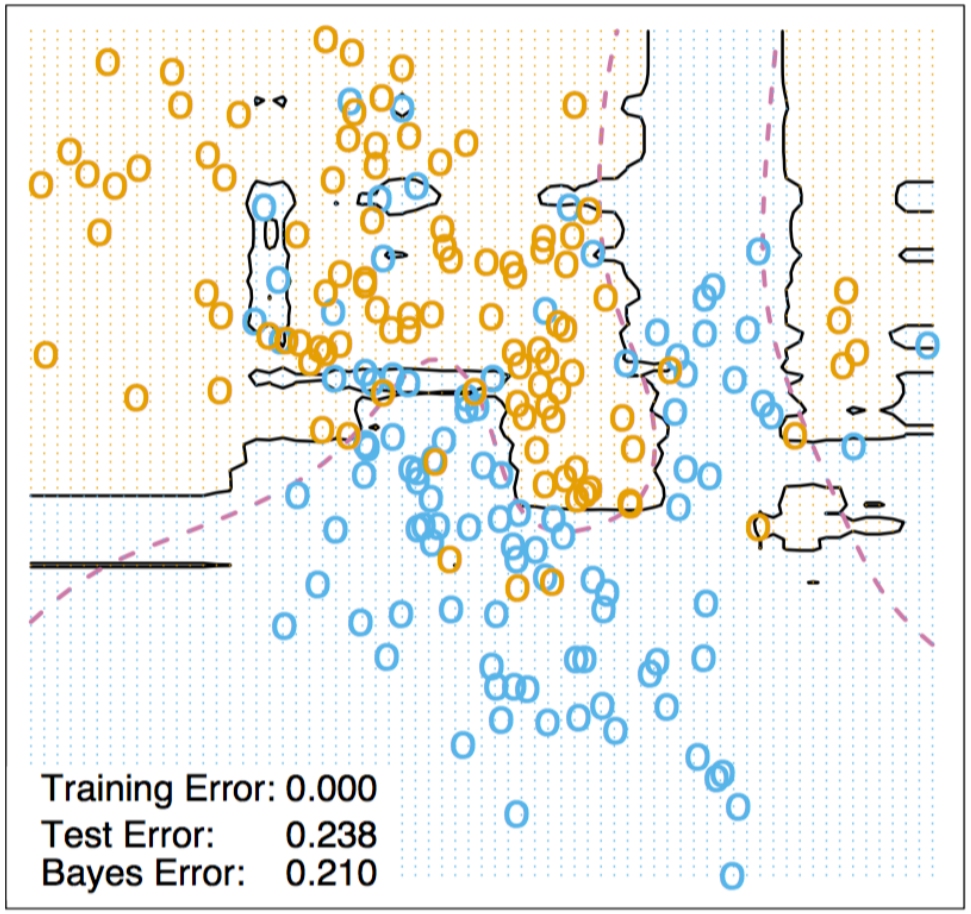
1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Random forests: Decision boundary example

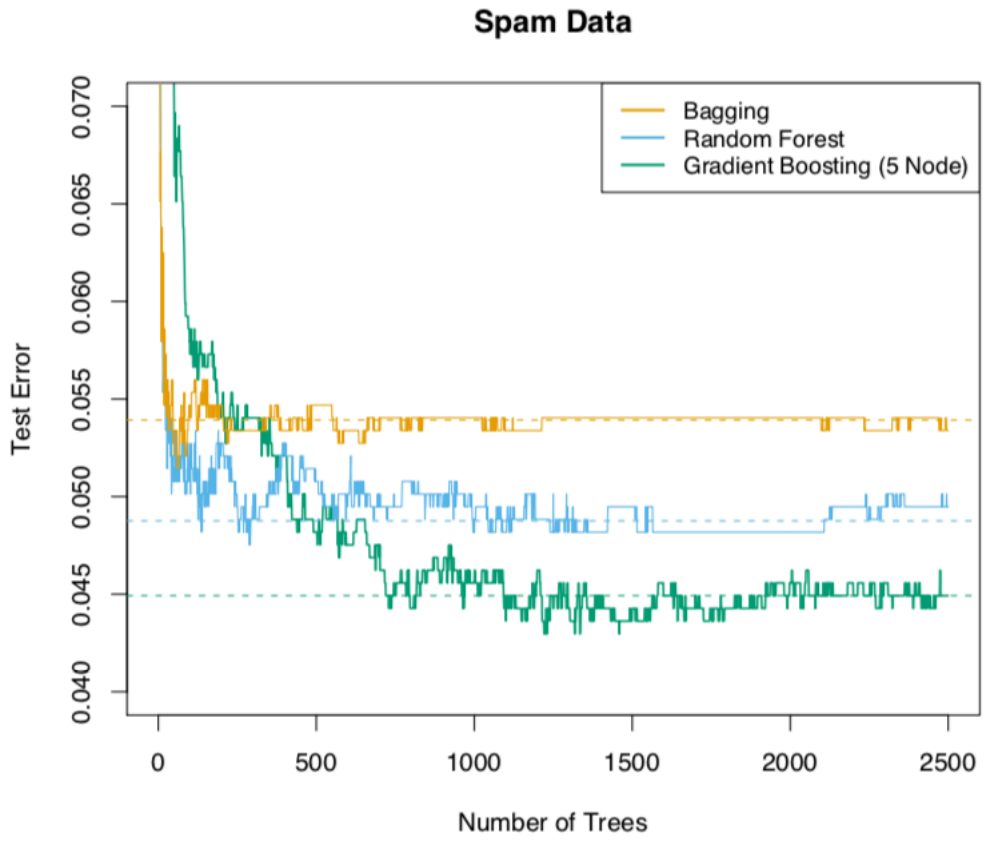


Random forests

Given random variables $Y_1, Y_2, \ldots, Y_B$ with
$\mathbb{E}[Y_i] = y$, $\mathbb{E}[(Y_i - y)^2] = \sigma^2$, $\mathbb{E}[(Y_i - y)(Y_j - y)] = \rho\sigma^2$

$$\mathbb{E}[(\frac{1}{B}\sum_{i=1}^{B} Y_i - y)^2] =$$

The power of weakly correlated predictors



Bagging: Averaged trees on bootstrapped datasets using all d features

Random forest: Averaged trees on bootstrapped datasets using m randomly selected features

Takeaways:

- Reducing correlation improves performance
- Ensembles are powerful

Summary so far

- Random forests have bias, variance
- Deal with categorial variables well
- Not that intuitive nor “interpretable”
- Gives some notion of confidence estimates
- Good software exists
- Some theoretical guarantees

Ensembles are powerful

For binary classification, if you have N weak classifiers, but each one is slightly better than random chance (gets right answer with $p > 0.5$), what happens if you take the majority vote?

As $N \to \infty$, what is the probability that the majority vote gets the right answer?

- Marquis de Condorcet, "Essay on the Application of Analysis to the Probability of Majority Decisions" (1785). Known as the Condorcet Jury Theorem.
- Schapire, "**The Strength of Weak Learnability**" (1990)

Boosting and additive models

Instead of ensembling bootstrapped models, can we:

- Keep the idea of ensembling / combining simpler models, but
- Not necessarily have the models be identically distributed?

Key idea: Given a current collection of models, add a new model that focuses on what the previous models got wrong

Additive models

Forward stagewise additive models

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:
   (a) Compute

$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)).$$

   (b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

Forward stagewise additive models

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:
   (a) Compute

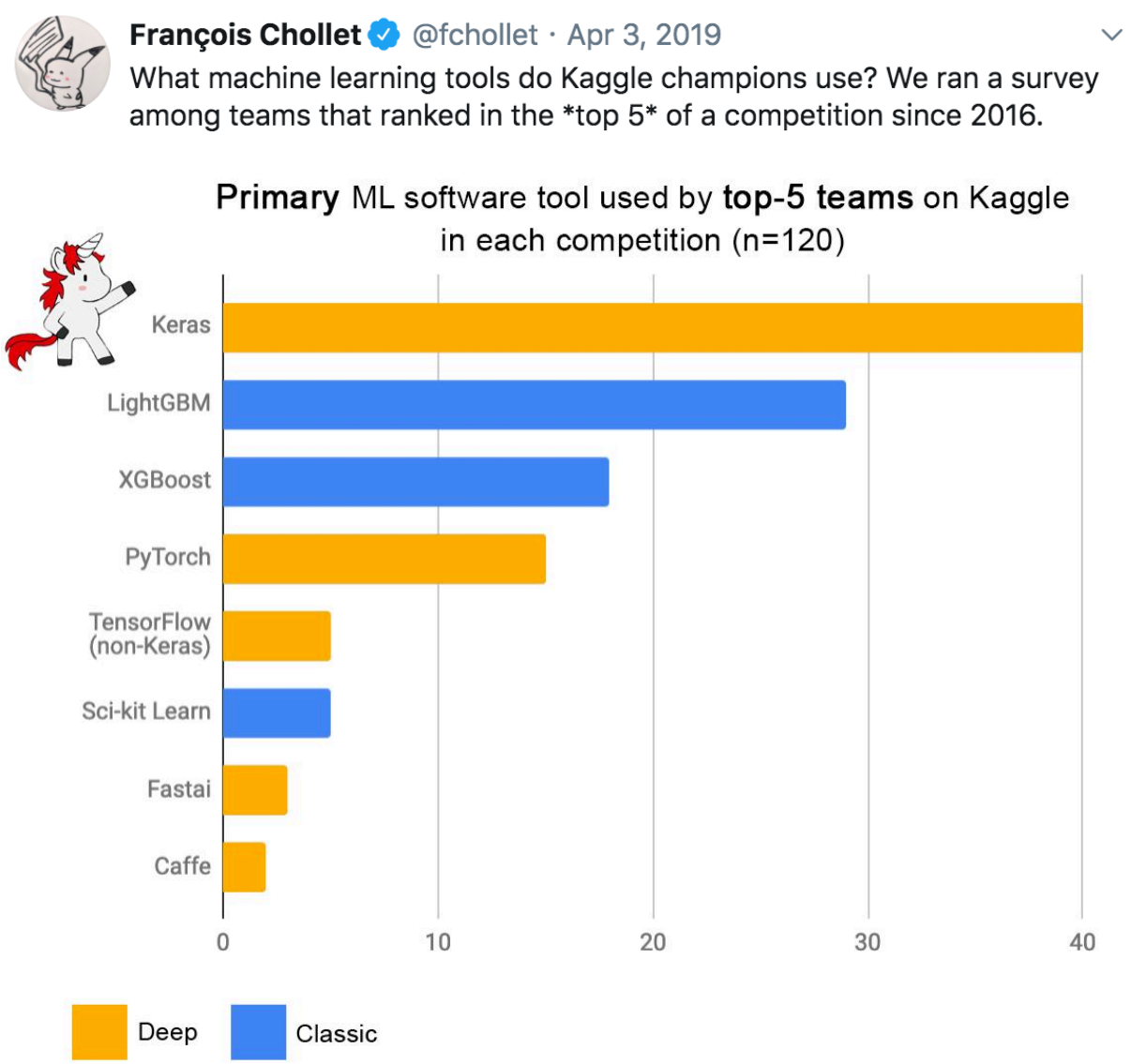
$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)).$$

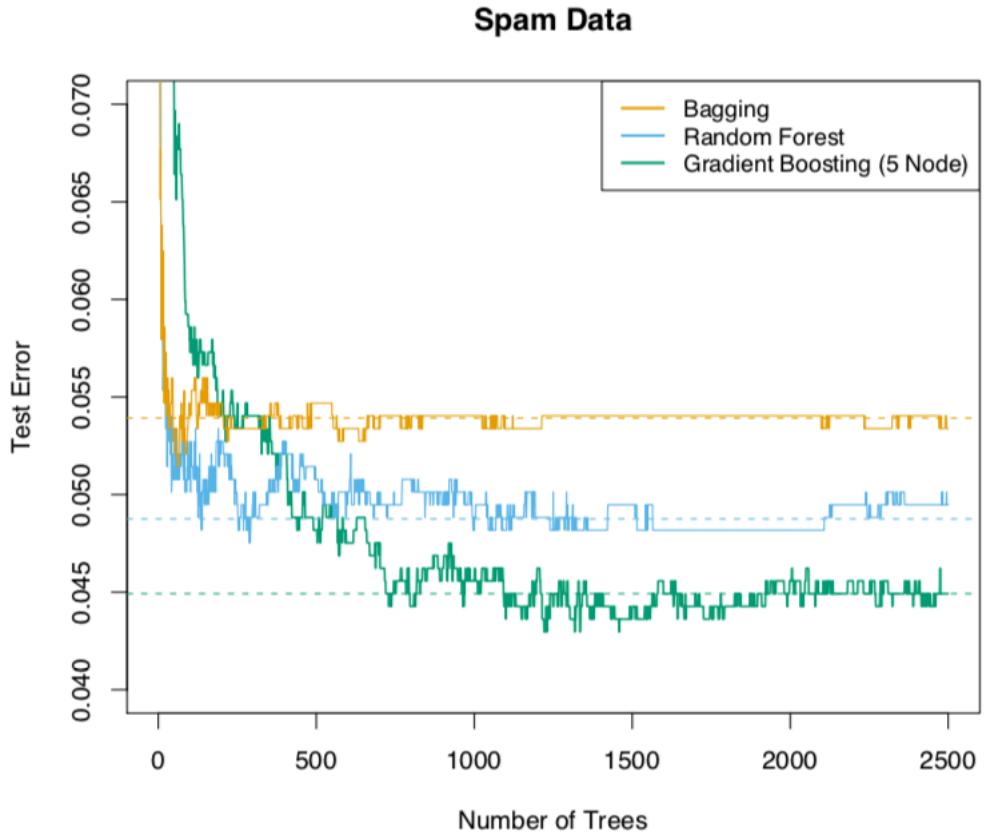
   (b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

Gradient boosting

A brief history of boosting

- 1988 Kearns and Valiant: “Can weak learners be combined to create a strong learner?”
- 1990 Schapire: “Yup, in theory”
- 1995 Schapire and Freund: “Practical for 0/1 loss” -> AdaBoost
- 2001 Friedman: “Practical for arbitrary losses”
- 2014 Tianqi Chen: “Scale it up!” -> XGBoost


François Chollet @fchollet · Apr 3, 2019
What machine learning tools do Kaggle champions use? We ran a survey among teams that ranked in the *top 5* of a competition since 2016.
Primary ML software tool used by top-5 teams on Kaggle in each competition (n=120)
Keras
LightGBM
XGBoost
PyTorch
TensorFlow (non-Keras)
Sci-kit Learn
Fastai
Caffe
0
10
20
30
40
Deep
Classic



Bagging: Averaged trees on bootstrapped datasets using all d features

Random forest: Averaged trees on bootstrapped datasets using m randomly selected features

Boosting: Learned combinations of trees

Takeaways

- Single trees: low bias, high variance
- Ensembles: low bias, (relatively) low variance
- Bagging averages many lightly dependent models to reduce variance
  - Random forests: same but with random subset of features
- Boosting learns a linear combination of high bias, highly dependent classifiers to reduce error
- Gradient boosted trees are commonly used for categorical data