

TREE BASED METHODS

E. Vegas, F. Reverter and A. Sanchez

INTRODUCTION TO DECISION TREES

MOTIVATION

- In many real-world applications, decisions need to be made based on complex, multi-dimensional data.
- One goal of statistical analysis is to provide insights and guidance to support these decisions.
- Decision trees provide a way to organize and summarize information in a way that is easy to understand and use in decision-making.

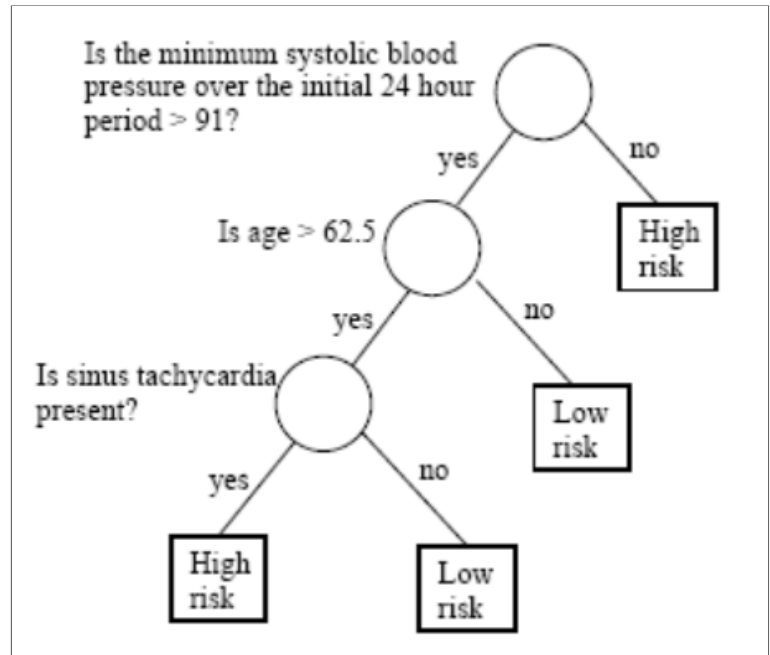
EXAMPLES (1)

- A bank need to have a way to decide if/when a customer can be granted a loan.
- A doctor may need to decide if an oncological patient has to undergo a surgery or a less aggressive treatment.
- A company may need to decide about investing in new technologies or stay with the traditional ones.

In all those cases a decision tree may provide a structured approach to decision-making that is based on data and can be easily explained and justified.

EXAMPLES (2)

- Similarly as would do a doctor, patients are classified into one of two classes: *high risk* versus *low risk* based on: systolic blood pressure, age and tachycardia.



SO, WHAT IS A DECISION TREE?

- A decision tree is a graphical representation of a series of decisions and their potential outcomes.
- It is obtained by recursively *stratifying* or *segmenting* the *feature space* into a number of simple regions.
- Each region (decision) corresponds to a *node* in the tree, and each potential outcome corresponds to a *branch*.
- The tree structure can be used to guide decision-making based on data.

WHAT DO WE NEED TO LEARN?

- We need **context** and also, we need to learn how to **build, evaluate** and **optimize** trees.
- **Context**
 - When is it appropriate to rely on decision trees?
 - When would other approaches be preferable?
 - What type of decision trees can be used?
- **Build the tree**
 - How do we construct the tree?
 - How do we optimize the tree?
 - How do we evaluate it?

MORE ABOUT CONTEXT

- Decision trees are non parametric, data guided predictors, well suited in many situations such as:
 - Non-linear relationships.
 - High-dimensional data.
 - Interaction effects.
 - Non-parametric data.
 - Mixed data types.
- They are not so appropriate for complex datasets, or complex problems, that require expert knowledge.

TYPES OF DECISION TREES

- Two main types: Classification and Regression Trees (CART).
 - **Classification Trees** are built when the response variable is categorical.
 - They aim to *classify a new observation* based on the values of the predictor variables.
 - **Regression Trees** are used when the response variable is numerical.
 - They aim to *predict the value* of a continuous response variable based on the values of the predictor variables.

CLASSIFICATION VS REGRESSION TREES

Aspect	Regression Trees	Classification Trees
Outcome var. type	Continuous	Categorical
Goal	To predict a numerical value	To predict a class label
Splitting criteria	Mean Squared Error, Mean Abs. Error, etc.	Gini Impurity, Entropy, etc.
Leaf node prediction	Mean or median of the target variable in that region	Mode or majority class of the target variable ...
Tree visualization	Sequence of splits leading to a numerical predic...	Sequence of splits leading to a categorical pr...
Examples of use cases	Predicting housing prices, predicting stock prices	Predicting customer churn, predicting the like...
Evaluation metric	Mean Squared Error, Mean Absolute Error, R-square...	Accuracy, Precision, Recall, F1-score, etc.
Overfitting	Can suffer from overfitting if the tree is too de...	Can suffer from overfitting if the tree is too ...
Pruning	Can be pruned to reduce overfitting	Can be pruned to reduce overfitting
Ensemble methods	Random Forest, Gradient Boosting, etc.	Random Forest, Gradient Boosting, etc.

BUILDING THE TREES

- Building the tree requires deciding:
 - how to partition (“split”) the space,
 - Which *impurity* measures to use,
 - When to stop splitting
- Evaluation is similar to other classifiers.
- Optimization involves deciding:
 - How to *prune* the tree,
 - Which features are most important.

TREE BUILDING WITH R

Package	Algorithm	Dataset size	Missing data handling	Ensemble methods	Visual repr	User interface
rpart	RPART	Medium to large	Poor	No	Yes	Simple
caret	Various	Various	Depends on algorithm	Yes	Depends on algorithm	Complex
tree	CART	Small to medium	Poor	No	Yes	Simple

A “QUICK AND DIRTY” EXAMPLE IN R

- The Pima Indian Diabetes data set contains 768 individuals (female) and 9 clinical variables.

```
Rows: 768
Columns: 9
$ pregnant <dbl> 6, 1, 8, 1, 0, 5, 3, 10, 2, 8, 4, 10, 10, 1, 5, 7, 0, 7,
1, 1...
$ glucose <dbl> 148, 85, 183, 89, 137, 116, 78, 115, 197, 125, 110, 168,
139,...
$ pressure <dbl> 72, 66, 64, 66, 40, 74, 50, NA, 70, 96, 92, 74, 80, 60,
72, N...
$ triceps <dbl> 35, 29, NA, 23, 35, NA, 32, NA, 45, NA, NA, NA, NA, 23,
19, N...
$ insulin <dbl> NA, NA, NA, 94, 168, NA, 88, NA, 543, NA, NA, NA, NA,
846, 17...
$ mass <dbl> 33.6, 26.6, 23.3, 28.1, 43.1, 25.6, 31.0, 35.3, 30.5, NA,
37....
$ pedigree <dbl> 0.627, 0.351, 0.672, 0.167, 2.288, 0.201, 0.248, 0.134,
0.158...
```

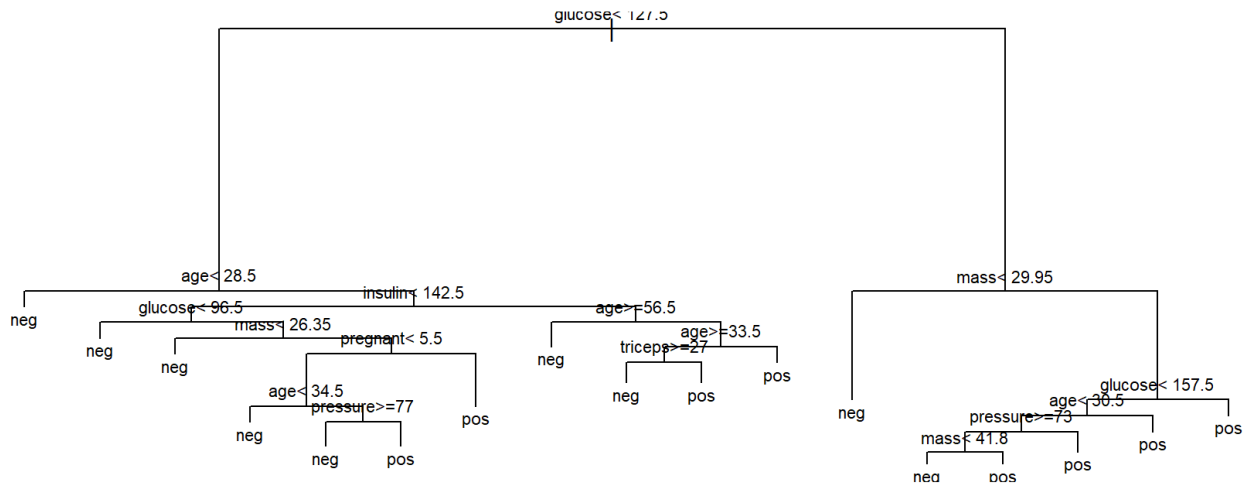
PREDICTING DIABETES ONSET

- We wish to predict the probability of individuals in being diabetes-positive or negative.

```
1 library(rpart)
2 model1 <- rpart(diabetes ~., data = PimaIndiansDiabetes2)
```

```
1 plot(model1)
2 text(model1, digits = 3, cex=0.7)
```

A SIMPLE, UNPRUNNED TREE



HOW ACCURATE IS THE MODEL?

```
1 predicted.classes<- predict(model1, PimaIndiansDiabetes2, "class")  
2 mean(predicted.classes == PimaIndiansDiabetes2$diabetes)
```

```
[1] 0.8294271
```


ALWAYS USE TRAIN/TEST SETS!

- A better strategy is to use train dataset to build the model and a test dataset to check how it works.

```
1 set.seed(123)
2 ssize <- nrow(PimaIndiansDiabetes2)
3 propTrain <- 0.8
4 training.indices <- sample(1:ssize, floor(ssize*propTrain))
5 train.data <- PimaIndiansDiabetes2[training.indices, ]
6 test.data <- PimaIndiansDiabetes2[-training.indices, ]
```

BUILD ON TRAIN, ESTIMATE ON TEST

- First, build the model on the train data

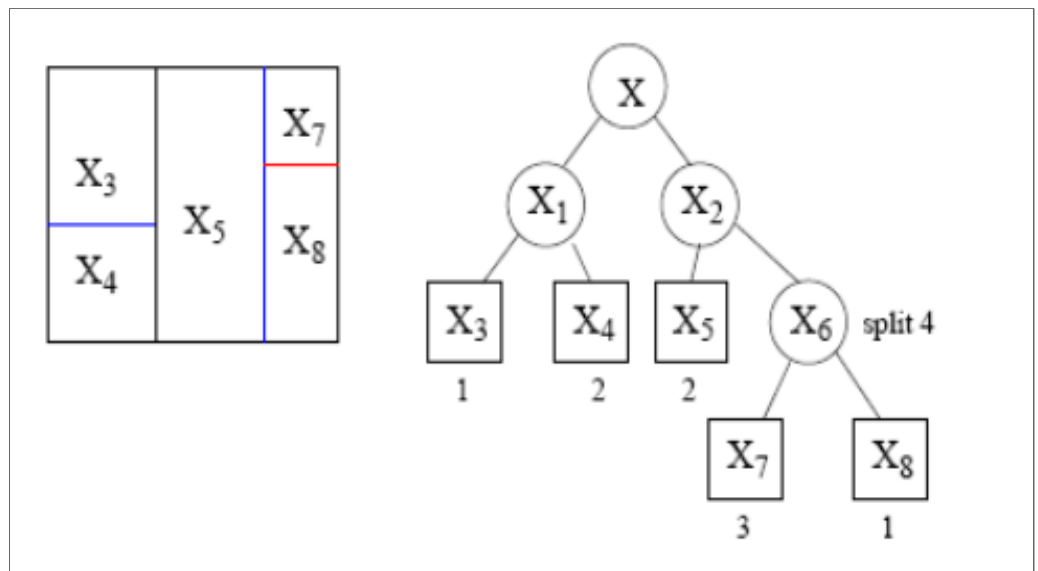
```
1 model2 <- rpart(diabetes ~., data = train.data)
```

- Then check its accuracy on the test data.

```
1 predicted.classes.test<- predict(model2, test.data, "class")
2 mean(predicted.classes.test == test.data$diabetes)
```

```
[1] 0.7272727
```

CONSTRUCTING TREES



NOTATION AND BASIC CONCEPTS

- \mathbb{X} : Space of variables, or *feature space*
 - Usually $\mathbb{X} \subseteq \mathbb{R}^p$
 - But it can contain numerical/categorical variables.
- $X \in \mathbb{X}$: Input vector: X_1, X_2, \dots, X_p .
- Tree-structured classifiers are constructed by repeated splits of the space X into smaller and smaller subsets, beginning with X itself.
 - That is by *recursive splitting*

ADDITIONAL NOTATION

- A node is denoted by t . We will also denote the left child node by t_L and the right one by t_R .
- Denote the collection of all the nodes in the tree by T and the collection of all the leaf nodes by \tilde{T}
- A split will be denoted by s . The set of splits is denoted by S .
- **Nice definition of basic terminology:**

TREES PARTITION THE SPACE

- *The tree represents the recursive splitting of the space.*
 - Every node of interest corresponds to one region in the original space.
 - Two child nodes occupy two different regions.
 - Together, yield same region as that of the parent node.
- In the end, every leaf node is assigned with a class and a test point is assigned with the class of the leaf node it lands in.

THE TREE REPRESENTS THE SPLITTING

- **Animation**

CONSTRUCTION OF A TREE

It involves the following three elements:

1. The selection of the splits, i.e., how do we decide which node (region) to split and how to split it?
 - How to select from the pool of candidate splits?
 - What are appropriate *goodness of split* criteria?
2. If we know how to make splits ('grow' the tree), how do we decide when to declare a node terminal and *stop splitting*?
3. How do we assign each terminal node to a class?

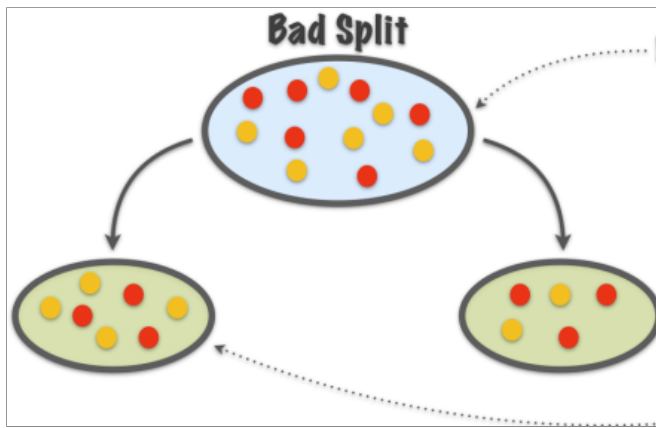
WHICH QUESTIONS TO SPLIT THE TREE

- To build a Tree, questions have to be generated that induce splits based on the value of a single variable.
- Ordered variable X_j :
 - Is $X_j \leq c$? for all possible thresholds c .
 - Restricts the split line to be parallel to the coordinates.
- Categorical variables, $X_j \in \{1, 2, \dots, M\}$:
 - Is $X_j \in A$? where A is any subset of $\{1, 2, \dots, M\}$.
- The pool of candidate splits for all p variables is formed by combining all the generated questions.

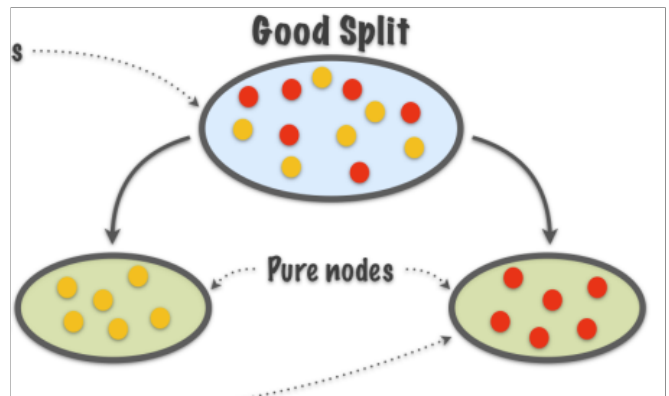
DETERMINING GOODNESS OF SPLIT

- The way we choose the split, is *to measure every split by a 'goodness of split' measure*, which depends on:
 - the split question as well as
 - the node to split.
- The 'goodness of split' in turn is measured by an *impurity function*.
- Intuitively, when we split the points we want the region corresponding to each leaf node to be "pure", that is, most points in this region come from the same class, that is, one class dominates.

GOOD SPLITS VS BAD SPLITS



Purity not increased



Purity increased

MEASURING HOMOGENEITY

- In order to measure homogeneity of splits we introduce
 - Impurity functions
 - Impurity measures
- These are used to measure the extent of purity for a region containing data points from possibly different classes.

IMPURITY FUNCTIONS

Definition: An **impurity function** is a function Φ defined on the set of all K -tuples of numbers (p_1, \dots, p_K) satisfying $p_j \geq 0$, $j = 1, \dots, K$, $\sum_j p_j = 1$ with the properties:

1. Φ achieves maximum only for the uniform distribution, that is all the p_j are equal.
2. Φ achieves minimum only at the points $(1, 0, \dots, 0), (0, 1, 0, \dots, 0), \dots, (0, 0, \dots, 0, 1)$, i.e., when the probability of being in a certain class is 1 and 0 for all the other classes.
3. Φ is a symmetric function of p_1, \dots, p_K , i.e., if we permute p_j , Φ remains constant.

SOME IMPURITY FUNCTIONS

Commonly used impurity functions:

1. **Entropy function:** $\sum_{j=1}^K p_j \log \frac{1}{p_j} = - \sum_{j=1}^K p_j \log p_j.$

If $p_j = 0$, use the limit $\lim p_j \rightarrow \log p_j = 0$.

2. **Misclassification rate:** $1 - \max_j p_j.$

3. **Gini index:** $\sum_{j=1}^K p_j (1 - p_j) = 1 - \sum_{j=1}^K p_j^2.$

IMPURITY MEASURES FOR A SPLIT

- Definition: Given an impurity function Φ , define the impurity measure, denoted as $i(t)$, of a node t as:

$$i(t) = \phi(p(1 | t), p(2 | t), \dots, p(K | t))$$

where $p(j | t)$ is the estimated posterior probability of class j given a point is in node t .

GOODNESS OF A SPLIT

- Once we have $i(t)$, we define the goodness of split s for node t , denoted by $\Phi(s, t)$:

$$\Phi(s, t) = \Delta i(s, t) = i(t) - p_R i(t_R) - p_L i(t_L)$$

- The best split for the single variable X_j is the one that has the largest value of $\Phi(s, t)$ over all $s \in \mathcal{S}_j$, the set of possible distinct splits for X_j .

ENTROPY BASED GOODNESS OF SPLIT

- Goodness of split measure based on Entropy impurity function.

$$H(X) = - \sum_{i=1}^n P(x_i) \log_2 P(x_i)$$

- Information theoretic approach that compares
 - the entropy of the parent node before the split to
 - that of a weighted sum of the child nodes after the split where the weights are proportional to the number of observations in each node.

INFORMATION GAIN

- For a feature F and a set of observations X , information gain is defined as:

$$IG(X, F) = (\text{original entropy}) - (\text{entropy after the split})$$

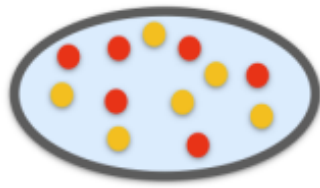
$$IG(X, F) = H(X) - \sum_{i=1}^n \frac{|x_i|}{X} H(x_i)$$

EXAMPLE

Consider the problem of designing an algorithm to automatically differentiate between apples and pears (class labels) given only their width and height measurements (features).

Width	Height	Fruit
7.1	7.3	Apple
7.9	7.5	Apple
7.4	7.0	Apple
8.2	7.3	Apple
7.6	6.9	Apple
7.8	8.0	Apple
7.0	7.5	Pear
7.1	7.9	Pear
6.8	8.0	Pear
6.6	7.7	Pear
7.3	8.2	Pear
7.2	7.9	Pear

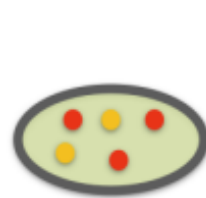
EXAMPLE. ENTROPY CALCULATION



$$\begin{aligned}
 H(X) &= \sum_{i=1}^n P(x_i) \log_2 P(x_i) \\
 &= -\frac{6}{12} \cdot \log_2\left(\frac{6}{12}\right) - \frac{6}{12} \cdot \log_2\left(\frac{6}{12}\right) \\
 &= \frac{1}{2} + \frac{1}{2} = 1
 \end{aligned}$$



$$\begin{aligned}
 H(X) &= -\frac{4}{7} \cdot \log_2\left(\frac{4}{7}\right) - \frac{3}{7} \cdot \log_2\left(\frac{3}{7}\right) \\
 &= 0.4613 + 0.5239 = 0.9852
 \end{aligned}$$



$$\begin{aligned}
 H(X) &= -\frac{3}{5} \cdot \log_2\left(\frac{3}{5}\right) - \frac{2}{5} \cdot \log_2\left(\frac{2}{5}\right) \\
 &= 0.4422 + 0.5288 = 0.9710
 \end{aligned}$$

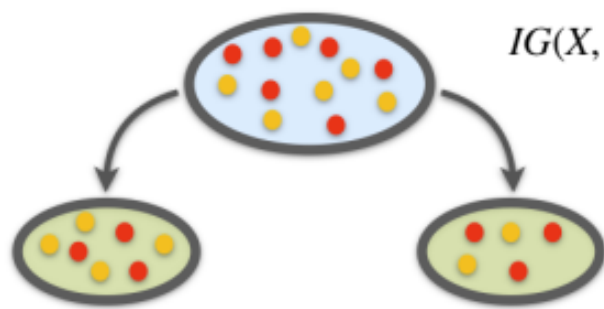


$$H(X) = -\frac{6}{6} \cdot \log_2\left(\frac{6}{6}\right) = 0$$



$$H(X) = -\frac{6}{6} \cdot \log_2\left(\frac{6}{6}\right) = 0$$

EXAMPLE. INFORMATION GAIN

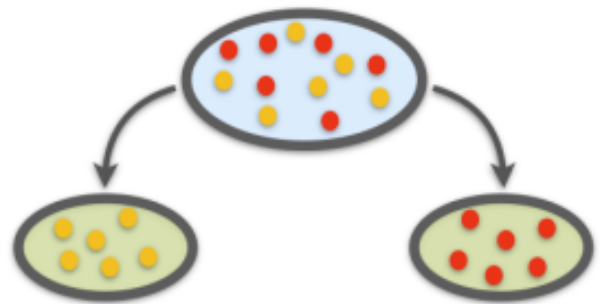


$$IG(X, F) = (\text{original entropy}) - (\text{entropy after split})$$

$$= H(X) - \sum_{i=1}^n \frac{|x_i|}{X} H(x_i)$$

$$= 1 - \frac{7}{12} \cdot (0.9852) - \frac{5}{12} \cdot (0.9710) = 0.0213$$

$$IG(X, F) = 1 - \frac{6}{12} \cdot (0) - \frac{6}{12} \cdot (0) = 1$$



WHEN TO STOP GROWING

- Maximizing information gain is one possible criteria to choose among splits.
- In order to avoid excessive complexity it is usually decided to stop splitting when “information gain does not compensate for increase in complexity”.
- In practice stop splitting if:

$$\max_{s \in S} \Delta I(s, t) < \beta$$

CLASS ASSIGNMENT RULES (1)

- The decision tree classifies new data points as follows.
 - We let a data point pass down the tree and see which leaf node it lands in.
 - The class of the leaf node is assigned to the new data point. Basically, all the points that land in the same leaf node will be given the same class. This is similar to k-means or any prototype method.

CLASS ASSIGNMENT RULES (2)

- A class assignment rule assigns a class $j = 1, \dots, K$ to every terminal (leaf) node $t \in \tilde{T}$.
- The class assigned to node t . \tilde{T} is denoted by $\kappa(t)$,
- E.g., if $\kappa(t) = 2$, all the points in node t would be assigned to class 2.

CLASS ASSIGNMENT RULES (3)

- If we use 0-1 loss, the class assignment rule is very similar to k-means (where we pick the majority class or the class with the maximum posterior probability):

$$\kappa(t) = \arg \max_j p(j \mid t)$$

ESTIMATING THE ERROR RATE (1)

- Let's assume we have built a tree and have the classes assigned for the leaf nodes.
- Goal: estimate *the classification error rate* for this tree.
- We need to introduce the resubstitution estimate $r(t)$ for the probability of misclassification, given that a case falls into node t . This is:

$$r(t) = 1 - \max_j p(j \mid t) = 1 - p(\kappa(t) \mid t)$$

ESTIMATING THE ERROR RATE (2)

- Denote $R(t) = r(t)p(t)$, that is the misclassification error rate weighted by the probability of the node.
- The resubstitution estimation for the overall misclassification rate $R(T)$ of the tree classifier T is:

$$R(T) = \sum_{t \in \tilde{T}} R(t)$$

OPTIMIZING THE TREE

- Trees obtained by looking for optimal splits tend to overfit: good for the data in the tree, but generalize badly and tend to fail more in predictions.
- In order to reduce complexity and overfitting while keeping the tree as good as possible tree pruning may be applied.
- Essentially pruning works by removing branches that are unlikely to improve the accuracy of the model on new data.

PRUNING METHODS

- There are different pruning methods, but the most common one is the *cost-complexity* pruning algorithm, also known as the *weakest link pruning*.
- The algorithm works by adding a penalty term to the misclassification rate of the terminal nodes:

$$R_{\alpha}(T) = R(T) + \alpha|T|$$

where α is the parameter that controls the trade-off between tree complexity and accuracy.

COST COMPLEXITY PRUNING

- Start by building a large tree that overfits the data.
- Then, use cross-validation to estimate the optimal value of α that minimizes the generalization error.
- Finally, prune the tree by removing the branches that have a smaller improvement in impurity than the penalty term multiplied by α .
- Iterate the process continues until no more branches can be pruned, or until a minimum tree size is reached.

REGRESSION TREES

- When the response variable is numeric, decision trees are *regression trees*.
- Some differences with classification trees:
 - Splitting criteria based on mean squared error or mean absolute error.
 - Leaf node prediction based on mean/median of the target variable.
 - Evaluation metrics based on measures of the difference between the predicted and actual values.

REFERENCES AND RESOURCES

REFERENCES

- **A. Criminisi, J. Shotton and E. Konukoglu (2011) Decision Forests for Classifcation, Regression ... Microsoft Research technical report TR-2011-114**
- Efron, B., Hastie T. (2016) Computer Age Statistical Inference. Cambridge University Press. **Web site**
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction. Springer.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112). Springer. **Web site**

COMPLEMENTARY REFERENCES

- Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). Classification and regression trees. CRC press.
- Brandon M. Greenwell (202) Tree-Based Methods for Statistical Learning in R. 1st Edition. Chapman and Hall/CRC DOI: <https://doi.org/10.1201/9781003089032>
- Genuer R., Poggi, J.M. (2020) Random Forests with R. Springer ed. (UseR!)

RESOURCES

- **Applied Data Mining and Statistical Learning (Penn State-University)**
- **R for statistical learning**
- **CART Model: Decision Tree Essentials**
- **An Introduction to Recursive Partitioning Using the RPART Routines**