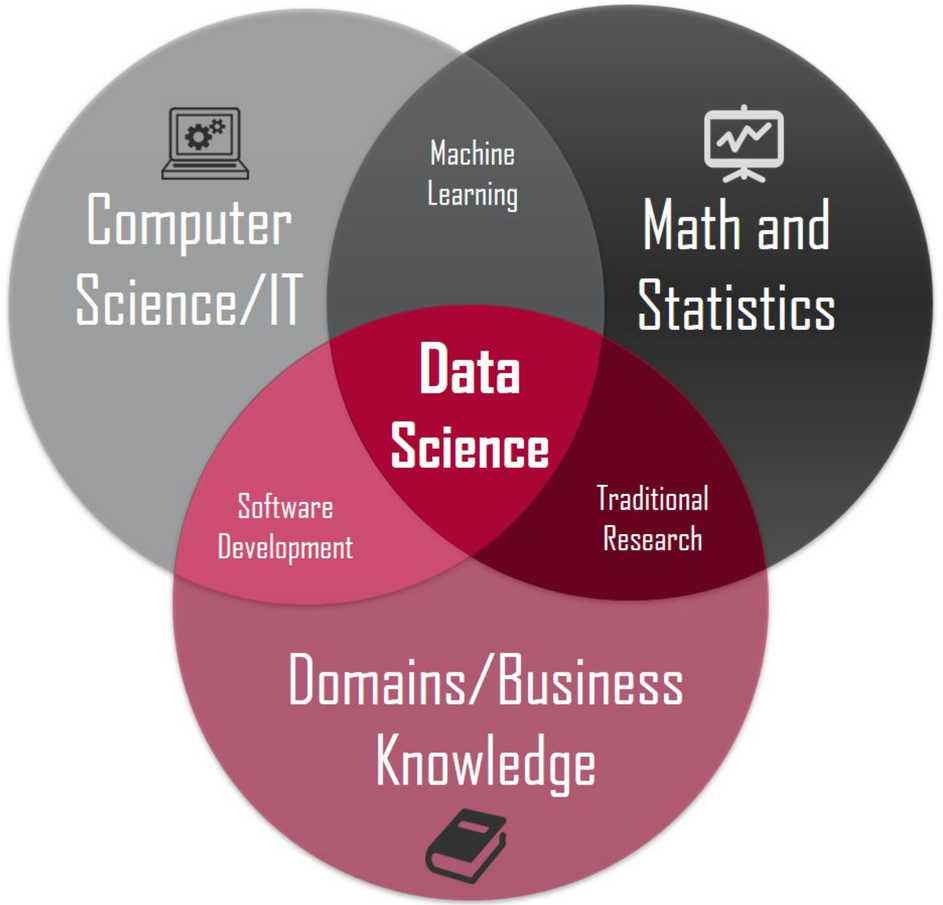
Intro to Machine Learning

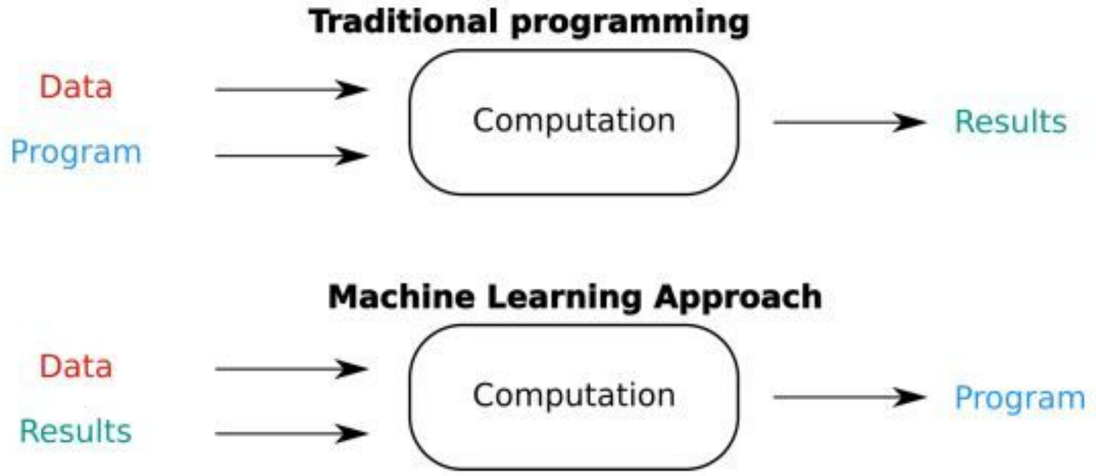
Providing universal access to AI education and practice

What is **Data Science?**

Data science, also known as data-driven science, is an interdisciplinary field about scientific methods, processes and systems to extract knowledge or insights from data in various forms, either structured or unstructured.



What is **Data Science?**



How is the traditional flow of **Data Science?**

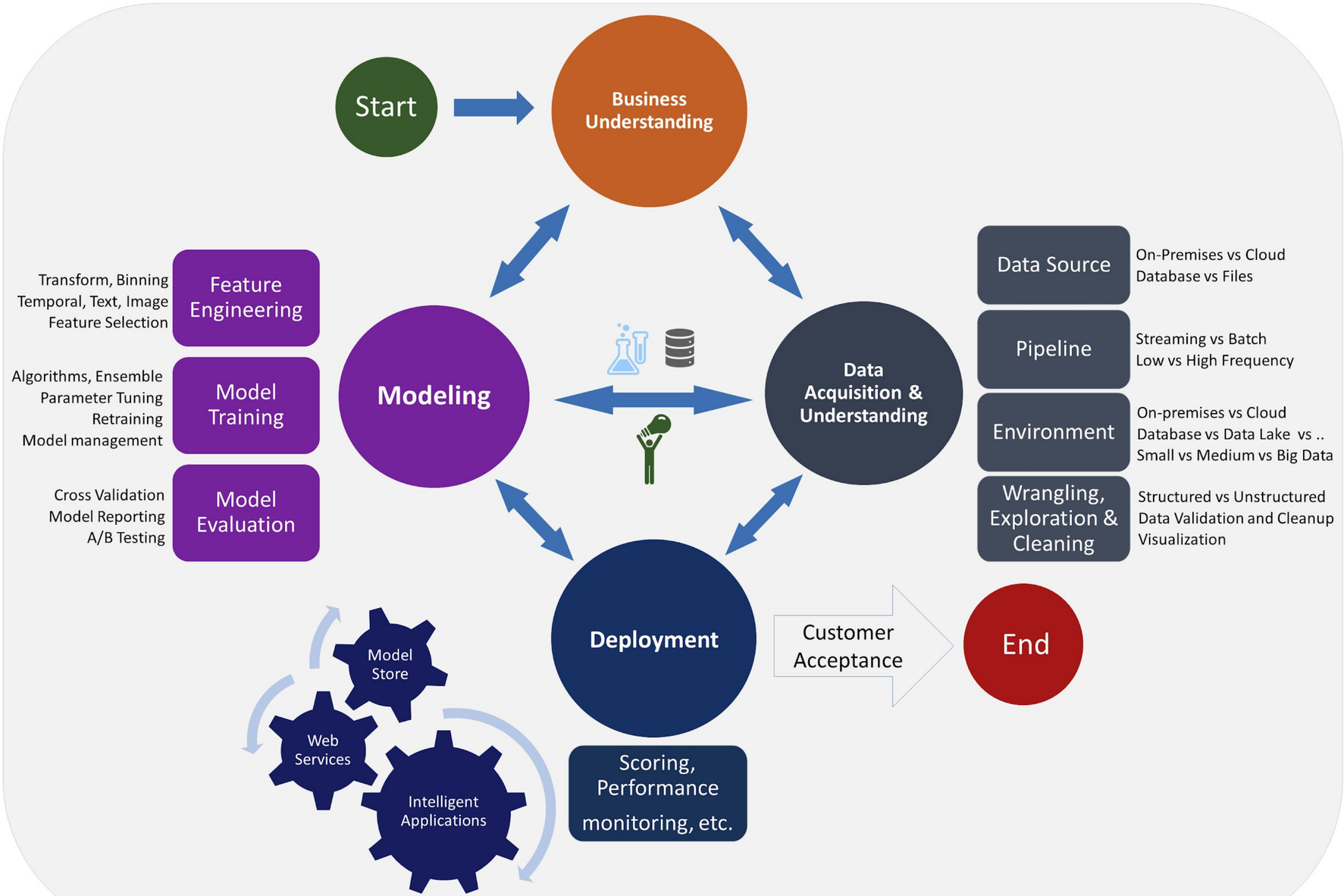
1. Business Problem
2. Data Acquisition
3. Data Preparation

.

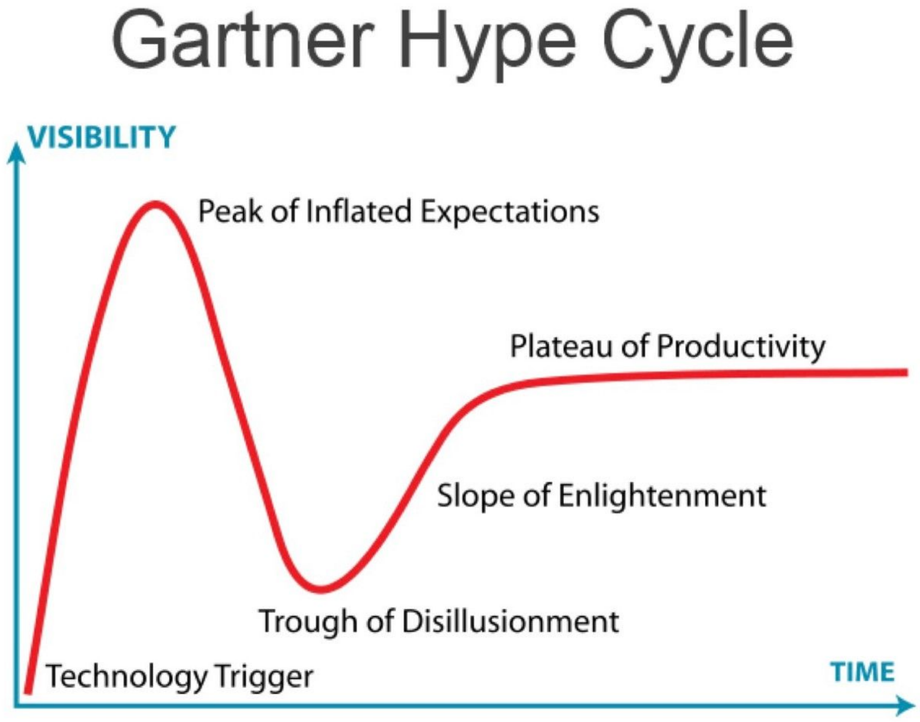
.

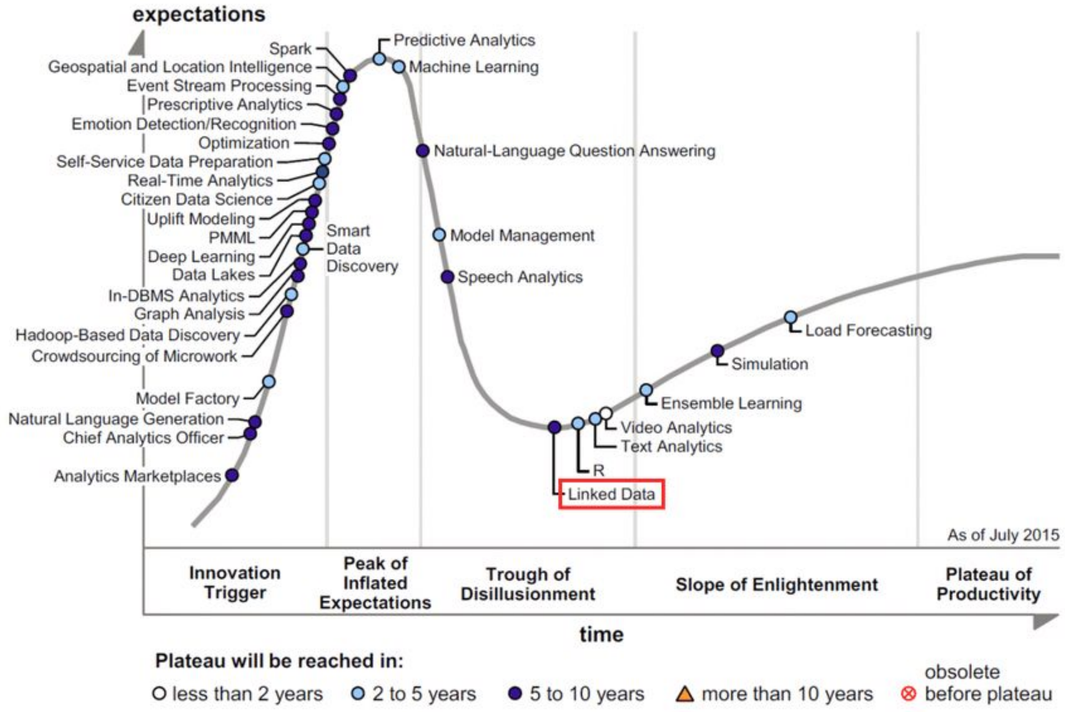
.

4. Data Analysis
5. Data Modelling
6. Visualization and Communication
7. Deployment and maintenance

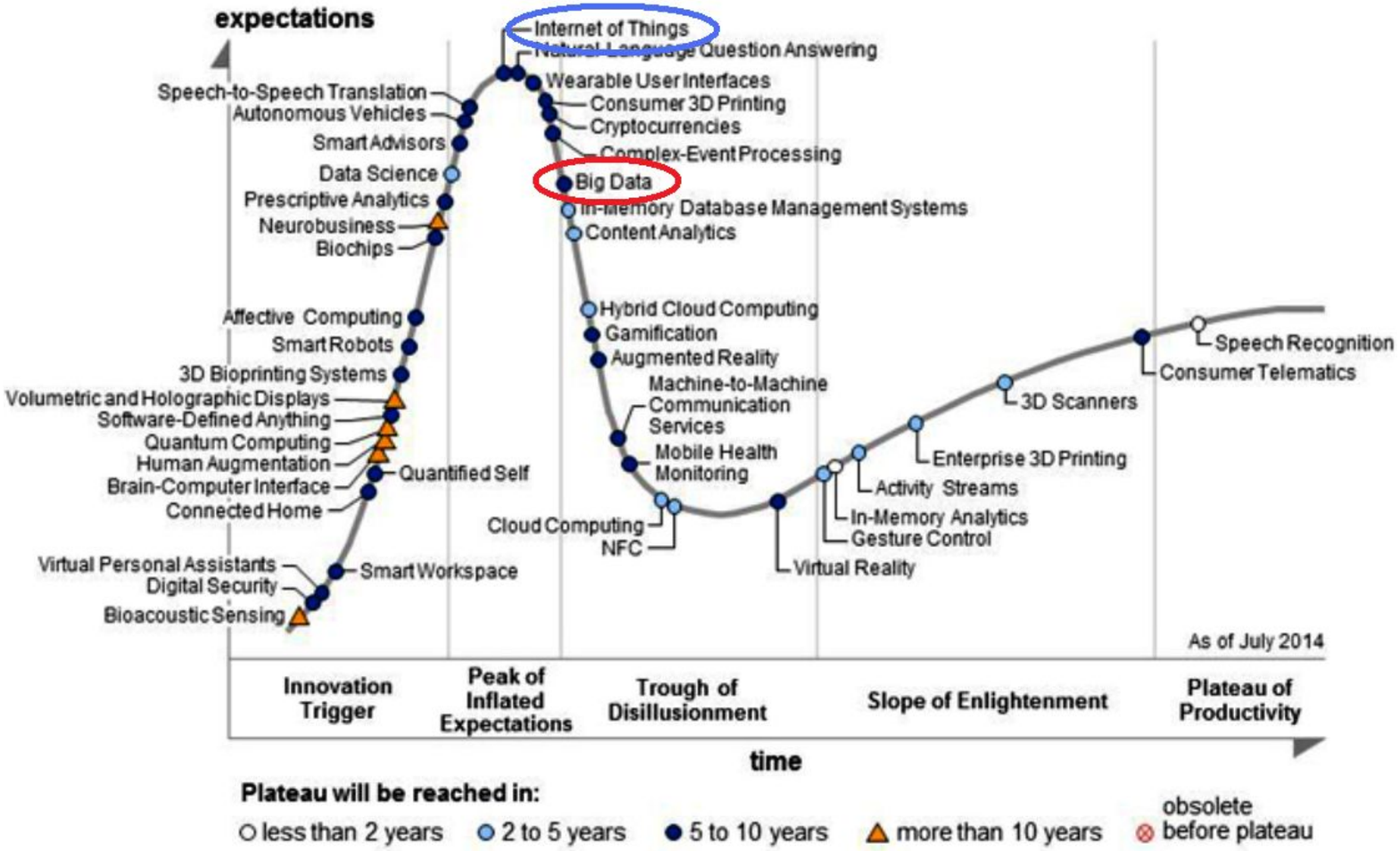


Data Science is so hyped right now...



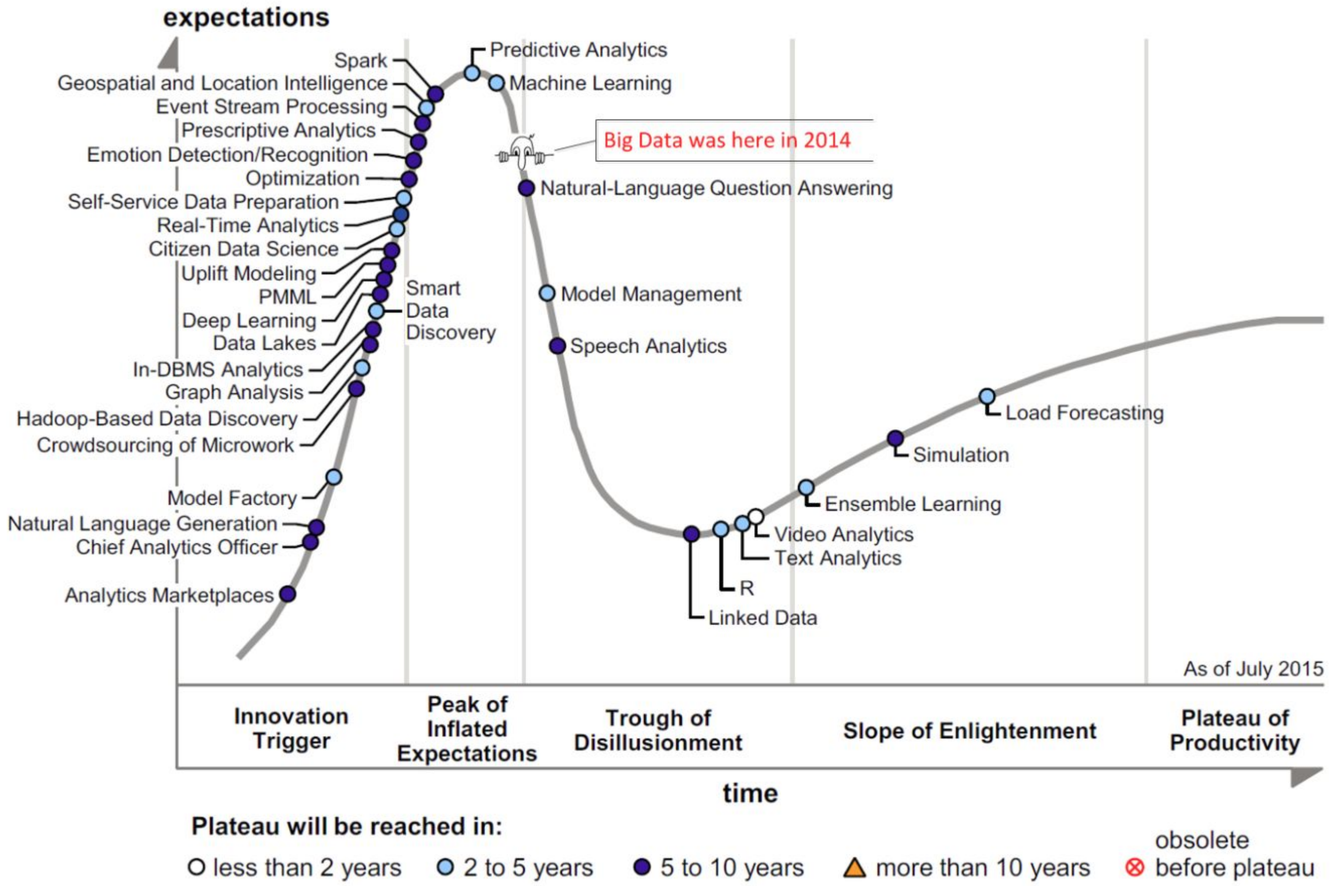


These graphs are also hyped...



These graphs are also hyped...

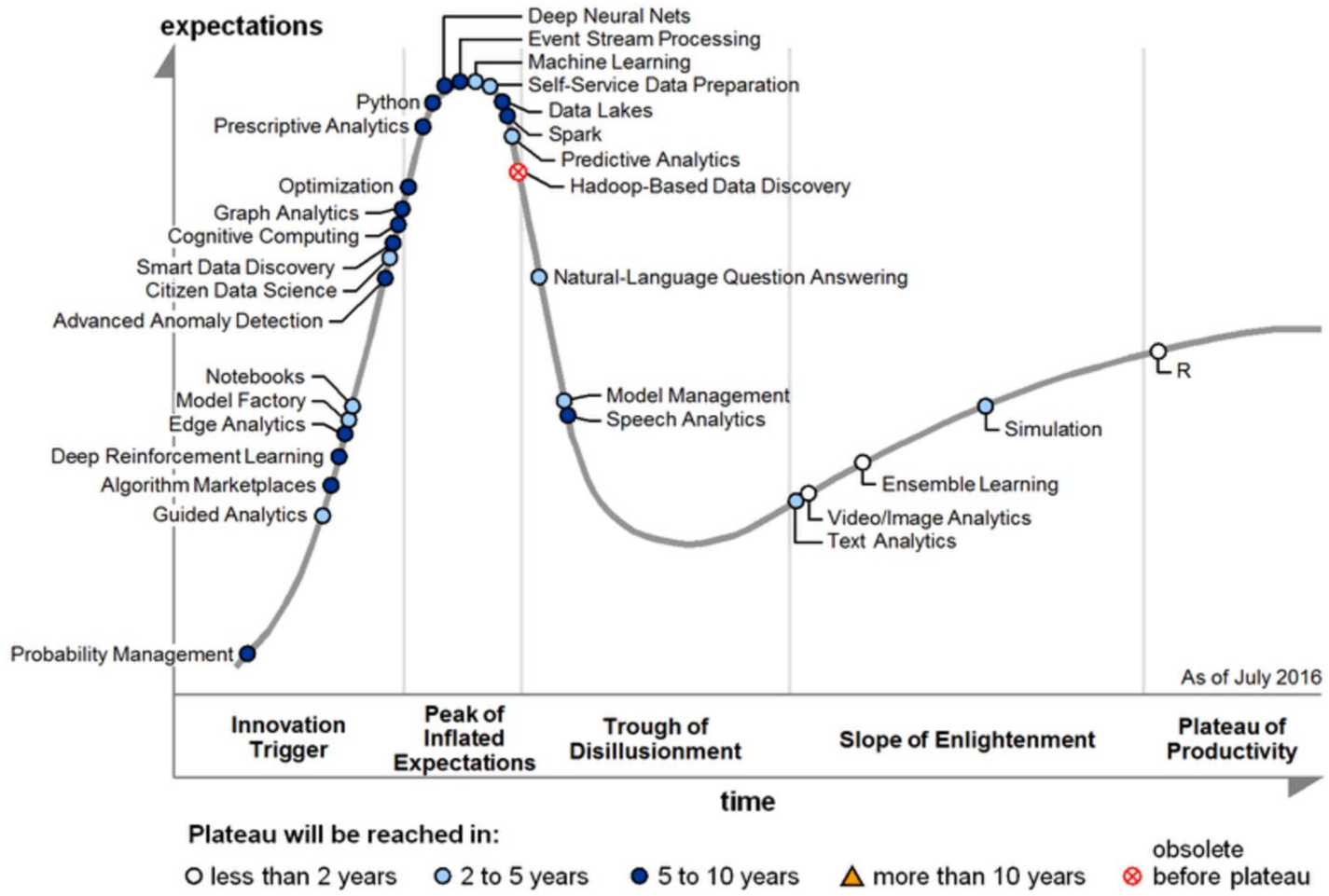
Figure 1. Hype Cycle for Advanced Analytics and Data Science, 2015



Source: Gartner (July 2015)

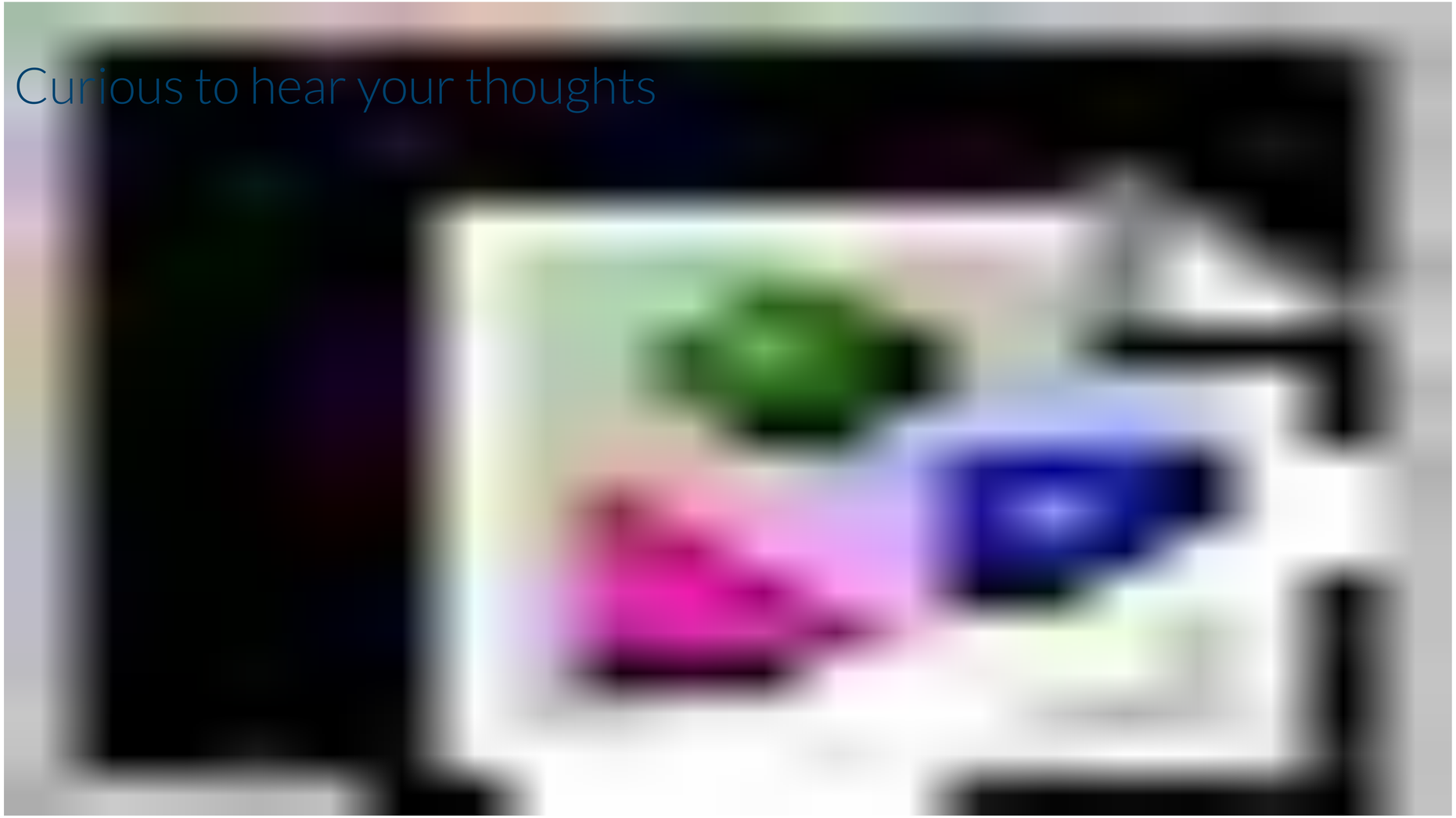
These graphs are also hyped...

Figure 1. Hype Cycle for Data Science, 2016

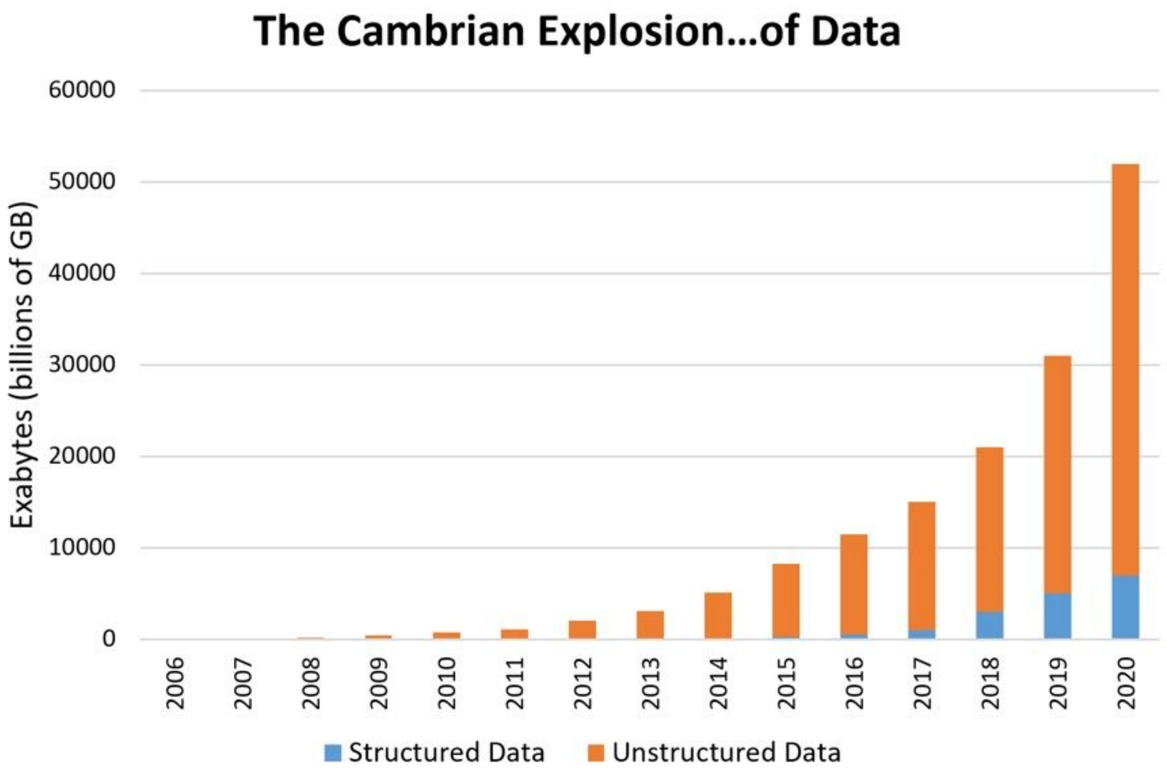


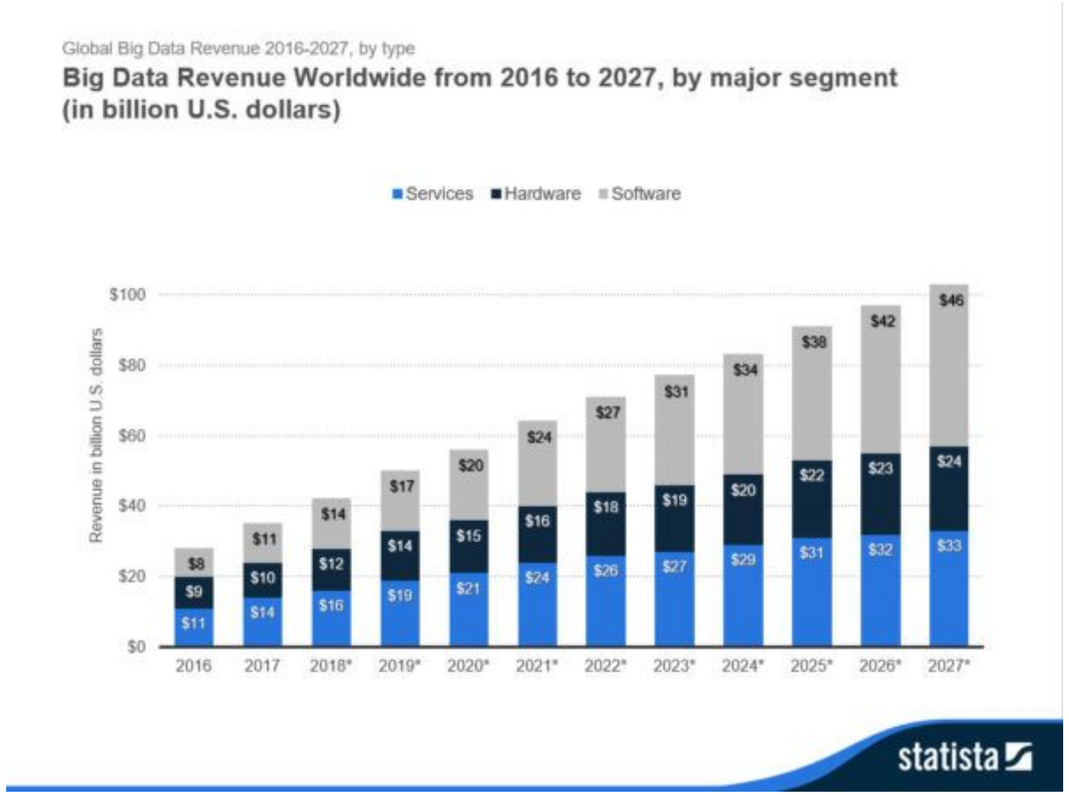
Source: Gartner (July 2016)

Why the hype in **Data Science?**



Why the hype in **Data Science?**





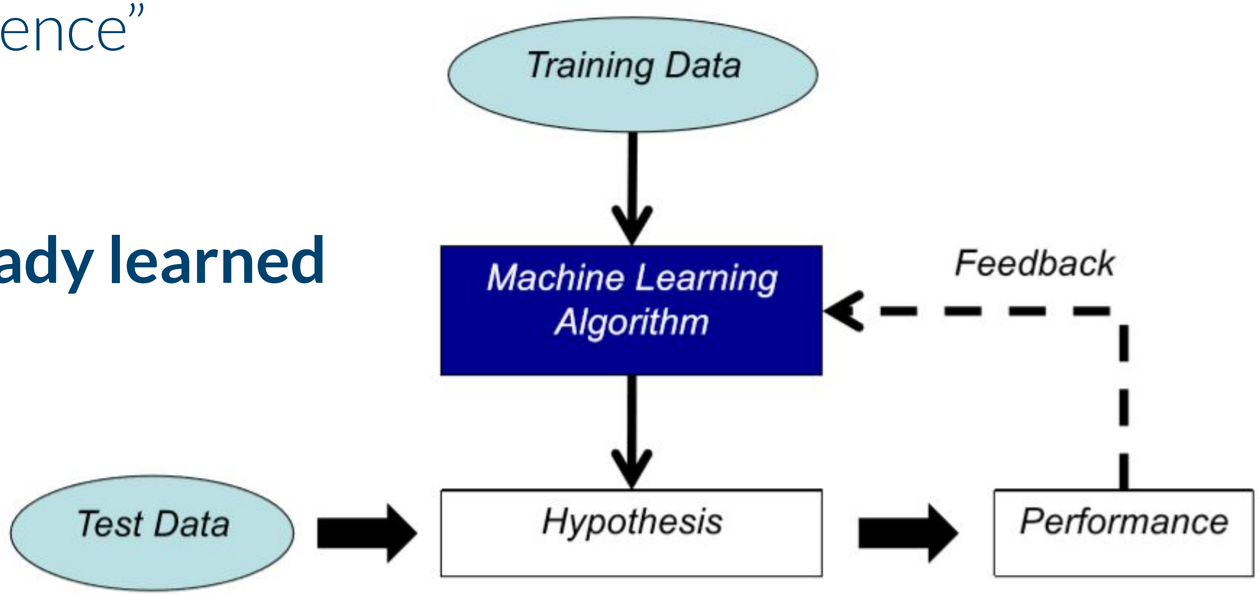
The true Data Science enabler: **ML**

Machine Learning: Field of study that gives the computer the ability to learn without being explicitly programmed - **Arthur Samuel (1959)**

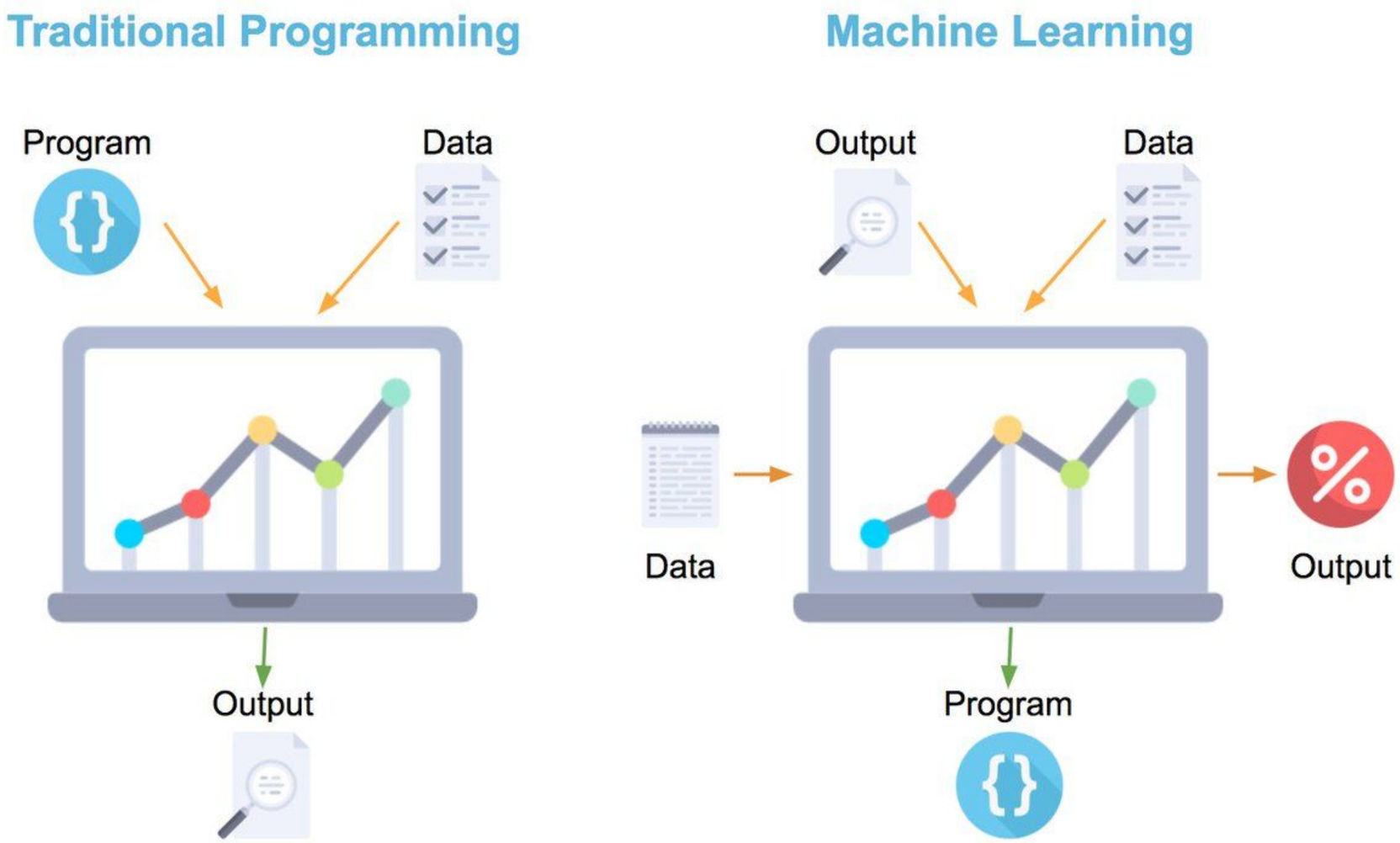
Okay so… **What is learning?**

By dictionary definition: To gain knowledge or understanding of, or skill in by study, instruction or experience”

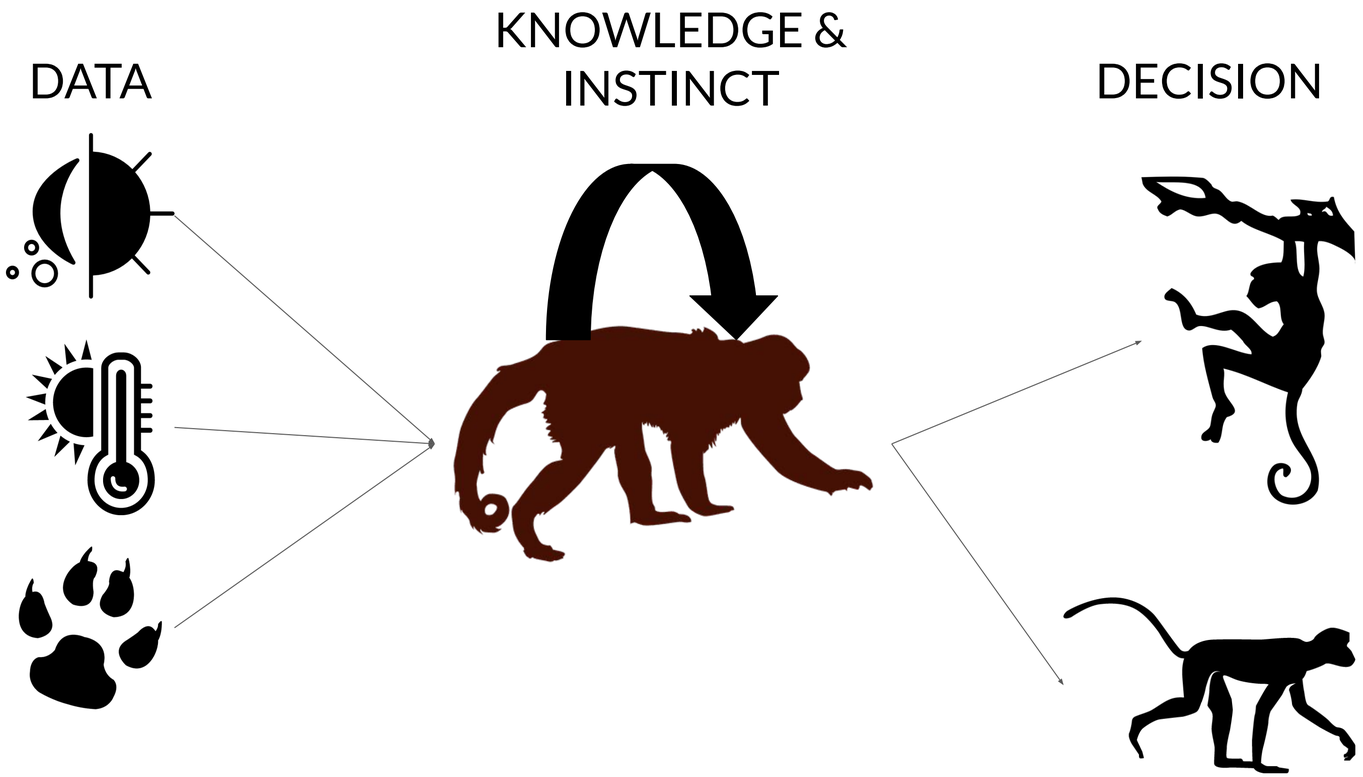
- **Learning a set of new facts**
- **Learning HOW to do something**
- **Improving ability of something already learned**



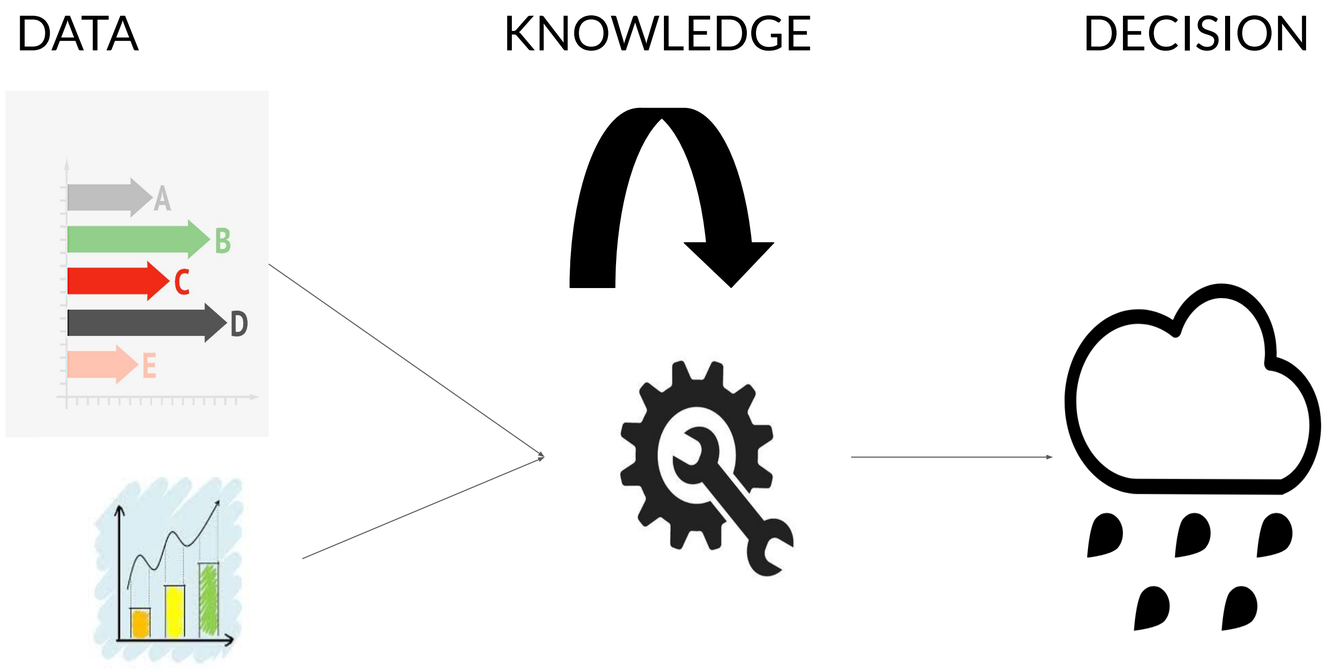
Traditional Programming vs ML



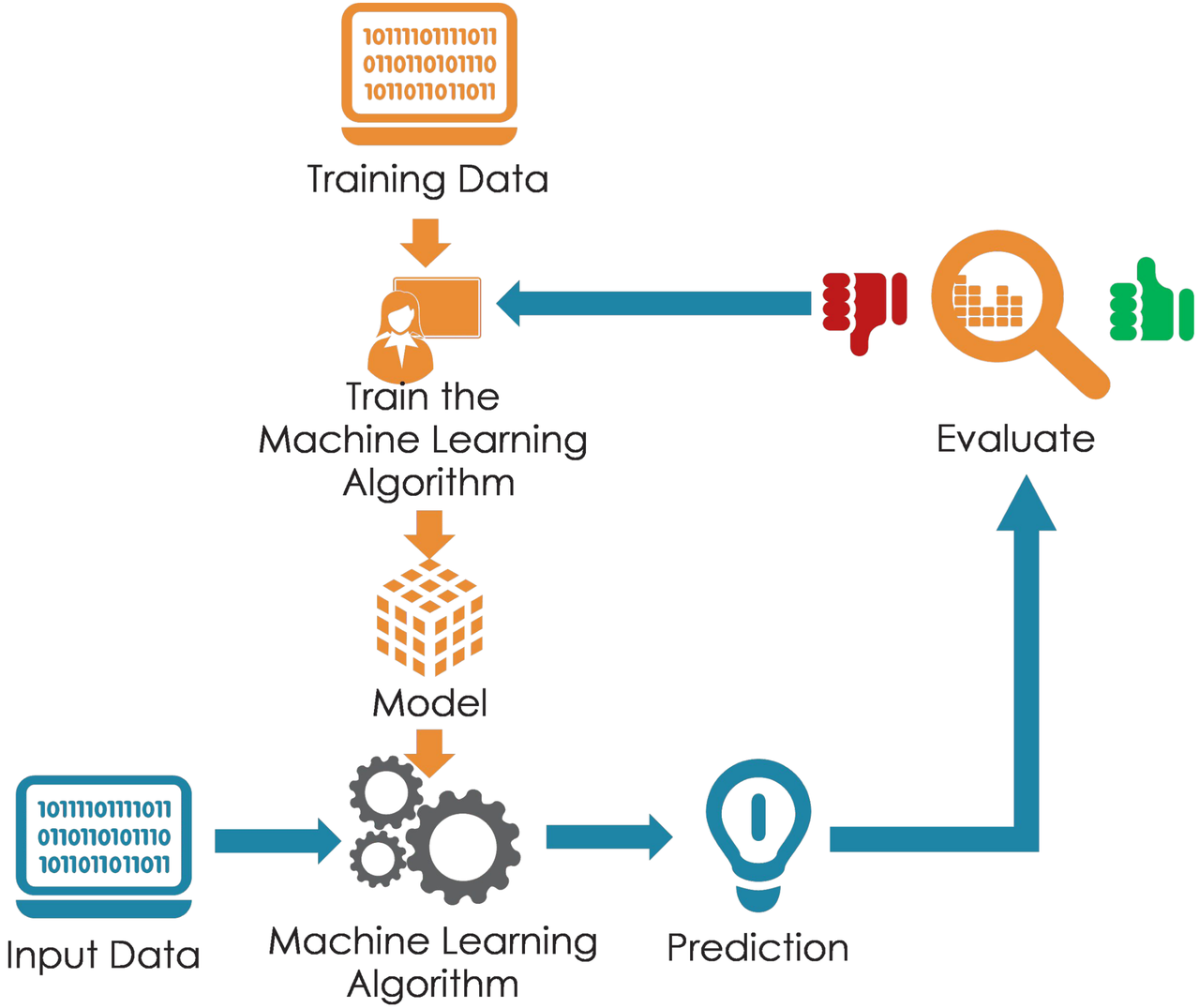
How do **humans learn?**

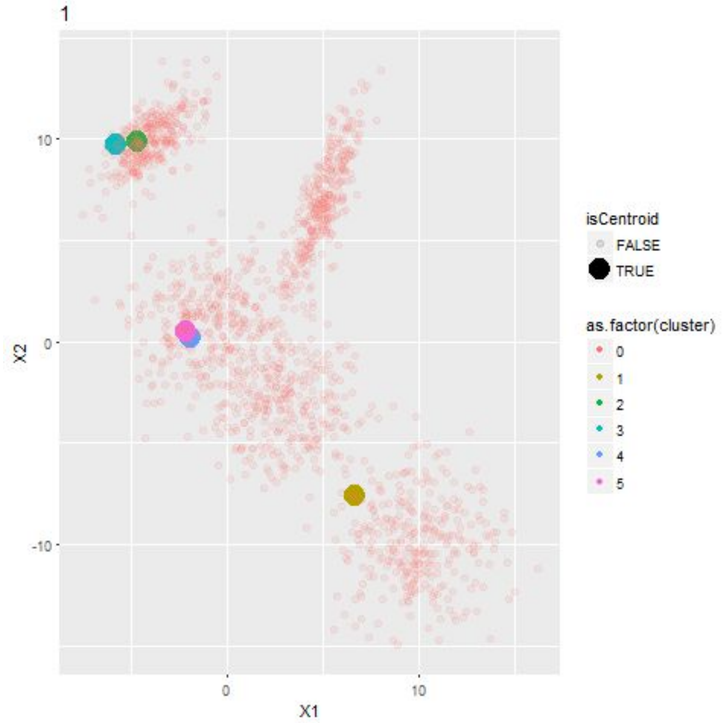


How can **machines learn?** (simplified)



Machine learning process





When to not use **Machine Learning**

- Option 1: **Manual Counting**
  - **Cost:** 4h+ and Anxiety
  - **Precision:** 100%
- Option 2: **Weighting**
  - **Cost:** 10’ + smile
  - **Precision:** 99.5%
- Option 3: **Machine Learning**
  - 3 days to design coin recognition engine
  - Computing power & not 100% prec.

When to not use **Machine Learning**

- Option 1: **Manual Counting**
  - **Cost:** 4h+ and Anxiety
  - **Precision:** 100%
- Option 2: **Weighting**
  - **Cost:** 10’ + smile
  - **Precision:** 99.5%
- Option 3: **Machine Learning**
  - 3 days to design coin recognition engine
  - Computing power & not 100% prec.

Machine Learning drawbacks:

- Expensive to implement
- Requires high domain expertise
- Can generate additional bias
- Can act as a black box

Don’t fish with dynamite…

… if you can avoid it

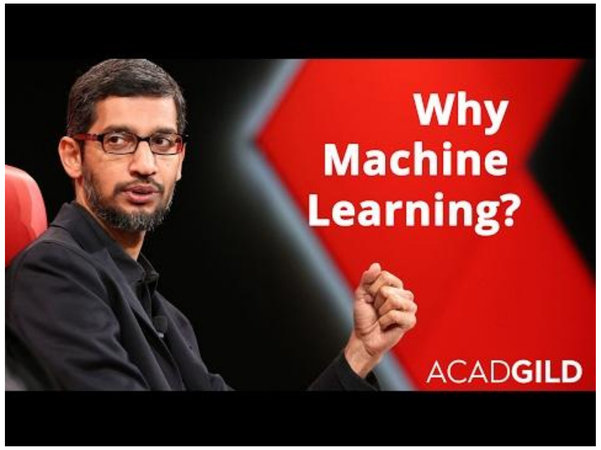
Why use **Machine Learning**?

Develop systems that are **too difficult** or **expensive** to construct manually because they require specific detailed skills or knowledge tuned to a specific task (knowledge engineering bottleneck). **How to we synthesise new proteins?**

Develop systems that can **automatically adapt** and **customize themselves** to individual users (personalization). **How does your inbox filter spam?**

Discover **new knowledge** from **large databases** (datamining).

How does each one of you surf youtube?



Why should you study **Machine Learning**?

- Many basic effective and efficient algorithms available
- Large amounts of online data available
- Large amounts of computational resources available
- Being in the right space matters more than doing exactly the right thing.
  - Startup in SF vs Startup Elsewhere in the world
  - Think about internet companies on 1998
  - Gold Mining in California
  - Hopefully we have also proved that this not a hype.

Machine Learning Definition

Study of algorithms that given a **task T**, they **improve their performance P** based on **experience E**.

In a way, learning implies : <T,P,E>

Some examples:

T: Handwritten words recognition

P: Percentage of words identified correctly

E: Database of handwritten words

T: Driving autonomously using LIDAR

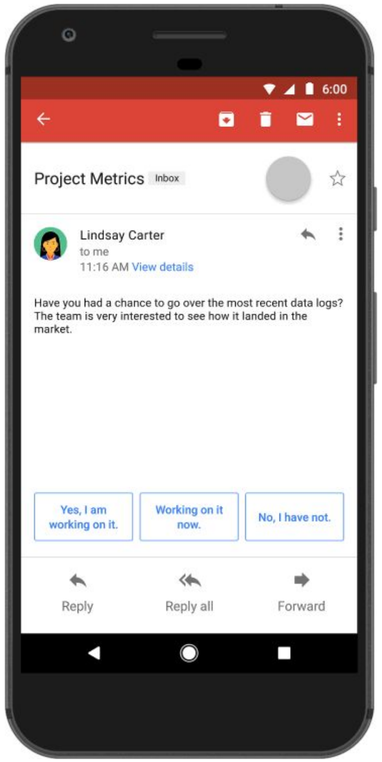
P: Average distance covered before a mistake

E: Sequence of images and direction commands recorded while a human was driving (millions of km and data actually)

Machine Learning Examples

Study of algorithms that given a **task T**, they **improve their performance P** based on **experience E**.

Let's think of some : <T,P,E>



Machine Learning in your Projects

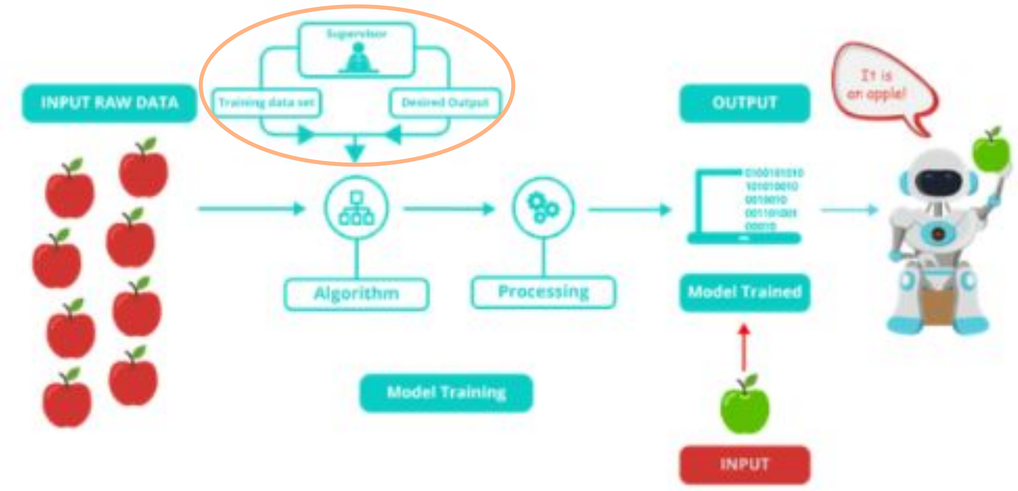
Study of algorithms that given a **task T**, they **improve their performance P** based on **experience E**.

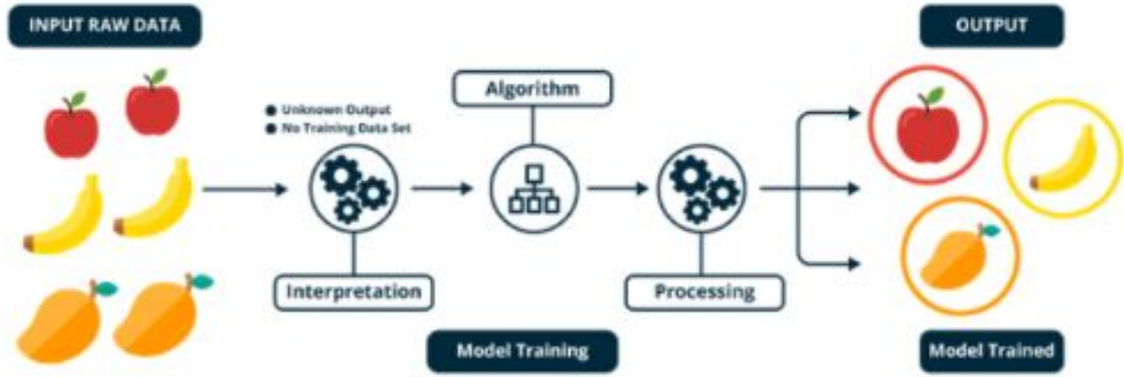
Let’s think of some : <T,P,E> …. With your projects!

Do you already know what you would like to do? **Let’s think of at least a field.**

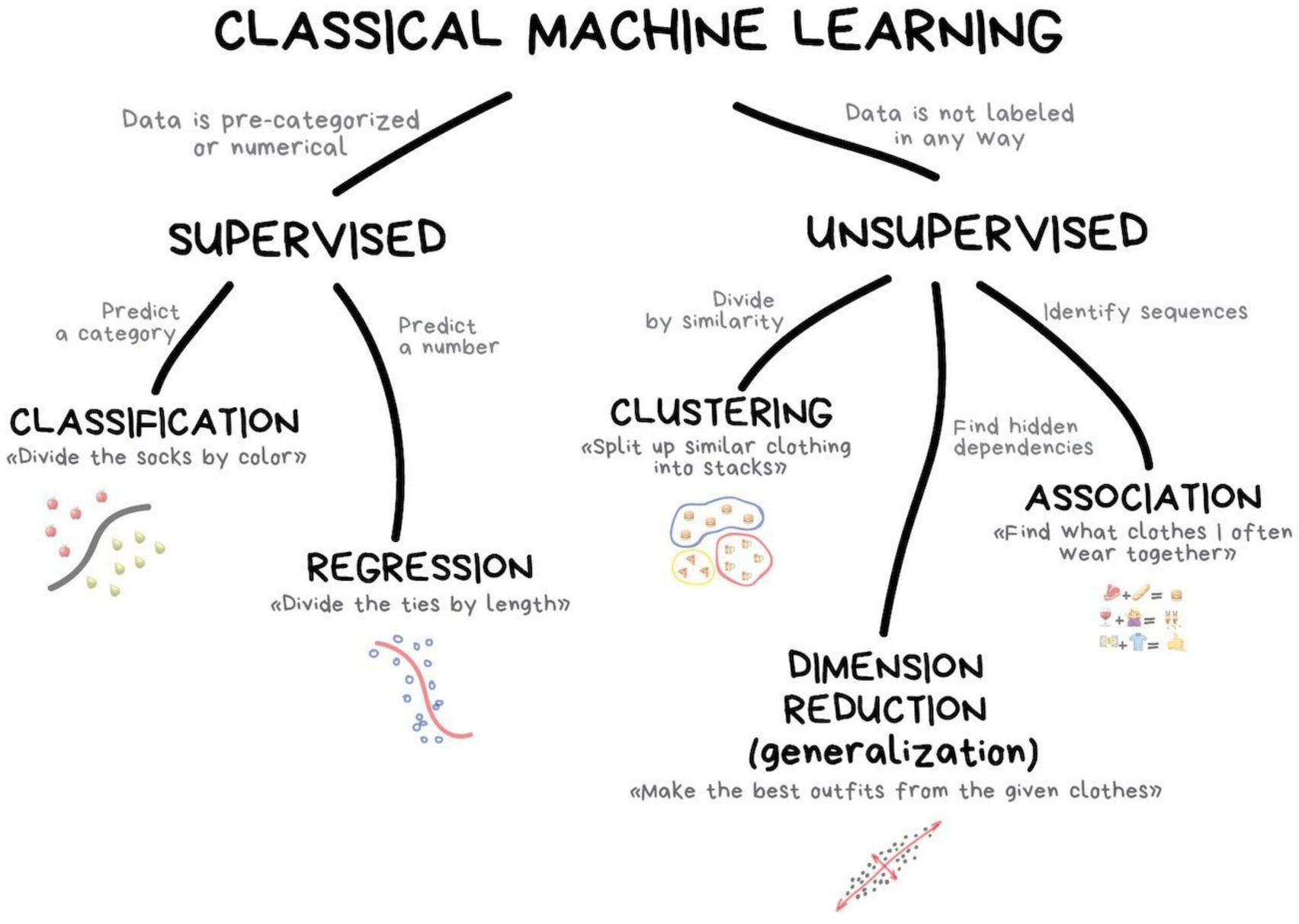
Two types of machine learning

- **Supervised Learning**
  - Labelled data
  - Classification
  - Regression
- **Self-supervised Learning**
  - Pattern Discovery
  - Clustering
  - Anomaly detection





Fast Machine Learning **Overview**



Actually, this is a lie. There's at least 3 types

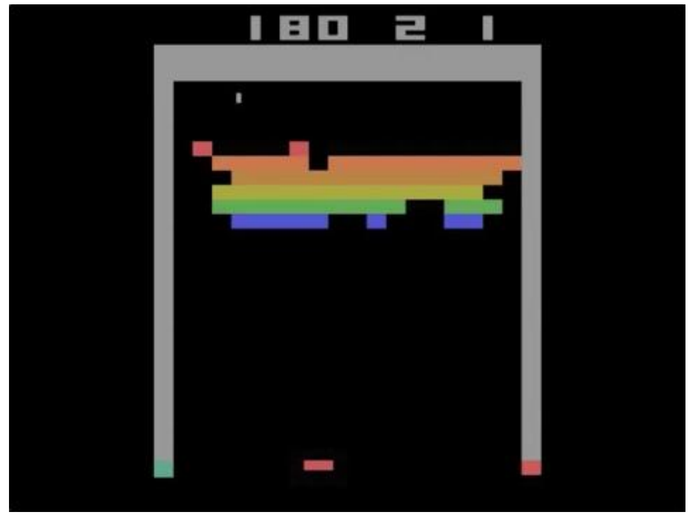
- **Bits = Information needed by the Machine Learning system**

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**

- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**

- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

Three **Examples**



Designing a Machine Learning System

Steps:

1. Picking the data (training experience)
2. Picking what we want to learn (target function)
3. Choosing how to represent the target function
4. Picking a learning algorithm to infer the target function from the training experience.

Maths for **Machine learning**

1. Selecting the right algorithm which includes giving considerations to **accuracy**, **training time**, **model complexity**, number of parameters and number of features.
2. Choosing **parameter settings** and validation strategies.
3. Identifying **underfitting** and **overfitting** by understanding the **Bias-Variance tradeoff**.
4. Estimating the **right confidence interval** and **uncertainty**.

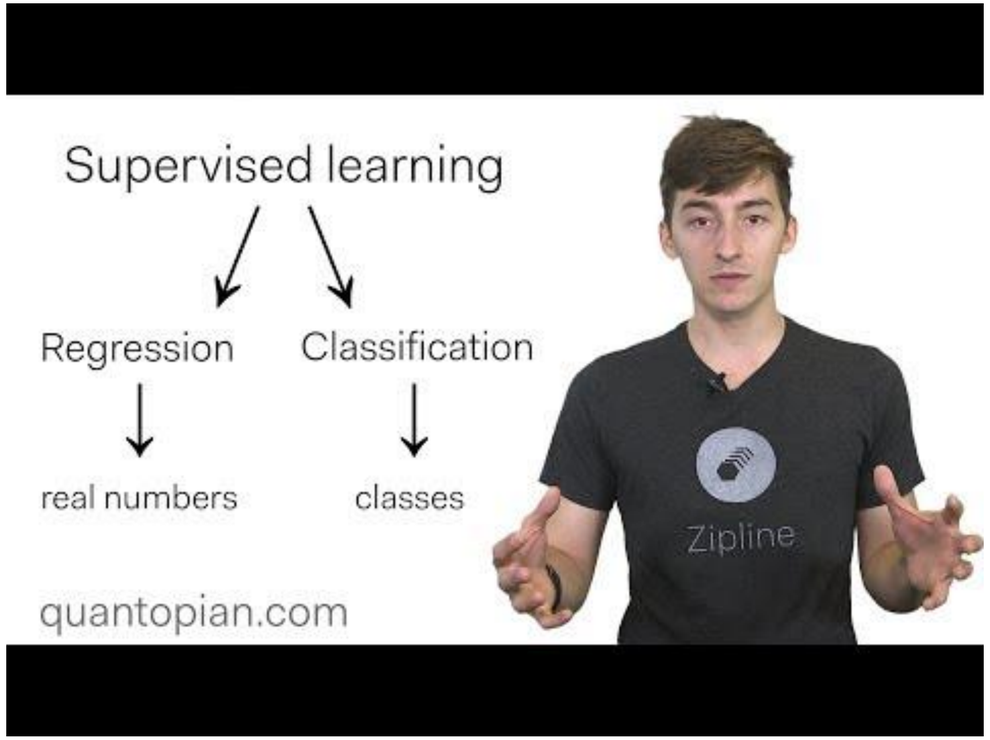
Relevant Concepts

1. What is a **task**?
   a. Classification
   b. Regression
   c. Problem solving / planning / control

2. How to evaluate **performance**?
   a. Classifying the right answers (or error)
   b. The validity of the solution
   c. Quality of the solution
   d. Performance velocity

3. How to represent **experience**?
   a. Neurons, cases, decision trees, etc

Tomato example



Influences to Machine Learning

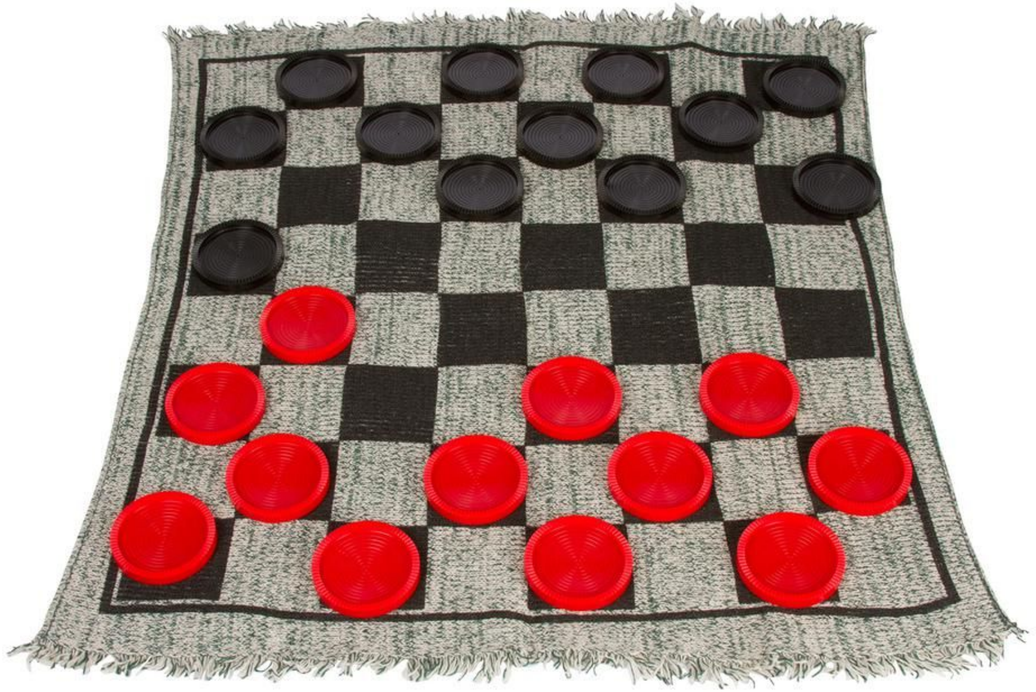
1. Models of intelligence
   - **Cognitive Psychology**: The process of human learning
   - **Neurobiology**: The brain, the neuron

2. Knowledge/concepts
   - **Cognitive Psychology**: What is a concept? How to represent them?
     Is there an explanation about how we represent them?
   - **Mathematical Logic**: How concepts can be combined and manipulated?
   - **Statistics**: What mathematical model represents them?
   - **Information theory**: How can we code them?

Sample Learning Problem

Learn to play checkers from self-play

We will develop an approach analogous to that used in the first machine learning system developed by Arthur Samuels at IBM in 1959



Influences to Machine Learning

Direct experience: Given sample input and output pairs for a useful target function.

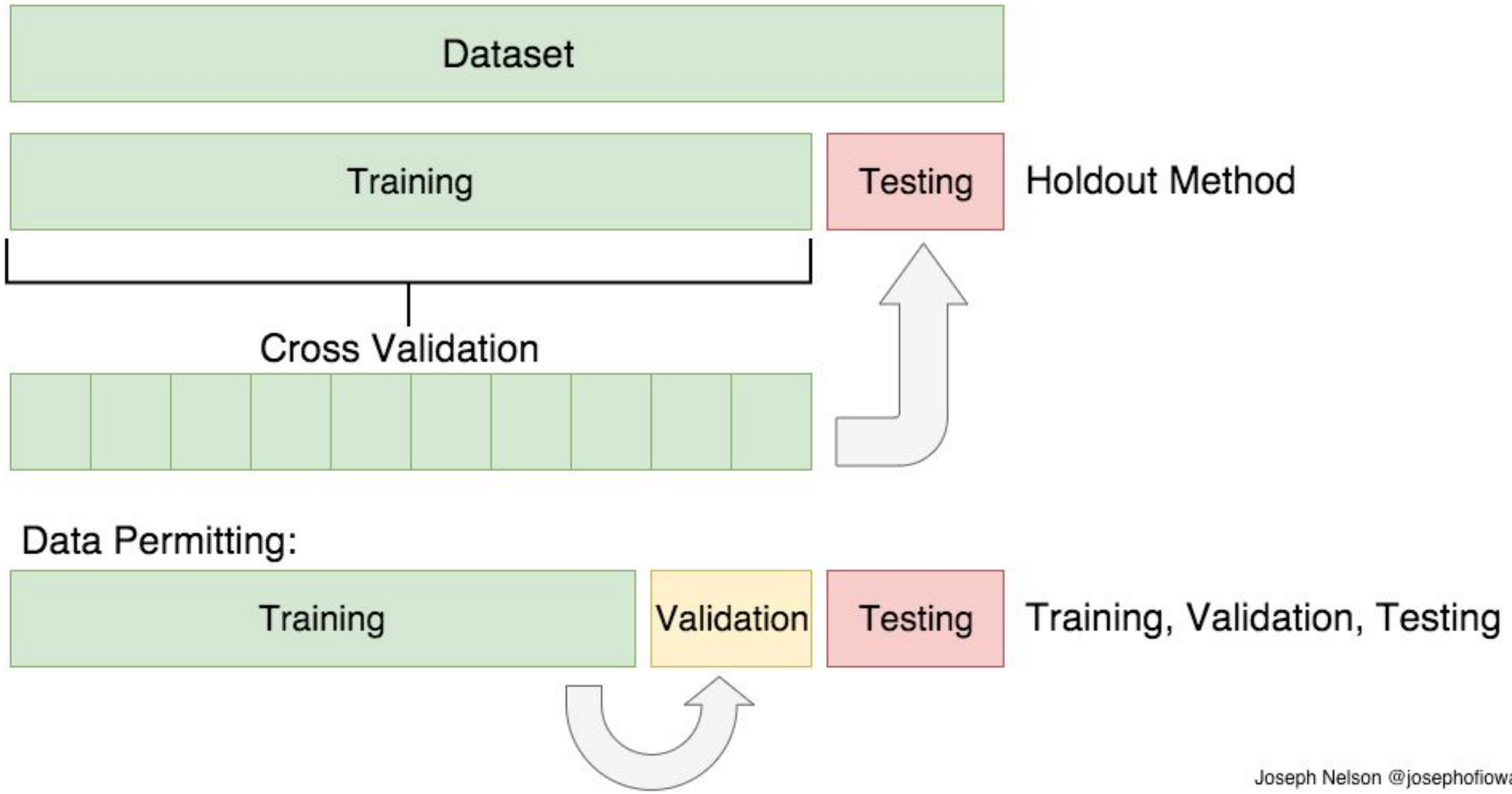
– *Checker boards labeled with the correct move*, e.g. extracted from record of expert play

Indirect experience: Given feedback which is not direct I/O pairs for a useful target function.

– *Potentially arbitrary sequences of game moves and their final game results.*

Train / Test Split

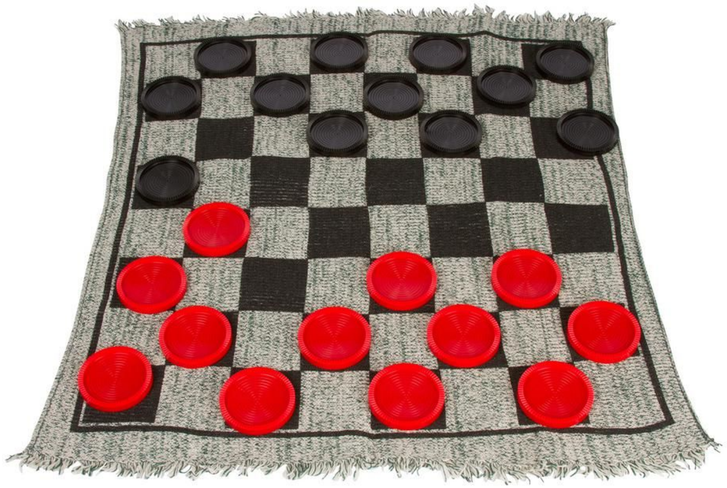
Like practicing for an exam with class exercises. Seeing the exam first would defeat the learning challenge



Choosing a target function

What function is to be learned and **how will it be used** by the performance system?

For checkers, assume we are **given a function for generating the lega moves** for a given board position and want to decide the **best mov**e.



– Could learn a function:

ChooseMove(board, legal-moves) → best-move

– Or could learn an evaluation function,

V (board) → R,

that gives each board position a score for how favorable it is

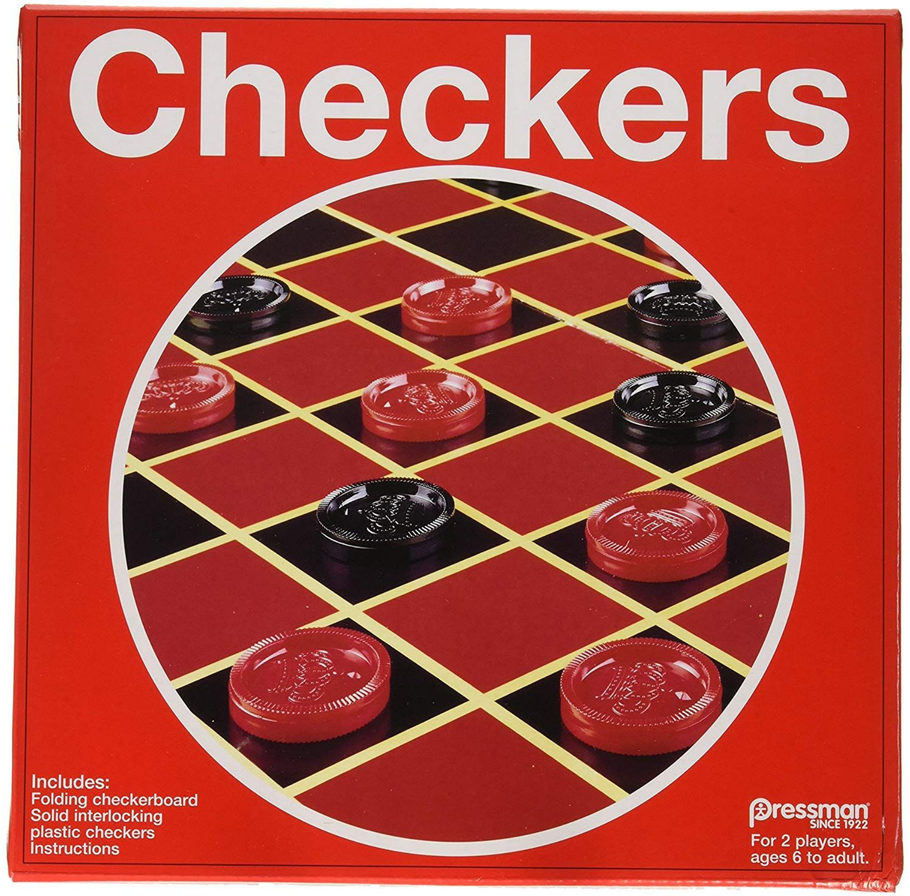
Ideal definition of V(board)

If b is a final winning board, then V(b) = 100

If b is a final losing board, then V(b) = –100

If b is a final draw board, then V(b) = 0

Otherwise, then V(b) = V(b´), where b´ is the **highest scoring final board position** that is achieved starting from b and playing optimally until the end of the game (assuming the opponent plays optimally as well).

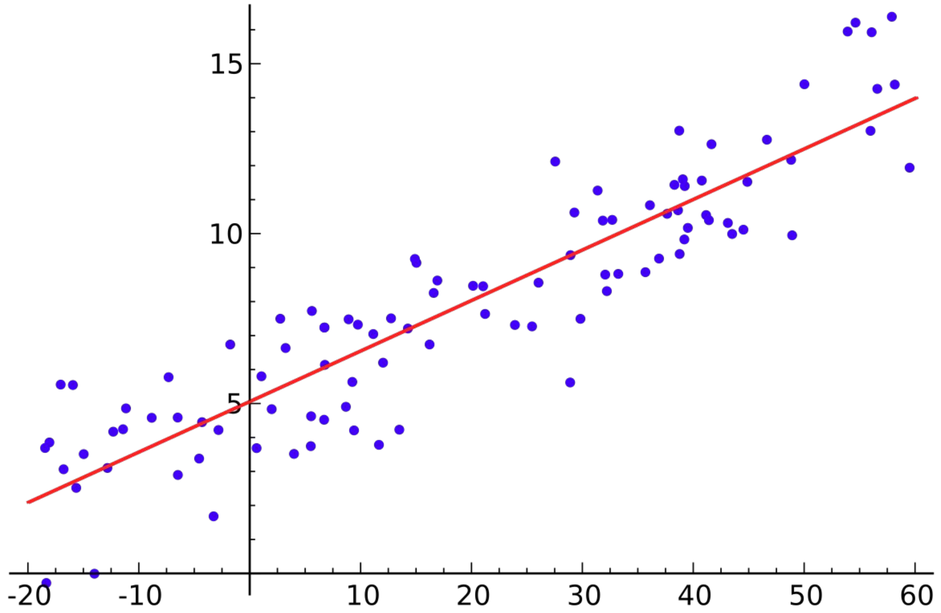


Target function

Which function must be learned and how will it be used in the system to incentivize performance?

Target function **can be represented in many ways**: lookup table, symbolic rules, numerical function -like in the picture- or neural network among others.

There is a trade-off between the **expressiveness** of a **representation** and the **ease of learning**.

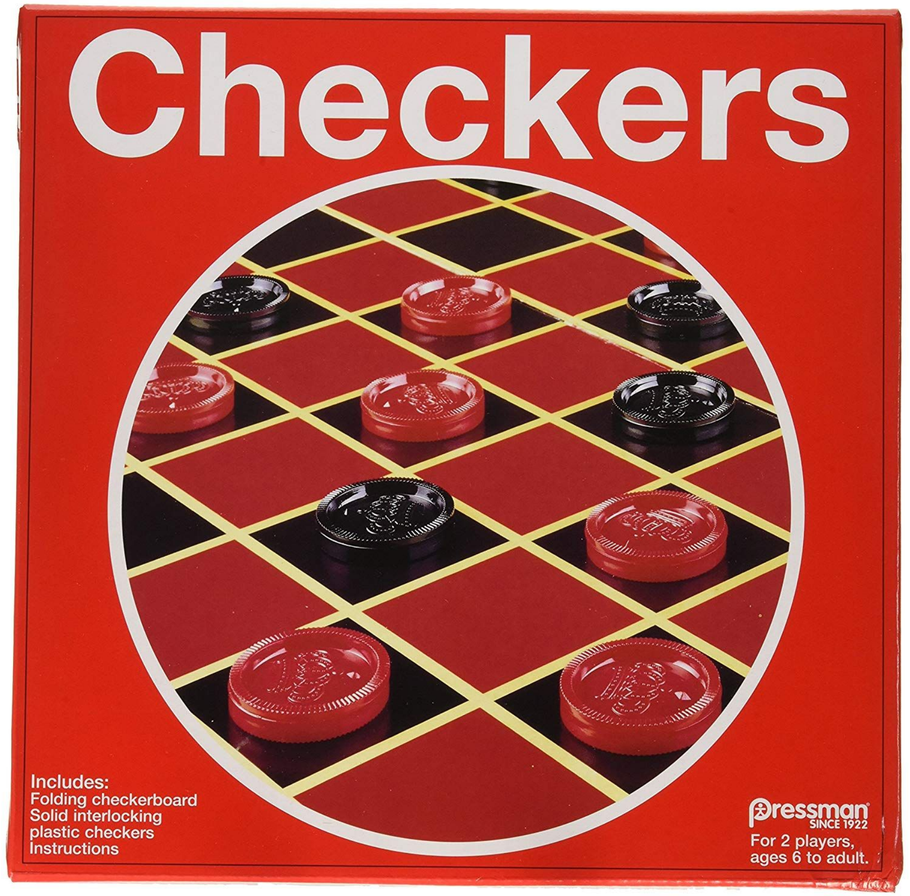


Linear Function for Representing V(b)

In checkers, use a linear approximation of the evaluation function.

$$\hat{V}(b) = w_0 + w_1 \cdot bp(b) + w_2 \cdot rp(b) + w_3 \cdot bk(b) + w_4 \cdot rk(b) + w_5 \cdot bt(b) + w_6 \cdot rt(b)$$

- **bp(b)**: number of black pieces on board b
- **rp(b)**: number of red pieces on board b
- **bk(b)**: number of black kings on board b
- **rk(b)**: number of red kings on board b
- **bt(b)**: number of black pieces threatened (i.e. which can be immediately taken by red on its next turn)
- **rt(b)**: number of red pieces threatened

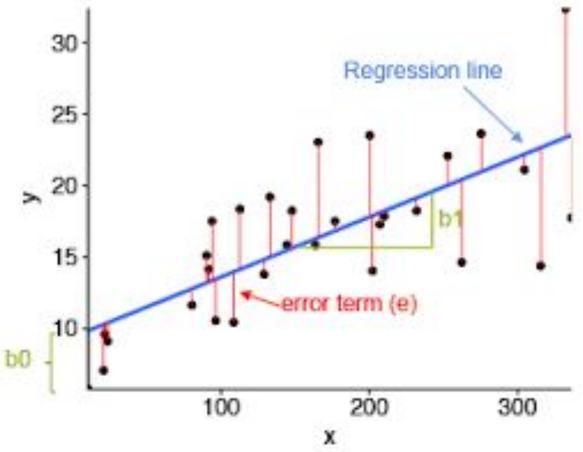


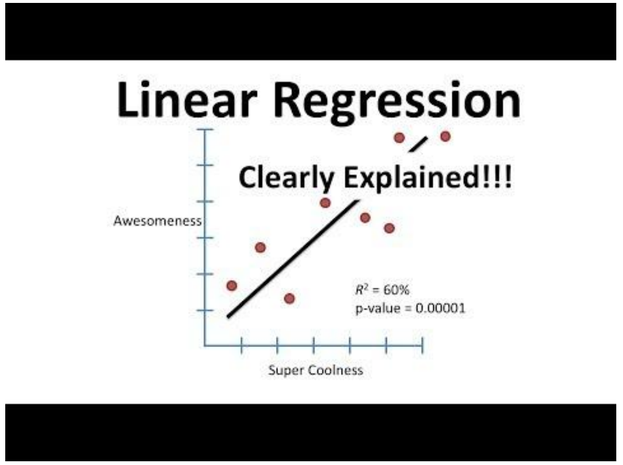
Learning Algorithm

Uses training values for the target function to induce a hypothesized definition that fits these examples and hopefully generalizes to unseen examples.

In statistics, learning to approximate a continuous function is called **regression**. Attempts to minimize some measure of error (**loss function**) such as **mean squared error**:

$$E = \frac{\sum_{b \in B} [V_{train}(b) - \hat{V}(b)]^2}{|B|}$$





Gradient Descent Algorithm

The **gradient descent algorithm** updates incrementally the weights of a linear function in an attempt to **minimize the mean squared error**.

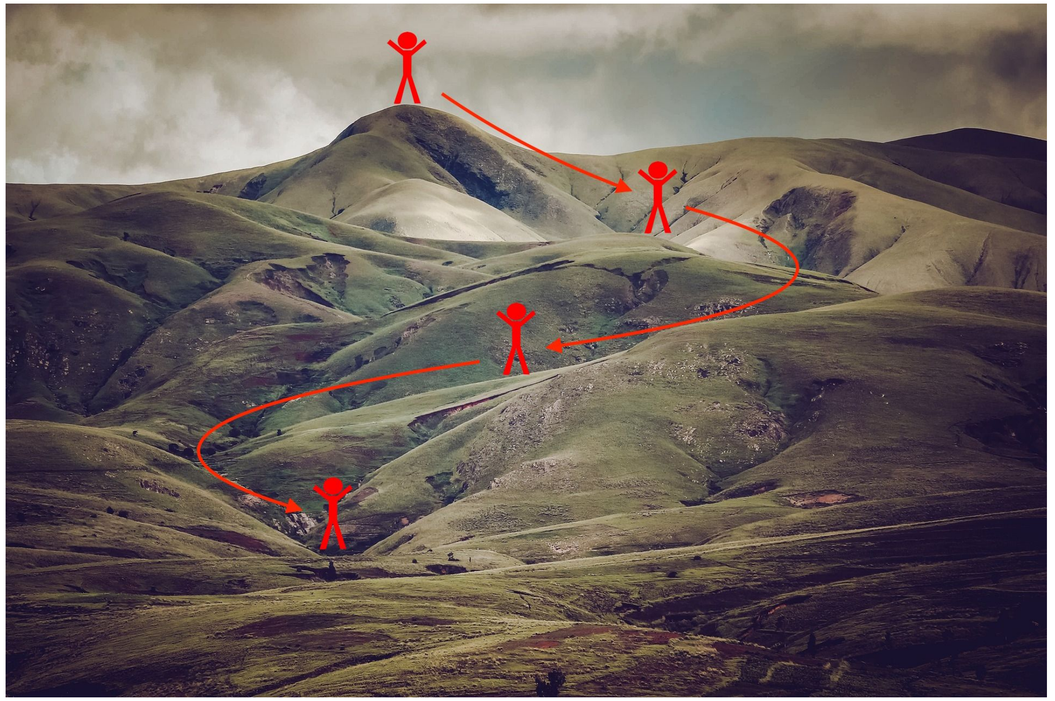
- **Repeat** until the points converge:
  - For each element of training **b**:
    - Calculate the absolute error:

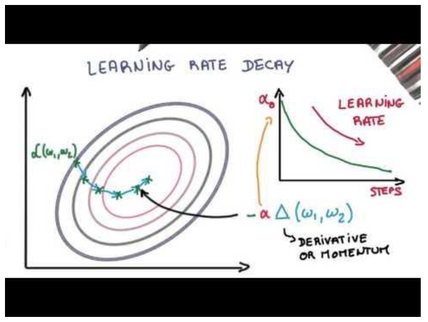
$$error(b) = V_{train}(b) - \hat{V}(b)$$

    - For each component of the function, update its weight:

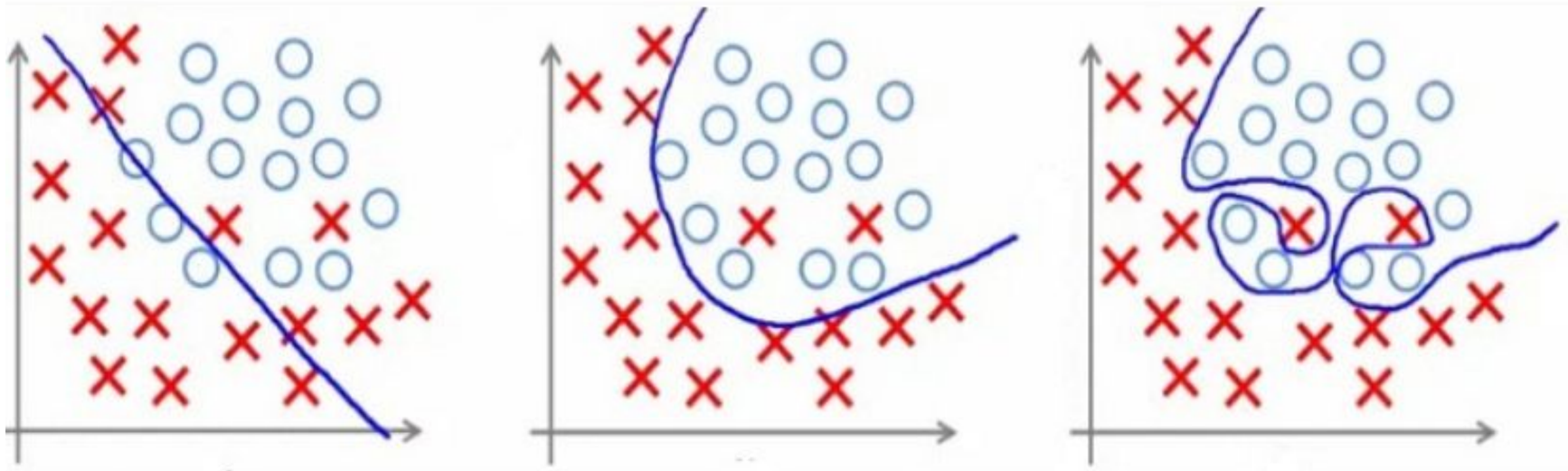
$$w_i = w_i + c \cdot f_i \cdot error(b)$$

c is the learning rate





Under-fitting and Overfitting



Under-fitting

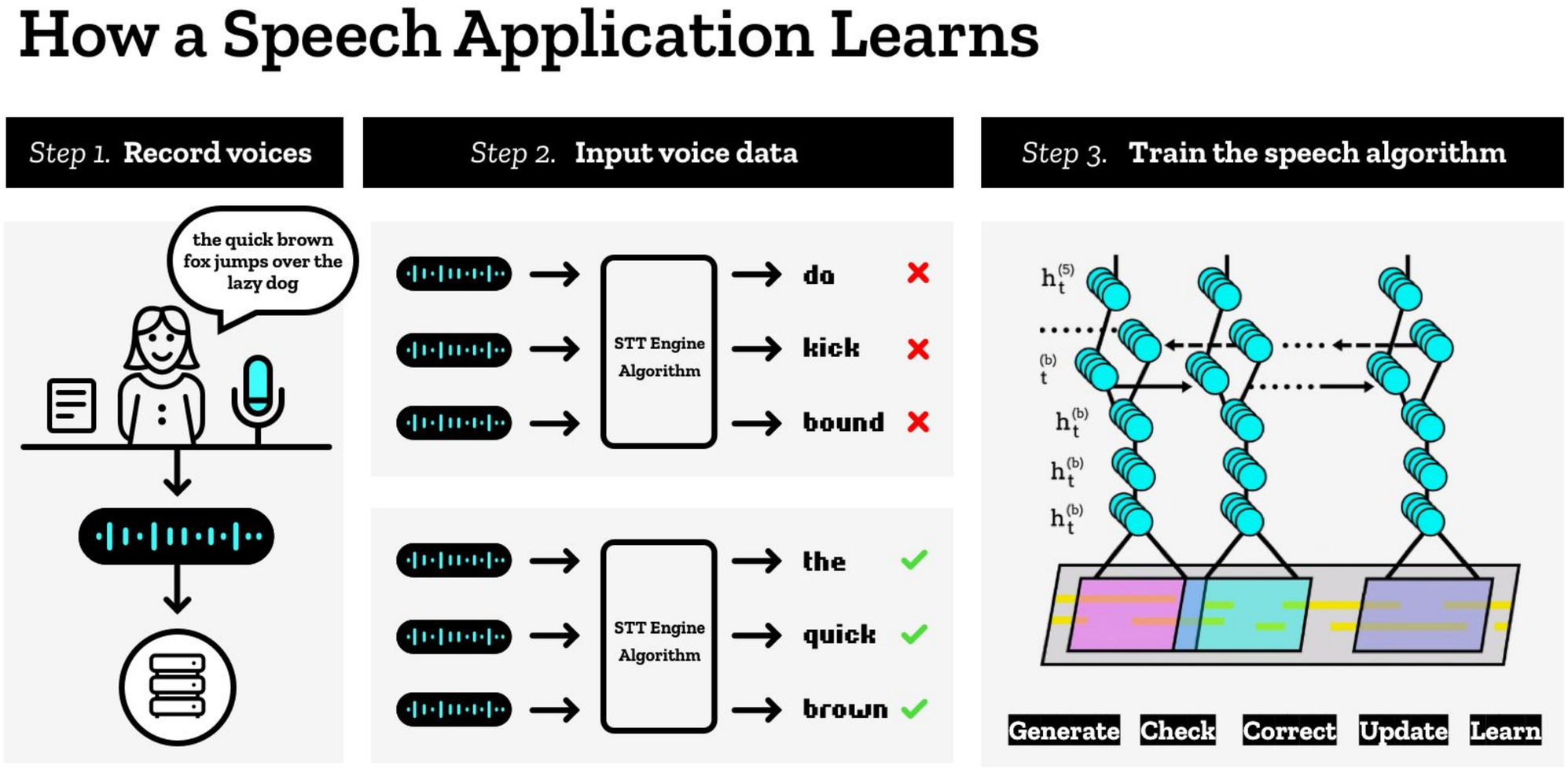
(too simple to explain the variance)

Appropriate-fitting

Over-fitting

(forcefitting -- too good to be true)

Example: Speech Application



Common Voice Project

Open Source STT Engine

Deep Learning Architecture

To **review** and **summarize**: Learning

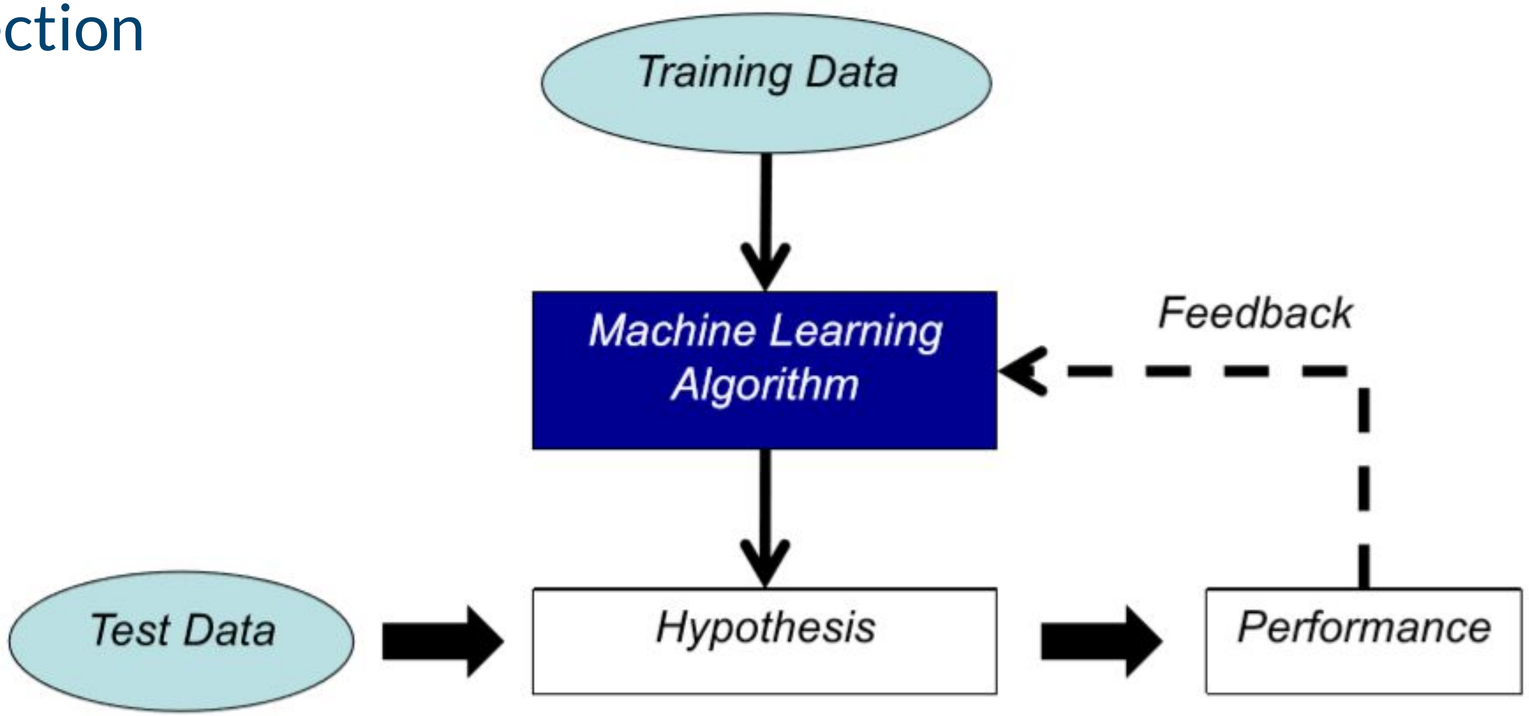
Learning can be viewed as using **direct or indirect experience** to approximate a chosen target function.

Function approximation can be viewed as a **search through a space of hypotheses** (representations of functions) for one that best fits a set of training data.

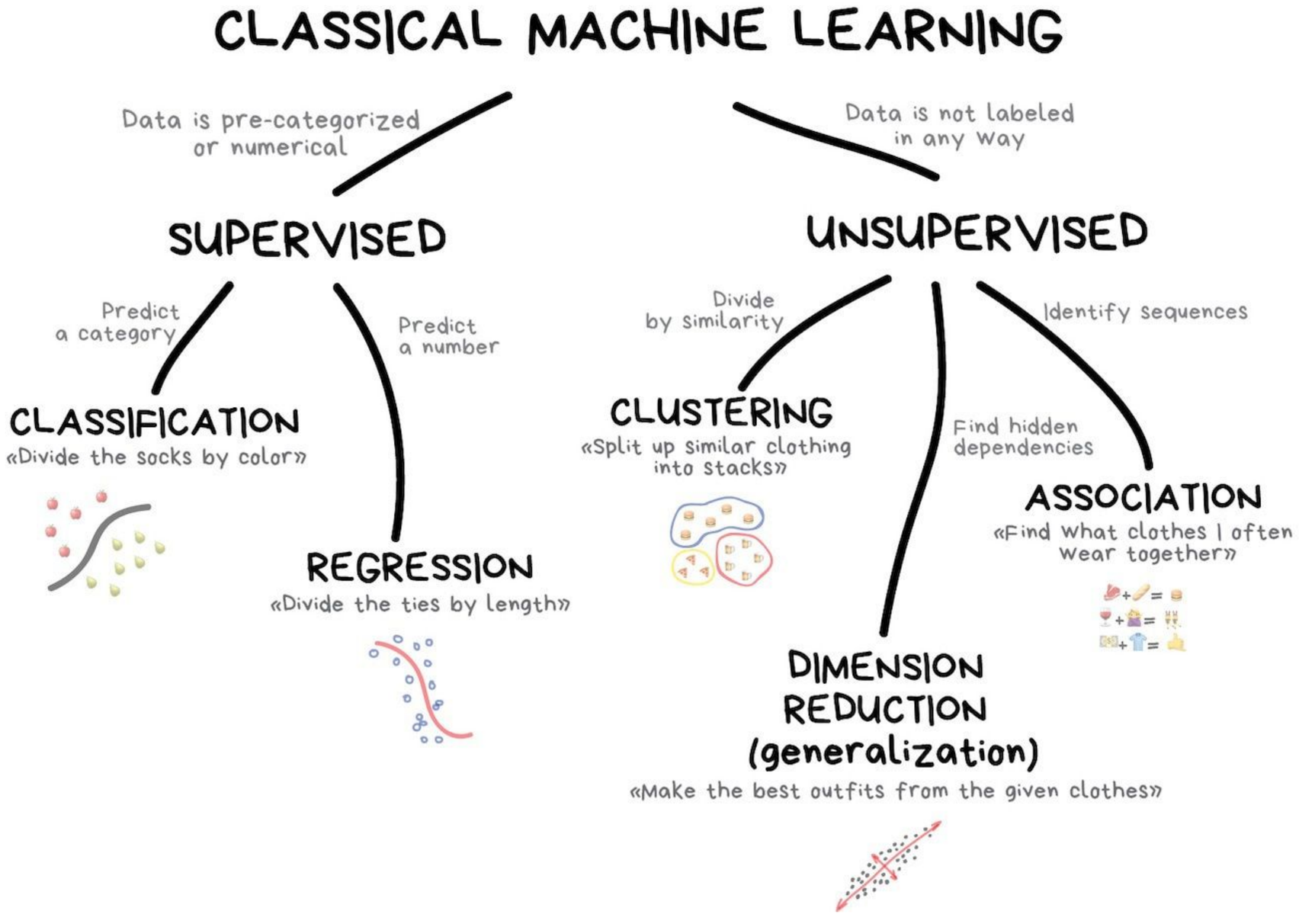
Different learning methods assume **different hypothesis spaces** (representation languages) and/or employ different **search techniques**.

To **review** and **summarize**: ML Process

1. Data collection and Preparation
2. Feature Selection
3. Algorithm Choice
4. Parameter and model selection
5. Training
6. Training Data
7. Evaluation



To **review** and **summarize**: Types of ML



Introduction to **Sklearn**

Sklearn is the Swiss knife of machine learning, it comes with dozens of models out of the box and a huge community. It is not the most powerful knife but great to get started. There is also an awesome set of tutorials.

Sklearn comes installed with the conda environment. In other scenario we need to install it by means of **pip** (which we won't), to install it we just need to run:

```
conda install scikit-learn
```

Sklearn: Types of Models

Models in sklearn are imported separately as for example.

```
from sklearn.ensemble import RandomForestClassifier
```

Inside of sklearn we will find different types of models. I will just introduce the high level API of them:

- **Supervised models** to perform predictions.
- **Self-supervised models** to group data automatically.
- **Transformation models** to perform transformations in the data

Sklearn: Supervised Models

This kind of models are the most intuitive ones. You train them with data and expected outputs and later it will predict outputs for unseen data. To train the algorithm we call the fit method and to predict with it we call the predict function

```
from sklearn.ensemble import RandomForestClassifier


clf = RandomForestClassifier().fit(X, y)
clf.predict(X)
```

Sklearn: Supervised Models

This kind of models are the most intuitive ones. You train them with data and expected outputs and later it will predict outputs for unseen data. To train the algorithm we call the fit method and to predict with it we call the predict function

```
from sklearn.ensemble import RandomForestClassifier


clf = RandomForestClassifier().fit(X, y)
clf.predict(X)
```

Sklearn: Self-supervised Models

Other type of models, in this case it will not predict but find groups of similar elements inside data. To train the algorithm we call the fit method and to get the group of an unseen element we call the predict method

```
from sklearn.cluster import KMeans


clf = KMeans().fit(X)
clf.predict(X)
```

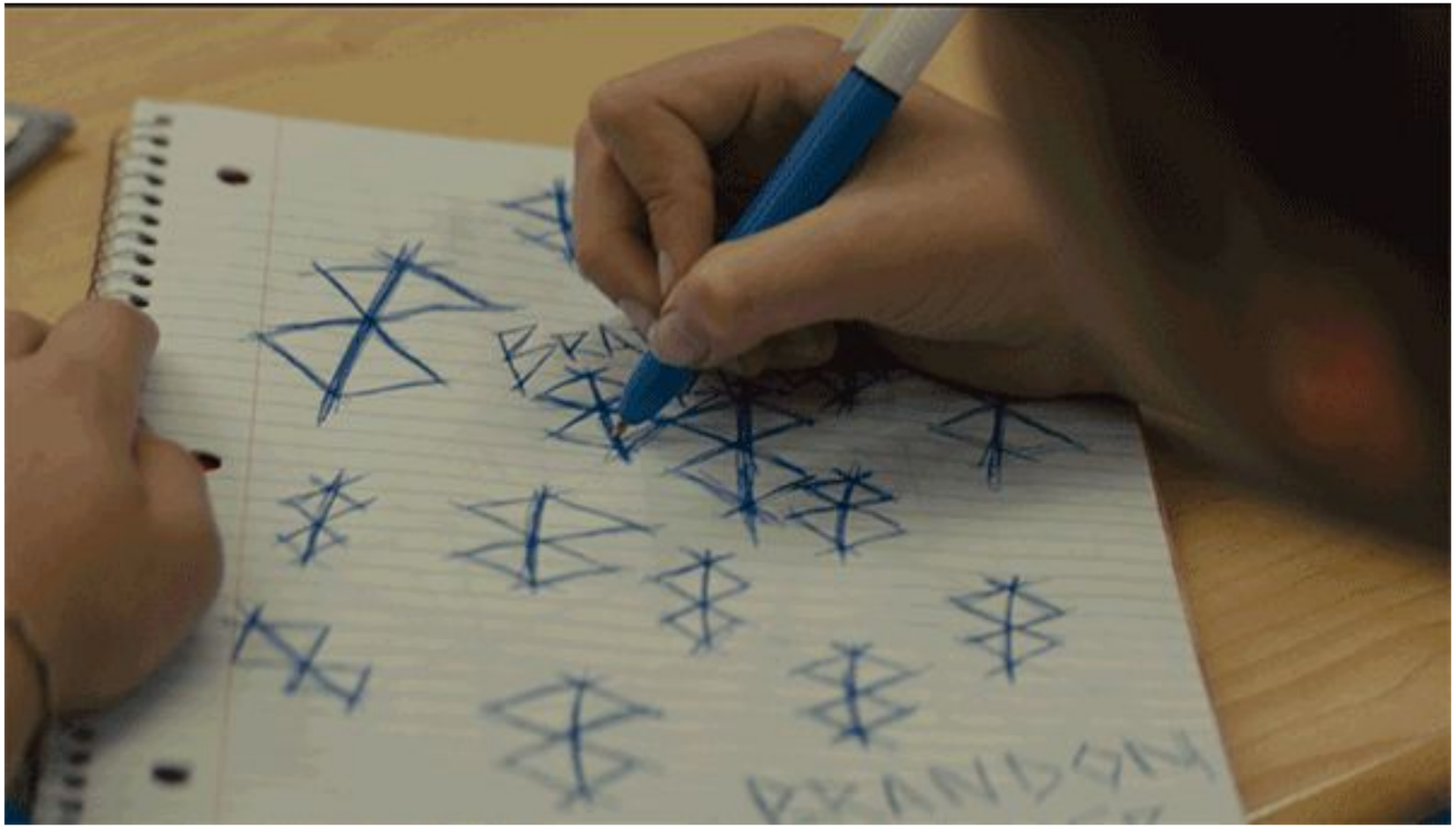
Sklearn: Transformation models

This last kind of models transform our data. For example for dimensionality reduction tasks or normalization tasks. Their main method is `fit_tranform`

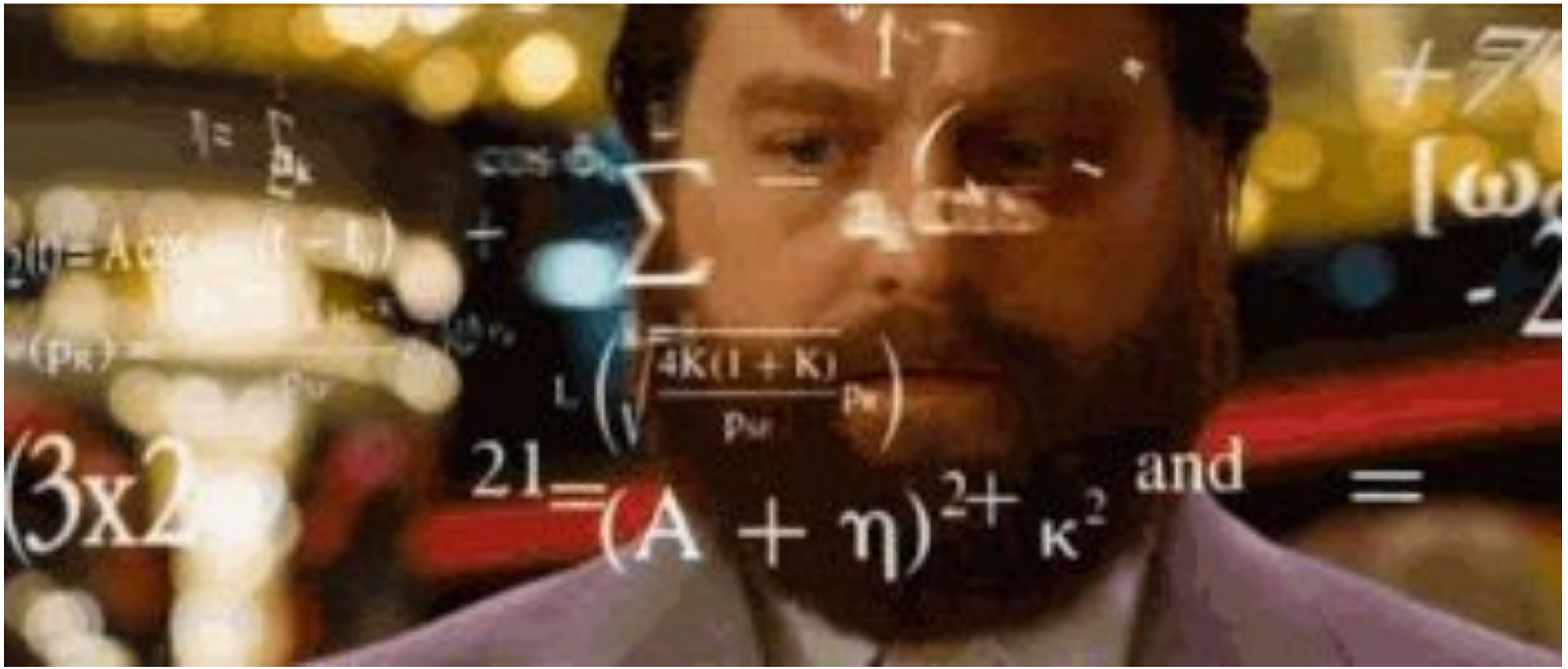
```
from sklearn.preprocessing import MinMaxScaler


transformed_data = MinMaxScaler().fit_transform(X)
```

Coding demo

Notebook



Exercise **time**!



Reviewing last week concepts (self correcting NBs)

Tensor Data Management

Data Analysis

Iris Dataset

Harder Challenge: Fashion MNIST

2) **Classification Exercise**

Classification Exercise

Given a dataset, make a program using the sklearn library, that divides the dataset in two parts, one for training and another for testing and train a classification algorithm of your choice from Sklearn (like SVM_Classification) to correctly classify the dataset.

1.- Iris dataset (load_iris)

2.- Wine dataset (load_wine)

3) **Regression Exercise**

Given the Boston dataset (load_boston), do a program using the library sklearn, that divides the dataset in two parts (one for training and another one for testing) and train the algorithm of regression to predict the housing prices given the missing values.

Harder Challenge: Boston Notebook (self-graded but with harder challenges)

Tip

-If there is some variable that is not correlated with the objective result, you can erase it to simply the algorithm in order to work in less dimensions.