Digital Forensics and Cybercrime

Niccoló Didoni

June 14, 2023

Contents

Part I

Cybercrime

Chapter 1

Introduction

1.1 Risk

The main variable we have to use when talking about security is risk. Risk is a combination of the damage one can do by attacking a system and the probability that such an attack is executed. More precisely, risk is a combination of three factors:

- The **value of an asset**.
- The **presence of vulnerabilities** and the ease of exploiting them.
- The **existence of threats**.

As defenders, we can control the first two variables (hence we can control the assets we have to protect and the vulnerabilities that affect them, by patching them) but we can't control the third one which is an external agent. In fact, we can only analyse the presence of threats, but we can't control them.

Doing security means balancing the reduction of risk and vulnerabilities and the cost required to do achieve such reduction.

1.2 Threat landscape

Let's start by analysing the part of risk that we can't control, which is threats. In particular, we want to give an overview of the possible threats that can menace a system. A threat can be classified along three dimensions:

- **Internal** or **external**. External threats are those that come from a source external to the system (e.g., the organisation) we are securing. On the other hand, internal threats come from inside the organisation that is attacked.
- **Generic** or **targeted**. Generic threats do not aim at specific victims but are generally attacking all possible victims or a subset of them with specific characteristics (e.g., a class of people). Generic threats are in fact considered the background noise of the Internet since they are always present. On the other hand, targeted attackers focus on specific victims (e.g., a specific person or company), hence they are more specialised.

- **Financially motivated** or **motivated by other factors**. Financially motivated attackers act only to gain some monetary profit from the attack. This means that they have to consider the money spent preparing and doing the attack with respect to the money gained from it. Not all attackers are motivated by money, in fact, some are motivated by emotions, politics, ethics or religion. In this case, the attackers might have a lot of resources and do not care about the profit they do from the attack, hence they are even more dangerous since it's not enough to make the attack more expensive than the asset to protect it.

These dimensions have been used to create the Gartner quadrant of threats, which splits threats into four categories and for each category it defines the type of attack and the motivation behind it:

- **Internal and generic**. Internal threats are not that generic since they require an attacker inside the target company, hence the attack must be a bit targeted, however, we can fit disgruntled (angry or dissatisfied) employees in this category. This type of threat **isn't always motivated by money** and in fact, it's usually driven by emotions like anger or dissatisfaction.
- **Internal and targeted**. Internal and targeted threads are usually identified as dishonest employees that want to steal money or information (which turns into money) from the company they work in. This scenario is different from the previous one (internal and generic) since the attacker isn't moved by emotions but by pure profit, hence it's **financially motivated**. Dishonest employees are always present in a company, but if we assume that they are not, we might still consider socially-engineered employees. Such employees are tricked or forced into betraying the company they work for by using, for instance, fishing or blackmail. Socially-engineered employees aren't the real attacker, however, they should be considered a threat anyways since they might cause harm to the company. An example of a socially-engineered employee is the person who is in charge of keeping the money of the company which receives a fake e-mail from the CEO which asks to send money to a specific bank account. If the accountant believes the mail is true, it sends the money to the real attacker.
- **External and generic**. External and generic threats are the background noise of the Internet since they are identified as generic criminals that try to steal money from various people using very different techniques. These threats are mostly **financially motivated**.
- **External and targeted**. External and targeted threats cover a variety of attackers which **might or might not be motivated by a monetary profit**. In this category, we find activists that want to disrupt the image or the reputation of a company or attackers that want to steal information from a company.

	General	Targeted
Internal	Disgruntled employees	Socially-engineered or dishonest employees
External	Criminals, usually looking to make money	A variety of advanced attackers

Table 1.1: Gartner quadrant of threats.

General threats are usually more present than specific ones. The same is true for external threats that are more frequent than internal ones. This means that external and general threats are the most common, however, they aren't the most dangerous. The most dangerous are in fact internal

attacks, and in particular internal and targeted threats because the attacker has a very specific target, tools set and goal. That being said, most companies still use the castle model to secure themselves. This is a big issue since the castle model secures the company from the outside, hence mitigating external threats, however, it doesn't consider internal threats which are more dangerous. It's also worth noting that external and specific threats might have a high degree of danger since the attacker might be a large organised group with a large number of resources.

1.2.1 Internal threads

Since internal threads are the most dangerous, it's better to analyse them more in-depth. Internal threats are generated by attackers (or tricked attackers, like socially-engineered ones) that act from inside the company they want to attack. Internal attackers can be further divided into:

- **Malicious insiders**. A malicious insider is an employee that uses his or her position, which is usually a privileged role in the company to gain an advantage (usually monetary) and harm the company. These attackers misuse their legitimate access to confidential information for personal gain.
- **Inside agents**. Inside agents are attackers that try to get hired or get a higher-level position in the company to gain access to privileged information and attack the company from the inside. Inside agents are therefore recruited by the real attacker to get hired in a company and attack it from inside. Inside agents are usually paid by the attacker but they might also get blackmailed and forced to betray the company they already work in. Note that in this scenario the amount of money an attacker has to give to a person to be an inside agent depends on the person itself, which might have a strong morality. It's also important to note that the company can defend itself from inside agents by paying a role with privileged access more so that an employee can't be paid to betray the company (since the company pays it more).
- **Emotional employees**. Emotional employees are driven by emotions and usually attack the company as an act of revenge.
- **Reckless employees**. Reckless employees are people that do not apply the security rules of the company. As a result, they cause (voluntarily or not) damage to the company they work in.
- **Third parties**. Modern companies usually work with third parties, like consulting companies. Third-party employees aren't part of the company, however, they work closely with the company's employees, hence the former ones can gain some useful information about the company and its employees, which might be used to cause damage.

1.2.2 Data breaches

Data breaches are more and more common in companies and analysing them is useful to understand what type of threat is more common. By analysing known data breaches we can say that:

- Most of the attacks are external and mostly done by large groups of people.
- Most data breaches are financially motivated.

The information comes from intelligence, which aims at gathering data of which we aren't sure. This means that intelligence also specifies what level of confidence we have about it. In the case of data

breaches, we can't be completely sure about the statistics since, for instance, internal attacks might be dealt with internally and privately, hence no information is disclosed to the public. This means that the statistic on external and internal threats might be biased towards external ones.

1.3 History of malware

The history of malware can be divided into three stages:

1. Initially, the malware was written by a small group of highly skilled people whose only goal was that of showing off their skills. This means that no (or little) monetary interest was involved.
2. After this initial stage, attackers started to realise mass attacks fuelled by a monetary return. Because attackers could gain money from an attack, they could use larger groups (paid by the return of the attack) to prepare large-scale attacks.
3. The last step involves strategic attacks targeted to high-value, specific targets. In this period we also see the advent of state-level attacks and cyber warfare. At this stage, most of the actors are financially motivated, however, there is a subset of them who are motivated by other factors. Such attackers are very dangerous since they usually have an unlimited budget and resources and they don't care about the monetary return of the attack. This means that they can attack a specific person without caring whether they will get some money back. This scenario is very difficult to defend since we can't just make it so that the cost of the attack is more than the value of the attacked asset. Finally, let us highlight that a financially-driven attack depends on the value of the attacked person.

1.4 Financially oriented threats

Most of the threats are financially oriented, hence it makes sense to analyse such threats. Financially motivated threats can be divided into:

- Direct monetisation threats.
- Indirect monetisation threats.

1.4.1 Direct monetisation threats

Direct monetisation threats allow an attacker to directly steal money from a victim. Some examples of direct monetisation threats are:

- Ransomware.
- Fake anti-viruses that tell the user that the computer has been infected (with a multitude of non-existing viruses) and require money to buy a complete version of the anti-virus, required to delete said viruses.
- Premium calls. Initially, users connected to the Internet by dialling a number. A fake modem could dial a user to a premium number instead of dialling to the Internet, hence taking money from the user. These attacks are basically extinct since we connect to the Internet using routers.
- Credit card or bank account frauds.

Ransomware

One of the most popular ways of stealing money from a victim is using ransomware.

> **Definition 1.1** (Ransomware)**.** *Ransomware is software, installed on the victim's machine, that encrypts the user's data and requires the victim to pay a ransom to get the data back.*

Ransomware has been theorised much earlier than becoming popular, however, it couldn't be initially implemented because the technology to handle payments wasn't available. In particular, ransomware requires payments which are:

- Easy to use by any user (even not experts).
- Digital.
- Non-reversible (i.e., the victim must not be able to get the money back).
- Untracked.

However, the invention of cryptocurrencies (starting with Bitcoin's paper in 2008) and of exchanges to use them allowed ransomware to be practically implemented to steal money from many victims. Ransomware attacks can be split into two categories:

- Private mass-market attacks that target a single person.
- Company attacks that target a whole company.

An attack using ransomware usually follows the same steps:

1. A victim is tricked into opening a file or a link attacked to a mail (which seems legitimate to the victim). The ransomware is then downloaded and installed on the victim's machine (e.g., using macros for files).
2. The ransomware encrypts the files on the victim's machine and prompts the user with a message that says that the computer has been encrypted and the only way of obtaining the data back is to pay a ransom. The ransomware also shows the instructions for buying cryptocurrencies and paying the attackers. Usually, this page contains a timer at the end of which the files will be deleted. This puts the victim under pressure so that he or she doesn't act rationally but impulsively. The victim has enough time to buy the crypto coins to pay the ransom but not enough time to seek help, hence the user has to act emotionally without thinking.
3. When the victim pays the ransom, he or she (usually) receives its data back.

Technically, the core of ransomware is the encryption mechanism, which has to be properly implemented to make the attack work. When the ransomware is installed and it has to encrypt the data:

1. A random symmetric key for a robust algorithm (usually AES) is picked on the victim's machine.
2. The files on the victim's machine are encrypted using that symmetric key. Some ransomware generate more than one key and encrypt different files with a different key.

3. The symmetric key is encrypted with an RSA public key obtained from a central control and command server. The C&C server holds the private key related to the public key used to encrypt the symmetric key. A public-private key pair is associated with a single victim machine.

4. When the user pays the ransom, the program on the victim's machine downloads the private key from the C&C server and decrypts the symmetric key. The symmetric key is then used to decrypt the victim's files.

Ransomware use symmetric encryption instead of applying RSA directly because the former is much faster than the latter (even 1000 times faster), hence using RSA would slow the whole process down by a lot (especially when the malware has to encrypt even terabytes of data) and allow the user to realise that something is happening.

A ransomware attack is, if well implemented, impossible to stop and the only way to get the data back is by paying a ransom. However, some implementations might do some mistakes like:

- Using a broken or weak encryption algorithm for which it's possible to write decryptors that work without knowing the symmetric key.
- Not deleting the symmetric key after encrypting it. This allows a defender to recover the symmetric key from a memory dump without knowing the private RSA key.
- Not deleting the data after having encrypted it. This allows a defender to recover the encrypted files from a memory dump without knowing either the symmetric key nor the private RSA key.

1.4.2 Indirect monetisation

Indirect monetisation means having a profit that doesn't come directly from stealing money. Some examples are:

- Gathering information that can be sold.
- Enabling abuse of computing resources.
- Selling or renting botnets.

Botnets

Botnets are collections of machines infected with malware controlled by a central control and command server that issues commands which are executed by the infected machines. Once an attacker has built a botnet, he or she can rent or sell some bots (i.e., some machines of the botnet, a section of the botnet) to other malicious agents which in turn use the bots for their attacks. This allows attackers to monetise the software used to infect the machines in the botnet.

Botnets come from the IRC protocol which uses IRC channels to which users can connect and communicate. When a user connects to the IRC network, he or she can choose a nickname among those that are not already taken, however, the nickname isn't permanently assigned to the user, but it's lost when the user disconnects from the network. Moreover, each IRC channel is identified by a unique name (preceded by an #) and, when the channel is empty it's destroyed. When a user joins an empty channel he or she gains operator privileges. This means that most users wanted to stay connected to the IRC network (to keep their username), be the first to join a crowded channel and never leave it so as to have operator privileges. Since normal computers weren't connected to the

Internet, people broke into systems constantly connected to the Internet (like universities) to install scripts which kept them connected to the IRC network. These scripts were called robots, from which we get the word **bot**. When bots are connected they are called botnets.

Apart from using universities' machines, bots and botnets weren't malicious on their own, however, with time, the idea has been reused to build large networks of regular machines. The size of such networks allowed attackers to compromise big companies by mounting Distributed Denial of Service attacks. DDOS attacks are nowadays mitigated by big companies thanks to more resilient infrastructures with many redundancies, however, other smaller companies are still vulnerable to DDOS attacks.

1.4.3 Cybercrime in different countries

When analysing the presence of botnets in different countries we always have to remember that, if data is not normalised with respect to the population, it's clear that the countries with more computers are more likely to have larger botnets.

Another important observation we can do is that the presence of a country in cybercrime depends on its evolution and on the circumstances in which the population lives. In any case, the problem is not about the people. Consider for instance Roumania. Twenty years ago, cybercrime was very popular in Roumania because the country just got out of the iron fence and young people had attended good STEM universities (as it was advertised by the URSS government). This means that Roumania was full of young people trained in maths, science and computer science but with a good living perspective and no future. This lead to many people using cybercrime to make money. Now Roumania has joined the European Union, the living perspective has improved and many companies have moved their headquarters there, hence young well-trained people don't have to commit cybercrimes to survive since they can be hired by the companies in the country.

We also have to consider that some countries, like Russia and China, have tolerant policies against cybercrime when not committed against their own nation. This means that cybercriminals attacking targets in foreign countries aren't persecuted.

1.5 Malware families

Malware can be divided into different families which are usually related to a single threat actor. More in general, each malware family falls into a more general description. The main are:

- **Credential stealers**. A credential stealer is a program that tries and steals the user's information (e.g., username and password, id, social security number, credit card pin or CVV) monitoring, for instance, keystrokes. In general, a user identity is not very worthy, however, an attacker can sell it, for a small price, to a large number of people to make a huge profit.
- **e-banking Trojans**. An e-banking Trojan is a credential stealer specifically built to steal credentials in bank operations.
- **Remote Access Tools**. A Remote Access Tool (RAT) is a more general term used to describe a botnet.
- **Loaders**. A loader is a piece of software that allows to download and install another software to a machine.

1.6 Cybercrime ecosystem

Cybercrime isn't committed by a single attacker that builds the whole attack on his or her own. An attacker usually buys from third parties tools to build the attack and then composes it. This means that the cybercrime scene can be seen as a complex ecosystem in which each actor buys something from other actors and sells something else that can be used by the ecosystem.

Let's start our analysis of the cybercrime ecosystem by describing the mechanism used to install malicious software on a victim's machine. Malware can be installed mainly in two ways:

- **Emails**. In this scenario, a victim receives a mail or even a reply to an email he or she previously sent. Most of the time the email is clearly fake since it comes from someone from which the user doesn't receive emails or the object doesn't match the sender's context. Sometimes the email makes sense to a general user, hence he or she opens it. The message contains a document or a link that when opened downloads the malware on the victim's computer.
- **Drive-by download**. Drive-by download exploits vulnerabilities in the browsers. In this scenario, a user is tricked into visiting a malicious website (or a legitimate website that has been compromised) that exploits the browser's vulnerability to install malware on the victim's machine (i.e., the one with the vulnerable browser). The malicious website is usually either custom-built by the attacker or compromised (e.g., using XSS).

As we can see in the second example, a drive-by download is made of many steps which require different programs and resources to work. An attacker, in fact, has to:

1. Obtain malware he or she wants to install on the victim's machine.
2. Obtain the exploit for the browser vulnerability.
3. Compromise a legitimate website or obtain a custom malicious website.

It could be hard for a single attacker to swiftly realise all these steps, hence they are usually outsourced. This allows the attacker to build an attack in a short time and obtain the resources needed from specialists. In the drive-by download we can recognise some fundamental parts of the cybercrime environment, however, there exist many more:

- **Software developers from which an attacker can buy malware**.
- **Exploits experts** from which an attacker can buy exploits.
- **Experts that write configuration files**.
- **Money launderers** that are able to transform digitally tracked money saved in bank accounts into clean cash.

Generally, these activities are hard to prosecute since, by themselves, they are legitimate and become a crime only when used jointly and to cause harm. Moreover, criminals tend to outsource their attack's components to countries where a certain activity is legitimate.

1.6.1 Malware developers

Malware developers are software developers specialised in writing malware. From an attacker's point of view, it's useful to outsource malware creation because malware can be very complex. If we consider the drive-by download example, some malware might, for instance, mount a man-in-the-browser attack which consists in replacing the user browser. The malicious browser works exactly as the legitimate one, however, it might also modify the pages searched by the user to steal information from him or her. Say for instance the user visits his or her bank website. The malicious software could add a form field to the page returned by the bank server which asks the user to insert the credit card security number. Such an attack is quite complex hence it's better if it's built by a specialised third party.

1.6.2 Exploit developers

Exploit developers are software developers that write exploits for known vulnerabilities. Usually, exploits are bought as exploits-as-a-service, which means that the buyer pays a periodic fee to regularly get fresh exploits such that the attacker is always provided with non-patched exploits. Note that even exploit creators might have bought vulnerabilities from third parties.

1.6.3 Configuration file experts

Some attacks or malware require complex configuration files, hence it's better to rely on specialised people. Bank Trojans, for instance, require a configuration file to be correctly installed on the victim's machine. An attacker could then turn to a specialised person to correctly write the configuration files for the attacker's purpose.

1.6.4 Money launderers

Money launderers transform digitally tracked money saved in bank accounts into clean cash. Money laundering isn't specific to cybercrime but is applied in every criminal activity, hence this is the only point of contact of cybercrime with classical crime. As for other activities, launderers can offer laundering-as-a-service. Money laundering is usually done in three different ways:

- The attacker **buys something that can't be revoked** (since bank payments can be reverted).
- The attacker buys something physical. This is usually harder directly from a bank account but it's easier from a credit card.
- The attacker uses **exotic travels**. Exotic travels are flights that are not very popular and are usually done only by people returning to their home country for visiting their parents. Travel companies buy a lot of exotic travel tickets so that they are able to sell them for cheap. This means that those purchasing exotic travel tickets know that a company offering cheap travel tickets can be trusted since this is the usual behaviour. An attacker could buy exotic travel tickets for their normal price and then sell them at a cheaper price (i.e., at the price a buyer would expect from a legitimate travel company that handles exotic travel). The buyer is required to pay cash, hence so that the attacker can clean the money in his or her bank account (the money in the bank account has been used to buy the tickets in the first place). The downside is that the attacker loses some money (since he or she sells tickets at a lower

price than the one at which he or she bought them), however, this always happens in money laundering.

- The attacker uses a **money mule**, namely a person that withdraws money from a bank account and sends them to the attacker. The mule can be aware of what he or she is doing (more rarely) or not (more frequently). A way to trick a person into becoming a money mule is a variation of the Nigerian king scam, which works as follows:

  1. The attacker contacts the mule promising to periodically send money to the mule's bank account.
  2. The attacker asks the mule to withdraw a part of the money sent and ship them to the attacker's address.

  This scam is very effective because the mule has no way of identifying the scammer, hence he or she (the victim) will be persecuted by the authorities whereas the scammer is free to use the cash he or she has received.

Chapter 2

Abuses of cryptocurrencies

2.1 Technical analysis

Cryptocurrencies are used in many fields and have been designed to solve a very specific problem, however, a technology can be abused and used for criminal intents. This means that cryptocurrencies aren't bad in general but can also be used for fuelling illegal activities. Since Bitcoin is the first cryptocurrency widely adopted, we will analyse it and then generalise the concepts seen for Bitcoin to describe how cryptocurrencies are used in illegal activities.

Bitcoin has been invented with the purpose of creating decentralised storage of information such that it can't be manipulated and it has some interesting properties. This means that Bitcoin isn't the first example of digital cash, however, it's the first one that doesn't use a centralised authority to handle cash (i.e., differently from what banks do). More precisely, Bitcoin defines a protocol over the Internet where nodes (i.e., users) mine bitcoins, and manage a group of addresses that holds coins. Each address is a hashed image of an underlying private-public pair of cryptographic keys and acts as a pseudonym of the coin's holder. This means that a key can be associated with a user, even if we don't know his or her identity. The address, which basically is an alphanumeric string, is used to receive and send Bitcoins.

2.1.1 Blockchain

As we have said, the network has no centralised authority to handle transactions hence the common state of the network has to be shared among all participants. The common state shared is called **blockchain** and it's an append-only ledger containing blocks of transactions used by everyone without a trusted authority managing insertions in the ledger.

2.1.2 Transactions

Let us now understand how Bitcoins are moved from one address to another. Each person can have a set of addresses (which are picked by the user itself by generating a private-public key pair) which are stored and managed in a wallet. The addresses stored in a wallet can be used to make transactions, which are the only way of transmitting Bitcoins from one address to another. Note that a wallet doesn't store the balance related to a key but, since the ledger is public, it can be obtained by parsing the whole ledger and summing up the Bitcoins sent and received by the address.

This is a very inefficient way of tracking a key balance, however, it's the only way of doing it in a distributed manner. The process of creating a transaction works as follows:

1. A user creates a transaction.
2. The transaction is signed with the user's private key.
3. The transaction is sent to the public ledger (which is not the blockchain).
4. The transaction is validated by the network. Namely, the nodes in the network check if the balance of the sender is not empty.

Note that a transaction moves all the Bitcoins specified at an address into the receiver's address. If one wants to send only some of its Bitcoins he or she can specify the real sender and a change address owned by the sender so that the sender gets back the Bitcoins he or she doesn't want to send. More precisely, a transaction contains:

- The public key of the user sending the Bitcoins.
- An hash of the previous transaction and the address of the sender. This allows nodes to check the order of transactions and check if a transaction is valid or not.
- The signature of the hash using the sender's private key.

Once transactions are published on the ledger, the network has to validate them to avoid situations in which a user sends the same Bitcoins to different users. Transactions are approved and verified by adding them to the blockchain. Each block of the blockchain contains several transactions, chosen by the miner among the transactions in the ledger, hence the blockchain block is the basic storage unit of Bitcoin. The blockchain makes it impossible to change a transaction that is already in the blockchain. This goal is achieved by implementing a blockchain as a linked list in which blocks can be added only to the head. Each block in the blockchain contains the hash of the previous block of the blockchain. This makes it impossible to change transactions already in the blockchain since, changing a block would require changing all successive blocks (which is, as we'll see, a very expensive operation). The order in which transactions are executed is defined by the order in which instructions are in blocks, namely, a transaction in block B_i is executed before one in block B_2, even if the former has been sent to the ledger after the former. This is because a block is created arbitrarily by a miner, which might choose to exclude some transactions from a block. The blockchain is therefore the distributed ledger which contains the transactions that have been executed and verified.

2.1.3 Mining

The core of Bitcoin is the blockchain. We've said that blocks are added to the blockchain by some people called miners, however, we haven't really explained how blocks are validated, namely how the other nodes can be sure that the miner has correctly added a block to the blockchain. The process of adding a block to the blockchain is called mining and requires the miner to append a number to the block such that the hash of the block starts with a certain number of zeros. The number of leading zeros increases with time, hence at the beginning blocks had to start with only one zero whereas now they have to start with multiple zeros. This means that the complexity of mining increases with time since finding a hash with multiple leading zeros is harder (we have more constraints). Note that, given that a hash function is non-invertible, the only way of finding one of the possible hashes is by brute-forcing all possible values to append to the block. This means that the mining process

is quite complex and blocks are added sometime after being created (usually around 10 minutes). Also, remember that the network has multiple miners that compete for one against each other to solve the problem first and get a reward and the difficulty of finding the next block increases or decreases in relation to the number of miners. In practice, a miner:

1. Picks some transactions and puts them in a block.
2. Adds the hash of the previous block (i.e., the hash computed by the miner who mined the previous block) to the new block.
3. Appends a number to the block.
4. Computes the hash of the block.
5. If the hash starts with a certain number of zeros, the mining process stops, otherwise the miner goes back to step 3 and tries to append a different number.

When the mining procedure ends, the block is added to the blockchain and the miner is rewarded with:

- Some Bitcoins (the exact number depends on the number of Bitcoins already in the network).
- The fees paid by the addresses sending Bitcoins, in fact, in a transaction, the sender has to add a fee to the money sent.

The fees and the reward obtained by the miner are added to the same block mined.

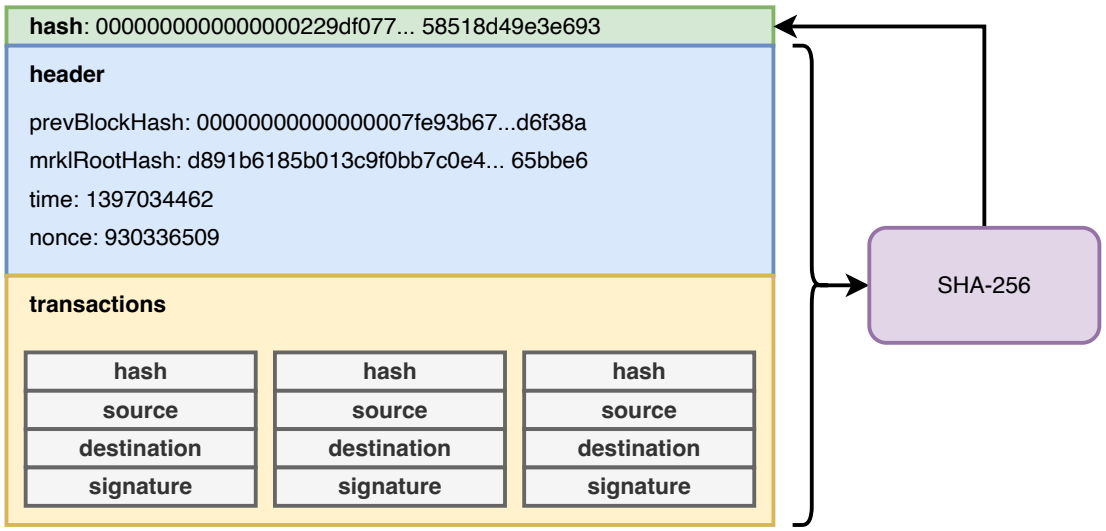


Figure 2.1: A Bitcoin block.

Note that mining is also used to give Bitcoins a real value, in fact, the value of a Bitcoin usually is the cost required to mine a Bitcoin.

2.1.4 Blockchain forks

Miners compete to compute blocks' hashes, hence it might happen that two miners find the hash of two blocks (which might or might not have some transactions in common) at the same time. When a miner computes a hash, it propagates the solution to all the other nodes in the network, which is a peer-to-peer network, however, this process takes time. This means that, when two miners find a solution at the same time and propagate the result, different nodes of the network might receive one block before the other. Let us consider an example to make things clearer. Let's start with a situation where all nodes of the network have the same blockchain and block B_0 is in the head of the blockchain. One miner in the US computes the hash of a block B_u, adds it to the blockchain and propagates the update to its neighbours (e.g., nodes in South America). At the same time, a miner in Australia computes the hash of a block B_a, adds it to the blockchain and propagates the results to the nodes in Asia. This means that:

- Nodes in the American continent have the US update.
- Nodes in Asia have the Australian update.
- Nodes in Europe and Africa don't have an update.

The updates keep propagating and

- The US update reaches the African continent, this means that America and Africa see only the US update.
- The Australian update reaches Europe, this means that Asia and Europe see only the Australian update.

When the US update reaches Europe and Asia (and the Australian reaches America and Africa), the blockchain has to be split because both updates contain the hash of B_0. This bifurcation is called a fork and is a problem since we want to have only one version of the ledger. To solve the problems with forks, Bitcoin considers the longer branch of a fork as the legitimate ledger, hence shorter branches are forgotten and only the longest is incremented, hence we always have a single ledger. Let us continue our example to understand how forks are solved. Say that a miner in Europe started to compute the hash of a block before receiving the US update, hence he or she used B_a hash to compute the hash of the new block B_e. When he or she finds the hash, the block is added after B_a and the update is propagated to everyone. The branch $B_a \to B_e$ is longer than the branch B_u, hence the US update is forgotten by the miners. Note that a miner is encouraged to work on the longest chain because, if he or she worked on the longest one, his or her work would be useless, since the other miner would be working on the longer chain.

Since a blockchain contains forks, a user can't be sure that a transaction has been validated until it's added to a block which is far from the head of the chain. This is because a fresh block might be on a short chain, hence it might be forgotten. On the other hand, if a block has been in the longer chain for a lot of time, we can be sure that no other chain can be longer than it, hence the transaction is verified.

2.2 Blockchain for cybercrimes

2.2.1 Anonymity and identity tracing

Bitcoins are used in many criminal activities because a criminal can have as many keys as he or she wants and each key is not connected to its identity (whereas a bank account is connected to

a person). Moreover, the key is generated by the owner itself, hence other users can't check if a certain key belongs to a person. One might think that, because of the properties we've listed, Bitcoin provides anonymity for its users. This is however false, in fact, it only provides **pseudo-anonymity** since a key is a pseudonym for a user. Differently from banks, which keep accounts balances and transactions secret, Bitcoin stores in clear each transaction together with the source and the destination of the transaction. This means that everyone can check the balance associated with a key and the Bitcoins that a key has sent or received.

Thanks to this fact, we can analyse the transactions to build a graph where each node contains a set of keys which we think belong to a single user. This analysis is done remembering that:

- In a transaction, all the Bitcoins of the sender address are transferred to the receiver.
- A transaction can have multiple receivers (each receiving a different share of the sender's Bitcoins). This is useful because, if the sender doesn't have to transfer all his or her Bitcoins, then he or she can specify as receivers the address of the true receiver and another address he or she owns where the remaining Bitcoins are sent. The second address (owned by the sender) acts as a change and is usually called **shadow address**. The shadow address specified in the transaction can be used to build the user graph since it belongs to the same wallet as the sender. Note that in general, it's hard to understand which receiver address is the indented receiver and which is the shadow address, however in earlier versions of the protocol, the shadow address was always put as a second address hence it was easier to associate the shadow address to the sender.
- A transaction can have multiple senders. This is usually done when one key doesn't contain enough Bitcoins, hence they are taken from multiple addresses. This means that, when a transaction has multiple sender addresses, we can reasonably associate the source keys to the same user.

A criminal can hide his or her keys using a key mixer which instead of making a transaction between two parties, distributes the transaction between different nodes such that the result is the same (the Bitcoins are sent from source to destination) but the Bitcoins are moved from one node to the other without moving directly from sender to receiver.

Note that user graphs don't tell us the true identity of a user but allow us to group all keys related to the same person or group. This is however a big advantage since, at some point, a criminal might want to convert his or her Bitcoins into real money, hence it must use a bank or other techniques which reveal his or her identity.

2.2.2 Silk Road

One famous example where user graphs helped find a criminal is Silk Road, a famous black market website. Silk Road, which has been shut down, used Bitcoins as a payment method. In particular, users had to deposit some credit which will then be used to pay for stuff. This means that payments weren't done directly from buyer to seller but through Silk Road. In particular:

- A buyer would deposit his or her Bitcoins on one of Silk Road's addresses.
- Silk Road would put the Bitcoins in the seller's account.

Since Silk Road acts as an intermediary, it also works as a mixer which can hide buyers' and sellers' addresses. Even if this looks like an advantage, it leads investigators into finding the keys used by Silk Road and its creator. This is because the Silk Road addresses participated in many

transactions (since they sent money to sellers and received money from buyers) hence it was possible to discover what addresses were more active. This analysis is usually easier when considering the source addresses, however, for destination addresses, it's possible to analyse the balance (since it's public) and find out the addresses with more Bitcoins. Address analysis wasn't enough, however the creator of Silk Road published one of the addresses that were taught to belong to Silk Road, hence the address was associated with him.

2.2.3 Ransomware

Bitcoins, and cryptocurrencies in general, are used as payment for ransomware. This means that it's possible to track down addresses that are related to ransomware. In particular, one can plot the balance of an account and compare it with ransom payments. If the oscillations of the two graphs are similar, then the key is probably related to a ransomware attack.

2.3 Other cryptocurrencies

Bitcoin isn't the only cryptocurrency, in fact, many others have been created. One of the most relevant is Monero which, differently from Bitcoin, verifies a block in the blockchain using zero-knowledge-proofs. This means that miners verify a block without checking the content of the block. As a result, it's impossible to track down transactions and addresses balance. On the flip side, Monero is much harder to use with respect to Bitcoin, hence it can't be used for ransomware. What criminals usually do is require payment in Bitcoin and then convert them to Monero so that they can be made disappear.

Part II

Digital forensics

Chapter 3

Digital Forensics

3.1 Introduction

Digital forensics (always remember that 's' at the end of the word) is a key element in the analysis and prosecution of a crime. First, let us define what forensics is.

> **Definition 3.1** (Forensics)**.** *Forensics is the application of scientific analysis methods to reconstruct evidence.*

Forensics can be applied in many fields, many giving an important contribution to court cases (e.g., ballistic forensics, pathology forensics). In our case, we are interested in digital forensics.

> **Definition 3.2** (Digital forensics)**.** *Digital forensics is the application of scientific analysis methods to digital data, computer systems, and network data to reconstruct evidence.*

The key part of forensics is its reliance on the scientific method. This means that we have to further specify what we mean by scientific analysis. This is particularly important when using forensics in courts of law since, in such a scenario, we should consider only:

- **Witnesses**. A witness is someone who has personally attended a crime and hence can speak for what he or she has seen.
- **Evidence**. Evidence is the only case in which a person, which is not a witness, can testify in a court of law. An expert is allowed to testimony only if the evidence he or she presents is based on a scientific method.

Since an expert can testimony without being a witness, it's important to define what evidence can be presented in a court, since the one presenting wasn't witnessing the crime. Namely, we have to define what scientific means. Note that the definition of what can be admitted depends on the country we are considering, however, most of the definitions used in literature are based (or at least biased towards) the US system. The definition of scientific analysis we will use is the Daubert standard.

Definition 3.3 (Daubert standard)**.** *A witness who is qualified as an expert by knowledge, skill, experience, training, or education may testify in the form of an opinion or otherwise if:*

1. *The expert's scientific, technical, or other specialised knowledge is relevant to the case and will help the trier of fact to understand the evidence or to determine a fact in issue.*
2. *The testimony is based on sufficient facts or data.*
3. *The testimony is the product of reliable principles and methods.*
4. *The expert has reliably applied the principles and methods to the facts of the case.*

Let us analyse more in-depth the properties an expert has to satisfy to be allowed as a witness in a court of law:

1. The first property tells us that the expertise of the witness has to be relevant to the case.
2. The second property tells us that the expert doesn't have to make guesses based on experience or skill if facts and data aren't presented. The expert shouldn't even give his or her own opinion if not supported by data.
3. The third property tells us that the methods applied by the expert should be well-tested, documented and standardised. Moreover, they should be based on scientific consensus and on publications.
4. The fourth property tells us that the expert should reliably apply the scientific techniques (with the characteristics defined by the third property), namely he or she should be able to do scientific experiments that prove his or her statements.

3.1.1 Scientific method

The Daubert standard defines the characteristics of an expert, however, it doesn't define what is scientific. The answer to this question is very complex, but we can present two simple solutions, which are not complete, but are enough for our purposes.

Repeatable

Galileo associated the concept of scientific with the concept of repeatability. He stated that:

Definition 3.4 (Repeatable)**.** *A process, experiment or method is scientific if it can be described so well and precisely that someone could repeat the experiment only by reading its description and another expert is able to tell if the experiment can be done or if it contains an error.*

Note that repeatability doesn't mean that the experiment must be necessarily repeated, but that, given the description, it could be repeated. Consider for instance an autopsy. If an autopsy is well documented and it's executed following the rules, it's a repeatable experiment since an expert can validate it, even if the same body can't be cut again (hence it can't actually be repeated).

Falsifiable

Another definition of scientific method comes from Karl R. Popper. He stated that:

> **Definition 3.5** (Falsifiable)**.** *A process, experiment or method is scientific if one is able to describe the experiment that would prove it false (independently from the outcome of the experiment).*

This means that a statement is scientific if its opposite can be proven (in the sense that we can define an experiment to prove the opposite, independently from the result of the experiment). Consider for instance a malware that does something malicious and then perfectly deletes itself (such that it's impossible to trace what it's done). In this case, after a successful attack by such malware, it's impossible to describe an experiment for proving that no malware perfectly deleted itself, hence we can't apply a scientific procedure for discovering such malware.

Extension to Daubert standard

Now that we have a better understanding of the scientific method, we can extend the properties listed in the Daubert standard (3.3):

1. The theory used for making a statement or identifying some evidence has to be accepted by the scientific community, in the sense that the majority of the experts must confirm that the theory is sound (according to the currently available instruments).
2. The theory should be subjected to peer review and published. It's also important that the publication from where the theory is taken isn't written by the same expert who is using the theory but by a third party which has no interest in the case. Namely, the scientist using the theory shouldn't be the same one who published it.
3. The theory has been or can be tested.
4. The error rate, when proving the theory, has to be acceptable.
5. The theory should be independent of the case in which it's used.

3.1.2 Usage of digital forensics

Digital forensics isn't limited to courts of law and can be used in many different fields. The differences between the fields of application are:

- In the **type of results**. In fact, finding evidence for sending someone to jail is different from finding evidence for firing someone.
- In the **constraints**. Different investigators might have different constraints. For instance, a police investigator might have more clearance (or less, depending on the country we are considering) than a private one.
- In the **aim**. In some cases, the expert has to find evidence in support or against a statement whereas in others the expert simply has to collect all evidence.

Courts of law

If we stick to courts and crimes, digital forensics can be applied at different levels. In some cases, the digital part is the key to the crime. Some instances are child pornography (which unfortunately is the area where digital forensics is more frequently applied), frauds, cyber extortion and cyber threats. In other cases, we have a digital incidence which might be more or less important but that isn't the main focus of the case. An example is espionage. Digital forensics is also applied to policy violations (e.g., an employee doesn't follow the company's rules) and copyright infringements.

3.1.3 Phases of an investigation

Each investigation can be divided into four phases:

1. **Acquisition of sources**. In this phase, the investigators get the sources of evidence, namely the sources from which they can extract some evidence. Some examples are hard drives and computers.
2. **Identification of evidence**. In this phase, the investigators extract evidence from the sources collected in the previous stage.
3. **Evaluation**. In this phase, evidence is compared against what the court has to prove and the elements of the crime. This is the phase in which the experts talk to the persecutors.
4. **Presentation**. In this phase, evidence is collected and presented to the court.

Let us now analyse each phase of an investigation more in detail.

3.2 Acquisition of sources

Before describing the first phase of an investigation and defining the best practices to use, we have to stress that everything we'll analyse is biased towards the US system since it has been developed in and for the States. In the States, crimes are evaluated by a jury which is allowed to see only the evidence that the judge deems admissible, hence, if the judge thinks that some evidence is not admissible, the jury will never see it or hear of it, hence the jury won't be influenced by it. In other parts of the world (e.g., in Italy), there is no jury and the final decision is in the hands of the judge who also decides if some evidence should be considered or not. This is very important since in this second case the judge will see the evidence in any case, hence he or she might be influenced even if the evidence isn't considered acceptable. In the US (and in the countries that use the same system), it's much more important to track evidence. In particular, in the US, evidence is tracked using a **chain of custody** which ensures that evidence is always preserved correctly and not disrupted when moved from one place to another. If the chain of custody is broken, the evidence can't be admitted to court, hence the acquisition of sources is much more important in the US than in other countries. This is because in Italy the judge will see some evidence even if it is compromised. On the other hand in the US, it's vital that the jury sees only what is admitted to court.

The acquisition of evidence in the cybercrime field is governed by different laws and standards. For instance, the Convention of Budapest on cybercrime is the international law applicable in the Council of Europe states. Moreover, there exist standards (under the 27000 ISO classification, which defines standards for cybersecurity) that help in correctly acquiring sources of evidence:

- **ISO/IEC 27037:2012**: Information technology — Security techniques — Guidelines for identification, collection, acquisition and preservation of digital evidence.

- **ISO/IEC 27035:2011**: Information technology — Security techniques — Information security incident management.
- **Guidelines for Evidence Collection and Archiving**.

3.2.1 Source brittleness

When acquiring sources of evidence we have to remember that digital sources are brittle. This means that there is no way, in general, to tell if they are modified. For instance, we can't tell if a bit of a hard disk is flipped. This is very different from other fields in which it's usually possible to understand if some source of evidence has been tampered with. As a result, digital evidence can be modified, hence we say that digital evidence is not **tamper evident**, meaning that, in general, it's not possible to understand if some source has been tampered with. In other words, it's theoretically possible to create, in a normal scenario, a perfect fake of a digital source. The result we just stated is very important since digital sources of evidence transcend scientificity, hence we might not be able to use them in a court of law. Luckily, it's possible to take proper precautions to build a trace of evidence which is able to highlight when a source of evidence has been tampered with.

Time

Time is very important in digital evidence. Investigators usually look for timestamps in digital sources since they are very useful in a court of law to establish when something has happened. Say for instance we want to understand if a computer was on at a certain point in time to validate someone's alibi. Another example is the assignment of an IP address to a person. By itself, an IP address can't be associated with a single person because the DHCP protocol periodically assigns IPs to different users. However, if we consider an IP address at a point in time, then we can link the IP to a single user. Time is however very hard to handle for computers and it's very simple to destroy evidence based on time. Let us now consider some examples to understand why handling time is hard and how things can go wrong. For starters, say we want to verify if a computer was on at a certain time. One could look at logs (logs store the time at which an operation has been done) and files (i.e., the time at which a file is written). An attacker could however tamper with logs and change the last access time for files (e.g., using the `touch` command). Usually, it's not possible to be sure that a file has not been tampered with, however, let us suppose that logs and file access times can't be modified. Even with this (strong) assumption, we can't be sure that an attacker has changed the BIOS to modify the time (e.g., moving the clock back) before editing the files and then modifying the time back. This means that, in any case, we can't be sure that a system hasn't been tampered with, unless some countermeasures are applied. One way to understand when a computer has done something is by analysing the interaction of said machine with other machines which we trust. For instance, the machine might have talked to a server, which has logged the interaction, hence proving that at a certain time, the machine was on.

In some cases, time is also hard to handle. For instance, Windows doesn't move the clock forward at 2 o'clock when observing daylight saving time but it does it at the beginning of the day, hence for a couple of hours, the time used by a computer doesn't match with the time of other machines (e.g., a Linux server).

3.2.2 Making sources of evidence tamper evident

As we have understood, we can't understand if a source of evidence has been modified, however, we can use methods to make sources of evidence as much tamper evident as possible. This allows us to

ensure:

- The integrity of the sources of evidence.
- The ethical behaviour of all parties.
- The detection of errors in good faith.
- The detection of natural decay of the sources (since cases last for years and drives fail over time).

Hashes

Some of the most commonly used techniques for making sources tamper evident are hashes and digital signatures. In practice, we want to

1. Compute, as soon as possible, the hash of the source of evidence.
2. Every time we use the source, compute the hash of the source and compare it with the hash initially computed.

This ensures that, whenever we use a source, its hash is the same as the one initially computed, hence the stored data hasn't changed. Note that hashes don't have to be stored on the same source of evidence since they could be tampered with and they would change the hash just computed. If we also sign the hashes, then it's possible to store them on the same source of evidence which is signed. This is because the signature is encrypted, hence it's not possible to tamper with it.

Note that it's not mandatory to use hashes in an investigation since without a hash we can prove no theory but we can only say that we are not able to prove whatever theory. In practice, without a hash, we can't prove a theory since we can't prove that a piece of evidence hasn't been tampered with. Namely, without the hash, we can't prove if a source of evidence has been tampered with. This is particularly relevant in some jurisdictions. In the US, if no hash or other method is used for making sources tamper evident, then the evidence has to be discarded. On the other hand, in other countries, such as in Italy, evidence can be used in a court case even if the source is not tamper evident. In other words,

- In the US we have to prove that the evidence hasn't been tampered with to be admitted to the court of law.
- In Italy we have to prove that the evidence has been tampered with to be rejected by the court of law.

Hashes aren't usually computed as soon as the source is collected. A source of evidence it's usually sealed in an evidence bag which is kept sealed until the source is analysed and the hash is computed. Let us conclude with two important comments on hashes:

- **Hashes are not a dogma**. We cannot just dismiss everything since there is no hash. We should always try and understand why is there no hash, if any other measure has been taken and if we can still reconstruct the chain of acquisition. **Hashes are not magic**. Computing a hash does not say anything about what happened before the hashing took place.

3.2.3 Standard procedure for acquiring sources of evidence

When we want to acquire evidence, we usually want to acquire a **bit-stream image**, also called evidence freezing, a clone copy or a forensics copy, which is a bit-by-bit copy of the source. This is because want to copy the whole source, even what isn't content. Investigators usually use forensics distributions of Linux (e.g., Tsurugi or BackBox) since they

- are open source,
- have extensive native file system support, and
- easily access drives and partitions without mounting them.

Let us now consider the process an investigator has to do to acquire a source of evidence making it tamper evident. Usually, we have to deal with drives since they are the devices where data is stored, hence let us consider this context. We have to:

1. Take the drive out of (i.e., disconnect from) the original system.
2. Connect the drive to another computer, equipped with a forensics Linux distribution. When the drive is connected to our computer we should use, in most cases, a write blocker, which is a bridge (usually implemented in hardware) between the drive and our computer that blocks the write operations performed by the computer. A write blocker is used to avoid situations in which the computer unintentionally tampers with the drive.
3. Compute the hash of the source. This can be done with different tools, however, a simple command to do this is

   ```
   dd if=/dev/sda conv=noerror,sync | sha256sum
   ```

   where `dd` is the command for obtaining a bit-stream and `sha256sum` is the command computing a hash.
4. Copy the drive to an image on our computer. This can be done with the command

   ```
   dd if=/dev/sda of=/tmp/acquisition.img conv=noerror,sync
   ```

5. Compute again the hash of the source with the command

   ```
   dd if=/dev/sda conv=noerror,sync | sha256sum
   ```

6. Compute the hash of the clone stored on our computer using the command

   ```
   sha256sum /tmp/acquisition.img
   ```

7. Compare the hashes computed at points 3, 5 and 6. If the first two hashes differ then something must have happened during the copy of the drive.

Note that using the right hash function is critical since we have to use the same hash function for all evidence used in the case. For instance, if the hash of a piece of evidence has been computed before the invention of SHA-256 and MD5 has been used, then we still have to use MD5 even if SHA-256 is more secure. In some cases it's possible to use non-cryptographically secure hashes, however, we shouldn't use functions like MD5 for which it's possible to compute meaningful conflicting messages.

This process works and is very secure, but it has some big issues which become more and more relevant with technological improvement. These challenges are:

- **Time**.
- **Space**.
- **Encryption**.

Time challenge

One of the main challenges we have to deal with when copying a drive and making it tamper evident is time. Say we want to copy a 1 TB drive with a SATA 3 interface which has a theoretical transfer speed of 600 MB/s. Then we need around half an hour per copy (but we need to do three copies since we call `dd` three times). This result is however purely theoretical and drives are usually less good, hence we get a peek write speed of around 100 MB/s and a total copy time of several hours (more than 3 hours). This means that the process of freezing the evidence might require several hours, even if it's possible to parallelise the process with custom hardware which can automatically handle the process. There exist also a command

```
dcfldd if=/dev/sda hash=md5,sha256 md5log=md5.txt sha256log=sha256.txt
of=/tmp/acquisition.img hashconv=after bs=512 conv=noerror,sync
```

for automating and parallelising the procedure. Note that this problem gets worse since drives are getting bigger hence the time needed for copying them increases.

Size challenge

Size is also a challenge since, in some cases, we need to deal with a large number of drives (even hundred or thousand of drives). We can use Storage Area Networks (SAN) or Network Attached Storage (NAS) to deal with a large number of drives and efficiently search data. Images can be moved along the network using the netcat command to setup an host

```
nc -lp 5678 > /tmp/acquisition.img
```

and running

```
dd if=/dev/sda conv=noerror,sync | nc -p 5678 <address>
```

to copy the drive on our computer.

Encryption challenge

Many drives are encrypted, hence simply obtaining the bit-stream of a drive is not enough to get the data. The main challenges we have to face are:

- Some drives are encrypted hence we need the encryption key to actually use the bit-stream. Note that the key might or might not be provided by the owner of the drive. Even if provided with key, performing acquisition in a repeatable way is sometimes challenging.
- Some drives are tied to the motherboard and aren't readable without the motherboard itself. In this case, copying the drive is useless.

3.2.4 Other procedures for acquiring tamper evident sources

The process described above is secure however it has to face different challenges and it can't be applied in every situation. We'll now describe some procedures to handle source acquirement when it's not possible to use the standard procedure.

Booting from live distribution

In some cases, it's not possible or we don't want to disconnect the drive from the computer. Some examples are drives linked to the motherboard or drives using RAID (since RAID drives are handled by a specific RAID controller without which it's impossible to use the drives). Let us focus on the problem of RAID disks and let us then focus on a more general solution. RAID drives are managed by a RAID controller and, since there exist a lot of them, it's hard to replicate the controller used by a specific drive. In case we have some bit-stream of SATA drives, we can give them to a piece of software that tries every possible RAID firmware to find the one used for the specific RAID drive input. This process is very lengthy but it works. Another solution is to copy the RAID firmware, if possible.

In general, if we can't disconnect the drive from the computer, we have to boot the computer to which the drive is connected with a forensics Linux distribution. Note that we want to use a forensics distribution since it doesn't automate drive access hence we can't unintentionally tamper with the target drive. Booting a different operating system with respect to the original one is not easy but the goal is usually to boot the machine and end in the BIOS, possibly immediately disconnecting the drive as soon as the BIOS is accessed. As soon as we can boot our forensics distribution we can then apply the standard procedure for copying the target drive (skipping the first two steps).

Analysing a live machine

As for now, we have considered devices which are already off. This isn't however always true since we might have to deal with machines that are on. This is a problem because a machine which is already on might be changing the evidence we are acquiring, namely it might be tampering with it. This is not a good thing since we'd like to preserve as much as possible the data stored on a device as it was when it was seized. When we deal with a live machine we have to ask ourselves if we can turn it off. In some cases, in fact, a machine generates some services that can't be turned off (e.g., a machine that handles the billing process of a mobile provider). We should also understand if it makes sense to turn the machine off in fact it might be of no use to shut down a computer. When a machine is on, a great deal of useful information (e.g., an encryption key) is stored in memory and, if we turn the machine off, such information is lost and we might not get it back.

In general, when dealing with a live system, we want to acquire at least some data before turning the system off. Data should be acquired in volatility order (i.e., memory before storage) because volatile memories change more frequently, hence we want to dump a state which is closer to the moment we want to analyse.

When working on a live machine, especially if we consider a Windows machine, we might not have the tools to dump the memory. This means that we should install some external software to acquire some evidence. This is a key choice and it usually requires some trade-off because a program might modify the storage and the memory we are trying to dump. This means that we have to balance the amount of software we want to install since:

- Installing too many programs might modify the drive too much.
- Without some programs we can't dump memory and storage.

In most cases a live machine is analysed without turning it off but, if we think that it's better to turn it off, we have to choose between:

- Shutting it down properly.
- Pulling the plug.

Once again we are in front of a trade-off because:

- Pulling the plug destroys caches and might corrupt the drive hence making the machine not bootable. On the other hand, this allows preserving the state of the files since the normal shut-down process can modify the storage (whereas a brutal shut-off doesn't).
- Shutting the machine down properly ensures that the machine can be turned on in the future but it might modify the state of the drive, hence modifying the evidence.

If we don't need to turn the machine on in the future, shutting it down improperly might be a good solution for forensics purposes since it preserves the state of the drive. If it's required that the machine boots after turning it off the only choice is shutting it down properly.

Independently from the fact that a machine is shut down or analysed when on, every action taken should be properly documented. This means that every time an action is taken, we should annotate the operation performed, the time at which such action has been taken and the reasons for which a certain choice has been done. This should be done even with non-live machines however it's particularly useful in this context because every action we take causes a change in the machine. We can say that we become part of the analysis since a particular operation might tamper with the evidence. Because the evidence that we provide has to be used in a court of law, if we document every step of its acquisition, we make the jury (or the judge) aware of the operations that have been done, which makes the whole process a scientific process (because it can be validated by others).

As we have understood, live analysis is way harder however it is quite rare in courts of law.

Live networks

The techniques we have described above are used for single machines however even networks of machines can be sources of evidence. When analysing a network of machines, which is a live network (otherwise the machines wouldn't be connected), we can find ourselves in two situations:

- **Something has already happened**. In this case, we can only observe what happened (or what the intruder hasn't deleted) and the only thing we can use is **log files**.
- **Something is happening**. In this case, we can observe logs but more often we observe network traffic. If we are observing the network while an intruder is attacking it we have to take into consideration that he or she might realise that we are observing and delete the evidence of what he or she has done.

In both cases we also have to consider that one or more machines might have rootkits installed, hence the tools on the machines might not give reliable results. This means that if we observe the logs or the network traffic from one machine of the network we might be using tools of a rootkit hence we might not be seeing the truth.

New scenarios

Together with the aforementioned scenarios, new technologies are bringing new challenges to evidence acquisition. Some fields which pose interesting challenges, which will be analysed later on, are:

- Cloud forensics.
- Mobile forensics.
- SSD forensics.

3.2.5 Acquisition of evidence on solid state drives

Introduction to SSDs

Since we are going to talk about SSDs, it's better to refresh how do SSDs work and how they differ from mechanical drives (especially hard disks).

SSDs are built using NAND-based flash technology (the same used for flash memory) and are extremely faster than hard drives, however, they have to face some challenges:

- SSDs have been used in mobile phones since forever and are recently getting used to replace hard drives in computers. This is a very relevant fact because, if we want to use SSDs as a plug-in replacement for HDs, we have to ensure that both technologies offer the same interface because a computer has to be able to talk to both (imagine we want to replace an HD of an old computer with an SSD). Namely, an operating system doesn't have to bother whether it's interfacing with an SSD or an HDD.
- SSDs have a limited lifespan which is around 10000 program-erase cycles. This means that we should be able to write the SSD as evenly as possible. For instance, some files are written once and never modified (e.g., a document) whereas others are frequently rewritten (e.g., a log file). In this situation, the memory cells containing the former files are never touched but the cells containing the latter type of files are frequently written. As a result, some cells are written a lot while others only a few times resulting in an uneven wear of the drive. To solve this problem drivers usually try to balance the writes, namely try to relocate files so that the drive can be written evenly.
- SSD memory cells are collected in blocks and, when we want to write a cell, we have to blank the entire block and completely rewrite the block, even if we want to write just one cell. Writing a block is quick but blanking it isn't.

To solve the problems we have just mentioned, manufacturers have added a chip, called Flash Transition Layer (FTL), that mediates the interaction with the operating system so that the OS can talk to an SSD as if it were talking to an HDD. The FTL handles the complexity related with write balancing and block blanking. When writing a cell in block B, since blanking is expensive, the FTL collects the cells that are in semi-empty blocks and writes them in B. This operation levels out the wearing of the drive and allows to reduce the number of blanks. In general, an FTL handles:

- **Write caching**.
- **Trimming**. Trimming is the process of blanking unused blocks while the drive is not used. Trimming is requested directly by the operating system (which has to support trimming) that tells the drive to delete some space which is not required anymore. This has a huge implication in file recovery since when using SSDs, differently from HDDs, the operating system is encouraged to blank some blocks hence the files in such blocks are completely deleted and not recoverable. Note that trimming isn't executed immediately when requested by the operating system but as soon as the drive is not used.
- **Garbage collection**. Some drives are able to do garbage collection without the intervention of the operating system. This feature rarely works.
- **Data compression**. Some drives can lower the wear of cells by transparently compressing the data so that it can be written in fewer cells.
- **Data encryption or obfuscation**. Since a drive might perform compression, it makes sense to also encrypt data. Note that, even when we do not encrypt data it might be hard to recover what's on the drive without the FTL since the FTL's logic might be very complex and most vendors keep it secret. The reason for keeping the FTL implementation secret is that the chips mostly use the same technology, hence the FTL is what differentiates a good SSD (i.e., with good performances) from a bad one.
- **Bad block handling**. In drives, since cells have a limited life, the FTL already knows that some cells will fail and will eventually have to deal with them.
- **Wear levelling**.

As a result, on an HDD we can access a specific sector and we get what is physically in that sector. On the other hand, on an SSD, because of the FTL mediation, when we want to access a section of the drive, the FTL reads wherever it has been written in that piece of the drive. Namely, the FTL hides where the section has been placed on the drive. This means that accessing the drive without the FTL is a mess since the FTL might have mapped and spread different parts of a section to cells in different blocks. This is a problem because forensics tools usually access raw data on a disk.

Since the FTL is the main barrier to an SSD analysis, we might want to bypass it, however, it is not an easy task. For starters, we can't completely disable the FTL (even if there exist comments to disable some features like caches) since it is not an optimisation but a need because the SSD wouldn't work without it. We could in theory extract the chip from the SSD (i.e., separate the memory chips from the FTL) and attach them to a custom board to extract the data on the chips but this operation is very expensive (the chip expects a very precise behaviour from the FTL, hence it's hard to replicate such behaviour) and might only partially recover data. Even if this process worked, we would get data which is probably encrypted, compressed and spread over different cells, hence difficult to work with. As a result, it's very hard to reconstruct practically usable files.

Impact of SSDs on forensics

Since it's hard to recover data from SSDs, we should understand what factors impact on SSD forensics analysis more. For starters, we should understand what can impact SSD analysis. The main influencing factors are:

- **Trimming**. Trimming is clearly a big problem for forensics analysis since the operating system can efficiently blank some blocks, which definitely deletes the data in that block. Note that, a drive will trim data as soon as it has nothing to do, hence if we connect the SSD to a write blocker (which is something we should do when using a normal HDD), the FTL will immediately trim data since the SSD can't be written, hence it must not be used. Note that the write blocker prevents the OS to write, but not the FTL. Moreover, the FTL might answer to a write with a different value every time. For instance, some FTL for USB drives (which are basically the same as SSDs') return a random number when the OS tries to access a bad block, hence the hash of a drive might be different every time we compute it.
- **Garbage collection**. Some SSDs can trim blocks even if the OS doesn't tell them so. This is even worse than OS-driven trimming since we have no clue when blocks are trimmed.
- **Erasing patterns**. Some SSDs don't trim the entire drive but just some parts, for unknown reasons (maybe some programming errors).
- **Compression**. Even if we are able to obtain the blocks of an SSD, it's very hard to obtain the actual files if the data is compressed since we don't know how it has been compressed (because we don't have the FTL).
- **Wear levelling**. Some theorised that wear levelling could be an advantage for forensics analysis since data is copied (this phenomenon is called write amplification), hence we have multiple instances available.
- **Files recoverability**.

The paper A Comprehensive Black-box Methodology for Testing the Forensic Characteristics of Solid-state Drives tests these factors to understand which influences forensics analysis more. SSDs more than often use caches which can be disabled, however, some FTL use caches that can't be disabled. To solve this problem, SSDs have been tested with files that are bigger than the cache available to the FTL so that such files can't be cached by the FTL.

Let us now understand, for each of the facts above, how they can be tested to understand if an SSD uses them and how much they impact the analysis:

- **Trimming**. Trimming is tested by:

  1. Formatting it.
  2. Filling it. We should even fill it at different percentages since some vendors might decide to trim only when there are not enough blocks left on the device.
  3. Deleting some files or performing formatting and testing how long it elapses before getting zeros instead of the files we have previously written.

  Note that the FTL might even return zeros even if a block hasn't actually been trimmed (for reasons). For us, this is like having trimmed the block since we can't bypass the FTL. The paper we are considering showed that trimming is usually activated after 1 to 10 seconds from when the file is deleted by the OS. One analysed drive showed that the fuller the disk was, the more aggressively the FTL trimmed files. Moreover, the trimming also depended on the operating system since it's the OS that tells the FTL that some files have to be trimmed.

- **Garbage collection**. Garbage collection can be tested by disabling trimming and checking if files were trimmed autonomously by the FTL (with the same procedure used for testing

trimming). The drive analysed in the paper (which is however 10 years old, hence things might have changed) showed that no drive can actually do garbage collection since it's hard to do it reliably. This makes sense because garbage collection would require the FTL to precisely guess what the OS wants to do with some files and delete files without the possibility to recover them if the guess is wrong.

- **Erasing patterns**. Some drives trim the drive in stripes (i.e., only some sections are trimmed) instead of doing it completely. The reason for this behaviour is unknown.
- **Compression**. We can check if compression is performed by writing high-entropy and low-entropy files. If we have a difference in the time required to write such files we can infer that the drive uses compression since writing takes time and low-entropy files can be compressed more than high-entropy ones, hence it should take less to transfer the former than the latter.
- **Wear levelling**. Testing for wear levelling is harder since we can only see what the FTL shows us but not what actually happens on the physical side. This isn't however an issue since wear levelling is performed by any FTL, otherwise, a drive could last very little. Even if we don't have to test for wear levelling, we still have to understand if wear levelling can help in forensics analysis. To understand if the write amplification effect can help us we can

  1. write unique patterns,
  2. dump the drive and,
  3. check if the patterns are repeated somewhere else in the drive.

  Even if researchers have done multiple tests specifically to trigger wear levelling (e.g., repeatedly writing on the same block), they couldn't notice the write amplification effect. This doesn't mean that wear levelling isn't performed but that the FTL might hide it.

As a result, the file recoverability of a drive with any of the characteristics above is 0, except for drives that trim files in strides in which case recoverability is more than 0. Long story short, if we have to analyse an SSD, except for extraordinary situations in which the chip code is buggy or something is wrong, we can't recover any file that has been deleted.

Also note that FTL chips can be upgraded but usually not downgraded. Moreover, the trimming process depends not only on the FTL version but also on the OS and driver version, hence the combination of the exact setting in which a drive is working is usually hard to predict.

3.2.6 Cloud forensics

Cloud computing is swiftly growing and is widely used by many companies. There exist many different cloud computing paradigms but we'll mainly consider the Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) paradigms. The main difference among these paradigms is the amount of hardware and software offered to and controlled by the user. Moreover, we will only consider public clouds like AWS and Azure but not private clouds. This is because in private systems we control the whole infrastructure, hence we can control every aspect of the system. When dealing with Cloud forensics we have to face different challenges in most of the phases of the analysis.

Acquisition issues

Depending on the Cloud paradigm we consider, the user might have more or less control over the resources and the infrastructure. In order, IaaS gives the user the most control since he or she can access the whole infrastructure, then we have PaaS and finally SaaS where a user can only use the software and can't access what's behind it. For instance, if we consider logs, we have that:

- **IaaS** offers logs until the OS level which is accessible to the customer and network and process logs at the provider level (e.g. load balancer logs).
- **PaaS** offers application logs (possibly), network logs, database logs, or operating system logs depending on the service provider.
- **SaaS** offers no logs since the user only controls the application but not what is behind it. In this case, we can't even get what is on the guest virtual machine.

Data acquisition can be done on the guest virtual machine but not on the host. This means that when we have to acquire a virtual machine, we can only get what is actually offered to the client. For instance, if we want to dump a drive, we can only dump what is actually saved on the system and we can't recover what has been deleted. This is because a virtual machine isn't mapped to a physical device but is allocated a resizable space which is enlarged or shrunk whenever a file is added or removed. As a result, when something is deleted, space is revoked from the virtual machine virtual drive, hence nothing can be recovered.

Another important difference with normal forensics is that data stored in the cloud is not at rest (i.e., deposited on a drive) but it's in flux, namely it only exists when we perform a transaction. When we recover drive images we recover data at rest (because it's stored and not involved in a data transfer) but when dealing with the Cloud we are more interested in data movement. In short, data flux means that something is built on the moment, for a specific user hence it's hard to reconstruct what has been shown to a specific user at a certain time. For instance, when we watch the comments under a YouTube video, we aren't looking at a page which is stored somewhere on a drive but we are looking at a page which is composed using different components coming from different parts of the YouTube infrastructure. This is a problem because it's easy to handle and analyse data at rest which is more stable and rarely changes, whereas data in flux is more volatile and harder to analyse since some association between different data can change or be deleted. This problem is quite relevant in courts since we might be asked to say what content has been presented to a user but, since data is in flux (i.e., composed on the fly and at the moment) we can't simply answer by printing out that content since the user might have had a different representation. This is because, for instance, a web page is made of different components dynamically loaded from different sources which might change the look for different users.

Note that the volatility of resources on the Cloud isn't just a matter of not being able to access some data (which is hidden by the provider) but also a matter of how Cloud services dynamically handle resources. Because resources are dynamically allocated, even with the collaboration of the vendors, we might not be able to get some data since it has been actually deleted or it's impossible to trace where it has been used (i.e., to which client it has been served).

Attribution issues

In forensics, we usually want to understand who has done something. In general, the acquisition of evidence is hard because of spoofing and stepping stones (i.e., mounting an attack from a machine

distant from the actual attacker). Namely, the attribution to the technical source is not the attribution to the agent. When dealing with the Cloud things get even worse since we add a layer of indirection. For instance, say an attacker uses a machine on the Cloud with a specific IP address. Later on, that address might be assigned to a different machine making it impossible to understand the source of an attack since we can see only the current IP-agent association, but not the one at attack-time. The only people that might give us the IP-agent association at attack-time is the provider of the service which has to collaborate and have the data available (since it might have been already deleted).

Legal issues

Different countries have different laws, hence what is legal in one country might not be in another. Moreover, in the physical world, it's always possible to understand where something has happened (i.e., where to do search and seizure). That isn't true for the digital world, especially when considering data in flux. If a resource is the result of a transaction that composes components coming from different parts of the world, it is hard to determine where the transaction took place. The Budapest Convention established the concept of **electronic search and seizure** to solve this problem. Electronic search and seizure allows investigators to do search and seizure from the place where the resource has been requested if the location of the searched component is available from the place where the resource has been requested. This allows investigators to remove obstacles when doing search and seizure but also doesn't solve all problems because the removal of obstacles (i.e., the legal option to forcefully access a system) can't be ordered across countries. For instance, a prosecutor in Italy might order a search and seizure on a system that is in Norway but it would be a crime in Norway. Namely, some things work internationally and others don't, especially on the Cloud.

Nested clouds

Many companies that offer Cloud services use other Cloud services (i.e., from other companies) themselves. This adds another layer of complexity since we might have to search files in different legal systems and in multiple places.

Forensics-driven Cloud

Some Cloud providers offer guarantees that help forensics analyst do their job. For instance, some providers:

- Make an effort to store snapshots of volatile virtual machine data in their infrastructure.
- Make an effort to provide proof of past data possession.
- Try to provide information about the location of data.
- Offer identity management services.
- Offer encryption and key management services.
- Have a legal provision and SLAs.

Cloud for forensics

Cloud services are also used by forensics analysts in fact they provide a lot of storage space and powerful tools to analyse the evidence. Even if the Cloud could be very helpful, it breaks the chain of custody since files are stored on a drive which is not controlled by the analyst, hence we can't ensure how it's preserved.

3.3 Identification of evidence

After having acquired the sources of evidence, we have to identify the evidence contained in the material we have acquired. This means applying computer science to whatever evidence to extract some information relevant for a court of law.

Some information might seem obvious to an expert, but not to the jury or the judge. In this sense, we have to remember that an expert witness has to speak for the discipline he or she is an expert in and answer the questions regarding that discipline, independently of how big or complex such questions are.

3.3.1 Tools for the identification of evidence

Before analysing the challenges related to the identification of evidence, let us define the setup and the tools required for this process. Let us consider a dump of a hard drive and let's say that we have to answer some questions regarding the content of said drive. When operating on a dump we usually want to use a **forensics Linux distribution** because:

- It's **open source**, hence everyone can check that the tools do what they should hence everyone can replicate the analysis done on the dump (which is required to have a scientific process).
- The drive might use any filesystem and **the Linux kernel can basically handle any filesystem**.
- It's **easy to mount files as drives**.

Linux is used to manage the access to the drive however it might not have the right tools to analyse the data on the drive (e.g., a file might be opened by a program that works only on Windows). To solve this problem we can use virtual machines that run the operating system required for opening the files on the drive. Note that virtualising the operating system is fundamental, especially in the case of Windows (which writes to a drive as soon as it's mounted), since we don't want it to write the drive and tamper with it. If we use virtualisation, we can use Linux and Samba to show the drive as read-only hence stopping Windows from writing the drive. In some cases, tools don't work if they can't write on the drive. To handle such cases we have to use a virtualisation software that records the writes to disk such that the guest OS thinks it has written something to a file, however, the hypervisor discards the writes. It's particularly important to keep the drive unchanged because:

- The scientific analysis has to be repeatable, hence the evidence can't be modified by the analysis itself.
- The experiment has to be described.

To make sure that these two principles are respected we should use open-source (and possibly free) tools which allow every other expert to validate the tools used, the experiment and its results. If

no open-source tool exists for a certain operation, we should use tools which are broadly available. Moreover, if the tool is not open-source, we should also provide a way to check that the result of an experiment is correct. Say for instance that we have a proprietary tool for searching child pornography content in a huge drive. At the end of the analysis, we have a set of folders where such material is located and we can check if the folders effectively contain the images we were looking for. In this case, we don't really know how the tool works but we can verify the result, hence it's fine to use it. Also note that if a tool extracts results that we can't explain, the results can't be used as evidence on their own and the tool can be used only to help in the process of obtaining evidence. A classic example is machine learning. Machine learning models are usually hard to interpret, hence the results returned can only be used as an aid in the investigation.

3.3.2 Recovery of deleted data

Evidence can be identified using everything which has to do with computer science, however, some techniques are very specific to digital forensics and are typically not studied in other contexts. Recovery of deleted data is the most important example. Data can be deleted from a disk:

- Intentionally (even not to hide evidence).
- Because operating systems delete some files when they are not used for some time.
- Because of failures.

In all of the cases above it could be possible to recover some files (or parts of some files) however the process is purely random and based on luck. Data recovery works because of the way in which files are handled by operating systems, hence it's good to understand how files are represented by an operating system (Linux in particular). A storage device is divided into blocks, each of which contains some data and blocks are pointed to by indexes. Blocks are accessed using a data structure called `inode`. Each file or folder is associated to an inode which stores some metadata and pointers to the blocks of which the file is made. More precisely a pointer in an inode can be:

- Direct if it points to a block.
- Indirect if it points to a list of pointers to blocks.
- Double indirect if it points to a list of pointers to list of pointers to blocks.
- Triple indirect if it points to a list of pointers to list of pointers to a list of pointers to blocks.

When a file is deleted, the operating system:

1. Writes that the file is deleted in the inode. This means that the file entry in the file system is flagged as deleted but the file is still in the storage. After this point, a file can be undeleted by simply removing the flag.
2. At some point in time, in random order,
    - Rebalances the file system removing all inodes referring to files that have been deleted. Namely, the file system entry will be removed and the file system structure is rewritten or rebalanced. Until this point, we can find the file metadata written in its inode.
    - The actual blocks (once) allocated to the file will be overwritten with other content. Until this point, we can retrieve the actual blocks on disk. After this point, the old blocks have been overwritten by other content and can't be recovered.

In many cases, the loss of metadata precedes the loss of the file (i.e., the overwrite of the file) but this is not a given and we can't be sure that metadata is deleted before the file is overwritten. Depending on what operation has already been executed we can obtain some files, parts of some files or nothing. In particular:

- If the inode has only been marked as deleted (i.e., only step 1 has been executed), then we can obtain the deleted files by marking the files as not deleted.
- If the metadata is still in the inode we can follow the pointers in the inode to build the file back. More precisely,
  - If some blocks have been overwritten we can reconstruct only a part of the file.
  - If all blocks have been overwritten we know that a file existed but we can't obtain the file back.
- If metadata has been deleted it's harder to tell if a file existed and we can only try to scan the whole memory looking for files.

Structure of disks

Mechanical disks are made of multiple disks stacked one on top of the other and each disk is divided into concentric tracks. The minimum arc that can be read by the magnetic head of the disk is called the sector. The operating system normalises sectors using groups of sectors called clusters. This is done because different disks have different sector sizes, hence if we take a cluster with a size that is a multiple of the most common sector sizes, then we can write to a cluster independently from the sector size (e.g., with a cluster of 8 kB, we'll write 2 sectors in disks with sectors of 4 kB and 4 sectors in disks with sectors of size 2 kB). The operating system allocates files cluster by cluster, which means that:

- A file smaller than a cluster still occupies a cluster.
- A file might occupy multiple clusters. Usually, the size of a file isn't a multiple of the size of a cluster which means that the last cluster is partially filled. The space at the end of the last cluster which is not written is called **slack space**.

Slack space recovery

Slack spaces are portions of a cluster which are not written since the file (or the last part of the file) is smaller than the cluster. These spaces can be used to recover some data since when a file is deleted its content isn't really deleted, hence slack spaces contain fragments of old data, i.e., of the file that was on that cluster before. Slack spaces are very useful to recover very old data, which is very useful. Unfortunately, slack spaces are usually small in size and might contain very little information.

Carving

Let us assume that the metadata related to some file has been lost but the file hasn't been overwritten yet. This means that we don't have the pointers to the blocks that compose the file but the file is still in memory. What we can do is exploit the fact that files usually have a header and a footer, hence we can

1. scan the whole drive,
2. as soon as we find a header, we look for the corresponding footer, and
3. we interpret all the data in between as a file.

This process works if a file is stored in adjacent clusters and it's not fragmented. This isn't however a big problem since, in the past, operating systems tended to fragment files but they they try to keep files in adjacent clusters and are only rarely split in two fragments (and almost never in more than two fragments).

If files don't have headers and footers there exist techniques, usually based on machine learning models, to determine the content of a file from the content of a block. This is usually hard but doable. Moreover, most of the file formats we are interested in have headers and footers.

Things get more complicated when we have to deal with header-less encryption and compression. In these cases, it's impossible to understand the content of a block since it looks like random noise (by definition of encryption). It's also impossible to understand where a file starts and the file format since we don't have a header. These cases are very rare but if it happens there is no way of recovering the data.

3.4 Antiforensics techniques

Criminals are aware of the fact that analysts can recover data from drives, hence they try and hide their actions. Antiforensics techniques are ways to contrast forensics techniques. This means that antiforensics techniques aim at

- creating confusion among the analysts, and
- defeating or bypassing the techniques and the tools used to recover data from disks.

Antiforensics techniques can mainly be split into two categories:

- **Transient antiforensics techniques**. Transient techniques make the job of an analyst more difficult but can be bypassed. In particular, if an analyst knows that a certain transient technique is applied, he or she can avoid, reverse or bypass it.
- **Definitive antiforensics techniques**. Definitive techniques make the job of an analyst impossible since they destroy evidence, hence they can't be bypassed or reversed.

These techniques can be mapped one-on-one to the first two phases of forensics analysis, namely acquisition and identification. More precisely:

- **Definitive tools are used to make the acquisition phase more difficult** since a technique that doesn't allow obtaining evidence makes it impossible to extract knowledge from the evidence, hence the evidence is lost. This means that the effect of an antiforensics technique at this phase is definitive, hence the tools are of the definitive type.
- **Transient tools are used to make the identification phase more difficult** since, even if some evidence has been modified, we still have it hence we can try and obtain the original data. Basically, we have the evidence but it's hidden, hence we can try to get it back.

Antiforensics techniques are applied to the first two phases of a forensics analysis since the last two (i.e., evaluation and presentation) are human-driven whereas the former are more automatic and machine-driven. Let us now consider some examples of both classes of techniques.

3.4.1 Definitive techniques

Tampering with the timeline

Operating systems have many tools to change the time at which a certain file has been accessed and criminals can also modify the time of a machine. A timeline allows us to

- understand at what time a certain file has been modified, and
- build a history of the modifications performed on the system.

In practice, a timeline gives another view of the filesystem by taking the files in the filesystem and ordering them by their access time. The closer the beginning of the timeline is from the current time, the more precise the timeline is.

For each file, the filesystem stores three values, called MAC, (four in case of NTFS, called MACE) in the file's metadata which allows to define when a file has been accessed:

- **Modified**. The modified value stores the last time at which the file has been modified.
- **Accessed**. The accessed value stores the last time at which the file has been accessed.
- **Created**. The created value stores the time at which the file has been created.
- **Entry changed** (only in NTFS). The entry changed value stores some metadata about the metadata as a security measure to ensure that the metadata hasn't been changed.

Since these values are stored in each file's metadata, the hash of a file is the same before and after modifying the MAC values but the hash of a disk is different before and after. All the MAC values are editable and the modifications are overridden, hence the old values are lost forever and this is the reason why this technique is of the definitive type. Usually, a criminal might want to change the M value to the C value so that it looks like the file hasn't been accessed. This scrambles the timeline and makes it hard to build an accurate representation of the modifications to the system. Long story short, we always have to keep in mind that dates on a file might have been changed, hence it could exist an artefact that let us say that something might have changed but we don't know if the change is legitimate or not. An example of a tool used to modify the MAC values of a file is `touch`.

Since timeline tampering techniques are related to the fact that a user can change the MAC values of a file, one might think that it'd be better to build OSes that don't allow editing the MAC values. It's in theory possible to build such an OS however it wouldn't bring that many advantages in fact:

- One could attach the disk to another OS, which allows modifying the MAC values and modifying the metadata on that OS.
- Not allowing users to change the MAC values means that the OS has to change such metadata. However, an admin should still have the same privileges as the OS, hence an attacker that can obtain admin privileges can still modify the MAC values.

This means that an OS that doesn't allow users to modify the MAC values can be built but the restrictions implemented can be bypassed.

Countering file recovery

Countering file recovery techniques aim at making it impossible to recover deleted files (i.e., data blocks that haven't been overwritten yet) from disk. There exist many techniques to achieve this goal but the most important are:

- **Secure file deletion**. Secure file deletion tools overwrite a file with all zeros before deleting its metadata. This means that it's impossible for an analyst to recover a file since it has been completely overwritten with all zeros. When using secure file deletion tools we always have to remember that there exist some fake tools that pretend to securely delete a file but that simply delete the file (without overwriting it).
- **Wiping unallocated space**.
- **File encryption**. If a file is encrypted, securely deleting the encryption key makes the file completely useless since it will look like noise (by definition of encryption) that can't be decrypted.
- **Virtual machine usage**. Virtual machine drives aren't actual drives but are represented as files that allow to dynamically allocate some space when needed. However, since files aren't actual drives, data remnants might not be there.

Countering file recovery techniques can be taken to an extreme. An example is reading the residual of magnetisation. Many years ago, physical magnetic drives weren't very dense, hence it was theoretically possible, as stated by Peter Gutmann in his paper Secure Deletion of Data from Magnetic and Solid-State Memory, to read the magnetic field on the single cells of a magnetic disk and, depending on the value read (of the magnetic field), understand if the previous value was a 1 or a 0. To avoid this type of analysis, governments have developed secure file deletion techniques that overwrite files with well-studied patterns that equalise the magnetisation levels of each cell such that it's impossible to recover the previous values. These files are overwritten multiple times to ensure the best security level. Nowadays, physical drives are so dense that it's practically impossible to read residuals of magnetisation hence multiple-passes secure deletion techniques are useless. Nevertheless, these techniques are still applied in some fields (e.g., military and government secrets), even if useless. This is an example of security inertia.

Fileless attacks

An attacker can be detected if he or she does some modifications on the disk. Fileless attacks allow the execution of some tasks (not in the sense of OS tasks) without leaving evidence of the hard drive and doing everything in memory (unless the attacker explicitly requires modifying the hard drive). For instance, Metasploit's meterpreter allows us to inject a program into a process memory space, hence we can have something like a shell inside a program without leaving evidence on the hard drive. Since fileless attacks only work on the memory, when the computer is shut down, everything is lost. This shows how important it is to analyse a target machine (i.e., do a memory dump or memory analysis) before shutting it down and acquiring the hard drive.

3.4.2 Transient techniques

Filesystem Insertion and Subversion Technologies

The Filesystem Insertion and Subversion Technologies tool collects a set of technologies that allow placing files where they should not be. In this way, an attacker can place some files he or she wants

to hide in places where an analyst wouldn't look. For instance, in partition tables, there is space for around 32 kilobytes of data where standard tools won't look for files. There also exist EXT2 or EXT3 filesystems that, once mounted on a system, can hide some files. Some examples are:

- **RuneFS** that marks the blocks containing the files to hide as bad blocks, hence blocks not to read.
- **WaffenFS** that leverages the differences between EXT3 and EXT2. In particular, the former uses around 32 megabytes of journaling data in a partition whereas the latter doesn't. When EXT3 journaling data is added to an EXT2 partition, it is ignored, hence we can hide there some files.
- **KY FS** that uses directory inodes to obtain unlimited space.
- **Data Mule FS** that puts data in padding and metadata structures of the filesystem ignored by forensic tools (up to 1MB of space on a typical FS).

In this case, the data is hidden but is still on the disk, hence an analyst, which knows that such a technique is applied, can recover the hidden files. For this reason, the techniques for hiding files are transient.

Note that these techniques are rarely used and it's possible to build automatic tools that detect if some file is hidden in places where it shouldn't be. These tools aren't however very popular since the attacks aren't popular either.

Log analysis

Logs aren't usually analysed by hand since it would be too long. Instead, we usually:

- use regular expressions or
- signatures

for recognising well-known patterns and messages. These patterns can be either ignored (if we search for patterns that we know do not generate errors, hence we want to remove them to focus on the errors) or kept (if we search for patterns that contain the event we are looking for). This means that an attacker can do some log injection to break the regular expression or the signature and make the job of an analyst hard since he or she has to manually analyse the logs. Let us consider an example to better understand how log injection works. Say a login system logs every user that tries to access the system with a string containing the name of the user and a phrase saying if the login is successful or not. We might want to look for an event where a user has failed a login for five times (meaning that the user is maybe trying to bruteforce the password). If an attacker injects a username with a carriage return in the middle, the regular expression used to find the five log entries breaks (since the carriage return isn't usually part of a username, hence it's not included in the letters searched by the regex) and we do not notice that the attacker is trying to break into the system. Other examples are available here (`https://owasp.org/www-community/attacks/Log_Injection`).

Once again, the data is still on the logs and one can always find it but it's hard. This shows why log analysis is a transient technique.

Partition table tricks

Partition tables can also be used to hide some data. There exist multiple ways to hide data using partition tables but the most important are:

- **Partitions not correctly aligned**. Partitions might overlap (which can also be a bug and happen not for malicious reasons). This can be used to hide some data in the overlapping region such that forensic tools can't find it. Using a partition restore tool we can read them, but they may escape a forensic analysis.
- **Adding multiple extended partitions might generate errors**. Windows and Linux manage them but many forensic tools don't.
- **Generating a high number of logical partitions in an extended partition might not be handled correctly**. In particular, some OSes might stop recognising partitions after reaching a certain number, hence we can store files in the partitions that are hidden. For instance, Windows calls drives starting from C and reaching Z. When we reach Z, the next drive is called AA, then AB and so on. In some cases, however, when the name Z is reached, no more disks are mounted, even if the partition exists, hence the files on the partitions after the one called Z are hidden.

In any case, the files are on different partitions, hence, if the analyst knows the technique applied, he or she can find them. That's why partition table tricks are transient techniques.

In general, it is enough to make it difficult to use some tools to make the analysis of a disk harder. In particular, an attacker only needs to make it hard for an analyst to use his or her favourite tools. If an analyst has to analyse many drives and can't use his or her favourite tools, then obtaining evidence might require a lot of time and he or she might even find only a part of all the hidden data.

3.5 Evaluation and presentation

The last step of forensics analysis is the evaluation and presentation phase.

Evaluation means matching the evidence elements (i.e., the facts we have established) to support a legal theory. Note that a legal theory is not a scientific theory. Evidence elements are used to demonstrate that a certain law has been broken. For instance, in Italy, the law punishes those who *willingly detain child pornography (i.e., videos or photos of people under the age of 18 involved in a sexual act)*, hence we might have to prove that a certain drive contains or not such material. Computer science isn't always enough, in fact, there might be cases in which we have to ask experts in other fields to prove some claim (e.g., it's not evident if the person is a minor). More precisely, an expert in a court, during the evaluation phase, can say if:

- The disk contains images.
- The images are pornography images.
- The person involved is a minor (with the help of another expert).

These are all facts and can be scientifically proven by the expert. The law however specifies that such material should be willingly detained, hence we also have to prove that part. This could be much harder because we have to prove the will of a person. The only thing an expert can do is to give the judge facts that might lead him or her to think that porn material has been willingly held. Some examples are access dates and the location of the material (if the file is in a temporary folder

used as a cache, the user might not have accessed such material but if the file is in a personal folder with a specific name, we might say that the user has willingly handled such material).

In any case, we, as experts, only have to give the judge some elements that he or she can use to pass the judgement (e.g., say if a hard drive contains some type of images, where such images are and when they've been accessed). In particular, an expert has to evaluate:

- **Elements to support the indictment**.
- **Alternative explanations**. An expert can also be asked to give an alternate explanation of the elements given to the judge. Say for instance a pen drive with some pornography material is found on someone's desk. He or she might be the holder of such material, or an alternate explanation could be that that person has found the drive in a parking lot of the place where he or she works and taken it to his or her desk waiting for someone to reclaim it. Another example could be the photo of bruises on the body of a child. One could think that it surely is a case of child pornography however that photo could have been taken by one of the parents to show that the other parent is violent.
- **What can be said, what can't and what requires more experimentation**. Some evidence is available to the expert for analysis but, in some cases, the expert is only aware of the fact that some evidence has been deposited without being allowed to analyse it. It's also possible that we don't know if some evidence has been analysed yet. Say for instance that a computer has been seized and we are working for the owner of the computer. Also assume that the computer has been (correctly) analysed by an expert chosen by the judge and some material has been found and correctly reported. Since we are working for the owner of the computer, we might be aware of other material which hasn't been found by the expert who analysed the computer. In this case, we can ask to extend the analysis but it wouldn't be a good idea. If the computer contains also some material that we don't want to be found, it's better not to ask to do another analysis. Basically, we have to evaluate, given that some material has been reported to be found, if it's worth asking to do a more in-depth analysis. In other words, we have to understand if proving something (by asking for a more precise analysis to find helpful material we know it's there) is more advantageous that taking the risk of revealing dangerous material. Moreover, we have to consider the fact that something might have been found but not reported yet.

3.5.1 Relationships with other parties

An expert doesn't only have to deal with the technical part of the process, but also with the law part. In particular, he or she has to talk with all the parties involved in the case, hence it's important to analyse how one should interact with each party.

Relationship with lawyers

The most important rule to follow when interacting with lawyers is that **the defence strategy is always chosen by the lawyer**. This happens because the defence strategy usually goes beyond technical details. Namely, technical defence is only a part of the whole defence strategy.

The lawyer also owns and handles the relationship with the client, even if the client directly hired the expert. This is because the client is defended by the lawyer, hence the lawyer is the only one authorised by the client to speak for them. Namely, an expert can't inject himself or herself into the relationship between lawyer and client and, if the expert is not able to do this, he or she must

resign. Moreover, we always have to remember that disparaging another professional in order to make the client reevaluate the relationship with that professional is an ethical violation which could even lead to expulsion from the order of engineers.

We also have to remember that an expert for a part is paid for by that part and should always work in favour of that part. Moreover, the expert is bound by professional secrecy, even if in some countries only lawyers, doctors and priests benefit from it. However, even if the lawyer and the client pay for our bills, they should not dictate what we do as technical experts in fact we are always responsible for what we do, hence we always have to write and say things of which we are convinced. In some cases we might be asked to omit something but, since an expert is a witness, we have to take care not to lie when omitting something. This is because, in some jurisdictions (e.g., in Italy), a witness can't lie nor not answer a question (whereas an accused person can lie or not answer a question). If the expert witness might have to reveal something that makes the client guilty, the lawyer might decide to use the expert only as a consultant and not as a witness in court.

Relationship with clients

Even if an expert should never handle the customer directly, he or she's still paid for by the customer. The expert should therefore remember that he or she should always work his or her best to defend the customer. This is fundamental since the law works this way. Both the prosecutor and the defender have to build their best scenario and the judge should evaluate what is closer to reality considering the collected evidence. This doesn't mean that the expert helps the client escape the law.

One important thing to keep in mind is the difference between historical truth and process truth:

- **Historical truth** is the way things happened really. Historical truth is very hard to know completely because of the many points of view, sides and granularity levels.
- **Process truth** is what is deposited in a court of law and can be used to evaluate if someone is guilty or not. The process truth is built by the parties involved in the process by filing many documents. The process truth also includes the testimonies of witnesses.

Processes are not held to find historical truth but process truth. In the best case, process truth is an approximation of historical truth, but it's not always like this. This means that when someone is found guilty or not guilty, it doesn't mean that he or she's actually guilty or not but it's simply the result of the analysis of the process truth.

Relationship with the police and the prosecutors

In some cases, an expert is called by the police or the prosecutors. In this case, everything we've said for clients and lawyers still holds, however, we have to take extra care and consider other rules.

First, we have to remember that assisting the prosecutor doesn't entail moral superiority. This means that we shouldn't think that we are on the right side of the process (i.e., on the side of the part which is not guilty) and we should only stick to facts extracted from evidence, independently from the fact that they are in favour or against the prosecutor. This is because in some jurisdictions (e.g., in Italy) the prosecutor should look for the truth, hence all facts should be presented.

A technical expert should not fall into the inquisitorial mindset nor think that he or she is like a policeman trying to find the other party guilty. In particular, an expert working for the prosecutor should not work to put someone in jail but to find the truth. This means that the expert should never try to punish the other party but only stick to the facts and answer the technical questions posed by the judge.

3.5.2 Evaluation process

One of the most important parts of the evaluation process is the analysis of all the documents that form the process truth. This means reading reports of other witnesses and investigators and reconstructing the picture of the case, namely the process truth. Note that at different stages of the process, different documents might be available (e.g., at the beginning we don't have the witnesses' testimonies). When analysing the documents we have to look for:

- **Errors**. Errors are pieces of information in the documents that are surely not true. If we can find an error in the documents we can use it in favour of our client. Some examples of errors are omissions or mistakes. The former is particularly interesting because if the other part's expert has omitted something, it's possible that it's something incriminating, hence something to further look into and investigate. Better said, the other part might have omitted something since it would incriminate his or her client. We can therefore look into it to defend our client.

- **Unclear methodology**. If a method for obtaining some evidence is not clearly described or wrongly applied, we can use it against the other part since the information obtained from such evidence could be far from the historical truth.

- **Suggestive writing**. Suggestive writing means writing a document in such a way that it implies a theory without explicitly supporting it with facts. Namely, the document implies something we are not sure of (i.e., that is not completely supported by facts) without saying that it's just a theory. It's important to find suggestive writing since experts should always stick to facts. Moreover, if a judge doesn't know that a document is written using suggestive writing, then he or she might be misled. Note that we, as technical experts, might also unconsciously fall into this error since we might feel sympathy for a client and extend it to what we write. This means that we have to split facts from what we think when writing a document. Let us consider an example to understand the difference between factual writing and suggestive writing. A document written with factual writing would write: *The images are located in the folder* `~/dir` *and have been created on date 10/2/20 and accessed on date 12/2/20.* A document written using suggestive writing would say: *The images have been added to the folder* `~/dir` *by the subject on date 10/2/20 and accessed on date 12/2/20.* As we can see in the second case we are assuming that a person has accessed the files whereas, in the former scenario, we have just reported facts.

- **Opinions and speculation**. A technical document should only report facts, hence finding an opinion or some speculation might benefit our client. Note that this is true for criminal law only, in fact in criminal law, an expert witness can't give an opinion whereas in civil law one can be asked to give an opinion. Since we are talking about opinions it's good to properly understand what we mean by opinion and what's the difference between an opinion and a hypothesis. Both should only be based on facts and experience, however, the latter is simply a way to explain facts and give an interpretation to them while the former is less factual. For instance, an opinion is the evaluation of a damage since it surely depends on facts but it depends more on the expert. Finding a hypothesis is very important since we, as experts, can find another interpretation that might help our client and change the opinion of the judge.

Note that lawyers and experts analyse the same documents but from a different perspective hence it's important to coordinate the efforts. Experts usually analyse more in-depth:

- Experts reports (of the same field).

- Police reports when they involve some technical elements.
- The paper tray of the evidence (i.e., a less strict chain of custody) acquired from search and seizure.

Analysis of errors in the documents

Searching for errors in the documents is very important, hence it's worth giving a better look at this phase. Errors can be made both in the acquisition and in the analysis phase:

- In the **acquisition phase** errors are usually made:
  - during the search and seizure,
  - in maintaining the chain of custody,
  - when describing the acquired material (an expert might notice that the received material is different from the one described in the document written in the acquisition phase),
  - in the procedure used to get evidence (e.g., the hard drive hasn't been hashed).

  Note that in some jurisdictions (e.g., in Italy) finding acquisition errors isn't enough to reject a piece of evidence and we should continue with the analysis phase. In fact, in Italy, evidence is rejected only if acquired by committing a crime.
- In the **analysis phase** the most common mistakes are:
  - hashes are not verified during every step,
  - a proprietary program is used,
  - the analysis process is not described,
  - technical mistakes.

  Technical mistakes are the most important and the most incriminating.

Errors can be made in the presentation phase, too. The most common errors done in the presentation phase are:

- **Not exploring alternative hypothesis**. An expert always gives the interpretation which is more favourable to his or her client. This means that another expert could find an alternative explanation which might be better for his or her client. When we present an alternative hypothesis we have to be sure that it's plausible otherwise we'll just make the original hypothesis more plausible (since we have shown that an alternative hypothesis is not plausible). In some cases, an alternative hypothesis might be non-scientific. The Trojan defence is a classical example of an alternative hypothesis which is not scientific. Say we are considering a hard drive and we want to show that our client hasn't accessed it. We could always state that the computer to which the drive was connected has been affected by a Trojan malware that has accessed some files and then erased itself together with any evidence. This means that we have no trace of the Trojan, hence the program that has accessed the drive can't be tested and the hypothesis can't be reproduced in a lab, which means that it's not scientific. In some specific cases, a Trojan defence is very plausible, hence it could be considered even if not scientific, however, if used for the wrong cases it might only yield the opposite result and make the original hypothesis more plausible.

- **Giving a biased presentation**. A presentation should only stick to facts, hence a biased presentation is an error.
- **The availability of counter-examples**. Finding counter-examples that fail the assumptions used to formulate a hypothesis can make the hypothesis less plausible. In general, examples and counter-examples are very useful since they allow non-technical people (like the judge, which is the one who decides, in the end, the fate of the client) to better understand what we've found. When something is understandable, it looks more plausible, hence a judge that understands our hypothesis might consider it more plausible, which is a big advantage for our client.
- **Missing explanations**. Finding missing explanations or even missing facts in one's presentation is very important since it means that something in favour of our client might have been omitted.

3.5.3 Presentation process

The presentation phase usually involves **writing a report** on what has been found and answering some technical questions, called *quesiti peritali*, asked by the judge. The most important things to remember when writing a report are:

- **The report has to be clear**. In particular, the report should focus on what we want to explain so that the reader can focus on what we have found.
- **The report should not use or limit, especially in Italy, non-national language** (e.g., use too many English terms in a report written in Italian).
- **Technical terms should be explained in footnotes**.
- **The report should be concise and simple but not simplistic**. This is because judges do not understand technical stuff but are very good at reading people and can understand if we are treating them as ignorants.
- **We always have to explain why something we are going to say is important** and then say what we want to say. This is important because reading technical stuff might be hard, hence it's better to give a reason to read something. Moreover, we have to remember that what is obvious to an expert isn't to a non-expert, hence giving a full explanation of what we are saying and why it's important helps everyone understand the report.
- **We have to stick to what is important to the case**. Reading technical reports is hard, hence it's better to concisely answer questions before actually explaining the motivation behind such answers. In practice, we should give a small answer to each *quesito peritale* and then give a technical explanation for each answer.

Writing a good report is not just a matter of style but it's important since the report we write is the only thing that matters in the courtroom. In fact, the judge doesn't see the discussions, the debates and the evidence we've used to write the report but only reads the results of our analysis on them. When writing a report, in addition to what we've stated beforehand, we have to consider that:

- We will have to read the report in the future (an expert could be asked to testimony even years after having written the report or the report might be used in a future appeal), hence we have to write things as if we were to explain them to our future self.

- The report will be read by the expert of the counterpart who will try and find errors in our work.

In addition to analysing what we should write, it's also important to give a look at what shouldn't be written in a report:

- **We should not write that something is not working without giving an explanation** of the reason for which it's not working. In general, when talking about technical stuff, we have to remember that we might know something that the judge doesn't, hence it's better to explain everything (without making the judge look too stupid). For instance, an expert could be asked why a suspect has dragged some files to the bin. The answer is easy for an expert however it's not for a judge, hence we should give a clear and complete answer so that the judge has every piece of information to evaluate the case.
- **We should not suggest and use suggestive writing**. In general, we should not use techniques we don't want other experts to use.
- **We should not be too technical**. Being very technical might be a good idea (but that's not a certainty) with other experts, however, it's not when talking to normal people (i.e., non-experts). In particular, being too technical might result in being obscure since the person we are talking to can't understand what is too technical. This is a problem because what is obscure usually looks false but we want others to believe in what we are saying. This is especially true for judges who have a lot to do but have no time nor will to read through the technical stuff we've written. A judge usually doesn't want to read technical reports but he or she has to, hence we should try to make the report as clear and easy to read as possible. We should also acknowledge that being obscure is a sign of lying, hence it's better to be clear, especially because judges are good at discovering lies.
- **We should not write in a biased way**. What we write in a report has an influence on humans (e.g., a report might help send someone to jail) and we might feel sympathy or solidarity with someone. This shouldn't however impact the way we write the report which should only stick to facts. This is easier for computer science experts since we have to deal with computers but it's harder for other experts, like psychologists, which have to deal with emotions.
- **We should not show deference** to the court. In particular, we should always remember to show respect to the court, especially to the judge. We shouldn't however show deference. This means that if some error is done in court we should correct the person making such an error, if we are sure of what we're saying.
- **We should not use sarcasm**. This is true but, in some cases, if we are really sure of a point we are making, a little irony can help and some judges might even appreciate it. Note that sarcasm is often used by lawyers since they also use emotions to convince the jury or the judge, however, we shouldn't do the same since our job is simply to report technical facts.
- **We should not use weak arguments** if stronger ones are available. In particular, if we have more than one argument, we should only mention those that are really strong. This is because a set of arguments is as strong as its weakest. Say for instance we have 4 strong arguments and a weak one. If the other expert shows that one of our arguments is weak, it will look like all the arguments are weak since we have lost credibility with that weak argument.

Structure of the report

Now that we know what and how to write the report, we can define a blueprint for its structure. A technical report is usually modelled upon a **scientific paper** and is divided into four parts:

1. **Introduction**. The introduction should drive home the point we are going to make in the rest of the document. The idea is to explain immediately the results of our analysis.

2. **Facts**. In this part, we put all the facts obtained from the evidence we worked with.

3. **Technical discussion and analysis**. This part should contain the technical explanation of the points we made in the introduction. What we write should be technical enough for supporting the facts we've collected but shouldn't be too technical since it has to be read by others. Each block in this part should contain a starting and ending paragraph with a summary of what will be said in the block and what has been said in the block, respectively. This allows people to skip the blocks they are not interested in while giving a complete picture of the document.

4. **Conclusion**. The conclusion should contain a summary of everything we've said previously. The conclusion is where the judge immediately looks to find answers, hence here is also a good place to answer the *quesiti peritali*. The answers should be as conclusive as possible (i.e., yes or no) since answering "it depends" is similar to saying that we can't answer the question. Not answering a question could be a problem for our client since it might look like that answering the question has a negative effect on our client. In this context, it's sometimes better to explicitly say that a question can't be answered and the reason why, since it might give us more credibility.

From the point of the defence, a report is usually seen as an obstacle course for the judge. In fact, the judge usually wants to send our client to jail, hence he or she (the judge) sees our report as a sequence of obstacles that he or she has to skip so that our client can be sent to jail. When we talk about skipping obstacles we mean that the judge tries to find excuses or shortcuts that allow us to ignore what we've written in the report. As a result, we should structure the main body of our report as an obstacle course with the sole purpose of tiring the judge. To achieve this result we should place obstacles (which are facts and explanations in favour of our client) in an incremental way (from the easiest to skip to the hardest to skip) so that the judge reaches a point where it's not possible anymore to ignore what we've written.

A sample report could be divided into the following sections:

1. **Foreword**. This section should contain what has been examined and the operations and experiments that have been done. This section should also include what questions we have been required to answer. The foreword section is important since it sets the limits of our work and allows us to say what we can answer and what we can't.

2. **Introduction**. The introduction section should contain what we are going to show, what we have found and why those things are relevant. Explaining why something is relevant is very important since we are dealing with non-experts who, in some cases, don't even know what questions to ask, hence we should give them the facts and the reasons why they are important for the case.

3. **Acquisition issues**. This section should highlight any error done during the evidence acquisition phase, hence on how evidence is acquired. This is a preliminary analysis. An example

could be showing that hashes haven't been computed. In this situation we don't only have to explain what a hash is but also why it's relevant that the hashes haven't been computed since the judge doesn't know what hashes are and there exists no law that forces to compute hashes on digital evidence.

4. **Technical analysis**. In the technical analysis, we can show what has been done wrongly by the other part's expert and how his or her conclusion is wrong. Here we can also highlight that, even assuming that every previous phase has been correctly executed and evidence has been correctly acquired, we can explain what has been found in another way. We can also show what experiments haven't been done or that some evidence is missing. Note that pointing out that something is missing might be very advantageous since, if it has been voluntarily avoided, we can assume that what's been avoided contains something against who has written the report. For instance, one might find some compromising material on our client's hard drive but we could point out that the expert hasn't checked if such material was in a user-defined folder or in a default folder (i.e., if the suspect has voluntarily downloaded such material) or if the computer was affected by a malware that automatically downloads such material.

5. **Conclusion**. In the conclusion we highlight what we have shown in the previous sections and we state a conclusive phrase that summarises what we have discovered. The conclusive phrase should be conclusive and inescapable (i.e., the judge should not be able to skip over that obstacle). This means that we should conclude with the strongest thing we can say supported by what we have found. In the conclusion we usually find statements like:

   - In this report we have shown X.
   - Evidence was not properly acquired and thus these manipulations may have contaminated it.
   - Item I did not happen as described, making conclusion C completely wrong (without fixing the conclusion).
   - Theory T fits the facts better.
   - Nobody checked if F was true or false, which could have led to G.

The example above is usually used in criminal court cases. In civil courts, we usually go through the counterpart's theory to take it apart. In some jurisdictions, the judge appoints an expert that writes a report which will be used by the judge to give its final verdict. The job of each part expert is therefore to go through the judge's expert to examine what he or she did wrong. The judge's expert will then correct its report. This means that each part's expert should write for the judge and the expert.

3.5.4 Testimony as a witness

In some jurisdictions, an expert witness is asked or required to testify in front of the court. For instance, in Italy, an expert witness must testify in order to submit a report. In other places, an expert can be called to testify but it's not mandatory. Moreover, in some places, a report or a testimony is a sworn testimony (i.e., the witness has to swear that he or she will not lie). In any case, a witness can't say or write a lie and a lie or an omission is a perjury, which is a crime. Also note that not all experts can claim professional secrecy. For instance, in Italy, only lawyers, doctors and priests can.

When asked to testify, a witness can undergo direct or cross-examination, depending on the jurisdiction. For instance, in Italy, one undergoes direct examination in civil courts but cross-examination in criminal courts.

Direct examination

When undergoing direct examination a witness is required to ask a predetermined list of questions (e.g., in Italy all questions are fixed by the law, start with "Is it true that" and must be written in such a way that the only answers are yes, no or don't know). Since the questions are predefined, we can prepare for them. In particular, when answering:

- We have to be clear and explain what we are saying.
- We have to talk to the judge since he or she is the one that has the final decision.
- We have to remember that a judge can ask something different from what is in the list of predefined questions, hence we should study the judge's previous cases to understand what he or she usually asks. This is important because the judge might ask something negative for our client, hence we should prepare to answer.
- We don't have to rush nor use too much time.

In general, apart from the questions asked directly by the judge, one should not be scared by direct examination since it can be prepared.

Cross-examination

Cross-examination is done by the prosecutor, i.e., by the opposite party, hence this is the most difficult type of testimony. Since there are no fixed questions it's much harder to get ready (even if it's possible by analysing the prosecutor's history). Moreover, the prosecutor usually becomes hostile quite fast. This is done to:

- Get the witness on their nerves so that he or she is forced to do some mistakes or to be more emotional.
- Discredit the witness.

This isn't done for a personal reason but only because this is how courts work. The prosecutor wants to send the suspect (i.e., our client) to jail, hence he or she will do anything to achieve this goal. This also includes leveraging the other's emotions or putting the other part in a bad light. To handle cross-examination, which is much harder than direct examination, we can:

- **Prepare**. Even if we don't have a set of questions we can be asked, we can still study the prosecutor's previous cases and understand how he or she usually approaches the examination.
- **Use the report**. When asked a question, an expert witness can use the report he or she has written since the witness might be required to testify long after having analysed the evidence. This means that if we are asked a negative question for our client we can gain some time to think by looking into the report.
- **Give complex answers when asked a negative question**. In general, it's better to be clear, however, we can avoid a negative question for our client by using really technical terminology and complex reasoning in the answer.

- **Give an exhaustive answer to positive questions**. If, by chance, the prosecutor asks a positive question for our client we shouldn't simply give a yes or no question but we should also explain the reason why in the details. This can help our client a lot.

In some cases, we could be tempted to try and avoid a question by saying that we haven't analysed some documents. However, if the documents have been submitted and the answer to the question has clearly been answered in some technical report, there is no reason to not answer (since the answer is already in the process truth). Moreover, saying that we haven't looked at a document will only make us look incompetent whereas answering correctly will increase our credibility (and, since the answer is already evident in the process truth, we can't do any more bad to our client).

Chapter 4

Malware analysis

4.1 Windows malware analysis

One way malware has to achieve persistence is to modify the Windows registers such that, when the laptop boots, it fetches some info from the registers and starts the malware.

Windows uses IAT to jump to functions just like Linux uses the GOT.

Some useful online VMs for reverse engineering are:

- Any run.
- Virus total.

Chapter 5

Mobile forensics

5.1 Introduction

Mobile forensics is currently one of the most important parts of digital forensics since mobile phones are a key part of everyone's life and we keep much information about us on our phones. Moreover, working in this field requires being up to date since new devices and operating systems updates are released every year, each with its characteristics and features which might differ from the previous devices.

Let's start by defying mobile forensics.

> **Definition 5.1** (Mobile forensics)**.** *Mobile forensics is a branch of digital forensics related to the recovery of digital evidence from a mobile device under forensically sound conditions.*

As we do with computers, we would like to analyse a mobile phone without modifying any bit of storage but recovering as much data as possible (ideally, every bit of storage). This isn't however always possible, especially when dealing with mobile devices. Because of this, we have to recover data in a forensically sound condition. This means that we have to document in detail the methodology used to recover data from a mobile device and the reasons why such a procedure has been used. Say for instance that we want to overcome a protection of an OS to extract all the data. In this case, we need to overcome some protections (i.e., modify something) of the OS since it doesn't allow us to read raw partitions or the full filesystem. In this instance, this action is justified by the fact that this is the only way of reading the whole storage.

Being able to obtain data from a mobile phone in a tamper-free way (i.e., without modifying anything) is very hard, hence we need validated and tested tools and methodologies. The methodology we use often depends on the characteristics of a device (e.g., the brand, the model, the OS, and many other parameters). Depending on the characteristics of a device we can exploit different vulnerabilities to obtain the information we need.

One of the keys to forensics soundness is documentation. In particular, we should document any changes in the device. Moreover, we should also follow international standards during the analysis process. A good starting point is the document Six Steps To Successful Mobile Validation. The following documents contain the main guidelines for mobile forensics:

- SWGDE Best Practices for Mobile Device Evidence and Collection Preservation and Acquisition (5.1).

- ISO/IEC 27037:2017 - Guidelines for identification, collection, acquisition and preservation of digital evidence.
- NIST Guidelines on Mobile Device Forensics.

In the next section we'll follow the first document but the others also contain useful material (the second is more general the third is older).

5.2 Device analysis

Mobile forensics poses many challenges. The most relevant ones are:

- **Market fragmentation**. Market fragmentation refers to the high number of manufacturers, hence different chipsets and different OS customisation. In fact, mobile devices usually run iOS or Android, hence we don't have a huge fragmentation in terms of operating systems (but we have in terms of OS versions). As a result, we must know how to approach each device based on its characteristics (mostly chipset and OS version).
- **New devices and operating systems** are frequently released. Manufacturers release new devices and operating systems very frequently (usually each manufacturer releases something new once a year), hence we could have a new device every month. This is a challenge because every new device or OS has new features or comes with some old features removed.
- There exist a **huge number of applications**. Since each app might contain data relevant to a process, we must be able to extract data from those applications, too. Unfortunately, every application works differently and represents data in a custom way, hence we have to develop specific tools for each application.
- We have to handle **huge volumes of data**. Modern phones have huge storage (up to terabytes) hence we have to develop tools which are able to handle these volumes of data. Recently, machine learning has come into play to swiftly analyse and classify data without human intervention (e.g., find images of drugs in a photo gallery).
- Data is being moved on **the Cloud**. Since modern applications generate a lot of data, some are usually moved to the Cloud where it's harder to access and analyse it.
- **Protection mechanisms**. Every device has its own protection mechanism (e.g., passcode, Touch ID, Face ID) which protects and encrypts data on the device. This means that without bypassing these barriers it's almost impossible to retrieve something from a device.

5.2.1 Evidence collection and preservation

The first stage of mobile forensics is the collection of the devices, namely when law enforcement is intervening in the field to collect some devices. In this phase, it's important to document everything that is done, for instance:

- the date showed on the device,
- if the device is on or off,
- if there is a passcode and if it was provided by the user,

- the device characteristics (e.g., manufacturer, model, serial number, the IMEI code),
- if the device is damaged,
- if any external peripheral has been removed.

During this phase is also important to take pictures of the devices and the peripherals to write a more complete documentation. The document 5.1 proposes a collection and preservation process for each operating system. Figures 5.1 and 5.2 show such a process as a flowchart.

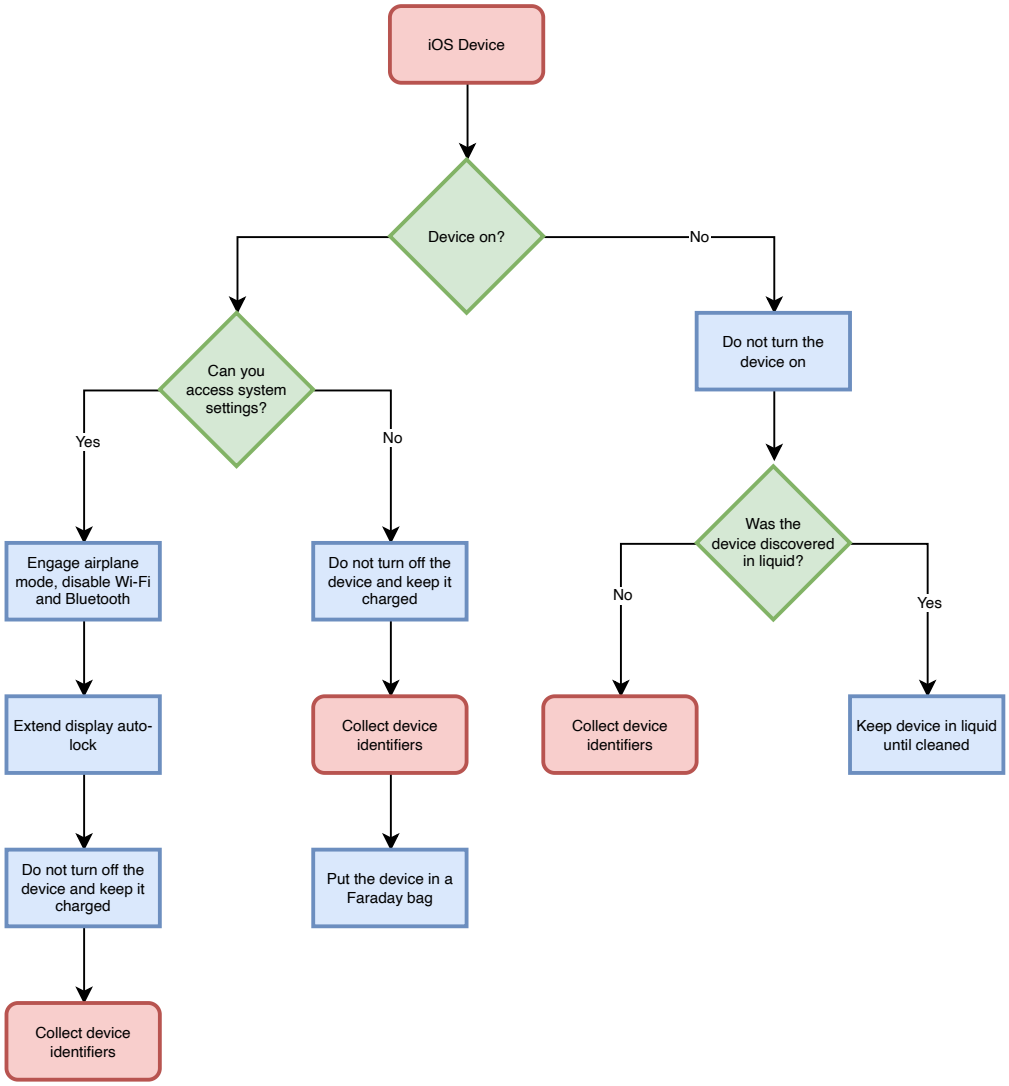


Figure 5.1: The collection and preservation process of devices running iOS.

When searching a crime scene, we shouldn't only look for the devices but also for the peripherals connected or linked to the devices, for instance, smart watches or IoT devices. Peripherals are useful

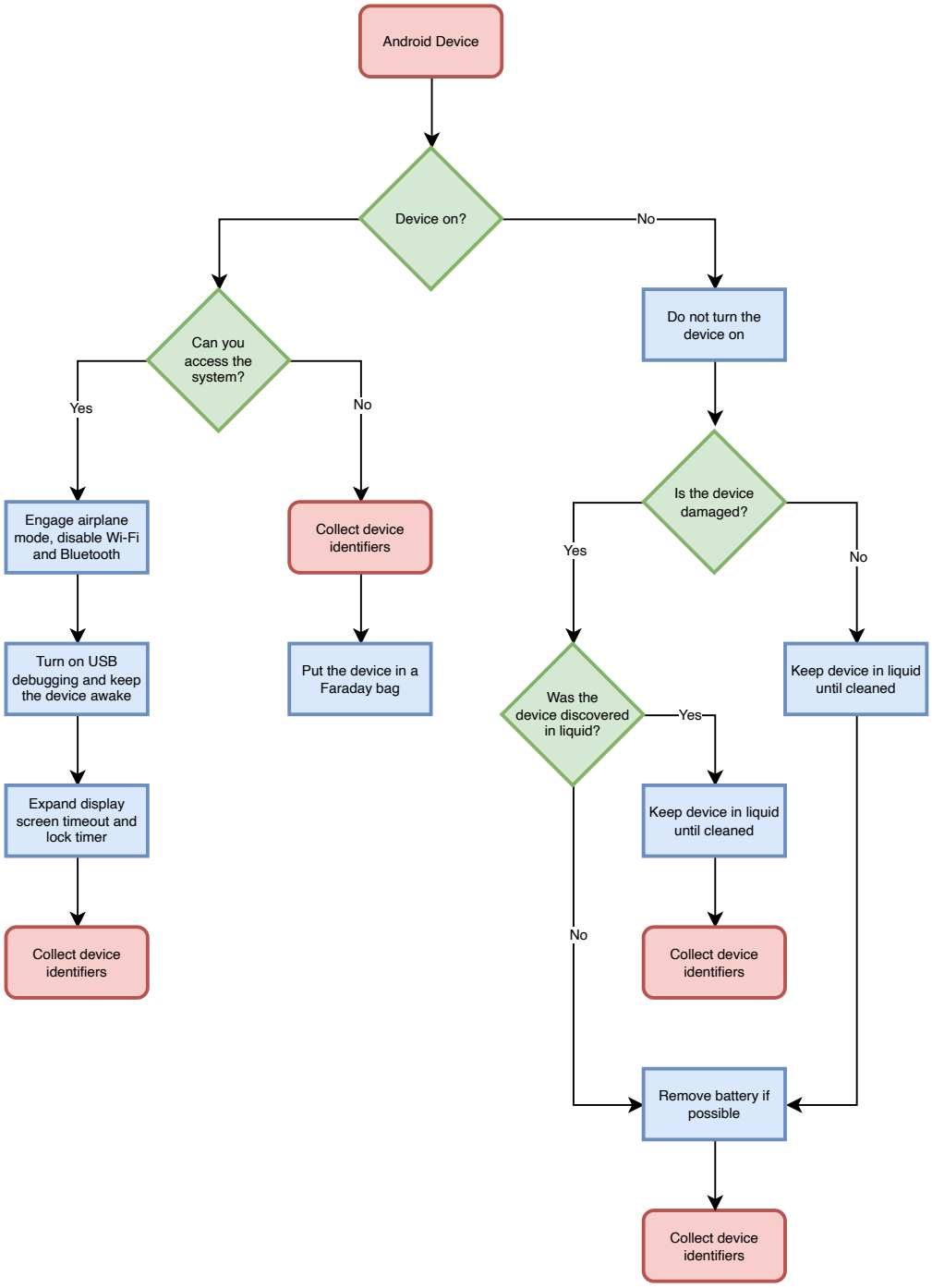


Figure 5.2: The collection and preservation process of devices running Android.

since they might use the same account used for the mobile phone but it might be easier to retrieve information.

When handling evidence we have to remember that classical forensics comes before digital forensics. This means that we should first collect DNA or fingerprints and then handle the devices.

Handling devices when powered on or not

Depending on the power status of a device (i.e., it's on or off), different procedures have to be taken:

- If the device is **powered on and unlocked** when found and it hasn't been turned off when collected, we should immediately isolate it to avoid cross-contamination or remote deletion. In this scenario, **data should be extracted as soon as possible or the phone can be turned off** (if we can't collect data immediately). Note that we should turn it off only if we have the passcode.
- If the device is **powered on and locked**, then it's in the After First Unlock (AFU) state, which means that the phone was booted at least once since the last turn-off and the user has inserted at least once the passcode. Technically, this means that the RAM contains the encryption keys to get access to the data. If this is the case, we should **leave the phone on and isolate it from the network**.
- If the device is powered off we should keep it turned off.

Network isolation

Another key procedure is isolating the devices from the network, especially if we want to keep the phone on. This way, no one can delete the data on the phone remotely, in fact, both iOS and Android allow users to fully erase a device using another device (i.e., the erase command is sent from another device).

5.2.2 Evidence acquisition

After having collected the devices on a crime scene we have to analyse them. In some cases, we don't even need to seize them (i.e., take them to a lab for later analysis) since the analysis can be done in place.

Equipment preparation

The analysis of what we have collected requires specific tools, which have to be prepared before being used. Some of these tools are specific for forensics, others come from other worlds (e.g., mobile repair) but are very useful in analysing a phone.

Most of the tools used for data extraction are commercial. These tools are used to overcome the protection mechanisms put in place by the OS to have root access to the system. This is because a user doesn't have full access (i.e., root access) to the operating system, hence we have to obtain it by exploiting vulnerabilities which allow us to overcome said protection mechanisms (which are not limited to the phone's passcode). Different tools are more suited for different phones or operating systems and that's why we have many different tools. These tools are quite expensive, hence we also have to find a trade-off between a complete analysis and a high cost.

Device identification

After having prepared the tools we need for the analysis, we have to identify what devices we have to analyse. Namely, we have to identify the characteristics of the phones to analyse.

For starters, we have to identify the physical device. This is typically done through the International Mobile Equipment Identity (IMEI) code, which is written somewhere on the device (e.g., on the sim card slot, on the back of the device, on the box of the device, on the mobile settings or typing the number *#06#). Retrieving the IMEI is typically quite easy. If available, we should also try to identify the serial number and the model number, which allow understanding when the user bought the device. In any case, the IMEI is the most important piece of information since it can be used to retrieve the physical characteristics of a device. Given an IMEI we can query http://www.imei.info/ to get a ton of information about a mobile phone. One of the most important pieces of information we can get from this page is the chipset since most of the vulnerabilities used to root a phone are chipset based. This is because once a vulnerability is discovered at the chipset level, there is nothing the manufacturer can do to fix it (i.e., the manufacturer can't release a security patch). Another important piece of information is the device warranty status. This information is helpful to understand when the user bought the phone since warranties expire some time after buying the phone.

Data extraction

After having extracted the physical characteristics of a device we need to extract the data. Data can be extracted at different levels, in fact, we can do:

- A **physical acquisition**. A physical acquisition allows us to extract every bit of data on the mobile storage (hence even deleted data which is still in memory). This isn't always possible. In some cases physical acquisition is even useless since the OS might use file-based encryption, meaning that there is one encryption key for each file, hence it's useless to get a physical dump since we can't recover deleted files since the encryption key has been deleted upon file deletion. As a result, physical acquisition is useful only when we have no encryption or full-disk encryption. A physical acquisition requires root access. If the device is not protected by passwords we can even use invasive techniques to read files directly from the chipset (e.g., JTAG, ISP or removing the chip). Since we need root access, physical acquisition requires rooting, jail-breaking or using some vulnerabilities to obtain root privileges.
- A **full file system acquisition**. A file system acquisition extracts files from a file system and may include data marked for deletion. A file system dump is the best we can get if the OS uses file-based encryption. A full file system acquisition requires root access. File system acquisition is usually performed with the device backup features or through the MTP or AFC protocols. Rooting or jailbreaking a system also allows to do a file system acquisition.
- A **logical acquisition**. A logical acquisition allows us to extract individual files and objects. A logical dump is the best we can do when we don't have root access and surely doesn't allow us to obtain full disk access. For instance, on Android, we can install an agent to get permission to read data from some content provided to get contacts, messages or media. It's however hard to get access to third-party apps. Logical acquisition is simple and fast, however, allows obtaining only some data (e.g., information from native apps) and no deleted data. Moreover, logical acquisition requires the passcode of the phone.
- A **manual acquisition**. A manual acquisition involves the manual operation of the keypad and handset display to document data present in the device's memory.

When extracting data, we also have to consider removable media (e.g., the SD card on an Android phone, the SIM card) which can contain useful information (even if now most of the interesting data is in the device memory or storage).

Part III

Fraud analysis and detection

Chapter 6

Fraud analysis

6.1 Frauds

An important part of digital forensics is the analysis of frauds. Let us then define what a fraud is.

> **Definition 6.1** (Fraud)**.** *A fraud is a wrong or criminal deception intended to result in financial or personal gain.*

This definition highlights two important characteristics of frauds, which are:

- A fraud has a criminal purpose.
- A fraud is related and aims to some gain.

The definition we just stated is correct but rather general and isn't enough to develop a fraud detection system. This means that we need another definition which highlights more characteristics of a fraud.

> **Definition 6.2** (Fraud)**.** *A fraud is a*
>
> - *uncommon,*
> - *well-considered,*
> - *imperceptibly concealed,*
> - *time-evolving and*
> - *carefully organised*
>
> *crime which appears in many forms.*

Also note that a fraud is a social phenomenon since it's something that is originated by an attacker but that affects a victim. The victim can be a single person, a community or the entire society, hence a fraud is a social phenomenon in the sense that it might effect the whole society.

6.1.1 Characteristics

Let us now analyse the main characteristics of a fraud.

Uncommon

A fraud is uncommon in the sense that only a minority of exceptional behaviours concerns frauds, of which only a limited number is known to be a fraud. This means that, among all cases that display an exceptional behaviour only a few are actually frauds and such frauds are only a part of all the frauds that really happen. Put it in another way, the detected frauds are only a part of the frauds that really happen. This means that we have a lot of false negatives (i.e., cases which are not considered frauds, but that actually are). Let us consider an example to better understand this property. Say we have a list of bank transactions. Only a part of these transactions are frauds and some transactions are marked as legitimate even if they are frauds.

This characteristic of frauds explains why it's hard to build a good fraud detection system. Since frauds are not that common, we have few examples to learn from, hence the algorithms (especially ML-based ones) have less data to leverage to recognise frauds. Moreover, we have to consider that something marked as fraud might be misclassified, hence the detection algorithm has an even harder time figuring out what is legitimate and what not.

Well considered and imperceptibly concealed

Fraudsters (i.e., those that commit a fraud) want to remain unnoticed so that they can work freely. To achieve this goal, fraudsters have to plan the fraud (i.e., the fraud has to be well considered) and have to act as legitimate users (i.e., non-fraudsters).

Time evolving

Frauds are not static phenomena since fraud detection system evolve and are able to recognise more frauds (thanks to the fraud samples collected). This results in a continuously evolving circle in which fraud detection system evolve, hence frauds have to evolve to avoid more powerful detection systems but the latter collect samples of more refined frauds hence they evolve too and the loop goes on.

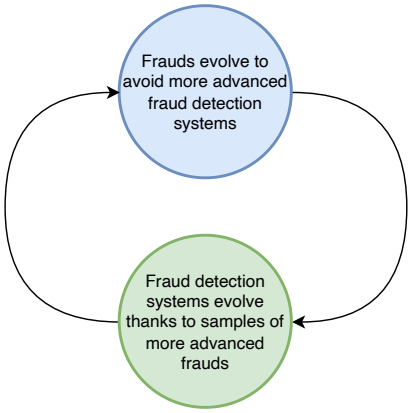


Figure 6.1: The loop that explains the continuous evolution of frauds and fraud detection systems.

Carefully organised

Frauds aren't usually committed by just one person but by a large and complex organisation made of many components, each of which handles a different aspect of the fraud. A fraud is usually made of many building blocks some of which can even be outsourced (as we have seen when talking about the cybercrime ecosystem), hence a fraud can be seen as a collection of many sub-frauds. For this reason, its important to analyse and understand the context in which a fraud is done. In some cases, one might even want to go back to the sub-frauds used to execute the main fraud and analyse them.

6.1.2 Fraudsters

When analysing a fraud, it's important to understand who is behind it. That is, it's important to understand who are the fraudsters. A fraudster is a person, moved by some sort of benefit, usually monetary but also personal, that commits a crime. In general, a fraudster is modelled using the fraud triangle, as shown in Figure 6.2.

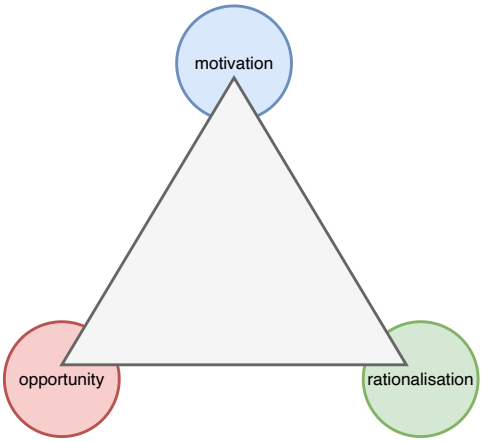


Figure 6.2: The fraud triangle.

The fraud triangle has, on its vertices, the characteristics that define a fraudster. These characteristics are:

- The **motivation**. Fraudsters are usually moved by greed (i.e., they want money independently from the fact that he or she needs it) or need.
- The **opportunity**. Fraudsters decide to do a fraud when they know that no real security solution is put in place to stop a fraud.
- The **rationalisation**. Fraudster, after having discovered a security lack, have to realise the fact that they are committing a crime and they have to believe that it's right to do it.

Fraud detection systems can only work on the opportunities.

6.2 Fraud categories

Frauds can be classified in many categories, the most relevant being:

- **Banking and credit card fraud**.
- **Insurance fraud**.
- **Corruption**.
- **Counterfeit**.
- **Product warranty fraud**.
- **Healthcare fraud**.
- **Telecommunications fraud**.
- **Money laundering**.
- **Click fraud**.
- **Identity theft**.
- **Tax evasion**.
- **Plagiarism**.

6.2.1 Social engineering

One of the most used vectors for perpetrating a fraud is social engineering.

> **Definition 6.3** (Social engineering)**.** *Social engineering refers to all techniques aimed at talking a target into revealing specific information or performing a specific action for illegitimate reasons.*

Another definition of social engineering is the following.

> **Definition 6.4** (Social engineering)**.** *Social engineering is the act of manipulating a person to take an action that may or may not be in the target's best interest. This may include obtaining information, gaining access, or getting the target to take certain actions.*

In other words, social engineering collects all the techniques used to psychologically manipulate a person into doing something. Basically, social engineering could be described as hacking a person. Social engineering consistently works because it's hard to teach humans how things should be done correctly (especially to non-experts), whereas it's easy to fix a machine. In other words, people are the most dangerous vulnerability of a system (and can't be easily fixed). This explains why social engineering is used and is one of the fraudsters' favourite vectors. Moreover, social engineers are usually also fraudsters.

The problem with many frauds involving social engineering is that the victim is usually arrested as the culprit, even if the real criminal is the one that tricked the victim using social engineering. This happens because it's usually hard for the victim to show who the real criminals are (remember that fraudsters are well organised and plan the fraud ahead to stay hidden) since they have little to no information on them.

Phishing

Phishing is one of the most used, if not the most used, social engineering techniques for committing frauds. A classical example is a link in a mail which seems legitimate (i.e., the mail is sent from an address which looks real and has a plausible content and object). When a victim clicks on the link, he or she is redirected to a fake site for stealing credentials or some vulnerability in the browser is exploited to gain access to the victim's machine. Phishing is so effective that it's hard, even for experts, to distinguish legitimate content from fraudulent one, especially when using a very conservative approach (in this case even legitimate content is flagged as fraud).

Chapter 7

Fraud prevention and detection

7.1 Anti-fraud strategies

Frauds are fought using a combination of

- **prevention** and
- **detection**

that collaborate to protect a system against frauds. When building a fraud protection and detection system we have to remember that an attacker (i.e., a fraudster) never targets the stronger point of a system but it's weakest. This means that, if we are able to secure a certain company and advertise the security measures put in place, fraudsters will be more inclined to attack another company which hasn't adopted the same security measure (or at least hasn't advertised it). This shows how important it can be to show off the security measures of a company (even if enlarged). As we have seen, protecting a system involves two components. Let us then briefly see the differences between them before analysing them more in-depth:

- **Detection**. Detection is the recognition of fraudulent activities. This means that fraud detection is an ex-post approach since it happens after the fraud has been committed.
- **Prevention**. Prevention is the act of not allowing frauds to happen. This means that fraud prevention is an ex-ante approach since it happens before the fraud has been committed.

Detection and prevention are complementary but not independent, this means that when detection is improved, which allows to collect more data and improve prevention, which generates more advanced frauds that require better detection systems and the loop continues. In other words, a change in one part of the system influences the fraudsters which impacts on the other part of the system. Let us consider an example to understand the difference between detection and prevention. Let's consider a banking fraud which attacks a bank that uses a website to use the online banking system and a phone for 2-factor authentication. Say an attacker tries to do a fraudulent transaction, then:

- The fraud detection part checks, in the database after the transaction's execution, if the transaction was fraudulent or not. This part is easier if we have more data, i.e., more previous fraudulent transactions.
- The fraud prevention part checks, before executing the transaction, if it is legitimate and should be executed.

Note that, in both cases, if we use a machine learning algorithm, it should be trustworthy, namely interpretable, because we have to be able to prove why a transaction has been flagged as fraud.

7.1.1 Fraud prevention

Frauds are usually prevented by securing authentication. In particular, we should always require strong user authentication. Lately, many technologies have been developed to ensure strong authentication some of which are:

- 3D secure (for payments).
- Multi-factor authentication.
- Apple Pay and Google Pay (for payments).

Let us consider 3D secure as an example to understand how fraudulent transactions are prevented from being executed. 3D secure asks a confirmation of the user's identity before approving and finalising the transaction. The identity is usually verified through a physical device. Like all techniques related to frauds, such devices have evolved with the fraudsters that tried to bypass them. The devices used for authentication usually generate a code using something like a pseudo-random number generator (or a ratchet) which is synchronised with the server. When the user wants to authenticate the device and the server generates the next number in the sequence and, if the number inserted by the user is the same as the one generated by the server then the user's identity is verified. The devices used for authentication should be tamper-proof or tamper-evident, namely the device should break if tampered with or it should be evident that the device has been tampered with. Moreover, authentication devices should be time dependent to make it harder to execute a man-in-the-middle attack by reusing previous codes. Recently, physical devices have been replaced by phones which work exactly like external devices. Using an app on a phone is a good idea because:

- It's time-dependent.
- The application is usually sand-boxed, hence, if the application is compromised, the attacker can't get access to all user's data.

The identity of a phone is related to the SIM which runs the phone, hence fraudsters have evolved to mount SIM replacement attacks which allow them to take control of a user's SIM and pass the authentication check. In particular, fraudsters exploit the fact that SMS are used to verify that the OPT application (i.e., the application on the phone that generates the OTP) is connected to a user. Fraudsters use phishing or social engineering in general to trick phone operators into getting a copy of the user's SIM.

7.2 Expert-based fraud detection

The most basic approach used when implementing fraud detection is based on human expertise, hence on analysts that analyse data and decide if a certain transaction is malicious. This solution directly exploits the domain knowledge of a fraud analyst, hence it's quite easy to implement since it only requires hiring some expert.

More precisely, an expert-based fraud detection system is based on a set of rules, for detecting a fraud, that are written by an expert who uses his or her knowledge to define the rules. A fraud

analyst knows how to gather and collect data to extract knowledge from it and detect frauds, hence one can exploit such expertise to build a fraud detection system.

Rules aren't however enough, in fact, an attacker could employ new fraudulent mechanisms which are not detected using our set of rules. This means that, whenever a transaction is suspected to be fraudulent, the expert has to manually analyse it to validate if it's actually a fraud. This might look like a disadvantage, however, it's also an opportunity to discover new fraud mechanisms. In particular, by analysing possible frauds, an expert analyst can discover new mechanisms and immediately address them.

7.2.1 Fraud management

Whenever an analyst confirms that a transaction is fraudulent, he or she can take two different actions:

1. **Corrective measures**. Corrective measures aim at resolving the fraud and correcting the consequences. When one case is handled we don't have to stop to that case but we also have to analyse the actions that led to the fraud. Such retrospective analysis is then used to understand if any previous case with similar characteristics has not been detected and correct them, too. Note that, when handling a fraud that happened in the past, the more we wait, the worst it is and the harder it is to resolve and correct it.

2. **Preventive measures**. Preventive measures aim at preventing future frauds. Practically, we want to extend the system used to fraud detection by integrating the mechanism used by the detected fraud.

7.2.2 Rule-based engine

Expert-based systems use a rule-based engine to detect frauds. This means that the analyst's expertise is used to build a set of rules which are then matched against a transaction to understand if such transaction is fraudulent or not.

Rule-based engines allow to detect the **red flags of frauds**. A red flag is something that deviates from the average behaviour, namely an anomaly or a deviation from normality. Red flags are always worth analysing to detect a fraud, hence they can be transformed to rules.

Disadvantages

Now that we know what expert-based systems are, we should analyse their characteristics and, in particular, their disadvantages. The main disadvantages of fraud detection systems are:

- They are **expensive to build**. A rule-based system relies on the rules built by an expert, hence it requires time to build such a system because the expert needs to list a set of rules to apply.

- **It's not always possible to recognise a fraud using a set of rules** and a if-then patter.

- They can **detect only known patterns** since they are based on the knowledge of the analyst, hence on what the analyst has already seen. As a consequence, analysts have to manually update and maintain the system because novel patterns have to be added when discovered. This means that expert-based systems are **difficult to maintain**.

Summarising these points, we can say that the main drawback of an expert-based system is that it requires continuous maintenance and upgrade to keep the pace with the fraudsters that use new patterns.

Since expert-based systems are updated by the analysts, we can notice that the more time has passed, the easier it is to detect a fraud and develop rules. This is because the more time has passed, the more data analysts have to say if a behaviour is fraudulent or not. Interestingly, this facts leads to a circular behaviour in which:

1. One fraudster uses a new fraud technique, which is not detected by fraud detection mechanisms.
2. Other fraudsters find out that the new fraud technique is effective (remember that the world of frauds is a complex environment in which people exchange knowledge) and start using it.
3. Since many fraudsters use the novel technique, it's easier for the analysts to detect the new mechanism and integrate it in a rule-based fraud detection engine.
4. The novel technique gets used less and less and the cycle repeats.

As a result, fraud-detection systems are becoming better and smarter since we have more data to built them.

7.3 Automated fraud-detection systems

Since data can be used to develop more accurate fraud detection systems, we are trying to move from expert-based fraud-detection systems to automated ones which

- **have higher performances** and,
- **require less human involvement**.

Namely, we can achieve better performances at a lower cost. Note that automated systems aren't completely independent from expert-based systems, in fact, when building an automated system we always have to start from an expert-based one. This is because we still need an expert to collect the information required to build the automated system. In other words, expert-based and automated fraud-detection systems are complementary.

7.3.1 Data-driven fraud-detection systems

Automated fraud-detection systems heavily rely on data, hence they are called data-driven fraud-detection systems. The reasons why data-driven systems are so successful are:

- They are very precise.
- They are operational efficient.
- They are cost efficient.
- There is a growing amount of interest in them.

Let us now analyse each point more in detail.

Precision

The precision is the performance of a fraud-detection system, hence the capability of detecting a fraud. Usually, organisations have a limited investigation capability since expert-based fraud-detection requires to hire a group of analysts and the organisation can only analyse the transactions that the analysts are able to analyse. Automated systems help analysts, in fact they allow to make better use of the analysts by highlighting the transactions which look more fraudulent. Basically, automated systems try to maximise the number of frauds discovered by the analysts.

Operational and cost efficiency

Automated fraud-detection systems reduce the cost to deploy and use them but also the time required to detect a fraud. This means that they increase the cost and operational (i.e., time constraint to satisfy when detecting a fraud) efficiency of a system, with respect to a system with no automated component. Note that operational efficiency is key when we want to analyse transactions that happen very fast, for instance, when analysing credit card transactions.

7.4 Fraud-detection techniques

Even if fraud detection systems are continuously innovated, fraud detection is still a very hard problem to solve.

7.4.1 Fraud-detection challenges

Fraud-detection systems have to face many challenges:

- Fraud detection systems have to deal with the **dynamic nature of frauds**.
- Fraud detection systems should have a **good detection power**.
- Fraud detection systems have to deal with the **skewness of data**.
- Fraud detection systems should be **operational efficient**.

Let's analyse these challenge one-by-one.

Dynamic nature of frauds

Fraudsters always try to build fraud techniques that evade fraud-detection systems. This fact has to be taken into consideration when building a fraud-detection system. In particular, fraudsters usually probe a fraud-detection system until they are able to find a transaction which is not detected. Recently, adversarial machine learning has also been used to probe a fraud-detection system. Fraudsters have gone even further by using poison attacks. These attacks use transactions that don't only evade the fraud-detection system but also are forged in a way that, when used for training the system itself, make the model shift in advantage of the attacker. An attacker might also develop his or her own fraud-detection model, trained on real world data (e.g., leaked transactions of a bank) and develop transactions that are able to evade the fraud-detection system. He or she can then try to use the same transactions on the target system (i.e., the real one the attacker wants to attack) since what works on a machine learning model might also work on another.

As a result, since frauds can be very dynamic, fraud-detection systems must be able to adapt and change.

Good detection power

Fraud-detection systems should be accurate, namely they should have:

- A **low false positives rate**. A false positive is a transaction that has been marked as non-fraudulent but that, in reality, is. A false positive is a cost since we have to deal with the client which has been annoyed even if the transaction was not fraudulent. Having an high false positives rate might encourage the client to change bank.
- A **low false negatives rate**. A false negative is a transaction that has been marked as fraudulent but that, in reality, isn't. A false negative is a cost since we are authorising a transaction that is not legitimate, hence we are losing money directly and indirectly (to repair the damage caused by the fraud).

Note that we aren't considering accuracy in the classical way (i.e., $\frac{TP+TN}{ALL}$) since fraudulent transactions are usually a very small part of a dataset (e.g., 1%), hence the accuracy of a system that marks every transaction as non-fraudulent is very high (which is something we don't want). Having both a low false and positive rate is impossible, hence we should find a trade-off.

Skewess of data

The datasets usually contain very few samples of frauds (we say that the data is skew), hence it's hard to find frauds. This problem is usually referred to with the name of **needle-in-the-haystack problem**.

Operational efficiency

Operational efficiency is key since the fraud-detection system might have to classify a transaction in a matter of seconds (e.g., a credit card transaction). Moreover, we have to consider that the system must be able to scale over large amounts of data and still operate efficiently.

7.4.2 Supervised and unsupervised techniques

Fraud-detection systems usually exploit some machine learning technique. The techniques used by ML-based systems can be divided into:

- In **supervised approaches** (also called predictive techniques) we have the set of features that define a fraud and we try to match them to the transactions of a data-set to find the transactions that are fraudulent. Consider, for instance, the game *Where is Wally?*. Note that, even if we know how a fraudulent case is done, we can still rise false alarms, which will cost money to the company.
- In **unsupervised approaches** (also called descriptive techniques) we don't know the set of features that define a fraud, hence we can only detect patterns that differ from the usual behaviour.

Unsupervised learning techniques

Unsupervised learning techniques can be applied whenever we don't have labels (or we don't need them) since these techniques don't require labels to train the model. This means that unsupervised

learning models are able to highlight behaviours that deviate from the standard ones (i.e., from the more present in the dataset).

An unsupervised approach is very useful in fraud detection since it allows recognising even novel fraud techniques since the model isn't trained to recognise a specific pattern but to recognise something that is different from the usual. This means that unsupervised learning techniques are complementary to expert-based systems since we can:

1. use expert-based systems to recognise known frauds based on the analysts' knowledge and,
2. use the unsupervised learning model to recognise patterns that are deviating from the normal behaviour but are not recognised by the expert-based engine.

It's not all positive though, in fact, unsupervised learning techniques also have some limitations:

- Unsupervised learning techniques still **require some human intervention** since not everything that is flagged as fraudulent (i.e., that deviates from the norm) is actually fraudulent, hence a human should confirm the output of the model. In other words, unsupervised learning techniques have a high false positive rate.
- An attacker can **mimic the behaviour of a user to bypass the fraud-detection mechanism**.

This means that unsupervised learning approaches aren't enough to build a complete fraud-detection system.

Supervised learning techniques

Supervised learning techniques try to distinguish between normal and unusual behaviour using data which is already labelled. This means that during training we have to tell the model what is a normal behaviour and what is not. This is very different from unsupervised learning since in the supervised learning case we are telling what is normal and what's not whereas in the unsupervised learning case we are letting the model understand what is normal and what's not.

Supervised learning models are very useful not only for detecting frauds. In particular, these techniques can be used to:

- **Detect frauds**, namely classify a behaviour as fraudulent or not fraudulent.
- **Predict frauds**, namely predict when a fraud will happen.
- **Estimate the amount stolen with the fraud**.

Supervised learning models aren't perfect, too. In fact, the main limitations of this class of machine learning techniques are:

- Supervised learning models **require historical data to learn**. Note that, in the case of supervised learning it's even harder to obtain data since we need to also label every data point.
- Supervised learning models **have a hard time detecting novel fraud techniques**.

7.4.3 Active learning techniques

Since every type of fraud-detection technique, namely rule-based, supervised learning and unsupervised learning, isn't enough to build a good system, we have to use the three systems together. Active learning exploits supervised and unsupervised learning techniques and adds some human intervention to build a system that has the best of each technique. In particular:

1. we start from an unsupervised learning model to recognise unusual behaviours,
2. we provide the output of the unsupervised learning model to an analyst that labels the fraudulent transactions and,
3. we train a supervised learning model using the data labelled by the analyst.

Active learning is particularly useful when we have a large labelled dataset. We can deploy an unsupervised learning model but it's not strong enough and it requires human intervention, hence we would like to train a supervised learning model, too. Active learning allows to build and maintain a dataset for supervised tasks, too. Let us consider a large dataset of unlabelled data. Giving the whole dataset to an analyst to label it would require a huge amount of time. We can then employ an unsupervised learning model to exclude transactions that are clearly legit (the model separates legitimate transactions from fraudulent ones). Now we have a much smaller set of transactions which the model thinks are fraudulent. These transactions (which are way less than those in the original dataset) might contain false positives (i.e., legitimate transactions flagged as fraudulent) hence we can give them to an analyst who can properly label them. This process can be iterated. For instance, once a certain number of transactions have been added to the unlabelled dataset (i.e., the transactions executed by the users) we can pass them to the unsupervised learning model, which selects the subset of possible fraudulent transactions and then pass the subset to an analyst who labels them.

7.4.4 Graph and network analysis

Graph and network analysis extend the abilities of the fraud-detection system by learning and detecting characteristics of fraudulent behaviour in a network of linked entities. Including an extra source of information in the analysis, i.e., the relationships between entities, contributes to uncovering particular patterns indicating fraud.

7.5 Developing a fraud-detection system

We've seen that a good fraud-detection system should integrate a rule-based engine, a supervised learning model and an unsupervised learning model. In particular, we should, in order:

1. develop and expert-based rule engine since, at the beginning we only have the expert's knowledge, then
2. design and train an unsupervised learning model since we can use the data collected by the rule engine, and then
3. design and train a supervised learning model since we finally have labelled data.

Note that this workflow considers a company that starts from 0, however, we can skip some steps if we already have some data. For instance, if we already have labelled data, we can directly deploy a supervised learning system since it is the most accurate one.

This scenario is different from the one presented when discussing active learning (hence it's not an issue that the procedure is different). In particular:

- In this case we are starting from an empty dataset and we want to collect data to build a complete fraud detection system.
- Active learning is applied when we already have a lot of unlabelled data and we want to use it for training a supervised learning model, too.

7.5.1 Fraud management cycle

Managing frauds doesn't only mean detecting them. In particular, we talk about a fraud management cycle (Figure 7.1), which is made of:

1. **Fraud detection**.
2. **Fraud investigation**. In this phase a fraud analyst analyses a a possibly fraudulent transaction.
3. **Fraud confirmation**. In this phase a fraud analyst says if a possibly fraudulent transaction is actually fraudulent.
4. **Fraud prevention**. If the analyst confirms the output of the fraud-detection system, then the fraud-prevention system is updated.
5. **Automated detection algorithm**. If the analyst confirms the output of the fraud-detection system, then the fraud-detection system is updated.

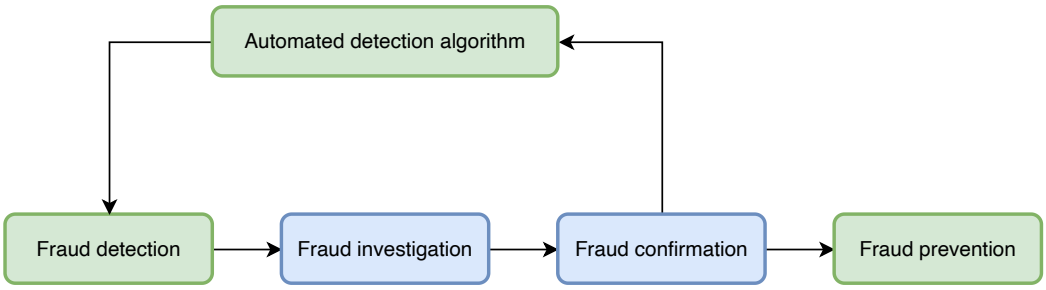


Figure 7.1: The fraud management cycle (blue tasks are performed by humans, green ones are automatic).

The fraud management cycle includes a phase in which we update the fraud-prevention and detection systems. Updating these systems might require a full retrain of a machine learning model (which might be very expensive), hence we should find the optimal update frequency. Namely, we need a trade-off between keeping an updated system at a very high cost or keep a system with old features, which is cheaper. Some way of understanding when we should update a systems are:

- When we have a good amount of new data.

- Analysing the time span in which a fraud is used (the more a fraud technique is used, the more important it is to update the system).
- Analyse the detection performance of the system. We should update the model when the model starts performing poorly.

Online models (i.e., non-parametric ones) don't require training hence updating a system is cheap. This allows to update the system very frequently but it might lead to some problems. In particular, an attacker might poison the dataset by sending appositely crafted transactions.

7.6 Fraud analytics process

Every time we want to deploy a data-driven approach we should structure our workflow in three main phases, each made of different steps:

- **Preprocessing**. In this phase we identify the data we want to work with, we clean, we filter, transform and integrate it. The preopocessing phase usually takes around 50 % of the total time required to develop a data-driven system. This is because the preprocessing phase is the most important part of the development and deployment of a data-driven system. An error in this phase influences the following phases and transforms in a bigger error.
- **Analytics**. In this phase we build the model using the preprocessed data.
- **Post-processing**.In this phase we analyse and interpret the model obtained to understand if it's working or not. In particular, in this phase we don't only want to understand if the model can detect previous fraudulent patterns but also unseen ones (i.e., we want to verify if the model is able to generalise the concept of fraud).

A model can be put in production only at the end of this workflow, hence when every output of the model is validated and approved.

Note that the success of a fraud-detection system doesn't depend only on how it performs with respect to the classical machine learning problems but also to:

- **How much user friendly is the output of the model**. This is because the output of a fraud-detection system has to be analysed by an analyst, hence it has to be human-friendly.
- **How the output of the model can be integrated with other applications**. This is because companies use many different applications hence it's better if the fraud-detection system can easily interact with the other components.
- **How easy it is to test every step of the model's development phase**.

7.6.1 Characteristics of a good fraud analytics process

The key characteristics of a good fraud analytics process are:

1. **Performance**.
2. **Interpretability**.
3. **Operational efficiency**.

4. **Economical cost**.
5. **Regulatory compliance**.

Let us now analyse each characteristic more in detail.

Performance

The performance of a process can be evaluated using accuracy, precision, recall, F1 score. These aren't the only metrics that we can use to evaluate the performance of a model, in fact some can't be always applied. In general, we want to build a model that is not overfitting. In fact the model should be able to generalise over the example shown.

When measuring the performance of a model we should also consider the the cost of handling the data, in fact if we have too many false positives, we have the cost of calling the clients for false alarms whereas, if we have too many false negatives, we have the cost of the money stolen.

Interpretability

When we build a model for fraud detection we want it to be interpretable (i.e., we want to make it easy to understand why it has generated a certain output). This characteristics limits the technologies that can be used to build a fraud detection system since many models (e.g., neural networks) are hard to interpret. Interpretability is key since the result of a fraud detection system has to be analysed by a human. It has to be easy, for an analyst, to understand why the model thinks a transaction is fraudulent to validate the machine's output.

With respect to interpretability, machine learning models can be divided into:

- **White box models**. White box models are less powerful than black box ones, however are more interpretable. In fact, it's easy, for white box models, to provide an explanation for an output. As a result, white box models are cheaper since analyst require less time to analyse the model's output but might lead to a money loss. In fact, being less powerful, white box models might miss fraudulent transactions. Because of their characteristics, white box models are usually used to analyse past incidents (i.e., not for real time analysis) since the analysts have more time to analyse the results, hence they can look at the interpretation of the model's output. On the other hand, in real-time analysis the analysts don't have time to analyse the reasons behind the results, hence it's useless to have an interpretable output.
- **Black box models**. Black box models are more powerful than white box ones but are less interpretable. This means that black box models are more accurate and faster at providing a result but it requires more time to analyse their output. For this reason, black box models are used in real-time analysis since we need a precise and fast result.

Operational efficiency

Every part of the fraud analytics process has to be operational efficient (i.e., it has to take a reasonable time) since having a perfect model that takes too much time to compute an output is useless. Operational efficiency is particularly important when dealing with real-time analysis since we need a result in a matter of seconds.

Economical cost

A fraud detection system is, after all, a security mechanism, hence it still follows the rules of risk. Risk can be seen as the combination of the *value of the asset to defend*, *the threats posed to the asset* and *the vulnerabilities of the asset*. Since we can't control the threats, we can only balance the cost of deploying new security measures and the cost of repairing a future damage. This means that we always have to consider the value of what we are protecting since, if the asset has a low value, it's worthless to deploy super expensive security measures. Moreover, we have to remember that more money spent on security doesn't necessarily translates to more security. For instance, we could build a very expensive system, which is however misconfigured or hard to use for the user (hence it's a indirect cost).

Finally, we have to remember that costs can be divided into:

- **Direct costs**.
- **Indirect costs**. Indirect costs aren't immediately paid by a company. For instance, indirect costs are generated by systems hard to use, with slow performance or that reduce productivity.

Regulatory compliance

Machine learning models require data for training. Data is usually protected, depending on the country we are considering, by regulations. One might think that we could anonymise data that contains sensitive information but this will lead to less powerful models. For instance, if we use anonymous data, we can't build one tailor-made model for each client.

Chapter 8

Machine learning for fraud detection

8.1 Building process

The process of building a machine learning model for fraud detection follows the same rules as any other system, hence we can still split the development process into three phases:

- **Preprocessing**.
- **Analysis**.
- **Post-preocessing**.

Let's now analyse each phase with its sub-steps.

8.1.1 Preprocessing

The prepossessing phase aims at preparing the data used to train the model. This phase takes most of the development phase (even half of it) because it's fundamental to build a good model. One might think that the more data we have the better it is but this isn't always true. In fact, it's true that we want a lot of data (since it can help reduce overfitting) but we also want good-quality data. In particular, we want data that is:

- **not dirty**,
- **not old**,
- **not inconsistent** and,
- **without replicas**.

If our data doesn't have these characteristics we might obtain a thrash-in-thrash-out model, hence a model that performs poorly since it has been trained on bad data. For this reason, we want to filter the data in the dataset to keep only the good data points.

Sources of data

Data can come from many different sources which provide different types of information. Some examples are:

- **Transactional data**.
- **Contractual, subscription, or account data**.
- **Socio-demographic information**.
- **Surveys**.
- **Behavioural information**.
- **Unstructured data**.
- **Contextual or network information**.
- **Qualitative, expert-based data**.
- **Publicly available data**.

Among these types of data, let us analyse more in-depth the first one which is relevant for fraud detection. Transactional data is structured and contains all the information regarding a transaction (e.g., amount, source, destination, time). Transactional data is meaningful when considered alone, but it can be also aggregated to obtain statistics about the whole dataset (e.g., we can compute the average, the mode, the median and the maximum of an attribute).

Type of data

Data can be:

- **Continuous**.
- **Categorical**, namely with discrete values. Categorical data can be further divided into:
  - **Nominal**. Nominal data can take a limited set of values on which we don't define an order. An example is IBANs.
  - **Ordinal**. Ordinal data can take a limited set of values on which it's possible to define an order.
  - **Binary**. Binary data can only take two values (usually yes or no).

Sampling

The first step in data preprocessing is sampling. Sampling allows selecting a subset of historical data of the dataset which will be used to build the model. One might think that sampling isn't necessary since we have enough computing power to use the whole dataset. It's true that we have that computing power but the reason for doing sampling isn't related to that. In fact, we want samples to be representative of what we want to analyse (i.e., of the behaviour we want to analyse) and to be in the correct time window. This means that a good sample has to be

- **representative of future and present entities** (i.e., we have to extract samples that are relevant to the current situation), and

- in the **optimal time window**.

Choosing a good set of samples is not an easy task. We should select samples that are representative of future entities (i.e., entities not in the dataset). When choosing samples we have to decide if we want:

- New data, which is less but closer to the time period in which the model is used.
- Old data, which is more but further from the time period in which the model is used.

This means that we have to find a trade-off between representative data (i.e., new data) and robust data (i.e., more data, since more data means building a more robust model).

When sampling a dataset we should also consider bias. This means that we should neither take samples too specific for a certain context (otherwise the model will be biased on a certain situation) nor too generic (otherwise the model will be too weak). Consider for instance a fraud detection system for credit card transactions. Say that the month in which the transaction is done is part of the model. Every month could display very different behaviours with respect to the average (e.g., in December people spend more money than in other months because of Christmas), hence a generic month isn't representative of the whole year. This means that we can:

- Build one model for each month. This solution has the best performance but the highest cost since we have to build multiple models.
- Build a single model that uses averages over a period. This solution is the cheapest but also the weakest since we aren't using precise data.

A good trade-off might use a model which is between these two extremes.

Stratified sampling When sampling data, we should extract a subset which has the same distribution as the original dataset. This is to say that the sampled dataset should contain the same percentage of fraudulent transactions as the original dataset (i.e., if the fraudulent transactions are 1% of the total transactions, then we should have also 1% of fraudulent transactions in the sampled dataset). This is because datasets are skew, hence we should keep this property in the sampled dataset, too. This concept is called stratified sampling.

Visual data exploration

After sampling data we can visually analyse it. This phase can reveal many useful information about the dataset which can be used when building the model. In this phase, we usually plot the data (either individually or aggregated) using pie charts, bar charts, histograms and many more (e.g., using Python's `matplotlib`).

This phase is very important since the patterns that we can visually recognise could be recognised by the model, too. This means that we can use the information we collect in this phase to tune the model.

Exploratory statistical analysis

After visually analysing the data, we can turn to a more statistical approach. In particular, we can compute:

- averages,

- standard deviations,
- minimum and maximum,
- percentiles,
- confidence intervals

over the whole dataset or over the single classes (if considering labelled data). This analysis can be used to recognise recurrent patterns which might be used to build the model.

Handling of missing values

The dataset we obtained after sampling isn't always perfect. In particular, some data might be missing. For instance, if we consider structured data (i.e., data with attributes), some values for some attributes might be missing. This means that some data points might be partially complete. For instance, some transactions might miss the age of the sender, others the location where the transaction was issued. Handling missing values correctly is key to obtaining a good model. Values might be missing for different reasons. It could be an error or some data might be undisclosed or not applicable.

We can do three things to deal with missing values:

- **Replace the missing data**. For instance, we can use the mean or the mode of the values of the same attribute that is missing, or use a default or known value.
- **Remove the whole data point**. When it's hard to fix the messy data point we can remove it to avoid having bad data.
- **Keep an incomplete data point**. If the missing values are very significant because they are related to the fraud (which we can tell after having analysed the dataset) we might keep an incomplete data point.

Deciding which action to take is a matter of understanding if a missing value is related to the target:

- If the value is related to the target we should keep it.
- If the value is not related to the target we either remove it or replace it.

Handling of outliers

A dataset could also contain outliers, namely values which deviate from the norm (i.e., that deviate from the usual values). Many outliers are completely normal and not related to a fraud, hence we should correctly deal with them.

The first thing we should do is check if something is clearly an error. For instance, the value 300 for the age of a person is clearly an outlier. Some outliers might not be outliers on their own but could be when combined with other data. In this case, we talk about multi-dimensional outliers. For instance, a tuple with `birth:1980, type:child` is clearly an outlier even if the single values are not. To deal with univariate outliers we observe the maximum and minimum values using:

- **Histograms**.

- **Box plots**. A box plot represents five important properties of a dataset: the median, the first and third quartile, the maximum and the minimum. First we sort the samples (using a certain feature as sorting key) and we find the median M. Then we take the samples before the median and we compute the median of that subset of samples. We call the first quartile Q_1 this median. We do the same for the samples after the median M. The median of that subset is called the third quartile Q_1. A box is drawn from the first to the third quartile. The minimum and maximum values can be computed straightforwardly as the maximum and the minimum or by using the interquartile range (IQR),

  $$IQR = Q_3 - Q_1$$

  The maximum value max can be defined as $1.5 \cdot IQR$. Figure 8.1 depicts an example of box plot.

- **Z-scores**, which measure how many standard deviations an observation lies away from the mean. The z-score for a feature x of sample i can be computed as

  $$z_i = \frac{x_i - \mu}{\sigma}$$

  were μ and σ are the the mean and the standard deviation of the feature over the whole dataset.

Dealing with multivariate (i.e., multi-dimensional) outliers is more complex and usually requires:

- An expert-based system (i.e., a human).
- Fitting regression lines and inspecting the observations with large errors (using, e.g., a residual plot).
- Clustering or calculating the Mahalanobis distance.

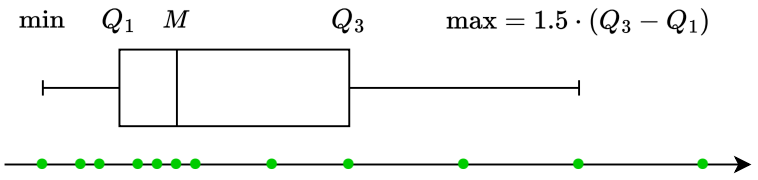


Figure 8.1: A box plot.

After having recognised and classified outliers (i.e., differentiated between valid and invalid outliers) we can:

- Treat invalid outliers, namely outliers which don't make sense (e.g., the guy born in 1980 who declared to be a child), as missing values.
- Impose an upper and lower value on an attribute and set the valid outliers, namely those that make sense, to the closest one.

Data standardisation

After having defined an upper and lower bound for each dimension, we should standardise or normalise data. Namely, we should shrink every dimension in a fixed interval (e.g., $[0, 1]$). This is very important, for instance (but not only), if we want to compute the distance between two points. Say for instance that a transaction is described by the time at which it has been done (as the distance in seconds from a specified date) and the value sent (in euros). These two attributes have a very different order of magnitude, hence if we compute the distance, using whatever distance metric (e.g., Euclidean, Manhattan), the time elapsed might be dominant over the value spent.

Categorisation

For categorical variables, it is needed to reduce the number of categories. The basic methods for doing categorisation are:

- **Equal interval binning**, which means creating bins with the same range. Practically, we consider all the output classes and we divide them into bins, each containing the same number of classes. This means that a bin might contain more samples than another. Consider for instance a dataset with 4 output classes C_0, C_1, C_2 and C_3. If we want to create two bins, we can evenly split the classes into two bins $B_0 = [C_0, C_1]$ and $B_1 = [C_2, C_3]$. If the dataset contains 10 points belonging to class C_0, 10 to class C_1, 1 to class C_2 and 3 to class C_3, then the first bin contains more samples.
- **Equal frequency binning**, which means creating bins with the same number of observations. Practically, we create bins in such a way that the bins contain more or less the same number of elements. If we consider the distribution used for the equal interval binning example, we can create $B_0 = [C_0, C_2]$ and $B_1 = [C_1, C_3]$.
- **Chi-squared analysis**.
- **Pivot Table**.

For continuous variables, by categorising the variable into ranges, non-monotonicity can be taken into account.

Variable selection

Datasets usually have many dimensions (i.e., many attributes) but only some are relevant to the model. This means that it's useful to select the attributes, also called features, that can contribute to the model's output and discard the others.

The easiest way to select features is by using filters. The main problem with filters is that they only consider one dimension, hence they don't consider the relationship between dimensions. Principal Component Analysis (PCA) instead considers the correlation between dimensions. This technique represents the dataset using principal components, namely directions with decreasing variance (the first principal component is the one with more variance, the second is the second with more variance and so on). Then, only the first N components are kept so that we keep only the directions with more variance (i.e., those that might contribute more to the output since a dimension which is constant isn't very interesting). This method is very useful however it comes at the cost of reducing the interpretability of the model since the new directions are not the original ones and might not have an immediate meaning. Figure 8.2 shows an example of principal component analysis.

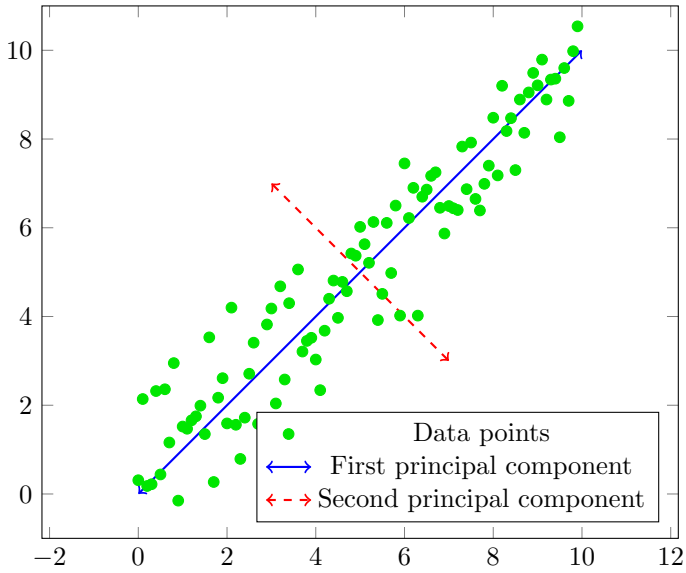


Figure 8.2: Principal components in PCA.

8.2 Unsupervised learning techniques

Unsupervised learning techniques for fraud detection aim at recognising behaviours which deviate from the norm. This means that one of the most important parts of unsupervised learning techniques is defining what is the norm, namely what is the normal behaviour. In particular, we can define the norm as:

- The behaviour of the **average customer** across a time period.
- The **average behaviour** of a given customer across a time period.

The difference between these definitions is that the former definition considers many customers whereas the latter is based on one customer only. In other words, in the first case, we are extracting an average customer, hence we are considering the median customer in our dataset in a particular period of time. The second definition considers instead one single customer and its average behaviour.

Once we have defined the norm, we can say that what is not the norm is anomalous, hence a possible fraud. For this reason, we can use the term anomaly detection. Anomalies are also called **outliers**, **exceptions** or **suspicious observations**. Formally,

> **Definition 8.1** (Anomalies)**.** *Anomalies (outliers, exceptions or suspicious observations) are observations that markedly appear to deviate from other members of the dataset where it occurs.*

Finding the norm is fundamental since a wrong norm means wrong anomalies. In general, we have to remember that we have to build a model that generalises as much as possible, hence we don't want the model to overfit (namely to fit the data instead of approximating the real norm). Unsupervised learning should be used when:

- Organisations are starting to do fraud detection.
- We don't have labelled data (i.e., when we can't use supervised learning).
- We know that the fraudsters' behaviour changes over time.

The norm can't be defined independently from the domain of application and from the time in which the model is used, hence the norm is dynamic, too. This makes finding the norm even harder. Moreover, unsupervised learning might be used by attackers to bypass a fraud detection system. In fact, when defining the norm, we are basically defining a boundary (i.e., a discriminant) between the norm and the anomalies. If an attacker knows or discovers the boundary, it can craft malicious transactions that are on the side of the boundary where the norm lies.

Note that, even if unsupervised learning models automatically define the norm, we still need to manually investigate the results returned by the model. In particular, we have to check if a transaction marked as fraudulent is actually fraudulent.

Now that we know the basics of unsupervised learning, we should look into some specific techniques. In particular, we'll analyse:

- **Graphical outlier detection**.
- **Statistical outlier detection** focusing on break-point analysis and peer-group analysis.
- **Clustering** focusing on hierarchical and non-hierarchical techniques.

8.2.1 Graphical outlier detection

Graphical outlier detection aims at finding the norm by visually analysing the dataset. The dataset can be plotted and then we can analyse what we see. The main **disadvantages** of this approach are:

- It's less formal and only limited to a few dimensions.
- It requires the active involvement of the end-user.
- It's cumbersome for a large dimensional data set.

8.2.2 Statistical outlier detection

Statistical outlier detection uses statistical measures to define the norm and detect outliers. The main statistical techniques are:

- **Z-score**. If the absolute value of the z-score is bigger than 3, the samples can be considered as outliers.
- **Fit a distribution, or mixture of distributions**. In this case, outliers are observations with small values for the probability density function.
- **Break-point analysis**.
- **Peer-group analysis**.
- **Association rule**.

Break-point analysis

Break-point analysis is an intra-account detection technique which allows us to understand if a user is doing something different from his or her normal behaviour. This means that we have to build a model for each user. Break-point analysis is based on the concept of break-point, which, in a time series, is a point in time where there is a sudden change in the behaviour of the time series. Figure 8.3 shows an example of a break-point. To apply break-point analysis on a time series (e.g., the money spent each day by a single customer) we have to:

1. Define a break-point, namely a point in time where we think we might find a sudden change in the user's behaviour.
2. Split the time series at the break-point to obtain a **old time series**, which contains the samples before the break-point, and a **new time series**, which contains the points after the break-point.
3. Compare the distributions of the new time series. The old time series is used as a model, hence as normal behaviour, to understand if the new time series behaves like the old one. Namely, we are trying to understand if the behaviour after the break-point is similar to the one before the break-point.

Deciding how to compare the distributions is key and we can use different metrics. For instance, we can measure distances or do a similarity test.

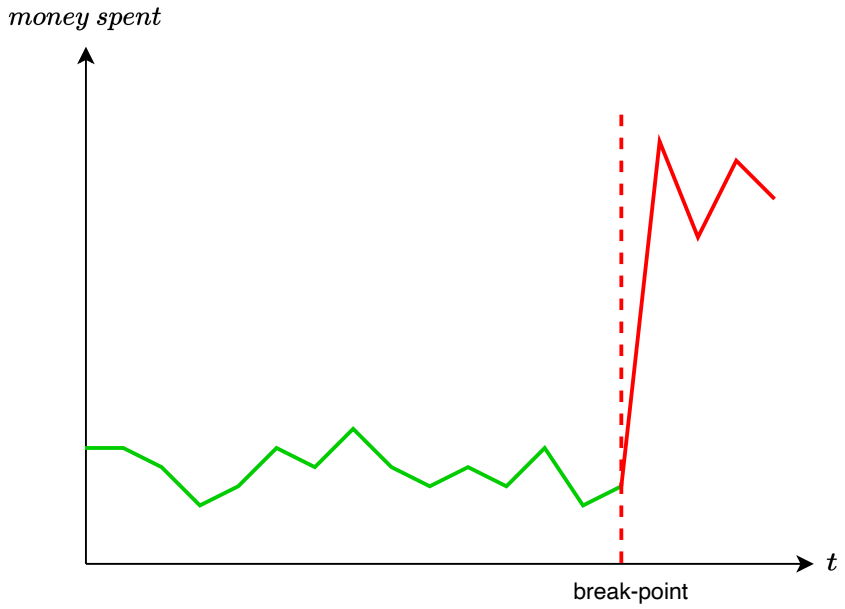


Figure 8.3: An example of break-point analysis.

Disadvantages The main **disadvantages** of this approach are:

- Defining the **break-point** is hard.

- We need **a model for each customer**, hence it's very expensive.

Peer-group analysis

Peer-group analysis is an inter-account detection technique which allows us to understand if a user is behaving differently from other users that normally have the same behaviour. The idea is to model a customer and pick N peers that behave similarly. When an account deviates from the behaviour of the group of peers, then we recognise it as anomalous behaviour. As we can see, in this case, we are considering multiple users together, hence we have to build fewer models (one for each group). More precisely, peer-group analysis can be divided into two steps:

1. **Peer recognition**. In this phase, we have to define the peer groups. This phase requires us to define:
   - How to build the group.
   - The number of peers that the group should contain. Finding the right number of peers isn't easy and usually boils down to a trade-off. In particular, if the group is too small the model is too sensitive to noise and if the group is too large, it's insensitive to outliers.
2. **Anomaly detection**. In this phase, we have to do a statistical test to define the anomalies comparing the behaviour of a user with the one of the group he or she belongs to.

Advantages The main advantages of peer-group analysis are:

- **Peer-group analysis can follow seasonality**. This happens because the norm of a user isn't modelled on the behaviour of that user in a specific time span but it's defined on the behaviour of multiple users in the whole time span. Consider for instance transactions over a year. During the month of December, expenses are usually higher. Break-point analysis would classify December's expenses as anomalies because they are very different from the norm (i.e., the other months). On the other hand, peer-group analysis uses the behaviour of many users during the whole year, hence the month of December is already normal since all the users buy more stuff in that month.
- **Peer-group analysis is less computationally intensive** because we need to create fewer models. On the flip side, creating one model requires more effort since finding peers is harder.

Disadvantages The main disadvantages of peer-group analysis are:

- **Attackers can inject peers and hide malicious transactions**. An attacker that controls multiple accounts can do expenses that mimic a malicious behaviour and then submit malicious transactions which are accepted if the user is associated with the malicious peers.
- **It's hard to define a good peer group**.
- **It's hard to say if a user deviates from the norm even if it's not malicious**. Namely, peer-group can generate many false positives.

8.2.3 Clustering

Break-point and peer-group analysis focus on local behaviour, namely on a single account or on a group of accounts. Clustering instead is able to analyse a whole set of transactions from different users. The idea behind clustering is to split data into groups called clusters. The clusters are built such that

- the homogeneity among nodes in the clusters, and
- the heterogeneity among clusters

are maximised. This means that we should build clusters that contain points as similar as possible and each cluster should be as different as possible from the other clusters.

Clustering techniques have to face the same problems as other machine learning techniques. In particular, we have to:

- **Define a metric to define clusters**. Clusters can be defined manually or using inter and intra-cluster similarity. Some distance measures for building clusters are the Euclidean distance, the Mahalanobis distance and the Minkowski distance. Each distance measure has its own advantages and should be used for different purposes. The distance measure depends also on the type of variable. For continuous variables, we can use, for instance, the Euclidean metric or the Pearson correlation or cosine measure. For binary variables, we can use the Simple Matching Coefficient or the Jaccard index.
- **Take care of the data by applying normalisation**, filtering data to keep only quality data and categorising discrete data.
- **Define a clustering technique**. Clustering techniques can be divided into hierarchical and non-hierarchical techniques.

When using clustering for fraud detection we would like to recognise large groups of legitimate transactions and small groups of fraudulent transactions. The former groups should be as far as possible from the latter.

8.2.4 Hierarchical clustering techniques

Hierarchical clustering techniques try to cluster data by aggregating or dividing data step by step. This means that we can:

- Start from as many clusters as the number of samples in the dataset and then start merging clusters to build larger clusters until we have only one big cluster containing all the samples. Techniques as such are called **agglomerative hierarchical clustering** techniques.
- Start from a single cluster containing the whole dataset and then iteratively split it until we have clusters containing one data point each. Techniques as such are called **divisive hierarchical clustering** techniques.

Deciding at which point the process should be stopped is of key importance and not trivial. Figure 8.4 shows an example of clustering by aggregation or division.

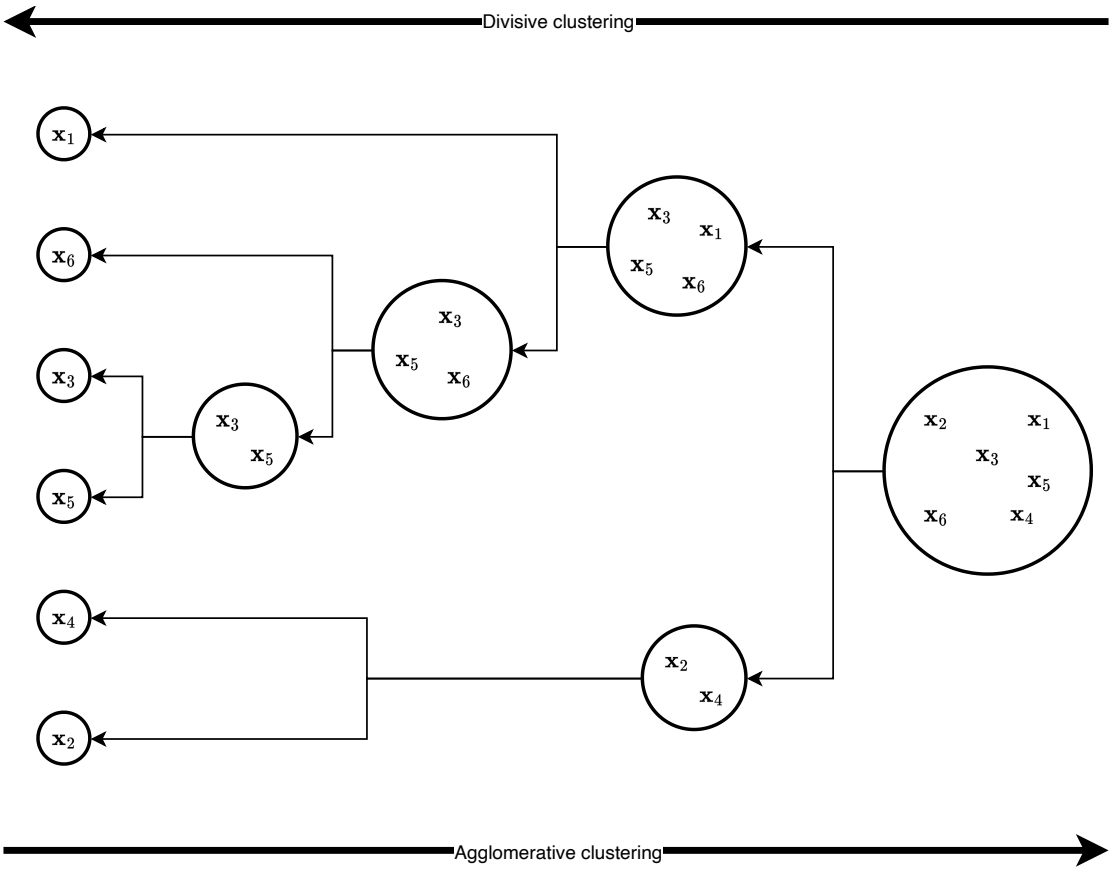


Figure 8.4: Agglomerative and divisive hierarchical clustering.

Measuring distances

We need to know how close two clusters (given that we have decided on a way to measure the distance between two points) are to build or split clusters. This is because we might want to join clusters that are close one to the other or build two sub-clusters of a cluster and evaluate their distance to understand if they should be split. As always, depending on the technique used for measuring the distance between clusters we might get very different results. The main techniques used for measuring the distance between clusters are:

- **Single linkage**. The single linkage distance is the distance between two clusters is the distance between the most similar elements of the two clusters.
- **Complete linkage**. The complete linkage distance is the distance between two clusters is the distance between the most dissimilar elements of the two clusters.
- **Average linkage**. The average linkage distance is the distance between two clusters is the average of the distances between all elements of the two clusters.
- **Centroid linkage**. The centroid linkage distance is the distance between the centroids of the two clusters.
- **Ward linkage**, computed as

$$D_{Ward}(C_i, C_j) = \sum_{x \in C_j} (x - c_i)^2 + \sum_{x \in C_j} (x - c_j)^2 - \sum_{c \in C_{ij}} (x - c_{ij})^2$$

Number of clusters

Deciding the right number of clusters is key to getting good models, in fact, if clusters are too small we might be too sensitive to noise whereas if clusters are too big we might not detect anomalies correctly. We can use two different tools for deciding the number of clusters:

- The **dendrogram**. A dendrogram graphically represents the process of splitting a cluster (or of merging clusters, if looked the other way around) using a tree. Each node of the tree represents a cluster with the root being the cluster with all the samples. Every time a cluster is split into two sub-clusters two branches are created. The vertical (or horizontal) scale gives the distance between two clusters amalgamated. Once we have built a dendrogram we can use it to decide the best number of clusters (in this case we choose directly the clusters). More precisely, we can do a horizontal cut at a level and take the clusters at that level. For instance, in Figure 8.6, we would take the clusters `{chicken, duck, pigeon}`, `{parrot, canary}`, `{own, eagle}`.
- The **screen plot**. A screen plot graphically represents the distance at which clusters are merged. We can find a good number of clusters by finding the elbow of the plot, namely the point where the plot has a rapid change in slope. Figure 8.7 shows an example of a screen plot for the dataset in Figure 8.5.

Advantages The main advantages of hierarchical clustering are:

- **We don't have to specify a-priori the number of elements in a cluster** nor the number of clusters.

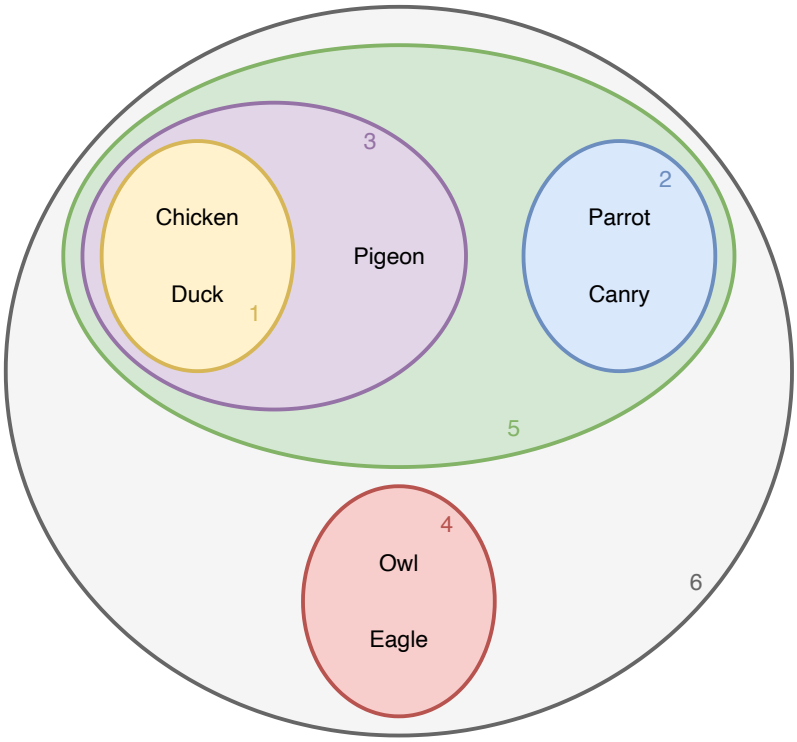


Figure 8.5: Birds dataset.

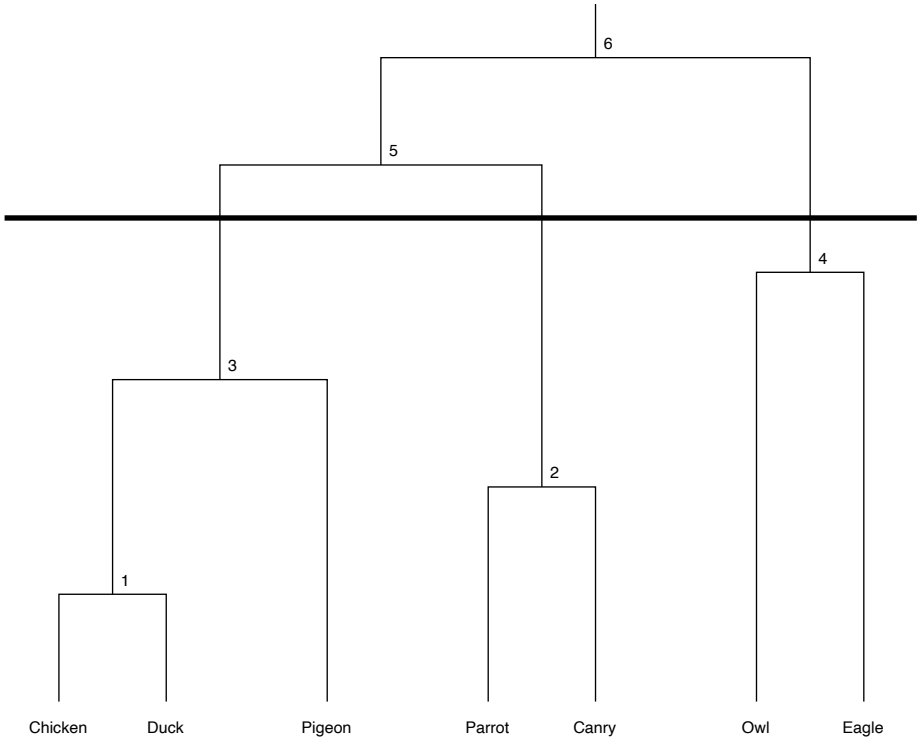


Figure 8.6: The dendrogram of the dataset in Figure 8.5.

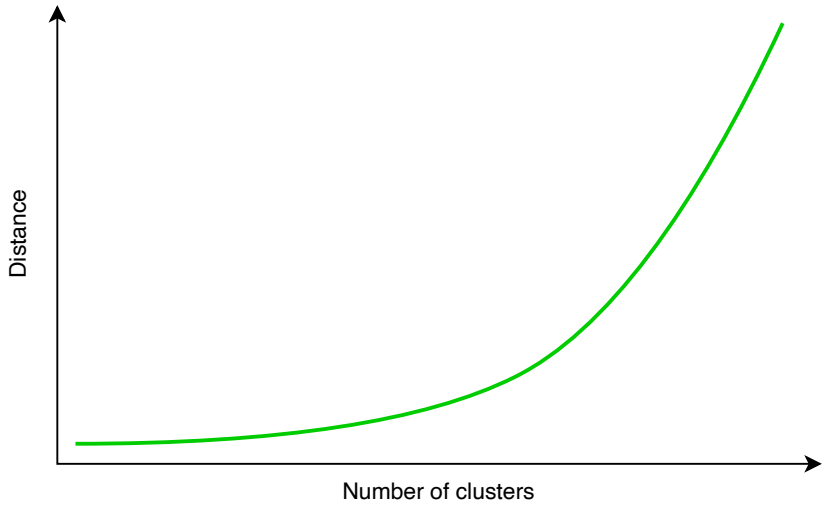


Figure 8.7: The screen plot of the dataset in Figure 8.5.

Disadvantages The main disadvantages of hierarchical clustering are:

- **Hierarchical clustering does not scale very well to large data sets**.
- **We need domain knowledge to interpret the results of the model**.
- **Hierarchical clustering is computationally expensive**.

8.2.5 Non-hierarchical clustering techniques

Non-hierarchical clustering techniques build clusters directly (i.e., no aggregation or division is involved). The two main techniques associated with non-hierarchical clustering are:

- **K-means**.
- **Self-Organising Maps**.

K-means

K-means allows us to iteratively find k clusters in a dataset. k-means clustering works as follows:

1. Select k observations (i.e., points in the dataset) at random. These are the initial centroids of the k clusters and are called seeds.
2. Assign each observation o to the cluster whose centroid is the closest to o. Namely, build the k clusters.
3. Compute a new centroid for each cluster using the points in the cluster.
4. Repeat from step 2 until stability, namely until centroids are stable, or after a certain number of iterations.

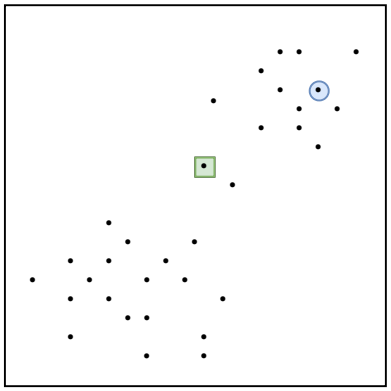
Figure 8.8 shows some iterations of k-means.

Disadvantages The main disadvantages of k-means are:

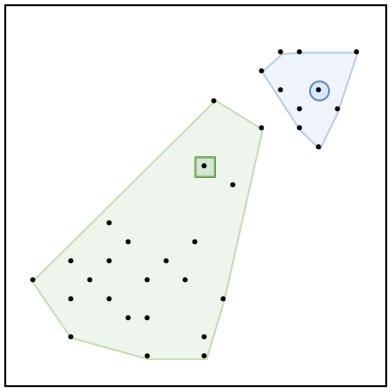
- **It's hard to define a distance measure for the data points**.
- **We need to decide the number of clusters** (i.e., k) **before building the model**. To solve this problem we can use hierarchical clustering to define k and then apply k-means. We could even repeat k-means for different values of k and keep the best model.
- **k-means is sensitive to outliers**. In fact, if the dataset contains outliers, the centroid might be shifted toward them, hence obtaining a less precise model. To solve this problem we can add a cluster for outliers.

Self-Organising Maps

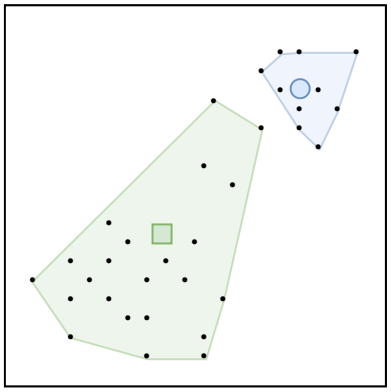
Self-Organising Maps (SOMs) exploit neural networks to cluster high-dimensional data on a low-dimensional grid of neurons. In particular, SOMs use a feed-forward neural network with two layers:



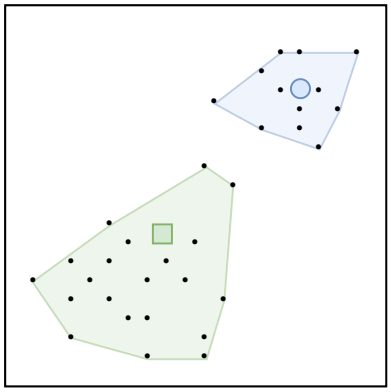
(a) The centroids are selected at random.



(b) Two clusters are built.



(c) New centroids are computed.



(d) New clusters are built using the new centroids.

Figure 8.8: Some iterations of the k-means algorithm.

- An **input** layer. The input layer stores a data point, hence it has as many nodes as the number of dimensions of the samples. We write

$$\mathbf{x} = [x_1, \ldots, x_n]$$

  to denote the input layer.

- An **output** layer. The output layer is a grid of neurons, which might have different shapes (e.g., rectangular as in Figure 8.9 or hexagonal).

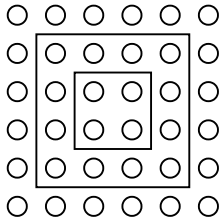


Figure 8.9: Different shapes of self-organising maps.

Each input is connected to each neuron in the output layer through a weighted connection. The weights that connect input i with the output neurons are denoted with

$$\mathbf{w}_i = [w_{1i}, \ldots, w_{Di}]$$

or equivalently we can consider a neuron and list the weights of the incoming connections.

$$\mathbf{w}_C = [w_{C1}, \ldots, w_{CN}]$$

Figure 8.10 shows an example of a self-organising map.

The weights are initially set to random values and then updated iteratively. To update the weights we have to:

1. Compute the distance between an input $\mathbf{x}$ and each neuron. The distance between an input $\mathbf{x}$ and a neuron c is computed as

$$d(\mathbf{x}, \mathbf{w}_c) = \sqrt{\sum_{i=1}^{N} (x_i - w_{ci})^2}$$

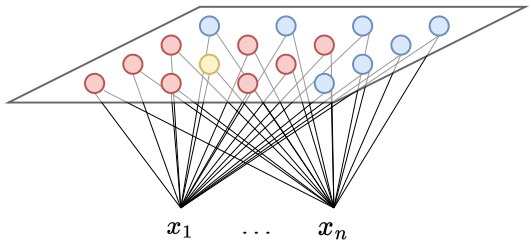
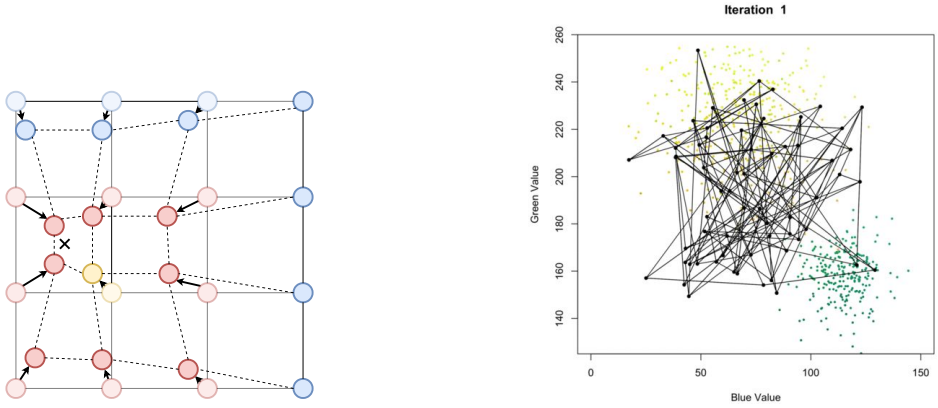


Figure 8.10: A self-organising map.



(a) A step of the training process of a self-organising network in a toy example.

(b) A step of the training process of a self-organising network in a real-world example.

Figure 8.11: A step of the training process of a self-organising network.

2. Take the neuron which is closer to $\mathbf{x}$. This neuron is called **Best Matching Unit**.
3. The weights of the weight vector of the BMU $\mathbf{w}_{BMU}$ and of its neighbours are updated using the learning rule
$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + h_{BMU,i}(t)[\mathbf{x}(t) - \mathbf{w}_i(t)]$$
where t is the time instant and $h_{BMU,i}$ is the neighbourhood of the BMU.

This process is repeated until stability, namely until the weights are stable, or after a fixed number of steps. Figures 8.11a and 8.11b (the second is applied on a real-world example) show one iteration of the algorithm used for updating the weights of the network.

Visualisation SOMs can be visualised using:

- **U-matrices**. A U-matrix adds a dimension (i.e., the height) to the matrix of neurons. Namely, each neuron is assigned a height which is computed as the average distance between the neuron and its neighbours. Dark and light colours can be used to indicate a large distance and can be interpreted as cluster boundaries. Figure 8.12 shows an example of a U-matrix.
- **Component planes**. A component plane visualises the weights between each specific input variable and its output neurons. Namely given an input point, we represent, for each neuron, the weight of the incoming arcs (for instance using colours, the darker, then bigger the weight). A component plane provides a visual overview of the relative contribution of each input attribute to the output neurons. Figure 8.13 shows an example of a component plane.

Advantages The main advantages of self-organising maps are:

- **We don't have to select a number of clusters a-priori**. Moreover, a SOM does not force the number of clusters to be equal to the number of output neurons.
- **They are interpretable** even when used with high-dimensional data.

For these reasons, SOMs are very popular.

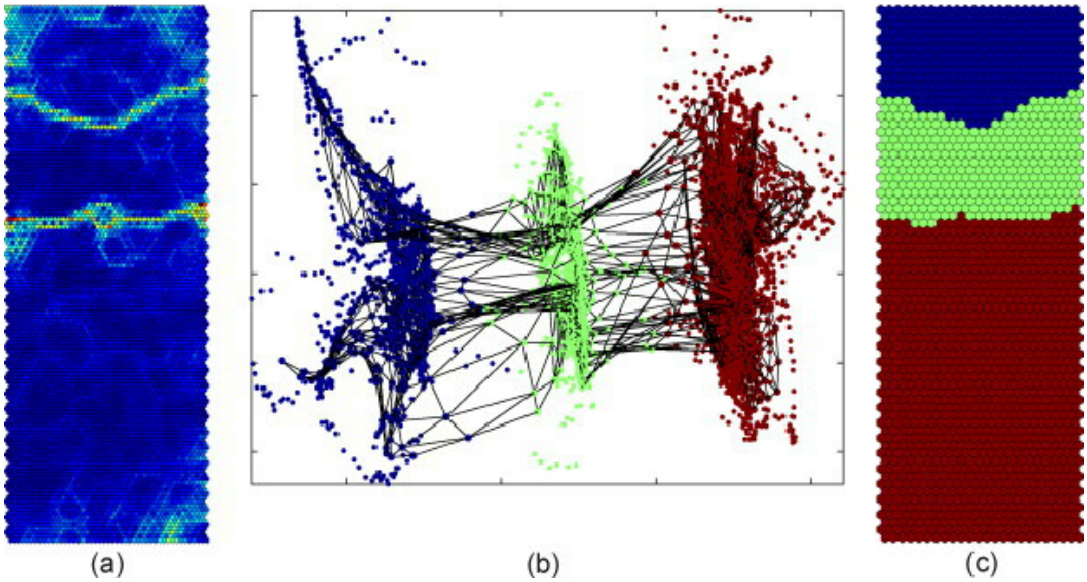
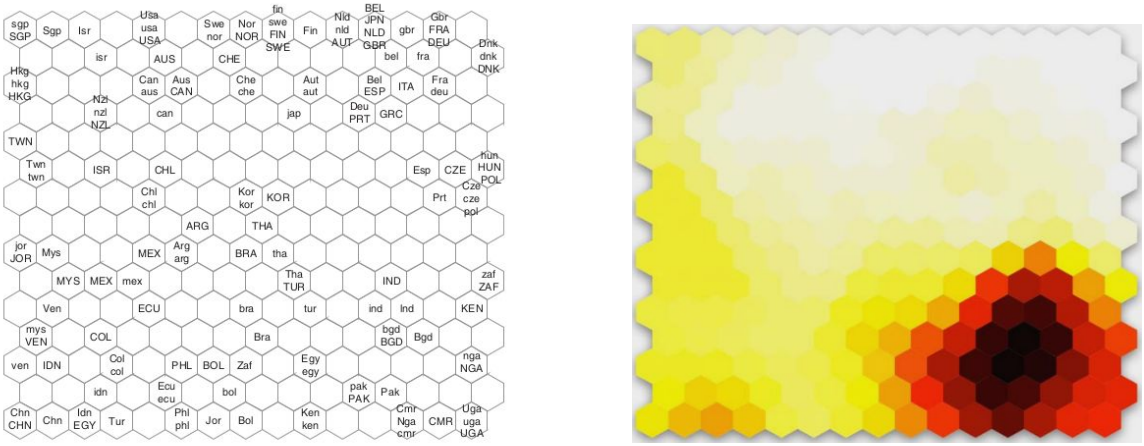


Figure 8.12: An example of visualisation using the SOM. (a) The U-matrix, (b) 2D projection which includes the SOM grid and the data, (c) The associated clusters map.



(a) The neurons of the self-organising map.

(b) The component plane for a specific feature.

Figure 8.13: Component planes used for clustering countries.

Disadvantages The main disadvantages of self-organising maps are:

- It is harder to compare various SOM solutions against each other since there is no objective function to minimise.
- Experimental evaluation and expert interpretation is needed to decide on the optimal size of the SOM (i.e., of the output layer).

8.2.6 Clustering with constraints

In some cases, we might have some domain expertise that can be useful to improve the power of the model. Clustering with constraints allows us to include this knowledge in the model by specifying constraints that the model has to follow. For instance, we can impose that two nodes must belong to the same cluster. In this case, each time an observation is (re-)assigned, the constraints are verified and the (re-)assignment is halted in case violations occur.

Clustering with constraints allows us to reduce the time required to find clusters and obtain clusters with some desired properties.

8.2.7 Evaluation

When building a model we also want to evaluate it. This is because we want to assess the goodness and the generalisation power of the model (i.e., how good the model is on unseen data). Evaluation can be performed:

- **Visually**. Visual evaluation consists of plotting the results obtained on unseen data and visually checking if they are coherent with what we expect. In other words, we explore data and graphically compare cluster distributions with population distributions across all variables on a cluster-by-cluster basis.
- **Statistically**. Statistical evaluation consists of using a statistical measure to understand if the results we obtained are coherent with what we expect. For instance, we can use the Sum of Squared Errors (SSE)

$$SSE = \sum_{i=0}^{K} \sum_{x \in C_i} distance^2(x, m_i)$$

to compute the sum of the distances between the centroid m_i of cluster C_i and the other points in C_i (where K is the number of clusters). Thanks to the SSE we can consider different clustering solutions, compute the SSE for each solution and then keep the solution with the lowest SSE.

As always we can't say if one technique is better than the other since the choice between the two depends on the context and the domain of application. Moreover, we might even use both in parallel.

8.3 Supervised learning techniques

Supervised learning techniques define a target variable, namely they assign each input to an output (which is the target value). For instance, we can assign a label to an input transaction to say if it's fraudulent or not. This means that we have to start from labelled data, namely from data for which we know the target value. As a result, we must use a dataset for which we know, for each transaction

if it is a fraud or not. This also means that we can only detect if a transaction is fraudulent or not but we can't define in general what is the norm.

Supervised learning tasks (or techniques, or problems) are split into two categories, depending on the nature of the target:

- **Regression problems** have a continuous target, which can be bounded or unbounded. An example of a regression problem is predicting the expense in a month by a certain user.
- **Classification problems** have a categorical target. This means that we are assigning an input to a class. In general, we can have as many target classes as we want and we talk about multi-class classification. A specific case of multi-class specification is binary classification in which we only have two classes. An example of binary classification is telling if a certain transaction is fraudulent or not.

As we have said, supervised learning algorithms require a labelled dataset, however, we don't always have one because:

- fraudsters can use the dataset to understand what frauds are classified as fraudulent and can use this information to elude the fraud detection system,
- releasing clients' transactions mines the clients' privacy, and
- banks do not want to show that some client has been defrauded.

Moreover, even if data is labelled, we can't be sure that the labels are correct since they are usually assigned by humans who might do mistakes or write labels for other purposes.

Now that we know the basics of supervised learning we should describe some techniques. In particular, we are going to analyse:

- **Linear regression**.
- **Logistic regression**.
- **Decision trees**.
- **Neural networks**.
- **Support vector machines**.
- **Ensemble methods**.

8.3.1 Linear regression

Linear regression is a regression model, hence it predicts a continuous value (i.e., the target is a continuous value). Linear regression models the target as a weighted sum of the input vector (i.e., of the features). If we call

- y the output of the model,
- $\mathbf{x} = [1, x_1, \ldots, x_D]$ an input point where x_i are the features (e.g., the amount spent and the date in a transaction),
- and $\mathbf{w} = [w_0, w_1, \ldots, x_D]$ the weights vector,

then we can model the target as

$$y = \mathbf{x} \cdot \mathbf{w}^T = \sum_{i=0}^{D} w_i x_i = w_0 + \sum_{i=1}^{D} w_i x_i$$

The weights vector is the model, in fact given $\mathbf{w}$ we can compute y for a certain input $\mathbf{x}$. This means that our goal is to find the best weights that better model the training data.

Training

Training a linear regression model means finding the weights $\mathbf{w}$ better approximate the target. In particular, we want to define a way to compute how good the model is. One example is using the Sum of Square Error SSE

$$SSE = \frac{1}{2}\sum_{s=1}^{D}(y_s - t_s)^2 = \frac{1}{2}\sum_{s=1}^{D}\left(\mathbf{x}_i\mathbf{w}^T - t_s\right)^2$$

where $\mathbf{x}_s$ is the s-th point in the dataset, t_s is the associated target and y_s is the value predicted by the model. The SSE computes how far the model is from the actual target value. Once we have defined a way to compute the goodness of a model we can minimise this function because we want the weight that minimises the error done by the model. This means that we have to compute

$$\frac{dSSE}{d\mathbf{w}} = \frac{1}{2}\frac{d}{d\mathbf{w}}\sum_{s=1}^{D}\left(\mathbf{x}_i\mathbf{w}^T - t_s\right)^2$$

Graphical representation

A linear regression model is a line in the input space that better models the dataset. Namely, the model is a line that better predicts the target of a given input $\mathbf{x}$. Figure 8.14 shows an example of linear regression.

8.3.2 Logistic regression

Linear regression can't be used for classification because even considering binary classification:

- **The target is not normally distributed** but follows a Bernoulli distribution. In fact, the target is either 0 or 1 (if we encode one class with the value 0 and the other with the value 1).
- **There are no guarantees that the output fits in the interval** $[0, 1]$. This means that we have to define a threshold and every input whose output is above the threshold is assigned to class 1, otherwise, it's assigned to class 0.

Logistic regression solves these problems by using a logistic function instead of a linear function to model the target. The new model is therefore

$$y = \frac{1}{1 + e^{-(\mathbf{x}\mathbf{w}^T)}} = \sigma(\mathbf{x}\mathbf{w}^T)$$

This function takes values between 0 and 1 (i.e., the output is bounded between 0 and 1). In fact, a logistic regression function models the probability of a point of belonging to a class. An example of a logistic regression model is shown in Figure 8.15.

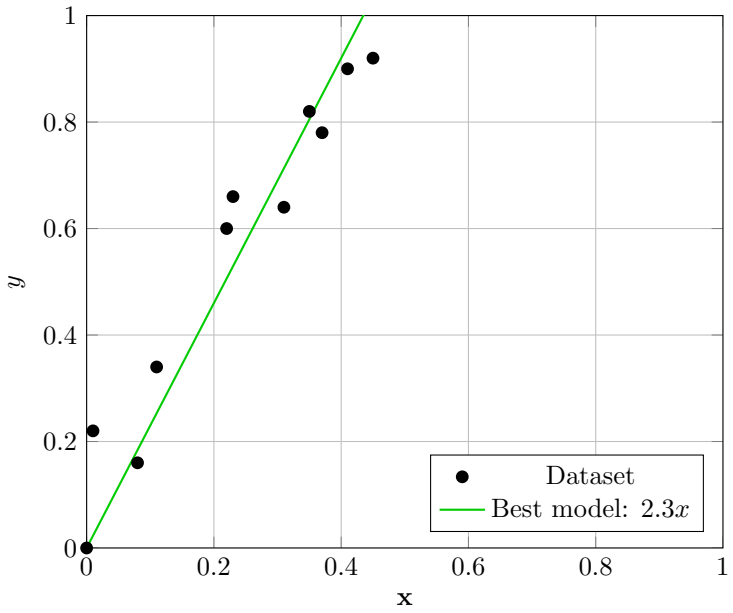


Figure 8.14: A linear regression model.

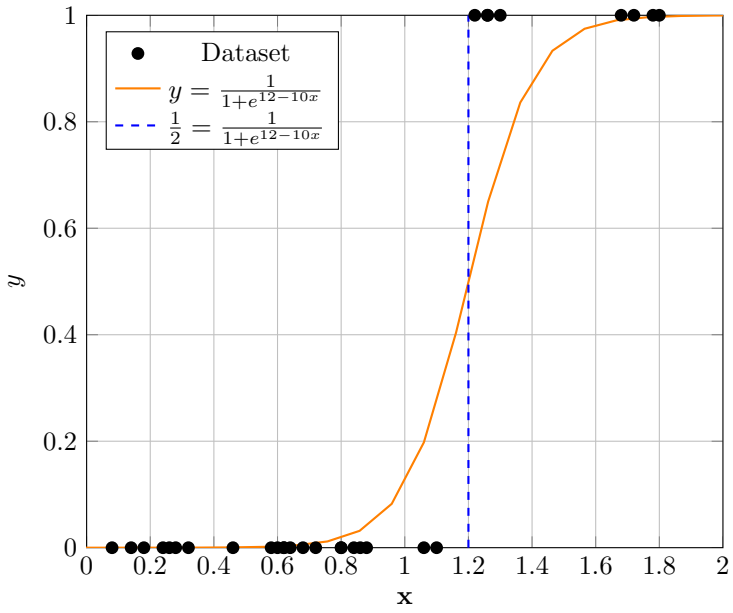


Figure 8.15: A logistic regression model.

Training

Training for classification works like training for regression. The only difference is the loss function we want to use for evaluating the model. Instead of SSE we use the maximum likelihood which is computed as

$$p(\mathbf{t}|\mathbf{x_1},\dots,\mathbf{x}_D,\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n} = \prod_{n=1}^{N} \sigma(\mathbf{x}\mathbf{w}^T)^{t_n}\big(1-\sigma(\mathbf{x}\mathbf{w}^T)\big)^{1-t_n}$$

The optimal weights, hence the optimal model, are obtained by minimising the maximum likelihood.

Graphical representation

A logistical regression model models the probability of belonging to a class. We can consider a value for such probability and say that a point belongs to class C_0 if the probability is higher than a certain value π. Let us consider $\pi = \frac{1}{2}$. Let us also consider a model with two input features,

$$y = \frac{1}{1+e^{-(w_0+w_1x_1+w_2x_2)}}$$

Having fixed a value for the probability, hence for y, we can write

$$y = \pi = \frac{1}{2} = \frac{1}{1+e^{-(w_0+w_1x_1+w_2x_2)}}$$

Now we can manipulate this expression to obtain

$$\begin{aligned}
\frac{1}{2} &= \frac{1}{1+e^{-(w_0+w_1x_1+w_2x_2)}} \\
2 &= 1+e^{-(w_0+w_1x_1+w_2x_2)} \\
1 &= e^{-(w_0+w_1x_1+w_2x_2)} \\
\ln(1) &= \ln\left(e^{-(w_0+w_1x_1+w_2x_2)}\right) \\
0 &= -(w_0+w_1x_1+w_2x_2) \\
0 &= w_0+w_1x_1+w_2x_2
\end{aligned}$$

This is the equation of a line in the plane (x_1, x_2). As a result we can state that a logistical regression model is the line (or hyperplane in case of multiple dimensions) that better separates the samples of a class from the sample of the other class. Figure 8.16 shows a regression model that splits the dataset in the case of data points with two features. If we were to represent the separating hyperplane in a dataset with one feature only we would obtain a point on the x axis. The separating hyperplane cal also be represented in the plane $(y, \mathbf{x})$ and, in case of a single feature, the hyperplane is a vertical line (the blue dashed line in Figure 8.15).

Variable selection

When building a regression model we might want to remove some variables which have little impact on the output. Reducing the number of variables

- simplifies the model (which also means that reduces its power) and,

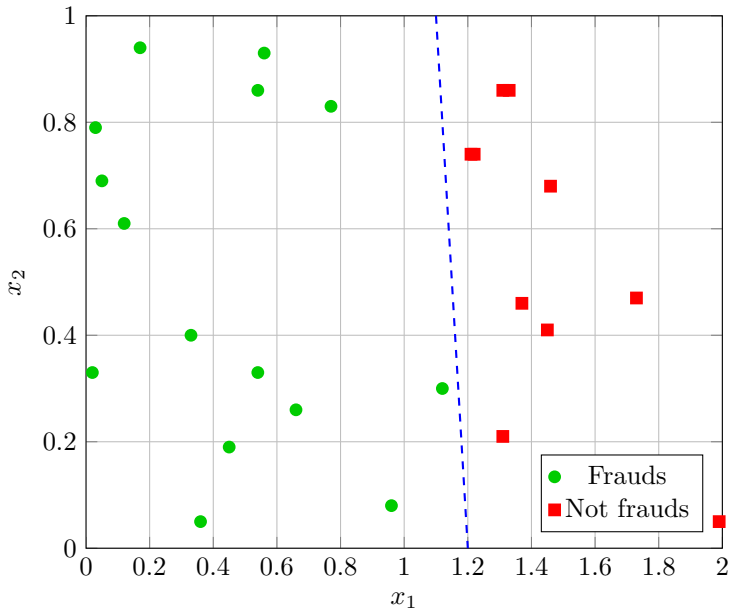


Figure 8.16: A logistic regression model that splits a dataset.

- makes the model faster to evaluate,

hence it's a very important task. Logistic and linear regression allow to automatically do variable selection. We do a statistical test based on a statistical hypothesis that verifies whether the coefficient of a variable is significantly different from 0. In particular, we perform

- a **t-student test** in the case of linear regression, and
- a **chi-squared test** in the case of logistic regression

to verify if it fits that distribution. Then we can compute the p-value, which can be interpreted as the probability of getting a more extreme value than the one observed. The p-value can be compared against a significance level to understand if a variable is significant or not. In particular,

- a low p-value represents a significant variable, and
- a high p-value represents an insignificant variable,

This means that, if we have a problem that is not complex from a feature space and allows us to apply such a simple regression model, then we have automatic ways to do feature selection before deploying the model.

The same approach can be applied to any model that has the same basic functioning.

We can use different approaches for deciding what variables should be removed (i.e., for navigating the variable space):

- **Forward regression**. Forward regression starts from the empty model and always adds variables based on low p-values.

- **Backward regression**. Backward regression starts from the full model and always removes variables based on high p-values.
- **Step-wise regression**. Step-wise regression is a mix of backward and forward regression. This technique is interesting because the problem of finding the best set of variables is an engineering problem, hence we have to deal with trade-offs between computational power and performance.

When selecting what variables should be kept and which should be removed, we also have to consider non-statistical measures. For instance, we should consider:

- **Legal issues**. Some features might not be used because the law doesn't allow them to. For instance, some data might be protected by the GDPR.
- **Operational efficiency**. It might be too expensive to keep a variable because it's too expensive to obtain or to pre-process.
- **Interpretability**. Some variables might be more significant and interpretable than others, hence it might be better to keep or remove them, respectively. Note that, the sign of the regression weights can indicate the relationship between a feature and a target. In particular:
  - If the weight is positive, when the feature increases, the target also increases.
  - If the weight is negative, when the feature increases, the target decreases.

8.3.3 Decision trees

Decision trees are classification machine learning models that model patterns inside the data with a tree. Each node of the tree specifies a branch condition and the leaves of the tree are the output classes. An input is forwarded from the root to the leaves and at each branch, the condition on that branch is evaluated. If the condition is true the input is forwarded on the one side of the branch, otherwise on the other.

Decision trees can be implemented in different ways. Each implementation handles in its own way three key aspects:

- **The splitting decision**.
- **The stopping decision**.
- **The assignment decision**.

Assignment decision

The assignment decision defines what class should be assigned to a leaf node. This decision is usually taken through majority voting. Practically we:

1. Pass every input through the tree and record, for each input, in which leaf it ends up.
2. For each leaf, we assign to a leaf the class with more representatives among the inputs that ended up in that leaf. Namely, if $(\mathbf{x}_1, C_1), (\mathbf{x}_2, C_4), (\mathbf{x}_3, C_1)$ ended up in leaf L_1, then we assign class C_1 to leaf L_1.

This type of learning is called winner-take-all learning.

Splitting decision

The splitting decision defines what feature should be used for splitting, at what value and at what level of the tree. The splitting decision is based on the concept of **impurity**. The impurity of a dataset measures the uncertainty of the classes in the dataset. Pure datasets, namely datasets with minimum impurity, contain data points with the same labels (i.e., all points have the same label). On the other hand, impure datasets, namely datasets with maximum impurity, have half samples of a class and half samples of the other.

The tree should minimise the impurity in the dataset at each step. Since we need to minimise the impurity, we need a way to compute it. Impurity is usually measured using:

- The **entropy**. Entropy is related to the concept of **information gain**, which is the weighted decrease in entropy. This means that the information gain is the difference between the entropy of two splits weighted on the number of elements that the split will influence in the decision tree. For this reason, the information gain is a measure used for selecting the split in the decision tree.
- The **Gini index**.

Both measures have a value of 0 when we have minimum impurity (i.e., when all inputs are positive samples or all are negative samples) and maximum value when we have maximum impurity (i.e., half of the inputs are positive samples, half are negative).

The information gain can be used to answer the splitting decision. In particular, we can compute different gains for different splits and keep the one with the highest gain (i.e., the one that mostly decreases the impurity in the data). Note that this process can be parallelised since each split is independent of the other, hence we can speed up the splitting process. Practically, to answer the splitting decision we

1. take all the features,
2. try different split candidates,
3. computes the information gain, and
4. select the split for which the information gain is higher.

This process can be applied if the number of features is small, otherwise, we have to introduce some heuristics. This process is repeated, starting from the root node. Note that this process can be parallelised.

Stopping decision

The stopping decision defines when the learning algorithm should stop adding nodes to the tree (i.e., when the algorithm should stop). Without any stopping mechanism, the algorithm would stop when the tree has as many leaves as the number of data samples (i.e., one class for each input in the dataset). This is however not desired since such a model would be highly overfitting. We can use **early stopping** to decide when the algorithm should stop. Early stopping evaluates the performance of a model (i.e., of the tree) during training and the training stops when the model starts performing badly. More precisely, the dataset is split into

- a training set, used for training and,
- a validation set, used for evaluating the model.

At each learning step, namely after having split the nodes at a certain level using the training set, the model is evaluated by means of the misclassification error computed on the validation set. When the performance of the model, evaluating using the validation set, starts decreasing (i.e., the validation error starts increasing), then we should stop slitting the nodes of the tree. In other words, we want to stop when the evaluation error is at its minimum. Note that, if we compute the evaluation error using the training set, we would see it decrease since we are using the same data used for learning.

Rule extraction

A decision tree can be seen as a rule set. In particular, the path from the root to a leaf is a condition made of the conjunction of the conditions on the inner nodes. For instance, if we consider the decision tree in Figure 8.17 we obtain the rules:

- If Transaction amount $>$ 100000\$ and unemployed = No Then no fraud.
- If Transaction amount $>$ 100000\$ and unemployed = Yes Then fraud.
- If Transaction amount $\leq$ 100000\$ and Previous fraud = Yes Then fraud.
- If Transaction amount $\leq$ 100000\$ and Previous fraud = No Then no fraud.

This property of decision trees is very important since it makes decision trees easy to interpret. In fact, we can say why a transaction has been marked fraudulent by simply following the path from the root to the leaf and reading the conditions in the inner nodes. This means that decision trees are a white-box model since we can understand why a class has been assigned to a sample.

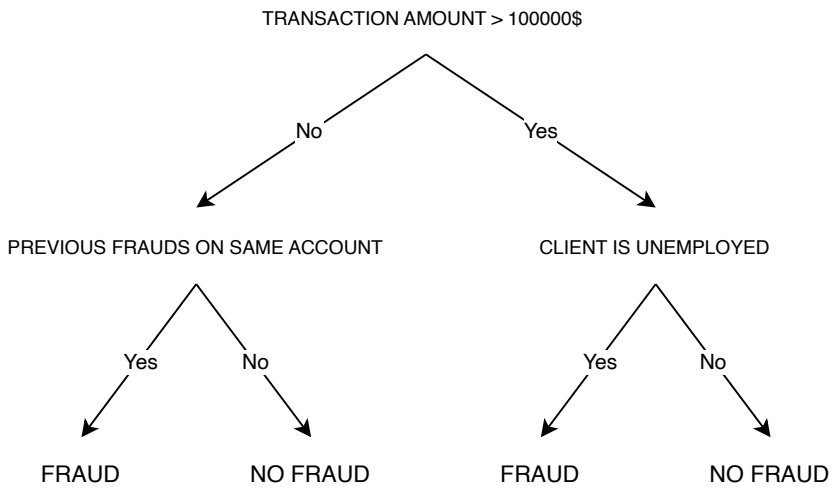


Figure 8.17: A decision tree.

Rule extraction can be used to interpret the results obtained through clustering, too. In fact, clustering divides points into clusters, which is like assigning a label to each point (if we consider a cluster as a class). In practice, we can:

1. Apply clustering to a dataset.

2. Use the results of clustering to train a decision tree.

3. Use the decision tree to interpret the results of clustering.

It's important to understand that, through this technique, we are not understanding why clustering spitted out such results but we are just trying to give an interpretation to such results. Namely, we are trying to understand if the decision tree can explain the results. In any case, we are not understanding the inner workings of the clustering algorithm we used in the first place.

Graphical representation

A decision tree finds multiple decision boundaries, orthogonal to the axes, which split the space of the dataset. For instance, the decision tree in Figure 8.18 is translated into the boundaries in Figure 8.19.

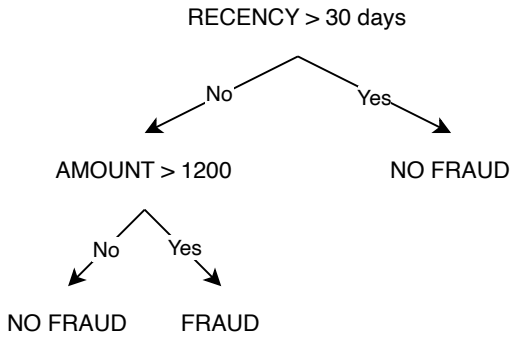


Figure 8.18: A decision tree.

Decision trees for variable selection

Decision trees minimise, at each level, the impurity of the dataset. This means that the features close to the root are more significant and useful to discriminate the class. As a result, we can use decision trees for variable selection, too. In fact, we can train a decision tree on a dataset and then keep the variables which are closer to the root.

Disadvantages

Given their properties, decision trees look like the definitive solution to fraud detection, but that is not true. Decision trees have one big problem, in fact, they are very sensitive to noise and always overfit, even if we apply early stopping. In other words, decision trees strongly depend on the training data and on the knowledge in the training data.

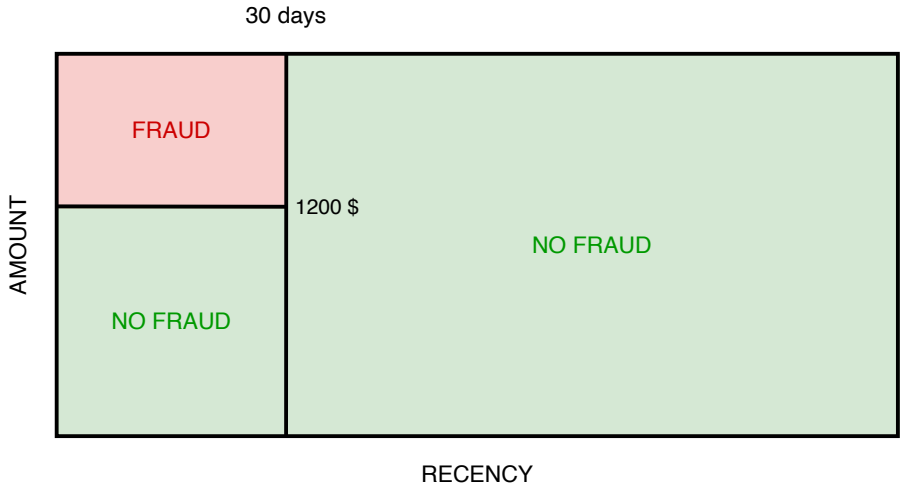


Figure 8.19: The graphical representation of the tree in Figure 8.18.

8.3.4 Neural networks

Interpretation of neural networks

Neural networks are very powerful models however they are black boxes. This means that we know how they are built but we can't clearly understand the reasons why a certain output value has been computed. Luckily there exist techniques to interpret the results of a neural network. The most important are:

- **Variable selection**.
- **Rule extraction**.
- **Two-stage models**.

Variable selection Variable selection allows us to select the features that mostly contribute to the output of the neural network since they might help us in understanding why a certain output has been computed. The most important techniques used for variable selection are:

- **Hinton diagrams**. Hinton diagrams allow us to visualise the weights of the connections from the input to the hidden layer. In particular, the weights are represented on a two-dimensional matrix with the input nodes on one dimension and the hidden neurons on the other. A cell in position (i, j) contains a representation of the weight from the input node i to the hidden neuron j. A weight is represented as a square. The size of the square is proportional to the weight and its colour represents the sign of the weight. Given a Hinton diagram (assuming the input nodes are on the columns) we can remove the columns with smaller squares. Practically, we

  1. Inspect the Hinton diagram and remove the variable whose weights are closest to zero.

2. Re-estimate the neural network with the variable removed. To speed up the convergence, it could be beneficial to start from the previous weights.
3. Continue with step 1 until a stopping criterion is met. The stopping criterion could be a decrease in predictive performance or a fixed number of steps.

- **Backward variable selection**. Backward variable selection removes one feature at a time to understand which feature mostly impacts the output. Practically, if we remove a feature and the output drastically changes, then the removed feature is relevant. More specifically, backward variable selection works as follows:

1. Build a neural network with all N variables.
2. Remove each variable in turn and estimate the network. This will give N networks each having $N-1$ variables.
3. Remove the variable whose absence gives the best-performing network (e.g., in terms of misclassification error, mean squared error).
4. Repeat this procedure until the performance decreases significantly.

Note that variable selection tells us what features are relevant for the network but doesn't explain how the neural network works internally. Figure 8.20 shows an example of a Hinton diagram.

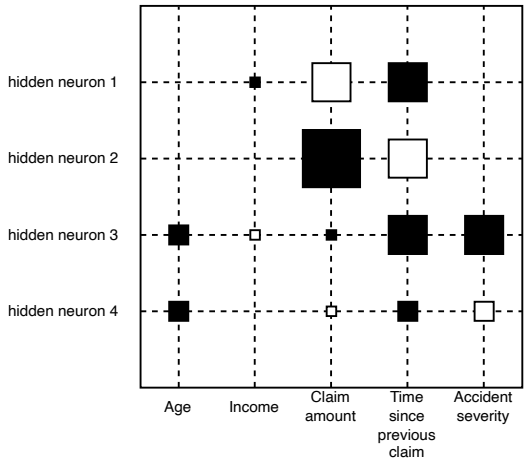


Figure 8.20: An example of a Hinton diagram. The rows of the matrix represent hidden neurons and the columns represent input features.

Rule extraction Rule extraction extracts if-then classification rules (like those obtained from decision trees) from a black-box model. Rule extraction techniques can be divided into:

- The **decompositional technique**. The decompositional technique decomposes the internal mechanisms of a neural network by inspecting its weights. The decompositional technique works as follows:

1. Start from the original data.
2. Build a neural network.

3. Categorise the activations of the hidden neurons. This means taking the values of the hidden neurons and discretise them. One way to discretise variables is by taking the interval of values (i.e., $[\min\{activations\}, \max\{activations\}]$), divide it into k subintervals of the same length and assign a discrete value to each interval (e.g., $(0, 2]$ is mapped to value 1, interval $(2, 4]$ is mapped to value 2 and so on). We can then assign to each neuron the discrete activation value related to the continuous activation value (e.g., we assign the activation value of 1 to a neuron with an activation of 1.2).
4. Extract the rules relating the network outputs to the categorised hidden units. For instance if we see that the output is always 1 when neuron 1 has a value of 2 and neuron 2 has a value of 5, then we can build the rule

```
if hidden_1 = 1 and hidden_2 = 5 then output = 1
```

5. Extract the rules relating the categorised units to the inputs.
6. Merge the rule sets obtained at the two previous steps.

- The **pedagogical technique**. The pedagogical technique uses a black box to train a white box (e.g., a decision tree), which can be then used to interpret the results (but not the mechanisms used by the black box to obtain them). The pedagogical technique works as follows:

  1. Start from the original data.
  2. Build a neural network.
  3. Get the network predictions and add them to the data set.
  4. Extract the rules relating the network predictions to the original inputs in a decision tree fashion. Generate additional data when necessary.

The main difference between these techniques is that the former builds if-then rules more similar to expert-based systems whereas the latter builds something similar to a decision tree. The rules extracted by the aforementioned techniques should be evaluated in terms of:

- **Accuracy**.
- **Conciseness**.
- **Fidelity**. Fidelity measures to what extent the extracted rule set succeeds in mimicking the neural network and is calculated as

$$\frac{TP + TN}{FN + FP}$$

  where TP is the number of legitimate transactions classified as legitimate by the neural network and the rules, TN is the number of fraudulent transactions classified as fraudulent by the neural network and the rules, FN is the number of transactions classified as fraudulent by the neural network but legitimate by the rule set and FP is the number of legitimate transaction classified as legitimate by the neural network but fraudulent by the rule set.

- **Banchmarking**. We can benchmark the extracted rules or the decision trees with a tree built directly on the original data.

Two-stage models Two-stage models use an interpretable model (i.e., a white box) to interpret the results of a black-box model. More precisely, to build a two-stage model we have to:

1. Estimate (i.e., train) a white-box model (e.g., logistic regression). This model provides interpretation.
2. Train a neural network (or in general a black-box model) to estimate the error done by the white-box model.
3. Compute the output of the whole model as the sum of the prediction of the white- and black-box models.

8.3.5 Support vector machines

Support Vector Machines (SVMs) mitigate the shortcomings of neural networks. In particular SVMs deal with:

- **not convex optimisation functions** and,
- the effort of **tuning the models hyper-parameters** (e.g., the number of hidden neurons, the activation function, the number of epochs).

A support vector machine can linearly split data, even if the data is not linearly separable. This seems counter-intuitive, however, it's possible thanks to kernels which map the input space to a linearly separable space where we can use an SVM. The idea behind SVMs is to use linear programming to find the classification hyperplane. Among all hyperplanes correctly splitting the dataset, we want to choose the one that maximises the distance between the hyperplane and the closest point, called the support vector, for each class, to the hyperplane. In practice, for a binary classification problem, given a candidate hyperplane we:

- Take the closest point of class C_1 to the hyperplane. This is the support vector for class C_1.
- Take the closest point of class C_2 to the hyperplane. This is the support vector for class C_2.
- Compute the distance between the support vectors and the hyperplane.
- Compare the distance obtained with the ones obtained for other hyperplanes.

Finding such a hyperplane is a convex optimisation problem with no local minima and only one global minimum. Figure 8.21 shows an example of how support vector machines work.

Support vector machines also work for datasets that contain outliers and aren't separable even when using kernels. Outliers are handled by adding a penalty to misclassified points (i.e., outliers). Figure 8.22 shows an example of SVM dealing with outliers.

Representation

Support vector machines are black boxes but we can use the same techniques used for neural networks to understand and interpret their output. An SVM can also be seen as a neural network where the hidden layer uses the kernel activation functions. The number of hidden neurons corresponds to the number of support vectors and doens't have to be tuned since it's obtained from the optimisation process (i.e., from learning). The output layer is a neuron with a linear activation function.

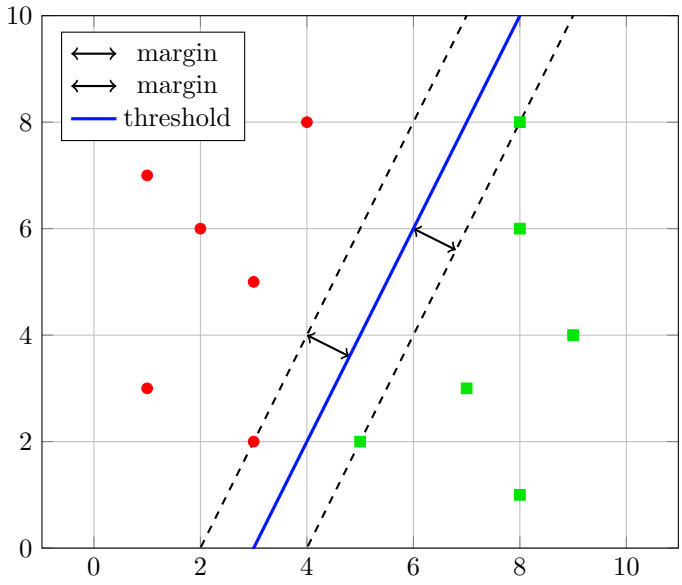


Figure 8.21: A support vector machine.

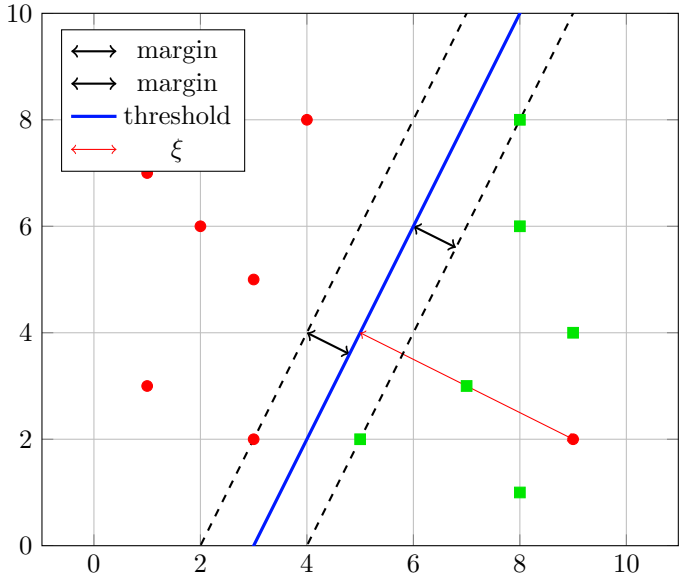


Figure 8.22: A support vector machine dealing with outliers.

8.3.6 Ensemble methods

Ensemble methods combine poor models to obtain better models. In fact, we can use different models to cover different parts and complement the behaviour of other models. Independently from the specific ensemble technique, the models used should be sensitive to changes in the data hence overfitting or underfitting, otherwise, we get a low-performance ensemble model. The most important ensemble techniques are:

- **Bagging**. Bagging can reduce the variance of overfitting models (i.e., with high variance) without increasing the bias.
- **Boosting**. Boosting can reduce the bias of underfitting models (i.e., with high bias) without increasing the variance.
- **Random forests**. Random forests can reduce the variance of decision trees without increasing their variance.

Ensemble methods are black-box models.

Bagging

Bagging performs bootstrap aggregation and works as follows:

1. We divide the dataset into B bootstraps, which are samples with replacements.
2. We build and train a model for each bootstrap.

During inference (i.e., when we want to use the ensemble model) we compute one output for each classifier and we use majority voting or the average over all the outputs to compute the output of the ensemble.

The key in bagging is that each of the B models is unstable (i.e., sensitive to noise, overfitting).

Boosting

Boosting estimates multiple models by weighting samples. Practically,

1. We initialise a weight for each model at random.
2. We re-weight the models according to the classification error. Ideally, difficult observations should get more attention.
3. Repeat for a fixed number of iterations (which can be found using a validation set).

The output of the ensemble is the weighted output of all the individual models.

The main advantage of boosting is that it's easy to implement however boosting has a high risk of overfitting, especially if the labels are noisy.

Random forests

Random forests are sets of decision trees initialised at random. In particular, for each tree:

1. We build a bootstrap.
2. We build a decision tree generated at random (i.e., random splits).

The idea behind random forests is to build models with high variance (hence overfitting models). In particular, random forests are based on:

- The dissimilarity amongst the base classifiers, which is obtained by adopting a bootstrapping procedure to select the training samples of the individual base classifiers
- The selection of a random subset of attributes at each node.
- The strength of the individual base models.

Evaluation of ensemble models

Various benchmarking studies have shown that random forests can achieve excellent predictive performance, in fact:

- They rank amongst the best-performing models across a wide variety of prediction tasks.
- They can deal with data sets having only a few observations but with lots of variables.
- They are highly recommended when high-performing analytical methods are needed for fraud detection.

8.4 Evaluation of fraud detection models

The evaluation of a fraud detection model starts before building the model itself. Before deploying the model we have to decide:

- How to **split the data**.
- What **performance metrics** we want to use.

Both decisions are taken before training the model but impact the evaluation phase.

8.4.1 Splitting the data

Splitting the dataset is key to evaluating a model on the correct data. Depending on the volume of data we have we can split the dataset in different ways.

Large dataset

If we have a large dataset, we can split it into three sets:

- A **training set**, usually 70% of the whole dataset. The training set is used only to train the model and should never be used for evaluating the model.
- A **validation set**, usually 10% of the whole dataset. The validation set should be used to tune the hyper-parameters of the model and to evaluate different models. The data in the validation set should not be used for training.
- A **test set**, usually 20% of the whole dataset. The test set should be used, after having trained the final model, to assess the performance of the model.

Small dataset

If we have a small dataset, we can't split the dataset into training, validation and test sets since we don't have enough samples. We can instead use:

- **Cross-validation**. Cross-validation doesn't differentiate between training and validation sets. Cross-validation works as follows:

  1. Data is randomly split into K folds, namely into K subsets.
  2. $K-1$ folds are used for training, the other for validation.
  3. Repeat from step 2 until every fold has been used for validation.
  4. The performance is evaluated averaging the performance of each iteration.

  When using machine learning models for fraud detection we usually want to use stratified cross-validation. When using stratified cross-validation, we have to make sure the fraud/no-fraud odds are the same in each fold.

- **Leave-one-out cross-validation**. Leave-one-out cross-validation works as normal cross-validation but each fold contains one data point only. This means that the model is validated using only one data point and trained on the remaining data points.

8.4.2 Performance metrics

The performance of a model is usually assessed using a confusion matrix. Using the confusion matrix we can compute many other performance metrics which are based on:

- True Positives TP. True positives (frauds) are the samples that belong to the positive class which are assigned to the positive class.
- True Negatives TN. True negatives (non-frauds) are the samples that belong to the negative class which are assigned to the negative class.
- False Positives FP. False positives are the samples that belong to the negative class which are assigned to the positive class.
- False Negatives FN. False negatives are the samples that belong to the positive class which are assigned to the negative class.

Starting from these metrics we can compute:

- The **classification accuracy** is the percentage of correctly classified observations.

$$Acc = \frac{TP + TN}{TP + FP + FN + TN}$$

- The **classification error** is the mis-classification rate.

$$Err = \frac{FP + FN}{TP + FP + FN + TN}$$

- The **sensitivity**, **recall** or **hit rate** measures how many of the positive samples are correctly labelled as positive samples.

$$Rec = \frac{TP}{TP + FN}$$

- The **specificity** looks at how many of the negative samples are correctly labelled by the model as negative samples.
$$Spe = \frac{TN}{FP + TN}$$
- The **precision** indicates how many of the predicted positive samples are actually positive.
$$Pre = \frac{TP}{TP + FP}$$

To decide which metric is best, we should understand the distribution of positive and negative samples in datasets for fraud detection. Datasets for fraud detection contain a small percentage of positive samples (since fraudulent transactions are way less than legitimate ones), hence even a model that flags every transaction as legitimate has a high accuracy. To understand which metric works well for our datasets we can evaluate the performance of a model at the cut-off, namely where every point is classified in the same class. If we obtain a good performance (or a completely bad one), the metric should be discarded.

The performance metrics we have seen can be used to build more complex metrics like the Receiver Operating Characteristic (ROC) Curve.

Receiver operating characteristic curve

The receiver operating characteristic curve plots the sensitivity over 1 minus the specificity (or over the false positive rate). A point in the $(1-specificity, sensitivity)$ space is an instance of a confusion matrix. A model with sensitivity and specificity 1 (i.e., in the top left corner) is perfect.

Figure 8.23 shows different ROC curves.

Given the ROC curve, we can use the Area Under the ROC Curve (AUC) to evaluate multiple curves that intersect. In general, the AUC provides a simple figure-of-merit for the performance. In particular, the higher the AUC, the better the performance. Note that the AUC is bounded between 0 and 1, hence can be interpreted as a probability. More precisely, it represents the probability that a randomly chosen fraudster gets a higher score than a randomly chosen non-fraudster.

A good classifier should have a ROC above the diagonal and AUC bigger than 50%.

Non-statistical performance metrics

The performance of a model should not be evaluated solely on statistical metrics. We should consider also:

- **Interpretability**. A very powerful model whose inner workings are unknown might be less useful than a less powerful model for which we understand the reasoning put in place to compute the output. In other words, black box models (which are usually more powerful but less interpretable) could be less useful than white box models (which are often less powerful but more interpretable). This is because a human usually has to validate the output of a fraud detection system. If the validator knows why a transaction has been marked as fraudulent, he or she can easily tell if the transaction is actually a fraud.
- **Justifiability**. Justifiability verifies to what extent the relationships modelled are in line with expectations by verifying the univariate impact of a variable on the model's output.

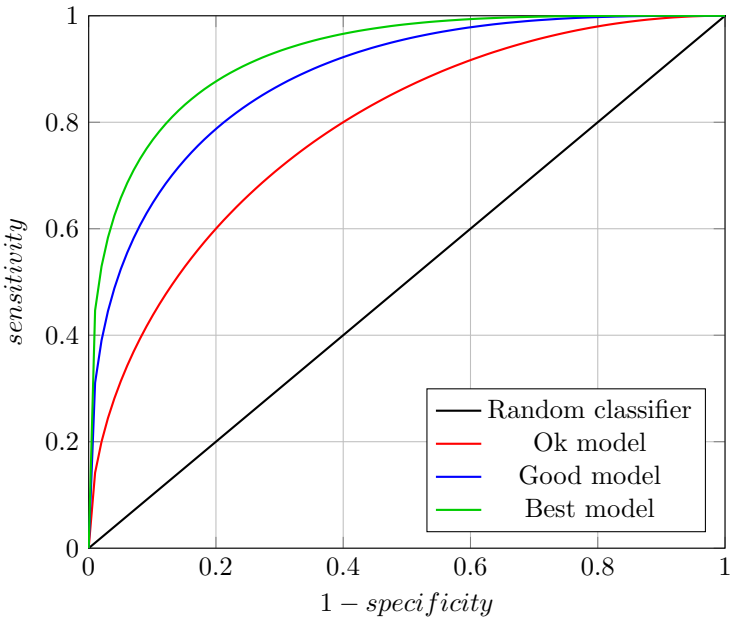


Figure 8.23: Different ROC curves.

- **Operational efficiency**. Fraud detection models are often used to validate transactions while they are executed, hence they have to act swiftly. For this reason, the model should be able to compute an output in a short time. Having a very powerful model that requires a lot of time to classify a transaction is useless in real-time operations and would make the bank lose clients (which prefer banks that allow the user to pay instantly).

8.4.3 Skewed datasets

The datasets used for fraud detection are usually unbalanced. This means that we have only a few positive samples (i.e., few fraudulent transactions), usually less than 1%. This can create problems since a model might not be powerful enough to learn what a fraudulent transaction is because of the high volume of legitimate transactions. Namely, the frauds are handled more like noise and all transactions are marked as legitimate because the model sees almost only legitimate transactions. To solve this problem we can try and rebalance the dataset. Note that this operation should be performed only on the training dataset. We have two ways of changing the distribution of positive samples:

- **Increasing the time horizon for prediction**. Instead of predicting fraud with a six-month forward-looking time horizon, we can use a 12-month time horizon.
- **Sampling every frauds twice** (or more). We predict fraud with a one-year forward-looking time horizon using information from a one-year backwards-looking time horizon. This can be done by shifting the observation point earlier or later, hence the same fraudulent observation can be sampled twice. The variables collected will be similar but not perfectly the same since they are measured in different time frames.

Finding the optimal number of positive samples to add is subject to a trial-and-error exercise and depends on the skewness of the target.

In general, we have three classes of techniques to change the positive distribution in the dataset:

- **Undersampling**. Undersampling techniques remove legitimate transactions two or more times so as to make the distribution less skew. Undersampling is based on business experience whereby obviously legitimate observations (e.g., low-value transactions or transactions done by inactive accounts) are removed.
- **Oversampling**. Oversampling techniques replicate frauds two or more times so as to make the distribution less skew.
- **Under- and oversampling**. Undersampling and oversampling can be mixed to get the best of both worlds. Undersampling usually results in better classifiers than oversampling.

Even if rebalancing can help in training, we shouldn't modify the original distribution that much since it would lead to biased models. To determine the best class distribution we can:

1. Build an analytical model on the original data set with the skew class distribution (e.g., 95%/5%).
2. Record the AUC of this model (possibly on an independent validation data set).
3. Use oversampling or undersampling to change the class distribution by 5% (for instance, 90% / 10%).
4. Record the AUC of the model.
5. Repeat from step 3 further reducing the distribution.
6. Once the AUC starts to stagnate (or drop), the procedure stops and the optimal odds ratio has been found.

This procedure is similar to early stopping.

SMOTE

The techniques we have studied re-use the samples already present in the dataset to decrease the skewness. However, there exist techniques that create new observations based on those already present in the dataset. One example is Synthetic Minority Oversampling Technique (SMOTE). SMOTE:

1. For each observation of a minority class, calculates the k nearest neighbours. Depending on the amount of oversampling needed, one or more of the k-nearest neighbours are selected to create the synthetic examples.
2. Randomly creates two synthetic examples along the line connecting the observation under investigation with the two random nearest neighbours.
3. Combines the synthetic oversampling of the minority class with undersampling of the majority class.

SMOTE usually works better than either undersampling or oversampling. Figure 8.24 shows an example of two samples created with the SMOTE technique.

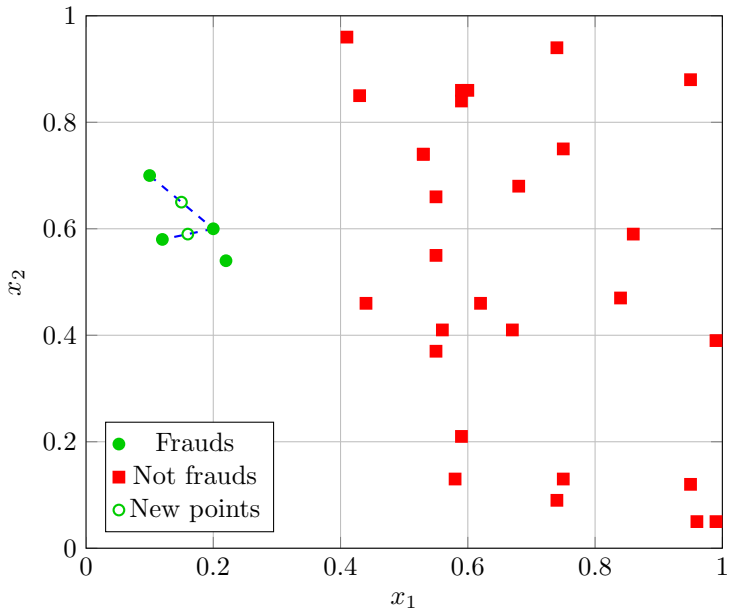


Figure 8.24: An example of application of the SMOTE technique.

Cost-sensitive training

The skewness can be reduced also during training. An example is cost-sensitive training that changes the evaluation of the minority class by assigning a higher misclassification cost to the minority class.