Contents

List of Figures

List of Tables

1 Survey of Quantitative Methods

The bread and butter of quantitative consulting is the ability to apply quantitative methods to business problems in order to obtain actionable insight. Clearly, it is impossible (and perhaps inadvisable, in a more general sense) for any given individual to have expertise in every field of mathematics, statistics, and computer science.

We believe that the best consulting framework is reached when a small team of consultants possesses expertise in 2 or 3 areas, as well as a decent understanding of related disciplines, and a passing knowledge in a variety of other domains: this includes keeping up with trends, implementing knowledge redundancies on the team, being conversant in non-expertise areas, and knowing where to find detailed information (online, in books, or through external resources).

In this section, we present an introduction for 9 "domains" of quantitative analysis:

- survey sampling and data collection;
- data processing;
- data visualisation;
- statistical methods;
- queueing models;
- data science and machine learning;
- simulations;
- optimisation, and
- trend extraction and forecasting;

Strictly speaking, the domains are not free of overlaps. Large swaths of data science and time series analysis methods are quite simply statistical in nature, and it's not unusual to view optimisation methods and queueing models as sub-disciplines of operations research. Other topics could also have been included (such as Bayesian data analysis or signal processing, to name but two), and might find their way into a second edition of this book.

Our treatment of these topics, by design, is brief and incomplete. Each module is directed at students who have a background in other quantitative methods, but not necessarily in the topic under consideration. Our goal is to provide a quick "reference map" of the field, together with a general idea of its challenges and common traps, in order to highlight opportunities for application in a consulting context. These subsections are emphatically NOT meant as comprehensive surveys: they focus on the basics and talking points; perhaps more importantly, a copious number of references are also provided.

We will start by introducing a number of motivating problems, which, for the most part, we have encountered in our own practices. Some of these examples are reported on in more details in subsequent sections, accompanied with (partial) deliverables in the form of charts, case study write-ups, report extract, etc.).

As a final note, we would like to stress the following: it is **IMPERATIVE** that quantitative consultants remember that acceptable business solutions are not always optimal theoretical solutions. Rigour, while encouraged, often must take a backseat to applicability. This lesson can be difficult to accept, and has been the downfall of many a promising candidate.

1.6 Data Science and Machine Learning

From Data to Wisdom

Data is not information, information is not knowledge, knowledge is not understanding, understanding is not wisdom.

– attributed to Cliff Stoll, *Nothing to Hide: Privacy in the 21st Century*, 2006

In October 2012, the *Harvard Business Review* published an article calling data science the "sexiest job of the 21st century", and comparing data scientists with the ubiquitous Wall Street "quants" of the '80s and '90s: a data scientist is a "hybrid of data hacker, analyst, communicator, and trusted adviser" [1] – data science is hot and everybody wants in!

Would-be data scientists are usually introduced to the field via machine learning algorithms and applications. Much could be said about those; the material presented here is not meant to constitute a complete survey of the field (multiple references are provided for the interested reader).

In particular, we will take a brief look at:

- the **fundamentals** of data science;
- **association rules** mining;
- **supervised learning and classification**, with a focus on **decision trees**;
- **unsupervised learning and clustering**, with a focus on *k*–**means**;
- some of the common **issues and challenges** linked to the data science process;
- an introduction to **artificial** (or augmented) **intelligence and deep learning**, *via* simple neural networks;
- **text mining and natural language processing**, with a focus on **sentiment analysis**;
- the basic principles of **distributed computing** for Big Data analysis, and
- a few other topics.

1.6.1 Fundamentals

From Data to Wisdom

We learn from failure, not from success!

– Bram Stoker, *Dracula*

Hilary Mason, a noted data scientist, has described data science as

> the **working intersection** of statistics, engineering, computer science, domain expertise, and "hacking."

In her view (which we share), it involves two main thrusts: **analytics** (which is to say, counting things) and **inventing new techniques** to draw insights from data [2].

A more pragmatic approach is to define data science as "the collection of processes by which we extract useful and **actionable insights** from data" (paraphrased from Ted Kartler).

Types of Learning As humans, we learn (at all stages) by first taking a look around, and then by answering questions, testing hypotheses, creating concepts, making predictions, creating categories and classifying objects, and grouping objects. In a way, the idea behind data science is to try to teach our machines (and thus, ultimately, ourselves) to gleam 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?

The simplest data science method, thus, is to **explore the data** (or at a representative sample), providing a summary through basic statistics – mean, mode, histograms, etc.; making its multi-dimensional structure evident through data data visualisation; and looking for consistency, considering what is in there and what is missing.

In the data science context, we identify two substantially more sophisticated types of leaning: **supervised** and **unsupervised**. **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 (i.e. "students give an answer to each exam question based on what they learned from worked-out examples provided by the teacher/textbook"); each algorithm's performance is evaluated using **test data** for which the label is known but not used in the prediction (i.e. "the teacher provides the correct answers and marks the students' exam questions using the key".)

Unsupervised learning 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 (i.e. "the teacher is not involved in the discovery process"); the drawback is that accuracy **cannot be evaluated** with the same degree of satisfaction (i.e. "students might end up with different groupings").

In the same vein, data scientists also recognise **semi-supervised learning** when some data points have labels but most do not, which is often the case when acquiring data is costly ("the teacher provides 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") and **reinforcement learning**, where an agent attempts to collect as much (short-term) reward as possible while minimising (long-term) regret ("embarking on a Ph.D. with an advisor... with highs and lows and *perhaps* a solution at the end of the day?"). Some data science techniques fit into both camps; others can be either supervised or unsupervised, depending on how they are applied.

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. 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 (such as 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 *vs.* ...). The distinction is **crucial**. Take some time to make sure you master it.

Data Science Tasks As we have mentioned previously, quantitative methods are only really interesting to clients when they help them 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 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.

IMPORTANT NOTE: this is as good a time as any to remind you that not every situation calls for data science (or even analytics). The recent drive towards data science has a "snake oil" feel to it, in our opinion (and in the opinion of a number of qualified commentators [3]); vendors and pundits have conducted a near-perfect marketing campaign for so-called "Big Data" and Data Science in general, but ask 10 stakeholders what these concepts mean and you're likely to get 10 different answers.

As a result, clients will often expect you to solve their problem using the latest data science gizmos even if they have no bearing on the problem at hand. Understanding the **limitations** of the methods and focusing on the client's explicit and tacit **needs** might help you avoid that particular quagmire.

Common data science tasks (with representative questions) include [4]:

- **classification** and **class 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 establishes best clients?
- **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 this customer's segment?
- **link prediction** – J. and K. have 20 friends in common: perhaps they'd be great friends?

Figure 1: *Amanita muscaria* (or fly agaric), in the wild. Does it look dangerous to you?

A classic example is provided by the UCI Machine Learning Repository Mushroom Dataset [5]. Consider *Amanita muscaria*, a specimen of which is shown in Figure 1. 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**.

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, and that fly agaric is actually edible in general, or *vice-versa*.

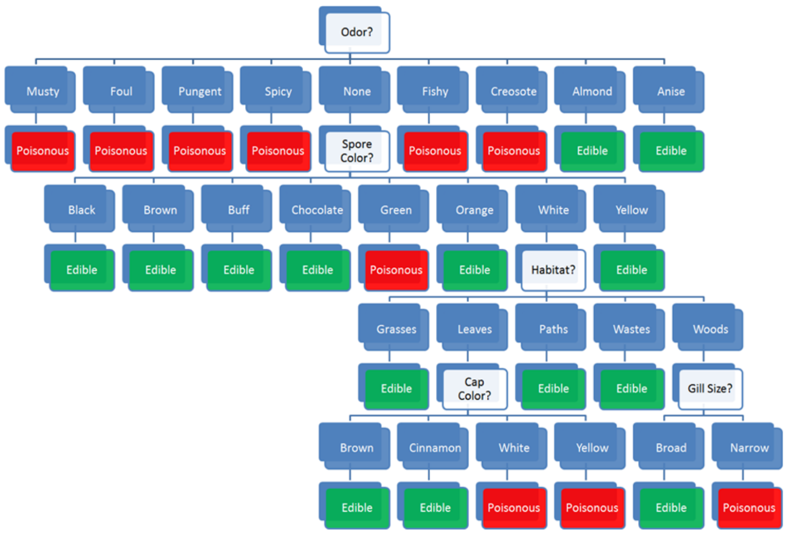
A predictive model, which would use features of a vast collection of mushroom species and specimens (including whether they were poisonous or edible) could help us some shed light on the matter: what do poisonous mushrooms have in common? what properties do edible mushrooms share? (note that this is not the same as understanding *why* a mushroom is poisonous or edible).

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

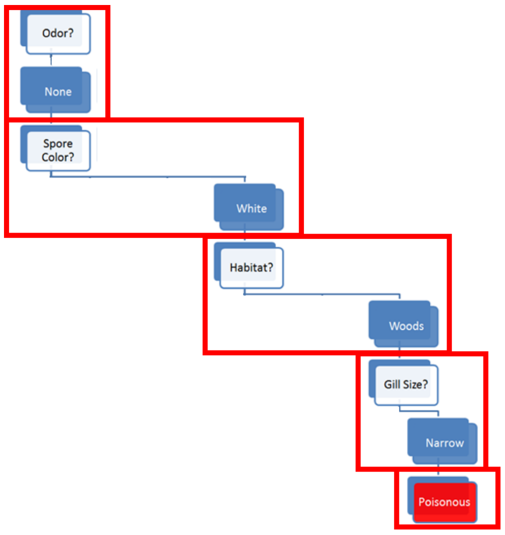
habitat: woods; **gill size**: narrow; **spores**: white; **odor**: none.

We do not know, *a priori*, whether it is **poisonous** or **edible**. Is the available information sufficient to answer the question? On its own, it is not (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 features in a vacuum). But we can use past data, with correct **edible** or **poisonous** labels, to build various supervised **classification** models to attempt to answer the question. Such a model, a **decision tree**, is shown in Figure 2a. The model prediction for *Amanita muscaria* follows the **decision path** shown in Figure 2b:

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 – that feature does not provide enough information to classify *Amanita muscaria* and we need to incorporate additional features into the decision path;
2. among mushrooms with no specific odor, some **spore colours** (black, brown, 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



(a) Decision tree



(b) Decision path

Figure 2: Decision tree for the mushroom classification problem, with decision path for *Amanita muscaria*.

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 forest mushrooms of no specific odor with white spores, a broad **gill size** is associated with edible mushrooms, whereas a narrow gill size is associated with poisonous mushrooms. As *Amanita muscaria* is a a narrow-gilled forest mushroom of no specific odor with white spores, the decision path predicts that it is **poisonous**.

The model **does not explain why** this particular combinations of features is associated with poisonous mushrooms – the decision path is not **causal**. Would you or a client have trusted an **edible** prediction? What's the true cost of making a classification mistake? Is the data on which the model is built representative? What do we need to know about the model in order to trust it?

1.6.2 Association Rules

Tufte's Rejoinder

Correlation isn't causation. But it's a big hint.

– E. Tufte

Association rules discovery is a type of unsupervised learning that finds **connections** among the attributes and levels (and combinations of attribute and levels) of a dataset's observations.

For instance, we might analyse a 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.

As Tufte reminds us on the previous page, correlation is not causation. Being a triathlete does not cause one to drive a Subaru, but a connection exists – enough so that in 2018, some Subaru dealerships in Québec were offering to pay the registration fee at an IRONMAN 70.3 competition with the purchase of a new vehicle!

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. Bread and milk might often be purchased together, for instance, but that is unlikely to be interesting given the frequency of market baskets containing milk or bread (in the mathematical sense of "or"). Wieners and mustard, on the other hand, might be purchased together more often than one would expect given how frequently both items are purchased; perhaps they are likely to be purchased as a pair, but not so likely to be purchased individually.

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 hot dogs while simultaneously (and quietly) raising the price of condiments 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 (banking on the fact that customers are unlikely to shop around to get the best deal on hot dogs AND condiments).

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. Rumour has it that the experiment was neither popular nor successful.

Other applications and uses exist, such as

- 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 (care must be taken to avoid a number of pitfalls);
- altering circumstances or environment to take advantage of these correlations (when a causal effect is suspected);
- using connections to modify the likelihood of certain outcomes;
- imputing missing data, and
- text autofill and autocorrect.

Other uses and examples can be found in [7–9].

Causation and Correlation Association rules can automate **hypothesis discovery**, but one must remain correlation-savvy (which is less prevalent among quantitative specialists than one might hope, in our experience). 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 other attributes $C_1, \ldots, C_n$ (which may not be available in the dataset) cause both A and B.

E. Siegel provides a number of examples of the confusion in [7]:

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

It can be tempting to try to **explain** these results (again from [7]): perhaps

- in preparation before an act of nature, 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 consumer 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 (?) it spread to them and so on.

These explanations *might* very well be the right ones, but there is nothing in the data to support them. Association rules discovery **finds** interesting rules, but it does not explain them.

IMPORTANT NOTE: you might not have much control over the matter as a consultant, but do whatever is in your power to avoid the accompanying headlines (or briefing to ministers) to be along the lines of

- "Pop-Tarts" help hurricane victims get 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, it is often the case that the "best" rules are those which are only accurate 10% of the time, as opposed to rules which are only accurate is only 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** (coverage) which measures the frequency at which a rule occurs in a dataset – low coverage values indicate rules that rarely occur;
- **confidence** (accuracy) 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 ... well, more interesting than rules with small absolute interest, and
- **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 was independent of the premises.

If N is the number of observations in the dataset and $\text{Freq}(A) \in [0, N]$ represents the count of the dataset's observations for which property A holds, then we can compute the metrics as follows:

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

The support is the proportion of instances where the premise and the conclusion occur together; the confidence is the proportion of instances where the conclusion occurs when the premise occurs, and so forth.

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

> "If an individual is born before 1986 (X), then they own a copy of Teresa Teng's *The Moon Represents My Heart*, in some format (Y)".

Let's assume further that

- $\text{Freq}(X) = 3888$ individuals were born before 1986;

- $\text{Freq}(Y) = 9092$ individuals own a copy of *The Moon Represents My Heart*, and
- $\text{Freq}(X \cap Y) = 2720$ individuals were born before 1986 and own a copy of *The Moon Represents My Heart*.

We can easily compute the 4 metrics for RM:

$$\begin{aligned}
\text{Support(RM)} &= \frac{2720}{15,536} \approx 18\% \\
\text{Confidence(RM)} &= \frac{2720}{3888} \approx 70\% \\
\text{Interest(RM)} &= \frac{2720}{3888} - \frac{9092}{15,356} \approx 0.11 \\
\text{Lift(RM)} &= \frac{15,356^2 \cdot 0.18}{3888 \cdot 9092} \approx 1.2.
\end{aligned}$$

These values are easy to interpret: RM occurs in 18% of the instances and it is true in 70% of the instances where the individual was born prior to 1986. This would seem to make RM a meaningful rule about the dataset – being older and owning that song are linked properties! But if being younger and not owning that song are not also linked, the statement is weaker than it appears at a first glance. As it happens, RM's lift is 1.2, which can be rewritten as $1.2 \approx \frac{0.70}{0.56}$, which means that 56% of younger individuals also own the song. The ownership rates between 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 low interest (0.11).

IMPORTANT NOTE: There will be times when an interest of 0.11 in a rule will be considered a smashing success, when a lift of 15 will not be considered that significant, and so forth. It's difficult to give hard and fast rules about absolute thresholds: it always depends on the context, and on comparing metric values with the dataset's other rules. In general, there is value in conducting preliminary exploration of the space of association rules (using domain expertise when appropriate) to determine reasonable threshold ranges for the specific situation; candidate rules are discarded or retained depending on these metric thresholds.

Rule Generation When we are given a rule (or rules), it is straightforward to evaluate it using various metrics. The real challenge of association rules discovery is in **generating** a set of candidate rules which are likely to be retained, without wasting time generating rules which are likely to be discarded. That is easier said than done.

An **item set** (or instance set) is a list of attributes and values. A set of rules can be created by adding 'IF ... THEN' blocks to the instances. From the instance set

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

we can create the rules

- IF (membership = True AND age = Youth) THEN purchasing = Typical;
- IF membership = True THEN (age = Youth AND purchasing = Typical);
- IF $\varnothing$ THEN (membership = True AND age = Youth AND purchasing = Typical); etc.

Now, consider an item set $\mathscr{C}$ with n members (that is to say, n attribute/level pairs). In a rule derived from $\mathscr{C}$, each of the n members shows up either in the premise or in the conclusion; there are thus 2^n such rules. The rule where each member is part of the premise (that is, the rule without a conclusion) is nonsensical and is not allowed, and so $2^n - 1$ rules can be derived from $\mathscr{C}$. 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 item sets in the data and generating all rules from those item sets) for any dataset with a realistic number of attributes. How do we generate **promising** candidate rules, in general?

The **apriori** algorithm is an early attempt to bypass that difficulty. It was developed initially for transaction data (but every reasonable dataset can be transformed into a transaction dataset using dummy variables). This algorithm attempts to find **frequent item sets** from which to build candidate rules, instead of building rules from *all* possible item sets.

It starts by identifying frequent **individual items** in the database and extends those that are retained into larger and larger item sets, assuming that these new sets are still found **frequently enough** in the data. In the technical jargon, we say that *apriori* uses a **bottom-up approach** and the **downward closure property of support**.

The savings come from the fact that the algorithm prunes candidates with **infrequent subpatterns** and removes them from consideration for any future item set: if a 1-item set is not considered to be frequent enough, any 2-item set containing it is also infrequent (see Table 1 for an illustration).

Of course, this requires a support threshold **input** (and there is no clear way to select that threshold, although it has to be set sufficiently high to minimise the number of frequent item sets that are being considered). The algorithm terminates when no further successful extensions are found.

- **Strengths:** easy to implement, easily parallelized [16].
- **Limitations:** slow, requires frequent data set scans, not ideal for finding rules for infrequent and rare item sets.

Other, more efficient, algorithms have since displaced it (although it does have historical value):

- **Max-Miner** tries to identify frequent item sets without enumerating them – it performs jumps in space instead of using bottom-up approach;
- **Eclat** is faster and uses depth-first search, but requires extensive memory storage (apriori and eclat are both implemented in the `R` package `arules`).

Notes 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 solve for association rules discovery, but **statistically sound association discovery** can help reduce the risk of finding spurious associations to a user-specified significance level [14, 15].

Since frequent rules correspond to instances that occur repeatedly in the dataset, algorithms that generate item sets often try to **maximise coverage**. When **rare events** are more meaningful (such

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

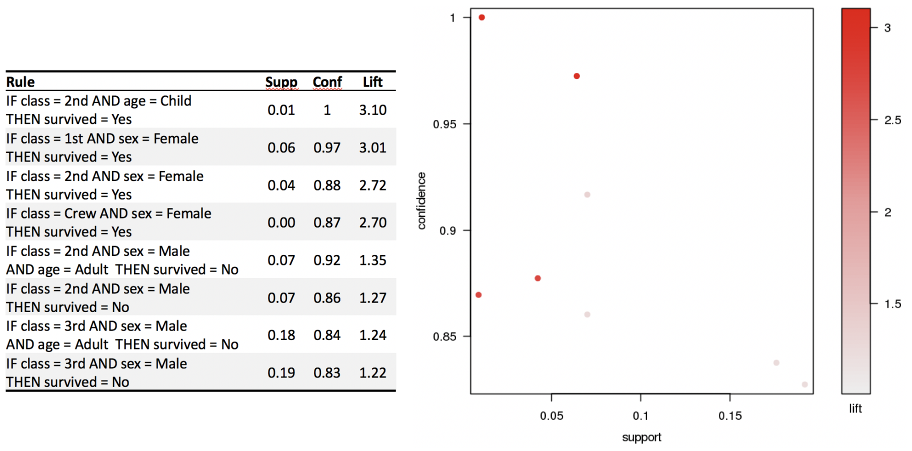
Table 1: Association rules for NHL playoff teams (1942-1967). A list of the 4 teams making the playoffs each year is shown on the left ($N = 20$). Frequent item sets are generated using the *apriori* algorithms, with a support threshold of 10. We see that there are 5 frequent 1-item sets, top row, in yellow (New York made the playoffs $6 < 10$ times – no larger frequent item set can contain New York). 6 frequent 2-item sets are found in the subsequent list of ten 2-item sets, top row, in green (note the absence of New York). Only 2 frequent 3-item sets are found, top row, in orange. Candidate rules are generated from the shaded item sets; the rules retained by the thresholds Support ≥ 0.5, Confidence ≥ 0.7, and Lift > 1 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 also Detroit make the playoffs (the presence or absence of Montreal in a rule is a red herring, as *les Habitants* made the playoffs every year in the data.

as detection of a rare disease or a threat), we need algorithms that can generate rare item sets. **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 the client.

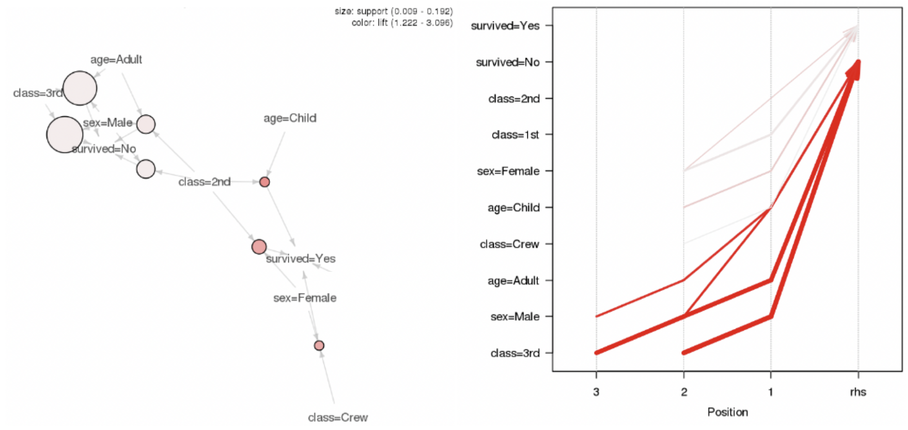
Other popular algorithms include: AIS, SETM, aprioriTid, aprioriHybrid, PCY, Multistage, Multihash, etc. Other metrics can be found in the `arules` documentation [11].

As a more realistic example, consider the *Titanic* dataset. Compiled by Robert Dawson in 1995, it 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).



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

(a) Scatterplot



(b) Network and Parallel Coordinates

Figure 3: Visualisations of the 8 *Titanic* association rules; scatterplot (top row), network diagram (bottom row, on the left), parallel coordinates (bottom row, on the right).

The attributes/levels are:

- **class** (first class, second class, third 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?

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); neither will we treat it as a predictive task (since the situation on the Titanic has little bearing on survival for new data). We will use fixed-structure association rules to **describe** explore the survival on the *Titanic* (compare with [17]). We use the `arules` implementation of the *apriori* algorithm in R to generate and prune candidate rules, eventually leading to **8 rules** (the results can be visualised in Figure 3). Who survived? (As a final reminder, **correlation is still not causation.**)

1.6.3 Supervised Learning and Classification

An Embarrassment of Riches

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.

– C.C. Aggarwal, *Data Classification: Algorithms and Applications*

Classification is a supervised learning endeavour in which a sample set of data (the **training** set) is used to determine rules and patterns that divide the data into predetermined groups, or classes. The training data usually consists of a **randomly** selected subset of the **labeled** data. **Value estimation** (regression) is similar to classification, except that the target variable is numerical. In the **testing phase**, the model is used to assign a class to observations for which the label is hidden, but ultimately known (the **testing** set).

The performance of the predictive model is then evaluated by comparing the predicted and the values for the testing set observations (but **never** using the training set observations). A number of technical issues need to be addressed: which features to select for inclusion in the model and, perhaps more importantly, which algorithm to choose, for example. The mushroom (classification) model from earlier in this section is a good example of a classification model, although no detail were provided using the training data and choice of algorithm.

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 assisting housing in old age based on demographic information/survey answers;
- **marketing and 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 organising and grouping instances into categories, or
- enhancing the detection of relevant objects:
  - **avoidance** – "this object is an incoming vehicle";
  - **pursuit** – "this borrower is unlikely to default on her mortgage";
  - **degree** – "this dog is 90% likely to live until it's 7 years old".

Other examples may be found in [18–21].

Some concrete examples may provide a clearer picture of the types of supervised learning problems that quantitative consultants may be called upon to solve.

- 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?
  - 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?
  - the amount by which a claim will be reduced if it is fraudulent?

  What kind and how much data do you think is required? What would constitute a predictive success? How would you help the client understand these risks associated with a predictive modeling approach?
- 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?

  How would you use standard statistical techniques to answer these questions?

In the absence of testing data, classification models cannot be used for predictive tasks, but may still be useful for **descriptive** tasks. When testing data exists, the process is often quite similar, independently of the choice of the algorithm (see the classification pipeline shown in Figure 4).

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 [27]. Here are common algorithms that data scientists should have at their command (full descriptions available in [4, 6, 38]):

- **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 multicolinearity, and the stepwise variable selection process that is typically used is problematic – regularisation methods would be better suited [40] (see Figure 6, top left);
- **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 section;
- **decision trees** are perhaps the most commons 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) – we will provide a bit more details shortly (see Figure 7);

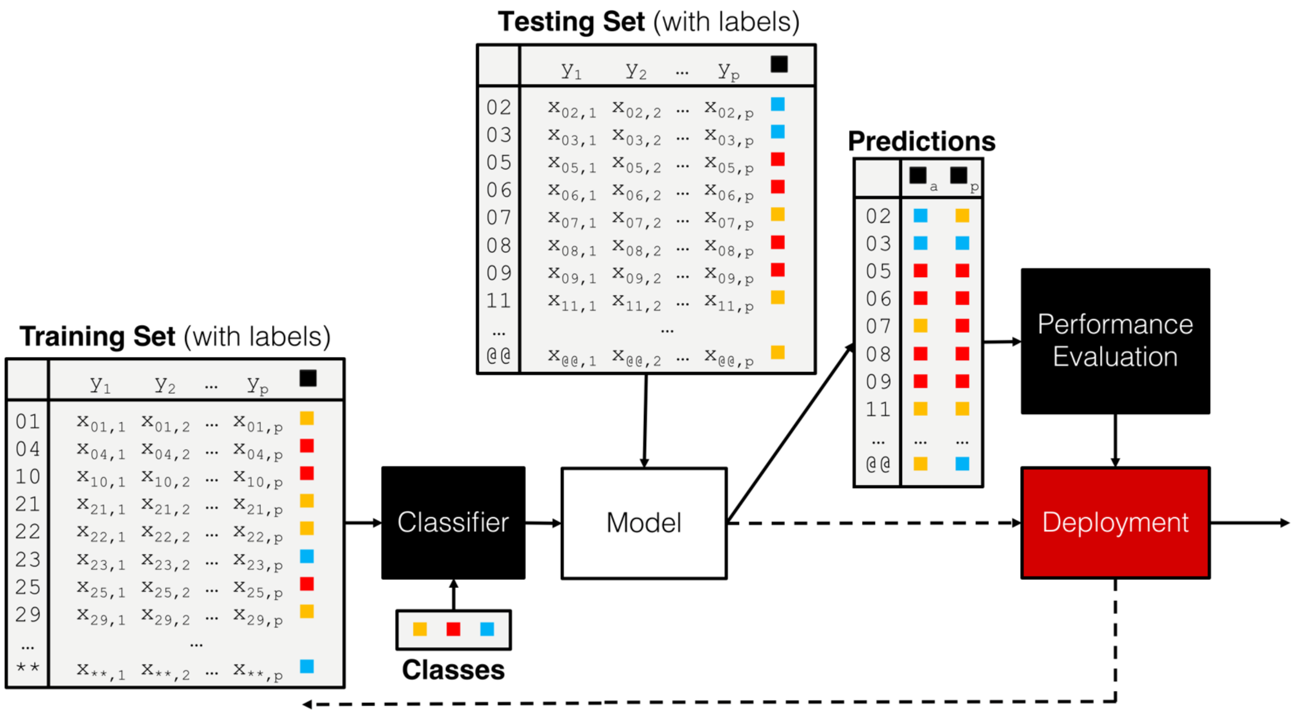
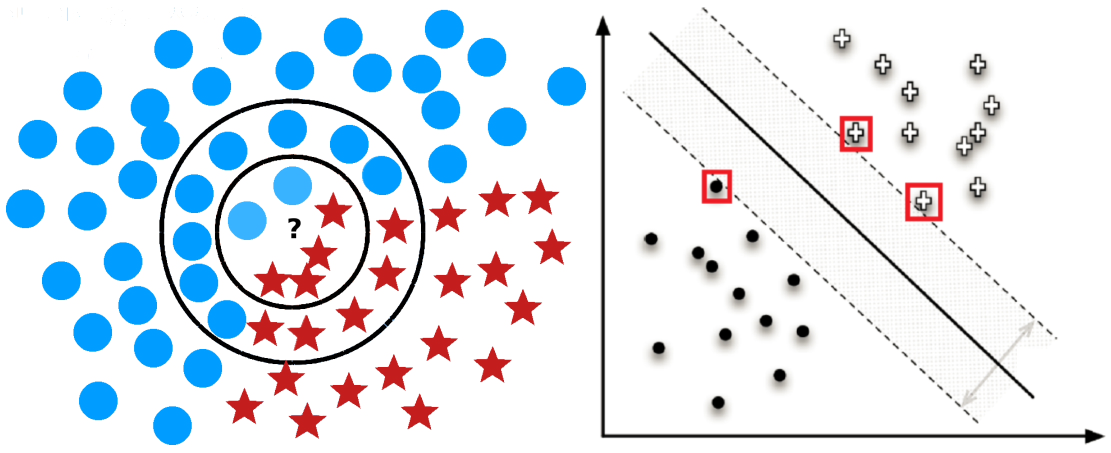


Figure 4: A classification pipeline, including training set, testing set, performance evaluation, and (eventual) deployment.



(a) *k*NN **(b)** SVM

Figure 5: Illustration of a k nearest neighbour (left) and a support vector machines classifier (right, based on [4]). What is the 6NN prediction for the location marked by a question mark? What about the 19NN prediction?

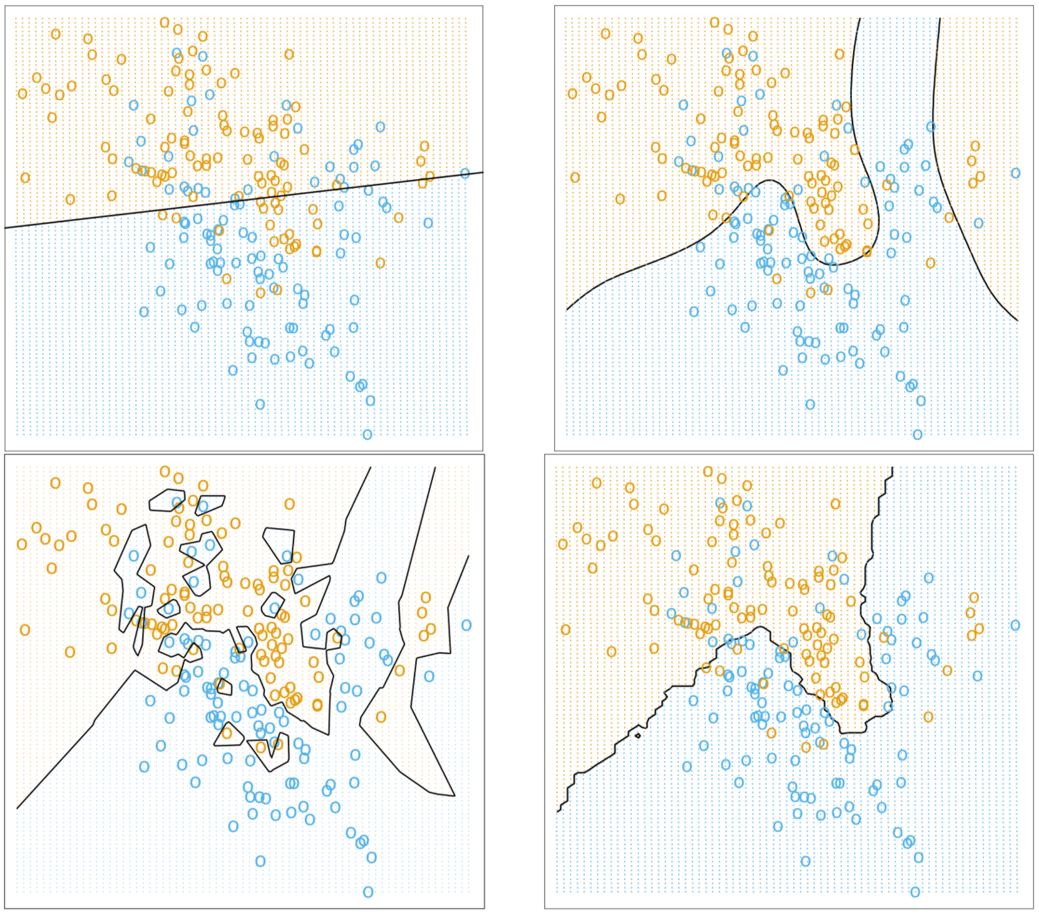


Figure 6: Illustrations of various classifiers (linear regression, top left; optimal Bayes (top right); 1NN and 15NN (bottom left and right, respectively) on an artificial dataset (from [6]). Note that linear regression is more stable, simpler to describe, and less accurate than *k*NN and optimal Bayes.

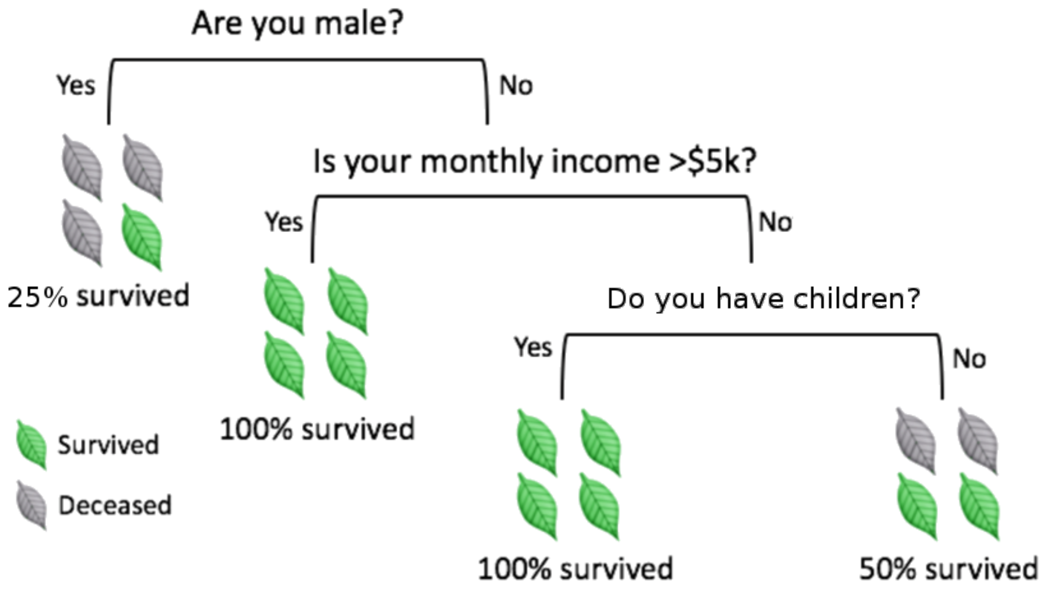


Figure 7: Illustration of a decision tree depicting the chances of survival for various disasters (fictional, based on [39]).

- **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 6, top right);
- **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 5a);
- **nearest neighbours classifiers** 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 5a and 6, bottom row)).

Decision Trees Decision trees are perhaps the most **intuitive** of these methods: classification is achieved by following a path up the tree, from its **root**, through its **branches**, and ending at its **leaves** (alghough 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 (that is to say, all instances in a leaf belong to the same leaf) 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 his widely-used C4.5) follows a simple path:

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.

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 $i = 1, \ldots, n$. The entropy of S is given by

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

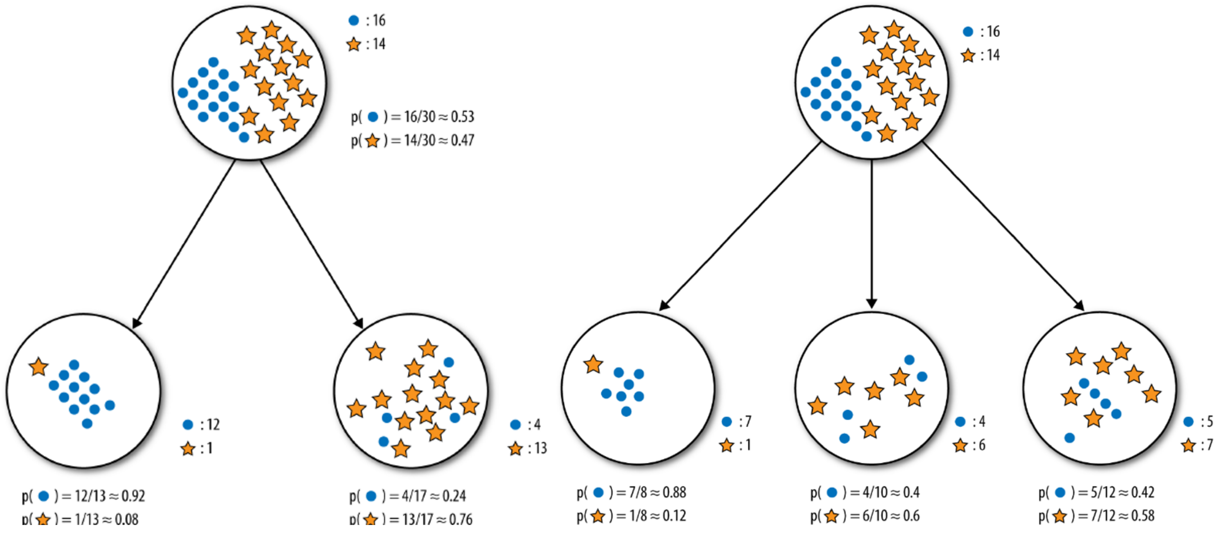


Figure 8: Picking the optimal information gain split (from [4]).

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

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

The second term on the right-hand side of 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 $\text{IG}(S;C_1,\ldots,C_k)$ is high; if the split leads to similar disorder in both children and parent, then $\text{IG}(S;C_1,\ldots,C_k)$ is low.

Consider, for instance, two splits shown for a parent set with 30 observations separated into 2 classes: $\circ$ and $\star$ (as in Figure 8). 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

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

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 when compared with the information gained from other splits that it becomes useful. A similar computation for the the ternary 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 – given those two choices, ID3 would select the first split 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 (being some of the algorithms with that property, although there is still some value in imputing missing observations);
- allow for **built-in feature selection** in that less relevant features do not typically tend to be used as splitting features;
- make **no assumption** about independence of observations, underlying distributions, multicolinearity, 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

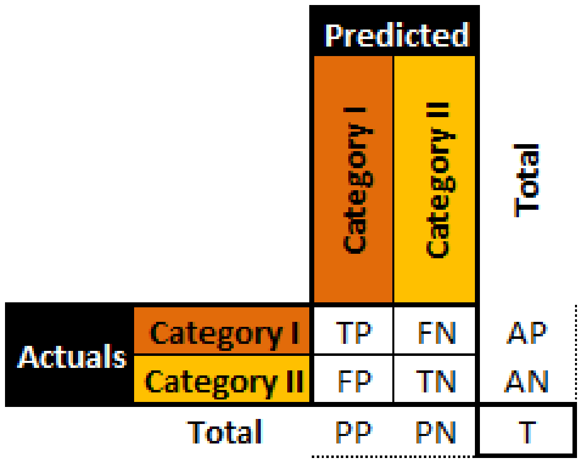
- **not usually as accurate** as other more complex algorithms, nor **as robust**, as small changes in the training data can lead to a completely different tree, with a completely different set of predictions (this can be problematic when presenting the results to a client whose understanding of these matters is slight);
- particularly **vulnerable to overfitting** in the absence of pruning – and pruning procedures are typically fairly convoluted (some models, such as conditional inference trees, 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 puruit of pure leaves, but is not the only splitting metric in use – common alternatives include **Gini impurity** and **variance reduction**. As mentioned previously, 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.

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 them are likely to make good predictions, on average.

There are still a few other points to ponder on the topic of decision trees;

- 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 (classical



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

Table 2: A general binary classifier.

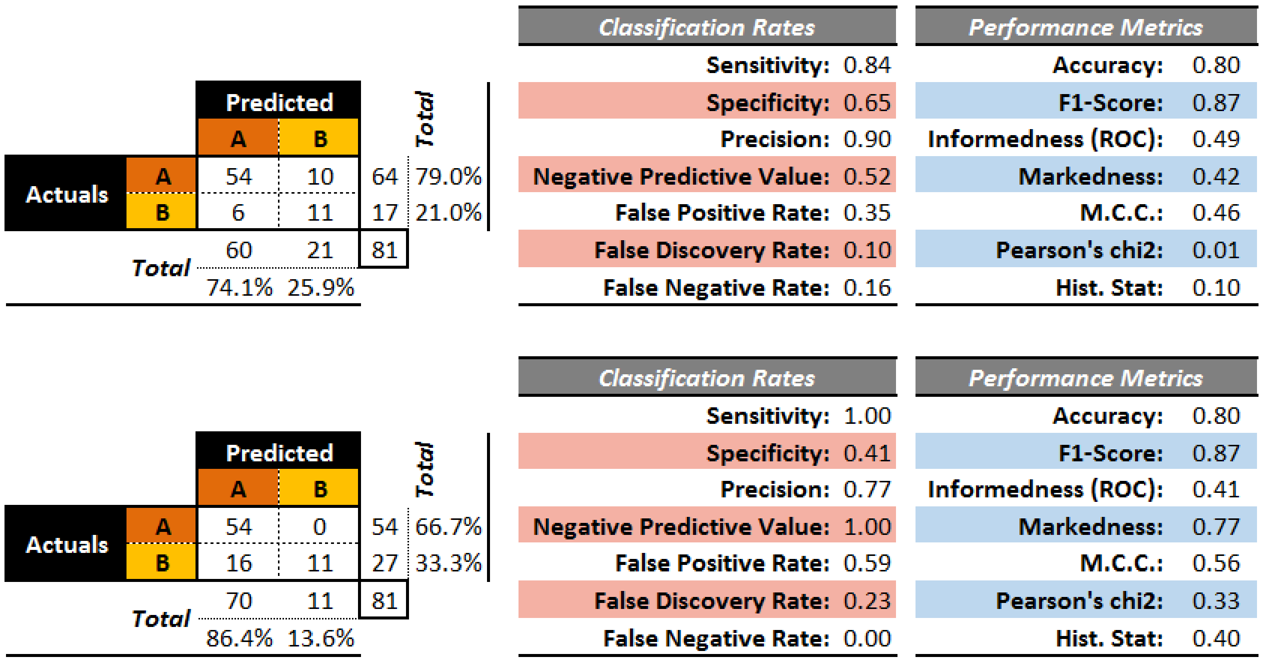
performance evaluation metrics can easily be fooled; if out of two classes one of the instances is only represented in 0.01% of the instances, predicting the non-rare class will yield correct predictions roughly 99.99% of the time, missing the point of the exercise altogether;

- with big datasets, algorithms must also consider efficiency – thankfully, decision trees are easily parallelisable.

IMPORTANT NOTE: as a consequence of the (so-called) **No-Free-Lunch Theorem**, no single classifier can be the best performer for every problem. The model selection process must take into consideration the nature of the available data, the relative frequencies of the classification sub-groups and the stated goals of the classification. Other considerations may include how easily the model lends itself to interpretation and statistical analysis; how much data preparation it requires; 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. The client might not be aware that past success is not a guarantee of future success – it is your responsibility to ensure that you try out various models.

Performance Evaluation 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 somewhat more unforgivable. Several metrics can be used to assess a classifier's performance, depending on the context.

Binary classifiers (such as the abstract example shown in Table 2) 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, TP, TN, FP and FN stand for *True Positives*, *True Negatives*, *False Positives*, and *False Negatives*, respectively. A perfect classifier would be one for which both $\text{FP}, \text{FN} = 0$. In practice, that rarely ever hAPpens (if at all).



		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		Performance Metrics	
Sensitivity:	0.84	Accuracy:	0.80
Specificity:	0.65	F1-Score:	0.87
Precision:	0.90	Informedness (ROC):	0.49
Negative Predictive Value:	0.52	Markedness:	0.42
False Positive Rate:	0.35	M.C.C.:	0.46
False Discovery Rate:	0.10	Pearson's chi2:	0.01
False Negative Rate:	0.16	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		Performance Metrics	
Sensitivity:	1.00	Accuracy:	0.80
Specificity:	0.41	F1-Score:	0.87
Precision:	0.77	Informedness (ROC):	0.41
Negative Predictive Value:	1.00	Markedness:	0.77
False Positive Rate:	0.59	M.C.C.:	0.56
False Discovery Rate:	0.23	Pearson's chi2:	0.33
False Negative Rate:	0.00	Hist. Stat:	0.40

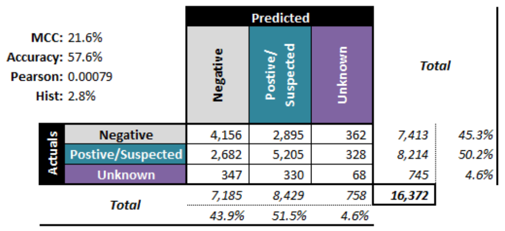
Table 3: Performance metrics for two (artificial) binary classifiers.

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: $(\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN})(\text{AP} \cdot \text{AN} \cdot \text{PP} \cdot \text{PN})^{-1/2}$
- 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 are shown in Table 3. Both classifiers have an accuracy of 80%, but the classifiers are not of the same quality: while the second classifier sometimes makes a wrong prediction for A, it never makes one for B, whereas the first classifier makes wrong 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.

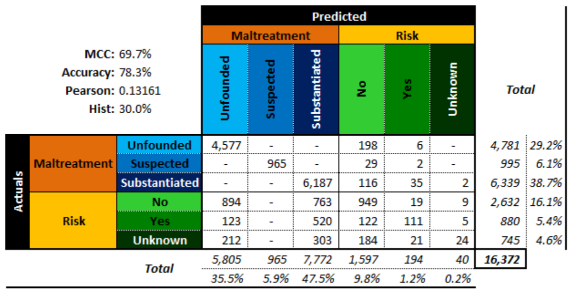
So which one is best? The performance metrics alone do not suffice to answer the question: a lot depends on the price associated with making a mistake. There's a good reason to use the Mathews Correlation Coefficient: of all the performance metrics, only it and the accuracy easily generalize to multi-level classifiers.



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

		Predicted				
		Negative	Postive/ Suspected	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%		

(a) Ternary classifier.



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%		

(b) Senary classifier.

Table 4: Performance metrics for (artificial) multi-level classifiers.

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* (this is especially problematic when one of the categories contains substantially fewer members than the others). 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 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: $\frac{1}{T}(|\text{PP}-\text{AP}|+|\text{PN}-\text{AN}|)$

For a given number of levels, the smaller these quantities, the more similar the actual and predicted distributions (see Tables 4 for examples of confusion matrices for multi-level classifiers).

For numerical targets y with predictions $\hat{y}$, the confusion matrix is not applicable, but a number of classical performance evaluation metrics are used: the

- mean squared and mean absolute errors
$$\text{MSE} = \text{mean}\left\{(y_i-\hat{y}_i)^2\right\}, \quad \text{MAE} = \text{mean}\{|y_i-\hat{y}_i|\};$$
- normalised mean squared and mean absolute errors
$$\text{NMSE} = \frac{\text{mean}\left\{(y_i-\hat{y}_i)^2\right\}}{\text{mean}\left\{(y_i-\overline{y})^2\right\}}, \quad \text{NMAE} = \frac{\text{mean}\left\{|y_i-\hat{y}_i|\right\}}{\text{mean}\left\{|y_i-\overline{y}|\right\}};$$
- mean average percentage error
$$\text{MAPE} = \text{mean}\left\{\frac{|y_i-\hat{y}_i|}{y_i}\right\};$$
- correlation $\rho_{y,\hat{y}}$.

An isolated performance metric does not provide enough of a rationale for model validation, unless it has first been normalised. There is much more to be said on the topic of model selection, but that is outside the scope of this document.

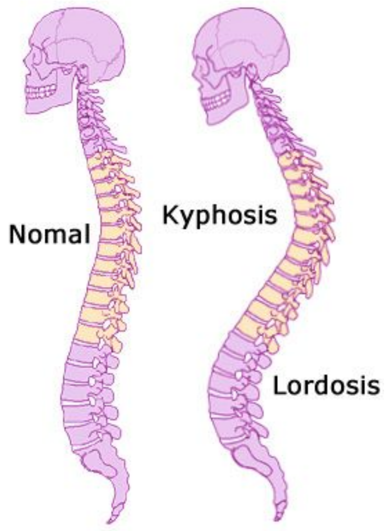


Figure 9: Kyphosis (medical condition). Surgery may cure the condition in children, but the procedure is not without risk. Are there factors that can determine if it is likely to be succesful prior to conducting it?

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 [41]). 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 question of interest for this natural dataset is

how do the three explanatory attributes impact the operation's success?

We use the `rpart` implementation of CART in `R` to generate some decision trees (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 10. Interestingly, it would appear that the `number` variable does not play a role in determining the success of the operation (for the observations in the dataset).

The visualisation of the decision tree certainly indicates that its leaves are not pure; more details can be found in a slightly different visualisation (see Figure 11a). A bit more (off-page) work suggests that the tree is somewhat overgrown and that it could benefit from being pruned after the first branching point (see Figure 11b). At any rate, it is mostly meaningless to discuss the performance of the tree if we are not using a holdout testing sample (not to say anything about the hope of generalising 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 5. Is this a good model?

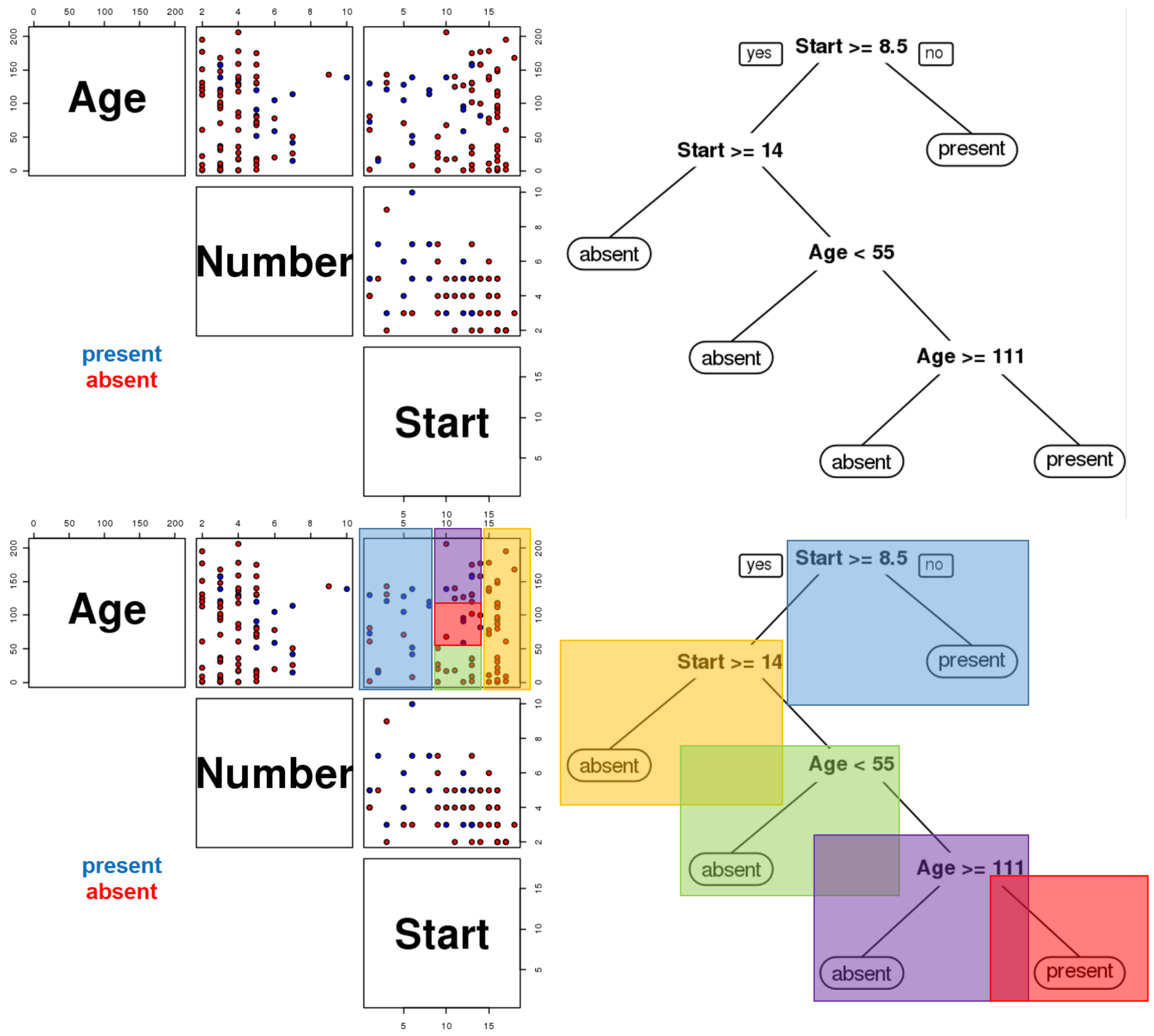
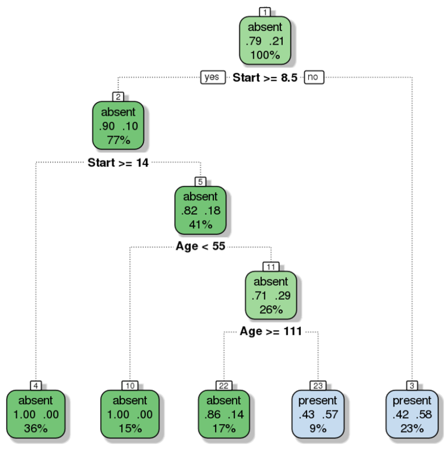
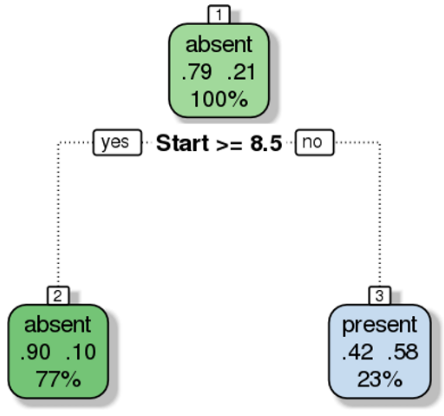


Figure 10: Kyphosis decision tree – visualisation. Only two features are used to construct the tree. We also note that the leaves are not pure – there are blue and red instances in 3 of the 5 classification regions.



(a) Decision tree



(b) Pruned decision tree

Figure 11: Pruning a decision tree – the original tree is more accurate, but substantially more complex than the pruned tree.

		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 5: Kyphosis decision tree – performance evaluation. The accuracy and $F1$ score are good, but the false discovery rate and false negative rate are not so great. This tree is good at predicting successful surgeries, but not fantatstic at predicting failed surgeries. Is it useful?

1.6.4 Unsupervised Learning and Clustering

A Never-Ending Story

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.

– V. Estivill-Castro, *Why So Many Clustering Algorithms?*

Quantitatively speaking, many statements can be made of a dataset: at the univariate level, we can compute frequency counts for the variables, identify measures of centrality (mean, mode, median) and dispersal (range, standard deviation), among others. At the multivariate level, various options are available, including n—way tabulations, correlation analysis, and data visualisation, for instance.

While these might provide insights on simple data, sets for which there is a large number of variables and mixed types (categorical and numerical) might not yield to such an approach. Instead, insight might come in the form of **aggregation** or **clustering** of similar observations. A successful clustering scheme tightly joins together any number of similarity profiles (tight in this context refers to small variability within the cluster). A typical application is one found in search engines, who attempt to group similar objects in one cluster and dissimilar objects far from each other – the listed results for the search are the nearest similar objects clustered around the search item. Implicit in this example (and in all applications) is the crucial factor of **closeness**: what does it mean for one observation to be near another one? Various **metrics** can be used, and not all of them lead to the same results.

Clustering is a form of unsupervised learning since the cluster labels are not determined ahead of the analysis. Among its advantages is the fact that clusters can be determined even when there are no natural ways to break down a dataset into constituent parts. One the one hand, if there is no natural way to break up the data into clusters, the results may be entirely arbitrary and fail to represent any underlying reality of the dataset. On the other hand, it could turn out that there were no recognized ways to naturally break up the data into clusters, and the clustering algorithm may discover such a grouping – clustering is sometimes called **automated classification** as a result. Figure 12 illustrates the main notion of clustering over a an artificial dataset.

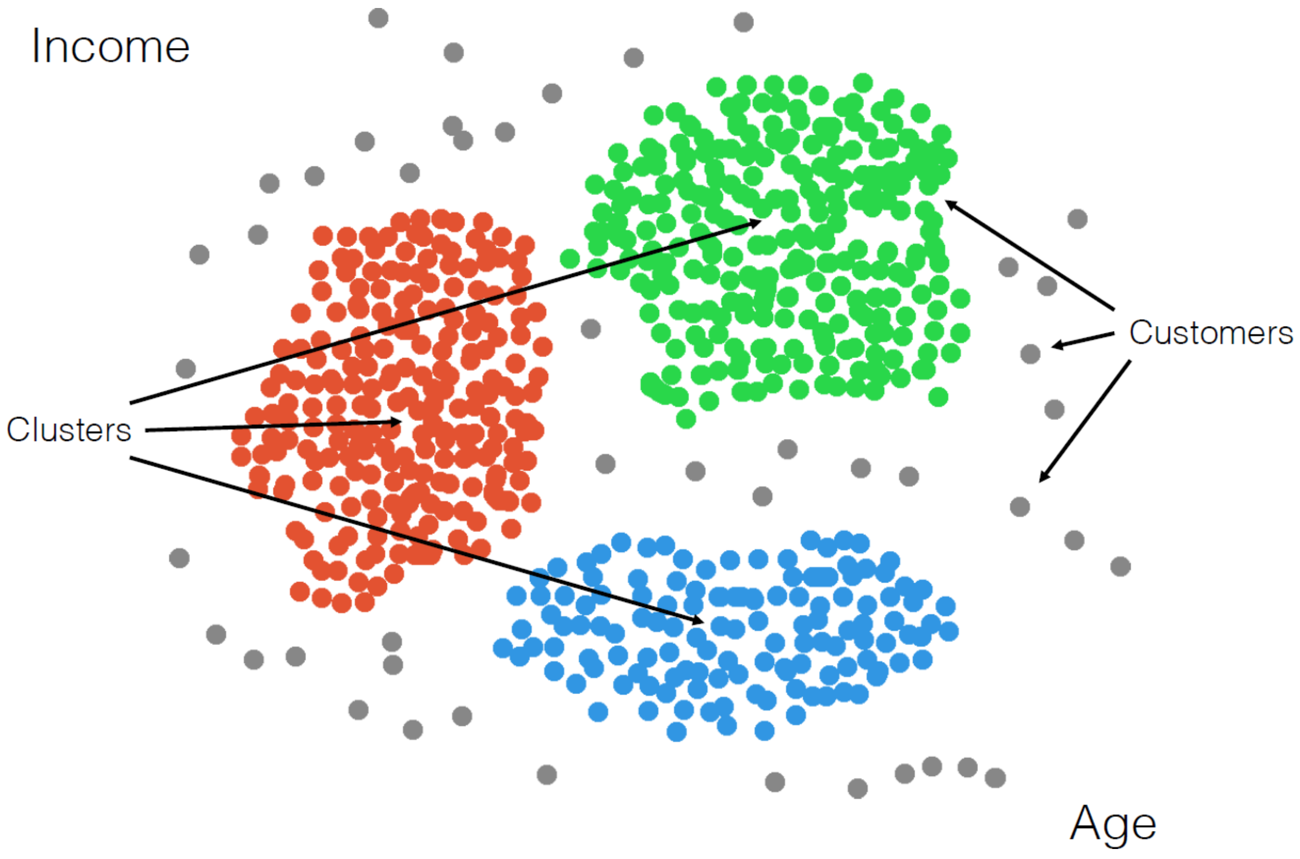


Figure 12: Clusters and outliers in an artificial dataset.

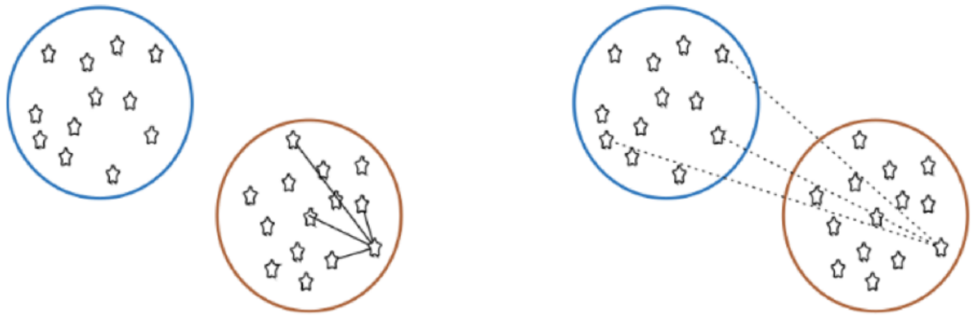


Figure 13: Average distance to points in own clusters (left, smaller is better) and to points in other clusters (right, larger is better).

In **clustering**, the data is divided into **naturally occurring groups**. Within each group, the data points are similar; from group to group, they are dissimilar (see Figure 13). Clustering as a learning mechanism is relatively **intuitive** for human beings as our brains unconsciously search for patterns. In general, people can handle **messy** data with the same relative ease as they handle clean data, but computers have a harder time doing so – part of the difficulty is that 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.

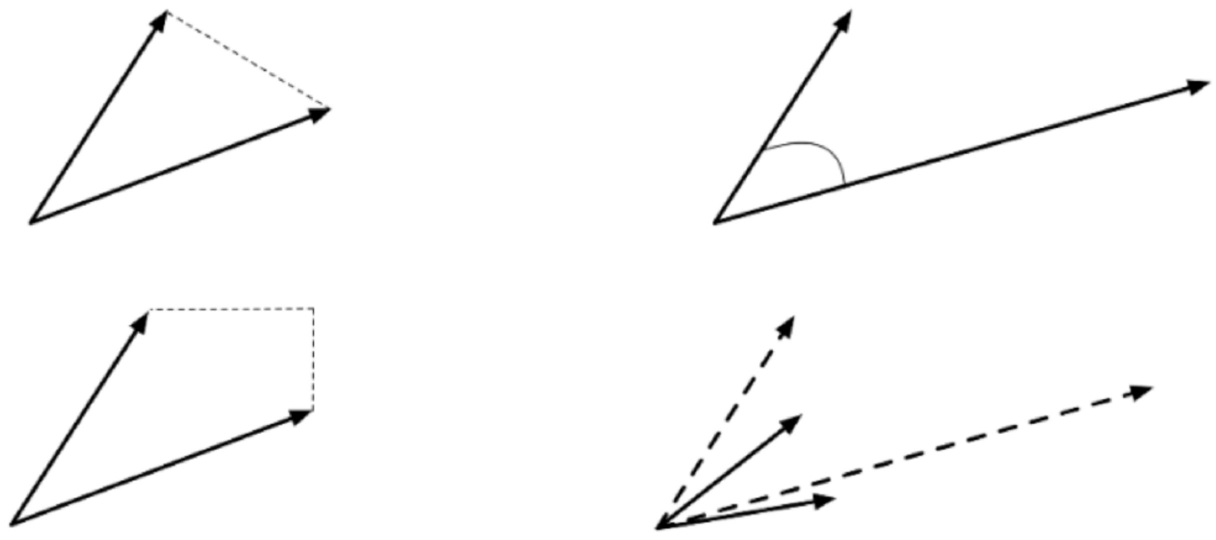


Figure 14: Distance metrics between observations: Euclidean (as the crow flies, top left); cosine (direction from a vantage point, top right); Manhattan (taxi-cab, bottom left). Observations should be transformed (scaled, translated) before distance computations (bottom right).)

Clustering 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 clustering results remains **strong** – we really, really want to reason with the data. Were you able to look at Figure 12 without assigning labels or trying to understand what type of customers were likely to be young and have medium incume? older and wealthier? Clustering algorithms are also (typically) **non-deterministic** – the same routine, applied twice (or more) to the same dataset, can discover completely different clusters (the order in which the data is presented can play a role, so can starting configurations)!

IMPORTANT NOTE: this (potential) non-replicability is not just problematic for validation, it can also leads to client dissatisfaction. If the consultant is tasked with finding customer clusters for marketing purposes and the clusters change every time the client asks for a report, there will be a very confused client unless you have explained the stochastic nature of the clusters before hand.

All clustering algorithms rely on the notion of **similarity** w between observations; in many instances, similarity is obtained *via* a **distance** (or metric) d. Typically, $w \to 1$ as $d \to 0$ and $w \to 0$ as $d \to \infty$, but there are similarity measures which are not derived from a distance metric. One additional source of clustering challenge is that there is no such thing as **the** distance or **the** similarity measure between observations, and that observations which are similar using a specific measure may not be similar at all using another. Commonly-used metrics include: Euclidean, Manhattan, Cosine, Canberra, Haming, Jaccard, Pearson, and so forth (see Figure 14). In general, however, no matter which similarity measure is selected, the data must first be **transformed** (scaled, centered, etc.), which introduces another layer of difficulty as there is not a canonical way to do this.

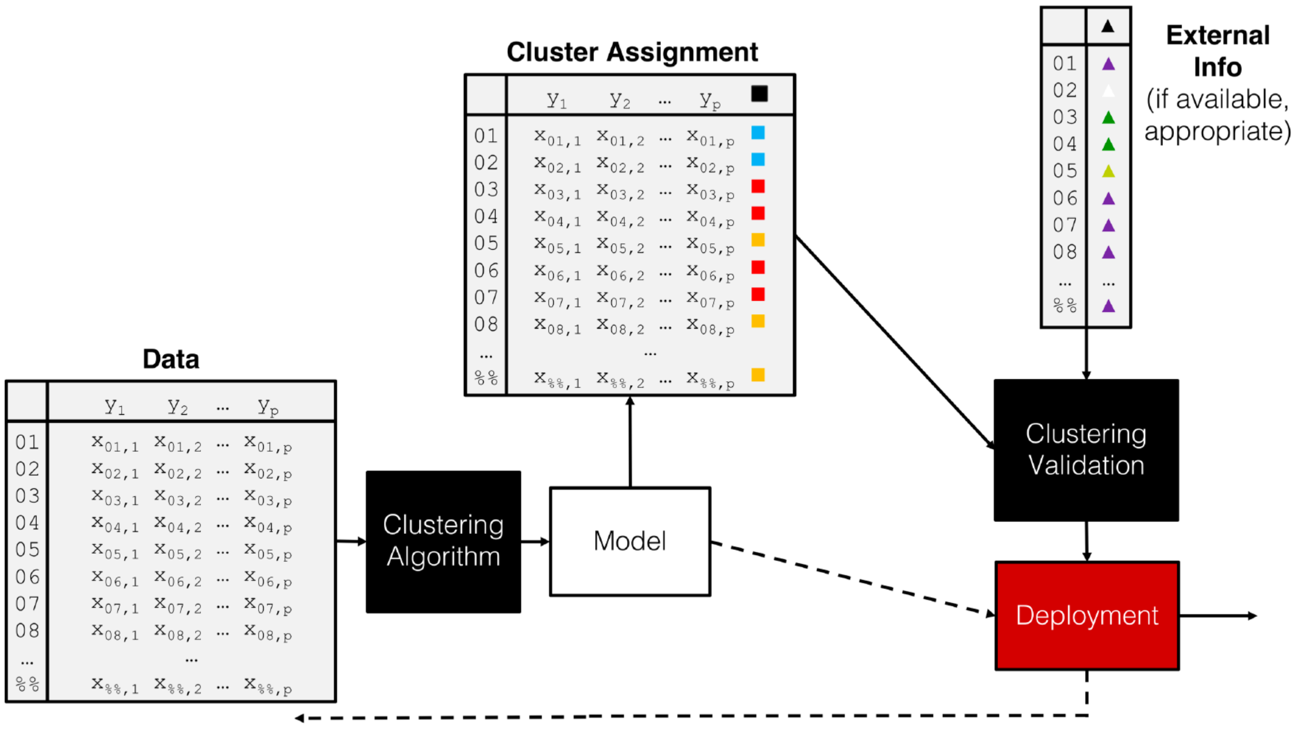


Figure 15: A clustering pipeline, including validation and (eventual) deployment

Clustering and other unsupervised learning tasks are frequently used, but often as preliminary steps in supervised learning problems – there are 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** – recognising 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.

Other examples may be found in [42, 46–57].

The clustering process is often the same, notwithstanding the choice of scaling strategy, similarity measure, and algorithm and parameters (see the pipeline shown in Figure 15).

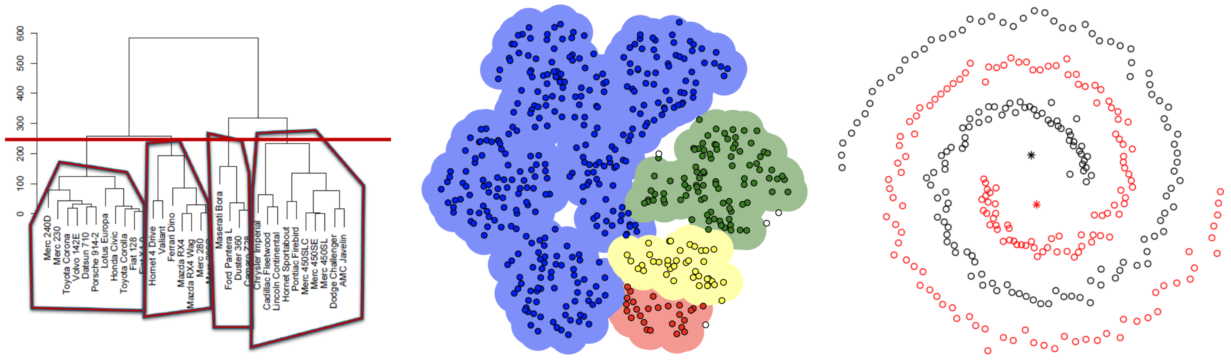


Figure 16: Illustration of hierarchical clustering (left), DBSCAN (middle, based on [62]), and spectral clustering (right).

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 [58]. The choice of algorithms (and associated parameters) is as much an art as it is a science, although domain expertise can come in handy [42]. Here are common algorithms that data scientists should have at their command (full descriptions available in [4, 38, 42]):

- 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 typically envision k—means with the Euclidean metric; variants exists such as k—mode (for categorical data), k—medians (for data with outliers), 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; there are divisive (DIANA) and agglomerative (AGNES) versions; there are no parameters to input, 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 16);
- **density-based spatial clustering** which is typically graph-based and attempts to identify closely-packed regions of the dataset; obvious advantages of DBSCAN (and variants OPTICS and DENCLUE) is robustness to outliers and not needing to be told how many clusters to search in the data; the main disadvantage is that the input parameters (neighbourhood radius and minimum number of points to be considered dense) are not easy to derive;
- **affinity propagation** is another clustering algorithm which selectes the optimal number of clusters directly from the data, but it does so by trying out and evaluating various scenarios and evaluating which may end up being time-consuming;
- **spectral clustering** can be used to recognise non-globular clusters; these are found by computing eigenvalues of an associated laplacian matrix – consequently, spectral clustering is fast.

Other methods include **latent Dirichlet allocation** (which is 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).

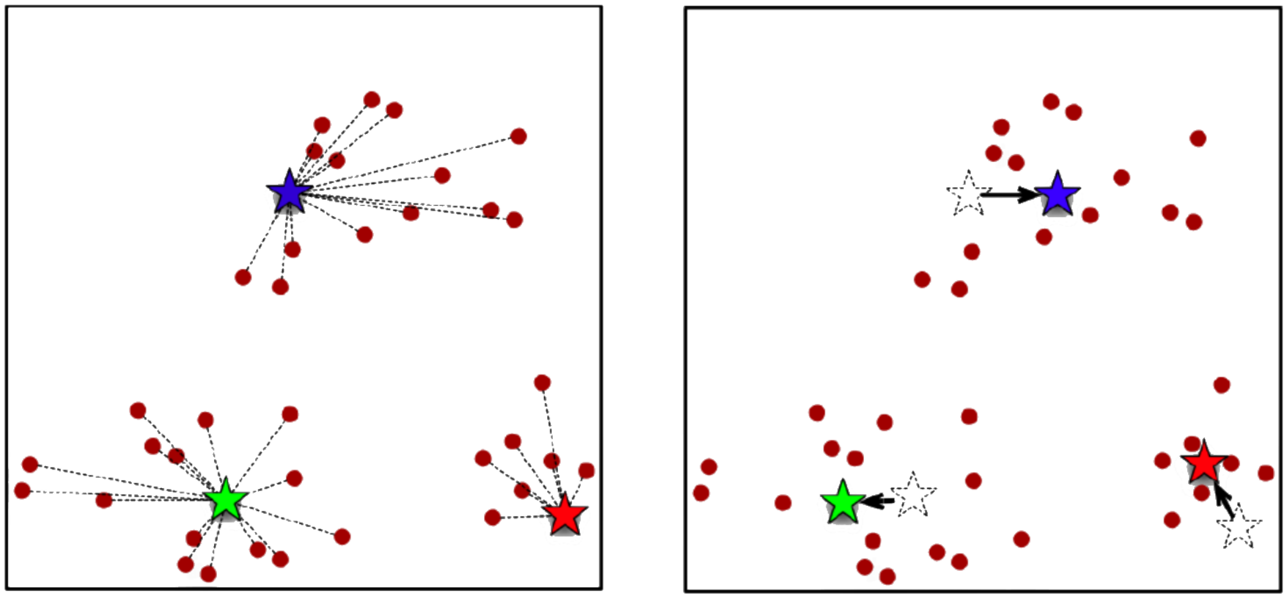


Figure 17: k–means cluster allocation (left) and updated centres (right).

k– **Means** As mentioned previously, k–means is a very natural way to group observations together (formally, k–means is linked to **Voronoi tilings**): 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 17);
7. repeating steps 4-6 until the clusters are **stable**.

For k–means, cluster centroids is 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 helps provide 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 to produce useful clusters if that assumption is not met in practice,
- it does not allow for **overlapping** or **hierarchical** groupings.

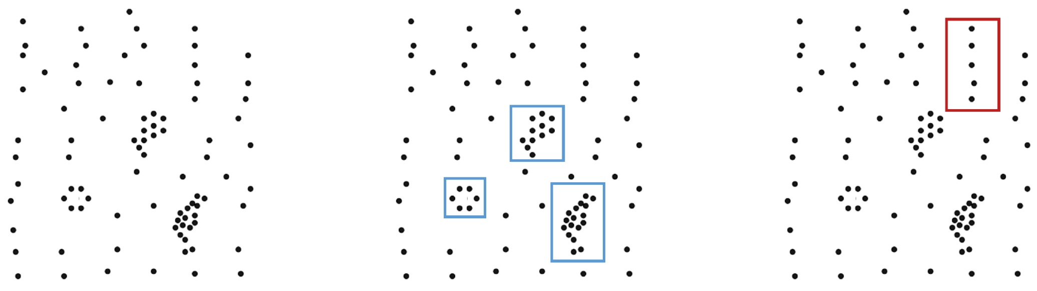


Figure 18: Artificial dataset; suggested clusters (blue), rejected suggestion (red).

Let us dwell some more on some of the issues relating to clustering **in general** (as opposed to specific to k—means):

- an implicit gamble is made, that of nearness of observations (in some metric) being linked with similarity, and large distances with dissimilarity – there are plenty of situations where this is an appropriate assumption to make (temperature readings on a map, for instance), but also many situations where it is less likely to be the case (physical proximity at a grocery's checkout counter, say... it is harder to find an example in this case due to our pattern-seeking habits, but the point remains solid);
- the **lack of a clear-cut definition** of what a cluster actually is makes it difficult to validate clustering results, although some guidelines exist (see Figure 18);
- the fact that various algorithms are **non-deterministic** is also problematic, and suggests that a clustering scheme should never be obtained using only one algorithmic pass; the outcome could be different depending on the **location** of random starting positions and the distance/similarity metric in use; essential patterns may emerge if the algorithms are implemented multiple times, with different starting positions and re-ordered data – for more information on the topic, see **cluster ensembles** in [42];
- for those algorithms that require the **number of clusters** as an input, it is difficult to determine what the optimal number should be (see Figure 19); 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 for a specific choice of parameter values in DBSCAN, say; a potential solution is to produce 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, and to look for "kinks" in the plot – such points provide cluster numbers at which an increase does not provide an in-step increase in clustering "resolution", so to speak;
- even when a cluster scheme has been accepted as valid, a **cluster description** might be difficult to come by – recall the examples provided when describing natural groups for unsupervised learning on 5 – should clusters be described using representative instances or average values or some combination of its' members most salient features?;
- most methods will find clusters in the data even if there are none – although DBSCAN is exempt from this **ghost clustering** (see Figure 20), and
- beware the temptation of ***a postiori*** **rationalisation** – once clusters have been found, it is tempting to try to "explain" them, but that is a job for domain experts, at best, and a waste of time and resources, at worst.

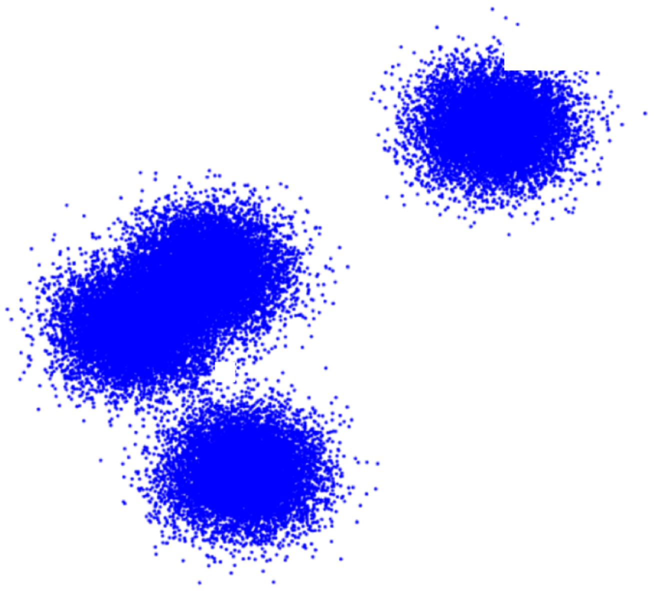


Figure 19: The number of clusters in a dataset is ambiguous: are there 2, 3, 4 or more clusters here?

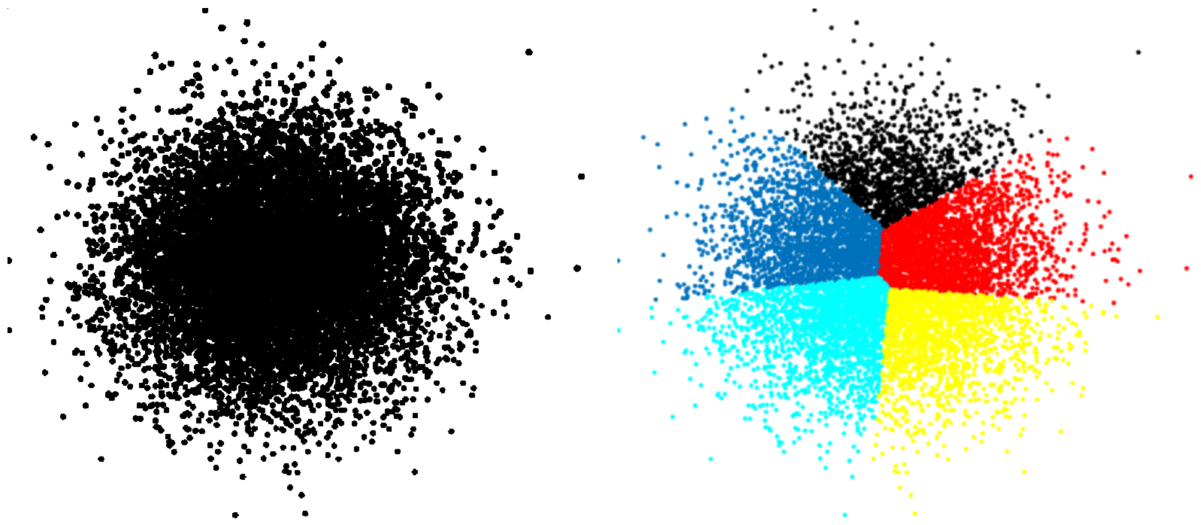


Figure 20: An illustration of ghost clustering with k—means, for $k = 5$.

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**.

An **optimal** clustering scheme is one which

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

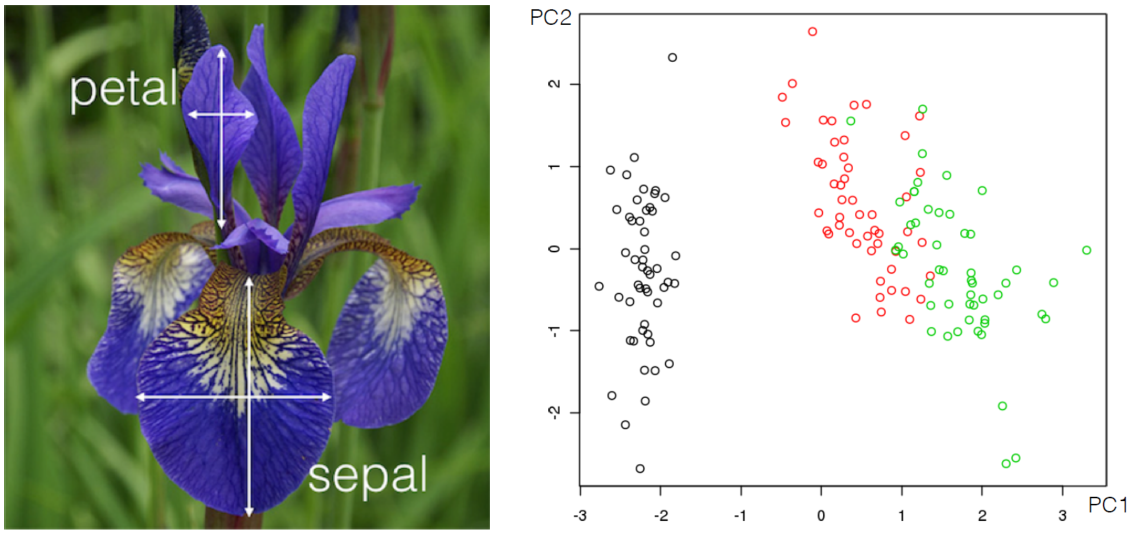


Figure 21: Illustration of iris measurements (left) and classification along 3 species, projected on 2 first principal components.

There are three main families of **clustering validation**:

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

Consider a clustering scheme $\mathscr{C} = \{\mathscr{C}_1, \ldots, \mathscr{C}_N\}$, with $\mathscr{C}_m$'s centroid denoted by c_m, and the average distance of $\mathscr{C}_m$'s members to c_m denoted by s_m. The **Davies-Bouldin Index** is defined as

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

where d is the appropriate distance metric. Since $\text{DB}_{\mathscr{C}}$ is only defined using the clustering results, it is an internal validation method. Heuristically, if the clusters are not greatly separated, we might expect $d(c_i, c_j)$ to be (relatively) small, and so $\text{DB}_{\mathscr{C}}$ to be (relatively) large. In the same vein, if the clusters are not homogeneous, we might expect $s_i + s_j$ to be (relatively) large, and so $\text{DB}_{\mathscr{C}}$ to be (relatively) large. In short, when the clustering scheme is not optimal, $\text{DB}_{\mathscr{C}}$ should be "large". This suggests another way to determine the optimal number of clusters – simply attempt to minimise $\text{DB}_{\mathscr{C}}$. Other methods have been suggested, let us briefly mention the **Sum of Squared Errors**, **Dunn's Index** and the **Silhouette Metric**.

Iris is a genus of plants with showy flowers. Fisher's iris dataset contains 150 observations of 5 attributes for specimens collected by Anderson, mostly from a Gaspé peninsula's pasture in the 1930s [63]. The attributes are **petal width**, **petal length**, **sepal width**, **sepal length**, and **species** (virginica, versicolor, setosa). A "description" of these features is provided by the picture in Figure 21 (left). This dataset has become synonymous with data analysis (to the point that the

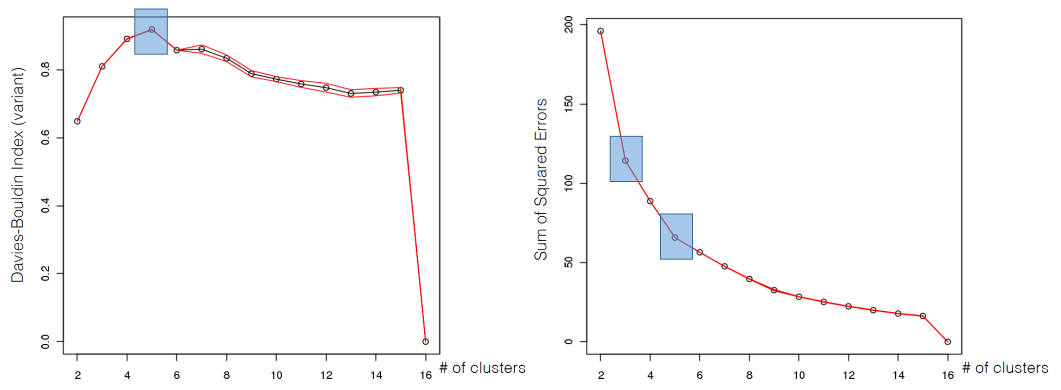


Figure 22: Optimal clustering results for the iris dataset: 5 clusters using (modified) Davies-Bouldin index and Sum of Squared Errors.

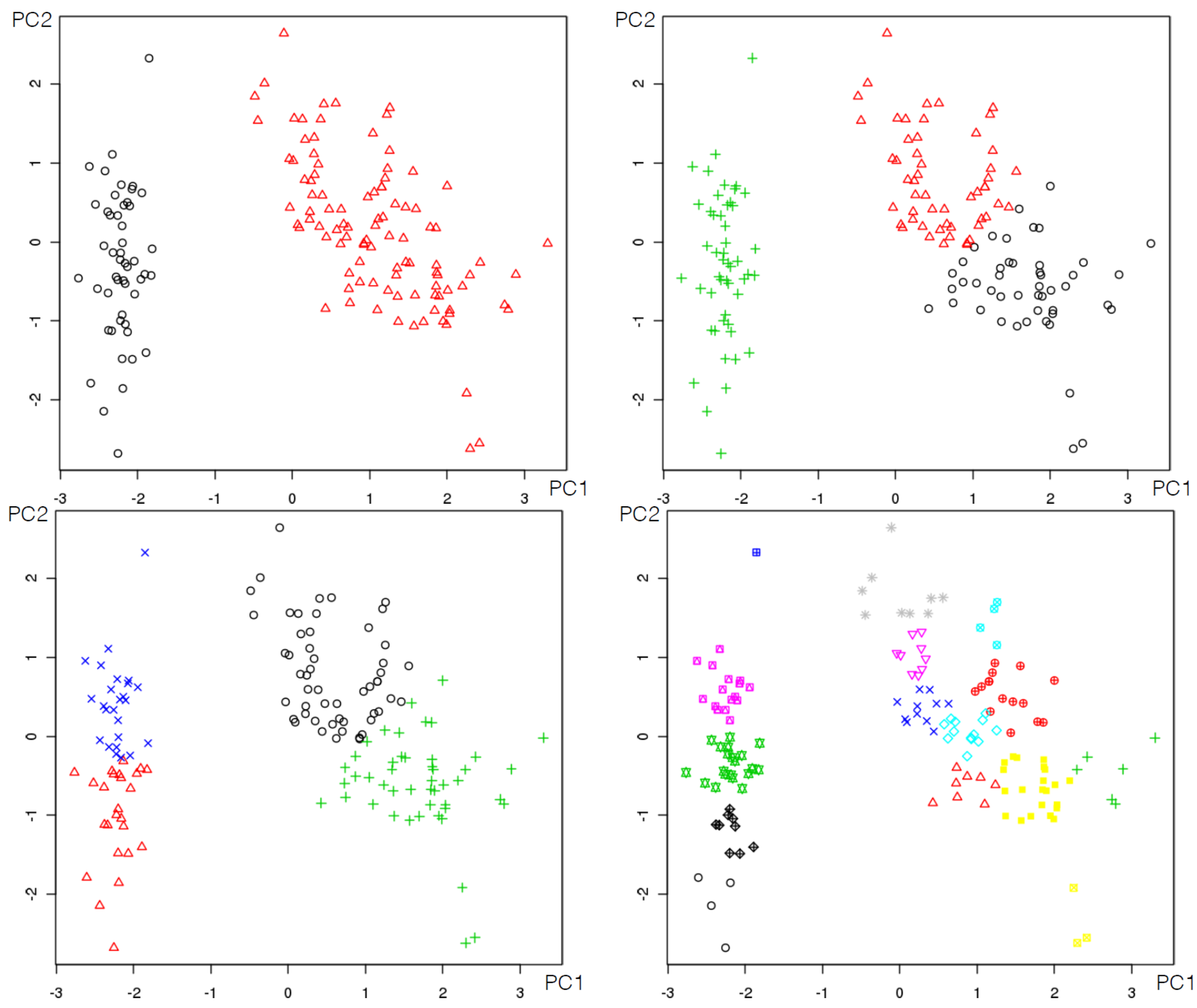


Figure 23: Clustering results on the iris dataset with k—means, for $k = 2, 3, 4, 15$ (from top left to bottom right, by row).

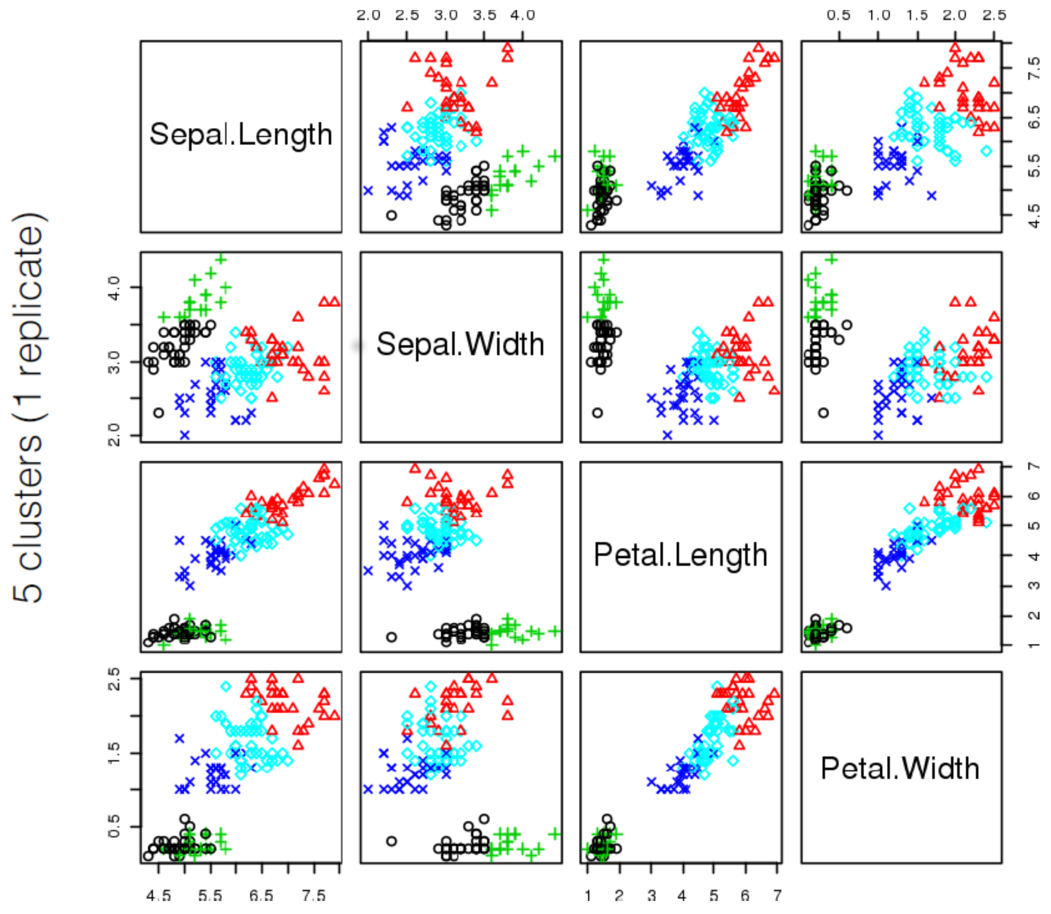


Figure 24: Optimal clustering results for the iris dataset: visualisation.

standard joke is that "it's not necessary to be a gardener to become a data analyst, but it helps"), being used to showcase just about every algorithm under the sun. This is, sadly, also what we are going to do.

A principal component projection of the dataset, with species indicated by colours, is shown in Figure 21 (right). 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 $k = 2, \ldots, 15$ and compute a (modified) Davies-Bouldin index and Sums of Squared Errors for each of the clustering schemes. The results for a single replicate are shown for $k = 2, 3, 4, 15$ in Figure 23; validation indicates that there seems to be either 3 of 5 natural k—means clusters in the data (see Figure 22). A single replicate with $k = 5$ is shown in Figure 24. Is it surprising that $k = 3$ was one of the candidates? Would you consider the final clustering scheme to be meaningful?

1.6.5 Issues and Challenges

The Stench of Bad Data

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.

– Q.E. McCallum, *Bad Data Handbook*

The data science landscape is littered with issues and challenges. Here are a few of them.

Bad Data We have touched upon data collection and data cleaning in Sections **??** and **??** and mentioned some of the most common data issues, namely that the available data is not always **representative** of the situation that the client would like to see modeled, and that it might not be sound. There are other potential issues, such as [83]:

- the data might formatted for human consumption, not machine readability;
- the data might contain lies and mistakes;
- the data might not reflect reality;
- there might be additional sources of bias and errors (not only imputation bias, but replacing extreme values with average values, proxy reporting, etc.);
- seeking perfection in the data can hamper the efforts of clients and consultants alike; there are different quality requirements for academic data, professional data, government data, military data, service data, commercial data, etc. – remember the engineering dictum: "close enough is good enough" (in terms of completeness, coherence, correctness, and accountability).

These accompany the **data analysis pitfalls**:

- analysing data **without understanding it**, or the context;
- using **one and only one tool** (by choice or by fiat) – neither the cloud nor Big Data will solve all of the client's problems!;
- analysing data for the sake of analysis;
- having **unrealistic expectations** of data analysis – consultants must bear some of the blame for this... don't oversell!

Overfitting In a traditional statistical model, p—values (or other statistics) can help validate how good a model is. For predictive data science models, such statistics cannot usually be computed. We recognise a "good" model based on how well it performs on unseen data. In practice, then, we hope for rules and models generated by data science techniques on a training set to be **generalisable 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 – the results are too specific to the training set (see Figure 25) for an illustration of **overfitting** and **underfitting**).

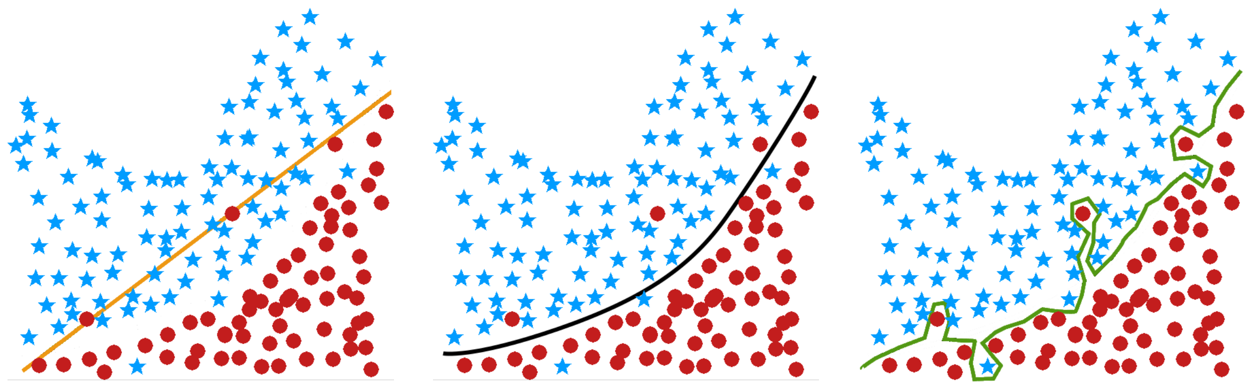


Figure 25: An illustration of underfitting (left) and overfitting (right); a reasonable fit is shown in the middle.

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

- **vague rule** – some people have black hair, some have brown hair, some blond, and some red (this underfit rule is probably too general to be useful);
- **reasonable rule** – in populations of European descent, approximately 45% have black hair, 45% brown hair, 7% blond and 3% red;
- **overly specific rule** – in every group of 100 individuals of European descent, we predict that there will be 46.3% with black hair, 47.2% with brown hair, 6.5% with blond hair, and 0% with red hair (this overfit rule is presumably based on some training set data without redheads).

With the overfit rule, we would predict that there are no redheads in populations of European descent, which is blatantly false. This result is simply specific to the particular training subset that was used – it was not representative of the entire dataset (assuming that the dataset itself was representative of the population of interest).

More formally, underfitting and overfitting can be viewed as resulting from the level of model complexity (see Figure 26). Overfitting can be overcome in several ways:

- **using multiple training sets**, 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 a small training set – a 70% - 30% split is often suggested.

Depending on the size of the data, there are different guidelines: 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**;
- **average-sized datasets** (less than a few thousand observations), use a few repetitions of 10-fold cross-validation (see Figure 27);
- **large datasets**, use a few repetitions of a holdout split (70%-30%, say).

IMPORTANT NOTE: remember to always, always (always!) evaluate the models on unseen data.

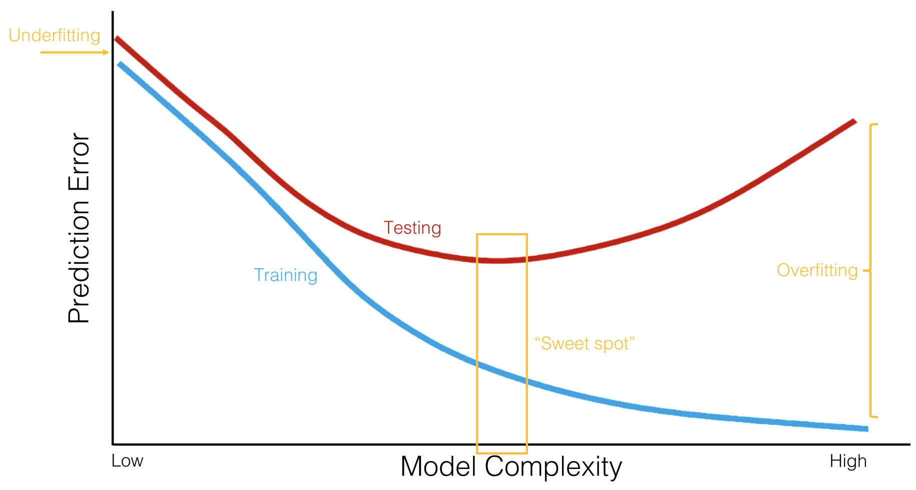


Figure 26: Underfitting and overfitting as a function of model complexity; error prediction on training sample (blue) and testing sample (red). High error prediction rates for simple models are a manifestation of underfitting; large difference between error prediction rates on training and testing samples for complex models are a manifestation of overfitting. Ideally, model complexity would be chosen to reach the situation's "sweet spot", but fishing for the ideal scenario might diminish explanatory power (based on [6]).

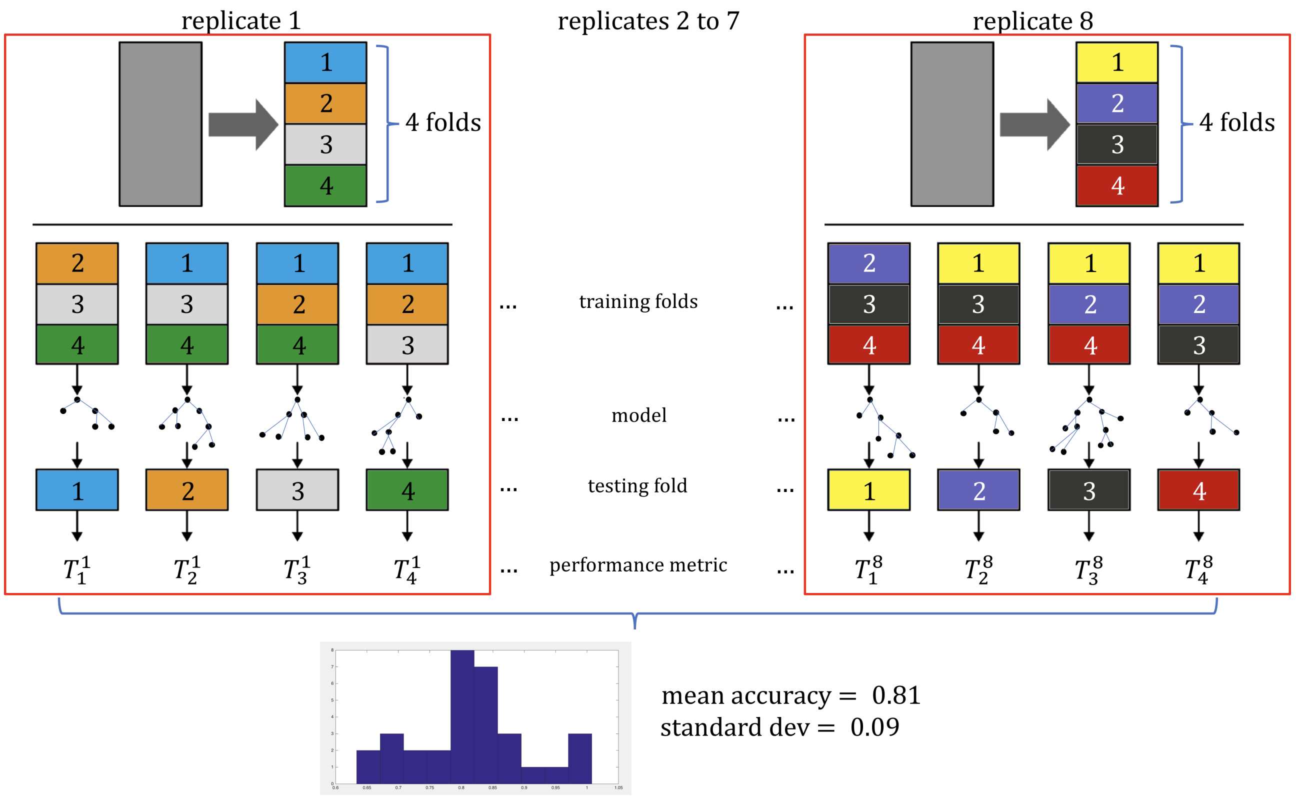


Figure 27: Schematic illustration of cross-fold validation, for 8 replicates and 4 folds; $8 \times 4 = 32$ models from a given family are built on various training sets (consisting of 3/4 of the available data – the training folds). Model family performance is evaluated on the respective holdout folds; the distribution of the performance metric values (in practice, some combination of the mean/median and standard deviation) can be used to compare various model families (based on [4, 38]).

Appropriateness and Transferability Data science models will be used heavily in the coming years (in fact, it has already started). We have discussed pros and cons of some of the applications on ethical and other non-technical grounds, but there are also **technical challenges**. In general, data science methods may **not** be appropriate if:

- you absolutely must use an existing (legacy) datasets instead of an ideal dataset (the dreaded "it's the best data we have!" response);
- the dataset has attributes that usefully predict a value of interest, but these attributes are not available at the time when a prediction is required (e.g. the total time spent on a website may be predictive of a visitor's future purchases, but the prediction must be made before the total time spent on the website is known);
- you will attempt to predict class membership using an unsupervised learning algorithm (e.g. 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?)

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 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 (e.g. 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). The problem is less academic than it might appear: **over-generalisations and inaccurate predictions can lead to harmful results**.

Myths and Mistakes We end this section with a short discussion of various data science (DS) myths and mistakes (based on [55], but they apply to quantitative consulting as a whole).

- Myth #1: DS is about algorithms
- Myth #2: DS is about predictive accuracy
- Myth #3: DS requires a data warehouse
- Myth #4: DS requires a large quantity of data
- Myth #5: DS requires technical experts
- Mistake #1: selecting the wrong problem
- Mistake #2: getting buried under tons of data without metadata understanding.
- Mistake #3: not planning the data analysis process
- Mistake #4: insufficient business and domain knowledge
- Mistake #5: using incompatible data analysis tools
- Mistake #6: using tools that are too specific
- Mistake #7: ignoring individual predictions/records in favour of aggregated results
- Mistake #8: running out of time
- Mistake #9: measuring results differently than the sponsor
- Mistake #10: naïvely believing what one is told about the data

IMPORTANT NOTE: it is your responsibility as a consultant to address these issues with the client. Do not assume that you are all on the same page – prod and ask.

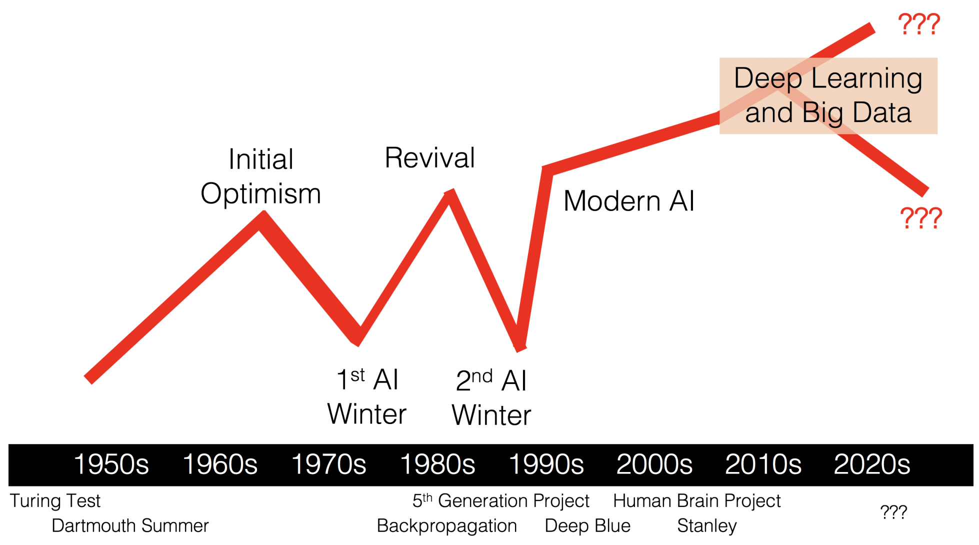


Figure 28: Conceptual timeline of the interest and optimism regarding artificial intelligence (A.I.); important milestones are indicated below the dates.

1.6.6 The Rest of the Data Science Picture

Even by the standards of this work, we have only scratched the surface of data science methods and tasks. In this section, we will briefly introduce a number of other related concepts. More information can be found in the references.

A.I. and Neural Networks

Deep Learning Indeed

Neural networks blow all previous techniques out of the water in terms of performance, but given the existence of adversarial examples, it shows we really don't understand what's going on.

– D. Gershgorn, *Quartz*

At various times since Alan Turing's seminal 1950 paper (in which he proposed the famous *Imitation Game* [94]), complete artificial intelligence has been announced to be "just around the corner" (see A.I winters, Figure 28). With the advent of **deep learning** and Big Data processing, optimism is as high as it's ever been. At the very least, it seems to be the topic *du jour*. Case in point, consider the following four recent headlines:

- "AlphaGo vanquishes world's top Go player, marking A.I.'s superiority over human mind" [South China Morning Post, May 27, 2017]
- "A Japanese A.I. program just wrote a short novel, and it almost won a literary prize" [Digital Trends, March 23, 2016]
- "Elon Musk: Artificial intelligence may spark World War III" [CNET, September 4, 2017]
- "A.I. hype has peaked so what's next?" [TechCrunch, September 30, 2017]

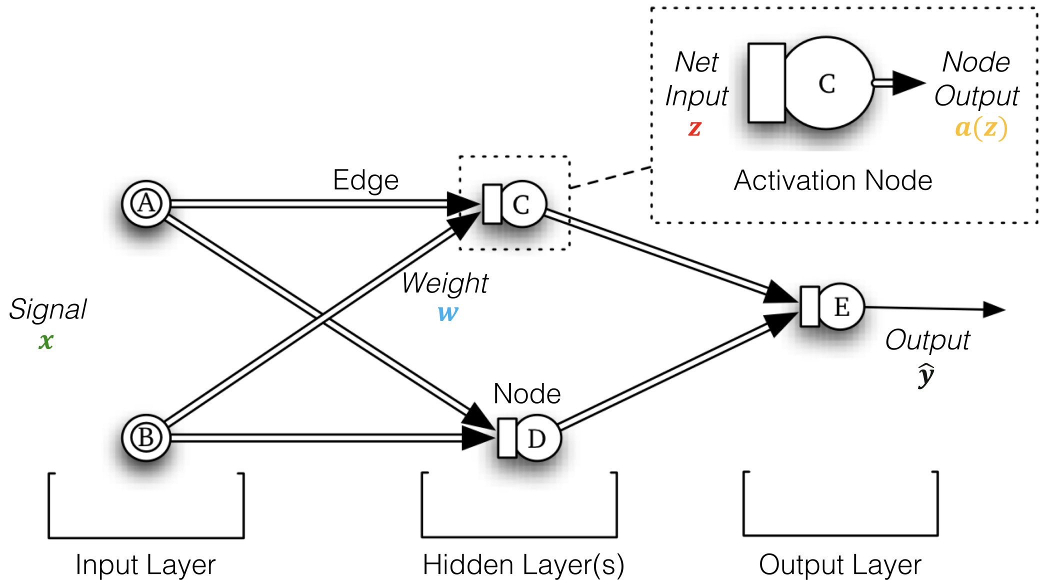


Figure 29: Artificial neural network topology – conceptual example. The number of hidden layers is arbitrary, as is the size of the signal and output vectors.

Opinions on the topic are varied – to some commentators, A.I. is both a brilliant success while to others it is a spectacular failure. What is A.I. and what is going on?

According to [84], the **essential qualities and skills of an intelligence** are that it:

- provides flexible responses in various scenarios;
- takes advantage of lucky circumstances;
- makes sense out of contradictory messages;
- recognises the relative importance of a situation's elements;
- finds similarities between different situations;
- draws distinctions between similar situations, and
- comes up with new ideas from scratch or by re-arranging previous known concepts.

A.I. research is defined as the study of *intelligent agents* – any device that perceives its environment and takes actions to maximise its chance of success at some task/goal [85]. Examples include

- **expert systems** – TurboTax, WebMD, technical support, insurance claim processing, air traffic control, etc.;
- **decision-making** – Deep Blue, auto-pilot systems, "smart" meters, etc.;
- **natural Language Processing** – machine translation, Siri, named-entity recognition, etc.;
- **recommenders** – Google, Expedia, Facebook, LinkedIn, Netflix, Amazon, etc.;
- **content generators** – music composer, novel writer, animation creator, etc.;
- **classifiers** – facial recognition, object identification, fraud detection, etc.

A trained **artificial neural network** (ANN) is a function that maps inputs to outputs in a useful way: it uses a Swiss-army-knife approach to providing outputs – plenty of options are available in the **architecture**, but it's not always clear which ones should be used. One of the reasons that

ANNs are so popular is that the user does not need to decide much about the function or know much about the problem space in advance – ANNs are **quiet models**.

Algorithms allow ANNs to **learn** (i.e. to generate the function and its internal values) automatically; technically, the only requirement is the ability to minimise a cost function (optimisation).

ANNs are formed from an **input layer** from which the **signal vector** x is inputted, an **output layer** which produces an **output vector** $\hat{y}$, and any number of **hidden layers**; each layer consists of a number of **nodes** which are connected to the nodes of other layers *via* **directed edges** with associated **weights** w (see Figure 29). Nodes from the hidden and output layers are typically **activation nodes** – the output $a(z)$ is some function of the input z. Signals propagate through the ANN using simple arithmetic, once a set of weights $\mathbf{w}$ and activation functions $\mathbf{a}(\cdot)$ have been selected (see Figure 30). In a nutshell, at each node, the neural net computes a weighted sum of inputs, applies an activation function, and sends a signal. This is repeated until the various signals reach the final output nodes.

That part is easy – given a signal, an ANN can produce an output, as long as the weights are specified. Matrix notation can simplify the expression for the output $\hat{y}$ in terms of the signal x, weights w, and activation function $a(\cdot)$. For instance, consider the network of Figure 29; if $a(z) =$, the network topology can be re-written as

Input: $\boldsymbol{X}_{(n\times p)} = \boldsymbol{X}_{(n\times 2)}$

$$\boldsymbol{X} = \begin{bmatrix} x_{A,1} & x_{B,1} \\ \vdots & \vdots \\ x_{A,n} & x_{B,n} \end{bmatrix}$$

Hidden units: $\boldsymbol{Z}^{(2)}_{(n\times M)} = \boldsymbol{Z}^{(2)}_{(n\times 2)}$

$$\boldsymbol{Z}^{(2)} = \begin{bmatrix} z_{C,1} & z_{D,1} \\ \vdots & \vdots \\ z_{C,n} & z_{D,n} \end{bmatrix} = \boldsymbol{X}\boldsymbol{W}^{(1)}$$

Weights: $\boldsymbol{W}^{(1)}_{(p\times M)} = \boldsymbol{W}^{(1)}_{(2\times 2)}$

$$\boldsymbol{W}^{(1)} = \begin{bmatrix} w_{AC} & w_{AD} \\ w_{BC} & w_{BD} \end{bmatrix}$$

Activation function: $\boldsymbol{a}^{(2)}_{(n\times M)} = \boldsymbol{a}^{(2)}_{(n\times 2)}$

$$\boldsymbol{a}^{(2)} = \begin{bmatrix} 1/\left(1+e^{-z_{C,1}}\right) & 1/\left(1+e^{-z_{D,1}}\right) \\ \vdots & \vdots \\ 1/\left(1+e^{-z_{C,n}}\right) & 1/\left(1+e^{-z_{D,n}}\right) \end{bmatrix} = g\left(\boldsymbol{Z}^{(2)}\right)$$

Activation function: $\boldsymbol{a}^{(2)}_{(n\times M)} = \boldsymbol{a}^{(2)}_{(n\times 2)}$

$$\boldsymbol{a}^{(2)} = g\left(\boldsymbol{Z}^{(2)}\right)$$

Output units: $\boldsymbol{Z}^{(3)}_{(n\times K)} = \boldsymbol{Z}^{(3)}_{(n\times 1)}$

$$\boldsymbol{Z}^{(3)} = \begin{bmatrix} Z_{E,1} \\ \vdots \\ Z_{E,n} \end{bmatrix} = \boldsymbol{a}^{(2)}\boldsymbol{W}^{(2)}$$

Weights: $\boldsymbol{W}^{(2)}_{(M\times K)} = \boldsymbol{W}^{(2)}_{(2\times 1)}$

$$\boldsymbol{W}^{(2)} = \begin{bmatrix} w_{CE} & w_{DE} \end{bmatrix}$$

Activation function: $\boldsymbol{a}^{(2)}_{(n\times K)} = \boldsymbol{a}^{(2)}_{(n\times 1)}$

$$\hat{\boldsymbol{y}} = \boldsymbol{a}^{(3)} = \begin{bmatrix} 1/\left(1+e^{-Z_{E,1}}\right) \\ \vdots \\ 1/\left(1+e^{-Z_{E,n}}\right) \end{bmatrix} = g\left(\boldsymbol{Z}^{(3)}\right)$$

from which we conclude that the output is

$$\hat{\boldsymbol{y}} = \boldsymbol{a}^{(3)} = g\left(\boldsymbol{Z}^{(3)}\right) = g\left[\boldsymbol{a}^{(2)}\boldsymbol{W}^{(2)}\right] = g\left[g\left(\boldsymbol{X}\boldsymbol{W}^{(1)}\right)\boldsymbol{W}^{(2)}\right].$$

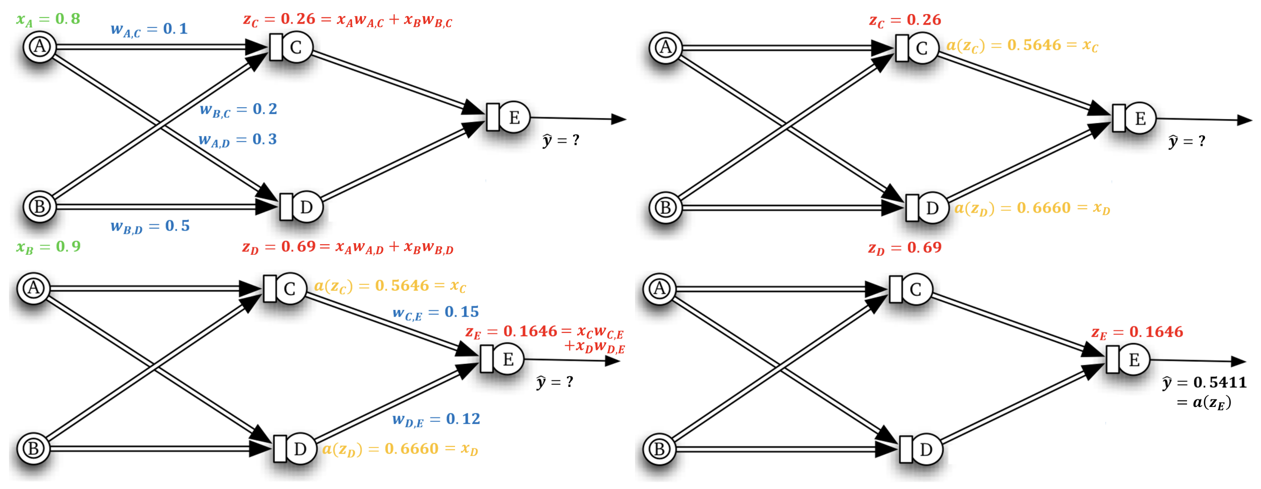


Figure 30: Signal propagating forward through an ANN; weights (in blue) and activation functions (in yellow) are given; inputs (in green), output (in black).

The problem is that unless the weights are judiciously selected, the output that is produced is unlikely to have anything to do with the desired output. For supervised learning tasks (i.e. when an ANN attempts to emulate the results of training examples), there has got to be some method to optimise the choice of the weights against an error function

$$R(W) = \sum_{i=1}^{n} \sum_{k=1}^{K} (\hat{y}_{ik}(W) - y_{ik})^2 \quad \text{or} \quad R(W) = -\sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \ln \hat{y}_{ik}(W)$$

(for value estimation and classification, respectively), where n is the number of observations in the training set, K is the number of output nodes in the ANN, y_{ik} is the known value or class label for the k^{th} output of the i^{th} observation in the training set – enter **backpropagation**, which is simply an application of the chain rule to $R(W)$ (the minimum is then found using numerical gradient descent).

ANNs can be quite accurate when making predictions – more than other algorithms, if given a proper set up (which can be hard to achieve). They degrade gracefully, and they often work when other things fail:

- when the relationship between attributes is **complex**;
- when there are a lot of dependencies/**nonlinear relationships**;
- when the inputs are **messy** and highlyconnected (images, text and speech), and
- when dealing with non-linear classification

On the other hand, they are relatively slow and prone to overfitting (unless they have access to large and diverse training sets). They are notoriously hard to interpret due to their **blackbox** nature, and there is no algorithm in place to select the optimal network topology. Finally, even when Even when they do perform better than other options, ANNs may not perform that much better due various **No Free-Lunch** theorems; and they are susceptible to various forms of adversarial attacks.

Text Mining and Natural Language Processing

Language and Data

Q: How many legs does a cat have if you call the tail a leg?
A: Four. Calling the tail a leg doesn't make it a leg.

– Old riddle, attributed to A. Lincoln

In this document, we take the position that **text mining** is the application of the typical data science tasks to text documents, with applications to **authorship questions** (classification), **sentiment analysis** (value estimation), **taxonomy creation** and **topic modeling** (clustering), **text description**, and **text visualisation**.

Natural language processing (NLP), in contrast, has a long history of lofty goals, which more or less boil down to developing machines that react "appropriately" while interacting with (natural) human languages. The focus of NLP tasks tends towards "understanding" languages; with common tasks including

- **syntax** (lemmatization, part-of-speech tagging, parsing, terminology extraction, sentence boundary disambiguation, stemming, word segmentation, etc.)
- **semantics** (machine translation, language generation, named entity recognition, optical character recognition, questions and answers, sentiment analysis, textual entailment, topic segmentation, word sense disambiguation, etc.)
- **discourse** (coreference resolution, discourse analysis, summarization, etc.)
- **speech** (recognition, segmentation, text-to-speech, etc.) [68]

Most natural human languages rules are dynamic, and usage may change drastically in space and time – a *poutine* is not the same delicious dish in New Brunswick as it is in Québec, for instance (see Figure 31) For another example, consider the meaning of the word *awful*, which drifted from

> "worthy of, or commanding, profound respect or reverential fear" to "frightful, very ugly, monstrous"

from 1000 AD onward. Other issues arise from dialect variations and individual-specific speech patterns, either due to linguistic drift, influence from other languages, sarcasm, idioms, figures of speech, and so forth. The intended meaning is often clear to experienced human speakers based on the specific context, but it is believed that natural language understanding is **AI-hard** – a complete resolution of the issues would require the ability to make computers as humans [69].

Thankfully, we rarely need a resolution at the most general level; in many areas, the current state of the art produces results which are acceptable to a large class of users. Since the 1990s, the NLP community has adopted a machine learning paradigm, which has provided advantages over the classical hard-coded hand-produced rules. Statistical machine translation, for instance, can take advantage of domain constraints and formal language habits to reduce the space of outputs and produce accurate translations of technical documents [70].

Let us take a look at two schools of thought regarding text mining: **semantic parsing** (SM) and **bag-of-words** (BoW) mining.

Figure 31: A poutine (left); an abomination in the eyes of all right-thinking sentient beings (right).

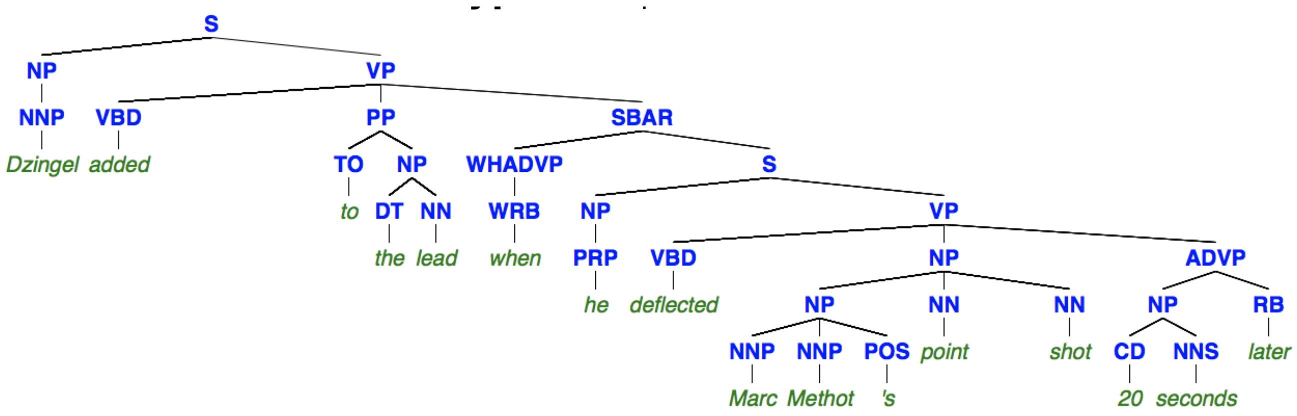


Figure 32: Syntactic parsing of a sentence using the Stanford parser [71].

In the SM view of text mining, word order and word type play a crucial role. The idea is to use a large number of hand-parsed sentences to train a model that outputs the most likely grammatical analysis of a sentence. Words are tagged along a tree structure, and may have multiple features. This information can then be used to extract insights about the sentence or document.

For instance, consider the sentence

> (S1) Dzingel added to the lead when he deflected Marc Methot's point shot 20 seconds later [72],

a syntactic parsing of which is shown in Figure 32 (the output of the Stanford parser [71]); another display is provided in Figure 33. From the tree diagram, a human observer can clearly see that "Marc Methot" is correctly parsed as a *noun phrase* (NP), that the "'s" is correctly identified as a *possessive marker* (POS), and that "Marc Methot's point shot" is correctly shown as a NP (built from 2 *singular proper nouns*, NNP), but the parser fails to recognize "point shot" as an NP. The two displays are, of course, equivalent. A computer program can be used to easily go from one to the other; a human with the right experience would find both as insightful. But it's certainly

```
(ROOT
 (S
  (NP (NNP Dzingel))
  (VP (VBD added)
    (PP (TO to)
      (NP (DT the) (NN lead)))
    (SBAR
      (WHADVP (WRB when))
      (S
        (NP (PRP he))
        (VP (VBD deflected)
          (NP
            (NP (NNP Marc) (NNP Methot)
                              ... (POS 's))
            (NN point) (NN shot))
          (ADVP
            (NP (CD 20) (NNS seconds))
            (RB later))))))))
```

Figure 33: Secondary view of syntactic parsing tree.

easier for a neophyte to comprehend the tree diagram. Why is that? Is it simply because we are a visual species? Or because most of us have parsed sentence fragments in our native languages as youths? In another parsing (using the Enju parser), "Marc Methot's point shot 20 seconds later" is tagged as a *simple declarative clause* (S), but "Marc Methot's point" and "shot 20 seconds later" are wrongly identified as a NP and a *verb phrase* (VP), respectively, underscoring the importance of parsing to our understanding of a sentence.

The **part-of-speech tagging** for the sentence is shown in the table below:

Word	Tag	Word	Tag
Dzingel	NNP	Marc	NNP
added	VBD	Methot	NNP
to	TO	's	POS
the	DT	point	NN
lead	NN	shot	NN
when	WRB	20	CD
he	PRP	seconds	NNS
deflected	VBD	later	RB

Notice how relational insight between the parts-of-speech has gotten lost (or is not displayed, at the very least).

The Stanford parser also provides a list of **universal dependencies**:

- nsubj(added-2, Dzingel-1)
- root(ROOT-0, added-2)
- case(lead-5, to-3)
- det(lead-5, the-4)

- nmod(added-2, lead-5)
- advmod(deflected-8, when-6)
- nsubj(deflected-8, he-7)
- advcl(added-2, deflected-8)
- compound(Methot-10, Marc-9)
- nmod:poss(shot-13, Methot-10)
- case(Methot-10, ’s-11)
- compound(shot-13, point-12)
- dobj(deflected-8, shot-13)
- nummod(seconds-15, 20-14)
- nmod:npmod(later-16, seconds-15)
- advmod(deflected-8, later-16)

For instance, “he” (the 7th token in the sentence) is the *nominal subject* (nsubj) of “deflected” (the 8th token), “point shot” is recognised as a *compound*, and “shot” (the 13th token) is the *direct object* (dobj, the second most core argument of a verb after the subject) of “deflected” (the 8th token).

In the BoW view of text mining, only the words are important – it is frequency (and relative frequency) that wins the day. In semantic parsing, the words have attributes depending on their position and role in the document’s sentences; in bag of words analysis, **the words themselves are attributes of the document**. Our sentence S1 is simply a collection of words, arranged here alphabetically:

’s, 20, added, deflected, Dzingel, he, later, lead, Marc, Methot, point, seconds, shot, the, to, when.

The fact that “point shot” is a noun phrase is not significant, but the fact that “point” and “shot” appear in the list is significant – it is the **relative frequencies** of the terms that provide information about the document or collection of documents (such as intent and themes).

No matter where text data comes from and what analyses we hope to run on it, the crucial first step requires extraction, formatting, and storage to a data structure with appropriate numerical properties:

- a **string** or vector of characters, with language-specific encoding;
- a collection of text documents (with meta information) called a **corpus** (‘permanent’ when stored on disk; ‘volatile’ when held in RAM);
- a **document-term matrix** (DTM) – or the transposed **term-document matrix** (TDM) – where the rows are the documents, the columns are the terms (see Figure 6), and the entries represent some text statistic;
- a **tidy text dataset** containing one **token** (single word, *n*-gram, sentence, paragraph) per row.

The DTM/TDM **representations** are **essential** to any statistical analysis of text data – it is on these entities that machine learning algorithms are unleashed.

Like every form of data, text data requires extensive cleaning and processing. Cleaning text data is, to put it mildly, even less pleasant a process than cleaning numeric or categorical data. There are challenges due to the nature of the data: how would one go about finding anomalies in the text? Outliers? Is the concept even definable for text data?

Character **encoding** may also produces surprises: a text (or a part of text) that looks completely normal to the naked eye may be unreadable to a computer because it was expecting a different coding system. There are probabilistic ways to detect a document's encoding, and ways to coerce a specified encoding – if you are working with text data and your code balks at doing something it should be able to do and none of the usual fixes apply, look into the encoding situation.

Another issue is that spelling mistakes and typographical errors are hard to catch in large documents (even with spell-checkers), to say nothing of:

- accent representation (*ya new cah's wicked pissa!*);
- neologisms and portmanteaus (*ruthfull*; *can't you tell that I'm planning prevenge?*);
- poor translations or foreign words (*I dou not spik inglishe*; *llongyfarchiadau*);
- puns and play-on-words (*they were yung and easily freudened!*);
- specialized vocabulary (*clopen*; *poset*);
- fictional names and places (*Qo'noS*; *Kilgore Trout*);
- slang and curses (*skengfire*; *#$&#!*);
- mark-up and tags (*<b>*; *\includegraphics*);
- uninformative text information (page number; ISBN blurb), etc.

The process can be simplified to some extent with the help of **regular expressions** and **text pre-processing functions** to:

- convert all letters to **lower case**, which should be avoided when seeking proper names;
- remove all **punctuation marks**, which should be avoided when seeking emojisl
- remove all **numerals**, which should be avoided when mining for quantities;
- remove extraneous **white spaces**;
- remove characters withing **brackets**, which should be avoided when seeking tags;
- replace all **numeral with words**, etc.

The purest BoW information about a term t in a document d is the raw **term frequency count** $\text{tf}_{t,d}$. Other measures are also used, such as the **relative term frequency** (or term proportion)

$$\text{tf}^*_{t,d} = \frac{\text{tf}_{t,d}}{M_d},$$

the **relative document frequency**

$$\text{df}^*_t = \frac{\sum_d \text{sign}\left(\text{tf}_{t,d}\right)}{N}$$

or the **term frequency-inverse document frequency**

$$\text{tf-idf}_{t,d} = -\text{tf}^*_{t,d} \times \ln \text{df}^*_t,$$

	Document 1	Document 2	Document 3	…	Document N	Sum
Token 1	0	0	1	62	3	66
Token 2	0	1	0	61	2	64
Token 3	1	0	3	101	0	105
…	112	24	38	84	0	258
Token M	2	2	0	12	3	19
Sum	115	27	42	320	8	

Table 6: Term-Document Matrix / Document-Term Matrix for a hypothetical corpus, with row sums and column sums.

where M_d is the number of retained terms in sentence d, and N is the number of documents in the corpus. If **all the corpus' documents** contain the term t, then $\text{df}^*_t = 1$ and

$$\text{tf-idf}_{t,d} = -\text{tf}^*_{t,d} \times \ln 1 = 0,$$

which is to say that this specific term does not provide information in the tf-idf framework. If a term t **rarely occurs** in a document d, then $\text{tf}^*_{t,d} \approx 0$ and

$$\text{tf-idf}_{t,d} \approx 0 \times \ln \text{df}^*_t = 0.$$

Terms that appear relatively often only in a small subset of documents are crucial to understanding those documents in the general context of the corpus – that's the interpretation of the tf-idf statistic (although we hasten to add that this argument is not rigorous and is considered at best a heuristic by many practitioners).

The following example provides an illustration of the various **text statistics**.

Consider a corpus $\mathscr{C} = \{d_1, \ldots, d_3\}$ consisting of of the following 3 documents:

> d_1 =“the dogs who have been let out”, d_2 =“who did that?”, and d_3 =“my dogs breath smells like dogs food” [sic].

Suppose that after processing, the following 14 terms are retained:

> t_1 =“been”, t_2 =“breath”, t_3 =“did”, t_4 =“dogs”, t_5 =“food”, t_6 =“have”, t_7 =“let”, t_8 =“like”, t_9 =“my”, t_{10} =“out”, t_{11} =“smells”, t_{12} =“that”, t_{13} =“the”, and t_{14} =“who”.

$\text{tf}^*_{t,d}$		t													
		1 been	2 breath	3 did	4 dogs	5 food	6 have	7 let	8 like	9 my	10 out	11 smells	12 that	13 the	14 who
d	1	1/7	0	0	1/7	0	1/7	1/7	0	0	1/7	0	0	1/7	1/7
	2	0	0	1/3	0	0	0	0	0	0	0	0	1/3	0	1/3
	3	0	1/7	0	2/7	1/7	0	0	1/7	1/7	0	1/7	0	0	0

(a) Relative term frequency

df^*_t	t													
	1 been	2 breath	3 did	4 dogs	5 food	6 have	7 let	8 like	9 my	10 out	11 smells	12 that	13 the	14 who
	1/3	1/3	1/3	2/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	2/3

(b) Relative document frequency

tf–idf^*_t		t													
		1 been	2 breath	3 did	4 dogs	5 food	6 have	7 let	8 like	9 my	10 out	11 smells	12 that	13 the	14 who
d	1	0.16	0	0	0.06	0	0.16	0.16	0	0	0.16	0	0	0.16	0.06
	2	0	0	0.37	0	0	0	0	0	0	0	0	0.37	0	0.14
	3	0	0.16	0	0.12	0.16	0	0	0.16	0.16	0	0.16	0	0	0

(c) Term frequency – inverse document frequency

Table 7: Illustration of text statistics on the toy corpus $\mathscr{C}$.

The text statistics of this corpus are shown in Table 7. Of course, the choice of a text statistic will have an effect on the numerical representation of the corpus, and so on the analytical results that are produced.

Let us take a look at one specific text mining task. Most of us have a good native understanding of the emotional intent of words, which leads us to infer **surprise**, **disgust**, **joy**, **pain** (and so forth) from a text segment. The process, when applied by machines to a block of text, is known as **sentiment analysis** (or opinion mining), and it used to answer questions such as:

- "Is this movie review positive or negative?"
- "Is this customer email a complaint?"
- "Have newspapers' attitudes about the Prime Minister changed since the election?"

Most humans would typically be able to answer these questions when presented with the appropriate text documentation. But there are challenges, which we may have a difficult time translating into the machine learning paradigm, such as:

- we don't always agree on the emotional content of a text;
- words may have different meaning/emotional value depending on the context (anti-antonyms);

- qualifiers can drastically change a term's emotional value;
- topic changes mid-text are not always easily identifiable, and neither is the use of rhetorical devices.

Sentiment analysis is a **supervised learning** problem, thus requiring dictionaries of emotional content to have been compiled ahead of time (internally or externally). Related tasks include:

- discarding subjective information (information extraction);
- recognising opinion-oriented questions (question answering);
- accounting for multiple viewpoints (summarisation);
- identifying suitability of videos for kids;
- identifying bias in news sources, and
- identifying appropriate content for ad placement.

For instance, consider the following three product reviews:

1. "Love the jeans, price, fit, but even more, love the suppliers. Simple concerns were not only answered immediately, they went beyond any expectations I had! Will definitely be buying through this supplier, highly recommended!"
2. "DON'T BUY. Great series aside, this special addition is pathetic. They're basically mass-market paperbacks: small and uncomfortable to hold. The regular paperback versions are far superior for basically the same price."
3. "Beginning the second use, the bowl keeps falling out 30 seconds after the mixing starts. A bit disappointed."

One of those reviews was given 5 stars, another 3 stars, and the final one 1 star. Can you determine which was which (and what product is being reviewed)? Would it surprise you to discover that the first one was the 5-star review, the second one the 3-star review, and the last one the 1-star review?

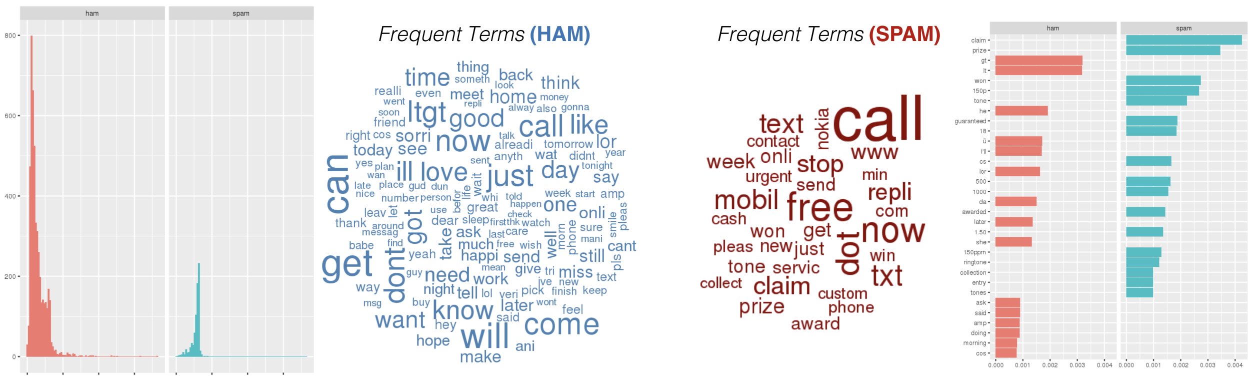
There are two types of sentiment analyses:

- **term-by-term** (TBT) looks at the emotional content of tokens and tries to deduce a score for passages containing them, whereas
- **document-by-document** (DBD) looks at scored passages and tries to find tokens which carry the emotional load or predict how a new passage would score on some emotional spectrum.

TBT is not a complicated technical task – it only requires the ability to match a lexicon score to a term, and to add the scores. TBT it relies heavily on **lexicons** – list of terms which have been ranked on some emotional scale

- AFINN: words on a scale from -5 (negative) to 5 (positive)
- BING: binary negative/positive
- NRC: words are assigned category(ies) of sentiments
- LOUGHRAN: categorical bins

For instance, "abandon" scores a -2 (AFINN), NA (BING), fear, negative, sadness (NRC), and negative (LOUGHRAN), whereas "not" scores NAs across the board. Each of these lexicons contains



(a) SMS lengths **(b)** Wordclouds **(c)** TF-IDF terms

Figure 34: Sentiment analysis of Ham and Spam dataset.

NBC		Actual		
		Ham	***Spam***	***Total***
Prediction	**Ham**	1437	32	88%
	Spam	10	194	12%
	Total	86%	14%	1673

SVM		Actual		
		Ham	***Spam***	***Total***
Prediction	**Ham**	1404	28	86%
	Spam	43	198	14%
	Total	86%	14%	1673

Table 8: Confusion matrices for classification of the Ham and Spam dataset; NBC (left), SVM (right).

a majority of **negative** terms, and context, as always, dictates the best choice. Once a lexicon has been selected, TBT is simply a matter of **chunking the text** and computing sentiment scores on each block (every 100 words, every 100 lines, every chapter, etc.)

DBD, on the other hand, is basically a classification problem, and so requires a training set with labeled outcomes. The Ham or Spam dataset consists of 5574 text messages from the Singapore area, with 3 variables: an SMS message, the length of the message (# of characters), and **Ham** or **Spam** labels (affixed by human observers after the fact). For instance, one of the spam messages reads:

> SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info

while one of the ham messages reads:

> Dear i am not denying your words please

The training set is built from 70% of the dataset, selected randomly. After processing, all terms appearing in fewer than 10 documents are removed, leaving a DTM with 613 columns. Classification is obtained via naive Bayes and support vector machines. The results are shown in Figure 34 and Table 8. Which of these classifiers is best? What if it's more important for you to never miss a true message? What if it's more important for you to never receive spam?

Big Data and Distributed Computing

Smokes and Mirrors

With too little data, you won't be able to make any conclusions that you trust. With loads of data you will find relationships that aren't real... Big data isn't about bits, it's about talent.

– D. Merrill, *Forbes*

Big data advocates are extremely enthusiastic about its potential to change the world and the way we analyse data. Interest in big data isn't about to simply fade away; we'd be remiss not to say a few words about this phenomenon. The portrait we paint will be incomplete, in particular we'll only briefly touch on the *MapReduce* paradigm, and say nothing of Spark, Hadoop, H20, and other framework.

Big data is often described using the **5-V paradigm**: volume, velocity, variety, veracity and value (some commentators have gotten in on the act and have added variability and visualisation).

- **volume** refers to large amounts of data produced by social media, sensors,etc.;
- **velocity** refers to the speed at which data is created, and the speed at which it can be accessed and processed;
- **variety** refers to the different types of available data, which can't all be saved in relational databases (tables, pictures, conversations, etc.);
- **veracity** refers to the fact that the quality and accuracy of big data is harder to control, but nonetheless analysable;
- **value** refers to the ultimate goal of many practitioners: to turn all of this data into something that can be used.

More specifically, small data consists of gigabytes to terabytes of centrally accessible and analysable structured data with known complex relationships, whereas big data consists of petabytes to exabytes of distributed semi-structured data with few known complex relationships. Big data tends to be incomplete and messy, and the focus is on managing (storage, access) and exploring the data; small data tends to be complete (or potentially complete) and clean, and the focus tends to be measuring and reporting on past performance.

At least, that's the idea. In the context of data analysis, a major difficulty is that a lot of the techniques were built around very small dataset: the theory remains correct, but the direct approach will leave even the best analysts waiting thousands of years for results.

Many computations happen instantly, others take a significant amount of time. Try doing some analysis in R or Python while steadily increasing the size of the data. *What happens*? Soon, your computer will start to lag. Eventually, the time required to complete the computations will become impractically long. Optimising your code and using a faster CPU can only get you so far. *Moore's Law* has 'tapered off' for CPU clock speeds (inasmuch as it ever applied in the first place, which it probably didn't). **This is the big data problem.**

Now, consider the two following performance metrics: MIPS (millions of instructions per second) and FLOPS (floating point operations per second, which is more relevant for numerical computations). MIPS/$ has increased by a factor of 10 every 5 years or so since 1940, FLOPS/$ every 8 years or so [66]. If you measure total capacity of a cluster by MIPS (which basically assumes you can parallelise with 100% efficiency), every five years the computational power of a cluster you can buy multiplies by 10. This is essentially the solution to the big data problem: **throw money at the problem and/or wait**.

But what does it mean to parallelise the analytical process? What if you could split your computation among multiple CPU cores, multiple CPUs, or multiple computers and divide the computation time by a factor of 4, or 32, or 1000? These factors can allow you to run algorithms on big data that keep your analytics, recommendations, smart services updated daily, hourly, or even in real time. The **election analogy to parallelisation** (see Figure 35) is simple to understand, yet carries most of the conceptual ideas:

- votes are counted at different polling stations in a riding;
- each station simultaneously counts its own votes and reports their total, and
- the totals of all polling stations are aggregated at Elections HQ

A single individual counting all the ballots would eventually get the same result (assuming that they don't make a mistake), but it would take far too long to get the result. The gains from parallelism depend on whether serial algorithms can be adapted to make use of parallel hardware. The **pizzeria analogy for limitations** of parallelisation also provides insights into (so-called) bottlenecks:

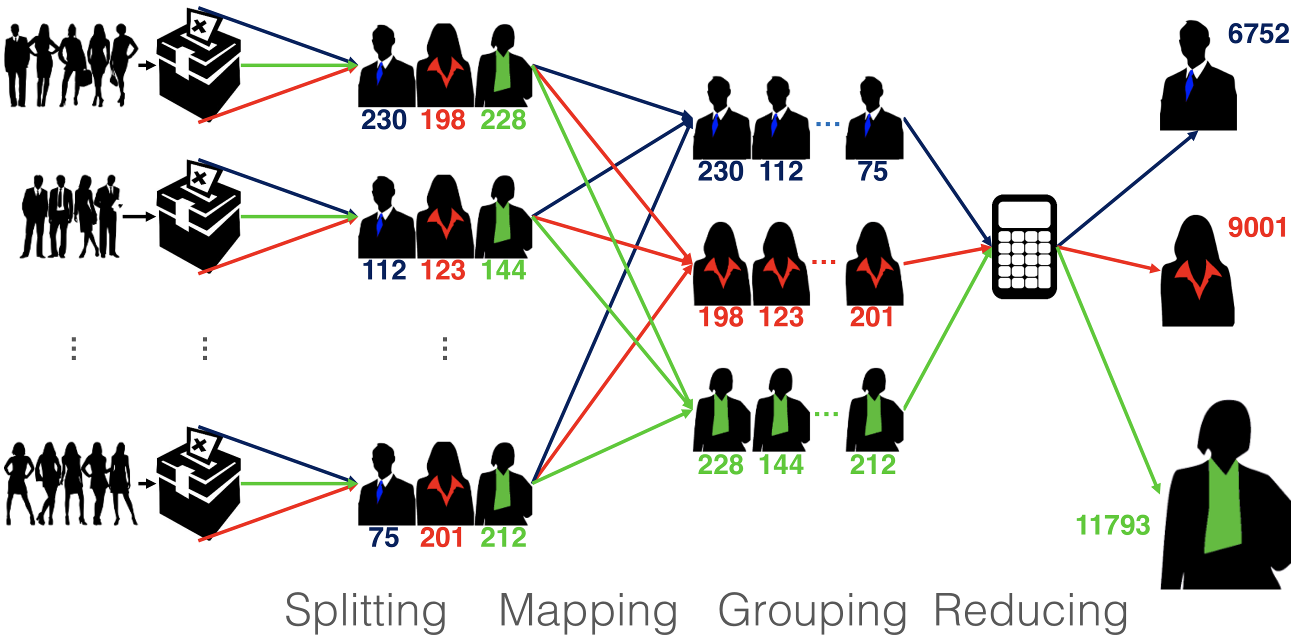
- multiple cooks can prepare toppings in parallel;
- but baking the crust can't be parallelised, and
- doubling oven space will increase the number of pizzas that can be made simultaneously but won't substantially speed up any one pizza.

Sometimes bottlenecks prevent parallelism from offering gains – if there is only one ladle, people lining up on both sides of a table to get some soup won't get their soup any faster than if they were just lining up on one side.

Thankfully, many of the regular data science techniques and quantitative methods have been **parallelised** and can now be used on large data sets. To use but one atypical example, the authors of [65] have proposed the concept of "computing near the data"

> to reduce the amount of data that needs to be transferred, we should perform as many operations as possible at the location where the data resides, and only communicate the reduced summary information across to the statistical system.

Statistical computations usually involve two types of computations: the first type is simpler in nature (such as summarising the data) but requires the whole data set; the second type is a more complicated manipulation of the summary results of the first type, but typically require a small amount of data.



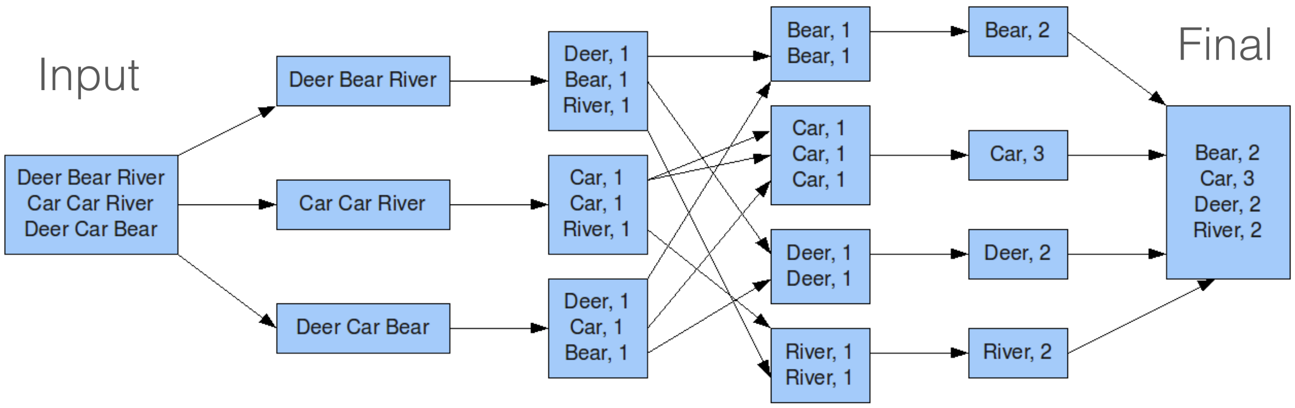


Figure 35: Ilustrations of the *MapReduce*; election analogy (above); word count (below).

For example, optimisation problems seek to maximise some objective function

$$\max_{\beta} L(\beta, X),$$

where β is a parameter vector and X is the available data. Typically, the solution is computed iteratively according to a scheme of the form

$$\beta^{(t+1)} = \beta^{(t)} + U(X, \beta^{(t)}),$$

where U depends on the selected algorithm and the function L; computing U is the data-intensive part of the algorithm. The calculation usually involves multiple steps, only some of which require X (such as computing the objective function or its derivative).

This suggests that the data-intensive operations which need to be performed on the entire data set near where this data set is located:

the summary results are sent back to the server responsible for the algorithmic flow. This server then in turn carries out more complicated actions such as constructing the correction factor U, inverting a matrix, or computing a linear interpolation. In other words, we break computation into pieces. Some pieces are carried out on the database side while others are carried out inside a statistics server.

Finally, we'd like to leave the floor to **big data sceptics**, who remind us that big data is not a panacea [64]:

- **Big Data is no crystal ball:** collected data can only come from the past. Analysing the data may provide a link between actions and consequences, which can be used to predict the Future all things being equal. But "past performance does not guarantee future results."
- **Big Data can't dictate your personal or organisational values:** the right answer (from a values perspective) may appear to be the wrong one (from the data science perspective). Data-based conclusions do not live in a vacuum: context matters. Blind obedience to data-driven results is just as dangerous as rejection based on gut-reaction.
- **Big Data can't solve every problem:** big data techniques have a successful track record. The approaches work most of the time. But as has been mentioned before, "when all you have is a hammer, everything looks like a nail." There is a huge number of different problems to be solved; not every single one of them calls for big data.

Feature Selection Removing **irrelevant** or **redundant** variables is a common data processing task. The motivations for doing so are varied, but usually fall into one of two categories:

- the modeling tools do not handle redundant variables well, due to **variance inflation**;
- as an attempt by the analysts to overcome the **curse of dimensionality** (see Section **??**) or to avoid situations where the number of variables is larger than the number of observations.

Filter methods inspect each variable individually and score them according to some importance metric; the less relevant features (i.e. the features whose importance scores below some preset threshold) are then removed. **Wrapper methods**, on the other hand, seek feature subsets for which the evaluation criterion used by the eventual analytical method is "optimised". The process is **iterative**, and typically computationally intensive: candidate subsets are used in the analysis until one produces an acceptable evaluation metric for the analysis.

The fundamentals of machine learning can also be used to select an optimal feature set. **Unsupervised methods** determine the importance of a feature based only on its values. **Supervised methods** evaluate each feature's importance by studying the relationship with a **target feature** (such as mutual information and correlation, in which features that are highly correlated with the target variable are retained, but this approach is limited if the relationship to the target variable is non-linear). Wrapper methods tend to be supervised. Unsupervised filter methods search for noisy features and include the removal of constant variables, of ID-like variables (i.e. different on all observations, say), of features with low variability, etc.

Stepwise selection is a form of *Occam's Razor*: at each step, a new feature is considered for inclusion or removal from the current features set based on some criterion (F—test, t—test, etc.) Such methods are extremely common, but they have severe limitations (which are not usually addressed):

- the tests are biased, since they are all based on the same data;
- the adjusted R^2 nly takes into account the number of features in the final fit, and not the degrees of freedom that have been used in the entire model;
- if cross-validation is used, stepwise selection has to be repeated for each sub-model but that is not usually done, and
- it represents a classic example of p—hacking.

An interesting solution is provided by the LASSO. In what follows, we assume that we N **centered** and **scaled** observations $x_i = (x_{1,i}, \cdots, x_{p,i})^\top$ and a target observation y_i. Let

$$\hat{\beta}_{\text{LS},j} = [(\mathbf{X}^\top\mathbf{X})^{-1}(\mathbf{X}^\top\mathbf{y}]_j$$

be the j^{th} ordinary least square (OLS) coefficient, and set a threshold $\lambda > 0$, whose value depends on the training dataset (there is more here than meets the eye [99]). Recall that $\hat{\beta}_{\text{LS}}$ is the solution to the OLS problem

$$\hat{\beta}_{\text{LS}} = \text{arg min}_\beta\{\|\mathbf{y}-\mathbf{X}\beta\|_2^2\}.$$

In general, there are **no restrictions** on the values taken by the coefficients $\hat{\beta}_{\text{LS},j}$ – large magnitudes imply that corresponding features **play an important role** in predicting the target.

Ridge regression is a method to **regularise** the OLS regression coefficients (the effect is simply to shrink the OLS coefficients). The problem consists in solving a modified version of the OLS scenario:

$$\hat{\beta}_{\text{RR}} = \text{arg min}_\beta\{\|\mathbf{y}-\mathbf{X}\beta\|_2^2 + N\lambda\|\beta\|_2^2\},$$

which yields **ridge coefficients**

$$\hat{\beta}_{\text{RR},j} = \frac{\hat{\beta}_{\text{LS},j}}{1+N\lambda}.$$

Regression with best subset selection is another method that sets some of the OLS regression coefficients to 0 (the effet is simply to select some of the OLS coefficients, potentially). The problem consists in solving another modified version of the OLS scenario:

$$\hat{\beta}_{\text{BS}} = \text{arg min}_\beta\{\|\mathbf{y}-\mathbf{X}\beta\|_2^2 + N\lambda\|\beta\|_0\}, \quad \text{where } \|\beta\|_0 = \sum_j \text{sgn}(|\beta_j|),$$

which yields **best subset** coefficients

$$\hat{\beta}_{\text{BS},j} = \begin{cases} 0 & \text{if } |\hat{\beta}_{\text{LS,j}}| < \sqrt{N\lambda} \\ \hat{\beta}_{\text{LS,j}} & \text{if } |\hat{\beta}_{\text{LS,j}}| \geq \sqrt{N\lambda} \end{cases}$$

Least absolute shrinkage and selection operator (LASSO). The problem consists in solving yet another modified version of the OLS scenario:

$$\hat{\beta}_{\text{BS}} = \text{arg min}_\beta\{\|\mathbf{y}-\mathbf{X}\beta\|_2^2 + N\lambda\|\beta\|_1\},,$$

which yields **lasso** coefficients

$$\hat{\beta}_{\text{L},j} = \hat{\beta}_{\text{LS},j} \cdot \max\left(0, 1 - \frac{N\lambda}{|\hat{\beta}_{\text{LS},j}|}\right).$$

LASSO combines the properties of ridge regression (**regularisation**) and best subset selection (**feature selection**) – it selects at most $\max\{p,N\}$ features, and usually selects no more than one feature in a group of highly correlated variables. Other **extensions** exist: elastic nets; group, fused and adaptive lassos; bridge regression, etc.

The main take-away is to be extremely wary of stepwise selection methods for dimensionality reduction.

LASSO and its variants are useful when the underlying model is linear or logistic regression, but for many data science applications, that might not necessarily ultimately be the case. **Principal component analysis** (PCA) can be used to find the combinations of variables along which the data points are **most spread out**; it attempts to fit a $p-$**ellipsoid** to a centered and scaled representation of the data. The ellipsoid axes are the principal components of the data. Small axes are components along which the variance is "small"; removing these component leads, in an ideal setting, to a "small" loss of information (although there are scenarios where it could be those "small" axes that are more interesting).

The procedure is simple:

1. centre and scale the data to obtain a matrix $\mathbf{X}$;
2. compute the data's covariance matrix $\mathbf{K} = \mathbf{X}^\top\mathbf{X}$;
3. compute $\mathbf{K}$ s eigenvalues $\mathbf{\Lambda}$ and orthonormal eigenvectors matrix $\mathbf{W}$;
4. each eigenvector $\mathbf{w}$ (also known as **loading**) represents an axis, whose variance is given by the associated eigenvalue λ.

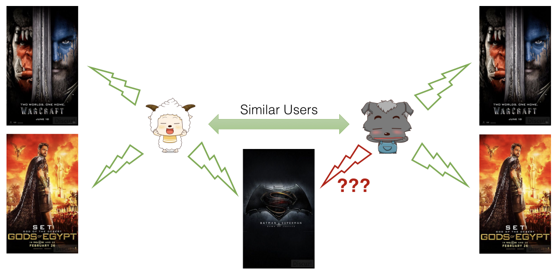
The loading that explains the most variance along a single axis is the eigenvector of the empirical covariance matrix corresponding to the largest eigenvalue, and that variance is proportional to the eigenvalue. The process is repeated, yielding **orthonormal** principal components $\text{PC}_1,\ldots',\text{PC}_r$, where $r = \text{rank}(\mathbf{X})$.

The **proportion of the spread in the data** which can be explained by each principal can be placed in a **scree plot**. How many principal components (PCs) should be retained in the analysis? We can either keep the PCs for which the cumulative proportion is below some threshold, or keep the PCs leading to a **kink** in the scree plot.

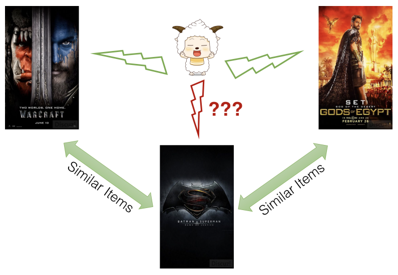
PCA is commonly used, but there are some limitations to keep in mind:

- it is dependent on scaling, and so is not uniquely determined;
- with little domain expertise, it may be difficult to interpret the PCs;
- it is quite sensitive to outliers, and
- the data assumptions are not always met – in particular, does it always make sense that important data structures and data spread be correlated (the **counting pancakes** problem), or that the components be **orthogonal**?

There are other methods to find the **principal manifolds** of a dataset, including UMAP, self-organizing maps, auto-encoders, curvilinear component analysis, manifold sculpting and kernel PCA [100].



(a) User-based



(b) Item-based

Figure 36: Illustration of collaborative filtering recommender engines – fair warning, we have neither seen nor heard of the movies depicted here.

Recommender Systems When we decide which movie to watch, which book or video game to purchase, where to take a vacation, etc., we often ask friends, colleagues, family for **suggestions**, on the assumption that their tastes are similar to ours. We may also opt to consult reviews of (or feedback on) similar products left by anonymous buyers or users, even though we do not know whether we necessarily share similarities with such reviewers.

Recommender systems use various data sources that link **users**, **items**, and **ratings** to infer potential **interest** in a product or service. What makes a **good** recommendation? Recommended items should be

- **relevant** and appealing to the user;
- **novel** and unknown to (or not have been considered by) the user prior to the suggestions;
- **serendipitous** and occasionally surprise the user, and
- **diverse** to minimise the chance of making only irrelevant suggestions.

There are various types of recommender systems:

- **user-based collaborative filtering** identify similar users based on the preference of a given user and recommend new items using ratings given by the similar users (see Figure 36a), whereas **item-based collaborative filtering** identify similar items based on item preferences and recommend top-rated items among non-rated items (see Figure 36b) – both of those methods require very little information on the user and provide fairly high recommendation satisfaction (accuracy), but they also tend to require many pairs of user-item ratings and may suffer from popularity bias, as well as providing black box recommendations;
- **content-based** systems recommend items based on similar items a user has liked in the past – in this case, there is no issue with popularity bias and the recommendations are transparent (white box), but the engines requires a fair amount of information about the content of the items (metadata, description, topics, etc.);
- in **model-based** systems, a non-rated item for a user can be seen as a missing value, to be imputed using various probabilistic models, supervised and unsupervised models, matrix factorization (alternating least square, etc.), single value decomposition, etc. – these systems are substantially faster than collaborative filtering or content-based engines, taking full advantage of distributed data science models, but the choice of the model can be somewhat arbitrary and produce completely different recommendations.

How good are the recommendation systems that you have encountered? A recent twitter joke by Justin Shanes made the round recently, which we found to be quite *à propos*:

> Amazon thinks my recent humidifier purchase was merely the inaugural move in a newfound hobby of humidifier collecting.

Data Streams When data is arriving **continuously** (or in small chunks), it might be useful to incorporate the new data into the existing models without having to re-compute all important statistics from scratch. Often this is done in **batches**:

1. data is collected until a threshold quantity is met, and
2. the batch of new data is then combined with the existing model to create a new model.

Data streams tasks include:

- **detecting a change** in the underlying data generating processes, which can be useful because keeping an outdated model on board can be costly (this problem is usually quite challenging);
- **maintaining histograms and simple statistics**, which is tricky because the optimal number of bins, the length of intervals, etc. may change with the addition of new data – maintaining statistics may require the phasing out of old data;
- **detecting novelty** – identifying new concepts that were not present in the training phase or identifying new profiles that are not explained by the current model;
- **frequent pattern mining** is interesting, because itemsets which were not frequent in the past may become frequent with the addition of new data, and *vice-versa*;
- **classification** is also problematic as it can be costly to rebuild global models (such as SVM, and naïve Bayes), especially since concept drift may affect only some region of the instance space – thankfully, some models such as decision trees can be modified locally instead of globally;
- **clustering** is perhaps one of the easier tasks to stream – at least for some of the models.

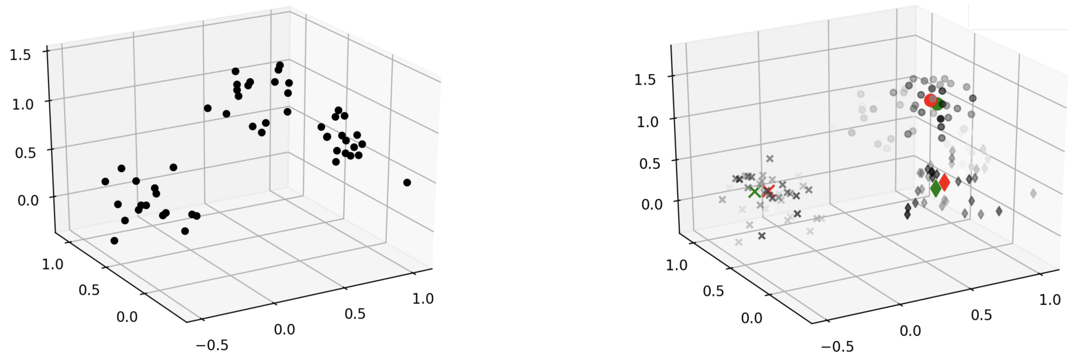
In **streaming k—means**, we let, at time t,

- $c_i(t)$ be the centroid of the i^{th} cluster;
- $n_i(t)$ be the number of points in the dataset that are assigned to the i^{th} cluster;
- $m_i(t)$ be the number of new points that are closer to $c_i(t)$ than to any other centroid, and
- $x_i(t)$ be the centroid of the new points that are closer to $c_i(t)$ than to any other centroid.

We can update c_i and n_i at time $t+1$ according to

$$\begin{aligned} c_i(t+1) &= \frac{\alpha c_i(t)n_i(t)+x_i(t)m_i(t)}{\alpha n_i(t)+m_i(t)} \\ n_i(t+1) &= n_i(t)+m_i(t) \end{aligned}$$

where α is a decay factor weighing old data relative to new data. The **heavy lifting**, algorithmically speaking, is in finding the closest centroids for the new points and the average of the new points in each of the k clusters.



(a) 50 original observations **(b)** 80 additional observations

Figure 37: Illustration of streaming k—means; the three clusters centroids are indicated by filled and coloured $\times$, $\circ$, and $\Diamond$.

As an example, we start by generating 50 points in space, normally distributed around three points; the standard deviation is chosen so that the data forms three natural clusters (see Figure 37a. Over time, 80 new points are added, but centered around slightly different locations, so that we should expect to see some movement in the three cluster centroids (see Figure 37b)

Others Let us mention in closing three other families of data science tasks which it would be useful for consultants to be familiar with at least:

- **social network analysis**, in which social relationships are analysed in network terms, with applications in customer churn prediction and prevention (understanding group behaviour), marketing campaigns (identifying influencers), risk and fraud detection (detecting money laundering or credit card fraud), etc.;
- **multi-media mining**, the application of data mining techniques to non-numerical and non-textual data, images, videos, and audio tracks, with use cases in behaviour analysis, intelligent compression, image classification, image segmentation, image annotation, video classification, audio classification, etc., and
- **outlier and anomaly detection**.

1.6.7 Final Thoughts

In many ways, the data science and machine learning story is just starting; in others, it has already peaked. Promises of disruption, artificial intelligence, self-driving cars, machine translation and complete language understanding, detection and prevention of ecosystem disturbances or astronomical catastrophes, and automated data analysis, among others, dance in the eyes of investors, public servants, politicians, and academics alike. The future seems extremely bright.

And it might very well be the case that all of these will come to pass. But remember that not every situation calls for data science methods, and that there are some skeletons in the closet, not least of which being that machine learning is not a rigorous science, for the most part, and that data science models tend to have a distinct black box feel to them.

In truth, predictive models are valid if they make good predictions – perhaps that is all that can be asked of our models. But there is no reason as of yet to abandon the classical statistical models that have served us well in the past – data science methods are not intended to supplant statistical methods, rather, they are meant to supplement them. Successful quantitative consultants will find a way to take advantage of this situation.

1.6.8 General Data Science Case Studies

We present four **case study write-ups** of various data science projects (the last write-up properly belongs to Section **??**). Such write-ups have two main functions: they show prospective clients

1. that other groups have also benefited from quantitative methods, and
2. that consultants have a good understanding of those methods.

How much technical details to provide in the write-ups can be tuned to each client, but remember that **visuals speak more than equations** with the average client.

1.6.9 Case Study: Analysis of CIS Data

The *Canadian Incidence Study of Reported Child Abuse and Neglect* (CIS) is a national surveillance program dedicated to the health of children in Canada. It examines the incidence of reported child maltreatment and the characteristics of the children and families investigated by Canadian child welfare sites from all thirteen sub-national jurisdictions.

Some minor methodological changes were introduced over the years: increased sample size every cycle, differences in jurisdictional oversampling strategies, increased sample from First Nations agencies, etc.

A crucial modification, however is that, prior to the third cycle of the CIS (2008), prevalence estimates were provided for five types of child maltreatment: physical abuse, exposure to family violence, neglect, sexual abuse and emotional abuse. Child protection workers provided data as to their findings: whether any type of abuse was

> **Substantiated** (evidence indicated it happened); **Unfounded** (did not happen) or, in some cases **Suspected** (may have happened but cannot be definitively proven).

From 2008 onwards, a new category was added: risk of maltreatment. Child protection workers could then indicate that the maltreatment allegation they were investigating was substantiated for risk. Possible outcomes for risk of future maltreatment are recorded as

> **Significant**; **Not Significant**, or **Unknown**.

The changes were brought about for two reasons: on the first hand, there could be a concern that a maltreatment incident may have occurred; on the other hand, even if such an incident was not substantiated or suspected, there may be significant risk of future maltreatment. Of all child maltreatment investigations in 2008, 74% focused on concern of abuse or neglect and 26% on risk of future maltreatment; 36% were substantiated, 8% had insufficient evidence to substantiate yet remained suspected, and 30% were unfounded; 5% of showed a significant risk, 17% were classified as not posing a significant risk, and the risk was unknown in the remaining 4% of cases.

Prior to 2008, the CIS variables did not include the type of investigation for specific cases. The *Public Health Agency of Canada* (PHAC) is interested in determining what that classification might have been for the 2003 dataset. **How would you approach this problem?**

Data Analysis Case Studies

Patrick Boily[1,2,3]

Abstract

In Data Science and Data Analysis, as in most technical or quantitative fields of inquiry, there is an important distinction between understanding the theoretical underpinnings of the methods and knowing how and when to best apply them to practical situations.

The successful transition from clean pedagogical toy examples to messy situations can be complicated by a misunderstanding of what a useful and insightful solution looks like in a non-academic context.

In this report, we provide examples of data analysis and quantitative methods applied to "real-life" problems. We emphasize qualitative aspects of the projects as well as significant results and conclusions, rather than explain the algorithms or focus on theoretical matters.

Keywords

case studies, data science, machine learning, data analysis, statistical analysis, quantitative methods

[1]Centre for Quantitative Analysis and Decision Support, Carleton University, Ottawa

[2]Department of Mathematics and Statistics, University of Ottawa, Ottawa

[3]Idlewyld Analytics and Consulting Services, Wakefield, Canada

Email: patrick.boily@carleton.ca

Contents

Data Analysis Case Studies

The case studies were selected primarily to showcase a wide breadth of analytical methods, and are not meant to represent a complete picture of the data analysis landscape. In some instances, the results were published in peer-reviewed journals or presented at conferences. In each case, we provide the:

- project title and citation references;
- author(s) and publication date;
- sponsors (if there were any), and
- methods that were used.

Depending on the case study, some of the following items are also provided:

- objective;
- methodology;
- advantages or disadvantages of specific methods;
- procedures and results;
- evaluation and validation;
- project summary, and
- challenges and pitfalls, etc.

Since the various sponsoring organizations have not always allowed the dissemination of specific results (for a variety of reasons), we have opted to follow their lead; when such results are available, the interested reader can consult them in the appropriate publications or presentations.

1. Classification: Tax Audits

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.

Title Data Mining Based Tax Audit Selection: A Case Study of a Pilot Project at the Minnesota Department of Revenue [1]

Authors Kuo-Wei Hsu, Nishith Pathak, Jaideep Srivastava, Greg Tschida, Eric Bjorklund

Date 2015

Sponsor Minnesota Department of Revenue (DOR)

Methods classification, data mining

Objective The U.S. Internal Revenue Service (IRS) estimated that there were huge gaps between revenue owed and revenue collected for 2001 and for 2006. The project's goals were to increase efficiency in the audit selection process and reduce the gap between revenue owed and revenue collected.

Methodology

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 it was realized that the size of the business being audited had an effect of the model accuracy: the task was split in two parts to model large businesses and smaller business separately.
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 1).

Strengths and Limitations of Algorithms

- The Naïve Bayes classification scheme assumes independence of the features, which rarely occurs in real-world situations. Furthermore, this approach tends to introduce bias to classification schemes. In spite of this, classification models built using Naïve Bayes have a successfully track record.
- MultiBoosting is an ensemble technique that uses forms a committees (i.e. groups of classification models) and group wisdom to make a prediction; unlike other ensemble techniques, it also uses a committee of sub-committee 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.

Procedures Classification schemes need a response variable for prediction: audits which yielded more than $500 per year in revenues during the audit period were *Good*; the others were *Bad*. The various models were tested and evaluated by comparing the performances of the manual audit (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. Department of Revenue (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 inadequate 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 [1]).

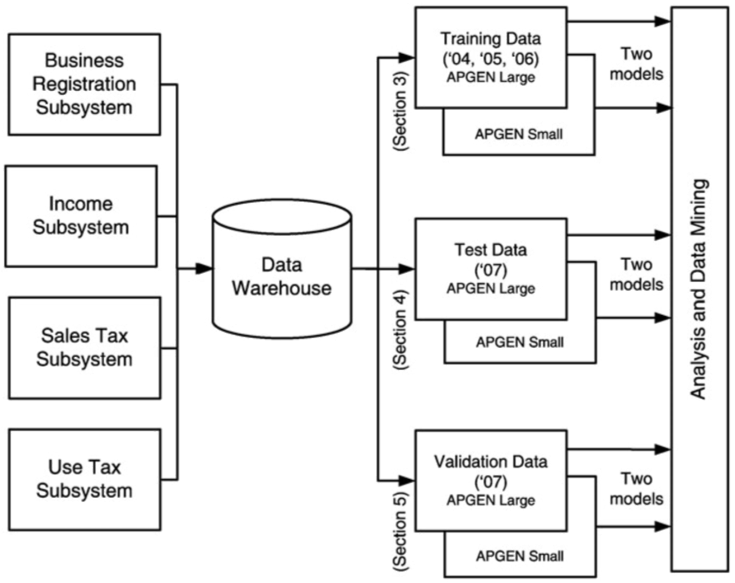


Figure 1. Data sources for APGEN mining [1]. Note the 6 final sets which feed the Data Analysis component.

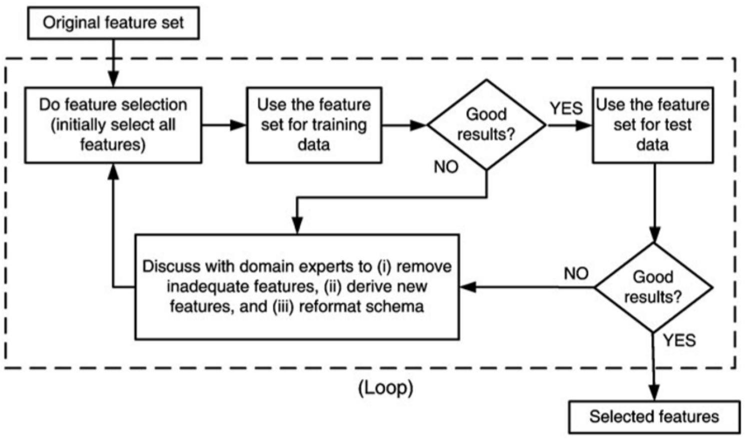

Original feature set
Do feature selection (initially select all features)
Use the feature set for training data
Good results?
YES
NO
Use the feature set for test data
Discuss with domain experts to (i) remove inadequate features, (ii) derive new features, and (iii) reformat schema
NO
Good results?
YES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

Figure 2. The feature selection process [1]. Note the involvement of domain experts.

4. **Selecting separate modeling features** for the APGEN Large and Small training sets. The feature selection process is shown in Figure 2.
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:

$$\text{Efficiency} = \text{ROI} = \frac{\text{Total revenue generated}}{\text{Total collection cost}} \quad (1)$$

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 would be required to reach 495 successful audits of APGEN Large (2007), and 295 would be required to reach 99 successful audits for APGEN Small (2007), as can be seen in Figure 3.

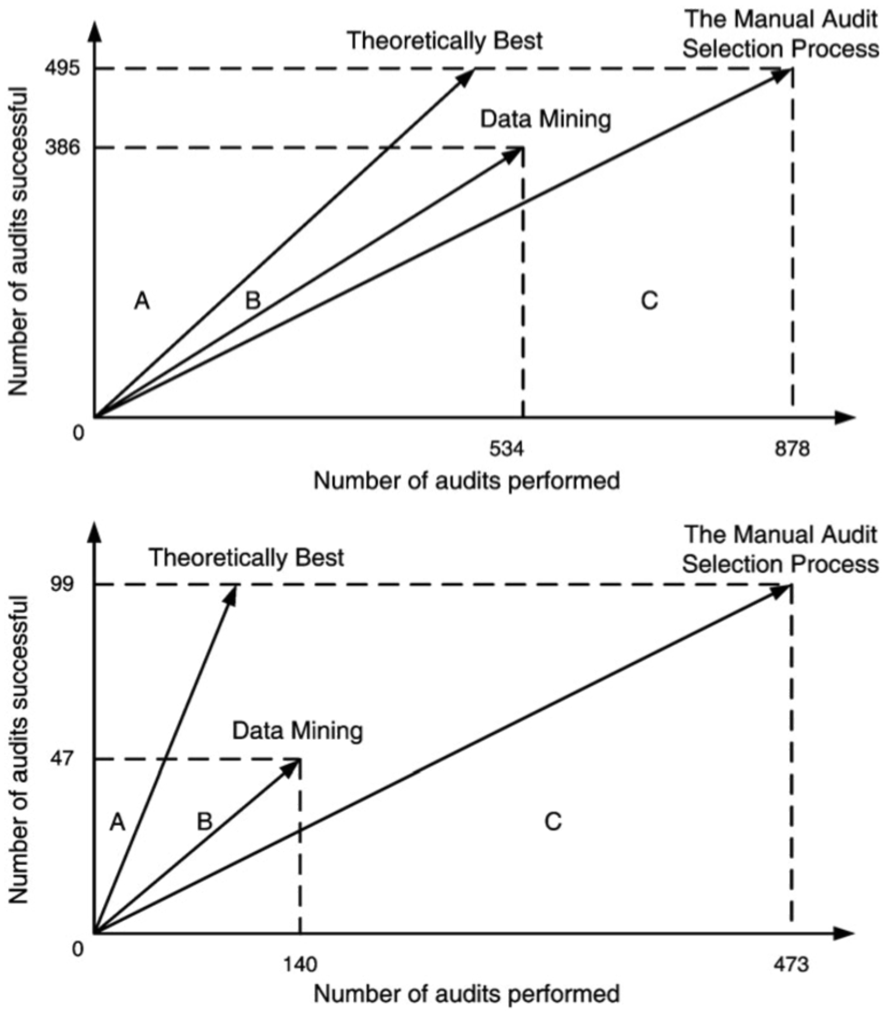


Figure 3. Audit resource deployment efficiency [1]. Top: APGEN Large (2007). Bottom: APGEN Small (2007). In both cases, the Data Mining approach was more efficient (the slope of the Data Mining vector is "closer" to the Theoretical Best vector than is the Manual Audit vector).

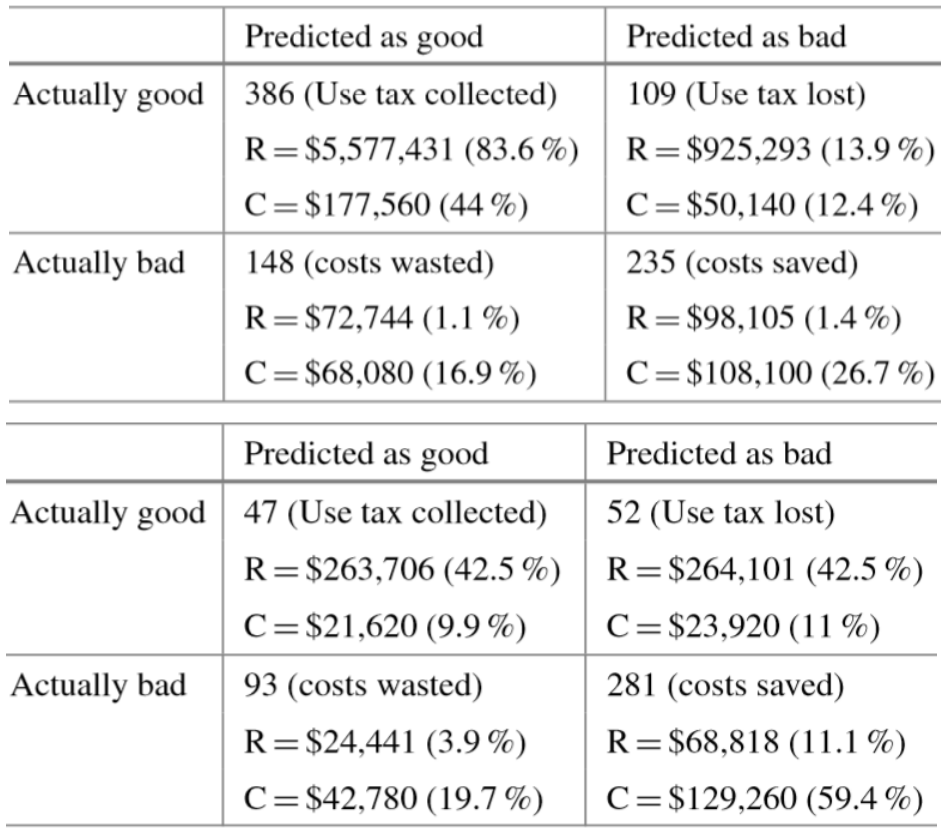
Table 1 presents the confusion matrices for the classification model on both the APGEN Large (2007) and APGEN Small (2007) data. Columns and rows represent predicted and actual results, respectively. The revenue R and collection cost C entries can be read as follows: the 47 successful audits which were 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 conpleted, 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 specific 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* [2]: 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 [3].

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!



	Predicted as good	Predicted as bad
Actually good	386 (Use tax collected)	109 (Use tax lost)
	R = $5,577,431 (83.6 %)	R = $925,293 (13.9 %)
	C = $177,560 (44 %)	C = $50,140 (12.4 %)
Actually bad	148 (costs wasted)	235 (costs saved)
	R = $72,744 (1.1 %)	R = $98,105 (1.4 %)
	C = $68,080 (16.9 %)	C = $108,100 (26.7 %)

	Predicted as good	Predicted as bad
Actually good	47 (Use tax collected)	52 (Use tax lost)
	R = $263,706 (42.5 %)	R = $264,101 (42.5 %)
	C = $21,620 (9.9 %)	C = $23,920 (11 %)
Actually bad	93 (costs wasted)	281 (costs saved)
	R = $24,441 (3.9 %)	R = $68,818 (11.1 %)
	C = $42,780 (19.7 %)	C = $129,260 (59.4 %)

Table 1. Confusion matrices for audit evaluation [1]. Top: APGEN Large (2007). Bottom: APGEN Small (2007). *R* stands for revenues, *C* for collection costs.

References

[1] Hsu, K.W., Pathak, N., Srivastava, J., Tschida, G., Bjorklund, E. [2015] *Data Mining Based Tax Audit Selection: A Case Study of a Pilot Project at the Minnesota Department of Revenue*, in: Abou-Nasr, M., Lessmann, S., Stahlbock, R., Weiss, G. (eds) *Real World Data Mining Applications*, Annals of Information Systems, v.17, Springer. doi:10.1007/978-3-319-07812-0_12

[2] Wolpert, D.H. [1996] The Lack of a priori distinctions between learning algorithms, *Neural Computation*, v.8, n.7, pp.1341-1390, MIT Press. doi:10.1162/neco.1996.8.7.1341

[3] Wolpert, D.H., Macready, W.G. [2005] Coevolutionary free lunches, *IEEE Transactions on Evolutionary Computation*, v.9, n.6, pp.721-735, IEEE Press. doi:10.1109/TEVC.2005.856205

2. Sentiment Analysis: BOTUS and Trump & Dump

In 2013, the BBC reported on various ways in which social media giant Twitter was changing the world, detailing specific instances in the fields of business, politics, journalism, sports, entertainment, activism, arts, and law [9].

It is not always clear what influence Twitter users have, if any, on world events or business and cultural trends; it was once thought (perhaps without appropriate evidence) that entertainers, athletes, and celebrities, that is to say, users with extremely high followers/following ratios, wielded more "influence" on the platform than world leaders [1].

Certainly, such users continue to be among the most popular – as of September 13, 2017, Twitter's 40 most-followed accounts tend to belong to entertainers, celebrities, and athletes, with a few exceptions [15].

One account has recently bridged the gap between celebrity and politics in an explosive manner: @realDonaldTrump, which belongs to the 45th President of the United States of America, has maintained a very strong presence on Twitter.

As of September 13, 2017, the account had 38,205,766 followers, and it was the 26th most-followed account on the planet, producing 35,755 tweets since it was activated in March 2009 [15].

Titles BOTUS [5], Trump & Dump Bot [16]

Authors Tradeworx (BOTUS), T3 (Trump & Dump)

Date 2017

Sponsor NPR's podcast *Planet Money* (BOTUS)

Methods sentiment analysis, social media monitoring, AI, real-time analysis, simulations

Objective There is some evidence to suggest that tweets from the 45th POTUS may have an effect on the stock market [8, 10]. Can sentiment analysis and AI be used to take real-time advantage of the tweets' unpredictable nature? Let's take a look at bots built for that purpose by NPR's *Planet Money* and by T3 (an Austin advertising agency).

Methodology Tradeworx followed these steps:

1. *Data collection*: tweets from @realDonaldTrump are collected for analysis.
2. *Sentiment analysis of tweets*: each tweet is given a sentiment score on the positive/negative axis.
3. *Validation*: the sentiment analysis scoring must be validated by observers: are human-identified positive or negative tweets correctly identified as such by BOTUS?
4. *Identification of the company in a tweet*: is the tweet even about a company? If so, which one?
5. *Determining the trading universe*: are there companies that should be excluded from the bot's trading algorithms?
6. *Classifying tweets as "applicable" or "unapplicable"*: is a tweet's sentiment strong enough for BOTUS to engage the trading strategy?
7. *Determining a trading strategy*: how soon after a flagged tweet does BOTUS buy a company's stock, and how long does it hold it for?
8. *Testing the trading strategy on past data*: how would BOTUS have fared from the U.S. Presidential Election to April 2017? What are BOTUS' limitations?

T3's Trump and Dump uses a similar process (see Figure 4).

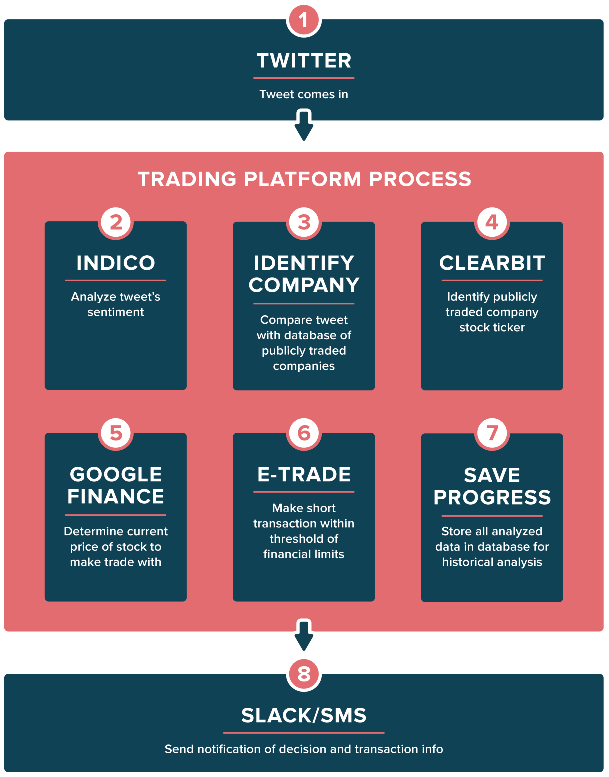

1
TWITTER
Tweet comes in
TRADING PLATFORM PROCESS
2
INDICO
Analyze tweet's sentiment
3
IDENTIFY COMPANY
Compare tweet with database of publicly traded companies
4
CLEARBIT
Identify publicly traded company stock ticker
5
GOOGLE FINANCE
Determine current price of stock to make trade with
6
E-TRADE
Make short transaction within threshold of financial limits
7
SAVE PROGRESS
Store all analyzed data in database for historical analysis
8
SLACK/SMS
Send notification of decision and transaction info

Figure 4. T3's Trump and Dump process [16].

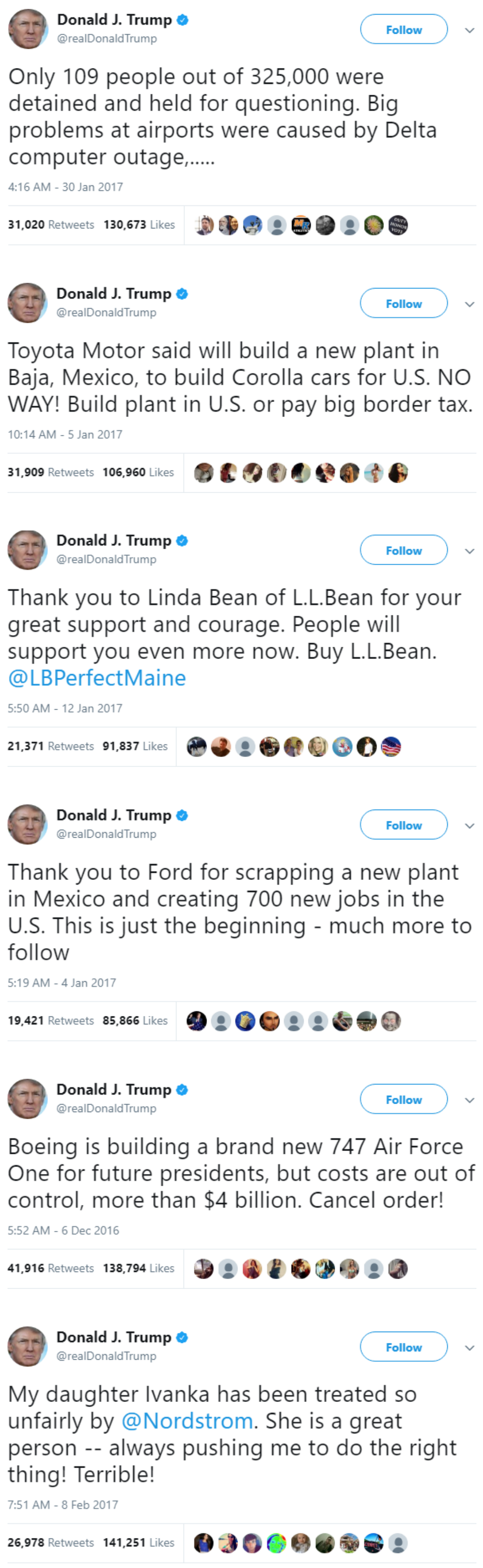
Data The data consists of:

- tweets by @realDonaldTrump (from around Election Day 2016 through the end of March 2017 for BOTUS; no details are given for T3) (see Figure 5 for sample);
- a database of publicly traded companies, such as can be found at [3, 14, 17], although which of these were used, if any, is not specified (no explicit mention is made for BOTUS), and
- stock market data for real-time pricing (Google Finance for T3) and backcasting simulation (for BOTUS, source unknown).

It is not publicly known whether the bots are upgrading their algorithms by including new data as time passes.

Strengths and Limitations of Algorithms and Procedure

- In sentiment analysis, an algorithm analyzes documenta in an attempt to identify the attitude they express or the emotional response they seek. It presents numerous challenges, mostly related to the richness and flexibility of human languages and their syntax variations, the context-dependent meaning of words and lexemes, the use of sarcasm and figures of speech, and the lack of perfect inter-rater reliability among humans [12]. As it happens, @realDonaldTrump is not much of an ironic tweeter – when he uses "sad", "bad" and "great", he usually means "sad", "bad" and "great" in their most general sense. This greatly simplifies the analysis.
- The bots have to learn to recognize whether a tweet is directed at a publicly traded company or not. In certain cases, the ambiguity can be resolved relatively easily with an appropriate training set (Apple the company vs. apple the food-item, say), but no easy solutions were found in others (Tiffany the company vs. Tiffany the daughter, for example). Rather than have humans step in and instruct BOTUS when it faces uncertainty (which would go against the purpose of the exercise), a decision was made to exclude these cases from the trading universe. What T3's bot does is not known.
- Once the bot knows how to rate @realDonaldTrump's tweets and to identify when he tweets about publicly-traded companies, the next question is to determine what the trading strategy should be. If the tweet's sentiment is negative enough T3 shorts the company's



Donald J. Trump @realDonaldTrump Follow

Only 109 people out of 325,000 were detained and held for questioning. Big problems at airports were caused by Delta computer outage,.....

4:16 AM - 30 Jan 2017

31,020 Retweets 130,673 Likes

Donald J. Trump @realDonaldTrump Follow

Toyota Motor said will build a new plant in Baja, Mexico, to build Corolla cars for U.S. NO WAY! Build plant in U.S. or pay big border tax.

10:14 AM - 5 Jan 2017

31,909 Retweets 106,960 Likes

Donald J. Trump @realDonaldTrump Follow

Thank you to Linda Bean of L.L.Bean for your great support and courage. People will support you even more now. Buy L.L.Bean. @LBPerfectMaine

5:50 AM - 12 Jan 2017

21,371 Retweets 91,837 Likes

Donald J. Trump @realDonaldTrump Follow

Thank you to Ford for scrapping a new plant in Mexico and creating 700 new jobs in the U.S. This is just the beginning - much more to follow

5:19 AM - 4 Jan 2017

19,421 Retweets 85,866 Likes

Donald J. Trump @realDonaldTrump Follow

Boeing is building a brand new 747 Air Force One for future presidents, but costs are out of control, more than $4 billion. Cancel order!

5:52 AM - 6 Dec 2016

41,916 Retweets 138,794 Likes

Donald J. Trump @realDonaldTrump Follow

My daughter Ivanka has been treated so unfairly by @Nordstrom. She is a great person -- always pushing me to do the right thing! Terrible!

7:51 AM - 8 Feb 2017

26,978 Retweets 141,251 Likes

Figure 5. Examples of @realDonaldTrump tweets involving Delta, Toyota Motor, L.L.Bean, Ford, Boeing, Nordstrom.

stock.[1] Of course, this requires first purchasing the stock (so that it can be shorted). Planet Money's decision was similar: buy once the tweet is flagged, and sell right away... but what does "right away" mean in this context? There is a risk involved: if the stock goes back up before BOTUS has had a chance to purchase the low-priced stock, it will lose money. To answer that question, Tradeworx simulated the stock market over the last few months, introducing the tweets, and trying out different trading strategies. It turns out that, in this specific analysis, "right away" can be taken to be 30 minutes after the tweet.

Results, Evaluation and Validation For a trading bot, the validation is in the pudding, as they say – do they make money? T3's president says that their bot is profitable (they donate the proceeds to the ASPCA) [16]: for instance, they netted a return of 4.47% on @realDonaldTrump's Delta tweet (see Figure 5); however, he declined to provide specific numbers (and made vague statements about providing monthly reports, which I have not been able to locate) [11].

The BOTUS process was more transparent, and we can point to Planet Money's transcript for a discussion on sentiment analysis validation (comparing BOTUS's sentiment rankings with those provided by human observers, or running multiple simulations to determine the best trading scenario) [5] – but it suffers from a serious impediment: as of roughly 4 months after going online, it still had not made a single trade [4]!

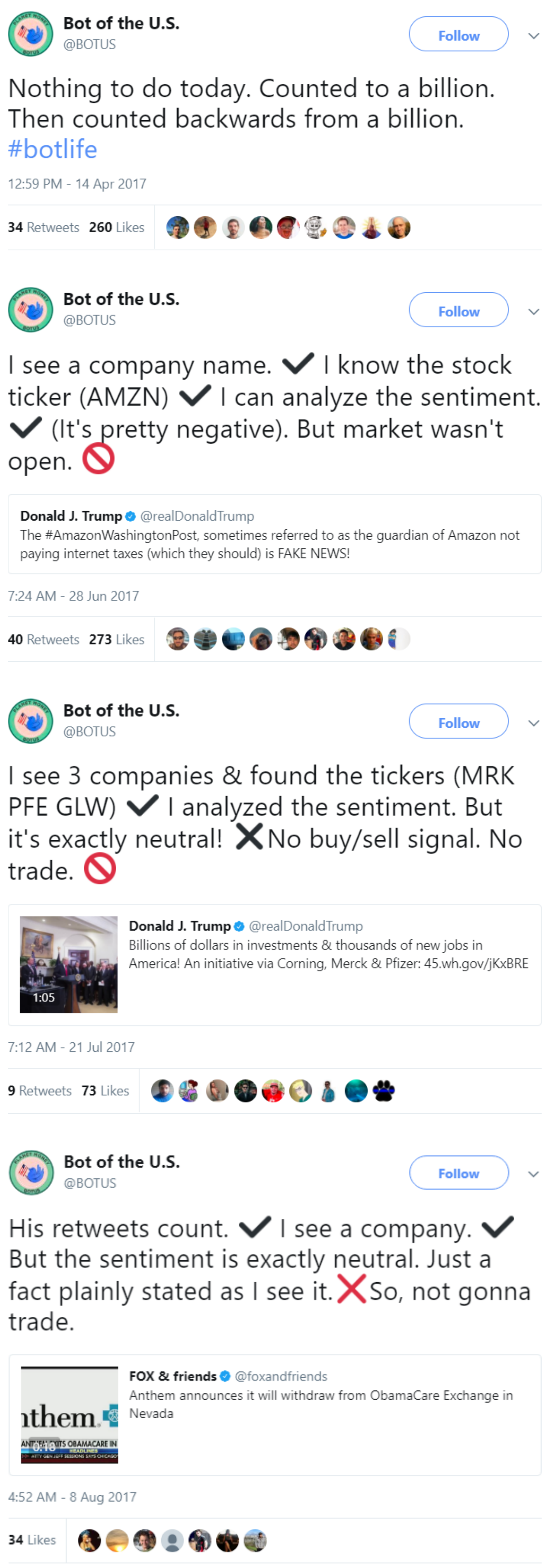
The reasons are varied (see Figures 6 and 7), but the most important setback was that @realDonaldTrump had not made a single valid tweet about a public company whose stock BOTUS could trade during the stock market business hours. Undeterred, Planet Money relaxed its trading strategy: if @realDonaldTrump tweets during off-hours, BOTUS will short the stock at the market's opening bell.

This is a risky approach, and so far it has not proven very effective: a single trade of Facebook's trade, on August 23rd, which resulted in a loss of 0.30$ (see Figure 7).

Take-Aways As a text analysis and scenario analysis project, both BOTUS and Trump & Dump are successful – they present well-executed sentiment analyses, and a simulation process that finds an optimal trading strategy. As predictive tools, they are sub-par (as far as we can tell), but for reasons that (seem to) have little to do with data analysis *per se*.

Unfortunately, this is not an atypical feature of descriptive data analysis: we can explain what has happened (or what is happening), but the modeling assumptions are not always applicable to the predictive domain.

[1] It sells the stock when the price is high, that is, *before* the tweet has had the chance to bring the stock down, and it repurchases it once the price has been lowered by the tweet, but before the stock has had the chance to recover.



Bot of the U.S. @BOTUS Follow

Nothing to do today. Counted to a billion. Then counted backwards from a billion. #botlife

12:59 PM - 14 Apr 2017

34 Retweets 260 Likes

Bot of the U.S. @BOTUS Follow

I see a company name. ✔ I know the stock ticker (AMZN) ✔ I can analyze the sentiment. ✔ (It's pretty negative). But market wasn't open. 🚫

Donald J. Trump @realDonaldTrump
The #AmazonWashingtonPost, sometimes referred to as the guardian of Amazon not paying internet taxes (which they should) is FAKE NEWS!

7:24 AM - 28 Jun 2017

40 Retweets 273 Likes

Bot of the U.S. @BOTUS Follow

I see 3 companies & found the tickers (MRK PFE GLW) ✔ I analyzed the sentiment. But it's exactly neutral! ✖No buy/sell signal. No trade. 🚫

Donald J. Trump @realDonaldTrump
Billions of dollars in investments & thousands of new jobs in America! An initiative via Corning, Merck & Pfizer: 45.wh.gov/jKxBRE

1:05

7:12 AM - 21 Jul 2017

9 Retweets 73 Likes

Bot of the U.S. @BOTUS Follow

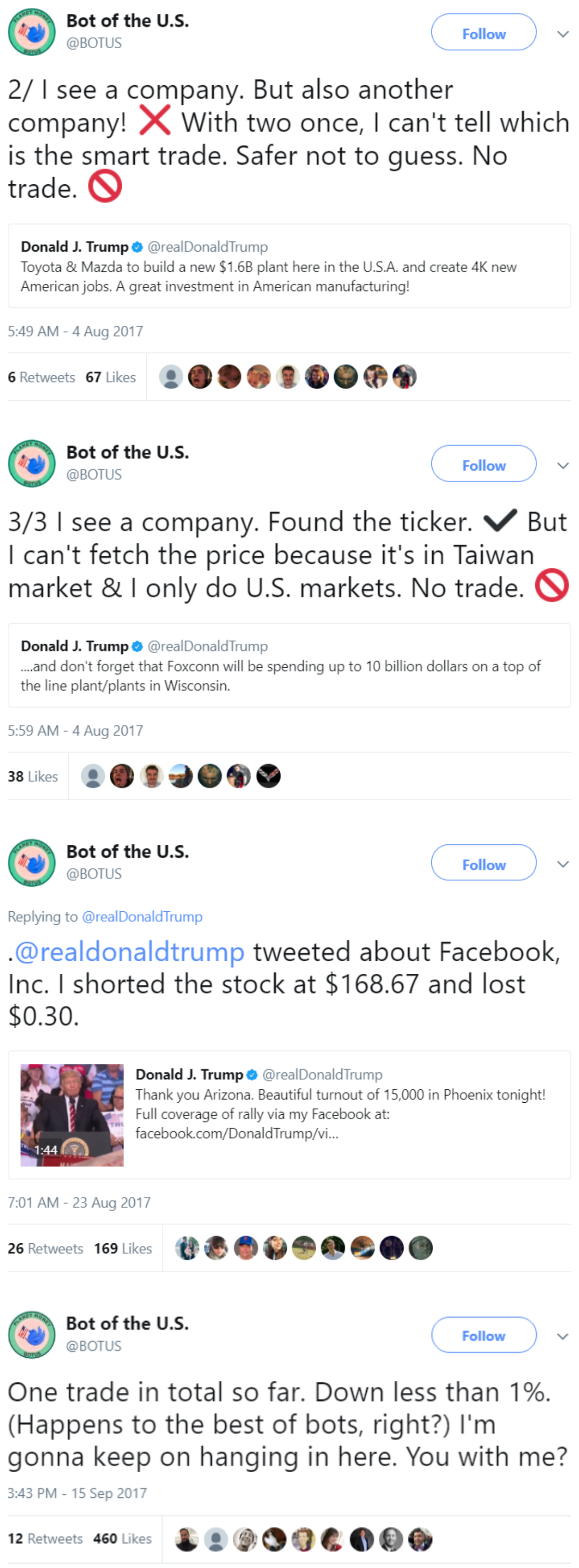
His retweets count. ✔ I see a company. ✔ But the sentiment is exactly neutral. Just a fact plainly stated as I see it.❌So, not gonna trade.

FOX & friends @foxandfriends
Anthem announces it will withdraw from ObamaCare Exchange in Nevada

4:52 AM - 8 Aug 2017

34 Likes

Figure 6. BOTUS reporting on its trades.



Bot of the U.S. @BOTUS Follow

2/ I see a company. But also another company! ❌ With two once, I can't tell which is the smart trade. Safer not to guess. No trade. 🚫

Donald J. Trump @realDonaldTrump
Toyota & Mazda to build a new $1.6B plant here in the U.S.A. and create 4K new American jobs. A great investment in American manufacturing!

5:49 AM - 4 Aug 2017

6 Retweets 67 Likes

Bot of the U.S. @BOTUS Follow

3/3 I see a company. Found the ticker. ✔ But I can't fetch the price because it's in Taiwan market & I only do U.S. markets. No trade. 🚫

Donald J. Trump @realDonaldTrump
....and don't forget that Foxconn will be spending up to 10 billion dollars on a top of the line plant/plants in Wisconsin.

5:59 AM - 4 Aug 2017

38 Likes

Bot of the U.S. @BOTUS Follow

Replying to @realDonaldTrump

.@realdonaldtrump tweeted about Facebook, Inc. I shorted the stock at $168.67 and lost $0.30.

Donald J. Trump @realDonaldTrump
Thank you Arizona. Beautiful turnout of 15,000 in Phoenix tonight! Full coverage of rally via my Facebook at: facebook.com/DonaldTrump/vi...

1:44

7:01 AM - 23 Aug 2017

26 Retweets 169 Likes

Bot of the U.S. @BOTUS Follow

One trade in total so far. Down less than 1%. (Happens to the best of bots, right?) I'm gonna keep on hanging in here. You with me?

3:43 PM - 15 Sep 2017

12 Retweets 460 Likes

Figure 7. BOTUS reporting on its trades.

References

[1] Arthur, C. [2010], The big bang visualisation of the top 140 Twitter influencers, retrived from The Guardian.com on September 13, 2017.

[2] Basu, T. [2017], NPR's Fascinating Plan to Use A.I. on Trump's Tweets, retrieved from inverse.com on September 12, 2017.

[3] Company List: NASDAQ, NYSE, & AMEX Companies, retrieved on NASDAQ.com on September 15, 2017.

[4] Dieker, N. [2017], Planet Money's BOTUS Bot Has Yet to Make a Single Stock Trade, retrieved from Medium.com's The Billfold on September 12, 2017.

[5] Goldmark, A. [2017], Episode 763: BOTUS, Planet Money podcast, retrieved from NPR.org's Planet Money on September 12, 2017.

[6] Green, S. [2017], Trump Tweets: Separate Positive and Negative Tweets, retrieved from Green Analytics on September 15, 2017.

[7] Greenstone, S. [2017], When Trump Tweets, This Bot Makes Money, retrieved from NPR.org on September 12, 2017.

[8] Ingram, M. [2017], Here's What a Trump Tweet Does to a Company's Share Price, retrieved from Fortune.com on September 15, 2017.

[9] Lee, D. [2013], How Twitter Changed the World, Hashtag-by-Hastag, retrieved from BBC.com on September 13, 2017.

[10] McNaney, B. [2017], A Negative Trump Tweet About Your Company Is An Eye Opener, Not A Crisis, retrieved from The Buzz Bin on September 22, 2017.

[11] Mettler, K. [2017], 'Trump and Dump': When POTUS tweets and stocks fall, this animal charity benefits, retrieved from the Washington Post on September 19, 2017.

[12] Ogneva, M. [2010], How Companies Can Use Sentiment Analysis to Improve Their Business, retrieved from Mashable on September 17, 2017.

[13] Popper, N. [2017] A Little Birdie Told Me: Playing the Market on Trump Tweets, retrieved from The New York Times on September 22, 2017.

[14] List of Publicly Traded Companies, retrieved on InvestorGuide.com on September 15, 2017.

[15] Top 100 Most Followed Twitter Accounts, retrieved from Twitter Counter on September 13, 2017.

[16] Trump & Dump Bot: Analyzes Tweets, Short Stocks, retrieved from T3 on September 14, 2017.

[17] The World's Biggest Public Companies List - Forbes 2000, retrieved on Forbes.com on September 15, 2017.

3. Clustering: The Livehoods Project

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?

Title The Livehoods Project: Utilizing Social Media to Understand the Dynamics of a City [2]

Authors Justin Cranshaw, Raz Schwartz, Jason I. Hong, Norman Sadeh

Date 2012

Sponsors National Science Foundation, Carnegie Mellon's CyLab, Army Research Office, Alfred P. Sloan Foundation, CMU/Portugal ICTI, with additional support from Google, Nokia, and Pitney Bowes.

Methods spectral clustering, social dynamics

Objective The project 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 a spectral clustering model (the method will be described later) to discover the distinct geographic areas of the city based on its inhabitants' collective movement patterns. Semi-structured interviews are used to explore, label and validate the resulting clusters, as well as the urban dynamics that shape them.

Livehood clusters are built and defined 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 approximately 11 million **Foursquare** (a recommendation site for venues based on users' experiences) check-ins from the dataset of Chen et al. [1] 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, livehood clusters from Pittsburgh, Pennsylvania, are examined using 42,787 check-ins of 3840 users at 5349 venues.

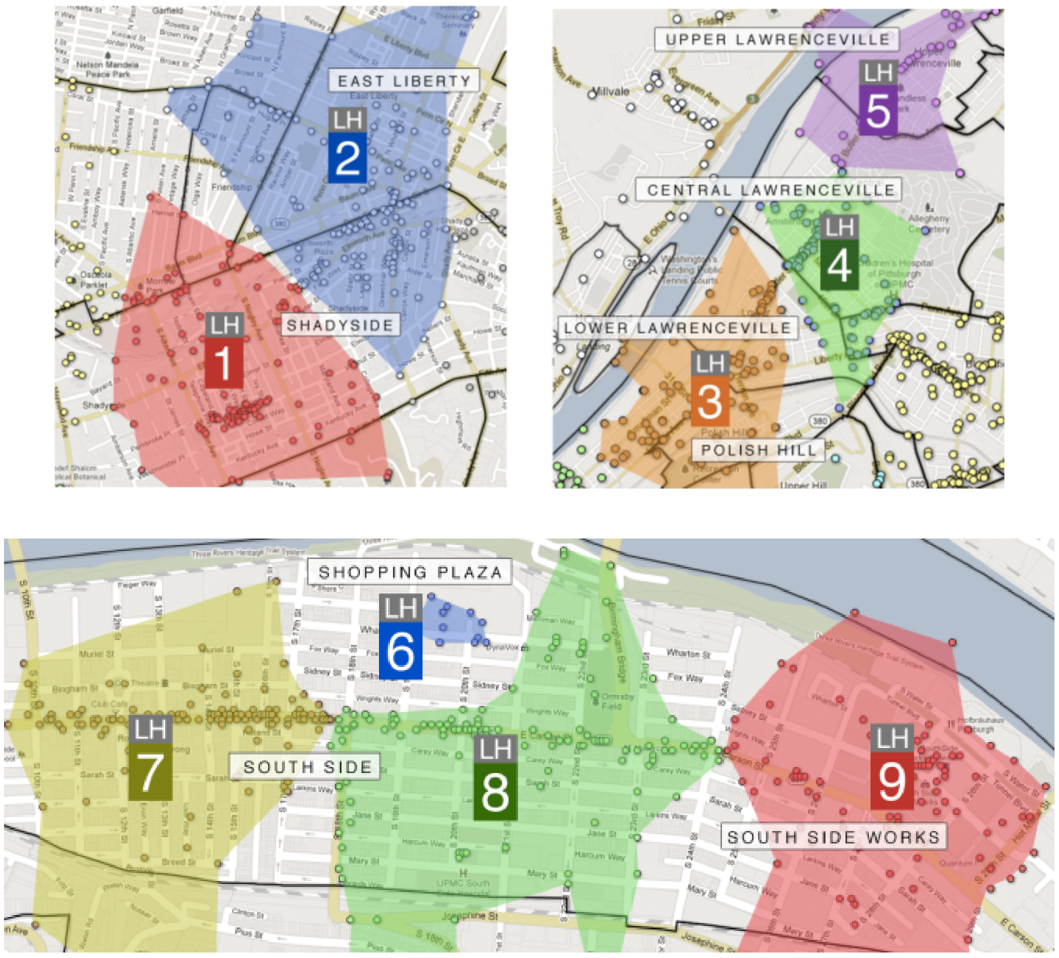


Figure 8. Some livehoods in metropolitan Pittsburgh, PA: in Shadyside/East Liberty, Lawrenceville/Polish Hill, and South Side. Municipal borders are shown in black.

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 or hiding minority behaviours.
- The data are based on a limited sample of check-ins shared on Twitter and are therefore biased towards the types of places 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.

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 derive 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 $G(A)$ be the graph obtained from the A by linking each venue to its nearest m neighbours. Spectral clustering is implemented by the following algorithm:

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

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

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 will induce a clustering $A_1, \ldots, A_k$ defined by $A_i = \{j | y_j \in C_i\}$.
7. For each A_i, let $G(A_i)$ be the subgraph of $G(A)$ 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 $m = 10$, $k_{\min} = 30$, $k_{\max} = 45$, and $\tau = 0.4$. The results for three areas of the city are shown in Figure 8. In total, 9 livehoods have been identified and validated by 27 Pittsburgh residents (see Figure 8; the original report has more information on the interview 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 interview 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 k—means is not the sole clustering algorithm in applications!

References

[1] Chen, Z., Caverlee, J., Lee, K., Su, D.Z. [2011], *Exploring millions of footprints in location sharing services*, ICWSM.

[2] Cranshaw, J., Schwartz, R., Hong, J.I., Sadeh, N. [2012], *The Livehoods Project: Utilizing Social Media to Understand the Dynamics of a City*, International AAAI Conference on Weblogs and Social Media, p.58.

4. Association Rules: Danish Medical Data

Title Temporal disease trajectories condensed from population wide registry data covering 6.2 million patients

Authors Anders Boeck Jensen, Pope L. Moseley, Tudor I. Oprea, Sabrina Gade Ellesøe, Robert Eriksson, Henriette Schmock, Peter Bjødstrup Jensen, Lars Juhl Jensen, and Søren Brunak

Date 2014

Sponsor Danish National Patient Registry

Methods association rules mining, clustering

Objective Estimating disease progression (trajectories) from current patient state is a crucial notion in medical studies. Trajectories have so far 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), this study finds connections between different diagnoses and how the presence of a diagnosis at some point in time might allow for the prediction of another diagnosis at a later point in time.

Methodology The following methodological steps were taken:

1. compute 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 14.9 years from January '96 to November '10 and consists of 68 million records for 6.2 million patients.

Challenges and Pitfalls

- Access to the National 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 exclusively (or at least, predominantly) be made 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 correlation using large groups for each pair of diagnoses. To compensate for multiple testing for close to 1 millions pairs and obtain a significant p—value, more than 80 million samples would have been required for each pair. This would have translated to a few thousand years' worth of computer running time. In order to avoid this pitfall, a pre-filtering step was included. Pairs included in the trajectories were eventually validated using the full sampling procedure, however.

Project Summaries 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. Three author quotes illustrate the importance of these results:

> The research could yield tangible health benefits as we move beyond one-size-fits-all medicine.
> — L.J.Jensen

> 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 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 clusters for two key diagnoses are shown in Figure 9.

References

[1] Jensen, A.B., Moseley, P.L., Oprea, T.I., Ellesoe, S.G., Eriksson, R., Schmock, H., Jensen, P.B., Jensen, L.J., Brunak, S. [2014], Temporal disease trajectories condensed from population-wide registry data covering 6.2 million patients, *Nature Communications*.

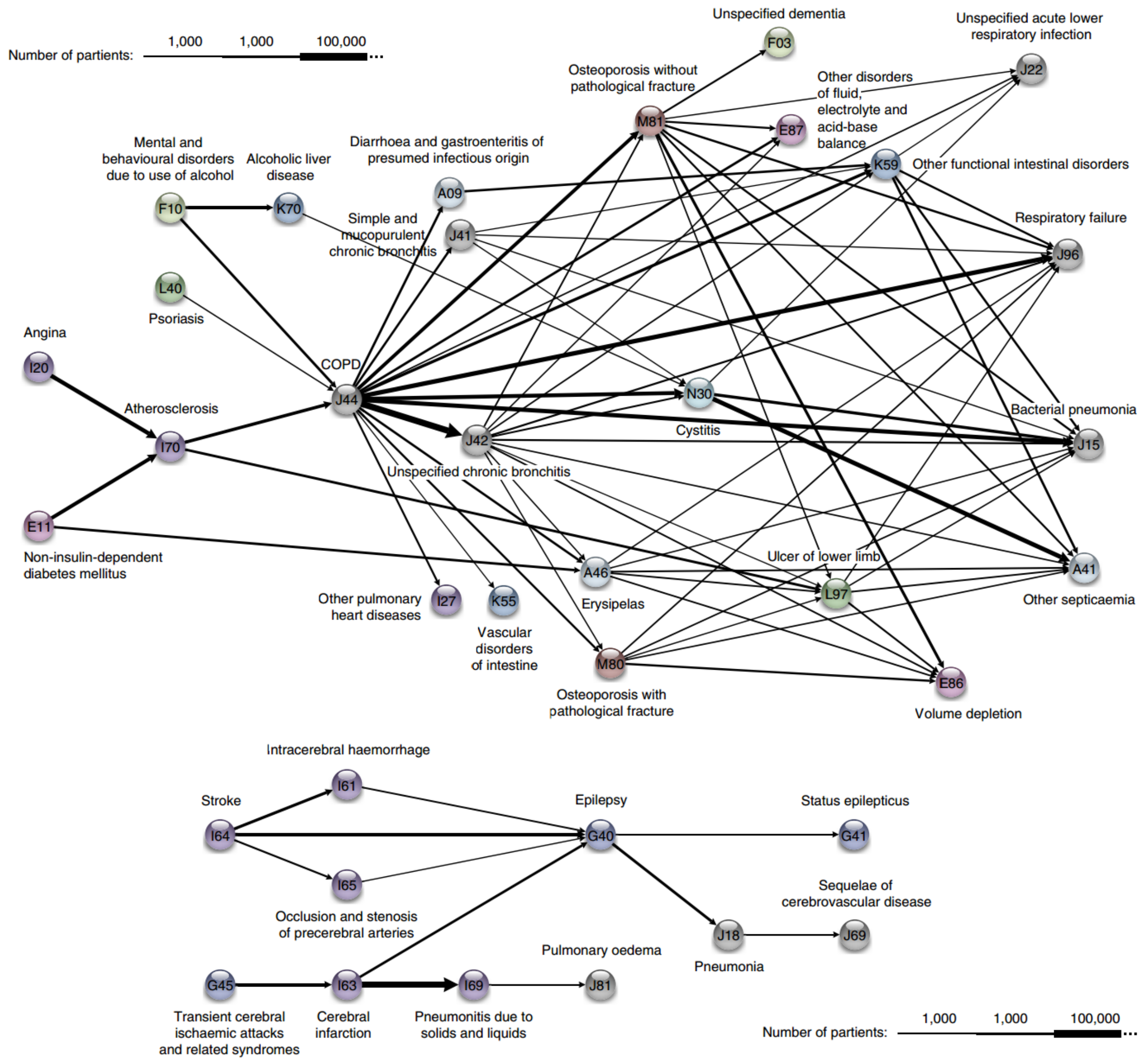

Number of partients: 1,000 1,000 100,000
Unspecified dementia
F03
Unspecified acute lower respiratory infection
J22
Osteoporosis without pathological fracture
M81
Other disorders of fluid, electrolyte and acid-base balance
E87
Mental and behavioural disorders due to use of alcohol
F10
Alcoholic liver disease
K70
Diarrhoea and gastroenteritis of presumed infectious origin
A09
K59
Other functional intestinal disorders
Simple and mucopurulent chronic bronchitis
J41
Respiratory failure
J96
L40
Psoriasis
Angina
I20
COPD
J44
N30
Atherosclerosis
I70
J42
Cystitis
Bacterial pneumonia
J15
Unspecified chronic bronchitis
E11
Non-insulin-dependent diabetes mellitus
A46
Ulcer of lower limb
L97
A41
Other septicaemia
Other pulmonary heart diseases
I27
K55
Erysipelas
Vascular disorders of intestine
M80
E86
Osteoporosis with pathological fracture
Volume depletion
Intracerebral haemorrhage
I61
Stroke
I64
Epilepsy
G40
Status epilepticus
G41
I65
Occlusion and stenosis of precerebral arteries
Sequelae of cerebrovascular disease
J18
J69
Pneumonia
Pulmonary oedema
G45
I63
I69
J81
Transient cerebral ischaemic attacks and related syndromes
Cerebral infarction
Pneumonitis due to solids and liquids
Number of partients: 1,000 1,000 100,000

Figure 9. The COPD cluster showing five preceding diagnoses leading to COPD and some of the possible outcomes; Cerebrovascular cluster with epilepsy as key diagnosis.

References

[1] Davenport, T.H., Patil, D.J. [2012], **Data Scientist: The Sexiest Job of the 21st Century**, in the *Harvard Business Review*, October.

[2] Mason, H. [2012], **What is a Data Scientist?**, Forbes.

[3] Few, S. [2017], Big Data, Big Dupe: a little book about a big bunch of nonsense, Analytics Press.

[4] Provost, F., Fawcett, T. [2015], Data Science for Business, O'Reilly.

[5] Dua, D., Karra Taniskidou, E. [2017], UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[6] Hastie, T., Tibshirani, R., Friedman, J. [2008], The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd ed., Springer.

[7] Siegel, E. [2016], *Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie or Die*, Predictive Analytics World.

[8] Brossette, S.E., Sprague, A.P., Hardin, J.M., Waites, K.B., Jones, W.T., Moser, S.A. [1998], Association Rules and Data Mining in Hospital Infection Control and Public Health Surveillance, Journal of American Medical Informatics Association, Vol.5, No.4, pp.373-381

[9] Garcia, E., Romero, C., Ventura, S., Calders, T. [2007], Drawbacks and solutions of applying association rule mining in learning management systems, Proceedings of the International Workshop on Applying Data Mining in e-Learning 2007.

[10] **http://www.rdatamining.com/examples/association-rules**

[11] **https://cran.r-project.org/web/packages/arules/vignettes/arules.pdf**

[12] **https://cran.r-project.org/web/packages/arulesViz/vignettes/arulesViz.pdf**

[13] **https://www.lynda.com/R-tutorials/Up-Running-R/120612-2.html**

[14] Webb, G.I. [2007], Discovering Significant Patterns, *Machine Learning* 68(1), Springer, pp. 1-33.

[15] Gionis, A., Mannila, H., Mielikäinen, T.; Tsaparas, P. [2007], Assessing Data Mining Results via Swap Randomization, ACM Transactions on Knowledge Discovery from Data (TKDD), Volume 1, Issue 3, Article No. 14.

[16] Leskovec, J., Rajamaran, A., Ullman, J.D. [2014], Mining of Massive Datasets, Cambridge Press.

[17] Risdal, M. [2016], Exploring Survival on the Titanic, Kaggle.com.

[18] Kitts, B., Zhang, J., Wu, G., Brandi, W., Beasley, J., Morrill, K., Ettedgui1, J., Siddhartha, S., Yuan, H., Gao, F., Azo, P., Mahato, R. (in press), Click Fraud Detection: Adversarial Pattern Recognition over 5 Years at Microsoft, Annals of Information Systems Special Issue on Data Mining in Real-World Applications.

[19] Kitts, B. (2013), The Making of a Large-Scale Ad Server, in Data Mining Case Studies Workshop and Practice Prize 5, Proceedings of the IEEE Thirteenth International Conference on Data Mining Workshops (ICDMW 2013), December, Dallas, TX, IEEE Press.

[20] Fefilatyev, S., Kramer, K., Hall, L., Goldgof, D., Kasturi, R., Remsen, A., Daly, K. (2011), Detection of Anomalous Particles from Deepwater Horizon Oil Spill Using SIPPER3 Underwater Imaging Platform, in Data Mining Case Studies IV, Proceedings of the Eleventh IEEE International Conference on Data Mining, Vancouver, Canada

[21] Kitts, B. (2005), Product Targeting From Rare Events: Five Years of One-to-One Marketing at CPI, Marketing Science Conference, Atlanta, June 2005.

[22] **https://algobeans.com/2016/07/27/decision-trees-tutorial/**

[23] **https://en.wikipedia.org/wiki/Decision_tree_learning**

[24] **https://en.wikipedia.org/wiki/Predictive_analytics**

[25] **https://en.wikipedia.org/wiki/Multivariate_adaptive_regression_splines**

[26] **https://en.wikipedia.org/wiki/Evaluation_of_binary_classifiers**

[27] Aggarwal, C.C. (ed.) [2015], Data Classification: Algorithms and Applications, CRC Press.

[28] NaÃŕve Bayes Classifier on Wikipedia

[29] Zhang, H. (2014) The optimality of NaÃŕve Bayes

[30] Domings, P., and Pazzani, M. (1997) Beyond independence: Conditions for the optimality of the simple Bayesian classifier

[31] Markham, K. Scikit-learn video #3: Machine learning first steps with the Iris dataset

[32] **http://www.ee.columbia.edu/ vittorio/BayesProof.pdf**

[33] **https://www.cs.cmu.edu/ epxing/Class/10701-08s/Lecture/lecture3-annotated.pdf**

[34] **http://www.cogsys.wiai.uni-bamberg.de/teaching/ss05/ml/slides/cogsysII-9.pdf**

[35] An Idiot's Guide to Support Vector Machines, by R. Berwick

[36] Support Vector Machine (and Statistical Learning Theory) Tutorial, by J.Weston

[37] Support vector regression for real-time flood stage forecasting, by P.S. Yu, S.T. Chen, I.F. Chang

[38] Torgo, L. [2016], Data Mining with R (2nd edition), CRC Press.

[39] Ng, A., Soo, K. [2016] Surviving a Disaster, in Numsense!, algobeans.

[40] Hastie, T., Tibshirani, T., Wainwright, M. [2015], Statistical Learning with Sparsity: The Lasso and Genarlizations, CRC Press.

[41] Chambers, J., Hastie, T. (eds.) [1992], Statistical Models in S, Wadsworth and Brooks/Cole.

[42] Aggarwal, C.C., Reddy, C.K. (eds.) [2014], Data Clustering: Algorithms and Applications, CRC Press.

[43] Aggarwal, C.C. [2015], Data Mining: the Textbook, Springer

[44] Maheshwari, A.K. [2015], Business Intelligence and Data Mining, Business Expert Press.

[45] Frank, E., Witten, I.H. [2005], Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed., Elsevier.

[46] Schubert, E., Sander, J., Ester, M., Kriegel, H.P., Xu, X [2017], DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN, ACM Trans. Database Syst. 42(3): 19:1–19:21

[47] Scientists Trace Society's Myths to Primordial Origins, by J. d'Huy. In Scientific American (Online). Published 2016-09-29. Retrieved 2016-10-11.

[48] Complex building's energy system operation patterns analysis using bag of words representation with hierarchical clustering, by U. Habib, K. Hayat and G. Zucker. Complex Adaptive Systems Modeling, 2016, 4:8.

[49] A Comparison of Antioxidant, Antibacterial, and Anticancer Activity of the Selected Thyme Species by Means of Hierarchical Clustering and Principal Component Analysis, by M. Orlowska, K. Pytlakowskae, A. Mrozek-Wilczkiewicz, R. Musiol, M. Waksmundzka-Hajnos, M. Sajewicz, T. Kowalska.Acta Chromatographica, 2016, 28.

[50] Automated detection of brain atrophy patterns based on MRI for the prediction of Alzheimer's disease, by C. Plant, S.J. Teipel, A. Oswald, C. BÃűhm, T. Meindl, J. Mourao-Miranda, A.W. Bokde, H. Hampel, M. Ewers.

[51] A Novel Approach for Predicting the Length of Hospital Stay With DBSCAN and Supervised Classification Algorithms, by Panchami V.U., N. Radhika, A.V. Vidyapeetham.

[52] Simulation of DNA damage clustering after proton irradiation using an adapted DBSCAN algorithm, by Z. Francis, C. Villagrasa, I. Clairand.

[53] Where traffic meets DNA: mobility mining using biological sequence analysis revisited, by A. Jawad, K. Kersting, N.V. Andrienko.

[54] Individual movements and geographical data mining. clustering algorithms for highlighting hotspots in personal navigation routes, by G. Schoier, G. Borruso.

[55] Justin Cranshaw, Raz Schwartz, Jason I. Hong, and Norman Sadeh, "The Livehoods Project: Utilizing Social Media to Understand the Dynamics of A City," Proceedings of the the Sixth International AAAI Conference on Weblogs and Social Media (ICWSM–12), Dublin, Ireland, pp. 1–8, 2012.

[56] Kung H. T., Vlah D.A, Spectral clustering approach to validating sensors via their peers in distributed sensor networks, Proceedings of the 18th IEEE International Conference on Computer Communications and Networks (ICCCN '09), 2009.

[57] F. R. Bach and M. I. Jordan, Spectral clustering for speech separation, in J. Keshet and S. Bengio (Eds.), Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods. New York: John Wiley, 2008.

[58] **https://en.wikipedia.org/wiki/Cluster_analysis_algorithms**

[59] **https://en.wikipedia.org/wiki/Davies–Bouldin_index**

[60] **https://algobeans.com/2015/11/30/k-means-clustering-laymans-tutorial/**

[61] **http://www.cs.umd.edu/ samir/498/10Algorithms-08.pdf**

[62] **https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/**

[63] Fisher, R.A. [1936], The use of multiple measurements in taxonomic problems, Annals of Eugenics. 7 (2): 179-188.

[64] Hassell, J. [2014], 3 Mistaken Assumptions About What Big Data Can Do For You, CIO.

[65] Chen, F., Ripley, B.D. [2003], Statistical Computing and Databases: Distributed Computing Near the Data, Proceedings of the 3rd International Workshop on Distributed Statistical Computing, Vienna.

[66] Grace, K. [2014], Trends in the Cost of Computing, aiimpacts.com.

[67] Kwartler, T., **Text Mining: Bag of Words**, retrieved from **DataCamp** on September 11, 2017.

[68] **Natural Language Processing**, retrieved from **Wikipedia** on October 5, 2017.

[69] Ide, N., Veronis, J. [1998], Introduction to the special issue on word sense disambiguation: the state of the art, *Computational Linguistics*, v.24, n.1, pp.2-40.

[70] Way, A., Gough, N. [2005], Comparing Example-Based and Statistical Machine Translation, *Natural Language Engineering*, v.11, n.3, pp.295-309.

[71] **The Stanford Parser: a statistical parser**, retrieved from **The Stanford Natural Language Processing Group** on October 5, 2017.

[72] Associated Press [2017], **Sens rally after blowing lead, beat Leafs to gain on Habs**, retrieved from **ESPN.com** on September 20, 2017.

[73] Anastasia, D.C., Tagarelli, A., Karypis, G. [2014], Document Clustering: The Next Frontier, in Data Clustering: Algorithms and Applications (Aggarwal, C.C., Reddy, C.K., eds.), CRC Press.

[74] Aggarwal, C.C., Zhai, C.X. [2015], Text Classification, in Data Classification: Algorithms and Applications (Aggarwal, C.C., ed.), CRC Press.

[75] Srivastava, A.N., Sahami, M. (eds.) [2009], Text Mining: Classification, Clustering, and Applications, CRC Press.

[76] Silge, J., Robinson, D. [2017], Text Mining with R: a Tidy Approach, O'Reilly.

[77] Jurafsky, D., Martin, J.H. [2009], Speech and Language Processing (2nd ed), Pearson.

[78] Aggarwal, C.C., Zhai, C.X. (eds.) [2012], Mining Text Data, Springer.

[79] Bird, S., Klein, E., Loper, E. [2009], Natural Language Processing with Python, O'Reilly.

[80] **http://aiplaybook.a16z.com/docs/guides/nlp#user-content-apiexamples**

[81] **https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/**

[82] **https://www.fastcompany.com/3037915/the-problem-with-sentiment-analysis**

[83] McCallum, Q.E. [2013], Bad Data Handbook, O'Reilly.

[84] Hoftsadter, D. [1979], Gödel, Escher, Bach: an Eternal Golden Braid, Basic Books.

[85] **https://en.wikipedia.org/wiki/Artificial_intelligence**

[86] Neural Networks Demystified, Welch Labs.

[87] Lawrence, S., Giles, C.L., Tsoi, A.C. (1996), What size neural network gives optimal generalization? Convergence properties of backpropagation. Technical Report UMIACS-TR-96-22 and CS-TR-3617, Institute for Advanced Computer Studies, University of Maryland, College Park.

[88] Elissee, A., Paugam-Moisy, H. (1997), Size of multilayer networks for exact learning: analytic approach. Advances in Neural Information Processing Systems 9, Cambridge, MA: The MIT Press, pp.162-168.

[89] Y. Bengio, Y. LeCun (2007), Scaling learning algorithms towards AI, in L. Bottou, O. Chapelle, D. DeCoste, J. Weston (eds), Large-Scale Kernel Machines, MIT Press

[90] Explaining and harnessing adversarial examples, I.J. Goodfellow, J. Shlens, C. Szegedy, ICLR 2015.

[91] No free lunch theorems for optimization, D.H. Wolpert, W.G. Macready, IEEE Transactions on Evolutionary Computation 1, 1997.

[92] Is AI riding a one-trick pony? MIT Technology Review, Sep 29, 2017.

[93] **https://en.wikipedia.org/wiki/Artificial_neural_network**

[94] Turing, A. [1950], Computing Machinery and Intelligence, *Mind*.

[95] Guyon, I., Elisseeff, A., An Introduction to Variable and Feature Selection, Journal of Machine Learning Research, 3(Mar):1157-1182, 2003.

[96] Cawley, G.C., Talbot, N.L.C., Gene selection in cancer classification using sparse logistic regression with Bayesian regularization, Bioinformatics, (2006) 22 (19): 2348-2355.

[97] Ambroise, C., McLachlan, G.J., Selection bias in gene extraction on the basis of microarray gene-expression data, PNAS, vol.99, no.10, pp.6562âĂŞ6566, 2002. Liu, H., Motoda, H. (eds), Computational Methods of Feature Selection, Chapman Hall/ CRC Press.

[98] Kononenko, I., Kukar, M. [2007], Machine Learning and Data Mining: Introduction to Principles and Algorithms, ch.6, Horwood Publishing.

[99] **https://en.wikipedia.org/wiki/Lasso_(statistics)**

[100] **https://en.wikipedia.org/wiki/Nonlinear_dimension_reduction**

[101] Gama, J. [2010], Knowledge Discovery from Data Streams, CRC Press/Chapman & Hall.

[102] Clark, C., September 2016, A Simple Content-Based Recommendation Engine in Python.

[103] Aggarwal,C.C. [2016] Recommender Systems: The Textbook, Springer.

[104] Koren Y. [2008], Factorization meets the neighborhood: a multi-faceted collaborative filtering model, KDD '08 Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining.

[105] Krashinsky, S. [2015], Privacy watchdog finds targeted ads still too personal, the Globe and Mail.

[106] Kargupta, H., Han, J., Yu, P.S., Motwani, R., Kumar, V. (eds) [2009], Next Generation of Data Mining, CRC/Chapman & Hall.