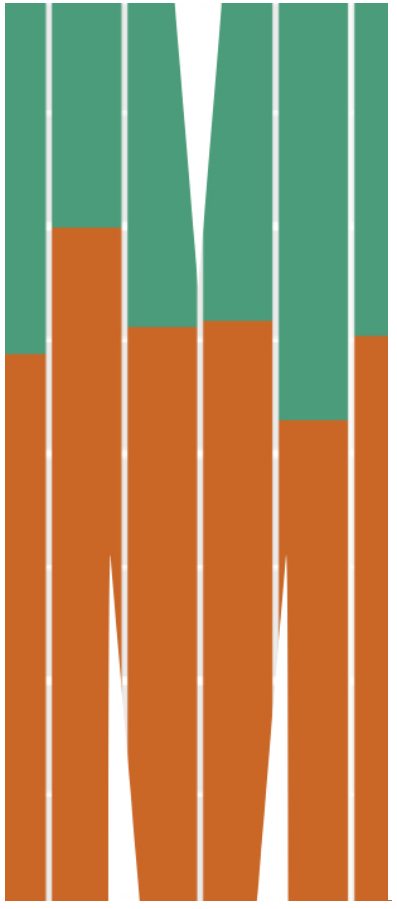


ETC3250: Ensemble Methods

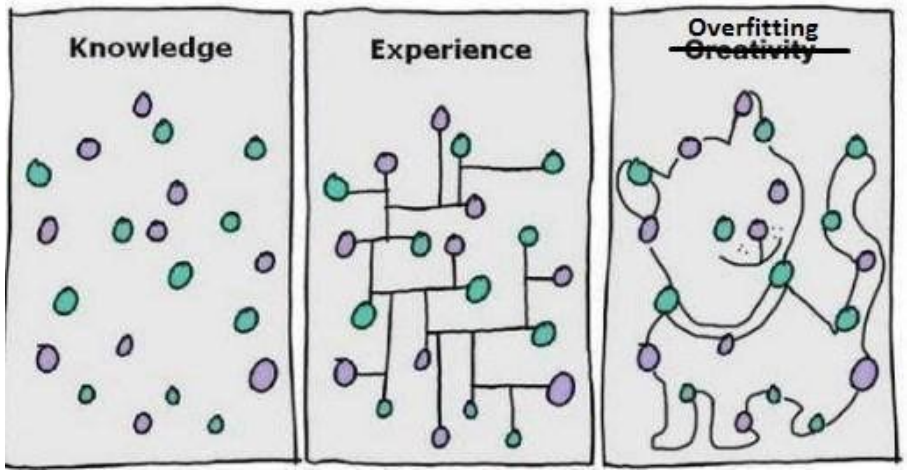
Semester 1, 2020

Professor Di Cook

Econometrics and Business Statistics
Monash University

Week 7 (a)

What's wrong with a single tree?



Source: Hugh MacLeod / Statistical Statistics Memes

Solution? Ensemble methods

Ensemble methods use multiple learning algorithms to obtain better predictive performance than any of the single constituents.

Roadmap

We will learn about different ensembles, increasing in complexity (but also potentially in predictive performance) as we go. These methods are

- Bagging
- Random Forests
- Boosted Trees

Bootstrap aggregation

- Take B different *bootstrapped* training sets:

$$D_1, D_2, \ldots, D_B$$

- Build a separate prediction model using each $D_{(\cdot)}$:

$$\hat{f}_1(x), \hat{f}_2(x), \ldots, \hat{f}_B(x)$$

- Combine resulting predictions, e.g. average

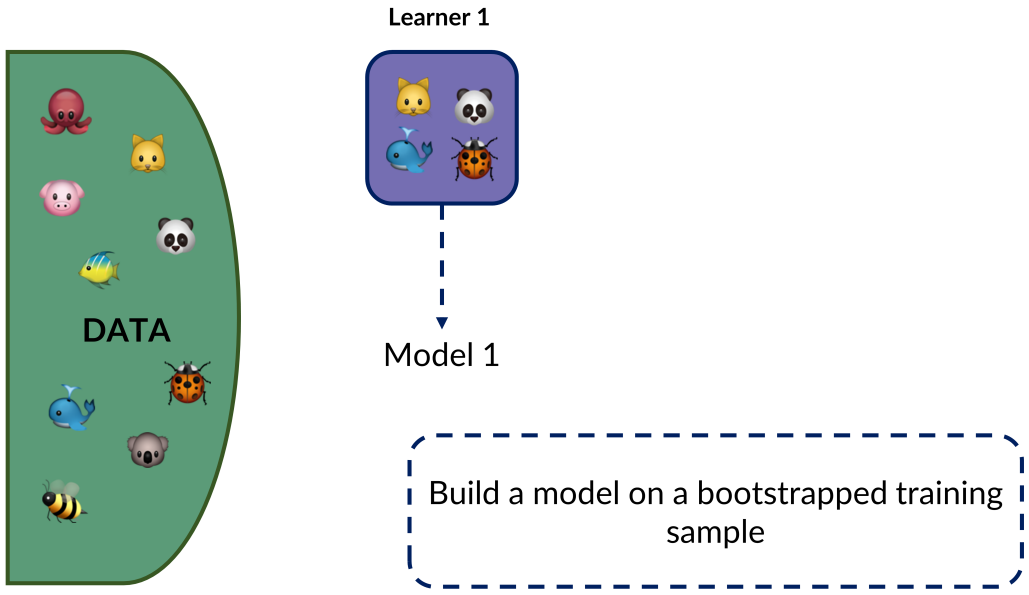
$$\hat{f}_{\text{avg}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x)$$

Bagging trees

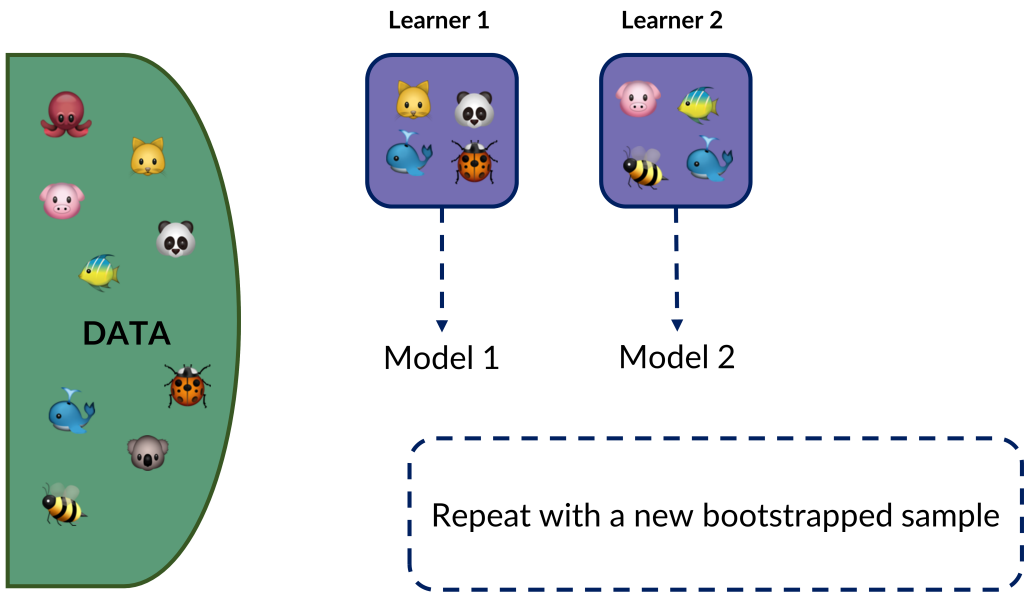
Bagged trees

- Construct B regression trees using B bootstrapped training sets, and average the resulting predictions.
- Each individual tree has high variance, but low bias.
- Averaging these B trees reduces the variance.
- For classification trees, there are several possible aggregation methods, but the simplest is the **majority vote**.

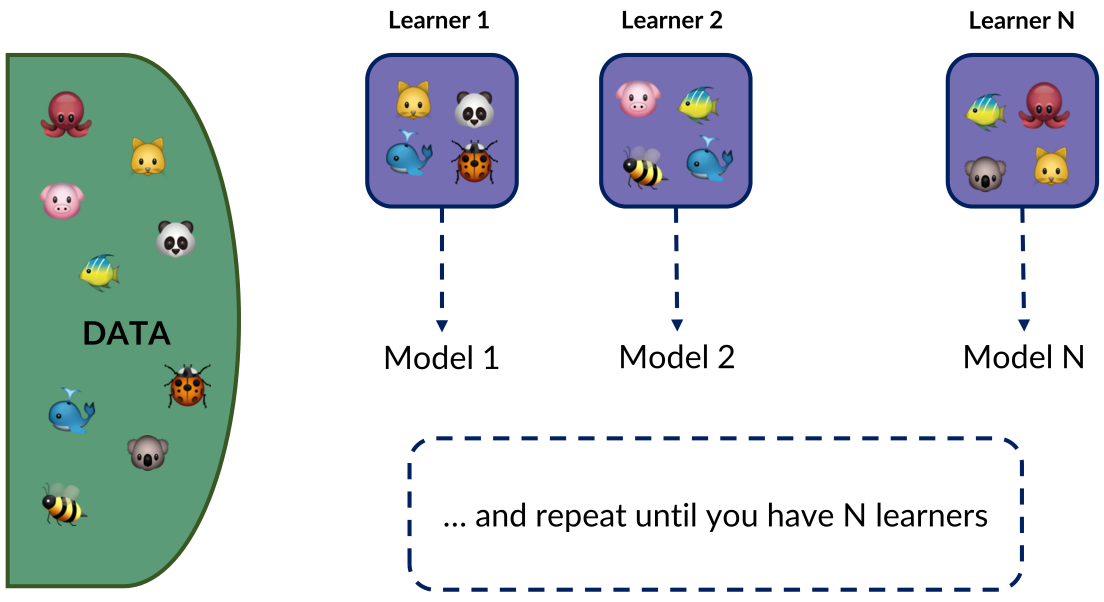
Bagged trees - construction



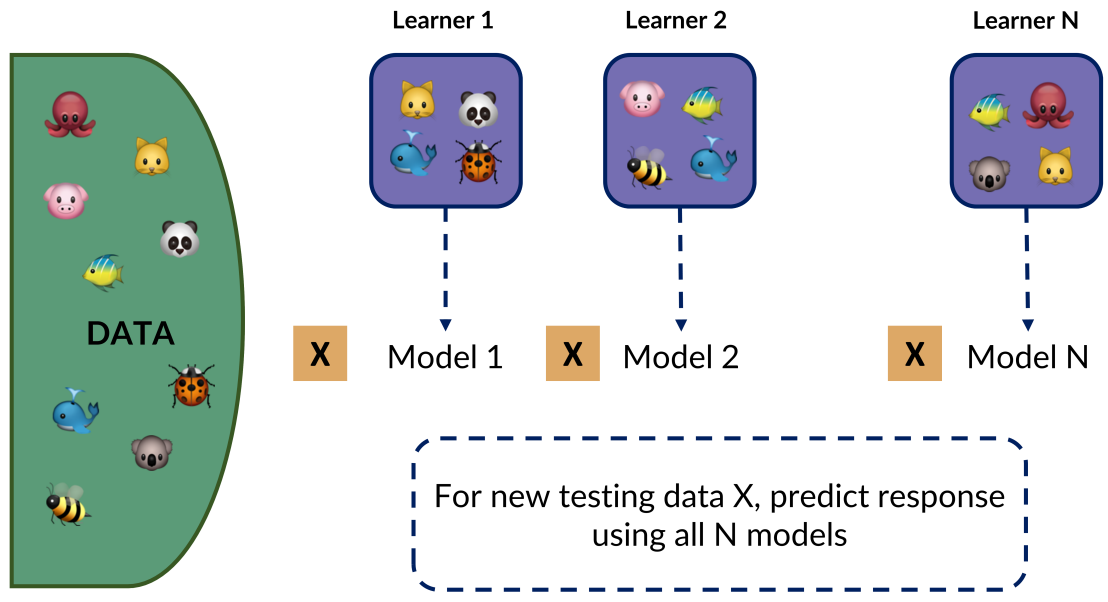
Bagged trees - construction



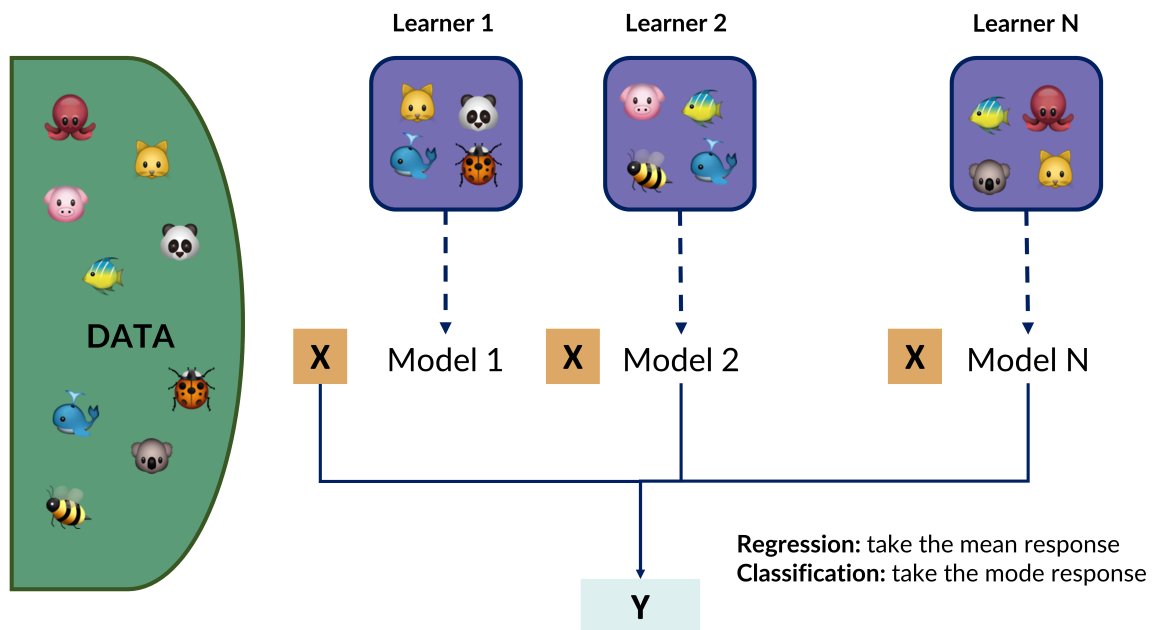
Bagged trees - construction



Bagged trees - construction



Bagged trees - construction



Out of Bag error

- No need to use (cross-)validation to **estimate the test error** of a bagged model (**debatable by some**).
- On average, each bagged tree makes use of around two-thirds of the observations. (Check the textbook exercise.)
- The remaining observations not used to fit a given bagged tree are referred to as the out-of-bag (OOB) observations.
- We can predict the response for the i^{th} observation using each of the trees in which that observation was OOB. This will yield around B/3 predictions for the i^{th} observation.
- To obtain a single prediction for the i^{th} observation, average these predicted responses (regression) or can take a majority vote (classification).

From bagging to Random Forests

However, when bagging trees, a problem still exists. Although the model building steps are independent, the trees in bagging are not completely independent of each other since all the original features are considered at every split of every tree. Rather, trees from different bootstrap samples typically have similar structure to each other (especially at the top of the tree) due to any underlying strong relationships.

To deal with this, we can use Random Forests to help over come this, by sampling the predictors as well as the samples!

Random Forests - the algorithm

1. Input: $L = (x_i, y_i), i = 1, \ldots, n, y_i \in \{1, \ldots, k\}, m < p$, number of variables chosen for each tree, B is the number of bootstrap samples.

2. For $b = 1, 2, \ldots, B$:

   i. Draw a bootstrap sample, L^{*b} of size n^{*b} from L.

   ii. Grow tree classifier, T^{*b}. At each node use a random selection of m variables, and grow to maximum depth without pruning.

   iii. Predict the class of each case not drawn in L^{*b}.

3. Combine the predictions for each case, by majority vote, to give predicted class.

Random Forest - Diagnostics

Useful by-products

- Error is computed automatically on the out-of-bag cases.
- Variable importance: more complicated than one might think
- Vote matrix, $n \times K$: Proportion of times a case is predicted to the class k.
- Proximities, $n \times n$: Closeness of cases measured by how often they are in the same terminal node.

Variable importance

1.For every tree predict the oob cases and count the number of votes cast for the correct class.

2.Randomly permute the values on a variable in the oob cases and predict the class for these cases.

3.Difference the votes for the correct class in the variable-permuted oob cases and the real oob cases. Average this number over all trees in the forest. If the value is large, then the variable is very important.

Alternatively, Gini importance adds up the difference in impurity value of the descendant nodes with the parent node. Quick to compute.

Variable importance

1.For every tree predict the oob cases and count the number of votes cast for the correct class.

2.Randomly permute the values on a variable in the oob cases and predict the class for these cases.

3.Difference the votes for the correct class in the variable-permuted oob cases and the real oob cases. Average this number over all trees in the forest. If the value is large, then the variable is very important.

Alternatively, Gini importance adds up the difference in impurity value of the descendant nodes with the parent node. Quick to compute.

Variable importance

1.For every tree predict the oob cases and count the number of votes cast for the correct class.

2.Randomly permute the values on a variable in the oob cases and predict the class for these cases.

3.Difference the votes for the correct class in the variable-permuted oob cases and the real oob cases. Average this number over all trees in the forest. If the value is large, then the variable is very important.

Alternatively, Gini importance adds up the difference in impurity value of the descendant nodes with the parent node. Quick to compute.

Variable importance

1.For every tree predict the oob cases and count the number of votes cast for the correct class.

2.Randomly permute the values on a variable in the oob cases and predict the class for these cases.

3.Difference the votes for the correct class in the variable-permuted oob cases and the real oob cases. Average this number over all trees in the forest. If the value is large, then the variable is very important.

Alternatively, Gini importance adds up the difference in impurity value of the descendant nodes with the parent node. Quick to compute.

- Proportion of trees the case is predicted to be each class, ranges between 0-1
- Can be used to identify troublesome cases.
- Used with plots of the actual data can help determine if it is the record itself that is the problem, or if method is biased.
- Understand the difference in accuracy of prediction for different classes.

Proximities

- Measure how each pair of observations land in the forest
- Run both in- and out-of-bag cases down the tree, and increase proximity value of cases i, j by 1 each time they are in the same terminal node.
- Normalize by dividing by B.

Example - Olive Oil data

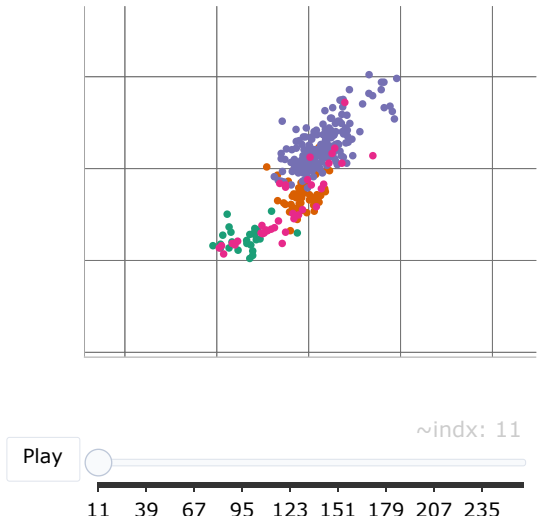
Distinguish the region where oils were produced by their fatty acid signature. Important in quality control and in determining fraudulent marketing.

Areas in the south:

1. North-Apulia
2. Calabria
3. South-Apulia
4. Sicily

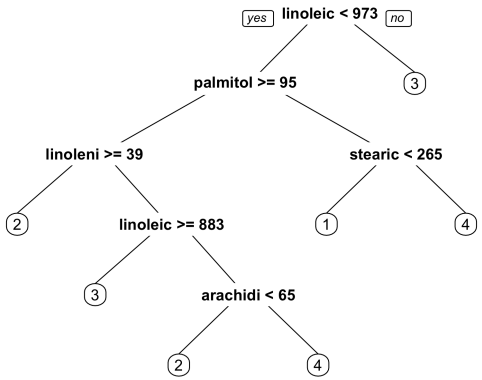
Example - Olive Oil data

Classifying the olive oils in the south of Italy - difficult classification task.



Example - Olive Oil data

Let's first examine the performance of a single decision tree.



Performance of a single tree

Training confusion and error

```
##           Reference
## Prediction   1   2   3   4
##          1  10   0   0   0
##          2   0  31   0   0
##          3   0   0 103   0
##          4   0   0   2  16
```

```
## [1] 0.012
```

```
## Class: 1 Class: 2
##        0        0
```

```
## Class: 3 Class: 4
##    0.019    0.000
```

Test confusion and error

```
##           Reference
## Prediction   1   2   3   4
##          1  12   1   0   2
##          2   0  23   0   2
##          3   0   2 100   1
##          4   1   5   5   7
```

```
## [1] 0.12
```

```
## Class: 1 Class: 2
##    0.077    0.258
```

```
## Class: 3 Class: 4
##    0.048    0.417
```

Example - Olive Oil data

We can then fit a Random Forest model. Provided from the package `randomForest`.

```
##
## Call:
##  randomForest(formula = area ~ ., data = olive_tr, importance = TRUE,      proximity = T
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 7.4%
## Confusion matrix:
##    1  2   3  4 class.error
## 1 8  1   0  1       0.200
## 2 0 29   1  1       0.065
## 3 0  0 103  0       0.000
## 4 1  4   3 10       0.444
```

Performance of Random Forest

Training confusion and error

```
##           Reference
## Prediction   1   2   3   4
##          1  10   0   0   0
##          2   0  31   0   0
##          3   0   0 103   0
##          4   0   0   0  18
```

```
## [1] 0
```

```
## Class: 1 Class: 2
##        0        0
```

```
## Class: 3 Class: 4
##        0        0
```

Test confusion and error

```
##           Reference
## Prediction   1   2   3   4
##          1  13   0   0   2
##          2   0  23   0   2
##          3   0   2 100   1
##          4   1   3   5   9
```

```
## [1] 0.099
```

```
## Class: 1 Class: 2
##    0.071    0.179
```

```
## Class: 3 Class: 4
##    0.048    0.357
```

Diagnostics - variable importance

```
##                 1     2      3     4
## palmitic    0.223 0.014 0.0098 0.026
## palmitoleic 0.256 0.073 0.1030 0.227
## stearic     0.032 0.035 0.0233 0.185
## oleic       0.223 0.149 0.0701 0.038
## linoleic    0.133 0.252 0.1667 0.095
## linolenic  -0.035 0.143 0.0154 0.057
## arachidic   0.014 0.019 0.0051 0.052
```

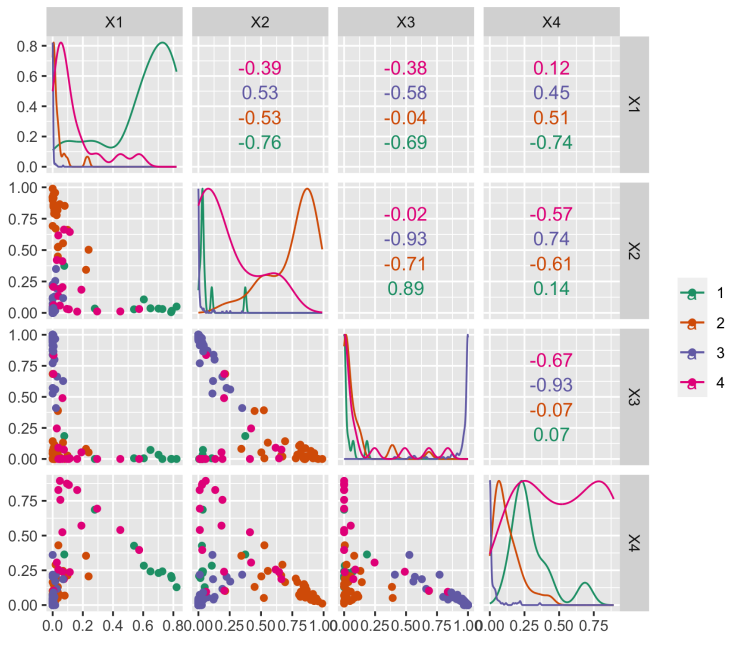
```
##             MeanDecreaseAccuracy MeanDecreaseGini
## palmitic                   0.025              7.9
## palmitoleic                0.119             18.6
## stearic                    0.044              9.0
## oleic                      0.089             18.4
## linoleic                   0.170             21.4
## linolenic                  0.041              7.2
## arachidic                  0.014              4.7
```

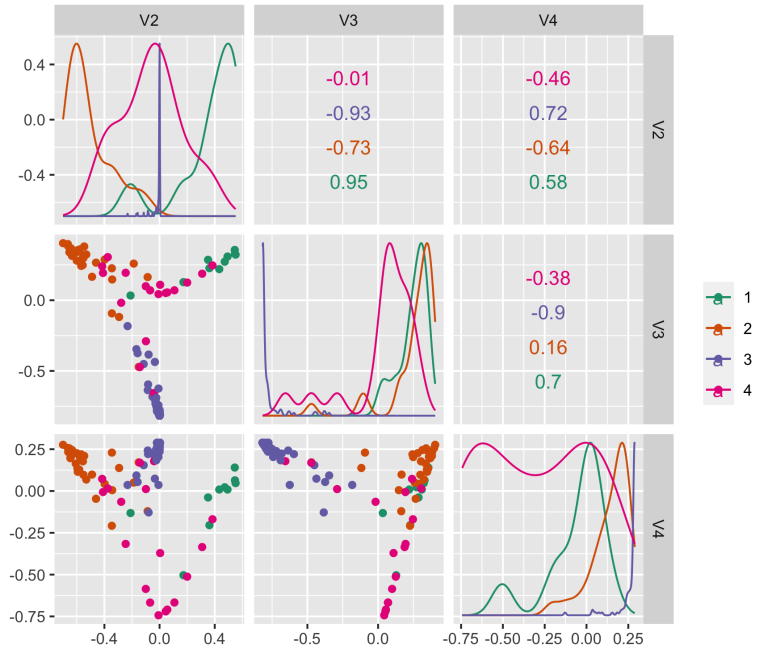
Diagnostics - vote matrix ✋

Examining the vote matrix allows us to see which samples the algorithm had trouble classifying.

Look at the two highlighted rows. How confident would you be in these classifications?

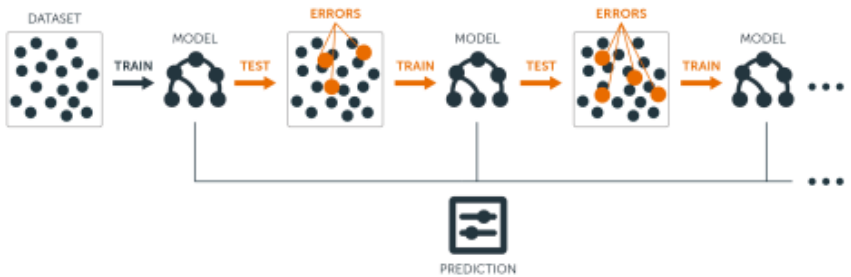
	1	2	3	4
1	0.505434783	0.054347826	0.038043478	0.402173913
2	0.897727273	0.022727273	0.000000000	0.079545455
3	0.677595628	0.010928962	0.000000000	0.311475410
4	0.631868132	0.027472527	0.000000000	0.340659341
5	0.565714286	0.051428571	0.011428571	0.371428571
6	0.897058824	0.014705882	0.000000000	0.088235294
7	0.141361257	0.256544503	0.314136126	0.287958115
8	0.691428571	0.045714286	0.034285714	0.228571429
9	0.573770492	0.005464481	0.000000000	0.420765027
10	0.788888889	0.011111111	0.000000000	0.200000000
11	0.005714286	0.605714286	0.331428571	0.057142857
12	0.000000000	0.984848485	0.005050505	0.010101010
13	0.005434783	0.940217391	0.000000000	0.054347826
14	0.000000000	0.793969849	0.000000000	0.206030151
15	0.015544041	0.704663212	0.025906736	0.253886010
16	0.000000000	0.994623656	0.000000000	0.005376344
17	0.000000000	0.951871658	0.010695187	0.037433155
18	0.017142857	0.674285714	0.011428571	0.297142857
19	0.000000000	0.823529412	0.096256684	0.080213904
20	0.000000000	0.931216931	0.000000000	0.068783069


X1
X2
X3
X4
-0.39
0.53
-0.53
-0.76
-0.38
-0.58
-0.04
-0.69
0.12
0.45
0.51
-0.74
-0.02
-0.93
-0.71
0.89
-0.57
0.74
-0.61
0.14
-0.67
-0.93
-0.07
0.07
1
2
3
4


V2
V3
V4
-0.01
-0.93
-0.73
0.95
-0.46
0.72
-0.64
0.58
-0.38
-0.9
0.16
0.7
1
2
3
4

From Random Forests to Boosting

Whereas random forests build an ensemble of deep independent trees, boosted trees build an ensemble of shallow trees in sequence with each tree learning and improving on the previous one.



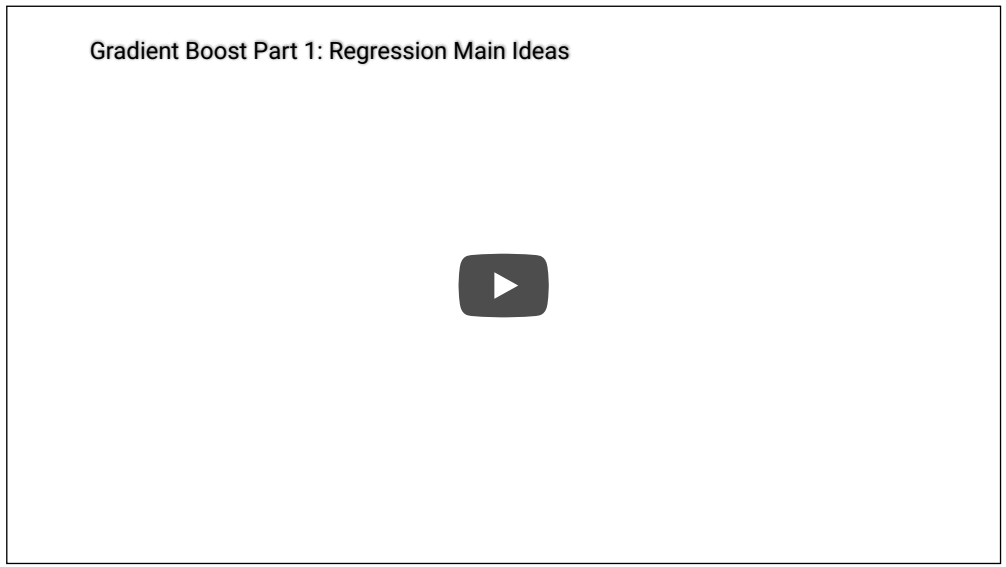
Source: Hands on Machine Learning with R

Boosted trees - the algorithm

Boosting iteratively fits multiple trees, sequentially putting more weight on observations that have predicted inaccurately.

1. Set $\hat{f}(x) = 0$ and $r_i = y_i \forall i$ in training set
2. For b=1, 2, ..., B, repeat:
   a. Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes)
   b. Update $\hat{f}$ by adding a shrunken version of the new tree $\hat{f}(x) = \hat{f}(x) + \lambda \hat{f}^b(x)$.
   c. Update the residuals $r_i = r_i - \lambda \hat{f}^b(x_i)$
3. Output boosted model, $\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$

Further boosting resources



More resources

Cook & Swayne (2007) "Interactive and Dynamic Graphics for Data Analysis: With Examples Using R and GGobi" have several videos illustrating techniques for exploring high-dimensional data in association with trees and forest classifiers:

- Trees video
- Forests video

👩‍💻 Made by a human with a computer

Slides at https://iml.numbat.space.

Code and data at https://github.com/numbats/iml.

Created using R Markdown with flair by **xaringan**, and **kunoichi** (female ninja) style.