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.

Chapter 3

Supervised learning

3.1 Logistic regression

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1. Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1). The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc. Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets. Logistic Regression can be used to classify the observations

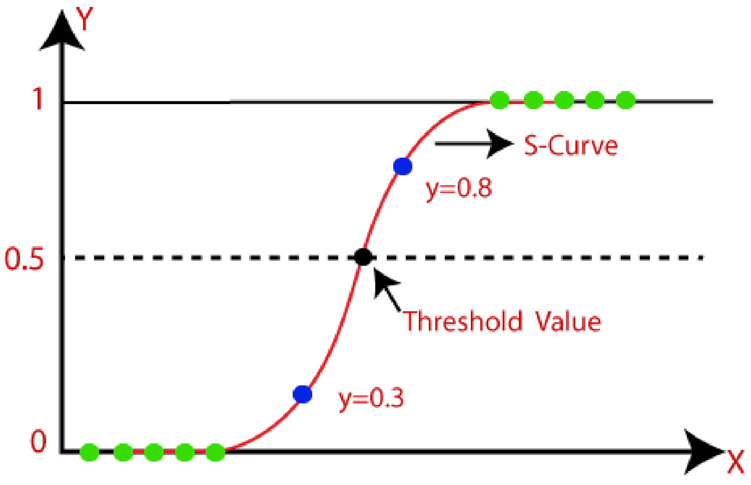

Figure 3.1: Logistic regression

using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:

3.1.1 Logistic Function (Sigmoid Function)

The sigmoid function is a mathematical function used to map the predicted values to probabilities. It maps any real value into another value within a range of 0 and 1. The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function. In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

3.1.2 Assumptions for Logistic Regression

1. The dependent variable must be categorical in nature.

2. The independent variable should not have multi-collinearity.

3.1.3 Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

We know the equation of the straight line can be written as:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

Figure 3.2:

In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y};\ \text{0 for y= 0, and infinity for y=1}$$

Figure 3.3:

But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

Figure 3.4:

The above equation is the final equation for Logistic Regression.

3.1.4 Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

Binomial: In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.

Multinomial: In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as ”cat”, ”dogs”, or ”sheep”

Ordinal: In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as ”low”, ”Medium”, or ”High”.

3.1.5 Procedure & R code

splitting the data set into ratio 0.80:0.20

```
split <- sample.split(df\$default, SplitRatio = 0.80)
```

creating training dataset

```
trainingSet <- subset(df, split == TRUE)
```

creating test data set

```
testSet <- subset(df, split == FALSE)
```

control parameters

```
library(caret)

objControl <- trainControl(method = "boot",
                           number = 2,
                           returnResamp = 'none',
                           summaryFunction = twoClassSummary,
                           classProbs = TRUE,
                           savePredictions = TRUE)
```

model building using caret package

```
set.seed(766)
caretLogitModel <- train(trainingSet[, 2:4],
                         trainingSet[, 1],
                         method = 'glm',
                         trControl = objControl,
                         metric = "ROC")

summary(caretLogitModel)
```

predicting the test set observations

```
logitModelPred <- predict(caretLogitModel, testSet,
type = "prob")
logitModelPred <- logitModelPred[,2]
```

plot of probabilities

```
plot(logitModelPred,
     main = "Scatterplot of Probabilities
of Default (test data)",
     xlab = "Customer ID", ylab = "Predicted
Probability of Default")
```

3.2 Decision Trees

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees. Below diagram explains the general structure of a decision tree:

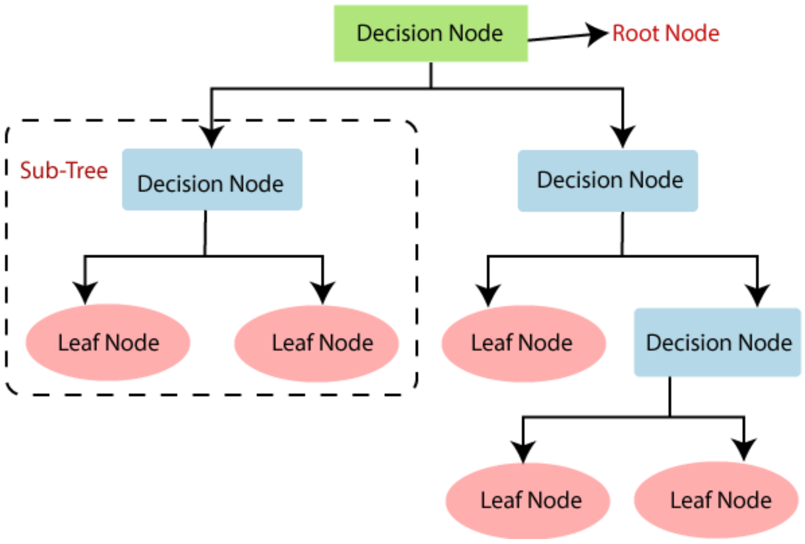

Figure 3.5:

3.2.1 Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

1. Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.

2. The logic behind the decision tree can be easily understood because it shows a tree-like structure.

3.2.2 Decision Tree Terminologies

Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

Branch/Sub Tree: A tree formed by splitting the tree.

Pruning: Pruning is the process of removing the unwanted branches from the tree.

Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

3.2.3 How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

3.2.4 Example

Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the 3.6 diagram.

3.2.5 Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM. By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

1. Information Gain
2. Gini Index

3.2.6 Information Gain

Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute. It calculates how much information a feature provides us about a class. According to the value of information gain, we split the node and build the decision tree. A decision tree algorithm always tries to maximize the value of

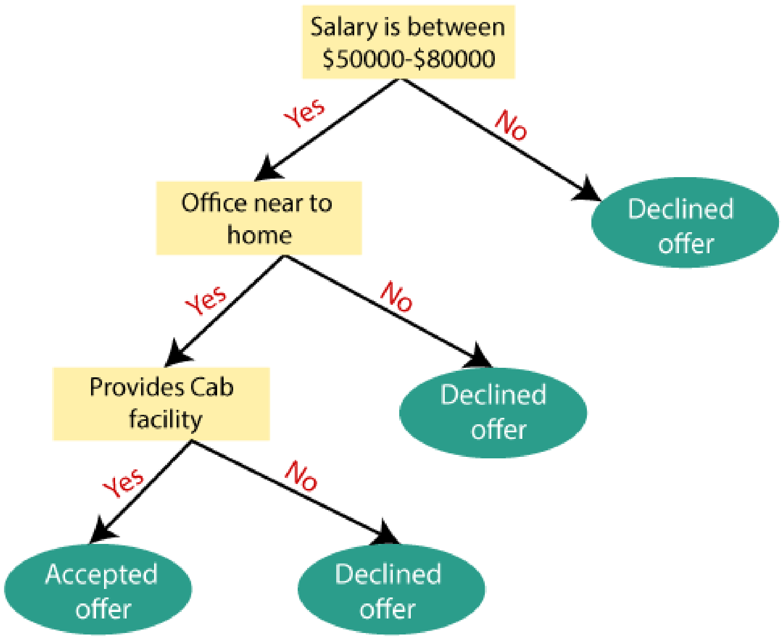

Figure 3.6: Decision tree example

information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

$InformationGain = Entropy(S) - [(WeightedAvg) * Entropy(eachfeature)]$

Entropy

Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

$Entropy(s) = -P(yes)log^2P(yes) - P(no)log^2P(no)$

Where,

S = Total number of samples; P(yes)= probability of yes; P(no)= probability of no;

3.2.7 Gini Index

Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm. An attribute with the low Gini index should be preferred as compared to the high Gini index. It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits. Gini index can be calculated using the below formula:

$GiniIndex = 1 - \sum_j P_j^2$

3.2.8 Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree. A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree pruning technology used:

1. Cost Complexity Pruning
2. Reduced Error Pruning.

3.2.9 Advantages of the Decision Tree

1. It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
2. It can be very useful for solving decision-related problems.
3. It helps to think about all the possible outcomes for a problem.
4. There is less requirement of data cleaning compared to other algorithms.

3.2.10 Disadvantages of the Decision Tree

1. The decision tree contains lots of layers, which makes it complex.
2. It may have an overfitting issue, which can be resolved using the Random Forest algorithm.

3. For more class labels, the computational complexity of the decision tree may increase.

3.2.11 Prcedure & R Code

Data

```
data("iris")
names(iris) = tolower(names(iris))
table(iris$species)
```

Data partition

```
suppressMessages(library(caret))
index = createDataPartition(y=iris$species, p=0.7,
list=FALSE)

train.set = iris[index,]
test.set = iris[-index,]

dim(train.set)
dim(test.set)

with(iris, qplot(petal.width, sepal.width,
colour=species, cex=2))
```

fit the model

```
iris.tree = train(species ~ .,
                  data=train.set,
                  method="rpart",
                  trControl = trainControl(method = "cv"))

iris.tree
summary(iris.tree$finalModel)
```

plot the model

```
plot(iris.tree$finalModel, uniform=TRUE,
     main="Classification Tree")
text(iris.tree\$finalModel, use.n.=TRUE, all=TRUE, cex=.8)
```

3.3 Random Forests

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, ”Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.” Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting. The figure 3.7 explains the working of the Random Forest algorithm.

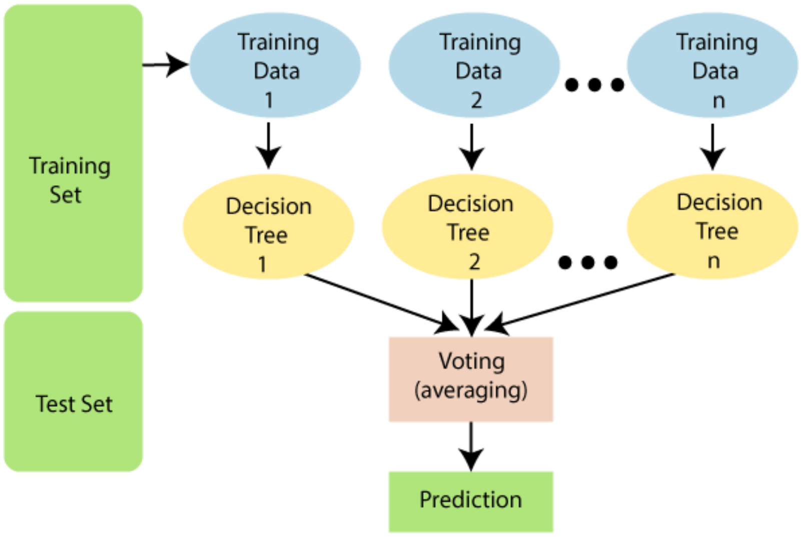

Figure 3.7:

3.3.1 Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

1. There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
2. The predictions from each tree must have very low correlations.

3.3.2 Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

1. It takes less training time as compared to other algorithms.
2. It predicts output with high accuracy, even for the large dataset it runs efficiently.
3. It can also maintain accuracy when a large proportion of data is missing.

3.3.3 How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

3.3.4 Example

Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the 3.8 image:

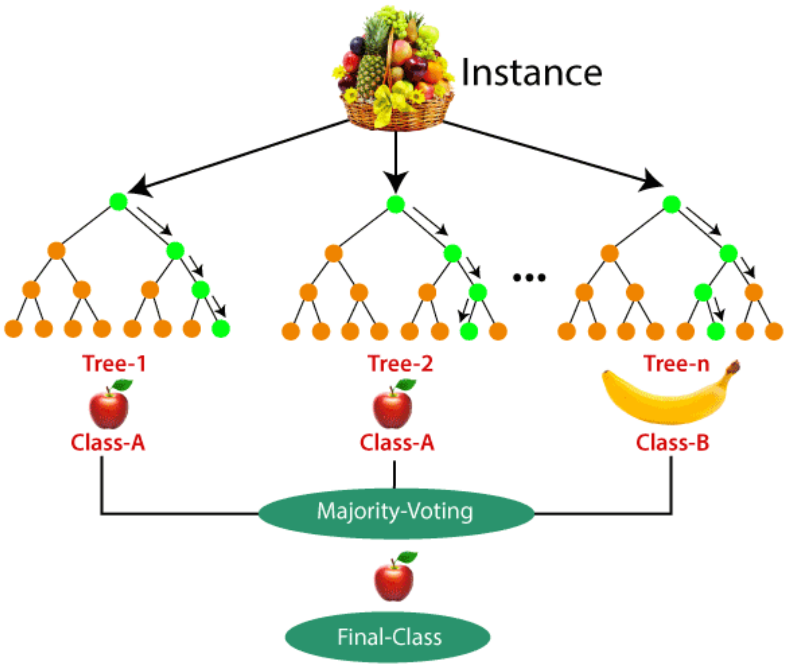

Figure 3.8:

3.3.5 Applications of Random Forest

There are mainly four sectors where Random forest mostly used:

1. Banking: Banking sector mostly uses this algorithm for the identification of loan risk.
2. Medicine: With the help of this algorithm, disease trends and risks of the disease can be identified.
3. Land Use: We can identify the areas of similar land use by this algorithm.
4. Marketing: Marketing trends can be identified using this algorithm.

3.3.6 Advantages of Random Forest

1. Random Forest is capable of performing both Classification and Regression tasks.
2. It is capable of handling large datasets with high dimensionality.
3. It enhances the accuracy of the model and prevents the overfitting issue.

3.3.7 Disadvantages of Random Forest

Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

3.3.8 Procedure & R Code

Required Libraries

```
library(ISLR)
library(caret)
```

Spliting data as training and test set. Using `createDataPartition()` function from caret

```
indxTrain <- createDataPartition(y = Smarket$Direction,
p = 0.75,list = FALSE)
```

```
training <- Smarket[indxTrain,]
testing <- Smarket[-indxTrain,]
```

Checking distibution in origanl data and partitioned data

```
prop.table(table(training$Direction)) * 100
prop.table(table(testing$Direction)) * 100
prop.table(table(Smarket$Direction)) * 100
```

Preprocessing

```
trainX <- training[,names(training) != "Direction"]
preProcValues <- preProcess(x = trainX,
method = c("center", "scale"))
preProcValues
```

Training and train control

```
ctrl <- trainControl(method="repeatedcv",repeats = 3)
```

Random forrest

```
rfFit <- train(Direction ~ ., data = training,
method = "rf", trControl = ctrl,
preProcess = c("center","scale"), tuneLength = 20)
rfFit
plot(rfFit)
```

Prediction & confusion matrix

```
rfPredict <- predict(rfFit,newdata = testing )
confusionMatrix(rfPredict, testing$Direction )
```

3.4 Support Vector Machines

In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis. Developed at AT&T Bell Laboratories by Vladimir Vapnik with colleagues (Boser et al., 1992, Guyon et al., 1993, Cortes and Vapnik, 1995, Vapnik et al., 1997. [2] SVMs are one of the most robust prediction methods, being based on statistical learning frameworks or VC theory proposed by Vapnik (1982,

1995) and Chervonenkis (1974). Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

3.4.1 Applications

SVMs can be used to solve various real-world problems:

- SVMs are helpful in text and hypertext categorization, as their application can significantly reduce the need for labeled training instances in both the standard inductive and transductive settings. Some methods for shallow semantic parsing are based on support vector machines. [34]
- Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. This is also true for image segmentation systems, including those using a modified version SVM that uses the privileged approach as suggested by Vapnik. [5]
- Classification of satellite data like SAR data using supervised SVM.
- Hand-written characters can be recognized using SVM. [6]
- The SVM algorithm has been widely applied in the biological and other sciences. They have been used to classify proteins with up to 90% of the compounds classified correctly. Permutation tests based on SVM weights have been suggested as a mechanism for interpretation of SVM models. Support vector machine weights have also been used to interpret SVM models in the past. Posthoc interpretation of support vector machine

models in order to identify features used by the model to make predictions is a relatively new area of research with special significance in the biological sciences.

3.4.2 Support Vector Machines in R

3.4.3 Linear SVM Classifier

Data

```
library(caret)

# Load the data
data("PimaIndiansDiabetes2", package = "mlbench")
pima.data <- na.omit(PimaIndiansDiabetes2)

# Inspect the data
sample_n(pima.data, 3)
```

Training data

```
# Set up Repeated k-fold Cross Validation
train_control <- trainControl(method="repeatedcv",
number=10, repeats=3)
```

Fitting data and viewing model

```
# Fit the model
svm1 <- train(diabetes ~., data = pima.data,
method = "svmLinear", trControl = train_control,
preProcess = c("center","scale"))
#View the model
svm1
```

Plotting fit

```
# Plot model accuracy vs different values of Cost
plot(svm2)
```

Notes

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

[2]Cortes, Corinna; Vapnik, Vladimir (1995). ”Support-vector networks” (PDF). Machine Learning. 20 (3): 273297. CiteSeerX 10.1.1.15.9362. doi:10.1007/BF00994018. S2CID 206787478.

[3]Joachims, Thorsten (1998). “Text categorization with Support Vector Machines: Learning with many relevant features”. Machine Learning: ECML-98. Lecture Notes in Computer Science. Vol. 1398. Springer. pp. 137142. doi:10.1007/BFb0026683. ISBN 978-3-540-64417-0.

[4]Pradhan, Sameer S.; et al. (2 May 2004). Shallow Semantic Parsing using Support Vector Machines. Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004. Association for Computational Linguistics. pp. 233240.

[5]Barghout, Lauren (2015). “Spatial-Taxon Information Granules as Used in Iterative Fuzzy-Decision-Making for Image Segmentation”. Granular Computing and Decision-Making. Studies in Big Data. Vol. 10. pp. 285318.

[6]DeCoste, Dennis (2002). “Training Invariant Support Vector Machines”. Machine Learning. 46: 161190.

Chapter 4

Unsupervised Learning

4.1 Gaussian Mixture Modeling

4.2 Manifold Learning

4.3 Clustering

4.4 Density Estimation

4.5 Neural Networks - Unsupervised Learning