# Introduction to Bioinformatics

## Data Mining

Lecturer: Jan Baumbach
Teaching assistant(s): Diogo Marinho

# Outline

- Intro;
- Naïve Bayes classifier;
- Decision Trees;
- Node Splitting;
- Overfitting & Tree Pruning
- Cross Validation & Bootstrapping
- Random Forest;
- Feature Selection;
- ROC plot;
- Label Randomization;
- Examples…

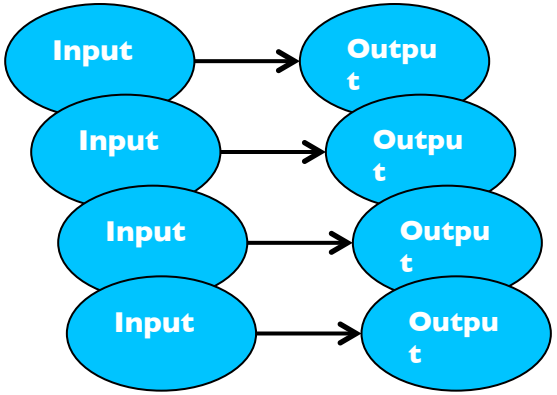
# Machine Learning – A Definition

**”Field of study that gives computers the ability to learn without being explicitly programmed.”**
**– Arthur Samuel, 1959**

**”A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.”**

# Types of Machine Learning

## Supervised Learning



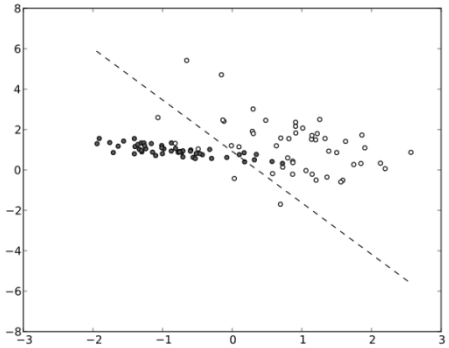

Training Set with labeled data

Often classification problems

## Semi-Supervised Learning

- Learn mainly on unlabeled data
- Provide small training set with labels
- Can lead to better training accuracy

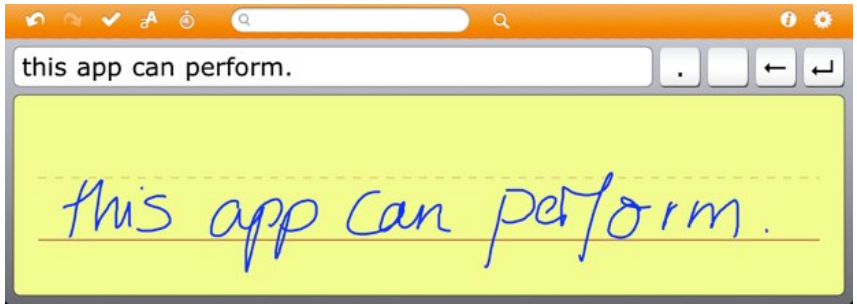
## Unsupervised Learning



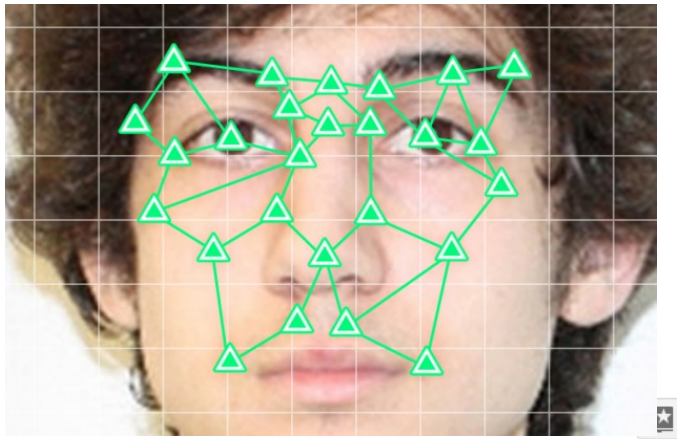
Find hidden structures in unlabeled data

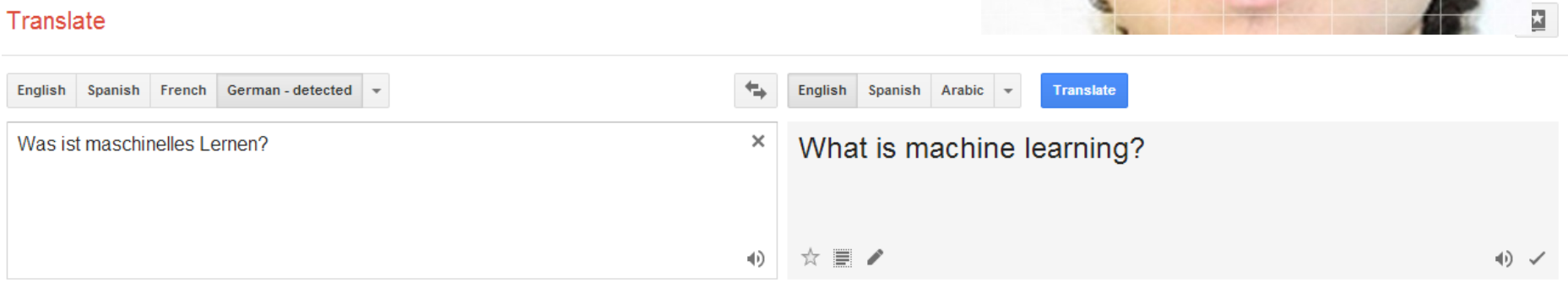
Often clustering problems

# Machine Learning is hip


There are hundreds of examples of machine learning in our daily life…





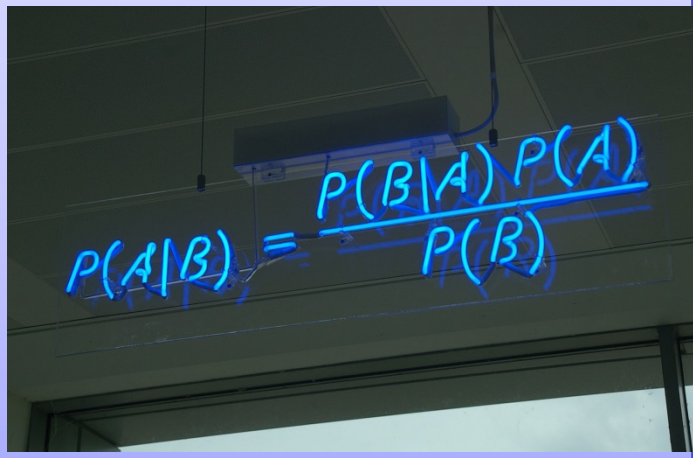
# Naïve Bayes

# Naive Bayes Classifier

- Supervised classification method
- Assumes total independence of features (naive)
- Works well for many real-world problems
- Outperformed by more sophisticated methods
- Requires only small amount of training data

**Do you remember Bayes Theorem?**

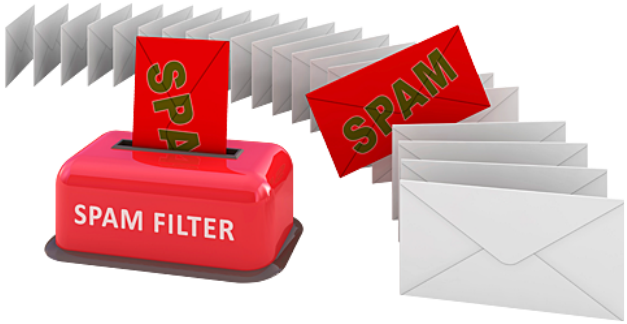
$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}.$$




**Bayesian Classification**

$$P(C|X_1, \ldots, X_n) = \frac{P(C)\ P(X_1, \ldots, X_n|C)}{P(X_1, \ldots, X_n)}$$

# Example: Spam Filter




First approach:

- C = {Spam, NotSpam}
- X is sequence of words in document
- X and $P(C|X_1, X_2, \ldots, X_n)$ are huge:
  - Article with 1000 words, $X=\{X_1,\ldots,X_{1000}\}$
  - $X_i$ represents i-th word in document
  - The domain of $X_i$ is the entire vocabulary, e.g. 10,000 words
  - $10{,}000^{1000} = 10^{4000}$
  - This won't work

# Bag of Words Model

- Additional assumption / simplification:

The position in a document does not matter

- $X_i \begin{cases} 1 \\ 0 \end{cases}$ Boolean: Word is in document

# Two Bags: Spam and NoSpam

- Priors: $P(C_i)$ easily estimated from data


# Conditional Independence

- Conditionals:
  - $X = X_1, \ldots, X_n$
  - Estimate $P(X_1, \ldots, X_n | C_i)$
  - Grows exponentially with n
  - (inspite of bag of words assumption)
- Is the occurance of word combinations(!) independent?

**Conditional Independence:**

- $P(X, Y \,|Z) = P(X|Z)\, P(Y|Z)$

# Naive Bayes Assumption

- $P(X_1, X_2 \mid C) = P(X_1|C)\, P(X_2 \,|C)$
- Generally: $P(X_1, \ldots, X_n \mid C) = \prod_i P(X_i \mid C)$
- As before: $P(X, Y \,|Z) = P(X|Z)\, P(Y|Z)$

# Naive Bayes Classifier

- Spam filter:

  spam/no-spam wrt. $max \begin{cases} P(C_{spam})P(X_1, \dots, X_n \mid C_{spam}) \\ P(C_{nospam})P(X_1, \dots, X_n \mid C_{nospam}) \end{cases}$

- MLE of parameters from training data:
  - Prior: $P(C = C_{spam}) = \frac{Count(C=C_{spam})}{\sum_{C'} Count(C')} = \frac{\#spam\ mails}{\#all\ mails}$
  - Likelihood: $P(X_i = x \mid C = C_{spam}) = \frac{Count(X_i=x, C=C_{spam})}{\sum_{x'} Count(X_i=x')}$

$$= \frac{\#Word\ X_i\ appears\ in\ spam\ mails}{\#Word\ X_i\ appears\ in\ any\ mail}$$

# Problems with Naive Bayes

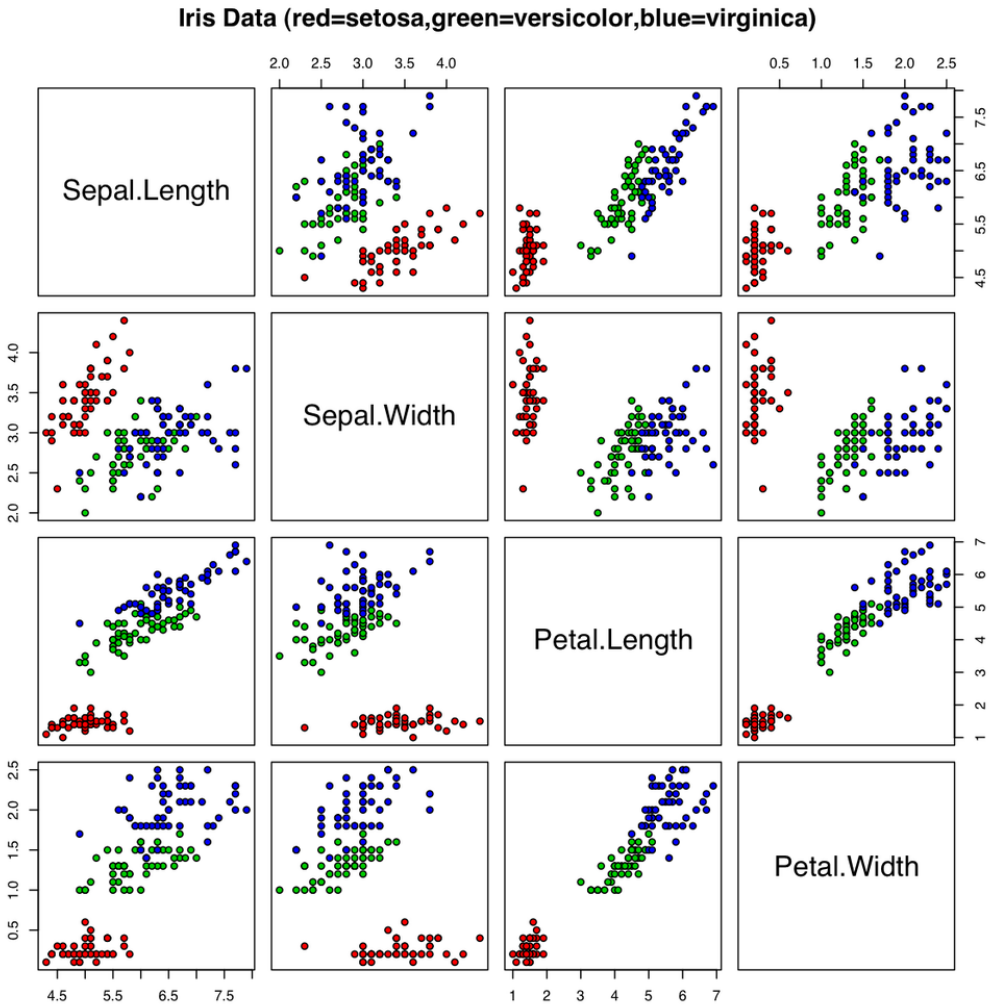
- Naive Bayes is the single most used classifier out there
- But,
  - Usually, features are not conditionally independent.
  - Probabilities P(C|X) often biased towards 0 or 1
  - Insufficient training data is problematic, e.g. "enlargement" was never part of the training set, therefore
  - $P(X_1 = "enlargement" \mid C_{spam}) = 0$ and ultimately
  - $P(X_1, \dots, X_n \mid C) = \prod_i P(X_i \mid C) = 0$
  - No matter what other words $X_2, \dots, X_n$ are included.

# Naive Bayes in R

```
install.packages("e1071")
library(e1071)
data(iris)

m <- naiveBayes(Species ~ ., data = iris)
table(predict(m, iris), iris[,5])
```

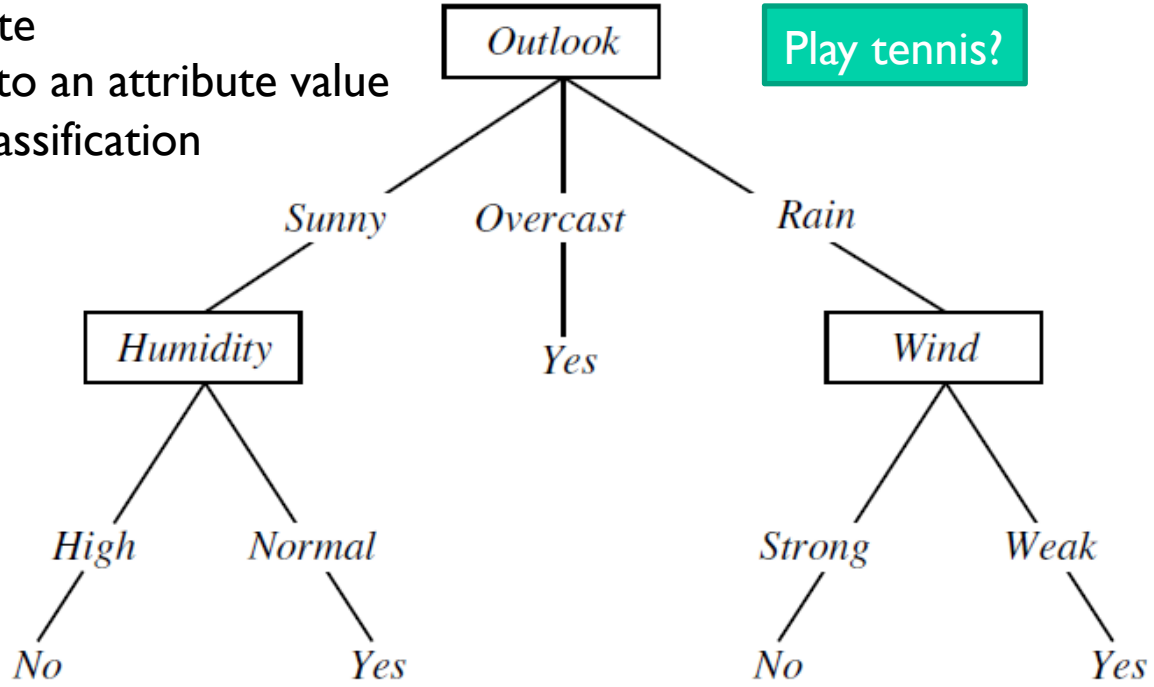
| | setosa | versicolor | virginica |
|---|---|---|---|
| setosa | 50 | 0 | 0 |
| versicolor | 0 | 47 | 3 |
| virginica | 0 | 3 | 47 |



# Decision Trees

# Decision Trees

- Each node tests an attribute
- Each branch corresponds to an attribute value
- Each leaf node assigns a classification

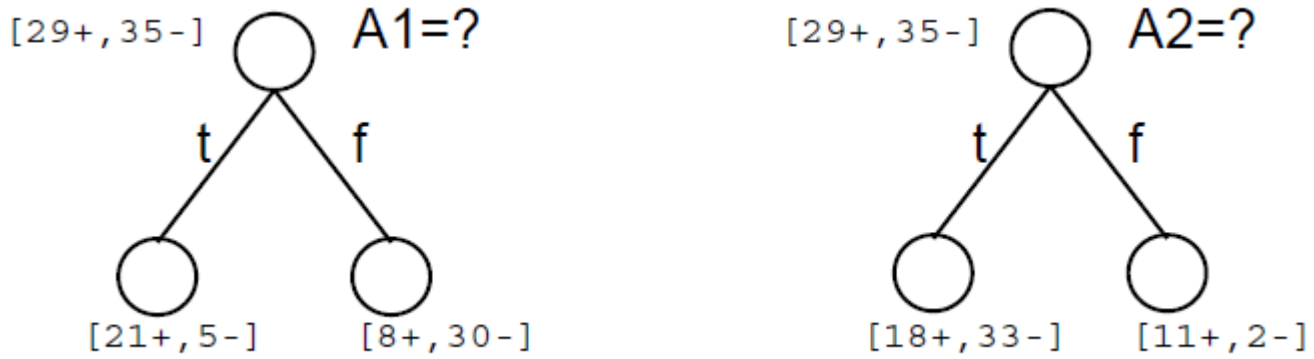



Requirements:
•Attribute – value pairs
•Target function is discrete

# Top-Down Induction

1. A ⟵ the "best" decision attribute for the next *node*
2. Assign A as decision attribute for *node*
3. For each value of A, create new descendant of *node*
4. Sort training examples to leaf nodes
5. IF training examples are perfectly classified STOP
   ELSE GOTO step 1

Which of the two attributes is best?



# Quality of a Split and Impurity

- **Objective:** Construct a tree with leaf nodes that contain cases of a single class only.
- **Node impurity** $i(t) = \phi(p_1, p_2, \dots, p_j)$ **of node t in split s is any function** $\phi(p_1, p_2, \dots, p_j)$ where $p_j$ are relative occurences (frequency of objects of a certain class j), with…
- **Requirements:**
  - $\phi$ maximal for even distribution $(\frac{1}{j}, \frac{1}{j}, \dots, \frac{1}{j})$
  - $\phi$ minimal when all observations belong to a single class, i.e. $(1, 0, \dots, 0), (0, 1, 0, \dots, 0), \dots, (0, 0, \dots, 1)$
  - $\phi$ is symmetric for $p_1, p_2, \dots, p_j$

# Reduction of Impurity

Goal: Find a tree with minimal impurity over all splits and all nodes.

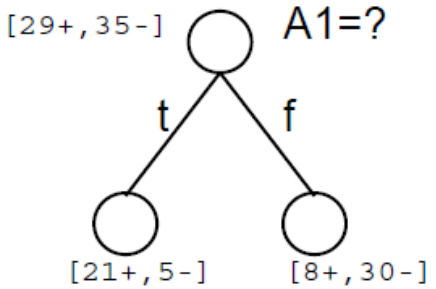
Note that we can measure the quality of a split s "below" node t as…

$$\Delta i(s,t) = i(t) - \pi(l)i(l) - \pi(r)i(r)$$

where

$\pi(l)$ proportion of cases sent to the left

$\pi(r)$ proportion sent to the right.



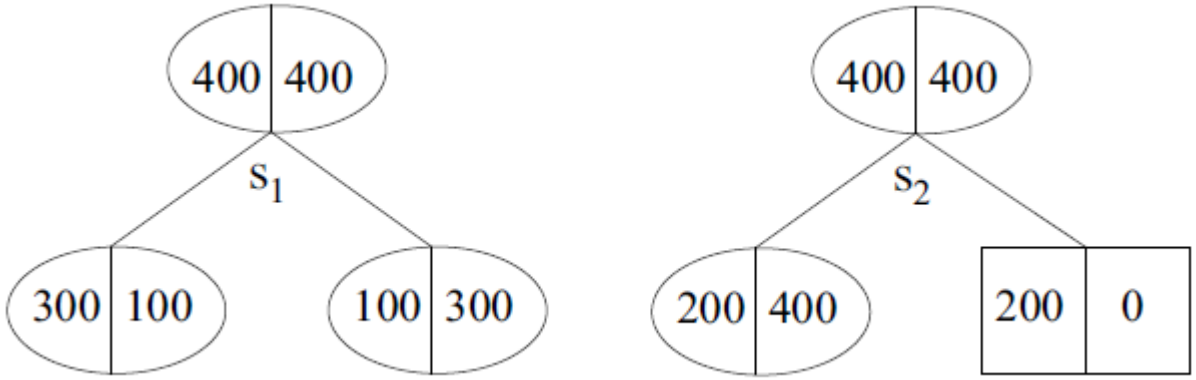
# Resubstitution Error

One possible impurity function:

$$i(t) = \phi(p_1, p_2, \ldots, p_j) = 1 - max_j\, p(j|t)$$

where $p(j|t)$ is the relative frequency of class j in node t.

Which split would you prefer?




Both splits have a the same resubstitution error, i.e. the same impurity. If we would have to stop here (max. tree depth = 2), we would prefer the right solution, as the right node is ”pure”.

# Gini impurity / Gini index

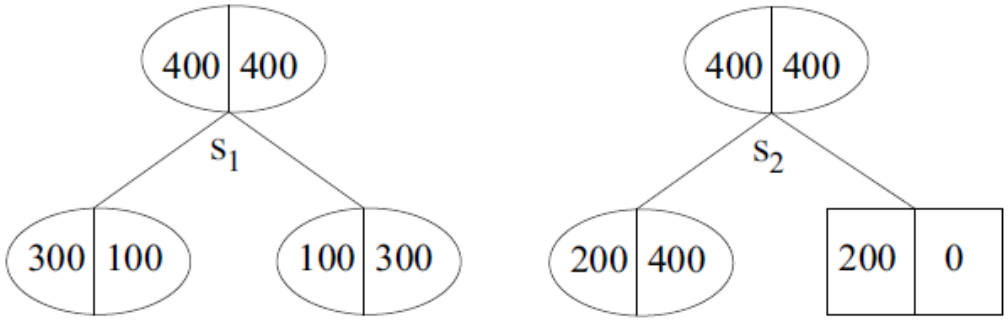
- In the previous example we intuitively prefer the 2nd split.
- We can achieve this with the Gini index. For two classes A,B:

$$i(t) = p(A|t)\, p(B|t) = p(A|t) * \big(1 - p(A|t)\big)$$

Recall: $p(A|t)$ is the frequency of cases of class A in node t.

- More general:

$$i(t) = \sum_{\cdot} p(j|t)(1 - p(j|t))$$



# Entropy / Information Gain

- A well-known alternative to the Gini index is entropy, also known as information gain. For two classes it is defined as:

$$i(t) = -p(A|t)\log p(A|t) - p(B|t)\log p(B|t)$$

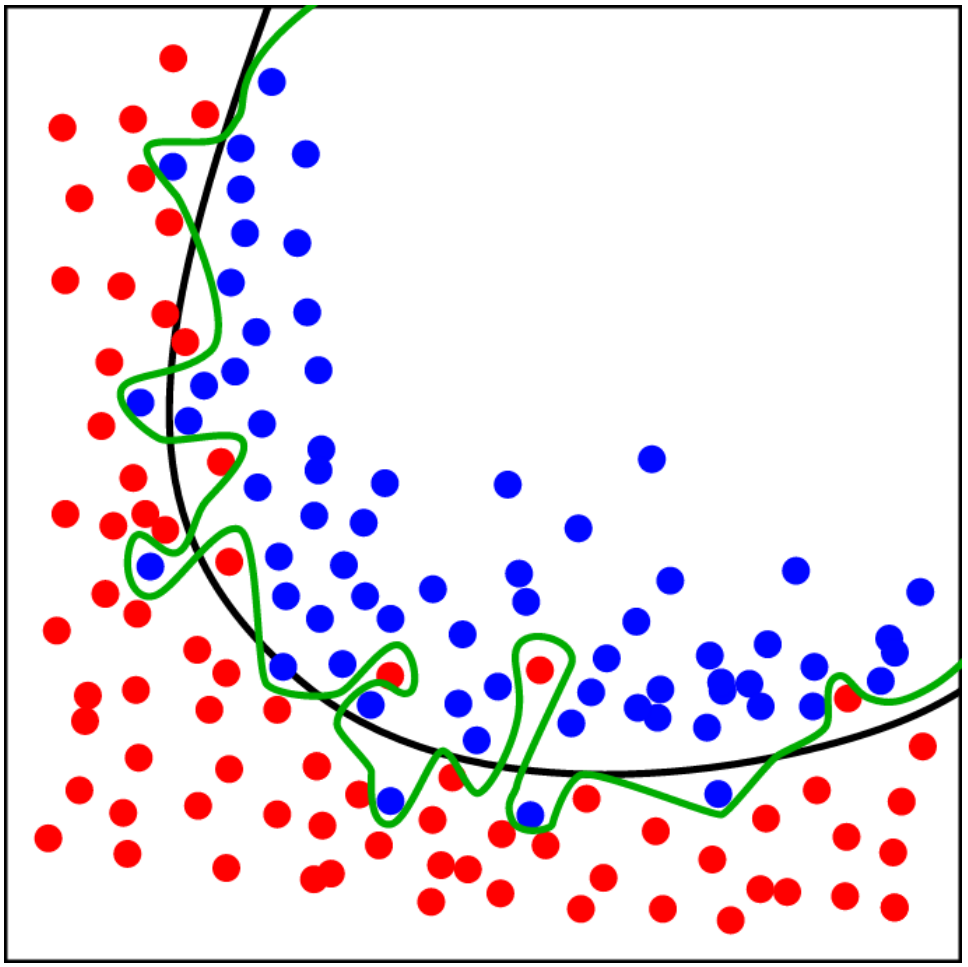
- Generalized for more than two classes:

$$i(t) = -\sum_j p(j|t)\log p(j|t)$$

- **Interpretation:** Just as with sequence logos. Measure of the average amount of information generated by drawing randomly with replacement from this node. If a node is pure,

# Overfitting

- The final model often perfectly explains the training data.
- But: Have we in fact been fitting noise?



# Stopping Rules & Pruning

- There are two basic approaches to avoid overfitting:
  - **Stopping Rules:** Don't expand a node if the impurity reduction of the best split is below some threshold
  - **Pruning:** Grow a very large tree and merge back nodes.

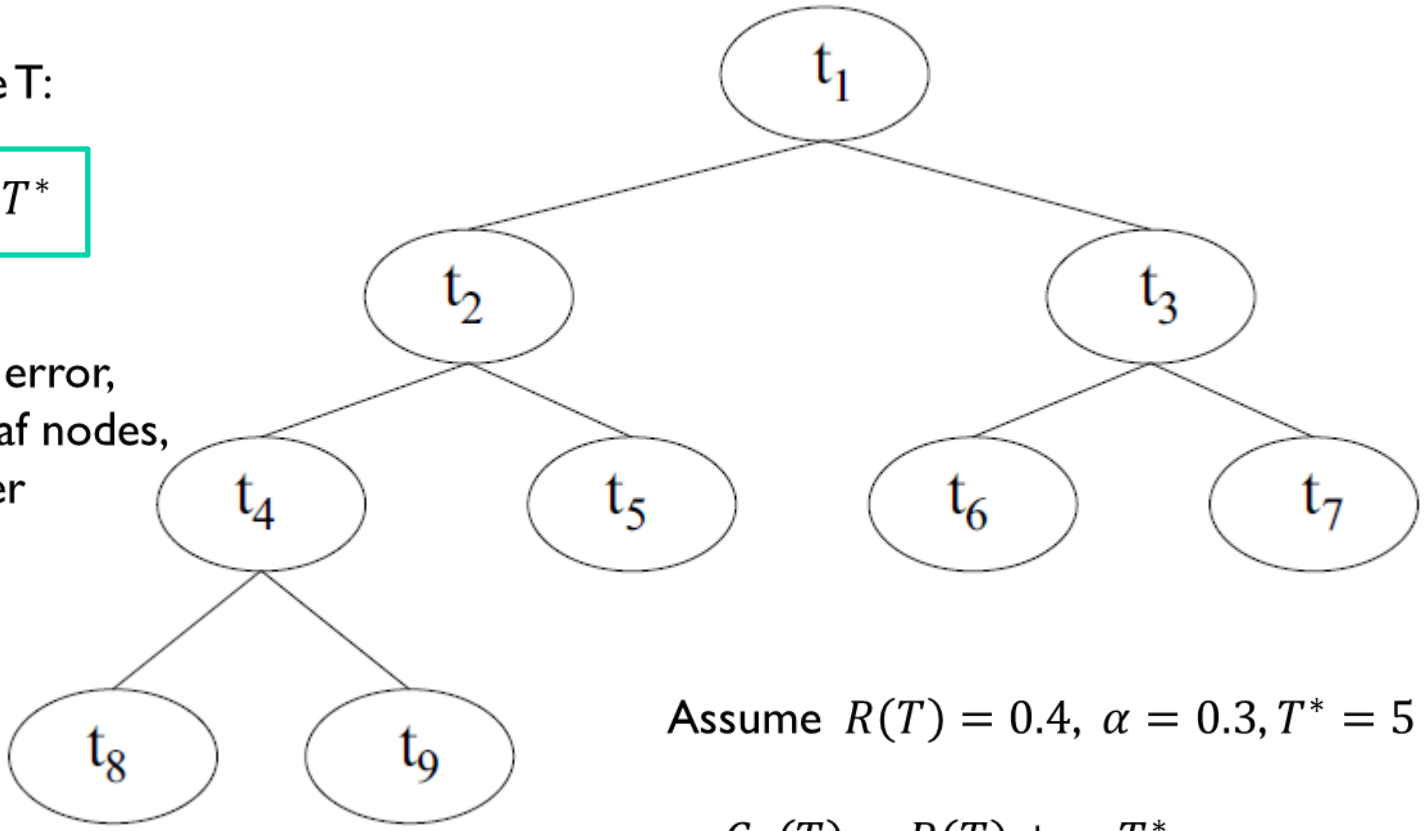
  Question: Why could tree pruning be better?

# Cost Complexity Pruning

Total Costs of a tree T:

$$C_\alpha(T) = R(T) + \alpha T^*$$

Where
$R(T)$ resubstitution error,
$T^*$ the number of leaf nodes,
$\alpha$ the cost parameter




Assume $R(T) = 0.4,\ \alpha = 0.3, T^* = 5$

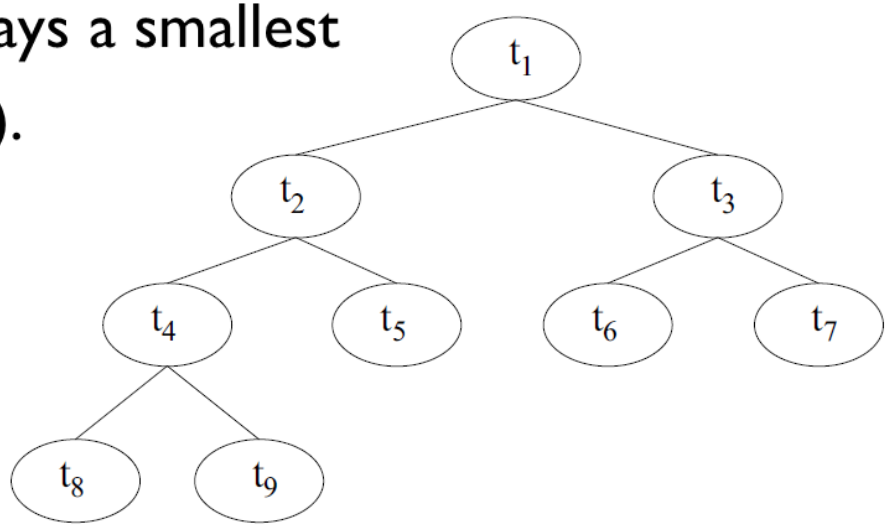
$$C_\alpha(T) = R(T) + \alpha T^*$$
$$= 0.4 + 0.3 * 5 = \mathbf{1.9}$$

# Cost Complexity Pruning

- For each node, compute how much pruning would increase the total costs.
- Note that $\alpha$ controls the balance between the tree costs $R(T)$ and the number of leaf nodes, as both cannot be minimized at the same time.
- Iteratively prune the "weakest link".
- For a given value of $\alpha$, there is always a smallest minimizing subtree (proof omitted).

Assume $R(T') = 0.5$
Now $T^* = 3$

$$C_\alpha(T') = R(T') + \alpha T'^* = 0.5 + 0.3 * 3 = \mathbf{1.4}$$

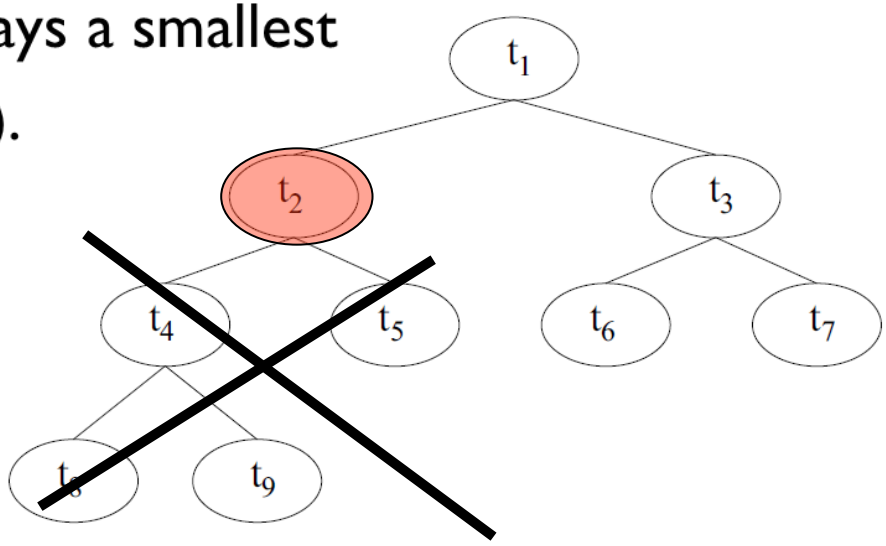


# Cost Complexity Pruning

- For each node, compute how much pruning would increase the total costs.
- Note that $\alpha$ controls the balance between the tree costs $R(T)$ and the number of leaf nodes, as both cannot be minimized at the same time.
- Iteratively prune the "weakest link".
- For a given value of $\alpha$, there is always a smallest minimizing subtree (proof omitted).

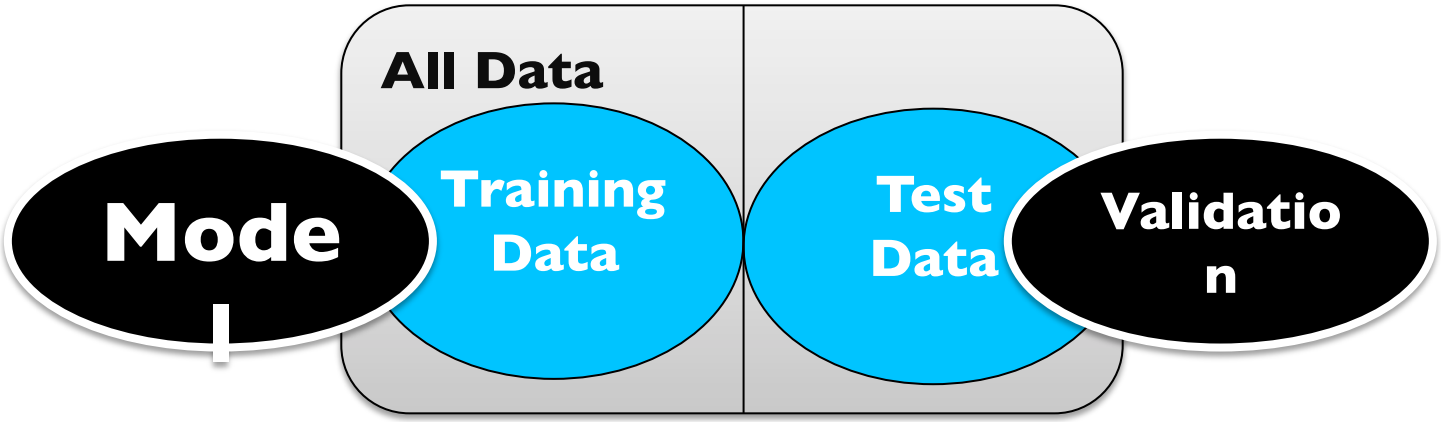
Assume $R(T') = 0.5$
Now $T^* = 3$

$$C_\alpha(T') = R(T') + \alpha T'^*$$
$$= 0.5 + 0.3 * 3 = \mathbf{1.4}$$



# Splitting the Data

- A quite effective way to assess ”overfitting” is splitting the data into a training and test set.
- The model is trained exclusively on the training data.
- And subsequently validated on the test data.
- Overfitted models will perform worse on the test data

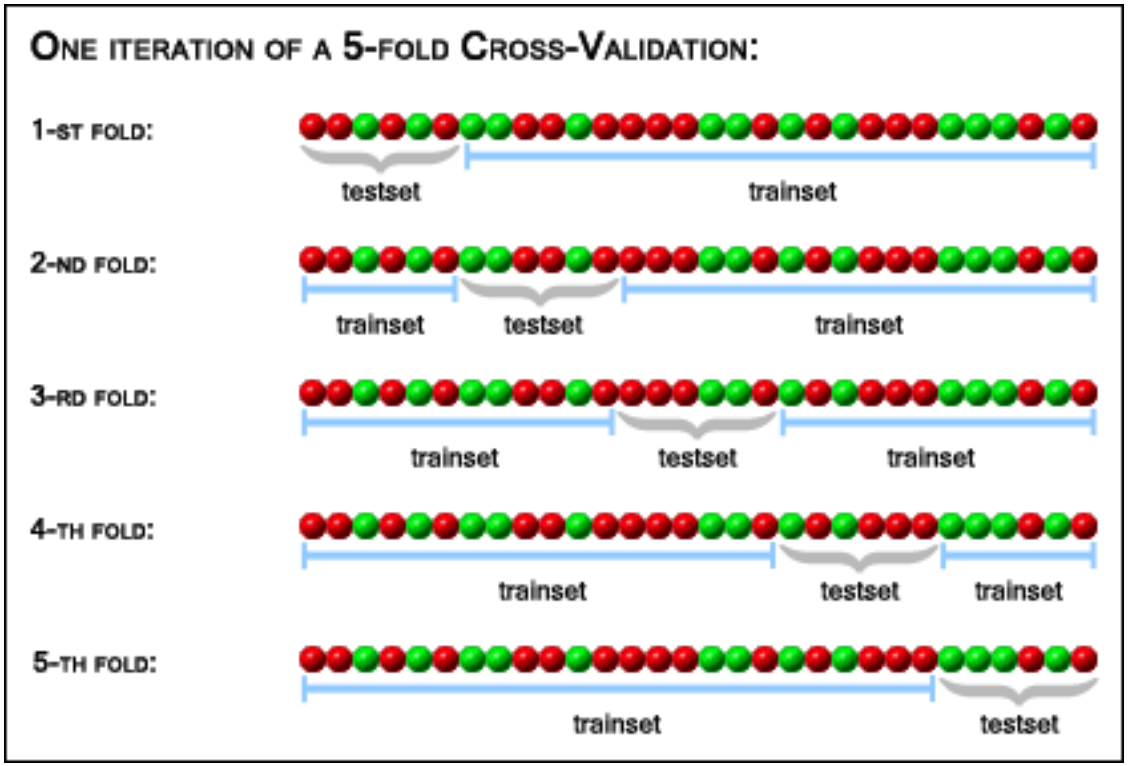


# Cross-Validation

- Problems with test data:
  - If you have a relatively small data set, you waste a large part that you want to use to train your model.
  - The model might look quite different depending on the samples used for the training set.
- Solution: Create multiple splits of the data

# K-fold Cross Validation

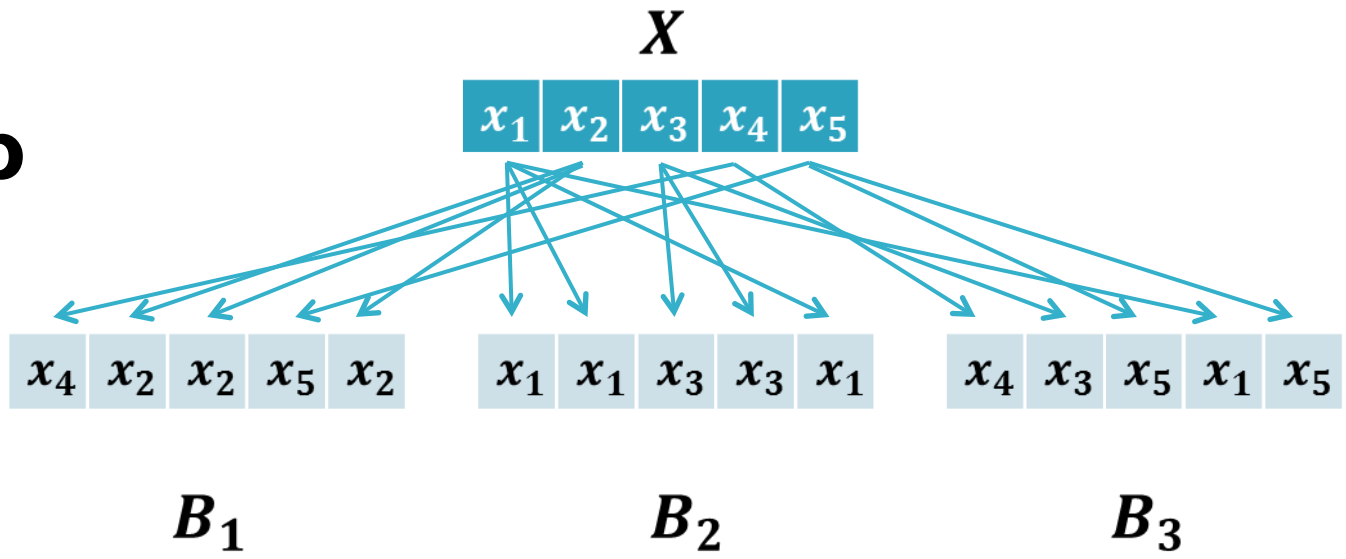
- Generate random permutation of the sample set
- Split into K subsets (called folds) of equal size.
- Of the K folds use one as test data and K-1 as train data
- Repeat until each fold has been used as test data once.
- Identify best features
- To improve accuracy: Repeat with new permutation.



# Leave-one-out Cross Validation

- Use a single sample as validation set, remaining samples as training data.
- Repeat until every sample has been used exactly once as validation data.
- Special case of K-fold cross validation with K = number of samples

# Bootstrap




- Draw bootstrap samples of size n repeatedly from the original data X by simple random sampling with replacement.
- The entire data set is available for model training
- Bootstrap leads to lower variance, but has a larger bias than cross validation.
- There are various implementations of the bootstrap that try to reduce this bias.

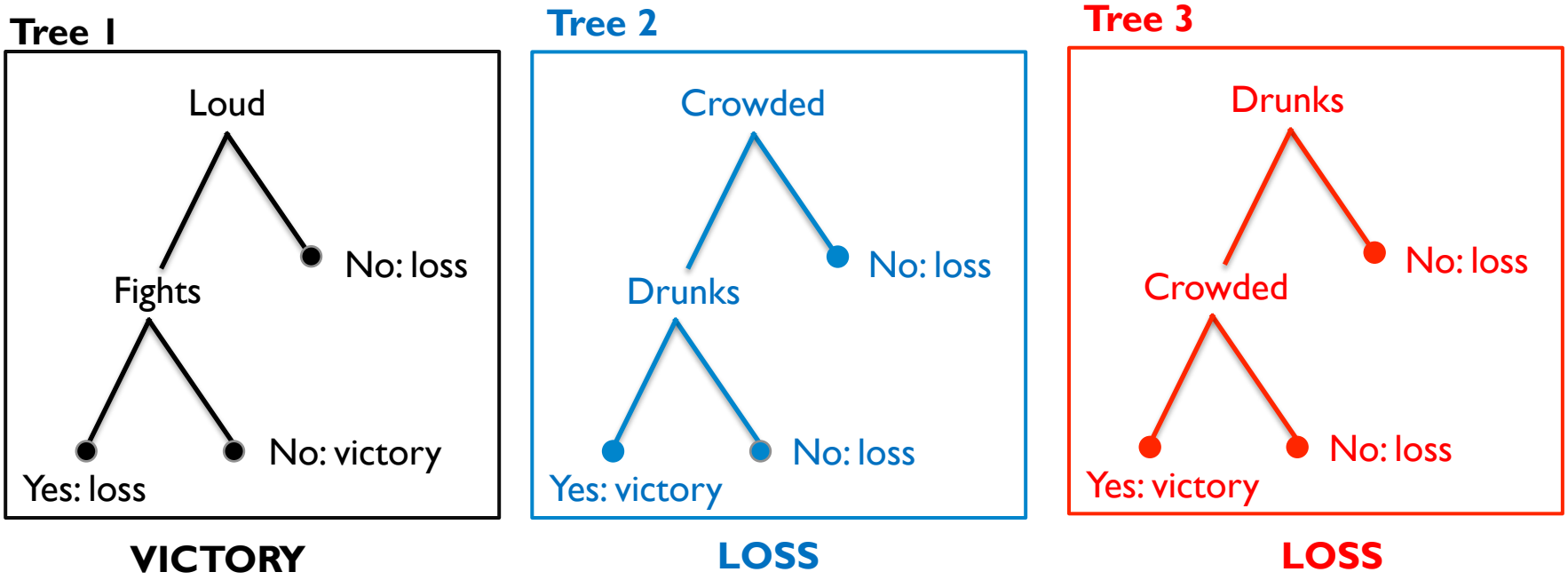
# Random Forest

# Random Forest
## (Main Features)

- State-of-the-art;
- Runs efficiently on large data bases;
- Gives estimates of what variables are important in the classification;
- Generates an internal unbiased estimate of the generalization error as the forest building progresses (out-of-bag);
- Has methods for balancing error in class population unbalanced data sets;
- Generated forests can be saved for future use on other data.

# Random Forest
## (Intuition)

Predict OB victory/loss based on sportsbar status after the game
- Crowded? Loud? Drunks? Fights?




New sample: empty, loud, no fights, full of drunks

**Majority Vote: LOSS!**

# Random Forest
## (Algorithm)

1. Grow B trees: For b = 1 to B:

   (a) Draw a bootstrap sample **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 leaf node of the tree, until the minimum node size $n_{min}$ is reached:

      i. Select m variables at random

      ii. Pick the best variable/split-point.

      iii. Split the node.

2. Output the ensemble of trees $\{T_b\}_1^B$

3. Apply majority vote, for instance. (Or more advanced: learn a decision tree

# Performance Evaluation

# Receiver Operating Characteristic (ROC) Curves

## History

Originally developed during WW2 for radar-signal detection methodology (signal-to-noise), hence "**Radar** Receiver Operator Characteristic").

| | | **OBSERVED** | |
|---|---|---|---|
| | | No event (0) | Event (1) |
| **PREDICTED** | No event (0) | **Good!** No resource wasted *(TN)* | **Bad!** Bombs dropped *(FN)* |
| | Event (1) | **Bad!** Resource wasted *(FP)* | **Good!** Fight back (TP) |

# ROC Curves

| | | OBSERVED | |
|---|---|---|---|
| | | No event (0) | Event (1) |
| PREDICTED | No event (0) | **Good!** No resources wasted *(TN)* | **Bad!** Bombs dropped *(FN)* |
| | Event (1) | **Bad!** Resources wasted… *(FP)* | **Good!** Now, fight back (TP) |

**Sensitivity :** $\text{True Positive [TP]} / (\text{True Positive [TP]} + \text{False Negative [FN]})$

**Specificity :** $\text{True Negatives [TN]} / (\text{True Negatives[TN]} + \text{False Negatives[FN]})$

# ROC Curves

**Sensitivity** = Probability of correctly interpreting the radar signal as enemy planes among those times when enemy planes were actually attacking.

$$\text{True Positive [TP]} / (\text{True Positive [TP]} + \text{False Negative [FN]})$$

**Specificity** = Probability of correctly interpreting the radar signal as no enemy planes among those times when no enemy planes were actually coming

$$\text{True Negatives [TN]} / (\text{True Negatives[TN]} + \text{False Negatives[FN]})$$

# ROC Curves

**Cancer detection** (two-part testing procedure)

1. A **screening** test with **high sensitivity**

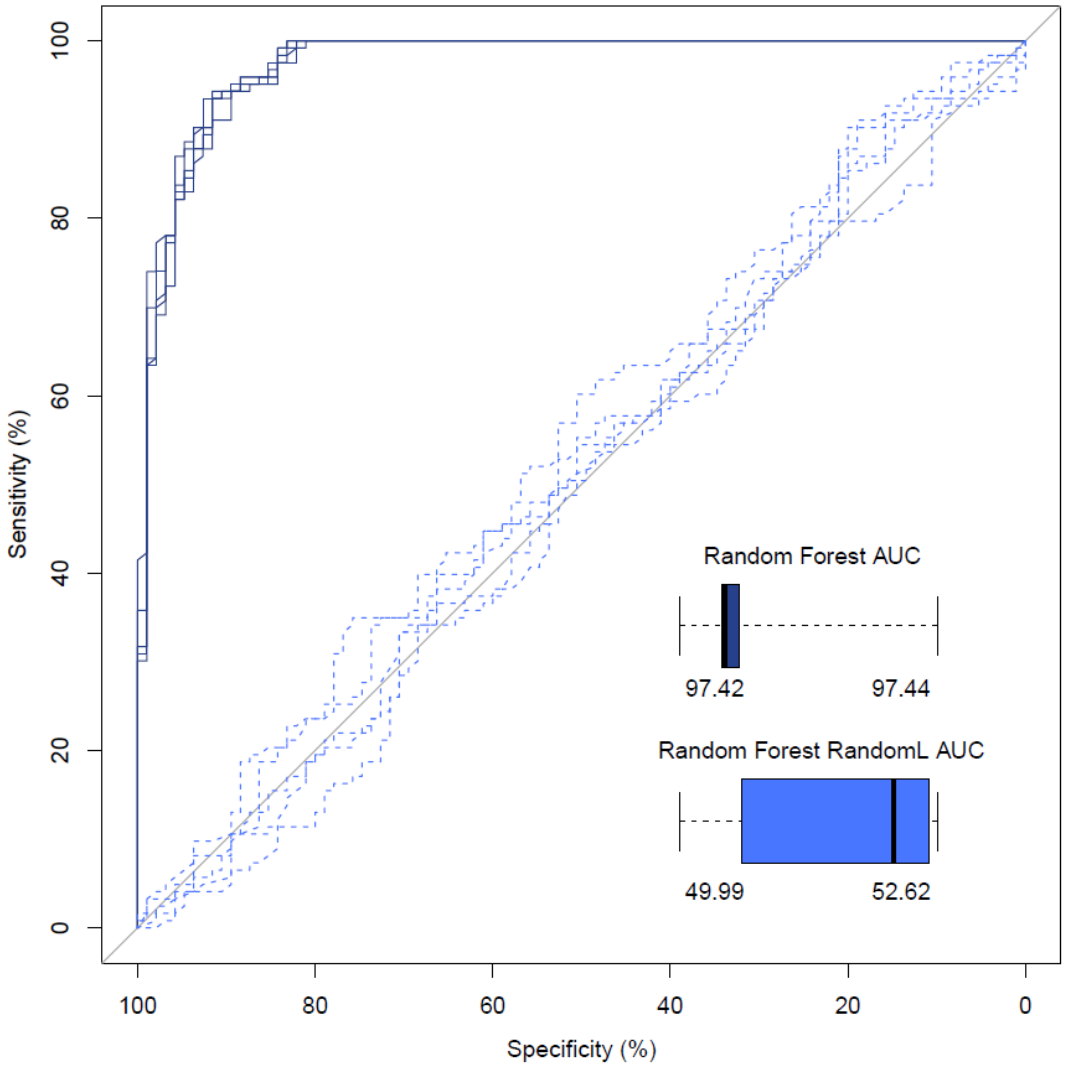
   - to detect as many cancer individuals as possible, allowing for some false positives but very few false negatives. Keep in mind that the false positives might receive unnecessary treatment. Therefore:

2. A **confirmatory** test with **high specificity**

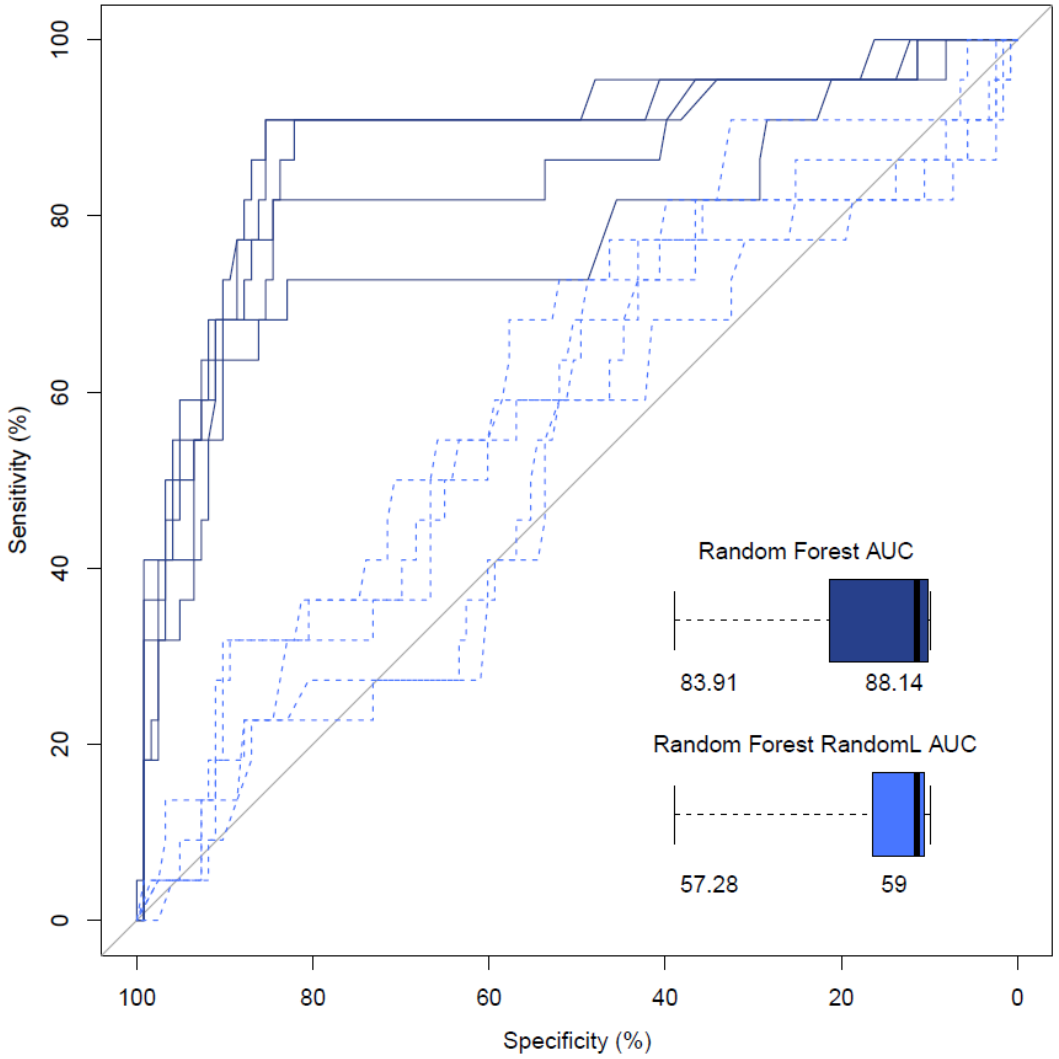
   - to eliminate as many of the false positives as possible.

http://www.aidsmap.com

# ROC Curves


Sensitivity (%)
100
80
60
40
20
0
Specificity (%)
100
80
60
40
20
0
Random Forest AUC
97.42
97.44
Random Forest RandomL AUC
49.99
52.62

# ROC Curves


Sensitivity (%)
100
80
60
40
20
0
Specificity (%)
100
80
60
40
20
0
Random Forest AUC
83.91
88.14
Random Forest RandomL AUC
57.28
59

# Thank you!

# Random Forest
## (Algorithm)

To make a prediction at a new point $x$:

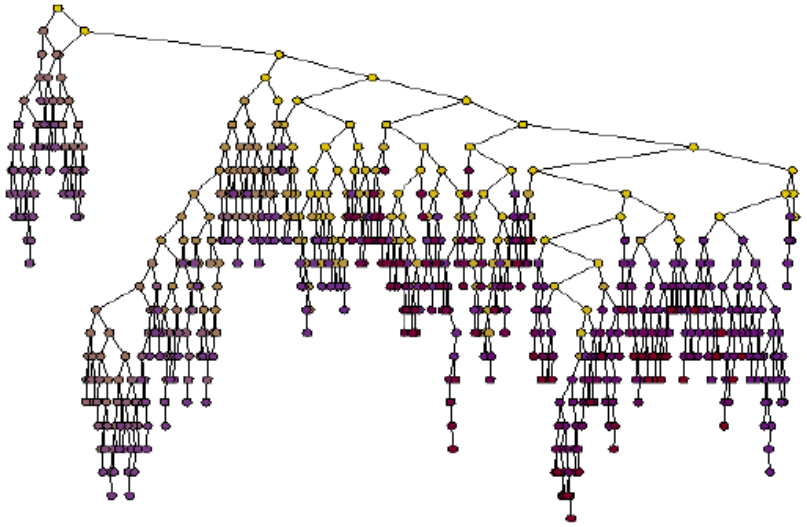
**Regression**: $\hat{f}_{rf}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$

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

# Random Forest

("Ingridients")

1. **Randomization and no-pruning**
   - For each tree, select at random, at each node, a small group of input coordinates to split.
   - Calculate the best split based on these features and cut.
   - The tree is grown to maximum size, without pruning.

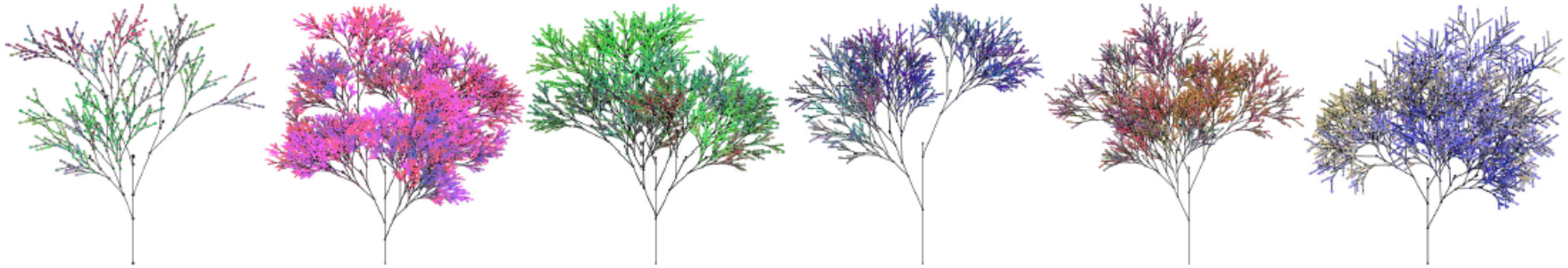


# Random Forest

("Ingridients")

## 2. Bootstrap aggregating (Bagging)

- Final predictions are obtained by aggregating over the ensemble.
- It is fast and easily parallelizable.



# Random Forest
("Ingridients")

## 2. Bootstrap aggregating (Bagging)

**When Bagging works?**

Main property of Bagging (proof omitted)

– Bagging decreases variance of the base model without changing the bias!!!

– Why? Averaging!

Bagging typically helps

– When applied with an over-fitted base model

- High dependency on actual training data

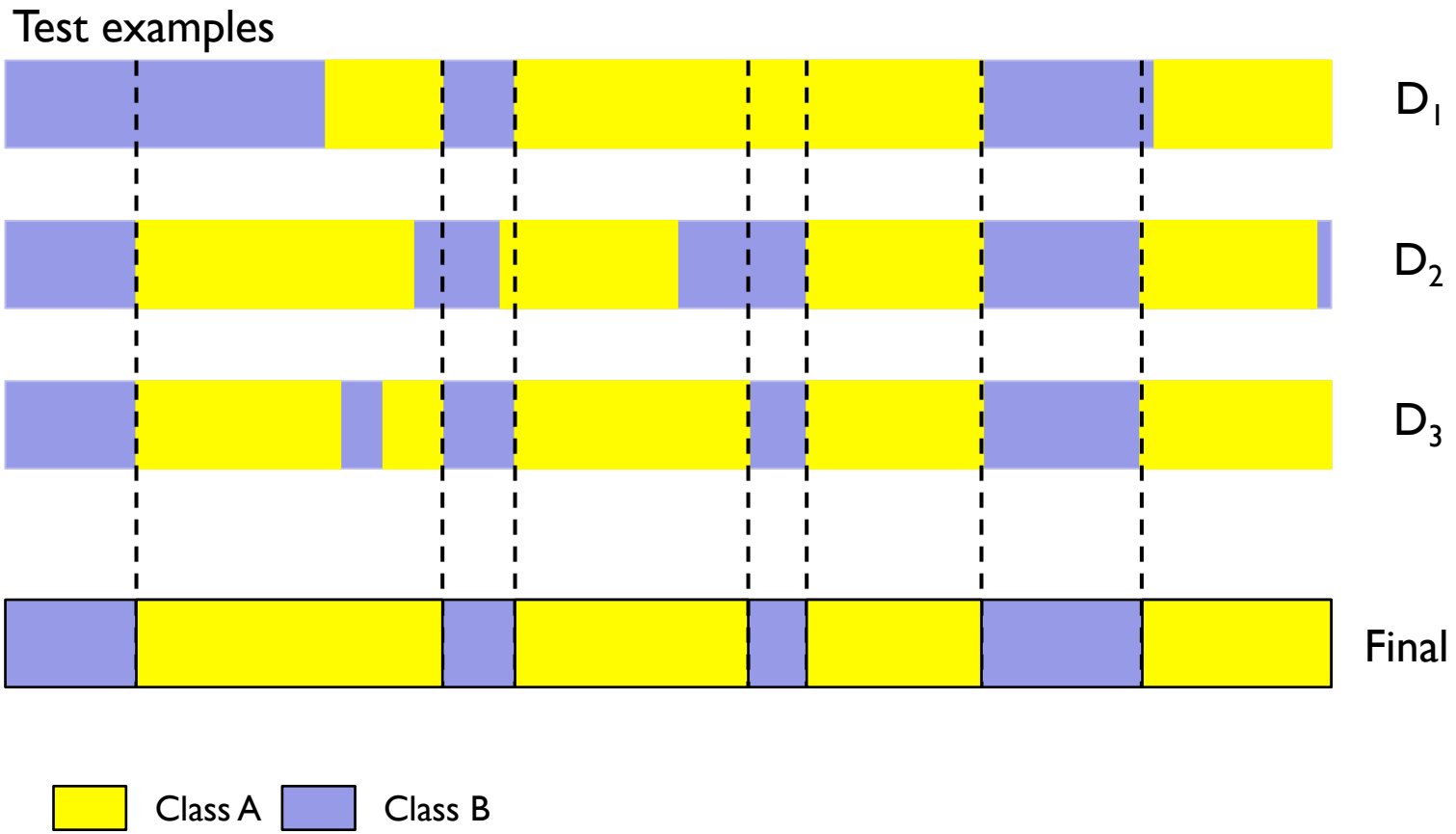
It does not help much

– High bias. When the base model is robust to the changes in the training data (due to sampling)

# Random Forest

("Ingridients")

## 2. Bootstrap aggregating (Bagging)



# Random Forest

("Ingridients")

**3. Out-of-bag error estimate**

Similar to leave-one-out cross-validation

- error estimated internally during run
- almost without any additional computational burden

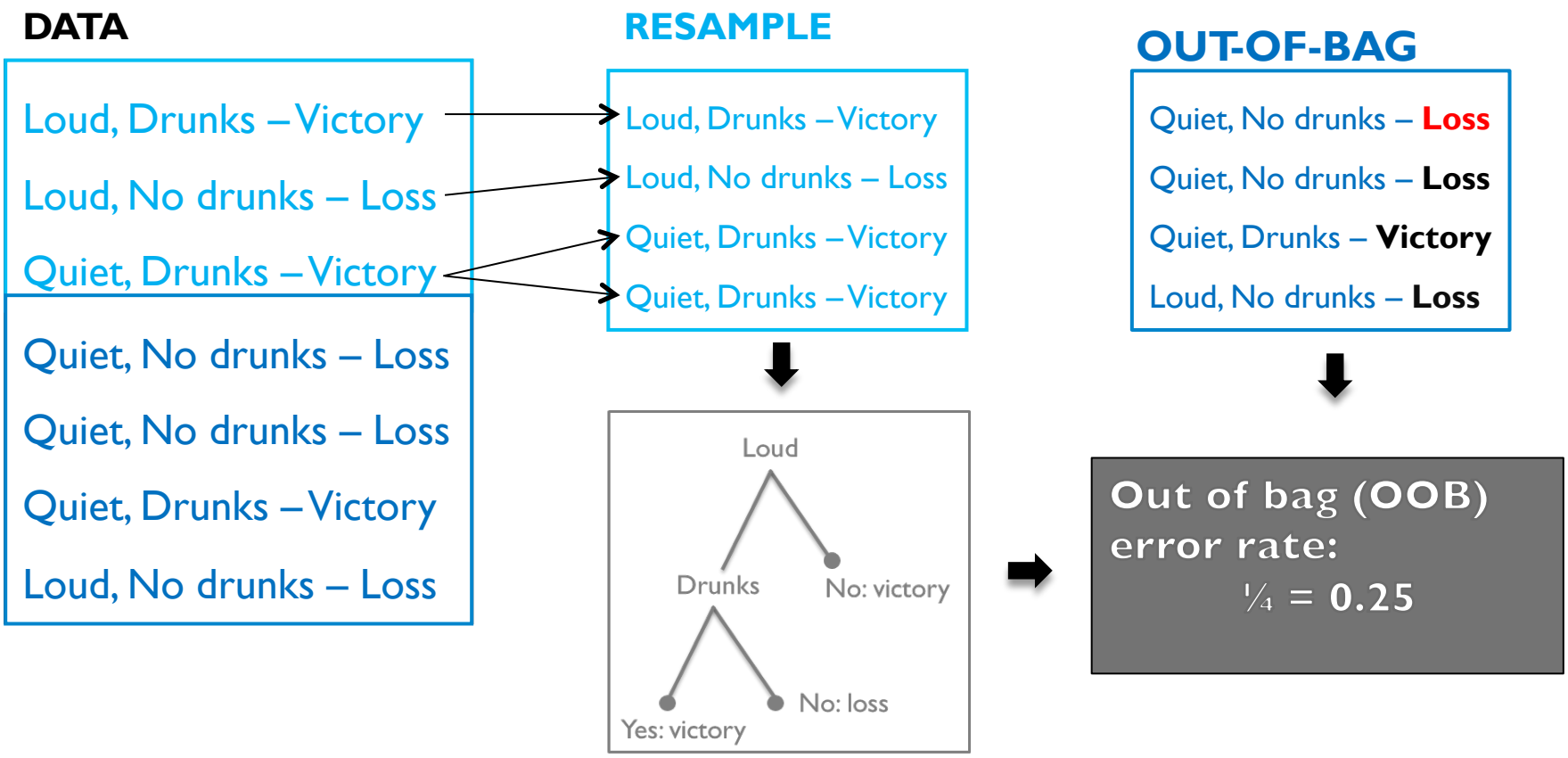
•Each tree is constructed using a different bootstrap sample from the original data (~2/3 of data);

•Put each case left out in the construction of the *k*th tree down the *k*th tree to get a classification…

# Random Forest

("Ingridients")

## 3. Out-of-bag error estimate



# Random Forest
("Ingridients")

## 4. Feature Selection

**1. Preliminary ranking and elimination:**

- Sort the variables in decreasing order of RF scores of importance (Gene Index, etc);
- Cancel the variables of small importance. Let *m* be the number of remaining variables.

**2. Variable selection**

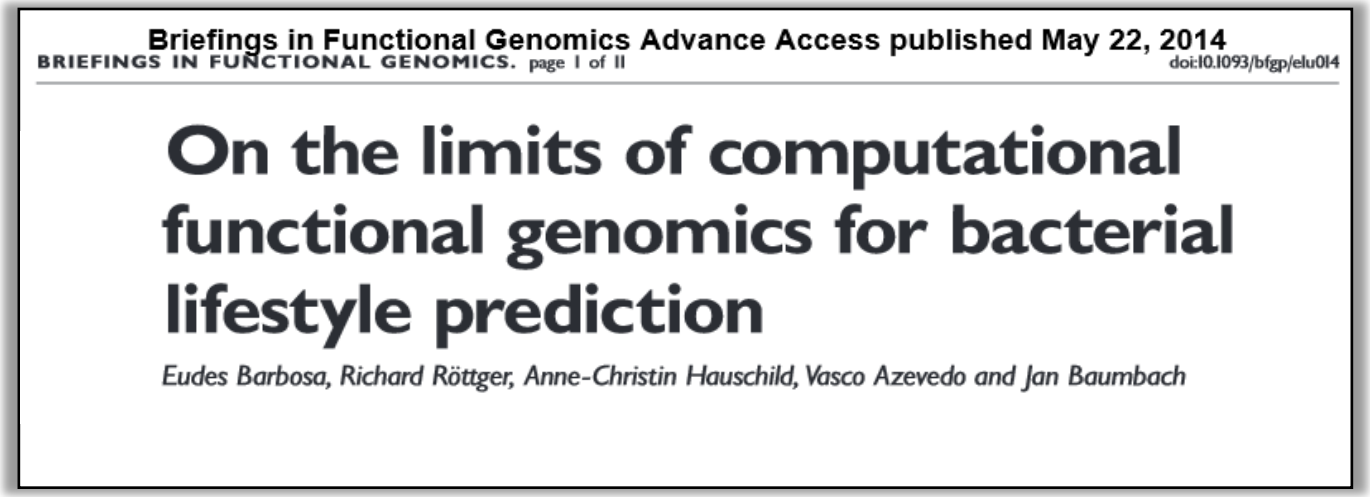
For interpretation

- Construct the nested collection of RF models involving the *k* fist variables, for *k* = 1 to *m*;
- Select the variables involved in the model leading to the smallest OOB error.

For prediction (conservative version)

- Starting from the ordered variables retained for interpretation, construct an ascending sequence of RF models by invoking and testing the variables stepwise;
- The variables of the last model are selected.

# Example I
# Bacterial Lifestyle



Briefings in Functional Genomics Advance Access published May 22, 2014
BRIEFINGS IN FUNCTIONAL GENOMICS. page 1 of 11
doi:10.1093/bfgp/elu014

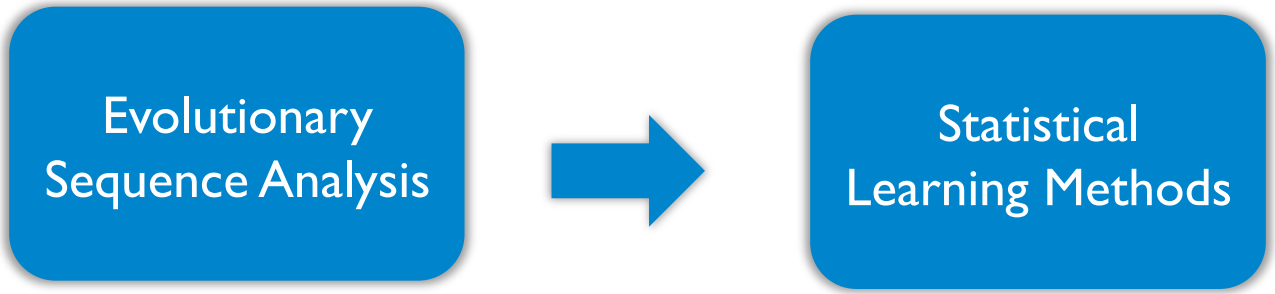
**On the limits of computational functional genomics for bacterial lifestyle prediction**

*Eudes Barbosa, Richard Röttger, Anne-Christin Hauschild, Vasco Azevedo and Jan Baumbach*

# Bacterial Lifestyle

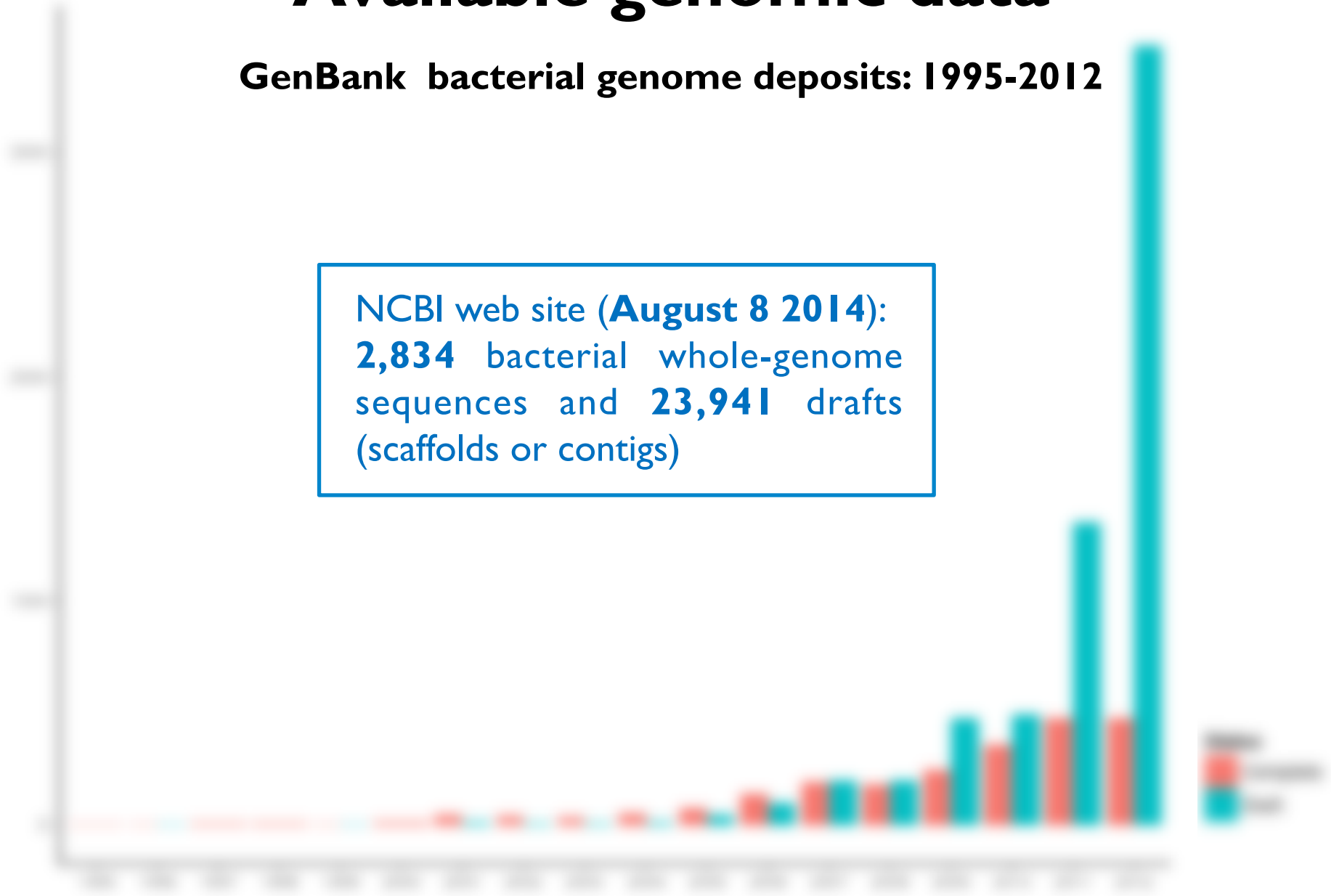
**Objectives:**

- Extract information from the large available genomic data;
- Predict lifestyle based on genome content;
- Feature selection for each lifestyle.



# Available genomic data

**GenBank bacterial genome deposits: 1995-2012**



NCBI web site (**August 8 2014**): **2,834** bacterial whole-genome sequences and **23,941** drafts (scaffolds or contigs)

# Actinobacteria

- One of the biggest clades of bacteria and with several genome sequences deposited (309 complete/3,419 drafts);
- High diversity throughout different lifestyles;
- Cope with a variety of different habitats;
- CMNR group.

*Corynebacterium pseudotuberculosis*
(Transmission electron microscopy).

*Yes, you saw this slide before…*

# Evolutionary sequence analysis
## (homology detection)

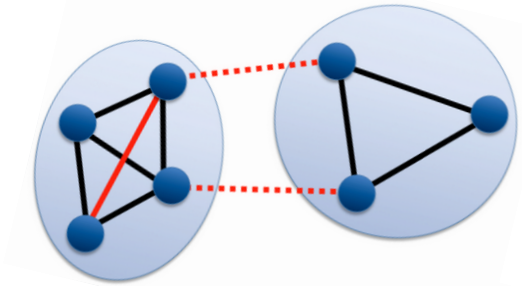
Blast + Fasta

```
AF178033  AF031394  99.63   811 3    0   1   811   99   909   0.0     1542.8
AF178033  AF031393  95.07   811 40   0   1   811   99   909   0.0     1249.4
AF178033  AF178031  94.82   811 42   0   1   811   1    811   0.0     1233.5
AF178033  AF178032  94.57   811 44   0   1   811   1    811   0.0     1217.7
AF178033  AF353195  85.93   803 113  0   1   803   99   901   0.0     670.5
AF178033  AF353192  85.86   806 114  0   1   806   99   904   0.0     670.5
AF178033  AF353196  85.48
AF178033  AF031391  85.48
AF178033  AF353201  85.36
AF178033  AF353200  84.99
AF178033  AF353197  84.99
```

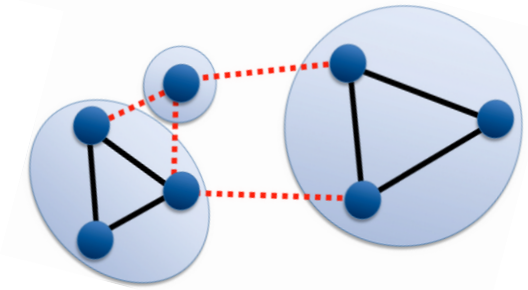
```
>gi77404488
MTTLTLDPHDNISSDDMDYDPRLDVEAQLLCALLWAQPAEARRVAGTVTASDFDRPVYASLYEKITELVN
AGKPHDAAQLAIALGKEGMTAGHKGKQLTTALADITTLGATANVEHYVDSVITQAYRRAFHLAARAMTEA
AEQLPERDLYEHMCAHGRQLRAATERLTAIRGGQL
>gi77404489
MNTPDDERDAWQMHSDGASWDQIGIEMGCSGATAQTLAAEYERRTDAAAHGAQDTLF
>gi77404490
MSDMQGPRVKRLWETIRSRMADVVADTAAAKALTPTDSLAHVKTLVDACTTQLDRSPYAVERQFLLQAVE
LADLPAGKPRAKYVKYFYELGERYSDRWQLNSHLVREKGALGAGRYRDDLGFWFRNNQPWASEHKCRWPR
```

Similarity Matrix

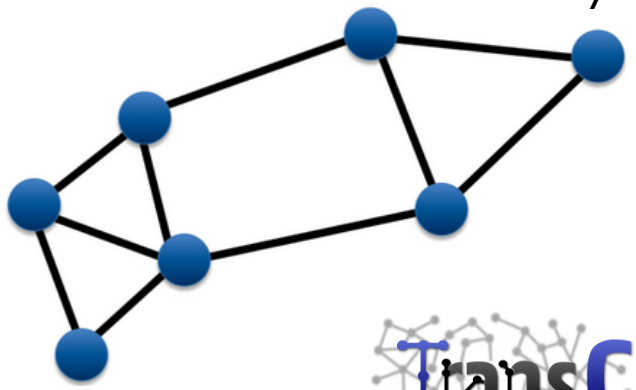
```
GI178033  GI031394  99.63   811
GI178033  GI031393  95.07   811
GI178033  GI178031  94.82   811
GI178033  GI178032  94.57   811
GI178033  GI353195  85.93   803
GI178033  GI353192  85.86   806
GI178033  GI353196  85.48   806
GI178033  GI031391  85.48   806
GI178033  GI353201  85.36   806
GI178033  GI353200  84.99   806
GI178033  GI353197  84.99   806
GI178033  GI353199  84.86   806
GI178033  GI031387  84.80   809
GI178033  GI353198  84.74   806
GI178033  GI031392  84.74   806
GI178033  GI080490  84.89   794
GI178033  GI031395  84.80   809
GI178033  GI353193  84.49   806
GI178033  GI353194  84.24   806
GI178033  GI031402  84.09   811
GI178033  GI031399  84.00   806
```



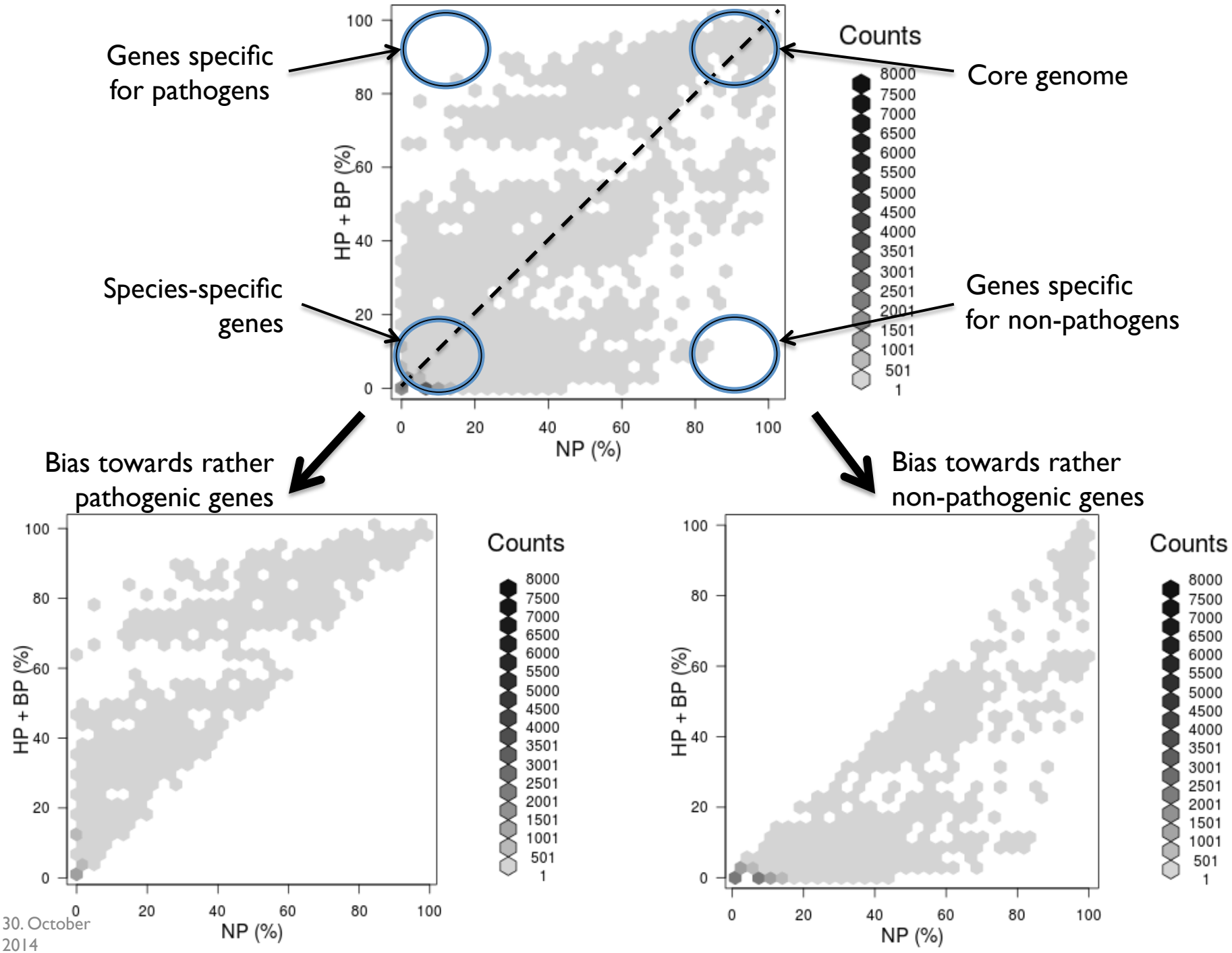
Transitivity Graph



Similarity Graph



TransClust


Genes specific for pathogens
Core genome
Species-specific genes
Genes specific for non-pathogens
Bias towards rather pathogenic genes
Bias towards rather non-pathogenic genes
Counts
8000
7500
7000
6500
6000
5500
5000
4500
4000
3501
3001
2501
2001
1501
1001
501
1
HP + BP (%)
NP (%)
0
20
40
60
80
100

# Statistical learning methods (Random Forest)

R package randomForest;

For each bias:

Data was evaluated using a 5-fold cross validation;

The procedure was repeated five times using different cross validation sets;

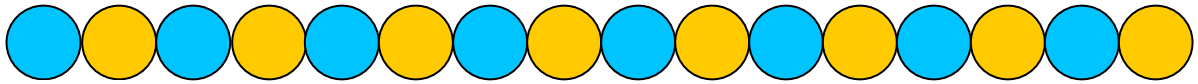
→5,000 trees

Real and random labels…

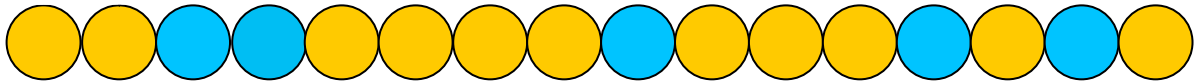
randomForest – http://cran.r-project.org/web/packages/randomForest/

# Statistical learning methods (Random Labels)

Two class problem:



**Real Label**



**Random Label**

# Statistical learning methods (Feature Selection)

Given RF's characteristic of random exploration of features, it can also be utilized for feature selection;

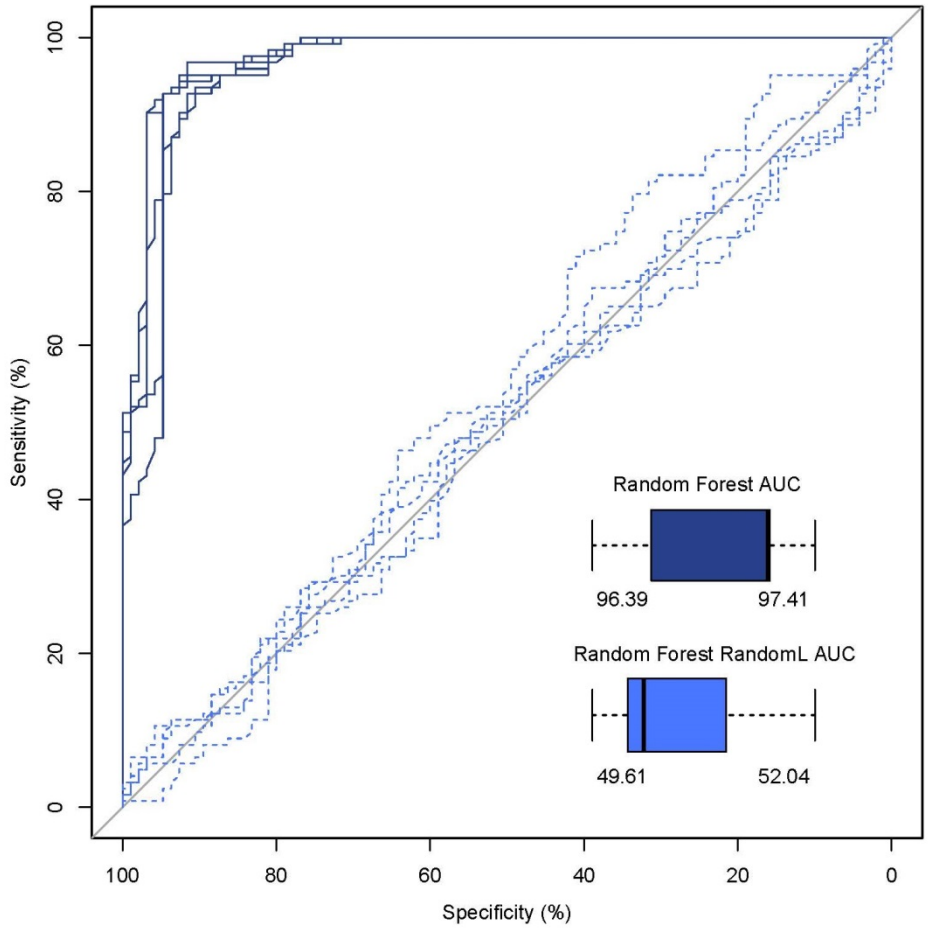
R package varSelRF;

The package successively eliminated the least important variables using a RF's internal error estimate (out-of-bag error) as minimization criterion.

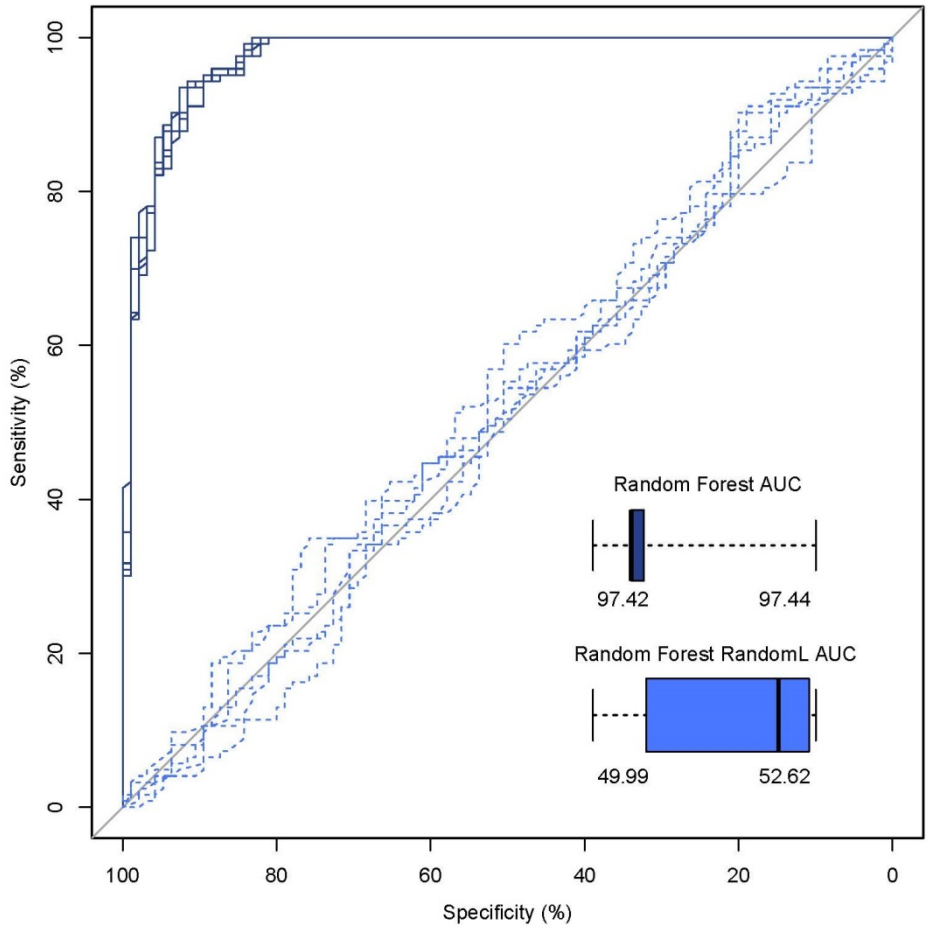
varSelRF – http://cran.r-project.org/web/packages/varSelRF/

# Statistical learning methods

Pathogen classifier (NP vs. HP+BP)

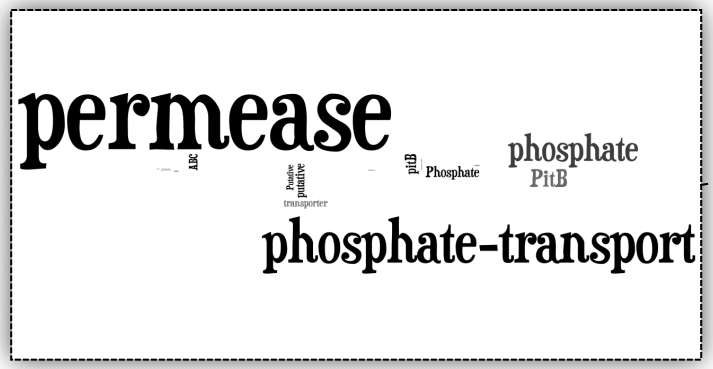


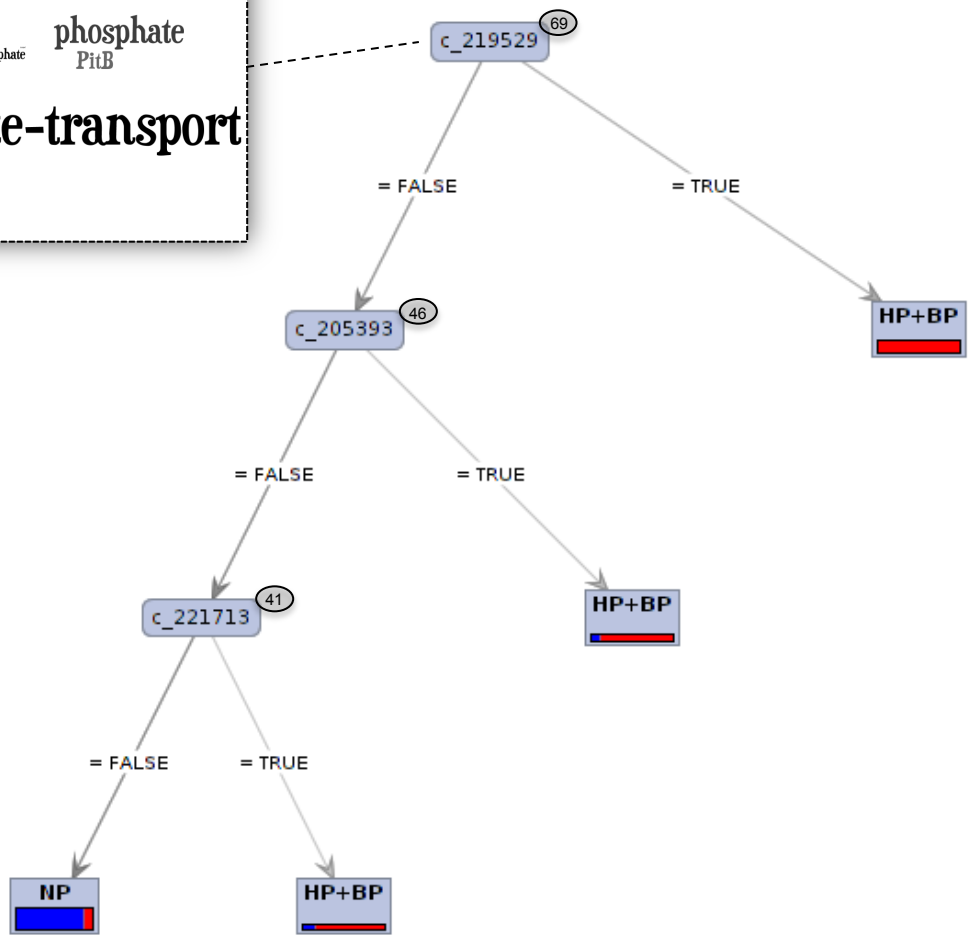

Non-pathogen classifier (NP vs. HP+BP)




Pathogens (HP+BP) vs. Non-Pathogens (NP)

# Statistical learning methods



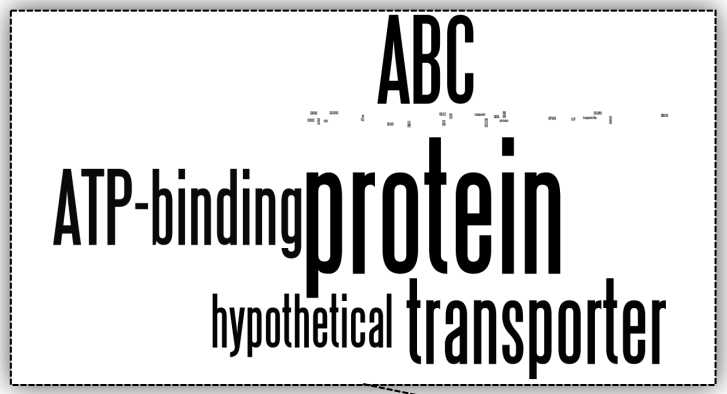



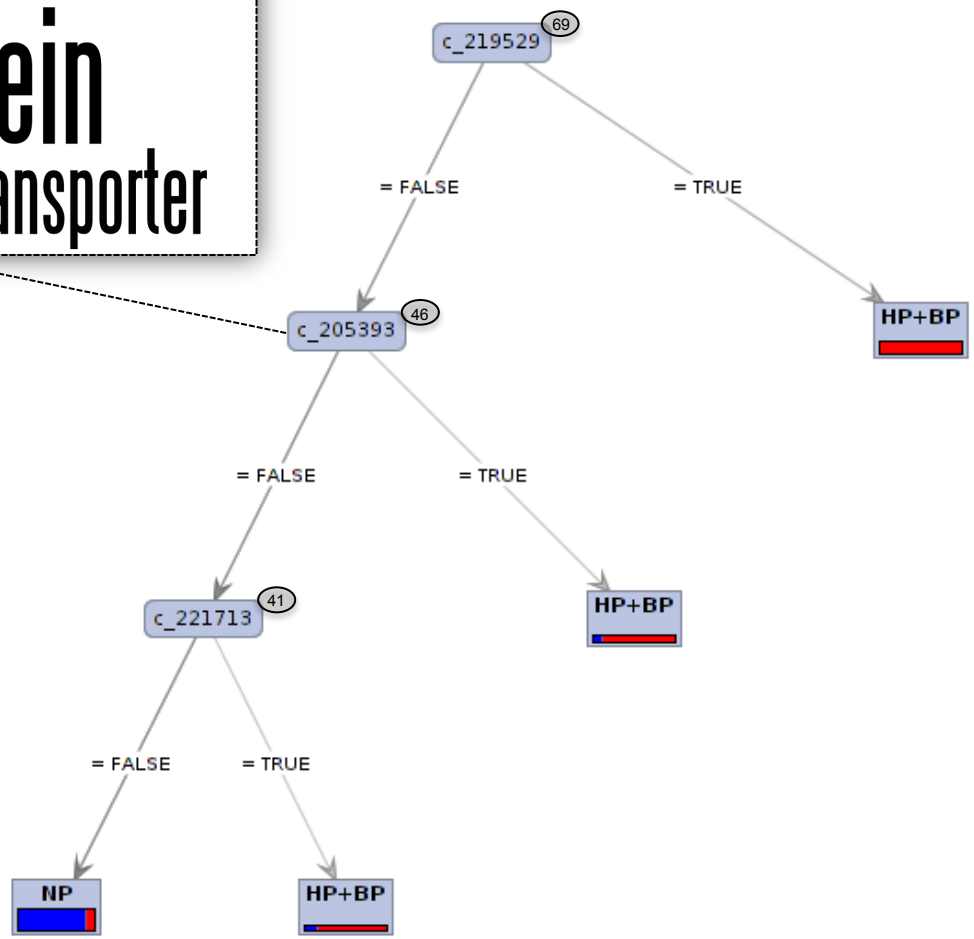

**Pathogens Classifier**

ACC = 93.9%

Pathogens (HP+BP) vs. Non-Pathogens (NP)

# Statistical learning methods



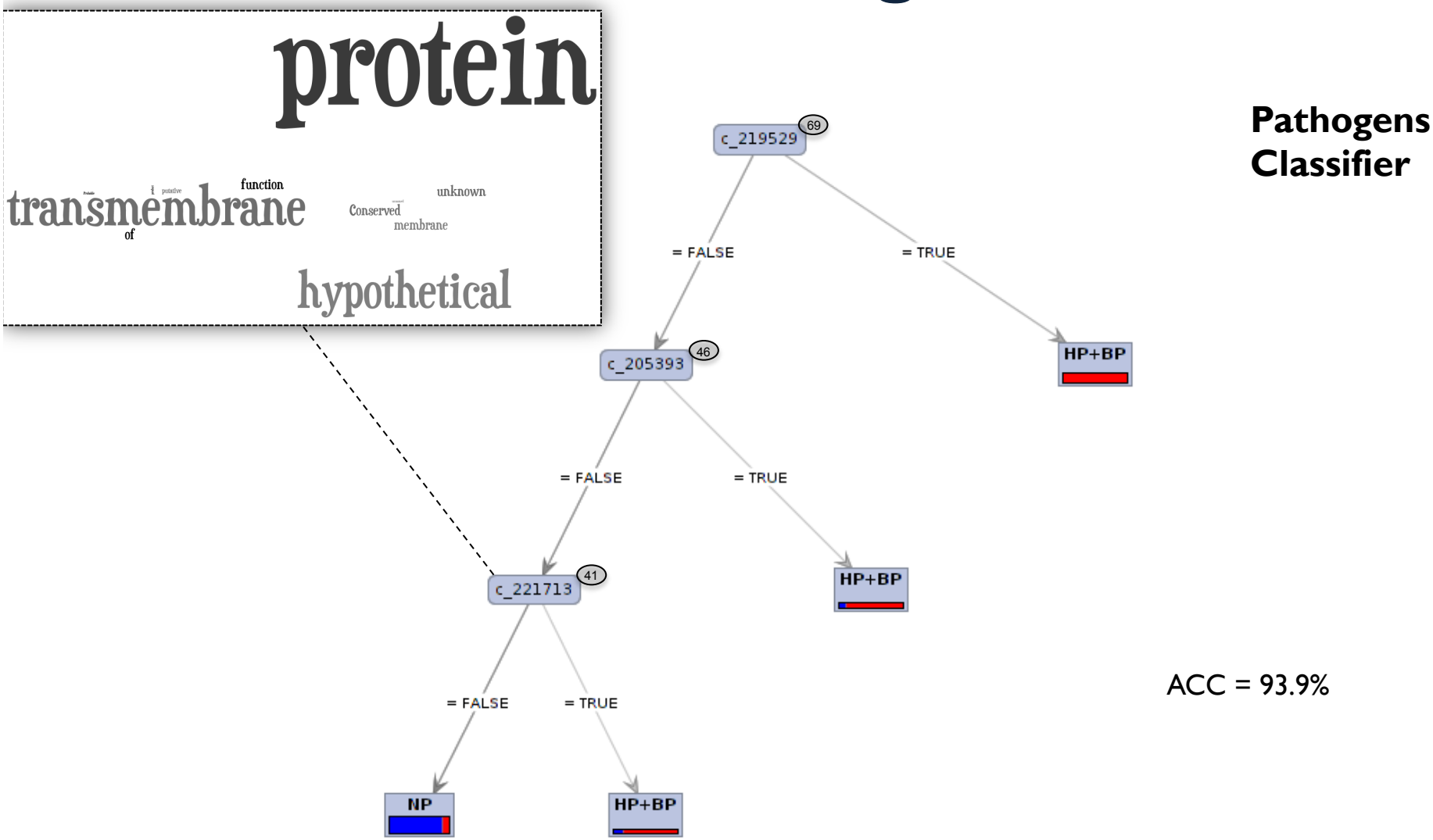




**Pathogens Classifier**

ACC = 93.9%

Pathogens (HP+BP) vs. Non-Pathogens (NP)

# Statistical learning methods

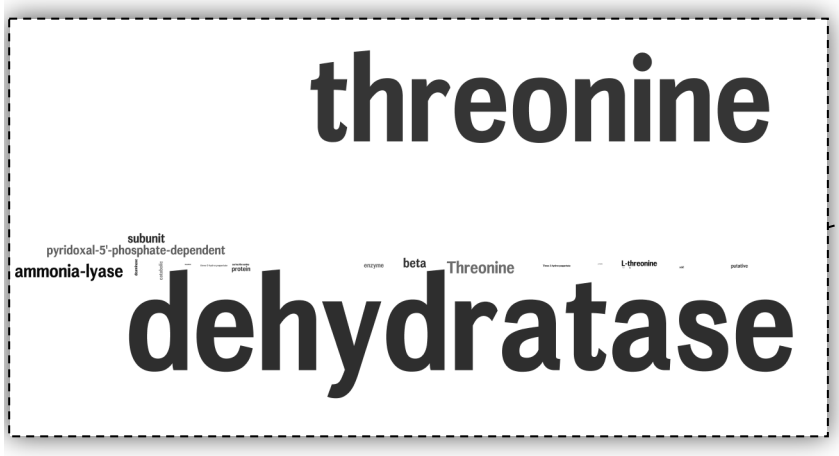



**Pathogens Classifier**

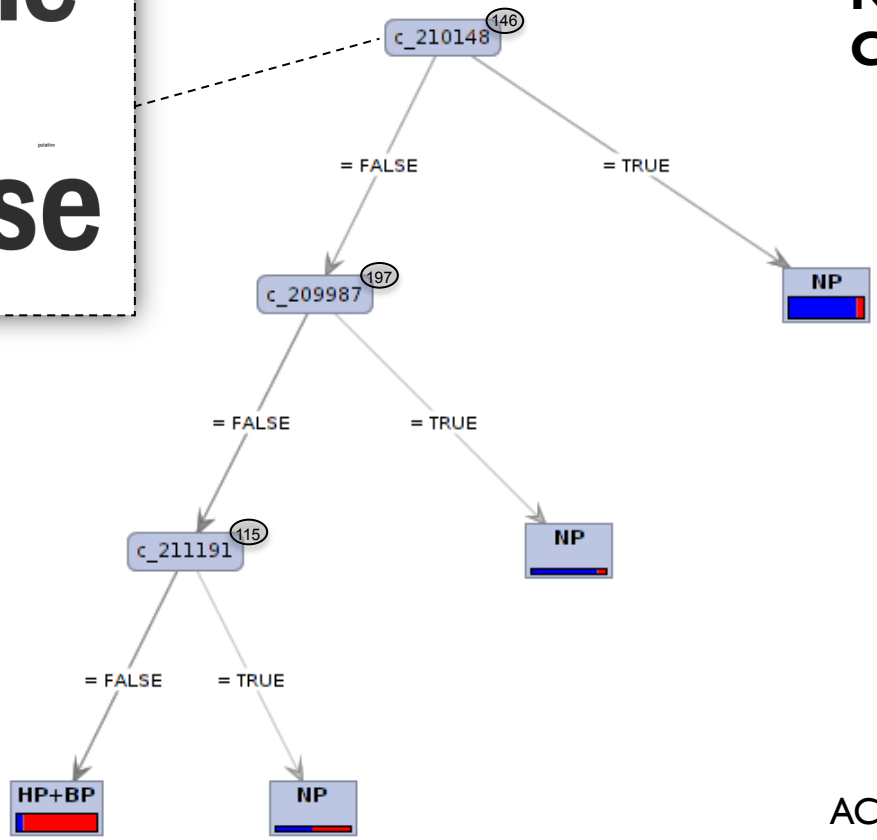
ACC = 93.9%

Pathogens (HP+BP) vs. Non-Pathogens (NP)

# Statistical learning methods



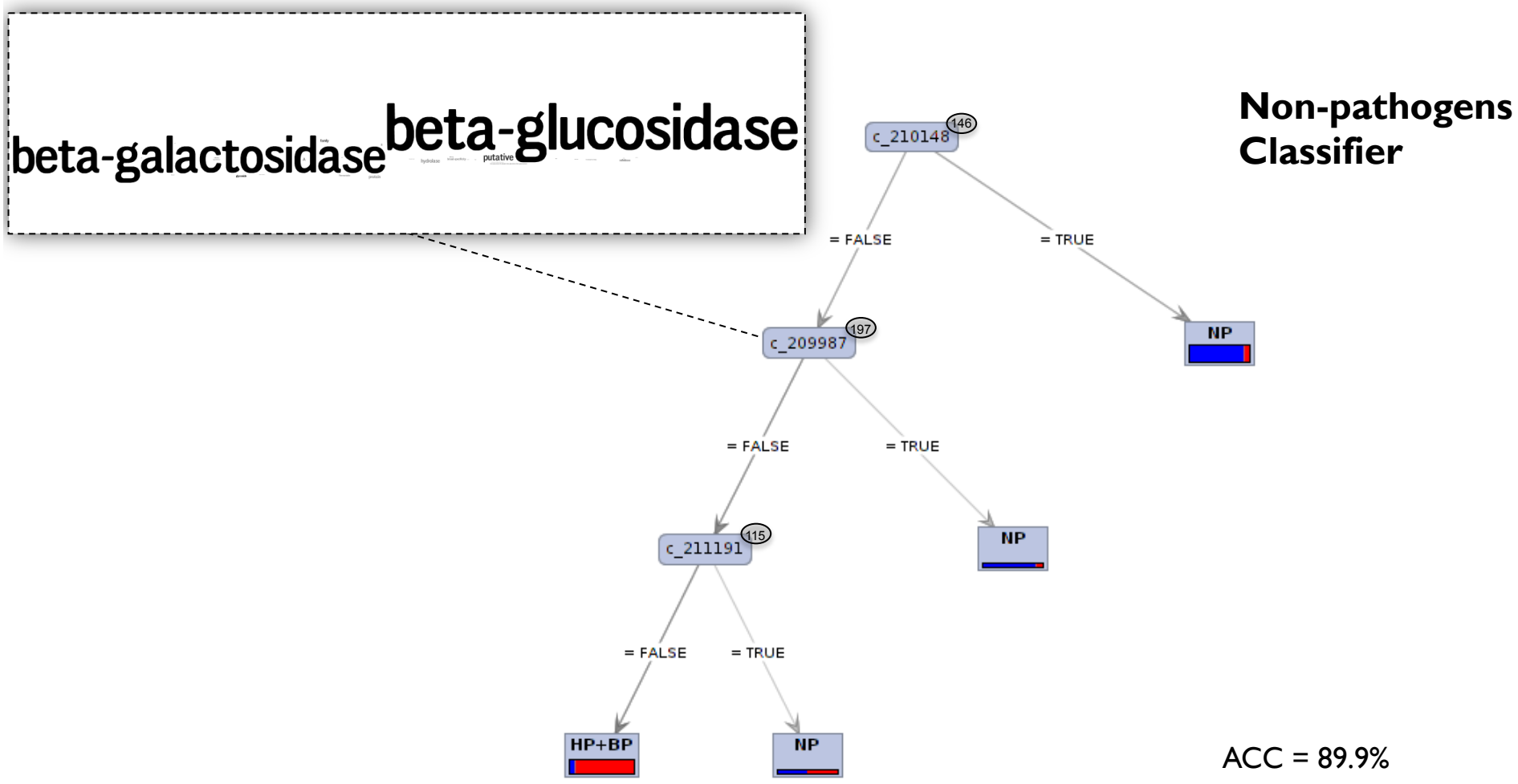

**Non-pathogens Classifier**




ACC = 89.9%

Pathogens (HP+BP) vs. Non-Pathogens (NP)

# Statistical learning methods

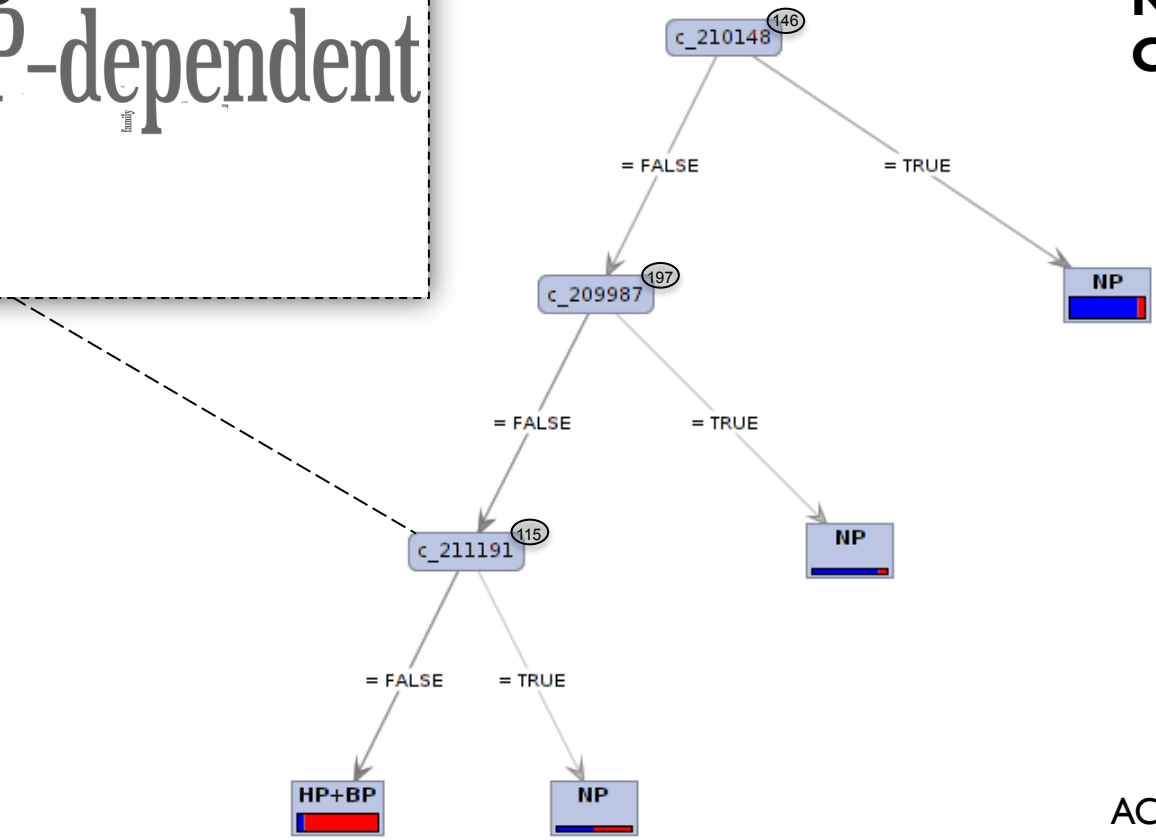



**Non-pathogens Classifier**

ACC = 89.9%

Pathogens (HP+BP) vs. Non-Pathogens (NP)

# Statistical learning methods


**Non-pathogens Classifier**

ACC = 89.9%

Pathogens (HP+BP) vs. Non-Pathogens (NP)