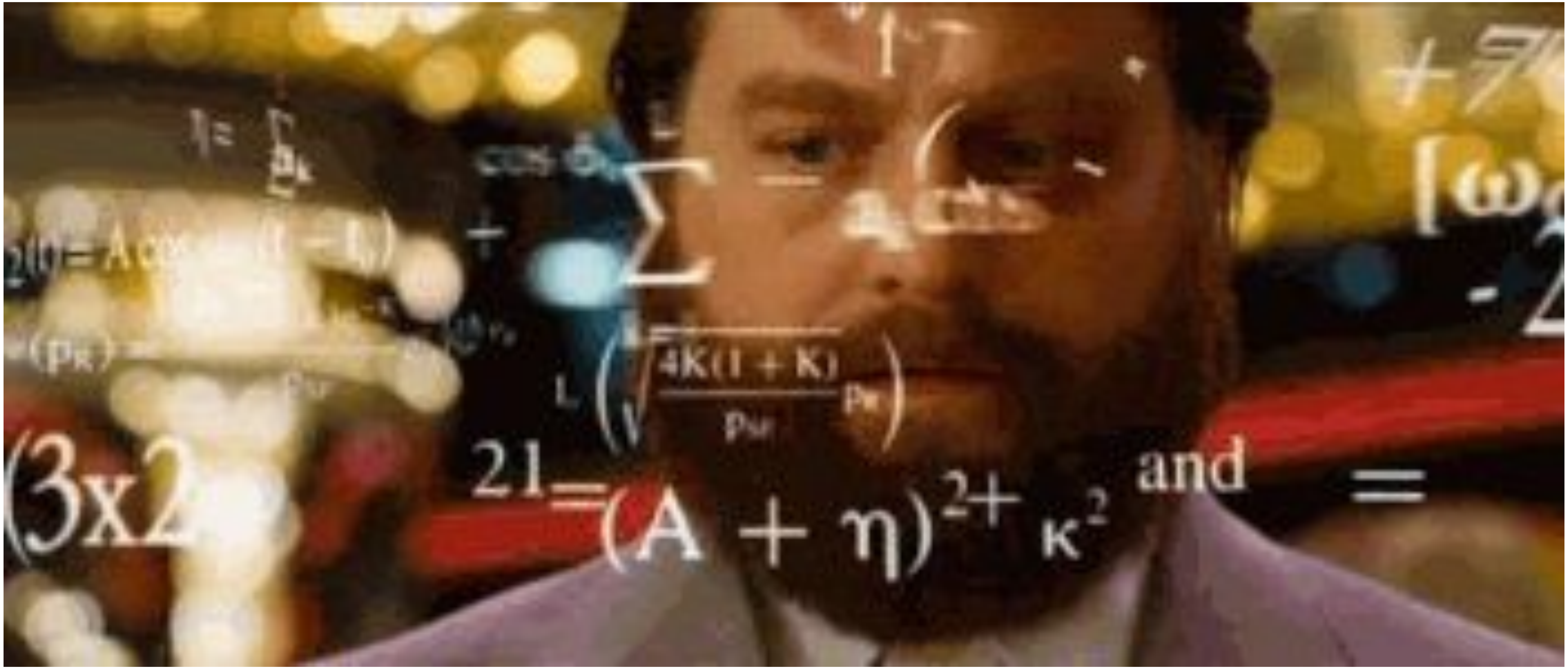
Intro to Machine Learning (2)

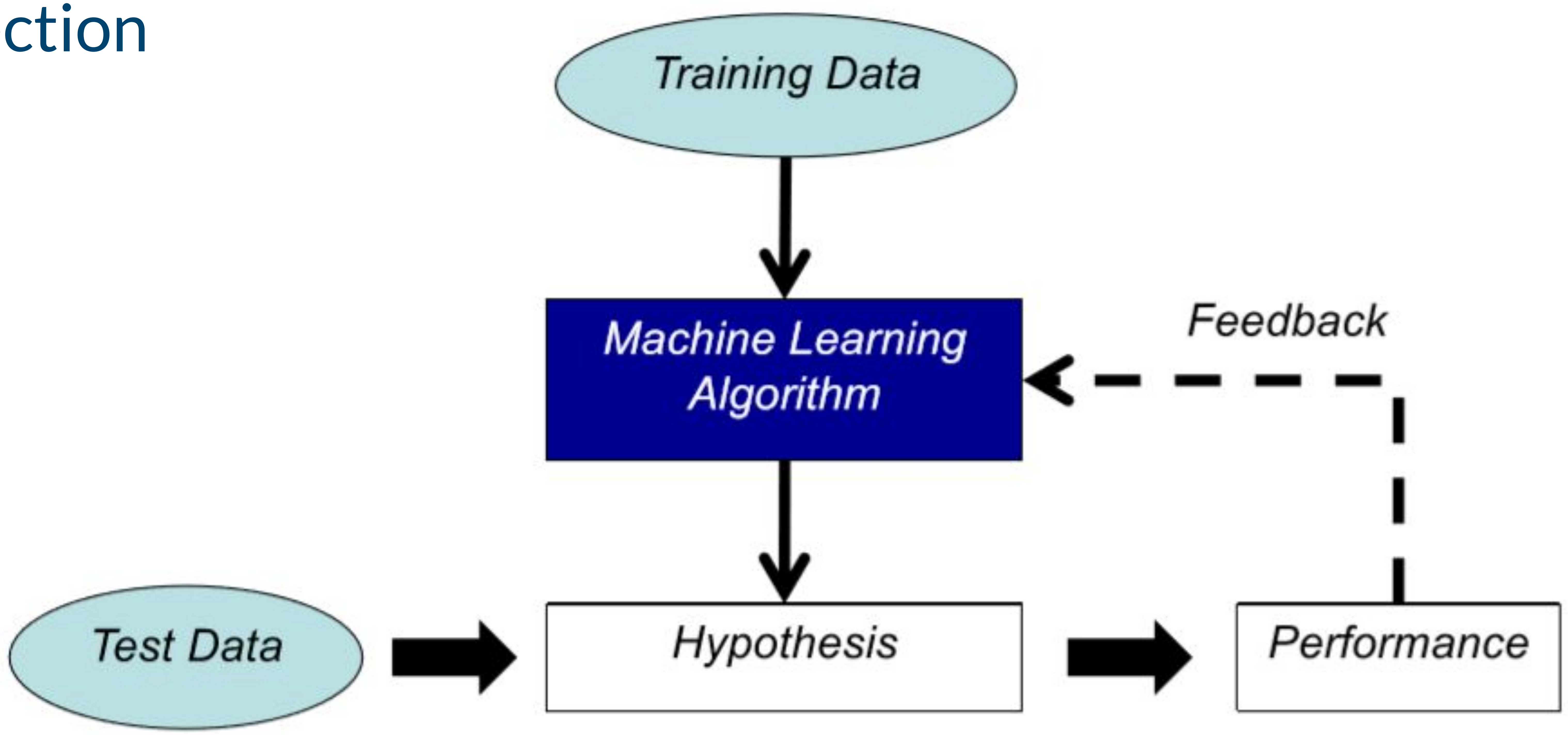
Providing universal access to AI education and practice

Important things to remember from yesterday

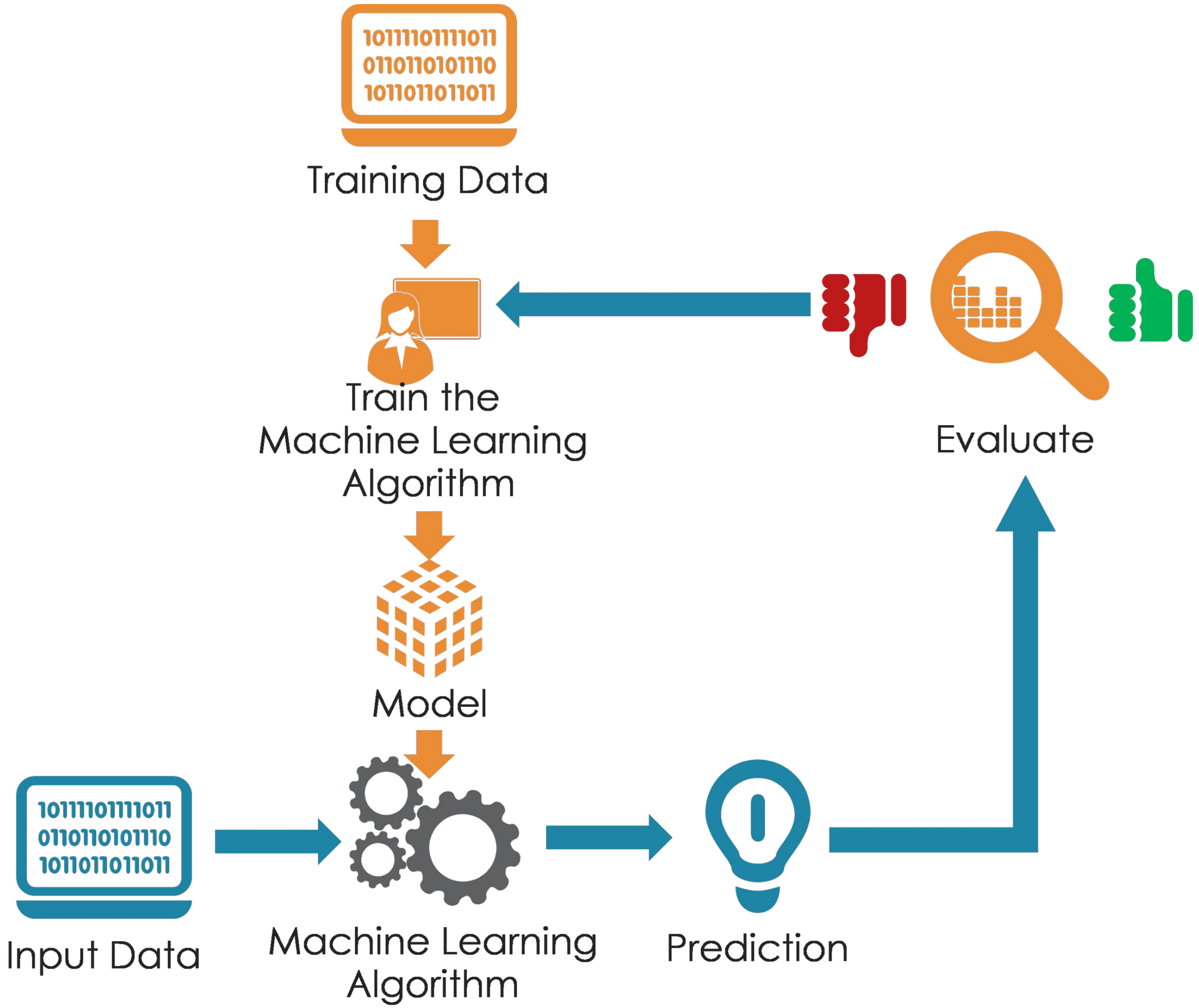


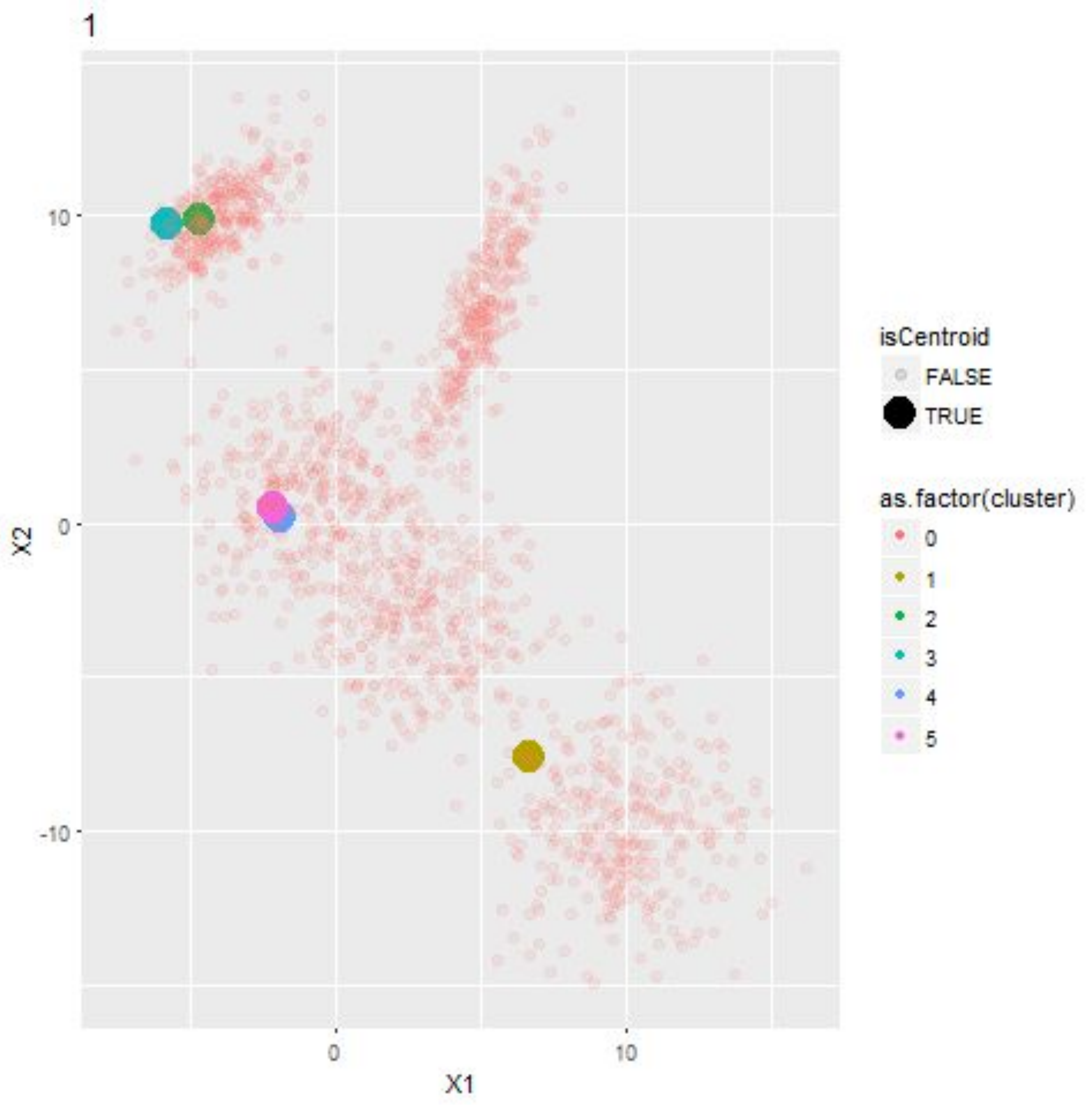
To **review** and **summarize**: ML Process

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

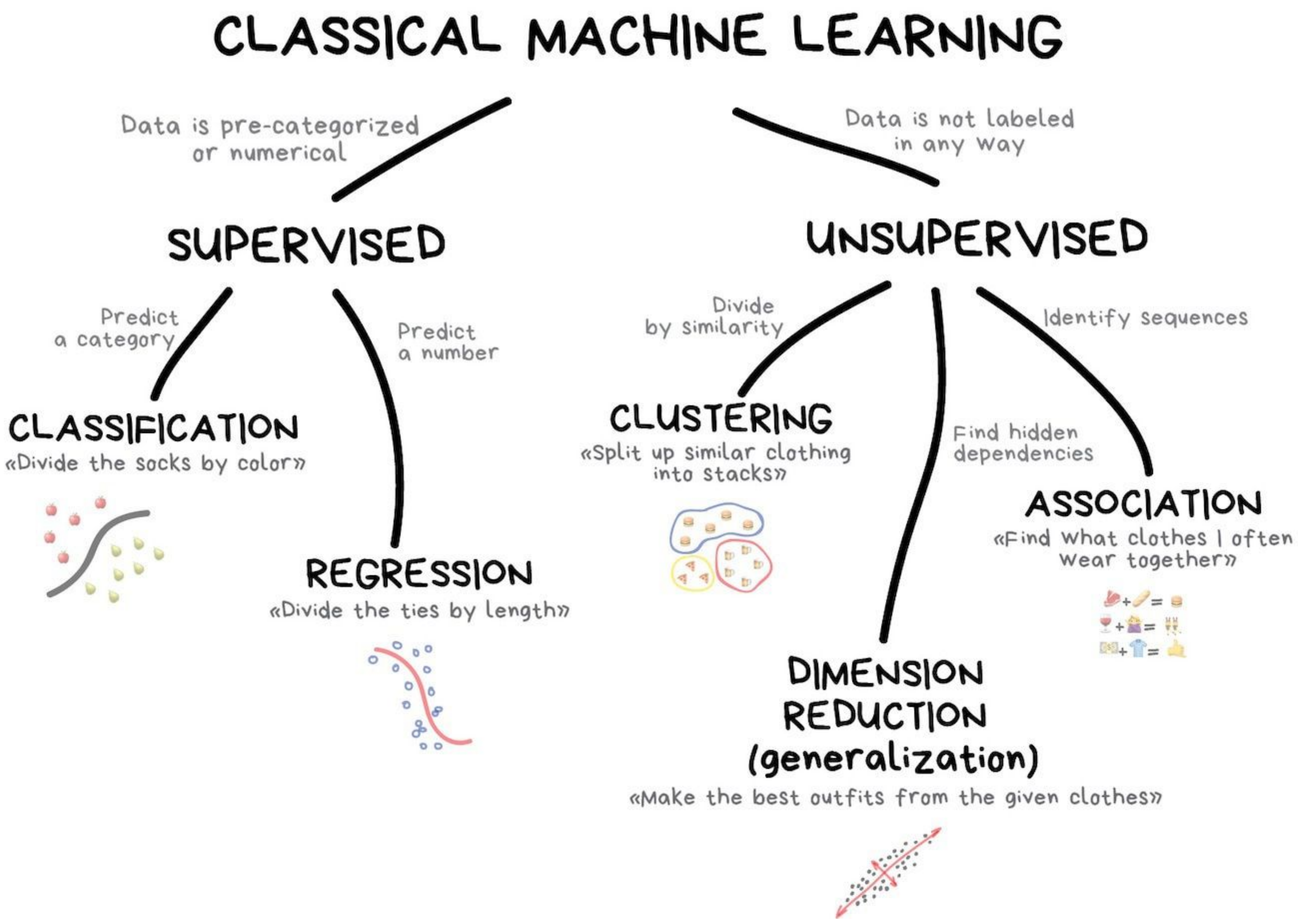


Machine learning process





Fast Machine Learning **Overview**

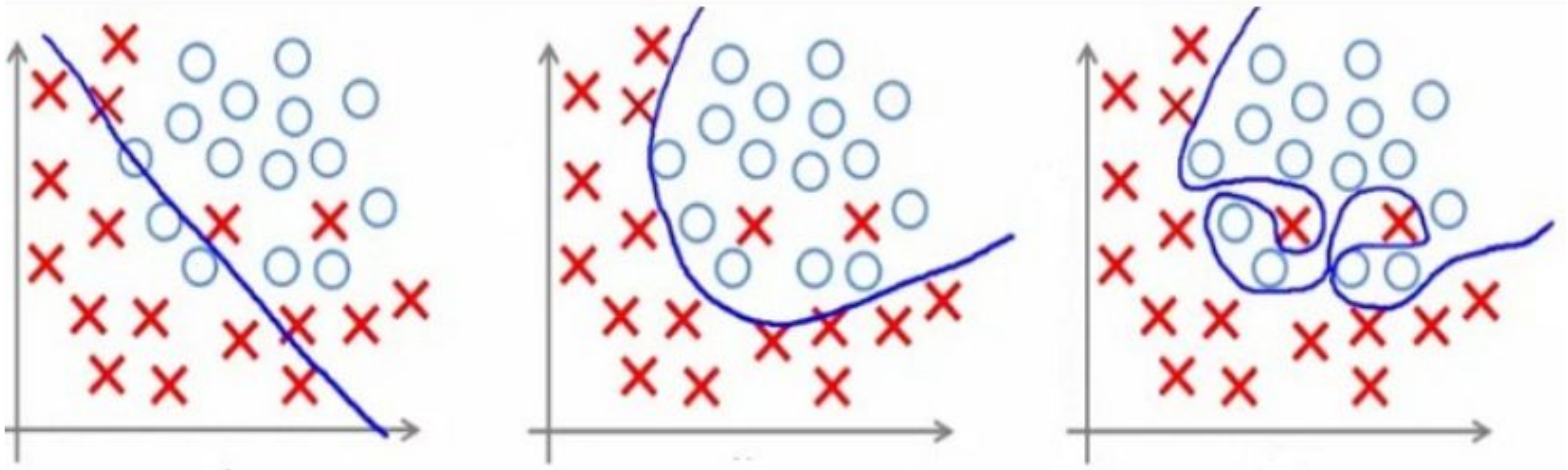


Designing a Machine Learning System

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.

Under-fitting and Overfitting



Under-fitting

(too simple to explain the variance)

Appropriate-fitting

Over-fitting

(forcefitting -- too good to be true)

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)
```

Resonable questions after yesterday

- What should be my X and y?
- Why do we transform the data between 0 and 1?
- What is one-hot encoding?
- Do I always have to do a train - test split?
- What model should we use? **You haven’t explained any of them**

I will try to remove your training wheels, **constantly**

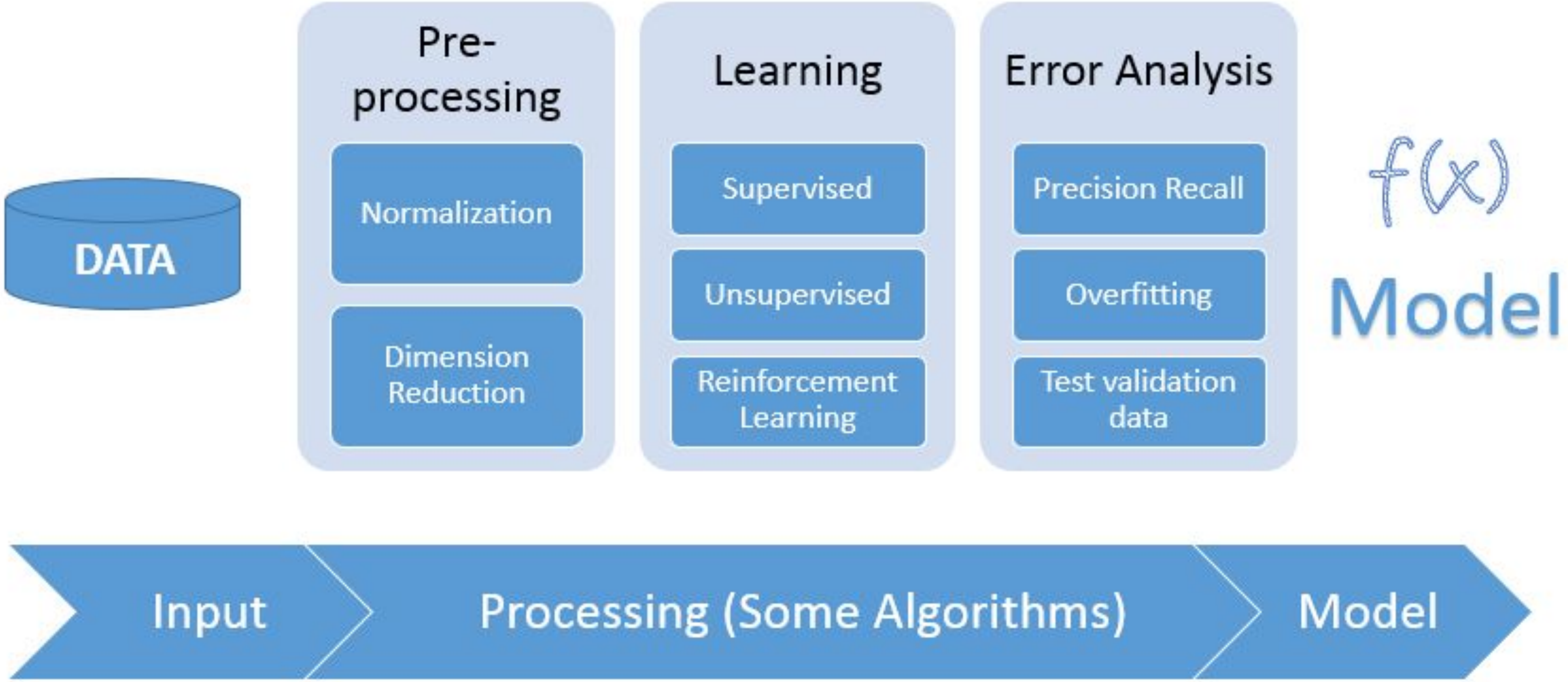
Sorry, not sorry!

Resonable questions after yesterday

- What should be my X and y?
- Why do we transform the data between Max - Min or 0 and 1?
- What is one-hot encoding?
- Do I always have to do a train - test split?
- What model should we use? **You haven’t explained any of them**

The **Golden Process**

TIP: It never changes. Normally, the more complex the data, the harder it is.



What should be the **x** and what should be the **y**?

What you want to feed into your system → x

What you want to get out of your system → y

```
Cool
1000 250
```

Out[19]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
92	Linda	Female	5/25/2000	5:45 PM	119009	12.506	True	Business Development
65	Steve	Male	11/11/2009	11:44 PM	61310	12.428	True	Distribution
445	Chris	Male	12/12/2006	1:57 AM	71642	1.496	False	NaN
732	Henry	Male	5/12/1986	2:04 AM	59943	1.432	False	Finance
352	NaN	Male	10/9/2011	9:29 AM	69906	4.844	NaN	Engineering
293	Jesse	Male	10/25/1999	3:35 PM	118733	9.653	False	Marketing
456	Deborah	NaN	2/3/1983	11:38 PM	101457	6.662	False	Engineering
171	Patrick	Male	8/17/2007	3:16 AM	143499	17.495	True	Engineering
562	Sara	NaN	10/7/1983	1:35 PM	87713	18.863	True	Legal
320	NaN	Female	7/8/2008	11:40 PM	62960	14.356	NaN	Sales
568	Susan	Female	4/18/1986	9:31 AM	90829	19.142	False	Marketing
775	Rose	Female	11/3/1999	9:06 AM	75181	6.060	True	Finance
32	NaN	Male	8/21/1998	2:27 PM	122340	6.417	NaN	NaN

Practice in **identifying x and y's**

What you want to feed into your system → x

What you want to get out of your system → y

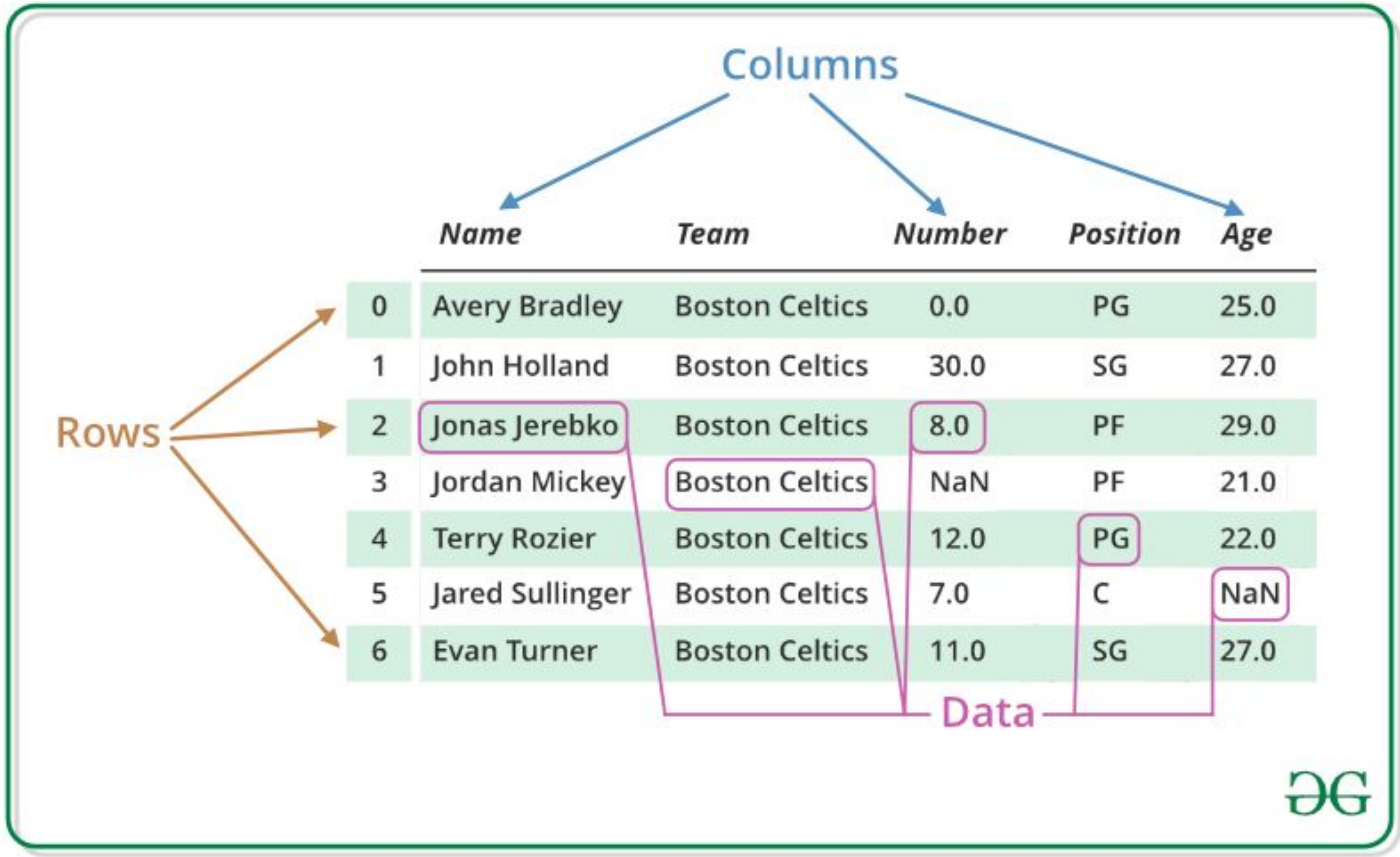
	pregnancies	glucose	diastolic	triceps	insulin	bmi	dpf	age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

	color	age	height
Jane	blue	30	165
Niko	green	2	70
Aaron	red	12	120
Penelope	white	4	80
Dean	gray	32	180
Christina	black	33	172
Cornelia	red	69	150

Practice in **identifying x and y's**

What you want to feed into your system → x

What you want to get out of your system → y



In [138]:

```
df = pd.read_csv('../../data/training_dataset_500.csv')

df[df['House']==1]
```

Out[138]:

	ID	Label	House	Year	Month	Temperature	Daylight	EnergyProduction
0	0	0	1	2011	7	26.2	178.9	740
1	1	1	1	2011	8	25.8	169.7	731
2	2	2	1	2011	9	22.8	170.2	694
3	3	3	1	2011	10	16.4	169.1	688
4	4	4	1	2011	11	11.4	169.1	650
5	5	5	1	2011	12	4.2	199.5	763
6	6	6	1	2012	1	1.8	203.1	765
7	7	7	1	2012	2	2.8	178.2	706
8	8	8	1	2012	3	6.7	172.7	788
9	9	9	1	2012	4	12.6	182.2	831
10	10	10	1	2012	5	17.6	214.2	955
11	11	11	1	2012	6	20.8	143.0	837

Practice in **identifying x and y's**

What you want to feed into your system → x

What you want to get out of your system → y

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
5	Amir Johnson	Boston Celtics	90.0	PF	29.0	6-9	240.0	NaN	12000000.0
6	Jordan Mickey	Boston Celtics	55.0	PF	21.0	6-8	235.0	LSU	1170960.0
7	Kelly Olynyk	Boston Celtics	41.0	C	25.0	7-0	238.0	Gonzaga	2165160.0
8	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
9	Marcus Smart	Boston Celtics	36.0	PG	22.0	6-4	220.0	Oklahoma State	3431040.0

Why do we **transform** the data? Max//Min or 0//1

- Normalization is a technique often applied as part of data preparation for machine learning.
- The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values.
- For machine learning, every dataset does not require normalization. It is required only when features have different ranges, which is **most** of them...

Normalization Formula

$$X_{normalized} = \frac{(X - X_{minimum})}{(X_{maximum} - X_{minimum})}$$

$$x_{new} = \frac{x - \mu}{\sigma}$$

What is one hot encoding?

- Another previous question could be… how can the machine understand categorical values? **They must be transformed into numbers!!**
- Actually, when making a classification, that is not needed since we can understand the numbers as classes. **What happens though if we want to "use" this data?** → We need to use techniques to transform categorical into numerical values.
- **Is it clear?** Tell me!

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

→

One Hot Encoding

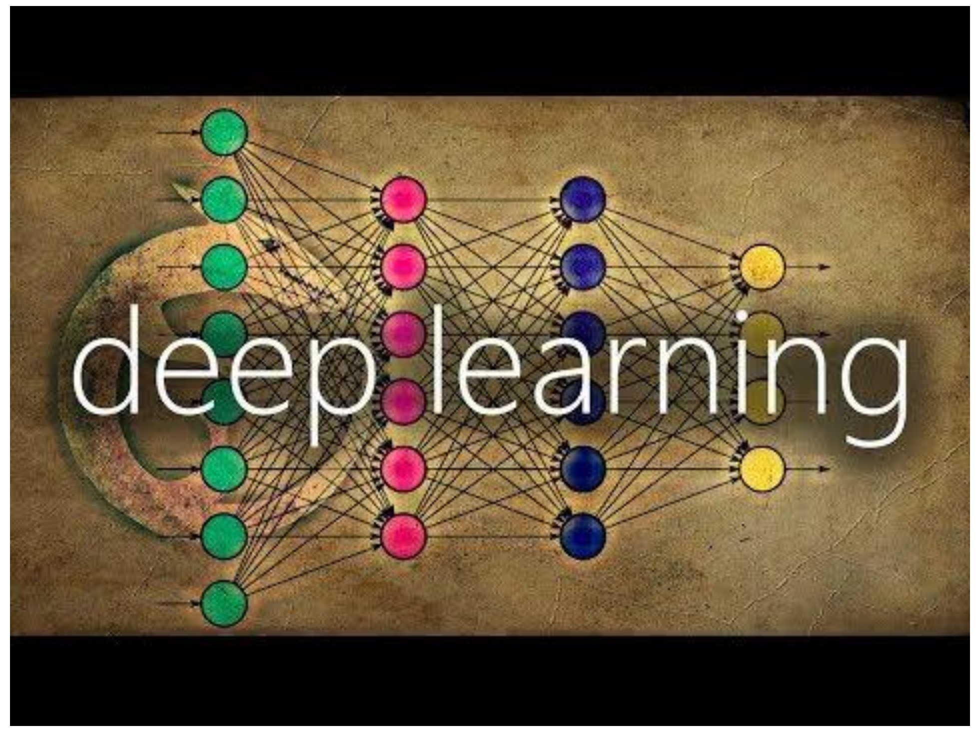
Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

Practical example of one-hot encoding

CompanyName	Categoricalvalue	Price
VW	1	20
Acura	2	10011
Honda	3	50000
Honda	3	10000

One hot encoding

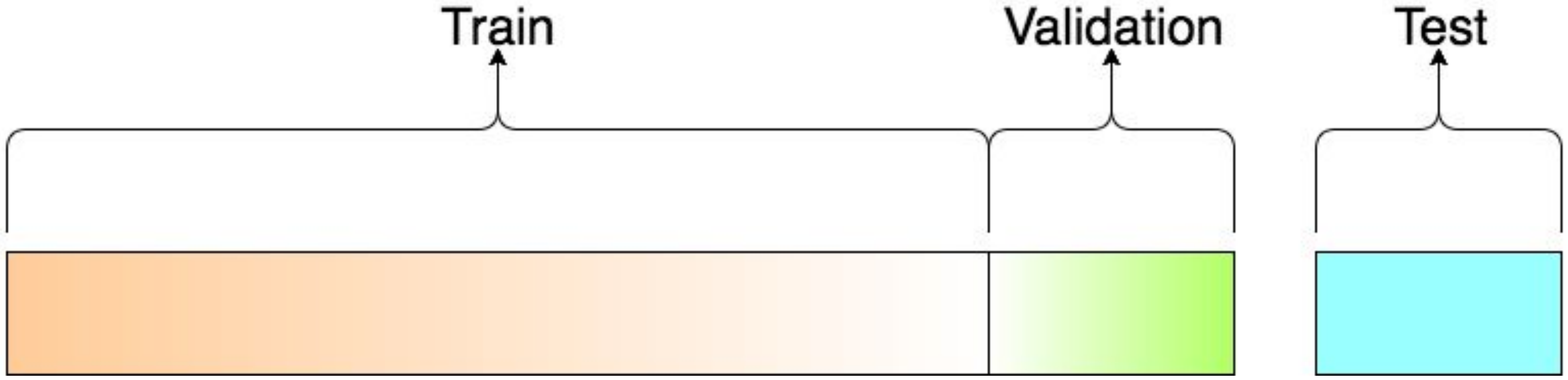
VW	Acura	Honda	Price
1	0	0	20000
0	1	0	10011
0	0	1	50000
0	0	1	10000



- If we don’t hot encode, and just transform the categories into numbers, internally our machine learning system is going to approximate the price. Imagine that our model calculates the average price between a VW and a Honda… (1+3)/2 = 2…. Is the average an Acura? **Not really**!

Do I always have to do a train - test split?

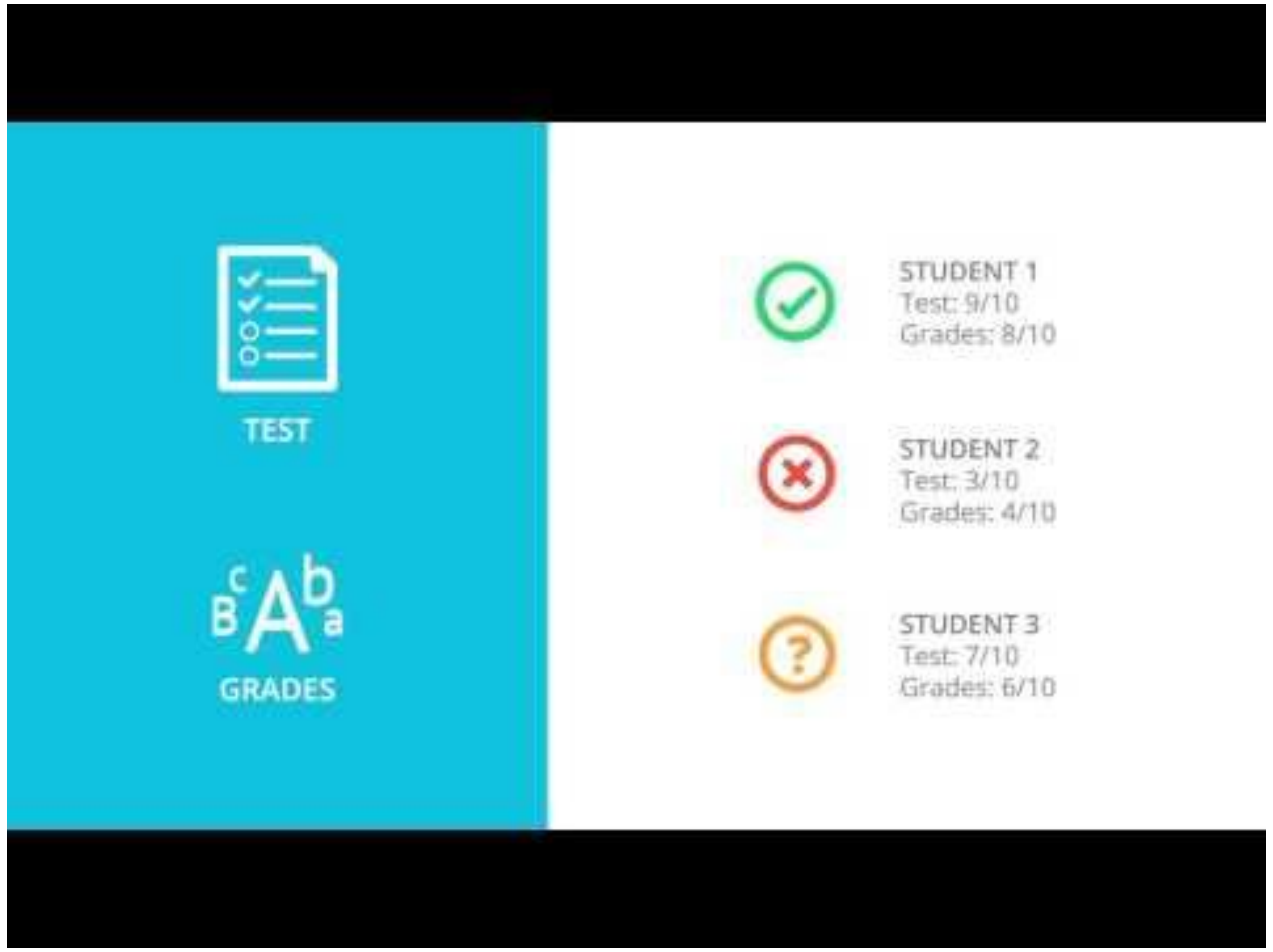
- Yes, there are several options but you must always **train / test**:
- So… what is the harm in combining them at the end for better accuracy? In real life if far from it as you never know if your data will evolve.
- We should strive to make models that generalize and perform well without As a good Data Scientist you should strive to make a model flow that is generalizable and performs well without any additional changes.
- This article includes a much more [detailed explanation] on several methodologies.



What model should we use?

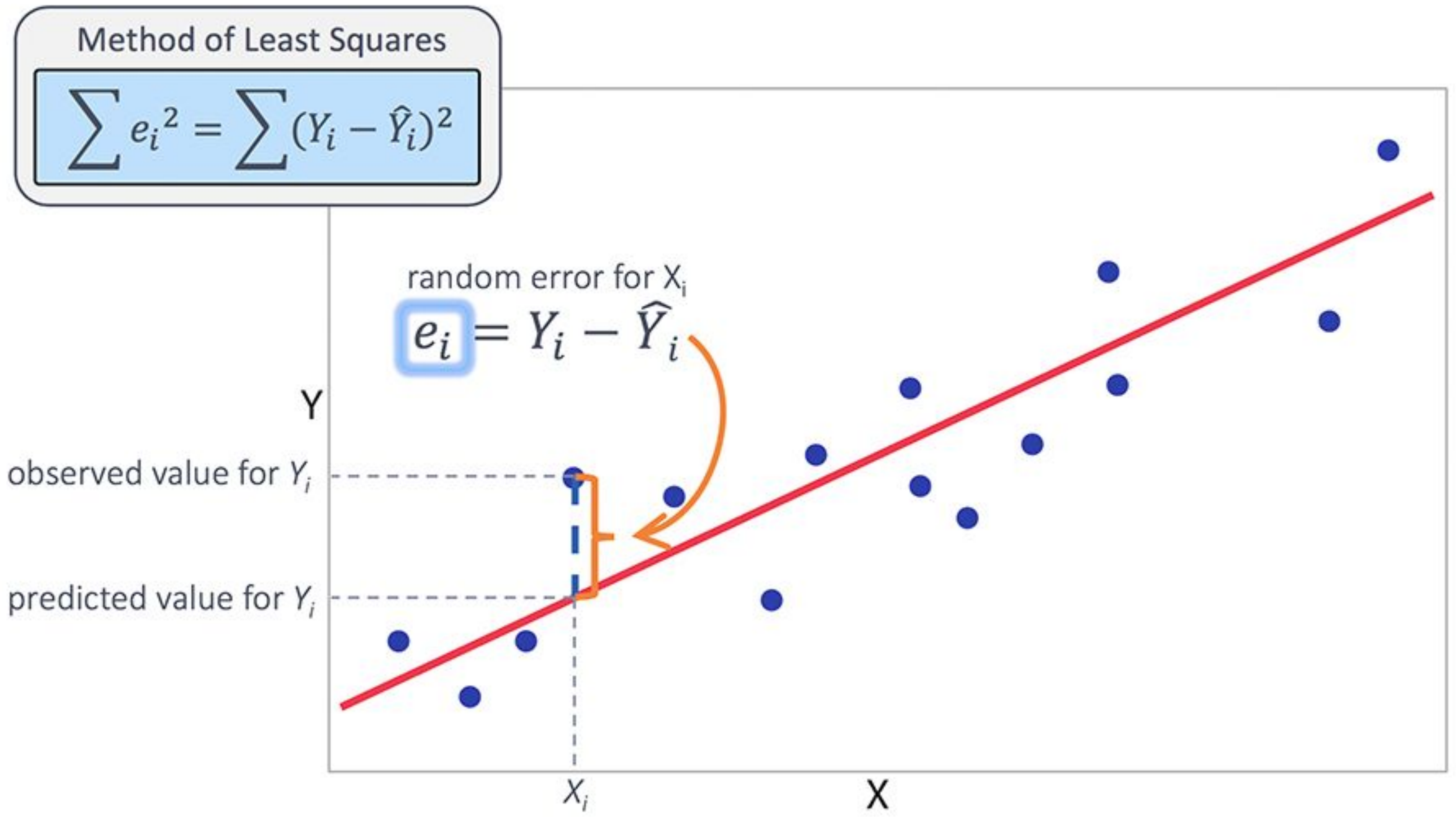
- We will actually go into quite some detail for all the possible models
- Knowing what each model does is a good idea to pick the better candidates
- But being extremely honest, in reality, you are likely to **pick the model** that **better performs with your data**.
- Today, let's place ourselves in the position of a data scientist striving to achieve the best performance. **How far will you go?**

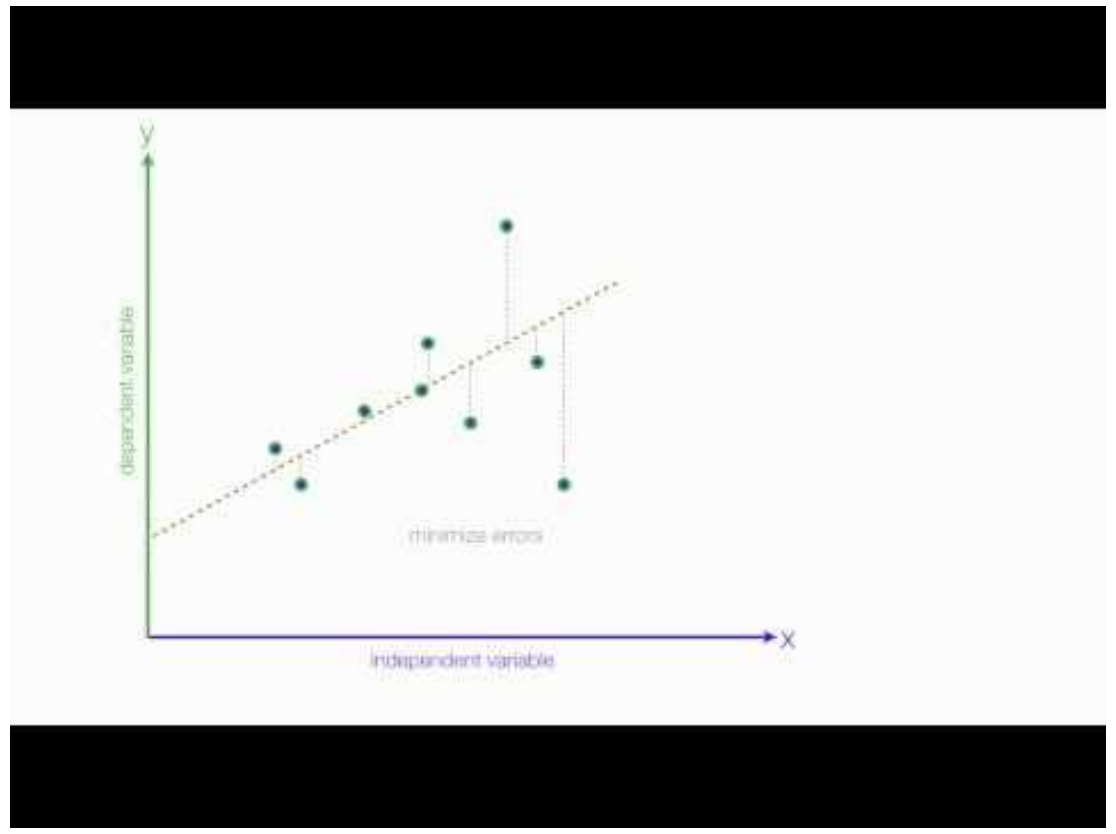
Okay okay, short overview first! **Classification**



- What do you think? A student 3 with a 7 on the test and a grade of 6, gets accepted or not?

Okay okay, short overview first! **Linear Regression**





Dependent Variable

Population Y intercept

Population Slope Coefficient

Independent Variable

Random Error term

$$Y_i = \underbrace{\beta_0 + \beta_1 X_i}_{\text{Linear component}} + \underbrace{\varepsilon_i}_{\text{Random Error component}}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

$$Y = X\beta + \varepsilon$$

Top - Down Method

Another challenge today

Kaggle Machine Learning (3h+)

Iris Dataset (30min)

Classification Exercise (30 min)

Regression Exercise (1h)

Titanic Challenge (1h)

6 hours of work starting now :)

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.- 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.

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.