Ensemble Algorithms

Dr Maria Anastasiadi
(m.anastasiadi@cranfield.ac.uk)

17th January 2025

www.cranfield.ac.uk

ML: Meta-Learning

Improving a model's performance with meta-learning

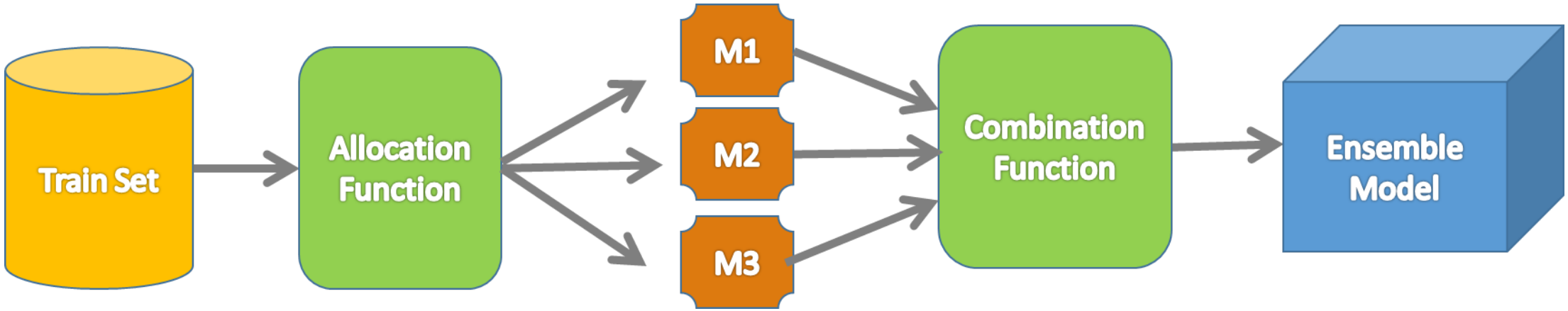
Learning how to learn.

Closely related to multi-task learning.

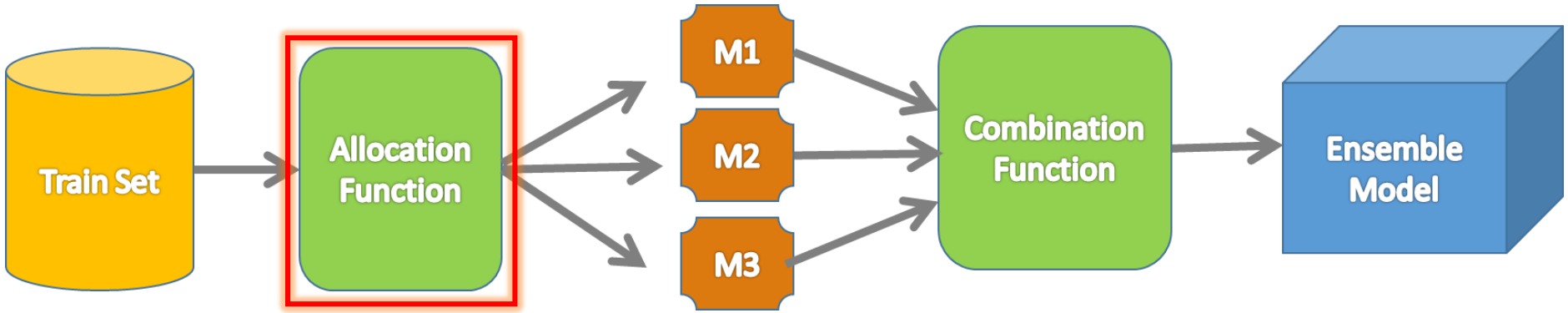
Crucial for Further Advances of Artificial Intelligence

ML: Ensemble Methods

Ensemble is the meta-learning approach which involves combining and managing the predictions of multiple models (weak learners) to create a stronger learner and boost performance.



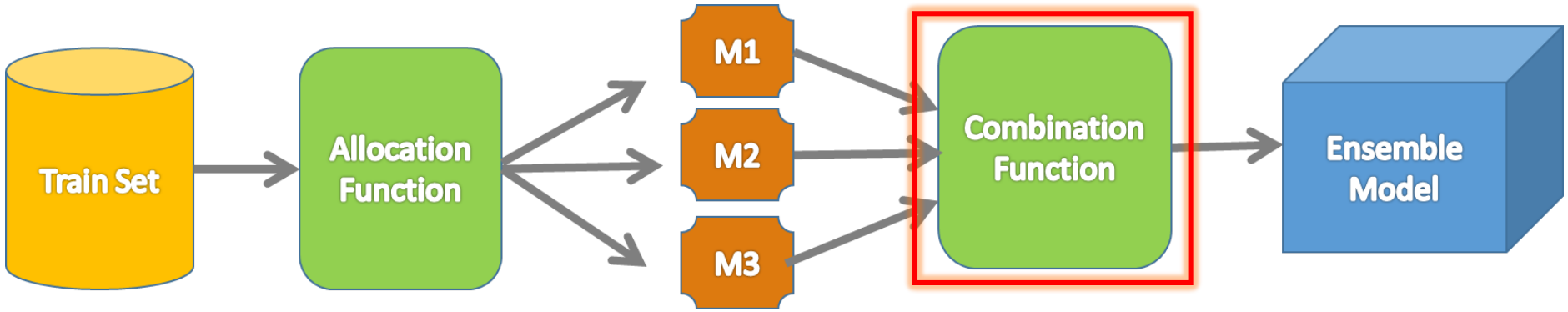
ML: Ensemble Methods



Allocation function

- Dictates how much of the training data each model receives.
- Can increase diversity by artificially varying the input data to bias the resulting learners (e.g. use bootstrap sampling, pass on different subset of features or samples to each model).
- If the ensemble already includes a diverse set of algorithms (e.g. kNN, NN, NB) the allocation function might pass the data on to each algorithm relatively unchanged.

ML: Ensemble Methods



Combination function

Governs how disagreements among the predictions are reconciled.

- Use majority vote.
- Add a weight to each model's votes based on its prior performance.
- Use another model to learn a combination function from various combinations of predictions (**stacking**).

ML: Ensemble Methods

Advantages of using ensembles over a single model

- **Better generalisability to future problems**: As the opinions of several learners are incorporated into a single final prediction, no single bias is able to dominate. This reduces the chance of overfitting to a learning task.

- **Improved performance on massive or very small datasets:** Training several small models more efficient than a single full model.

- **Better suited to difficult learning tasks**: Real-world phenomena are often extremely complex. Models that divide the task into smaller portions are likely to more accurately capture subtle patterns that a single global model might miss.

ML: Ensemble Methods

Bagging

Bootstrap aggregating (Bagging) was one of the first ensemble methods to gain widespread acceptance (Leo Breiman, 1994).

- Bagging generates a number of training datasets by bootstrap sampling the original training data. These datasets are then used to generate a set of models using a single learning algorithm.
- The models' predictions are combined using voting (for classification) or averaging (for numeric prediction).
- Bagging performs well when used with relatively **unstable** learners such as decision trees.
- Unstable models are essential in order to ensure the ensemble's diversity in spite of only minor variations between the bootstrap training datasets.

ML: Ensemble Methods

Boosting

Boosts the performance of weak learners to attain the performance of stronger learners (R. Schapire & Y. Freund).

- Similar to bagging, boosting uses ensembles of models trained on resampled data and a vote to determine the final prediction.

Key differences to bagging:

- The resampled datasets in boosting are constructed specifically to generate **complementary** learners.
- Boosting gives each learner's vote a **weight** based on its past performance.
- Models that perform better have greater influence over the ensemble's final prediction.

ML: Ensemble Methods

Boosting

- Boosting will result in performance that is often better than the best of the models in the ensemble.
- It is possible to increase ensemble performance simply by adding additional classifiers to the group, assuming that each classifier performs better than random chance.

⚠ Caution!
Constantly adding learners results in small gains in accuracy and may result in the model being over-fitted!

ML: Ensemble Methods

AdaBoost

- **AdaBoost or adaptive boosting** (Freund & Schapire, 1997) is an algorithm focused on generating weak learners that iteratively learn a larger portion of the **difficult-to-classify** examples by giving more weight to frequently misclassified examples.
- AdaBoost was the first really successful boosting algorithm developed for binary classification. Modern boosting methods are build on AdaBoost, most notably stochastic gradient boosting machines.
- AdaBoost can be used to boost the performance of any machine learning algorithm but is best used with decision trees.

ML: Ensemble Methods

adabag package in R

AdaBoost

How it works

1. Beginning from an unweighted dataset, the first classifier attempts to model the outcome.

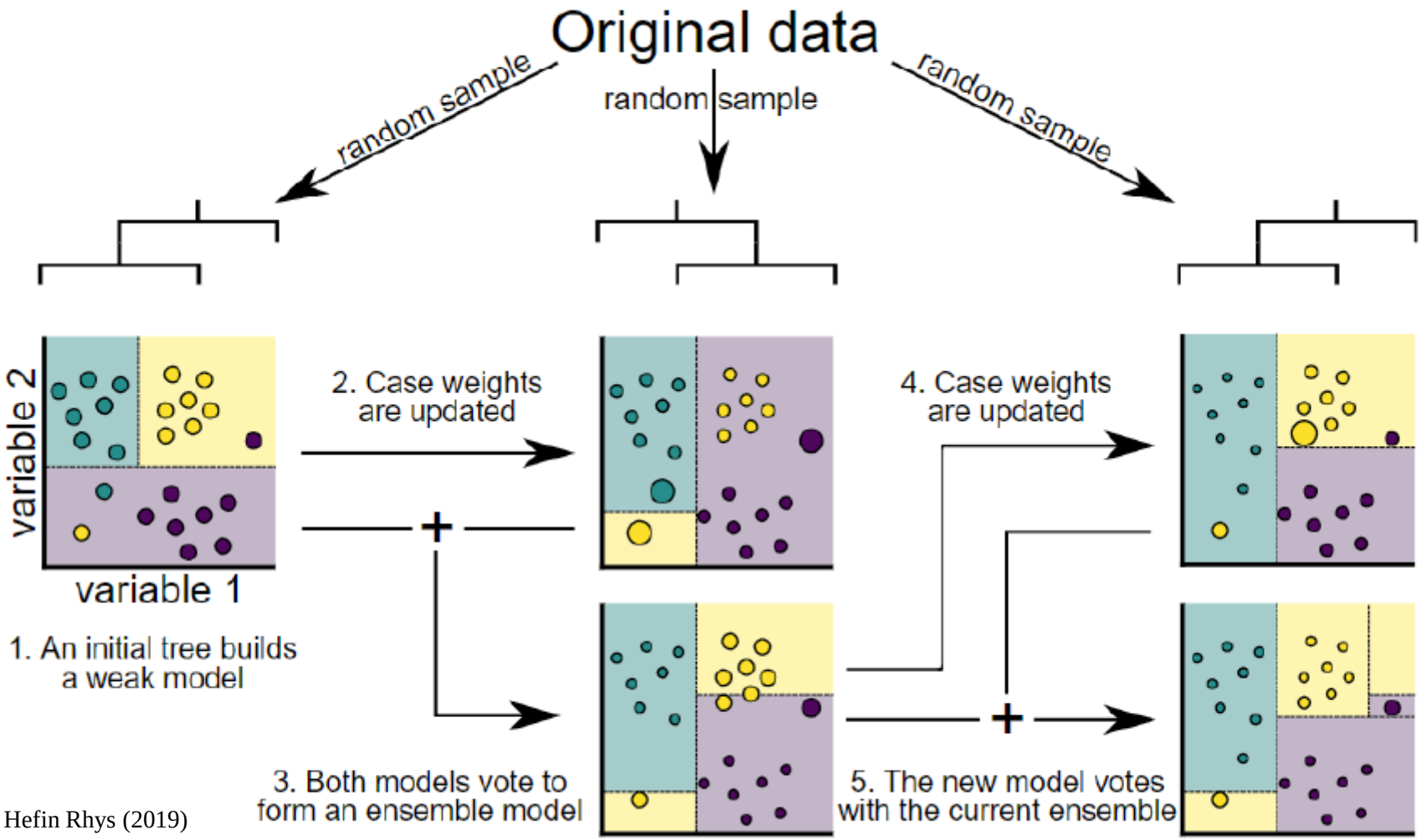
2. Instances which the classifier predicted correctly will be less likely to appear in the training dataset for the following classifier. As a result, the difficult-to-classify examples will appear more frequently.

3. As additional rounds of weak learners are added, they are trained on data with successively more difficult examples.

4. The process continues until the desired overall error rate is reached or performance no longer improves.

ML: Ensemble Methods

AdaBoost with Decision Trees



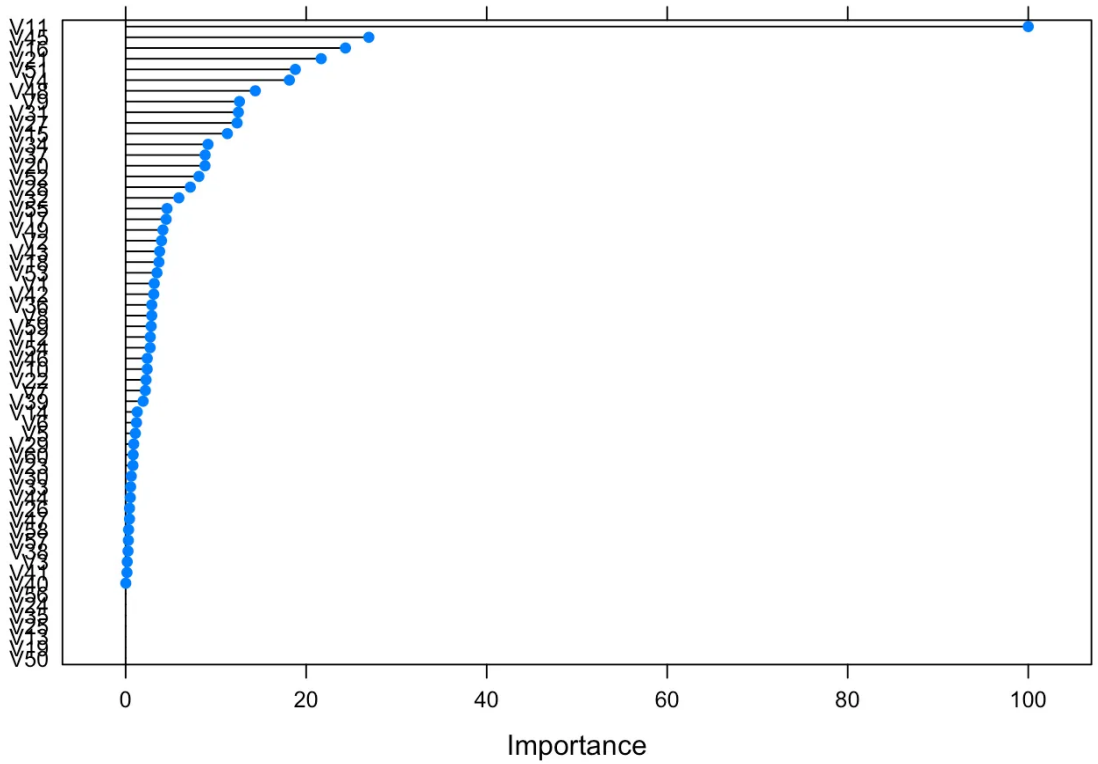
Schematic: Hefin Rhys (2019)

ML: Ensemble Methods

Calculating Variable Importance for XGBoost

- Feature importance is calculated for a single decision tree by the amount that a split on this feature improves the performance measure, weighted by the number of observations the node is responsible for.

- The final feature importance is then averaged across all the decision trees within the model.



ML: Ensemble Methods

- Use *caret* to train xgboost model:

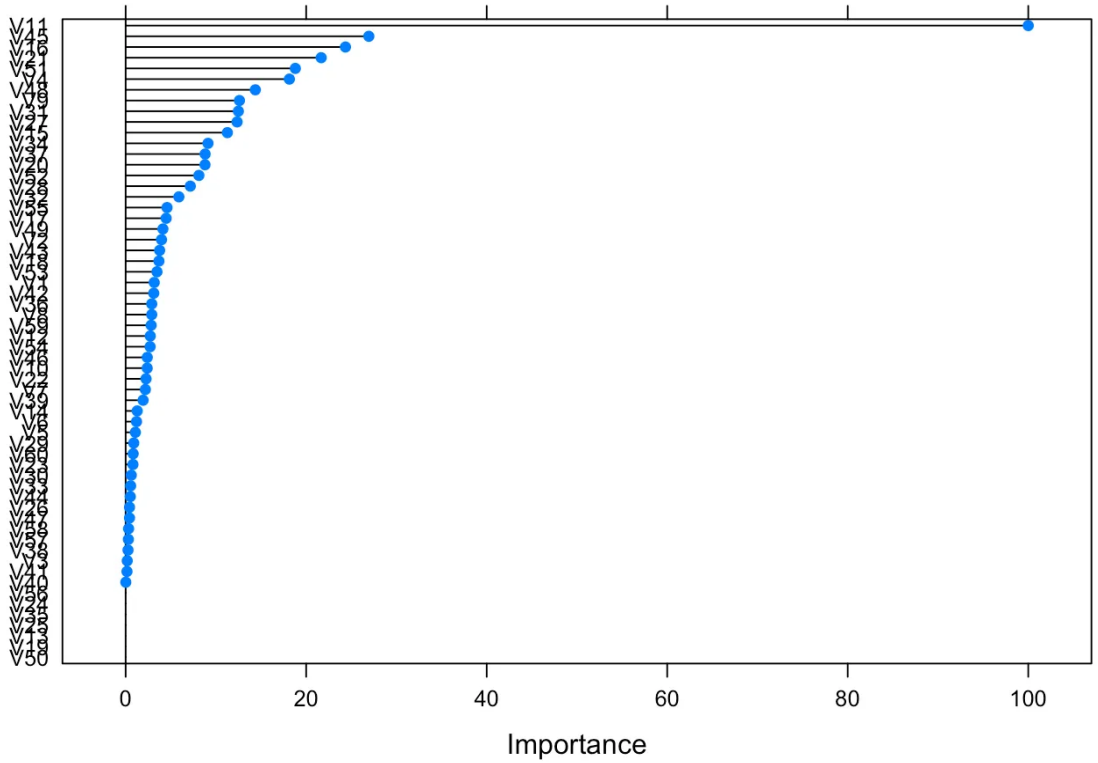
```
xgb_fit <- train(Class ~ .,
                 data = Data,
                 method = "xgbLinear")
```

- Extract Variable Importance

```
caret_imp <- varImp(xgb_fit)
caret_imp
```

- Plot Variable Importance

```
plot(caret_imp)
```



ML: Ensemble Methods

Example of using grid search in a gradient boosting machine (GBM) model.

There are four main tuning parameters:

- **n.trees** = number of iterations, i.e. trees,
- **interaction.depth** = complexity of the tree (the maximum depth of each tree). Default is 1.
- **shrinkage** = a shrinkage parameter applied to each tree in the expansion (learning rate: how quickly the algorithm adapts)
- **n.minobsinnode** = the minimum number of observations in the terminal nodes of the trees

```
expand.grid(interaction.depth = c(1, 5, 9),
                        n.trees = (1:30)*50,
                        shrinkage = 0.1,
                        n.minobsinnode = 20)
```

ML: Ensemble Methods

Random Forests (or Decision Tree Forests)

An ensemble-based method which focuses only on ensembles of *decision trees* (Leo Breiman 2001).

- Random forests combine versatility and power into a single machine learning approach.
- As the ensemble uses only a small, random portion of the full feature set, random forests can handle extremely large datasets.
- At the same time, its error rates for most learning tasks are on par with nearly any other method.

Due to their power, versatility, and ease of use, random forests are quickly becoming one of the most popular machine learning methods.

ML: Ensemble Methods

`randomForest` package in R

Random Forests

How it works (Andy Liaw & Matthew Wiener, 2002, *R News*)

Step1. Draw **n** bootstrap samples from the original data.

Step2. For each of the bootstrap samples, grow an unpruned classification or regression tree, with the following modification:

At each node, rather than choosing the best split among all predictors, randomly sample $\mathbf{m_{try}}$ of the predictors and choose the best split from among those variables.

Step3. Predict new data by aggregating the predictions of the **n** trees (i.e., majority votes for classification, average for regression).

ML: Ensemble Methods

Random Forests

Error rate

To calculate the error rate from the training data we follow these steps:

1) At each bootstrap iteration, use the tree grown with the bootstrap sample to predict the data not included in the bootstrap sample (“out-of-bag”, or OOB, data).
2) **Aggregate the OOB predictions**. Calculate the error rate and call it the OOB estimate of error rate.

ML: Ensemble Methods

Example of Random Forests hyperparameter tuning with *caret*

```
1. Define hyperparameter tuning settings
grid <-expand.grid(mtry=2:40)

control <- trainControl(method='repeatedcv',
                        number=10, repeats=5,
                        search='grid')


2. Train a Random Forests classification model
rf.model <- train(Class~., data = TrainSet,
                  method = 'rf',
                  metric = 'Accuracy',
                  ntrees = 800,
                  tuneGrid = grid,
                  trControl = control )
```

Random Forests

Extra Information

The **`randomForest`** package optionally produces two additional pieces of information:

a) **Variable importance**: a measure of the importance of the predictor variables.

b) **Proximity measure**: a measure of the internal structure of the data (the proximity of different data points to one another).

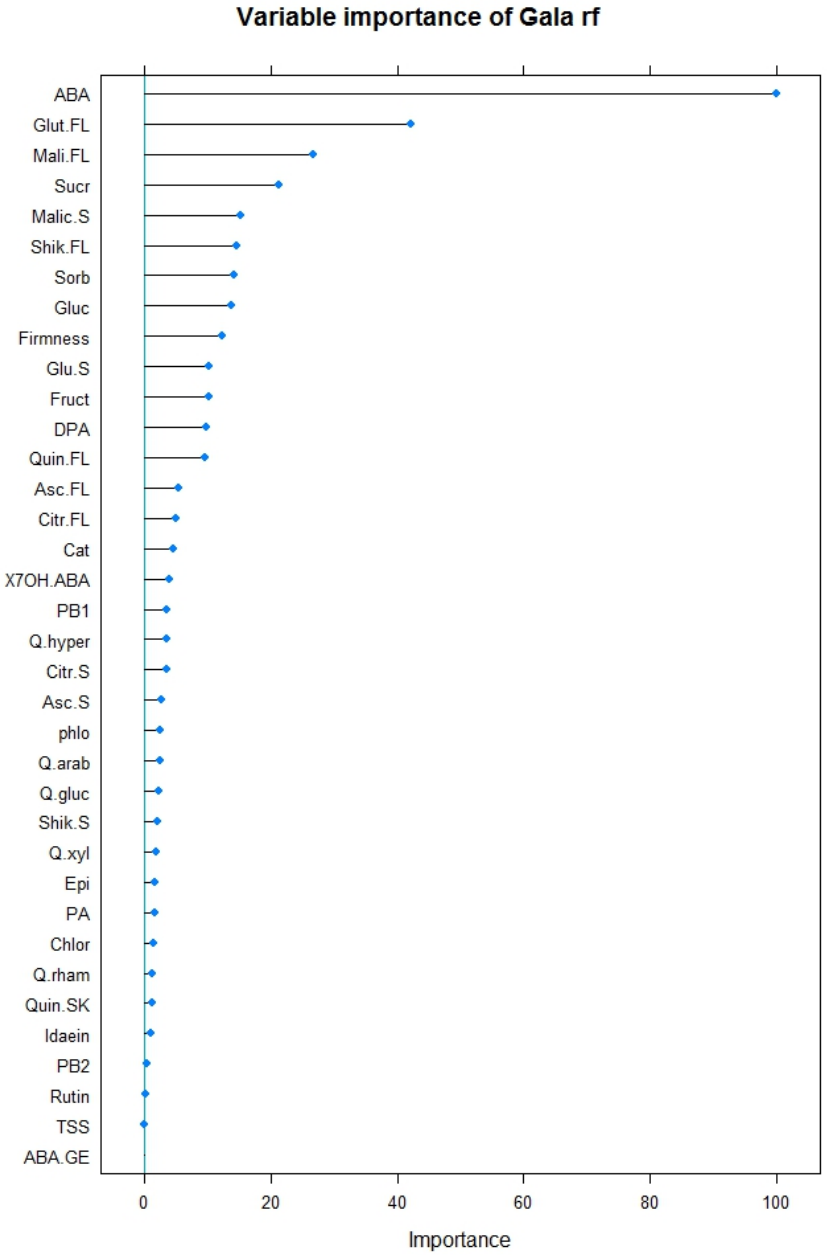
ML: Ensemble Methods

Random Forests

How to calculate variable importance :

Variable importance: The random forest algorithm estimates the importance of a variable by looking at how much the prediction error increases when OOB data for that variable is permuted while all others are left unchanged (Breiman 2002).

Important to remember that the importance of a variable may be due to its (possibly complex) interaction with other variables.



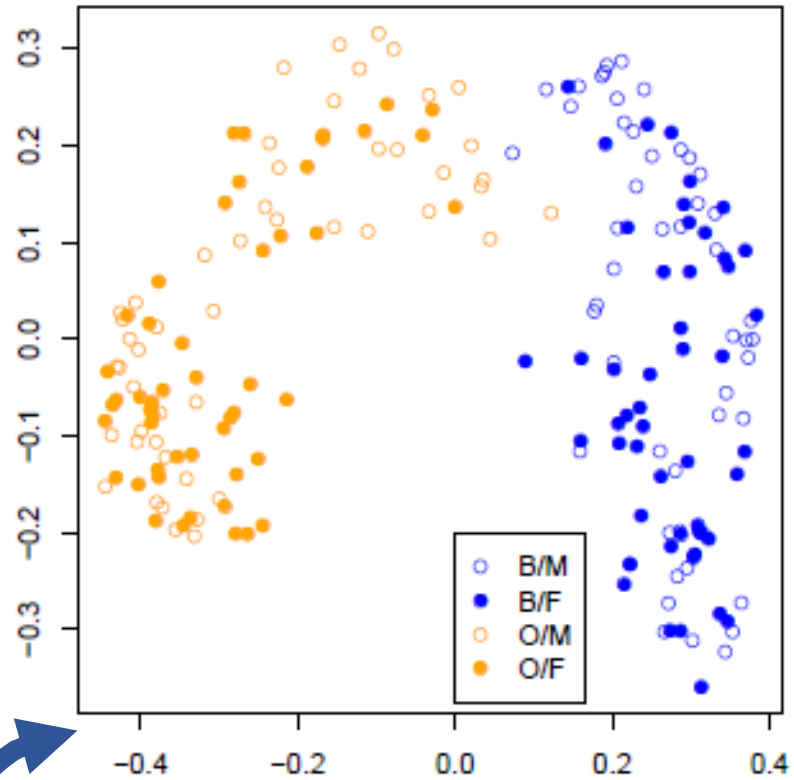
ML: Ensemble Methods

Random Forests

How to calculate variable proximity measure:

Proximity measure:

- The **`randomForest`** package produces a proximity matrix.
- Each (i, j) element of the proximity matrix is the fraction of trees in which elements i and j fall in the same terminal node.
- The intuition is that “similar” observations should be in the same terminal nodes more often than dissimilar ones.
- Useful for *unsupervised* learning applications.



A scatterplot of the (1-proximity) matrix scaled using a multidimensional scaling method. (A. Liaw & M.Wiener, 2002, *R News)*

ML: Ensemble Methods

Random Forests

Strengths	Weaknesses
An all-purpose model that performs well on most problems	The model outcome is not easily interpretable unlike a decision tree
Can handle noisy or missing data	May take some time to tune the model to the data
Only selects the most important features	
Effective on massive datasets with large number of features and/or samples	

Summary

- Principles of Ensemble methods
- Main Ensemble Techniques:
  - Bagging
  - Boosting
  - Adaboost
  - Random Forest

www.cranfield.ac.uk

T: +44 (0)1234 750111

@cranfielduni

@cranfielduni

/cranfielduni