Introduction to Machine Learning | 19

by **Patrick Boily**

Data scientists are often introduced to the field *via* machine learning concepts, algorithms and applications, which we introduce in this chapter.

In Chapters 20, 21, and 22, we will discuss other technical aspects of machine learning, as well as more sophisticated algorithms (abundant details can also be found in [2–5, 191], among others).

19.1 Preliminaries

> "Data is not information, information is not knowledge, knowledge is not understanding, understanding is not wisdom." (attributed to C. Stoll [192])

One of the challenges of working in the **data science** (DS), **machine learning** (ML), and **artificial intelligence** (AI) fields is that nearly all quantitative work can be described with some combination of the terms DS/ML/AI (often to a ridiculous extent). As such, it can be difficult to differentiate the discipline from other quantitative fields, which makes studying and learning it properly harder than it should.

Robinson [166] suggests that their relationships follow an inclusive **hierarchical** structure:

- in a first stage, **DS** provides "insights" *via* visualization and (manual) inferential analysis;
- in a second stage, **ML** yields "predictions" (or "advice"), while reducing the operator's analytical, inferential and decisional workload (although it is still present to some extent), and
- in the final stage, **AI** removes the need for oversight, allowing for automatic "actions" to be taken by a completely unattended system.

The goals of AI are laudable in an academic setting, but in practice, we believe that stakeholders should not seek to give up their agency in the decision-making process; as such, we follow the lead of various thinkers and suggest further splitting AI into "general AI" (which we will not be pursuing further at this stage) and "**augmented** intelligence".[1]

With this in mind, our definition of the DS/ML/AI approach is that it consists of quantitative processes[2] that can help users **learn actionable insights** about their situation without completely abdicating their decision-making responsibility.

In this chapter, we will take a brief look at:

[1] Which can be viewed as ML "on steroids".

[2] What H. Mason has called "the **working intersection** of statistics, engineering, computer science, domain expertise, and "hacking" [167].

- the **fundamentals** of data science (see Section 19.2);
- **association rules** mining (see Section 19.3);
- **supervised learning** and **classification**, with a focus on **decision trees** (see Section 19.4);
- **unsupervised learning** and **clustering**, with a focus on k**−means** (see Section 19.5), and
- some of the common **issues** and **challenges** encountered during the data science and machine learning process (see Section 19.6).

19.2 Statistical Learning

> "We learn from failure, not from success!" (B. Stoker, *Dracula*)

As humans, we learn (at all stages) by first taking in our environment, and then by:

- answering questions about it;
- testing hypotheses;
- creating concepts;
- making predictions;
- creating categories, and
- classifying and grouping its various objects and attributes.

In a way, the main concept of DS/ML/AI is to try to teach our machines (and thus, ultimately, ourselves) to glean insight from data, and how to do this properly and efficiently, free of biases and pre-conceived notions – in other words, **can we design algorithms that can learn?**[3]

In that context, the simplest DS/ML/AI method is **exploring the data** (or a representative sample) to:

- provide a summary through basic statistics – mean, variance, histograms, etc.;
- make its multi-dimensional structure evident through data visualization, and
- look for consistency, considering what is in there and what is missing.

19.2.1 Types of Learning

In the **statistical learning** context,[4] more sophisticated approaches traditionally fall into a **supervised** or an **unsupervised** learning framework.

Supervised learning is akin to "learning with a teacher." Typical tasks include **classification**, **regression**, **rankings**, and **recommendations**.

In supervised learning, algorithms use **labeled training data** to build (or train) a predictive model;[5] each algorithm's performance is evaluated using **test data** for which the label is known but not used in the prediction.[6]

In supervised learning, there are fixed **targets** against which to train the model (such as age categories, or plant species) – the categories (and their number) are known prior to the analysis.

[3] Note that this is not the same thing as asking whether we *should* design such algorithms.

[4] A term sometimes used to describe general DS/ML/AI approaches, for no particular reason other than letting the audience know that the user has a background in mathematics and statistics.

[5] For instance, students may need to answer each exam question based on what they learned from worked-out examples provided by the teacher/textbook.

[6] The teacher provides the correct answers and marks the exam questions using the key, to continue the example.

Unsupervised learning, on the other hand, is akin to "self-learning by grouping similar exercises together as a study guide." Typical tasks include **clustering**, **association rules discovery**, **link profiling**, and **anomaly detection**. Unsupervised algorithms use **unlabeled data** to find natural patterns in the data; the drawback is that accuracy **cannot be evaluated** with the same degree of satisfaction.[7]

In unsupervised learning, we don't know what the target is, or even if there is one – we are simply looking for **natural groups** in the data.[8]

Other Learning Frameworks Some data science techniques fit into both camps; others can be either supervised or unsupervised, depending on how they are applied, but there are other conceptual approaches, especially for AI tasks:

- **semi-supervised learning** in which some data points have labels but most do not, which may occur when acquiring data is costly;[9]
- **reinforcement learning**, where an agent attempts to collect as much (short-term) reward as possible while minimizing (long-term) regret.[10]

19.2.2 DS and ML Tasks

Outside of academia, DS/ML/AI methods are only really interesting when they help ask and answer useful questions. Compare, for instance:

- **Analytics** – "How many clicks did this link get?"
- **Data Science** – "Based on the previous history of clicks on links of this publisher's site, can I predict how many people from Manitoba will read this specific page in the next three hours?" or "Is there a relationship between the history of clicks on links and the number of people from Manitoba who will read this specific page?"
- **Quantitative Methods** – "We have no similar pages whose history could be consulted to make a prediction, but we have reasons to believe that the number of hits will be strongly correlated with the temperature in Winnipeg. Using the weather forecast over the next week, can we predict how many people will access the specific page during that period?"

Data science and machine learning models are usually **predictive** (not **explanatory**): they show connections, and exploit correlations to make predictions, but they don't reveal why such connections exist.

Quantitative methods, on the other hand, usually assume a certain level of causal understanding based on various **first principles**. That distinction is not always understood properly by clients and consultants alike.

Common data science tasks include [168]:

- **classification** and **probability estimation** – which undergraduates are likely to succeed at the graduate level?
- **value estimation** – how much is a given client going to spend at a restaurant?
- **similarity matching** – which prospective clients are most similar to a company's established best clients?

[7] The teacher would not be involved in the discovery process, say, and the students might end up with different groupings, as an example.

[8] Perhaps junior students who like literature, have longish hair, and know how to cook *vs.* students who are on a sports team and have siblings *vs.* financial professionals with a penchant for superhero movies, craft beer and Hello Kitty backpack *vs.* ...

[9] The teacher could provide worked-out examples and a list of unsolved problems to try out; the students try to find similar groups of unsolved problems and compare them with the solved problems to find close matches.

[10] Embarking on a Ph.D. with an advisor, with all of the highs and the lows (and **maybe** a diploma at the end of the process?).

- **clustering** – do signals from a sensor form natural groups?
- **association rules discovery** – what books are commonly purchased together at an online retailer?
- **profiling** and **behaviour description** – what is the typical cell phone usage of a certain customer segment?
- **link prediction** – J. and K. have 20 friends in common: perhaps they'd be great friends?

A classic example is provided by the UCI Machine Learning Repository *Mushroom Dataset* [193]. Consider *Amanita muscaria* (commonly known as the fly agaric), a specimen of which is shown below.

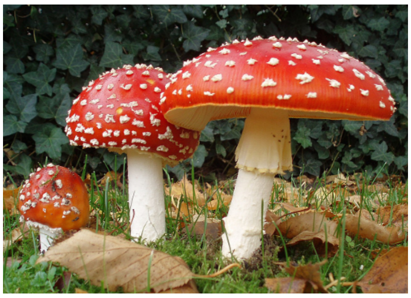


Figure 19.1: *Amanita muscaria* (fly agaric), in the wild. Does it look dangerous to you?

Is it **edible**, or **poisonous**? There is a simple way to get an answer – eat it, wait, and see: if you do not die or get sick upon ingestion, it was **edible**; otherwise it was **poisonous**.

But this test in unappealing for various reasons, however. Apart from the obvious risk of death, we might not learn much from the experiment; it is possible that this specific specimen was poisonous due to some mutation or some other factor (or that the ingester had a pre-existing condition which combined with the fungus to cause discomfort, etc.), and that fly agaric is actually edible in general (unlikely, but not impossible).

A predictive model, which would use features of a vast collection of mushroom species and specimens (including their class) could help shed light on the matter: what do poisonous mushrooms have in common? What properties do edible mushrooms share?[11]

For instance, let's say that *Amanita muscaria* has the following features:

- **habitat**: woods;
- **gill size**: narrow;
- **spores**: white;
- **odor**: none,
- **cap color**: red.

We do not know **a priori** whether it is poisonous or edible. Is the available information sufficient to answer the question? Not on its own, no.[12]

But we could use **past data**, with correct **edible** or **poisonous** labels and the **same set of predictors** to build various supervised **classification** models to attempt to answer the question.

A simple form of such model, a **decision tree**, is shown in Figure 19.2.

The model prediction for *Amanita muscaria* follows the **decision path** shown in Figure 19.3.

[11] Note that this is not the same as understanding **why** a mushroom is poisonous or edible – the data alone cannot provide an answer to that question.

[12] A mycologist could perhaps deduce the answer from these features alone, but she would be using her experience with fungi to make a prediction, and so would not be looking at the features in a *vacuum*.

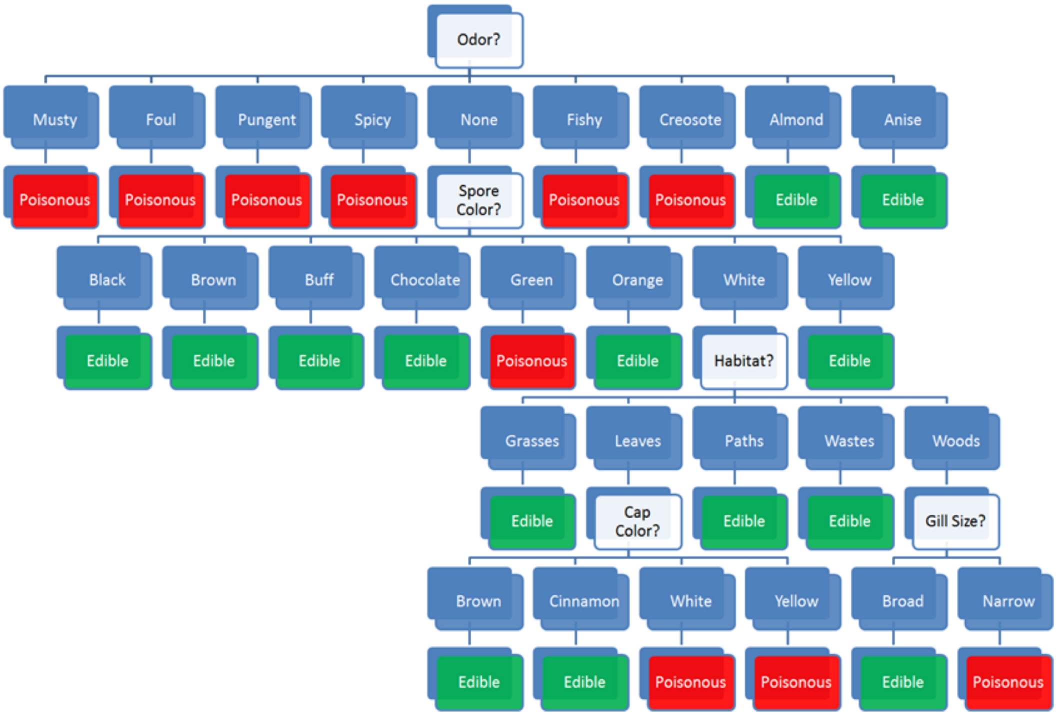


Figure 19.2: Decision tree for the mushroom classification problem [author unknown].

1. some mushroom **odors** (musty, spicy, etc.) are associated with poisonous mushrooms, some (almond, anise) with edible mushrooms, but there are mushrooms with no specific odor in either category – for mushroom with 'no odor' (as with *Amanita muscaria*), odor does not provide enough information for proper classification and we need to incorporate additional features into the decision path;
2. among mushrooms with no specific odor, some **spore colours** (black, etc.) are associated with edible mushrooms, some (almond, anise) with poisonous mushrooms, but there are mushrooms with 'white' spores in either category – the combination 'no odor and white spores' does not provide enough information to classify *Amanita muscaria* and we need to incorporate additional features into the decision path;
3. among mushrooms of no specific odor with white spores, some **habitats** (grasses, paths, wastes) are associated with edible mushrooms, but there are mushrooms in either category that are found in the 'woods' – the combination 'no odor, white spores, found in the woods' does not provide enough information to classify *Amanita muscaria* and we need to incorporate additional features into the decision path,
4. among white-spored forest mushroom with no specific odor, a broad **gill size** is associated with edible mushrooms, whereas a 'narrow' gill size is associated with poisonous mushrooms – as *Amanita muscaria* is a narrow-gilled, white-spored forest mushroom with no specific odor, the decision path predicts that it is **poisonous**.

Note that the **cap color** does not affect the decision path, however.[13]

But the decision tree model **does not explain why** this particular combinations of features is associated with poisonous mushrooms – the decision path is not **causal**.

[13] It would have had *Amanita muscaria*'s habitat been 'leaves'.

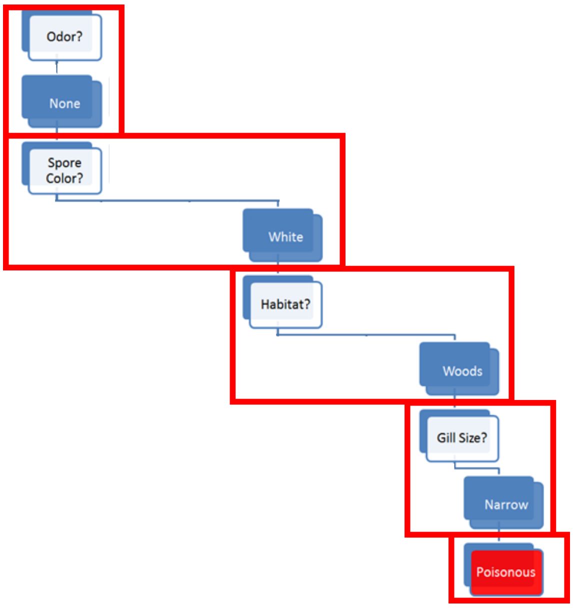


Figure 19.3: Decision path for *Amanita muscaria*.

At this point, a number of questions naturally arise:

- Would you have trusted an **edible** prediction?
- How are the features measured?
- What is the true cost of making a mistake?
- Is the data on which the model is built representative?
- What data is required to build trustworthy models?
- What do we need to know about the model in order to **trust it**?

The stage has now been set: this mushroom classification problem has all the hallmarks of a ML problem. Keep it in mind as a representative of the discipline in the sections that follow.

19.3 Association Rules Mining

> "Correlation isn't causation. But it's a big hint." (E. Tufte)

19.3.1 Overview

Association rules discovery is a type of unsupervised learning that finds **connections** among the attributes (variables) and levels (values), and combinations thereof, of a dataset's observations. For instance, we might analyze a (hypothetical) dataset on the physical activities and purchasing habits of North Americans and discover that

- runners who are also triathletes (the **premise**) tend to drive Subarus, drink microbrews, and use smart phones (the **conclusion**), or
- individuals who have purchased home gym equipment are unlikely to be using it 1 year later, say.

But the presence of a **correlation** between the premise and the conclusion does not necessarily imply the existence of a **causal relationship** between them. It is rather difficult to "demonstrate" causation *via* data analysis; in practice, decision-makers pragmatically (and often erroneously) focus on the second half of Tufte's rejoinder, which basically asserts that "there's no smoke without fire."

Case in point, while being a triathlete does not cause one to drive a Subaru, Subaru Canada thinks that the connection is strong enough to offer to reimburse the registration fee at an IRONMAN 70.3 competition (since at least 2018)! [194]

Market Basket Analysis Association rules discovery is also known as **market basket analysis** after its original application, in which supermarkets record the contents of shopping carts (the **baskets**) at check-outs to determine which items are frequently purchased together.

For instance, while bread and milk might often be purchased together, that is unlikely to be of interest to supermarkets given the frequency of market baskets containing milk **or** bread.[14]

Knowing that a customer has purchased bread does provide some information regarding whether they also purchased milk, but the individual probability that each item is found, separately, in the basket is so high to begin with that this insight is unlikely to be useful.

If 70% of baskets contain milk and 90% contain bread, say, we would expect **at least**

$$90\% \times 70\% = 63\%$$

of all baskets to contain milk **and** bread, should the presence of one in the basket be **totally independent** of the presence of the other.

If we then observe that 72% of baskets contain both items (a 1.15-fold increase on the expected proportion, assuming there is no link), we would conclude that there was at best a **weak correlation** between the purchase of milk and the purchase of bread.

Sausages and hot dog buns, on the other hand, which we might suspect are not purchased as frequently as milk and bread, might still be purchased as a pair more often than one would expect given the frequency of baskets containing sausages **or** buns.

If 10% of baskets contain sausages, and 5% contain buns, say, we would expect that

$$10\% \times 5\% = 0.5\%$$

of all baskets would contain sausages **and** buns, should the presence of one in the basket be **totally independent** of the presence of the other.

If we then observe that 4% of baskets contain both items (an 8-fold increase on the expected proportion, assuming there is no link), we would obviously conclude that there is a **strong correlation** between the purchase of sausages and the purchase of hot dog buns.

It is not too difficult to see how this information could potentially be used to help supermarkets turn a profit: announcing or advertising a sale on sausages while **simultaneously** (and quietly) raising the price of buns could have the effect of bringing in a higher number of customers into the store, increasing the sale volume for both items while keeping the combined price of the two items constant.[15]

[14] In the mathematical sense of "or": one, or the other, or both.

[15] The marketing team is banking on the fact that customers are unlikely to shop around to get the best deal on hot dogs AND buns, which may or may not be a valid assumption.

A (possibly) apocryphal story shows the limitations of association rules: a supermarket found an association rule linking the purchase of beer and diapers and consequently moved its beer display closer to its diapers display, having confused correlation and causation.

Purchasing diapers does not cause one to purchase beer (or *vice-versa*); it could simply be that parents of newborns have little time to visit public houses and bars, and whatever drinking they do will be done at home. Who knows? Whatever the case, rumour has it that the experiment was neither popular nor successful.

Applications Typical uses include:

- finding **related concepts** in text documents – looking for pairs (triplets, etc) of words that represent a joint concept: {San Jose, Sharks}, {Michelle, Obama}, etc.;
- detecting **plagiarism** – looking for specific sentences that appear in multiple documents, or for documents that share specific sentences;
- identifying **biomarkers** – searching for diseases that are frequently associated with a set of biomarkers;
- making predictions and decisions based on association rules (there are pitfalls here);
- altering circumstances or environment to take advantage of these correlations (suspected causal effect);
- using connections to modify the likelihood of certain outcomes (see immediately above);
- imputing missing data,
- text autofill and autocorrect, etc.

Other uses and examples can be found in [163, 195, 196].

Causation and Correlation Association rules can automate **hypothesis discovery**, but one must remain correlation-savvy.[16]

If attributes A and B are shown to be correlated in a dataset, there are four possibilities:

- A and B are correlated entirely by chance in this particular dataset;
- A is a relabeling of B (or *vice-versa*);
- A causes B (or *vice-versa*), or
- some combination of attributes $C_1, \ldots, C_n$ (which may not be available in the dataset) cause both A and B.

Siegel [195] illustrates the confusion that can arise with a number of real-life examples:

- Walmart has found that sales of strawberry Pop-Tarts increase about seven-fold in the days preceding the arrival of a hurricane;
- Xerox employees engaged in front-line service and sales-based positions who use Chrome and Firefox browsers perform better on employment assessment metrics and tend to stay with the company longer, or
- University of Cambridge researchers found that liking "Curly Fries" on Facebook is predictive of high intelligence.

[16] Which remains less prevalent among quantitative specialists and data scientists than one might hope, unfortunately, in our experience.

It can be tempting to try to **explain** these results (again, from [195]). Perhaps:

- when faced with a coming disaster, people stock up on comfort or nonperishable foods;
- the fact that an employee takes the time to install another browser shows that they are an informed individual and that they care about their productivity, or
- an intelligent person liked this Facebook page first, and her friends saw it, and liked it too, and since intelligent people have intelligent friends (?), the likes spread among people who are intelligent.

While these explanations *might* very well be the right ones (although probably not in the last case), there is **nothing in the data** that supports them. Association rules discovery **finds** interesting rules, but it does not explain them. **The point cannot be over-emphasized**: correlation does not imply causation.

Analysts and consultants might not have much control over the matter, but they should do whatever is in their power so that the following headlines do not see the light of day:

- "Pop-Tarts" get hurricane victims back on their feet;
- Using Chrome of Firefox improves employee performance, or
- Eating curly fries makes you more intelligent.

Definitions A rule $X \to Y$ is a statement of the form "if X (the **premise**) then Y (the **conclusion**)" built from any logical combinations of a dataset attributes.

In practice, a rule **does not need to be true for all observations** in the dataset – there could be instances where the premise is satisfied but the conclusion is not.

In fact, some of the "best" rules are those which are only accurate 10% of the time, as opposed to rules which are only accurate 5% of the time, say. As always, **it depends on the context**. To determine a rule's strength, we compute various rule metrics, such as the:

- **support**, which measures the frequency at which a rule occurs in a dataset – low coverage values indicate rules that rarely occur;
- **confidence**, which measures the reliability of the rule: how often does the conclusion occur in the data given that the premises have occurred – rules with high confidence are "truer", in some sense;
- **interest**, which measures the difference between its confidence and the relative frequency of its conclusion – rules with high absolute interest are ... more interesting than rules with small absolute interest;
- **lift**, which measures the increase in the frequency of the conclusion which can be explained by the premises – in a rule with a high lift (> 1), the conclusion occurs more frequently than it would if it were independent of the premises, and
- **conviction** [197], **all-confidence** [198], **leverage** [199], **collective strength** [200], and many others [201, 202].

In a dataset with N observations, let $\text{Freq}(A) \in \{0, 1, \dots, N\}$ represent the count of the dataset's observations for which property A holds.

This is all the information that is required to compute a rule's evaluation metrics:

$$\begin{aligned}
\text{Support}(X \rightarrow Y) &= \frac{\text{Freq}(X \cap Y)}{N} \in [0,1] \\
\text{Confidence}(X \rightarrow Y) &= \frac{\text{Freq}(X \cap Y)}{\text{Freq}(X)} \in [0,1] \\
\text{Interest}(X \rightarrow Y) &= \text{Confidence}(X \rightarrow Y) - \frac{\text{Freq}(Y)}{N} \in [-1,1] \\
\text{Lift}(X \rightarrow Y) &= \frac{N^2 \cdot \text{Support}(X \rightarrow Y)}{\text{Freq}(X) \cdot \text{Freq}(Y)} \in (0, N^2) \\
\text{Conviction}(X \rightarrow Y) &= \frac{1 - \text{Freq(Y)}/N}{1 - \text{Confidence}(X \rightarrow Y)} \geq 0
\end{aligned}$$

British Music Dataset A simple example will serve to illustrate these concepts. Consider a (hypothetical) music dataset containing data for $N = 15,356$ British music lovers and a **candidate rule** RM:

> "If an individual is born before 1976 (X), then they own a copy of the Beatles' *Sergeant Peppers' Lonely Hearts Club Band*, in some format (Y)".

Let's assume further that

- $\text{Freq}(X) = 3888$ individuals were born before 1976;
- $\text{Freq}(Y) = 9092$ individuals own a copy of *Sergeant Peppers' Lonely Hearts Club Band*, and
- $\text{Freq}(X \cap Y) = 2720$ individuals were born before 1976 and own a copy of *Sergeant Peppers' Lonely Hearts Club Band*.

We can easily compute the 5 metrics for RM:

$$\begin{aligned}
\text{Support(RM)} &= \frac{2720}{15,356} \approx 18\% \\
\text{Confidence(RM)} &= \frac{2720}{3888} \approx 70\% \\
\text{Interest(RM)} &= \frac{2720}{3888} - \frac{9092}{15,356} \approx 0.11 \\
\text{Lift(RM)} &= \frac{15,356^2 \cdot 0.18}{3888 \cdot 9092} \approx 1.2 \\
\text{Conviction(RM)} &= \frac{1 - 9092/15,356}{1 - 2720/3888} \approx 1.36
\end{aligned}$$

These values are easy to interpret: RM occurs in **18%** of the dataset's instances, and it holds true in **70%** of the instances where the individual was born prior to 1976.

This would seem to make RM a **meaningful rule** about the dataset – being older and owning the album are linked properties. But if being younger and not owning that song are not also linked properties, the statement is actually weaker than it would appear at a first glance.

As it happens, RM's lift is **1.2**, which can be rewritten as

$$1.2 \approx \frac{0.70}{0.56},$$

i.e. 56% of younger individuals also own the song.

The ownership rates of the two age categories are different, but perhaps not as significantly as one would deduce using the confidence and support alone, which is reflected by the rule's "low" interest, whose value is **0.11**.

Finally, the rule's conviction is **1.36**, which means that the rule would be incorrect 36% more often if X and Y were completely independent.

All this seems to point to the rule RM being not entirely devoid of meaning, but to what extent, exactly? This is a difficult question to answer.[17]

It is nearly impossible to provide **hard** and **fast** thresholds: it always depends on the context, and on comparing evaluation metric values for a rule with the values obtained for some other of the dataset's rules. In short, evaluation of a lone rule is **meaningless**.

In general, it is recommended to conduct a **preliminary exploration** of the space of association rules (using domain expertise when appropriate) in order to determine reasonable threshold ranges for the specific situation; candidate rules would then be discarded or retained depending on these metric thresholds.

This requires the ability to "easily" generate potentially meaningful candidate rules.

19.3.2 Generating Rules

Given association rules, it is straightforward to evaluate them using various metrics, as discussed in the previous section.

The real challenge of association rules discovery lies in **generating** a set of candidate rules which are likely to be retained, without wasting time generating rules which are likely to be discarded.

An **itemset** (or instance set) for a dataset is a list of attributes and values. A set of **rules** can be created from the itemset by adding "IF ... THEN" blocks to the instances.

As an example, from the instance set

$$\{\text{membership} = \text{True}, \text{age} = \text{Youth}, \text{purchasing} = \text{Typical}\},$$

we can create the 7 following 3−item rules:

- IF (membership = True AND age = Youth) THEN purchasing = Typical;
- IF (age = Youth AND purchasing = Typical) THEN membership = True;
- IF (purchasing = Typical AND membership = True) THEN age = Youth;
- IF membership = True THEN (age = Youth AND purchasing = Typical);

[17] There will be times when an interest of 0.11 in a rule would be considered a smashing success; a lift of 15 would not be considered that significant but a support of 2% would be, and so forth.

- IF age = Youth THEN (purchasing = Typical AND membership = True);
- IF purchasing = Typical THEN (membership = True) AND age = Youth);
- IF ∅ THEN (membership = True AND age = Youth AND purchasing = Typical);

the 9 following 2−item rules:

- IF membership = True THEN age = Youth;
- IF age = Youth THEN purchasing = Typical;
- IF purchasing = Typical THEN membership = True;
- IF membership = True THEN purchasing = Typical;
- IF age = Youth THEN membership = True;
- IF purchasing = Typical THEN age = Youth;
- IF ∅ THEN (age = Youth AND purchasing = Typical);
- IF ∅ THEN (purchasing = Typical AND membership = True);
- IF ∅ THEN (membership = True) AND age = Youth);

and the 3 following 1−item rules:

- IF ∅ THEN age = Youth;
- IF ∅ THEN purchasing = Typical,
- IF ∅ THEN membership = True.

In practice, we usually only consider rules with the same number of items as there are members in the itemset: in the example above, for instance, the 2−item rules could be interpreted as emerging from the 3 separate itemsets

$$\begin{aligned}\{\text{membership} = \text{True}, \text{age} = \text{Youth}\}\\ \{\text{age} = \text{Youth}, \text{purchasing} = \text{Typical}\}\\ \{\text{purchasing} = \text{Typical}, \text{membership} = \text{True}\}\end{aligned}$$

and the 1−item rules as arising from the 3 separate itemsets

$$\{\text{membership} = \text{True}\}, \{\text{age} = \text{Youth}\}, \{\text{purchasing} = \text{Typical}\}.$$

Note that rules of the form $\varnothing \longrightarrow X$ (or IF ∅ THEN X) are typically denoted simply by X.

Now, consider an itemset $\mathscr{C}_n$ with n members (that is to say, n attribute/level pairs). In an n−item rule derived from $\mathscr{C}$, each of the n members appears either in the premise or in the conclusion; there are thus 2^n such rules, in principle.

The rule where each member is part of the premise (i.e., the rule without a conclusion) is nonsensical and is not allowed; we can derive exactly $2^n - 1$ n−item rules from $\mathscr{C}_n$. Thus, the **number of rules increases exponentially** when the **number of features increases linearly**.

This combinatorial explosion is a problem – it instantly disqualifies the **brute force** approach (simply listing all possible itemsets in the data and generating all rules from those itemsets) for any dataset with a realistic number of attributes.

How can we then generate a small number of **promising** candidate rules, in general?

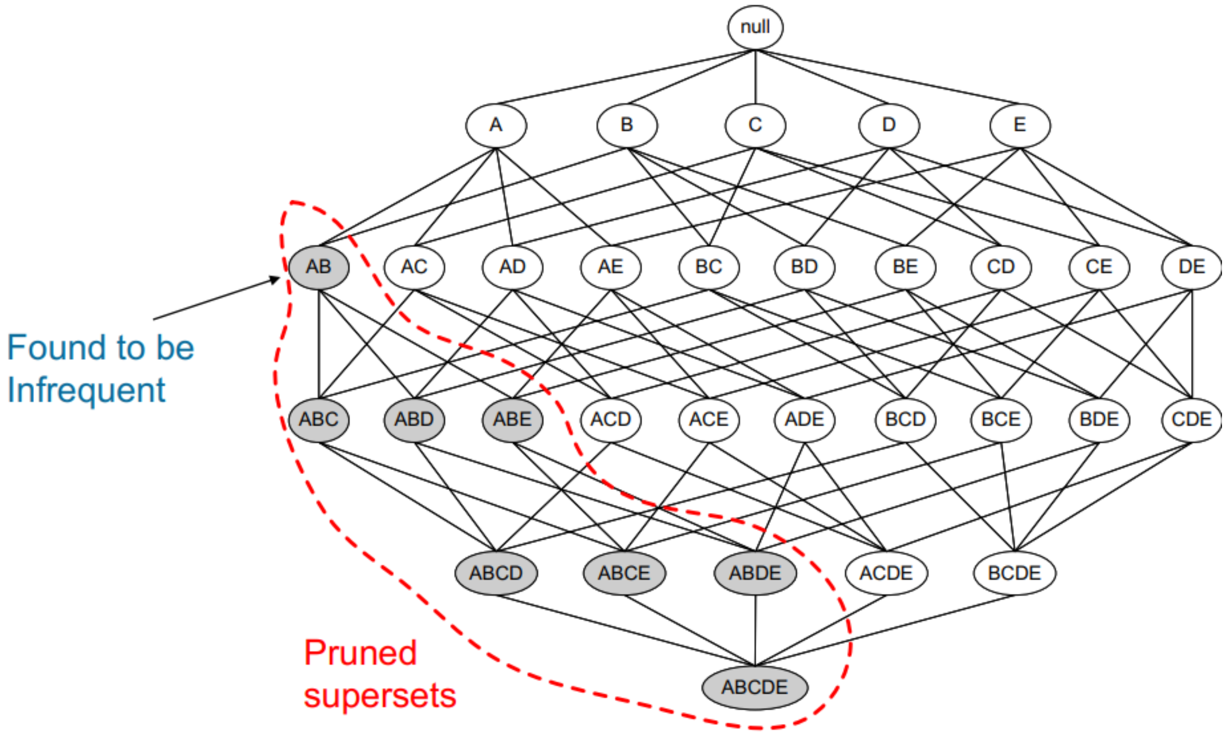


Figure 19.4: Pruned supersets of an infrequent itemset in the *a priori* network of a dataset with 5 items [203]; no rule would be generated from the grey itemsets.

19.3.3 The *A Priori* Algorithm

The ***a priori*** algorithm is an early attempt to overcome that difficulty. Initially, it was developed to work for **transaction data** (i.e. goods as columns, customer purchases as rows), but every reasonable dataset can be transformed into a transaction dataset using dummy variables.

The algorithm attempts to find **frequent itemsets** from which to build candidate rules, instead of building rules from **all** possible itemsets.

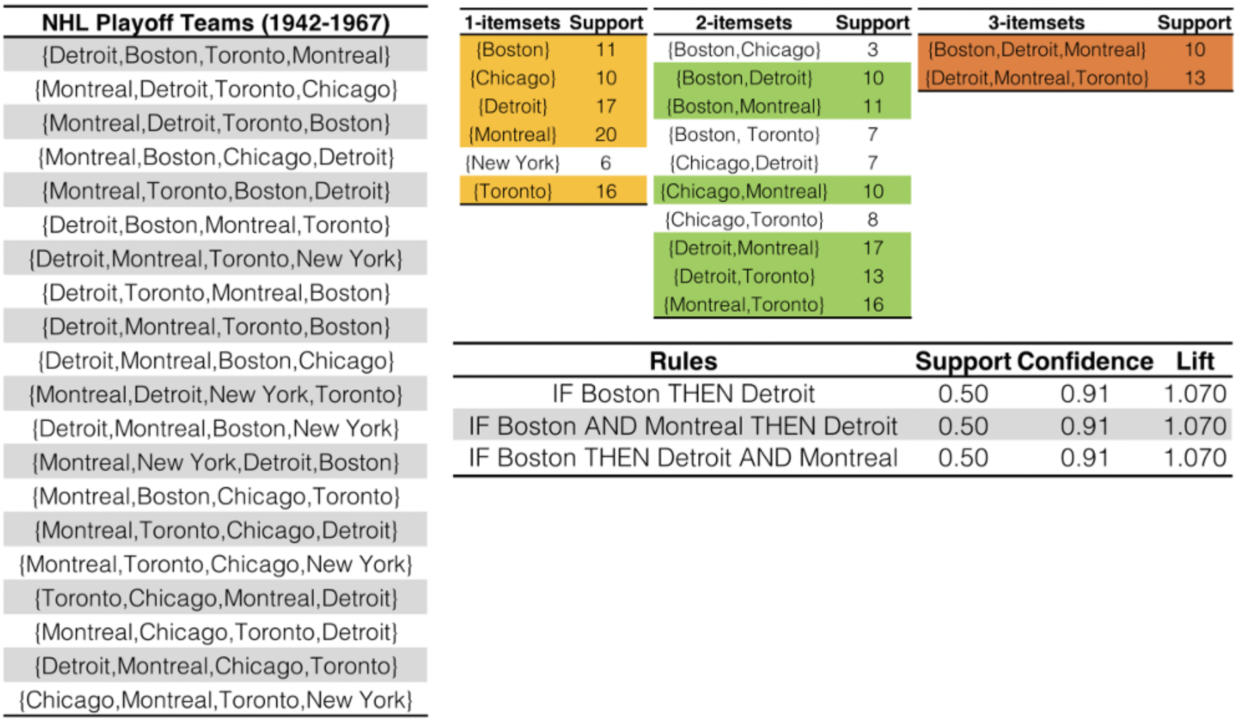
It starts by identifying frequent **individual items** in the database and extends those that are retained into larger and larger **item supersets**, who are themselves retained only if they occur **frequently enough** in the data.

The main idea is that "all non-empty subsets of a frequent itemset must also be frequent" [203], or equivalently, that all supersets of an infrequent itemset must also be infrequent (see Figure 19.4).

In the technical jargon of machine learning, we say that *a priori* uses a **bottom-up approach** and the **downward closure property of support**.

The memory savings arise from the fact that the algorithm prunes candidates with **infrequent sub-patterns** and removes them from consideration for any future itemset: if a 1−itemset is not considered to be frequent enough, any 2−itemset containing it is also infrequent (see Figure 19.5 for another illustration).

A list of the 4 teams making the playoffs each year is shown on the left ($N = 20$). Frequent itemsets are generated using the *a priori* algorithms,



NHL Playoff Teams (1942-1967)
{Detroit,Boston,Toronto,Montreal}
{Montreal,Detroit,Toronto,Chicago}
{Montreal,Detroit,Toronto,Boston}
{Montreal,Boston,Chicago,Detroit}
{Montreal,Toronto,Boston,Detroit}
{Detroit,Boston,Montreal,Toronto}
{Detroit,Montreal,Toronto,New York}
{Detroit,Toronto,Montreal,Boston}
{Detroit,Montreal,Toronto,Boston}
{Detroit,Montreal,Boston,Chicago}
{Montreal,Detroit,New York,Toronto}
{Detroit,Montreal,Boston,New York}
{Montreal,New York,Detroit,Boston}
{Montreal,Boston,Chicago,Toronto}
{Montreal,Toronto,Chicago,Detroit}
{Montreal,Toronto,Chicago,New York}
{Toronto,Chicago,Montreal,Detroit}
{Montreal,Chicago,Toronto,Detroit}
{Detroit,Montreal,Chicago,Toronto}
{Chicago,Montreal,Toronto,New York}

1-itemsets	Support
{Boston}	11
{Chicago}	10
{Detroit}	17
{Montreal}	20
{New York}	6
{Toronto}	16

2-itemsets	Support
{Boston,Chicago}	3
{Boston,Detroit}	10
{Boston,Montreal}	11
{Boston, Toronto}	7
{Chicago,Detroit}	7
{Chicago,Montreal}	10
{Chicago,Toronto}	8
{Detroit,Montreal}	17
{Detroit,Toronto}	13
{Montreal,Toronto}	16

3-itemsets	Support
{Boston,Detroit,Montreal}	10
{Detroit,Montreal,Toronto}	13

Rules	Support	Confidence	Lift
IF Boston THEN Detroit	0.50	0.91	1.070
IF Boston AND Montreal THEN Detroit	0.50	0.91	1.070
IF Boston THEN Detroit AND Montreal	0.50	0.91	1.070

Figure 19.5: Association rules for NHL playoff teams (1942-1967): 4 teams (out of 6) made the playoffs each year.

with a support threshold of 10. We see that there are 5 frequent 1−itemsets, in yellow;[18] 6 frequent 2−itemsets are found in the subsequent list of ten 2−itemsets, in green.[19] Only 2 frequent 3−itemsets are found, in orange. Candidate rules are generated from the shaded itemsets; the rules retained by the thresholds

$$\text{Support} \geq 0.5, \ \text{Confidence} \geq 0.7, \ \text{and Lift} > 1 \text{ (barely)},$$

are shown in the table on the bottom row – the main result is that when Boston made the playoffs, it was not surprising to see Detroit also make the playoffs.[20] Are these rules meaningful at all?

Of course, this process requires a support threshold **input**, for which there is no guaranteed way to pick a "good" value; it has to be set sufficiently high to minimize the number of frequent itemsets that are being considered, but not so high that it removes too many candidates from the **output list**.[21]

The algorithm terminates when no further itemsets extensions are retained, which must occur in datasets with a finite # of categorical levels.

- **Strengths:** easy to implement and to parallelize [204]
- **Limitations:** slow, requires frequent data set scans, not ideal for finding rules for infrequent and rare itemsets

More efficient algorithms have since displaced *a priori* in practice:[22]

- **Max-Miner** tries to identify frequent itemsets without enumerating them – it performs jumps in itemset space instead of using a bottom-up approach;
- **Eclat** is faster and uses depth-first search, but requires extensive memory storage.[23]

[18] 18: New York made the playoffs $6 < 10$ times – no larger frequent itemset can contain New York.

[19] 19: Note the absence of New York.

[20] 20: The presence or absence of Montreal in a rule is a red herring, as Montreal made the playoffs every year in the data.

[21] 21: As ever, optimal threshold values are **dataset-specific**.

[22] 22: Although it retains **historical value**.

[23] 23: *A priori* and eclat are both implemented in the R package `arules` [198].

19.3.4 Validation

How **reliable** are association rules? What is the likelihood that they occur entirely **by chance**? How **relevant** are they? Can they be generalised **outside** the dataset, or to **new** data streaming in?

These questions are notoriously difficult to answer for association rules discovery, but **statistically sound association discovery** can help reduce the risk of finding spurious associations to a user-specified significance level [201, 202].

We end this section with a few comments.

- Since frequent rules correspond to instances that occur repeatedly in the dataset, algorithms that generate itemsets often try to **maximize coverage**. When **rare events** are more meaningful (such as detection of a rare disease or a threat), we need algorithms that can generate rare itemsets. **This is not a trivial problem**.
- Continuous data has to be binned into **categorical** data to generate rules. As there are many ways to accomplish that task, the same dataset can give rise to completely different rules. This could create some credibility issues with clients and stakeholders.
- Other popular algorithms include: AIS, SETM, aprioriTid, aprioriHybrid, PCY, Multistage, Multihash, etc.
- Additional evaluation metrics can be found in the `arules` documentation [198].

19.3.5 Case Study: Danish Medical Data

In *temporal disease trajectories condensed from population wide registry data covering 6.2 million patients* [157], A.B. Jensen *et al.* study diagnoses in the Danish population, with the help of association rules mining and clustering methods.

Objectives Estimating **disease progression** (trajectories) from current patient state is a crucial notion in medical studies. Such trajectories had (at the time of publication) only been analyzed for a small number of diseases, or using large-scale approaches without consideration for time exceeding a few years. Using data from the *Danish National Patient Registry* (an extensive, long-term data collection effort by Denmark), the authors sought connections between different **diagnoses**: how does the presence of a diagnosis at some point in time allow for the prediction of another diagnosis at a later point in time?

Methodology The authors took the following methodological steps:

1. compute the **strength of correlation** for pairs of diagnoses over a 5 year interval (on a representative subset of the data);
2. test diagnoses pairs for **directionality** (one diagnosis repeatedly occurring before the other);
3. determine reasonable **diagnosis trajectories** (thoroughfares) by combining smaller (but frequent) trajectories with overlapping diagnoses;

4. **validate** the trajectories by comparison with non-Danish data,
5. **cluster** the thoroughfares to identify a small number of **central medical conditions** (key diagnoses) around which disease progression is organized.

Data The Danish National Patient Registry is an electronic health registry containing administrative information and diagnoses, covering the whole population of Denmark, including private and public hospital visits of all types: inpatient (overnight stay), outpatient (no overnight stay) and emergency. The data set covers 15 years, from January '96 to November '10 and consists of 68 million records for 6.2 million patients.

Challenges and Pitfalls

- Access to the **Patient Registry** is protected and could only be granted after approval by the *Danish Data Registration Agency the National Board of Health*.
- Gender-specific differences in diagnostic trends are clearly identifiable (pregnancy and testicular cancer do not have much cross-appeal), but many diagnoses were found to be made exclusively (or at least, predominantly) in different sites (inpatient, outpatient, emergency ward), which suggests the importance of stratifying by **site** as well as by **gender**.
- In the process of forming small diagnoses chains, it became necessary to compute the correlations using **large groups** for each pair of diagnoses. For close to 1 million diagnosis pairs, more than 80 million samples would have been required to obtain significant $p-$values while compensating for **multiple testing**, which would have translated to a thousand years' worth of computer running time. A pre-filtering step was included to avoid this pitfall.[24]

Project Summary and Results The dataset was reduced to **1,171 significant trajectories**. These thoroughfares were clustered into patterns centred on 5 key diagnoses central to disease progression:

- **diabetes**;
- **chronic obstructive pulmonary disease** (COPD);
- **cancer**;
- **arthritis**, and
- **cerebrovascular disease**.

Early diagnoses for these central factors can help reduce the risk of adverse outcome linked to future diagnoses of other conditions.

Two author quotes illustrate the importance of these results:

> "The sooner a health risk pattern is identified, the better we can prevent and treat critical diseases." [S. Brunak]

> "Instead of looking at each disease in isolation, you can talk about a complex system with many different interacting factors. By looking at the order in which different diseases

[24] The final trajectories were validated using the full sampling procedure.

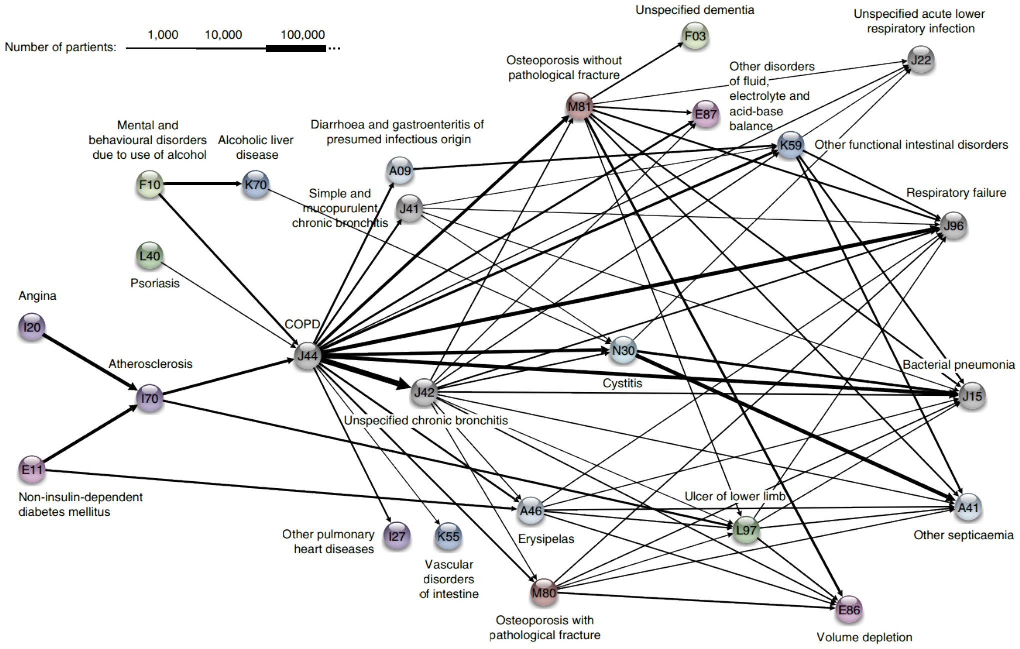

Number of partients:
1,000
10,000
100,000
Unspecified dementia
F03
Unspecified acute lower respiratory infection
J22
Osteoporosis without pathological fracture
M81
Other disorders of fluid, electrolyte and acid-base balance
E87
Mental and behavioural disorders due to use of alcohol
Alcoholic liver disease
Diarrhoea and gastroenteritis of presumed infectious origin
K59
Other functional intestinal disorders
F10
K70
A09
Simple and mucopurulent chronic bronchitis
J41
Respiratory failure
J96
L40
Psoriasis
Angina
I20
COPD
J44
N30
Bacterial pneumonia
Atherosclerosis
I70
J42
Cystitis
J15
Unspecified chronic bronchitis
E11
Non-insulin-dependent diabetes mellitus
A46
Ulcer of lower limb
A41
L97
Other pulmonary heart diseases
I27
K55
Erysipelas
Other septicaemia
Vascular disorders of intestine
M80
E86
Osteoporosis with pathological fracture
Volume depletion

Figure 19.6: The COPD cluster showing five preceding diagnoses leading to COPD and some of its possible outcomes [157].

> appear, you can start to draw patterns and see complex correlations outlining the direction for each individual person."
> [L.J. Jensen]

Among the specific results, the following "surprising" insights were found:

- a diagnosis of anemia is typically followed months later by the discovery of colon cancer;
- gout was identified as a step on the path toward cardiovascular disease, and
- COPD is under-diagnosed and under-treated.

The disease trajectories cluster for COPD is shown in Figure 19.6.

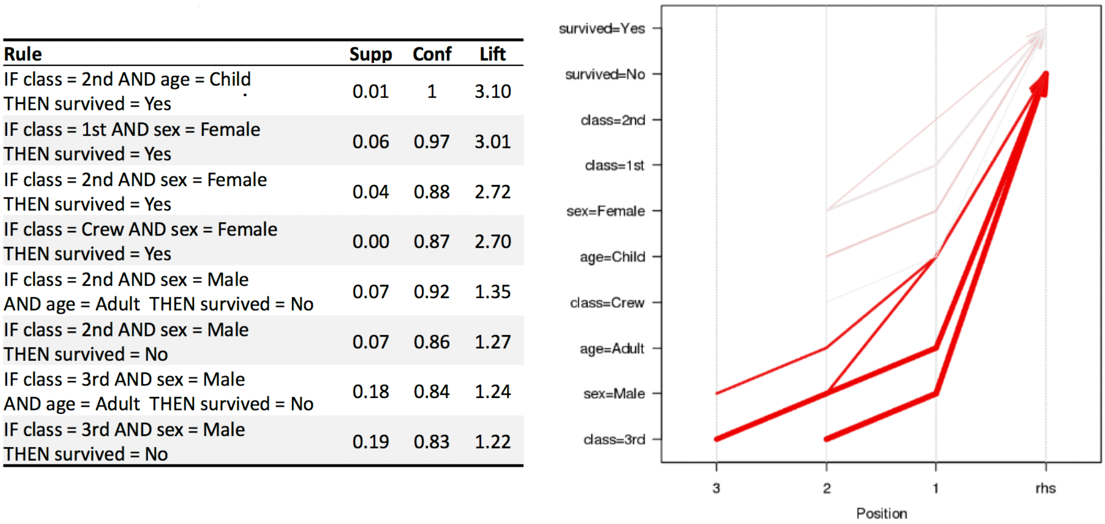
19.3.6 Toy Example: Titanic Dataset

Compiled by Robert Dawson in 1995, the *Titanic* dataset consists of 4 categorical attributes for each of the 2201 people aboard the Titanic when it sank in 1912 (some issues with the dataset have been documented, but we will ignore them for now):

- **class** (1st class, 2nd class, 3rd class, crewmember)
- **age** (adult, child)
- **sex** (male, female)
- **survival** (yes, no)

The natural question of interest for this dataset is:

> "How does survival relate to the other attributes?"



Rule	Supp	Conf	Lift
IF class = 2nd AND age = Child THEN survived = Yes	0.01	1	3.10
IF class = 1st AND sex = Female THEN survived = Yes	0.06	0.97	3.01
IF class = 2nd AND sex = Female THEN survived = Yes	0.04	0.88	2.72
IF class = Crew AND sex = Female THEN survived = Yes	0.00	0.87	2.70
IF class = 2nd AND sex = Male AND age = Adult THEN survived = No	0.07	0.92	1.35
IF class = 2nd AND sex = Male THEN survived = No	0.07	0.86	1.27
IF class = 3rd AND sex = Male AND age = Adult THEN survived = No	0.18	0.84	1.24
IF class = 3rd AND sex = Male THEN survived = No	0.19	0.83	1.22

Figure 19.7: Visualization of 8 *Titanic* association rules with parallel coordinates.

This is not, strictly speaking, an unsupervised task (as the interesting rules' structure is fixed to conclusions of the form survival = Yes or survival = No).

For the purpose of this example, we elect not to treat the problem as a **predictive task**, since the long removed situation on the Titanic has little bearing on survival for new data – as such, we use fixed-structure association rules to **describe** and **explore** survival conditions on the *Titanic* (compare with [205]).

We use the `arules` implementation of the *a priori* algorithm in `R` to generate and prune candidate rules, eventually leading to **8 rules** (the results are visualized in Figure 19.7). Who survived? Who didn't?[25] We show how to obtain these rules *via* `R` in Section 19.7.1 (*Association Rules Mining: Titanic Dataset*).

19.4 Classification and Value Estimation

> "The diversity of problems that can be addressed by classification algorithms is significant, and covers many domains. It is difficult to comprehensively discuss all the methods in a single book." [6]

19.4.1 Overview

The principles underlying classification, regression and class probability estimation are well-known and straightforward. **Classification** is a supervised learning endeavour in which a sample **training** set of data is used to determine rules and patterns that divide the data into predetermined groups, or classes. The training set usually consists of a **randomly** selected subset of the **labeled** data.[26]

[25] Again, with feeling: **correlation does not imply causation.**

[26] **Value estimation** (regression) is similar to classification, except that the target variable is numerical instead of categorical.

In the **testing phase**, the model is used to assign a class to observations in the **testing set**, in which the label is hidden, in spite of being actually known.

The performance of the predictive model is then **evaluated** by comparing the predicted and the actual values for the testing set observations (but **never** using the training set observations). A number of technical issues need to be addressed in order to achieve optimal performance, among them: determining which features to select for inclusion in the model and, perhaps more importantly, which algorithm to choose.

The **edible**/**poisonous** mushroom model from *Data Science and Machine Learning Tasks* provides a clean example of a **classification model**, albeit one for which no detail regarding the training data and choice of algorithm were made available.

Applications Classification and value estimation models are among the most frequently used of the data science models, and form the backbone of what is also known as **predictive analytics**. There are applications and uses in just about every field of human endeavour, such as:

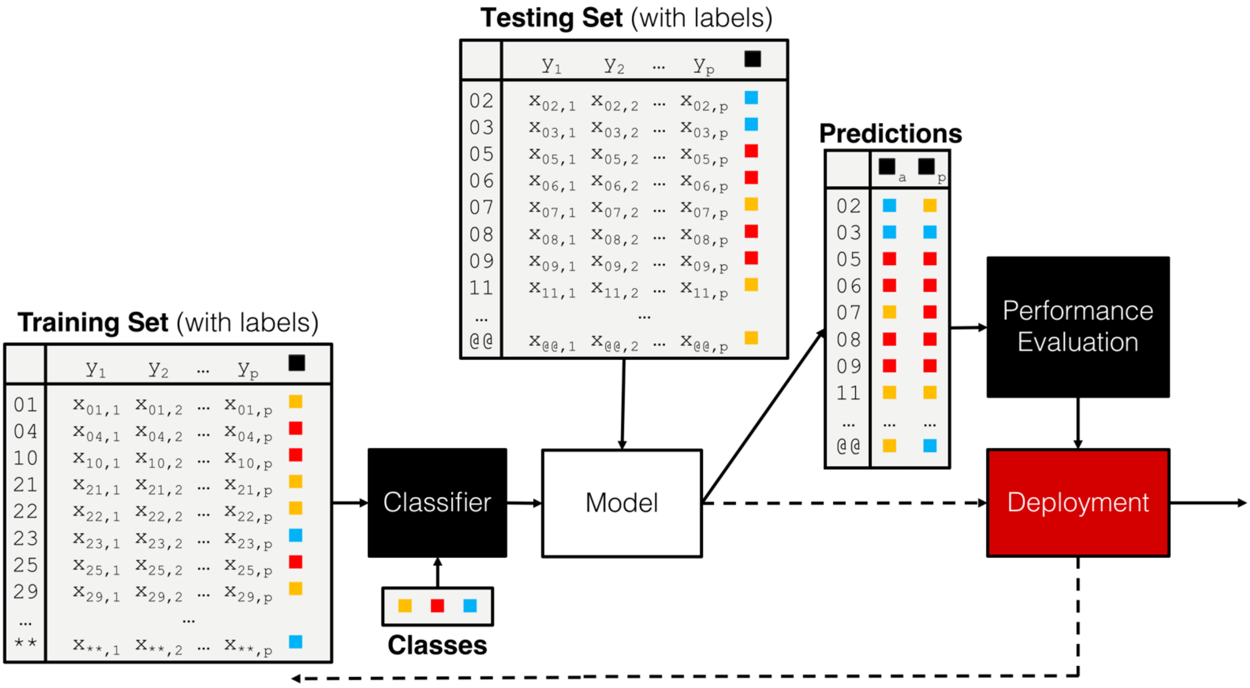
- **medicine and health science** – predicting which patient is at risk of suffering a second, and this time fatal, heart attack within 30 days based on health factors (blood pressure, age, sinus problems, etc.);
- **social policies** – predicting the likelihood of required assisted housing in old age based on demographic information/survey answers;
- **marketing/business** – predicting which customers are likely to cancel their membership to a gym based on demographics and usage;
- in general, predicting that an object belongs to a particular class, or organizing and grouping instances into categories, or
- enhancing the detection of relevant objects:
  - **avoidance** – "this object is an incoming vehicle";
  - **pursuit** – "this object is leaving the scene of a collision";
  - **degree** – "this object is 90% likely to run in front of the car",
- **economics** – predicting the inflation rate for the coming two years based on a number of economic indicators.

Other examples may be found in [206–209].

Concrete Examples Some concrete examples may provide a clearer picture of the types of SL problems encountered by analysts.

- A motor insurance company has a fraud investigation department that studies up to 20% of all claims made, yet money is still getting lost on fraudulent claims. To help better direct the investigators, management would like to determine, using past data, if it is possible to predict whether a claim is likely to be fraudulent?[27]

[27] And/or whether a customer is likely to commit fraud in the near future? Whether an application for a policy is likely to result in a fraudulent claim? If the amount by which a claim will be reduced if it is fraudulent?



Training Set (with labels)

	y_1	y_2	...	y_p	■
01	$x_{01,1}$	$x_{01,2}$	...	$x_{01,p}$	🟨
04	$x_{04,1}$	$x_{04,2}$	...	$x_{04,p}$	🟥
10	$x_{10,1}$	$x_{10,2}$	...	$x_{10,p}$	🟥
21	$x_{21,1}$	$x_{21,2}$	...	$x_{21,p}$	🟨
22	$x_{22,1}$	$x_{22,2}$	...	$x_{22,p}$	🟨
23	$x_{23,1}$	$x_{23,2}$	...	$x_{23,p}$	🟦
25	$x_{25,1}$	$x_{25,2}$	...	$x_{25,p}$	🟥
29	$x_{29,1}$	$x_{29,2}$	...	$x_{29,p}$	🟨
...			...		
**	$x_{**,1}$	$x_{**,2}$	...	$x_{**,p}$	🟦

Testing Set (with labels)

	y_1	y_2	...	y_p	■
02	$x_{02,1}$	$x_{02,2}$	...	$x_{02,p}$	🟦
03	$x_{03,1}$	$x_{03,2}$	...	$x_{03,p}$	🟦
05	$x_{05,1}$	$x_{05,2}$	...	$x_{05,p}$	🟥
06	$x_{06,1}$	$x_{06,2}$	...	$x_{06,p}$	🟥
07	$x_{07,1}$	$x_{07,2}$	...	$x_{07,p}$	🟨
08	$x_{08,1}$	$x_{08,2}$	...	$x_{08,p}$	🟥
09	$x_{09,1}$	$x_{09,2}$	...	$x_{09,p}$	🟥
11	$x_{11,1}$	$x_{11,2}$	...	$x_{11,p}$	🟨
...			...		
@@	$x_{@@,1}$	$x_{@@,2}$	...	$x_{@@,p}$	🟨

Predictions

	$■_a$	$■_p$
02	🟦	🟨
03	🟦	🟦
05	🟥	🟥
06	🟥	🟥
07	🟨	🟥
08	🟥	🟥
09	🟥	🟥
11	🟨	🟨
...	...	...
@@	🟨	🟦

Figure 19.8: A classification pipeline, including training set, testing set, performance evaluation, and (eventual) deployment.

- Customers who make a large number of calls to a mobile phone company's customer service number have been identified as churn risks. The company is interested in reducing said churn. Can they predict the overall lifetime value of a customer? Which customers are more likely to churn in the near future? What retention offer a particular customer will best respond to?

In every classification scenario, the following questions must be answered before embarking on analysis:

- What kind of data is **required**?
- How much of it?
- What would constitute a **predictive success**?
- What are the **risks** associated with a predictive modeling approach?

These have no one-size-fits-all answers; they have to be considered on a case-by-case basis.

In the absence of testing data, classification models cannot be used for predictive tasks, but may still be useful for **descriptive** tasks (see Titanic example above).

When testing data exists, the overall process is often quite similar, independently of the choice of the algorithm (see the classification pipeline shown in Figure 19.8).

This clearly points to the importance of obtaining **good test data**, but keep in mind that this process may be costly and/or difficult to implement, in general. Data scientists often have to enact clever schemes to collect the right "stuff" – consider, for instance, the current procedures used to prove that an online user is not a bot, such as identifying all traffic lights, motorcycles, crosswalks, store fronts, etc. in a picture.[28]

[28] It is not in fact the answers that identify a user as human; rather, it is reaction times and mouse movements that betray bots. The answers are used to collect data to train self-driving vehicle AIs.

19.4.2 Classification Algorithms

The number of classification algorithms is truly staggering – it often seems as though new algorithms and variants are put forward on a monthly basis, depending on the task and on the type of data [6].

While some of them tend to be rather esoteric, there is a fairly small number of commonly-used workhorse algorithms/approaches that data scientists and consultants should at least have at their command:[29]

- **logistic regression** and linear regression are classical models which are often used by statisticians but rarely in a classification setting (the estimated coefficients are often used to determine the features' importance); one of their strengths is that the machinery of standard statistical theory (hypothesis testing, confidence intervals, etc.) is still available to the analyst, but they are easily affected by variance inflation in the presence of predictor multi-collinearity, and the stepwise variable selection process that is typically used is problematic – regularization methods would be better suited in general [210] (see Figure 19.9 for illustrations);
- **neural networks** have become popular recently due to the advent of deep learning; they might provide the prototypical example of a **black box** algorithm as they are hard to interpret; another issue is that they require a fair amount of data to train properly – we will have more to say on the topic in a later chapter;
- **decision trees** are perhaps the most common of all data science algorithms, but they tend to overfit the data when they are not pruned correctly, a process which often has to be done manually (see Figure 19.9 for an illustration) – we shall discuss the pros an cons of decision trees in general in *Decision Trees*;
- **naïve Bayes classifiers** have known quite a lot of success in text mining applications (more specifically as the basis of powerful spam filters), but, embarrassingly, no one is quite sure why they should work as well as they do given that one of their required assumptions (independence of priors) is rarely met in practice (see Figure 19.9 for an illustration);
- **support vector machines** attempt to separate the dataset by "fitting" as wide of a "tube" as possible through the classes (subjected to a number of penalty constraints); they have also known successes, most notably in the field of digital handwriting recognition, but their decision boundaries (the tubes in question) tend to be non-linear and quite difficult to interpret; nevertheless, they may help mitigate some of the difficulties associated with big data (see Figure 19.10 for an illustration);
- **nearest neighbours classifiers** (kNN) basically implement a voting procedure and require very little assumptions about the data, but they are not very stable as adding training points may substantially modify the boundary (see Figures 19.9 and 19.10 for illustrations),

Boosting methods [3, 211] and **Bayesian methods** [49, 212, 213] are also becoming increasingly more popular.

[29] Full descriptions: [2, 103, 168].

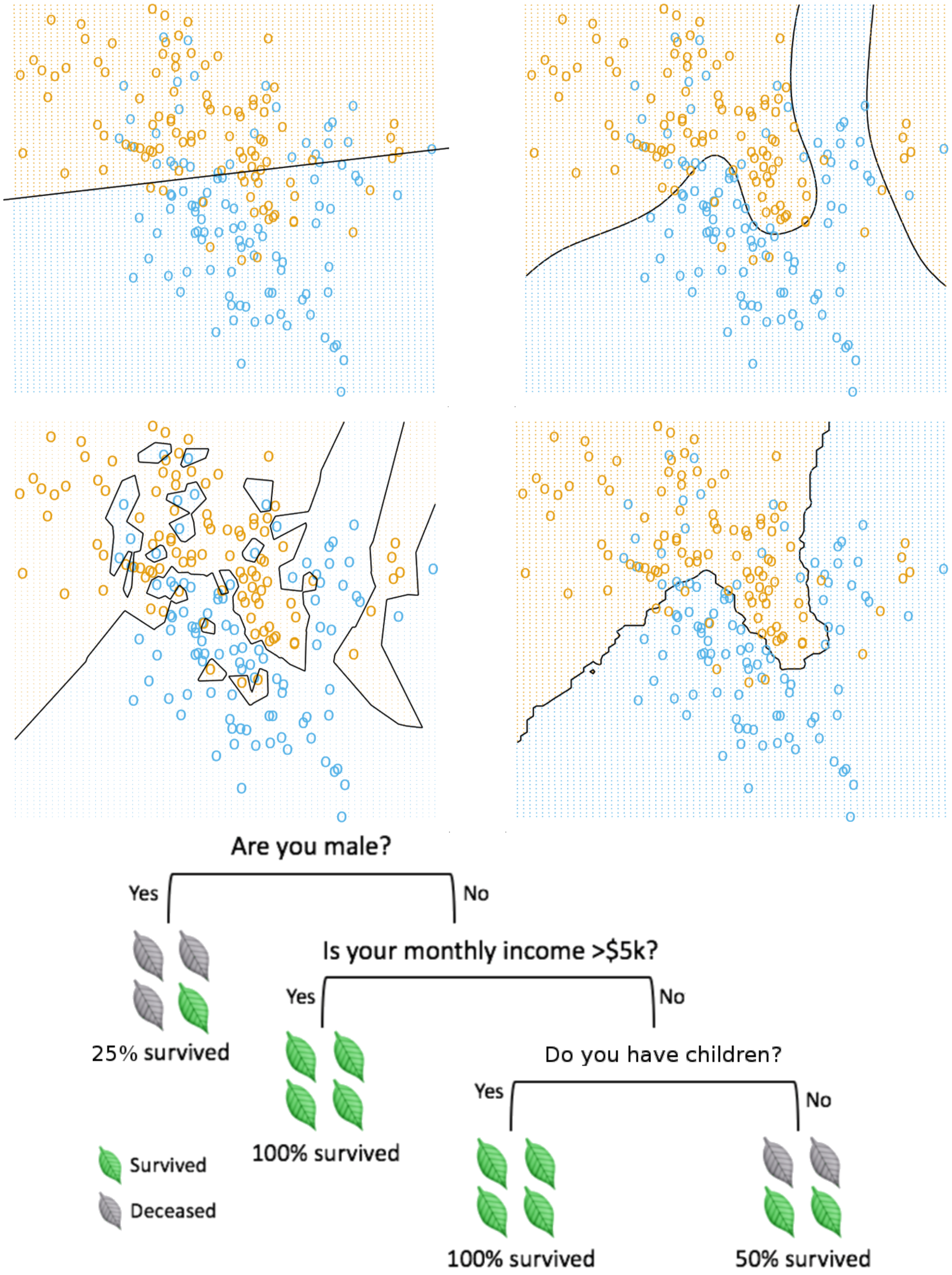


Figure 19.9: Illustrations of various classifiers – linear regression, top left; optimal Bayes, top right; 1NN and 15NN, middle left and right, respectively, on an artificial dataset (from [2]); decision tree depicting the chances of survival for various disasters (fictional, based on [214]). Note that linear regression is more stable, simpler to describe, but less accurate than *k*NN and optimal Bayes.

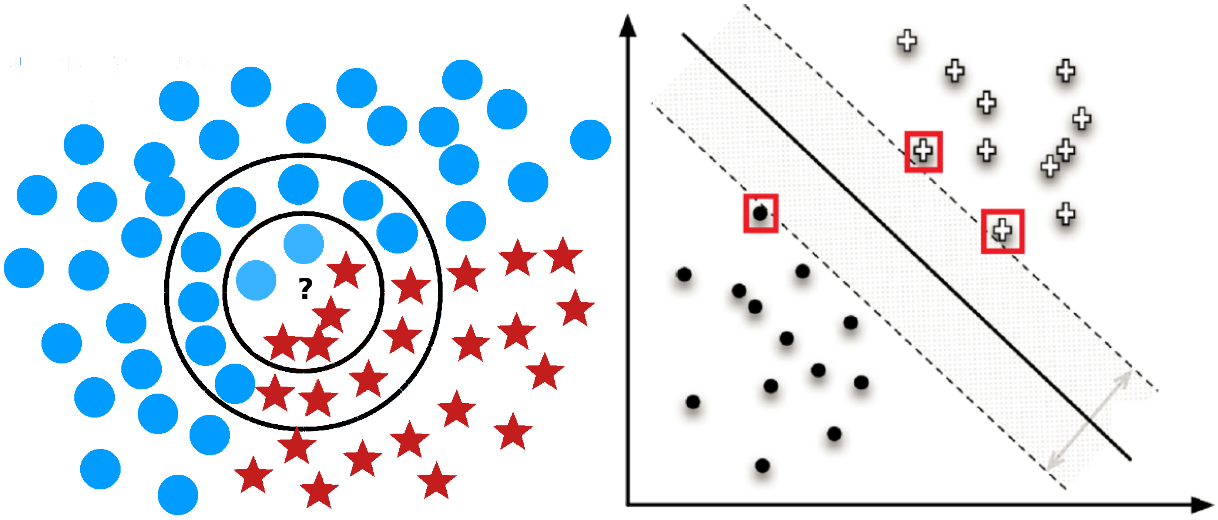


Figure 19.10: Illustration of a k nearest neighbour (left) and a support vector machines classifier (right, based on [168]). What is the 6NN prediction for the location marked by a question mark? What about the 19NN prediction?

19.4.3 Decision Trees

In order to highlight the relative simplicity of most classification algorithms, we will discuss the workings of **ID3**, a historically significant decision tree algorithm.[30]

Classification trees are perhaps the most **intuitive** of all supervised methods: classification is achieved by following a path up the tree, from its **root**, through its **branches**, and ending at its **leaves** (although typically the tree is depicted with its root at the top and its leaves at the bottom).

To make a **prediction** for a new instance, it suffices to follow the path down the tree, reading the prediction directly once a leaf is reached. It sounds simple enough in theory, but in practice, creating the tree and traversing it might be **time-consuming** if there are too many variables in the dataset (due to the criterion that is used to determine how the branches split).

Prediction accuracy can be a concern in trees whose growth is **unchecked**. In practice, the criterion of **purity** at the leaf-level[31] is linked to bad prediction rates for new instances. Other criteria are often used to prune trees, which may lead to impure leaves.

How do we grow such trees?

For predictive purposes, we need a training set and a testing set upon which to evaluate the tree's performance. Ross Quinlan's **Iterative Dichotomizer 3** (a precursor to the widely-used C4.5 and C5.0) follows a simple procedure:

1. split the training data (**parent**) set into (**children**) subsets, using the different levels of a particular attribute;
2. compute the **information gain** for each subset;
3. select the **most advantageous** split, and
4. repeat for each node until some **leaf criterion** is met.

[30] 30: ID3 would never be used in a deployment setting, but it will serve to illustrate a number of classification concepts.

[31] 31: That is to say, all instances in a leaf belong to the same leaf.

Entropy is a measure of disorder in a set S. Let p_i be the proportion of observations in S belonging to category i, for $i = 1, \dots, n$. The entropy of S is given by

$$E(S) = -\sum_{i=1}^{n} p_i \log p_i.$$

If the **parent set** S consisting of m records is split into k children sets $C_1, \dots, C_k$ containing $q_1, \dots, q_k$ records, respectively, then the **information gained** from the split is

$$I(S : C_1, \dots, C_k) = E(S) - \frac{1}{m}\sum_{j=1}^{k} q_j E(C_j).$$

The sum term in the information gain equation is a weighted average of the entropy of the children sets.

If the split leads to little disorder in the children, then $\mathrm{IG}(S; C_1, \dots, C_k)$ is high; if the split leads to similar disorder in both children and parent, then $\mathrm{IG}(S; C_1, \dots, C_k)$ is low.

Consider, as in Figure 19.11, two splits shown for a parent set with 30 observations separated into 2 classes: $\circ$ and $\star$.

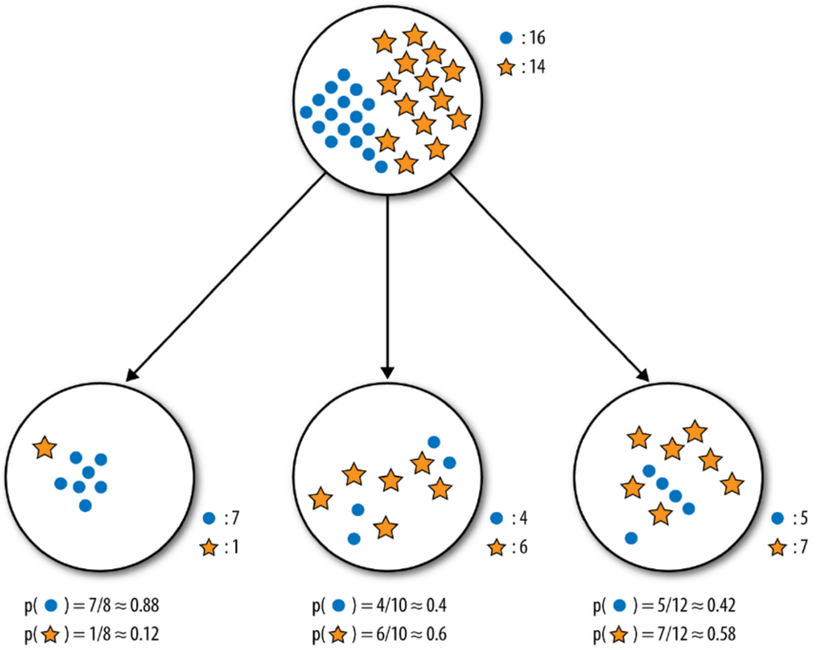


Figure 19.11: Picking the optimal information gain split. [168]

Visually, it appears as though the binary split does a better job of separating the classes. Numerically, the entropy of the parent set S is

$$\begin{aligned} E(S) &= -p_\circ \log p_\circ - p_\star \log p_\star \\ &= -\frac{16}{30}\log\frac{16}{30} - \frac{14}{30}\log\frac{14}{30} \approx 0.99. \end{aligned}$$

For the binary split (on the left), leading to the children set L (left) and R (right), the respective entropies are

$$E(L) = -\frac{12}{13}\log\frac{12}{13} - \frac{1}{13}\log\frac{1}{13} \approx 0.39$$

and

$$E(R) = -\frac{4}{17}\log\frac{4}{17} - \frac{13}{17}\log\frac{13}{17} \approx 0.79,$$

so that the information gained by that split is

$$\text{IG}(S; C_L, C_R) \approx 0.99 - \frac{1}{30}[13 \cdot 0.39 + 17 \cdot 0.79] = 0.37.$$

On its own, this value is not substantially meaningful – it is only in comparison to the information gained from other splits that it becomes useful. A similar computation for the three-way split leads to $\text{IG}(S; C_1, C_2, C_3) \approx 0.13$, which is indeed smaller than the information gained by the binary split – of these two options, ID3 would select the first as being **most advantageous**.

Decision trees have numerous strengths: they

- are easy to interpret, providing, as they do, a **white box model** – predictions can always be explained by following the appropriate paths;
- can handle numerical and categorical data **simultaneously**, without first having to "binarise" the data;
- can be used with **incomplete** datasets, if needed (although there is still some value in imputing missing observations);
- allow for **built-in feature selection** as less relevant features do not tend to be used as splitting features;
- make **no assumption** about independence of observations, underlying distributions, multi-collinearity, etc., and can thus be used without the need to verify assumptions;
- lend themselves to **statistical validation** (in the form of cross-validation), and
- are in line with **human decision-making approaches**, especially when such decisions are taken deliberately.

On the other hand, they are

- **not usually as accurate** as other more complex algorithms, nor **as robust**, as small changes in the training data can lead to a completely different tree, with a completely different set of predictions;[This can become problematic when presenting the results to a client whose understanding of these matters is slight.]
- particularly **vulnerable to overfitting** in the absence of pruning — and pruning procedures are typically fairly convoluted (some algorithms automate this process, using statistical tests to determine when a tree's "full" growth has been achieved), and
- **biased** towards categorical features with high number of levels, which may give such variables undue importance in the classification process.

Information gain tends to grow small trees in its pursuit of pure leaves, but it is not the only splitting metric in use (**Gini impurity**, **variance reduction**, etc.).

Notes ID3 is a precursor of C4.5, perhaps the most popular decision tree algorithm on the market. There are other tree algorithms, such as C5.0, CHAID, MARS, conditional inference trees, CART, etc., each grown using algorithms with their own strengths and weaknesses.

Regression trees are grown in a similar fashion, but with a **numerical** response variable (predicted inflation rate, say), which introduces some complications [2, 3].

Decision trees can also be combined together using **boosting algorithms** (such as **AdaBoost**) or **random forests**, providing a type of voting procedure also known as **ensemble learning** – an individual tree might make middling predictions, but a large number of judiciously selected trees are likely to make good predictions, on average [2, 3, 211] – we will re-visit these concepts in a later chapter.

Additionally:

- since classification is linked to **probability estimation**, approaches that extend the basic ideas of regression models could prove fruitful;
- **rare occurrences** are often more interesting and more difficult to predict and identify than regular instances – historical data at Fukushima's nuclear reactor prior to the 2011 meltdown could not have been used to learn about meltdowns, for obvious reasons,[32]
- with big datasets, algorithms must also consider **efficiency** – thankfully, decision trees are easily parallelizable.

19.4.4 Performance Evaluation

As a consequence of the (so-called) **No Free-Lunch Theorem**, no single classifier can be the best performer for every problem [215, 216]. Model selection must take into account:

- the **nature** of the available data;
- the **relative frequencies of the classification sub-groups**;
- the **stated classification goals**;
- how easily the model lends itself to **interpretation** and **statistical analysis**;
- how much **data preparation** is required;
- whether it can accommodate various data types and missing observations;
- whether it performs well with large datasets, and
- whether it is **robust** against small data departures from theoretical assumptions.

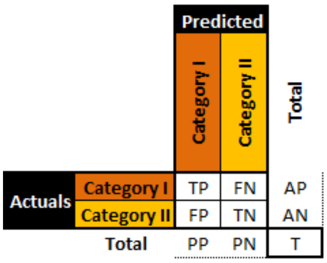
Past success is not a guarantee of future success – it is the analyst's responsibility to try a variety of models. But how can the "best" model be selected?

When a classifier attempts to determine what kind of music a new customer would prefer, there is next to no cost in making a mistake; if, on the other hand, the classifier attempts to determine the presence or absence of cancerous cells in lung tissue, mistakes are more consequential. Several metrics can be used to assess a classifier's performance, depending on the context.

[32] Classical performance evaluation metrics can easily be fooled; if out of two classes one of the instances is only represented in 0.01% of the instances, predicting the non-rare class will yield correct predictions roughly 99.99% of the time, missing the point of the exercise altogether.

Binary classifiers (presented in the abstract in Table 19.1) are simpler and have been studied far longer than multi-level classifiers; consequently, a larger body of evaluation metrics is available for these classifiers.

In the medical literature, for instance, TP, TN, FP and FN stand for **True Positives**, **True Negatives**, **False Positives**, and **False Negatives**, respectively.



		Predicted		
		Category I	Category II	Total
Actuals	Category I	TP	FN	AP
	Category II	FP	TN	AN
	Total	PP	PN	T

Table 19.1: A general binary classifier.

A perfect classifier would be one for which both $\text{FP}, \text{FN} = 0$, but in practice, that rarely ever happens (if at all).

Traditional **performance metrics** include:

- sensitivity: TP/AP
- specificity: TN/AN
- precision: TP/PP
- negative predictive value: TN/PN
- false positive rate: FP/AN
- false discovery rate: $1 - \text{TP}/\text{PP}$
- false negative rate: FN/AP
- accuracy: $(\text{TP} + \text{TN})/\text{T}$
- F_1−score: $2\text{TP}/(2\text{TP} + \text{FP} + \text{FN})$
- MCC: $\frac{\text{TP}\cdot\text{TN}-\text{FP}\cdot\text{FN}}{\sqrt{\text{AP}\cdot\text{AN}\cdot\text{PP}\cdot\text{PN}}}$
- informedness/ROC: $\text{TP}/\text{AP} + \text{TN}/\text{AN} - 1$
- markedness: $\text{TP}/\text{PP} + \text{TN}/\text{PN} - 1$.

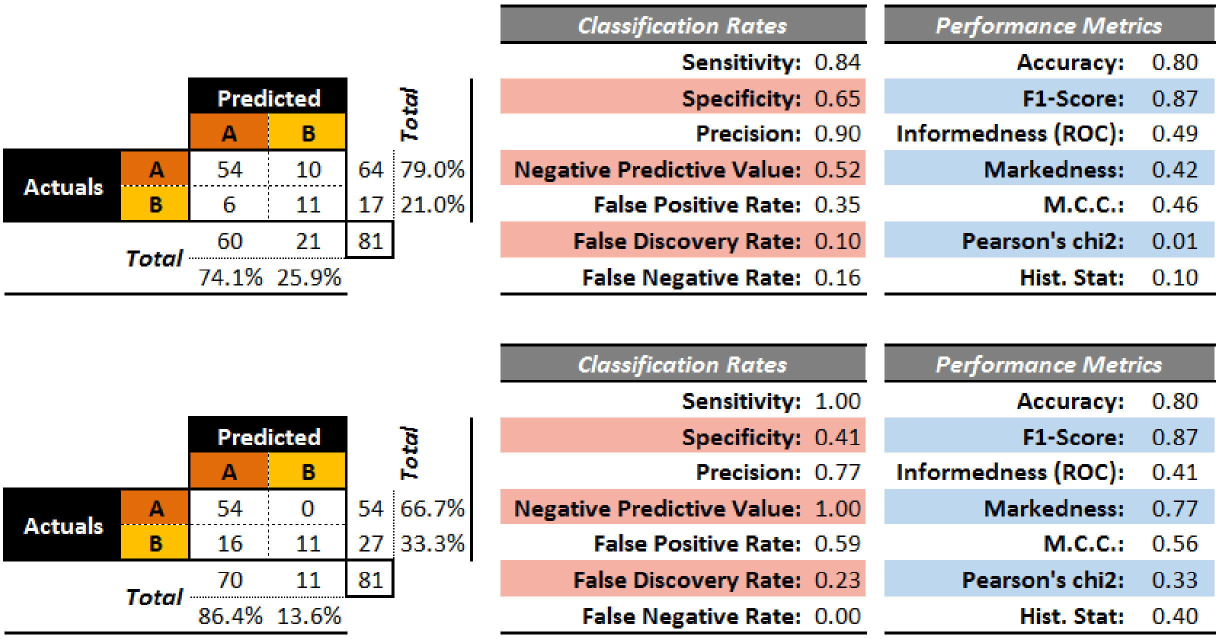
The **confusion matrices** of two artificial binary classifers for a testing set are shown in Table 19.2.

Both classifiers have an accuracy of 80%, but while the second classifier sometimes makes a wrong prediction for A, it never does so for B, whereas the first classifier makes erroneous predictions for both A and B. On the other hand, the second classifier mistakenly predicts occurrence A as B 16 times while the first one only does so 6 times. Which is best?

The performance metrics alone do not suffice to answer the question: the cost associated with making a mistake must also be factored in. Furthermore, it could be preferable to select performance evaluation metrics that generalize more readily to **multi-level classifiers** (see Table 19.3 for examples of associated confusion matrices).

The **accuracy** is the proportion of correct predictions amid all the observations; its value ranges from 0% to 100%. The higher the accuracy, the better the match, and yet, a predictive model with high accuracy may nevertheless be useless thanks to the **Accuracy Paradox** (see rare occurrence sidenote on page 808).

The **Matthews Correlation Coefficient** (MCC), on the other hand, is a statistic which is of use even when the classes are of very different sizes. As



		Predicted A	Predicted B	Total	
Actuals	A	54	10	64	79.0%
	B	6	11	17	21.0%
Total		60	21	81	
		74.1%	25.9%		

Classification Rates	
Sensitivity:	0.84
Specificity:	0.65
Precision:	0.90
Negative Predictive Value:	0.52
False Positive Rate:	0.35
False Discovery Rate:	0.10
False Negative Rate:	0.16

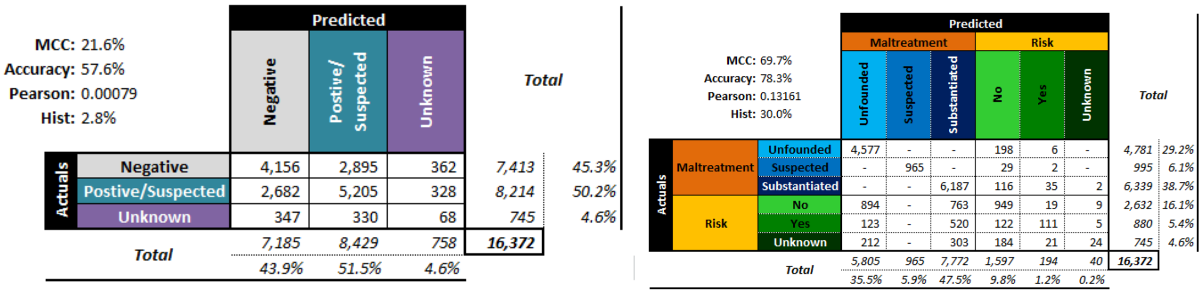
Performance Metrics	
Accuracy:	0.80
F1-Score:	0.87
Informedness (ROC):	0.49
Markedness:	0.42
M.C.C.:	0.46
Pearson's chi2:	0.01
Hist. Stat:	0.10

		Predicted A	Predicted B	Total	
Actuals	A	54	0	54	66.7%
	B	16	11	27	33.3%
Total		70	11	81	
		86.4%	13.6%		

Classification Rates	
Sensitivity:	1.00
Specificity:	0.41
Precision:	0.77
Negative Predictive Value:	1.00
False Positive Rate:	0.59
False Discovery Rate:	0.23
False Negative Rate:	0.00

Performance Metrics	
Accuracy:	0.80
F1-Score:	0.87
Informedness (ROC):	0.41
Markedness:	0.77
M.C.C.:	0.56
Pearson's chi2:	0.33
Hist. Stat:	0.40

Table 19.2: Performance metrics for two (artificial) binary classifiers.



MCC: 21.6%
Accuracy: 57.6%
Pearson: 0.00079
Hist: 2.8%

		Predicted Negative	Predicted Postive/ Suspected	Predicted Unknown	Total	
Actuals	Negative	4,156	2,895	362	7,413	45.3%
	Postive/Suspected	2,682	5,205	328	8,214	50.2%
	Unknown	347	330	68	745	4.6%
Total		7,185	8,429	758	16,372	
		43.9%	51.5%	4.6%		

MCC: 69.7%
Accuracy: 78.3%
Pearson: 0.13161
Hist: 30.0%

			Predicted							
			Maltreatment			Risk				
			Unfounded	Suspected	Substantiated	No	Yes	Unknown	Total	
Actuals	Maltreatment	Unfounded	4,577	-	-	198	6	-	4,781	29.2%
		Suspected	-	965	-	29	2	-	995	6.1%
		Substantiated	-	-	6,187	116	35	2	6,339	38.7%
	Risk	No	894	-	763	949	19	9	2,632	16.1%
		Yes	123	-	520	122	111	5	880	5.4%
		Unknown	212	-	303	184	21	24	745	4.6%
		Total	5,805	965	7,772	1,597	194	40	16,372	
			35.5%	5.9%	47.5%	9.8%	1.2%	0.2%		

Table 19.3: Performance metrics for (artificial) multi-level classifiers: ternary - left; senary - right [personal files].

a correlation coefficient between the actual and predicted classifications, its range varies from −1 to 1.

If MCC = 1, the predicted and actual responses are identical, while if MCC = 0, the classifier performs no better than a random prediction ("flip of a coin").

It is also possible to introduce two non-traditional performance metrics (which are nevertheless well-known statistical quantities) to describe how accurately a classifier preserves the classification distribution (rather than how it behaves on an observation-by-observation basis):

- Pearson's χ^2: $\frac{1}{T}\left((\text{PP}-\text{AP})^2/\text{PP}+(\text{PN}-\text{AN})^2/\text{PN}\right)$
- Hist: $\frac{1}{T}\left(|\text{PP}-\text{AP}|+|\text{PN}-\text{AN}|\right)$

Note, however, that these are non-standard performance metrics. For a given number of levels, the smaller these quantities, the more similar the actual and predicted distributions.

For **numerical targets** y with predictions $\hat{y}$, the confusion matrix is not defined, but a number of classical performance evaluation metrics can be used on the testing set: the

- **mean squared** and **mean absolute errors**

$$\text{MSE} = \text{mean}\left\{(y_i - \hat{y}_i)^2\right\}, \quad \text{MAE} = \text{mean}\{|y_i - \hat{y}_i|\};$$

- **normalized mean squared/mean absolute errors**

$$\text{NMSE} = \frac{\text{mean}\left\{(y_i - \hat{y}_i)^2\right\}}{\text{mean}\left\{(y_i - \overline{y})^2\right\}},$$
$$\text{NMAE} = \frac{\text{mean}\left\{|y_i - \hat{y}_i|\right\}}{\text{mean}\left\{|y_i - \overline{y}|\right\}};$$

- **mean average percentage error**

$$\text{MAPE} = \text{mean}\left\{\frac{|y_i - \hat{y}_i|}{y_i}\right\};$$

- **correlation** $\rho_{y,\hat{y}}$, which is based on the notion that for good models, the predicted values and the actual values should congregate around the lines $y = \hat{y}$ (as in Figure 19.12).

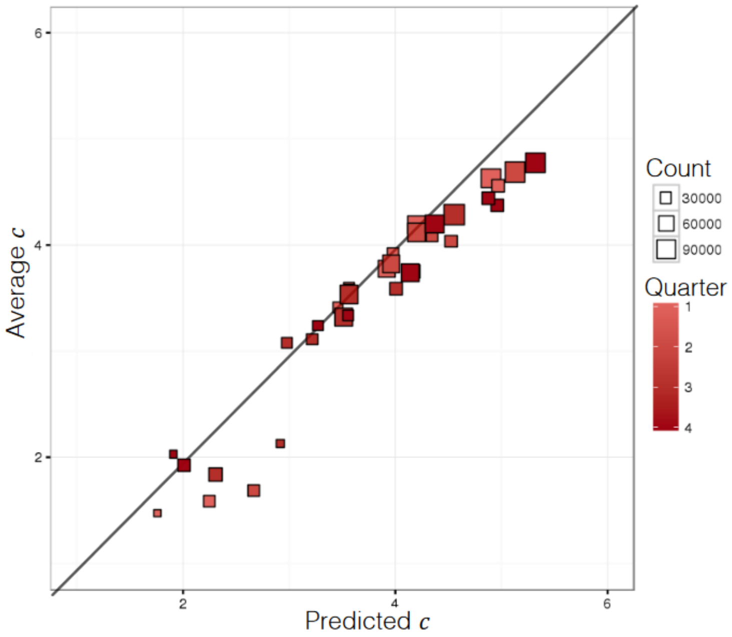


Figure 19.12: Predicted and actual numerical responses [personal file].

As is the case for classification, an isolated value estimation performance metric does not provide enough of a rationale for model validation/selection. One possible exception: **normalized evaluation metrics** do provide some information about the relative quality of performance [2, 3].

19.4.5 Case Study: Minnesota Tax Audit

Large gaps between revenue owed (in theory) and revenue collected (in practice) are problematic for governments. Revenue agencies implement various fraud detection strategies (such as audit reviews) to bridge that gap.

Since business audits are rather costly, there is a definite need for algorithms that can predict whether an audit is likely to be successful or a waste of resources.

In *Data Mining Based Tax Audit Selection: A Case Study of a Pilot Project at the Minnesota Department of Revenue* [158], Hsu et al. study the Minnesota Department of Revenue's (DOR) tax audit selection process with the help of classification algorithms.

Objective The U.S. Internal Revenue Service (IRS) estimated that there were large gaps between **revenue owed** and **revenue collected** for 2001 and for 2006. Using DOR data, the authors sought to increase **efficiency** in the audit selection process and to **reduce the gap** between revenue owed and revenue collected.

Methodology The authors took the following steps:

1. **data selection and separation:** experts selected several hundred cases to audit and divided them into training, testing and validating sets;
2. **classification modeling** using MultiBoosting, Naïve Bayes, C4.5 decision trees, multilayer perceptrons, support vector machines, etc;
3. **evaluation of all models** was achieved by testing the model on the testing set – models originally performed poorly on the testing set until the size of the business being audited was recognized to have an effect, leading to two separate tasks (large and small businesses),
4. **model selection and validation** was done by comparing the estimated accuracy between different classification model predictions and the actual field audits. Ultimately, MultiBoosting with Naïve Bayes was selected as the final model; the combination also suggested some improvements to increase audit efficiency.

Data The data consisted of selected tax audit cases from 2004 to 2007, collected by the audit experts, which were split into training, testing and validation sets:

- the **training data** set consisted of *Audit Plan General* (APGEN) *Use Tax* audits and their results for the years 2004-2006;
- the **testing data** consisted of APGEN Use Tax audits conducted in 2007 and was used to test or evaluate models (for Large and Smaller businesses) built on the training dataset,
- while **validation** was assessed by actually conducting field audits on predictions made by models built on 2007 Use Tax return data processed in 2008.

None of the sets had records in common (see Figure 19.13).

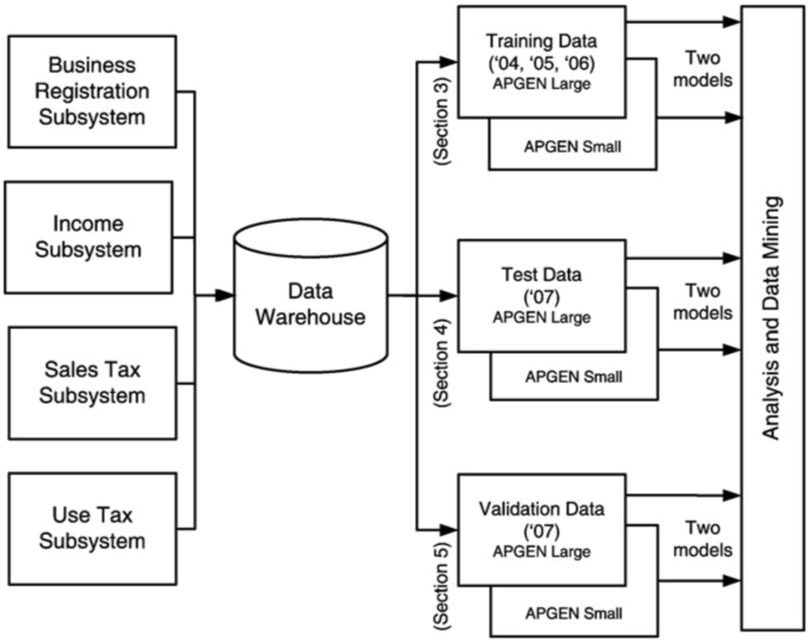


Figure 19.13: Data sources for APGEN mining [158]. Note the 6 final sets which feed the Data Analysis component.

Strengths and Limitations of Algorithms

- The Naïve Bayes classification scheme assumes independence of the features, which rarely occurs in real-world situations. This approach is also known to potentially introduce bias to classification schemes. In spite of this, classification models built using Naïve Bayes have a successful track record.
- MultiBoosting is an **ensemble technique** that uses committee (i.e. groups of classification models) and "group wisdom" to make predictions; unlike other ensemble techniques, it is different from other ensemble techniques in the sense that it forms a committee of sub-committees (i.e., a group of groups of classification models), which has a tendency to reduce both bias and variance of predictions (see [6, 211] for more information on these topics).

Procedures Classification schemes need a response variable for prediction: audits which yielded more than $500 per year in revenues during the audit period were classified as *Good*; the others were *Bad*. The various models were tested and evaluated by comparing the performances of the manual audits (which yield the actual revenue) and the classification models (the predicted classification).

The procedure for manual audit selection in the early stages of the study required:

1. DOR experts selecting several thousand potential cases through a query;
2. DOR experts further selecting several hundreds of these cases to audit;
3. DOR auditors actually auditing the cases, and
4. calculating audit accuracy and return on investment (ROI) using the audits results.

Once the ROIs were available, data mining started in earnest. The steps involved were:

1. **Splitting the data** into training, testing, and validating sets.
2. **Cleaning the training data** by removing "bad" cases.
3. **Building** (and revising) **classification models** on the training dataset. The first iteration of this step introduced a separation of models for larger businesses and relatively smaller businesses according to their **average annual withholding amounts** (the threshold value that was used is not revealed in [158]).
4. **Selecting separate modeling features** for the APGEN Large and Small training sets. The feature selection process is shown in Figure 19.14.
5. **Building classification models** on the training dataset for the two separate class of business (using C4.5, Naïve Bayes, multilayer perceptron, support vector machines, etc.), and assessing the classifiers using **precision** and **recall** with improved estimated ROI:

$$\text{Efficiency} = \text{ROI} = \frac{\text{Total revenue generated}}{\text{Total collection cost}}$$

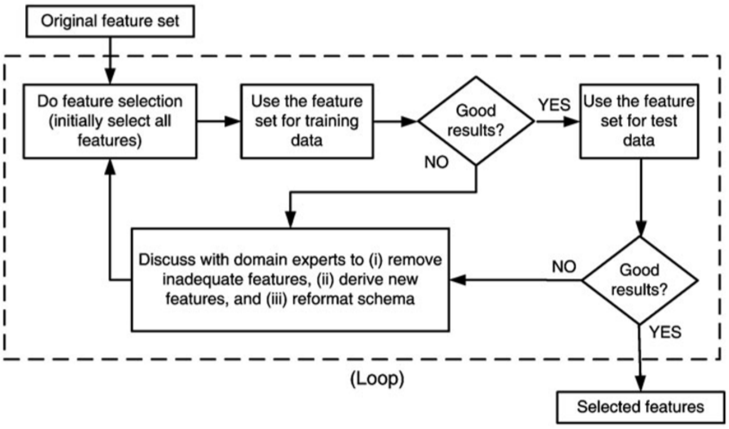

Original feature set
Do feature selection (initially select all features)
Use the feature set for training data
Good results?
YES
NO
Use the feature set for test data
Discuss with domain experts to (i) remove inadequate features, (ii) derive new features, and (iii) reformat schema
NO
Good results?
YES
(Loop)
Selected features

Figure 19.14: Feature selection process in [158]; note the involvement of domain experts.

Results, Evaluation and Validation The models that were eventually selected were combinations of MultiBoosting and Naïve Bayes (C4.5 produced interpretable results, but its performance was shaky). For APGEN Large (2007), experts had put forward 878 cases for audit (495 of which proved successful), while the classification model suggested 534 audits (386 of which proved successful). The theoretical best process would find 495 successful audits in 495 audits performed, while the manual audit selection process needed 878 audits in order to reach the same number of successful audits.

For APGEN Small (2007), 473 cases were recommended for audit by experts (only 99 of which proved successful); in contrast, 47 out of the 140 cases selected by the classification model were successful. The theoretical best process would find 99 successful audits in 99 audits performed, while the manual audit selection process needed 473 audits in order to reach the same number of successful audits.

In both cases, the classification model improves on the manual audit process: roughly 685 data mining audits to reach 495 successful audits of

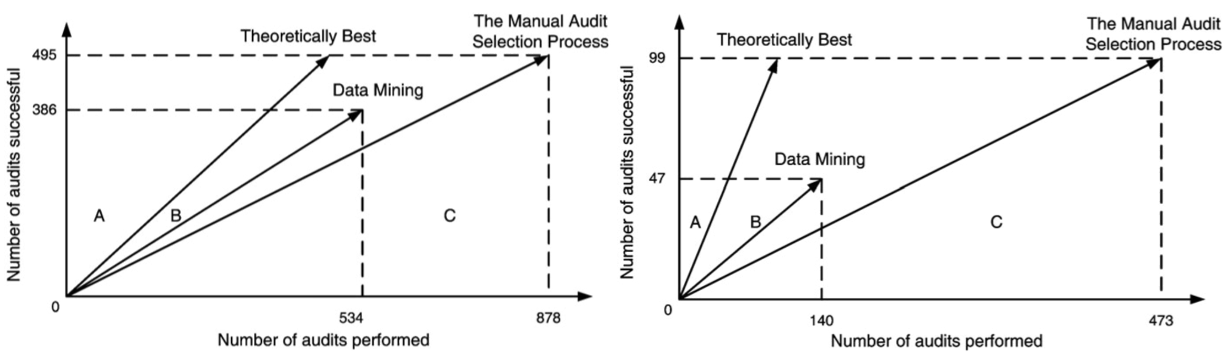


Figure 19.15: Audit resource deployment efficiency [158]; left – APGEN Large (2007); right – APGEN Small (2007). In both cases, the Data Mining approach was more efficient (the slope of the Data Mining vector is "closer" to the Theoretical Best vector than is the Manual Audit vector).

	Predicted as good	Predicted as bad

Actually good	386 (Use tax collected)
	R = $5,577,431 (83.6 %)
	C = $177,560 (44 %)
		109 (Use tax lost)
		R = $925,293 (13.9 %)
		C = $50,140 (12.4 %)

Actually bad	148 (costs wasted)
	R = $72,744 (1.1 %)
	C = $68,080 (16.9 %)
		235 (costs saved)
		R = $98,105 (1.4 %)
		C = $108,100 (26.7 %)

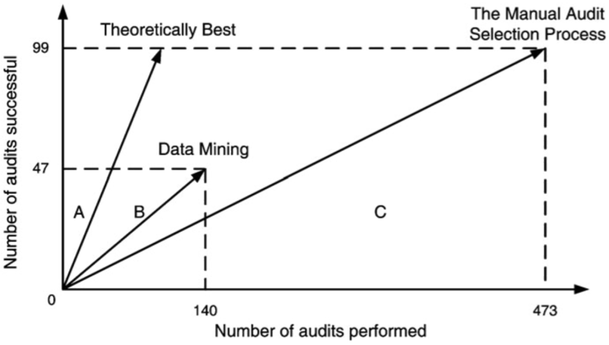


Table 19.4: Confusion matrices for audit evaluation [158]; left – APGEN Large (2007); right – APGEN Small (2007). *R* stands for revenues, *C* for collection costs.

APGEN Large (2007), and 295 would be required to reach 99 successful audits for APGEN Small (2007), as can be seen in Figure 19.15.

Figure 19.4 presents the confusion matrices for the classification model on both the APGEN Large and Small 2007 datasets.

The revenue *R* and collection cost *C* entries can be read as follows: the 47 successful audits which were correctly identified by the model for APGEN Small (2007) correspond to cases consuming 9.9% of collection costs but generating 42.5% of the revenues. Similarly, the 281 bad audits correctly predicted by the model represent notable collection cost savings. These are associated with 59.4% of collection costs but they generate only 11.1% of the revenues.

Once the testing phase of the study was completed, the DOR validated the data mining-based approach by using the models to select cases for actual field audits in a real audit project. The prior success rate of audits for APGEN Use tax data was 39% while the model was predicting a success rate of 56%; the actual field success rate was 51%.

Take-Aways A substantial number of models were churned out before the team made a final selection. Past performance of a model family in a previous project can be used as a guide, but it provides no guarantee regarding its performance on the current data – remember the *No Free Lunch (NFL) Theorem* [215]: nothing works best all the time!

There is a definite iterative feel to this project: the feature selection process could very well require a number of visits to domain experts before the feature set yields promising results. This is a valuable reminder that the data analysis team should seek out individuals with a good understand of both data and context. Another consequence of the NFL is that domain-specific knowledge has to be integrated in the model in order to beat random classifiers, on average [216].

Finally, this project provides an excellent illustration that even slight improvements over the current approach can find a useful place in an organization – data science is not solely about Big Data and disruption!

19.4.6 Toy Example: Kyphosis Dataset

As a basic illustration of these concepts, consider the following example. **Kyphosis** is a medical condition related to an excessive convex curvature of the spine. Corrective spinal surgery is at times performed on children.

A dataset of 81 observations and 4 attributes has been collected (we have no information on how the data was collected and how representative it is likely to be, but those details can be gathered from [217]).

The attributes are:

- **kyphosis** (absent or present after presentation);
- **age** (at time of operation, in months);
- **number** (of vertebrae involved),
- **start** (topmost vertebra operated on).

The natural question of interest for this dataset is:

> "How do the three explanatory attributes impact the operation's success?"

We use the `rpart` implementation of Classification and Regression Tree (CART) in `R` to generate a decision tree. Strictly speaking, this is not a predictive supervised task as we treat the entire dataset as a training set for the time being – there are no hold-out testing observations.

The results are shown in Figure 19.16. Interestingly, it would appear that the variable `number` does not play a role in determining the success of the operation (for the observations in the dataset).

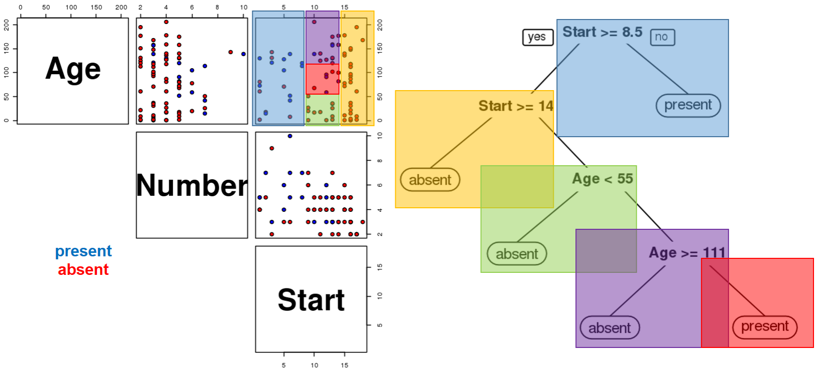


Figure 19.16: Kyphosis decision tree visualization. Only two features are used to construct the tree. We also note that the leaves are not pure – there are blue **and** red instances in 3 of the 5 classification regions.

Furthermore, the decision tree visualization certainly indicates that its leaves are not pure (see Figure 19.17. Some additional work suggests that the tree is somewhat overgrown and that it could benefit from being pruned after the first branching point.

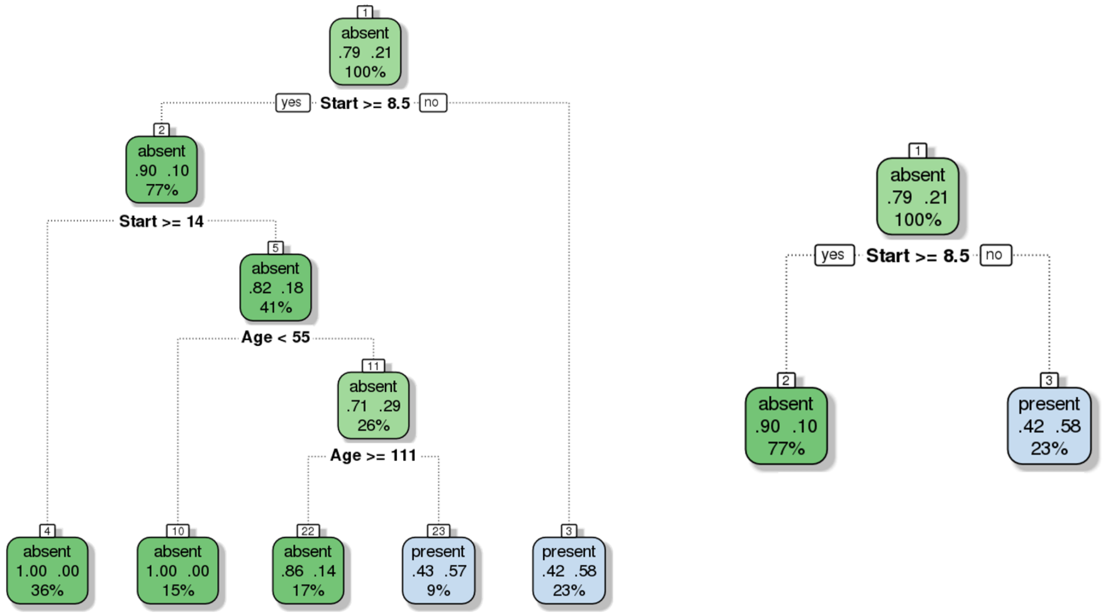
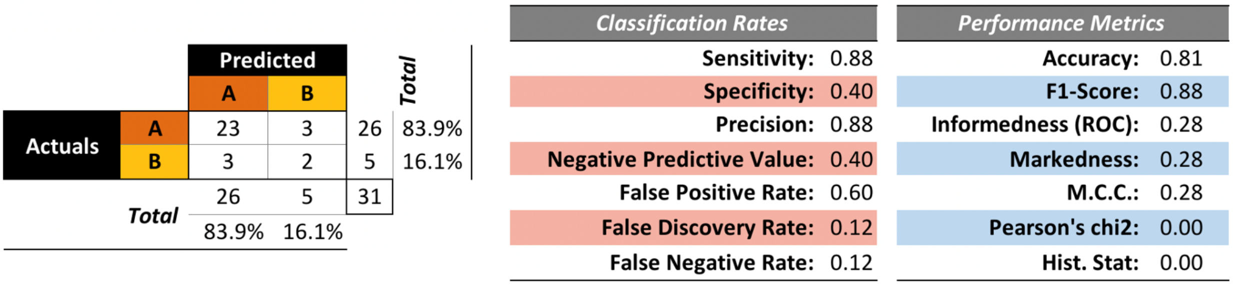


Figure 19.17: Pruning a decision tree – the original tree (left) is more accurate/more complex than the pruned tree (right).



		Predicted		*Total*	
		A	B		
Actuals	A	23	3	26	83.9%
	B	3	2	5	16.1%
Total		26	5	31	
		83.9%	16.1%		

Classification Rates	
Sensitivity:	0.88
Specificity:	0.40
Precision:	0.88
Negative Predictive Value:	0.40
False Positive Rate:	0.60
False Discovery Rate:	0.12
False Negative Rate:	0.12

Performance Metrics	
Accuracy:	0.81
F1-Score:	0.88
Informedness (ROC):	0.28
Markedness:	0.28
M.C.C.:	0.28
Pearson's chi2:	0.00
Hist. Stat:	0.00

Table 19.5: Kyphosis decision tree – performance evaluation. The accuracy and F_1 scores are good, but the false discovery and false negative rates are not so great. This tree is good at predicting successful surgeries, but not fantastic at predicting failed surgeries. Is it still useful?

At any rate, it remains meangingless to discuss the performance of the tree for **predictive purposes** if we are not using a holdout testing sample (not to say anything about the hope of generalizing to a larger population).

To that end, we trained a model on 50 randomly selected observations and evaluated the performance on the remaining 31 observations (the structure of the tree is not really important at this stage). The results are shown in Table 19.5. Is the model "good"?

It is difficult to answer this question in the machine learning sense without being able to compare its performance metrics with those of other models (or families of models).[33]

In the *Model Selection* subsection, we will briefly discuss how estimate a model's true predictive error rate through **cross-validation**. We will also discuss a number of other issues that can arise when ML/AI methods are not used correctly.

We show how to obtain these decision trees *via* R in Section 19.7 (*Classification: Kyphosis Dataset*).

[33] 33: The relative small size of the dataset should give data analysts pause for thought, at the very least.

19.5 Clustering

> "Clustering is in the eye of the beholder, and as such, researchers have proposed many induction principles and models whose corresponding optimisation problem can only be approximately solved by an even larger number of algorithms." [218]

19.5.1 Overview

We can make a variety of **quantitative statements** about a dataset, at the **univariate** level. For instance, we can

- compute **frequency counts** for the variables;
- identify **measures of centrality** (mean, mode, median), and
- measure the **dispersion** (range, standard deviation), among others.

At the **multivariate** level, the various options include n−**way tabulations**, **correlation analysis**, and **data visualization**, among others.

While these can provide insights in simple situations, datasets with a **large number of variables** or with **mixed types** (categorical and numerical) might not yield to such an approach. Instead, insights might come in the form of **aggregation** or **clustering** of similar observations.

A successful **clustering scheme** is one that tightly joins together any number of similarity profiles – "tight" in this context refers to small variability within the cluster, see Figure 19.18 for an illustration.

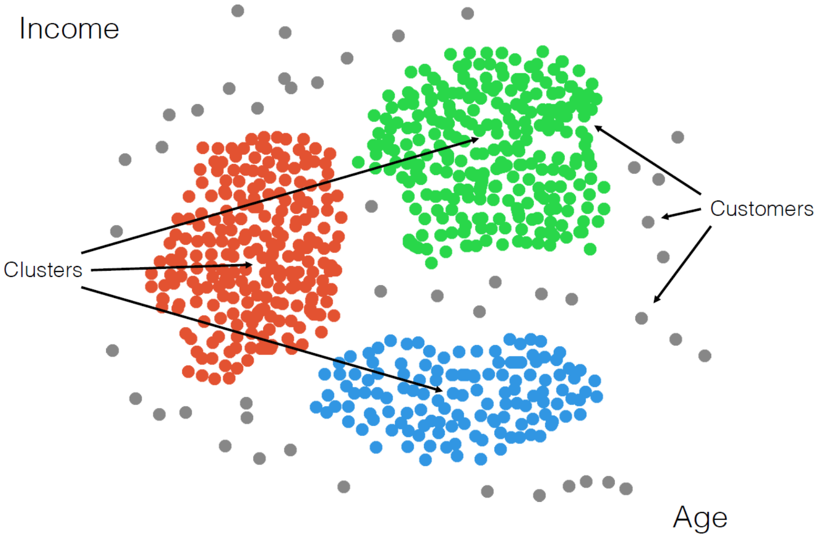


Figure 19.18: Clusters and outliers in an artificial dataset [personal file].

A typical application is one found in **search engines**, where the listed search results are the nearest similar objects (**relevant webpages**) clustered around the search item.

Dissimilar objects (**irrelevant webpages**) should not appear in the list, being "far" from the search item. Left undefined in this example is the crucial notion of **closeness**: what does it mean for one observation to be near another one? Various **metrics** can be used (see Figure 19.19 for some simple examples), and not all of them lead to the same results.

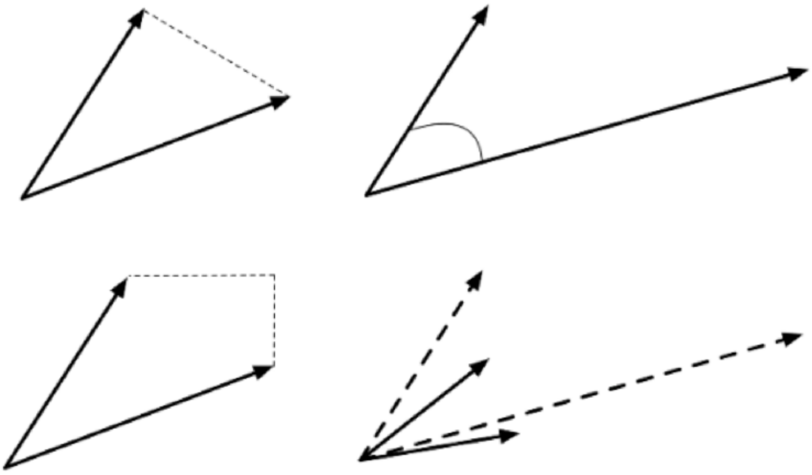


Figure 19.19: Distance metrics between observations: Euclidean (as the crow flies, top left); cosine (direction from a vantage point, top right); Manhattan (taxi-cab, bottom left). Observations should be transformed (scaled, translated) before distance computations (bottom right).

Clustering is a form of **unsupervised learning** since the cluster labels (and possibly their number) are not determined ahead of the analysis.

The algorithms can be **complex** and **non-intuitive**, based on varying notions of similarities between observations, and yet, the temptation to provide a simple *a posteriori* explanation for the groupings remains **strong** – we really, really want to reason with the data.[34]

They are also (typically) **non-deterministic** – the same routine, applied twice to the same dataset, can discover completely different clusters.[35]

This (potential) non-replicability is not just problematic for validation – it can also leads to client dissatisfaction. If the analyst is tasked with finding customer clusters for marketing purposes and the clusters change every time the client or the stakeholders ask for a report, they will be very confused (and will be doubting the results) unless the **stochastic nature** of the process has already been explained.

Another interesting aspect of clustering algorithms is that they often find clusters even when there are no natural ways to break down a dataset into constituent parts.

When there is no natural way to break up the data into clusters, the results may be **arbitrary** and fail to represent any underlying reality of the dataset. On the other hand, it could be that while there was no **recognized** way of naturally breaking up the data into clusters, the algorithm **discovered** such a grouping – clustering is sometimes called **automated classification** as a result.

The aim of clustering, then, is to divide into **naturally occurring groups**. Within each group, observations are similar; between groups, they are dissimilar (see Figure 19.20 for an illustration).

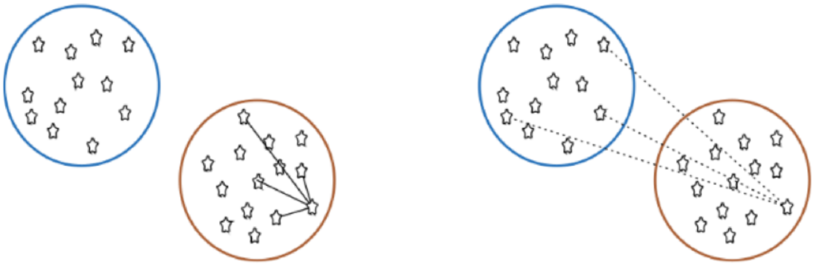


Figure 19.20: Distance to points in own clusters (left, smaller is better) and to points in other clusters (right, larger is better).

[34] Is it possible to look at Figure 19.18 without assigning labels or trying to understand what type of customers were likely to be young and have medium income? Older and wealthier?

[35] The order in which the data is presented can play a role, as can starting configurations.

As a learning process, clustering is fairly **intuitive** for human beings – our brains unconsciously search for patterns and they can generally handle **messy** data with the same relative ease as clean data. Computers have a harder time of it, however, partly because there is no **agreed-upon definition of what constitutes a cluster**, and so we cannot easily code their recognition into algorithms – to paraphrase Justice Potter Stewart,

> "I may not be able to define what a cluster is, but I know one when I see one."

All clustering algorithms rely on the notion of **similarity** w between observations; in many instances, similarity is obtained *via* a **distance** (or metric) d, with $w \longrightarrow 1$ as $d \longrightarrow 0$, and $w \longrightarrow 0$ as $d \longrightarrow \infty$. However, there are similarity measures which are not derived from a distance metric.

One additional clustering challenge is that there is no such thing as **the** distance or **the** similarity measure between observations – observations which are similar using a specific measure **may not be similar at all using another**. Commonly-used metrics include:

> euclidean, Manhattan, cosine, Canberra, Hamming, Jaccard, Pearson, and so on.

Note, however, that no matter which similarity measure is selected, the data must first be **transformed**: scaled, centered, etc. (see Figure 19.19). This introduces another layer of arbitrary choices, as there are multiple available options and no **canonical** way to perform this.

Applications Frequently, we use clustering and other unsupervised learning tasks as **preliminary steps** in supervised learning problems, but there exist stand-alone applications as well:

- **text analysis** – grouping similar documents according to their topics, based on the patterns of common and unusual terms;
- **product recommendations** – grouping online purchasers based on the products they have viewed, purchased, liked, or disliked, or grouping products based on customer reviews;
- **marketing** – grouping client profiles based on their demographics and preferences;
- **social network analysis** – recognizing communities within large groups of people;
- **medical imaging** – differentiating between different tissue types in a 3D voxel;
- **genetics** – inferring structures in populations;
- dividing a larger group (or area, or category) into smaller groups, with members of the smaller groups having similarities of some kind, as analytical tasks may then be solved separately for each of the smaller groups, which may lead to increased accuracy once the separate results are aggregated, or
- creating (new) taxonomies on the fly, as new items are added to a group of items, which could allow for easier product navigation on a website like Netflix, for instance.

Numerous other applications may be found in [4, 156, 159–162, 219–225].

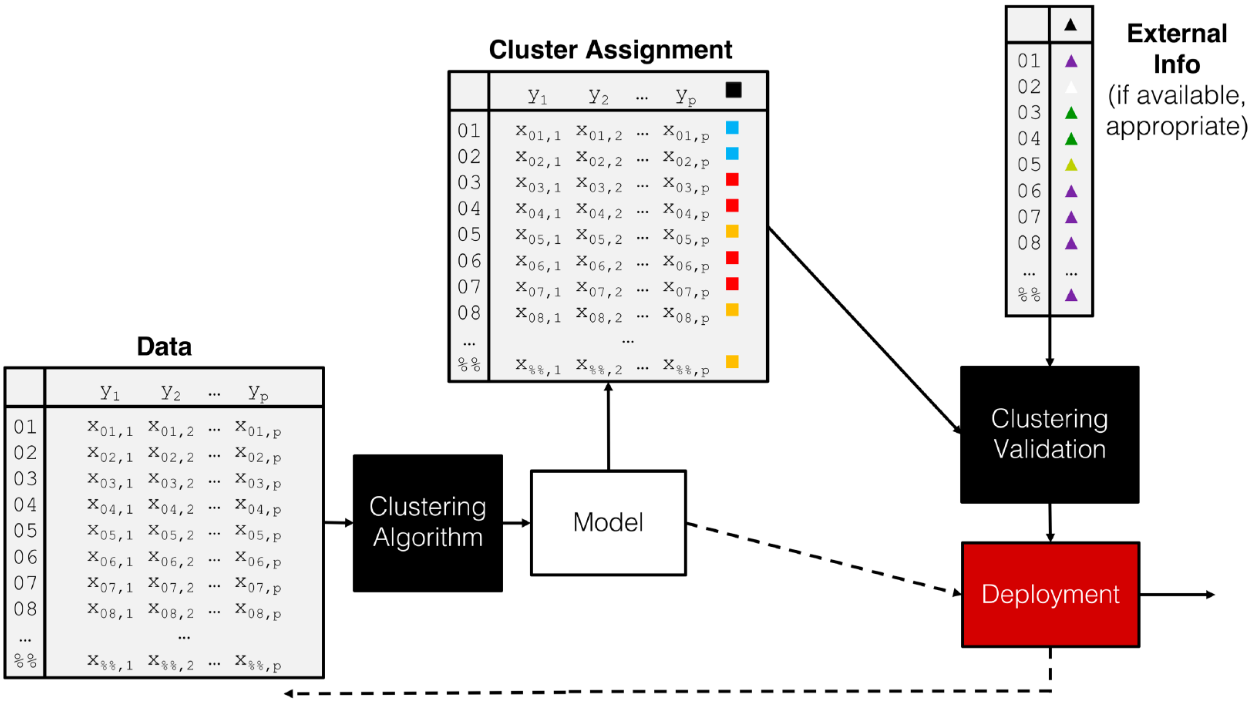


Figure 19.21: A clustering pipeline, including validation and (eventual) deployment.

When all is said and done, the clustering process is quite standard, notwithstanding the choice of scaling strategy, similarity measure, and algorithm and parameters (see the pipeline shown in Figure 19.21).

19.5.2 Clustering Algorithms

As is the case with classification, the number of clustering algorithms is quite high; the Wikipedia page lists 40+ such algorithms as of August 2018 [175]. The choice of algorithms (and associated parameters) is as much an art as it is a science, although domain expertise can come in handy [4].

There is a smaller list of common algorithms that data scientists and consultants should have in their toolbox:[36]

- k−**means**, close on the heels of decision trees for the title of "most-used data science algorithm", is a **partition clustering method** which tends to produce equal-sized clusters; when clients ask for their data to be clustered, they are typically envisioning k−means with the Euclidean metric; variants include k−mode (for categorical data), k−medians (for data with outliers), and k−means|| and k−means++ for large data sets; the number of clusters k (and the similarity measure/distance metric) must be provided by the user; works fairly well for "blob"-like data;
- **hierarchical clustering** is one of the few deterministic algorithms on the market, with **divisive** (DIANA) and **agglomerative** (AGNES) versions; no parameters need to be inputted, but the users must select a **linkage** strategy (roughly speaking, a metric that computes the distance between clusters) and a level at which to read off the clusters (see Figure 19.22 for an illustration);

[36] Full descriptions: [4, 103, 168].

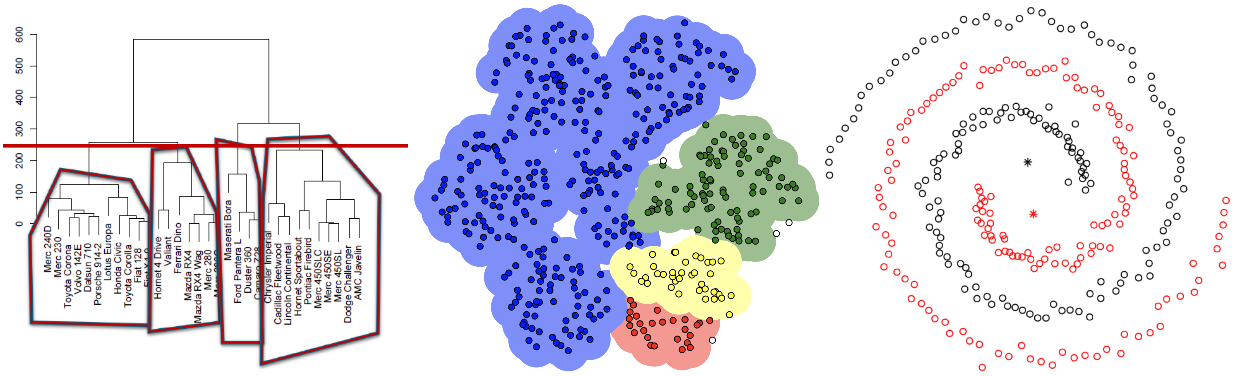


Figure 19.22: Illustration of hierarchical clustering (left), DBSCAN (middle, based on [226]), and spectral clustering (right).

- **density-based spatial clustering** (DBSCAN) is a graph-based approach which attempts to identify densely-packed regions in the dataset; its most obvious advantages (and of its variants OPTICS and DENCLUE) are **robustness to outliers** and not needing to input a **number of clusters** to search for in the data; the main disadvantage is that the optimal input parameters (**neighbourhood radius** and **minimum number of points** to be considered dense) are not easy to derive (see Figure 19.22);
- **affinity propagation** is another algorithm which selects the optimal number of clusters directly from the data, but it does so by trying and evaluating various scenarios, which may end up being **time-consuming**,
- **spectral clustering** can be used to recognize **non-globular** clusters (see Figure 19.22 for an illustration); these are found by computing eigenvalues of an associated Laplacian matrix – consequently, spectral clustering is **fast**.

Other methods include **latent Dirichlet allocation** (used in topics modeling), **expectation-maximisation** (particularly useful to find gaussian clusters), **BIRCH** (a local method which does not require the entire dataset to be scanned) and **fuzzy clustering** (a soft clustering scheme in which the observations have a degree of belonging to each cluster).

19.5.3 k-Means

As mentioned previously, $k-$means is a very natural way to group observations together (formally, $k-$means is linked to **Voronoi tilings**). $k-$means clustering is achieved by:

1. selecting a **distance metric** d (based on the data type and domain expertise);
2. selecting a **number of clusters** k;
3. randomly choosing k data instances as initial **cluster centres**;
4. calculating the **distance** from each observation to each centre;
5. placing each instance in the cluster whose centre it is **nearest** to;
6. computing/updating the **centroid** for each cluster (see Figure 19.23 for an illustration),
7. repeating steps 4-6 until the clusters are **“stable”**.

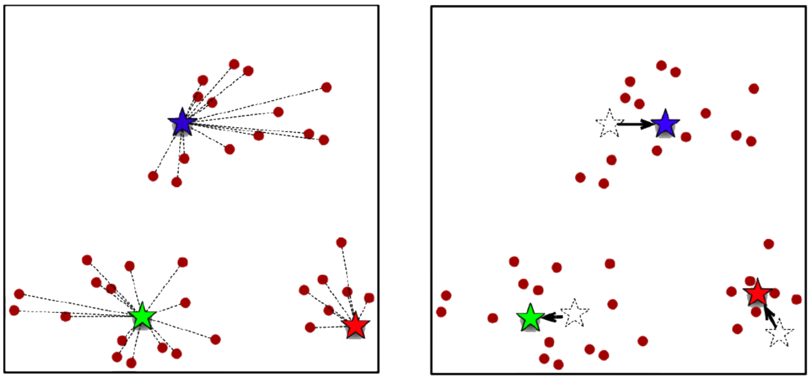


Figure 19.23: $k-$means cluster allocation (left) and updated centres (right) [author unknown].

For $k-$means, cluster centroids are obtained by averaging all points in the cluster. For $k-$medians and $k-$mode, the centrality measure is replaced by the obvious candidate.

This simple algorithm has numerous strengths:

- it is elegant and **easy to implement** (without actually having to compute pairwise distances), and so is extremely common as a result;
- in many contexts, it is a **natural way** to look at grouping observations, and
- it provides a first-pass **basic understanding of the data structure**.

On the other hand,

- it can only assign an instance to **one** cluster, which can lead to overfitting – a more robust solution would be to compute the probability of belonging to each cluster, perhaps based on the distance to the centroid;
- it requires the "true" underlying clusters to be **gaussian-** or blob-shaped, and it will fail to produce useful clusters if that assumption is not met in practice,
- it does not allow for **overlapping** or **hierarchical** groupings.

Notes Let us now return to some issues relating to clustering **in general** (and not just to $k-$means):

No matter the choice of algorithm, clustering rests on the assumption that **nearness of observations** (in whatever metric) is linked with **object similarity**, and that **large distances** are linked with **dissimilarity**. While there are plenty of situations where this is an appropriate assumption to make (temperature readings on a map, for instance), there are others where it is unlikely to be the case (chocolate bars and sensationalist tabloids at a grocery's checkout, say).

The **lack of a clear-cut definition** of what a cluster actually is (see Figure 19.24 for an example) makes it difficult to validate clustering results. Much more can be said on the topic [4].

The fact that various algorithms are **non-deterministic** is also problematic – clustering schemes should never be obtained using **only one** algorithmic pass, as the outcome could be different depending on the **location** of random starting positions and the distance/similarity metric in use.

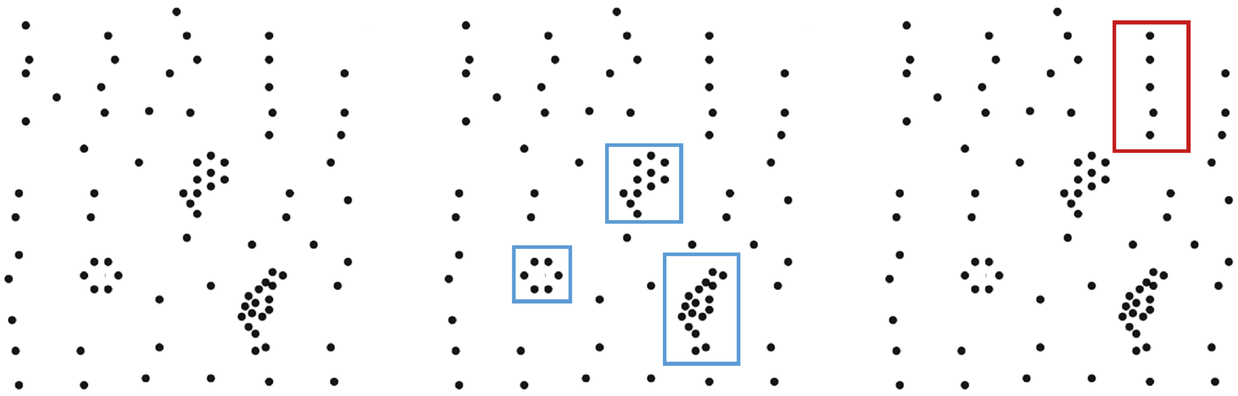


Figure 19.24: Cluster suggestions in an artificial dataset: suggested (blue), rejected (red).

But this apparent fickleness is not necessarily a problem: **essential patterns** may emerge if the algorithms are implemented multiple times, with different starting positions and re-ordered data (see **cluster ensembles** [4]). For those algorithms that require the **number of clusters** as an input, it may be difficult to determine what the optimal number should be (see Figure 19.25 for an illustration).

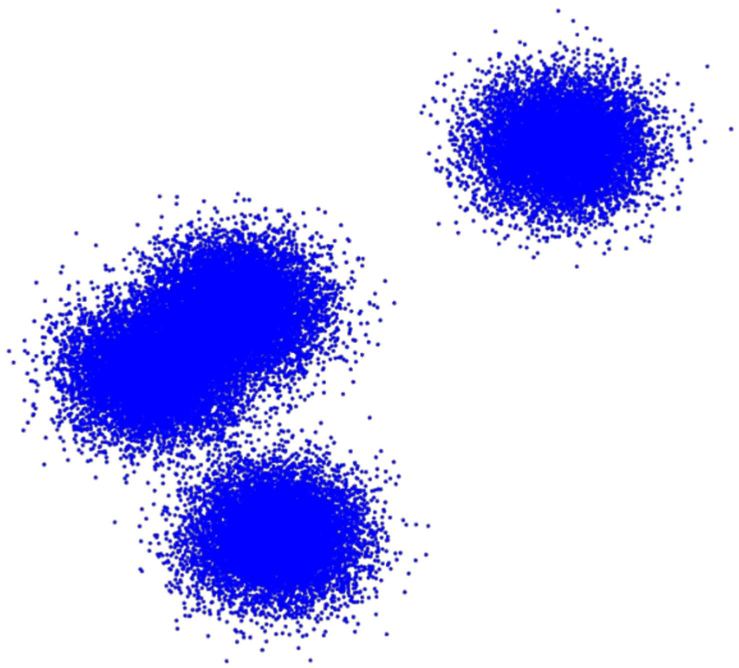


Figure 19.25: The number of clusters in a dataset is ambiguous: are there 2, 3, 4+ clusters in this example?

This number obviously depends on the choice of algorithm/metric, the underlying data, and the use that will be made of the resulting clusters; a dataset could have 3 natural groups when seen through the lens of $k-$means, but only 2 clusters for a specific choice of parameter values in DBSCAN, and so on.

This problem could be overcome by producing clustering schemes (from the same family of algorithms) with an increasing number of clusters and to plot the average distance of a cluster member to its cluster representative (centroid) against the number of clusters. Any kink in the plot represents a number of clusters at which an increase does not provide an in-step increase in clustering "resolution", so to speak (see Figure 19.30 in the *Toy Example: Iris Dataset* subsection for an illustration).

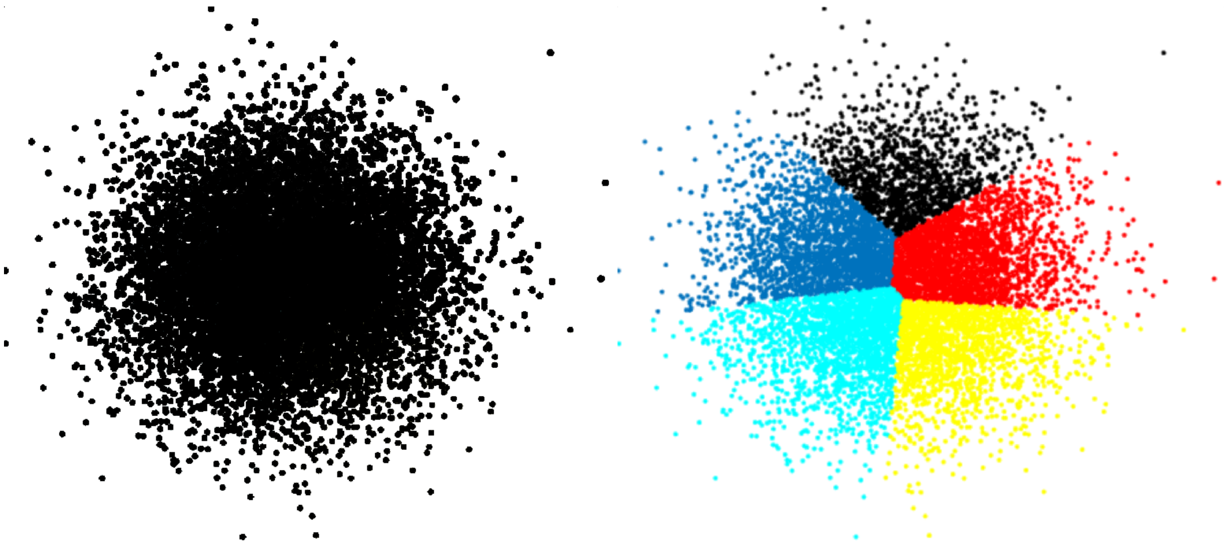


Figure 19.26: An illustration of ghost clustering with $k-$means, for $k = 5$.

And even when a cluster scheme has been accepted as valid, a **cluster description** might be difficult to come by – should clusters be described using representative instances or average values or some combination of its' members most salient features? Although there are exceptions, the ease with which clusters can be described often provides an indication about how **natural** the groups really are.

One of the most frustrating aspects of the process is that most methods will find clusters in the data even if there are none – although DBSCAN is exempt from this **ghost clustering** phenomenon (see Figure 19.26 for a $k-$means example).

Finally, analysts should beware (and resist) the temptation of ***a posteriori* rationalisation** – once clusters have been found, it is tempting to try to "explain" them; why are the groups as they have been found? But that is a job for domain experts, at best, and a waste of time and resources, at worst. Tread carefully.

19.5.4 Clustering Validation

What does it mean for a clustering scheme to be **better** than another? What does it mean for a clustering scheme to be **valid**? What does it mean for a single cluster to be **good**? **How many** clusters are there in the data, really?

These are not easy questions to answer. In general, asking if a clustering scheme is the right one or a good one is meaningless – much better to ask if it is **optimal** or **sub-optimal**, potentially in comparison to other schemes.

An **optimal** clustering scheme is one which

- **maximizes separation** between clusters;
- **maximizes similarity** within groups;
- agrees with the **human eye test**, and
- is **useful** at achieving its goals.

There are 3 families of **clustering validation** approaches:

- **external**, which use additional information (but the labels in question might have very little to do with the similarity of the observations);
- **internal**, which use only the clustering results (shape and size of clusters, etc), and
- **relative**, which compare across a number of clustering attempts.

In order to illustrate some of the possibilities, consider a dataset with **clustering scheme** $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_N\}$, where $\mathcal{C}_m$'s centroid is denoted by c_m, and the average distance of $\mathcal{C}_m$'s members to c_m is denoted by s_m. The **Davies-Bouldin Index** is defined as

$$\text{DB}_{\mathcal{C}} = \frac{1}{N} = \sum_{i=1}^{N} \max_{j \neq i} \left\{ \frac{s_i + s_j}{d(c_i, c_j)} \right\},$$

where d is the selected distance metric. Since $\text{DB}_{\mathcal{C}}$ is only defined using the clustering results, it is an **internal validation method**.

Heuristically, if the **cluster separation is small**, we might expect $d(c_i, c_j)$ to be (relatively) small, and so $\text{DB}_{\mathcal{C}}$ should be (relatively) large. In the same vein, if the clusters are **heterogeneous**, we might expect $s_i + s_j$ to be (relatively) large, and so $\text{DB}_{\mathcal{C}}$ should be (relatively) large.

In short, when the clustering scheme is **sub-optimal**, $\text{DB}_{\mathcal{C}}$ is "large". This suggests another way to determine the optimal number of clusters – pick the scheme with minimal $\text{DB}_{\mathcal{C}}$ (see Figure 19.30, which uses a modified version of the index, for an illustration).

Other **cluster quality metrics** exist, including **SSE**, **Dunn's Index**, the **Silhouette Metric**, etc. [4, 227].

19.5.5 Case Study: Pittsburgh Livehoods

When we think of similarity at the urban level, we typically think in terms of neighbourhoods. Is there some other way to identify similar parts of a city?

In *The Livehoods Project: Utilizing Social Media to Understand the Dynamics of a City* [156], Cranshaw *et al.* study the social dynamics of urban living spaces with the help of clustering algorithms.

Objective The researchers aims to draw the boundaries of **livehoods**, areas of similar character within a city, by using clustering models. Unlike **static** administrative neighborhoods, the livehoods are defined based on the habits of people who live there.

Methodology The case study introduces **spectral clustering** to discover the **distinct geographic areas** of the city based on its inhabitants' collective **movement patterns**. Semi-structured interviews are also used to **explore**, **label**, and **validate** the resulting clusters, as well as the urban dynamics that shape them.

Livehood clusters are built using the following methodology:

1. a **geographic distance** is computed based on pairs of check-in venues' coordinates;
2. **social similarity** between each pair of **venues** is computed using cosine measurements,
3. spectral clustering produces **candidate livehoods** clusters;
4. interviews are conducted with residents in order to **validate** the clusters discovered by the algorithm.

Data The data comes from two sources, combining 11 million (a recommendation site for venues based on users' experiences) check-ins from the dataset of Chen et al. [228] and a new dataset of 7 million Twitter check-ins downloaded between June and December of 2011.

For each check-in, the data consists of the **user ID**, the **time**, the **latitude and longitude**, the **name of the venue**, and its **category**.

In this case study, it is livehood clusters from the city of Pittsburgh, Pennsylvania, that are examined *via* 42,787 check-ins of 3840 users at 5349 venues.

Strengths and Limitations of the Approach

- The technique used in this study is **agnostic** towards the particular source of the data: it is not dependent on meta-knowledge about the data.
- The algorithm may be prone to "majority" bias, consequently misrepresenting/hiding minority behaviours.
- The dataset is built from a **limited** sample of check-ins shared on Twitter and are therefore biased towards the types of visits/locations that people typically want to share **publicly**.
- Tuning the clusters is non-trivial: experimenter bias may combine with "confirmation bias" of the interviewees in the validation stage – if the researchers are themselves residents of Pittsburgh, will they see clusters when there are none?

Procedures The Livehoods project uses a **spectral clustering model** to provide structure for local **urban areas** (UAs), grouping close Foursquare venues into clusters based on both the **spatial proximity** between venues and the **social proximity** which is derived from the distribution of people that check-in to them.

The guiding principle of the model is that the "character" of an UA is defined both by the types of venues it contains and by the people frequent them as part of their daily activities. These clusters are referred to as **Livehoods**, by analogy with more traditional neighbourhoods.

Let V be a list of Foursquare venues, A the associated **affinity matrix** representing a measure of similarity between each venue, and $G_m(A)$ be the graph obtained from the A by linking each venue to its nearest m neighbours. Spectral clustering is implemented as follows:[37]

1. Compute the diagonal degree matrix $D_{ii} = \sum_j A_{ij}$;
2. Set the Laplacian matrix $L = D - A$ and

$$L_{\text{norm}} = D^{-1/2} L D^{-1/2};$$

[37] We will discuss spectral clustering and other clustering algorithms in detail in Chapter 22, *Spotlight on Clustering*.

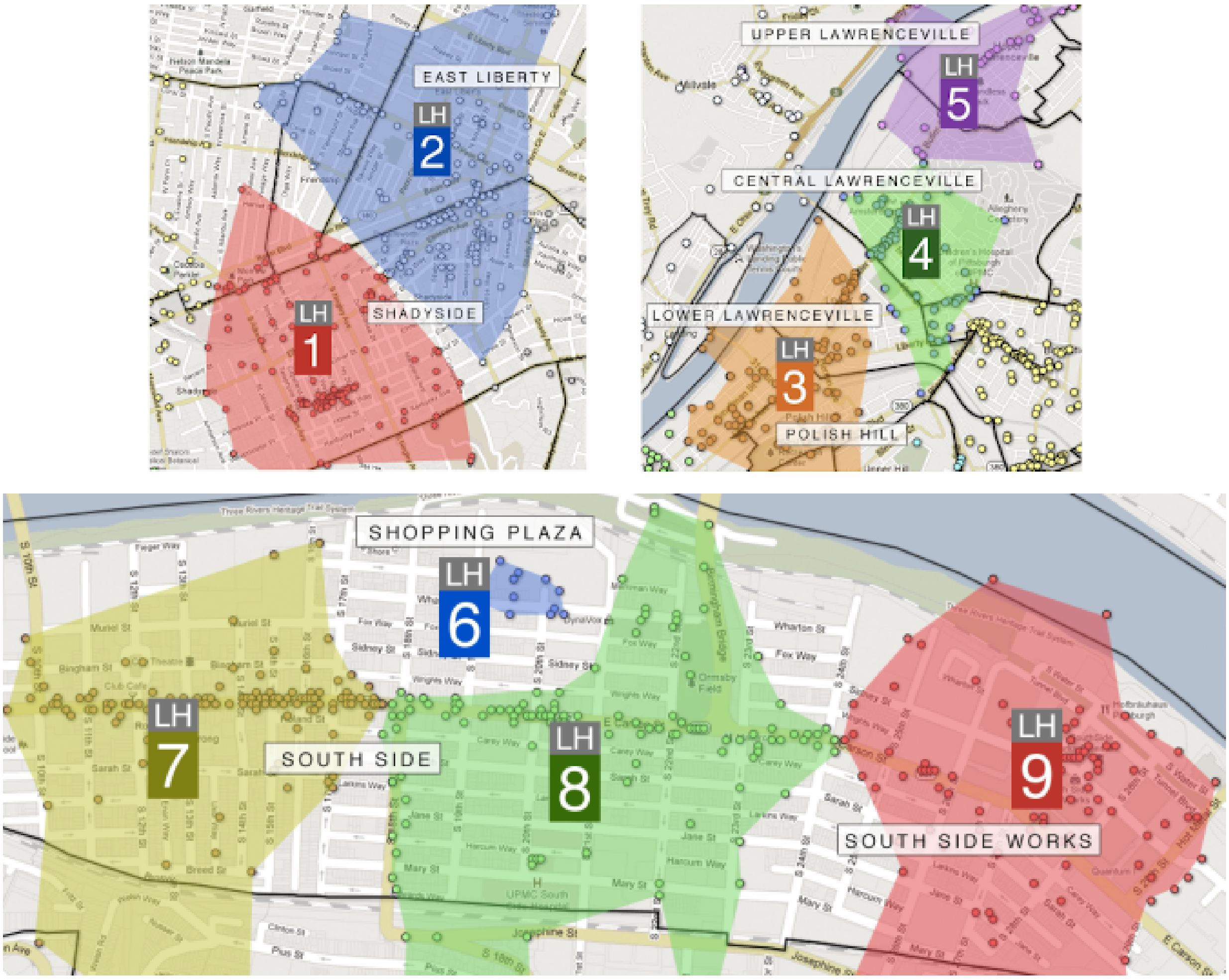


Figure 19.27: Some livehoods in metropolitan Pittsburgh, PA: Shadyside/East Liberty, Lawrenceville/Polish Hill, and South Side. Municipal borders are shown in black.

3. Find the k smallest eigenvalues of L_{norm}, where k is the index which provides the biggest jump in successive eigenvalues of eigenvalues of L_{norm}, in increasing order;
4. Find the eigenvectors $e_1, \ldots e_k$ of L corresponding to the k smallest eigenvalues;
5. Construct the matrix E with the eigenvectors $e_1, \ldots e_k$ as columns;
6. Denote the rows of E by $y_1, \ldots, y_n$, and cluster them into k clusters $C_1, \ldots, C_k$ using k-means. This induces a clustering $\{A_1, \ldots, A_k\}$ defined by
$$A_i = \{j \mid y_j \in C_i\};$$
7. For each A_i, let $G(A_i)$ be the subgraph of $G_m(A)$ induced by vertex A_i; split $G(A_i)$ into connected components; add each component as a new cluster to the list of clusters, and remove the subgraph $G(A_i)$ from the list;
8. Let b be the area of bounding box containing coordinates in the set of venues V, and b_i be the area of the box containing A_i; if $\frac{b_i}{b} > \tau$, delete cluster A_i, and redistribute each of its venues $v \in A_i$ to the closest A_j under the distance measurement.

Results, Evaluation and Validation The parameters used for the clustering were $m = 10$, $k_{\text{min}} = 30$, $k_{\text{max}} = 45$, and $\tau = 0.4$. The results for three areas of the city are shown in Figure 19.27. In total, 9 livehoods have been identified and validated by 27 Pittsburgh residents; the article has more information on this process.

- **Municipal Neighborhoods Borders:** livehoods are dynamic, and evolve as people's behaviours change, unlike the fixed neighbourhood borders set by the city government.
- **Demographics:** the interviews displayed strong evidence that the demographics of the residents and visitors of an area often play a strong role in explaining the divisions between livehoods.
- **Development and Resources:** economic development can affect the character of an area. Similarly, the resources (or lack there of) provided by a region has a strong influence on the people that visit it, and hence its resulting character. This is assumed to be reflected in the livehoods.
- **Geography and Architecture:** the movements of people through a certain area is presumably shaped by its geography and architecture; livehoods can reveal this influence and the effects it has over visiting patterns.

Take-Away While this is a neat example of practical clustering, its main take-away, from our perspective, is to remind everyone that $k-$means is not the sole clustering algorithm in applications!

19.5.6 Toy Example: Iris Dataset

Iris is a genus of plants with showy flowers. The `iris` dataset contains 150 observations of 5 attributes for specimens collected by Anderson, mostly from a Gaspé peninsula's pasture in the 1930s [229].[38]

The attributes are:

- **petal width**
- **petal length**
- **sepal width**
- **sepal length**
- **species** (virginica, versicolor, setosa)

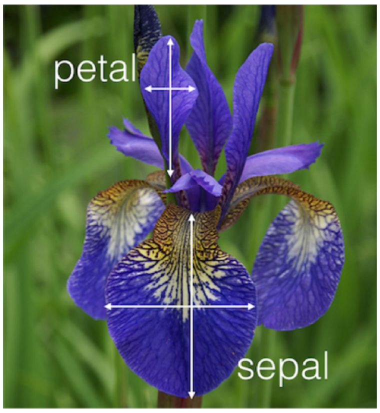
This dataset has become synonymous with data analysis, being used to showcase just about every algorithm under the sun. That is, sadly, also what we are going to do in this section.[39]

A **principal component projection** of the dataset, with species indicated by colours, is shown in Figure 19.28 (left).

From an **unsupervised learning** point of view, one question of interest is whether the observations form natural groupings, and, if so, whether these groupings correspond to the (known) species.

We use the $k-$means algorithm with Euclidean distance to resolve the problem. Since we do not know how many clusters there should be in the data (the fact that there are 3 species does not mean that there should be 3 clusters), we run 40 replicates for $k = 2, \dots, 15$.

[38]:



[39] Note that the iris dataset has started being phased out in favour of the penguin dataset [230], for reasons that do not solely have to do with its overuse (hint: take a look at the name of the journal that published Fisher's paper).

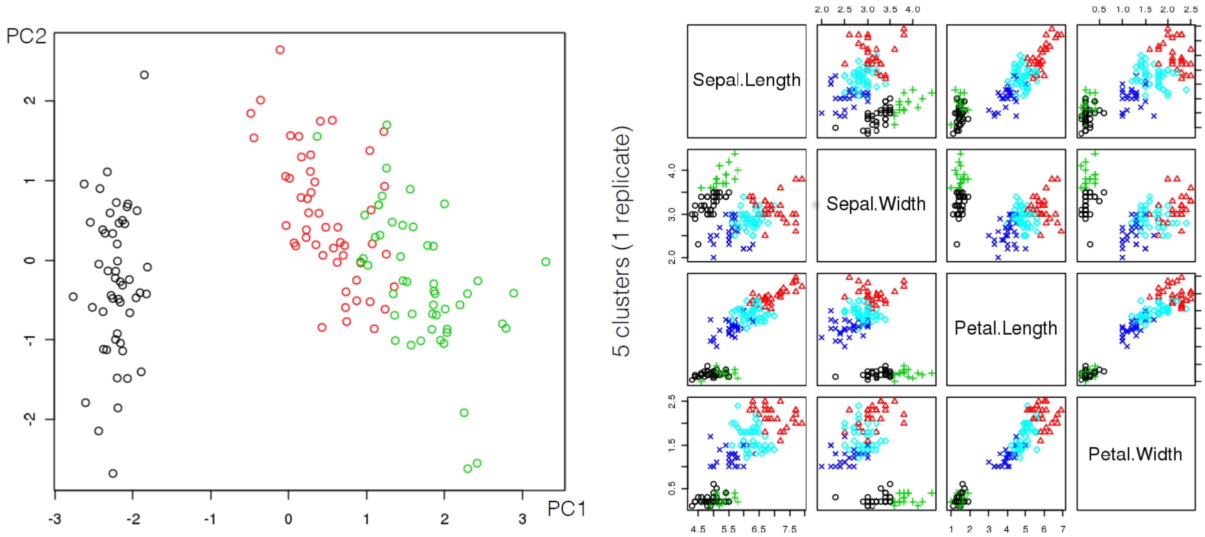


Figure 19.28: Classification of the iris dataset's 3 species, projected on the first 2 principal components (left); optimal clustering results for the iris dataset – one replicate, $k = 5$ (right).

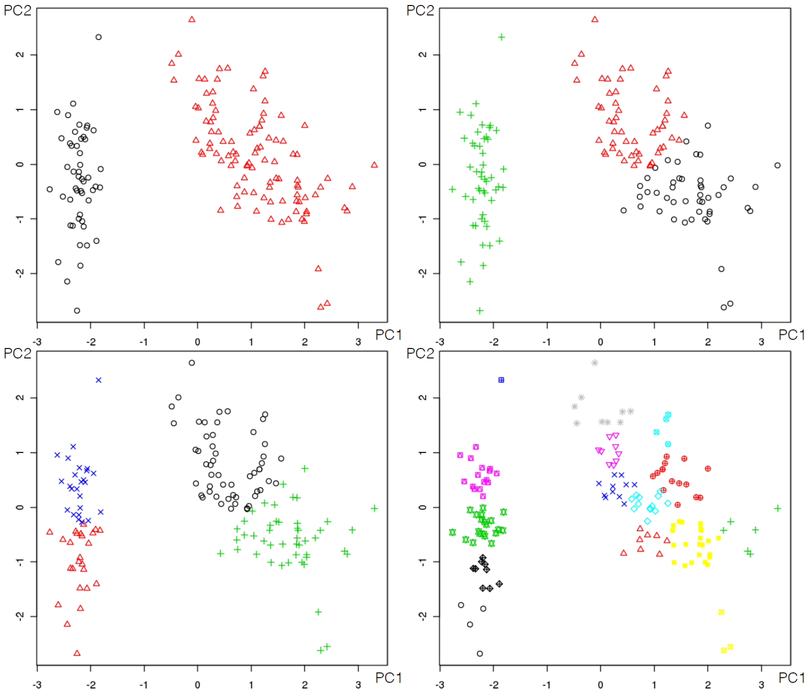


Figure 19.29: Clustering results on the iris dataset with k−means, for $k = 2, 3, 4, 15$ (from left to right).

For each replicate, we compute a (modified) **Davies-Bouldin Index** and the **Sum of Squared Errors** of the associated clustering schemes (see Figure 19.30 for the output) – the validation curves seem to indicate that there could be either 3 of 5 natural k−means clusters in the data. Is this a surprising outcome?

A single replicate with $k = 5$ is shown in Figure 19.28 (right). Would you consider this representative final clustering scheme to be meaningful? We show how to obtain these clustering results *via* R in Section 19.7 (*Clustering: Iris Dataset*).

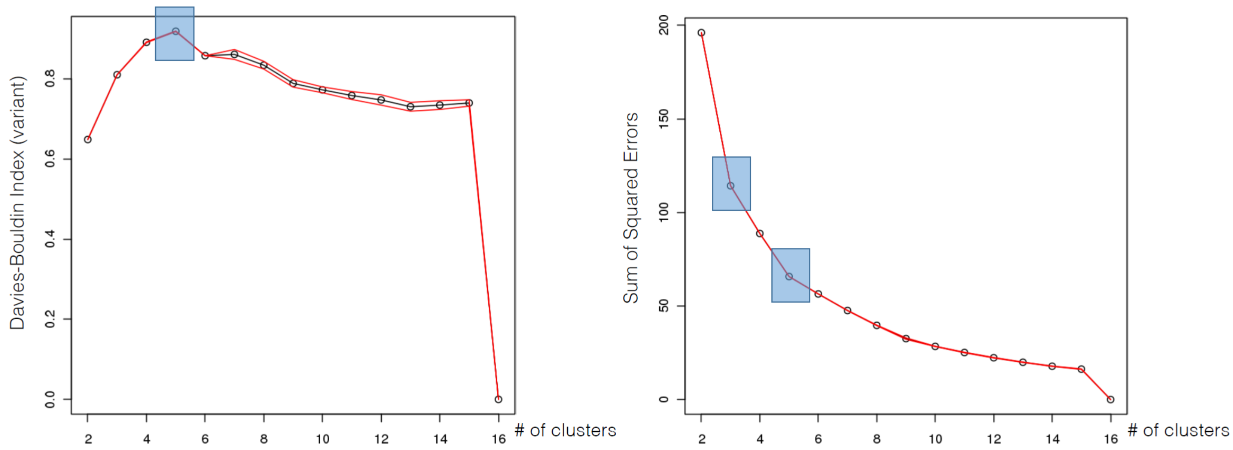


Figure 19.30: Optimal clustering results for the iris dataset: 5 clusters using (modified) Davies-Bouldin index and Sum of Squared Errors.

19.6 Issues and Challenges

> "We all say we like data, but we don't. We like getting insight out of data. That's not quite the same as liking data itself. In fact, I dare say that I don't quite care for data, and it sounds like I'm not alone." [231]

19.6.1 Bad Data

The main difficulties with data is that it is not always **representative** of the situation that we would like to model and that it might not be **consistent** (the collection and collation methods may have changed over time, say). There are other potential data issues [231]:

- the data might be formatted for human consumption, not machine readability;
- the data might contain lies and mistakes;
- the data might not reflect reality, and
- there might be additional sources of bias and errors (imputation bias, replacing extreme values with average values, proxy reporting, etc.).

Seeking perfection in the data beyond a "reasonable" threshold[40] can hamper the efforts of analysts: different quality requirements exist for academic data, professional data, economic data, government data, military data, service data, commercial data, etc. It can be helpful to remember the engineering dictum: "close enough is good enough"![41] The challenge lies in defining what is "close enough" for the application under consideration.

Even when all (most?) data issues have been mitigated, there remains a number of common **data analysis pitfalls**:

- analyzing data **without understanding the context**;
- using **one and only one tool** (by choice or by fiat) – neither the "cloud", nor Big Data, nor Deep Learning, nor Artificial Intelligence will solve all of an organization's problems;

[40] This threshold is difficult to establish exactly, however.

[41] In terms of completeness, coherence, correctness, and accountability.

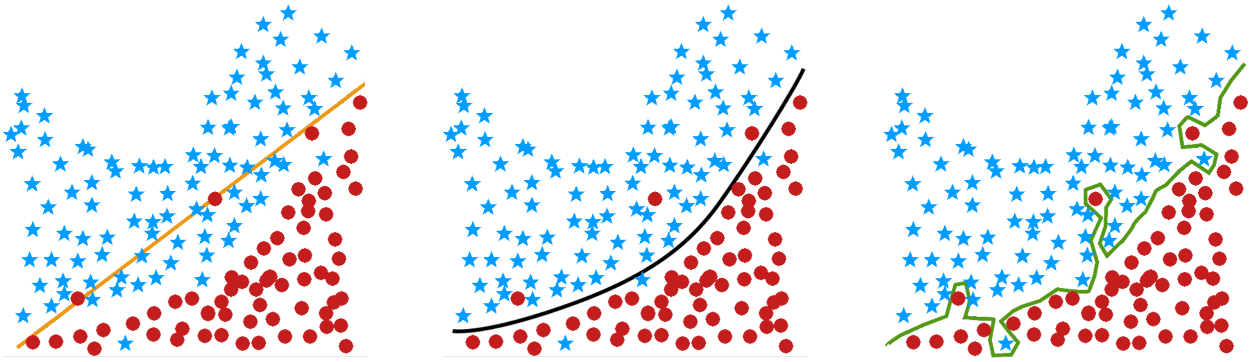


Figure 19.31: Illustration of underfitting (left) and overfitting (right) for a classification task – the optimal classifier (middle) might reach a compromise between accuracy and simplicity.

- analyzing data just for the sake of analysis,
- having **unrealistic expectations** of data analysis/DS/ML/AI – in order to optimize the production of actionable insights from data, we must first recognize the methods' domains of application and their limitations.

19.6.2 Overfitting/Underfitting

In a traditional statistical model, $p-$values and goodness-of-fit statistics are used to validate the model. But such statistics cannot always be computed for predictive data science models. We recognise a "good" model based on how well **it performs on unseen data**.

In practice, training sets and ML methods are used to search for **rules** and **models** that are **generalizable to new data** (or validation/testing sets).

Problems arise when knowledge that is gained from supervised learning does not generalize properly to the data. Ironically, this may occur if the rules or models **fit the training set too well** – in other words, the results are too specific to the training set (see Figure 19.31 for an illustration of **overfitting** and **underfitting**).

A simple example may elucidate further. Consider the following rules regarding hair colour among humans:

- **vague rule** – some people have black hair, some have brown hair, some blond, and some red;[42]
- **reasonable rule** – in populations of European descent, approximately 45% have black hair, 45% brown hair, 7% blond and 3% red, and
- **overly specific rule** – in every 10,000 individuals of European descent, we predict there are 46.32% with black hair, 47.27% with brown hair, 6.51% with blond hair, and 0.00% with red hair.[43]

With the overly specific rule, we would predict that there are no redheads in populations of European descent, which is false. This rule is **too specific** to the particular training subset that was used to produce it.[44]

More formally, underfitting and overfitting can be viewed as resulting from the **level of model complexity** (see Figure 19.32).

[42] This is obviously "true", but too general to be useful for predictions.

[43] This rule presumably emerges from redhead-free training data.

[44] We could argue that the data was simply not representative – using a training set with redheads would yield a rule that would make better predictions. But "over-reporting/overconfidence" (which manifest themselves with the use of significant digits) is also part of the problem.

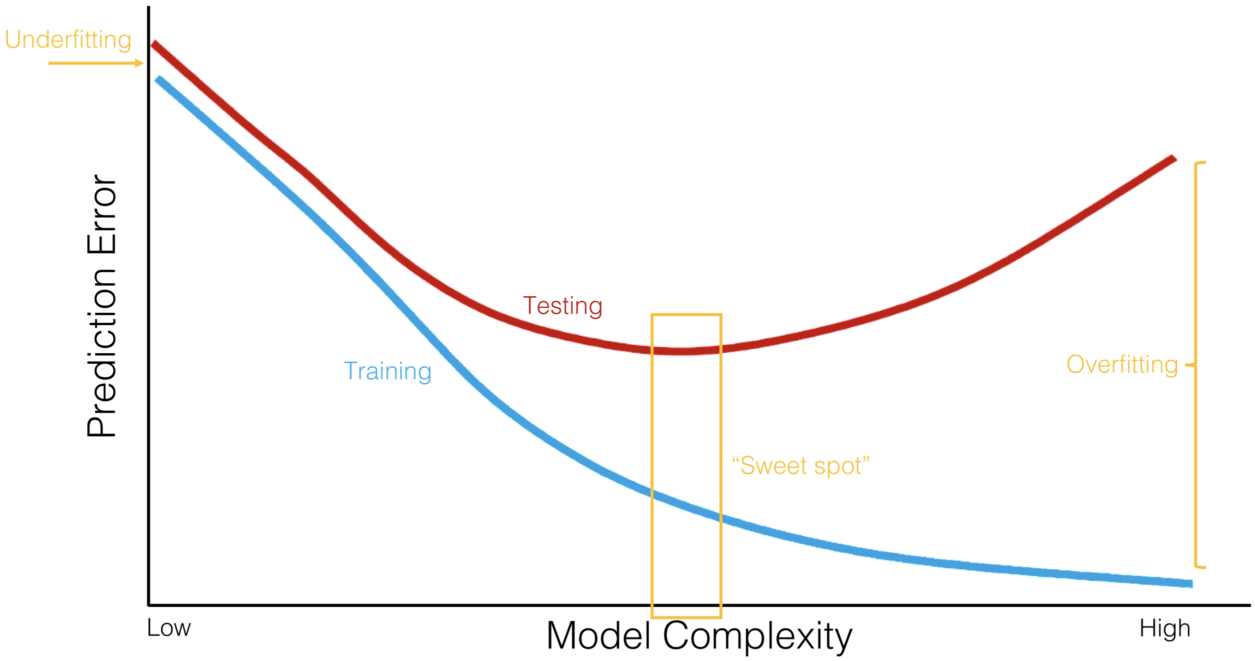


Figure 19.32: Underfitting and overfitting as a function of model complexity; error prediction on training sample (blue) and testing sample (red). High error prediction rates for simple models are a manifestation of underfitting; large difference between error prediction rates on training and testing samples for complex models are a manifestation of overfitting. Ideally, model complexity is chosen to reach the situation's 'sweet spot'; fishing for the ideal scenario might diminish explanatory power (based on [2]).

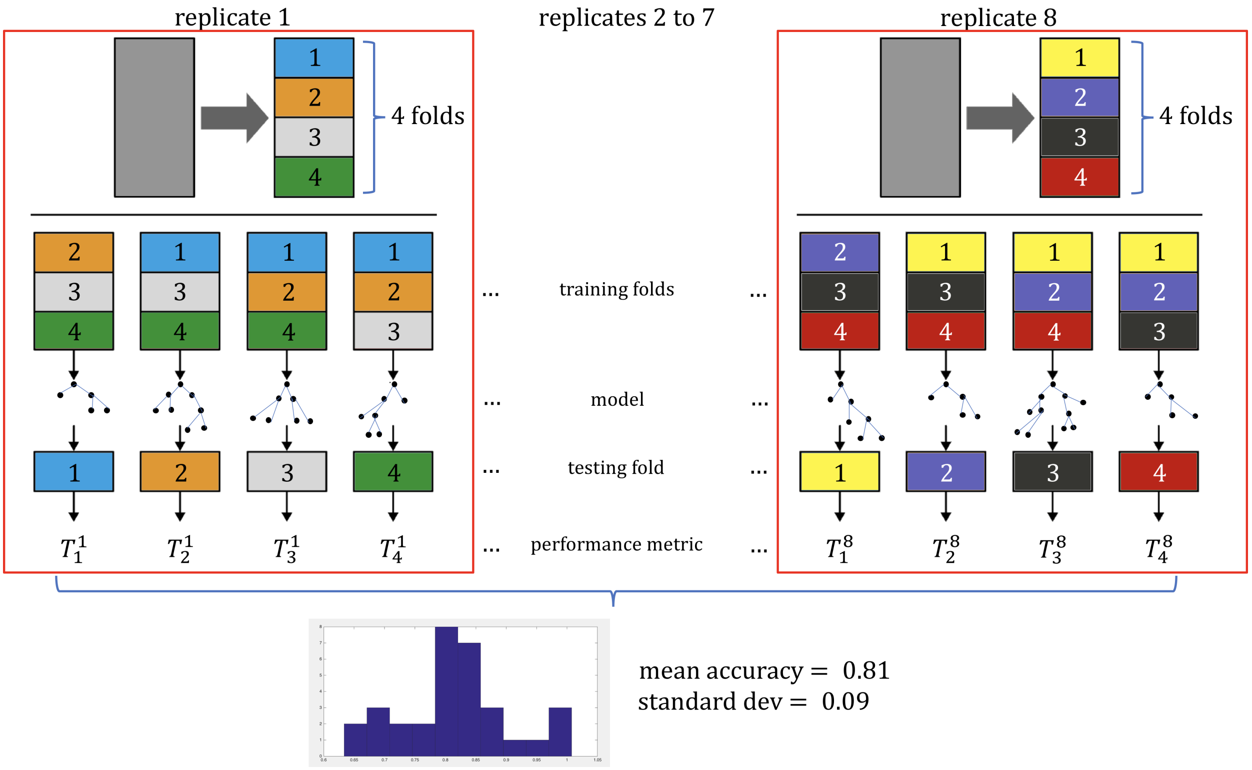


Figure 19.33: Schematic illustration of cross-fold validation, for 8 replicates and 4 folds; $8 \times 4 = 32$ models from a given family are built on various training sets (consisting of 3/4 of the available data – the training folds). Model family performance is evaluated on the respective holdout folds; the distribution of the performance metrics (in practice, some combination of the mean/median and standard deviation) can be used to compare various model families (based on [103, 168]).

Underfitting can be overcome by using more complex models (or models that use a larger proportion of a dataset's variables). Overfitting, on the other hand, can be overcome in several ways:

- **using multiple training sets** (ensemble learning approaches), with overlap being allowed – this has the effect of reducing the odds of finding spurious patterns based on quirks of the training data;
- **using larger training sets** may also remove signal which is too specific to small training sets: a 70%/30% split is often suggested, and
- **using simpler models** (or models that use a dataset with a reduced number of variables as input).

When using multiple training sets, the size of the dataset may also affect the suggested strategy: when faced with

- **small datasets** (less than a few hundred observations, say, but that depends on numerous factors such as computer power and number of tasks), use 100-200 repetitions of a **bootstrap procedure** [3];
- **average-sized datasets** (less than a few thousand observations), use a few repetitions of 10-fold cross-validation [3, 103] (see Figure 19.33 for an illustration), and
- **large datasets**, use a few repetitions of a holdout split (70%/30%?).

No matter which strategy is eventually selected, the machine learning approach requires ALL models to be evaluated on **unseen data**.[45]

19.6.3 Appropriateness and Transferability

Data science models will continue to be used heavily in the near future; while there are pros and cons to their use on ethical and other non-technical grounds, their applicability is also driven by **technical considerations**.

DS/ML/AI methods are **not** appropriate if:

- existing (legacy) datasets absolutely must be used instead of ideal/appropriate datasets;[46]
- the dataset has attributes that usefully predict a value of interest, but these attributes **are not available** when a prediction is required (e.g. the total time spent on a website may be predictive of a visitor's purchases, but the prediction must be made before the total time spent on the website is known), and
- class membership or numerical outcome is going to be predicted using an unsupervised learning algorithm.[47]

Every model makes certain assumptions about what is and is not **relevant** to its workings, but there is a tendency to only gather data which is **assumed** to be relevant to a particular situation. If the data is used in other contexts, or to make predictions depending on attributes for which no data is available, then there might be no way to **validate the results**.[48] This is not just an esoteric consideration: **over-generalizations and inaccurate predictions can lead to harmful results**.

[45] These issues will be revisited in Chapters 20 (*Regression and Value Estimation*) and 21 (*Spotlight on Classification*).

[46] "It's the best data we have!" does not mean that it is the right data, or even good data.

[47] For instance, clustering loan default data might lead to a cluster contains many defaulters – if new instances get added to this cluster, should they automatically be viewed as loan defaulters?

[48] For instance, can we use a model that predicts whether a borrower will default on a mortgage or not to also predict whether a borrower will default on a car loan or not? The problem is compounded by the fact that there might be some link between mortgage defaults and car loan defaults, but the original model does not necessarily takes this into account.

19.6.4 Myths and Mistakes

We end this section by briefly repeating various **data science myths**, originally found in [232]:

1. DS is about algorithms;
2. DS is about predictive accuracy;
3. DS requires a data warehouse;
4. DS requires a large quantity of data, and
5. DS requires only technical experts,

as well as common **data analysis mistakes** [same source]:

1. selecting the wrong problem;
2. getting by without metadata understanding;
3. not planning the data analysis process;
4. insufficient business/domain knowledge;
5. using incompatible data analysis tools;
6. using tools that are too specific;
7. favouring aggregates over individual results;
8. running out of time;
9. measuring results differently than the client, and
10. naïvely believing what one is told about the data.

It remains the analyst's and/or the consultant's responsibility to address these issues with the stakeholders and/or clients, **the earlier, the better**. It is safer to assume that not everyone is on the same page – prod and ask, early and often.

19.7 R Examples

We provide the R code that was used to produce the outputs of the toy examples in Sections 19.3, 19.4, and 19.5.

19.7.1 ARM: Titanic

This example refers to the Titanic dataset toy example of Section 19.3. The very first step in programming withRis to import data.

Setting up the Titanic dataset

```
class = as.factor(c(rep("3rd",52),rep("1st",118),
   rep("2nd",154),rep("3rd",387),rep("Crew",670),
   rep("1st",4),rep("2nd",13.01),rep("3rd",89),
   rep("Crew",3),rep("1st",5),rep("2nd",11),
   rep("3rd",13),rep("1st",1),rep("2nd",13),
   rep("3rd",14),rep("1st",57),rep("2nd",14),
   rep("3rd",75),rep("Crew",192),rep("1st",140),
   rep("2nd",80),rep("3rd",76),rep("Crew",20)))
sex = as.factor(c(rep("Male",35),rep("Female",17),
   rep("Male",1329),rep("Female",109),rep("Male",29),
   rep("Female",28),rep("Male",338),rep("Female",316)))
```

```
age = as.factor(c(rep("Child",52),rep("Adult",1438),
   rep("Child",57),rep("Adult",654)))
survived = as.factor(c(rep("No",1490),rep("Yes",711)))
titanic = data.frame(class,sex,age,survived)
```

We briefly explore the structure of data.

Summary data

```
summary(titanic)
table(titanic$age,titanic$survived)
table(titanic$class,titanic$survived)
```

class	sex	age	survived
1st: 325	Female: 470	Adult: 2092	No: 1490
2nd: 285	Male: 1731	Child: 109	Yes: 711
3rd: 706			
Crew: 885			

age/survived	No	Yes	class/survived	No	Yes
Adult	1438	654	**1st**	122	203
Child	52	57	**2nd**	167	118
			3rd	528	178
			Crew	673	212

Then we use the `arules` package function `apriori()`, which returns all possible rules (built by the apriori algorithm). By default, `apriori()` creates rules with minimum support of 0.1, minimum confidence of 0.8, and maximum of 10.

Original apriori rules

```
rules.titanic <- arules::apriori(titanic)
```

```
Apriori parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.8    0.1    1 none FALSE            TRUE       5     0.1      1
 maxlen target  ext
     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 220

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[10 item(s), 2201 transaction(s)] done [0.00s].
sorting and recoding items ... [9 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
```

```
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [27 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

For a rule $X \longrightarrow Y$, arule evaluates a number of default metrics. We can extract the rules using the function inspect(): there are 27 in total.

Inspecting the original rules

```
arules::inspect(rules.titanic)
```

```
     lhs                                        rhs           support   confidence
[1]  {}                                      => {age=Adult}   0.9504771 0.9504771
[2]  {class=2nd}                             => {age=Adult}   0.1185825 0.9157895
[3]  {class=1st}                             => {age=Adult}   0.1449341 0.9815385
...
[26] {class=Crew, sex=Male, survived=No}  => {age=Adult}   0.3044071 1.0000000
[27] {class=Crew, age=Adult, survived=No} => {sex=Male}    0.3044071 0.9955423

     coverage  lift      count
[1]  1.0000000 1.0000000 2092
[2]  0.1294866 0.9635051  261
[3]  0.1476602 1.0326798  319
...
[26] 0.3044071 1.0521033  670
[27] 0.3057701 1.2658514  670
```

We set our own parameters to create a new list of rules, with minimum support of 0.005 and minimum confidence of 0.8. As we are naturally interested in finding combinations of attributes associated with survival (either Yes or No), we only retain rule for which the conclusion is {survived=No} or {survived=Yes}.

There are 12 such rules.

Constrained apriori rules

```
rules.titanic.2 <- arules::apriori(titanic,
     parameter = list(minlen=2, supp=0.005, conf=0.8),
     appearance = list(rhs=c("survived=No","survived=Yes"),
                       default="lhs"),
     control = list(verbose=F))
```

We then sort the list by the lift value (in descending order) and print the results:

Sorting the constrained rules

```
rules.titanic.2 <- arules::sort(rules.titanic.2,
                      by=c("lift"), decreasing=TRUE)
```

Inspecting the constrained rules

```
arules::inspect(rules.titanic.2)
```

```
     lhs                                    rhs
[1]  {class=2nd, age=Child}              => {survived=Yes}
...
[11] {class=3rd, sex=Male, age=Adult}    => {survived=No}
[12] {class=3rd, sex=Male}               => {survived=No}

     support      confidence coverage    lift     count
[1]  0.010904134  1.0000000  0.010904134 3.095640  24
...
[11] 0.175829169  0.8376623  0.209904589 1.237379 387
[12] 0.191731031  0.8274510  0.231712858 1.222295 422
```

Are all these rules independent? If we know that

```
{class=3rd,sex=Male} => {survived=No}
```

is a rule, say, then we would not be surprised to find out that

```
{class=3rd,sex=Male,age=Adult} => {survived=No}
```

is also a rule.

The following chunk of code identifies which rules have an antecedent which is a subset of another rule's antecedent, marking one of them as **redundant**, and removing those from the set of rules, which brings us down to 8 rules:

Pruned apriori rules

```
subset.matrix <- as.matrix(arules::is.subset(
                 rules.titanic.2, rules.titanic.2))
subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA
redundant.titanic <- colSums(subset.matrix, na.rm=T) >= 1
rules.titanic.2.pruned <- rules.titanic.2[!redundant.titanic]
arules::inspect(rules.titanic.2.pruned)
```

```
    lhs                                    rhs            supp conf cove lift count
[1] {class=2nd, age=Child}              => {survived=Yes} 0.01 1.00 0.01 3.10  24
[2] {class=1st, sex=Female}             => {survived=Yes} 0.06 0.97 0.07 3.01 141
[3] {class=2nd, sex=Female}             => {survived=Yes} 0.04 0.88 0.05 2.72  93
[4] {class=Crew, sex=Female}            => {survived=Yes} 0.01 0.87 0.01 2.69  20
[5] {class=2nd, sex=Male, age=Adult} => {survived=No}  0.07 0.92 0.08 1.35 154
[6] {class=2nd, sex=Male}            => {survived=No}  0.07 0.86 0.08 1.27 154
[7] {class=3rd, sex=Male, age=Adult} => {survived=No}  0.18 0.84 0.21 1.24 387
[8] {class=3rd, sex=Male}            => {survived=No}  0.19 0.83 0.23 1.22 422
```

We have presented the "interesting" associations in tabular format, but there are a variety of graphical representations as well (available in package `arulesViz`), such as:

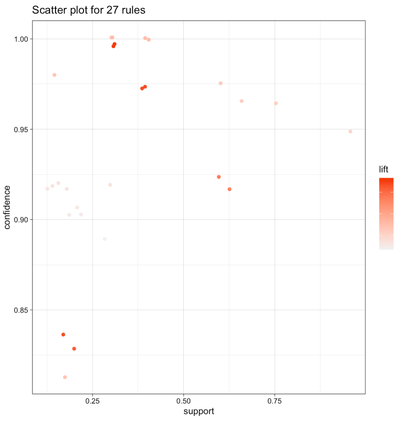
- a bubble chart;

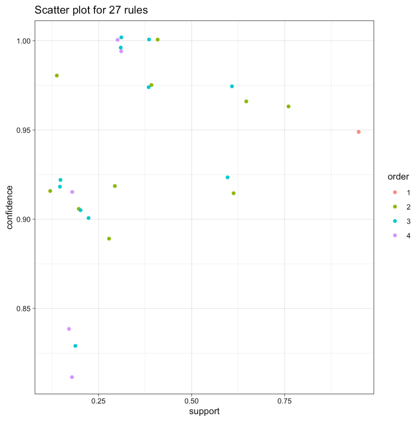
- a two-key plot (taking into account the rules' lengths);
- a graph structure, or
- parallel coordinates (where the width of the arrows represents support and the intensity of the colour represent confidence).

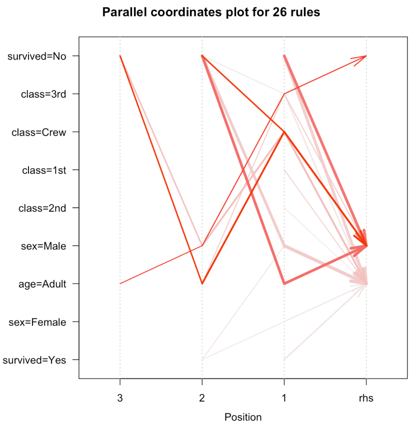
The original rules are shown below:

Visualizing the original rules

```
library(arulesViz)
plot(rules.titanic)
plot(rules.titanic, method="graph")
plot(rules.titanic, method="paracoord", control = list(reorder = TRUE))
```

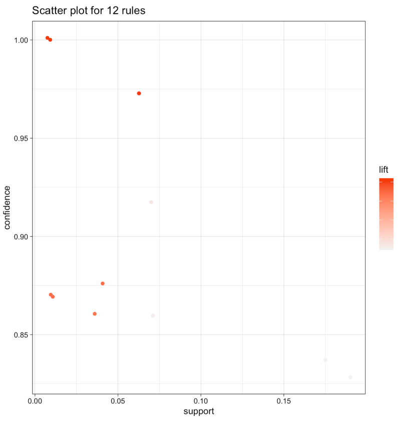


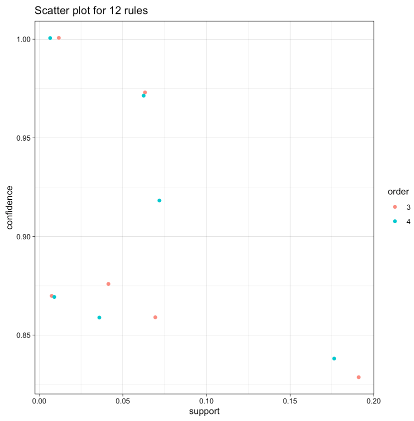


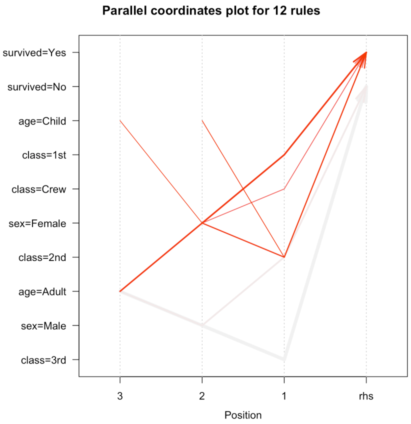


For the constrained and the pruned rules, we obtain:

```
plot(rules.titanic.2)
plot(rules.titanic.2, method="graph")
plot(rules.titanic.2, method="paracoord")
```

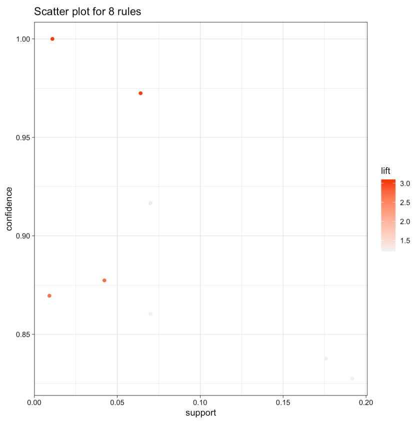


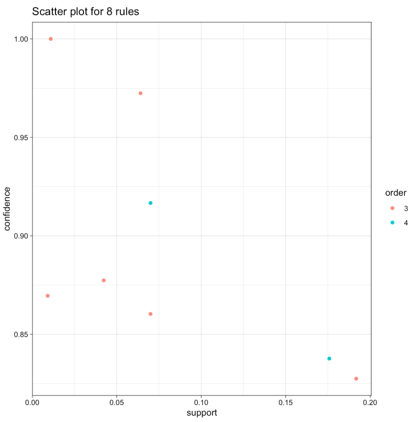


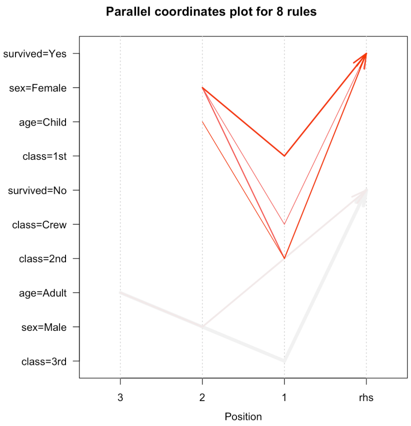


```
plot(rules.titanic.2.pruned)
plot(rules.titanic.2.pruned, method="graph")
plot(rules.titanic.2.pruned, method="paracoord")
```

Is anything surprising about these outcomes?







19.7.2 Classification: Kyphosis Dataset

This example refers to the Kyphosis dataset toy example of Section 19.4; we explore the built-in `kyphosis` dataset with two decision tree methods (`rpart()`,`ctree()`).

Let's get some information on the `kyphosis` dataset.

Getting the help file

```
?rpart::kyphosis
```

We can also determine its structure and summary statistics:

Kyphosis dataset structure

```
str(rpart::kyphosis)
```

```
'data.frame':   81 obs. of  4 variables:
 $ Kyphosis: Factor w/ 2 levels "absent","present": 1 1 2 ...
 $ Age     : int  71 158 128 2 1 1 61 37 113 59 ...
 $ Number  : int  3 3 4 5 4 2 2 3 2 6 ...
 $ Start   : int  5 14 5 1 15 16 17 16 16 12 ...
```

Summary data

```
summary(rpart::kyphosis)
```

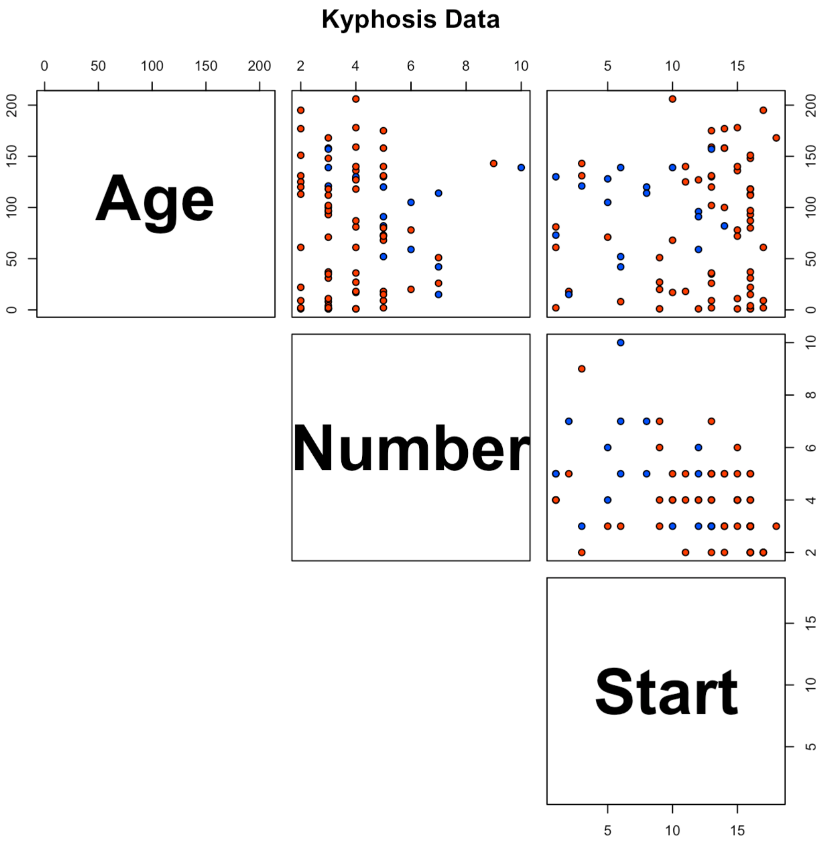
Kyphosis	Age	Number	Start
absent: 64	Min. : 1.00	Min. : 2.000	Min. : 1.00
present: 17	1st Qu.: 26.00	1st Qu.: 3.000	1st Qu.: 9.00
	Median : 87.00	Median : 4.000	Median : 13.00
	Mean : 83.65	Mean : 4.049	Mean : 11.49
	3rd Qu.: 130.00	3rd Qu.: 5.000	3rd Qu.: 16.00
	Max. : 206.00	Max. : 10.000	Max. : 18.00

As always, we should take the time to visualize the dataset. In this case, since there are 4 variables (one of which is categorical), a scatterplot matrix is probably a good approach.

Visualizing the kyphosis data

```
pairs(rpart::kyphosis[,2:4],
      main = "Kyphosis Data",
      bg = c("red", "blue")[unclass(rpart::kyphosis[,1])],
      pch = 21, lower.panel=NULL,
      cex.labels=4.5, labels=c("Age","Number","Start"),
      font.labels=2)
```

What should the legend of this scatterplot matrix be (**red**=?, **blue**=?).



We build a tree using the recursive partitioning algorithm implemented in `rpart`.[49] For the time being, we're assuming that the training set is the dataset as a whole (so there is no reason to expect that the decision trees should have predictive power, only descriptive power).

Building a recursive partition tree

```
set.seed(2) # for replicability
tree <- rpart::rpart(Kyphosis ~ Age + Number + Start,
                     method="class", data=rpart::kyphosis)
tree
```

[49] The package is called `rpart`, the function... also `rpart()`.

```
n= 81

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 81 17 absent (0.79012346 0.20987654)
   2) Start>=8.5 62  6 absent (0.90322581 0.09677419)
     4) Start>=14.5 29  0 absent (1.00000000 0.00000000) *
     5) Start< 14.5 33  6 absent (0.81818182 0.18181818)
      10) Age< 55 12  0 absent (1.00000000 0.00000000) *
      11) Age>=55 21  6 absent (0.71428571 0.28571429)
        22) Age>=111 14  2 absent (0.85714286 0.14285714) *
        23) Age< 111 7  3 present (0.42857143 0.57142857) *
   3) Start< 8.5 19  8 present (0.42105263 0.57894737) *
```

We can access the method results by using `printcp()` (although how informative this output will prove depends on the expectations. . .)

Getting information about the tree

```
rpart::printcp(tree)
```

```
Classification tree:
rpart::rpart(formula = Kyphosis ~ Age + Number + Start, data = rpart::kyphosis,
    method = "class")

Variables actually used in tree construction:
[1] Age   Start

Root node error: 17/81 = 0.20988

n= 81

        CP nsplit rel error xerror    xstd
1 0.176471      0   1.00000 1.0000 0.21559
2 0.019608      1   0.82353 1.2353 0.23200
3 0.010000      4   0.76471 1.2941 0.23548
```

Details on the nodes and the splits can be obtained using `summary()`.

Summarizing the tree

```
summary(tree)
```

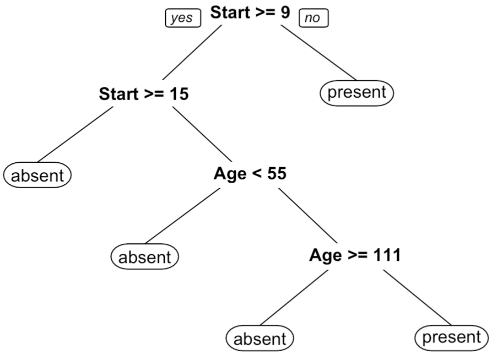
```
n= 81

          CP nsplit rel error   xerror      xstd
1 0.17647059      0 1.0000000 1.000000 0.2155872
2 0.01960784      1 0.8235294 1.235294 0.2320031
3 0.01000000      4 0.7647059 1.294118 0.2354756

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 23: 7 observations
  predicted class=present  expected loss=0.4285714  P(node) =0.08641975
    class counts:     3     4
   probabilities: 0.429 0.571
```

What are these nodes that are being referred to? Plotting the tree provides more information. Here is a basic plot:

Plotting the tree

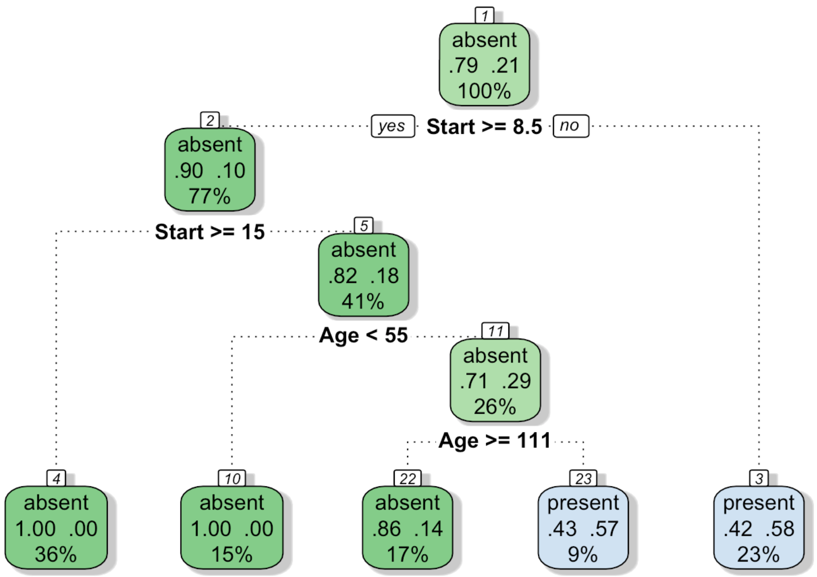
```
rpart.plot::prp(tree)
```



and a fancier one:

Plotting a fancier tree

```
rattle::fancyRpartPlot(tree,
          main="Classification Tree for Kyphosis")
```



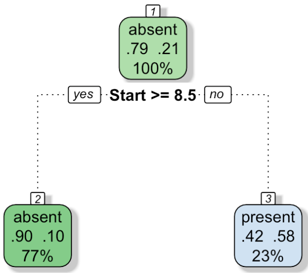
In the fancy plot, does the **intensity** of the colour play a role? What about the **percentages**? What about the **decimals**?

Unchecked tree growth usually leads to overfitting. This is typically a problem when datasets contain too many variables.[50] But overfitting is unlikely to be an issue in the case of the `kyphosis` dataset because it contains only three variables.

Nevertheless, the code below shows you how you would **prune** the growth of the tree in general, by finding a value of `cp` which maximizes `xerror` (this will be revisited in Section 21.4, *Tree-Based Methods*).

Pruning and plotting the pruned tree

```
tree2 = rpart::prune(tree, cp = 0.02)
rattle::fancyRpartPlot(tree2)
```



How good is the classification model provided by the tree? We don't have access to p−values or confidence intervals – we need to rely on the model's **confusion matrix**.

We can obtain the predictions made by the model on the object tree by using the `predict()` function. This procedure takes each observation

[50] The corresponding decision tree will contain too many splits in that case.

and feeds it to the model, outputting the likelihood of kyphosis being `absent` or `present`.

Note that the probabilities are **calibrated** - compare with the Naive Bayes method which we will see later.

The following model predicts the class probabilities for all observations:

Getting the class probabilities

```
predictions1 = predict(tree, type = "prob")
```

The first 10 predictions are:

```
head(predictions1, 10)
```

absent	present
0.4210526	0.5789474
0.8571429	0.1428571
0.4210526	0.5789474
0.4210526	0.5789474
1.0000000	0.0000000
1.0000000	0.0000000
1.0000000	0.0000000
1.0000000	0.0000000
1.0000000	0.0000000
0.4285714	0.5714286

In general, the confusion matrix requires a specific prediction (`absent` or `present`), against which we compare the actual classification. Here, we have probabilities. How can we take the probabilities and transform them into specific predictions?

Here is one way to do this.

Building the confidence matrix

```
# uniformly generate a random number (between 0 and 1)
# for each of the observations
random1 <- runif(81)

# extract the actual classification of the observations
real <- rpart::kyphosis$Kyphosis

# join together the prediction probabilities,
# the random numbers, and the actual classification
# since we're joining together text and numbers,
# cbind coerces the factors to numerical values
# absent = 1, present = 2
test1 <- cbind(predictions1,random1,real)

# this code takes advantage of the numerical presentation
# of the factors to output a specific prediction
# if random1 < prob of absent, then
# real = absent (1), otherwise real = present (2)
pred1 <- 2-(test1[,3]<test1[,1])

# add the specific predictions to the test1 dataset
test1 <- cbind(test1,pred1)
```

The confusion matrix actually depends on the random variables generated in the previous chunk of code.

In this case, the first 10 predictions would be:

```
head(test1[,c(1,2,4,5)],10)
```

absent	present	real	pred1
0.4210526	0.5789474	1	1
0.8571429	0.1428571	1	1
0.4210526	0.5789474	2	2
0.4210526	0.5789474	1	2
1.0000000	0.0000000	1	1
1.0000000	0.0000000	1	1
1.0000000	0.0000000	1	1
1.0000000	0.0000000	1	1
1.0000000	0.0000000	1	1
0.4285714	0.5714286	2	1

We can now build the confusion matrix on the model predictions using `real` (actual classification) and `pred1` (specific model prediction).

The function `table()` produces a joint distribution, where the rows correspond to the first variable in the call (actual), and the columns to the second variable (predicted).

Confusion matrix

```
table(test1[,4],test1[,5])
```

```
     1  2
  1 59  5
  2  9  8
```

Note that the confusion matrix could be different every time a new set of predictions are made. Why would that be the case?

Is this a good classification model or not?

In this example, we computed the confusion matrix using the **entire dataset**. In a sense, we should not have been surprised that the results were decent, because we were using the same data to build the model and to evaluate it.

From a predictive perspective, classification models are built on a subset of the data (the **training set**) and evaluated on the remaining data (the

testing set). The idea is that if there IS a strong classification signal, it should be found in any representative subset. As a rule, we look for training sets making up between 70% and 80% of the data. They should be selected randomly.

We start by creating a training set with 50 instances, and we fit the `rpart()` algorithm to this data:[51]

Training a recursive partition tree

```
sub <- c(sample(1:81, 50))
(fit <- rpart::rpart(Kyphosis ~ ., data = rpart::kyphosis,
                     subset = sub))
```

```
n= 50

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 50 9 absent (0.8200000 0.1800000)
  2) Start>=8.5 40 3 absent (0.9250000 0.0750000) *
  3) Start< 8.5 10 4 present (0.4000000 0.6000000) *
```

The confusion matrix has to be built on the testing set.[52] There are 2 ways to make a prediction: we either use the most likely outcome, or we generate a random vector of predictions using the probabilities for each class, as above.

We can predict on the basis of class:

Predictions on the testing set

```
table(predict(fit, rpart::kyphosis[-sub,], type = "class"),
              rpart::kyphosis[-sub, "Kyphosis"])
```

```
          absent present
  absent      19       3
  present      4       5
```

Or we can predict on the basis of probability:

```
prob.fit <- predict(fit, rpart::kyphosis[-sub,],
                    type = "prob")
random1 <- runif(31)
real <- rpart::kyphosis[-sub,"Kyphosis"]

# absent = 1, present = 2
test1 <- cbind(prob.fit,random1,real)
pred1 <- 2-(test1[,3]<test1[,1])
test1 <- cbind(test1,pred1)
table(test1[,4], test1[,5])
```

[51] If we use all variables, we do not need to specify them in the model call; we only need to use the “.”

[52] Iindicated here by `-sub`, that is, the opposite of the `sub` indices.

```
    1  2
1 20  3
2  5  3
```

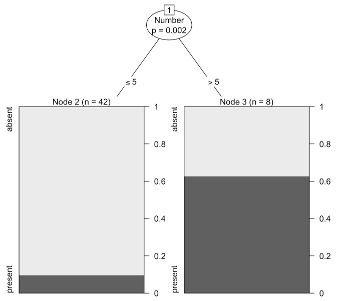
These are the confusion matrices that should be used to evaluate the decision's tree performance.

Another way to build decision trees is *via* party's `ctree()` function. Conditional inference trees have the property that they will **automatically prune** themselves once a statistical criterion is met by the tree as a whole. The downside is that they do not usually pick fully reasonable splits (they may not conform to contextual understanding).

Building and plotting a conditional inference tree

```
kyphosis.ctree <- party::ctree(Kyphosis ~ .,
        data = rpart::kyphosis, subset = sub)
kyphosis.ctree
plot(kyphosis.ctree)
```

```

     Conditional inference tree with 2 terminal nodes

Response:  Kyphosis
Inputs:  Age, Number, Start
Number of observations:  50

1) Number <= 5; criterion = 0.998, statistic = 11.641
  2)*  weights = 42
1) Number > 5
  3)*  weights = 8
```



A very simple tree, as can be seen.

Confusion matrix on the testing set

```
table(predict(kyphosis.ctree, rpart::kyphosis[-sub,]),
              rpart::kyphosis[-sub,1])
```

```
         absent present
 absent      21       7
 present      2       1
```

How good of a model is this one? Which of the `rpart()` or `ctree()` model is preferable? Does this depend on the training set?

19.7.3 Clustering: Iris Dataset

This example refers to the Iris dataset toy example of Section 19.5; we cluster the ubiquitous (built-in) `iris` dataset, *via* k-means.

The procedure is straightforward:

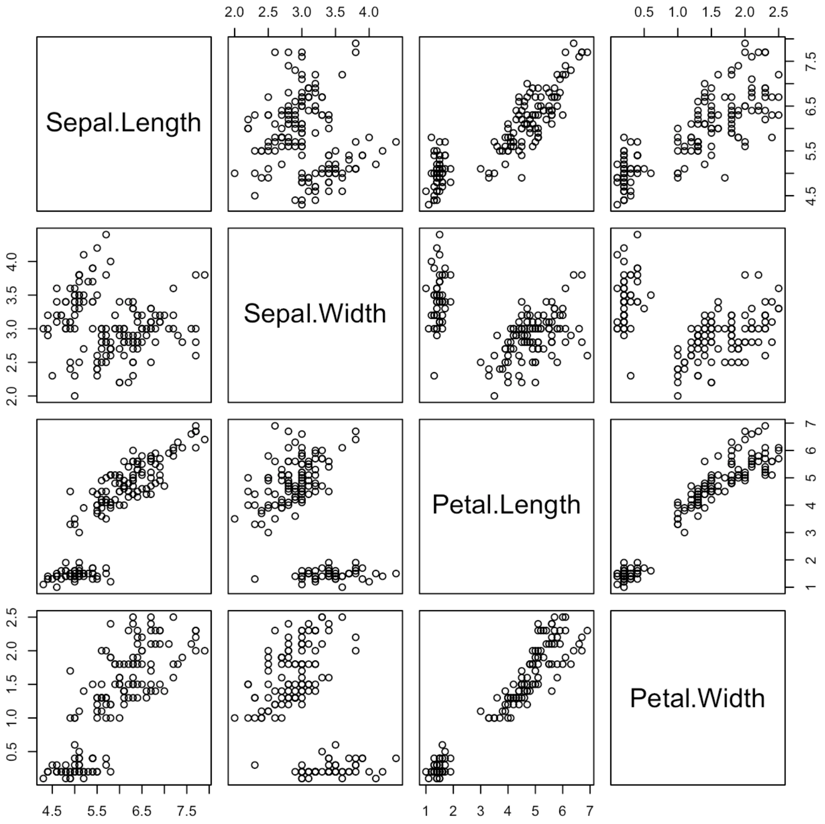
1. cluster with $n = 2, \ldots, 15$ clusters;
2. display the **Within Sum of Squares** curve, as a function of the # of clusters;
3. display the **Davies-Bouldin curve**, as a function of the # of clusters,
4. select the optimal number of clusters on the basis of these curves.

Let us load the data and take a look at the iris dataset.[53]

Visualizing the iris data

```
my.data<-iris[,1:4]
head(my.data)
pairs(my.data)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5.0	3.6	1.4	0.2
6	5.4	3.9	1.7	0.4



[53] Without the species labels, as this is an unsupervised problem.

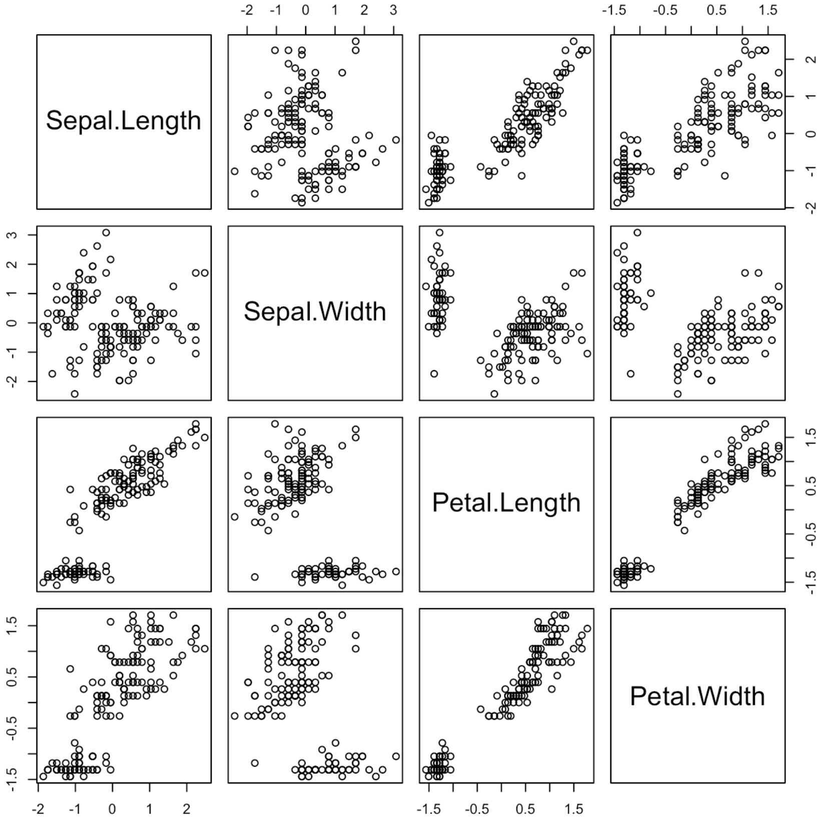
The eye test would most likely identify 2 clusters.

In preparation for the cluster analysis, we will scale the data so that all the variables are represented on the same scale. This can be done using the `scale()` function.

Scaling the data

```
my.data.scaled<-scale(my.data)
head(my.data.scaled)
pairs(my.data.scaled)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	-0.8976739	1.01560199	-1.335752	-1.311052
2	-1.1392005	-0.13153881	-1.335752	-1.311052
3	-1.3807271	0.32731751	-1.392399	-1.311052
4	-1.5014904	0.09788935	-1.279104	-1.311052
5	-1.0184372	1.24503015	-1.335752	-1.311052
6	-0.5353840	1.93331463	-1.165809	-1.048667



The “shape” of the dataset is the same, but the axis ranges have changed.

One way to reduce the dimension of the problem is to work with the **principal components** (see Chapter 23, *Feature Selection and Dimension Reduction*). The hope is that most of the variation in the data can be explained by a smaller number of derived variables, expressed as linear combinations of the original variables.

This can be accomplished in R with the `princomp()` function.

Principal Components

```
pc.agg.data = princomp(my.data.scaled)
summary(pc.agg.data)
```

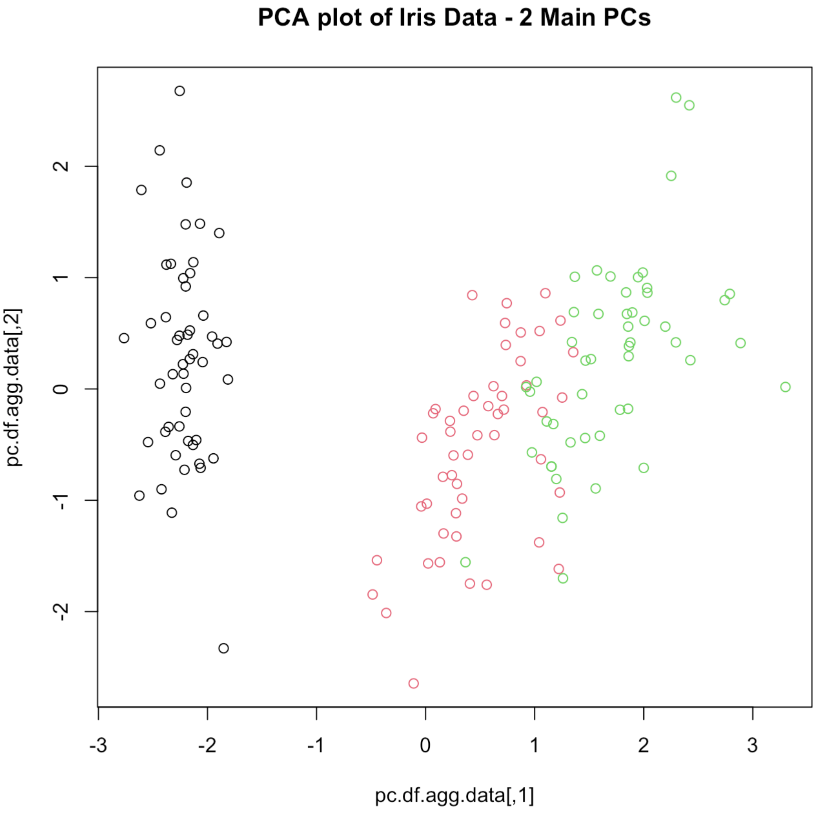
```
Importance of components:
                          Comp.1    Comp.2     Comp.3      Comp.4
Standard deviation     1.7026571 0.9528572 0.38180950 0.143445939
Proportion of Variance 0.7296245 0.2285076 0.03668922 0.005178709
Cumulative Proportion  0.7296245 0.9581321 0.99482129 1.000000000
```

This provides a summary of the "strength" of the signal in each component. The cumulative proportion of the variance is the value of interest. If 2 principal components are needed to explain 95% of the variance, we would expect that roughly 95% of the set is "2-dimensional".

As this is indeed the case, we opt to cluster the data projected on only the first 2 components, which can be accessed via the scores attribute of `pc.agg.data`. The plot below shows a 2D representation of the iris dataset obtained *via* principal components analysis (PCA).

Visualizing the PCA data

```
pc.df.agg.data = cbind(pc.agg.data$scores[,1],
                       pc.agg.data$scores[,2])
plot(pc.df.agg.data,col=iris[,5])
title('PCA plot of Iris Data - 2 Main PCs')
```



The **Davies-Bouldin** (DB) index is a measure that is used to determine the optimal number of clusters in the data. For a given dataset, the optimal number of clusters is obtained by maximizing the DB index (there are different versions of the DB index, all giving equivalent results).

```
Davies.Bouldin <- function(A, SS, m) {
  # A  - the clusters’ centres
  # SS - the within sum of squares
  # m  - the sizes of the clusters
  N <- nrow(A)   # number of clusters

  # intercluster distance
  S <- sqrt(SS/m)

  # Get the distances between centres
  M <- as.matrix(dist(A))

  # Get the ratio of intercluster/centre.dist
  R <- matrix(0, N, N)
  for (i in 1:(N-1)) {
    for (j in (i+1):N) {
      R[i,j] <- (S[i] + S[j])/M[i,j]
      R[j,i] <- R[i,j]
    }
  }
  return(mean(apply(R, 1, max)))
}
```

But we do not yet know how many clusters we will ultimately be using,[54] so we will construct $k-$means clusters for a variety of values k.

We will in fact produce 40 replicates for each $k = 2, \ldots, 15$ and track both the DB index and the **Within Sums of Squares** (SS), which is a measure of how similar observations are within each cluster, and how different they are from observations in other clusters.[55]

If the DB index does not provide a clear-cut winner, the optimal number of clusters is obtained when the slope of the SS curve flattens “drastically”.[56]

We start by setting up the number of repetitions and the loop.

Initializing the k-means process

```
# setting up the repetitions and display options
oldpar <- par(mfrow = c(4, 4))
N = 40              # Number of repetitions
max.cluster = 15    # Maximum number of desired clusters

# initializing values
m.errs <- rep(0, max.cluster)
m.DBI <- rep(0, max.cluster)
s.errs <- rep(0, max.cluster)
s.DBI <- rep(0, max.cluster)
```

[54] Although the visual inspection above suggested $k = 2$ or $k = 3$.

[55] The value for the SS can be found by calling the attribute `withinss` on a `kmeans` object.

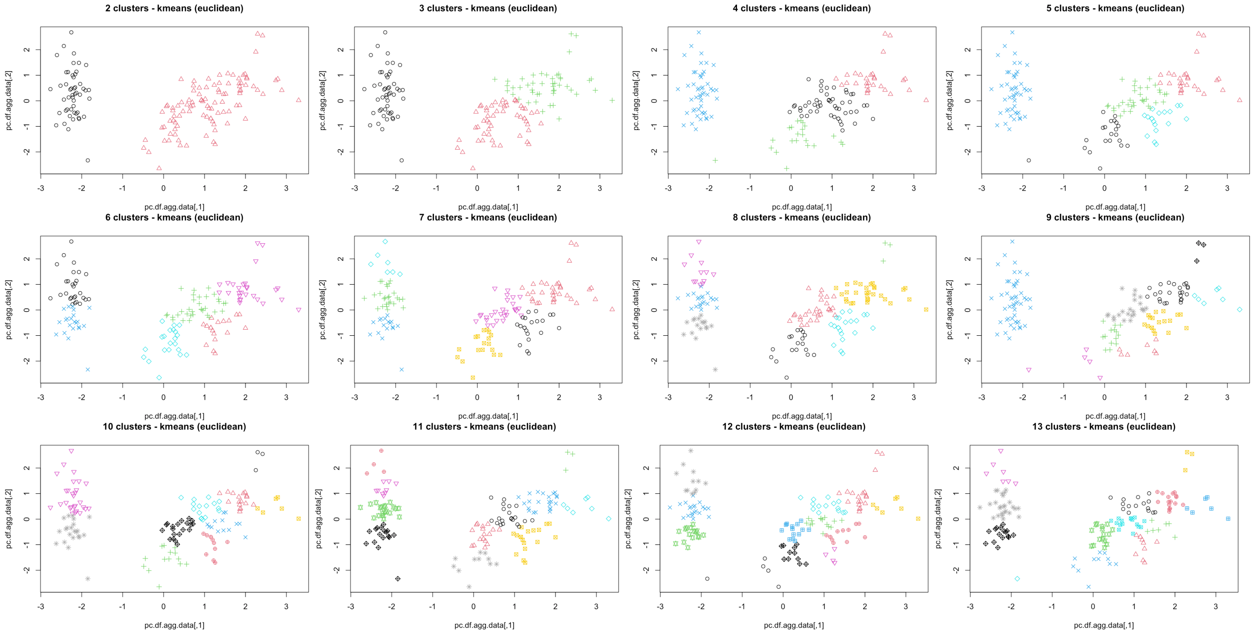
[56] So that adding more clusters does not provide as big a decrease in SS.

Now we run 40 replicates for each number of clusters (so 560 calls to the `kmeans()` clustering algorithm in total). For each clustering schemes, we compute the DB index and the SS, and store them in memory.

We also print **one** of the clustering schemes for each of the number of clusters in the iteration.

Clustering the iris data

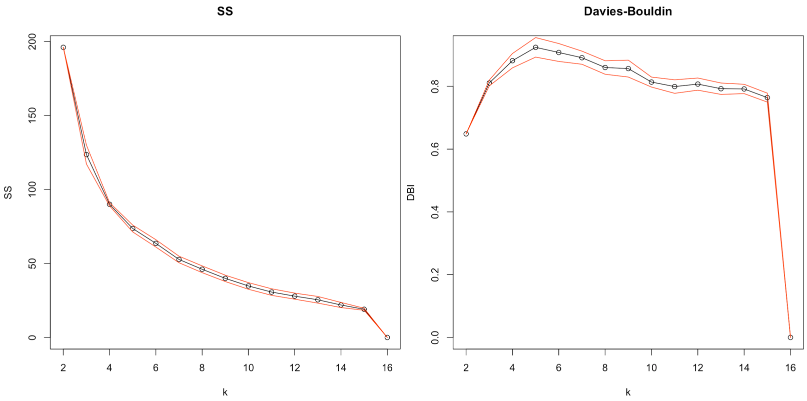
```
set.seed(0) # for replicability

## clustering and plotting
for (i in 2:max.cluster) {
  errs <- rep(0, max.cluster)
  DBI <- rep(0, max.cluster)

  for (j in 1:N) {
    KM <- kmeans(pc.df.agg.data, iter.max = 10,  i)
    errs[j] <- sum(KM$withinss)
    DBI[j] <- Davies.Bouldin(KM$centers, KM$withinss,
                             KM$size)
  }

  m.errs[i - 1] = mean(errs)
  s.errs[i - 1] = sd(errs)
  m.DBI[i - 1] = mean(DBI)
  s.DBI[i - 1] = sd(DBI)

  plot(pc.df.agg.data,col=KM$cluster, pch=KM$cluster,
       main=paste(i,"clusters - kmeans (euclidean)"))
}
```



Since we have replicates, we can compute **confidence bonds** for both the average DB index and the average SS.

Confidence bands for Within SS and DB curves

```
MSE.errs_up = m.errs + 1.96 * s.errs / sqrt(N)
MSE.errs_low = m.errs - 1.96 * s.errs / sqrt(N)

MSE.DBI_up = m.DBI + 1.96 * s.DBI / sqrt(N)
MSE.DBI_low = m.DBI - 1.96 * s.DBI / sqrt(N)

# Within SS curve
plot(2:(max.cluster+1), m.errs, main = "SS",
    xlab="k", ylab="SS")
lines(2:(max.cluster+1), m.errs)
par(col = "red")
lines(2:(max.cluster+1), MSE.errs_up)
lines(2:(max.cluster+1), MSE.errs_low)
par(col = "black")

# DBI curve
plot(2:(max.cluster+1), m.DBI, main = "Davies-Bouldin",
    xlab="k", ylab="DBI")
lines(2:(max.cluster+1), m.DBI)
par(col="red")
lines(2:(max.cluster+1), MSE.DBI_up)
lines(2:(max.cluster+1), MSE.DBI_low)
par(col = "black")
```



Where is the DB curve maximized? Does it match what the SS curve shows? We pick the optimal number of clusters using the following:

```
(i_choice <- which(
    m.DBI==max(m.DBI[1:(length(m.DBI)-1)]))+1)
```

```
[1] 5
```

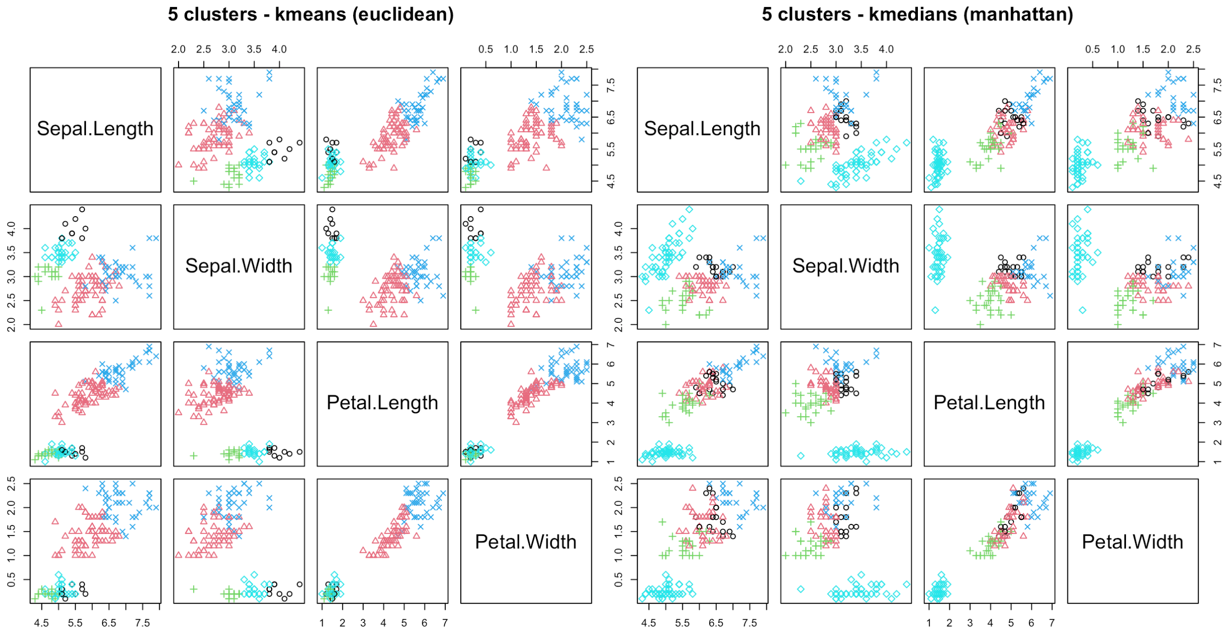
Finally, let us plot a "final" realization of the clustering scheme with the optimal number of clusters. We cluster on the PCA-reduced scaled data, but we plot the results with the original iris data.

We will also verify if we get similar clustering schemes when we use a different distance measure (the default measure in `kmeans()` is the

Euclidean metric). Let us try the manhattan distance (k-medians) with the `cclus()` method (available with the `flexclust` package, which provides a more flexible clustering approach, including different algorithms and distances).

Comparison of k-means and k-medians

```
# k-means
KM <- kmeans(agg.data, iter.max = 10, i_choice)
plot(iris[,1:4],col=KM$cluster, pch=KM$cluster,
    main=paste(i_choice,"clusters - kmeans (euclidean)"))

# k-medians
library(flexclust)
KMed <- cclust(pc.df.agg.data, i_choice, dist="manhattan")
plot(iris[,1:4], col=predict(KMed), pch=predict(KMed),
     main=paste(i_choice,"clusters - kmed (manhattan)"))
```


5 clusters - kmeans (euclidean)

5 clusters - kmedians (manhattan)

How do they compare to one another?

19.8 Exercises

1. What are some examples of supervised and unsupervised learning tasks in the business world? In a public policy/government setting? In a scientific setting?
2. Assuming that data mining techniques are used in the following cases, identify whether the required task falls under supervised or unsupervised learning [232].

    a. Deciding whether to issue a loan to an applicant based on demographic and financial data (with reference to a database of similar data on prior customers).

b. In an online bookstore, making recommendations to customers concerning additional items to buy based on the buying pattern in prior transactions.

c. Identifying a network data packet as dangerous (virus, hacker attack) based on comparison to other packets with a known threat status.

d. Identifying segments of similar customers.

e. Predicting whether a company will go bankrupt based on comparing its financial data to those of similar bankrupt and non-bankrupt firms.

f. Estimating the repair time required for an aircraft based on a trouble ticket.

g. Automated sorting of mail by zip code scanning.

h. It is more difficult and expensive to win new customers than it is to retain existing customers. Scoring each customer on their likelihood to quit can help an organization design effective interventions, such as discounts or free services, to retain profitable customers in a cost-effective manner.

i. Some medical practitioners conduct unnecessary tests and/or over-bill their government or insurance companies. Using audit data, it may be possible to identify such providers and take appropriate action.

j. A market basket analysis can help develop predictive models to determine which products often sell together. This knowledge of affinities between products can help retailers create promotional bundles to push non-selling items along a set of products that sell well.

k. Diagnosing the cause of a medical condition is the crucial first step in medical engagement. In addition to the current condition, other factors can be considered, including the patient's health history, medication history, family's history, and other environmental factors. A predictive model can absorb all of the information available to date (for this patient and others) and make probabilistic diagnoses, in the form of a decision tree, taking away most of the guess work involved.

l. Schools can develop models to identify students who are at risk of not returning to school. Such students can be flagged to be on the receiving end of potential corrective measures.

m. In addition to customer data, telecom companies also store call detail records (CDR), which precisely describe the calling behaviour of each customer. The unique data can be used to profile customers, who may be marketed to based on the similarity of their CDR to other customers'.

n. Statistically, all equipment is likely to break down at some point in time. Predicting which machine is likely to shut down is a complex process. Decision models to forecast machinery failure could be constructed using past data, which can lead to savings provided by preventative maintenance.

o. Identifying which tweets contain disinformation and which tweets are legitimate.

3. Would the results of the *Danish medical study* (see Section 19.3) be applicable to the Canadian context? To the Chinese context? What do you think some of the ethical/technical challenges were?

4. Evaluate the following candidate association rules for the British Musical Dataset introduced in Section 19.3:

   - If an individual owns a classical music album (W), then they also own a hip-hop album (Z), given that Freq(W) = 2010, Freq(Z) = 6855, and Freq($W \cap Z$) = 132.
   - If an individual owns both the Beatles' *Sergeant Peppers' Lonely Hearts Club Band* and a classical music album, then they were born before 1976, given that Freq($Y \cap W$) = 1852 and Freq($Y \cap W \cap X$) = 1778.

5. Out of the 3 rules that have been established in the previous question ($X \to Y$, $W \to Z$, and (Y AND W) $\to X$), which do you think is more useful? Which is more surprising?
6. A store that sells accessories for smart phones runs a promotion on faceplates. Customers who purchase multiple faceplates from a choice of 6 different colours get a discount. The store managers, who would like to know what colours of faceplates are likely to be purchased together, collected past transactions in the file `Transactions.csv`. Consider the following rules:

   - {red, **w**hite} → {**g**reen}
   - {**g**reen} → {**w**hite}
   - {red, **g**reen} → {**w**hite}
   - {**g**reen} → {red}
   - {**o**range} → {red}
   - {**w**hite, **b**lack} → {**y**ellow}
   - {**b**lack} → {**g**reen}

   a. For each rule, compute the support, confidence, interest, lift, and conviction.
   b. Amongst the rules for which the support is positive (> 0), which one has the highest lift? Confidence? Interest? Conviction?
   c. Build an additional 5-10 candidate rules (randomly), and evaluate them. Which of the 12-17 candidate rules do you think would be most useful for the store managers?
   d. How would one determine reasonable threshold values for the support, coverage, interest, and lift of rules derived from a given dataset?

7. Consider the following datasets:

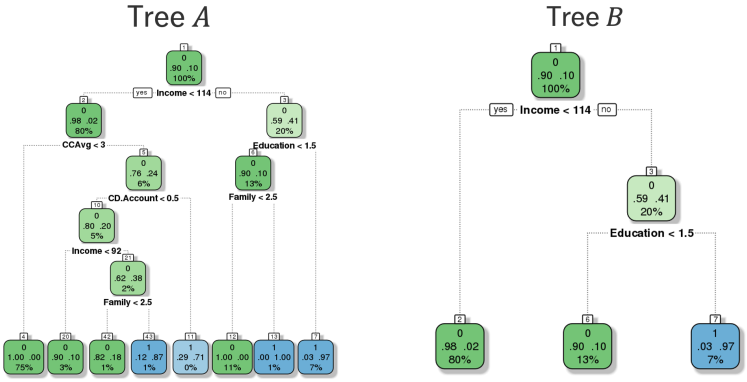
   - `GlobalCitiesPBI.csv`
   - `2016collisionsfinal.csv`
   - `polls_us_election_2016.csv`
   - `HR_2016_Census_simple.xlsx`, and
   - `Transactions.csv`.

   a. Determine what the data is reporting on / what it is about / create a "data dictionary" to explain the different fields and variables in the dataset.
   b. Develop a list of questions you would like to answer about the data.
   c. Investigate variables (individual, bivariate, multivariate) through charts, distributions, variable interactions, summary statistics, etc.

d. Do you trust the data or not? Why? If you don't trust it, flag some potential issues with the data/specific entries.

e. Conduct an association rule mining analysis of the datasets. Using either the brute force approach or the apriori algorithm, determine 10-20 strong association rules. Visualize them, and interpret their results.

8. *UniversalBank* is looking at converting its liability customers (i.e., customers who only have deposits at the bank) into asset customers (i.e., customers who have a loan with the bank). In a previous campaign, *UniversalBank* was able to convert 9.6% of 5000 of its liability customers into asset customers. The marketing department would like to understand what combination of factors make a customer more likely to accept a personal loan, in order to better design the next conversion campaign. `UniversalBank.csv` ↗ contains data on 5000 customers, including the following measurements: age, years of professional experience, yearly income (in thousands of USD), family size, value of mortgage with the bank, whether the client has a certificate of deposit with the bank, a credit card, etc. They build 2 decision trees on a training subset of 3000 records to predict whether a customer is likely to accept a personal loan (1) or not (0).



a. Explore the `UniversalBank.csv` ↗ dataset. Can you come up with a reasonable guess as to what each of the variables represent?

b. How many variables are used in the construction of tree *A*? Of tree *B*?

c. Are the following decision rules valid or not for trees *A* and/or *B*?

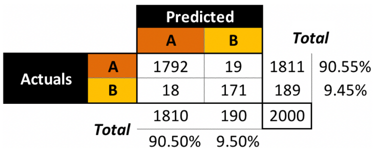
- IF (Income $\geq$ 114) AND (Education $\geq$ 1.5) THEN (Personal Loan = 1)
- IF (Income $<$ 92) AND (CCAvg $\geq$ 3) AND (CD.Account $<$ 0.5) THEN (Personal Loan = 0)

d. What prediction would trees *A* and *B* make for a customer with:

- a yearly income of 94,000$USD (Income = 94);
- 2 kids (Family = 4);
- no certificate of deposit with the bank (CD.Account = 0);
- a credit card interest rate of 3.2% (CCAvg = 3.2), and
- a graduate degree in Engineering (Education = 3)?

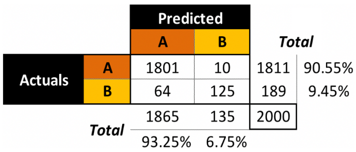
9. The confusion matrices for the predictions of trees A and B on the remaining 2000 testing observations are shown below.

Tree A



		Predicted			
		A	B	Total	
Actuals	A	1792	19	1811	90.55%
	B	18	171	189	9.45%
Total		1810	190	2000	
		90.50%	9.50%		

Tree B



		Predicted			
		A	B	Total	
Actuals	A	1801	10	1811	90.55%
	B	64	125	189	9.45%
Total		1865	135	2000	
		93.25%	6.75%		

   a. Using the appropriate matrices, compute the 9 performance evaluation metrics for each of the trees (on the testing set).
   b. If customers who would not accept a personal loan get irritated when offered a personal loan, what tree should *UniversalBank*'s marketing group use to help maintain good customer relations?

10. Consider the `algae_blooms.csv` of Section 16.5.3. We try to build a model to predict the presence/absence of algeas based on various chemical properties of river water. What is the data science motivation for such a model? After all, we can simply analyze water samples to determine if various harmful algae are present or absent. The answer is simple: chemical monitoring is cheap and easy to automate, whereas biological analysis of samples is expensive and slow. Another answer is that analyzing the samples for harmful content does not provide a better understanding of algae drivers: it just tells us which samples contain algae.
   a. Load the data and summarize/visualize it: you will be tasked with predicting the presence/absence of algae a1 and a2.
   b. Clean the data and impute missing values, as needed.
   c. Remove 20% of the observations for a validation set.
   d. Create a training/testing pair on the remaining 80% of the observations and train 2 decision trees to predict the presence/absence of algaes a1 and a2, respectively. Evaluate the performance of each model. Which models performs best on your training/testing pair?
   e. Repeat step d) on at least 20 distinct training/testing pairs. Evaluate the performance of each model, and save them.
   f. For each algae, pick the best of the models (how would you determine this) and use it to make predictions for the readings in the validation set. Evaluate.
   g. Instead of picking the best of the 20+ models, find some way to combine the results of the 20 models and to make predictions for the readings in the validation set. Evaluate these predictions.
   h. Which of the resulting models of steps 6 or 7 provide the best performance? Which are easier to interpret?

11. Repeat question 10, using the same validation set in part c). In part d), use the remaining 80% of the data to build a decision tree (do not split into a training/testing pair first). Use these models to make predictions for the readings in the validation set. Evaluate these predictions. Is there evidence of overfitting?

12. Repeat question 10, using the same validation set in part c). In parts d) to g), use decision stumps (decision trees with only 1 branching point) instead of full growth trees. Is there evidence of underfitting?

13. The population of Canada is divided physically into provincial and territorial areas, most of which are further subdivided into health regions. The Census information (from 2016) ↗ is available for those health regions. The equivalent 2018 dataset has been clustered to produce peer groups: the result is shown here ↗. The data is found in the file `HR_2016_Census_simple.xlsx` ↗.

    a. Load the data and summarize/visualize it (extract the rows with a 4-digit geocode).
    b. Clean the data and impute missing values (if necessary). Scale the data and assign to a new set.
    c. Run the $k-$means algorithm (with Euclidean distance) on the scaled data, using ALL the features, for $k = 3, \ldots, 16$. Use the Davies-Bouldin index and the Within-SS index to determine the optimal number of clusters. Is the optimal clustering scheme plausible?

14. Reduce the dimension of the health region dataset by running a principal component analysis (PCA) and keep the principal components that explain up to 80% of the variability in the data. Repeat step c). Are the results significantly different than they were for question 13?
15. Run $k-$means on the original health regions data (previous question) and on the reduced data, for the same range of $k-$values, but replicate the process 30+ times per value of k. What are the optimal k values in the aggregate runs?
16. Save the cluster assignments for each run with the optimal values of k found in question 13. Say that two observations A and B have similarity $w(A, B) \in [0, 1]$ if A and B lie in the same cluster in $w(A, B)$% of the runs. What are some observations with high similarity measurements? With low similarity measurements?
17. Provide a $k-$means clustering schema for the *UniversalBank* dataset.
18. The remaining exercises use the Gapminder Tools ↗ (there is also an offline version ↗).

    a. In the default configuration, we can identify some potential association rules. Using visual and ballpark estimates, evaluate the performance of the following rules:

        - Income > 8000 ⟶ Life Expectancy > 70
        - Income < 8000 AND Life Expectancy < 70 ⟶ Region = Africa

    b. Play around with various charts and variables and identify and evaluate 5+ additional association rules.
    c. Identify groups of similar countries, in 2018 [be sure to validate your groups using various charts].
    d. In the default configuration, follow the trajectories of Finland, Sweden, Iceland, Norway, and Denmark between 1900 and 2018. Do the countries appear to follow similar trajectories? Are there outliers or anomalous trajectories?
    e. Repeat step d) for Brazil, Paraguay, Uruguay, Venezuela, Colombia, Peru, and Ecuador.
    f. Based on your results in steps 4 and 5, would you expect the trajectory for Argentina to be more like those of the Nordic countries or those of the South American countries? Or perhaps neither? Is your answer the same over all time horizons?