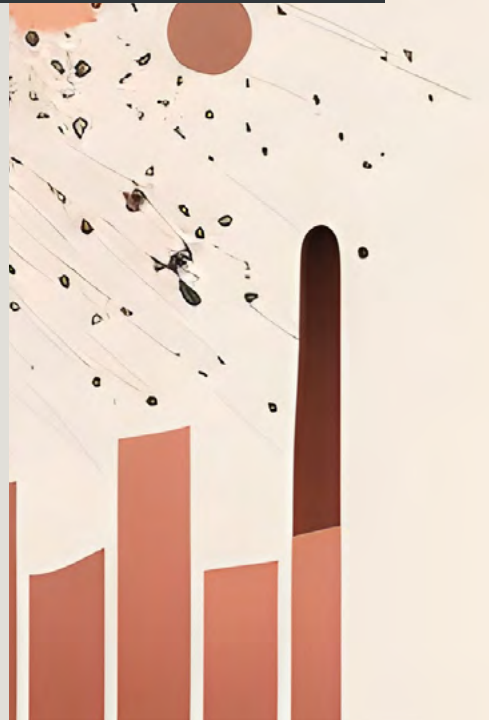


# DATA SCIENCE 2023



**The big data revolution is now at least a decade old, and most companies and public institutions have fully developed data warehouses or lakes where organisational data is collected. But what is it being used for? What value does it create? We have gotten better and more detailed reporting on the daily operations, more management reporting and more dashboards. All of this is good and valuable, but we can do so much more. This is what data science is all about.**

Data science promises to bring operational solutions and even deeper insights into organisations. As such, the promise of data science is closer to the original promise of mechanical automation. Data science produces solutions. Solutions that go into productions and directly affect an operational part of the organisation, supporting, improving or maybe automating a work process. Sometimes even enabling new work processes. This is not the only feature of data science, but it is probably the core promise and why it is being compared to the fourth industrial revolution (Forum, 2016).

An example could be a doctor. A patient comes through the door, and a data science solution not only presents a dashboard based on the patient's journal but also a set of predicted points of attention for the doctor based on the patient's history, the latest research and the current disease landscape in the world. The patient gets a scan of his/her lungs causing trouble, and as the scanner delivers the image, it instantly suggests that a region of the image be examined for potential cancerous growth and immediately highlights the area. The machine is trained on millions of images worldwide, and the doctor is more alert. After the patient has left, the doctor dictates a note which is automatically saved as text. An algorithm notes that a scan of the lungs is mentioned and automatically adds a procedure code to the patient's history. It is also noted that cancer is suspected, and a flag of attention is raised on the patient for future reference.



Data science is an interdisciplinary field collecting classical statistics, machine learning and scientific methods to understand and analyse real world phenomena via structured and unstructured data.

(Wikipedia: Data Science, 2022)

In the example above, data science was first used to give the users an overview based on an otherwise incomprehensibly large information base. Subsequently, decision support was provided as the area was highlighted. Finally, there was a complete automation of the logging and archiving of the work. In this case, the doctor was able to work more precisely, spend more time with the patient and see more patients in a day thanks to intelligent use of data.

We are hardly there yet. But I am in no doubt that we are on the way. This guide will give you an idea of how data science works in organisations today, and how you can get started in your own organisation.

## Who is this guide for?

This guide is first and foremost for people who want to increase their organisation's data science skills. Either by becoming better data scientists or software developers themselves or by leading a data science department, small or large. This guide provides deep insight into a complete data science workflow – not just from data to model but from identifying the right problem to setting up the model to realising value and maintenance.

As such, it is also relevant for managers working with data and digitalisation. Data science is a rapidly developing field, and it can be a full-time job just to keep up with what is happening. We will provide an overview of what can be expected from modern data science solutions, and what it takes to achieve success with them.

## Why this guide?

At Implement, we experience a great demand for data science competencies. This is driven by the rapid growth of data in most modern organisations which naturally raises the question: how can we leverage all this data?

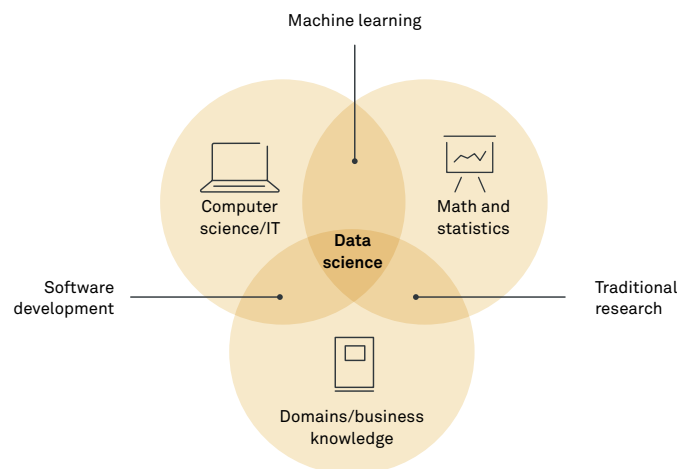
The next thing that usually happens is that either it is decided that a business problem needs to be solved with data science or perhaps an enterprising employee proposes a concrete data science solution. In some cases, a team is formed, and solution identification begins.

In any case, the core challenge is the need to have three competencies present in the process: (1) software development and IT, (2) statistics and machine learning and (3) the business or domain within which they work.

If one of these elements are missing, we see some common challenges arise:

- Lacking the **software developer skills**, the data science project risks becoming a fantastic solution but which cannot be implemented in the business and is somewhere between difficult and impossible to maintain. The solution may emerge, and it may even create some value, but it will have a short life if it ever gets into operations.
- Lacking the **statistics and machine learning skills**, the data science project will often find it difficult to provide the adequate quality of solutions that is in demand. The project will be prevented from tackling the new and difficult problems that they potentially could if they had a greater understanding of machine learning.
- Lacking **business knowledge**, the project risks developing great solutions that are not applicable in business and where the benefits of development are never realised.

Unsurprisingly, finishing projects, achieving the expected quality and realising the desired impact are some of the most widespread problems in new data science projects.



## Content of this guide

In this guide, you will be taken on a journey from the earliest start of a data science project with use case identification and all the way to advice on governance of models. The guide consists of eight chapters with the following headings:

- **Chapter 1: Use case identification:** How do you ensure that you are working on a valuable idea?
- **Chapter 2: Data exploration:** Get an overview of your data from patterns and content to creation and management.
- **Chapter 3: Machine learning:** What are the techniques we use to solve the problems? How do you hit the right amount of complexity and ensure that your model works in real life.
- **Chapter 4: Definition of purpose:** What should your model optimise for? What exactly is the problem we are trying to solve? And how do you make sure that your model actually does what you expect.

- **Chapter 5: Modern deep learning and fine tuning:** The toolbox of data science is rapidly growing and reuse of models across domains is changing things up greatly.
- **Chapter 6: MLOps and testing:** We look at how to deploy our models and properly update and maintain them.
- **Chapter 7: Ethics:** Data science touches on many human aspects from personal data to automation. Chapter 7 gives an introduction to the main considerations and some concrete advice on how to proceed in a human-centric way.
- **Chapter 8: Sustainable data science:** Investigates the environmental impact of our modelling work, and what we can do to minimise our footprint.

The guide can be read chapter by chapter, but it is also written such that each individual chapter can be read in isolation. If you just need a brush-up on a particular concept, feel free to skip to the relevant chapter.



# Chapter 1: Use case identification

How do we ensure that our data science solutions make an impact? This is often easier said than done, and data science has some specific pitfalls to keep in mind. Part of the challenge is the limited overlap between people with knowledge of artificial intelligence and business processes. This challenge means that many data science projects start by trying to solve the wrong problem which either does not solve anything valuable for the business or proves impossible or too expensive to solve compared to the value it presents.

## What is a (data science) use case?

A use case is two things: a goal and a path to it. Alternatively: **a problem and a solution.**

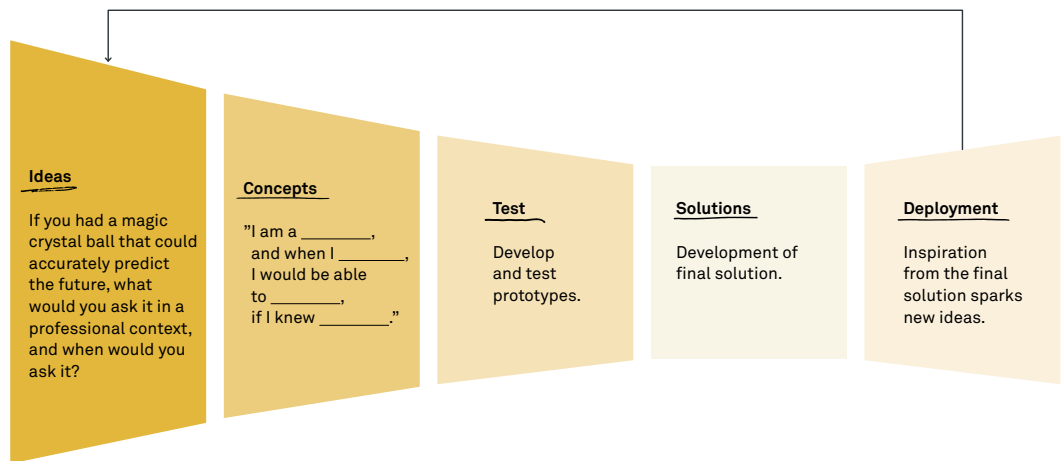
An example could be translating texts from French to English, and the solution could be a machine translation system such as Google Translate.

Use case identification is about first identifying interesting and valuable goals and problems and then identifying well-designed solutions for them.

## A place to start, but not to stop

If someone out there reads this and feels that we overlook a lot of possibilities (for example, unsupervised learning or reinforcement learning), the answer is yes. The tools presented here are not an exhaustive list for finding any data science use case in an organisation, but it is a way to get started.

## Use case identification



Use case identification often looks like a classic innovation process. A wealth of ideas must be generated, sorted, prioritised and enriched until some of them can be tested and, finally, a subset can be sent out into the world for feedback and inspiration.

## Ideas: Getting started with a data science brainstorm

The first rule of data science use case identification is: **never do it alone.** Get out there and meet the potential users. Run workshops or invite people in.

Start by explaining what data science and machine learning is followed by a question that can help to inspire, such as:

““ If you had a magic crystal ball that could accurately predict the future, what would you ask it in a professional context, and when would you ask it?

After the initial jokes about lottery numbers or the private lives of their friends, it often turns out that this question is harder to answer than you might expect. It requires people to think quite specifically about the question they want to ask the magic crystal ball. The great thing about the exercise is that it puts aside discussions about what is possible and what is not, and it allows people to be creative. Often, many interesting questions and discussions come up, many of which are obviously impossible, and this is part of the point. It is much better to discard an impossible idea early than to develop something realistic but worthless.

### Concepts: Creative enrichment

At this point, we would like to have 20-40 ideas written down, describing situations where knowledge from a magic crystal ball could be useful. This needs to be further clarified, and we need to be critical of our issues. We know from the previous exercise that they are interesting, but are they also *valuable*? The purpose of this step is to validate that the problem we are solving is valuable. That is why we use this phrase:

*I am a \_\_\_\_\_,  
and when I \_\_\_\_\_,  
I would be able to \_\_\_\_\_  
if I knew \_\_\_\_\_.*

The key is to have an **action** in a **situation**. See this example:

*I am an **operator at 112**, and  
when I **make a call**, I would be  
able to **send an ambulance  
faster** if I knew it was a cardiac  
arrest.*

(See (Corti.ai, 2020) for more on this example)

If such a phrase cannot be constructed, there is a risk that what has been identified is a “nice to know” type of problem. A situation where someone would like to know something because it would be nice or interesting but may not be able to use the information for a valuable action. These phrases also make it possible to go to specific people to get the idea verified. Are these people in fact in the situation being described? Is the information proposed enough to make a decision/ take action? Is the action available? If all these questions can be answered with a “yes”, then we have an enriched idea.

**Tip:** If possible, ask the person in the phrase if they know any data that could give a clue as to what we would like to know. People are often very creative and can have insights that we do not have. So take the opportunity to talk to users!

### How about complete automation?

In a complete automation situation, one in which there is no human to consider the information and act, a solution might look different. This type of use case also falls outside the tools here. Not because they are not interesting, but the identification follows a somewhat different pattern and lends itself more easily to a classic cost-focused business case. It is – generally speaking – a discipline that leans more on process management than innovation because once the solution is designed correctly, the value or business case is often very straightforward.

### Critical questions

Ideally, we should now have a handful of good concepts to move forward with, and it is time to look at the data science feasibility behind each concept. These rules of thumb can be useful:

- How long would it take a human to answer the same question?
- Would ten people be somewhat in agreement, or would we expect disagreement?
- Does data exist that can reasonably explain the phenomenon?

### How long would it take a human to answer the same question?

This question is about the expected complexity of the problem. An old rule in data science says: if it takes a human more than two seconds, then

the computer probably cannot solve it. Today, we can do better, and I would be willing to say a few minutes, but the principle is still relevant. If people find it to be a difficult assessment, then the machines are probably at least as challenged. The best use cases are often easy for people. Assessing whether an image is a dog or a cat is easy, and we do it almost without thinking. Conversely, assessing whether the content of an article is factually correct may require us to think more carefully. The value in data science often lies in being able to scale the solution and increase standardisation and consistency by decreasing the human element.

**Would ten people be somewhat in agreement, or would we expect disagreement?** The question attempts to assess the degree of subjectivity in a problem. Note that the world is full of things that we know are subjective, but there are even more things that we think are objective but which in practice are not. We might like to think that hiring and firing in our company follows a strict objective process, but a lot of studies show how subjective these decisions are. These areas are difficult for data science because they (for good or bad) contain a lot of subjectivity.

It is worth noting that **subjectivity is not a dealbreaker for data science**. If a task is subjective, it is cause for awareness for the data scientist, and he/she should show respectful humility as to what can be expected of a solution. Algorithms can also be used to identify and control bias. More on that in the Ethics chapter.

**Does data exist that can reasonably explain the phenomenon?** Some phenomena do not lend themselves to prediction because they are

fundamentally unpredictable. Earthquakes and volcanic eruptions have long been the goal for science not only to understand but also to predict, but so far without major success. Major political events such as elections and civil wars are also notoriously difficult to predict. The stock market is also a classic challenge. Common to all is that we could probably compile some data, but we would hardly be able to compile enough data to obtain a particularly strong model, either because data is particularly hard to come by and/or the phenomenon is known to be incredibly complex. The best use cases are those where the source of knowledge is very clear. If we need to diagnose if a leg is broken, we have an X-ray. We know that the answer to our question (“is the leg broken?”) can be found in the image. We might just not know exactly how. This is where machine learning is strongest.

#### **Are we solving the right problem?**

It goes without saying that by using data science solutions, we can solve a plethora of problems. But all solutions might not always fit into the business landscape, and good data science is enabled by being close to business decisions. Consider working in a company that has an extensive master data management system paired with solid data warehouse holding transactional data. If the business wants to “increase revenue of current customers and reduce the cost of onboarding new customers”, as data scientists, we need to be close to the business when deciding to employ either a customer churn or a customer lifetime value analysis. A churn analysis is probably not what we are looking for in this case, but on the other hand, as data scientists, we might highlight the pairing of customer lifetime value and churn analysis in order to achieve a deeper understanding of the customer base.

## **Solution development, testing and deployment**

Hopefully, you have at least one idea left after all of this, and congratulations, you probably have a great use case! You are reasonably sure that there is a group of users who, given your solution, could take valuable action with it, and you are reasonably sure that it can be realistically developed using a data science approach. It could now be extended into a formal business case with estimates of effort needed to develop the solution, expected business impact and prerequisites and requirements for development.

More on all of that in the coming chapters.

## **Summary**

If you need to identify valuable data science use cases, it is helpful to first initiate a creative process by putting aside the limitations of the technology and focusing on identifying situations where better decisions can be made based on information. It can be done, for example, with a phrase like: “I am a \_\_\_, and when I \_\_\_, I would be able to \_\_\_ if I knew \_\_\_.”

Typically, such phrases can be addressed by a data scientist who, based on a critical assessment, will be able to prioritise them and begin collecting data.

# Chapter 2: Data exploration

Data exploration is probably one of the most underappreciated data science disciplines. Specifically, it is underestimated how much time can be saved by getting a strong feel for your data before you start modelling. Are certain variables distributed highly unevenly? Is much data missing? Are there obvious and strong correlations present in the data? Are some variables completely or near-completely stationary?

You can save a lot of time by knowing your data well and designing more clever models and data transformations earlier while avoiding serious mistakes or bias later in the work. It is hard to make an exact recipe for data exploration. We have instead compiled five important pieces of advice that we try to follow as well as some of the most important tools.

## Understand the source of the data

All data comes from somewhere. Whether it is by manual entry, a sensor in a machine or digital system monitoring behaviour.

It is crucial to understand the quality and context of the source. If possible, try to set up a meeting with the people who either enter data into or work with the systems that create the data. What is their impression of the work associated with the data they create? Do they have confidence in the data? Are there exceptions and dark numbers (meaning non-recorded cases) in their work?

If possible, follow the data from the source to the point where you expect to access it.

Are there business decisions along the way that you need to be aware of?

For example, you can ask the source or source owner the following questions:

- Would you trust the data you provide yourself?
- Are there things that are not captured and expressed in data?
- Is there any data that will not be delivered?

## Ask local data experts

Even if you understand the data sources, it can still be important to ask experts in the respective data areas.

If you use data from multiple sources or make transformations of your data yourself, make sure that the transformations you make are known and approved by people in the field. Do you retrieve labels from a table and link it to data in another system? Does that link make sense?

You should also take the opportunity to enquire about data availability. If the problem you are trying to solve is time-sensitive, is the data then updated frequently enough to make sense for your solution?

For example, you can ask the data expert the following questions:

- Can I link these sources with other sources? Is there anything I need to be aware of?
- How long is the history of the different sources? Is it consistent and reliable over time?

- How often is data from the sources updated?

## Data mesh, data warehouses and the like

Individual organisations with a high degree of maturity may have reached a level where their data warehouse, data mesh or analytical environment is so mature that significant data science activity can be done on pre-approved sources. This solution can be pursued with great advantage, but it is still relatively rare. Essentially, it is achieved when a team of employees (the data warehouse style) or a decentralised organisation with strong principles (data mesh style) provide data of very high quality. This data will be so thoroughly documented, cleaned and standardised that the users or consumers can trust it without necessarily understanding the origin of the data any further. This will greatly speed up analytical development, as data discovery, data cleaning and transformation can be done in well-defined analytical environments.

## Keep track of your data structures

If you are to take one thing away from this, let it be this: if possible, **set up an Analytical Base Table (ABT)**.

An ABT is a very safe and easy way to handle your data in a data science project, and it is hugely suitable for data exploration. An ABT is basically a large table that has all your observations as rows, and the columns are the properties that we know about our observations.





## TIDY DATA

Tidy data is a data structure that we always strive to adhere to in all data science tables. Tidy data has one row per observation, one variable per column and one value per cell. Thus, it is a flat two-dimensional table. It is not always that we can keep our data tidy when analyses and needs get complicated, but we should always strive for that. If most of our tables are tidy, we are in a much better place (Grolemund & Wickham, 2020).

You build an ABT in the following way:

1. **Establish a base population.** What are you analysing? Are these support cases? Process steps? Customers? Or customers per day? This question is as difficult as it is essential to answer. You will have an answer when you can answer exactly how many observations your base population has (hint: it must be a counting number).
2. **Left-join features onto your base population.** A left join ensures that observations from your base population (your left table) are never removed when you add new properties from a source (the right table). Instead, missing data is inserted if there is a discrepancy between your base population and your source. It is an important safety feature that ensures that you do not suddenly lose data. At the same time, it is easy to check if duplicates are triggered. You can always count the rows in your base population, and it must not be larger than when you established it.

If you have done it right, you have so-called tidy data (Grolemund & Wickham, 2020). Each observation is in a row, each property is in a column, and each cell is a property for a given observation.

For complex data science solutions, it can become significantly more advanced. Initially, it can help to think about things like sound and images as elements that can be contained in a row, although the way they are held may just be a link to a location where the media is loaded from.

These kinds of media files have their own table-like formats. For example, an image is a matrix of pixels with colour values, and sound is usually a single column table with amplitudes as values.

Two columns for stereo sound and so on. In these cases, you can have a tidy table, but where the contents of a cell can be a table. Conceptually, there is nothing wrong with that, but technically it may require special tools.

### Analytical Base Table (ABT)

An ABT is a table that strictly adheres to the principles of tidy data. An ABT thus has one row per observation and one variable per column. Among the columns are one or more “targets” that are what we want to predict or analyse and one or more “features” that are what we know about a given observation that we should use as a source for our predictions or analyses. It is a simple and very effective data presentation that makes the use of a wide range of data science solutions very intuitive.

Country	Year	Cases	Population
Afghanistan	1999	745	19957071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	218766	128042683

**Variables**

Country	Year	Cases	Population
Afghanistan	1999	745	19957071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	218766	128042683

**Observations**

Country	Year	Cases	Population
Afghanistan	1999	745	19957071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	218766	128042683

**Values**

### Visual inspection and tools

If you have an ABT (or something similar) established, you can do a visual inspection of your data. It is important to look at data. Visualisations are much richer than lists of, for example, averages, standard deviations etc. If you want to see a terrifying example of how descriptive statistics can lie, just look at [Anscombe's Quartet](#) (Anscombe, 1973) or even worse [Autodesk Research – Datasaurus](#) (Matejka & Fitzmaurice, 2020).

Below are three excellent and free tools for data exploration:

- **Google Facets:** A very simple and fast data profiling tool. You upload the ABT (or download the tool and run it locally) and get quick descriptive statistics and simple distributions across all your key features. However, the real value comes when you cross variables. In Google Facets, you have all the data in a simple

interface, making facets the easiest visual inspection tool (Google, 2022). SandDance is a similar tool from Microsoft (Microsoft, 2022).

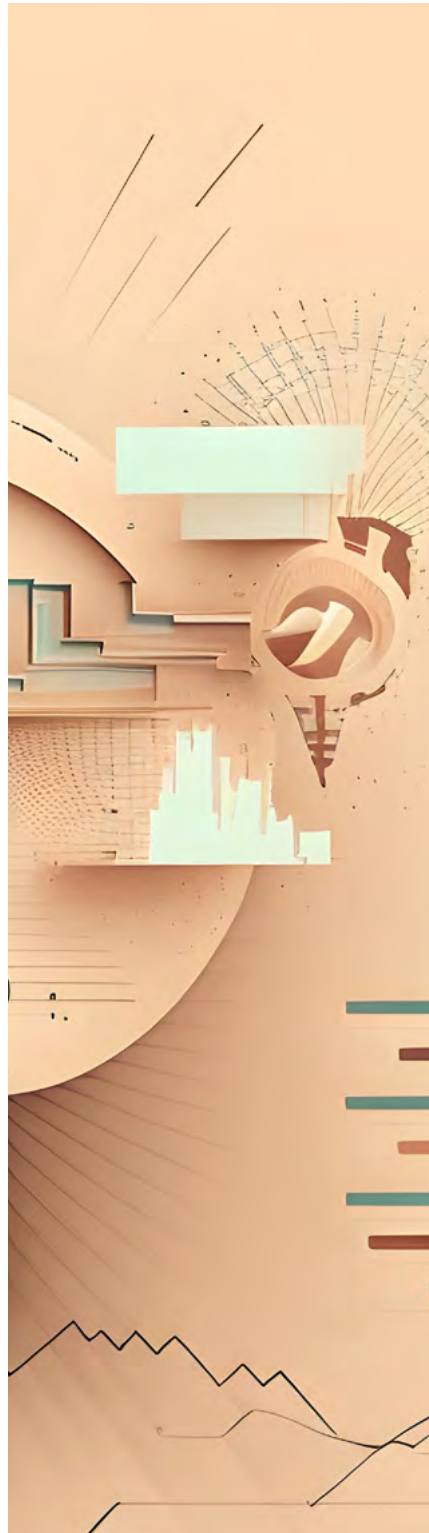
- **Tensorflow Projector:** A substantially more specialised tool designed for complex data with a lot of interdependent features. Tensorflow Projector can “distill” complex datasets into 2D or 3D visualisations, grouping similar data points together

(Tensorflow, 2020). The technique is called T-SNE (Wattenberg, Viegas, & Ian, 2016) and is a powerful visualisation algorithm. We use Projector in two ways:

- » To ensure that there is a good and varied structure of data. If Projector makes an artificial or unnatural-looking form, it is often because the structure of data is not as rich as we might otherwise expect.
- » Projector is uniquely suited to handle extremely complex data such as text, audio or images once it has been processed and prepared for analysis, and it can give a human-friendly overview of the dataset.

- **Microsoft Power BI:** In all fairness, PowerBI is a complete Business Intelligence tool, but the fast calculator and powerful universe of visualisations make Power BI an important data exploration tool. The tool is especially suitable because you can make regular reports in Power BI which means that without additional effort you can have dashboards monitor data quality as you develop. (Microsoft, PowerBI, 2020).

- **Jupyter Notebook:** For a more python-like experience, Jupyter Notebook allows one to mix code, text and imagery in one document. This means that the complexity and specificity of the profile is only limited by what Python can do and fit on a page. Jupyter Notebook is also well suited for automation and can be refreshed through various automation workflows.



## Automation

We highly recommend setting up automation around your data exploration, and especially data profiling, so that it happens often and in a reproducible fashion. Packages such as Pandas Profiling (Profilling, 2020) for Python make it easy, but a good Jupyter Notebook with a series of key visualisations that can easily be updated might also do the trick on smaller projects (Jupyter, 2022). The whole point is to make it easy to do data exploration so that discipline is kept, and you get it done.


The data profile can be very basic, but make sure that you always keep track of how many observations you have, how many – and which – properties, what the minimum and maximum of each attribute is and how many missing data points each attribute has. It may seem trivial to note this information, and something which can easily be requested on demand if needed. Firstly, the key is that automation allows you to keep a history of data profiles to keep track of changes, and secondly, perhaps you are a bit lazy, and if the profile does not exist, you are not going to produce when you should. Therefore, do yourself a favour: set it up and automate it. You will be happy that you did.

Furthermore, automation has the advantage that it can be done through an analysis flow. It is a great help to have consistent data reports for all sources for the overall ABT and for all significant transformation steps towards the final model and final output.

These reports are also an essential aid in handover and maintenance, helping others to understand the data that you have been working on.

# Chapter 3: Machine learning

Machine learning is the process of training an algorithm to identify or learn patterns in data without explicit instruction. For example, an algorithm might be given a series of numbers (x) which it is asked to adjust by multiplying with a number (a) and adding a number (b) to produce a result that closely approximates the target (y). To exemplify, we might expect income to be an (approximate) function of age and a minimum subsistence level of income which everyone is expected to earn. At that point, we can estimate people's income based on their age. This implies a formula of the type:


$$f(x) = ax + b$$

This is also known as linear regression. We know from school that we can work out an analytical solution to this, but there is another solution too. How about taking a fast computer and trying 1,000 different numbers in place of (a) and 1,000 different numbers in place of (b)? Then simply calculate which combination of (a) and (b) that, on average, made (x) closest to (y). On a modern laptop, it takes well under a second and gives a numerical result that is very close to the exact analytical solution. We can improve it even further by using an algorithm that looks at how different combinations of (a) and (b) might give a more correct answer and use that information to make either (a) or (b) bigger or smaller. With that, we can get a more accurate answer in as few tries as possible.

For simple linear regression, it might be unnecessary because we have the exact mathematical solution, but for many statistical models there is no exact mathematical solution. Instead, we have to rely on approximation. Thus, solutions like the one above where we optimise **our guesses can be our only option. It is just our luck then that it turns out that such optimised guesswork performs so incredibly well** (Wigner, 1960).

So now your data is ready, and the problem is in place. It is time to estimate the right model.

- How do we identify the right model?
- Overfitting and underfitting
- What to do about fit?
- Classical models
- A warning
- Bonus: Cross-validation

## How do we identify the right model?

The best model is the model that is most accurate when encountering new data. We are interested in using the data we have collected to create a model that can inform us about data that is beyond what we already have today. It might be that we have 100,000 e-mails which we have read and then assessed that 10% are spam. The good model is the model which, based on the data, can develop a real “understanding” of spam such that we can implement it on our e-mail server to correctly identify and process all the e-mails sent to our e-mail server in the future. E-mails that the model, by definition, has never seen before.

We then estimate our model on training data alone and try to use this model to guess our test data. The charm is that **we know the truth of our test data** so we can measure very accurately how accurate the model is.

Take the example of the 100,000 e-mails. We would take 20,000 e-mails randomly (spam is still around 10%) and save it for testing. We would then “train” a model using the 80,000 remaining e-mails by showing it the contents of the e-mail and whether it was spam or not. When we are happy with the model's ability to predict whether an e-mail is spam or not in the training set (maybe it is right 95% of the time), then it is time for the exam. We take the model and test it against the 20,000 e-mails we saved. On the test set, it may hit right in 91% of cases. Then we can assume that we have a model that is in fact correct about 91% of the time.

## Overfitting and underfitting

The reason why we have to split out data in two (also referred to as test-train-split) is because modern machine learning is really good. So good, in fact, that with a sufficiently advanced model we can promise that any data scientist can guarantee 100% accuracy on training data. The most advanced models can completely “memorise” large datasets. In this case, they do not learn anything useful in general about the data. The phenomenon is called overfitting and is absolutely crucial in the quality of a machine learning model.

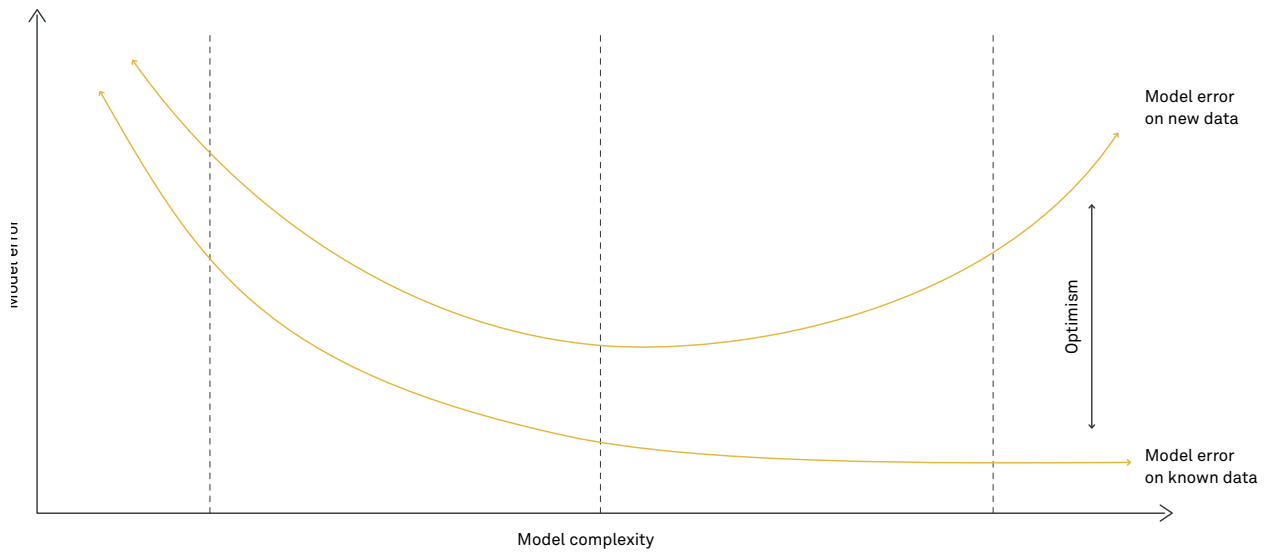
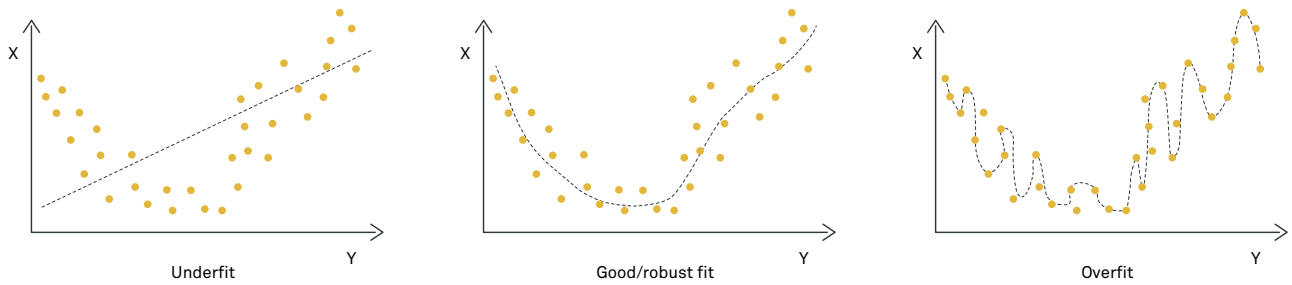
It can be illustrated as below.



### Overfitting and underfitting

Overfitting and underfitting are the two most important phenomena to balance when building models. Overfitting means that the model is too complex and learns nuances in data that have nothing to do with the real world but are about specific patterns in the examples we have. Intuitively, one can say that the answers are learnt by heart without learning the contexts that give the answers. Underfitting is the opposite where the model does not have sufficient complexity to capture the patterns that are in the data. Intuitively, it can be said that it is not smart enough to learn the information available in data.





The bottom diagram shows the essence of the challenge: the greater the model complexity, the better the model will “fit” on training data and initially test data because there is complexity to capture. After a point, learning starts to go awry. The model gets better and better on training data but begins to “overinterpret” data. When it later encounters new data (like the test set), it suddenly creates a lot more errors. It has not actually “learnt” the substance of the data but rather just memorised the particular data it was given.

The phenomenon is illustrated in the three figures above. For example, the complexity of the model can be seen as how “complicated” a line we allow the model to draw through our data points. In the first figure (“Underfit”), we only allow the model to draw a straight line. Here the model is just bad. It does not fit very well with data and both testing and training have many large errors. In the next figure, we allow the model more freedom. It must not make sharp turns, but it is allowed to bend the line. Now, we can see that it follows data

really well, and we can assume that if only future data follows roughly the same structure as our training data, then the model will probably do an excellent job on it as well.

In the third and last figure, we see the overfitting scenario. **Here, the model has been given too much freedom, and it has started to simply connect the dots.** It is probably 100% accurate (it hits all the dots), but if we put in a new dot, the chance that it would be particularly good on the line is very small.

## What to do about fit?

There are a few classic tricks in these scenarios:

1. **Test often.** Vary the parameters in your model and observe how the difference between testing and training changes. Keep increasing the complexity of your model until you see no improvements, or the improvements are only on training data. It is normal for training performance to be slightly better than test performance but too large a difference should be cause for concern.
2. **Support the model with additional information.** Also known as “feature engineering”. This is a manual process in which we try to help the model by removing distracting information and highlighting important elements. This assumes that we know what features in our data are merely distractions and which are important. It requires strong domain knowledge. In our spam e-mail example, a classic trick is to throw away words like “that”, “and”, “it”, “there” and “are” because they are not important to the content. This leaves fewer words left to distract the model.
3. **Get more data.** Overfitting is always relative to a given amount of data. Keep in mind that the challenge is that the model “cheats” by learning data by heart instead of the general patterns and intuitions in the data. The best solution is to give the model more data. With enough data, even a highly complex model can no longer “remember” everything and is instead forced to learn more general and useful properties instead.

## Feature engineering

Feature engineering is when a data scientist manually performs mathematical operations to help a model see the relevant patterns in data. For example, in a situation where we want to assess whether a credit card transaction is fraudulent, relevant features to engineer could be average transaction size for the cardholder, average transaction size for the seller, the number of times the cardholder has purchased from the seller in the past and how long it has been since the last purchase. Information like this would probably be very helpful to our model, while the name of the seller would probably just distract it. In the previous example with income and age, we might express age as “years since the 18th birthday”. We would do this because we know that it is not actually age that determines income but rather experience in the workforce. We can reasonably assume that a lot of people enter the workforce around the age of 18, so this might be closely matching the pattern in the data. We used our domain knowledge about employment and wages to “assist” the model in finding the pattern in the data.

## Classical models

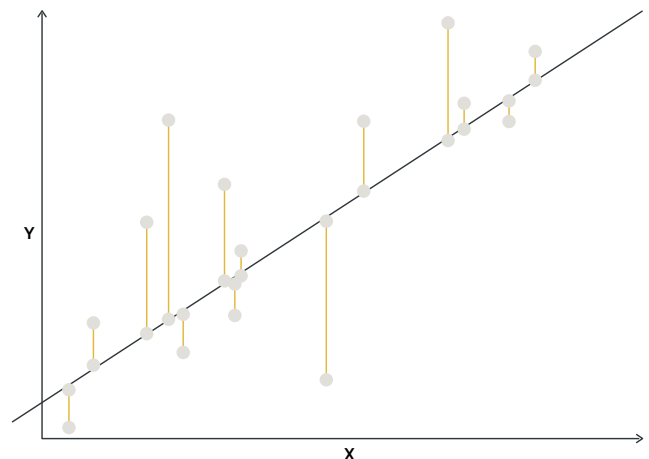
There seems to be an inexhaustible range of data science models, all of which are ideal within their own little niche. There are a handful of classics that everyone should know, which can often solve most problems adequately.

## Logistic and linear regression

Classical linear and logistic regression are still some of the most widespread and helpful models today. Both models work by trying to estimate a given target (for example, the price of a stock tomorrow) based on input (for example, prices over the past seven days) and then fitting a line through the data points.

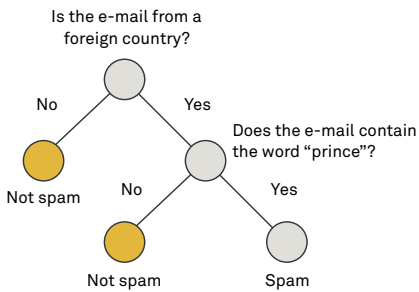
The line is drawn by estimating the line which most closely fits the inputs to the outputs like in the example in the beginning of the article. The figure below illustrates the model.

Logistic regression is a bit more complicated, although the principle is the same, but here the line can take other forms than a straight line. For example, it may take the form of an S-curve.



## Decision trees and random forests

A less famous alternative to linear regression are tree-based models. They are called tree-based because they are all based on a so-called **decision tree**. For example, a (very simple) decision tree to determine if an e-mail is spam might look like this:



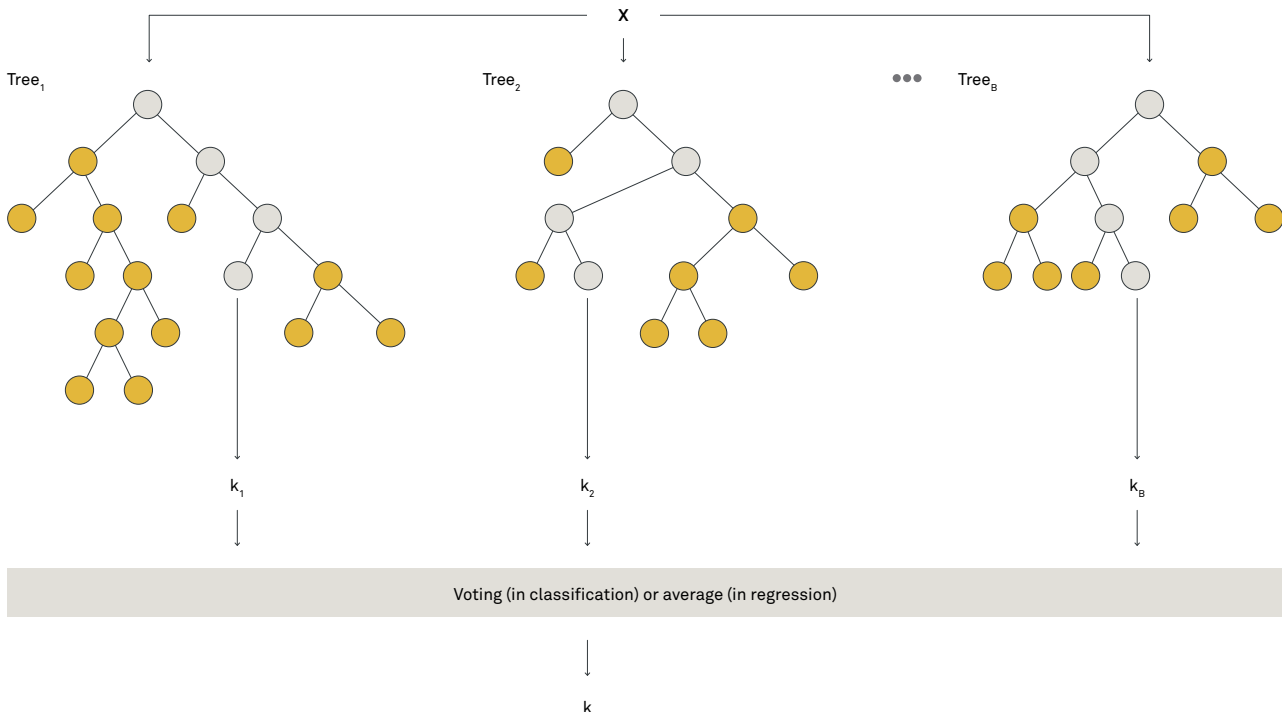
There are techniques for learning that kind of decision tree from data by constantly asking: "What do I need to split on now so that the groups I split into are most pure in terms of what I am interested in (for example, spam/non-spam)." As such, a decision tree resembles a flow chart where we ask different questions about the data in order to make a single prediction.

A development on the decision tree is the **random forest**. Here, we take a sample of our data (for example, 20% of rows and 50% of columns) and estimate a small decision tree based on this. Then we take a new sample of data and make a new decision tree. We continue this process until we have, for example, 50 small trees.

When we want predictions, we ask each tree and let them vote (or take the average of their values) such that the

prediction is based on the consensus of all the trees.

Exactly why random forest is such a great model is a mathematically complicated story, but random forest manages the overfitting problem much better than a single decision tree. Intuitively, the key is that no single tree is able to know everything, making it much harder for the model to merely memorise everything. Instead, each tree "does its best", which is alright in itself, but with a lot of them, rather accurate estimates can be obtained. Additionally, random forest has a lot more settings to tweak such as how much data does each tree have? How many trees in the forest? How complex is each tree allowed to grow? We have a very natural way to increase complexity (like adding trees to more data to each tree).





**Gradient boosting** is a further development from the random forest which is worth mentioning. It is a rather complicated model in which each tree knows the prediction errors of the previous trees, meaning that the trees are allowed to support each other and specialise in certain nuances of the data. It is an extremely powerful method known for winning a lot of smaller data science competitions on the internet.

## Deep learning

Deep learning is probably one of the most famous techniques right now and is the driving force behind almost all major artificial intelligence breakthroughs in recent years. The technique has become very advanced, and it is a whole field in itself, which is why it is almost unfair to describe just one technique.

In short, deep learning works by “stacking” mathematical functions on top of one another with non-linear transformations in between. If that did not make sense, think about it like this; maybe you have three “inputs” for

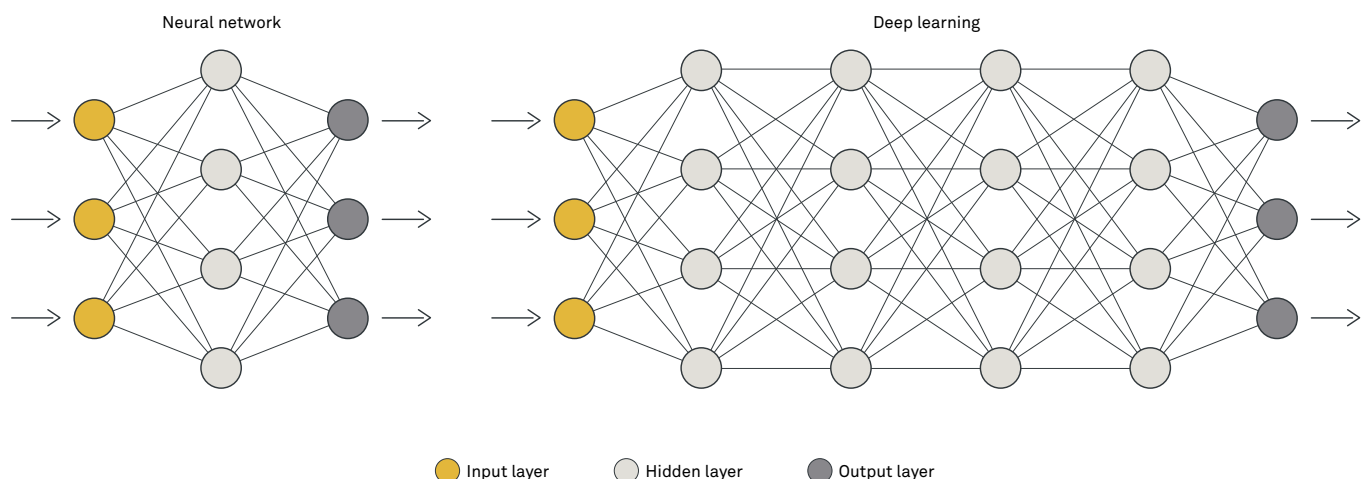
your model (height, age and income), and you want to guess what gender people have (male/female). In a deep learning model like the one below, you take the three parameters, cross them in a number of ways (in the illustration, four ways) and then transform them through one of the different “neurons” (the first layer after input). Each neuron has a non-linear transformation, so the output becomes between 0 and 1, and high and low numbers come closer together, while numbers around the middle spread out between 0.2 and 0.8 (this is also called a “squashing function”, as it squashes the numbers into a given range). The four new outputs are then combined with each other in the second layer of the model, whereupon they are transformed through new non-linear functions, then on to the third layer etc. Finally, they reach the last output neuron, where a number close to 1 means female, while a number close to 0 means male (and when we use it later, we would set a cut-off point at 0.5).

When training the model, weights are identified for all inputs between all layers (so that inputs from the previous

layer are not just naively put together). These weights might increase or decrease the signal from the previous input. The specific properties of the non-linear functions are also estimated (when should they begin to “squash”? At 0.2 or 0.3?). It quickly becomes very complicated, and therefore deep learning is an extremely complex method. The advantage is that it can capture even the most incredible nuances in data. This is why it is often used in, for example, languages and images that contain a lot of information and complexity. The disadvantage is that it is very prone to overfitting and is therefore extremely data-intensive.

## A warning

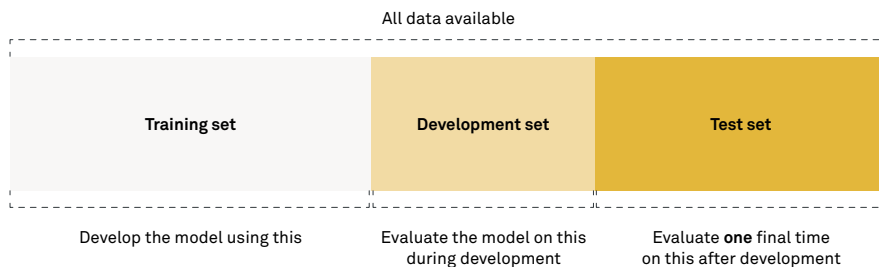
Above, we have described a process where a single chunk of data is taken aside for testing while training on the rest. If this goes on for a very long time, and a large number of model types are tested, then the process of selecting a model can be driven by one’s test set rather than something real in the data. Many people therefore work with an extra split, a so-called development set.



Basically, data is divided into three sets: 60% training, 20% development and 20% testing.

Training is used as before to estimate the model. Development is used to “test on” during development of the model (it is used multiple times). The test set is now used only once – as a final exam.

This ensures that the test set really is new data, and that the precision we report for our model will be in the final set.



### Bonus: Cross-validation

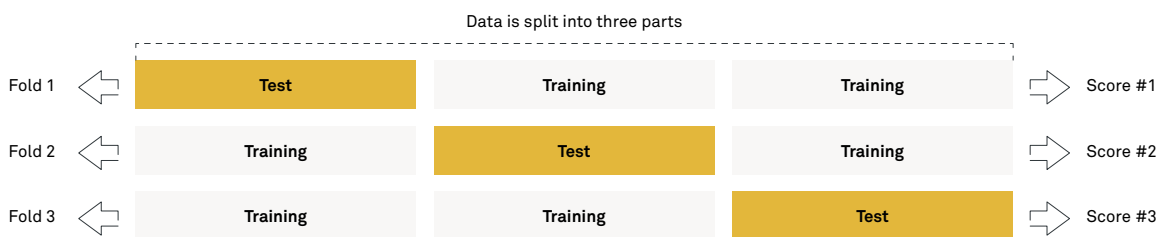
An alternative to the test-train-split is cross-validation. Data is split into a number of parts (for example three, 33% data in each). The model is then estimated and tested once for each chunk of data – all the time with a new chunk for testing and all others inside as training kits.

So, for each “fold”, a test score is obtained, and we then calculate an average across all folds as our final score.

Cross-validation is generally considered both more accurate and safer than a simple test-train-split, but it can also be time-consuming and

complicated to implement because the model has to be estimated many times.

Note that it is often recommended to split data into 5-10 pieces when doing cross-validation. This ensures that the test set has a reasonable 10-20% size each time but once again increases computation time by an equal factor.

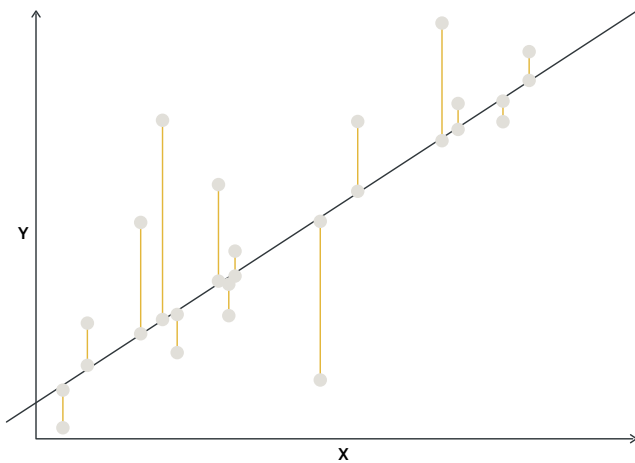


# Chapter 4: Definition of purpose

If there is one thing we have learnt during our time in Implement's data science team, it is the importance of a clearly defined purpose all the way down to the model specification. Data science is relatively unique in the way that goals and purpose are often a part of a formula called a loss function. In short, it is the art of ensuring that **our data science solution solves the valuable problem** we have identified, and not a simpler (but related) problem, and that it happens under realistic conditions. If we are not intensely conscious of our loss function, we risk, for example, prioritising our customers incorrectly, or we risk optimising for situations that do not arise.

So far, we have discussed what data science is, how we identify a valuable problem and how we make sure that we have the data we need to solve our problem. The next step in the process is to define the goal and the boundaries of our problem.

- What is a loss function?
- Why are loss functions important to the business?
- Tip #1. Weighting of cases
- Tip #2. Not all errors are equal
- Tip #3. Allow the model to gracefully give up
- Baselines



## What is a loss function?

When training a machine learning model, we often set up a loss function. The purpose of a loss function is to describe mathematically to the model what we are trying to optimise for. Typically, we deal with three types of problems in machine learning:

1. **Regression:** Guess a number. For example, the price of a stock or the age of a person.
2. **Classification:** Guess a type out of multiple possible. For example, the diagnosis of a patient or whether a picture contains a cat, bird or dog.
3. **Binary classification:** Yes/no questions. For example: "Is this spam?" or "Is this a dog?"

For each of these types of problems are some canonical formulations of loss functions. For example, to the right is an example of a regression problem. The diagonal line is our model, and the dots are our observations (i.e. the input data). We want a line (model) so that the distance between the line and the points is as small as possible for all the dots. Technically, the distance between the dots and our line is called our "residuals" or errors, and that is what we are minimising when estimating the slope of our line.

For classification, there are several variants, but a natural solution is simply to look at the proportion of correct answers and try to estimate an algorithm that minimises the number of incorrect answers.

## Why are loss functions important to the business?

It is often surprising to people closest to the business that when we make regression models (i.e. guess a number like the number of goods sold tomorrow), standard practice among data scientists today is not to minimise the average error but to weigh more errors exponentially heavier the larger they are. This formulation has some appealing mathematical properties, but it is a very important business decision that often just slips through. In practice, the business often has strong opinions. Maybe errors below a certain size are truly insignificant, or maybe errors above a certain size are all equally “bad”. If this is done wrong, a data scientist may mistakenly develop an algorithm that is rejected by the business because the algorithm optimises too much for some extreme periods (or worse, non-existent scenarios!) at the expense of everyday operations. On average, the model becomes too imprecise, and the project is abandoned. It is likely that a proper wording and discussion of the loss function would have solved it.

### Tip #1. Weighting of cases

Ask the business early: “Are there certain things more important to identify than others?” The answer is almost always yes. Maybe finding the cancer cases is more important than avoiding taking in too many patients. Exactly how much more important? This is an important business question that needs to be answered. For example, a project in Denmark found that it was 16 times more costly to overlook a “yes” that should actually be a “yes” than to say “yes” to something that should have actually been a “no”.

When asked the question, it appeared that there was a clear business calculus that could provide the exact answer. Obviously, it dramatically changed the algorithm compared to if “yes” and “no” had been equally weighted. Most loss functions allow weights to be applied to individual cases during development to allow the model to focus on the most important problems.

### Tip #2. Not all errors are equal

Commonly, the size of the errors is also not of equal importance.

#### Errors so small that they do not matter:

Depending on the solution’s use, a +/- 1% error may be negligible. Then it might be helpful to just ignore them in the loss function. Techniques such as support vector machines make this very explicit.

#### Errors so big that they do not matter:

An even more important category is the big mistakes. Maybe obviously large errors are easily caught by humans or are possible to check using rules. It might even be preferable for a solution to provide an obviously false answer rather than deliver something close. An individual model can often be designed, detecting if a case or prediction is highly likely to be obviously wrong and can send it for manual handling.

Between these two filters, the model gets a chance to focus on the cases where it can make a valuable difference. The cases can be sorted from data (here you have to think carefully!) or implemented as part of the loss function.

### Tip #3. Allow the model to gracefully give up

If a problem proves particularly difficult, the introduction of a “don’t know” or “gray” category can make business sense. In a classification problem, this is often done by taking a large number of rare types and adding them to one large “gray” class, which is then handled manually or according to some rules. In regression, it can be done by first estimating a model that tries to estimate the exact number, after which you inspect the cases in which the model makes big mistakes. Next, an individual model is trained to assess whether a case is likely to have a large or small error. Finally, the original model is only retrained on the proportion of cases that had a small error (the errors are then reexamined, and we would expect to generally see better performance). When the model is to be used, the “big or small error” model is used first. Cases with expected large errors are handled manually, while only those with expected small errors are allowed to continue. This can be seen as a form of intelligent filtering.

For a more comprehensive overview of different models, we recommend [scikit-learn](#) (scikit-learn, 2020).

### Baselines

A baseline can be whatever the current solution is, or it may refer to industry standards. This might be human judgement or a simple-to-implement rule like always predicting “A” based on a few conditions.

Baselines are hugely important. The business is rarely interested in knowing that our model is right 84%

of the time in isolation. In fact, relevant stakeholders will often interpret “84% accuracy” as “being wrong 16% of the time”, and this may dissuade them from pursuing a data science solution to the business problem. They want to know how much better than the current practice our proposed solution is – the so-called “model lift”. (Or how close can we get to the human baseline with an algorithm that can work for free 24/7?)

Deciding what the current baseline performance is can be tricky. Human baselines can be especially tricky, but as a part of the data exploration process, we can often get a good estimate. Look out for differences in performance between individual humans in the previous process. Consider if there are some unique qualities in having a human handling the process like explainability or handling of edge cases and consider how this can be emulated using data science techniques. In a lot of cases, businesses will be surprised by their human baseline, as we tend to assume that humans are very good at or sometimes perfect for their job. A model with 85% predictive accuracy might not sound great, but if it turns out that the human baseline was 75% accurate, it can be an excellent lift but a politically daunting thing for the organisation to face if it was previously assumed that the process accuracy was near perfect.

### **Model lift**

A concept of how much better a model is compared to a baseline. It may be tempting to focus solely on the precision of a model, but in practice it is often the improvement compared to the starting point that is the deciding factor. Like how much better than our previous methods or a known industry standard our solution is.

### **Summary**

Before we start selecting and designing models, we should make it clear what the purpose is right down to the mathematical level. Are all types of errors equally important? How good is the baseline we are up against? What is the cost of making a mistake versus sending a case for manual processing?

Those questions are at the heart of designing the loss function and purpose of a data science solution.



# Chapter 5: Modern deep learning and fine tuning

Over recent years, a new significant trend in data science, and specifically in deep learning, has gained a foothold. The models have become larger and more complex, and fine tuning technology is used to carry information from one domain to another. The effect of this trend is better models in niche areas than we have ever had before. We can suddenly achieve fantastic results with one hundredth or one thousandth of the data we have used before, and this opens up completely new avenues.

The text below focuses on this development within language analytics, which has seen the greatest breakthroughs recently. This is not to say that we have not seen incredible applications of the technology in computer vision (Microsoft ResNet (Lenyk & Park, 2023)), speech-to-text (OpenAI Whisper (Alec Radford, 2022)) and image generation (Stable Diffusion (Esser & Rombach, 2022)).

## Deep dive on deep learning

In order to understand the new possibilities, we need to understand deep learning a little better. In chapter 4, we talked about how deep learning is a series of layers of alternating linear and non-linear transformations. In the lower layers, raw data is fed in, and in the upper layer, predictions are extracted. Deep learning uses numerical optimisation methods to find out how the linear and non-linear transformations should be adapted so that the raw data makes the right predictions. Given enough layers of transformations, deep learning is a so-called universal function approximator, which means that, in principle, it can learn approximately any function of unlimited complexity.

If it sounds a bit abstract, think of some of the biggest models today, for example, GPT-3 (Brown, 2020) or ChatGPT. A gigantic model with over a trillion simulated neurons. In the lower layers, the model takes words from a sentence, and in the upper layer, it tries to guess what the next word in the sentence is. To solve that task, the model must first learn basic grammar. But to become really good, it needs to also learn context and a form of approximating “common sense”.

Grammatically correct sentence:  
Anders Fogh Rasmussen is a Danish ...  
butcher/ballet dancer/schoolteacher.

Common sense phrase:  
Anders Fogh Rasmussen is a Danish ...  
*politician*.

There can be many points for the model, i.e. correct words one after the other, in just learning how to link simple words together correctly, such as “is”, “a”, “the”, “that”, “and”, “but” and “so” one after the other. However, if the model is to include the last details, it will learn that “Anders Fogh” is most often followed by “Rasmussen” and not “Mogensen”. And it will learn that when people write about Anders Fogh Rasmussen, words like “politics” and “left” often follow. When a model gets as big as GPT-3, it even starts to show an understanding of concepts you would not expect. Below is, for example, a real excerpt from GPT-3. (Bold text is our input. Italic text is written by GPT-3):

**“Implement Consulting Group is a world-leading specialist in the field of management consulting, providing services in the areas of Strategy, Sales, Marketing and Organizational**

*Transformation. Implement Consulting Group has offices in Denmark, Norway, Sweden and the Netherlands, and employs around 500 people.”*

Given just four words, GPT-3 correctly deduces that it is a management consultancy, although it is probably not the most difficult. It “knows” a lot about Implement’s business areas, but most notably it finds that we are based in Denmark, Norway and Sweden. We currently have no office in the Netherlands but are located in Germany and Switzerland.

We are not quite there where the model has real “human style” general intelligence. But we are far beyond just creating grammatically correct sentences. GPT-3 has embedded real “knowledge” about a relatively obscure concept such as Implement together with every single piece of knowledge on the internet from medical procedures, material science, history and geography to business.

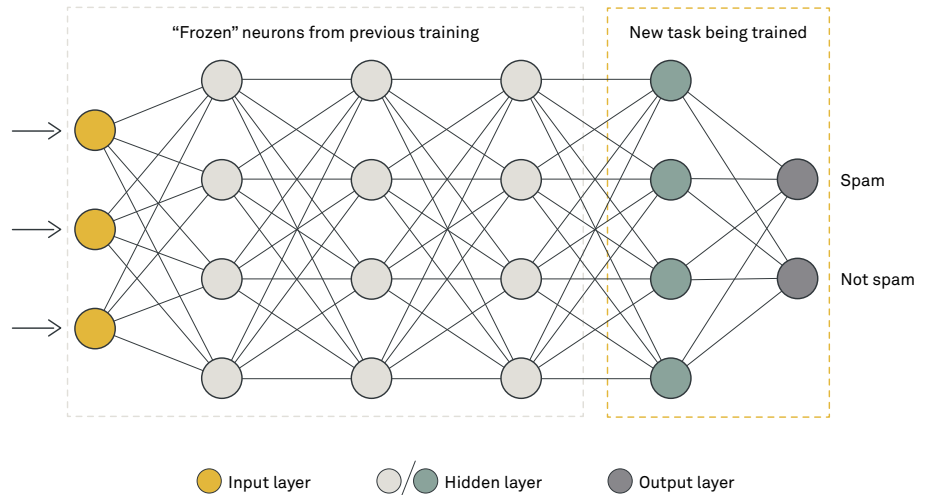
## Fine tuning

But what are we going to use it for? It may be fascinating that a model has learnt grammar and has a sufficient understanding of all the information in the world to have a cursory knowledge of Implement, but how do we get any value out of it?

Once again, the answer is found in the hierarchical structure of deep learning. Gigantic models such as GPT-3 are trained on correspondingly gigantic amounts of raw text data retrieved from the internet. More data than any company can generate on more expensive hardware than most people can afford.

But when this is done, we have new opportunities. It is possible to move knowledge from one domain to another. In the case of GPT-3, we can, for example, replace the upper layers in the model so that instead of predicting the next word in the sentence, it tries to predict whether the sentence it has just been given is part of a spam e-mail or not.

This is “pre-training” on a simple task where we have close to unimaginable amounts of data (predict the next word in a random sentence from the internet). During pre-training, the model learnt basic grammar, the mutual meaning of different words and context-dependent meaning. After this, we can then fine-tune our model for the final task, like spam filtering.



### Fine tuning

Fine tuning is when we use models trained on one domain and apply them to another. We might need to develop a model that can recognise trucks. It can be difficult to get enough pictures of trucks. But if starting from a model trained to recognise cars (of which we have many images), and the training continues our modest collection of truck images, then we might be able to reach the goal. Many of the experiences from the car model will be able to be transferred nicely. For example, wheels, lights and windows suggest a car, while asphalt and green leaves suggest other things. We can make do with a fraction of the data it would require training a model from scratch.

### Fine tuning in practice and standard models

The possibility of fine tuning has created a practice of creating so-called “model hubs” or “model zoos”, where companies or research institutions make various basic models available on open-source terms. These models can then be downloaded and fine-tuned for one’s specific purposes.

#### Important basic models

Some of the most important foundation models are mentioned here:

- **BERT:** A completely generally applicable, fast, flexible and well-documented language model. It clearly works best in English but can also handle Danish. There are many additional variants of BERT, for example, BioBERT, RoBERTa, BART, BERT-Chinese or DistillBERT.
- **RoBERTa and Ælæctra:** A version of BERT already fine-tuned in Danish, which can be further fine-tuned for specific tasks.
- **GPT-Neo:** An open-source variant of GPT-3.

- **Big Science BLOOM:** The largest open-source text-generative model.
- **Whisper:** A speech-to-text model with sublime performance in English and also really strong performance in Danish.
- **Stable Diffusion:** Text-to-image model that allows users to create images from scratch based solely on text input.
- **T5/OPUS:** Machine translation.

Developing and fine tuning these giant models from scratch is a complicated process traditionally reserved for very mature and resourceful organisations. Fortunately, there are tools that help and methods to get started in other ways. Despite its bizarre name, **HuggingFace** has become a central tool in modern data science for this very purpose, and thanks to their Transformers python library, it is not nearly as complex as it was just a year ago to work with these models.

**A note on simple and advanced concepts**

In this document, some liberties have been taken. In practice, we cannot count on – and especially not in language – grammar to necessarily lie in the lower layers and advanced concepts in the upper ones. How the model distributes its learning is a complicated field to study, where the research is longest in image recognition. It is a helpful pedagogical parallel here to understand fine tuning and modern deep learning.

In summary, fine tuning and pre-trained models allow us to leverage much larger and more powerful machine learning structures than previously. It is one of those rare times in data science when we really might just get something for free in terms of performance, simply because someone, somewhere, has already spent a lot of resources.





# Chapter 6: MLOps and testing

MLOps, or DevOps for machine learning, is a methodology that aims to ensure that machine learning models get placed into production and continue to provide the expected business value over time. The goal of MLOps is to facilitate the deployment, monitoring and maintenance of machine learning models in a production environment.

MLOps involves using tools and processes to automate the entire lifecycle of a machine learning model from initial development and training to testing and deployment to monitoring and maintenance. This can involve things like version control for machine learning models, automated testing and validation and infrastructure automation.

By using MLOps, organisations can improve the speed and reliability of their model development and deployment, allowing them to incorporate machine learning faster and more reliably into their operations.

A key aspect of MLOps is the continuous effort of the work. Once a model is developed, it is deployed and put under monitoring. Based on this monitoring, maintenance will be performed, and we might even further develop the model with additional functionality. In any case, it will not take long before we will have to do another round of deployment and update our monitoring process.

This follows closely the canonical DevOps methodologies, but we want to highlight some specifics here.

## A word on version control and common software development practices

Below, we assume that the reader is familiar with common software development practices and tools like version control through tools like Git and good practices like code review, documentation and a general idea about what software testing is.

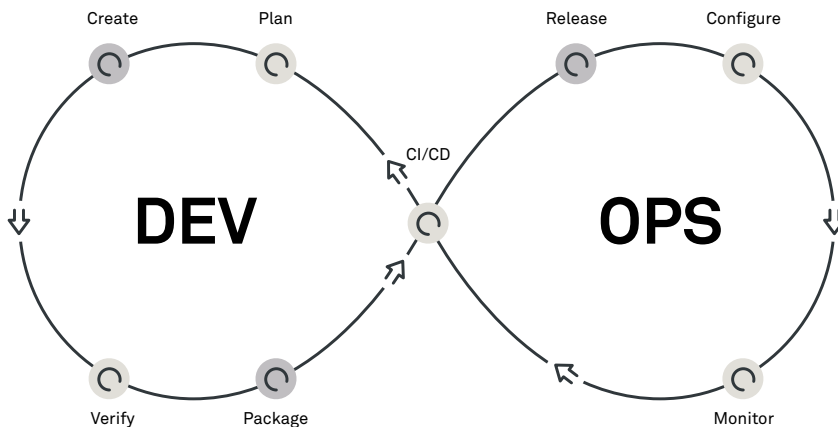
## Deployment and testing

A good deployment is **automated** and thus something we can do often and easily with little to no cost, and a good deployment is **safe**, meaning that we have no risk in terms of the quality of the product or the work we put into it.

## Automated deployment

A lot of modern software development tools allow the construction of various pipelines. These usually function such that when code on the system is updated to a certain point, an automated system picks it up, verifies the integrity of the code, packages it and then releases it for deployment.

For a fast and agile setup, the full end-to-end automation of this process is critical. Manual steps greatly slow the development process and introduce the risk of human error, as we forget to perform certain deployment steps. Tools for performing automated deployment are found on all major cloud vendors and have strong open-source alternatives. Within Microsoft Azure, it is known as DevOps. In the Amazon system, it is known as SageMaker. A great open-source alternative would be Jenkins.



The image above shows the canonical DevOps methodology.

Common steps to an automated flow are as follows:

1. **Build the code:** To the extent that we can have compile errors, syntax errors and generally ensure that the code is technically able to execute.
2. **Automated unit testing:** A suite of micro-tests are performed on individual functions of the code.
3. **Performance testing:** A test to verify the predictive quality of the model.
4. **End-to-end test:** Deployment to a test or pre-production environment where the solution can be tested and integrated with other modules.
5. **Manual approval:** Ensures that any potential non-automated checks can be done and a proper release to deployment can be scheduled for a convenient time.
6. **Release for deployment:** The code being moved to a production environment.

These steps are further elaborated below on safe deployment.

Ideally, this means that a data scientist or developer merely pushes their code to the main working repository, and once a fellow colleague has read over their code and approved it, the pipeline automatically detects the change, and usually within minutes, the system can be updated and kept running either in a test or an actual production environment.

## Safe deployment

In order to avoid confusion and system degradation when potentially performing multiple deployments per day, a lot of safety is required in

connection with deployments. This is usually done through testing. It should be stressed that testing is a large field in itself. Many small hobby or start-up data science projects might initially have little or no testing of their code and merely focus on getting a functional system up and running. This is completely normal. On the other hand, on large enterprise systems, entire functions are often dedicated to testing with test engineers, test designers and test managers.

In MLOps, testing can be broadly divided into three groups: unit testing, CI/CD testing and performance testing.

**Unit testing** is a common software development practice in which individual pieces of code are tested. We might have a function which unpackages a data file, and a unit test might give this function a small known static data file and check that when the function is run the result returned corresponds to a known value. Or maybe a function transforms a particular column feature of our dataset, and we might feed the function three static example rows with known results and check that these are returned as expected. Most programming languages used for machine learning have well-established unit testing frameworks like Python's built-in unit test or open-source PyTest.

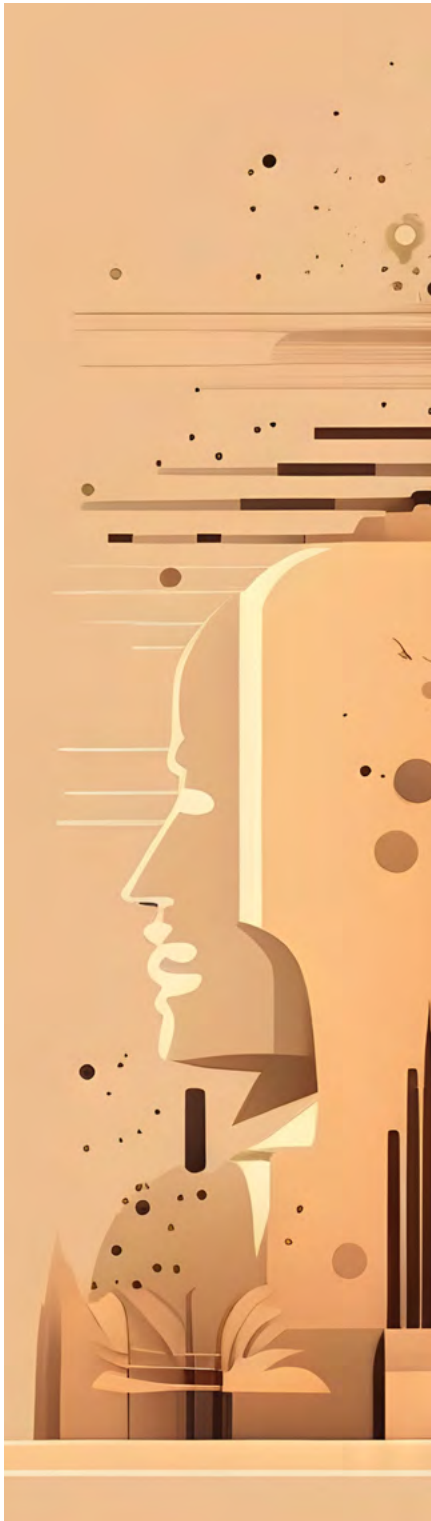
**Performance testing** is a less well-established methodology. It has several other names and depends a lot on the solution at hand. It might merely automatically push through a test dataset of the entire pipeline and measure the performance in terms of a metric like accuracy and only continue deployment if a certain threshold is reached. It might compare to an automatic baseline like a similar

model or the existing solution and only allow deployment if the new code sees improvements or identical performance to the baseline.

In a lot of cases, performance testing is also done over a longer period of time. Code can be pushed to a "test" environment before going to production. An organisation might have a convention that code must reside in the test environment for between 24 hours to a week before getting moved into production. A test environment will shadow the production environment and mirror user behaviour on the production system, but not actually provide input to users. Common design patterns also include **canary deployment** in which a subset of real users get to reach the new system initially and this group is expanded over time as the deployment is deemed successful. Another pattern is the **blue/green deployment** in which both systems run in parallel, and some process intelligently decides which to serve to individual users.

**End-to-end testing** is usually conducted over a time period spanning from an hour or two or (more commonly) a day to a week. The new solution changes are placed in a test or pre-production environment, mirroring the activity of the main production system and thus simulate an actual production deployment. This includes scaling with users, and it includes integration to various other systems in the IT landscape. The system will collect detailed logs of everything from predictive performance or memory usage to a final evaluation at the end.

Once all tests are passed, we should know that the system is functioning and that it improves on the current system according to our standards. The only thing missing is any potential



manual approvals and we can then release the solution to our users. In some cases, we might not have nor need a manual approval step, and our deployment can be fully automated.

**Manual approval** is a common step to collecting various human-centric quality checks which might be required for regulatory purposes. Ideally, this step should be fast and agile. As many tests and approval should be collected in the previous steps as possible. It can, however, be difficult to automate security screenings or reviews on the outcome of end-to-end tests. It might also often be necessary to schedule the actual production deployment, for example, to a period with low user pressure.

### Monitoring

Once a model reaches users, we start to generate usage data. This is incredibly valuable for future development and to fix mistakes, before they grow too large.

Most programming languages and data science tools allow various forms of logging. Python has built-in logging libraries and deployment tools like Tensorflow Extended and PyTorch Monitor module or is often part of deployment pipelines in the various cloud providers' solutions.

Common design patterns to MLOps monitoring include:

**Model performance:** It is important to monitor the performance of your machine learning models in a production environment to ensure that they are providing accurate and reliable results. This can involve tracking metrics like model accuracy and monitoring for drift, or changes in the distribution of your data over time. It can also include systemic measures

like latency, throughput and the volume of user requests against the system, how long time each prediction takes to obtain or how much memory the solution consumes.

**Use alerts and notifications:** Set up alerts and notifications to alert you when the performance of your machine learning model deviates from expected levels. This can help you quickly identify and address any issues that may arise. More detailed information can be found in dashboards.

In summary, we are going to want to deploy often so we can get the best possible system. This means that our deployment needs to be automated from code change to system update with several testing and monitoring steps in between.



# Chapter 7: Ethics

In the field of data science, ethics are very important at the moment and for good reason. When we work with data science, we often get very close to people, and we use tools that we do not always fully understand. The fields of data ethics and data regulation are in about as rapid a development as the field of data science, and we see legislation from both the EU and other actors trying to establish some frameworks specifically around the use of artificial intelligence and data collection.

Within moral philosophy, ethics is the discipline that deals with the question of what is good and bad or right and wrong. Here, however, we will specifically deal with applied ethics, which are questions about what a person is required to do (or allowed to do) in specific real-world situations.

We should note that what we write here is very much our own take on ethical data science. We have studied and worked actively with ethics both in general and in data science for several years. In addition, we lean on [People and AI Research](#) (the PAIR group) and their work with the [People + AI Guidebook](#). However, we have not contributed philosophical breakthroughs ourselves, and we cannot claim to systematically rest on an established moral philosophical system.

## Privacy

We want to go over privacy issues lightly here. It is not because it is not important (it is really important), but for the sake of space and focus in the guide. For a comprehensive insight into the privacy discussion, the [UK Information Commissioner's Office](#) (ICO, 2020) is a great place to start. Privacy takes other forms beyond the purely legal aspect like GDPR. Fundamentally, as with a lot of applied ethics, it is about finding the balance between the good that we are able to do and the risks and actions we must take to do it. The discussion is most fruitful when it centres on the things that we can do best to allow people control and agency in our solutions. Like allowing people to see their data and obtain an explanation behind an algorithm's decision. It is also fruitful to discuss technologies which might aid in the safety of solutions from basic things like encryption to advanced solutions like [differential privacy](#) in which we add some wrong information (so-called noise) to the data. That is, if data were to get into the wrong hands, it is not known which is real and which is false. Adding the false information correctly will not affect the data science value of data.

A very basic way to view things is with a simple dos and don'ts of ethical data science:

### Dos:

- Create value.
- Prolong life.
- Make people's lives easier.
- Make expertise available for everyone.
- Automate tasks to allow people to do more valuable things.

### Don'ts:

- Bias (meaning systematic inaccuracy in the model).
- Discriminate (meaning the specific kind of bias related to protected groups like gender, race and religion).
- Obscure or manipulate.
- Lie.

## Make sure what you do is valuable

When we consider the justification for our data science solutions, it is important to assess both the good and the bad. Data science has tremendous potential, and we have examples of great goods being obtained like [AI saving people's lives](#) (Corti.ai, 2020), [making people's lives easier](#) (Muhajlovic, 2020), [making expert intuition available for more people](#) (Seeing Potential, 2020) and [automating tasks to free people to do more valuable things](#) (Enversion, KAUNT, 2020).

AI and data science create – and must create – a lot of value. If a solution is not valuable, it can hardly be justified under any circumstances. ([DeepNude](#), (Vincent, 2019)).

There is a stark difference between mining patient journals to advertise phony nutritional supplements to them and trying to prevent strokes (Enversion, Sundhed, 2020).

Almost all data science projects try to be valuable, and almost all have a firm grasp on their impact beyond the immediate profit motives. It is so much more important to keep it this way.

### Avoid bias and unreliability

The first class of ethical problems to deal with are quite statistical in nature. Bias is when a model exhibits systematic errors where it might overclassify cats or spam or consistently predict the rise of a stock too low. Unreliability happens when our model makes more errors than we anticipated, but there is no systematic structure to these errors.

Both challenges are a business challenge but by extension also often an ethical challenge, because it means that we are potentially releasing a model that is less valuable than we expect, making any judgements about its merits questionable.

Both unreliability and bias usually occur in two ways: (1) we have been careless and made a mistake in our data processing, or (2) our data was poor and did not fit with the reality in which the model was to be used.

We can remedy point 1 by using a thorough method. Thoroughly test the models (chapter on machine learning) and make sure that the models are doing as expected (chapter on purpose definition). Another important technique is to focus on outliers and edge cases, making sure that there is

an elegant handling of these as well. Either making sure that the model handles this well, or that there are alternatives in place (see the People + AI Guidebook, the chapter [Graceful Failure](#) for great inspiration). Finally, finished models can be subject to simulator studies where counterfactual cases are processed through the model, and we are able to closely investigate the provided results.

Point 2 is best remedied by knowing your data intimately. Good data exploration is key (see chapter on data exploration). Both the actual statistical structure of the incoming data as well as its origin.

We consider it to be an **ethical responsibility** of a data scientist to ensure that his or her models are **robust**, and that the data on which they are built is **clearly understood**.

### Avoid discrimination

In this context, we use the term discrimination to imply that a model has learnt an undesirable behaviour to the detriment of protected social groups. Classical examples of this are racism and sexism. Discrimination can occur because of bias (either in the data or in the model development), but we will extend it even further. Even a perfectly designed model on a well understood data foundation can exhibit discriminative properties if it systematically disadvantages protected social groups.

This means that we have two types of discrimination:

- 1. Discrimination in model development.** When the data is good, the question is well formulated, but some error introduced by the data scientist introduces discrimination. This is a relatively easy case as we can identify it through testing and data exploration with special attention and care to protected social groups. Once we have identified the issue, we can look for a solution and fix whatever error introduces the discriminative property.
- 2. Discrimination in data.** Often, our history is undesirable. We might have had discriminative policing or crime patterns. Cathy O'Neil so thoroughly demonstrates in her book, *Weapons of Math Destruction* (O'Neil, 2016), that cleaning data can be difficult, impossible or make things worse. In fact, she ends up concluding that it is best to include the discrimination and, instead, pay attention to it and deal with it by building functionality to support the model. This could be additional automated checks on input and output, being more vigilant about moving things to manual review, or in some cases adding counter-biases manually to certain predictions. This obviously requires a thorough study of the model's discrimination (for example with simulations of counterfactual cases).

When there is discrimination in data, it is often because there is an existing discriminatory practice, and as such, data science has an opportunity (some might even argue an obligation) to cast light on this and try to improve on it.

In the words of [John Shawe-Taylor](#), [UNESCO's \(UN\)'s Chairman of Artificial Intelligence](#) (Synced, 2019):

“Humans don't realize how biased they are until AI reproduces the same bias.”

Before we attack a discriminatory model, it is often worth asking why data is so unjustifiably discriminatory in the first place. Is it possible to make a less discriminating algorithm than the human alternative? The ethical question is often about which of the following three solutions is least problematic:

1. The improvement is worth taking despite the concern about automating discrimination.
2. The discriminatory process can be completely abolished or radically changed.
3. Stick to the discriminatory practice that there is.

All three options should be considered and can be valid. In the following, we assume option 1 was chosen, and that we have decided to develop a model and want to pursue an as ethically defensible solution as possible.

### **Avoid lies, manipulation and obfuscation**

Even if our model is great, it is very important how we talk about it.

Communication around our model is very important. Miscommunication is one of the top reasons why people get frustrated or hurt by data science solutions, so let us make sure that our users understand our solution.

We can imagine a data science solution that deals with people's claims for compensation from an insurance company. We assume that the solution is relatively well functioning and does two things: (1) assesses whether people have filled out their application with sufficient information and (2) gives them an immediate assessment of their claims.

Here are three examples of bad ways and one good way to talk about data science.

#### **Avoid turning data science into black magic.**

✗ *Thank you for your application. We will now process it with our Auto-Request Analysis\_AI™ and give you an answer faster than ever before!*

This response risks leaving people confused as to what exactly happened. What is this system? Why is it happening? What exactly is it doing? What if I do not like it? It is an unhelpful marketing hype.

#### **Avoid hiding the model.**

✗ *Thank you for your application. We will process it and get back to you as soon as possible.*

If a user then a few seconds later receives a message that their application is faulty, or that an initial assessment looks poor, it will cause confusion. Was it really processed? Or is this an error? Why did it happen so quick? If you are automating tasks, do not hide it, or it is going to cause concern and confusion.

#### **Avoid explaining the technology.**

✗ *Thank you for your application. We have recently implemented a non-negative matrix factorisation algorithm that helps us sort incoming applications and identify missing features. We will be able to respond to you in a short moment with 96% accuracy.*

This is closely related to the previous example as it leaves most users questioning what happened. While having a highly accurate technical explanation available for expert users is good practice, frontloading it for everyone is a recipe for confusion, frustration and misunderstandings.

#### **Explain what is happening, why it is happening and what people can do.**

✓ *Thank you for your application. We are now processing it automatically for any missing information and – if possible – we will provide you with an initial assessment of your claim. If you have any questions about this process, simply answer this e-mail. One of our representatives will be in contact with you soon.*

Be honest about using an automated system. It is rarely important to people whether it is machine learning, a set of simple rules or a robot. Make sure that it is clear what the analysis is about (here: missing information and initial assessment) and equally important: what does the analysis depend on (here: the application). It is also great practice to give the user an opportunity to communicate on automation. Most data science solutions sometimes make mistakes, and it gives the user room to help deal with them in addition to a chance to respond to the results in general which can both help users gain necessary agency as well as provide important feedback.

## Final notes on model communication

We believe that many of the ethical challenges in data science can be solved through good user experience design (UX). If you make sure that your users/citizens/customers understand what is happening, give them the ability to intervene and provide feedback and ensure graceful handling of failures, you have a very good opportunity to inspire the kind of trust in the model that is the foundation for ethical development. For additional inspiration, see chapters [Feedback + Control](#), [Errors + Graceful Failure](#) and [Explainability + Trust in People + AI Guidebook](#).

# Chapter 8: Sustainable data science

The rise of data science has raised concerns about the sustainability of this practice. The environmental impact of the data centres of the world is a growing concern, and especially the supersizing of some deep learning models over recent years has required a staggering amount of energy to produce. In a growing global environmental crisis, the data science profession, like any other, needs to pay close attention to the resources they use when creating our solutions.

## Recycling in data science

We estimate that a model like OpenAI's GPT-3 cost USD 4.6 million and about 1 GWh of total energy to produce over the course of about 30 days. Maybe horrifyingly, that is the equivalent of 500 tonnes of coal if run exclusively on coal power. Add to that embedded emissions of the hardware and the need to be considerate becomes clear.

The good news is that once models like these are trained, we can use and adapt them for many purposes. Open-source models like Stability.ai's Stable Diffusion and Big Science BLOOM have had great environmental cost to produce, but once created, they can be distributed widely, and a lot of people can now fine-tune that model at much lower cost for their purposes. General purpose models like these means that we do not have to spend 500 tonnes of coal every time someone needs to utilise a highly advanced model. The original environmental cost of the model can be spread across many users over the following years.

It also means that we must generally recommend any organisation who undertakes large- scale training of

models to sincerely consider the opportunity to open-source their work to avoid unnecessary environmental damage.

## Don't use what you don't need

While we generally do not recommend using more advanced models than what is suited for the problem, an additional argument is the environmental impact. While it is preferable to fine-tune a large open-source model instead of training one from scratch for your purposes, if a simpler model like logistic regression or linear regression suffices, these will be several orders of magnitude more efficient and thereby also more environmentally friendly.

## Hardware matters

Fortunately for the environment, high energy consumption has been the enemy of computing ever since the beginning. The energy going into processors becomes heat that degrades performance. Larger, more modern chips are generally significantly more power efficient than older chips. While the total power consumption of cloud data centres is staggering, they are more efficient per calculation than what anyone can produce in a local setup (David Patterson, 2021). Scale matters in these settings, and the data centres can leverage the newest chips. Many of them even allow you to pick and choose locations with a high degree of renewable energy.

The production of new hardware carries its own emissions. If you are running a large compute operation, knowing when to decommission



hardware to balance your power budget has probably always been core to your skillset, but considering the environmental impact of the actual hardware might require some rethinking.

### **Monitor your usage**

An important step in sustainable data science is reporting on energy consumption. Several tools exist to improve this, but especially worth mentioning is the work from open-source project CarbonCode (CarbonCode, 2022). CarbonCode is a simple python-based utility which monitors python processes and reports on their energy consumption. It is also able to consider the geographic location of the process and optionally report it into a central registrar of data science processes to help everyone understand the carbon footprint of data science.

### **Different clouds**

If you are running your operations efficiently in the cloud, you might also have noticed that not all clouds are created equal from a sustainability point of view. The Big Science BLOOM model was trained on a French supercomputer running exclusively on nuclear power. Most cloud vendors label their different data centre regions according to their environmental friendliness, and there can be huge gains here.

**In summary,** sustainable data science is best achieved by not using more powerful compute tools than necessary, recycling models whenever possible through open-source initiatives, using modern and efficient hardware and, finally, monitoring the energy consumption of our processes through tools like CarbonCode.



# Data science for you

I hope that this guide has helped to provide an overview of the data science process – not just from data to model but from problem to solution. As I wrote in the beginning, data science is a combined discipline of both software development, machine learning and statistics and business/domain knowledge. None of the three parts must be forgotten. We hope that this guide can serve as a starting point both for your own data science journey and to help others create value for organisations through clever use of data.

Finally, here is a list of references to various resources mentioned in the guide. These are both articles and books and, perhaps more importantly, a comprehensive list of all the software tools that have been mentioned.

Should you have any thoughts, questions, wishes, corrections or anything else, do not hesitate to reach out! We would love to hear from you and help wherever we can.



## References

- Ajitsaria, A. (2018). Logging in Python. Retrieved from <https://realpython.com/python-logging/>
- Alec Radford, J. W. (2022). Introducing Whisper. Retrieved from OpenAI Research Blog: <https://openai.com/blog/whisper/>
- Ascombe, F. J. (1973). Graphs in Statistical Analysis. The American Statistician, 27:1, 17-21.
- Azure. (2020). Azure DevOps. Retrieved from <https://azure.microsoft.com/da-dk/services/devops/>
- CarbonCode. (2022). Retrieved from <https://codecarbon.io/>
- CircleCI. (2020). CircleCI. Retrieved from <https://circleci.com/>
- Corti.ai. (2020). Corti.ai. Retrieved from <https://www.corti.ai/>
- David Patterson, J. G.-M. (2021). Carbon Emissions and Large Neural Network Training. ArXiv.
- Delgan. (2020). Loguru. Retrieved from <https://github.com/Delgan/loguru>
- Enversion. (2020). KAUNT. Retrieved from <http://enversion.dk/okonomi/>
- Enversion. (2020). Sundhed. Retrieved from <http://enversion.dk/sundhed/>
- Esser, P., & Rombach, R. (2022). Stable Diffusion Launch Announcement. Retrieved from Stability.ai: <https://stability.ai/blog/stable-diffusion-announcement>
- Forum, W. E. (2016). weforum.org. Retrieved 2020, from <https://www.weforum.org/focus/fourth-industrial-revolution>
- Foundation, A. (2020). Anaconda. Retrieved from <https://www.anaconda.com/>
- Git. (2020). Git. Retrieved from <https://git-scm.com/>
- GitHub. (2020). GitHub. Retrieved from <https://github.com/>
- GitLab. (2020). GitLab. Retrieved from <https://about.gitlab.com/>
- Goodfellow, I. (2017). Adversarial Example Research. Retrieved from <https://openai.com/blog/adversarial-example-research/>
- Google. (2022). Facets. Retrieved from <https://pair-code.github.io/facets/>
- Grolemund, G., & Wickham, H. (2020). R for Data Science. Retrieved from <https://r4ds.had.co.nz/tidy-data.html>
- Hede, A. (2020, April). Implement Learning Institute, Data Science Master Class.
- ICO. (2020). Information Commissioners Office. Retrieved from <https://ico.org.uk/>
- Jenkins. (2020). Jenkins. Retrieved from <https://jenkins.io/>
- JetBrain. (2020). PyCharm. Retrieved from <https://www.jetbrains.com/pycharm/>
- Jupyter. (2022). Retrieved from <https://jupyter.org/>
- Knight, M. (2017). Dataversity: What is Data Governance? Retrieved from <https://www.dataversity.net/what-is-data-governance/>
- Krekel, H. (2020). PyTest. Retrieved from <https://docs.pytest.org/en/latest/>



- Lenyk, Z., & Park, J. (2023). Microsoft Vision Model ResNet-50 combines web-scale data and multi-task learning to achieve state of the art. Retrieved from Microsoft Research blog: <https://www.microsoft.com/en-us/research/blog/microsoft-vision-model-resnet-50-combines-web-scale-data-and-multi-task-learning-to-achieve-state-of-the-art/>
- Loukides, M. (2012). O'Reilly, What is DevOps? Retrieved from <http://radar.oreilly.com/2012/06/what-is-devops.html>
- Matejka, J., & Fitzmaurice, G. (2020). Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics through Simulated Annealing. Autodesk Research.
- Matplotlib. (2020). Matplotlib. Retrieved from <https://matplotlib.org/>
- Microsoft. (2020). PowerBI. Retrieved from <https://powerbi.microsoft.com/>
- Microsoft. (2020). Visual Studio Code. Retrieved from <https://code.visualstudio.com/>
- Microsoft. (2022). SandDance. Retrieved from <https://microsoft.github.io/SandDance/>
- Muhajlovic, I. (2020). Towards Data Science. Retrieved from <https://towardsdatascience.com/how-artificial-intelligence-is-impacting-our-everyday-lives-eae3b63379e1>
- MySQL. (2020). MySQL. Retrieved from <https://www.mysql.com/>
- O'Neil, C. (2016). Weapons of Math Destruction. Crown Books.
- PAIR\_Research. (2020). People + AI Guidebook. Retrieved from <https://pair.withgoogle.com/guidebook/>
- Pandas. (2020). Pandas. Retrieved from <https://pandas.pydata.org/>
- Profiling, P. (2020). Pandas profiling. Retrieved from <https://github.com/pandas-profiling/pandas-profiling>
- Project, R. (2020). The R Project for Statistical Computing. Retrieved from <https://www.r-project.org/>
- Python. (2020). Python.org. Retrieved from <https://www.python.org/>
- Robot. (2020). Robot. Retrieved from <https://robotframework.org/>
- scikit-learn. (2020). scikit-learn.org/stable. Retrieved from <https://scikit-learn.org/stable/>
- Seeing Potential, G. (2020). Seeing Potential. Retrieved from <https://about.google/stories/seeingpotential/>
- Spotify\_Labs. (2019). The Winding Road to Better Machine Learning Infrastructure Through Tensorflow Extended and Kubeflow. Retrieved from <https://labs.spotify.com/2019/12/13/the-winding-road-to-better-machine-learning-infrastructure-through-tensorflow-extended-and-kubeflow/>
- Synced. (2019). Humans Don't Realize How Biased They Are Until AI Reproduces the Same Bias, Says UNESCO AI Chair. Retrieved from <https://medium.com/syncedreview/humans-dont-realize-how-biased-they-are-until-ai-reproduces-the-same-bias-says-unesco-ai-chair-9968bb1f5da8>
- TensorFlow. (2020). TensorFlow Extended. Retrieved from <https://www.tensorflow.org/tfx>
- Tensorflow, G. (2020). Google Tensorflow. Retrieved from <https://projector.tensorflow.org/>
- Vincent, J. (2019). The Verge. Retrieved from <https://www.theverge.com/2019/6/27/18760896/deepfake-nude-ai-app-women-deepnude-non-consensual-pornography>
- W3Schools. (2020). SQL LEFT JOIN keyword. Retrieved from [https://www.w3schools.com/sql/sql\\_join\\_left.asp](https://www.w3schools.com/sql/sql_join_left.asp)
- Wattenberg, M., Viegas, F., & Ian, J. (2016). How to use t-SNE effectively. Distill.
- Wigner, E. P. (1960). The unreasonable effectiveness of mathematics in the natural sciences. Communications in Pure and Applied Mathematics.
- Wikipedia. (2020). Differential Privacy. Retrieved from [https://en.wikipedia.org/wiki/Differential\\_privacy](https://en.wikipedia.org/wiki/Differential_privacy)
- Wikipedia. (2020, April 21). Wikipedia: Data Science. Retrieved from Wikipedia: [https://en.wikipedia.org/wiki/Data\\_science](https://en.wikipedia.org/wiki/Data_science)
- Wikipedia: Data Science. (2022, April 21). Retrieved from Wikipedia: [https://en.wikipedia.org/wiki/Data\\_science](https://en.wikipedia.org/wiki/Data_science)



## Contact

For more information, please contact:

**Lead author Adam Hede**  
Implement Consulting Group  
+45 2929 9395  
adhe@implement.dk

**Nikolaj Skov Purup**  
+45 2935 0526  
nipu@implement.dk

**Marie Normann Gadeberg**  
+45 2074 3484  
maga@implement.dk

**Tobias Søndergaard**  
+45 2618 7793  
tobs@implement.dk

**Victor Emil Funch**  
+45 5221 6349  
vifu@implement.dk

**Jonas Vagn Jensen**  
+45 5217 7595  
jovj@implement.dk

**Mante Abisalaite**  
+45 9192 2862  
maab@implement.dk