

Topic 19-Conducting DS Interview,  
<https://www.youtube.com/watch?v=HROTqpBLQYI>

## CONDUCTING DATA SCIENCE INTERVIEWS

Objectives: You will learn:

- **How to conduct an efficient data science interview?**
- Explore how a hiring process can be tied to real working experience so that candidates you seek will clearly understand what kind of job they will need to do,
- Get rid of unnecessary interviewing rituals and make the hiring process more efficient.

- - - - -

- How long has it been since you went through a technical interview?
- Was it a pleasant experience?
- can you judge the relevance of your interview questions to your day-to-day work?



- We imagine that we are using a rational way of thought to pass candidates through a sieve, selecting only the stellar experts who will do the job.
- In fact, interviews are biased (Technical interviews are deeply flawed) There is almost no correlation between success in an interview and success at the actual job
- Do random candidate selection show better results than the hiring processes of many companies

We will cover the following topics:

- Common flaws of technical interviews
- Introducing values and ethics into the interview
- Designing good interviews

### Common flaws of technical interviews


- If you have a SE background, how often have interviewers asked you to reverse a binary tree on a whiteboard, or to find a maximum subarray sum?
- If you come from a DS background, how many central limit theorem proofs did you lay out on a piece of paper or a whiteboard?
- If you are a team leader, have you asked such questions yourself?

**It is not implying that those questions are bad, but quite the opposite!**

Knowledge of core CS algorithms and ability to derive proofs may be important for some jobs.

- But for what purpose do we ask those questions?
- What do we want to know about the person on the other side of the table?

For most companies, the ability to give answers to those questions is not relevant at all!

Then  What is the reason for asking them?

Well, because stereotypical programmers must know algorithms, and data scientists must know their mathematics and statistics; this way of thinking is logical and straightforward.

When we think in this way, we imagine a general collective image of a rock star programmer and a data science wizard.



**Reversing binary trees may be relevant for some projects in some company, but why is it relevant for yours?**

So, if you are open-minded, critical, and honest with yourself, this discussion and course content will not give you secret knowledge that will improve your interviews.

**It will give you tools.** If you invest in them, you'll get back more straightforward, simple, and enjoyable interviews that will find the right people for the right job.

## Searching for candidates you don't need

Most companies are looking for rock star developers, stellar data scientists, and consist purely of the best people in the entire world.



Let's look at a data scientist job description from a real job board.

The primary focus for a candidate will be on applying different techniques (data mining/statistical analysis/build prediction systems/recommendation systems) using large company datasets.

Apply machine learning models and test the effectiveness of different actions.

- The candidate must have strong technical expertise and be able to use a wide set of tools for data mining/data analysis methods.
- The candidate must be able to build and implement mathematical models, algorithms, and simulations.

### We seek candidates with the following skill set:

1. Work with business cases: seek opportunities and leverage company data to create business solutions.
2. Mine and analyse data from company databases to **drive optimization** and improvement of product development and sales techniques.
3. Assess the effectiveness and accuracy of new data sources and data gathering.
4. Extending the company's data with third-party sources of information when needed.
5. Use predictive modelling to **increase revenue generation, ad targeting**, and other business outcomes.
6. Analyze business cases & identify data sources (int | ext) and data mining/analysis methods to se.
7. Develop a normalization engine to execute cleansing/deduplication for a raw data through **extract transform load (ETL)** process for data sources.

Topic 20-Requirements of Ds interview and Scenario,  
<https://www.youtube.com/watch?v=jS78E9kjdok>



## Requirements of Data Science Interviews Scenario

### We seek candidates with the following skill set:

1. Work with business cases: seek opportunities and leverage company data to create business solutions.
2. Mine and analyse data from company databases to **drive optimization** and improvement of product development and sales techniques.
3. Assess the effectiveness and accuracy of new data sources and data gathering.
4. Extending the company's data with third-party sources of information when needed.
5. Use predictive modelling to **increase revenue generation, ad targeting**, and other business outcomes.
6. Analyze business cases & identify data sources (int | ext) and data mining/analysis methods to se.
7. Develop a normalization engine to execute cleansing/deduplication for a raw data through **extract transform load (ETL)** process for data sources.
8. Create, train, and test predictive models to solve defined business cases.
9. Develop algorithms to apply to data sets.
10. Design data structure models for collected data.
11. Facilitate the build of a solution from **proof of concept (POC) to production.**
12. Work with business owners to gather additional information about business cases.
13. Work with data that generated by core business.
14. Be ready to work in agile style (**daily, sprint planning, sprint review, retrospective**).
15. Work in an environment that adapts quickly to creative change using agile principles.
16. **Actively works with different development groups inside of organization.**
17. Be ready to adapt a new tool/library/technology/platform.
18. Excellent understanding of ML techniques and algorithms, such as **k-nearest neighbors**

**(kNNs), Naive Bayes, support vector machines (SVM),** decision tree, clustering, ANNs.

- 19.Strong understanding of math statistics (such as **distributions, statistical testing, regression,** and so on).
- 20.Experience creating and using advanced ML algorithms and statistical methods such as regression, simulation, scenario analysis, modeling, clustering, decision trees, NNs, and so on.
- 21.Proficiency in using query languages such as **SQL**.
- 22.Experience working with and creating data architectures, data models, DWs /data lakes.
- 23.Ability to work with minimal supervision.
- 24.Strong data analytical skills.
25. **Creative and analytical thinker** with strong problem-solving capabilities.
- 26.Background in technology or professional services preferably in one or more domains of AWS, Azure, Security, or AI/ML.
- 27.Strong understanding of **consulting business**.
- 28.Strong structural work methods, multitasking, and time management skills.
- 29.Self-driven independent work ethic that drives internal and external accountability.
- 30.Experience with common data science toolkits and libraries, such as pandas, Keras, SciPy, scikit-learn, TensorFlow, NumPy, MatLab, and other popular data mining technologies.
- 31.Experience using statistical computer languages (R, Python, and so on) to manipulate and draw insights from large data sets.
- 32.Knowledge and experience using SQL language.
- 33.Experience using Azure/AWS services.
34. Experience with C++/C# as a plus.
35. Based on this description, the candidate must know 4 to 5 programming languages to a level that allows them to start a POC and finish with a production ready system. More so, machine learning, deep learning, ETL, and business analysis skills are also a must. The candidate should be able to learn any new technology not present in the list. And he/she should be self-driven and independent.

If you think about it, this is an ideal candidate. A one-man army, a unicorn.

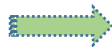
Do such people exist?



Yes, but they are extremely rare.

Does every company need people like this to complete their projects and achieve their goals?

The answer is : **with a bold no.**

 Instead of seeking the best fit for their goals, they seek the best fit for all possible goals.

Being general and all-encompassing, this job search will be long, tiresome, and stressful.

After reading this long list of requirements, the candidate won't be able to figure out what their job will be like.

All descriptions are vague and hard to understand. This description does not give a hint about what this job will be like.

The job description should give a clear listing of the following aspects:

- **The future responsibilities of the candidate**
- **Requirements for handling these responsibilities**

When the job description is vague and convoluted, testing all required skills will surely require several interview sessions, which will make the process even more tiring.

Adequate assessment of multiple candidates based on this list is close to impossible.

After a long search, the company will just hire someone randomly. After this, the troubles won't end.

The greatest problem of this job description is a lack of direction. The employer is not sure about the candidate's functions and responsibilities. This problem is deeper than it seems to be; the job description is actually only a symptom, not the illness itself.



Topic 1-Introduction to managing DS Projects and Team building,  
<https://www.youtube.com/watch?v=vsgJOZIFJaY>

(Professional Elective-II)

**Dr. Ajeet K Jain**

*Associate Professor, CSE, KMIT , Hyderabad*

**Objectives: The course will help to learn:**

1. How to build and manage a data science team
2. How to handle various data science projects in an organization
3. About minimum viable products for data science and creating (developing) infrastructure for
4. data science projects
5. Technology for securing data science environment
6. How to involve data science in cyber security

**Outcomes:** After learning the concepts , you will be able to understand :

1. Importance of building and managing data science teams.
2. How to handle data science projects in organization.
3. Developing infrastructure for data science projects.
4. Prominence of cyber security for data science

**TEXT BOOKS:**

1. **Managing Data Science:** Effective strategies to manage data science projects and build a sustainable team, Kirill Dubovikov, Packt Publishing Ltd, 2019



2. **SECURE DATA SCIENCE: Integrating Cyber Security and Data Science**, Bhavani Thuraisingham, Murat Kantarcioglu, and Latifur Khan, CRC Press, 2022



#### REFERENCE BOOKS:

1. CYBER-RISK INFORMATICS: Engineering Evaluation with Data Science, MEHMET SAHINOGLU, John Wiley & Sons, Inc, 2016
2. Think Like a Data Scientist, BRIAN GODSEY, MANNING, 2017

#### **Building and Sustaining a Team**

- Defining data science, The influence of data science, Limitations of data science, Review of Machine Learning and Deep Learning, Review of Offline model testing, Online model testing.
- **Building and Sustaining a Team:** Defining data science team roles, Exploring data science team roles and their responsibilities, Common flaws of technical interviews, Introducing values and ethics into the interview, Designing good interviews, Achieving team Zen, Leadership and people management, Facilitating a growth mindset, Case study—creating a data science department.

DS and ML can transform any organization and open new opportunities.

Any DS project is a unique mix of research, s/w engg., and business expertise.

A substantial managerial effort is needed to guide the solution from prototype development to production.

Traditional approaches often fail as they have different conditions and requirements in mind.

We shall learn about proven approaches to DS project management, with tips and best practices to

guide us along the way.

The process of DS begins with preparation you need to establish:

- what you know,
- what you have,
- what you can get,
- where you are, and
- where you would like to be.

This last one is of utmost importance; a project in DS needs to have a purpose and corresponding goals. Only when you have well defined goals can you begin to survey the available resources and all the possibilities for moving toward those goals.

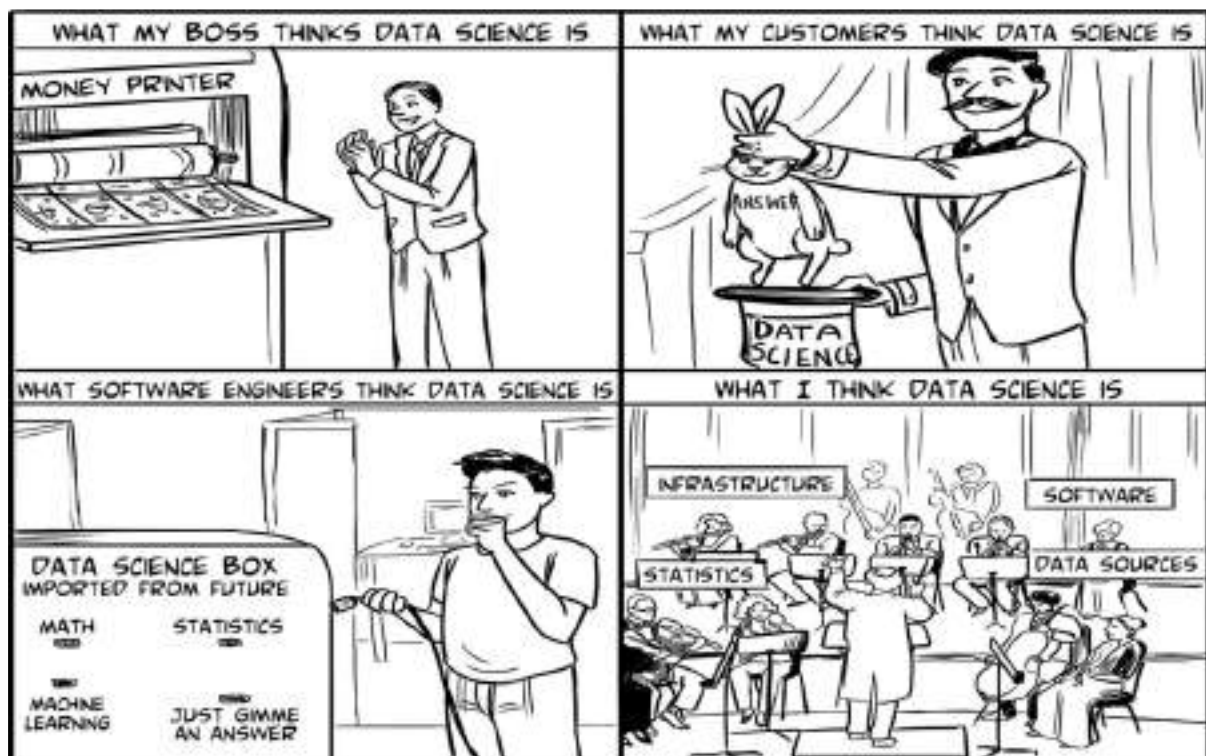


Fig. 1.1 Perspective view of DS : what is it?

## 1.1 What is Data Science

Before diving into the management issues of building systems around machine learning algorithms, we need to explore the topic of data science itself.

- What are the main concepts behind data science and machine learning?
- How do you build and test a model?
- What are the common pitfalls in this process?
- What kinds of models are there?
- What tasks can we solve using machine learning?



#### What You Can Do with Data Science : KNOW the DOMAIN

Before building a house, it would be better to know how a hammer works. Having basic knowledge of the domain you manage is vital for any kind of manager.

- A software development manager needs to understand computer programming.
- A factory manager needs to know the manufacturing processes.
- A data science manager is no exception.

DS has become popular, and many business people and technical professionals have an increasing interest in understanding data science and applying it to solve their problems. People often form their first opinions about DS from the information that they collect through the background: **news sites, social networks**, and so on.

Unfortunately, most of those sources misguide, rather than give a realistic picture of DS and ML.

Instead of explaining, the (social) media describes the ultimate magical tools that easily solve all our problems :

- ❖ The technological singularity is near.
- ❖ A universal income economy is around the corner.

Well, only if machines learned and thought like humans.

In fact, we are far from creating general-purpose, self-learning, and self-improving algorithms.

This session explores current possibilities and modern applications of the main tools of data science: machine learning and deep learning.

We will cover the following topics:

- Defining AI
- Introduction to machine learning
- Introduction to deep learning
- Deep learning use case
- Introduction to causal inference



## Defining A I

Media and news use AI as a substitute buzzword for any technology related to data analysis. In fact, AI is a sub-field of computer science and mathematics. It all started in the 1950s, when several researchers started asking whether computers can **learn**, **think**, and **reason**. 74 years later, we still do not know the answer.

However, we have made significant progress in a specific kind of AI that solves thoroughly specified narrow tasks: weak AI.

Science fiction novels tell about machines that can reason and think like humans. In scientific language, they are described as strong AI. Strong AI can think like a human, and its intellectual abilities may be much more advanced. The creation of strong AI remains the main long-term dream of the scientific community. However, practical applications are all about weak AI. While strong AI tries to solve the problem of general intelligence, weak AI is focused on solving one narrow cognition task, such as vision, speech, or listening. Examples of weak AI tasks are diverse: speech recognition, image classification, and customer churn prediction. Weak AI plays an important role in our lives, changing the way we work, think, and live. We can find successful applications of weak AI in every area of our lives.

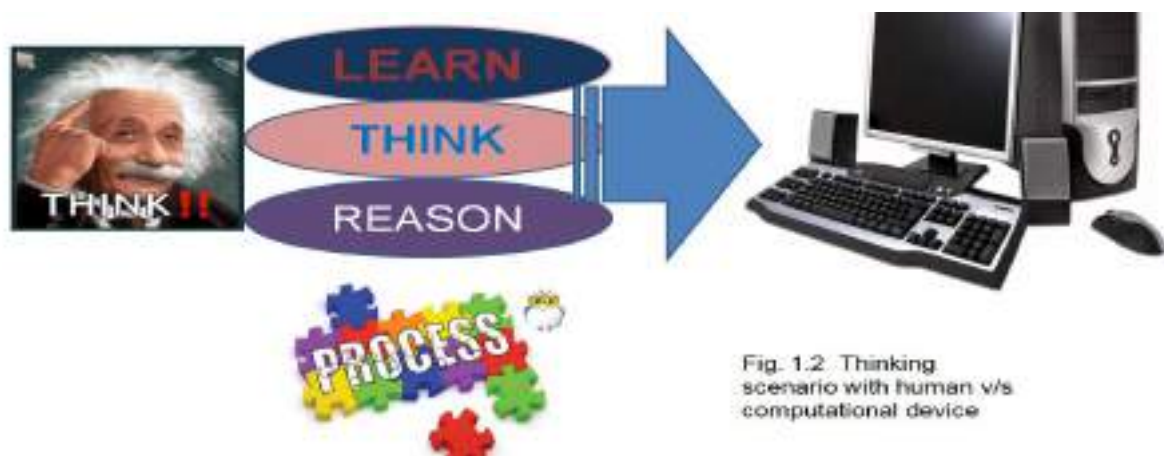


Fig. 1.2: Thinking scenario with human v/s computational device

### 1.3 Defining Data Science (DS)

- *How does AI relate to ML?*
- *What is deep learning (DL)?*
- *How do we define data science?*

These popular questions are better answered graphically

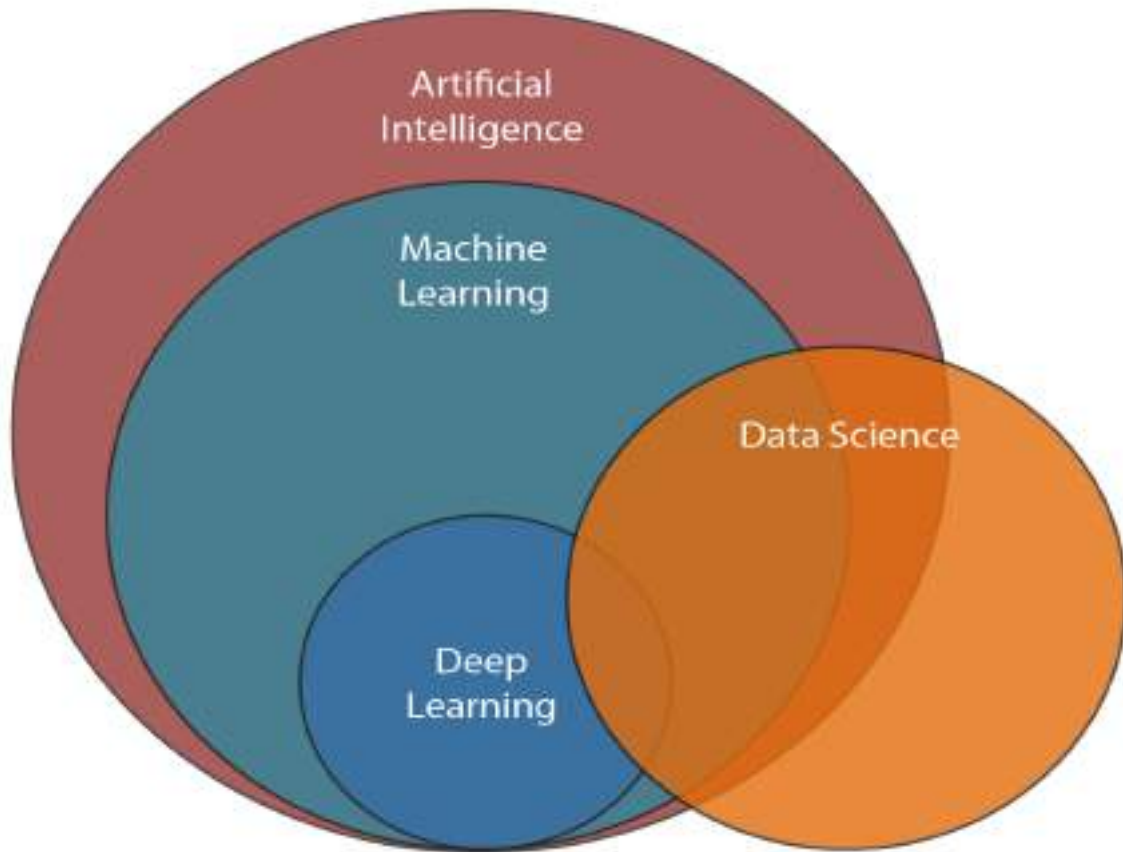


Fig. 1.2 DS : intermixing view

AI is a general scientific field that covers everything related to weak and strong AI.

Machine learning is a subfield of AI that studies algorithms that can adapt their behavior based on incoming data without explicit instructions from a programmer.

Deep learning is a subfield of machine learning that studies a specific kind of machine learning model called deep neural networks.

Data science is a multidisciplinary field that uses a set of tools to extract knowledge from data and support decision making. Machine learning and deep learning are among the main tools of data science.

The ultimate goal of DS is to solve problems by extracting knowledge from data and giving support for complex decisions.

The first part of solving a problem is getting a good understanding of its domain. You need to:

- ***understand the insurance business before using DS for risk analysis.***
- ***know the details of the goods manufacturing process before designing an automated quality assurance process.***



TAKE AWAY : You understand the domain. Then, you find a problem.
--

If you skip this part, you have a good chance of solving the **wrong problem**.

After coming up with a good problem definition, you seek a solution.

Suppose that you have created a model that solves a task. A ML model in a vacuum is rarely interesting for anyone. So, it is not useful.

To make it useful, we need to wrap our models into something that can be seen and acted upon.

**In other words, we need to create software around models.**

Data science always comes hand-in-hand with creating software systems.

Any ML (DL) model needs s/w . Without this, models would just lie in computer memory, not helping anyone. So, DS is never only about science.

Business knowledge and s/w development are also important. Without them, no solution would be complete.

Topic 2-DS Concepts and Social Effects,  
[https://www.youtube.com/watch?v=sFJlpLT6j\\_I](https://www.youtube.com/watch?v=sFJlpLT6j_I)

## MDSS : Video Session # 2

Defining Data Science (DS)

- *How does AI relate to ML?*
- *What is deep learning (DL)?*
- *How do we define data science?*

These popular questions are better answered graphically:

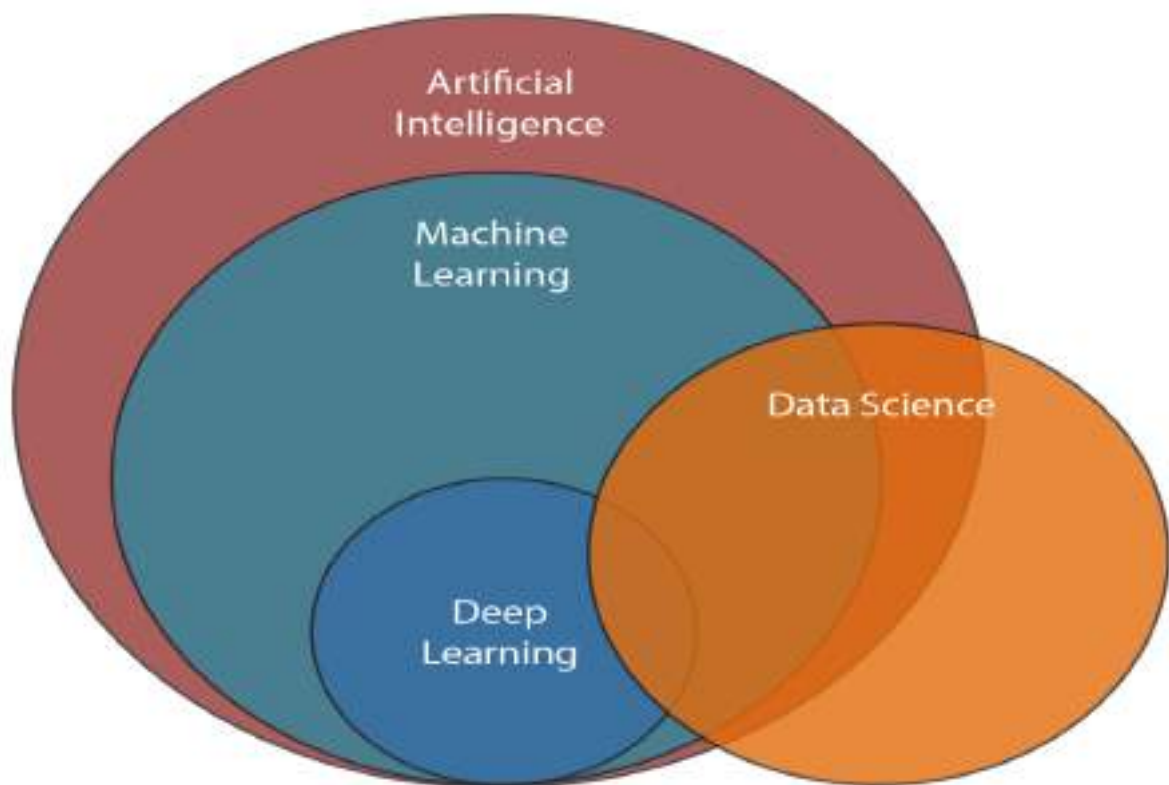


Fig. 1.2 DS : intermixing view

AI is a general scientific field that covers everything related to weak and strong AI.

Machine learning is a subfield of AI that studies algorithms that can adapt their behavior based on incoming data without explicit instructions from a programmer.

Deep learning is a subfield of machine learning that studies a specific kind of machine learning model called deep neural networks.

Data science is a multidisciplinary field that uses a set of tools to extract knowledge from data and support decision making. Machine learning and deep learning are among the main tools of data science.

The ultimate goal of DS is to solve problems by extracting knowledge from data and giving support for complex decisions.

The first part of solving a problem is getting a good understanding of its domain. You need to: understand the insurance business before using DS for risk analysis.

Know the details of the goods manufacturing process before designing an automated quality assurance process.

TAKE : You understand the domain.

AWAY Then, you find a problem.

If you skip this part, you have a good chance of solving the wrong problem.

After coming up with a good problem definition, you seek a solution.

Suppose that you have created a model that solves a task. A ML model in a vacuum is rarely interesting for anyone. So, it is not useful.

To make it useful, we need to wrap our models into something that can be seen and acted upon.

**In other words, we need to create software around models.**

Data science always come hand-in-hand with creating software systems.

Any ML (DL) model needs s/w . Without this, models would just lie in computer memory, not helping anyone. So, DS is never only about science.

Business knowledge and s/w development are also important. Without them, no solution would be complete.

### **1.3 The influence of DS**

DS has huge potential and affects our daily lives.

Healthcare companies are learning to diagnose and predict major health issues. Businesses use it to find new strategies for winning new customers and personalize their services.

Recently formed new Telengana Govt. has given FREE travel in RTC buses to women passengers across entire state.

We use big data analysis in genetics and particle physics.

Thanks to advances in DS, **self-driving cars are now a reality.**

Thanks to the internet and global computerization, we create vast amounts of data daily.

Ever-increasing volumes of data allow us to automate human labor

Sadly, for each use case that improves our lives, we can easily find two that make them worse.

To give you a disturbing example, let's look at China. The Chinese government is experimenting with a new **social credit system**. It uses surveillance cameras to track the daily lives of its citizens on a grand scale. Computer vision systems can recognize and log every action that you make while commuting to work, waiting in lines at a government office, or going home after a party.

A special social score is then calculated based on your monitored actions. This score affects the lives of real people. In particular, public transport fees can change depending on your score; low scores can prohibit you from interviewing for a range of government jobs.

**Another disturbing example:**

Women passengers are occupied the male passenger seats and fighting among themselves in order to grab the seat and fast and easy travel.

This Free-bee is actually a social-cum-political manifesto to win the election as a strategy, but have major ever lasting effects on the travel pattern and crowd management and busses run.

**End of Session # 2**

### Topic 3- Limitations of DS,

<https://www.youtube.com/watch?v=rYtx8QOpBAQ>

## MDSS: Session 3

### 1.3 The influence of DS ( contd...)

Another disturbing example:

Women passengers are occupied the male passenger seats and fighting among themselves in order to grab the seat and fast and easy travel.

This Free-bee is actually a social-cum-political manifesto to win the election as a strategy, but have major ever lasting effects on the travel pattern and crowd management and busses run.

On the other hand, this same technology can be used to help people. For example, it can be used to track criminals in large crowds. The way you apply this new technology can bring the world closer to George Orwell's 1984, or make it a safer place. The general public must be more conscious of these choices, as they might have lasting effects on their lives.

Another example of some disturbing uses of machine learning is businesses that use hiring algorithms based on ML. Months later, they discovered that the algorithms introduced bias against women. It is becoming clear that we do not give the right amount of attention to the ethics of data science.

While companies such as Google create internal ethics boards, there is still no govt. control over the unethical use of modern technology. Before such programs arrive, it is strongly encouraged to consider the ethical implications of using DS. We all want a better world to live in. Our future, (future of our children), depends on small decisions we make every day.

### 1.4 Limitations of DS

Like any set of tools, DS has limitations too! Before diving into a project with ambitious ideas, it is important to consider the current limits of possibility. A task that seems easily solvable may be unsolvable in practice. Insufficient understanding of the technical side of data science can lead to serious problems in your projects.

You can start a project only to discover that you cannot solve the task at all.

Even worse, you can find out that nothing works as intended only after deployment.

Depending on your use case, it can affect real people.

Understanding the main principles behind DS will rid you of many technical risks that predetermine a project's fate before it has even started

## 2.0 Introduction to ML

ML is by far the most important tool of a data scientist.

It allows us to create algorithms that discover patterns in data with thousands of variables.

### Different types and capabilities of ML algorithms.

ML is a scientific field that studies algorithms that can learn to perform tasks without specific instructions, relying on patterns discovered in data. Eg., we can use algorithms to predict the likelihood of having a disease or assess the risk of failure in complex manufacturing equipment.

Every machine learning algorithm follows a simple formula. In Fig. 1.3, you can see a high-level decision process that is based on a machine learning algorithm. Each machine learning model consumes data to produce information that can support human decisions or fully automate them.

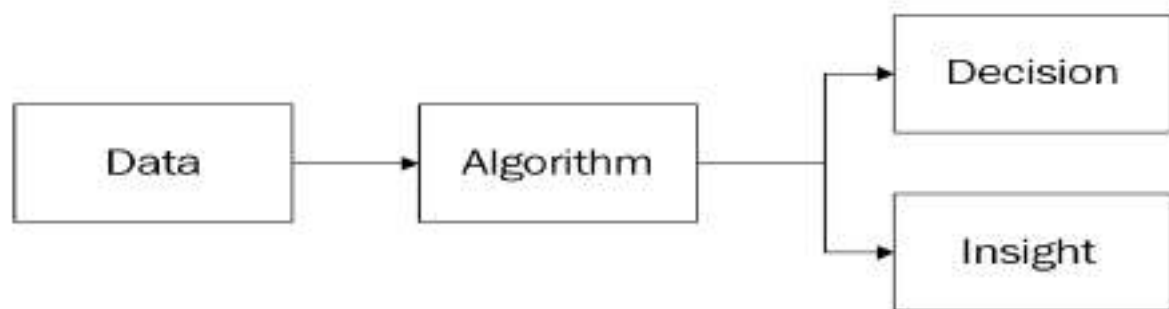


Fig. 1.3 ML Decision making process



## 2.1 Decisions and insights provided by a ML model

When solving a task using machine learning, you generally want to automate a decision making process or get insights to support your decision. For example, you may want an algorithm to output a list of possible diseases, given a patient's medical history and current condition. If machine learning solves your task completely, or end to end, this means that the algorithm's output can be used to make a final decision without further thinking; in our example, determining the disease the patient is suffering from and prescribing suitable medication automatically. The execution of this decision can be manual or automatic. We say that machine learning applications like these are end to end. They provide a complete solution to the task.

Let's look at digital advertising as an example. An algorithm can predict whether you will click on an ad. If our goal is to maximize clicks, we can make automated and personalized decisions about which specific advertisement to show to each user, solving the click-through rate maximization problem end to end.



Another option is to create an algorithm that provides you with an insight. You can use this insight as part of a decision-making process. This way, the outputs of many machine

learning algorithms can take part in complex decision making. To illustrate this, we'll look at a warehouse security surveillance system. It monitors all surveillance cameras and identifies employees from the video feed. If the system does not recognize a person as an employee, it raises an alert.



This setup uses two machine learning models: face detection and face recognition. At first, the face detection model searches for faces in each frame of the video. Next, the face recognition model identifies a person as an employee by searching the face database. Each model does not solve the employee identification task alone. Yet each model provides an insight that is a part of the decision-making process.



Topic 4-Datatypes,

<https://www.youtube.com/watch?v=EskZoKVcj-4>

## MDSS: Session 4

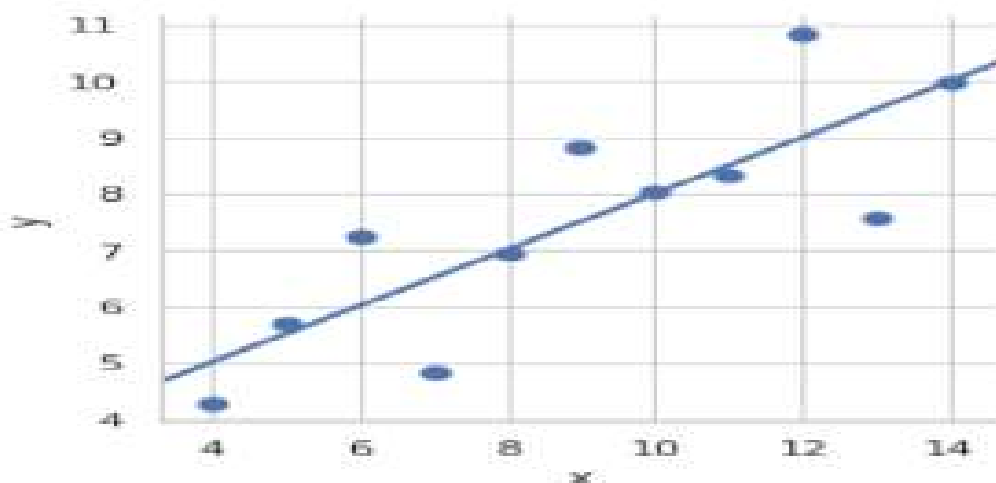
### Data for machine learning models

You may have noticed that algorithms in our examples work with different data types. In the digital advertising example, we have used structured customer data. The surveillance example used video feeds from cameras. In fact, machine learning algorithms can work with different data types.

We can divide the entire world's data into two categories: structured and unstructured. Most data is unstructured. Images, audio recordings, documents, books, and articles all represent unstructured data. Unstructured data is a natural byproduct of our lives - Smartphones and social networks facilitate the creation of endless data streams. Nowadays, you need little to snap a photo or make a video. Analyzing unstructured data is so difficult that we didn't come up with a practical solution until 2010.

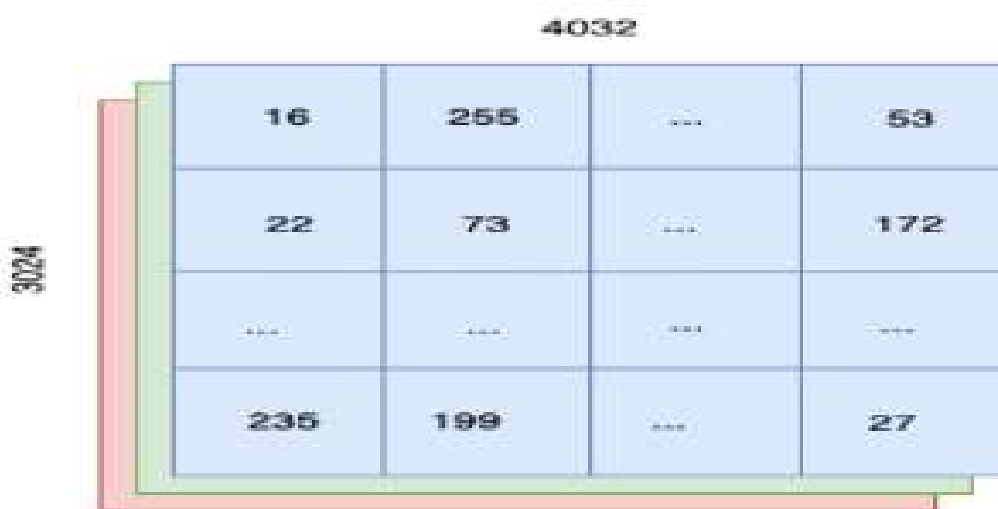
Structured data is hard to gather and maintain, but it is the easiest to analyze. The reason is that we often collect it for this exact purpose. Structured data is typically stored inside computer databases and files. In digital advertising, ad networks apply huge effort to collect as much data as possible. Data is gold for advertising companies. They collect your browsing history, link clicks, time spent on site pages, and many other features for each user. Vast amounts of data allow the creation of accurate click probability models that can personalize your ad browsing experience. Personalization increases click a probability, which increases advertisers' profits.

Each value in this dataset is displayed on the following plot:



As you can see, we can have an accurate guess of what the value of **y** will be given a value of **x**. To do this, we can look at the corresponding point on the line. For example, for **x = 10**, **y** will be equal to **8**, which matches the real data points depicted as blue dots.

Now, let's shift to a photo made with an iPhone. This image will have a resolution of 4,032 x 3,024 pixels. Each pixel will have three color channels: red, green, and blue. If we represent pixels in this image as numbers, we will end up with over twelve million of them in each photo. In other words, our data has a dimensionality of 12 million.



Using machine learning algorithms on high-dimensional data can become problematic. Many machine learning algorithms suffer from a problem called the curse of dimensionality. To create a good model that recognizes objects in a photo, you need a model that's much more complex than a simple line. The complexity of the model increases the **data hunger** of the model, so models that work well on unstructured data usually require vast amounts of training data samples.

Topic 5-ML Models & Prediction,  
<https://www.youtube.com/watch?v=t-89rl8kXGU>

## MDSS: Session 5

### Anatomy of ML

Most common use of ML principles are in prediction domain. Predictive algorithms tell us when something will happen, but not necessarily why it will happen. Some examples of prediction tasks are:

- *Will this user churn in the next month?*
- *Does this person have a high risk of developing Alzheimer's disease?*
- *Will there be a traffic jam in the next hour?*

Often, we want to have explanations instead of predictions.

*Solving an inference task means to find supporting evidence for some claim in the data by asking Why? questions.*

- *Why did this person win the election?*
- *Why do this medicine work and the others don't?*

Statistical inference helps us to find an explanation or to prove the efficiency of our actions.

When we do inference, we seek answers for the present or the past.

When we try to look into the future, prediction comes into play.

Sometimes, we are not interested in predicting the future or finding evidence. We want machines to recognize complex patterns in data, such as objects in an image, or to analyze the sentiment of a text message. This group of tasks is called recognition.

ML covers many types and flavors of models. But why do we need such variety of different algorithms?

The reason lies in a theorem called the No Free Lunch Theorem.

It states that there is no best model that will consistently give you the best results on each task for every dataset. Each algorithm has its own benefits and pitfalls. It may work flawlessly on one task, but fail miserably at another.

**IMP:** One of the most important goals of a data scientist is to find the best model to solve the problem at hand.

## 2.4 Main types of tasks you can solve with ML

The no free lunch theorem states that there is no best model that solves all tasks well. The consequence of this is that we have many algorithms that specialize in solving specific tasks

**Example # 1** For instance, let's look at fashion retail warehouse demand forecasting. A retailer sells a fixed set of clothes at their stores. Before an item makes it to the shelves, it must be bought from the manufacturer and transferred to a warehouse. Let's assume that their logistics cycle takes two weeks. How do we know the best quantity of each item to order? There is a good chance that we have item sales data for each store.



We can use it to create a predictive model that estimates average customer demand for each item in the catalogue of our warehouse over the next two weeks.

That is, we forecast an average number of to-be bought items over the next two weeks.

End of Session # 5



Topic 6-Regression and Prediction Model,  
<https://www.youtube.com/watch?v=9wTaYA-xcN8>

## MDSS: Session # 6

### Regression and Prediction model

#### Example # 1 ( contd...)

The simplest model would be to take an average demand for each item from the last two weeks as an estimate of average future demand.

Simple statistical models like this are frequently used at real retail stores, so we do not oversimplify.

To be more general, let's call the thing we want to forecast the target variable.

In this case, the target variable is the demand.

To build a forecasting model, we will use two previous weeks of historical data to calculate arithmetic averages for each item. We then use those averages as estimates for the future values. In a way, historical data was used to teach our model about how it should predict the target variable.

When the model learns to perform a given task using input/output examples, this process is supervised learning.



Supervised learning is not limited to simple models.

- ❖ In general, the more complex your model is, more data it requires to train.
- ❖ More training data you have, better your model will be !

## Example # 2 World of Information Security

Our imaginary client is a large telecoms company. Over the years, they have experienced many security breaches in their network. Thankfully, specialists have recorded and thoroughly investigated all fraudulent activity on the network.

**Security experts labelled each fraudulent activity in network logs.**



Having lots of data with labelled training examples, we can train a supervised learning model to distinguish between normal and fraudulent activity. This model will recognize suspicious behaviour from vast amounts of incoming data. However, this will only be the case if experts labeled the data correctly. Our model won't correct them if they didn't. This principle is called garbage in, garbage out. Your model can only be as good as your data.

Both the retail and security examples use supervised learning, but let's look at the target variables more closely.

1. The forecasting algorithm used demand as the target variable.

Demand is a continuous number ranging from 0 to  $\infty$  (infinity).

2. On the other hand, the security model has a fixed number of outcomes.

Network activity is either normal or fraudulent. We call the first type of the target variable—continuous, and the second type—categorical.

The target variable type strongly indicates which kind of task we can solve. Prediction tasks with continuous target variables are called regression problems.

And when the total number of outcomes is limited, we say that we solve a classification problem. Classification models assign data points to categories, while regression models estimate quantities.

Here are some examples:

House price estimation is a regression task | Predicting user ad clicks is a classification task.

Predicting HDD utilization in a cloud storage service is a regression task.

Identifying the risk of credit default is a classification task.

We can use it to create a predictive model that estimates average customer demand for each item in the catalogue of our warehouse over the next two weeks.

That is, we forecast an average number of to-be bought items over the next two weeks.

You can consider yourself lucky if you have found a good labelled dataset. You're even luckier if the dataset is large, contains no missing labels, and is a good match for an end-to-end solution to a problem. The labels we use for supervised learning are a scarce resource. A total absence of labels is a lot more common than fully labelled datasets.

This means that often, we cannot use supervised learning. But the absence of labels does not mean that we are doomed! One solution is to label data by hand.

You can assign this task to your employees if the data cannot be shared outside of the company.

Otherwise, a much simpler and faster solution is to use crowd funding services such as Amazon Mechanical Turk.

There, you can outsource data labelling to a large number of people, paying a small fee for each data point.

### ML to Games.

If we take a single game, we may label some data and use supervised learning. But scaling this approach for all games is not possible in practice. On a typical gaming console, you use the same controller to play different games. Try to recall when you played for the first time in your life. It is

likely that you were unsure of what to do. You might have tried pressing a few buttons and looked at a jumping character. Piece by piece, you must have figured out the rules of the game and started playing. I wouldn't be surprised if you felt confident playing and could finish the first levels after a few hours of experience.



Using the knowledge we have gained so far, can we design a machine learning algorithm that will learn how to play games? It might be tempting to use supervised learning, but think first. You had no training data when you took the controller for the first time in your life. But can we create algorithms that would figure out game rules by themselves?

End of Session #6

Topic 7-NLP,

<https://www.youtube.com/watch?v=Td3WZQ3czCM>

## MDSS: Session # 7

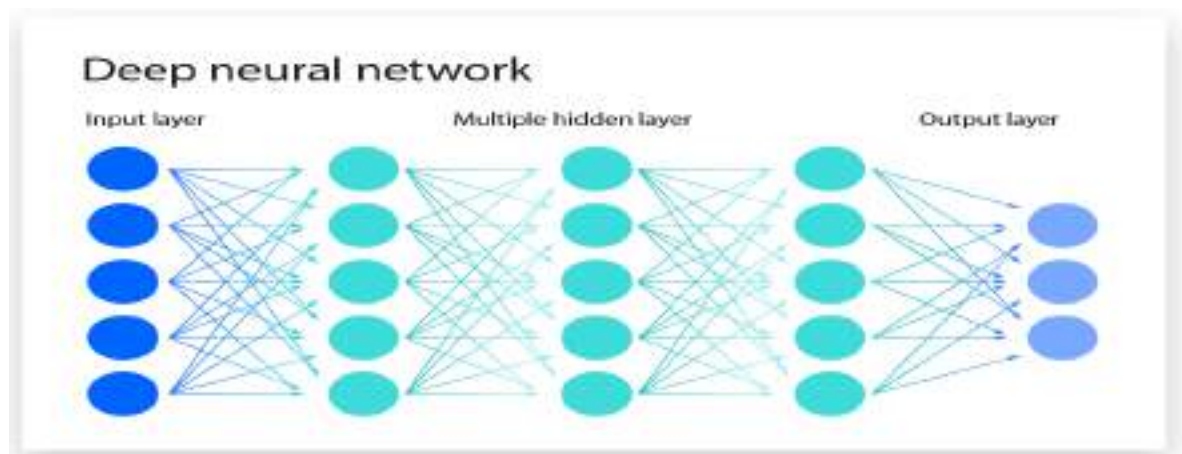
### Natural Language Processing

#### Introduction to Deep Learning (DL)

DL deals with study and implementation of a specific set of models using neural networks (NN).

Training big NNs uses a great amount of computation power. But not any computation power will suffice as NNs do a lot of matrix operations involved.

Strangely, rendering computer graphics also involves many matrix operations, so many, in fact, that each computer has a dedicated circuit inside: a GPU.



Nvidia has developed a special programming framework **CUDA** allowing you to harness the power of your GPU not only for computer graphics, but for general computing tasks as well. GPUs can do insane amounts of parallel matrix operations. Modern graphics cards have thousands of cores, so you can perform thousands of operations in parallel. And all individual operations also work quickly. Modern GPUs can execute thousands of parallel, floating-point computations. GPUs specialize in solving one specific task much faster than general-purpose CPUs

DL is fantastic at solving tasks with unstructured datasets. To illustrate this, let's look at a ML competition called **ImageNet**. It contains over **14 million images**, classified into **22,000** distinct categories. To solve ImageNet, an algorithm should learn to identify an object in the photo. While human performance on this task is around 95% accuracy, the best neural network model surpassed this level in 2015.





### Diving into natural language processing (NLP)

We write every day, whether it is documents, tweets, electronic messages, books, or emails. The list can go on and on. Using algorithms to understand natural language is difficult because our language is ambiguous, complex, and contains many exceptions and corner cases. The first attempts at **natural language processing (NLP)** were about building rule based systems. Linguists carefully designed hundreds and thousands of rules to perform seemingly simple tasks, such as part of speech tagging.





The following are a few of the NLP steps:

- **Text preprocessing:** Text preprocessing entails cleaning up the text and getting it ready for additional processing. This could entail actions like eliminating stop words, stemming words, and punctuation.
- **Tokenization:** Tokenization is the process of dividing a text into separate words or phrases.
- **Part-of-speech tagging:** Tagging each word in the text with its part of speech, such as a noun, verb, adjective, or adverb, is known as part-of-speech tagging.
- **Named entity recognition:** Named entity recognition entails locating and categorizing named objects in the text, including persons, places, businesses, and items.
- **Semantic analysis:** This means understanding the text's meaning. This could entail activities like figuring out how words and phrases relate to one another and figuring out the tone of the text.
- **Text generation:** Text creation is the process of creating new text, such as summaries, translations, or chatbot responses.

NNs also do a good job of solving comprehension problems:

#### **Question Answering and Text Summarization.**

In Q & A, the model gets a chunk of text and a question about its contents.

E.G., given the introduction to this section, *Introduction to deep learning*, a question-answering model could correctly answer the following queries:

*How many cores do modern desktop CPUs have? (less than 10)*

*When did neural networks originate? (1980s)*

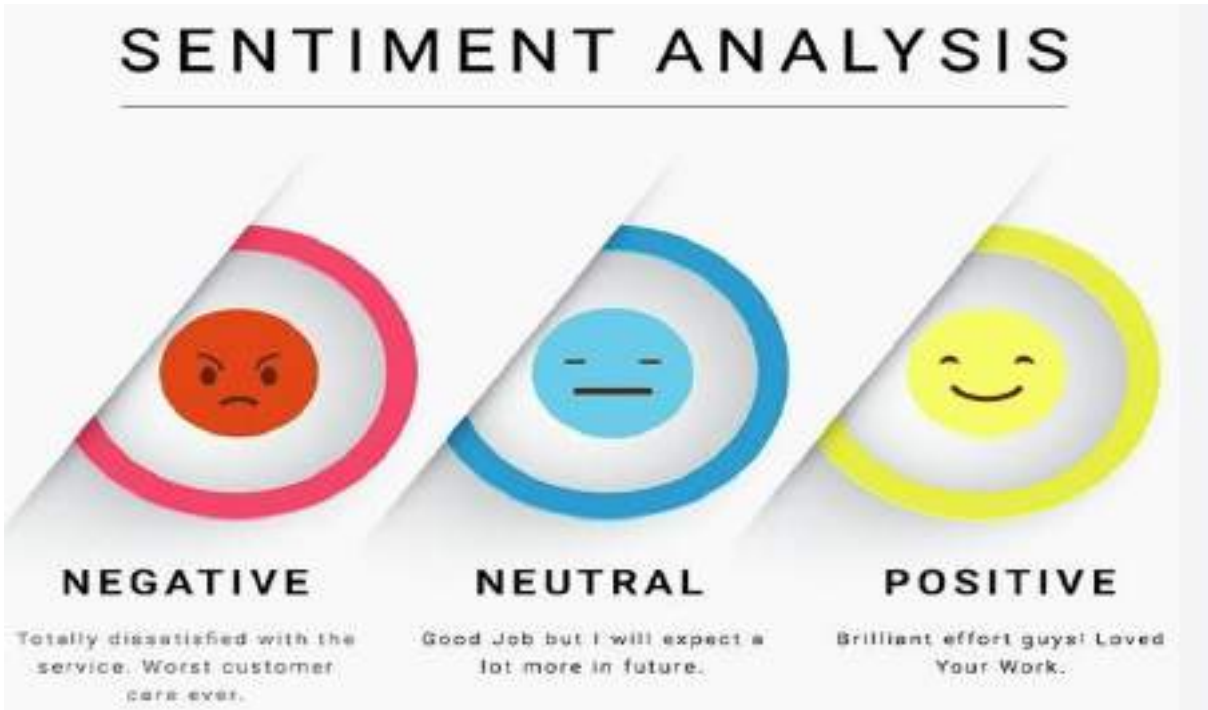
Text summarization seeks to extract the main points from the source. If we feed the first few paragraphs of this section into a text summarization model, we will get the following results:

*Now it is time to draw a line between ML and DL. In truth, we can't do this, because deep learning is a sub-field of machine learning. Formally, deep learning studies a specific set of models called neural networks. The theory behind neural networks originated in the 1980s. You can try out a sample text summarization model online at <http://textsummarization.net/>.*

Another intersecting NLP problem is text classification. By labeling many texts as emotionally positive or negative, we can create a sentiment analysis model.

We can train this kind of model using supervised learning. Sentiment analysis models can give powerful insights when used to measure reactions to news or the general mood around Twitter hashtags.

Text classification is also used to solve automated email and **document tagging**. We can use NNs to process large chunks of emails and to assign appropriate tags to them.



Top-level Category	Sub-Category				
Year	2019	2020			
Quarter	Q1	Q2	Q3	Q4	
Customer Journey Stage	Discovery	Learn	Try	Buy	Adapt, Advocate
Campaign	Back to school	Halloween	Black Friday	Holidays	
Channel	Email	Social	SEO	Print	Pod
Persons	Buyer	User			
Workflow	Draft	In Progress	Approved	Final	
Team	Creative	Lead Gen	Retention	Marketing	Sales

End of Session 7

Topic 8-NLP Chatbots & Computer Vision,  
<https://www.youtube.com/watch?v=qGFji9sOKII>

## MDSS: Session # 8

### NLP Applications

The pinnacle of practical NLP is the creation of dialog systems, or chatbots. Chatbots can be used to automate common scenarios at IT support departments and call centres. However, creating a bot that can reliably and consistently solve its task is not an easy task. Clients tend to communicate with bots in rather unexpected ways, so you will have a lot of corner cases to cover. NLP research is not quite at the point of providing an end-to-end conversational model that can solve the task.



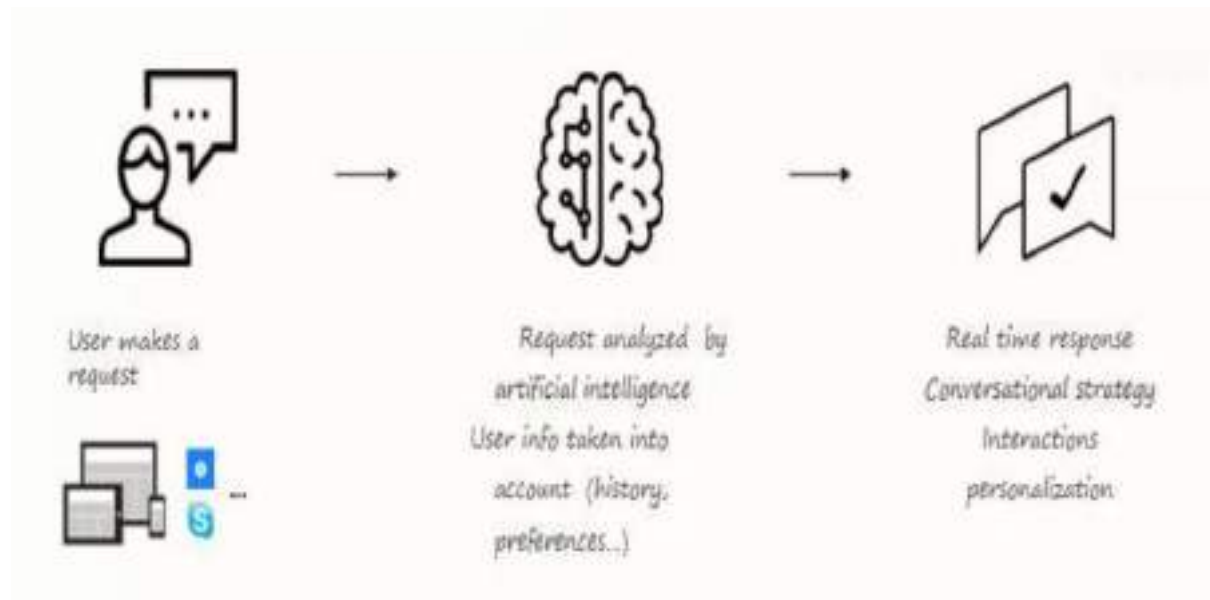
General chatbots use several models to understand user requests and prepare an answer:



The intent classification model determines the user's request.

The entity recognition model extracts all named entities from the user's message. The response generator model takes the input from the intent classification and entity recognition models and generates a response. The response generator can also use a knowledge database to look up extra

information to enrich the response. Sometimes, the response generator creates several responses. Then, a separate response ranking model selects the most appropriate one



NLP models can also generate texts of arbitrary length based on initial input. State-of-the-art models output results that are arguably indistinguishable from human-written texts



You can look at the model output samples in an OpenAI blog post about the GPT-2 model: <https://openai.com/blog/better-language-models/#sample1>.

While the results are very compelling, we are yet to find useful and practical applications for text generation models. While writers and content creators could potentially benefit from these models by using them to expand a list of key points into a coherent text, unfortunately, these models lack the means to control their output, making them difficult to use in practice.

## Exploring Computer Vision

In 2010, the first ImageNet Large Scale Visual Recognition Challenge was held. The task was to create a classification model that solved the ambitious task of recognizing an object in an image. In total, there are around 22,000 categories to choose from. The dataset contains over 14 million labeled images. If someone sat and chose the top 5 objects for each image, they would have an error rate of around 5%.



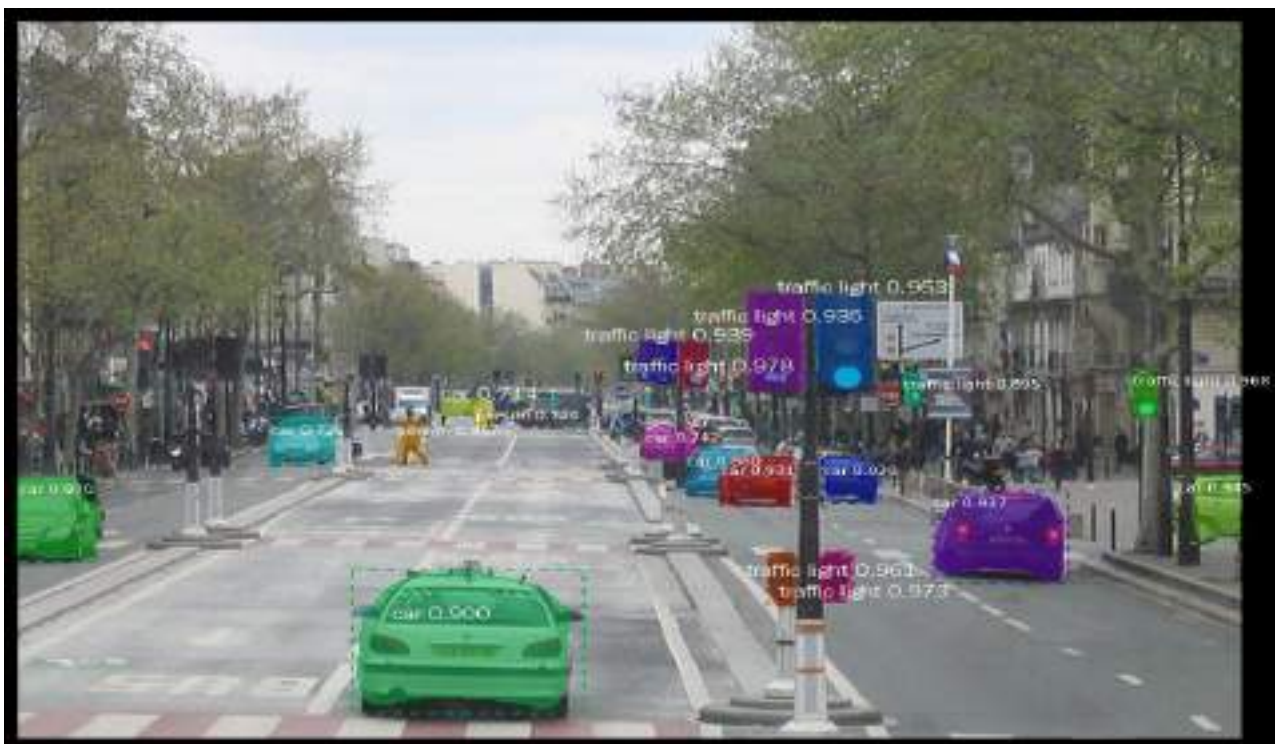
In 2015, a DNN surpassed human performance on ImageNet. Since then, many computer vision algorithms have been rendered obsolete. Deep learning allows us not only to classify images, but also to do object detection and instance segmentation.

The following two pictures help to describe the difference between object detection and instance segmentation:





Object detection models recognize objects and place bounding boxes around them.



Instance segmentation models find the exact outlines of objects.

Thus, the main practical uses of deep learning in computer vision are essentially the same tasks at different levels of resolution:



**Image classification:** Determining the class of an image from a predetermined set of categories

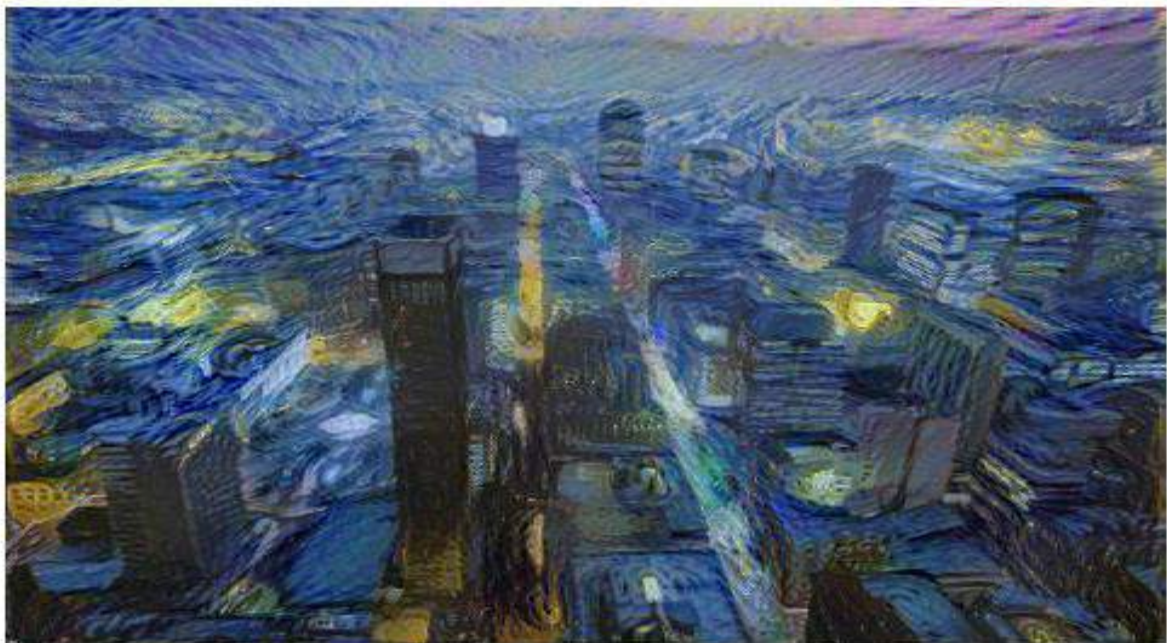
**Object detection:** Finding bounding boxes for objects inside an image and assigning a class probability for each bounding box

**Instance segmentation:** Doing pixel-wise segmentation of an image, outlining every object from a predetermined class list

Computer vision algorithms have found applications in cancer screening, handwriting recognition, face recognition, robotics, self-driving cars, and many other areas.

Another interesting direction in computer vision is generative models. While the models we have already examined perform recognition tasks, generative models change images, or even create entirely new ones. Style transfer models can change the stylistic appearance of

an image to look more like another image. This kind of model can be used to transform photos into paintings that look and feel like the work of an artist.



Another promising approach for training generative models is called Generative

Adversarial Networks (GANs). You use two models to train GANs: Generator and Discriminator. The generator creates images. The discriminator tries to distinguish the real images from your dataset from the generated images. Over time, the generator learns to create more realistic images, while the discriminator learns to identify more subtle mistakes in the image generation process. The

results of this approach speak for themselves. State-of-the-art models can generate realistic human faces, as you can see on page 3 of Nvidia's paper, <https://arxiv.org/pdf/1812.04948.pdf>.

Take a look at the following images. These photos are not real. A deep neural network generated these images:



We can also use GANs to perform conditional image generation. The word *conditional* means that we can specify some parameters for the generator. In particular, we can specify a type of object or texture that is being generated. For example, Nvidia's landscape generator software can transform a simple color-coded image, where specific colors represent soil, sky, water, and other objects, to realistic-looking photos

We can also use GANs to perform conditional image generation. The word *conditional* means that we can specify some parameters for the generator. In particular, we can specify a type of object or texture that is being generated. For example, Nvidia's landscape generator software can transform a simple color-coded image, where specific colors represent soil, sky, water, and other objects, to realistic-looking photos.

End of Session # 8

Topic 9-Model Testing,

<https://www.youtube.com/watch?v=ydSX5XsnD3E>

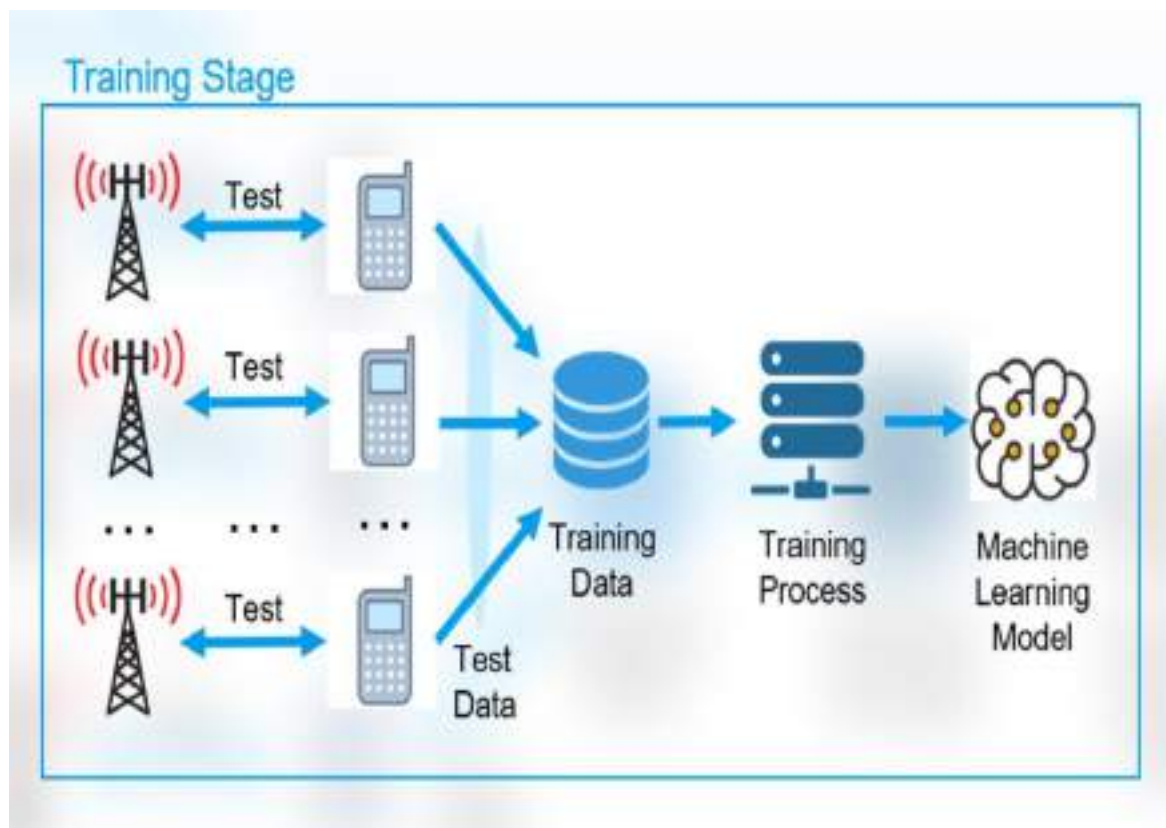
## MDSS: Session # 9

### Testing Your Models

#### Introduction to Model Testing

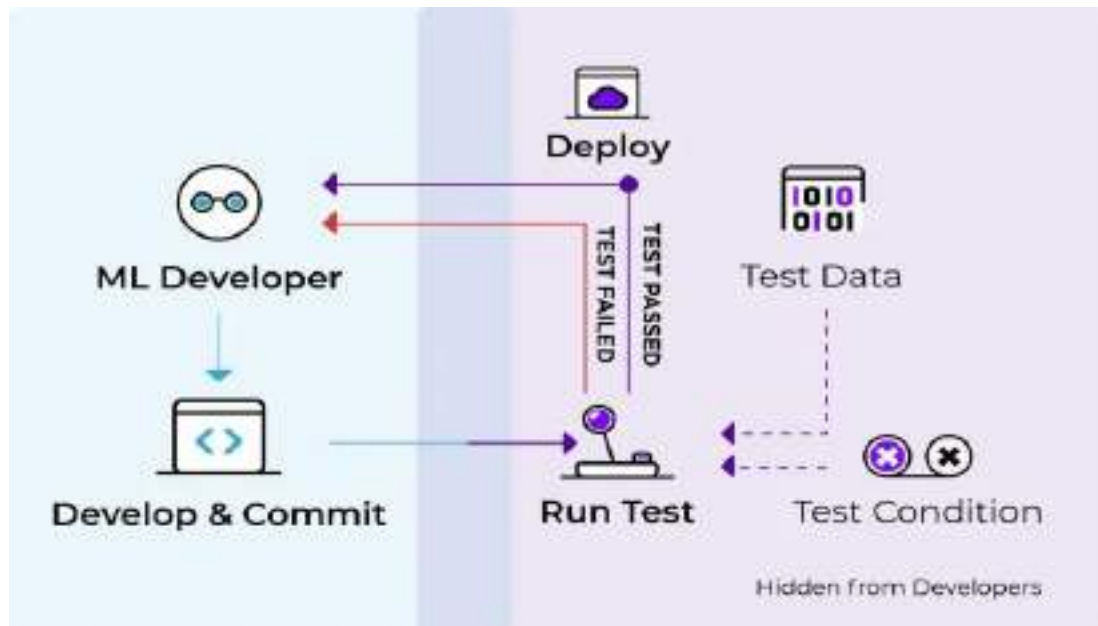
Coming up with a perfect ML model is not simple if you do not use a good testing methodology. This seemingly perfect model will fail the moment you deploy it.

Testing the model's performance is not an easy task, but it is an essential part of every data science project. Without proper testing, you can't be sure whether your models will work as expected, and you can't choose the best approach to solve the task at hand.



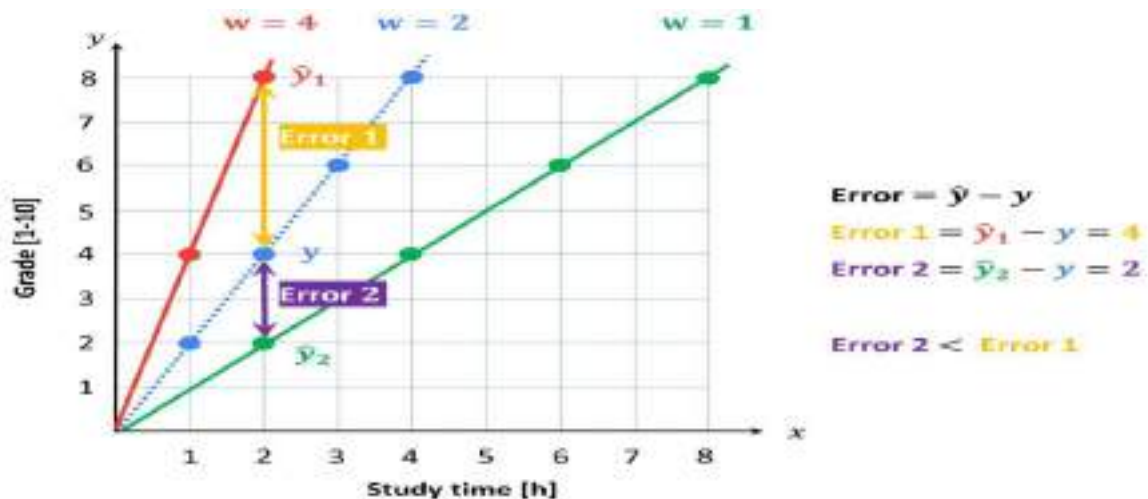
#### Offline Model Testing

Offline model testing encompasses all the model-evaluation processes that are performed before the model is deployed. Before discussing online testing in detail, we must first define model errors and ways to calculate them.



### Understanding Model Errors

Every model can make mistakes because collected data, and the model itself, introduces implications about the nature of your problem. The best example of a good working model is inside your brain.



You use modeling in real time—the brain renders everything you see by interpreting electromagnetic impulses recorded by your eyes. While this picture of the world is imperfect, it is useful as we receive over 90% of information through the visual channel. The last 10% comes from

hearing, touch, and our other senses. Thus, each model, **M**, tries to predict real value, **Y**, by making a guess,  $\hat{Y}$ .

The difference between the real value and model's approximation makes up the model error:

$$\text{Error} = Y - \hat{Y}$$

For regression problems, we can measure the error in quantities that the model predicts. E.g., if we predict house prices using a ML model and get a prediction of \$300,000 for a house with a real price of \$350,000, we can say that the error is \$350,000 - \$300,000 = \$50,000.

For classification problems in the simplest setting, we can measure the error as 0 for a guess, and 1 for a wrong answer. E.g., for a cat/dog recognizer, we give an error of 1 if the model predicts that there is a cat in a dog photo, and 0 if it gives a correct answer.

### Decomposing errors

You won't find a ML model that perfectly solves your problems without making even a single mistake, no matter how small. Since every model makes mistakes, it is critical to understand their nature. Suppose that our model makes a prediction and we know the real value. If this prediction is incorrect, then there is some difference between the prediction and the true value:

$$\text{Error} = \text{real value} - \text{prediction}$$

*The other part of this error will come from imperfections in our data, and some from imperfections in our model. No matter how complex our model is, it can only reduce the modeling error. **Irreducible error** is out of our control, hence its name.*

End of Session # 9

Topic 10-Decomposition of Errors,

<https://www.youtube.com/watch?v=8FPqGJzHF4o>

## MDSS: Session # 10

### Decomposing errors

You won't find a ML model that perfectly solves your problems without making even a single mistake, no matter how small. Since every model makes mistakes, it is critical to understand their nature. Suppose that our model makes a prediction and we know the real value. If this prediction is incorrect, then there is some difference between the prediction and the true value:

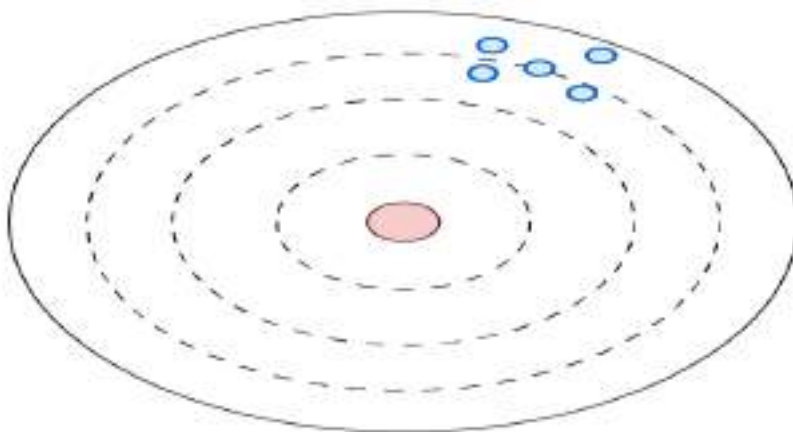
$$\text{Error} = \text{real value} - \text{prediction}$$

*The other part of this error will come from imperfections in our data, and some from imperfections in our model. No matter how complex our model is, it can only reduce the modeling error. **Irreducible error** is out of our control, hence its name.*

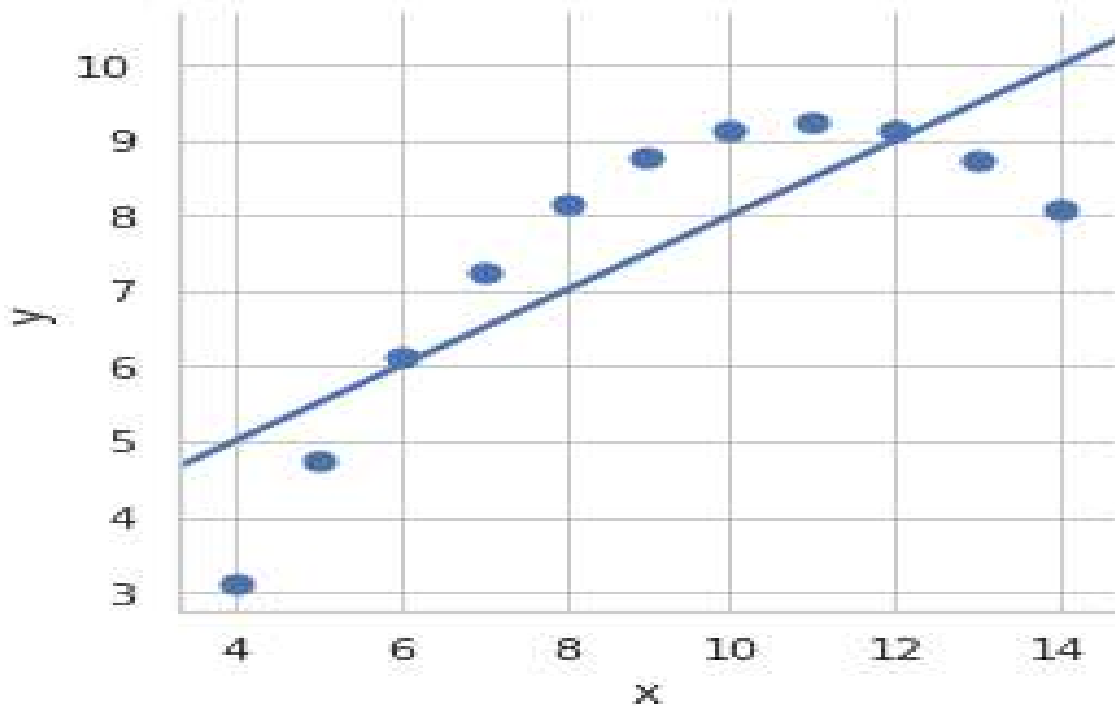
Let's look at it in the following formula:

$$\text{Error} = \text{Error}_{\text{reducible}} + \text{Error}_{\text{irreducible}}$$

The **red** center of each target represents our goal (real value), and the **blue** shots represent the model predictions. In the target, the model's aim is off—all predictions are close together, but they are far away from the target. This kind of error is called **bias**. The simpler our model is, the more bias it will have. For a simple model, the bias component can become prevailing .

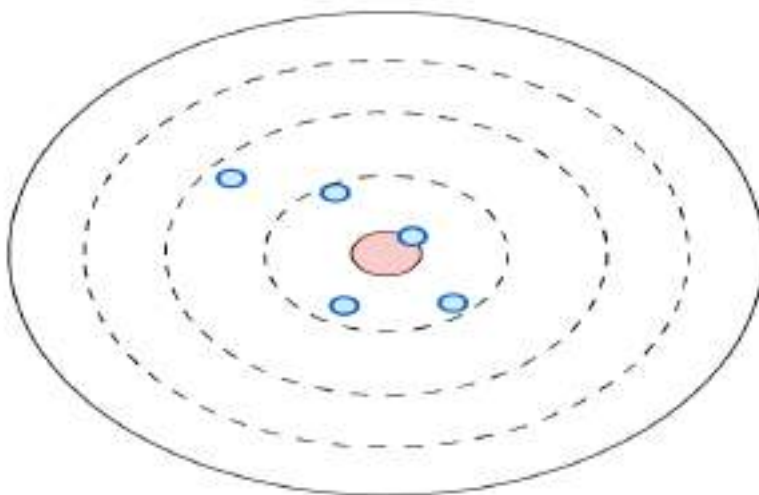






*In this plot, we try to model a complex relationship between variables with a simple line. This kind of model has a high bias.*

The second component of the model error is **variance**:



All predictions appear to be clustered around the true target, but the spread is too high. The source of this error comes from the model's sensitivity to fluctuations in data. If the model has high variance, randomness in measurements can lead to very different predictions.

So far, we have decomposed model error into the three following numbers:

$$\text{Error} = \text{bias} + \text{variance} + \text{irreducible error}$$

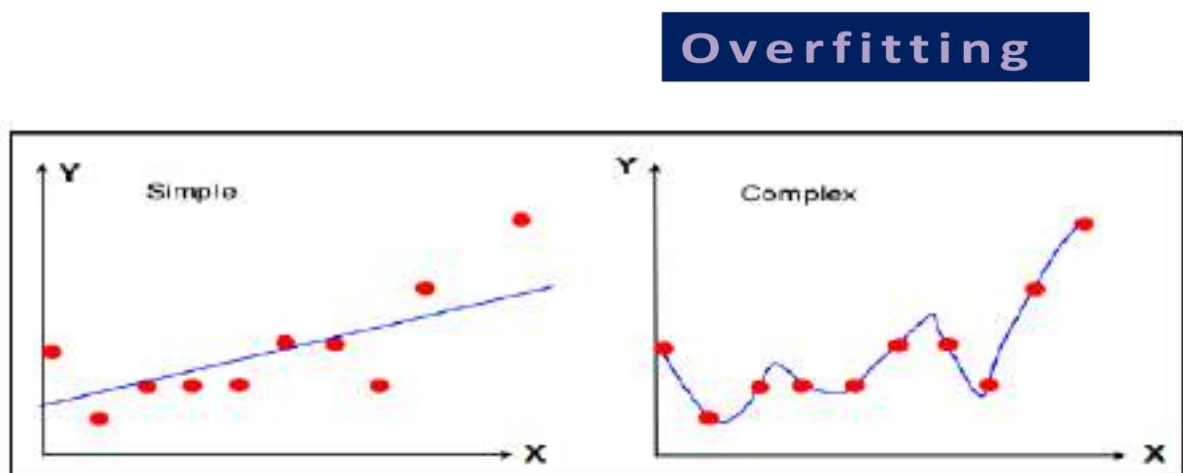
It is not a coincidence that bias and variance sit close together in the same formula. There is a relationship between them. Predictive models show a property called the **bias-variance tradeoff**—the more biased a model, the lower the variance component of the error. And in reverse, the more variance it has, the lower its bias will be.

Typically, models that impose some kind of structure in the data have a high bias (they assume certain laws that the data conforms to). Biased models will work well, as long as the data does not contradict the underlying logic of the model. To give you an example of such a model, think of a simple line. For example, we will predict a housing price as a linear function of its size in square feet:

$$\text{Housing price} = \$10000 \times \text{number of square feet}$$

Notice that if we change the square footage by a little, say 0.1, then the prediction won't change by much. Thus, this model has low variance. When the model is sensitive to changes in its input, its variance will outgrow the bias. The variance component will grow with your model increases in complexity and the total number of parameters grows.

In the following plot, you can see how two different models fit the same dataset. The first simple model has low variance, and the second complex model has high variance:



In the preceding plot, slight changes in **X** can lead to large fluctuations of **Y**. Models with high variance are robust and imply that the data is much less structured.



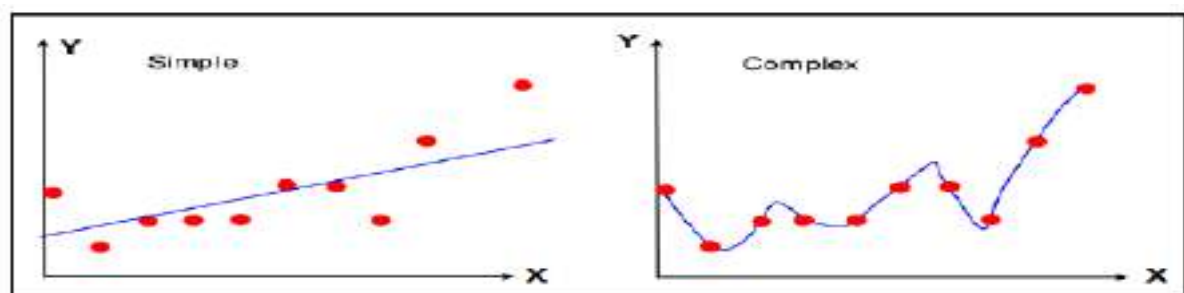
Topic 11-Overfitting,

<https://www.youtube.com/watch?v=jG1zLio9F4g>

## Understanding Overfitting

The bias-variance trade-off goes hand in hand with a very important problem in ML called overfitting. If your model is too simple, it will cause large errors. If it is too complex, it will memorize the data too well. An over fitted model remembers data too well and acts like a database. Suppose that our housing dataset contains some lucky deals where previous houses had a low price because of circumstances not captured in the data. An over fit model will memorize those examples too closely and predict incorrect price values on unseen data.

Now, having understood the error decomposition, can we use it as a stepping stone to design a model-testing pipeline?



**Overfitting**

We need to determine how to measure model error in such a way that it will correspond to the real model performance on unseen data. The answer comes from the question itself. We will split all the available data into two sets: a training set and a test set, as shown in the following screenshot:



We will use data in the training set to train our model. The test set acts as unseen data and you should not use the test set in the training process. When the model's training is

finished, you can feed the test data into your model. Now you can calculate errors for all predictions. The model did not use the test data during training, so the test set error represents the model error on unseen data. The drawback to this approach is that you take a significant amount of data, usually up to 30%, to use for testing. This means less training data and lower model quality. There is also a caveat – if you use your test set too much, error metrics will start to lie. For example, suppose that you did the following:

1. Trained a model
2. Measured the error on the test data
3. Changed your model to improve the metrics
4. Repeated steps 1-3 ten times
5. Deployed the model to production

It is likely that the quality of your model will be much lower than expected. Why did this happen? Let's look more closely *step 3*. *You looked at a score, and changed your model or data processing code several consecutive times*. In fact, you did several learning iterations by hand. By repeatedly improving the test score, you indirectly disclosed information about the test data to your model. When the metric values measured on a test set deviate from the metrics measured on the real data, we say that the test data has leaked into our model. Data leaks are notoriously hard to detect before they cause damage. To avoid them, you should always be mindful of the possibility of a leak, think critically, and follow best practices.

We can use a separate piece of data to fight test set leakage. Data scientists use validation sets to tune model parameters and compare different models before choosing the best one. Then, the test data is used only as a final check that informs you about model quality on unseen data. After you have measured the test metric scores, the only decision left is to make is whether the model will proceed to testing in a real-world scenario.

In the following screenshot, you can see an example of a train/validation/test split of the dataset:



Unfortunately, the following two problems persist when we use this approach:

- The information about our test set might still leak into our solution after many iterations. Test-set leakage does not disappear completely when you use the validation set, it just becomes slower. To overcome this, change your test data from time to time. Ideally, make a new test set for every model-deployment cycle.
- You might over fit your validation data quickly, because of the train-measure change feedback cycle for tuning your models.

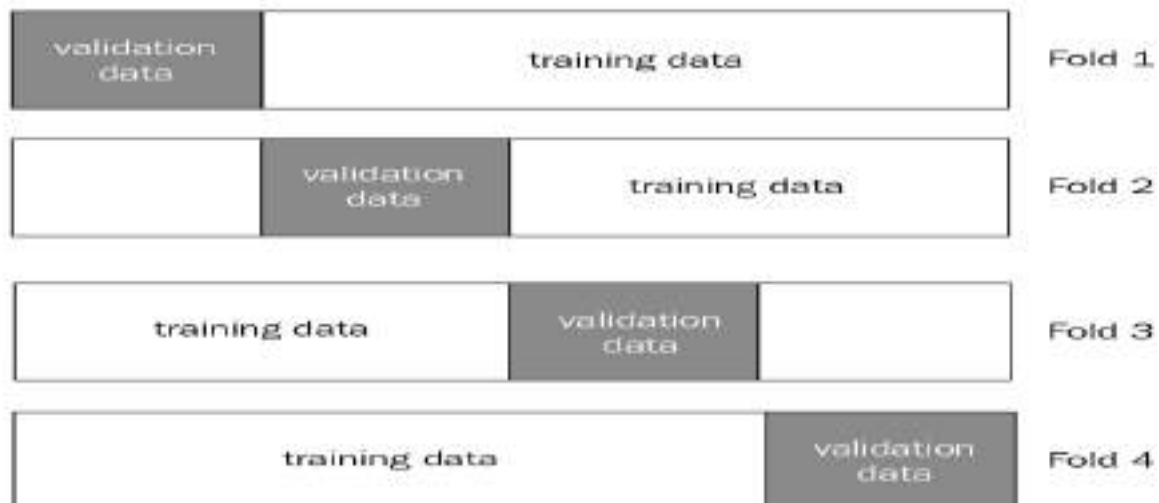
Topic 12-Accuracy Metrics,

<https://www.youtube.com/watch?v=a5R55Ehi4fl>



### Avoiding Overfitting

The following screenshot explains cross-validation visually:



Cross-validation has one main drawback: it requires significantly more computational resources to assess model quality. In our example, to make a single assessment we needed to fit three models. With a regular train/test split, we would train only one model. In addition, cross-validation accuracy will grow with the number of iterations you use (also called folds). So cross-validation allows you to use more data for training, while requiring more computational resources. How do we choose between cross-validation and train validation- test splits for projects?

In cross-validation, is a variable parameter that is set up by a data scientist. The lowest possible value is 1, which is equivalent to a simple train/test split. The largest extreme is equal to the number of data points in the dataset. This means that if we have points in the dataset, the model will be trained and tested times. This special case of cross validation is called leave-one-out cross-validation. In theory, a larger number of folds means that the cross-validation will return more accurate metric values. While leave-one-out cross-validation is the most theoretically accurate method, it is seldom used in practice because of the large computational requirements. In practice, the values of range from 3 to 15 folds, depending on the dataset size. Your project may need to use more, so take this as advice and not as a rule.



We will use data in the training set to train our model. The test set acts as unseen data and you should not use the test set in the training process. When the model's training is finished, you can feed the test data into your model. Now you can calculate errors for all predictions. The model did not use the test data during training, so the test set error represents the model error on unseen data. The drawback to this approach is that you take a significant amount of data, usually up to 30%, to use for testing. This means less training data and lower model quality. There is also a caveat – if you use your test set too much, error metrics will start to lie. For example, suppose that you did the following:

1. Trained a model
2. Measured the error on the test data
3. Changed your model to improve the metrics
4. Repeated steps 1-3 ten times
5. Deployed the model to production

It is likely that the quality of your model will be much lower than expected. Why did this happen? Let's look more closely *step 3*. You looked at a score, and changed your model or data processing code several consecutive times. In fact, you did several learning iterations by hand. By repeatedly improving the test score, you indirectly disclosed information about the test data to your model. When the metric values measured on a test set deviate from the metrics measured on the real data, we say that the test data has leaked into our model. Data leaks are notoriously hard to detect before they cause damage. To avoid them, you should always be mindful of the possibility of a leak, think critically, and follow best practices.

We can use a separate piece of data to fight test set leakage. Data scientists use validation sets to tune model parameters and compare different models before choosing the best one. Then, the test data is used only as a final check that informs you about model quality on unseen data. After you have measured the test metric scores, the only decision left is to make is whether the model will proceed to testing in a real-world scenario.

In the following screenshot, you can see an example of a train/validation/test split of the dataset:



Unfortunately, the following two problems persist when we use this approach:

- The information about our test set might still leak into our solution after many iterations. Test-set leakage does not disappear completely when you use the validation set, it just becomes slower. To overcome this, change your test data from time to time. Ideally, make a new test set for every model-deployment cycle.
- You might over fit your validation data quickly, because of the train-measure change feedback cycle for tuning your models.

To prevent overfitting, you can randomly select train and validation sets from your data for each experiment. Randomly shuffle all available data, then select random train and validation datasets by splitting the data into three parts according to proportions you have chosen.

There is no general rule for how much training, validation, and testing data you should use. Often, more training data means a more accurate model, but it means that you will have less data to assess the model's performance. The typical split for medium-sized datasets (up to 100,000 data points) is to use 60-80% of the data to train the model and use the rest for validation.

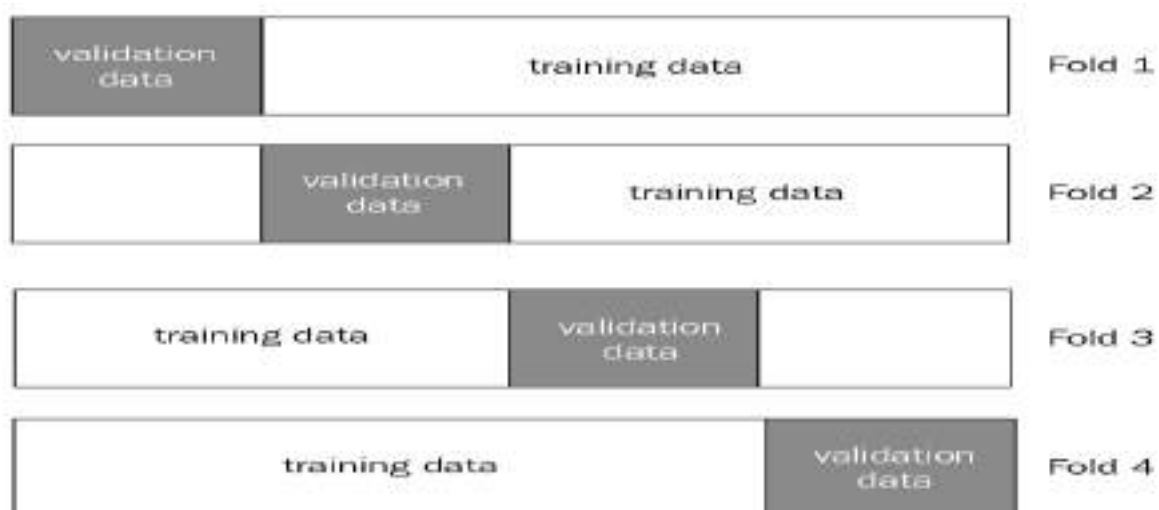
The situation changes for large datasets. If you have a dataset with 10,000,000 rows, using 30% for testing would comprise 3,000,000 rows. It is likely that this amount would be overkill. Increasing test and validation test sizes will yield diminishing returns. For some problems, you will get good results with 100,000 examples for testing, which would amount for a 1% test size. The more data you have, the lower the proportion you should use for testing.

Often, there is too little data. In those situations, taking from 30%-40% data for testing and validation might severely decrease the model's accuracy. You can apply a technique called cross-

validation in data-scarce situations. With cross-validation, there's no need to create a separate validation or test set. Cross-validation proceeds in the following way:

1. You choose some fixed number of iterations—three, for example.
2. Split the dataset into three parts.
3. For each iteration, cross-validation uses 2/3 of the dataset as a training data and 1/3 as validation data.
4. Train model for each of the three train-validation set pairs.
5. Calculate the metric values using each validation set.
6. Aggregate the metrics into a single number by averaging all metric values

The following screenshot explains cross-validation visually:



Cross-validation has one main drawback: it requires significantly more computational resources to assess model quality. In our example, to make a single assessment we needed to fit three models. With a regular train/test split, we would train only one model. In addition, cross-validation accuracy will grow with the number of iterations you use (also called folds). So cross-validation allows you to use more data for training, while requiring more computational resources. How do we choose between cross-validation and train validation- test splits for projects?

In cross-validation, is a variable parameter that is set up by a data scientist. The lowest possible value is 1, which is equivalent to a simple train/test split. The largest extreme is equal to the number of data points in the dataset. This means that if we have points in the dataset, the model will be

trained and tested times. This special case of cross validation is called leave-one-out cross-validation. In theory, a larger number of folds means that the cross-validation will return more accurate metric values. While leave-one-out cross-validation is the most theoretically accurate method, it is seldom used in practice because of the large computational requirements. In practice, the values of range from 3 to 15 folds, depending on the dataset size. Your project may need to use more, so take this as advice and not as a rule.

The following table sums up a general way of thinking

	Model training requires low to moderate computational resources and time	Model training requires large computational resources and takes a long time
Small to medium dataset	Cross-validation	Either
Large dataset	Either	Train/validation/test split

Another important aspect related to model testing is how to split the data. A slight error in your splitting logic can mean all your testing efforts were in vain. Splitting is easy, if all observations in your dataset are independent. Then you can use random data splits. But what if we are solving the stock-price prediction problem? When our data rows are tied to time, we can't look at them as independent values. Today's stock prices depend on their past values. If this wasn't true, the prices would randomly jump from \$0 to \$1,000. In this situation, suppose we have two years' worth of stock data, from January 2017 to December 2018. If we use random splits, it is possible that our model will train in September 2018 and test on February 2017. This makes no sense. We must always think about causal relationships and dependencies between your observations and be sure to check whether your validation procedure is correct.

## Using technical metrics

Each model, no matter how complex and accurate, makes mistakes. It is natural to expect that some models will be better than others when solving a specific problem. Currently, we can measure errors by comparing individual model predictions with the ground truth. It would be useful to summarize them into a single number for measuring the model's performance. We can use a metric to do this. There are many kinds of metrics that are suitable for different machine learning problems.

In particular, for regression problems the most common metric is the **root mean square error**, or **RMSE**:

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (\text{predicted}_i - \text{actual}_i)^2}{N}}$$

Let's examine the elements of this formula:

- $N$  is the total number of data points.
- $\text{predicted} - \text{actual}$  measures the error between ground truth and model prediction.
- The Sigma sign at the start of the formula means sum.

Another popular way to measure regression errors is **mean absolute error (MAE)**:

$$MAE = \frac{1}{N} \sum_{i=1}^N |\text{predicted}_i - \text{actual}_i|$$

Note that MAE is very similar to RMSE. Compared to MAE, RMSE has a square root instead of absolute value and it squares errors. While MAE and RMSE may seem identical, there are some technical differences between them. Data scientists can choose best metrics for a problem, knowing their trade-offs and shortcomings. You don't need to learn them all, but I would like to highlight one difference to give you a general feel of the thought process. RMSE penalizes large errors more than MAE. This property comes from the fact that RMSE uses squared errors, while MAE uses absolute values. To illustrate, an error of 4 would be 4 in MAE, but in RMSE it will turn into 16 because of the square.

For classification problems, the metric-calculation process is more involved. Let's imagine that we are building a binary classifier that estimates the probability of a person having pneumonia. To calculate how accurate, the model is, we may just divide the total of correct answers by the number of rows in the dataset:

$$Accuracy = \frac{n_{correct}}{N}$$

Here,  $n_{correct}$  is the amount of correct predictions, and  $N$  is the total number of predictions. Accuracy is simple to understand and calculate, but it has a major flaw. Let's assume the average probability of having pneumonia is 0.001%. That is, one person out of 100,000 has the illness. If you had collected data on 200,000 people, it is feasible that your dataset would contain only two positive cases. Imagine you have asked a data scientist to build a machine learning model that estimates pneumonia probability based on a patient's data. You have said that you would only accept an accuracy of no less than 99.9%. Suppose that someone created a dummy algorithm that always outputs zeros.

This model has no real value, but its accuracy on our data will be high as it will make only two errors:

$$Accuracy = \frac{199998}{200000} = 99.999\%$$

The problem is that accuracy considers only global fraction of answers. When one class outnumbers the others, accuracy outputs misleading values.

This model has no real value, but its accuracy on our data will be high as it will make only two errors:

$$Accuracy = \frac{199998}{200000} = 99.999\%$$

The problem is that accuracy considers only global fraction of answers. When one class outnumbers the others, accuracy outputs misleading values.

Let's look at model predictions in more detail by constructing a confusion table:

	Model prediction: Has pneumonia	Model prediction: Does not have pneumonia
Real outcome: Has pneumonia	0	2
Real outcome: Does not have pneumonia	0	199,998

After looking at this table, we can see that the dummy model won't be helpful to anyone. It didn't identify two people with the condition as positive. We call those errors **False Negatives (FN)**. The model also correctly identified all patients with no pneumonia, or **True Negatives (TN)**, but it has failed to diagnose ill patients correctly.



Now, suppose that your team has built a real model and got the following results:

	Model prediction: Has pneumonia	Model prediction: Does not have pneumonia
Real outcome: Has pneumonia	2	0
Real outcome: Does not have pneumonia	30	199,968

This model correctly identified two cases, making two **True Positive (TP)** predictions. This is a clear improvement over the previous iteration. However, the model also identified 30 people as having pneumonia, while they were not ill in reality. We call such an error a **False Positive (FP)** prediction. Is having 30 false positives a significant disadvantage? That depends on how physicians will use the model. If all subjects will be automatically prescribed with heavy medication with side-effects, false positives can be critical.

It may be less severe if we consider a positive model only as a possibility of having a disease. If a positive model answer only signals that the patient must go through a specific set of diagnostic procedures, then we can see a benefit: to achieve the same level of pneumonia identification, therapists will diagnose only 32 patients, where previously they had to investigate 200,000 cases. If we had not used the confusion table, we might have missed dangerous model behavior that would negatively affect people's health.

Next, your team has done another experiment and created a new model:

	Model prediction: Has pneumonia	Model prediction: Does not have pneumonia
Real outcome: Has pneumonia	0	2
Real outcome: Does not have pneumonia	100,000	99,998

Does this model perform better? The model would have missed one patient that needed therapy and assigned 100,000 healthy people to a treatment group, making physicians do unnecessary work. In truth, you can make the final decision only after presenting results to the people who will use the model. They may have a different opinion on what is best. It would be best to define this at the first stages of the project by creating a model-testing



methodology document by collaborating with experts in the field.

You will face binary classification problems everywhere, thus having a good understanding of terminology is important.

You can see all new concepts summed up in the following table:

	Model prediction: 1 (positive case)	Model prediction: 0 (negative case)
Real outcome: 1 (positive case)	TP	FN
Real outcome: 0 (negative case)	FP	TN

It is crucial to note that you can control the amount of false positive and false negative responses for a single model. Classifiers output a probability of a data point belonging to a class. That is, the model prediction is a number between 0 and 1. You can decide whether a prediction belongs to a positive or negative class by comparing it with a threshold. For example, if the threshold is 0.5, then any model prediction greater than 0.5 will belong to class 1 and to 0 otherwise

By changing the threshold, you can change the proportions between the cells in the confusion table. By choosing a large threshold, like 0.9, the volume of false positive responses will decrease, but false negative responses will increase. Threshold selection is essential for binary classification problems. Some environments, such as digital advertising, will be more forgiving to false positives, while in others, such as healthcare or insurance, may find them unacceptable.

Confusion tables provide deep insights into classification problems but require your attention and time. This can be limiting when you want to do numerous experiments and compare many models. To simplify the process, statisticians and data scientists have designed many metrics that sum up classifier performance without suffering from problems like accuracy metrics do. First, let's examine some ways to summarize confusion table rows and columns. From there, we will explore how to condense it into a single statistic. In the following table, you can see two new metrics for summarizing different kinds of errors, precision and recall:

	Model prediction: 1 (positive case)	Model prediction: 0 (negative case)	Combined metric
Real outcome: 1 (positive case)	True Positive	False Negative	$precision = \frac{TP}{TP+FP}$
Real outcome: 0 (negative case)	False Positive	True Negative	

Combined metric	$recall = \frac{TP}{TP + FN}$ , also called True Positive Rate (TPR)		
-----------------	--	--	--

Precision measures a proportion of positive (relevant) cases that your model has identified. If your model predicted 10 positive cases and 2 positive predictions turned out to be negative in reality, then its precision would be 0.8. Recall represents a probability of correctly predicting a positive case. If out of 10 positive cases, the model had predicted all 10 correctly (10 true positives) and marked 5 negative cases as positive (5 false positives), then its recall would be 0.67. A recall of 0.67 means that if our model predicts a positive case, it will be correct 67 times out of 100.

For binary classification, precision and recall diminish the amount of metrics we must work with to two. This is better, but not ideal. We can sum up everything into a single number by using a metric called F1-score. You can calculate F1 using the following formula:

$$F1 = 2 \frac{precision \times recall}{precision + recall}$$

F1 is 1 for a perfect classifier and 0 for the worst classifier. Because it considers both precision and recall, it does not suffer from the same problem as accuracy and is a better default metric for classification problems.

Topic 13-Online Model Testing,

<https://www.youtube.com/watch?v=8WNJe7WzIV8>

## MDSS: Session # 13

### Online Model Testing

Even a great offline model testing pipeline won't guarantee that the model will perform exactly the same in production. There are always risks that can affect your model performance, such as the following:

**Humans:** We can make mistakes and leave bugs in the code.

**Data collection:** Selection bias and incorrect data-collection procedures may disrupt true metric values

**Changes:** Real-world data may change and deviate from your training dataset, leading to unexpected model behaviour.

The only way to be certain about model performance in the near future is to perform a live test. Depending on the environment, such test may introduce big risks. For example, models that assess airplane engine quality or patient health would be unsuitable for real world testing before we become confident in their performance.

When the time for a live test comes, you will want to minimize risks while making statistically valid conclusions. Thankfully, there is a statistical framework for that purpose known as hypothesis testing. When performing a hypothesis test, you check the validity of some idea (hypothesis) by collecting data and executing a statistical test. Imagine you need to check whether your new advertising model increases revenues from the ad service. To do this, you randomly split all your clients into two groups: one group uses the old advertising algorithm, while the others see ads recommended by a new algorithm. After you have collected a sufficient volume of data, you compare two groups and measure differences between them. Why do we need to bother with statistics, you may ask?

Because we can answer the following questions only with the help of stats:

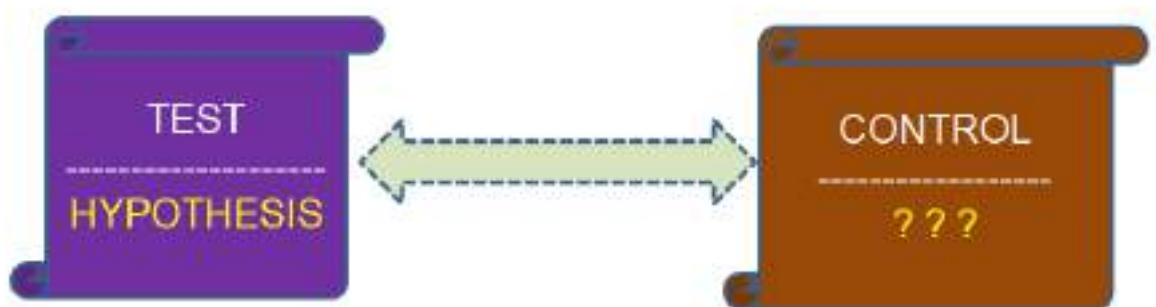
- How should I sort (sample) individuals into each group? Can my sampling process distort results of the test?

- What is the minimum number of clients in each group? Can random fluctuations in the data affect my measurements?
- How long should I run the test for to get a confident answer?
- What formula should I use to compare results in each group?

The experiment setup for a hypothesis test splits test targets into two groups on purpose. We can try to use a single group instead. For instance, we can take one set of measurements with the old model. After the first part of the experiment is finished, we can deploy the new algorithm and measure its effect. Then, we compare two measurements made one after another. What could go wrong? In fact, the results we get wouldn't mean anything. Many things could have changed in between our measurements, such as the following:

- User preferences
- General user mood
- Popularity of our service
- Average user profile
- Any other attribute of users or businesses

All these hidden effects could affect our measurements in unpredictable ways, which is why we need two groups: test and control.



We must select these groups in such a way that the only difference between them is our hypothesis.

➤ ***It should be present in the test group and missing from the control group.***

Ex. In medical trials, control groups are the ones who get the placebo. Suppose we want to test the positive effect of a new painkiller. Here are some examples of bad test setups:

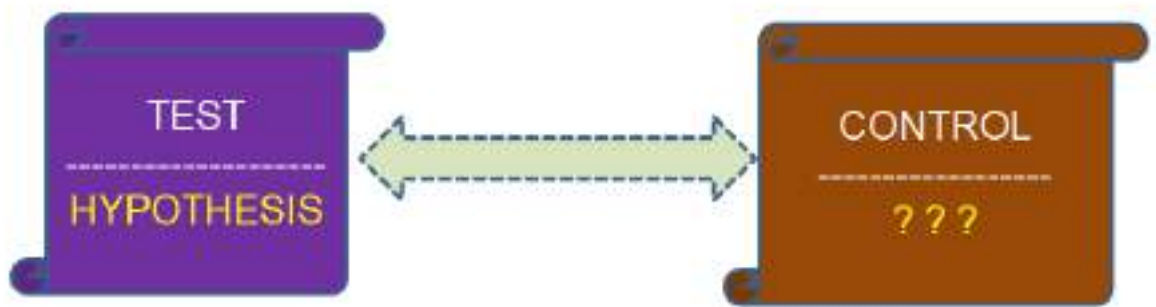
- The control group consists only of women.
- The test and control groups are in different geographical locations.
- You use biased interviews to preselect people for an experiment.

Topic 14-Hypothesis Testing & MAB Problem,  
<https://www.youtube.com/watch?v=W1hKN9wCqqA>

## MDSS: Session # 14

### Hypothesis testing & MAB problem

All these hidden effects could affect our measurements in unpredictable ways, which is why we need two groups: test and control.



We must select these groups in such a way that the only difference between them is our hypothesis.

- ***It should be present in the test group and missing from the control group.***

Ex. In medical trials, control groups are the ones who get the placebo. Suppose we want to test the positive effect of a new painkiller. Here are some examples of bad test setups:

- The control group consists only of women.
- The test and control groups are in different geographical locations.
- You use biased interviews to preselect people for an experiment.

The easiest way to create groups is random selection. Truly random selection may be hard to do in the real world, but is easy if you deal with internet services.

- There, you may just randomly decide which version of your algorithm to use for each active user.
- Be sure to always design experiment setups with an experienced statistician or data scientist, as correct tests are notoriously hard to execute, especially in offline settings.

Statistical tests check the validity of a null hypothesis, that is, that the results you got are by



chance. The opposite result is called an alternative hypothesis. For instance, here is the hypothesis set for our ad model test:

**Null hypothesis: The new model does not affect the ad service revenue.**

**Alternative hypothesis: The new model affects the ad service revenue.**

Typically, a statistical test measures the probability of a null hypothesis being true. If the chances are low, then the alternative hypothesis is true. Otherwise, we accept the null hypothesis. If, according to a statistical test, the probability that the new model does not affect service revenue would be 5%, we would say that we accept the alternative hypothesis at a 95% confidence level. This means the model affects the ad service revenue with a 95% probability. The significance level for rejecting the null hypothesis depends on the level of risk you want to take. For an ad model, a 95% significance may be enough, while no less than a 99% significance is satisfactory for a model that tests patient health conditions.

**The most typical hypothesis test is comparing two means.** If we use this test in our ad model example, we would measure average revenues with and without the new ranking algorithm. We may accept or reject the null hypothesis using a test statistic when the experiment is finished.

The amount of data you need to collect for conducting a hypothesis test depends on several factors:

**Confidence level:** The more statistical confidence you need, the more data is required to support the evidence.

**Statistical power:** This measures the probability of detecting a significant difference, if one exists. The more statistical power your test has, the lower the chance of false negative responses.

**Hypothesized difference and population variance:** If your data has large variance, you need to collect more data to detect a significant difference. If the difference between the two means is smaller than population variance, you would need even more data.

You can see how different test parameters determine their data hunger in the following table:

Confidence level	Statistical power	Hypothesized difference	Population variance	Recommended sample size
95%	90%	\$10	\$100	22 ad demonstrations to clients
99%	90%	\$10	\$100	30 ad demonstrations to clients
99%	90%	\$1	\$100	2,976 ad demonstrations to clients

While powerful, hypothesis tests have limitations:

- You need to wait until the experiment ends before you can apply its results.
- If your model is bad, you won't be able to reduce damage without compromising the test procedure.
- Another limitation is that you can test only one model at a time with a single hypothesis test

In situations where you can trade off statistical rigor for speed and risk-aversion, there is an alternative approach called **Multi-Armed Bandits (MABs)**. To understand how **MABs** work, imagine yourself inside a casino with lots of slot machines. You know that some of those machines yield better returns than others. Your task is to find the best slot machine with a minimal number of trials. Thus, you try different (multi) arms of slot machines (bandits) to maximize your reward. You can extend this situation to testing multiple ad models: for each user, you must find a model that is most likely to increase your ad revenue.

### **Multi-armed bandit (MAB) examples**

One real-world example of a multi-armed bandit problem is when a news website has to make a decision about which articles to display to a visitor. With no information about the visitor, all click outcomes are unknown. The first question is, which articles will get the most clicks?

The most popular MAB algorithm is called an epsilon-greedy bandit – whose workings method are simple:



1. Select a small number called **epsilon**. Suppose we have chosen **0.01**.
2. Choose a random number between **0** and **1**. This number will determine whether MAB will explore or exploit a possible set of choices.
3. If the number is lower or equal to epsilon, make a choice at random and record a reward after making an action tied to your choice. We call this process exploration – MAB tries different actions at random with a low probability to find out their mean reward.
4. If your number is greater than epsilon, make the best choice according to the data you have collected. We call this process exploitation – MAB exploits knowledge it has collected to execute an action that has the best expected reward. MAB selects the best action by averaging all recorded rewards for each choice and selecting a choice with the greatest reward expectation.

Frequently, we start with large values of epsilon and decrease it to smaller values. In this way, MAB explores lots of random choices at the start and exploits the most profitable actions toward the end. The exploration frequency is gradually diminishing, becoming closer to zero.

When you first launch MAB, it collects rewards from random actions. As time passes, you will see that average rewards for all choices converge to their true values. The major benefit of MABs is that they change their behavior in real time. While someone is waiting for a hypothesis test results, MAB gives you a changing picture while covering to the best choice. Bandits are one of the most basic reinforcement learning algorithms. Despite their simplicity, they can provide good results.

Topic15-Building and Sustaining a DS Team,  
<https://www.youtube.com/watch?v=qNVjrC6cQQI>

## MDSS: Session # 15

### Building and Sustaining a DS Team

#### Building and Sustaining a Team

- Defining DS team roles,
- Exploring team roles and their responsibilities,
- Common flaws of technical interviews,
- Introducing values and ethics into the interview,
- Designing good interviews, Achieving team Zen,
- Leadership and people management,
- Facilitating a growth mindset.

#### Case study—creating a DS department

DS is an innovation for most organizations. However, every innovation requires deep and careful thinking – *not all ideas are equally good, and not all of them have the necessary resources for implementation.*

This session will help you to identify the best ideas and strip them down to the minimum valuable product.

Another important consideration is how successfully you can sell your idea to all stakeholders who might get benefit from it.

Merging knowledge of modern data analysis algorithms and business domain expertise is a necessary step for every project. The sessions will provide the importance of using a scientific approach in business and shall help you to answer the following questions:

- How can you find an efficient data science application for your business?
- What are business and technical metrics and how should we define them?
- How can we introduce project goals that align well with our business?

#### Importance of teamwork

You can complete a complex project with a team more efficiently than in isolation.

Of course, one person can build a house alone, but by working with others, they will finish the house faster, and the result will be better.

When you work with a team, everyone can specialize in performing several closely-related types of work. To explore different specialties, ***let's look at an example of how houses are built.***

*Building a roof requires one set of skills, while setting up electricity is completely different.*

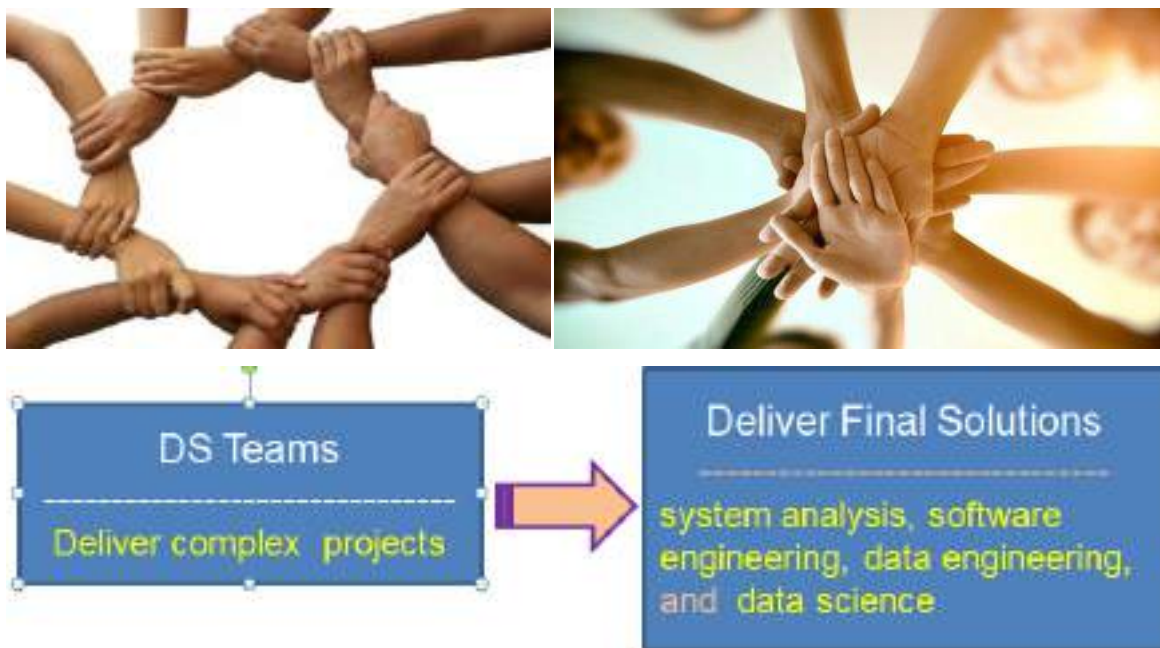
As a manager, you need to have a basic understanding of all specialties, so as to understand all components necessary for completion. **How is this concept related to DS?**

Use this knowledge to derive roles and specialties in a building a DS team.

- **Define, explore, and understand (DEU)** different team roles as well as each role's key skills and responsibilities.

### Defining DS Team Roles

DS teams need to deliver complex projects where system analysis, software engineering, data engineering, and data science are used to deliver the final solution.



We will need to explore the main DS project roles.

**The project role depicts a set of related activities that can be performed by an expert.** Role-expert is not strictly a one-to-one correspondence, as many experts have the expertise to handle multiple roles at once.

## Average DS Team Composition



An average data science team will include:

- a business analyst (BA),
- a system analyst (SA),
- a data scientist (DSc),
- a data engineer (DE), and
- a data science team manager (DSc Team Mgr).



More complex projects may also benefit from the participation of a software architect and backend / frontend development teams.

Topic 16-Role & Responsibilities of Team Members,  
[https://www.youtube.com/watch?v=b\\_Ajj93BIYA](https://www.youtube.com/watch?v=b_Ajj93BIYA)

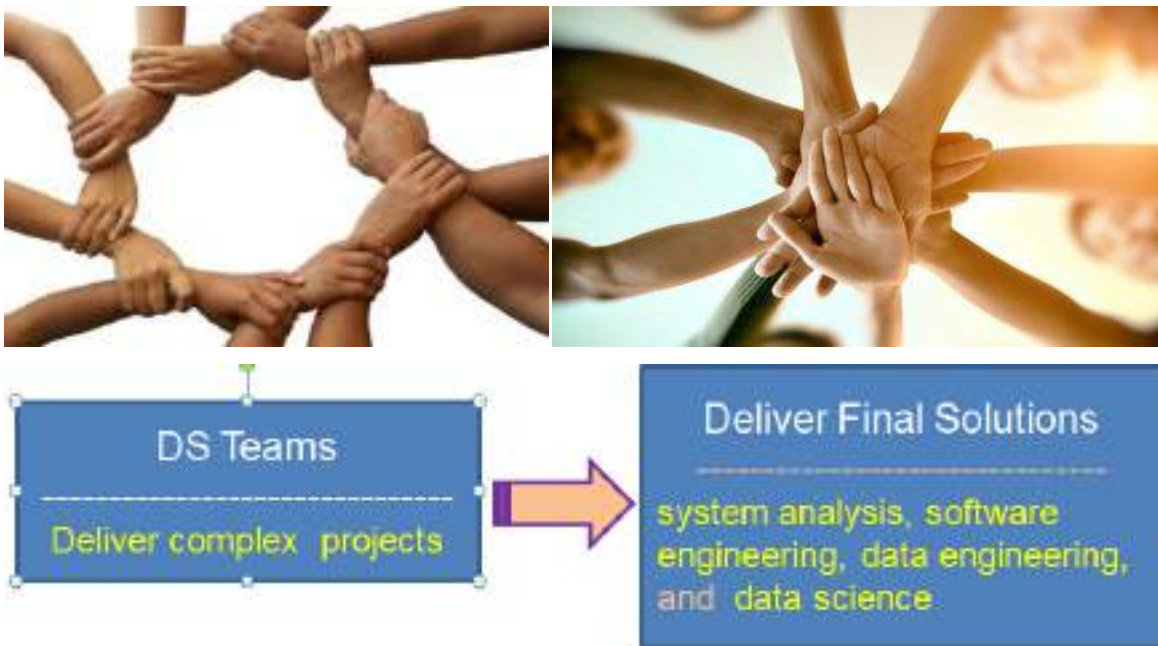


## MDSS: Session # 16

### Roles and responsibility of team members

#### Defining DS Team Roles

DS teams need to deliver complex projects where system analysis, software engineering, data engineering, and data science are used to deliver the final solution.



We will need to explore the main DS project roles.

**The project role depicts a set of related activities that can be performed by an expert.** Role-expert is not strictly a one-to-one correspondence, as many experts have the expertise to handle multiple roles at once.

#### Average DS Team Composition

An average data science team will include:

1. a business analyst (BA),
2. a system analyst (SA),
3. a data scientist (DSc),
4. a data engineer (DE), and
5. a data science team manager (DSc Team Mgr).



More complex projects may also benefit from the participation of a software architect and backend / frontend development teams.

- CORE RESPONSIBILITIES
  - PROJECT STACKHOLDER
  - PROJECT USERS
- CORE RESPONSIBILITY OF ANALYSIS TEAM
  - BUSINESS ANALYSTS
  - SYSTEM ANALYSTS
- CORE RESPONSIBILITY OF ANALYSIS TEAM
  - Data SCIENTIST
  - DATA ENGINEER
- CORE RESPONSIBILITY OF SOFTWARE TEAM

## CORE RESPONSIBILITIES

Here are the core responsibilities of each team role:

**Project stakeholders:** Represent people who are interested in the project; in other words, your customers. They generate and prioritize high-level requirements and goals for the project.

**Project users:** People who will use the solution you are building. They should be involved in the requirements-specification process to present a practical view on the system's usability.

Let's look at the core responsibilities in the analysis team:

**Business analysts:** The main business expert of the team. They help to shape business requirements and help data scientists to understand the details about the problem domain. They define business requirements in the form of a **business requirements document (BRD)**, or stories, and may act as a product owner in agile teams.

**System analysts:** They define, shape, and maintain software and integration requirements. They create a **software requirements document (SRD)**. In simple projects or Proof of Concepts (**PoC**), this role can be handled by other team members.

**Data analysts:** Analysis in data science projects often requires building complex database queries and visualizing data. Data analysts can support other team members by creating data marts and interactive dashboards and derive insights from data.

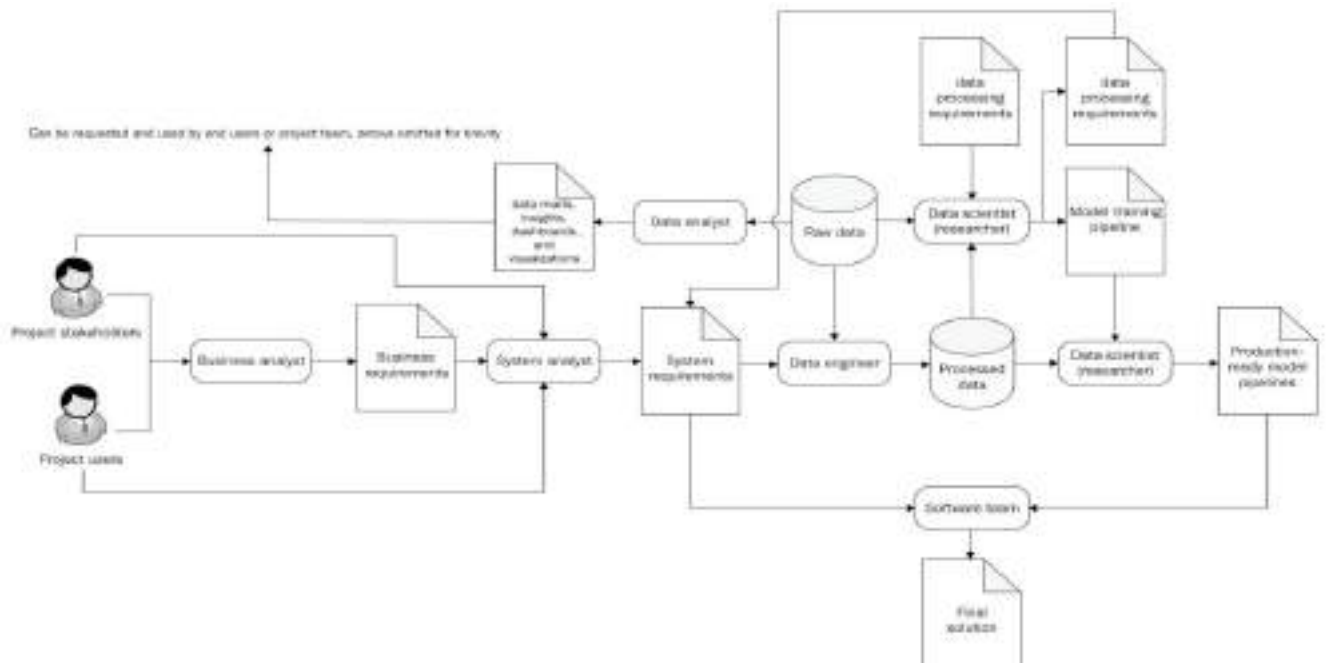
**Data scientists:** They create models, perform statistical analysis, and handle other tasks related to data science. For most projects, it will be sufficient to select and apply existing algorithms. An expert who specializes in applying existing algorithms to solve practical problems is called a ML (DL) engineer. However, some projects may ask for research and the creation of new state-of-the-art models.

For these tasks, a machine or deep learning researcher will be a better fit. For those readers with computer science backgrounds, we can loosely describe the difference between a machine learning engineer and research scientist as the difference between a software engineer and a computer scientist.

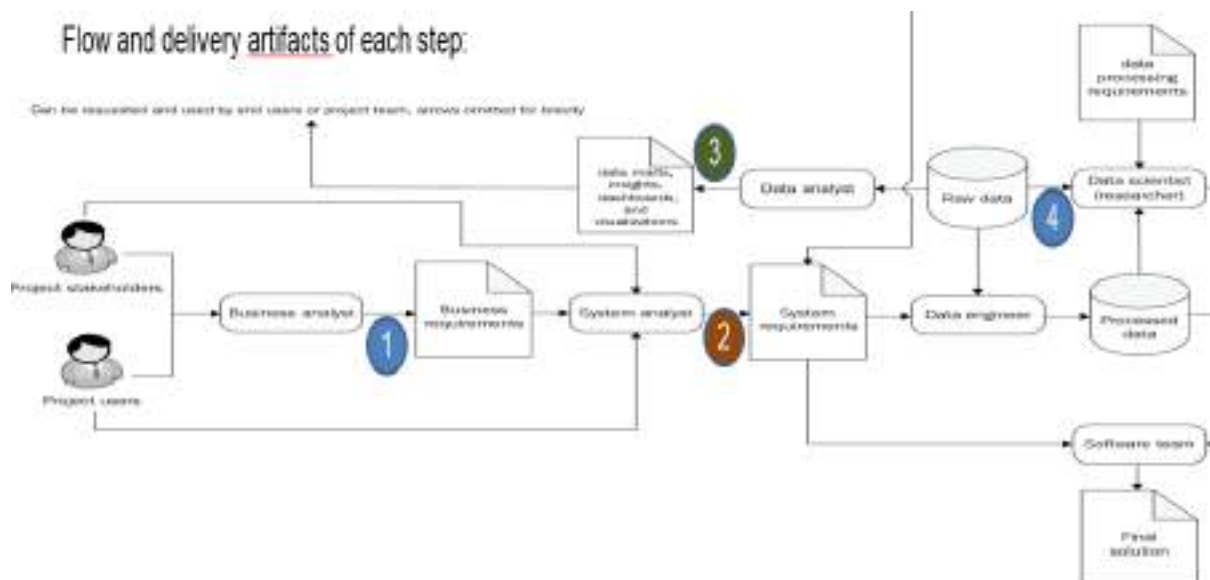
## Topic 17-Individual Responsibilities,

<https://www.youtube.com/watch?v=H7XsWSN45sA&feature=youtu.b>

## Individual Responsibilities



Flow and delivery artifacts of each step:



1. **Business analyst documents business requirements** based on querying project stakeholders and users.
2. **System analyst documents system (technical) requirements** based on business

requirements and querying project stakeholders and users.

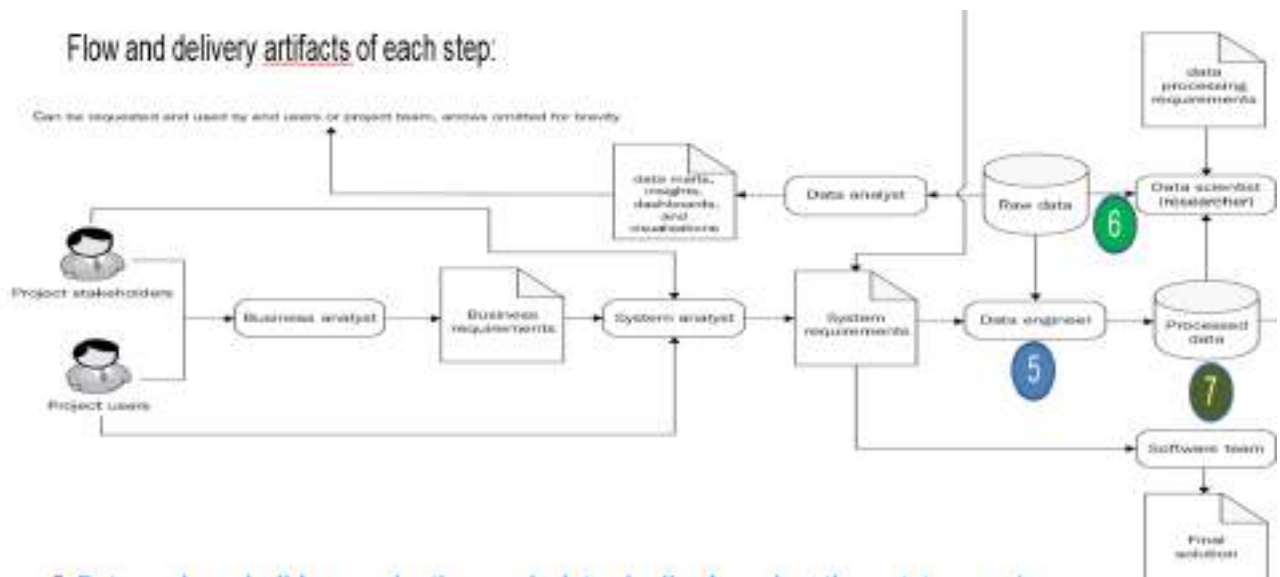
3. **Data analyst supports the team by creating requested data marts and dashboards.** They can be

used by everyone on the team in the development process, as well as in production.

If the data analyst uses a Business Intelligence tool, they can build dashboards directly for end users.

4. **Data scientist (researcher) uses documented requirements and raw data to** build a model training

pipeline and document data processing requirements that should be used to prepare training, validation, and testing datasets.



5. **Data engineer builds a production-ready data pipeline based on the prototype made** in Step 3.

6. **Data scientist (engineer) uses processed data to build a production-ready** model for training and prediction pipelines and all necessary integrations, including model APIs.

7. **Software team uses the complete model training and prediction pipelines to** build the final solution.

### Exploring data science team roles and their responsibilities

To complete a data science project, you will need a data scientist.

### ***Can a single expert lead a project?***

To answer this question, we can break down data science projects into stages and tasks that are, to some extent, present in all projects.

Before starting a project, you need an idea that will allow your client to achieve their goals and simplify their life. In business, you will look to improve key business processes within a company. Sometimes, the idea is already worked out, and you may start directly from Implementation, but more often, your team will be the driver of the process. So, our ideal expert must be able to come up with an idea of a data science project that will provide value for the client.

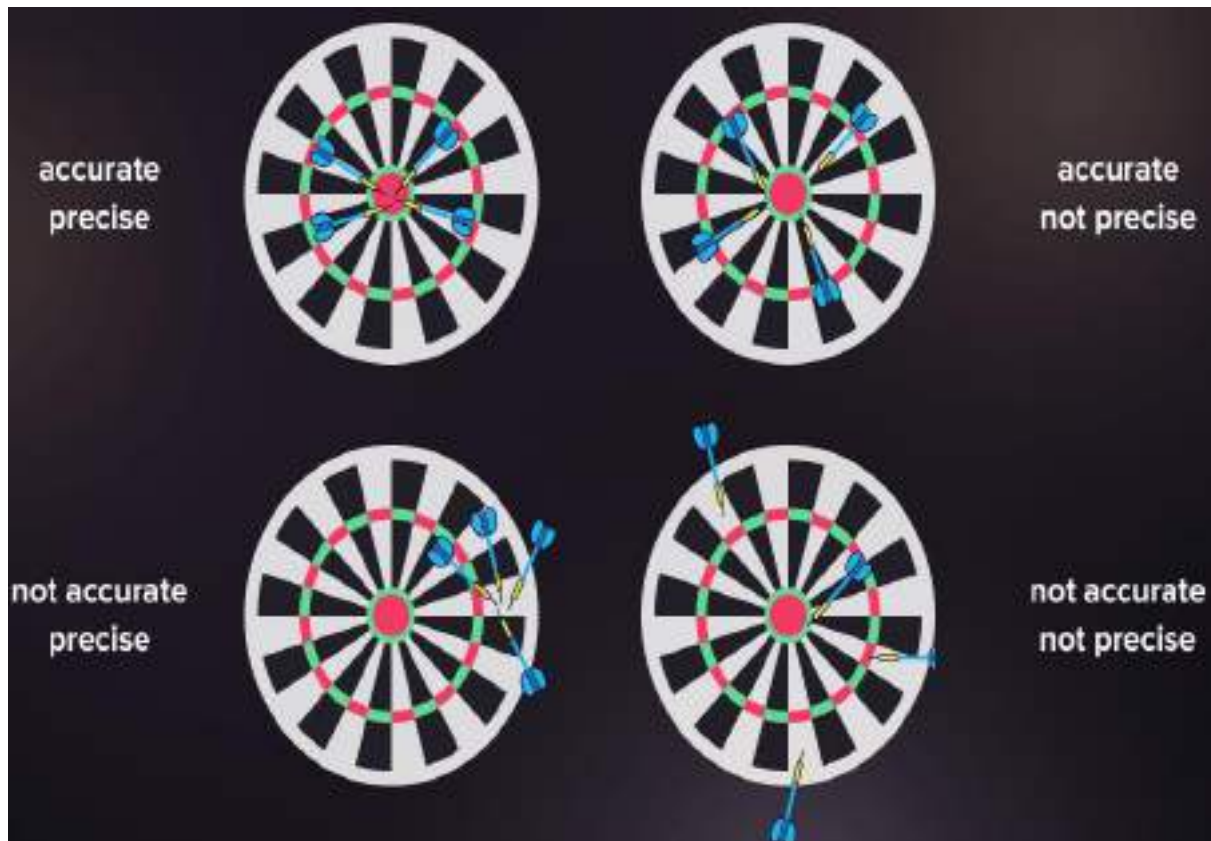
Next, we will study two project examples to look at how simple projects can be handled with small teams or even one cross-functional expert, while larger projects need more diverse teams, where each team member handles one or two specific roles.

#### **Case Study 1 : fraud in banks**

To explore what data science projects can be like, we will look at a case study. Mary is working as a data scientist in a bank where the fraud analysis department became interested in **ML**. **She is experienced in creating ML models** and integrating them into existing systems by building APIs. Mary also has experience in presenting the results of her work to the customer.

One of the main activities of this department is to detect and prevent credit card fraud. They do this by using a rule-based, fraud detection system. This system looks over all credit card transactions happening in the bank and checks whether any series of transactions should be considered fraudulent. Each check is hardcoded and predetermined. They have heard that ML brings benefits over traditional, rule-based, fraud detection systems. So, they have asked Mary to implement a fraud detection model as a plugin for their existing system. Mary has inquired about the datasets and operators of the current fraud detection system, and the department confirmed that they will provide all necessary data from the system itself. The only thing they need is a working model, and a simple software integration. The staffs were already familiar with common classification metrics, so they were advised to use F1-score with k-fold cross-validation.

A F1 score assesses the predictive ability of a model by examining its performance on each class individually rather than considering overall performance like accuracy does. The F1 score combines two competing metrics, precision and recall.



Imagine you are throwing darts at a **bullseye** with the goal of achieving both accuracy and precision, meaning you want to consistently hit the bullseye. Accuracy refers to landing your throws near the **bullseye**, but not necessarily hitting it every time. On the other hand, precision means your throws cluster closely together, but they may not be near the **bullseye**. However, when you are both accurate and precise, your darts will consistently hit the **bullseye**.



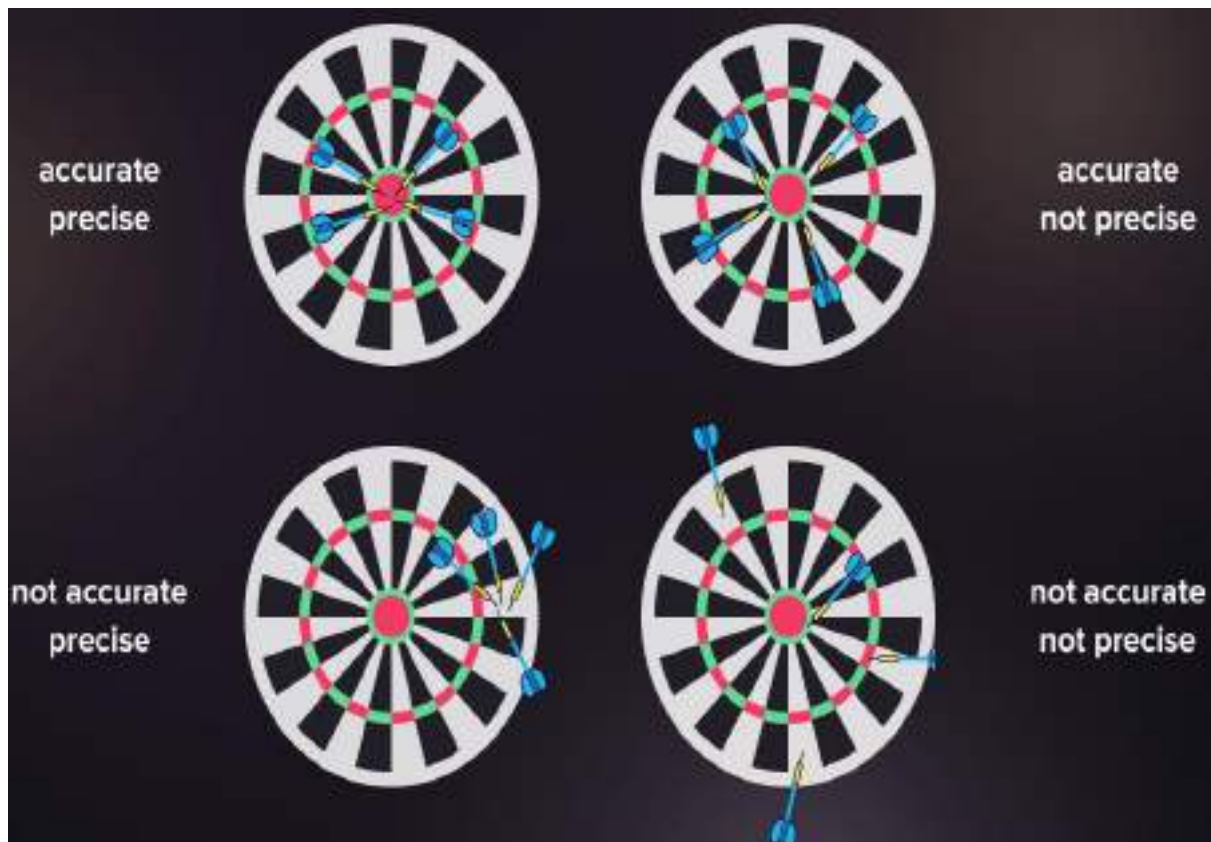
Topic 18-Using F1 Score,

<https://www.youtube.com/watch?v=deXqnkoKbtU>

## MDSS: Session # 18

### USING F1 SCORE AND QUALIFICATIONS OF DS PROFESSIONALS

#### Review of F1 Score in Metrics



Imagine you are throwing darts at a **bullseye** with the goal of achieving both accuracy and precision, meaning you want to consistently hit the bullseye. Accuracy refers to landing your throws near the **bullseye**, but not necessarily hitting it every time. On the other hand, precision means your throws cluster closely together, but they may not be near the **bullseye**. However, when you are both accurate and precise, your darts will consistently hit the **bullseye**.

#### Why use F1 score instead of accuracy?

Accuracy is used when the True Positives and True negatives are more important while

F1-score is used when the **False Negatives** and **False Positives** are crucial. Accuracy can be used when the class distribution is similar while F1-score is a better metric when there are imbalanced classes as in the above case.

		Actual	
		+ve	-ve
Predicted	+ve	TP	FP
	-ve	FN	TN

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

TP: Number of samples *correctly* predicted as “positive.”

FP: Number of samples *wrongly* predicted as “positive.”

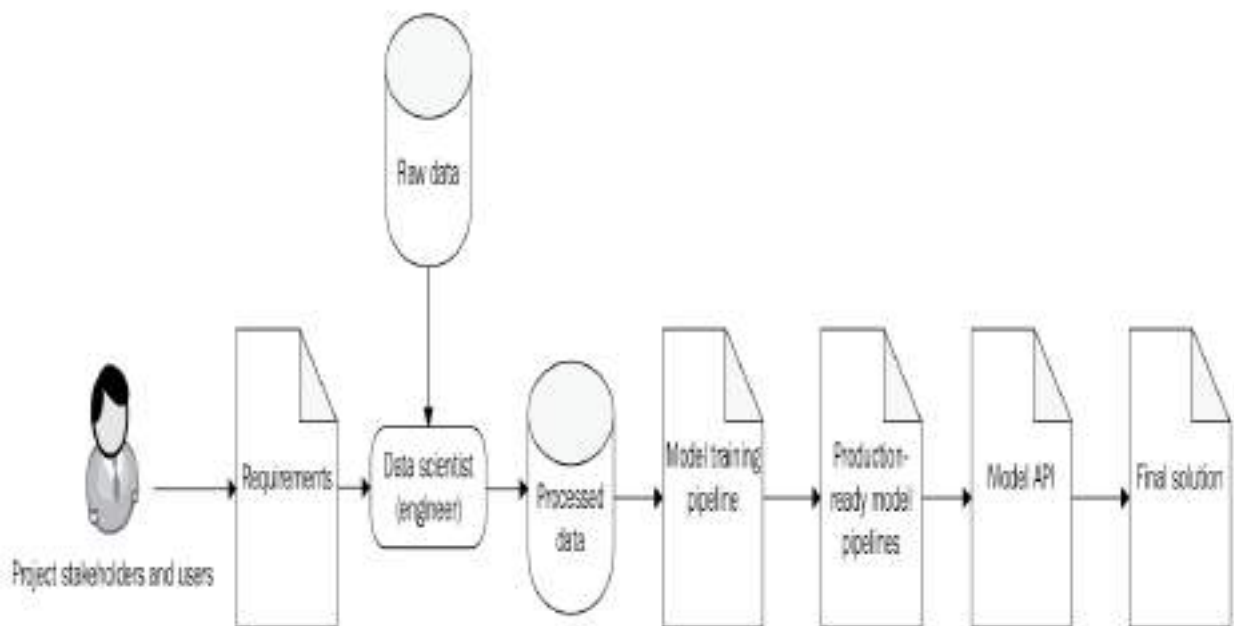
TN: Number of samples *correctly* predicted as “negative.”

FN: Number of samples *wrongly* predicted as “negative

The F1 score is calculated as the harmonic mean of the precision and recall scores. It ranges from 0-100%, and a higher F1 score denotes a better quality classifier.

$$\begin{aligned} \text{F1 Score} &= \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} \\ &= \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \end{aligned}$$

In this project setup, project stakeholders have already finished the business analysis stage. They have an idea and a success criterion. Mary has clean and easy access to the data source from a single system, and stakeholders can define a task in terms of a classification problem. They have also defined a clear way to test the results. The software integration requirements are also simple. Thus, the role flow of the project is simplified to just a few steps, which are all performed by a single data scientist role:



As a result, Mary has laid out the following steps to complete the tasks:

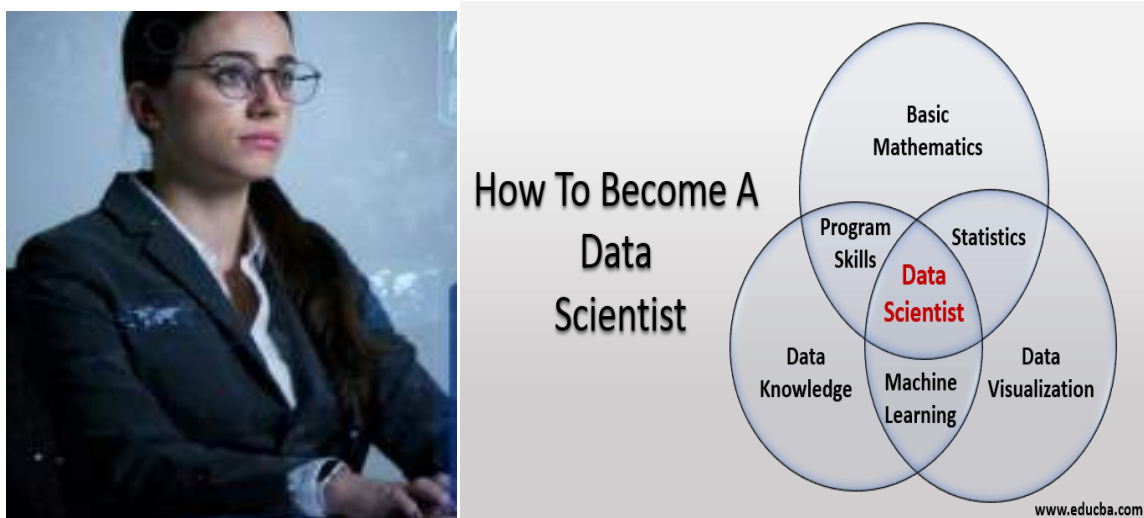
1. Create a machine learning model.
2. Test it.
3. Create a model training pipeline.
4. Create a simple integration API.
5. Document and communicate the results to the customer.

This project looks simple and well-defined. Based on this description, we can believe that Mary can handle it by herself, with little supervision.

### Key skills of a Data Scientist

A data scientist is a relatively new profession, and vague definitions of their responsibilities are common. This uncertainty leads to many problems. Generic position descriptions on job

boards are unappealing to good candidates. Professionals looking for a job won't be able to figure out what's required from them to get this position if you don't crystallize this definition in your head first. If someone goes to a job without knowing what is expected of them, it becomes even worse. Without a clear goal, or at least a clear definition, of the next milestone, your team will get lost. Defining clear responsibilities for team roles is the basis that will allow your team to function coherently.



### Key skills of a Data Scientist (DSct)

A data scientist is a relatively new profession, and *vague* definitions of their responsibilities are common. This uncertainty leads to many problems.

Generic position descriptions on job boards are unappealing to good candidates.

Professionals looking for a job won't be able to figure out what's required from them to get this position if you don't crystallize this definition in your head first.

*If someone goes to a job without knowing what is expected of them, **it becomes even worse!***

Without a clear goal, or at least a clear definition, of the next milestone, your team will get lost.

Therefore, defining clear responsibilities for team roles is the basis that will allow your team to function coherently.

An ideal data scientist is often described as a mix of the following **three things**:

- **Domain expertise:** This includes knowledge of an environment data scientists are working in, such as healthcare, retail, insurance, or finance.
- **Software engineering:** Even the most advanced model won't make a difference if it can only present pure mathematical abstractions. They need to know how to shape their ideas into a usable form.

- **Data science:** Need to be proficient in mathematics, statistics, and one or more key areas of data science, such as ML, DL, or time series analysis.

**IMP :** It is important to understand why each area of expertise is present in this list.

We will start with the **domain** expertise.

*You are very unlikely to find a world-class business expert who can also train state-of-the-art DL. The good thing is that you don't need to search for **unicorns**.*

*A data scientist needs domain expertise mainly to understand data.*

*It is also handy when working with requirements and stakeholder expectations. A basic to medium understanding of insurance business can help data scientists throughout the process of building a model.*

**Example :** How business expertise can help to build models for an insurance company:

- Speaking with insurance experts in the same language can help to discover their pain and desires, so the data scientist can adjust their goals to achieve better results.
- Insurance expertise will help in exploring and understanding the raw data from company's databases.
- Insurance expertise can help in data pre-processing. It aids data scientists in assembling datasets for machine learning models.

Thus, domain expertise is not the prime skill for a data scientist, but a basic to medium level of understanding can improve results by a large margin.

### **Key skills of a Data Engineer**

As a project becomes more complex, managing data becomes difficult. Your system may consume data from many sources, some of them being real-time data streams, while others may be static databases. Volumes of data that your system will need to process can also be large. All this leads to the creation of a data processing subsystem that manages and orchestrates all data streams.

Managing large volumes of data and creating systems that can process large volumes of data quickly requires the usage of highly specialized software stacks suited to this task.

Hence, a separate role of a data engineer emerges. The key areas of knowledge for data engineers are as follows:

- **Software engineering**
- **Big data engineering: This includes distributed data processing frameworks,**

data streaming technologies, and various orchestration frameworks. Data engineers also need to be proficient with the main software architecture patterns related to data processing.

**Database management and data warehousing: Relational, NoSQL, and inmemory** databases.

### **Key skills of a data science manager**

**Management:** A data science team manager should have a good understanding of the main software management methodologies, such as SCRUM and Kanban. They should also know approaches and specific strategies for managing data science projects.

**Domain expertise:** They should know the business domain well. Without it, task decomposition and prioritization becomes impossible, and the project will inevitably go astray.

**Data science: A good understanding of the basic concepts behind data science** and machine learning is essential. Without it, you will be building a house without knowing what a house is. It will streamline communication and help to create good task decompositions and stories.

### **Key skills of a data science manager**

**Management:** A data science team manager should have a good understanding of the main software management methodologies, such as SCRUM and Kanban. They should also know approaches and specific strategies for managing data science projects.

**Domain expertise:** They should know the business domain well. Without it, task decomposition and prioritization becomes impossible, and the project will inevitably go astray.

**Data science: A good understanding of the basic concepts behind data science** and machine learning is essential. Without it, you will be building a house without knowing what a house is. It will streamline communication and help to create good task decompositions and stories.

Topic 21-Discovering the Purpose of Interview Process,  
<https://www.youtube.com/watch?v=bongqMusdBfg>



## DISCOVERING THE PURPOSE OF THE INTERVIEW SCORE PROCESS

First, you should figure out why you need to do interviews. Let's do a quick exercise. Ask yourself why and write out answers until there is nothing left to say. After that, recall the last interview that you did. Was it well aligned with your goals?

The results should give you an instant vector for improvement.

From the employer's side, the sole purpose of the interview is to fill the position with a capable candidate.

From the side of the employee, the purpose of the interview is to find a good team, an interesting project, a reliable company to work with, and satisfactory compensation.

We often forget that the interview is a dialogue, not a mere test of skill.

Unfortunately, there is no clear and ubiquitous definition of the main interview goal, i.e., because each goal is unique for each position.

You should define this goal based on the detailed understanding of the job you are interviewing for.

If you are looking for a data scientist, define rigorously what you expect them to do.

What tasks will the new team member solve?

Think deeply about what the first working day in this position will look like.

What are the core responsibilities?

What skills are useful but not mandatory?

Based on that, you can start drafting a set of desired skills for the candidates. Be specific. If you demand knowledge of SQL, then be sure to have a definite reason for this requirement.

If your interview goal is to find an ideal, top 0.1 percentile, world-class expert, then think again.

The primary goal is to find an able person who can do the job. It might be that this person must be the best expert in the world. If so, this requirement should be explained and justified. To create this understanding, you need to describe what the job will be like.

After you decide that the description is finished, rewrite it with the candidate in mind. It should be short, easy to read, and give a complete idea of that the job will be like and what the

expectations are. The ambiguity of job descriptions often comes from a lack of this understanding. We just need someone; their job description says. Yours should say We know exactly what we need, and you can rely on us. If you can't, do you really need to post a new job?

**Concrete understanding of your goals will be helpful in several ways:**

- You will understand what kind of expert you should search for.
- Candidates looking at your job description will know exactly what you expect of them.

It will be easier to judge whether their experience is relevant or not.

- Clear goal definitions will allow you to design purposeful and insightful interview questions.

If your requirements list looks too long, try to simplify it. Maybe you were overzealous and some skills you have listed are not vital.

Or perhaps you could consider hiring two people with different backgrounds?

If you are posting a job description with a wide cross functional set of requirements, be sure to have a reason behind it.

If your job description is complex, then you will find the following is true:

- ❖ The candidate search will take longer since you are demanding more skills and experience.
- ❖ The candidates will expect a higher salary.
- ❖ The interview process will be long and could extend to several sessions.
- ❖ You will have fewer options to choose from.

All those restrictions should have good supporting arguments.

***If you have none, consider simplifying.***

***Having a good goal is not all you need, the execution matters too.***

We will now look at the interview process from the candidate's standpoint to discuss the topics of bias and empathy.

## Introducing values and ethics into the interview

We often look at interviews from a one-sided perspective. We want to find a reliable team member who can do the job well. It is easy to forget that bad interviews can scare good candidates. More so, a constant flow of potential candidates makes judgments about your company through the interviews. Bad interviews lead to bad publicity. The key to effective and smooth interviews is to think about the candidate experience.

If you have successfully defined honest requirements for the candidate, the next step is to think about the interview experience as a whole. To see a new teammate in the best light, the interview should look like a real working process, not like a graduate examination.

Consider how most technical interviews are conducted:

- You go through a resume filter. The best people have **PhDs** and **at least 5 years** of (questionable) experience in major tech companies.

- You finish some kind of online test for your technical skills.

- You submit your GitHub repositories and portfolio projects for review.

- Then, you go through an online screening interview. You answer a series of technical questions related to a multitude of technologies and scientific areas that might be useful in your future work.

End of session 21

Topic 22-Introducing Values and Ethics into Interview Process,  
<https://www.youtube.com/watch?v= 2ADgfHZL1M>

### Introducing values and ethics into the interview

We often look at interviews from a one-sided perspective. We want to find a reliable team member who can do the job well. It is easy to forget that bad interviews can scare good candidates. More so, a constant flow of potential candidates makes judgments about your company through the interviews. Bad interviews lead to bad publicity. The key to effective and smooth interviews is to think about the candidate experience.

If you have successfully defined honest requirements for the candidate, the next step is to think about the interview experience as a whole. To see a new teammate in the best light, the interview should look like a real working process, not like a graduate examination.

Consider how most technical interviews are conducted:

- You go through a resume filter. The best people have **PhDs** and **at least 5 years of** (questionable) experience in major tech companies.
- You finish some kind of online test for your technical skills.
- You submit your GitHub repositories and portfolio projects for review.
- Then, you go through an online screening interview. You answer a series of technical questions related to a multitude of technologies and scientific areas that might be useful in your future work.
- Next, you find yourself in a room with a whiteboard, where you answer a set of complicated and very specific problems on software development, software architecture, machine learning, combinatorial optimization, category theory, and other important topics that are used to select the best from the best.

- Multiple people, each an expert in their field, will test you for different skills. It is likely that the interviewers will test you with tricky problems from their experience, each of which took them days to tackle.
- Finally, a company executive will see if you are a good cultural fit and go along with the company values and goals.

The interview process must respect the candidate's emotional condition and treat them as a human. It should be humanistic. The moment you start doing this, the potential candidate lists will grow and the interviews will be shorter and more effective.

Be open-minded and clear yourself of prejudices. Some of the most talented and successful software engineers, software architects, and data scientists I have worked with had unrelated job experiences or a lack of formal education. I am not stating that those things do not matter. They do, but they are not determining factors. Correlation does not mean causation. By rejecting candidates using unnecessary pre-screening filters, you miss opportunities to meet talented people who will stay with you for many years. Of course, some organizations need pre-screening filters. For example, you would not hire a surgeon without a diploma. In some fields, such as medicine, education gives essential experience that you can't get by yourself. But the situation changes drastically with fields such as software engineering and data science, where the knowledge and experience is accessible with a few keystrokes.

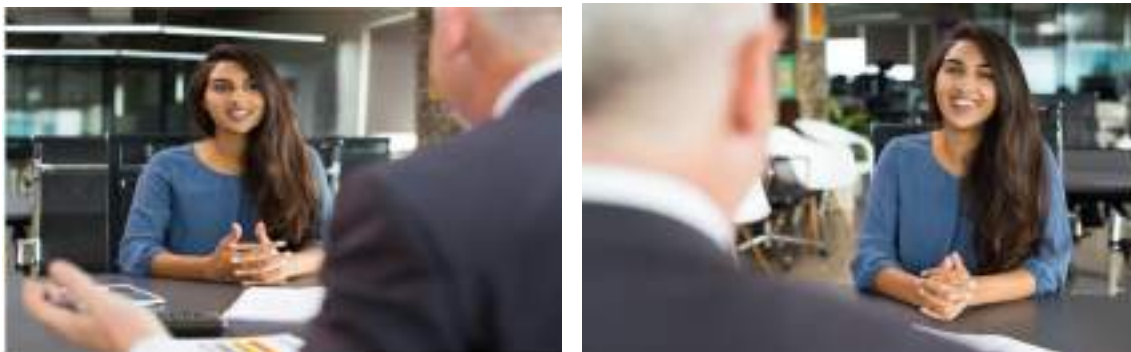
At last, remember to be human. Be nice to people you are talking with. Create a pleasant experience and you will get much better results. This can be hard if you are interviewing people for the whole day, so it is better to limit yourself to one or two interviews per day at the most.

You may think such interviews will take an unreasonable amount of your time and looking at every candidate is impossible. In fact, it is the opposite. The key to designing humanistic interviews is to shift the focus from your organization to the candidate, and we will explore ways to do this in the next section.

## Designing good interviews

How can we make interviews more relevant and time-intensive? An ideal interview would be testing in a real working environment for a couple of months. While there are several companies that can afford to operate without interviews, using paid probation periods instead, this is a very costly hiring strategy that not every business can afford. A good interview should serve as a substitute for real working experience. It should not be a test of a person's skill, but a test for a person's ability to perform a specific job well. If an ideal test for a candidate is a probation period, then the worst kind is a whiteboard interview (unless you are interviewing a computer science lecturer). To design a great interview, bring it as close as possible to your working process and to the issues you solve on a daily basis.

## Designing test assignments



Of course, there might have been a couple of times when you or your colleagues in the same position needed to prove a theorem or implement a complex algorithm from scratch, but is it enough to include this problem in your interviewing process?

If your team has encountered this problem only a few times, it deserves to be a supplementary question at most. Create most of the interview questions from the most frequent problems that your team solves daily.

By no means search the web for interview questions. Your interview should be as specific as possible and generic questions will make it dull and unrelated to what you are searching for.

A frequent source of questions is interviewer's own knowledge resources. You may have read many intricate & interesting books and blog posts. If the knowledge you gained from those materials, no matter how intellectually valuable, is unrelated to the candidate's job, please keep it out of the interview.

Imagine a ML engineer's interview where the candidate gets a question about mathematical details of the reparametrization trick in variational auto encoders when his job will be about delivering pretrained classifiers to production as scalable services.

This case might seem unrealistic to you, but it may happen to you!

After some initial technical questions, the interviewer started asking about the subtleties of native code integration in Java applications:

Developer: *Do you integrate native code into your applications?*

**Interviewer: No.**

**Developer:** *That's good. Because I do not really know much about this topic.*

Interviewer: *I'd like to ask about it anyway.*

**Developer:** *But I know little about native code integration.*

*And, as you said, this is not relevant. Could we move on?*

**Interviewer asks the question.**

This question was bad, no matter how you look at it:

It was irrelevant to the skill requirements in this position.

It was irrelevant to the company.

The interviewer already knew that the candidate could not provide an answer.



It increased the stress factor of the interview, possibly affecting the answers to the more relevant questions that followed.

It diminished the value of this company for the candidate.

On the contrary, a good interview question should be the following:

- Related to the problems that the candidate will solve in his/her job
- As hard or as simple as the candidate's daily working experience

The largest problem of any interview question is its time span. We expect a typical interview problem to be solved in less than 10 minutes. The real working environment does not have such time constraints. To narrow the gap, you can create a test assignment. Ideally, this assignment should be implementable in 2 to 4 hours.

To create a test assignment, do the following:

1. Take a core task that is performed by similar roles (or directly by teammates with the same title) in your company.
2. Then, describe this task in one paragraph. You want your task description to be **SMART (Specific, Measurable, Achievable, Relevant, and Time bound)**.
3. Try to estimate how long it will take to implement. Since you are basing it on a real working experience, it could take days or weeks to finish the task. Only very dedicated candidates will be ready to take on an assignment with such a time span. You can shorten its length by decomposing it into several tasks and taking only the most important ones as a part of the assignment. You can transform all additional tasks into questions that you will ask while examining the test assignment results.

Topic 23- Building a DS Team,

<https://www.youtube.com/watch?v=Oylqz6k3DKI>

## MDSS : Chapter 6

### Building Your Data Science Team

#### You will LEARN

- About 3 key aspects of building a successful team & explain the role of a leader in a DS team.
- How to build and sustain a balanced team that's capable of delivering end-to-end solutions.

Then, we will set out a plan that we will improve and add several useful management skills to our arsenal:

leadership by example, active listening, empathy and emotional quotient, trust, and delegation.

#### Topics covered:

- ▶ Achieving team Zen
- ▶ Leadership and people management
- ▶ Facilitating a growth mind set

#### Achieving Team Zen

A balanced team solves all incoming tasks efficiently and effortlessly. Each team member complements others and helps others to find a solution with their unique capabilities. However, balanced teams do not just magically come into existence when a skillful leader assembles their elite squad.



Team Zen is a collaboration of expertise that enables clients to realize the potential of their human capital.

Finding team Zen is not a momentary achievement; you need to work to find it.

It may seem that some teams are stellar performers while others are underwhelming.

It is crucial to realize that team performance is not a constant state.

The best teams can become the worst, and vice versa. Each team should work to improve. Some teams may need less work, while others will be harder to change, but no team is hopeless.

So, what makes a balanced team? Each team member is unique in their soft and hard skill sets.

Some people are a universal jack of all trades, while others are highly specialized in a single area of expertise.

**Also, we all have vast emotional differences.**

In a balanced team, everyone complements and augments each other. Each team member knows at what point they can participate to yield the best results.

Balanced teams are self-organizing. They can efficiently plan and act without direct interventions from the team leader.

The leader unites the team and makes it coherent, but the actual work process of a balanced team looks highly decentralized.

The leader frequently takes part in teamwork and helps everyone reach a common goal. The team members have good relationships and want to advance themselves to new heights. Another feature of balanced teams is that they do not have a single point of failure. If someone is ill or has decided to move on, it is not a big deal regarding how the core team functions.

Unbalanced teams can be hard to distinguish at first glance. An unbalanced team can function properly and even be a top performer, but unbalanced teams can be very efficient but fragile. People do not stick around long in an unbalanced team because it is hard to do so. Unbalanced teams have many forms, so they are hard to define.

Unbalanced teams are those where role preferences congregate in one part of the Wheel.

Ex. Thruster- Organizer, Concluder-Producer and Assessor-Developer.

**Case I : Expertise imbalance in a team.**

The team leader is the best expert in the team. They do most of the important work, while other team members work on the simplest delegated

tasks. The team leader decomposes the task into many simple subtasks and assigns them to their team. No one sees the whole picture but them. The unwillingness to share important parts of the work and to involve the team in the task's decomposition process creates a strong imbalance.

The **team is very fragile** and will not function properly without its leader. Employees in this team burn out quickly because they do not see the result of their work. Most of the tasks are boring and repetitive, so they seek challenges in another team, project, or company. This is a case of expertise imbalance in a team.

Case-II Situational Leadership, Leading by Example and Empathy,  
<https://www.youtube.com/watch?v=gZMQaZ7BZHg>

## MDSS : Chapter 6

### Building Your Data Science Team

Case 11 : Cultural Imbalance Team.

The team is building a very large project. The development has been going on for 3 years. The team leader and several data scientists who were working on the project from the start developed strong relationships and a team culture. However, not all of the team members stayed with them for the whole 3 years, so the team leader found several new experts eager to take on the project. However, these new team members found it hard to assimilate into the team. The core team unwillingly communicates with the new team. They find it hard to do so because of the cultural gap. It is a lot easier to interact with familiar people than with the new team. Thus, cultural imbalance has emerged.

Another important component of **each team is its roles**. Roles define sets of activities that team members can perform. You may think you do not need roles because your team is agile and cross-functional, so everyone can and should be able to do everything. This may be crucial for small teams to survive while being adroit at what they do. But being cross-functional does not contradict having clearly defined roles. In fact, cross-functionality and size are different dimensions of a team, with the latter affecting the former through the number of roles each team member performance.

E.g., Our DS team should handle everything from the definition of business requirements to the delivery of the final solution. We have loosely defined the project delivery process and assigned each stage to a role:

- **Defining business requirements: BA**
- **Defining functional requirements: SA**
- **Defining nonfunctional requirements: SA**
- **Discovering and documenting incoming data sources, creating data marts, and reports:**  
**Data analyst**
- **Exploratory data analysis and modeling: Data scientist**
- **Building software around the model: Software Engineer**

- **Creating documentation for the final product: Software engineer and systems analyst**
- **Delivery and deployment: DevOps engineer and software engineer**
- **Management and team leadership: Team leader**

Suppose that this team is limited to only 3 members (2+1). We can combine several roles into one to fulfill the team size constraint, while simultaneously creating goals for individual team members.

Recall that forming goals for each position is the foundation of a good hiring process, and creating a team goal, defining roles, and mapping them to positions is a prerequisite for that.

For our example, there are many options, but the most realistic would be the following:

1. **Defining business, functional, and nonfunctional requirements: Analyst**
2. **Delivering software and models: Machine learning engineer**
3. **Project management, team leadership, and the acceptance of final and intermediate results: Team leader**

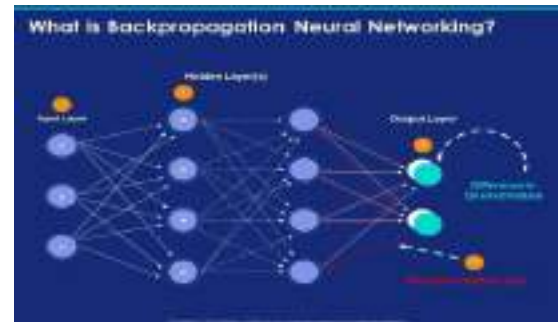
Creating individual goals that are aligned with team goals can be done through the following set of transformations:

Team goal → Team roles → Positions → Individual goals

The more roles your team has, and the smaller its size, the vaguer the boundaries between the team members will be, and the harder it will be to find teammates to fill in those roles.

Defining roles helps you to be realistic regarding the amount of work and the budget it will take you to create a good team, so never leave this step out. Mapping roles to positions is equally helpful if your team is large and the responsibilities of every employee are clearly isolated or if your team is small and responsibilities are less segregated.

By following this process, you can create a feedback system that will optimize not only global team goals but also individual goals, providing everyone with an understanding of what their part in the big picture is. When designing deep neural networks, you need to create a loss function and make sure that the gradient passes through each layer. When building teams, you need to make sure that feedback passes through each part of your team, starting from yourself.



Understanding **roles**, **goals**, and **size** constraints will also help you grow and develop your team. Over time, your team will grow, but if you maintain the core **goal-role-position framework**, adding new responsibilities and team members will be straightforward with growth, and you will assign positions to fewer roles.

*Goal-position-role maintenance and revision is also necessary to keep your team balanced. Changes in organizational structure or in the main goals of a team always bring a risk of creating an imbalance, so be careful when introducing changes and plan ahead.*

The concepts of team balance and a feedback system are simple but powerful tools that will help you build a balanced team and scale it in the right way.

## Leadership and People Management

Leadership and people management are related, as people management **involves selecting, training, and motivating employees to improve their skills and output**. Leaders need to be able to motivate, inspire, and delegate tasks, as well as communicate effectively and resolve conflicts.



Team Zen

*that a team leader should not be at the core of*

the team because this situation leads to severe organizational imbalance.

Nonetheless, a team leader should be everywhere and nowhere at the same time.



A team leader should facilitate team functioning, help every team member take part in the process, and mitigate any risks that threaten to disrupt the team functions before they become real issues.

A good team leader can substitute and provide support for all or most of the roles in their team.

They should have good expertise regarding the core roles of the team so that they can be helpful in as many of the team activities as possible.

The team leader should make sure that information propagates through the team effortlessly, and that all communication takes place as needed. If we take the internet as an analogy, the team leader should provide a fast and reliable communication network for all the servers.

Another important activity for a team leader is expanding the team constantly and in a balanced manner.

All team members should have sufficient motivation, be challenged, and have a clear understanding of how they can grow according to their individual goals.

Most importantly, a team leader should set goals and achieve those goals together with their team. The team leader should unify everyone's efforts into a single focal point and make sure that focal point is the right one.

### **Leading by example**

The most simple and effective leadership advice was likely given to you in childhood:

if you want to have good relationships with others, take the first step:

- ❖ Want people to trust you?
- ❖ Build incrementally by trusting other people.
- ❖ Want your team to be motivated?
- ❖ Be involved in the workflow and give everyone a helping hand; show them progress.
- ❖ Management literature calls this principle leadership by example.

- ❖ Walk your talk, and everyone will follow the example.



This kind of leadership is very effective for team building:

- ❖ It does not require you to learn and apply complex theoretical methodologies
- ❖ It builds relationships and trust in your team
- ❖ It ensures that the team leader is involved in the workflow of every role in the team
- ❖ It helps everyone grow professionally, including yourself

***The only downside of leadership by example is that it can be hard. It requires you to be deeply involved in your team—emotionally and intellectually—and this can be debilitating.***

It is important to avoid exhaustion because your team will feel it instantly. If you burn out,

so will everyone on your team. Thankfully, there is a simple way to avoid leadership burnout. Make sure that you have time during your workday that allows you to take a rest from team leadership activities. Maybe it is a small research project that will help your team members or maybe it is a prototype you can handle yourself.

### **Using Situational Leadership**

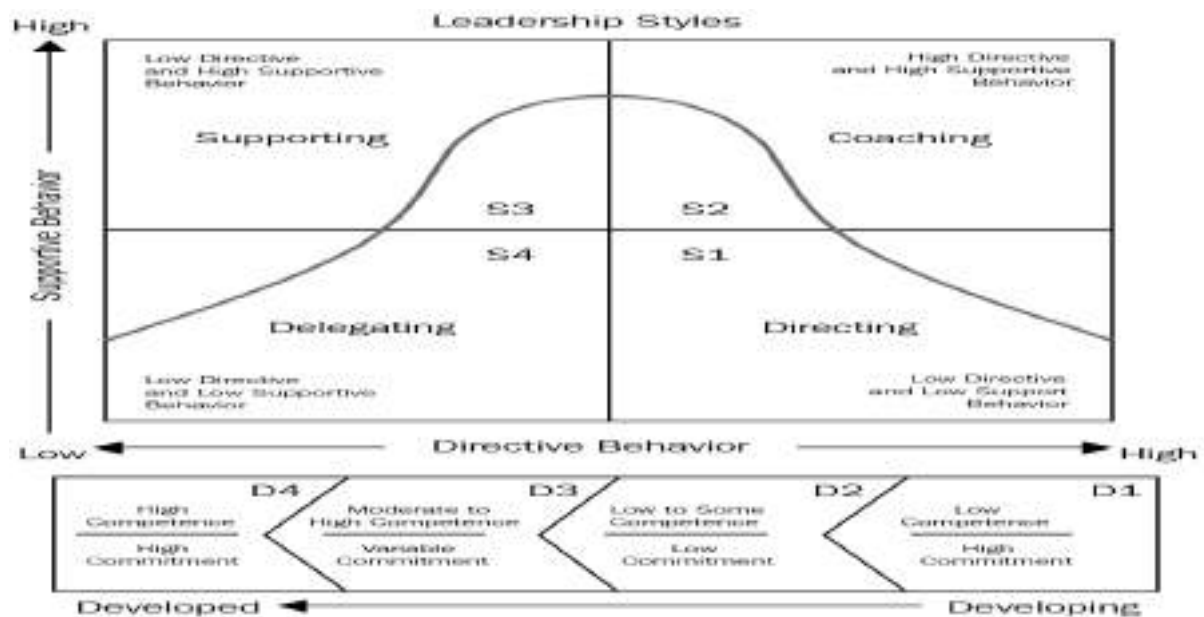
Thankfully, in most cases, problems regarding task delegation are mostly emotional. **To delegate without fear, you should build trust.** And to build trust, you need to take the first step. If you find it hard to delegate, just force yourself to try it. It may seem like a leap of faith, but jump and see what your team is capable of.

At first, you may be scared, but I assure you, they will surprise you. The tricky part of delegation is the task description step.

Delegation will cause failure if you cannot describe what needs to be done correctly and without over-complicating the matter. To delegate effectively, you can make use of situational leadership. At its core,

it is a very simple concept: you must measure your teammate's competence and commitment levels.

Then, you need to choose your leadership style according to those levels.



Commitment stands for the employee's level of motivation. People with high commitment levels find their work interesting and are ready to try hard until the task is completed. A low commitment level, on the other hand, indicates that the person is not very interested in completing a task, finds it boring or repetitive, or is close to burning out.

The other side of the coin is competence, which indicates the employee's ability to handle tasks by themselves. Low competence indicates that the person has no or low prior experience in the area, regarding the task they need to complete. A high competency level indicates that the employee has all the necessary knowledge and experience to complete the task. Situational leadership helps you keep motivation levels stable while ensuring that the employee gets all the necessary instructions so that their competence can grow without harming the task's resulting quality.

The following example shows how we can apply the situational leadership technique by determining an employee's competence and commitment before delegating a task:

1. Imagine that you've got a new team member, Jane. She does not have much experience and is taking a junior position in your team. Her motivation is high, and she is eager to learn new things. Here, the commitment level is high and the competence level is low, so we must take a directive leadership style. Obviously, she cannot solve complex tasks alone. It is better to take something simple, split the task into as many subtasks as you can, thoroughly describe everything, and Check the correctness of every step. Employees from this category will take up a lot of your time, so be aware of your own resources. A team of 10 high commitment- low-competency people can easily siphon all of your time.

2. Over some time, Jane has gained some skills, but her motivation has started to fall recently. This is the perfect time to coach. At this stage, you want to loosen control a bit to allow your teammate to make commitments.

3. Jane's competency levels have grown further, and you can tell that she has somewhere between a moderate to a high level of technical skill compared to others in your team. It is important to catch this transition because Jane's motivation is at risk again. To keep it up, you will want to change your leadership style to one that's supportive and focuses less on particular assignments. You will also want to share decision-making tasks with her. Jane's performance will probably be inconsistent at this stage, so provide any support she needs and help her transition into the last stage.

4. At this stage, Jane is already a senior expert. She is highly skilled, can make good decisions, and can provide a consistent output of good quality. This is the stage when you can delegate tasks. Your main goals at this stage are to set high-level goals, monitor progress, and take part in high-level decision-making.

3. Jane's competency levels have grown further, and you can tell that she has somewhere between a moderate to a high level of technical skill compared to others in your team. It is important to catch this transition because Jane's motivation is at risk again. To keep it up, you will want to change your leadership style to one that's supportive and focuses less on particular

assignments. You will also want to share decision-making tasks with her. Jane's performance will probably be inconsistent at this stage, so provide any support she needs and help her transition into the last stage.

4. At this stage, Jane is already a senior expert. She is highly skilled, can make good decisions, and can provide a consistent output of good quality. This is the stage when you can delegate tasks. Your

main goals at this stage are to set high-level goals, monitor progress, and take part in high-level decision-making.

### **Developing empathy**

Basic concepts such as leadership by example, situational leadership, and the SMART criteria will help you structure and measure your work as a team leader. However, there is another crucial component, without which your team can fall apart, even with the perfect execution of formal leadership functions. This component is empathy.

Being empathetic means to understand your emotions, as well as other people's emotions. Our reactions are mostly irrational, and we often confuse our own feelings and emotions.

For example, it is easy to confuse anger for fear; we can act angrily when in reality we are just afraid. Understanding your own emotions and learning to recognize others will help you understand people and see subtleties and motives behind their actions. Empathy helps us find the logic behind irrational actions so that we can react properly. It may be easy to answer with anger in regard to aggressive behavior unless you can understand the motives behind those actions. If you can see those motives, acting angrily may seem foolish in the end. Empathy is the ultimate conflict-resolution tool; it helps build trust and coherency in the team.

Another important tool that will help you build empathy and understand others is **active**

**listening.** We listen to others each day, but do we extract all the available information from their speech? In fact, we leave most of it behind. When listening, we are often distracted by our own thoughts. We do not pay attention to the speaker and are eager to talk.

To listen actively, do the following:

**Pay full attention to the speaker:** We often drift into our own thoughts when others speak for long periods of time. The speaker will immediately notice that you are in the clouds, and it will negatively affect their urge to share information with you.

**Show that you are paying attention:** Give feedback using *uh-huhs*, *saying yes*, by nodding, and by smiling. The other person shouldn't feel like they are talking to a concrete wall.