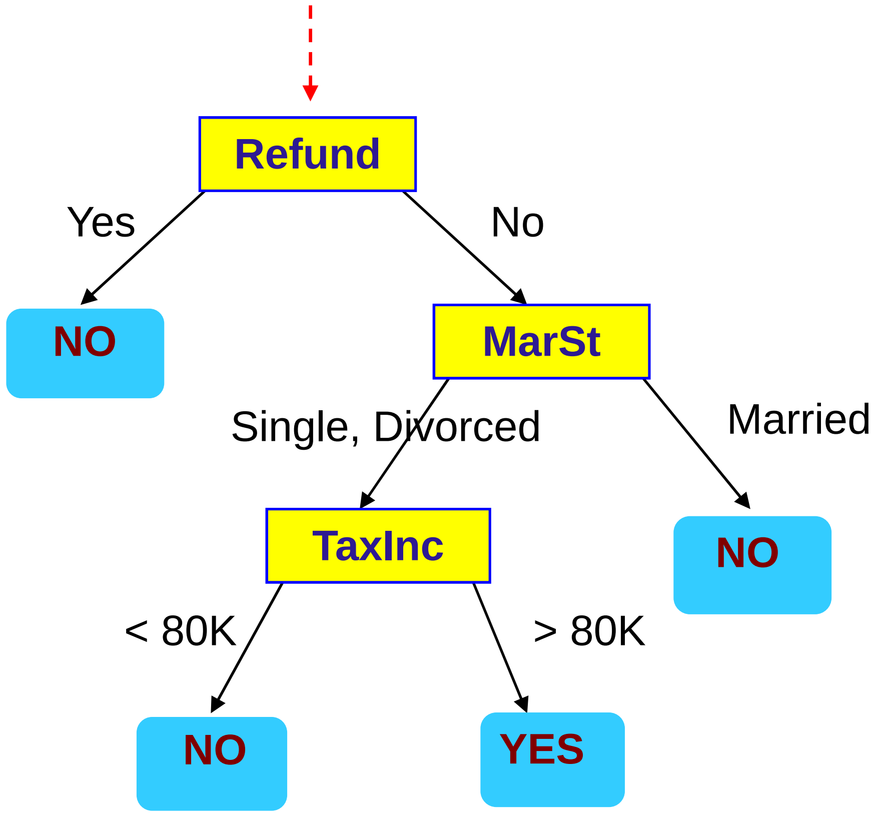
Introduction to the Random Forest method

Zurich University of Applied Sciences

- **First look at trees**
- **Ensemble idea: bagging, boosting**
- **Random Forest**
  - **tree bagging**
  - **evaluation -> out of bag (oob) error**
  - **variable importance measures -> variable selection**
  - **proximity measure**
    - from supervised random forest
    - from unsupervised random forest
  - **imputation**

Example of a classification tree

Start from the root of tree.



Test Data

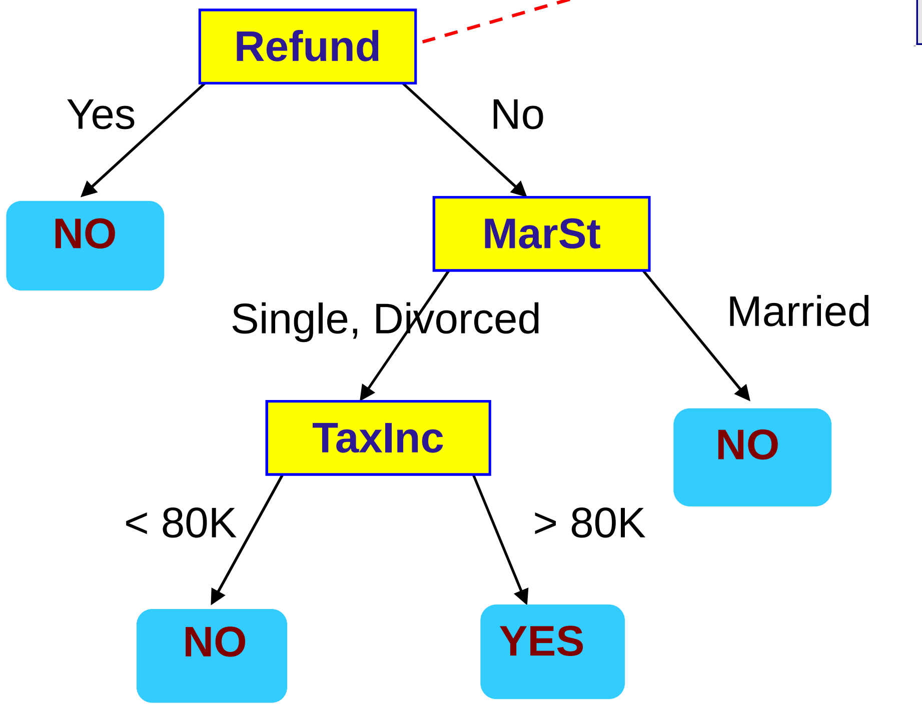
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	**?**

The response is categorical

Example:
Model "cheat" by some properties of the insured person (tax, sex, …)

Source: Tan, Steinbach, Kumar

Apply Model to Test Data

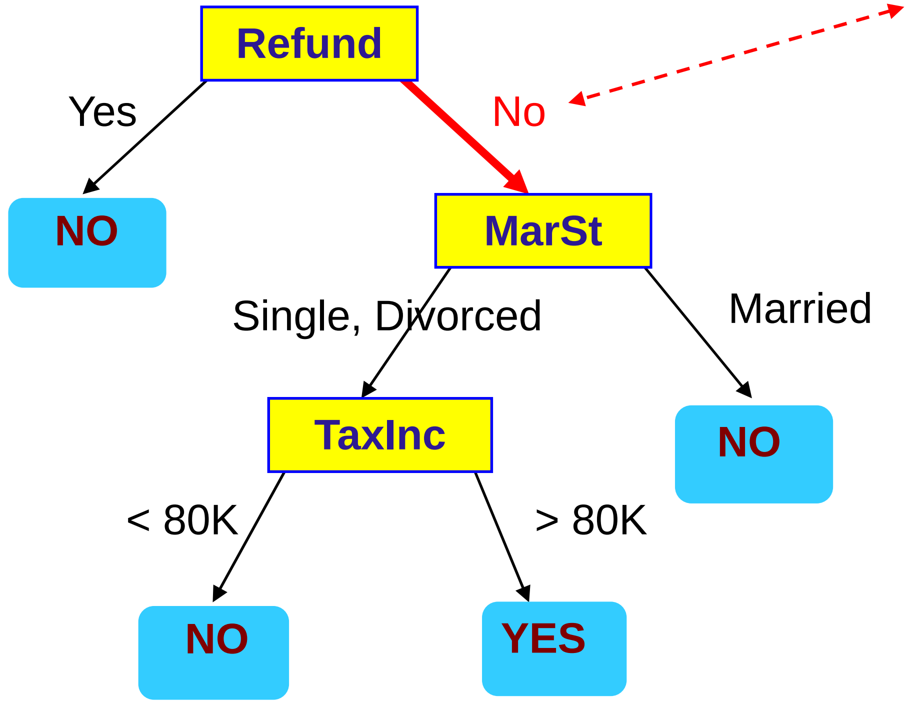


Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	**?**

Source: Tan, Steinbach, Kumar

Apply Model to Test Data

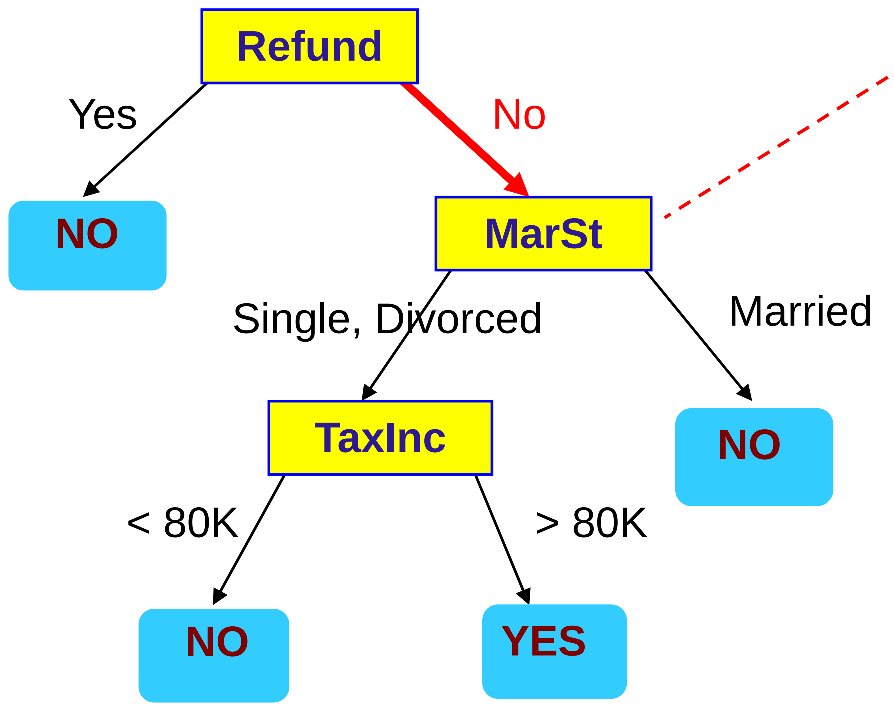


Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Source: Tan, Steinbach, Kumar

Apply Model to Test Data



Test Data

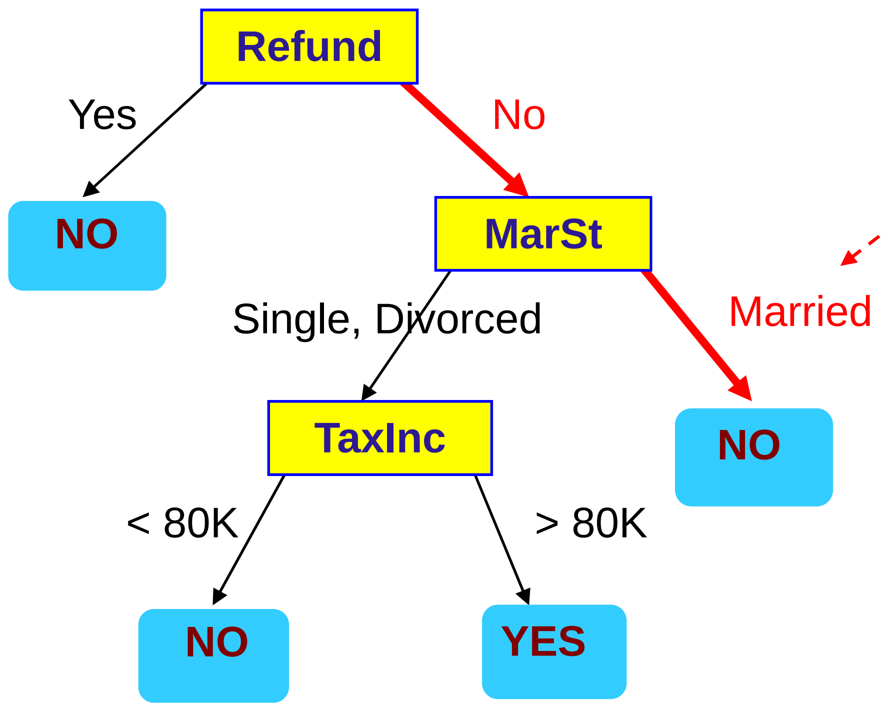
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Source: Tan, Steinbach, Kumar

Apply Model to Test Data

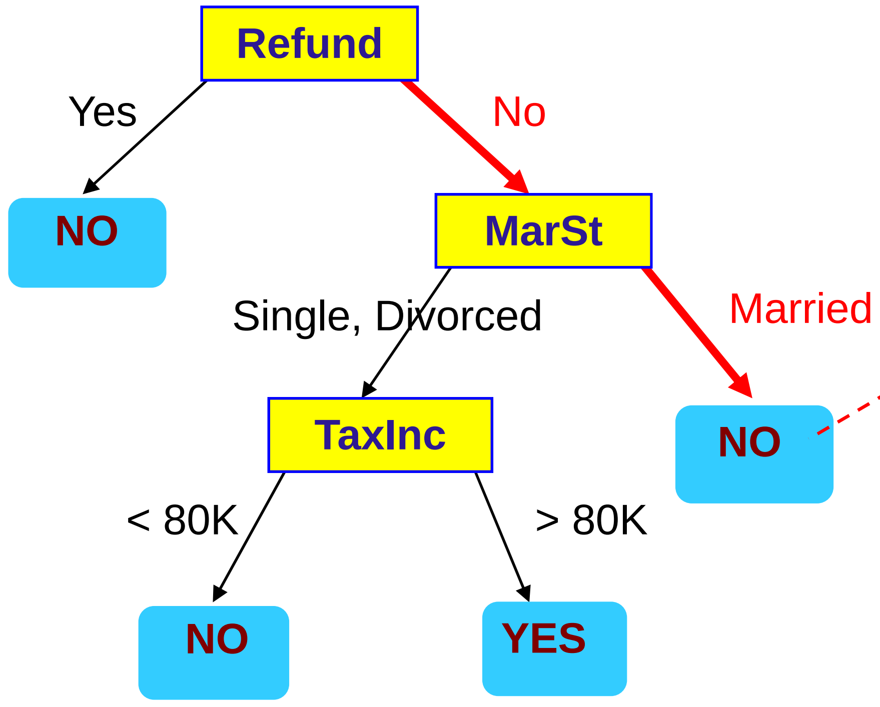
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Source: Tan, Steinbach, Kumar

Apply Model to Test Data



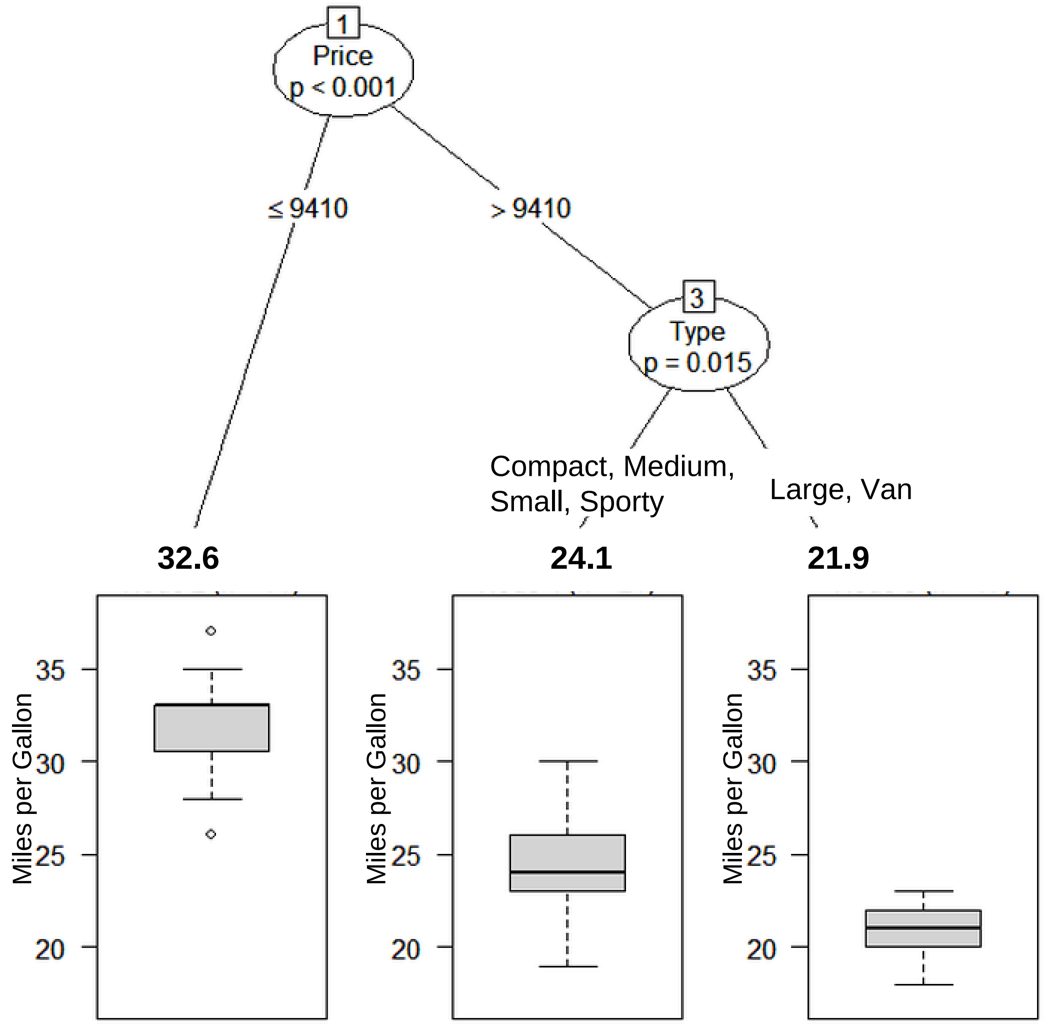
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to “No”

Source: Tan, Steinbach, Kumar

Example of a regression tree



The response is continuous

Example:
Model mileage by the car properties (price, type, …)

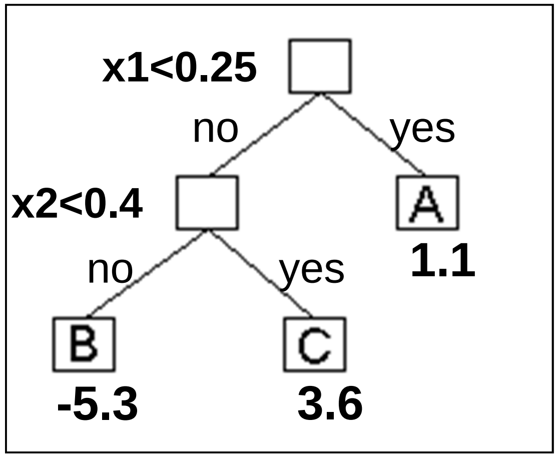
How to find the tree structure of a regression tree?

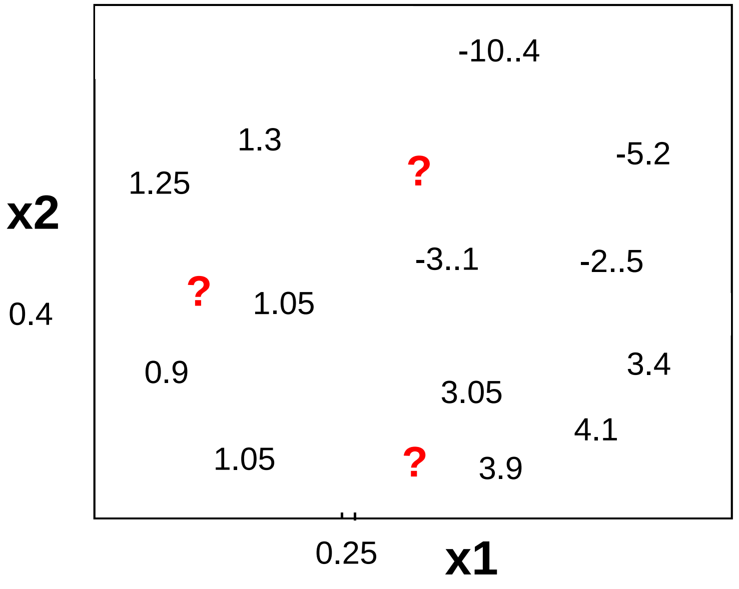
- Starting with a single region -- i.e., all given data
- At the m-th iteration:

```
for each region $R$
    for each attribute $x_j$ in $R$
        for each possible split $s_j$ of $x_j$
            record change in score when we partition $R$ into $R^l$ and $R^r$
Choose $(x_j , s_j)$ giving maximum improvement to fit
Replace $R$ with $R^l$; add $R^r$
```

Score: MSE (mean squared error)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2$$



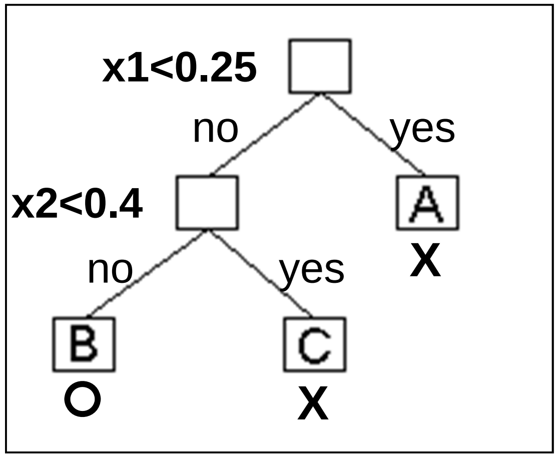


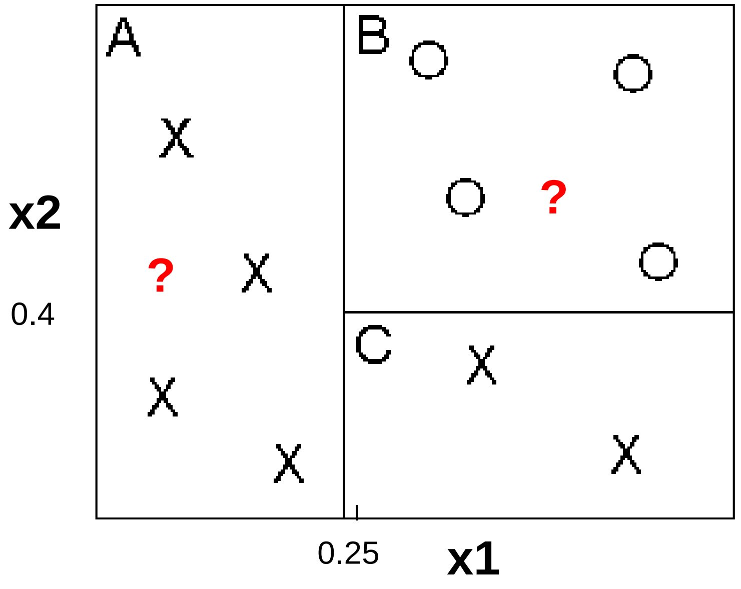
How to find the tree structure of a classification tree?

- Starting with a single region -- i.e., all given data
- At the m-th iteration:

```
for each region $R$
    for each attribute $x_j$ in $R$
        for each possible split $s_j$ of $x_j$
            record change in score when we partition $R$ into $R^l$ and $R^r$
Choose $(x_j, s_j)$ giving maximum improvement to fit
Replace $R$ with $R^l$; add $R^r$
```

Score: Missclassification rate





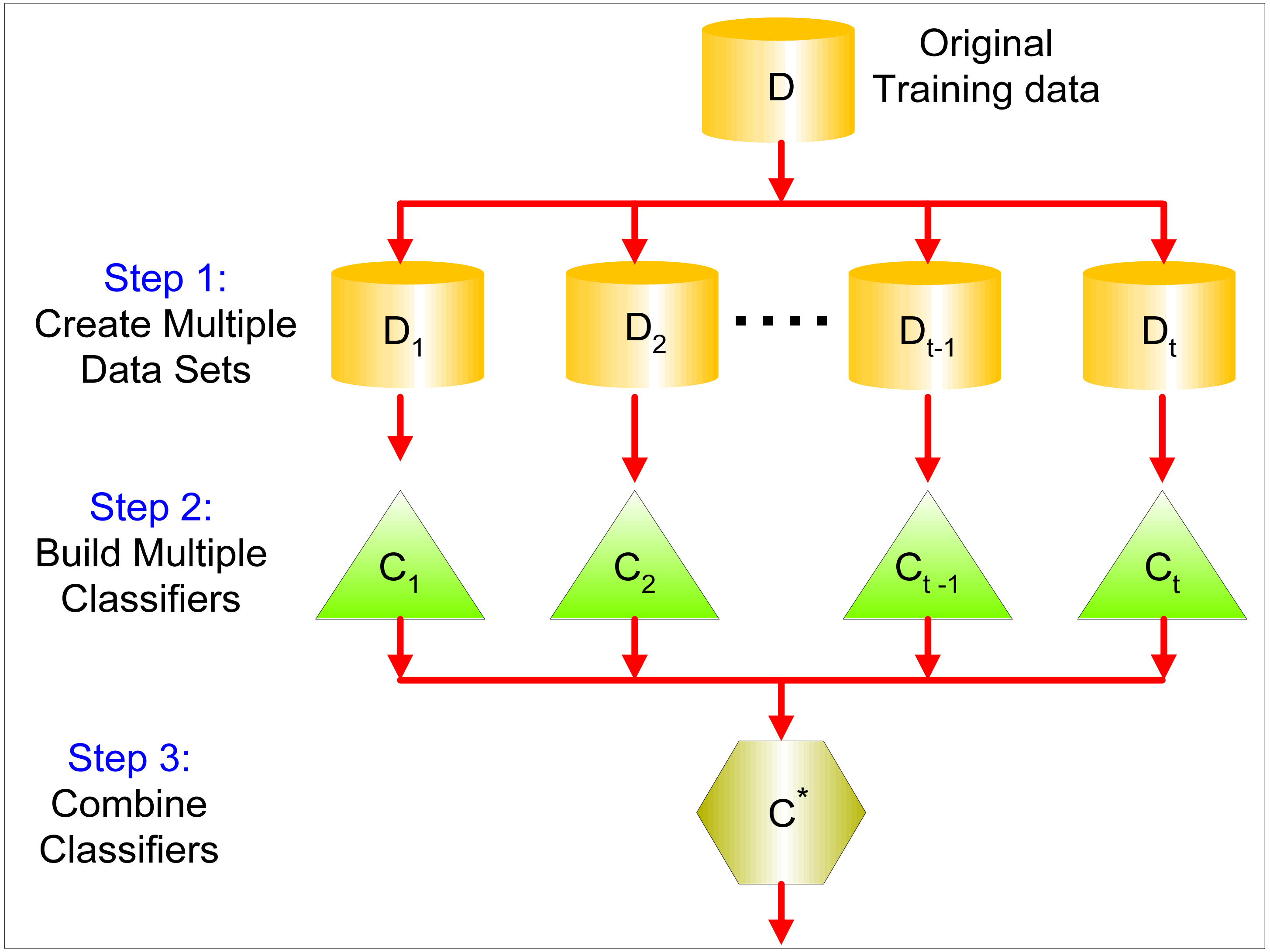
Pros and cons of tree models

- Pros
  - Interpretability
  - Robust to outliers in the input variables
  - No distribution assumptions
  - No variable transformation necessary (invariant to monoton trafos)
  - Can capture non-linear structures
  - Can capture local interactions very well
  - Low bias if appropriate input variables are available and tree has sufficient depth.
- Cons
  - High variation (instability of trees)
  - response surface is not smooth
  - Tend to overfit
  - Needs big datasets to capture additive structures
  - Inefficient for capturing linear structures

Ensemble Methods

- Construct a set of classifiers from the training data
- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers

General Idea

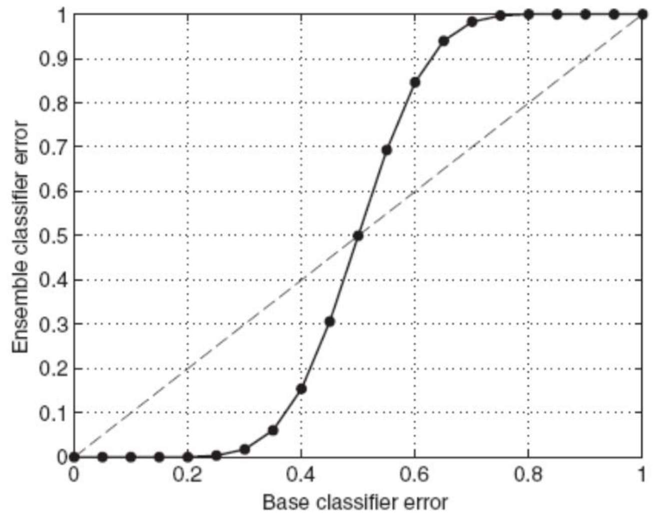


Source: Tan, Steinbach, Kumar

Why does it work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, ε = 0.35
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction (that is if >50%, here 13 or more classifiers out of 25 make a wrong prediction)

$$P(\text{wrong prediction}) = \sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$



=> Ensembles are only better than one classifier, if each classifier is better than random guessing!

Source: Tan, Steinbach, Kumar

Which supervised learning method is best?

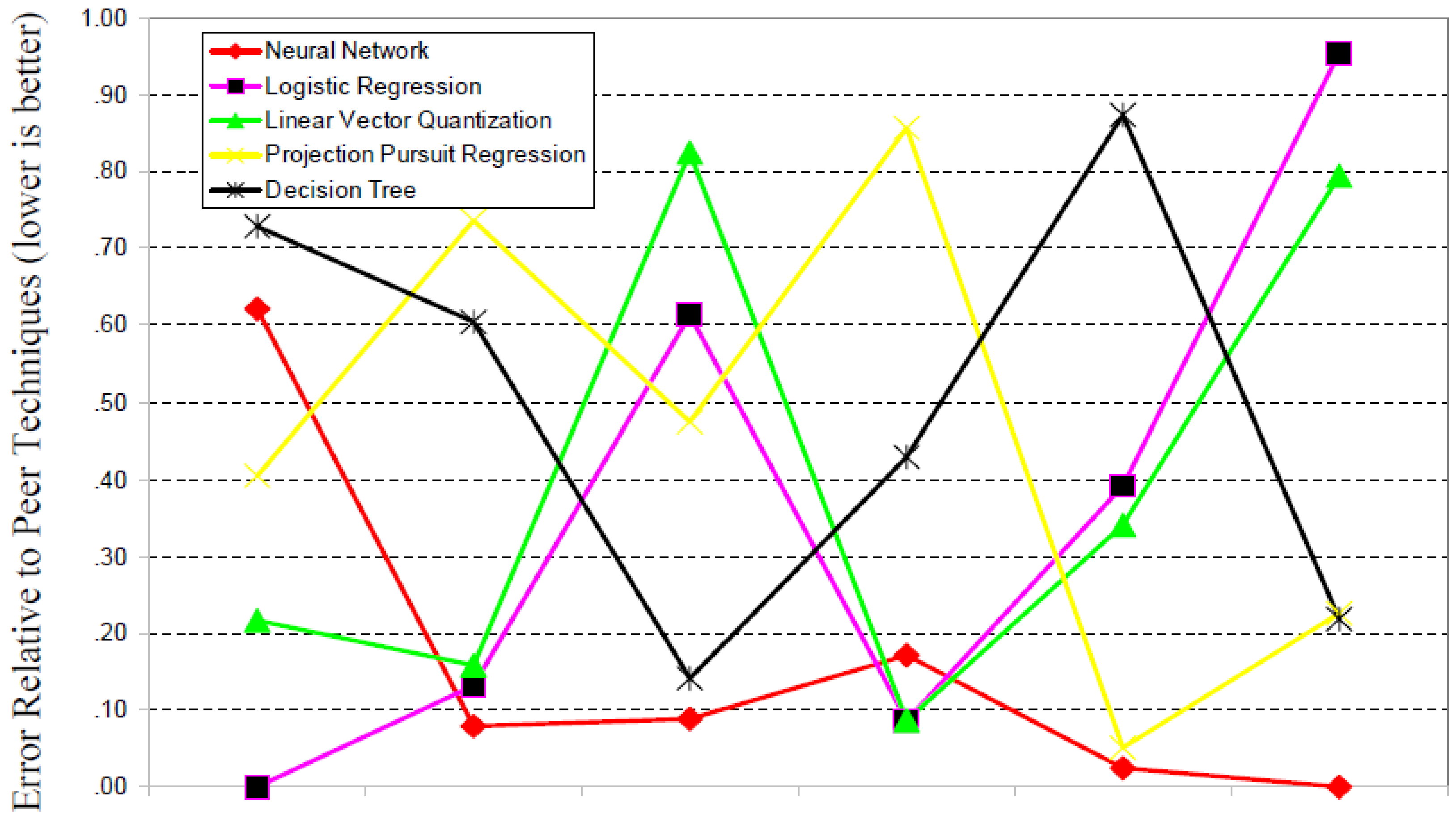


Bild: Seni & Elder

Bundling improves performance

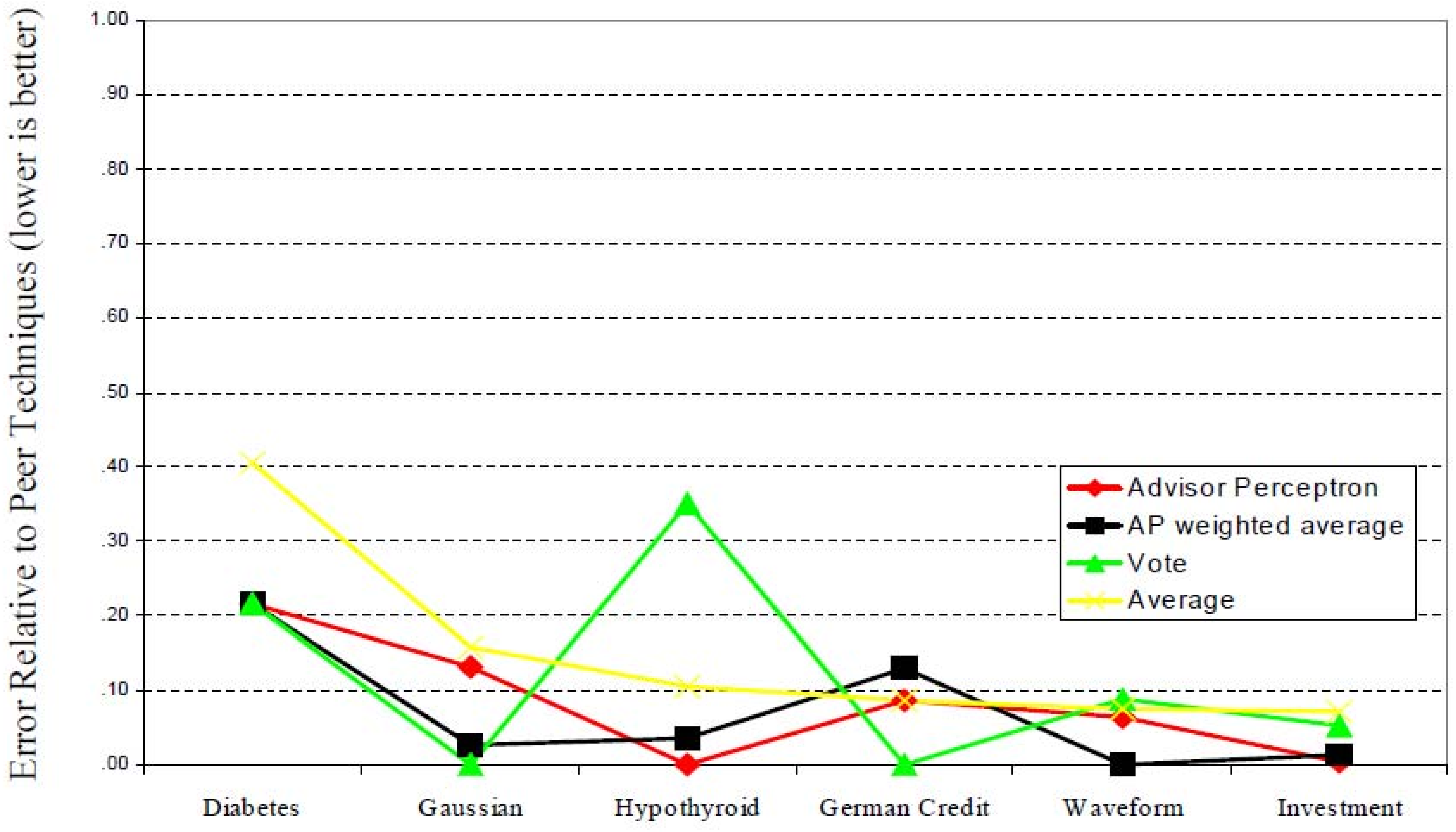
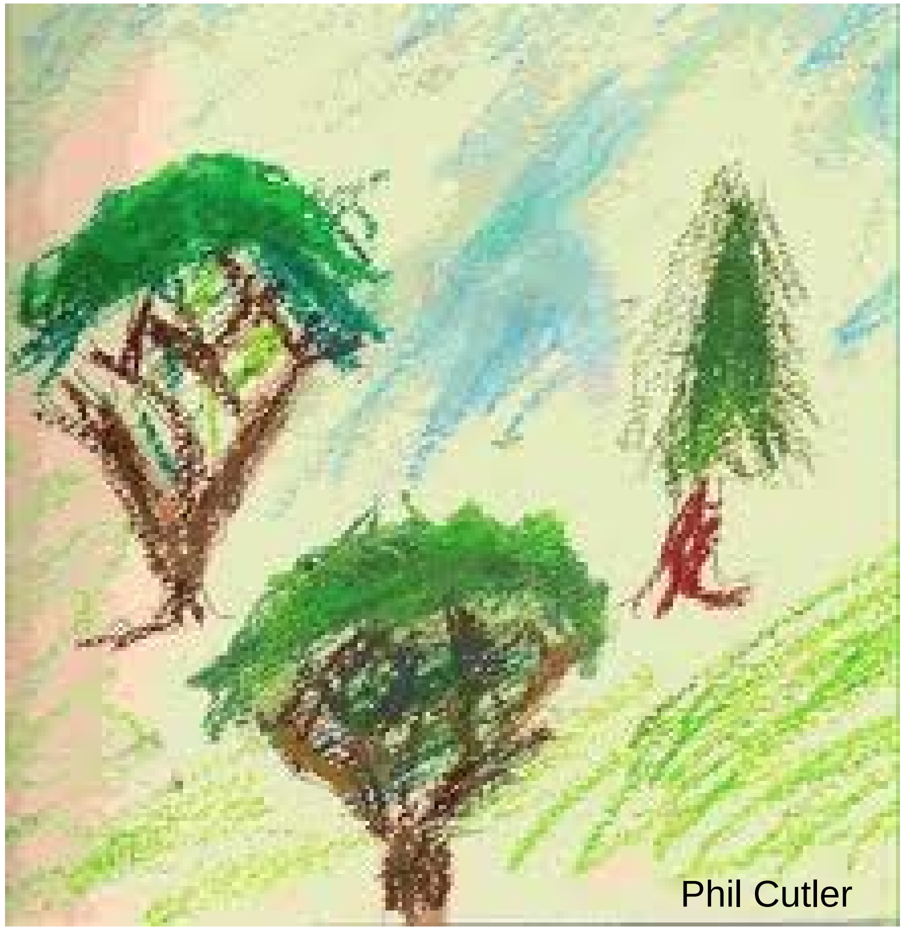
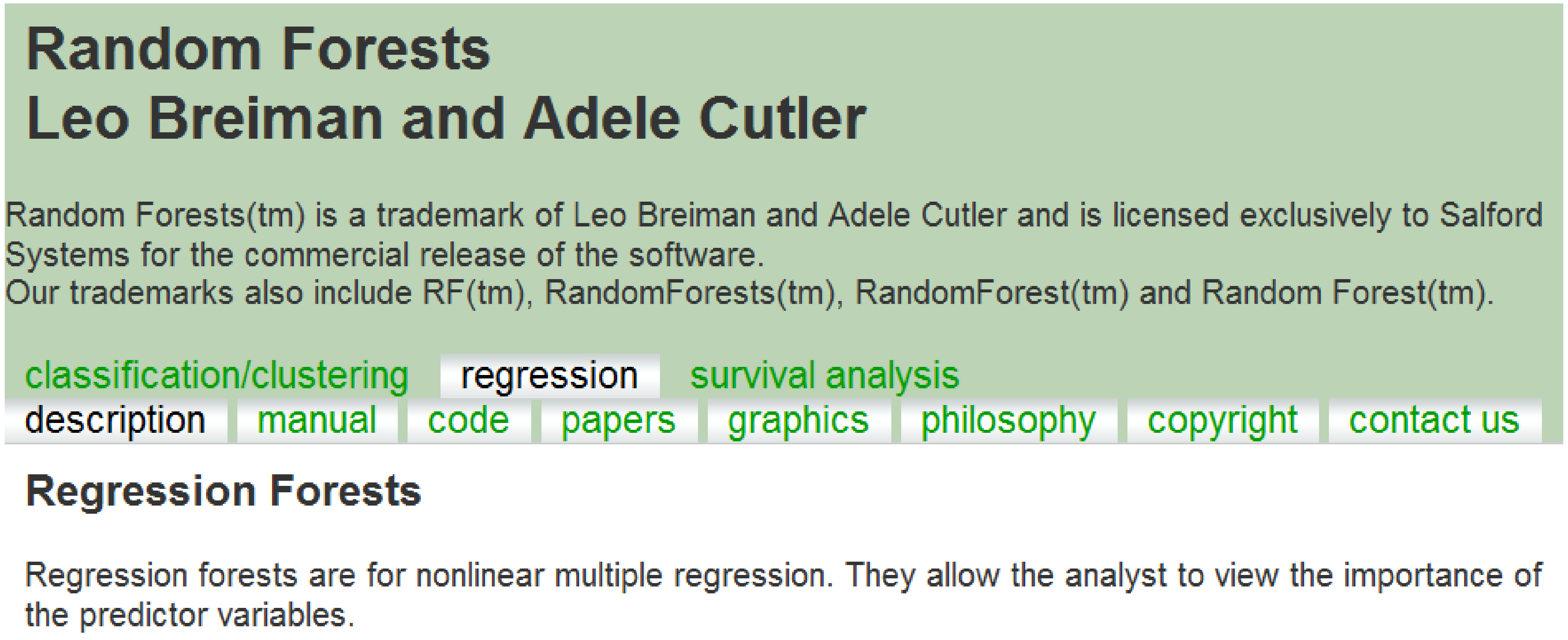


Bild: Seni & Elder

Random Forest as ensemble method for classification and regression



Random Forest: Development 1999-2001



Random Forests
Leo Breiman and Adele Cutler

Random Forests(tm) is a trademark of Leo Breiman and Adele Cutler and is licensed exclusively to Salford Systems for the commercial release of the software.
Our trademarks also include RF(tm), RandomForests(tm), RandomForest(tm) and Random Forest(tm).

Regression Forests

Regression forests are for nonlinear multiple regression. They allow the analyst to view the importance of the predictor variables.

In R: library(randomForest)
Good article: http://www.r-project.org/doc/Rnews/Rnews_2002-3.pdf

„Random Forest“ as ensemble method

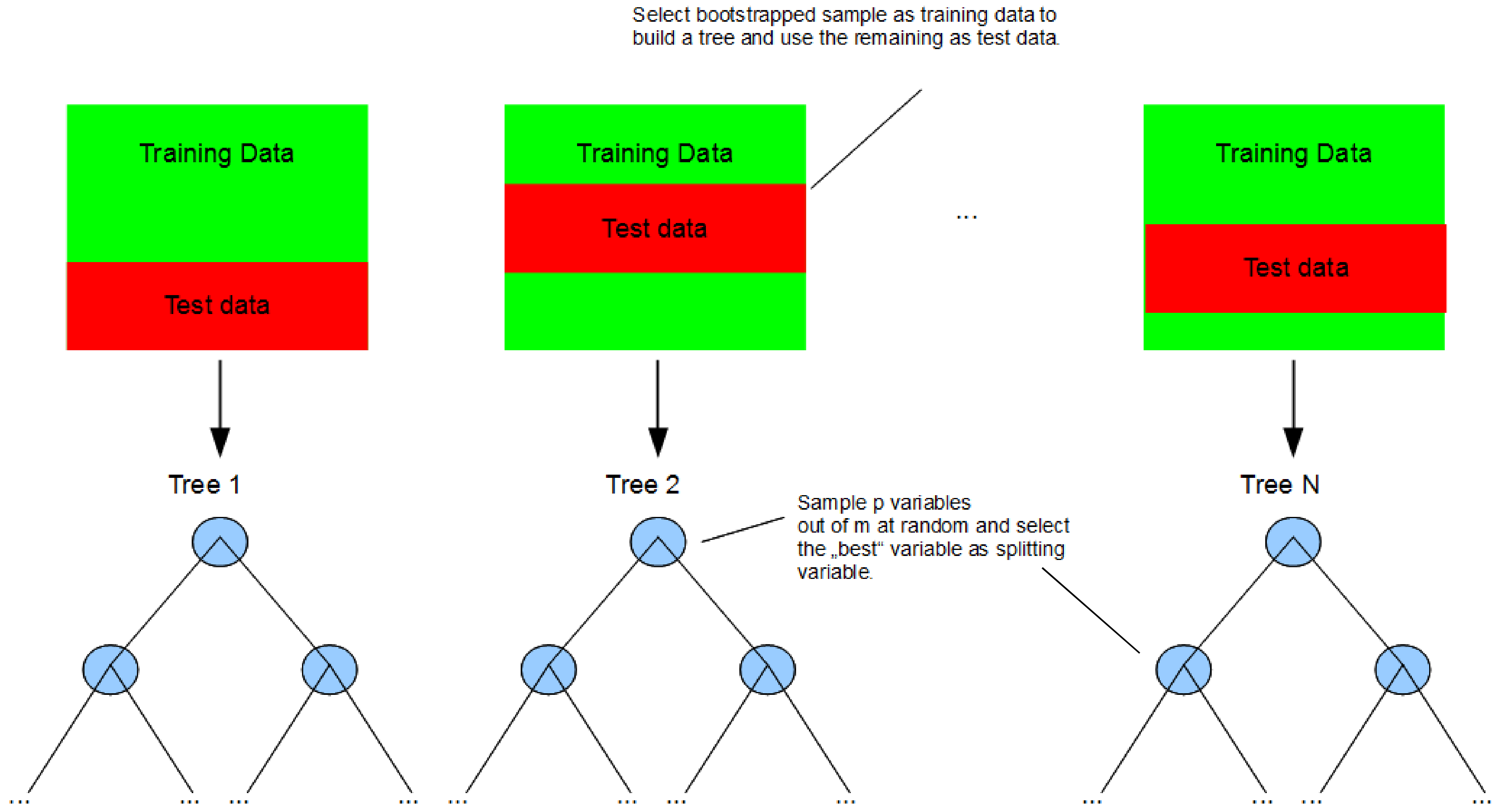
Bagging: bootstrapping and averaging

Basic idea:

1) Grow many iid trees on bootstrap samples of training data
2) Minimize Bias by growing trees sufficiently deep
3) Reduce variance of noisy but unbiased trees by averaging
4) Maximize variance reduction by minimizing correlation between trees by means of bootstrapping data for each tree and sampling available variable-set at each node

remark: highly non-linear estimators like trees benefit the most by bagging

Random Forest: Learning algorithm


Select bootstrapped sample as training data to build a tree and use the remaining as test data.
Training Data
Test data
Training Data
Test data
...
Training Data
Test data
Tree 1
Tree 2
Tree N
Sample p variables out of m at random and select the „best“ variable as splitting variable.
...

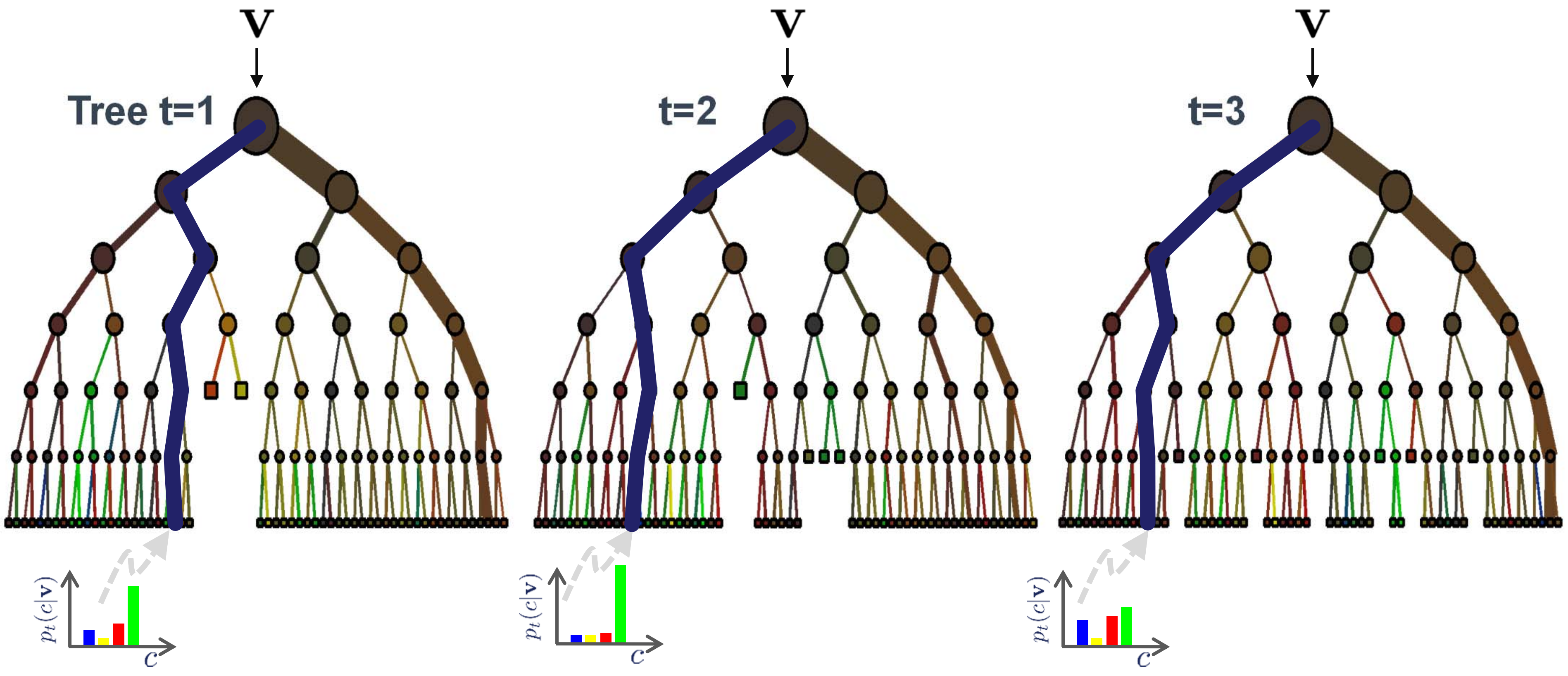
Random Forest: Learning algorithm

Each tree of the random forest is constructed using the following algorithm:

1. Let the number of training cases be N, and the number of variables in the classifier be M.
2. Choose a training set for this tree by choosing n times with replacement from all N available training cases (i.e. take a bootstrap sample). Use the rest of the cases to estimate the error of the tree, by predicting their classes.
3. For each node of the tree, randomly choose m variables on which to base the decision at that node. Calculate the best split based on these m variables in the training set.
4. Each tree is fully grown and not pruned.

For prediction a new sample is pushed down the tree. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the average vote of all trees is reported as random forest prediction.

How to classify a new observation v with a random forest?

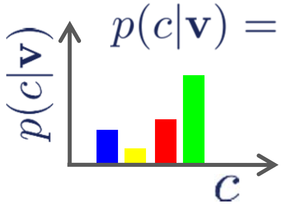


The to derive the ensemble result:

a) Each tree has a “winner-class”
Take the class which was most often the winner

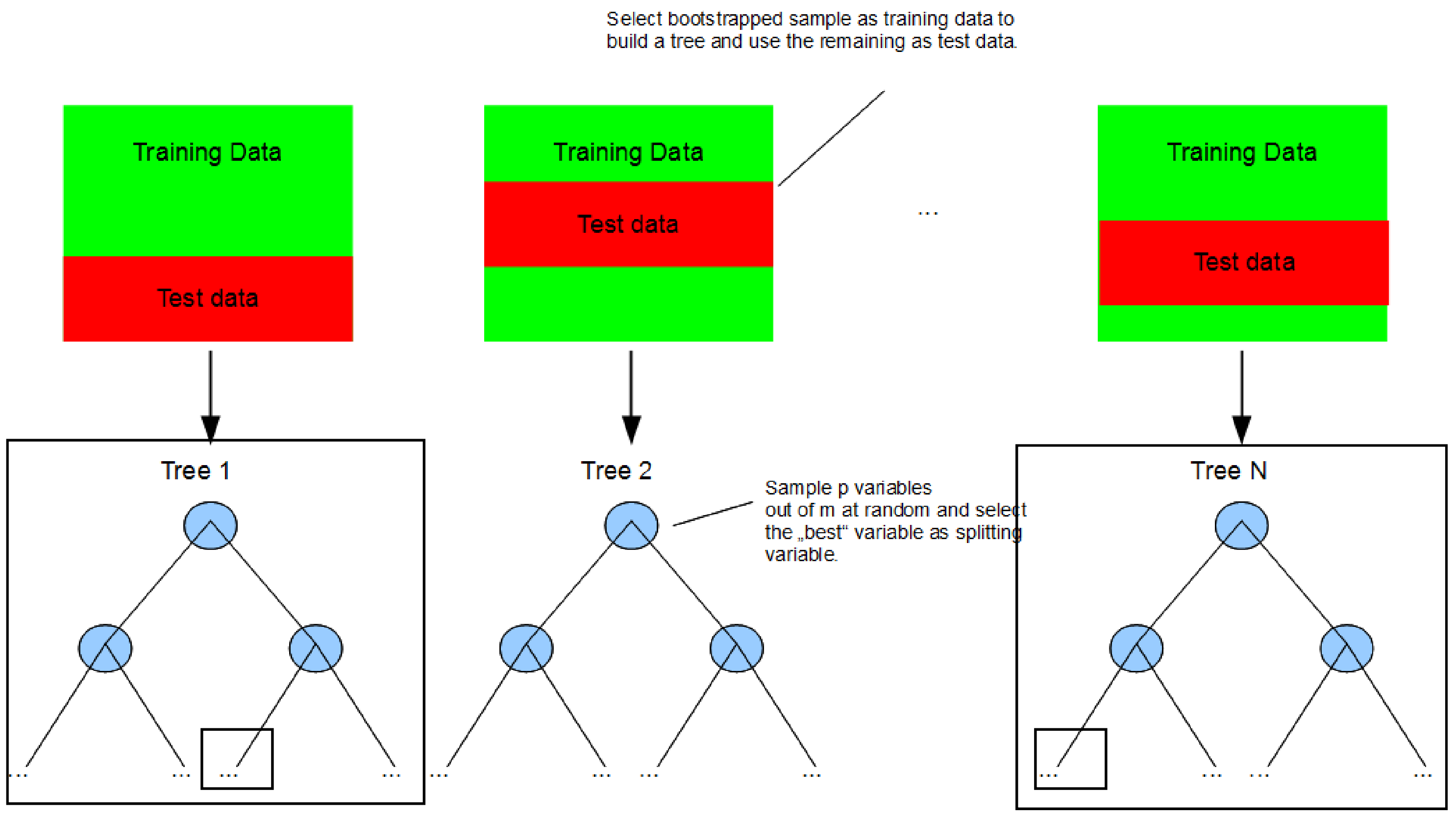
b) average probabilities:

$$p(c|\mathbf{v}) = \frac{1}{T}\sum_{t}^{T} p_t(c|\mathbf{v})$$



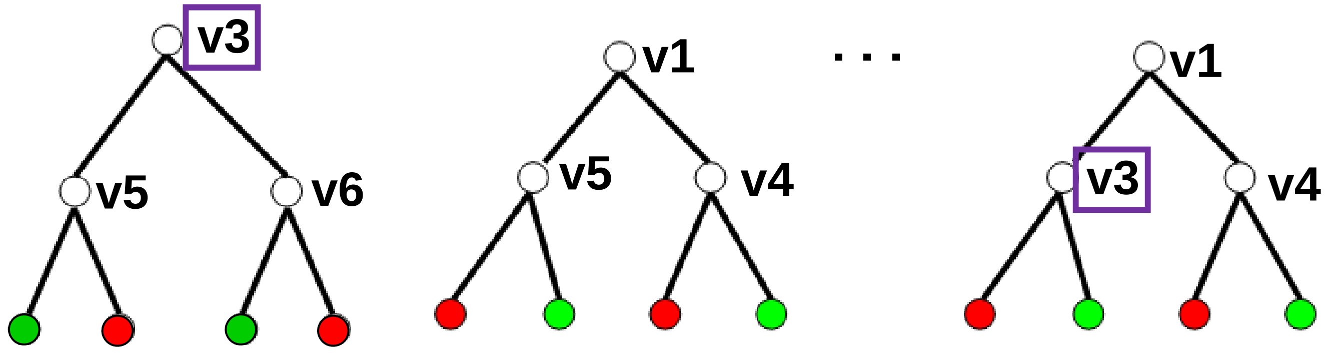
Animation taken from microsoft

Oob-error: Out of Bag Evaluation of the Random Forest


Select bootstrapped sample as training data to build a tree and use the remaining as test data.
Training Data
Test data
Training Data
Test data
...
Training Data
Test data
Tree 1
Tree 2
Sample p variables out of m at random and select the „best" variable as splitting variable.
Tree N

For each observation, construct its random forest oob-predictor by averaging only the results of those trees corresponding to bootstrap samples in which the observation was not contained.

Variable Importance 1: Score improvement at each Split



At each split where the variable, e.g. v3 is used, the improvement of the score is measured. The average over all these v3-involving splits in all trees is the importance-1 measure of v3.

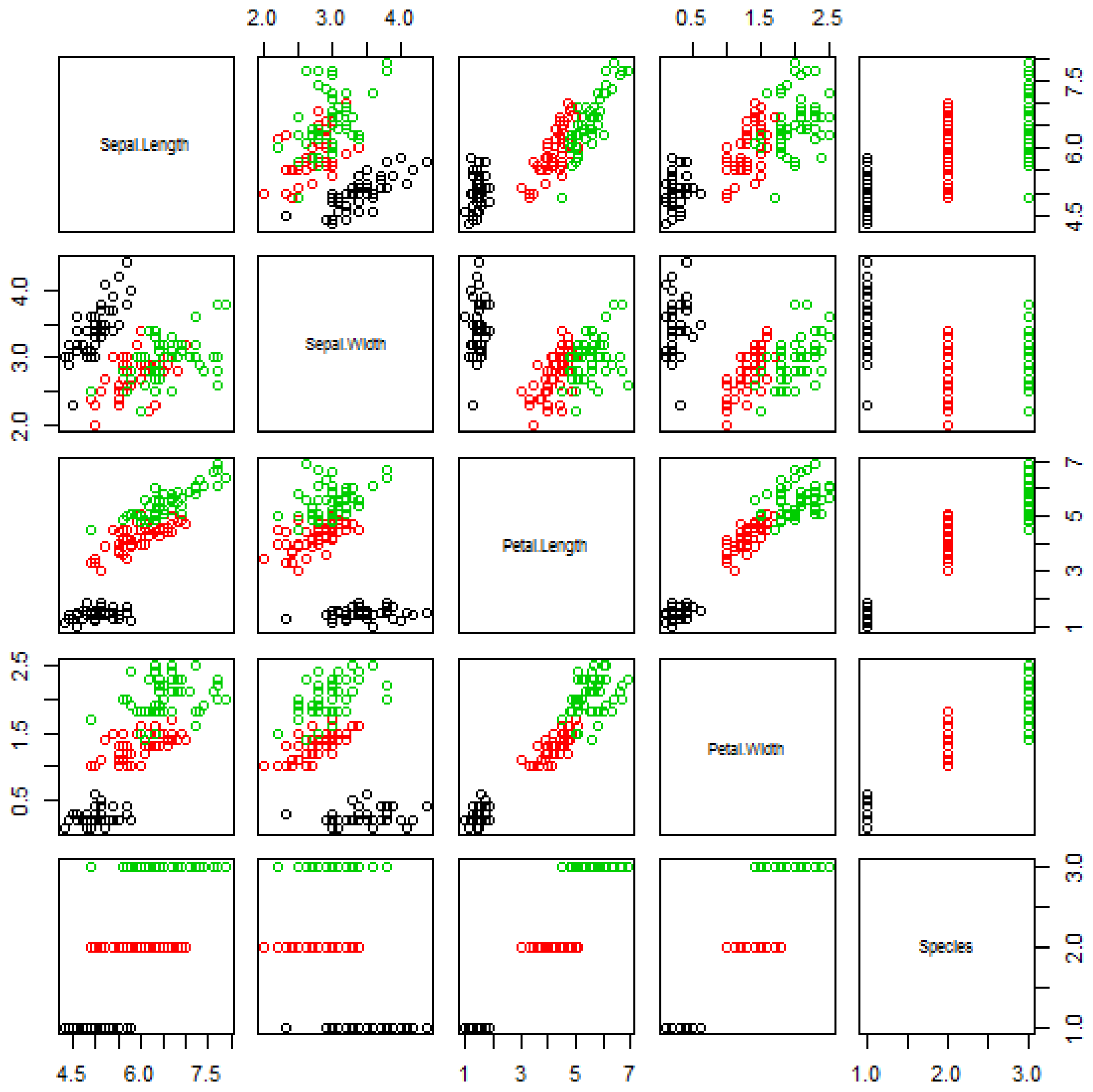
Variable Importance 2: performance-loss by permutation

Determine importance-2 for v4:

1) obtain standard oob-performance
2) values of v4 are randomly permuted in oob samples and oob-performance is again computed
3) Use decrease of performance as important measure of v4

v1	v2	v3	v4	v5	v6	v7
0	1	1	2	0	1	0
0	2	2	1	2	0	1
1	0	0	1	1	2	0
1	0	0	1	1	0	2
0	2	1	0	2	0	1

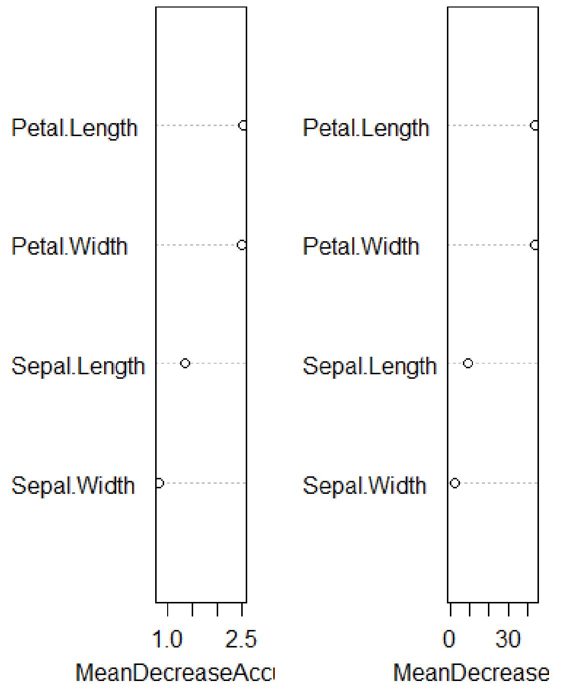
Analyzing the iris data set



Which leaf-length can be best predicted from the other measures?
Which measures are most important to identify the species?

Random Forest for classifying iris Species

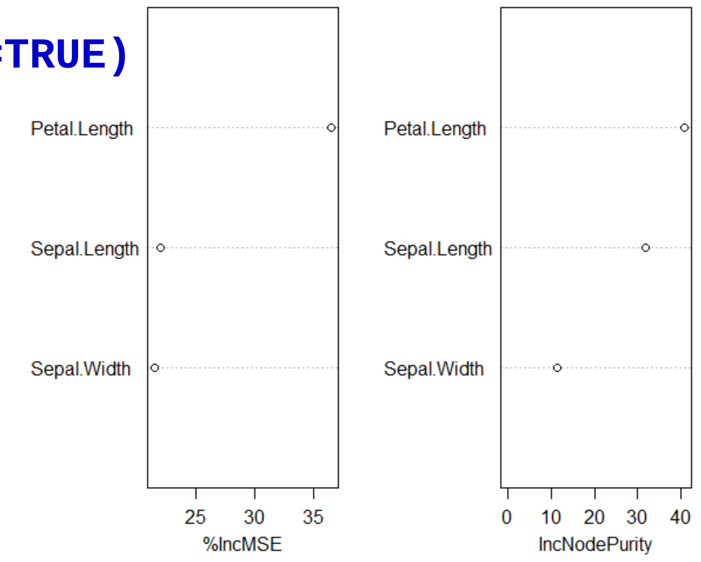
```
➢ library(randomForest)
> randomForest(Species ~ .,
           data = iris, importance = TRUE)
> print(iris.rf)
Call:
 randomForest(formula = Species ~ .,
           data = iris, importance = TRUE)
       Type of random forest: classification
             Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 6%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         50          0         0        0.00
versicolor      0         47         3        0.06
virginica       0          6        44        0.12
> varImpPlot(iris.rf)
```

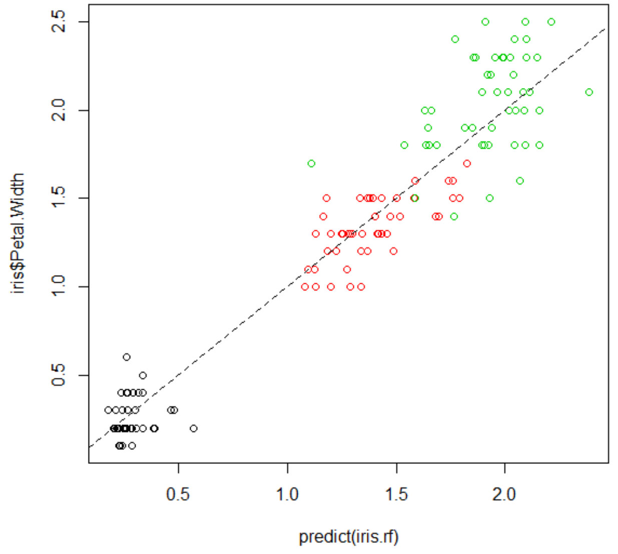


Measures on petal leafs are more important for classification than sepal measures

Random Forest for predicting Petal.Width via Regression

```
> iris.rf <- randomForest(Petal.Width ~ .,
                 data=iris[,1:4], importance=TRUE)
> print(iris.rf)
Call:
 randomForest(formula = Petal.Width ~ .,
data = iris[, 1:4], importance = TRUE)
     Type of random forest: regression
          Number of trees: 500
No. of variables tried at each split: 1
   Mean of squared residuals: 0.03995001
        % Var explained: 93.08
```





RF-regression allows quite well to predict the width of petal-leafs from the other leaf-measures of the same flower.

The model will probably improve, if we include the species as predictor variable.

Known Pros of Random Forest

- It is one of the most accurate learning algorithms available
- It is very easy to handle – no distribution assumptions -> no transformation
- Random Forest does not tend to overfit, cv incorporated
  (but the observations must be independent!)
- It can handle thousands of input variables without variable deletion
  - can handle lots of noise variable even with quite few relevant variables (6 out of 100 already works fine)
- with increasing number of variables the bias usually decreases
- can be used for variable selection -> variable importance
- can handle strong and local interactions very well
- robust against outliers in the predictive variables
- It can compute proximities between observations -> clustering data

Most important R functions for Random Forest

Package: “randomForest”

Functions “**`randomForest()`**” and “**`varImpPlot()`**”

What kind of missing data can we have

Missing (completely) at Random: can be handled

MCAR $P(R|Y_{com}) = P(R)$
Missingness does not depend on data

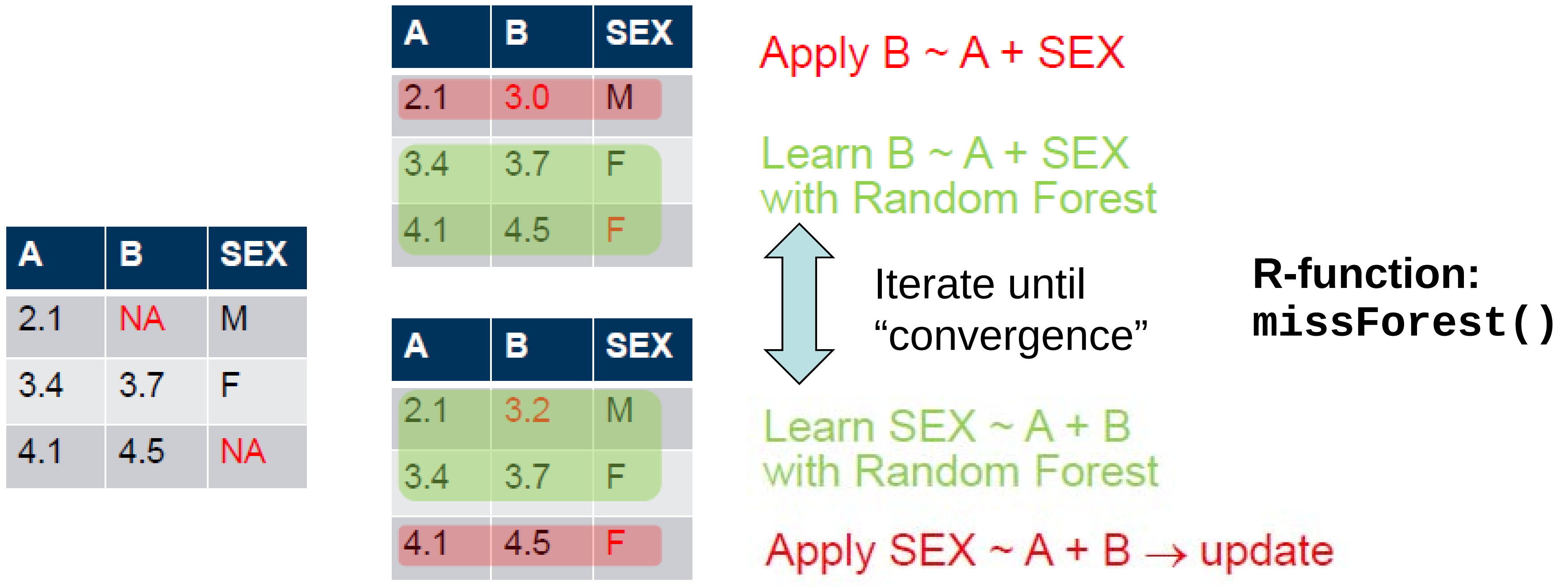
MAR $P(R|Y_{com}) = P(R|Y_{obs})$
Missingness depends only on observed data

MNAR $P(R|Y_{com}) = P(R|Y_{obs})$
Missingness depends on missing data

A	B	SEX
2.1	NA	M
3.4	3.7	F
4.1	4.5	NA

Missing Not at Random: serious problem!

Use RF for imputation if value are missing at random



+ Easy to use, works with mixed data types, gives an oob-imputation-error

- Single imputation (underestimates Variance) , over-optimistic oob

Or only using rf-proximities (randomForest package) to do weighted averaging: **`rfImpute()`**

missRF pseudo code & definition of oob imputation error

Stopping criterion: difference of successive imputed values stops to get smaller

Algorithm 2 Impute missing values with random forest.

```
Require: $\mathbf{X}$ an $n \times p$ matrix, stopping criterion $\gamma$
 1: Make initial guess for missing values;
 2: $\mathbf{k} \leftarrow$ vector of sorted indices of columns in $\mathbf{X}$
    w.r.t. increasing amount of missing values;
 3: while not $\gamma$ do
 4:     $\mathbf{X}^{imp}_{old} \leftarrow$ store previously imputed matrix;
 5:     for $s$ in $\mathbf{k}$ do
 6:         Fit a random forest: $\mathbf{y}^{(s)}_{obs} \sim \mathbf{x}^{(s)}_{obs}$;
 7:         Predict $\mathbf{y}^{(s)}_{mis}$ using $\mathbf{x}^{(s)}_{mis}$;
 8:         $\mathbf{X}^{imp}_{new} \leftarrow$ update imputed matrix, using predicted $\mathbf{y}^{(s)}_{mis}$;
 9:     end for
10:     update $\gamma$.
11: end while
12: return the imputed matrix $\mathbf{X}^{imp}$
```

Normalized Root Mean Squared Error (NRMSE):

$$NRMSE = \sqrt{\frac{mean(Y_{com} - Y_{imputed})^2}{var(Y_{com})}}$$

Proportion of falsely classified entries (PFC) over all categorical values

$$PFC = \frac{nmb.\ missclassified}{nmb.\ categorical\ values}$$

PhD-thesis @ ETH, 2012 by DANIEL J. STEKHOVEN

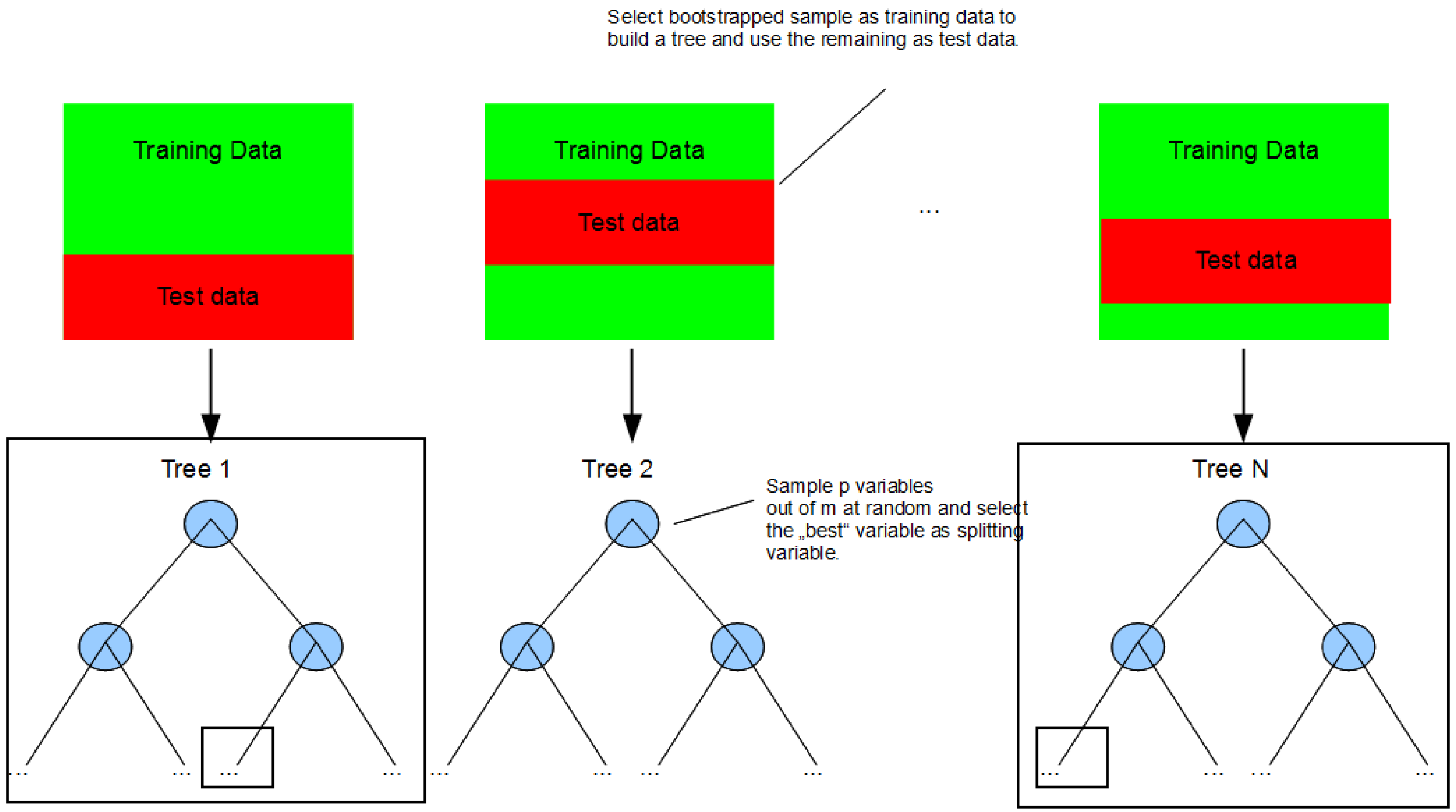
How to run an unsupervised RF?

Key Idea (Breiman)

- Label observed data as class 1
- Generate synthetic observations and label them as class 2

  independent sampling from each of the univariate marginal distributions of the variables
- Construct a RF predictor to distinguish class 1 from class 2
- Use the resulting RF-dissimilarity measure only for pairs of real observations out of class 1 – now the distance between two observations are only determined by their features and not by their class labels.

Proximity: Similarity between two observation according to supervised RF


Select bootstrapped sample as training data to build a tree and use the remaining as test data.
Training Data
Test data
Training Data
Test data
...
Training Data
Test data
Tree 1
Tree 2
Sample p variables out of m at random and select the „best" variable as splitting variable.
Tree N
...

The test data contains now a pair of observation random forest determines proximity by counting in how many trees both observation end up in the same leaf. Since the RF was built in a supervised modus, the proximity is also influenced by the class label of the observations.

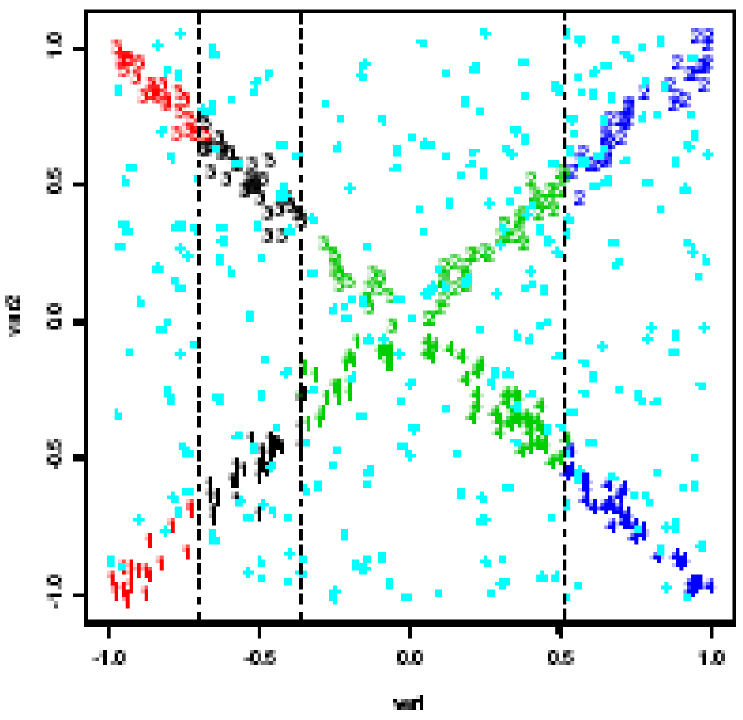
Proximity: Similarity between two observation according to unsupervised RF

In the unsupervised modus we do not provide a class label to the observations. In this modus RF creates new artificial observations by sampling from the marignal distribution of each feature and assigning the class label 0 to these new artificial observations. The original observations get class label 1.

The proximity between two observations from the original data set are now determined in the same manner without using different class labels for different observation of the provided datat – hence, only the similarity of their feature vector should have impact on the proximity score.

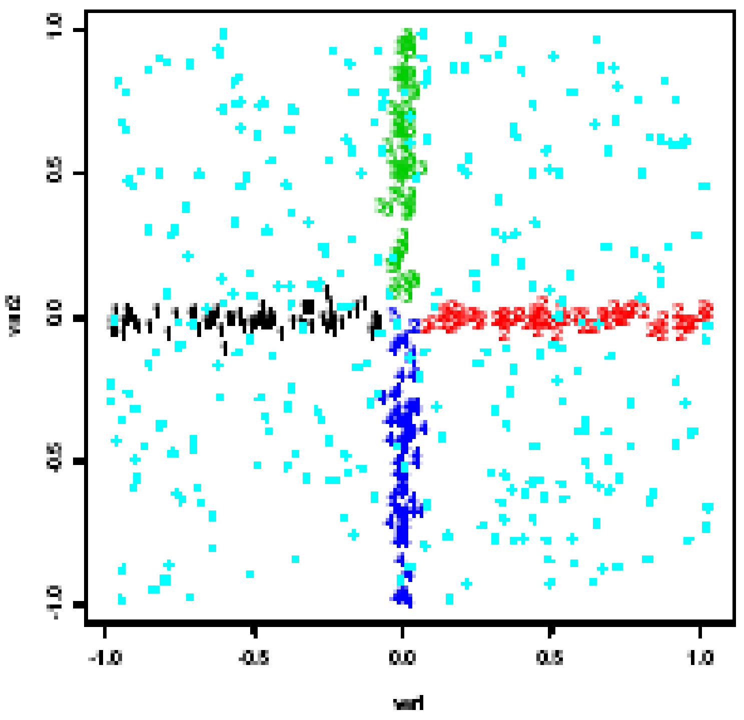
How do the synthetic data look like and what are the consequences? RF clustering is not rotationally invariant

- Synthetic Data (Synthetic Data)



a)

Cuts along the axes do not separate observed from synthetic (turquoise) data.



b)

Cuts along the axes succeed at separating observed data from Synthetic data.

Possible reasons for using distances from usRF

- **Properties of usRF distances:**
  - robust against outliers
  - resulting clusters can be described by a set of threshold rules
  - chooses cut-off values automatically
  - invariant against monotone transformations -> based on ranks
  - no need to transform the often highly skewed features
  - can handle variables with «creative coding» i.e. “-1” for not existent
  - can handle mixed variable (numeric and factor) types well
  - Ignores noise feature (up to >80% of all features can only be noise)
  - only part of the feature set may contribute the distances
  - distances will slightly vary from usRF to usRF due to sampling steps
  - distance focuses on most dependent and interacting variables
  - Is not invariant against rotations (see example)