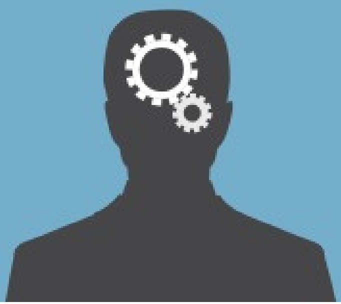
Simple | Smart | Speed
CHOOLS

MACHINE LEARNING MASTERY



Contents

Part I

Intro duction

Chapter 1

Welcome

Welcome to Machine Learning Mastery With R. This book is your guide to applied machine learning with R. You will discover the step-by-step process that you can use to get started and become good at machine learning for predictive modeling on the R platform.

1.1 Learn R The Wrong Way

Here is what you should NOT do when you start studying machine learning in R.

1. Get really good at R programming and R syntax.
2. Deeply study the underlying theory and parameters for machine learning algorithms in R.
3. Avoid or lightly touch on all of the other tasks needed to complete a real project.

I think that this approach can work for some people, but it is a really slow and a roundabout way of getting to your goal. It teaches you that you need to spend all your time learning how to use individual machine learning algorithms. It also does not teach you the process of building predictive machine learning models in R that you can actually use to make predictions. Sadly, this is the approach used to teach machine learning that I see in almost all books and online courses on the topic.

1.2 MachineLearninginR

This book focuses on a specific sub-field of machine learning called predictive modeling. This is the field of machine learning that is the most useful in industry and the type of machine learning that the R platform excels at facilitating. Unlike statistics, where models are used to understand data, predictive modeling is laser focused on developing models that make the most accurate predictions at the expense of explaining why predictions are made. Unlike the broader field of machine learning that could feasibly be used with data in any format, predictive modeling is primarily focused on tabular data (e.g. tables of numbers like a spreadsheet).

This book was written around three themes designed to get you started and using R for applied machine learning effectively and quickly. These three parts are as follows:

Lessons: Learn how the sub-tasks of a machine learning project map onto R and the best practice way of working through each task.

Projects: Tie together all of the knowledge from the lessons by working through case study predictive modeling problems.

Recipes: Apply machine learning with a catalog of standalone recipes in R that you can copy-and-paste as a starting point for new projects.

1.2.1 Lessons

You need to know how to complete the specific subtasks of a machine learning project on the R platform. Once you know how to complete a discrete task using the platform and get a result reliably, you can do it again and again on project after project. Let's start with an overview of the common tasks in a machine learning project. A predictive modeling machine learning project can be broken down into 6 top-level tasks:

1. Define Problem: Investigate and characterize the problem in order to better understand the goals of the project.
2. Analyze Data: Use descriptive statistics and visualization to better understand the data you have available.
3. Prepare Data: Use data transforms in order to better expose the structure of the prediction problem to modeling algorithms.
4. Evaluate Algorithms: Design a test harness to evaluate a number of standard algorithms on the data and select the top few to investigate further.
5. Improve Results: Use algorithm tuning and ensemble methods to get the most out of well-performing algorithms on your data.
6. Present Results: Finalize the model, make predictions and present results.

A blessing and a curse with R is that there are so many techniques and so many ways to do something with the platform. In part II of this book you will discover one easy or best practice way to complete each subtask of a general machine learning project. Below is a summary of the Lessons from Part II and the sub-tasks that you will learn about.

Lesson 1: Install and Start R.

Lesson 2: R Language Crash Course.

Lesson 3: Load Standard Datasets.

Lesson 4: Load Custom Data. (Analyze Data)

Lesson 5: Understand Data With Descriptive Statistics. (Analyze Data)

Lesson 6: Understand Data With Visualization. (Analyze Data)

Lesson 7: Pre-Process Data. (Prepare Data)

Lesson 8: Resampling Methods. (Evaluate Algorithms)

Lesson 9: Algorithm Evaluation Metrics. (Evaluate Algorithms)

Lesson 10: Spot-Check Algorithms. (Evaluate Algorithms)

Lesson 11: Model Selection. (Evaluate Algorithms)

Lesson 12: Algorithm Parameter Tuning. (Improve Results)

Lesson 13: Ensemble Methods. (Improve Results)

Lesson 14: Finalize Model. (Present Results)

These lessons are intended to be read from beginning to end in order, showing you exactly how to complete each task in a predictive modeling machine learning project. Of course, you can dip into specific lessons again later to refresh yourself. Some lessons are demonstrative, showing you how to use specific techniques for a common machine learning task (e.g. data loading and data pre-processing). Others are in a tutorial format, building throughout the lesson to culminate in a final result (e.g. algorithm tuning and ensemble methods). Each lesson was designed to be completed in under 30 minutes (depending on your level of skill and enthusiasm). It is possible to work through the entire book in one weekend. It also works if you want to dip into specific sections and use the book as a reference.

1.2.2 Projects

Recipes for common predictive modeling tasks are critically important, but they are also just the starting point. This is where most books and courses stop.

You need to piece the recipes together into end-to-end projects. This will show you how to actually deliver a model or make predictions on new data using R. This book uses small well-understood machine learning datasets from the UCI Machine learning repository[1] in both the lessons and in the example projects. These datasets are available for free as CSV downloads, and most are available directly in R by loading third party packages. These datasets are excellent for practicing applied machine learning because:

They are small, meaning they fit into memory and algorithms can model them in reasonable time.

They are well behaved, meaning you often don't need to do a lot of feature engineering to get a good result.

They are benchmarks, meaning that many people have used them before and you can get ideas of good algorithms to try and accuracy levels you should expect.

In Part III you will work through three projects:

[1] http://archive.ics.uci.edu/ml

Hello World Project (Iris flowers dataset) : This is a quick pass through the project steps without much tuning or optimizing on a dataset that is widely used as the hello world of machine learning.

Regression (Boston House Price dataset) : Work through each step of the project process with a regression problem.

Binary Classification (Wisconsin Breast Cancer dataset) : Work through each step of the project process using all of the methods on a binary classification problem.

These projects unify all of the lessons from Part II. They also give you insight into the process for working through predictive modeling machine learning problems which is invaluable when you are trying to get a feeling for how to do this in practice. Also included in this section is a template for working through predictive modeling machine learning problems which you can use as a starting point for current and future projects. I find this useful myself to set the direction and setup important tasks (which are easy to forget) on new projects, and I'm sure you will too.

1.2.3 Recip es

Recipes are small code snippets in R that show you how to do one specific thing and get a result. For example, you could have a recipe that demonstrates how to use Random Forest algorithm for classification. You could have another for normalizing the attributes of a dataset.

Recipes make the difference between a beginner who is having trouble and a fast learner capable of making accurate predictions quickly on any new project. A catalog of recipes provides a repertoire of skills that you can draw from when starting a new project. More formally, recipes are defined as follows:

Recipes are code snippets not tutorials.

Recipes provide just enough code to work.

Recipes are demonstrative not exhaustive.

Recipes run as-is and produce a result.

Recipes assume that required packages are installed.

Recipes use built-in datasets or datasets provided in specific packages.

You are starting your journey into machine learning with R with my personal catalog of machine learning recipes provided with this book. All of the code from the lessons in Part II are available in your R recipe catalog. There are also recipes for techniques not covered in this book, including usage of a very large number of algorithms and many additional case studies. Recipes are divided into directories according to the common tasks of a machine learning project as listed above. The list below provides a summary of the recipes available.

Analyze Data: Recipes to load, summarize and visualize data, including visualizations using univariate plots, multivariate plots and projection methods.

Prepare Data: Recipes for data preparation including data cleaning, feature selection and data transforms.

Algorithms: Recipes for using a large number of machine learning algorithms both standalone and within the popular R package caret, including linear, nonlinear, trees, ensembles for classification and regression.

Evaluate Algorithms: Recipes for re-sampling methods, algorithm evaluation metrics and model selection.

Improve Results: Recipes for algorithm tuning and ensemble methods.

Finalize Model: Recipes to make final predictions, to finalize the model and save and load models to disk.

Other: Recipes for managing packages and getting started with R syntax.

Case Studies: Case studies for binary classification, multiclass classification and regression problems.

This is an invaluable resource that you can use to jump-start your current and future machine learning projects. You can also build upon this recipe catalog as you discover new techniques.

1.2.4 YourOutcomesFromThisProcess

This book will lead you from being a developer who is interested in machine learning with R to a developer who has the resources and capability to work through a new dataset end-to-end using R and develop accurate predictive models. Specifically, you will know:

How to work through a small to medium sized dataset end-to-end.

How to deliver a model that can make accurate predictions on new unseen data.

How to complete all subtasks of a predictive modeling problem with R.

How to learn new and different techniques in R.

How to get help with R.

From here you can start to dive into the specifics of the functions, techniques and algorithms used with the goal of learning how to use them better in order to deliver more accurate predictive

models, more reliably in less time..

1.3 What This Book is Not

This book was written for professional developers who want to know how to build reliable and accurate machine learning models in R.

This is not a machine learning textbook. We will not be getting into the basic theory of machine learning (e.g. induction, bias-variance trade-off, etc.). You are expected to have some familiarity with machine learning basics, or be able to pick them up yourself.

This is not an algorithm book. We will not be working through the details of how specific machine learning algorithms work (e.g. random forest). You are expected to have some basic knowledge of machine learning algorithms or how to pick up this knowledge yourself.

This is not an R programming book. We will not be spending a lot of time on R syntax and programming (e.g. basic programming tasks in R). You are expected to be a developer who can pick up a new C-like language relatively quickly.

You can still get a lot out of this book if you are weak in one or two of these areas, but you may struggle picking up the language or require some more explanation of the techniques. If this is the case, see the Resources Chapter at the end of the book and seek out a good companion reference text.

1.4 Summary

I hope you are as excited as me to get started. In this introduction chapter you learned that this book is unconventional. Unlike other books and courses that focus heavily on machine learning algorithms in R and focus on little else, this book will walk you through each step of a predictive modeling machine learning project.

Part II of this book provides standalone lessons including a mixture of recipes and tutorials to build up your basic working skills and confidence in R.

Part III of this book will introduce a machine learning project template that you can use as a starting point on your own projects and walks you through three end-to-end projects.

The recipes companion to this book provides a catalog of more than 150 machine learning recipes in R. You can browse this invaluable resource, find useful recipes and copy-and-paste them into your current and future machine learning projects.

Part IV will finish out the book. It will look back at how far you have come in developing your new found skills in applied machine learning with R. You will also discover resources that you can use to get help if and when you have any questions about R or the platform.

1.4.1 NextStep

In the next Chapter you will take a closer look at R. You will discover what R is, why it is so powerful as a platform for machine learning and the different ways you should and should not use the platform.

Chapter 2

The R Platform

R is one of the most powerful machine learning platforms and is used by the top data scientists in the world. In this Chapter you will get an introduction to R and why you should use R for machine learning.

2.1 WhyUseR

There are five reasons why you should use R for your predictive modeling machine learning problems:

R is used by the best data scientists in the world. In surveys on Kaggle (the competitive machine learning platform), R is by far the most used machine learning tool[1] When professional machine learning practitioners were surveyed in 2015, again the most popular machine learning tool was R[2].

R is powerful because of the breadth of techniques it offers in third-party packages . Any techniques that you can think of for data analysis, visualization, data sampling, supervised learning and model evaluation are provided in R. The platform has more techniques than any that you will come across.

R is state-of-the-art because it is used by academics. One of the reasons why R has so many techniques is because academics who develop new algorithms are developing them in R and releasing them as R packages. This means that you can get access to state-of-the-art algorithms in R before other platforms. It also means that you can only access some algorithms in R until someone ports them to other platforms.

R is free because it is open source software. You can download it right now for free and it runs on any workstation platform you are likely to use.

R is a lot of fun. I think the fun comes from the sense of exploration (you're always finding out about some new amazing technique) and because of the results you get (you can run very powerful methods on your data in a few lines of code). I use other platforms, but I always come back to R when I've got serious work to do.

Convinced?

[1] http://blog.kaggle.com/2011/11/27/kagglers-favorite-tools

[2] http://www.kdnuggets.com/2015/05/poll-r-rapidminer-python-big-data-spark.html

2.2 What Is R

R is a language, an interpreter and a platform.

R is a computer language . It can be difficult to learn but is familiar and you will figure it out quickly if you have used other scripting languages like Python, Ruby or BASH.

R is an interpreter . You can write scripts and save them as files. Like other scripting languages, you can then use the interpreter to run those scripts any time. R also provides a REPL (read-evaluate-print loop) environment where you can type in commands and see the output immediately.

R is also a platform . You can use it to create and display graphics, to save and load state and to interface with other systems. You can do all of your exploration and development in the REPL environment if you so wish.

2.2.1 WhereRCameFrom

R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand as an implementation of the S programming language. Development started in 1993. A version was made available on FTP released under the GNU GPL in 1995. The larger core group and open source project was setup in 1997.

It started as an experiment by the authors to implement a statistical test bed in Lisp using a syntax like that provided in S. As it developed, it took on more of the syntax and features of S, eventually surpassing it in capability and scope.

2.2.2 Power In The Packages

R itself is very simple. It provides built in commands for basic statistics and data handling. The machine learning features of R that you will use come from third party packages. Packages are plug-ins to the R platform. You can search for, download and install them within the R environment.

Because packages are created by third parties, their quality can vary. It is a good idea to search for the best-of-breed packages that provide a specific technique you want to use. Packages provide documentation in the form of help for each package function and often vignettes that demonstrate how to use the package.

Before you write a line of code, always search to see if there is a package that can do what you need. You can search for packages on the Comprehensive R Archive Network or CRAN for short[3] .

2.2.3 NotForProduction

R is probably not the best solution for building a production model. The techniques may be state-of-the-art but they may not use the best software engineering principles, have tests or be scalable to the size of the datasets that you may need to work with.

[3] https://cran.r- project.org

That being said, R may be the best solution to discover what model to actually use in production. The landscape is changing and people are writing R scripts to run operationally and services are emerging to support larger datasets.

2.3 Summary

This chapter provided an introduction to R. You discovered:

5 Good reasons why you should be using R for machine learning, not least because it is used by some of the best data scientists in the world.

That R is 3 things, a programming language, interpreter and platform.

That the power of R comes from the free third-party packages that you can download.

That R is excellent for R&D and one-off projects, but probably inappropriate for running production models.

2.3.1 Next Step

Next is Part II where you will begin working through the lessons, the main body of this book. The first lesson is quick and will assist you in installing R and running it for the first time.

Part II

Lessons

Chapter 3

Understand Your Data Using Data Visualization

You must understand your data to get the best results from machine learning algorithms. Data visualization is perhaps the fastest and most useful way to summarize and learn more about your data. In this lesson you will discover exactly how you can use data visualization to better understand your data for machine learning using R. After completing this lesson, you will know:

1. How to create plots in order to understand each attribute standalone.

2. How to create plots in order to understand the relationships between attributes.

Let’s get started.

3.1 UnderstandYourDataToGetTheBestResults

A better understanding of your data will yield better results from machine learning algorithms. You will be able to clean, transform and best present the data that you have. The better that the data exposes the structure of the problem to the machine learning algorithms, the more accurate your models will be. Additionally, a deeper understanding of your data may even suggest specific machine learning algorithms to try on your data.

3.1.1 VisualizeYourDataForFasterUnderstanding

The fastest way to improve your understanding of your dataset is to visualize it. Visualization means creating charts and plots from the raw data. Plots of the distribution or spread of attributes can help you spot outliers, strange or invalid data and give you an idea of possible data transformations you could apply.

Plots of the relationships between attributes can give you an idea of attributes that might be redundant, resampling methods that may be needed and ultimately how difficult a prediction problem might be. In the next section you will discover how you can quickly visualize your data in R. This lesson is divided into three parts:

Visualization Packages: A quick note about your options when it comes to R packages for visualization.

Univariate Visualization: Plots you can use to understand each attribute standalone.

Multivariate Visualization: Plots that can help you to better understand the interactions between attributes.

3.2 VisualizationPackages

There are many ways to visualize data in R, but a few packages have surfaced as perhaps being the most generally useful.

graphics package: Excellent for fast and basic plots of data.

lattice package: More pretty plots and more often useful in practice.

ggplot2 package: Beautiful plots that you want to generate when you need to present results.

I recommend that you stick with simple plots from the graphics package for quick and dirty visualization, and use wrappers around lattice (via the caret package) for more useful multivariate plots. I think ggplot2 plots are excellent and look lovely, but overkill for quick and dirty data visualization.

3.3 UnivariateVisualization

Univariate plots are plots of individual attributes without interactions. The goal is to learn something about the distribution, central tendency and spread of each attribute.

3.3.1 Histograms

Histograms provide a bar chart of a numeric attribute split into bins with the height showing the number of instances that fall into each bin. They are useful to get an indication of the distribution of an attribute.

```
# load the data
data(iris)
#   create   histograms   for   each
attribute par(mfrow=c(1,4))
for(i in 1:4) {
    hist(iris[,i], main=names(iris)[i])
}
```

Listing 3.1: Calculate histograms.

You can see that most of the attributes show a Gaussian or multi-modal Gaussian distribution. You can see the measurements of very small flowers in the Petal width and length column.

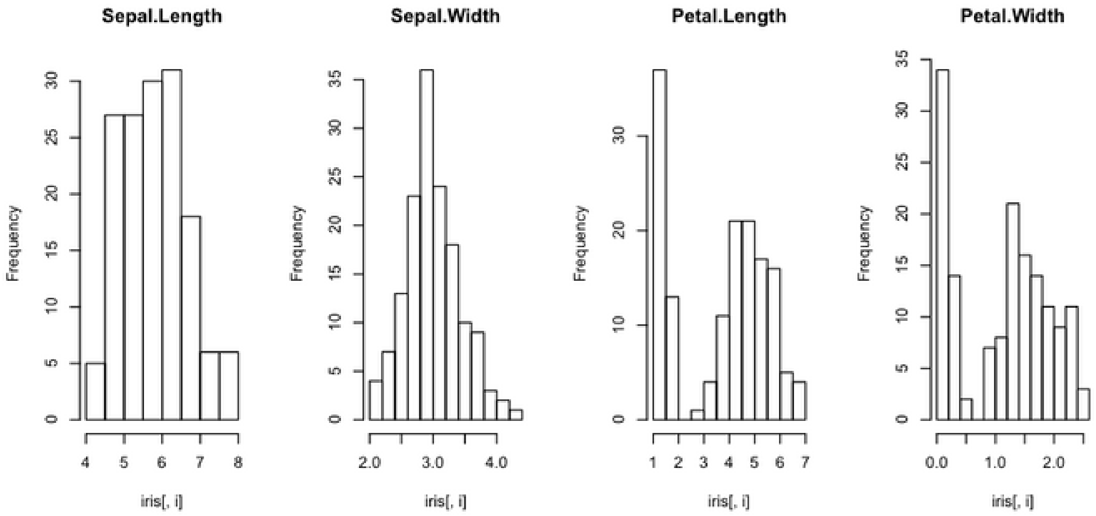


Figure 3.1: Histogram Plot in R

3.3.2 Density Plots

We can smooth out the histograms to lines using a density plot. These are useful for a more abstract depiction of the distribution of each variable.

```
# load packages
library(lattice)
# load dataset
data(iris)
#  create  a  layout  of  simpler  density  plots  by
attribute par(mfrow=c(1,4))
for(i in 1:4) {
plot(density(iris[,i]), main=names(iris)[i])
}
```

Listing 3.2: Calculate density plots.

Using the same dataset from the previous example with histograms, we can see the double Gaussian distribution with petal measurements. We can also see a possible exponential (Lapacian-like) distribution for the Sepal width.

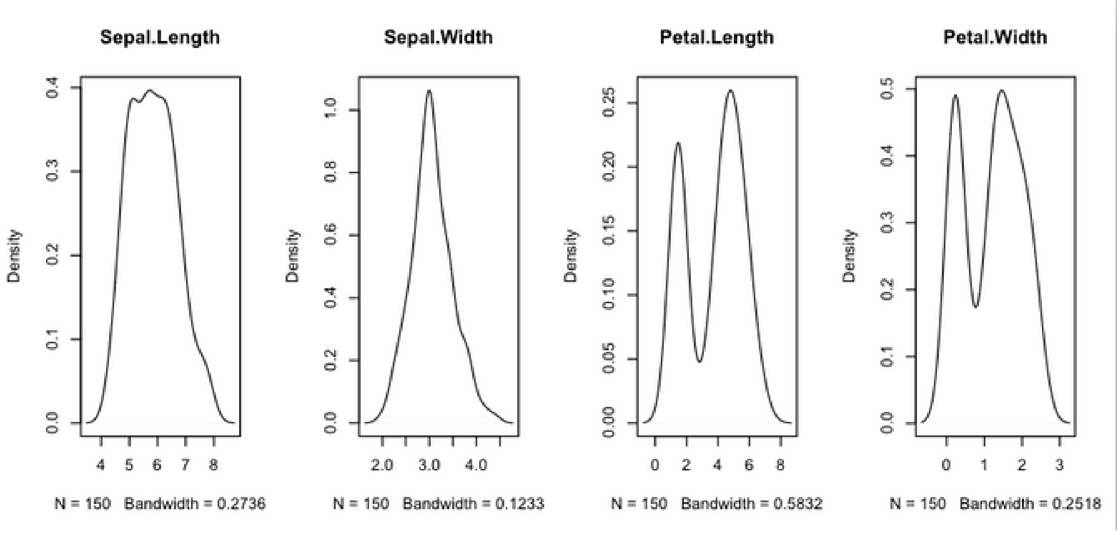


Figure 3.2: Density Plots in R

3.3.3 Box And Whisker Plots

We can look at the distribution of the data a different way using box and whisker plots. The box captures the middle 50% of the data, the line shows the median and the whiskers of the plots show the reasonable extent of data. Any dots outside the whiskers are good candidates for outliers.

```
# load dataset
data(iris)
# Create separate boxplots for each
attribute par(mfrow=c(1,4))
for(i in 1:4) {
boxplot(iris[,i], main=names(iris)[i])
}
```

Listing 3.3: Calculate box and whisker plots.

We can see that the data all has a similar range (and the same units of centimeters). We can also see that Sepal width may have a few outlier values for this data sample.

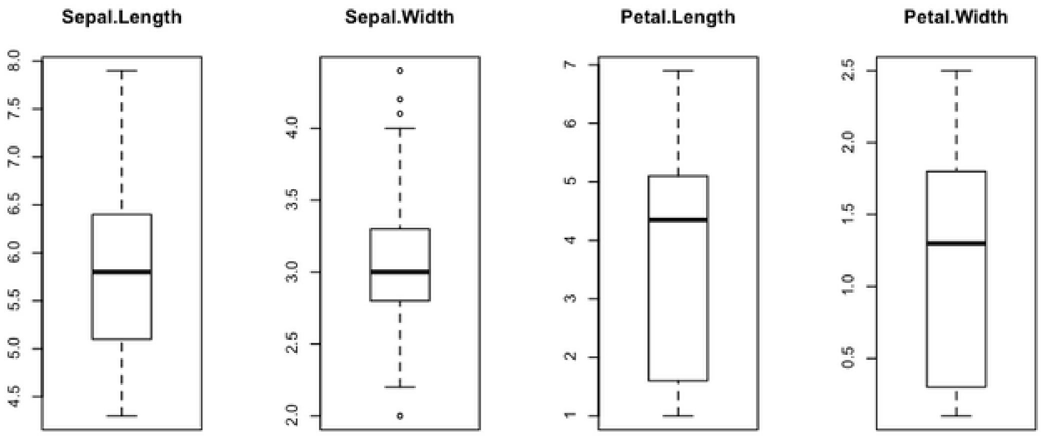


Figure 3.3: Box and Whisker Plots in R

3.3.4 Bar Plots

In datasets that have categorical rather than numeric attributes, we can create bar plots that give an idea of the proportion of instances that belong to each category.

```
# load the package
library(mlbench)
# load the dataset
data(BreastCancer)
# create a bar plot of each categorical attribute
par(mfrow=c(2,4))
for(i in 2:9) {
counts <- table(BreastCancer[,i])
name <- names(BreastCancer)[i]
barplot(counts, main=name)
}
```

Listing 3.4: Calculate bar plots.

We can see that some plots have a good mixed distribution and others show a few labels with the overwhelming number of instances.

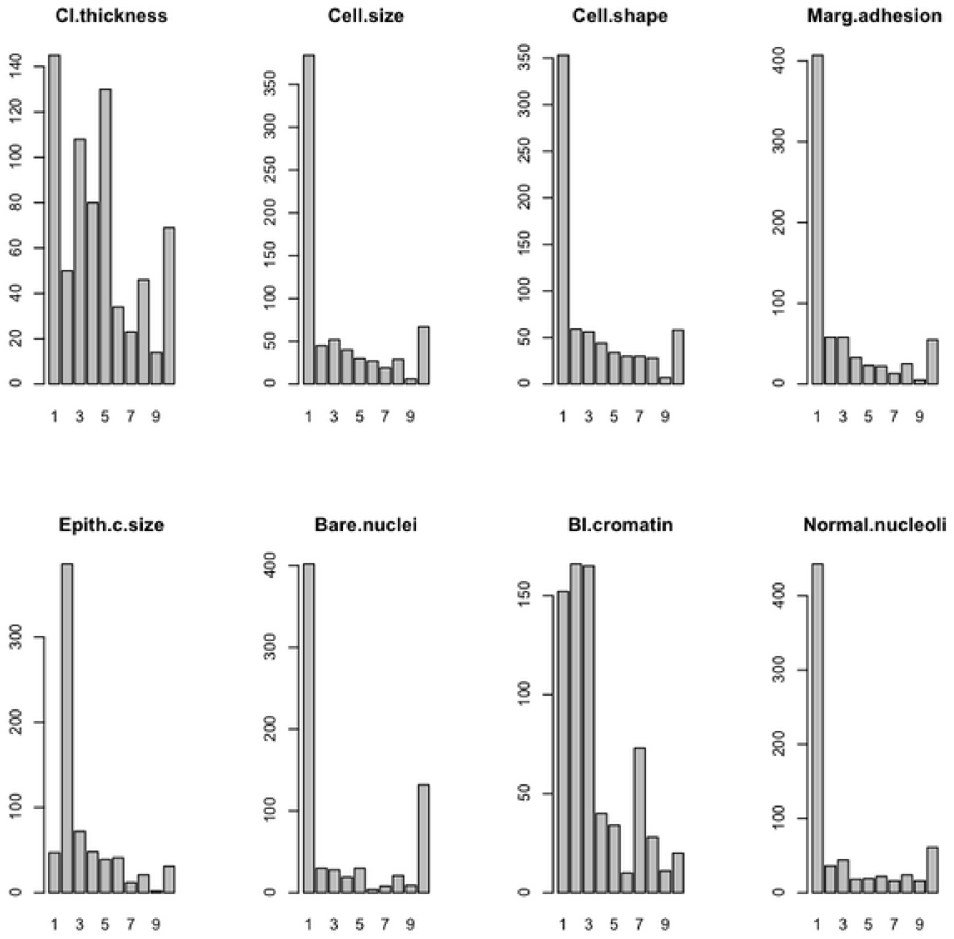


Figure 3.4: Bar Plots in R

3.3.5 MissingPlot

Missing data can have a big impact on modeling. Some techniques ignore missing data, others break. You can use a missing plot to get a quick idea of the amount of missing data in your dataset. The x-axis shows attributes and the y-axis shows instances. Horizontal lines indicate missing data for an instance, vertical blocks represent missing data for an attribute.

You may need to install the Amelia package. Refer to Section ?? for help installing packages.

```
# load packages
library(Amelia)
library(mlbench)
# load dataset
data(Soybean)
# create a missing map
missmap(Soybean, col=c("black", "grey"), legend=FALSE)
```

Listing 3.5: Calculate missing plot.

We can see that some instances have a lot of missing data across some or most of the attributes.

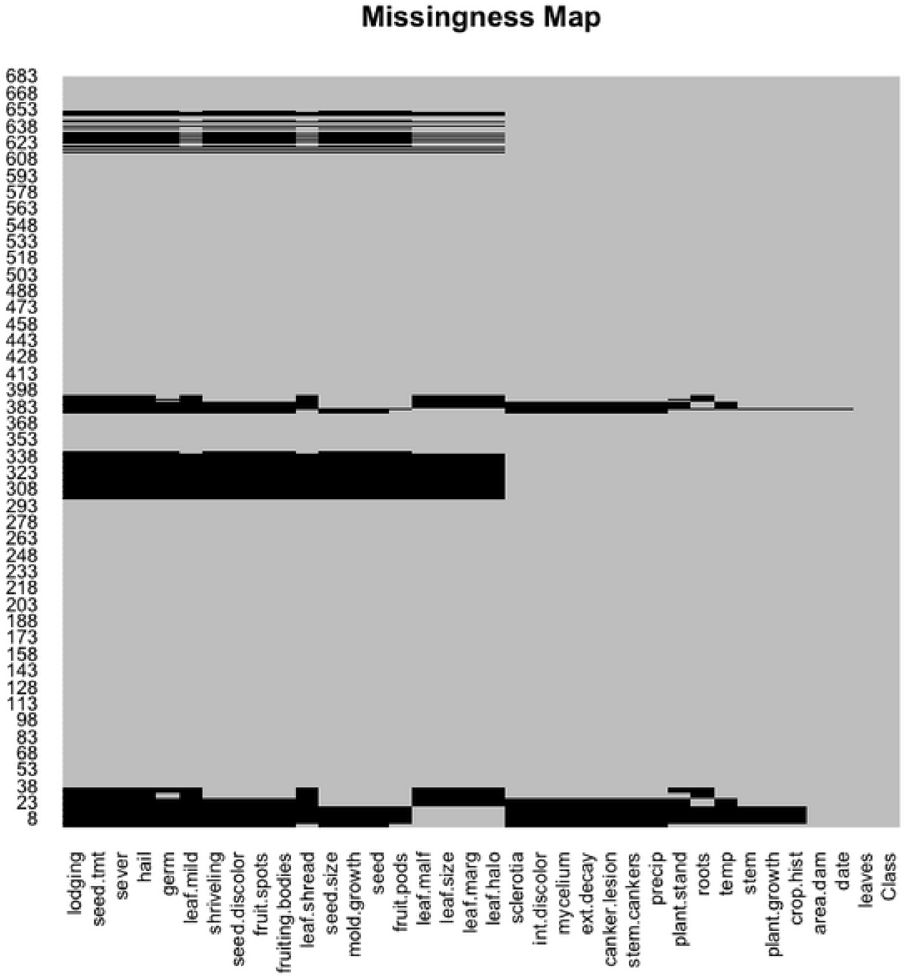


Figure 3.5: Missing Map in R

3.4 MultivariateVisualization

Multivariate plots are plots of the relationship or interactions between attributes. The goal is to learn something about the distribution, central tendency and spread over groups of data, typically pairs of attributes.

3.4.1 Correlation Plot

We can calculate the correlation between each pair of numeric attributes. These pairwise correlations can be plotted in a correlation matrix to given an idea of which attributes change together.

You may need to install the corrplot package. Refer to Section ?? for help installing packages.

```
# load package
library(corrplot)
# load the data
data(iris)
```

```
# calculate correlations
correlations <- cor(iris[,1:4])
# create correlation plot
corrplot(correlations, method="circle")
```

Listing 3.6: Calculate correlation plot.

A dot-representation was used where blue represents positive correlation and red negative. The larger the dot the larger the correlation. We can see that the matrix is symmetrical and that the diagonal attributes are perfectly positively correlated (because it shows the correlation of each attribute with itself). We can see that some of the attributes are highly correlated.

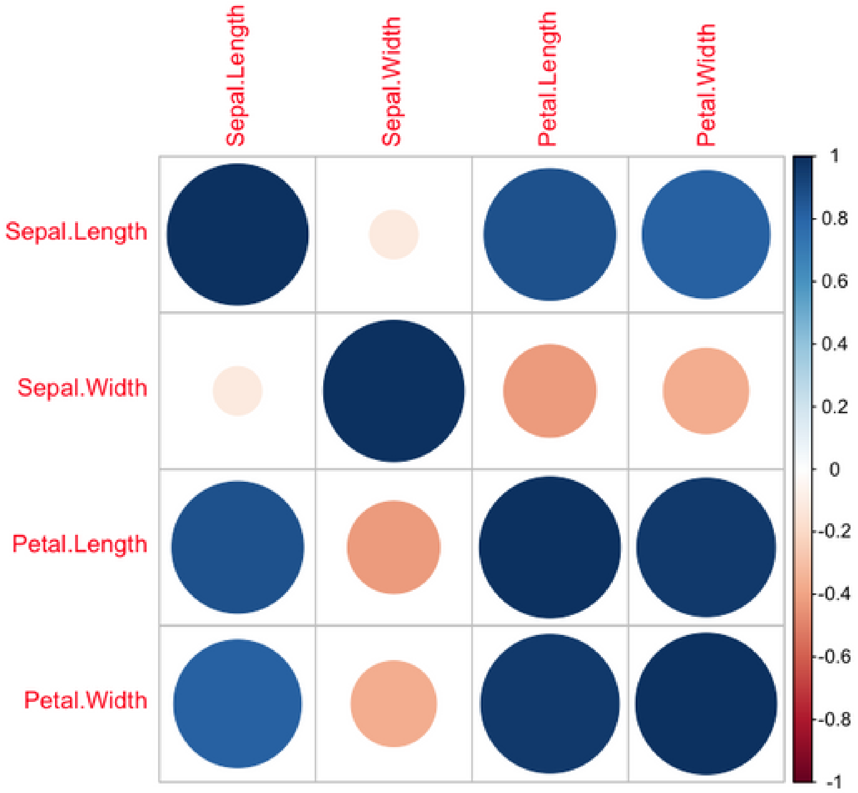


Figure 3.6: Correlation Matrix Plot in R

3.4.2 Scatter Plot Matrix

A scatter plot plots two variables together, one on each of the x- and y-axes with points showing the interaction. The spread of the points indicates the relationship between the attributes. You can create scatter plots for all pairs of attributes in your dataset, called a scatter plot matrix.

```
# load the data
data(iris)
# pairwise scatter plots of all 4 attributes
pairs(iris)
```

Listing 3.7: Calculate a scatter plot matrix.

Note that the matrix is symmetrical, showing the same plots with axes reversed. This aids in looking at your data from multiple perspectives. Note the linear (diagonal line) relationship between petal length and width.

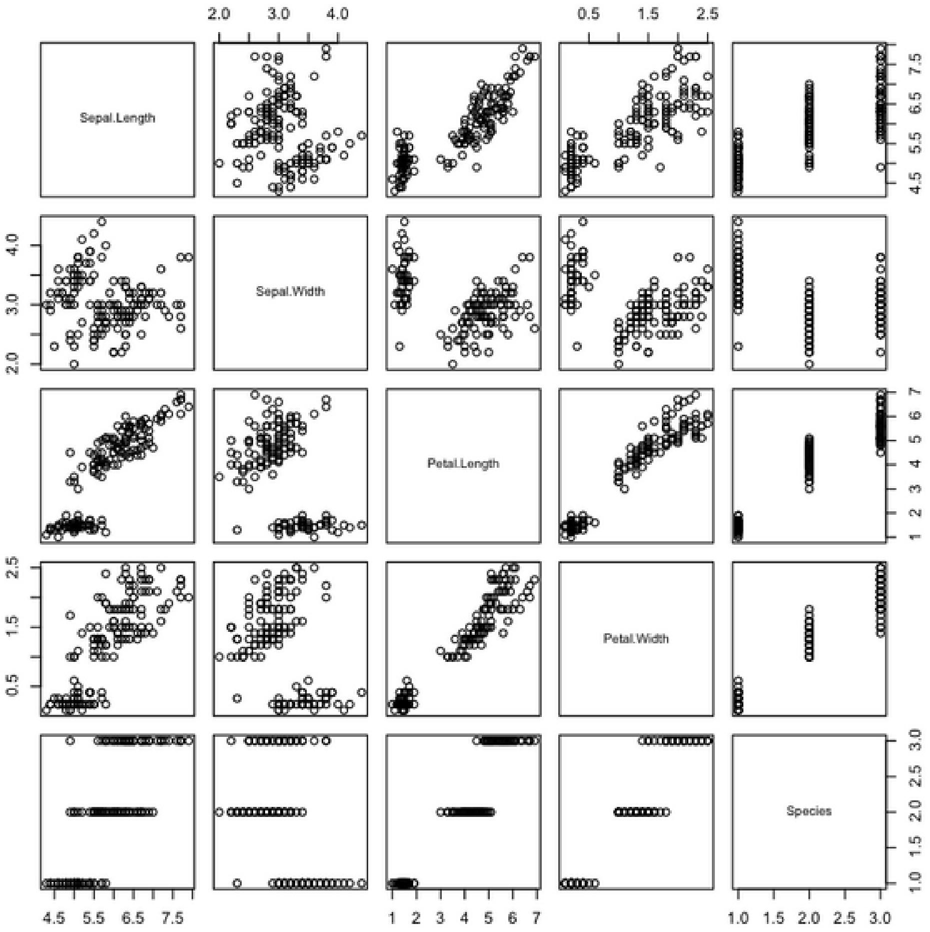


Figure 3.7: Scatter plot Matrix Plot in R

3.4.3 Scatter plot Matrix By Class

The points in a scatter plot matrix can be colored by the class label in classification problems. This can help to spot clear (or unclear) separation of classes and perhaps give an idea of how difficult the problem may be.

```
# load the data
data(iris)
# pairwise scatter plots colored by class
pairs(Species~., data=iris, col=iris$Species)
```

Listing 3.8: Calculate a scatter plot matrix by class.

Note the clear separation of the points by class label on most pairwise plots.

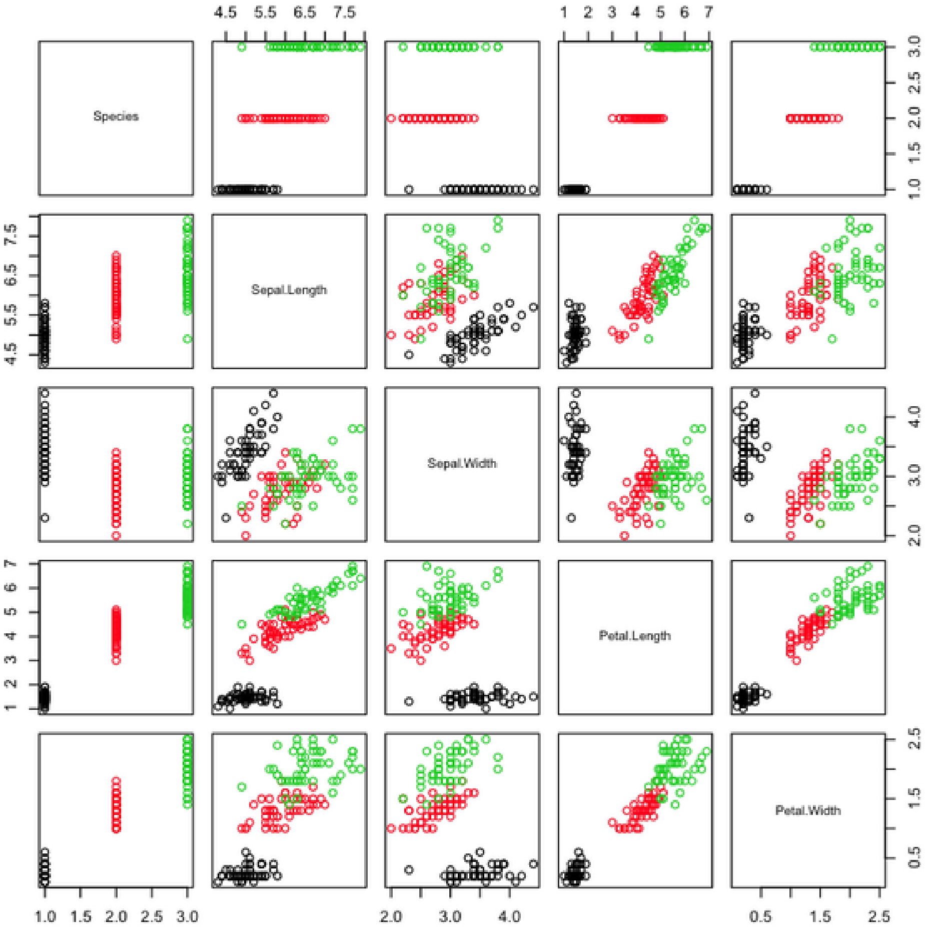


Figure 3.8: Scatter plot Matrix Plot By Class in R

3.4.4 DensityPlotsByClass

We can review the density distribution of each attribute broken down by class value. Like the scatter plot matrix, the density plot by class can help see the separation of classes. It can also help to understand the overlap in class values for an attribute.

```
# load the package
library(caret)
# load the data
data(iris)
# density plots for each attribute by class value
x <- iris[,1:4]
y <- iris[,5]
scales    <-    list(x=list(relation="free"),    y=list(relation="free"))
featurePlot(x=x, y=y, plot="density", scales=scales)
```

Listing 3.9: Calculate density plots by class.

We can see that some classes do not overlap at all (e.g. Petal Length) where as with other attributes there are hard to tease apart (e.g. Sepal Width).

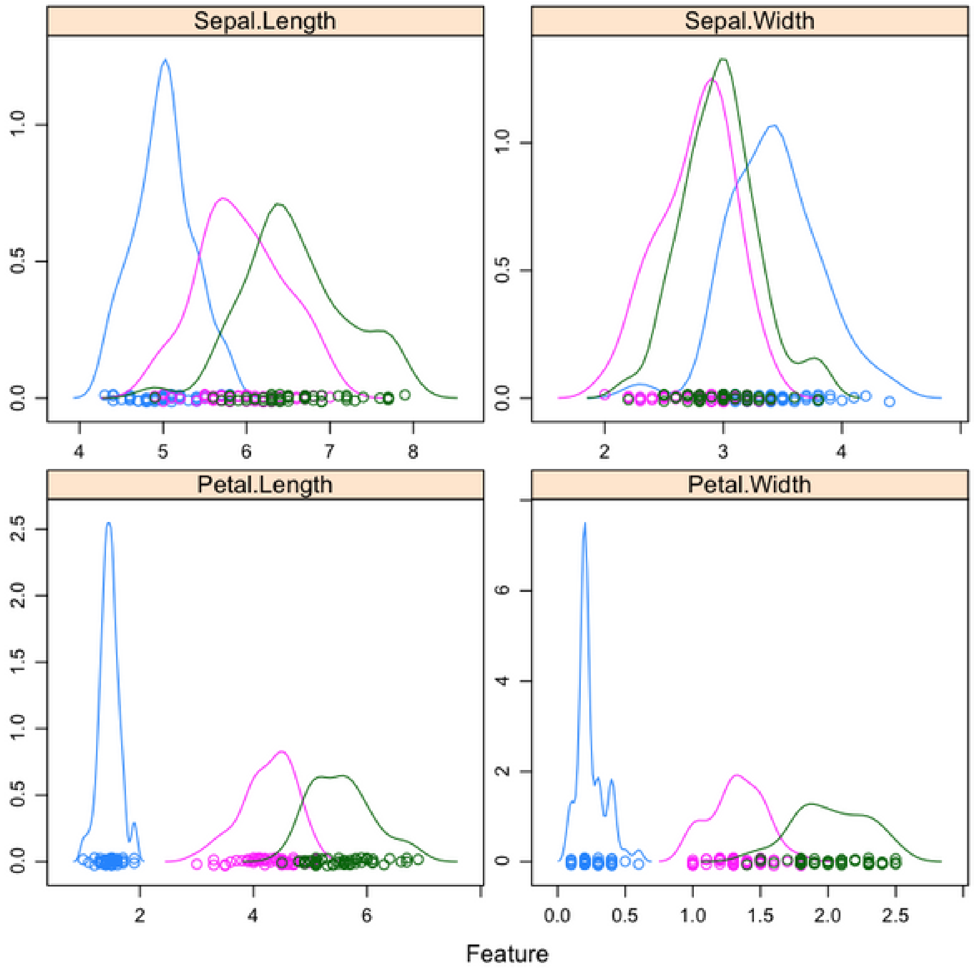


Figure 3.9: Density Plot By Class in R

3.4.5 Box And Whisker Plots By Class

We can also review the boxplot distributions of each attribute by class value. This too can help in understanding how each attribute relates to the class value, but from a different perspective to that of the density plots.

```
# load the package
library(caret)
# load the iris dataset
data(iris)
# box and whisker plots for each attribute by class
value x <- iris[,1:4]
y <- iris[,5]
featurePlot(x=x, y=y, plot="box")
```

Listing 3.10: Calculate box and whisker plots by class.

These plots help to understand the overlap and separation of the attribute-class groups. We can see some good separation of the Setosa class for the Petal Length attribute.

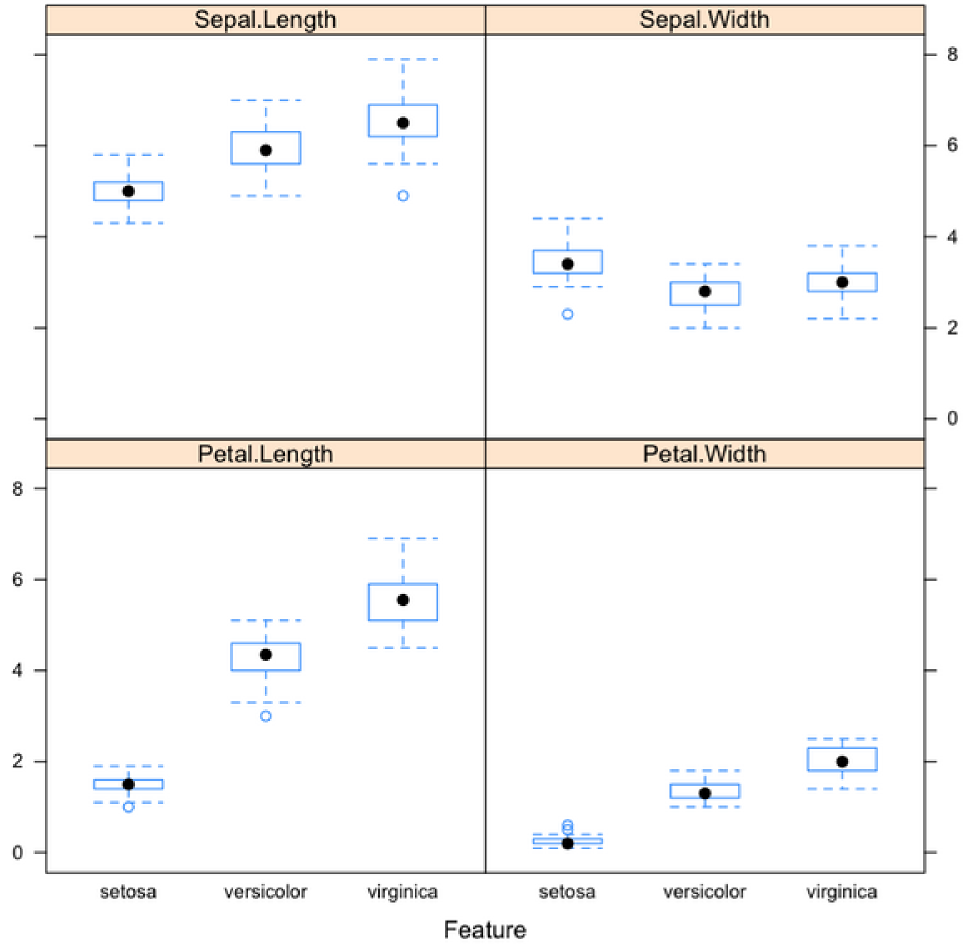


Figure 3.10: Box and Whisker Plots By Class in R

3.5 Tips For Data Visualization

Review Plots. Actually take the time to look at the plots you have generated and think about them. Try to relate what you are seeing to the general problem domain as well as specific records in the data. The goal is to learn something about your data, not to generate a plot.

Ugly Plots, Not Pretty. Your goal is to learn about your data not to create pretty visualizations. Do not worry if the graphs are ugly. You a not going to show them to anyone.

Write Down Ideas. You will get a lot of ideas when you are looking at visualizations of your data. Ideas like data splits to look at, transformations to apply and techniques to test. Write them all down. They will be invaluable later when you are struggling to think of more things to try to get better results.

3.6 Summary

In this lesson you discovered the importance of data visualization in order to better understand your data. You discovered a number of methods that you can use to both visualize and improve your understanding of the attributes in your data using univariate plots and their interactions using multivariate plots.

Univariate Plots: Histograms, Density Plots, Box And Whisker Plots, Bar Plots and

Missing Plot

Multivariate Plots: Correlation Plot, Scatter Plot Matrix, Scatter Plot Matrix By Class, Density By Class and Box And Whisker Plots By Class.

3.6.1 NextStep

You have now seen two ways that you can use to learn more about your data: data summarization and data visualization. In the next lesson you will start to use this understanding and pre-process your data in order to best expose the structure of the problem to the learning algorithms.