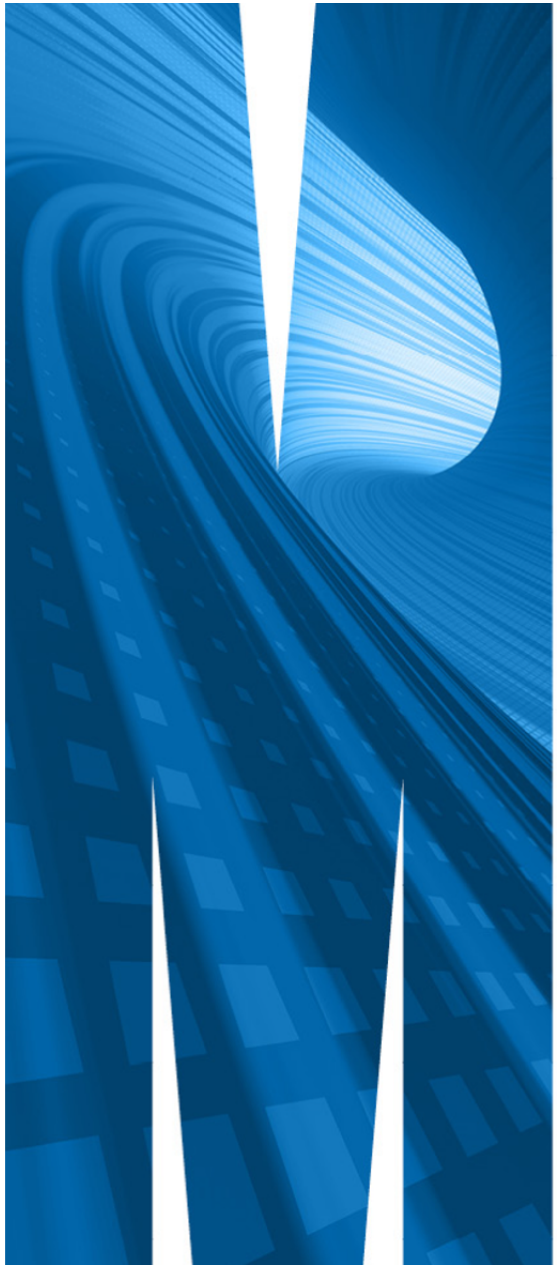
MONASH University

ETC3250/5250: Introduction to Machine Learning

Random forests



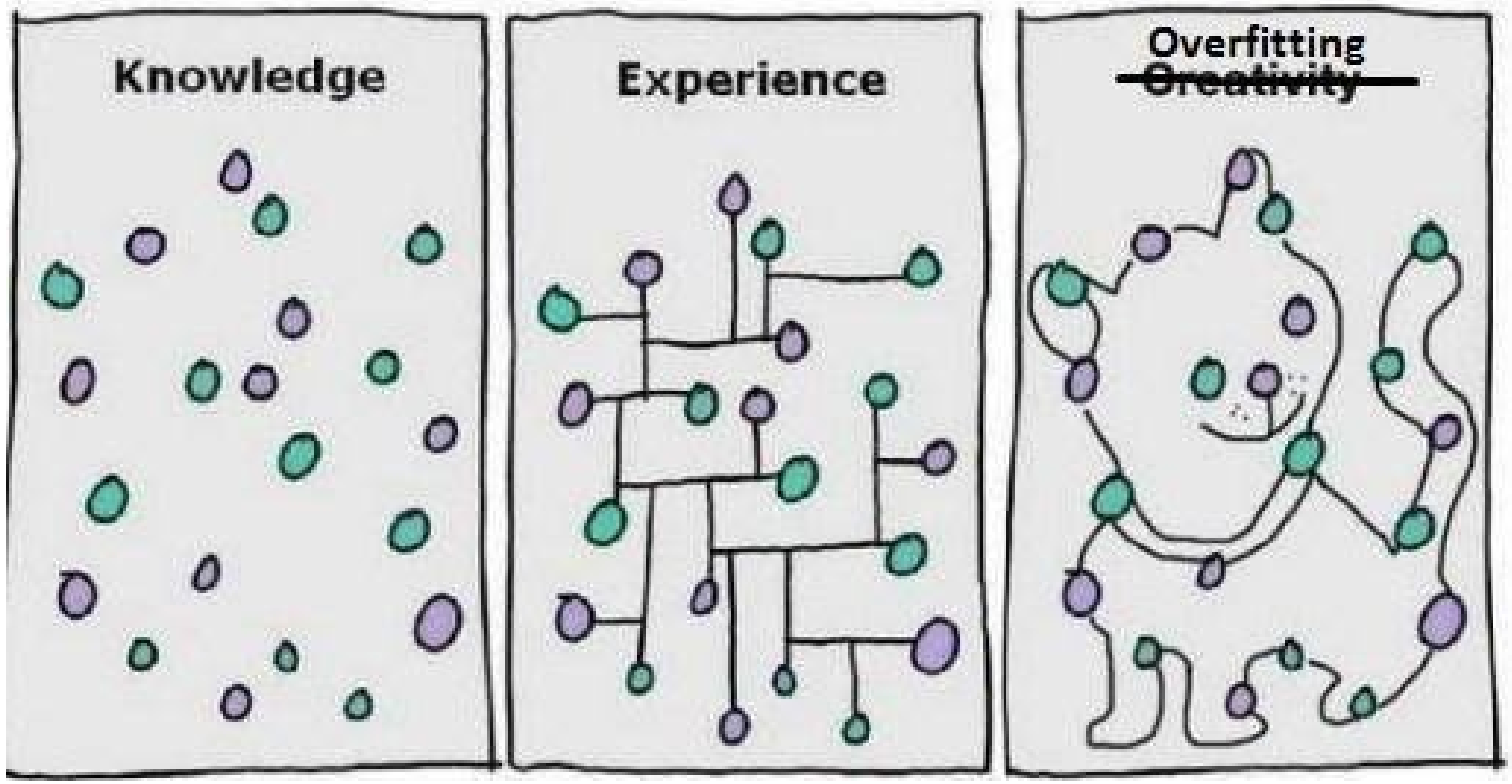
Lecturer: *Professor Di Cook*

Department of Econometrics and Business Statistics

ETC3250.Clayton-x@monash.edu

Week 7a

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: combine the predictions of multiple trees, fitted on bootstrap samples.
- Random forests: combine predictions from bagged trees, plus random samples of predictors.
- Boosted trees: combine predictions from trees sequentially fit to residuals from previous fit.

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.

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.

Read a fun explanation here by Harriet Mason

Vote Matrix

- 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

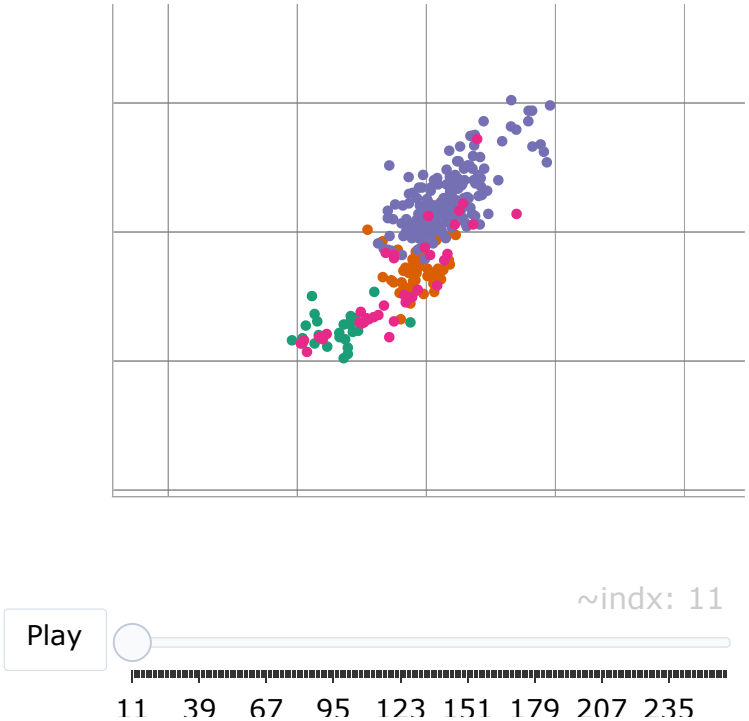
Distinguish the region where oils were produced by their fatty acid signature. Important in quality control and in determining fraudulent marketing.

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 - random forest fit

```
set.seed(2021)
olive <- olive %>%
  mutate(area = factor(area)) %>%
  dplyr::select(area:arachidic)
olive_split <- initial_split(olive, 2/3,
                             strata = area)
olive_tr <- training(olive_split)
olive_ts <- testing(olive_split)

olive_rf <- rand_forest() %>%
  set_engine("randomForest",
             importance=TRUE, proximity=TRUE) %>%
  set_mode("classification") %>%
  fit(area~., data=olive_tr)
```

```
## parsnip model object
##
## Fit time:  85ms
##
## Call:
##  randomForest(x = maybe_data_frame(x), y = y, importance = ~TRUE,      proxim
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 8.8%
## Confusion matrix:
##    1  2   3  4 class.error
## 1 17  0   1  2  0.15000000
## 2  0 35   1  1  0.05405405
## 3  0  2 131  2  0.02962963
## 4  2  4   4 14  0.41666667
```

Test set confusion and accuracy

```
olive_ts_pred <- olive_ts %>%
  mutate(.pred = predict(olive_rf, olive_ts)$.pred_class)
conf_mat(olive_ts_pred, area, .pred)$table %>% addmargins()
```

```
##            Truth
## Prediction   1   2   3   4 Sum
##        1     5   0   0   0   5
##        2     0  16   0   1  17
##        3     0   2  71   3  76
##        4     0   1   0   8   9
##        Sum   5  19  71  12 107
```

```
bal_accuracy(olive_ts_pred, area, .pred)$.estimate
```

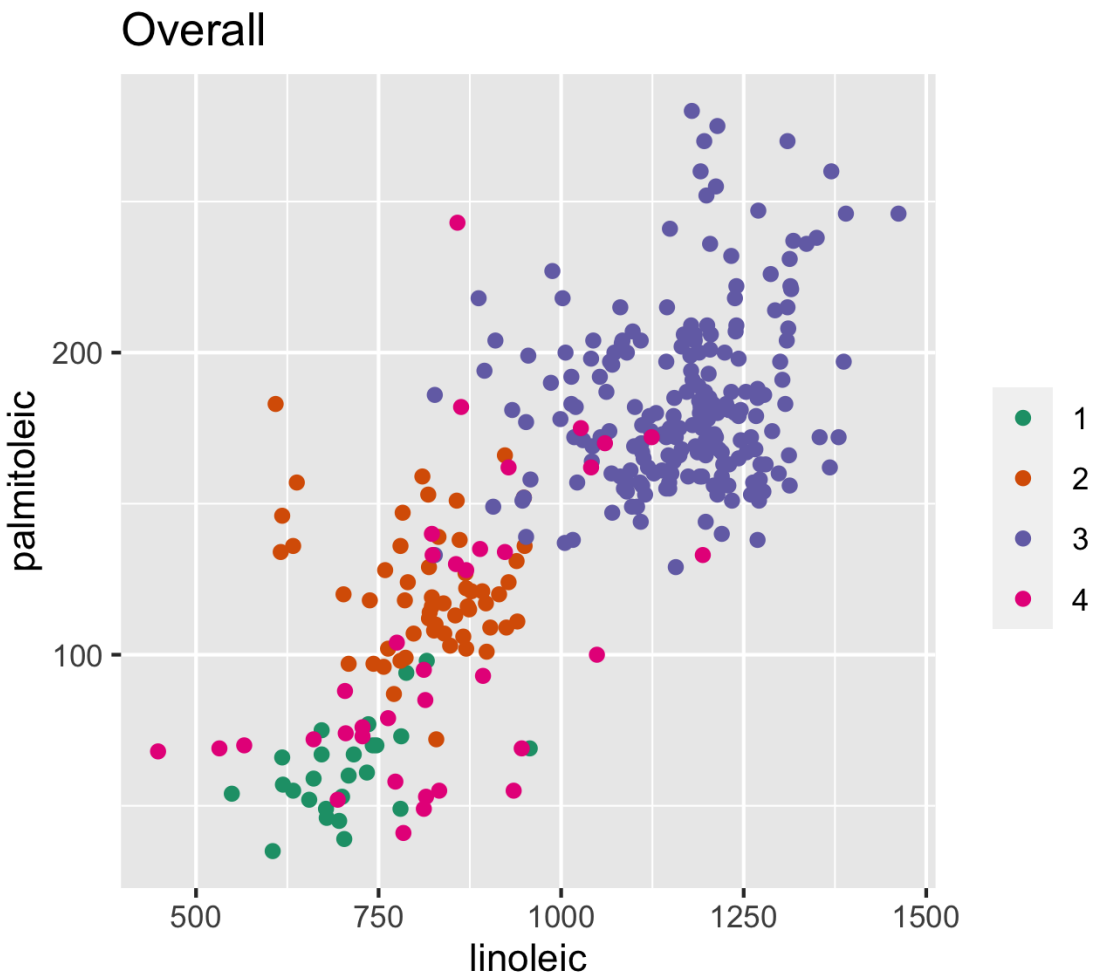
```
## [1] 0.9184991
```

Diagnostics - variable importance

```
##                   1     2     3     4
## palmitic     0.2686 0.027 0.020 0.042
## palmitoleic  0.2371 0.085 0.118 0.143
## stearic     -0.0031 0.053 0.025 0.113
## oleic        0.2905 0.125 0.074 0.020
## linoleic     0.2100 0.264 0.182 0.055
## linolenic   -0.0013 0.144 0.012 0.049
## arachidic    0.0727 0.040 0.012 0.109

##             MeanDecreaseAccuracy MeanDecreaseGini
## palmitic                   0.046             12.6
## palmitoleic                0.125             24.2
## stearic                    0.036             10.8
## oleic                      0.095             22.9
## linoleic                   0.183             29.9
## linolenic                  0.037              9.3
## arachidic                  0.033             10.5
```

Important variables

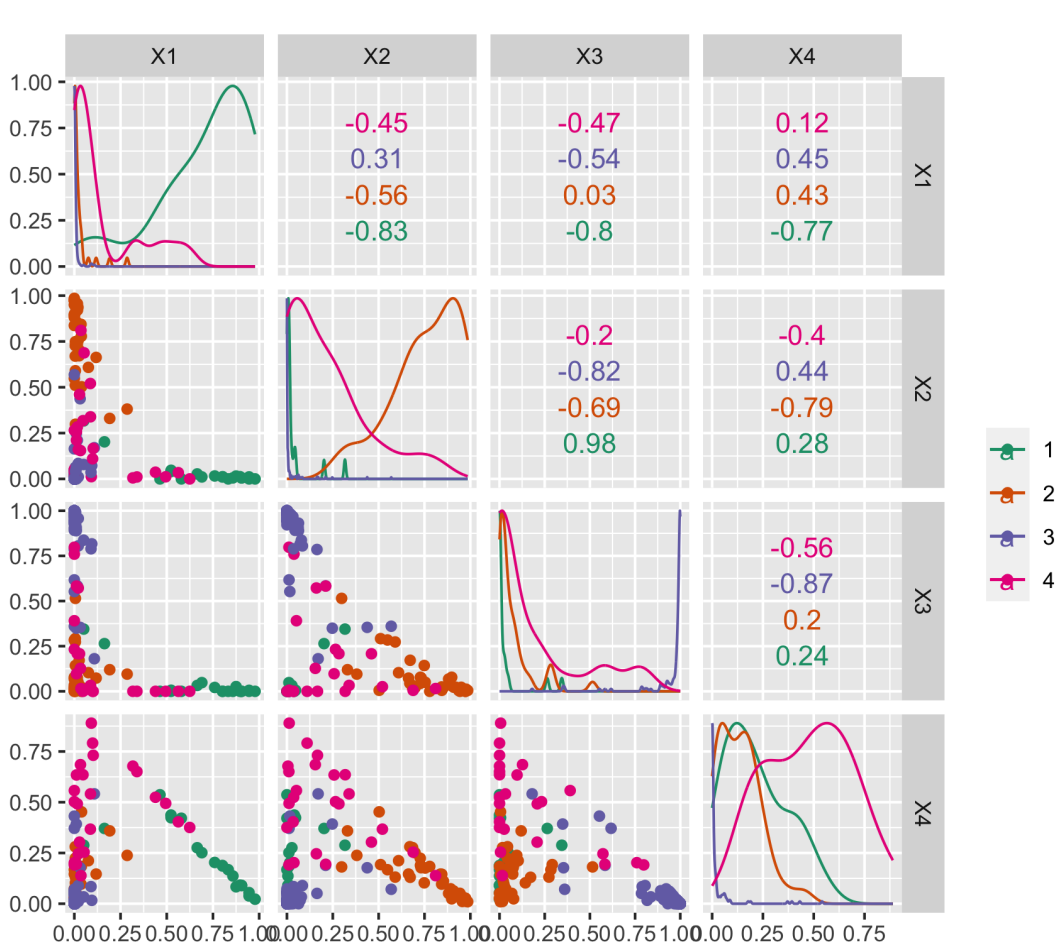


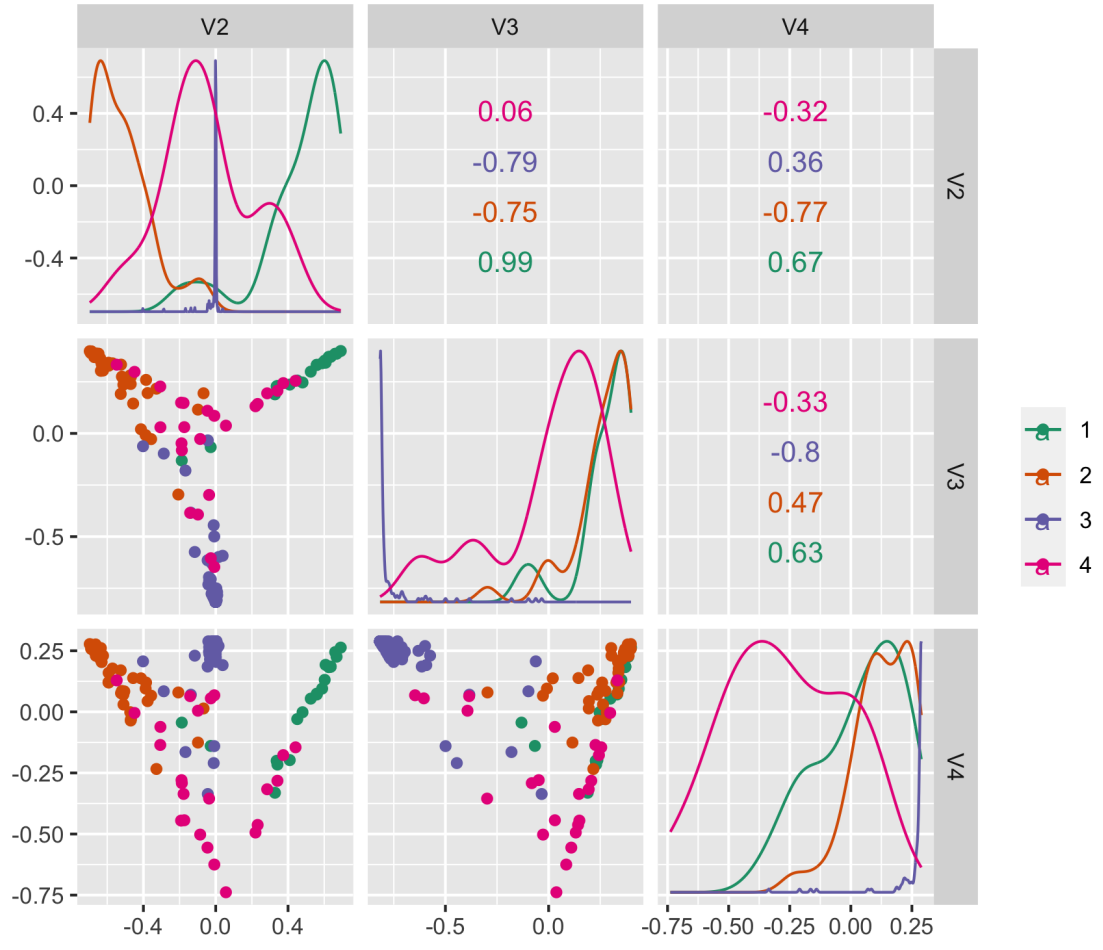
Diagnostics - vote matrix

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

Look rows 3 and 5. How confident would you be in the classifications of these two observations?

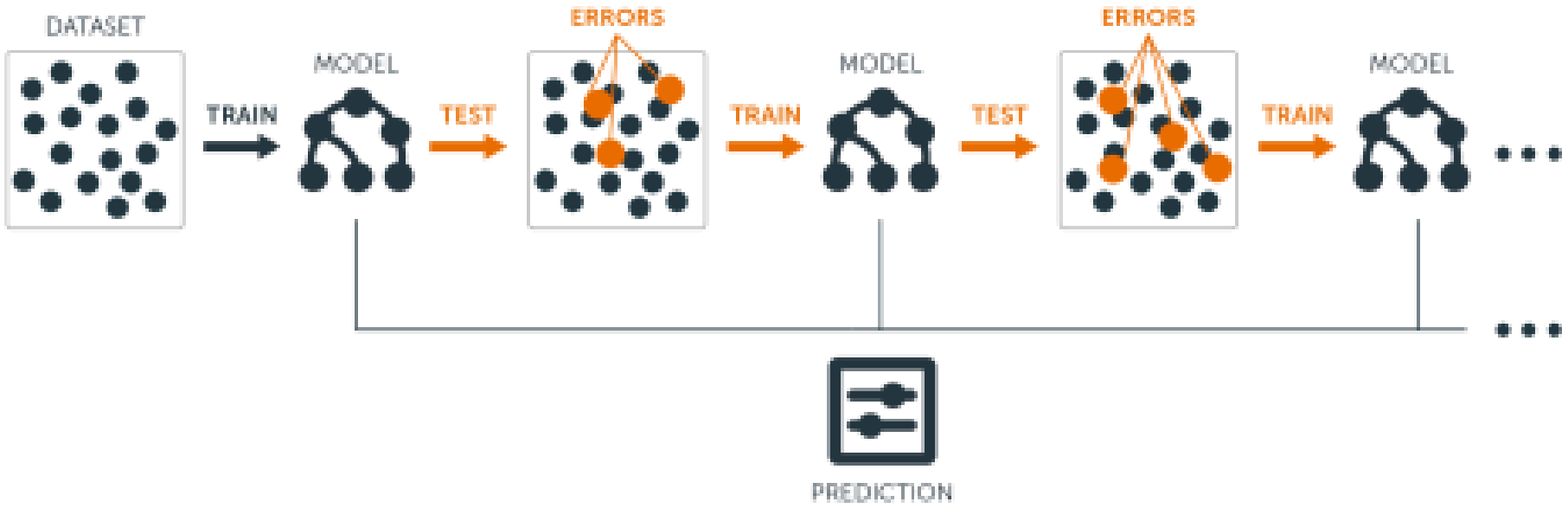
```
## # A tibble: 10 x 4
##    `1`      `2`      `3`      `4`
##    <matrix> <matrix> <matrix> <matrix>
##  1 0.6898   0.010695 0.048128 0.25134
##  2 0.7594   0.016043 0.021390 0.20321
##  3 0.9778   0.000000 0.000000 0.02222
##  4 0.8324   0.000000 0.000000 0.16757
##  5 0.4633   0.000000 0.000000 0.53672
##  6 0.5235   0.047059 0.005882 0.42353
##  7 0.5230   0.040230 0.000000 0.43678
##  8 0.9043   0.005319 0.000000 0.09043
##  9 0.8750   0.015625 0.026042 0.08333
## 10 0.8963   0.012195 0.000000 0.09146
```


X1
X2
X3
X4
-0.45
0.31
-0.56
-0.83
-0.47
-0.54
0.03
-0.8
0.12
0.45
0.43
-0.77
-0.2
-0.82
-0.69
0.98
-0.4
0.44
-0.79
0.28
-0.56
-0.87
0.2
0.24
1
2
3
4


V2
V3
V4
0.06
-0.79
-0.75
0.99
-0.32
0.36
-0.77
0.67
-0.33
-0.8
0.47
0.63
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: Boehmke (2020) 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 weighted 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, $\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$

Read a fun explanation of boosting here by Harriet Mason.

Boosting a regression tree - watch this!

StatQuest by Josh Starmer

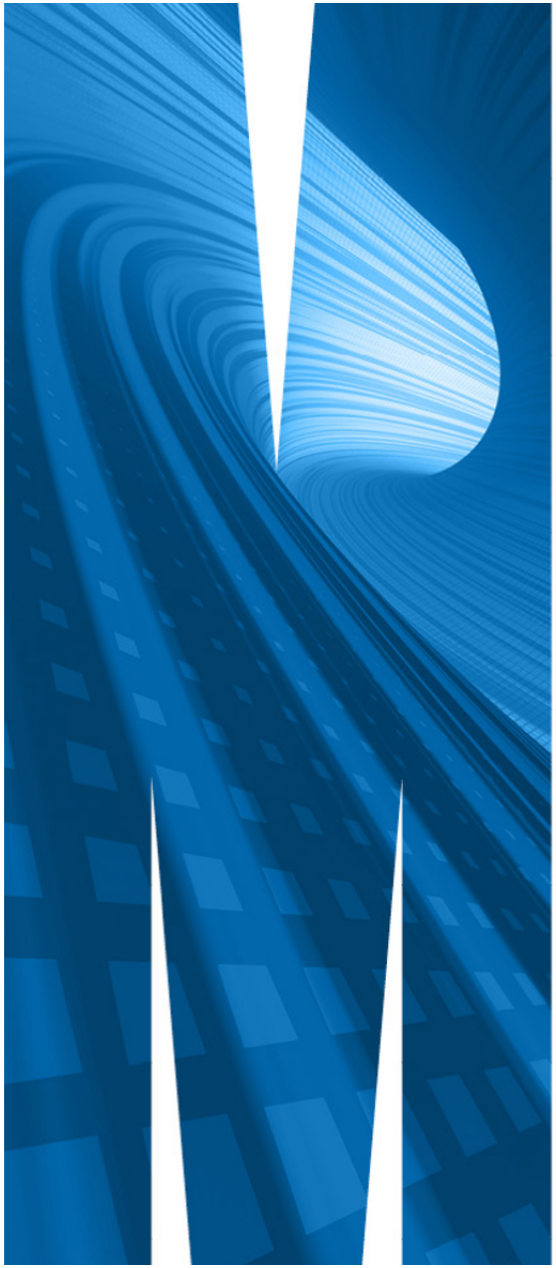
Boosting a classification tree - watch this!

StatQuest by Josh Starmer

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



Lecturer: *Professor Di Cook*

Department of Econometrics and Business Statistics

ETC3250.Clayton-x@monash.edu

Week 7a