# MACH

## 1. The ML Landscape

1 Allez sur wooclap.com

2 Entrez le code d'événement dans le bandeau supérieur

Code d'événement
STUSTG

# Bonjour !

# What was the first ML application that became mainsteam ?

- What was the first ML application that became mainsteam ?
  - Tips: back in the 1990s
- Nowadays,  hundreds of ML applications
  - Cite few of them you  use regularly.
- What exactly does it mean for a machine to learn something ?
  - If I download a copy of all Wikipedia articles, has my computer really learned something?  Is it suddenly smarter ?
- In this lesson :
  - Clarify what ML is and why you may use it.
  - Introduction to the fundamental concepts and jargon that every data scientist should know by heart.

# What is Machine Learning ?

- Machine Learning is the science and art of programming computers so that **they can learn from data**
- A more general definition:
  - ML is the « field of study that gives computers the ability to learn without being explicitly programmed » -- Arthur Samuel, 1959
- A more engineering oriented one:
  - A computer program is said to learn from **experience E** with respect to some **task T** and some **performance measure P**, if its performance of T, as measured by P, improve with experience E. -- Tom Mitchell, 1997
  - The spam filter example:
    - What is T, P and E ?

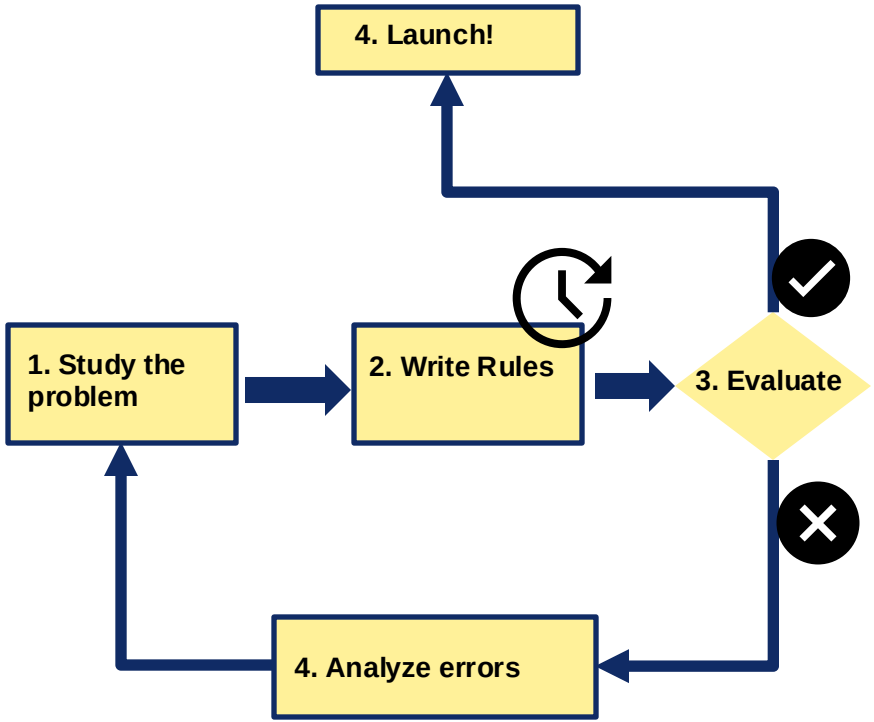
EPITA


# Why use Machine Learning?

# Traditional approach (without ML)

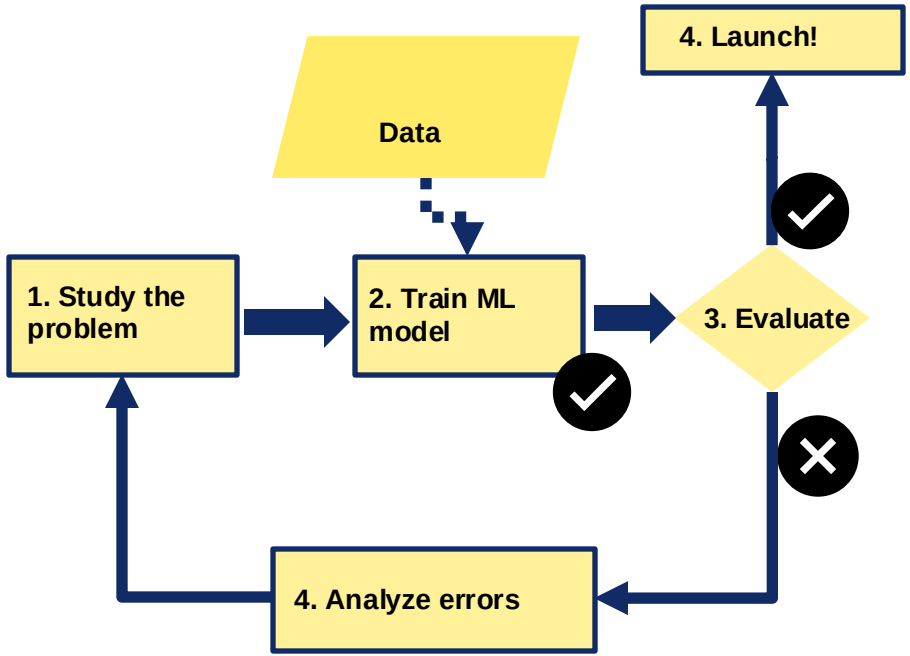
1. **Examine** what spam looks like. Notice some words or phrase (such as 4U, enlarge, free, amazing, …)
2. **Write** a detection algorithm for each the patterns noticed: the program would flag emails as spam if a number of these **rules** were detected.
3. **Test** the program and repeat (1) and (2) until it was good enough to launch.

**Issues: pb difficult → long list of complex rules – hard (impossible) to maintain.**

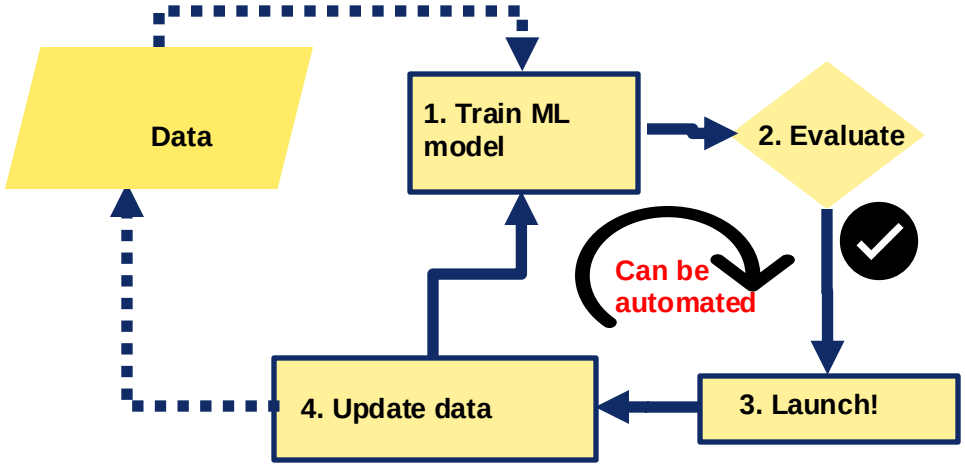


# The Machine Learning approach

- Automatically learns which words and phrases are good predictors
- Program more shorter and most likely more accurate

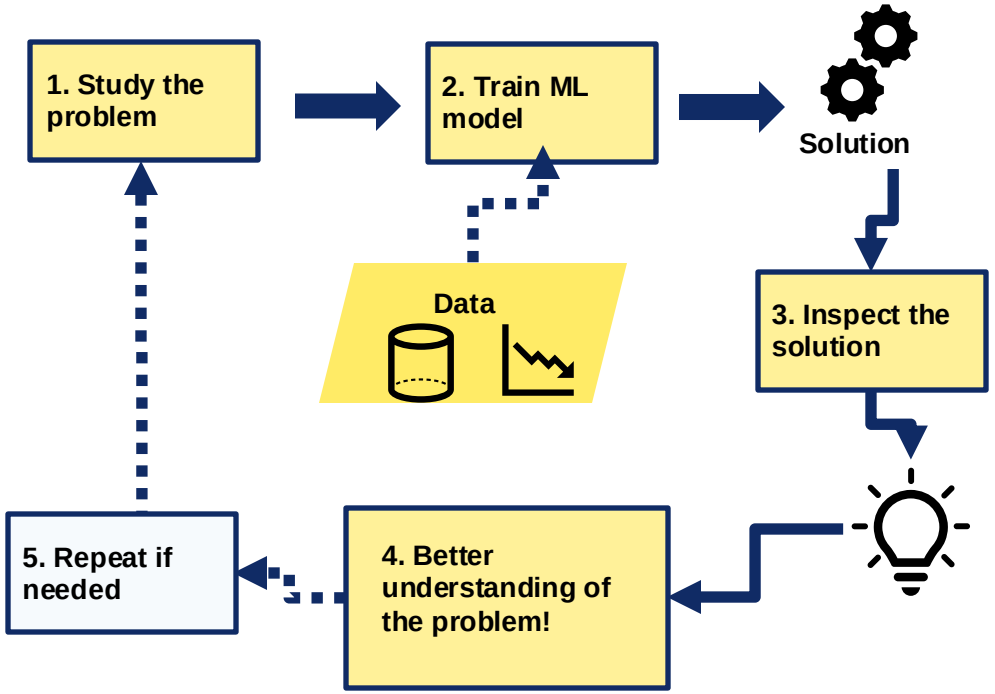


# Automatically adapting to change



# ML can help humans learn

**Discovery of hidden patterns**
**Data mining**



# Summary

- ML is great for:
  - Problems for which a solution require a lot of fine-tunnig or long lists of rules: a ML model can often simplify code and perform better than the traditional approach
  - Complex problems for which using a traditional approach yields no good solution: the best machine learning techniques can perhaps find a solution
  - Fluctuating environnments: a ML system can be easily retrained on new data, always keeping it up to date.
  - Getting insights about complex problems and large amount of data: ML can support new discoveries.


# Examples of Applications

# Concrete examples

- Analyzing images of products on a production line to automatically classified them (DNN S9 IF, SCIA)
- Detecting tumors in brain scans (DNN S9, MLRF S8?)
- Automatically classifying news articles (S8, NLP1, NLP2)
- Automatically flagging offensive comments on discussion forums (S8 NLP2)
- Summarizing long documents (S8, NLP2)
- Creating a chatbot or a personal assistant, (Question/answer modules, NLP2)
- Forecasting your company's revenues next year based on many performance metrics (MACH)
- Making your app react to voice commands (RNN, CNN, Transformers, S9 DNN)
- Detecting credit card fraud (MACH)
- Segmenting clients based on their purchased so that you can design a different marketing strategy (Clustering, MACH)
- Recommending a product that a client may be interested in, based on past purchases (S8, RecSys)
- Building an intelligent bot for a game (S9 RL)
- Building associations and discover knowledge (EDA, S9)
- Traffic forecasting (GNN, S9)
- Molecule toxicity (GNN, S9)
- Link prediction ⇔ item/friend recommendation (MLG S8, GNN S9)


# Types of Machine Learning Systems

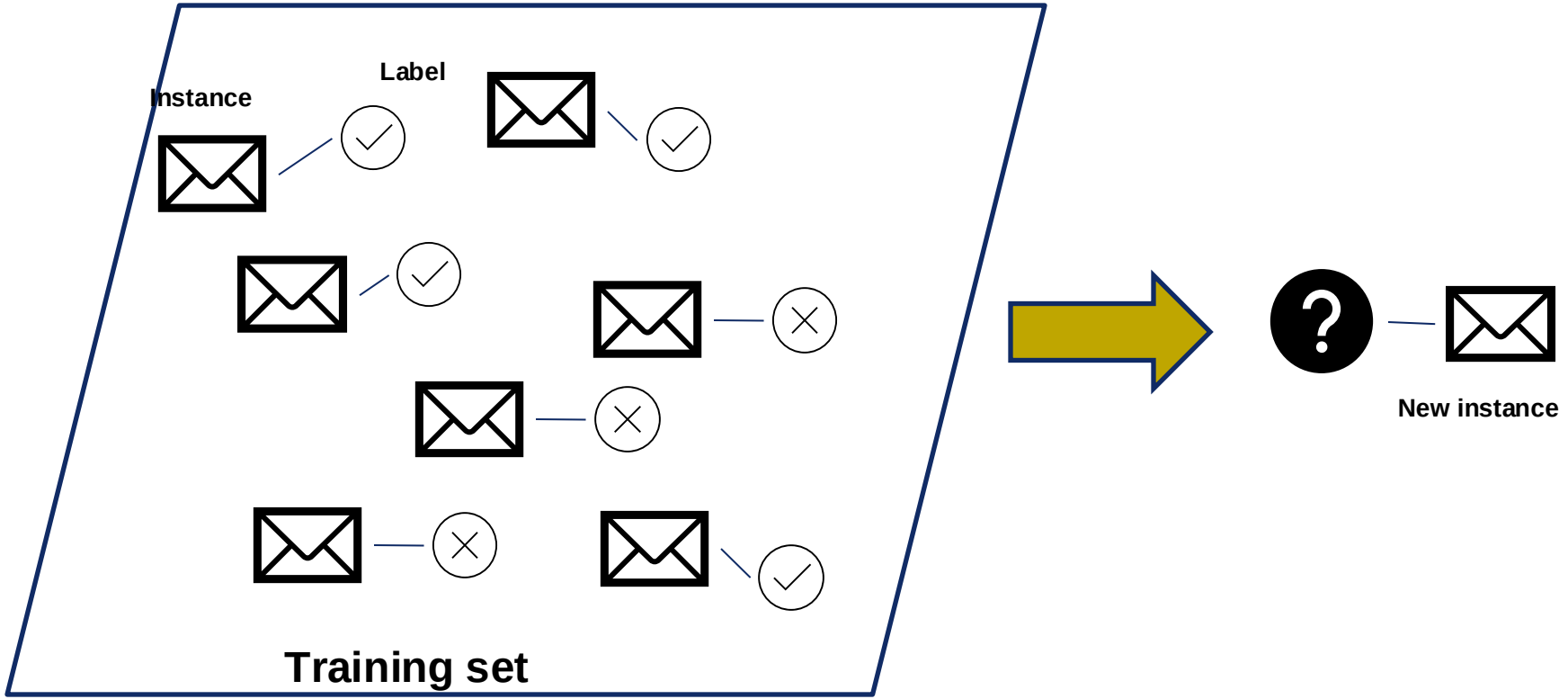
# Criteria to classify ML systems in broad categories

- How they are supervised during training (supervised, unsupervised, semi-supervised, self-supervised, …)
- Whether or not they can learn incrementally on the fly (online vs batch learning)
- Whether they work by simply comparing new data points to known data points , or instead by detecting patterns in the training data and building a predictive model (instance-based vs model-based learning)

Non exclusive criteria:

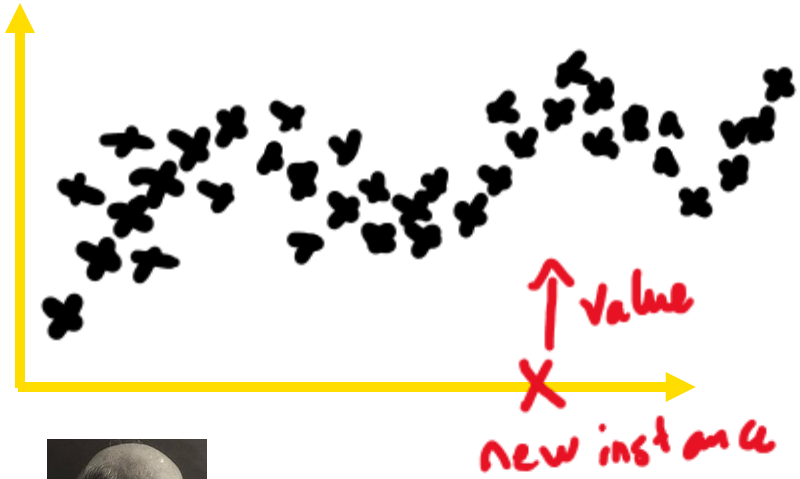
- Spam filter may learn on the fly using a DNN trained on labeled data: online, model-based, supervised learning

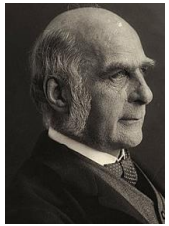
# Training supervision: supervised learning (classification)



# Supervised Learning: **regression**

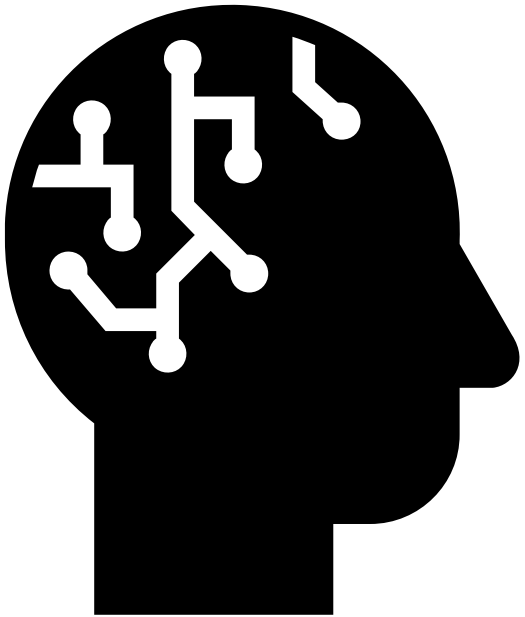
- Predict a **target** numeric value (price of a car, revenue, grades) given a set of **features** (mileage, age, brand, ..)
- Regression models can be used for classification as well and vice versa
  - Logistic regression is commonly used for classification as the output value corresponds to the probability belonging to a given class (e.g., 30% chance of being a spam)
- **Fun fact:** Regression is a statistics term introduced by Francis Galton while he was studying the fact that children of tall people tend to be shorter than their parents. Since the children were shorter, he called this regression to the mean. This name was then appplied to the methods he used to analyze correlation between variables.



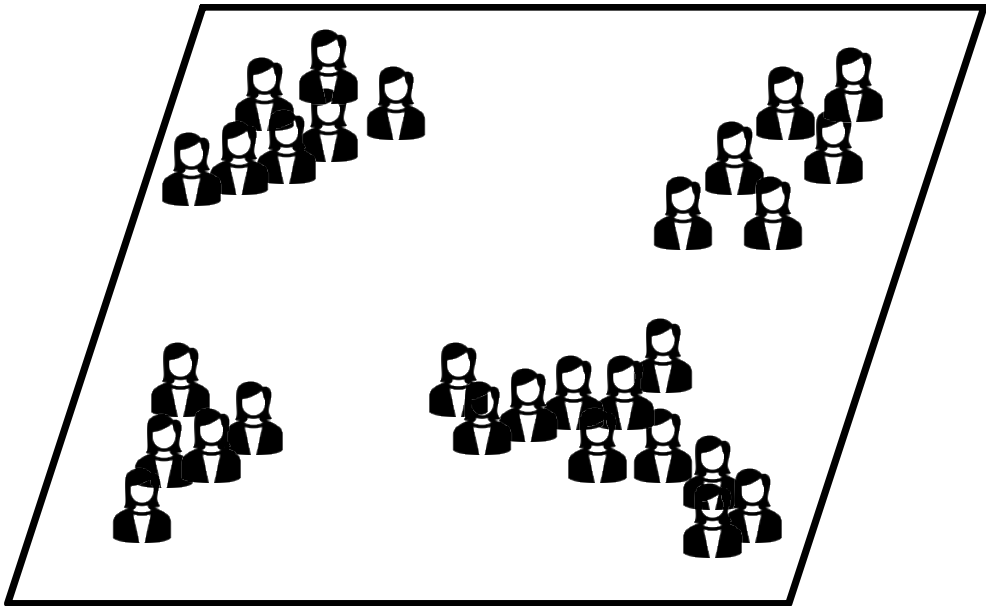
# Vocabulary

- **Target** and **label** are synonyms in supervised learning but:
  - **Target** is more common in **regression** tasks;
  - **Label** is more related for **classification** tasks;
- **Features** a.k.a. *predictors* or *attributes*



# Unsupervised Learning

- The training set is unlabeled
- The system will then tries to learn without a teacher.
  - Clustering to detect groups of similar individual samples
  - Finding association between attributes: association rule mining
  - Dimensionality reduction for insightful visualizations;
  - Anomaly detection

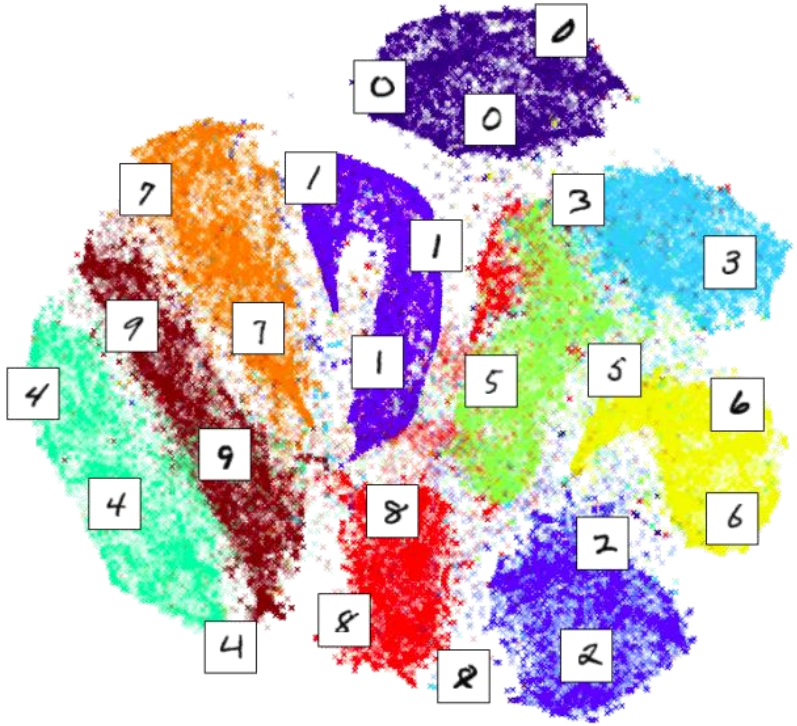


# Clustering

- Aim at finding groups of similar objects
- Relies on a similarity / distance measure
- Kmeans, hierarchical clustering, spectral clustering, density based clustering …
- How to assess the results without no ground truth ?
- We will have a lesson on clustering !

# Dimensionality reduction

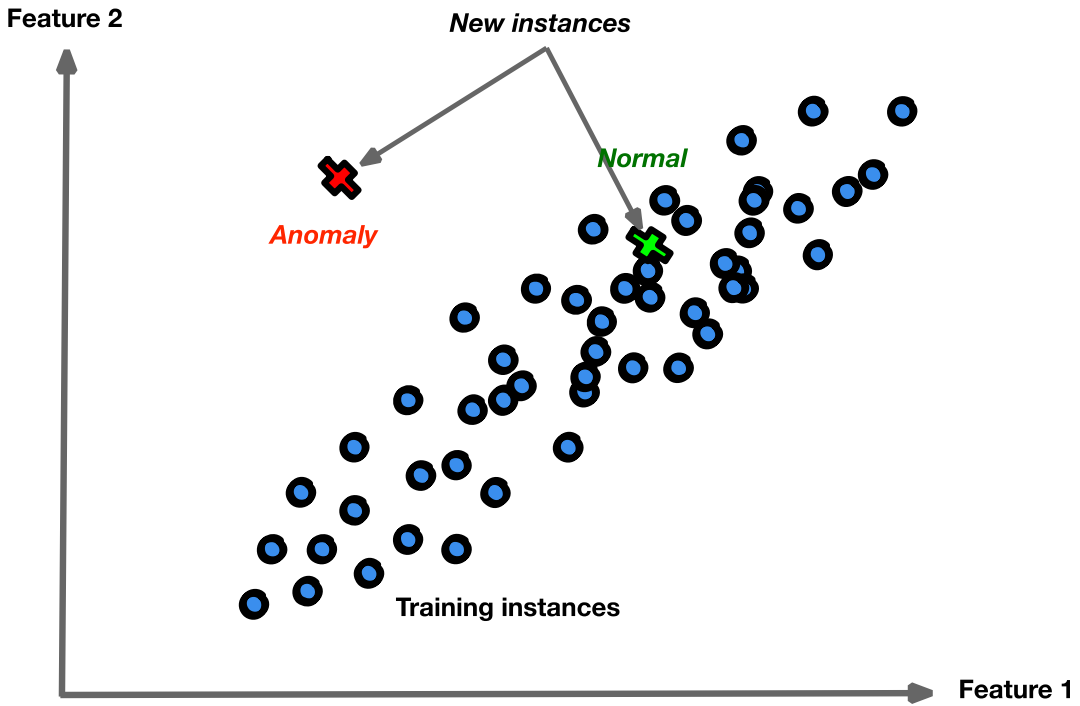
- Simplify the data without loosing too much information
- PCA
- t-SNE
- Often a good idea to reduce the dimensionality before performing another ML algorithm



*MNIST images visualised in two dimensions using t-SNE.*

http://colah.github.io/posts/2014-10-Visualizing-MNIST/

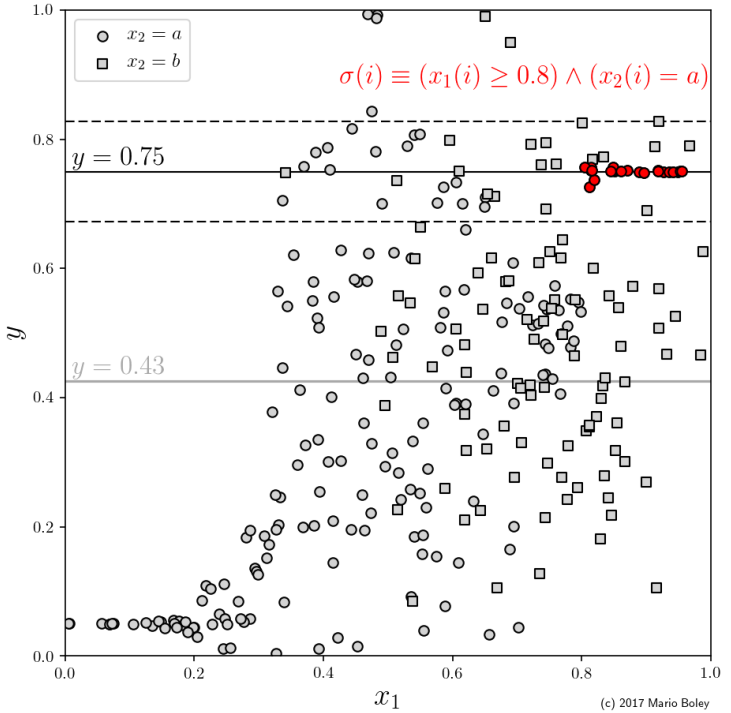
# Anomaly Detection



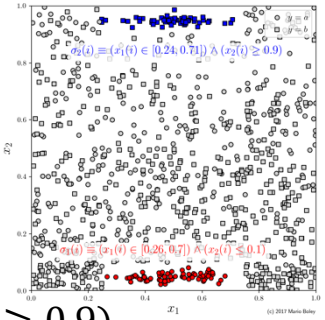
# Association learning /Subgroup discovery / Pattern Mining

Looking for **local models** within the data

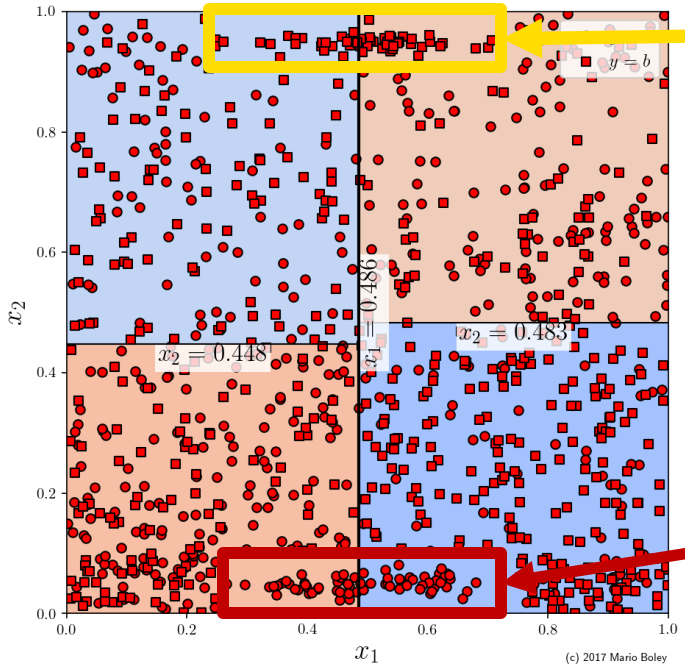
- «A complex theory of everything might be of less value than a simple observation about a specific part of the data space»
- Identifying interesting subspace and the power of saying *« I don't know for other points »*
- 1 lesson (all) + GDM (S8) EDA (S9) for SCIAg



Global Modelling is guided by the global picture and may not uncover some interesting insights while local patterns make it possible.



Decision tree with 0.7 accuracy




$q_1 = (x_1(i) \in [0.24, 0.71] \wedge x_2 \geq 0.9)$

**Two subgroups with 1.0 accuracy**

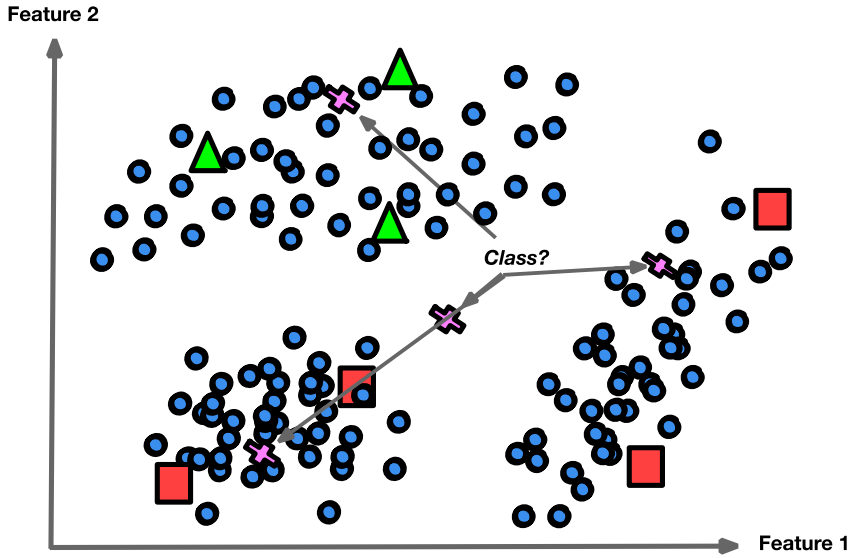
$q_2 = (x_1(i) \in [0.26, 0.7 \wedge x_2(i) \leq 0.1)$

# Fun fact (or success story)

Diappers and beers often bought together: the first examples of association rules that support the success of this concept.

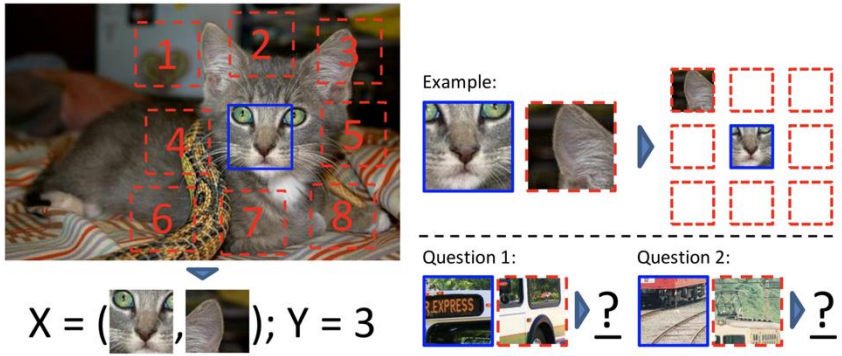
# Semi-supervised learning

- Labelling data is usually time-consumming and costly
- ➔ often have much more unlabeled instances than few labeled ones.



# Self-supervised learning

- Generating a fully labelled dataset from a fully unlabelled one

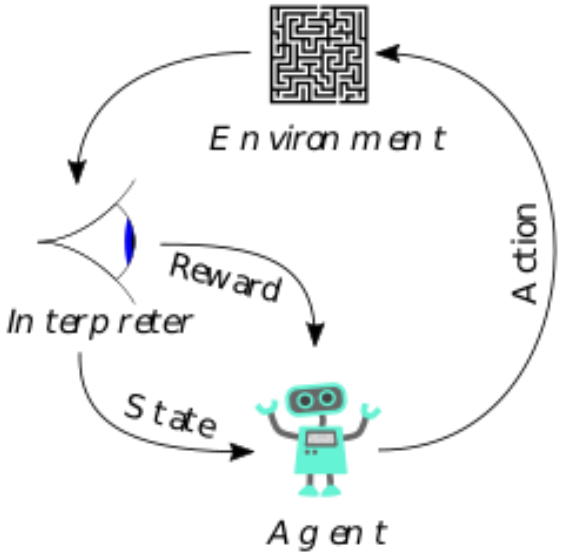


# Transfer learning

- Transferring knowledge from one task to another

# Reinforcement Learning

- Learn (sequences of) actions to take from a experienc
- Games, robots, …
- Q-learning, MCTS, Markov models





# Batch vs Online Learning

Another criterion to classify ML systems: whether or not the system can learn incrementally from a stream of data

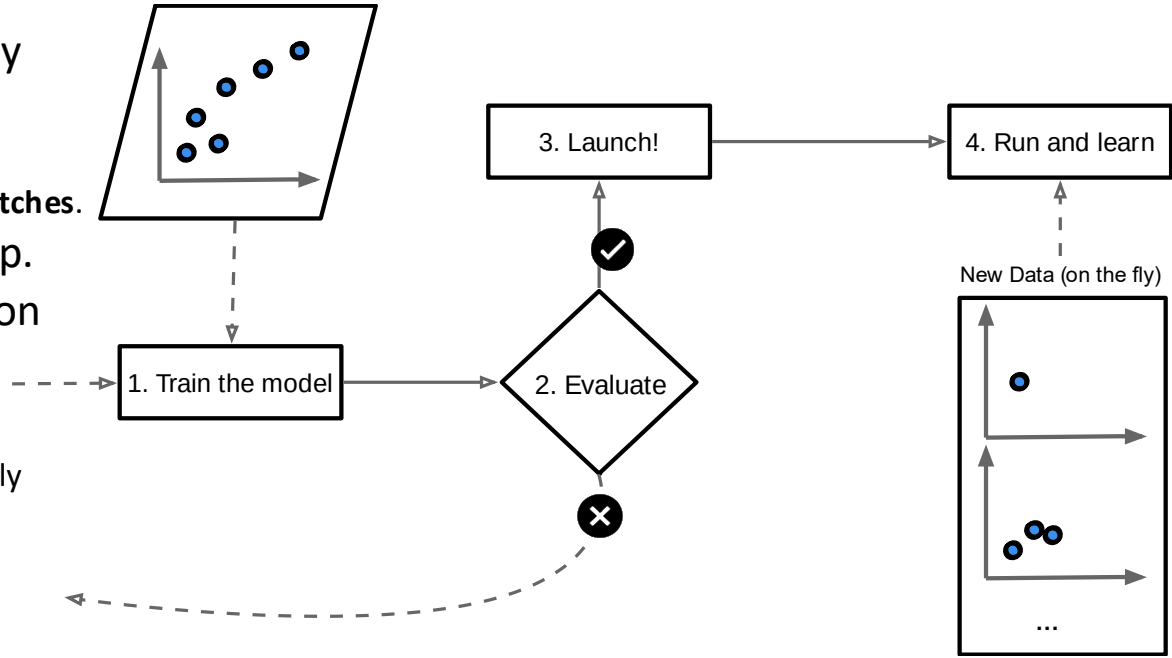
# Batch Learning

- System not able to learn incrementally: it must be trained using all the available data
- Costly in time and computing resources
  - Done Offline
- Pb: **a model's performance tends to decay slowly over time**
  - The world continues to evolve while the model remains unchanged
  - Phenomenon a.k.a. **model rot** or **data drift**.
  - With fast evolving systems (financial market), the decay is faster
  - Need to retrain with updated data.
- When your resources are limited

.....

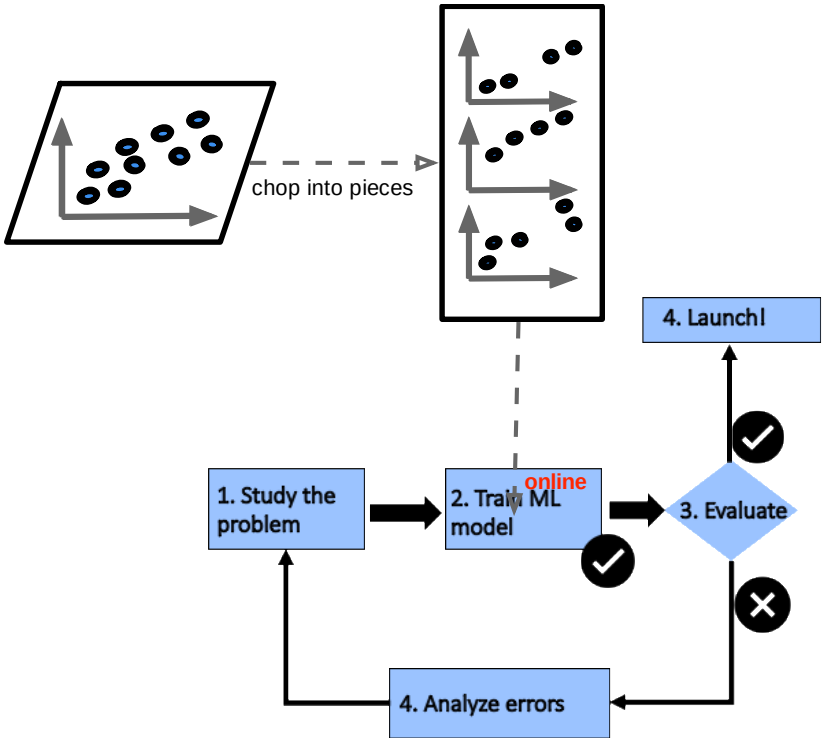
# Online Learning

- The system is trained incrementally according to a sequence of:
  - Individual instances,
  - Small groups of instances aka **mini-batches**.
- Each learning step is fast and cheap.
- The system learns about the data on the fly
- Useful for:
  - Systems that need to change extremely rapidly
  - Systems with limited resources



# To train model on huge datasets that cannot fit in main memory

- Aka out-of-core learning
- The algorithm loads part of the data, runs a training step on that data, and repeats until it has run on all of the data.

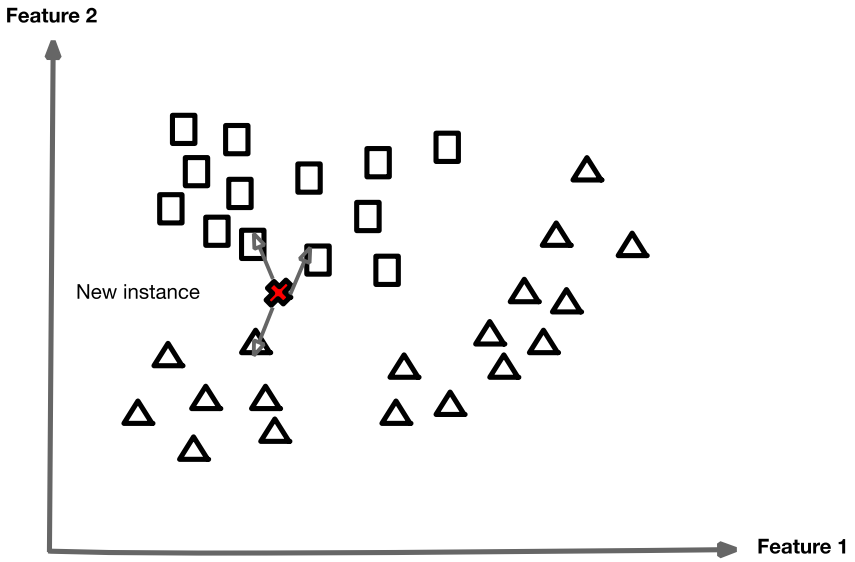


# Instance-based vs Model-based learning

What about generalization?

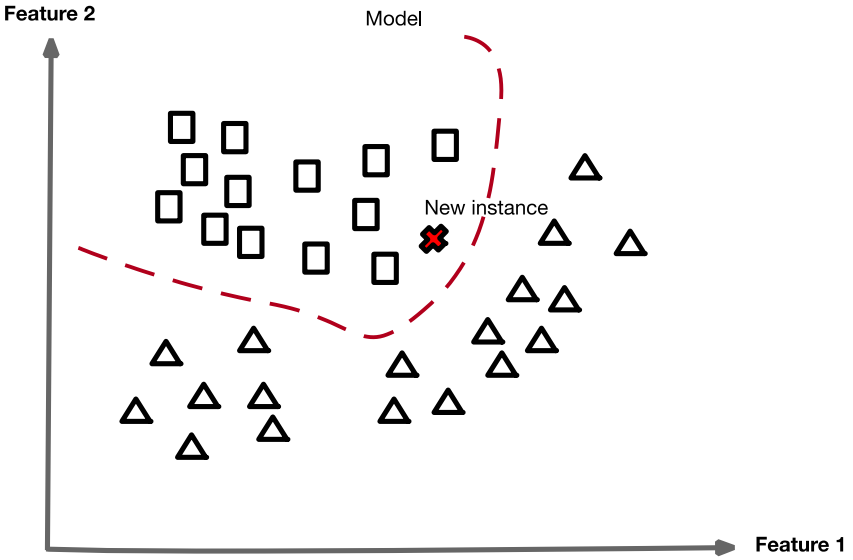
# Instance-based Learning

- The most trivial (and natural) form of learning : classify **similarly** to what you already observed
- The system generalizes to new cases by using a similarity measure to compare them to the learned example (or a subset of them).
- A new instance would be classified according to the majority class of the most similar instances.
- A.k.a. K-Nearest Neighbors (KNN)
    - What about k ?



# Model-based learning

- Build a model of the trained example and then use that model to make **predictions**.

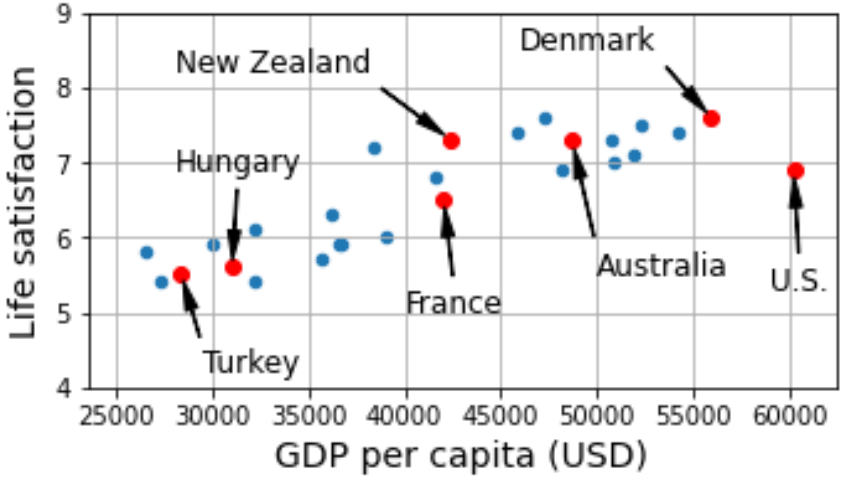


# A typical machine learning workflow

Let’s go through the code …

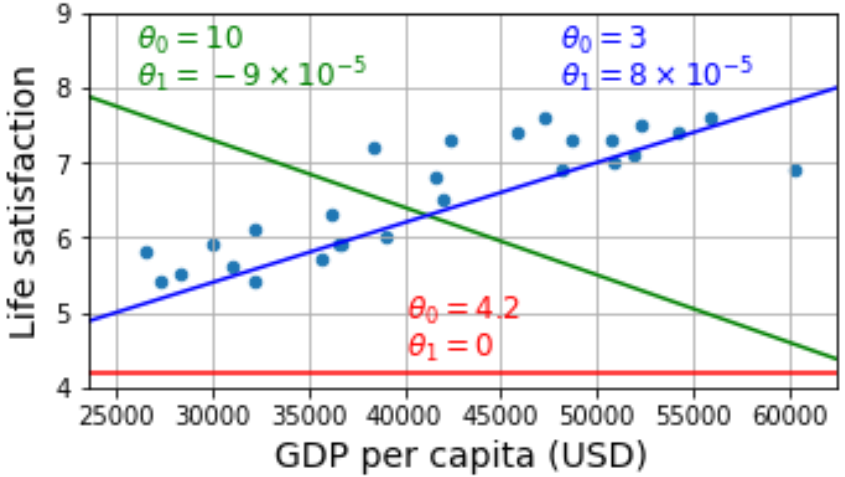
# Does money make people happier?

- Do you see a trend ?
  - Although the data are noisy, it looks like life satisfaction goes up more or less linearly as the country's GDP per capita increases/
- Equation of a simple linear model:
- $life\_satisfaction = \Theta_0 + \Theta \times GDP\_per\_capita$
- $\Theta_i$ are used by convention to represent the model parameters.



# Which is the best model ?

- We need to define the parameters Θ
- How can we know which values make the model perfom best?
  - A performance measure:
    - Utility function (or fitness function) that measures how good is the model,
    - Cost function that measures how bad is the model.
  - For linear models, a cost function that measures the distance between predictions and ground truth. The objective is to minimize it.



# The linear model that fits the training data best

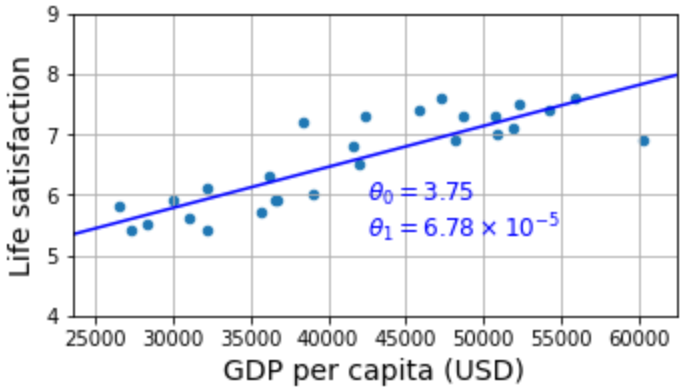
```
from sklearn import linear_model

X_sample = country_stats[[gdppc_col]].values
y_sample = country_stats[[lifesat_col]].values

lin1 = linear_model.LinearRegression()
lin1.fit(X_sample, y_sample)

t0, t1 = lin1.intercept_[0], lin1.coef_[0][0]
print(f"θ0={t0:.2f}, θ1={t1:.2e}")
```

```
θ0=3.75, θ1=6.78e-05
```



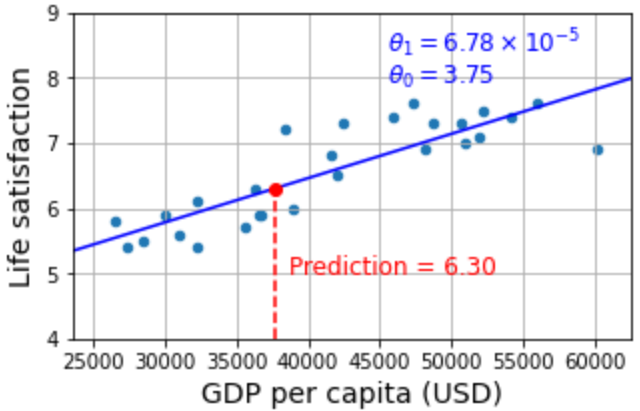
# Making new predictions (aka inference)

```
cyprus_gdp_per_capita = gdp_per_capita[gdppc_col].loc["Cyprus"]
cyprus_gdp_per_capita
```

```
37655.1803457421
```

```
cyprus_predicted_life_satisfaction = lin1.predict([[cyprus_gdp_per_capita]])[0, 0]
cyprus_predicted_life_satisfaction
```

```
6.301656332738056
```



What can go wrong …

Bad model or bad data …

# Insufficient quantity of training data

- Data matters more than algorithms
- Trade-off between spending time ad money on algorithm development vs spending it on corpus development.
- However, small and medium-sized datasets are still very common and it is not always easy to get extra training data, so algorithms are still important.

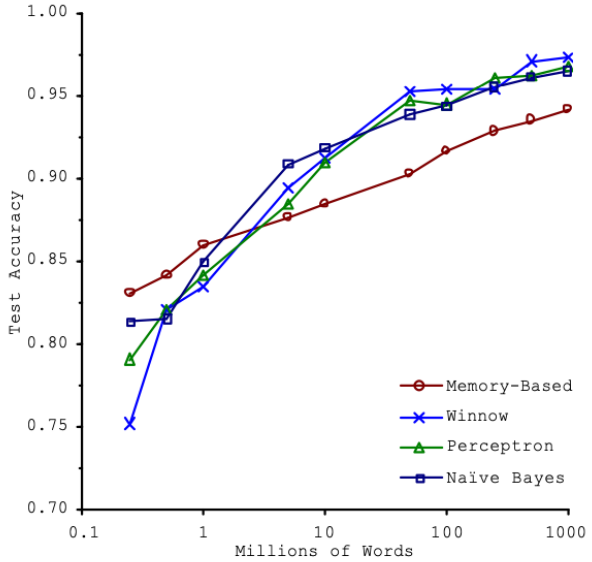
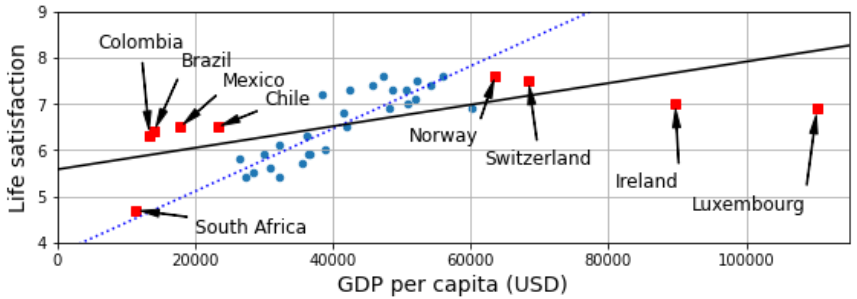



Figure 1. Learning Curves for Confusion Set Disambiguation

*Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (ACL '01). Association for Computational Linguistics, USA, 26–33*

# Non representative training data

- Generalize well <=> training data must be representative of new cases
- We add countries with lower or bigger GDP per capita:
  - The old model is bad on these data
- It makes clear that such a simple linear model is probably never going to work well.
- By using a non representative training set, we train a model that is unlikely to make accurate predictions




Crucial to use a training set that is representative of the cases we want to generalize.
Issues:

- Too small sample => sampling noise (non representative data as a result of chance)
- Very large samples can also be non representative if the sampling method is flawed (sampling bias)

# Example of sampling bias: the US presidential election in 1936

- Landon against Roosevelt:
  - The *Literary Digest (LT)* conducted a very large poll, sending mail to about 10 million people
  - It got 2.4 million answers, and predicted with high confidence that Landon would get 57% of the votes.
  - Instead Roosevelt won with 62% of the votes.
  - The flaw was the sampling method:
    - To obtain addresses to send the polls => LT used telephone directories, list of magazine subscribers, club membership lists. All of these lists tended to favor wealthier people, who were more likely to vote Republican .
    - Less than 25%  of people answered. This again introduces a sampling bias, by potentially rulling out people who did not care about politics, or people who did not like the LT, and other key groups. => aka **nonresponse biase**

# Poor-quality data

- Training data set is often full of errors, outliers, noise (due to poor quality measurements)
- => spend time to clean up the data.
  - Outliers => it may help to discard them or to fix the errors manually
  - Missing values : decide whether ignore the attribute or fill the missing values, …

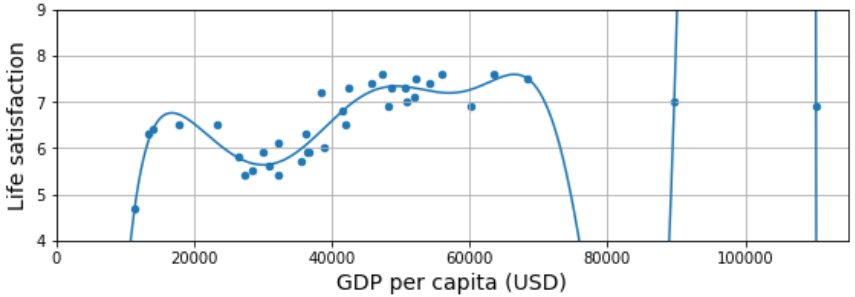
# Irrelevant features: garbage in garbage out!

- The system will only be capable of learning if the training data contains enough relevant features and not too many irrelevant ones.
- Feature engineering:
  - Feature selection (selecting the most useful one among all the features)
  - Feature extraction (combine existing features to build a more useful one)
  - Creating new features bu gathering new data.

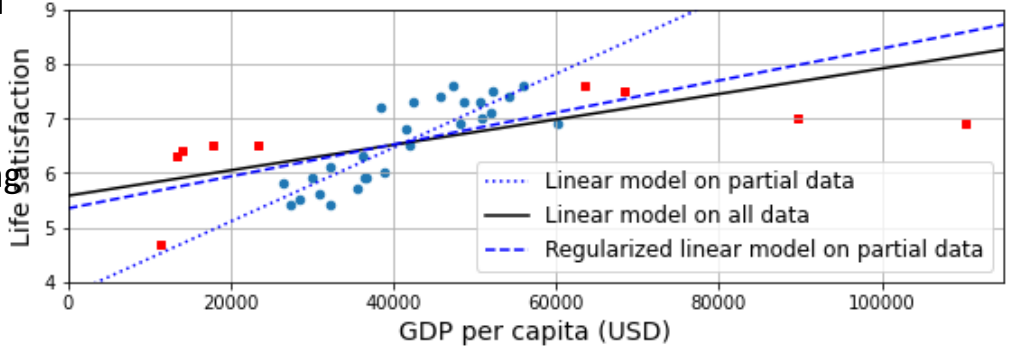
# Bad algorithms: overfitting the training data

- Overgeneralisation (**overfitting**): the model performs (too) well on the training data, but it does not generalize well on new instances.

- A high degree polynomial life satisfaction model that strongly overfits the training data
  - Would we really trust its predictions?




- Constraining a model to make it simpler and reduce the risk of overfitting is called **regularization**.
- Regularization amount: an hyperparameter
- Hyperparameter: a parameter of the learning algorithm (not of the model)
- Hyperparameter tuning is important.



# Underfitting the training data

- The opposite of overfitting:
  - It occurs when the model is too simple to learn the underlying structure of the data.
    - Linear model of life satisfaction is prone to underfit.
    - Reality is just more complex than the model, so its predictions are bound to be inaccurate, even on the training examples.
  - The main options to fix  this problem:
    - Select a more powerful model, with more parameters
    - Feed better features to the learning algorithm (feature engineering).
    - Reduce the constraints on the model (reducethe regularization hyperparameter).

# Stepping back

- ML is about making machines get better at some tasks by learning from data.
- Different types of ML systems: supervised or not, batch or online, instance-based or model-based.
- System will not perfom well if the training set is not representative, noisy or polluted with irrelevant features.
- Overfitting vs underfitting.

- An important topic not discussed yet!
  - The evaluation of the model (and its finetuning).

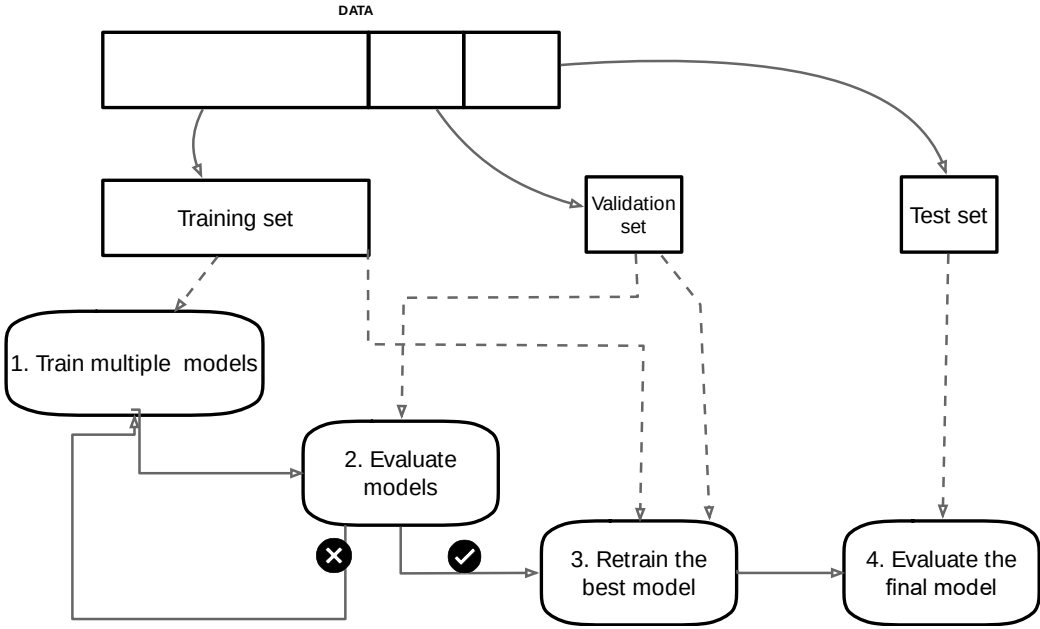
# Testing and validating

- The only way to know how well a model will generalize to new cases is to actually try it on new cases!

- Put it in production, wait and see? No a good idea … (dramatic image for the company/business)
- Better option: **Split** your data into **two sets**
  - The **training** set
  - The **test** set
- The error rate on on new cases (aka generalization error or out-of-sample error) can be estimated by evaluating the model on the test set.
- Training error low and generalization error high => overfitting on the training data.

# Hyperparameter tuning and model selection

- Evaluating a model is simple enough: just use a test set
- What about hyperparameter?
  - Possibly test 100 different values …
  - Another set is needed to keep the generalization error correct.
  - Common solution : holdout validation (train/validation/test sets)
  - Validation set aka dev set or development set
- Pb of too small validation set: cross validation.



# No Free Lunch Theorem

- A model is a simplified representation of the data
  - Simplifications discard the superfluous details to better generalize to new instances
- Selecting a particular type of model ± implicitly making assumptions about the data.
  - Linear model …


No Free Lunch Theorem: If you make absolutely no assumption about the data, there is no reason to prefer one model over any other.

➔ There is no model that is a priori guaranteed to work better.

*The lack of a priori distinctions between learning algorithms. DH Wolpert - Neural computation, 1996*

# What you should know after this lesson

- ✓ What Machine Learning is (with a smart definition).
- ✓ Give at least 4 types of ML applications.
- ✓ What a labeled training set is.
- ✓ The two most common supervised tasks.
- ✓ Four common unsupervised tasks.
- ✓ What type of algorithm to segment customers into several groups.
- ✓ What an online learning system is.
- ✓ Algorithm that relies on a similarity measure to make predictions.
- ✓ Differences between model parameters and model hyperparameters.
- ✓ Purpose of Train / Validation / Test sets.
- ✓ Overfitting vs underfitting.

- Tutorials
- (simple) exercises
- Make Cheat Sheets from Tutorials:
  - Numpy
  - Mathplot
  - Panda
  - Web scrapping : https://realpython.com/python-web-scraping-practical-introduction/

# TODO (Lab Session)


EPITA
ÉCOLE D'INGENIEURS EN INFORMATIQUE