Machine Learning and Artificial Intelligence *through* R

A concise practice notes for data analysts

Dr. Kamakshaiah Musunuru

Preface

This book is an outcome of my preparation of class notes for the course "Machine Learning" for one of the classes I was dealing at certain University in India. For long I am craving to indulge in machine learning especially through R platform. Why R? There are two reasons; one, R is world's best platform or environment for quantitative techniques. Second, I have fair level of skills using R for quantitative analysis.

The Python community seems to be so strong in this area of operations, i.e., machine learning, for a simple reason that machine learning requires quit a bit of coding skills and also computationally intense. It is rare to find someone from arts background writing code related to machine learning. However, this will surely change. Also, the arena of computational statistics changing very fast. Today, interdisciplinary skills are topic for discussion.

Today, ML & AI is so advancing. Let it be a smart mirror in a nearby textile showroom, or smart glasses in a movie like *Jerusalem*, autonomous or pilot less vehicles such as the one in the movie Stealth etc. I was flabbergasted when I watched A.I. shot by Steven Spielberg (2001). It had an invincible influence on my understanding of Artificial Intelligence in specific and Intelligence in general. Anyway, Terminator is all time hit in the movie history. Apart from movies, today, we have sophisticated computer algorithms that are advancing and used for training machines. Russel and Norvig's "Artificial Intelligence - A Modern Approach" is one of my favorite books I always prefer to time pass.

To be precise, I always think AI as a perfect blend of technology and psychology. The word intelligence is so charismatic and also bewitching to me. Intelligence is what makes human unique. The human species on earth always running ahead of other species due to its intelligence. The human race can make huge strides in science and humanity through their little tiny brain.

This book deals with certain examples related to few methods of ML & AI in R. As I said I like R due to the fact that it is all time companion to me in academics. R is *lingua franca* of statistics. Sorry, not only for statistics but for entire edifice of numerical analysis and quantitative techniques. R is a programming language and that makes it so special, compared to other tools, and suitable for programming advanced mathematics.

I am open source software evangelist and also a freelance software developer. All my code is not only free but open source. Visit my Github portal at `https://github.com/Kamakshaiah` and find the application which is relevant to you.

Wish you happy reading ...

Author
Dr. Kamakshaiah Musunuru

Contents

Chapter 1

Introduction

1.1 Machine Learning

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed. Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data - such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions, through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or infeasible; example applications include email filtering, detection of network intruders or malicious insiders working towards a data breach, optical character recognition (OCR), learning to rank, and computer vision.

Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization,

which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning. Machine learning can also be unsupervised and be used to learn and establish baseline behavioral profiles for various entities and then used to find meaningful anomalies.

Within the field of data analytics, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytics. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.

1.1.1 Types of problems and tasks

Machine learning tasks are typically classified into three broad categories, depending on the nature of the learning "signal" or "feedback" available to a learning system. These are

1. *Supervised learning*: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.

2. *Unsupervised learning*: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

3. *Reinforcement learning*: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). The program is provided feedback in terms of rewards and punishments as it navigates its problem space.

Between supervised and unsupervised learning is *semi-supervised learning*, where the teacher gives an incomplete training signal: a training set with some (often many) of the target outputs missing. Transduction is a special case of this principle where the entire set of problem

instances is known at learning time, except that part of the targets are missing.

Among other categories of machine learning problems, learning to learn learns its own inductive bias based on previous experience. Developmental learning, elaborated for robot learning, generates its own sequences (also called curriculum) of learning situations to cumulatively acquire repertoires of novel skills through autonomous self-exploration and social interaction with human teachers and using guidance mechanisms such as active learning, maturation, motor synergies, and imitation.

Tasks can be categorized into deep learning (the application of artificial neural networks to learning tasks that contain more than one hidden layer) and shallow learning (tasks with a single hidden layer).

Another categorization of machine learning tasks arises when one considers the desired output of a machine-learned system.

1. In classification, inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are ”spam” and ”not spam”.

2. In regression, also a supervised problem, the outputs are continuous rather than discrete.

3. In clustering, a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.

4. Density estimation finds the distribution of inputs in some space.

5. Dimensionality reduction simplifies inputs by mapping them into a lower-dimensional space. Topic modeling is a related problem, where a program is given a list of human language documents and is tasked to find out which documents cover similar topics.

1.1.2 Approaches

A core objective of a learner is to generalize from its experience. Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution (considered representative of the space of occurrences) and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases.

The computational analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory. Because training sets are finite and the future is uncertain, learning theory usually does not yield guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common. The bias-variance decomposition is one way to quantify generalization error.

For the best performance in the context of generalization, the complexity of the hypothesis should match the complexity of the function underlying the data. If the hypothesis is less complex than the function, then the model has underfit the data. If the complexity of the model is increased in response, then the training error decreases. But if the hypothesis is too complex, then the model is subject to overfitting and generalization will be poorer.

In addition to performance bounds, computational learning theorists study the time complexity and feasibility of learning. In computational learning theory, a computation is considered feasible if it can be done in polynomial time. There are two kinds of time complexity results. Positive results show that a certain class of functions can be learned in polynomial time. Negative results show that certain classes cannot be learned in polynomial time.

1. Decision tree learning
2. Association rule learning
3. Artificial neural networks
4. Deep learning

5. Inductive logic programming
6. Support vector machines
7. Clustering
8. Bayesian networks
9. Reinforcement learning
10. Representation learning
11. Similarity and metric learning
12. Sparse dictionary learning
13. Genetic algorithms
14. Rule-based machine learning

1.2 Artificial Intellegence

Artificial intelligence (AI, also machine intelligence, MI) is apparently intelligent behaviour by machines, rather than the natural intelligence (NI) of humans and other animals. In computer science AI research is defined as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of success at some goal. Colloquially, the term "artificial intelligence" is applied when a machine mimics "cognitive" functions that humans associate with other human minds, such as "learning" and "problem solving".

The scope of AI is disputed: as machines become increasingly capable, tasks considered as requiring "intelligence" are often removed from the definition, a phenomenon known as the AI effect, leading to the quip "AI is whatever hasn't been done yet." For instance, optical character recognition is frequently excluded from "artificial intelligence", having become a routine technology. Capabilities generally classified as AI as of 2017 include successfully understanding human speech, competing at a high level in strategic game systems (such as chess and Go), autonomous cars, intelligent routing in content delivery networks, military simulations, and interpreting complex data.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. For most of its history, AI research has been divided into subfields that often fail to communicate with each other. However, in the early 21st century statistical approaches to machine learning became successful enough to eclipse all other tools, approaches, problems and schools of thought.

The traditional problems (or goals) of AI research include reasoning, knowledge, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence is among the field's long-term goals. Approaches include statistical methods, computational intelligence, and traditional symbolic AI. Many tools are used in AI, including versions of search and mathematical optimization, neural networks and methods based on statistics, probability and economics. The AI field draws upon computer science, mathematics, psychology, linguistics, philosophy, neuroscience, artificial psychology and many others.

The field was founded on the claim that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the nature of the mind and the ethics of creating artificial beings endowed with human-like intelligence, issues which have been explored by myth, fiction and philosophy since antiquity. Some people also consider AI a danger to humanity if it progresses unabatedly.

Chapter 2

About R

R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R. [1]

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, $\cdots$) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS.

2.1 The R environment

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

The term "environment" is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.

R, like S, is designed around a true computer language, and it allows users to add additional functionality by defining new functions. Much of the system is itself written in the R dialect of S, which makes it easy for users to follow the algorithmic choices made. For computationally-intensive tasks, C, C++ and Fortran code can be linked and called at run time. Advanced users can write C code to manipulate R objects directly.

Many users think of R as a statistics system. We prefer to think of it of an environment within which statistical techniques are implemented. R can be extended (easily) via packages. There are about eight packages supplied with the R distribution and many more are available through the CRAN family of Internet sites covering a very wide range of modern statistics.

R has its own LaTeX-like documentation format, which is used to supply comprehensive documentation, both on-line in a number of

formats and in hardcopy.

2.2 Basics

2.2.1 CRAN - Packages

2.2.2 Data types

Vectors

Factors

Matrices

Lists

2.2.3 User Defined Functions

Chapter 3

Generalized Linear Models

3.1 Ridge Regression

3.2 Lasso

3.3 Robust Regression

3.4 Polynomial Regression

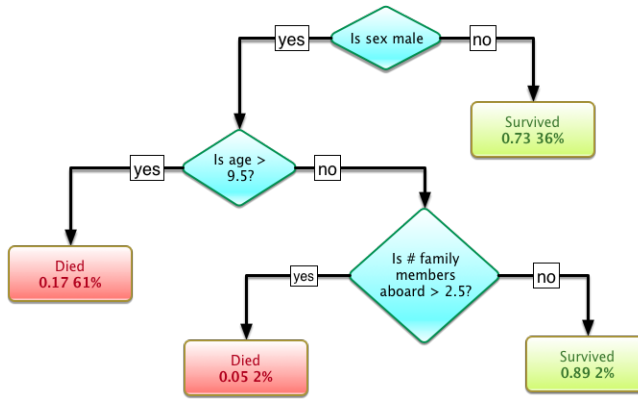
Chapter 4

Supervised learning

4.1 Decision Trees

Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modelling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. [2] In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data (but the resulting classification tree can be an input for decision making).

Decision tree learning is a method commonly used in data mining. The goal is to create a model that predicts the value of a target variable based on several input variables. An example is shown in the diagram below. Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the



values of the input variables represented by the path from the root to the leaf.

A decision tree is a simple representation for classifying examples. But for now, assume that all of the input features have finite discrete domains, and there is a single target feature called the "classification". Each element of the domain of the classification is called a class. A decision tree or a classification tree is a tree in which each internal (non-leaf) node is labeled with an input feature. The arcs coming from a node labeled with an input feature are labeled with each of the possible values of the target or output feature or the arc leads to a subordinate decision node on a different input feature. Each leaf of the tree is labeled with a class or a probability distribution over the classes[why?].

A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called *recursive partitioning*. See the examples illustrated in the figure for spaces that have and have not been partitioned using recursive partitioning, or recursive binary splitting. The recursion is completed when the subset at a node has all the same value of the target variable, or when splitting no longer adds value to the predictions. This process of *top-down induction of decision trees* (TDIDT) is an example of a greedy algorithm, and it is by far the most common strategy for learning decision trees from data.

In data mining, decision trees can be described also as the combination of mathematical and computational techniques to aid the description, categorization and generalization of a given set of data.

Data comes in records of the form:

$$(\mathbf{x}, Y) = (x_1, x_2, x_3, ..., x_k, Y) \tag{4.1}$$

The dependent variable, Y, is the target variable that we are trying to understand, classify or generalize. The vector x is composed of the input variables, x1, x2, x3 etc., that are used for that task.

4.1.1 Types

Decision trees used in data mining are of two main types:

1. ***Classification tree*** analysis is when the predicted outcome is the class to which the data belongs.

2. ***Regression tree analysis*** is when the predicted outcome can be considered a real number (e.g. the price of a house, or a patient's length of stay in a hospital).

The term *Classification And Regression Tree* (CART) analysis is an umbrella term used to refer to both of the above procedures, first introduced by *Breiman et al.* [3] Trees used for regression and trees used for classification have some similarities - but also some differences, such as the procedure used to determine where to split. Some techniques, often called *ensemble methods*, construct more than one decision tree:

4.1.2 Methods

1. ***Boosted trees*** Incrementally building an ensemble by training each new instance to emphasize the training instances previously mis-modeled. A typical example is AdaBoost. These can be used for regression-type and classification-type problems.

2. ***Bootstrap aggregated*** (or bagged) decision trees, an early ensemble method, builds multiple decision trees by repeatedly re-

sampling training data with replacement, and voting the trees for a consensus prediction.

3. ***Rotation forest*** in which every decision tree is trained by first applying principal component analysis (PCA) on a random subset of the input features.

4.1.3 Metrics

This is perhaps most important part of learning methods. Algorithms for constructing decision trees usually work top-down, by choosing a variable at each step that best splits the set of items. Different algorithms use different metrics for measuring "best". These generally measure the homogeneity of the target variable within the subsets. Some examples are given below. These metrics are applied to each candidate subset, and the resulting values are combined (e.g., averaged) to provide a measure of the quality of the split.

Gini impurity

Used by the CART (classification and regression tree) algorithm, Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. Gini impurity can be computed by summing the probability p_i of an item with label i being chosen times the probability $1 - p_i$ of a mistake in categorizing that item. It reaches its minimum (zero) when all cases in the node fall into a single target category.

To compute Gini impurity for a set of items with J classes, suppose $i \in \{1, 2, ..., J\}$ and let p_i be the fraction of items labeled with class i in the set.

$$\mathrm{I}_G(p) = \sum_{i=1}^{J} p_i(1 - p_i) = \sum_{i=1}^{J}(p_i - {p_i}^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} {p_i}^2 = 1 - \sum_{i=1}^{J} {p_i}^2 = \sum_{i \neq k} p_i p_k (4.2)$$

Information Gain

Used by the *ID3, C4.5* and *C5.0* tree-generation algorithms. Information gain is based on the concept of entropy from information theory. [4] [5] [6] Entropy is defined as below

$$H(T) = I_E(p_1, p_2, ..., p_J) = -\sum_{i=1}^{J} p_i \log_2 p_i \tag{4.3}$$

where $p_1, p_2, ...$ are fractions that add up to 1 and represent the percentage of each class present in the child node that results from a split in the tree.

Information Gain = Entropy(parent) - Weighted Sum of Entropy(Children)

IG(T,a)=H(T)-H(T|a) IG(T,a) = H(T) - H(T|a)

Information gain is used to decide which feature to split on at each step in building the tree. Simplicity is best, so we want to keep our tree small. To do so, at each step we should choose the split that results in the purest daughter nodes. A commonly used measure of purity is called information which is measured in bits, not to be confused with the unit of computer memory. For each node of the tree, the information value "represents the expected amount of information that would be needed to specify whether a new instance should be classified yes or no, given that the example reached that node".

4.1.4 Example

Consider an example data set with four attributes: outlook (sunny, overcast, rainy), temperature (hot, mild, cool), humidity (high, normal), and windy (true, false), with a binary (yes or no) target variable, play, and 14 data points. To construct a decision tree on this data, we need to compare the information gain of each of four trees, each split on one of the four features. The split with the highest information gain will be taken as the first split and the process will continue until all children nodes are pure, or until the information gain is 0.

The split using the feature windy results in two children nodes, one for a windy value of true and one for a windy value of false. In this data set, there are six data points with a true windy value, three of which have a play value of yes and three with a play value of no. The eight remaining data points with a windy value of false contain two no's and six yes's. The information of the ***windy=true*** node is calculated using the entropy equation above. Since there is an equal number of yes's and no's in this node, we have

$$I_E([3,3]) = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = \\ -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1$$

For the node where ***windy=false*** there were eight data points, six yes's and two no's. Thus we have

$$I_E([6,2]) = -(6/8)\log_2(6/8) - (2/8)\log_2(2/8) = \\ -(3/4)\log_2(3/4) - (1/4)\log_2(1/4) = 0.8112781$$

To find the ***information of the split***, we take the *weighted average* of these two numbers based on how many observations fell into which node.

$$I_E([3,3],[6,2]) = I_E(windyornot) = (6/14)(1) + (8/14)(0.8112781) = \\ 0.8921589$$

To find the ***information gain of the split using windy***, we must first calculate the information in the data before the split. The original data contained nine yes's and five no's.

$$I_E([9,5]) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940286$$

Now we can calculate the ***information gain achieved by splitting on the windy feature***.

$$IG(windy) = I_E([9,5]) - I_E([3,3],[6,2]) = 0.940286 - 0.8921589 = \\ 0.0481271$$

To build the tree, the information gain of each possible first split would need to be calculated. The best first split is the one that provides the most information gain. This process is repeated for each impure node until the tree is complete.

Variance reduction

Introduced in CART, variance reduction is often employed in cases where the target variable is continuous (regression tree), meaning that use of many other metrics would first require discretization before being applied. The variance reduction of a node N is defined as the total reduction of the variance of the target variable x due to the split at this node:

$$I_V(N) = \frac{1}{|S|^2} \sum_{i \in S} \sum_{j \in S} \frac{1}{2}(x_i - x_j)^2 - \left(\frac{1}{|S_t|^2} \sum_{i \in S_t} \sum_{j \in S_t} \frac{1}{2}(x_i - x_j)^2 + \frac{1}{|S_f|^2} \sum_{i \in S_f} \sum_{j \in S_f} \frac{1}{2}(x_i - x_j)^2 \right)$$

where S, S_t, and S_f are the set of presplit sample indices, set of sample indices for which the split test is true, and set of sample indices for which the split test is false, respectively. Each of the above summands are indeed variance estimates, though, written in a form without directly referring to the mean.

4.1.5 Decision tree advantages

Amongst other data mining methods, decision trees have various advantages:

1. ***Simple to understand and interpret***. People are able to understand decision tree models after a brief explanation. Trees can also be displayed graphically in a way that is easy for non-experts to interpret.

2. ***Able to handle both numerical and categorical data***. Other techniques are usually specialised in analysing datasets that have only one type of variable. (For example, relation rules can be used only with nominal variables while neural networks can be used only with numerical variables or categoricals converted to 0-1 values.)

3. ***Requires little data preparation***. Other techniques often require data normalization. Since trees can handle qualitative predictors, there is no need to create dummy variables.[16]

4. ***Uses a white box model***. If a given situation is observable in a model the explanation for the condition is easily explained by

boolean logic. By contrast, in a black box model, the explanation for the results is typically difficult to understand, for example with an artificial neural network.

5. ***Possible to validate a model using statistical tests***. That makes it possible to account for the reliability of the model.
6. ***Non-statistical approach*** that makes no assumptions of the training data or prediction residuals; e.g., no distributional, independence, or constant variance assumptions
7. *Performs well with large datasets*. Large amounts of data can be analysed using standard computing resources in reasonable time.
8. ***Mirrors human decision making*** more closely than other approaches.[16] This could be useful when modeling human decisions/behavior.
9. ***Robust against co-linearity***, particularly boosting
10. *In built feature selection*. Additional irrelevant feature will be less used so that they can be removed on subsequent runs.

4.1.6 Implementation in R

Enough! we had lot of reading on decision trees. Now it is the time for implementation.

Tree-Based Models: Classification

Recursive partitioning is a fundamental tool in data mining. It helps explore the stucture of a set of data, while developing easy to visualize decision rules for predicting a categorical (classification tree) or continuous (regression tree) outcome. This section briefly describes CART modeling, conditional inference trees, and random forests.

We will be performing decision trees through three packages namely (1) `rpart` for CART, (2) `party` for CIT, (1) `RandomForest` for Random Forests.

CART Modeling via `rpart`

`rpart` is helpful in implementing Classification and Regression Trees (CART) methodology. CART (as described by Brieman, Freidman, Olshen, and Stone) can be generated through the `rpart` package. Detailed information on rpart is available in An Introduction to Recursive Partitioning Using the RPART Routines. The general steps are provided below followed by two examples. [7] [8]

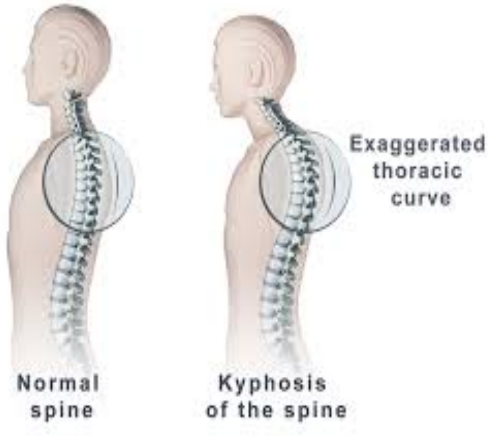
Let us get into implementation. We will be examining two types of decision trees i.e. (1) Classification trees (2) Regression trees through `rpart`. The following are some of the important steps in performing decision trees in R.

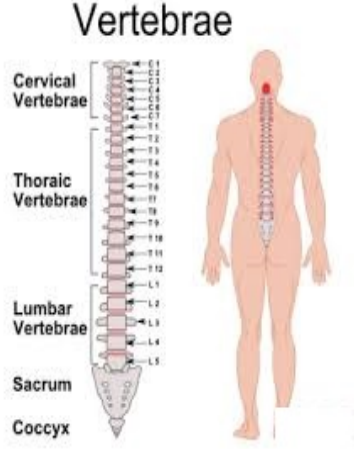
1. Grow the Tree
2. Examine the results
3. prune tree

There is one data set known as `kyphosis`. Read more about the data set from http://www.stat.umn.edu/RegGraph/data/Kyphosis.lsp or for short description you may execute `help("kyphosis")` in R terminal.

Kyphosis is the medical term for "hunchback." It happens when the upper portion of the spine (the thoracic) reason is abnormally curved forward. Some forward curvature in this region is normal but if the curve is greater than 50 degrees, the condition is considered to be "kyphotic" which is abnormal. Kyphosis is thought to affect men and women equally, although this has been debated in some literature. The disorder affects approximately 4 to 8% of the overall population. The most common symptom of kyphosis, and the easiest to recognize is a rounded upper back. The extreme curvature in the upper spine causes a "humpback" appearance or causes the individual to hunch forward while the other being *severe pain in the back, tiredness or fatigue, Stiffness and tenderness of the spine, inter alia.* One of the treatments is surgery if the curve is extreme, measuring 75 degrees or greater on an x-ray, and it is continuing to progress in spite of bracing, surgery will likely be recommended. This rarely occurs in cases of kyphosis. Data were collected on 83 patients undergoing corrective

spinal surgery . The objective was to determine important risk factors for kyphosis following surgery. The risk factors *are age in years, the starting vertebrae level* of the surgery and *the number of levels* involved.






```
> library(rpart)
> data(kyphosis)
> # help("kyphosis")
```

```
> head(kyphosis)

  Kyphosis Age Number Start
1   absent  71      3     5
2   absent 158      3    14
3  present 128      4     5
4   absent   2      5     1
5   absent   1      4    15
6   absent   1      2    16

>
```

There are four variables in the data set namely *kyphosis, Age, Number* and *Start*. You may go to the aforementioned URL and read about the variables of this data set. The very first row shows that the risk factor is *absent* for patient whose *age* is 71 years and surgery was done at *level* 3 for 5 levels (*number of levels*).

```
> # classification thru assimption that the risk
        depends on AGE, NO. OF LEVELS and STARTING LEVEL
> fit <- rpart(Kyphosis ~ Age + Number + Start,
        method = "class", data =kyphosis)
```

Now we might be able to investigate the object that we created `fit`. Two important features of this object are `cptable` and `variable.importance`.

```
> fit$cptable

          CP nsplit rel error xerror      xstd
1 0.17647059      0 1.0000000      1 0.2155872
2 0.01960784      1 0.8235294      1 0.2155872
3 0.01000000      4 0.7647059      1 0.2155872

> fit$variable.importance

   Start      Age   Number
8.198442 3.101801 1.521863
```

While coming to the `rpart()` function; we had supplied an arguments i.e. `method`. This might need certain discussion. `rpart()` perform decision tree analysis through certain methods namely, *class, exp, poisson, anova* and *any user level functions*. If the dependent variable (Y)

is *survival* object rpart by default choses "exp", or if Y has 2 columns then method "poisson" is assumed, or if Y a factor then method "class" is selected for rest of the cases method "anova" is selected. You may find more description to these methods by executing `help("rpart")` at R console.

There are many functions to describe the analysis in R for `rpart`. You may visit https://cran.r-project.org/web/packages/rpart/rpart.pdf for CRAN manual and https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf for vignette. One of the functions which, perhaps, important are `summary()`. We may call `fit` object through this function to know the very gist of the analysis.

```
> summary(fit)

Call:
rpart(formula = Kyphosis ~ Age + Number + Start, data = kyphosis,
    method = "class")
  n= 81
          CP nsplit rel error xerror      xstd
1 0.17647059      0 1.0000000      1 0.2155872
2 0.01960784      1 0.8235294      1 0.2155872
3 0.01000000      4 0.7647059      1 0.2155872
Variable importance
 Start    Age Number
    64     24     12
Node number 1: 81 observations,    complexity param=0.1764706
  predicted class=absent   expected loss=0.2098765  P(node) =1
    class counts:    64    17
   probabilities: 0.790 0.210
  left son=2 (62 obs) right son=3 (19 obs)
  Primary splits:
      Start  < 8.5  to the right, improve=6.762330, (0 missing)
      Number < 5.5  to the left,  improve=2.866795, (0 missing)
      Age    < 39.5 to the left,  improve=2.250212, (0 missing)
  Surrogate splits:
      Number < 6.5  to the left,  agree=0.802, adj=0.158, (0 split)
Node number 2: 62 observations,    complexity param=0.01960784
  predicted class=absent   expected loss=0.09677419  P(node) =0.7654321
    class counts:    56     6
   probabilities: 0.903 0.097
  left son=4 (29 obs) right son=5 (33 obs)
  Primary splits:
      Start  < 14.5 to the right, improve=1.0205280, (0 missing)
      Age    < 55   to the left,  improve=0.6848635, (0 missing)
      Number < 4.5  to the left,  improve=0.2975332, (0 missing)
  Surrogate splits:
      Number < 3.5  to the left,  agree=0.645, adj=0.241, (0 split)
      Age    < 16   to the left,  agree=0.597, adj=0.138, (0 split)
Node number 3: 19 observations
  predicted class=present  expected loss=0.4210526  P(node) =0.2345679
    class counts:     8    11
   probabilities: 0.421 0.579
Node number 4: 29 observations
  predicted class=absent   expected loss=0  P(node) =0.3580247
    class counts:    29     0
   probabilities: 1.000 0.000
```

```
Node number 5: 33 observations,    complexity param=0.01960784
  predicted class=absent   expected loss=0.1818182  P(node) =0.4074074
    class counts:    27     6
   probabilities: 0.818 0.182
  left son=10 (12 obs) right son=11 (21 obs)
  Primary splits:
      Age    < 55   to the left,  improve=1.2467530, (0 missing)
      Start  < 12.5 to the right, improve=0.2887701, (0 missing)
      Number < 3.5  to the right, improve=0.1753247, (0 missing)
  Surrogate splits:
      Start  < 9.5  to the left,  agree=0.758, adj=0.333, (0 split)
      Number < 5.5  to the right, agree=0.697, adj=0.167, (0 split)
Node number 10: 12 observations
  predicted class=absent   expected loss=0  P(node) =0.1481481
    class counts:    12     0
   probabilities: 1.000 0.000
Node number 11: 21 observations,    complexity param=0.01960784
  predicted class=absent   expected loss=0.2857143  P(node) =0.2592593
    class counts:    15     6
   probabilities: 0.714 0.286
  left son=22 (14 obs) right son=23 (7 obs)
  Primary splits:
      Age    < 111  to the right, improve=1.71428600, (0 missing)
      Start  < 12.5 to the right, improve=0.79365080, (0 missing)
      Number < 3.5  to the right, improve=0.07142857, (0 missing)
Node number 22: 14 observations
  predicted class=absent   expected loss=0.1428571  P(node) =0.1728395
    class counts:    12     2
   probabilities: 0.857 0.143
Node number 23: 7 observations
  predicted class=present  expected loss=0.4285714  P(node) =0.08641975
    class counts:     3     4
   probabilities: 0.429 0.571
```

This function gives half-a-ton output for investigation. The very first part of the output is `cptable` this has certain useful measures like `reliability erro` and `error associated with the predictor (x)`. For instance, in out output we got *CP* getting decresed for increasing *number of splits*. At the same time the *reliability error* increasing. This, intuitively, shows that *error inversly related to complexity parameter* for this problem. We can also try manually for different CP values. Suppose we wish to 10 splits and at a fixed complexity parameter of 1 percent.

```
> fit1 <- rpart(Kyphosis ~ Age + Number + Start,
        method = "class", control=rpart.control(minsplit=10, 0.01),
                  data =kyphosis)
> fit1$cptable

          CP nsplit rel error   xerror      xstd
1 0.17647059      0 1.0000000 1.000000 0.2155872
2 0.11764706      1 0.8235294 1.529412 0.2471591
3 0.07843137      2 0.7058824 1.470588 0.2445528
4 0.05882353      5 0.4705882 1.470588 0.2445528
5 0.01000000      8 0.2941176 1.352941 0.2387187
```

Now it might be possible for us to make sense out of new output.

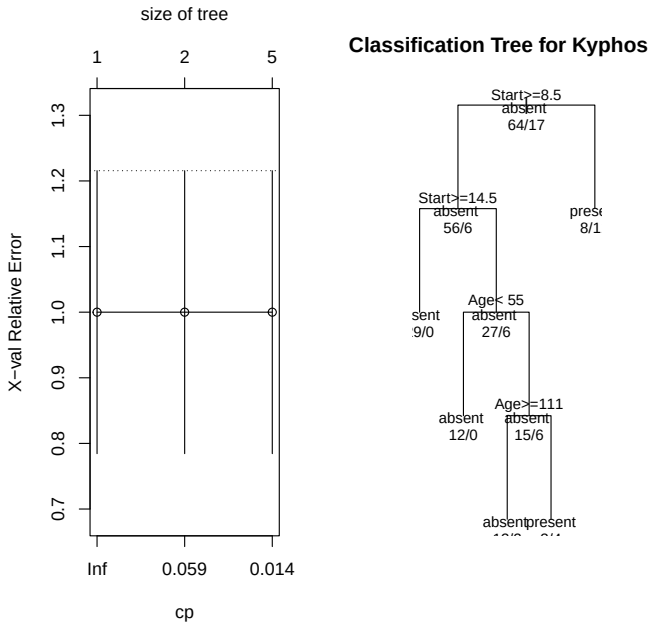
If CP value is 1 percent, then it might be possible reduce *reliability error* to 0.294 for 8 splits. Now coming to the part of "xerror" in the `cptable`; this column deals with overfitting. Lesser is the error lesser the overfitting in the model. We might be able to verify the minimum possible error (xerror) as below:

```
> fit1$cptable[which.min(fit1$cptable[, "xerror"]), "CP"]
```

```
[1] 0.1764706
```

The above statement retrieves the right value for CP to assess the overfitting. This might be able to help to refit the model to that particular level. We might be ablt to discuss this later after dealing with plots. The rest of the information in `summary` is to do with graphing just like *node 0, 1, 2, ...* We have pretty well defined methods for obtaining visuals.

```
> par(mfrow=c(1,2))
> plotcp(fit) # plot 1
> plot(fit, uniform=TRUE, main="Classification Tree for Kyphosis");
        text(fit, use.n=TRUE, all=TRUE, cex=.8) # plot 2
>
> # post(fit, file = "D:/tree.ps",
        title = "Classification Tree for Kyphosis")
```



The last method i.e. `post()` is the method that shows you that it is also possible to save the file in *Post Script* format. You need application like Inkskape to open such files. Now you got to cross check the summary output while observing the plot (decision tree). For the sake of simplicity, the output shows that at node 1 there are 81 observations, CP = 0.176; predicted class = absent; expected loss = 0.209; class counts = 64, 17 (which sums to 81); *left son*, i.e. the node left side to the parent node has 62 observation and *right son* the node right side to the parent node has 19 observations.

Now what make interesting is the *probabilities*, given a question how likely that it turns out to be a person with a start level for vertebrae is greater than 8 and less than 9 but has absense of kyphosis risk condition? the answer could be 79 percent. The answer for the same question for start value 14.5 is 90 percent likely and 9 percent unlikely. But, what makes rather more interesting is node 3: this node throws certain interesting outcomes. It is more likely (58 percent) that a

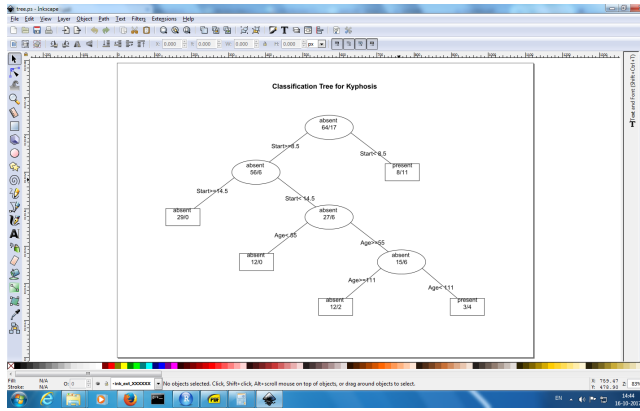


Figure 4.1: Decision tree in Inkskape

person develops *kyphosis* risk condition when the operation is done at vertebrae level greater than 8 but less than 9.

Cross Validation Error

The purpose of cross validation is not to help select a particular instance of the classifier (or decision tree, or whatever automatic learning application) but rather to qualify the model, i.e. to provide metrics such as the average error ratio, the deviation relative to this average etc. which can be useful in asserting the level of precision one can expect from the application. One of the things cross validation can help assert is whether the training data is big enough.

Cross validation isn't used for buliding/pruning the decision tree. It's used to estimate how good the tree (built on all of the data) will perform by simulating arrival of new data by building the tree without some elements just as with a minimum value for class. It doesn't really make sense to pick one of the trees generated by it because the model is constrained by the data you have (and not using it all might actually be worse when you use the tree for new data).

The tree is built over the data that you choose (usualy all of it). Pruning is usually done by using some heuristic (i.e. 90% of the elements

in the node belongs to class A so we don't go any further or the information gain is too small).

Earlier we have interpreted the data st different nodes such as left son and right son, such nodes are known as *folds*. These folds serves better at different levels of predictor. However, the parent node i.e. the level at 100% data always serves as a better predictor. Okay, given all that how are we going to do *pruning*.

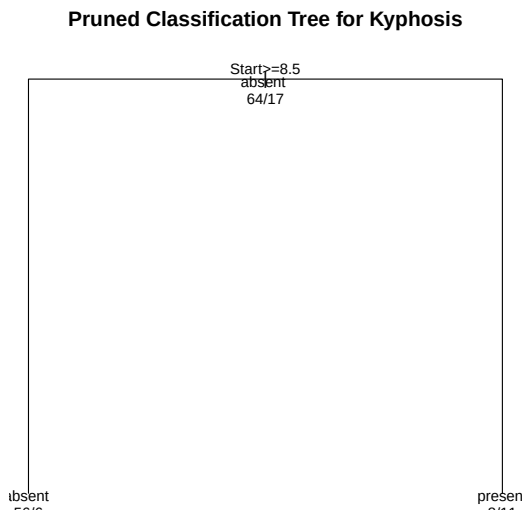
Pruning

Let us assume that we would like to take minimum possible error

```
> fit$cptable[which.min(fit$cptable[, "xerror"]), "CP"]
```

```
[1] 0.1764706
```

```
> pfit <- prune(fit, cp=0.1764)
> plot(pfit, uniform=TRUE,
       main="Pruned Classification Tree for Kyphosis")
> text(pfit, use.n=TRUE, all=TRUE, cex=.8)
> post(pfit, file = "D:/Work/Books/R/ML_AI/11.ps",
       title = "Pruned Classification Tree for Kyphosis")
```



So if we choose to prune for minimum value of cross validation error, the predition level is going to be the node 1. The details are clear from the below figure and can be obtrained through `summary(fit)`

Tree-Based Models: Regression Tree

Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modelling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.

In this section we will be working with a peculiar data set known as

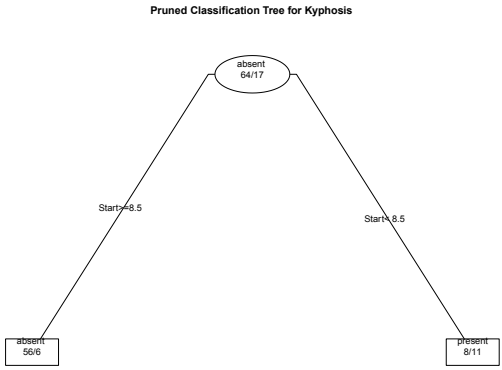


Figure 4.2: The PS figure

`cu.summary`. The `cu.summary` data frame has 117 rows and 5 columns, giving data on makes of cars taken from the April, 1990 issue of Consumer Reports. For more information you may try `help("cu.summary)`.

```
> data(cu.summary)
> head(cu.summary)
```

```
                Price Country Reliability Mileage  Type
Acura Integra 4 11950   Japan Much better      NA Small
Dodge Colt 4     6851   Japan        <NA>      NA Small
Dodge Omni 4     6995     USA  Much worse      NA Small
Eagle Summit 4   8895     USA      better      33 Small
Ford Escort   4  7402     USA       worse      33 Small
Ford Festiva 4   6319   Korea      better      37 Small
```

You may find the type of the rows and columns in the output. *Price* and *Mileage* are continuous, numerical and non-categorical data, whereas, *Country, Reliability* and *Type* are non-numerical, descrete and categorical data.

```
> levels(cu.summary$Country)
```

```
 [1] "Brazil"    "England"   "France"    "Germany"   "Japan"     "Japan/USA"
```

```
 [7] "Korea"     "Mexico"    "Sweden"    "USA"
```

```
> levels(cu.summary$Reliability)
```

```
[1] "Much worse"  "worse"       "average"     "better"      "Much better"
```

```
> levels(cu.summary$Type)
```

```
[1] "Compact" "Large"   "Medium"  "Small"   "Sporty"  "Van"
```

We might try a crosstab through `table()`.

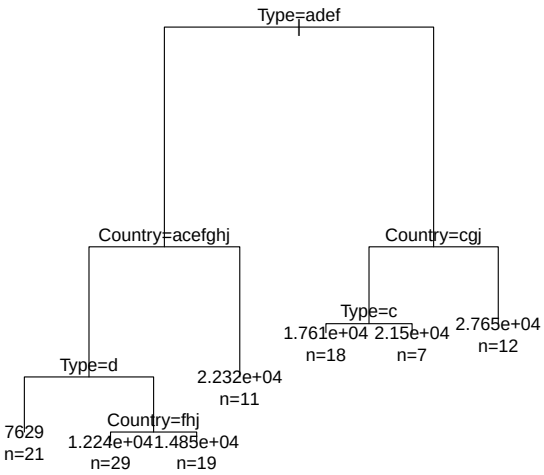
```
> car.tab <- table(cu.summary$Type, cu.summary$Reliability)
> chisq.test(car.tab)
```

```
        Pearson's Chi-squared test
data:  car.tab
X-squared = 35.119, df = 20, p-value = 0.01948
```

We can reject null hypothesis (H_0) that in favor of alternative hypothesis (H_a). Reliability is not independent of Type of the car. In otherwords, Reliability, which is the user perception that it is *better* or *worst*, pretty much depend on Type of the car such as *Large* or *Small*.

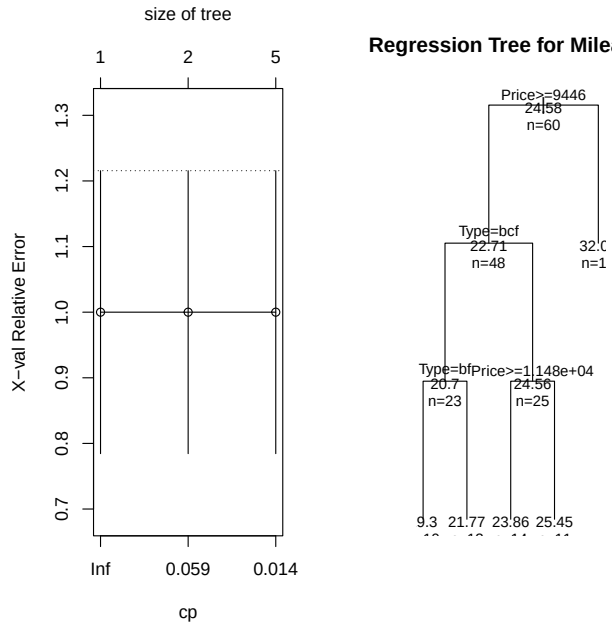
Now let us try *Regression Tree* on the same data.

```
> fit.reg.tree <- rpart(Price ~ Mileage + Type + Country, cu.summary)
> par(xpd = TRUE)
> plot(fit.reg.tree, compress = TRUE)
> text(fit.reg.tree, use.n = TRUE)
```



Let us try with anova method.

```
> fit.reg.tree1 <- rpart(Mileage~Price + Country + Reliability + Type,
        method="anova", data=cu.summary)
> par(mfrow=c(1,2))
> plotcp(fit) # plot 1
> plot(fit.reg.tree1, uniform=TRUE,
        main="Regression Tree for Mileage ") # plot 2
> text(fit.reg.tree1, use.n=TRUE, all=TRUE, cex=.8)
> post(fit.reg.tree1, file = "D:/Work/Books/R/ML_AI/12.ps",
        title = "Regression Tree for Mileage ")
>
```



The last figure helps us to make sense on regression tree.

```
> mil.na.omit <- na.omit(cu.summary[, "Mileage"])
> length(na.omit(cu.summary[, "Mileage"]))
```

```
[1] 60
```

```
> length(which(cu.summary$Price[mil.na.omit] >= 9446))
```

```
[1] 41
```

```
> length(which(cu.summary$Price[mil.na.omit] < 9446))
```

```
[1] 19
```

We are confused!

```
> summary(fit.reg.tree1)

Call:
rpart(formula = Mileage ~ Price + Country + Reliability + Type,
    data = cu.summary, method = "anova")
  n=60 (57 observations deleted due to missingness)
         CP nsplit rel error    xerror       xstd
```

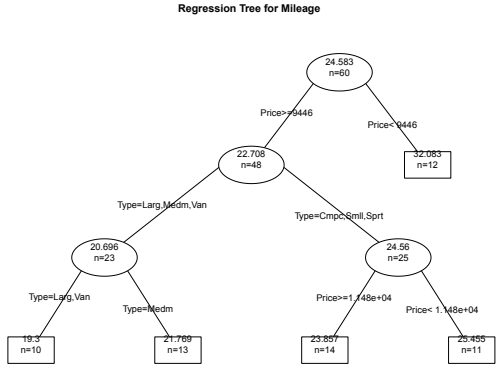


Figure 4.3: Regression tree for Mileage open in Inkscape

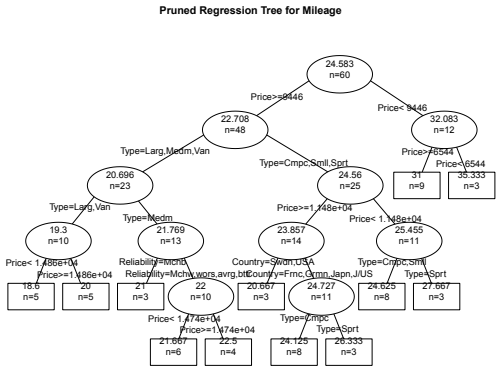
```
1 0.62288527      0 1.0000000 1.0251235 0.17570130
2 0.13206061      1 0.3771147 0.5397837 0.10391179
3 0.02544094      2 0.2450541 0.4218018 0.08634316
4 0.01160389      3 0.2196132 0.4291810 0.08844338
5 0.01000000      4 0.2080093 0.4335482 0.08780693
Variable importance
  Price    Type Country
     48      42      10
Node number 1: 60 observations,    complexity param=0.6228853
  mean=24.58333, MSE=22.57639
  left son=2 (48 obs) right son=3 (12 obs)
  Primary splits:
      Price       < 9446.5  to the right, improve=0.6228853, (0 missing)
      Type        splits as  LLLRLL,      improve=0.5044405, (0 missing)
      Reliability splits as  LLLRR,       improve=0.1263005, (11 missing)
      Country     splits as  --LRLRRRLL,  improve=0.1243525, (0 missing)
  Surrogate splits:
      Type    splits as  LLLRLL,     agree=0.950, adj=0.750, (0 split)
      Country splits as  --LLLLRRLL, agree=0.833, adj=0.167, (0 split)
Node number 2: 48 observations,    complexity param=0.1320606
  mean=22.70833, MSE=8.498264
  left son=4 (23 obs) right son=5 (25 obs)
  Primary splits:
      Type        splits as  RLLRRL,      improve=0.43853830, (0 missing)
      Price       < 12154.5 to the right, improve=0.25748500, (0 missing)
      Country     splits as  --RRLRL-LL,  improve=0.13345700, (0 missing)
      Reliability splits as  LLLRR,       improve=0.01637086, (10 missing)
  Surrogate splits:
      Price   < 12215.5 to the right, agree=0.812, adj=0.609, (0 split)
      Country splits as  --RRLRL-RL,  agree=0.646, adj=0.261, (0 split)
Node number 3: 12 observations
  mean=32.08333, MSE=8.576389
Node number 4: 23 observations,    complexity param=0.02544094
  mean=20.69565, MSE=2.907372
```

```
  left son=8 (10 obs) right son=9 (13 obs)
  Primary splits:
      Type    splits as  -LR--L,      improve=0.515359600, (0 missing)
      Price   < 14962   to the left,  improve=0.131259400, (0 missing)
      Country splits as  ----L-R--R,  improve=0.007022107, (0 missing)
  Surrogate splits:
      Price < 13572   to the right, agree=0.609, adj=0.1, (0 split)
Node number 5: 25 observations,    complexity param=0.01160389
  mean=24.56, MSE=6.4864
  left son=10 (14 obs) right son=11 (11 obs)
  Primary splits:
      Price       < 11484.5 to the right, improve=0.09693168, (0 missing)
      Reliability splits as  LLRRR,       improve=0.07767167, (4 missing)
      Type        splits as  L--RR-,      improve=0.04209834, (0 missing)
      Country     splits as  --LRRR--LL,  improve=0.02201687, (0 missing)
  Surrogate splits:
      Country splits as  --LLLL--LR, agree=0.80, adj=0.545, (0 split)
      Type    splits as  L--RL-,     agree=0.64, adj=0.182, (0 split)
Node number 8: 10 observations
  mean=19.3, MSE=2.21
Node number 9: 13 observations
  mean=21.76923, MSE=0.7928994
Node number 10: 14 observations
  mean=23.85714, MSE=7.693878
Node number 11: 11 observations
  mean=25.45455, MSE=3.520661
```

The summary shows that there exist 48 respondents out of 60 of valid Mileage cases whose Price is less than 9446 dollars. And the mean value of Mileage for this category is 22.708. But 12 of the respondents has 32.08 mileage which greater compared to the category where Price is greater than or equal to 9446 dollars. So a vehicle with less Price has more mileage. Moreover, Compact, Small and Sport vehicles seems to yield more mileage compared to Large, Medium, Van type of vehicles. Finally a vehicle whose Price is 11480 dollars and belongs to either of Compact, Small and Sport appers to be efficient in terms of mileage. However, the better choise might be any vehicle less than 9446 dollars (mileage = 32.083). Let us prune the tree for 1 splits (cp = 0.13206061)

```
> fit.reg.tree2 <- rpart(Mileage~Price + Country + Reliability + Type,
        method="anova", control=rpart.control(minsplit=10, cp=0.001),
                data=cu.summary)
> pfit.mil <- prune(fit.reg.tree2, cp=0.001)
> post(pfit.mil, file = "D:/Work/Books/R/ML_AI/13.ps",
        title = "Pruned Regression Tree for Mileage")
```

Finally, a vehicle whose price is greater than equal to 9446 dollars and possibly less than 11480 dollars “Sport” type specifically surveyed in France, Germany, Japan and US appears to be efficient in terms of Mileage.



Tree-Based Models: Conditional Inference Trees

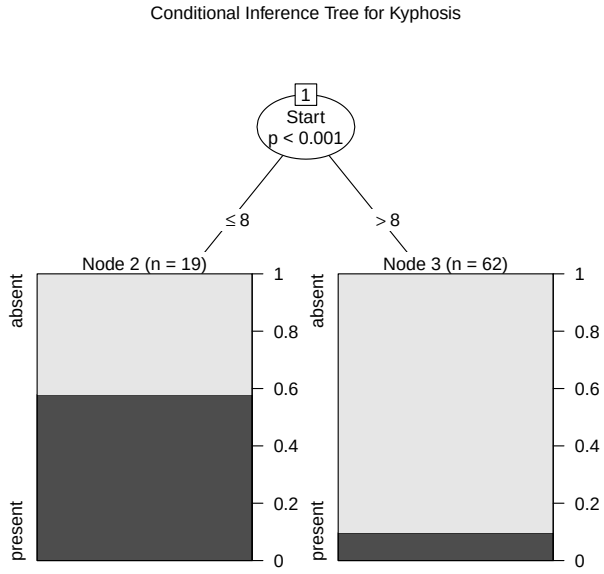
The `party` package provides nonparametric regression trees for nominal, ordinal, numeric, censored, and multivariate responses. You can create a regression or classification tree via the function `ctree()`. Statistics-based approach that uses non-parametric tests as splitting criteria, corrected for multiple testing to avoid overfitting. This approach results in unbiased predictor selection and does not require pruning. [9]

Decision-tree learners can create over-complex trees that do not generalize well from the training data. (This is known as overfitting.[20]) Mechanisms such as pruning are necessary to avoid this problem (with the exception of some algorithms such as the Conditional Inference approach, that does not require pruning

We will try to analyze both `kyphosis` and `cu.summary` using `ctree()`.

```
> library(party)
> fit3 <- ctree(Kyphosis ~ Age + Number + Start, data=kyphosis)
> plot(fit3, main="Conditional Inference Tree for Kyphosis")
> fit3

        Conditional inference tree with 2 terminal nodes
Response:  Kyphosis
Inputs:  Age, Number, Start
Number of observations:  81
1) Start <= 8; criterion = 1, statistic = 15.909
  2)*  weights = 19
```

```
1) Start > 8
  3)*  weights = 62
```



It seems that out of 81 patients 19 of them undergone surgery at less than or equal to 8 levels and the same is 62 for greater than 8. Let us see this:

```
> length(which(kyphosis[, "Age"] <8) == "TRUE")
        + length(which(kyphosis[, "Age"] <= 8) == "TRUE")

[1] 19

> length(which(kyphosis[, "Age"] > 8) == "TRUE")
        - length(which(kyphosis[, "Age"] <8) == "TRUE")

[1] 62

> table(kyphosis$Kyphosis[kyphosis[, "Age"] <= 8])

 absent present
     10       0

> 10/19

[1] 0.5263158

> 9/19
```

```
[1] 0.4736842
```

The above code shows the number of patients at left and right nodes along with proportions. It is not straighly intuitive to make any sense from the figure. The proportion of individuals absent are 0.473 and that absent are 0.526. The same way we can also carry out analysis for right node.

```
> table(kyphosis$Kyphosis[kyphosis[, "Age"] > 8])

 absent present
     54      17

> 54/62

[1] 0.8709677

> 17/62

[1] 0.2741935
```

The above calculations provides description for right side node i.e. Node 3 with 62 individuals.

4.2 Random Forests

Random forests or random decision forests are an *ensemble learning* method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

The first algorithm for random decision forests was created by Tin Kam Ho using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg. An extension of the algorithm was developed by Leo Breiman and Adele Cutler, and "Random Forests" is their trademark. The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho and later independently by Amit and Geman in order to construct a

collection of decision trees with controlled variance. Although random forests have been inherently designed to work only with multidimensional data, it has been shown that one can also use them for arbitrary objects (like graphs or time series) with the use of only pairwise similarities between objects. This variant is referred to as a similarity forest.

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = x_1, \cdots, x_n$ with responses $Y = y_1, \cdots, y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For b = 1, ..., B:

1. Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.
2. Train a classification or regression tree fb on X_b, Y_b.

   After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} f_b(x') \tag{4.4}$$

   An estimate of the uncertainty of the prediction can be made as the standard deviation of the predictions from all the individual regression trees on x':

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B} (f_b(x') - \hat{f})^2}{B - 1}}. \tag{4.5}$$

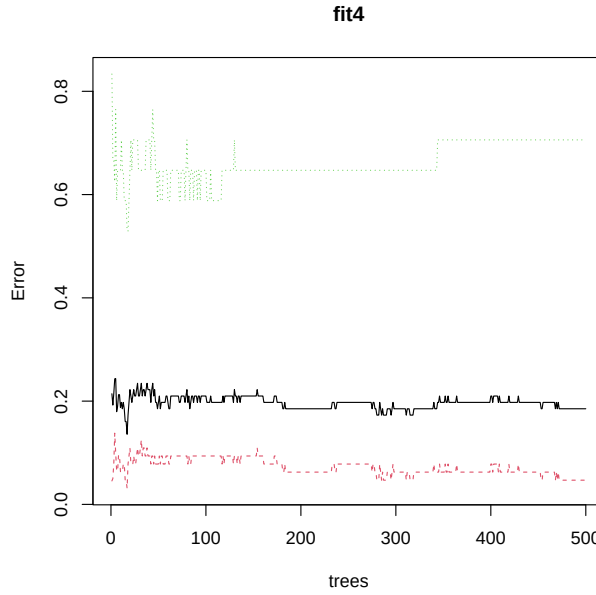
   The number of samples/trees, B, is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set. An optimal number of trees B can be found using cross-validation, or by observing the *out-of-bag error*: the mean prediction error on each training

sample x_i, using only the trees that did not have x_i in their bootstrap sample. The training and test error tend to level off after some number of trees have been fit.

You got a dedicated website for theory on *Random Forest* visit https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm to read and understand about this method. Let us try Random Forest on `kyphosis` data set.

Unsupervised Case

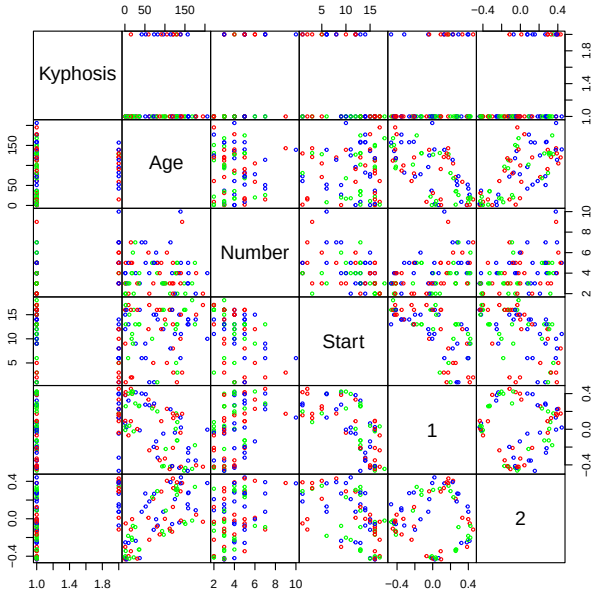
```
> library(randomForest)
> fit4 <- randomForest(Kyphosis ~ Age + Number + Start,   data=kypho
> print(fit4) # view results

Call:
 randomForest(formula = Kyphosis ~ Age + Number + Start, data = kyph
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 1

        OOB estimate of  error rate: 18.52%
Confusion matrix:
        absent present class.error
absent      61       3   0.0468750
present     12       5   0.7058824

> importance(fit4)

       MeanDecreaseGini
Age            8.328868
Number         5.688869
Start          9.856295

> plot(fit4)
```



Let us go ahead.

```
> kyp.rf <- randomForest(Kyphosis ~ ., data=kyphosis, importanc
> kyp.mds <- cmdscale(1 - kyp.rf$proximity, eig=TRUE)
> op <- par(pty="s")
> pairs(cbind(kyphosis[,1:4], kyp.mds$points), cex=0.6, gap=0,
> par(op)
> print(kyp.mds$GOF)
```

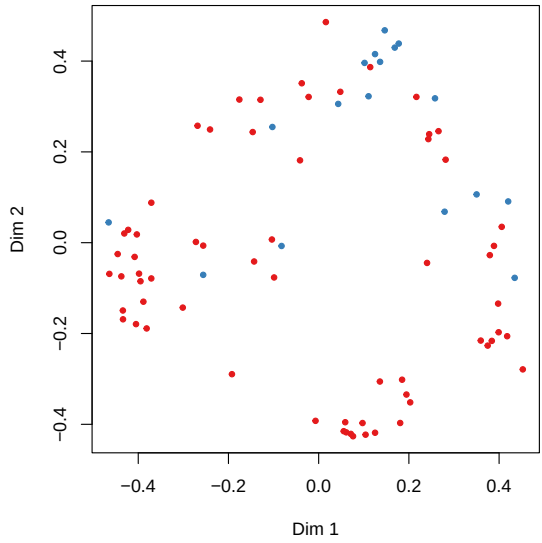
```
[1] 0.3052175 0.3599026
```



Supervised Case

```
> kyp.srf <- randomForest(Kyphosis ~ ., proximity = TRUE, data = kyp
> MDSplot(kyp.srf, kyphosis$Kyphosis)
> ## stratified sampling: draw 20, 30, and 20 of the species to grow
> for (i in 1:3){
+   x = i*20
+   print(randomForest(kyphosis[2:3], kyphosis$Kyphosis, sampsize=c(
+ }

Call:
 randomForest(x = kyphosis[2:3], y = kyphosis$Kyphosis, sampsize = c
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 1

        OOB estimate of  error rate: 23.46%
```

```
Confusion matrix:
        absent present class.error
absent      61       3   0.0468750
present     16       1   0.9411765

Call:
 randomForest(x = kyphosis[2:3], y = kyphosis$Kyphosis, sampsiz
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 1

        OOB estimate of  error rate: 23.46%
Confusion matrix:
        absent present class.error
absent      60       4   0.0625000
present     15       2   0.8823529

Call:
 randomForest(x = kyphosis[2:3], y = kyphosis$Kyphosis, sampsiz
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 1

        OOB estimate of  error rate: 24.69%
Confusion matrix:
        absent present class.error
absent      59       5   0.0781250
present     15       2   0.8823529

>
```



But the problem is the result is same irrespective of the sample size. So we can follow the earlier i.e. unsupervised method.

4.3 Support Vector Machines

Notes

[1]Source: https://www.r-project.org/about.html

[2]Source: Wikipedia; https://en.wikipedia.org/wiki/Decision_tree_learning

[3]Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). Classification and regression trees. Monterey, CA: Wadsworth Brooks/Cole Advanced Books Software. ISBN 978-0-412-04841-8.

[4]In decision tree learning, ID3 (Iterative Dichotomiser 3) is an algorithm invented by Ross Quinlan[1] used to generate a decision tree from a dataset. ID3 is the precursor to the C4.5 algorithm, and is typically used in the machine learning and natural language processing domains. Read more about ID3 algorithm athttps://en.wikipedia.org/wiki/ID3_algorithm

[5]C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan.[1] C4.5 is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a statistical classifier. Authors of the Weka machine learning software described the C4.5 algorithm as "a landmark decision tree program that is probably the machine learning workhorse most widely used in practice to date". Read more about C4.5 algorithm at https://en.wikipedia.org/wiki/C4.5_algorithm

[6]Perhaps the most recent algorithm must be C5 which is much improved version of C4.5 which Ross Quinnlan did. Quinnlan is commercially selling C5 through GNU-GPL. He added several features to his slgorithm such as high performance computing, boosting, winnowing (noise reduction) apart from other features pruning and attribute level handling *ala* discrete & continuious attribute handling and missing attribute values in C4.5. Read more about C5 at https://en.wikipedia.org/wiki/Ross_Quinlan

[7]Read more aboout `rpart` at http://cran.r-project.org/web/packages/rpart/index.html

[8]http://www.mayo.edu/hsr/techrpt/61.pdf

[9]Hothorn, T.; Hornik, K.; Zeileis, A. (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework". Journal of Computational and Graphical Statistics. 15 (3): 651-674. JSTOR 27594202. doi:10.1198/106186006X133933.

Chapter 5

Neural Networks - Supervised

5.1 Single Layer Perceptrons

5.2 Multi-layer Perceptron

Chapter 6

NLP

Chapter 7

Deep Learning

7.1 Speech Recognition

7.2 Image Recognition

Chapter 8

Genetic Algorithms

Chapter 9

Swarm Intellegence

Part I

UNSUPERVISED LEARNING

Chapter 10

Gausian Mixture Modelling

Chapter 11

Manifold Learning

Chapter 12

Clustering

Chapter 13

Density Estimation

Chapter 14

Neural Networks - Unsupervised Learning