Machine Learning

Prediction of Phenotype by Transcriptome classification using Random Forest Machine Learning

Uwe Menzel, 2012

uwe.menzel@matstat.de

www.matstat.org

Identification of longevity biomarkers in *Nothobranchius furzeri* by Transcriptome Measurement and Random Forest Analysis

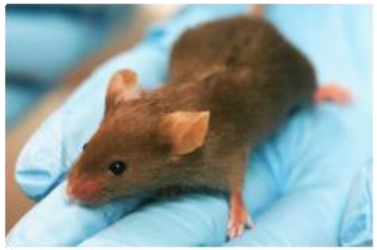
U. Menzel[1], S. Priebe[1], M. Baumgart[2], M. Groth[2], A. Cellerino[2], R. Guthke[1]
[1]Leibniz Institute for Natural Product Research and Infection Biology - Hans-Knöll-Institute
[2]Leibniz Institute for Age Research - Fritz-Lipmann Institute

- Transcriptomes (RNA-Seq, Illumina), different ages, 5 species
- > 1000 transcriptomes

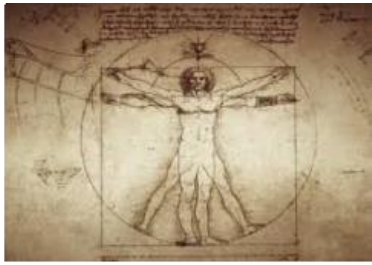
C. elegans

N. furzeri (killifish)

D. rerio (zebrafish)



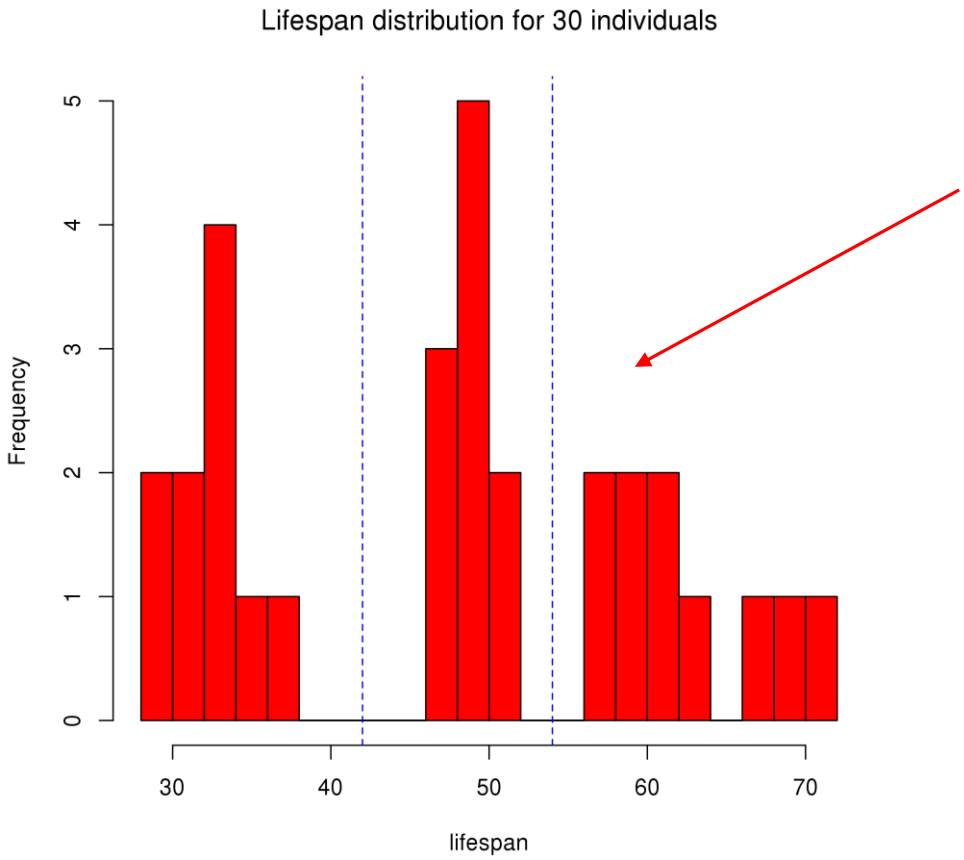
M. musculus



H. sapiens

“Finclip study”

- Fin biopsis from 152 individuals of *N. furzeri* aged 10 or 20 weeks (max. lifespan 60 weeks) without sacrificing the fish → record lifespan data
- → for each fish: transcriptome at 10 & 20 weeks + lifespan



Fishes were subdivided into 3 lifespan groups:

- short-lived,
- medium-lived,
- long-lived

Note: grouping into (G1, G2,G3) is not mandatory. RF can also be used for regression.

Transcriptome data

	Nfu_g_1_015316	Nfu_g_1_022387	Nfu_g_1_007729	Nfu_g_1_011508	Nfu_g_1_017364	Nfu_g_1_011804	lifespan
LS10_10	1.783383	3.783560	0.7551712	2.539021	10.02691	0.2986646	G1
LS10_20	4.610213	3.680158	0.5084912	3.256471	17.02645	0.3995374	G1
LS1_10	3.487960	2.028367	0.9885908	1.576127	15.94995	0.6600103	G1
LS11_10	7.491374	2.719286	0.6367047	1.566329	14.29085	1.4492038	G2
LS11_20	5.550955	1.585267	0.4532070	1.492758	11.00229	0.6734699	G2
LS1_20	3.962182	4.192136	0.3688653	2.719002	12.47558	0.8255347	G1
LS12_10	6.148398	7.738688	0.7528698	3.050520	14.01485	1.9913062	G2
LS12_20	5.691901	4.942013	0.4244077	2.673257	12.76144	0.4833154	G2
LS13_10	6.176109	7.046949	0.9157133	3.299551	18.65433	1.8501543	G2
LS13_20	4.258711	5.333459	0.4729034	1.737829	14.75942	1.2208889	G2

samples, obervations (fishes)

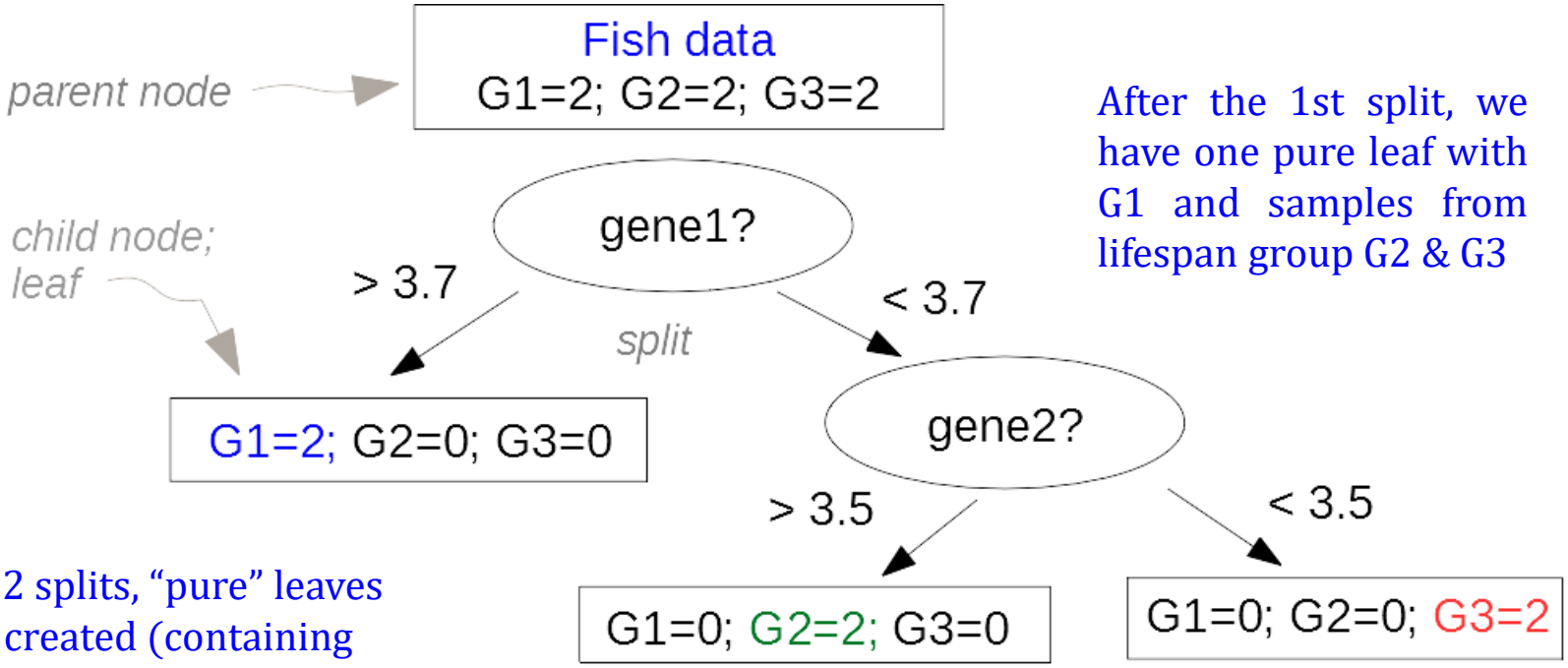
predictors, features, variables (genes)

Predict this symbol (G1, G2, G3) from other columns

TODO: Identify those genes whose expression is predictive for lifespan
→ establish statistical model

Decision tree: Training set

Fish#	gene1	gene2	gene3	class
1	6.1	7.1	1.9	G1 (long)
2	5.4	3.9	4.2	G1 (long)
3	1.1	5.0	2.4	G2 (medium)
4	2.0	4.0	7.4	G2 (medium)
5	1.5	2.1	1.2	G3 (short)
6	1.2	3.0	5.3	G3 (short)



After the 1st split, we have one pure leaf with G1 and samples from lifespan group G2 & G3

With 2 splits, "pure" leaves were created (containing one class only)

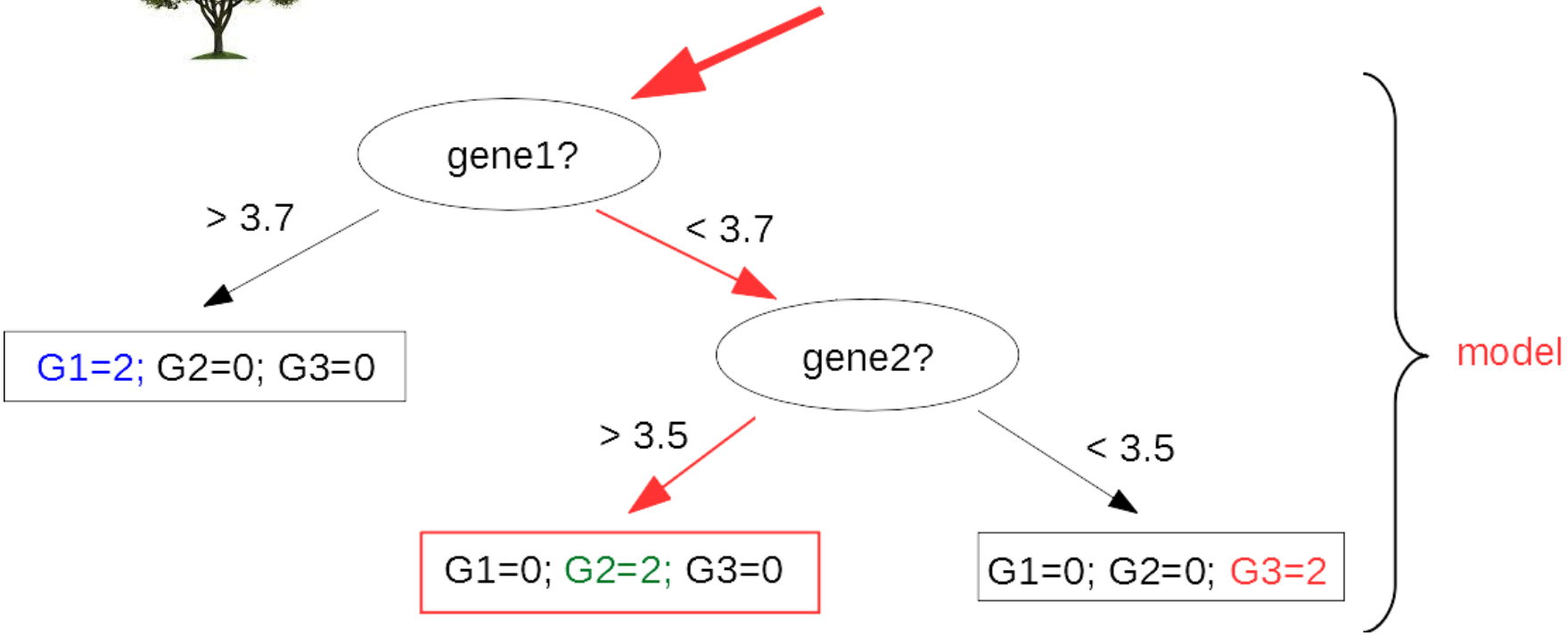
These are now our decision rules – **the model**

Test set, prediction

A new sample with still unknown class (lifetime)

Fish#	gene1	gene2	gene3	class
7	3.2	4.7	5.3	??

We apply the **decision rules** "learned" above



- Using the model (the decision tree), we can predict the lifespan class of some indvidual based on it's transcriptome (at early age).
- In order to make unique predictions, we must achieve pure leafs (with a few splits).

Creating pure leaves: Measuring purity

$$H = -\sum_{i=1}^{N} p_i \cdot \ln p_i$$

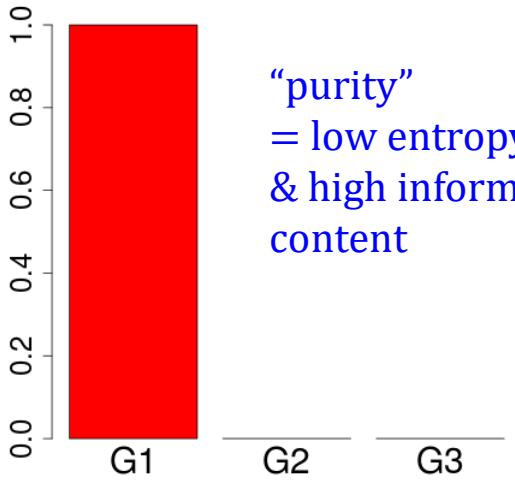
H: entropy ; p_i: probability (fraction)

	Leaf #1		
Class	G1	G2	G3
#samples	3	0	0
p_i	3/3	0/3	0/3

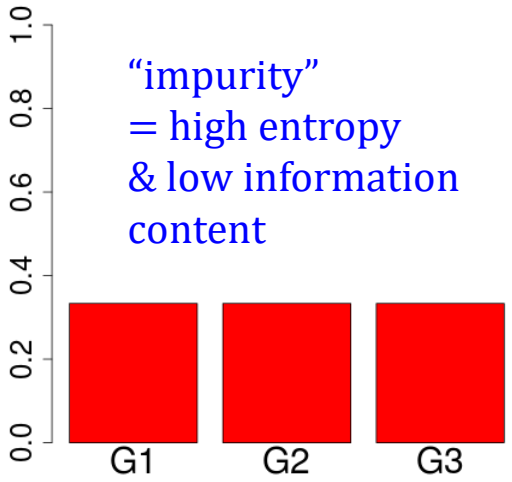
	Leaf #2		
Class	G1	G2	G3
#samples	1	1	1
p_i	1/3	1/3	1/3

$$H_1 = -\frac{3}{3} \cdot \ln\left(\frac{3}{3}\right) - \frac{0}{3} \cdot \ln\left(\frac{0}{3}\right) - \frac{0}{3} \cdot \ln\left(\frac{0}{3}\right) = 0$$

$$H_2 = -\frac{1}{3} \cdot \ln\left(\frac{1}{3}\right) - \frac{1}{3} \cdot \ln\left(\frac{1}{3}\right) - \frac{1}{3} \cdot \ln\left(\frac{1}{3}\right) \approx 1.1$$

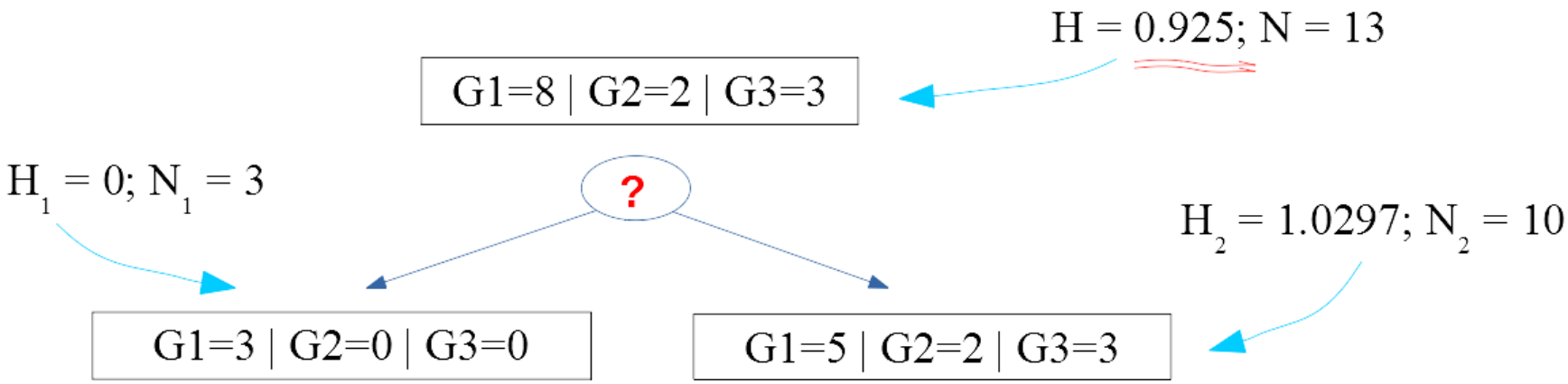


"purity"
= low entropy
& high information content



"impurity"
= high entropy
& low information content

Creating pure leaves: Entropy before and after split



$$I = \sum_{i=1}^{childs} \frac{N_i}{N} \cdot H_i$$

$$= \frac{N_1}{N_1 + N_2} \cdot H_1 + \frac{N_2}{N_1 + N_2} \cdot H_2$$

$$= \frac{3}{13} \cdot 0 + \frac{10}{13} \cdot 1.0297 = 0.792$$

Weighted average of child node entropies:

- N_i = nr. samples in a leaf
- H_i = entropy in a leaf
- $N = \sum N_i$

Information gain = 0.925 − 0.792 = 0.133 ; Information gain = entropy reduction from parent to child nodes.

Creating pure leaves: Finding the (best) splits

	Nfu_g_1_015316	Nfu_g_1_022387	Nfu_g_1_007729	Nfu_g_1_011508	Nfu_g_1_017364	Nfu_g_1_011804	lifespan
LS10_10	1.783383	3.783560	0.7551712	2.539021	10.026911	0.2986646	G1
LS3_20	2.912910	3.311607	0.2488983	1.865725	17.114136	0.7669024	G1
LS18_20	3.130268	1.026144	0.2906958	2.092429	14.161903	0.3040331	G2
LS3_10	3.179805	3.360687	0.4278054	2.074309	17.036349	0.8413103	G1
LS1_10	3.487960	2.028367	0.9885908	1.576127	15.949953	0.6600103	G1
LS9_10	3.566403	4.069780	0.5727213	1.710495	13.471091	0.8510692	G1
LS23_10	3.658328	1.870489	0.4121220	2.977915	9.516363	0.5995242	G3
LS16_10	3.710652	20.824557	0.8620429	2.575994	10.774487	0.7982864	G2
LS30_20	3.845079	8.204667	0.5187278	2.234556	12.295116	0.6729676	G3
LS1_20	3.962182	4.192136	0.3688653	2.719002	12.475581	0.8255347	G1

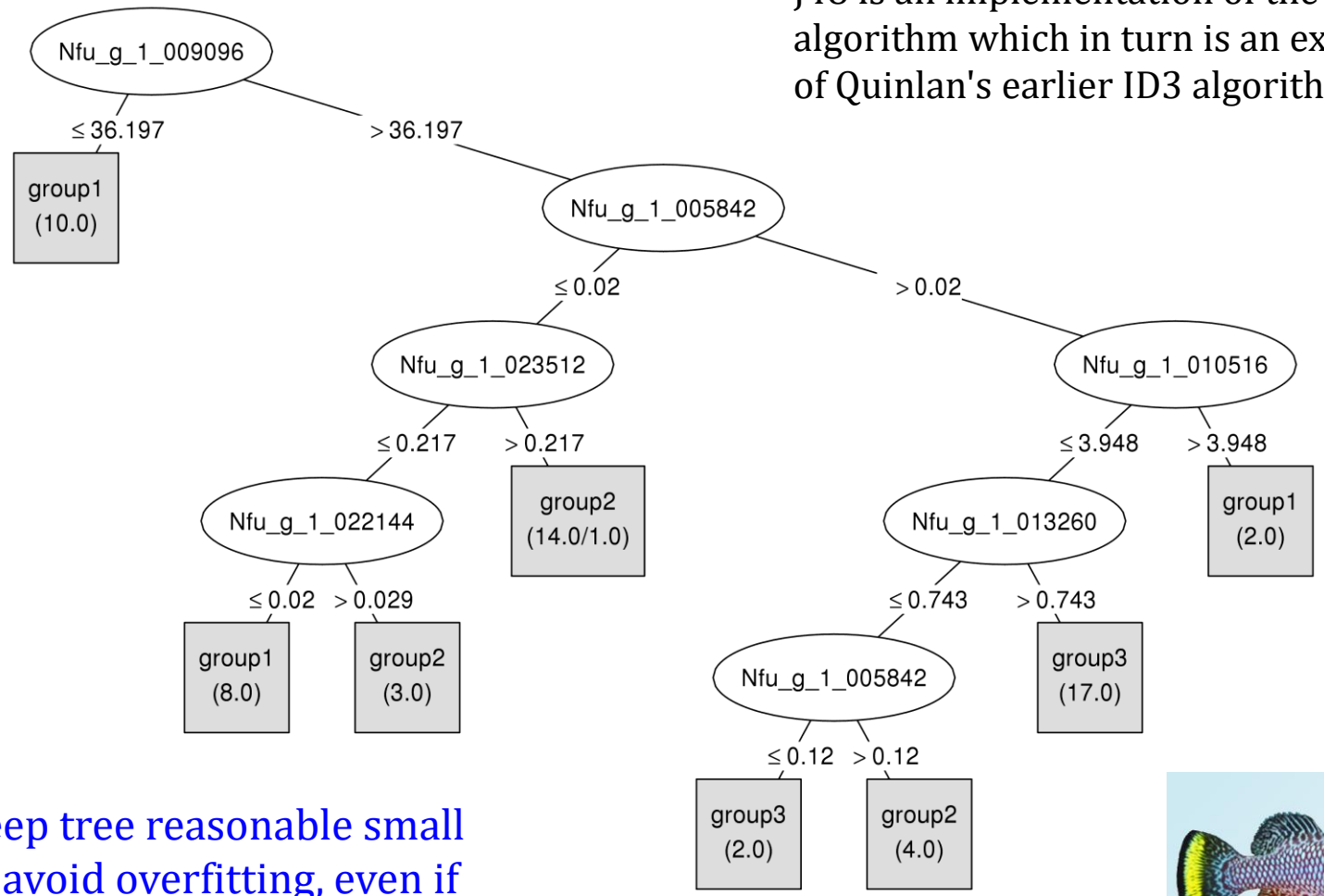
order when this variable is probed

- Try out:
  - all variables (Nfu_ .....)
  - all splits (midpoints of ordered expression values)
- choose the split with highest information gain
- ID3, C4.5 algorithms (Ross Quinlan*)
  - Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81-106
  - https://en.wikipedia.org/wiki/ID3_algorithm

Finclip-study

```
library(RWeka)
tree <- J48(lifespan ~ ., data = df)
summary(tree)
plot(tree)
```

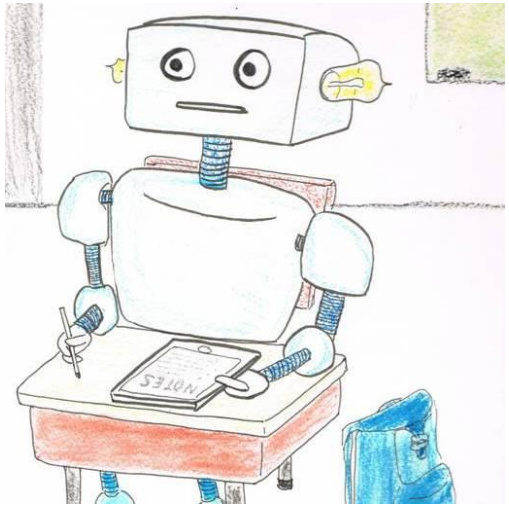
J48 is an implementation of the C4.5 algorithm which in turn is an extension of Quinlan's earlier ID3 algorithm.



Keep tree reasonable small to avoid overfitting, even if impure leaves remain!

Classification by Machine Learning

- Once the tree is created, we can classify new samples by "running them down the tree" (Breiman, Cutler*) → classification
- https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm
- We habe established a “model” and are now able to automatically classify new samples → machine learning



http://gureckislab.org/blog

Ups & Downs of Decision Trees

Advantages:

- classifier that is favourable when the number of variables (genes) is higher than the number of observations (samples) (90 samples, 15.000 genes)
- tree pinpoints genes that are informative with regard to some attribute (here with regard to age).

Drawback:

- Overfitting, bias-variance dilemma
- Having huge data with big variance, correlation between predictors and response variables can occur which is purely casual!

Solution: Use Random Forest!

https://en.wikipedia.org/wiki/Pruning_(decision_trees)

Random Forest (RF)

- Breiman, Cutler, 2001*
- ensemble classifier, creates many decision trees (typically 1000)
- prediction by aggregating multiple deep (unpruned) trees
  - prevention of overfitting to training set
- use subset of data for every tree:
  1. sample with replacement from the observations → reduces variance
  2. select random subset of variables → identify additional predictors

* Breiman, Leo (2001). "Random Forests". Machine Learning 45 (1): 5–32

Sampling in RF

	Nfu_g_1_015316	Nfu_g_1_022387	Nfu_g_1_007729	Nfu_g_1_011508	Nfu_g_1_017364	Nfu_g_1_011804	lifespan
LS10_10	1.783383	3.783560	0.7551712	2.539021	10.02691	0.2986646	G1
LS10_20	4.610213	3.680158	0.5084912	3.256471	17.02645	0.3995374	G1
LS1_10	3.487960	2.028367	0.9885908	1.576127	15.94995	0.6600103	G1
LS11_10	7.491374	2.719286	0.6367847	1.566329	14.29085	1.4492838	G2
LS11_20	5.550955	1.585267	0.4532070	1.492758	11.00229	0.6734699	G2
LS1_20	3.962182	4.192136	0.3688653	2.719002	12.47558	0.8255347	G1
LS12_10	6.148398	7.738688	0.7528698	3.050520	14.01485	1.9913062	G2
LS12_20	5.691901	4.942013	0.4244077	2.673257	12.76144	0.4833154	G2
LS13_10	6.176109	7.046949	0.9157133	3.299551	18.65433	1.8501543	G2
LS13_20	4.258711	5.333459	0.4729034	1.737829	14.75942	1.2208889	G2
LS14_10	7.619397	15.069436	0.6466326	2.421878	12.22194	1.1121912	G2
LS14_20	5.671981	6.596495	0.4721143	1.693458	16.63263	0.7399827	G2
LS15_10	4.985908	2.185439	0.2989142	1.673799	18.81677	0.8052556	G2
LS15_20	6.147395	1.433401	0.2440111	2.743580	19.95349	0.4995860	G2
LS16_10	3.710652	20.824557	0.8620429	2.575994	10.77449	0.7982864	G2
LS16_20	4.483383	12.629638	0.5761642	1.922418	17.73283	0.4466388	G2
LS17_10	5.319336	3.401925	0.4684633	3.693792	13.93959	0.9389352	G2
LS17_20	4.515645	1.197883	0.4089893	2.412451	14.00179	0.3559631	G2
LS18_10	4.190156	1.514862	0.2463951	1.185550	19.16160	1.1947903	G2
LS18_20	3.130268	1.026144	0.2906958	2.092429	14.16190	0.3040331	G2

Ignore about 1/3 of the observations (fishes) by sampling with replacement

Randomly choose $\sqrt{N}$ of the N variables (genes)

⇨ every tree is build with a subset of the data

```
## Default S3 method:
randomForest(x, y=NULL,  xtest=NULL, ytest=NULL, ntree=500,
             mtry=if (!is.null(y) && !is.factor(y))
             max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
             replace=TRUE, classwt=NULL, cutoff, strata,
             sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)),
             nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
             maxnodes = NULL,
             importance=FALSE, localImp=FALSE, nPerm=1,
             proximity, oob.prox=proximity,
             norm.votes=TRUE, do.trace=FALSE,
             keep.forest=!is.null(y) && is.null(xtest), corr.bias=FALSE,
             keep.inbag=FALSE, ...)
```

Draw $n_{\texttt{tree}}$ bootstrap samples from the original data.

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 $m_{\texttt{try}}$ of the predictors and choose the best split from among those

```
## S3 method for class 'formula'
randomForest(formula, data=NULL, ..., subset, na.action=na.fail)
## Default S3 method:
randomForest(x, y=NULL,  xtest=NULL, ytest=NULL, ntree=500,
             mtry=if (!is.null(y) && !is.factor(y))
             max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
             replace=TRUE, classwt=NULL, cutoff, strata,
             sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)),
             nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
             maxnodes = NULL,
             importance=FALSE, localImp=FALSE, nPerm=1,
             proximity, oob.prox=proximity,
```

Draw $n_{\texttt{tree}}$ bootstrap samples from the original data.

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 $m_{\texttt{try}}$ of the predictors and choose the best split from among those

Liaw, Wiener: Classification and Regression by randomForest

Predict new data by aggregating the predictions of the $n_{\texttt{tree}}$ trees (i.e., majority votes for classification, average for regression).

Error estimation for RF

An estimate of the error rate can be obtained, based on the training data, by the following:

1. At each bootstrap iteration, predict the data not in the bootstrap sample (what Breiman calls "out-of-bag", or OOB, data) using the tree grown with the bootstrap sample.

2. Aggregate the OOB predictions. (On the average, each data point would be out-of-bag around 36% of the times, so aggregate these predictions.) Calcuate the error rate, and call it the OOB estimate of error rate.

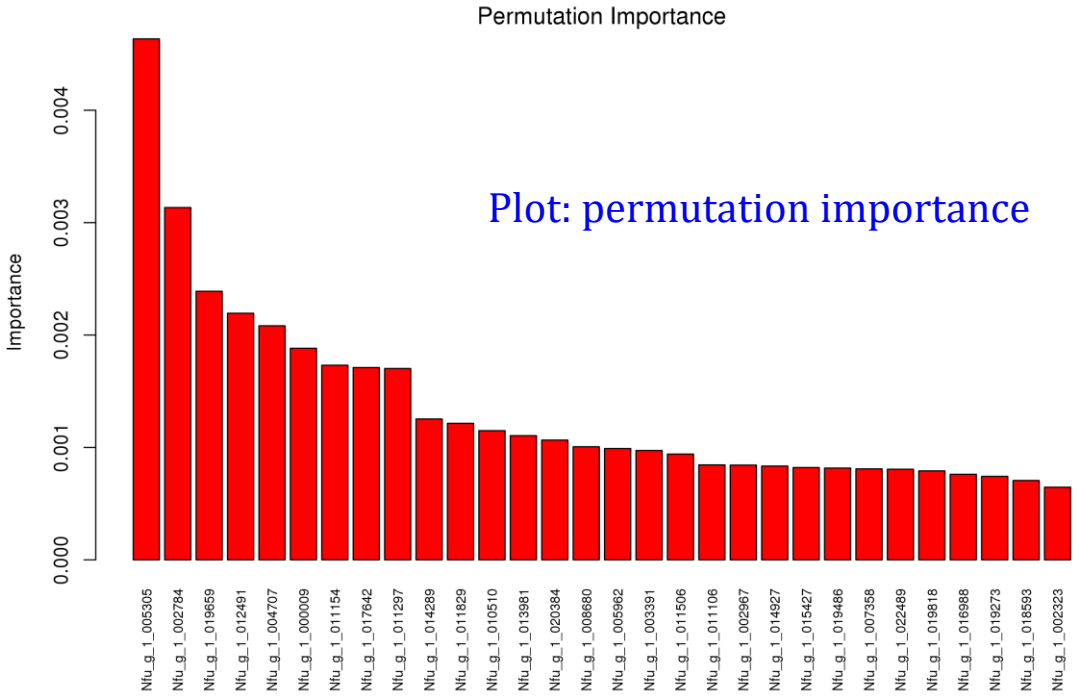
Our experience has been that the OOB estimate of error rate is quite accurate, given that enough trees have been grown (otherwise the OOB estimate can bias upward; see Bylander (2002)).

Liaw, Wiener: Classification and Regression by randomForest

Variable Importance in RF

- a measure of the explanatory power of a variable (gene) with regard to the response (lifespan)
- Permutation importance:
  - scramble i-th variable in training set and record change in the out-of-bag error

	Nfu_g_1_015316
LS10_10	1.783383
LS10_20	4.610213
LS1_10	3.487960
LS11_10	7.491374
LS11_20	5.550955
LS1_20	3.962182
LS12_10	6.148398
LS12_20	5.691901
LS13_10	6.176109
LS13_20	4.258711
LS14_10	7.619397
LS14_20	5.671981
LS15_10	4.985908



Plot: permutation importance

gene names

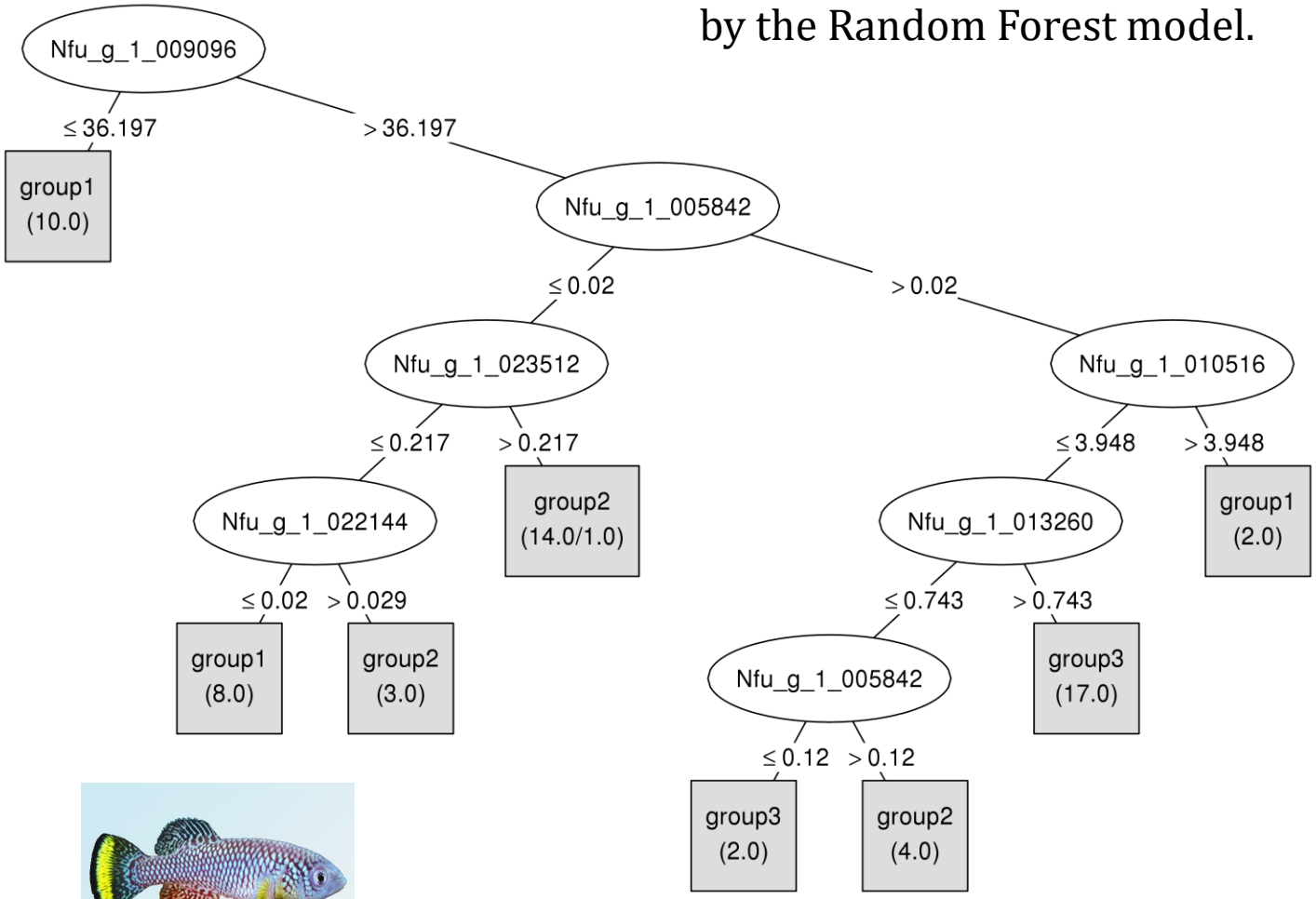
Feature selection (variable selection)

- selecting a subset of relevant features (variables, predictors) for use in the model (Wikipedia)
- here: select relevant genes (= variables) for prediction
- R package varSelRF (Diaz-Uriarte, 2006)
- builds RF models recursively, and abandons the variables (genes) with the lowest predictive power on every step
- predictive power is measured by out-of-bag (OOB) error

```
set.seed(99)
rf.model <- varSelRF(expr.data.df, class,
                     ntree = 5000,
                     vars.drop.frac = 0.2,
                     keep.forest = T)
best.variables = rf.model$selected.vars
length(best.variables)
```

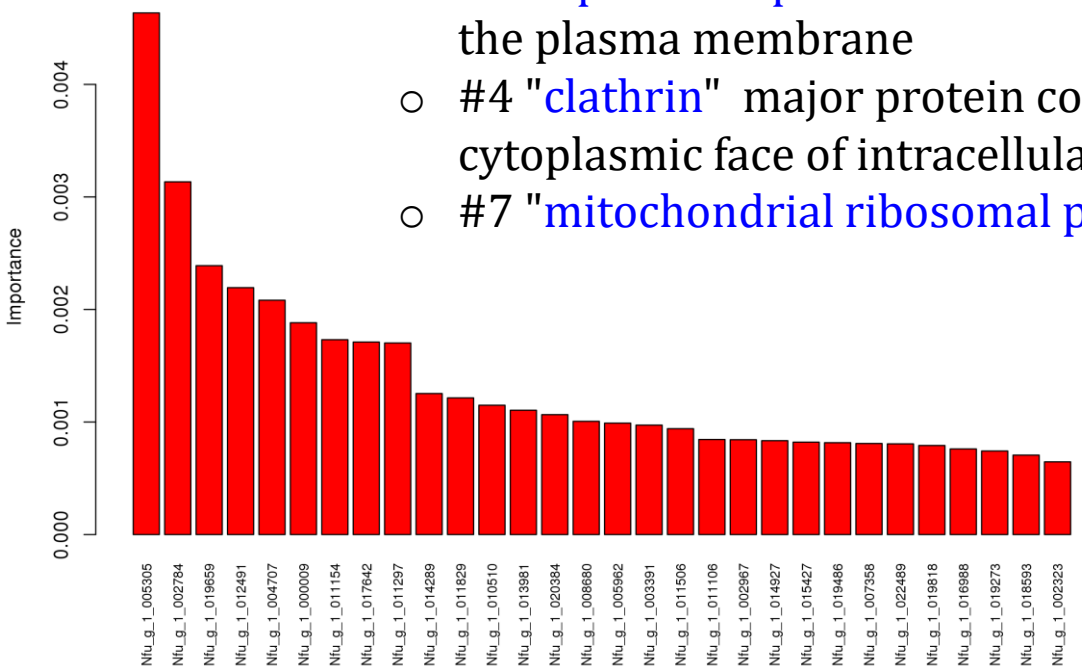
“Averaged tree”

The tree is based on the genes with the largest importance values calculated by the Random Forest model.



Important Genes = Biomarkers

- Biomarkers for ageing
- Enrichment analysis: GO, KEGG



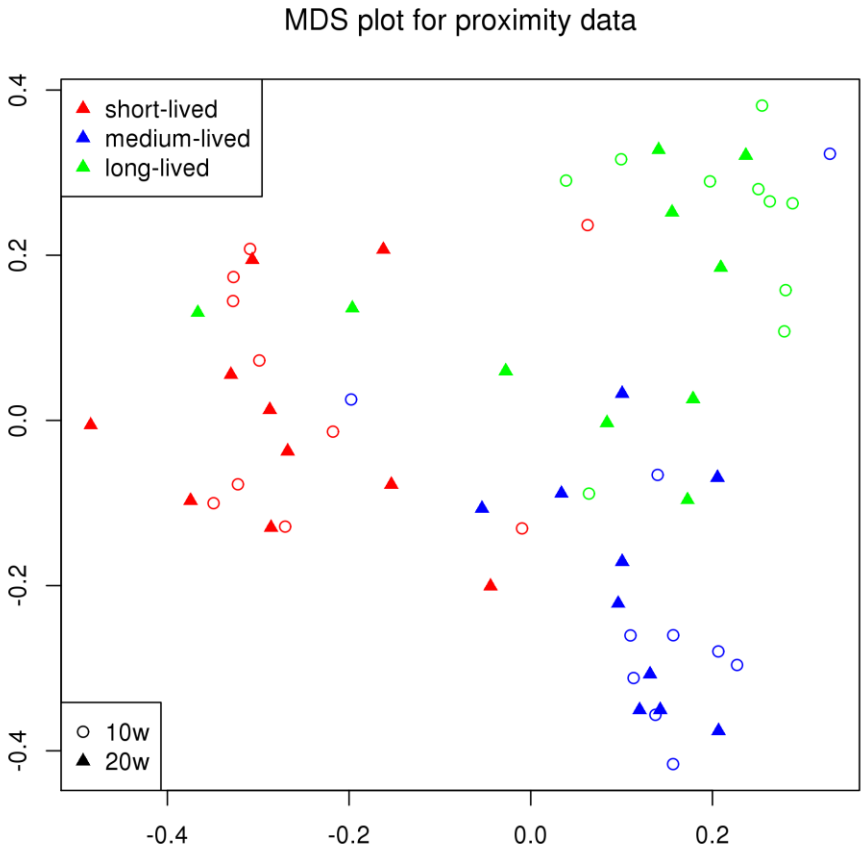
- #1: "spectrin alpha 2": scaffold proteins that stabilize the plasma membrane
- #4 "clathrin" major protein component of the cytoplasmic face of intracellular organelles
- #7 "mitochondrial ribosomal protein" ....

Plot: permutation importance vs. genes

Proximity of the samples

RF calculates a proximity matrix (how close is every pair of samples?)

- if samples end up in the same leaf frequently, they are considered similar
- proximity matrix → Multi-Dimensional Scaling (MDS-) plot
- MDS-plot presents samples in a 2- or 3-D plot by preserving the relative distances between samples
- a point represents the whole transcriptome of a sample
- samples cluster well according to lifespan group



Cross Validation

_g_1_007729	Nfu_g_1_011508	Nfu_g_1_017364	Nfu_g_1_011804	lifespan
0.7551712	2.539021	10.02691	0.2986646	G1
0.5084912	3.256471	17.02645	0.3995374	G1
0.9885908	1.576127	15.94995	0.6600103	G1
0.6367047	1.566329	14.29085	1.4492038	G2
0.4532070	1.492758	11.00229	0.6734699	G2
0.3688653	2.719002	12.47558	0.8255347	G1
0.7528698	3.050520	14.01485	1.9913062	G2
0.4244077	2.673257	12.76144	0.4833154	G2
0.9157133	3.299551	18.65433	1.8501543	G2
0.4729034	1.737829	14.75942	1.2208889	G2
0.6466326	2.421878	12.22194	1.1121912	G2
0.4721143	1.698458	16.63263	0.7399827	G2
0.2989142	1.678799	18.81677	0.8052556	G2
0.2440111	2.748580	19.95349	0.4995860	G2
0.8620429	2.575994	10.77449	0.7982864	G2
0.5761642	1.922418	17.73283	0.4466388	G2
0.4684633	3.693792	13.93959	0.9389352	G2
0.4089893	2.412451	14.00179	0.3559631	G2
0.2463951	1.185550	19.16160	1.1947903	G2
0.2906958	2.092429	14.16190	0.3040331	G2

Training set:
build RF model

Test set: “forget” the response (for a while)

10-fold CV: subdivide samples randomly into 10 parts

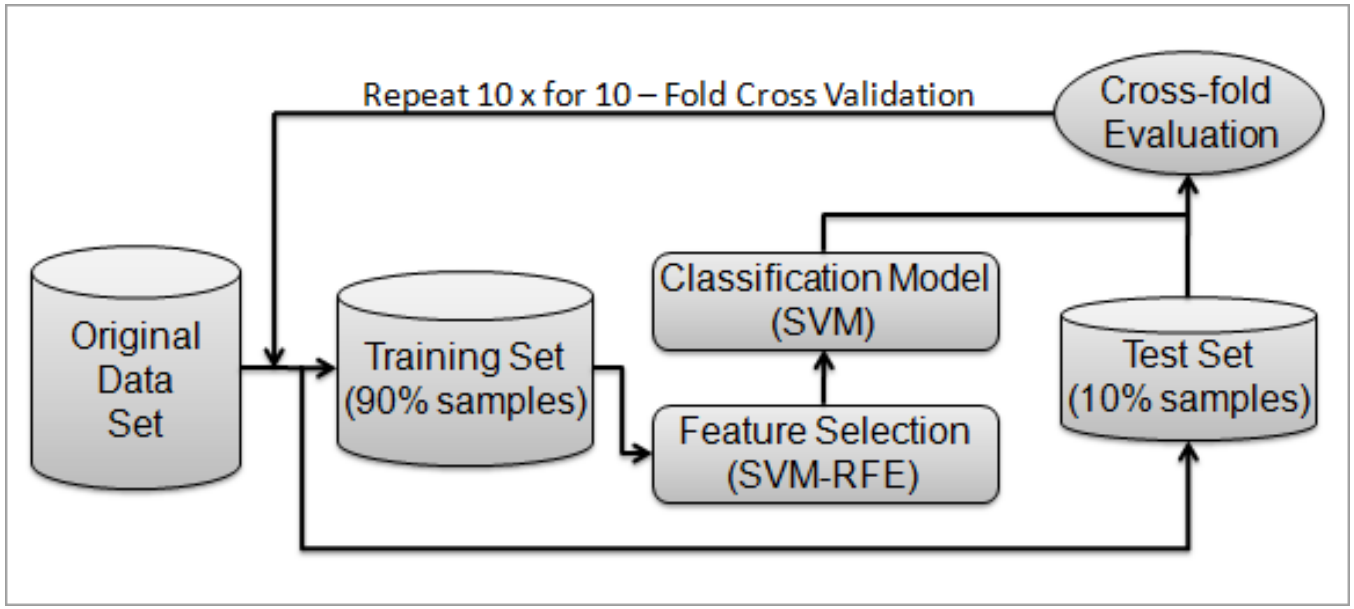
- Use 90% as training set, 10% as test set
- Repeat classification 10 times
- Calculate average classification errror

ca. 70% correctly classified in the finclip study

Pitfall in Cross Validation

Common mistake: knowledge leaking[1]

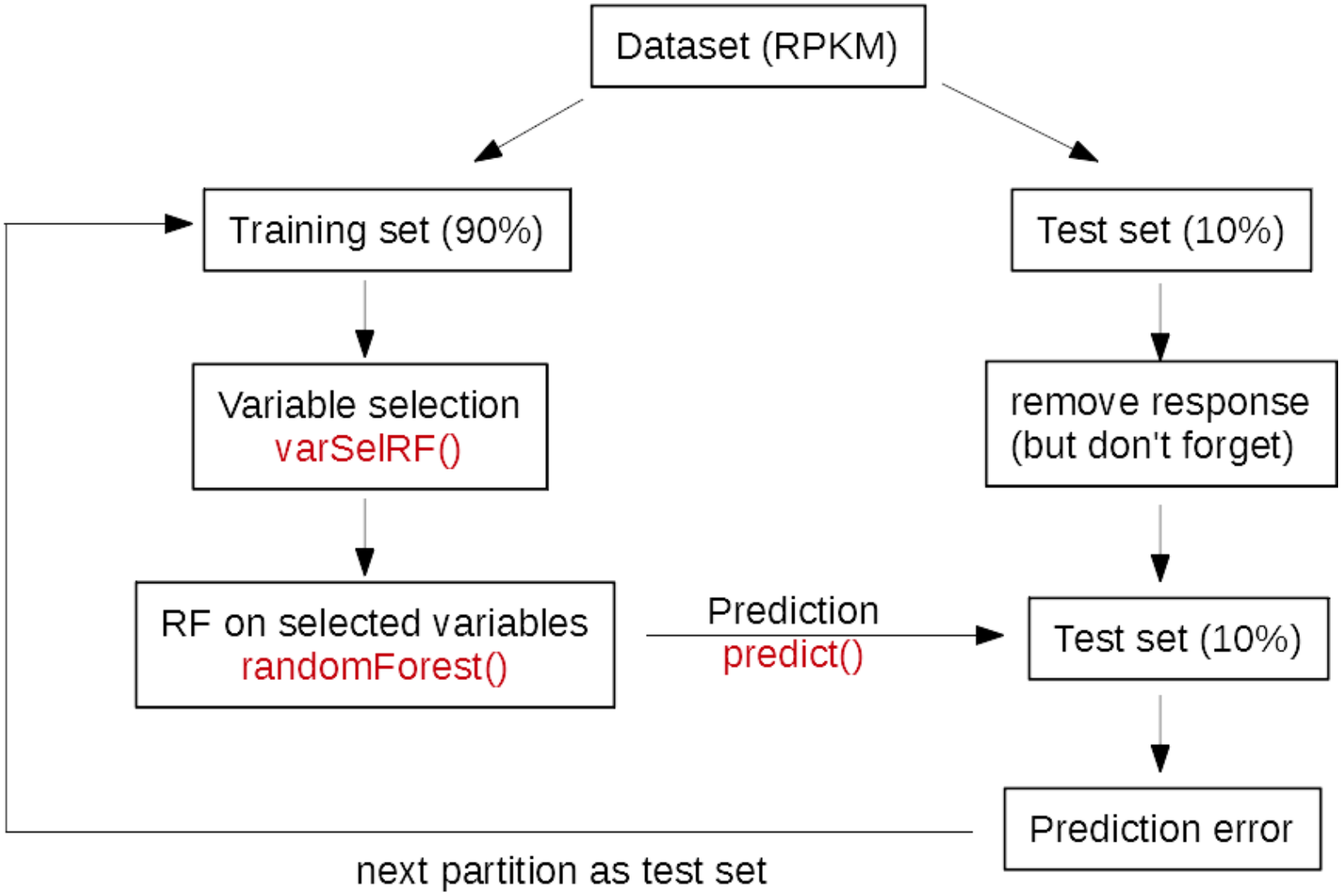
- Feature selection must be made on the training set only
- The test set must **not** be included in feature selection



http://gemler.fzv.uni-mb.si/results.php

[1] http://www.alfredo.motta.name/cross-validation-done-wrong

Variable Selection and Cross Validation



Summary

- Expression levels at early age (10/20 weeks) can be used to predict, with some accuracy, lifespan of individuals of killifish (N. furzeri).
- Samples cluster fairly well to the recorded lifespan (in MDS-plot), confirming the lifespan-predictive power of the identified biomarkers.
- 10-fold cross validation shows that almost 70% of the samples of the test set can be classified correctly (but not for a held-out dataset!)
- Classification performance is similar for 10 & 20 week-transcriptomes, and for the change of expression between 10 & 20 weeks.
- Complete separation of the groups is unlikely, as some of the short-lived animals may not have survived for reasons not related to ageing.
- Validation using an independent test set is desirable in order to obtain a more solid assessment of the prediction performance.

Resources

- Rweka: https://cran.r-project.org/web/packages/RWeka/index.html
- randomForest: https://cran.r-project.org/web/packages/randomForest/index.html
- varSelRF: https://cran.r-project.org/web/packages/varSelRF/index.html
- Breiman/Cutler RF: https://www.stat.berkeley.edu/~breiman/RandomForests/
- "RWeka Odds and Ends" by Kurt Hornik (R core team), 2014
- Liaw, Wiener: "Classification and Regression by randomForest", R News, 2002
- Diaz-Uriarte, "GeneSrF and varSelRF ..." http://www.ncbi.nlm.nih.gov/pubmed/17767709
- "A Brief Tour of the Trees and Forests", R-Bloggers, http://www.r-bloggers.com/a-brief-tour-of-the-trees-and-forests/

Appendix

Prediction of Phenotype by Transcriptome classification using Random Forest Machine Learning

Uwe Menzel, 2012

uwe.menzel@matstat.de

www.matstat.org

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

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

Splitting on continuous attributes

- Binary splits: A <= x and A > x
- Attribute A with (sorted) values a_1, a_2, a_3, …, a_n
  consider all possible (n-1) split points
  For efficiency, consider values from a sample

Data Mining Techniques, by M.J.A. Berry and G.S Linoff, 2004

ID3, C4.5 (Quinlan), RF (Breiman)

The ID3 algorithm begins with the original set S as the root node. On each iteration of the algorithm, it iterates through every unused attribute of the set S and calculates the entropy H(S) (or information gain IG(A)) of that attribute. It then selects the attribute which has the smallest entropy (or largest information gain) value. The set S is then split by the selected attribute (e.g. age < 50, 50 <= age < 100, age >= 100) to produce subsets of the data. The algorithm continues to recur on each subset, considering only attributes never selected before.

C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of information entropy. The training data is a set S = {s_1, s_2, ...} of already classified samples. Each sample s_i consists of a p-dimensional vector (x_{1,i}, x_{2,i}, …,x_{p,i}) , where the x_j represent attribute values or features of the sample, as well as the class in which s_i falls.
At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain (difference in entropy). The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurs on the smaller sublists.

The RF bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

https://en.wikipedia.org/wiki/ID3_algorithm
https://en.wikipedia.org/wiki/C4.5_algorithm
https://en.wikipedia.org/wiki/Random_forest

Leo Breiman
Statistics Department
University of California
Berkeley, CA 94720

January 2001

Abstract

Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error for forests converges a.s. to a limit as the number of trees in the forest becomes large. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them. Using a random selection of features to split each node yields error rates that compare favorably to Adaboost (Freund and Schapire[1996]), but are more robust with respect to noise. Internal estimates monitor error, strength, and correlation and these are used to show the response to increasing the number of features used in the splitting. Internal estimates are also used to measure variable importance. These ideas are also applicable to regression.