Homework 1
-
Hands-on machine learning chapter I

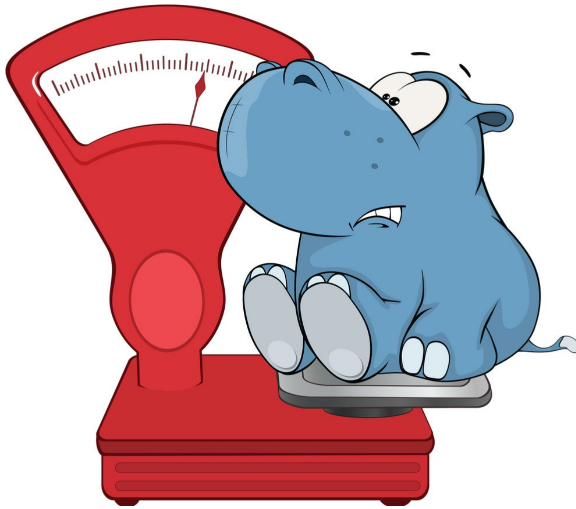
Damiano Rousselet - 12225451

universität innsbruck

1. Definition and Motivation of Machine Learning

- Read through the definition of machine learning. You want to estimate the weight of a person based on his or her height. Now try to describe the terms task, performance measure, training set, training instance, and model using this concrete example.

→ engineering-oriented definition:
A computer program is said to learn from *experience E* with respect to *some task T* and some *performance measure P*, if its performance on *T*, as measured by *P*, improves with experience *E*.
—Tom Mitchell, 1997



→Task: Estimate weight as $f(Height)$

→Possible Performance measure (Ex.): $\mathrm{P} = \frac{1}{n}\sum(Weight\_i - \widehat{Weight\_i})^2 = "MSE"$

→Training set: Set of pairs of $height\_1 : weight\_1, height\_2 : weight\_2, \ldots, height\_n : weight\_n$

→Training instance: Each single pair: $height\_i : weight\_i$

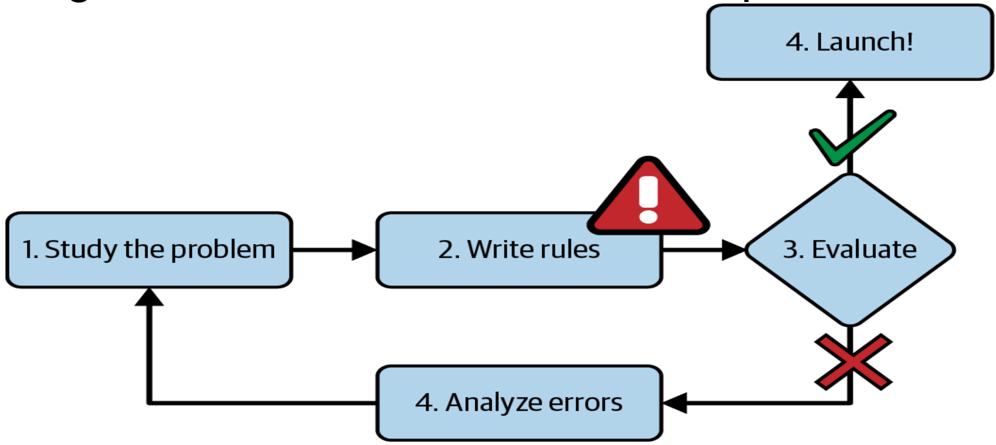
→ Model: Part of ML system that makes predictions
(E.g. Linear regression, *NN, Random forests*,…..)

- What is the difference between the traditional programming and the machine learning approach?

→In traditional programming we would examine the situation and write an algorithm for each case, for example:

```
If x:
   do 1
If y:
   do 2
[…]
```

Then test if the algorithm is good enough to work with all/enough cases, else modify code / approach



Traditional programming

→ In ML

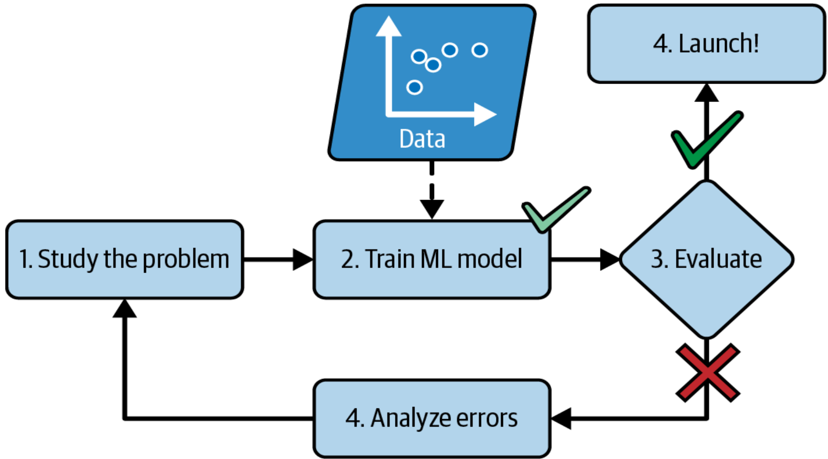
1. Analyse problem

2. Train model to automatically learn what elements / patterns predict the outcome best

3. Evaluate model outcome / predictions

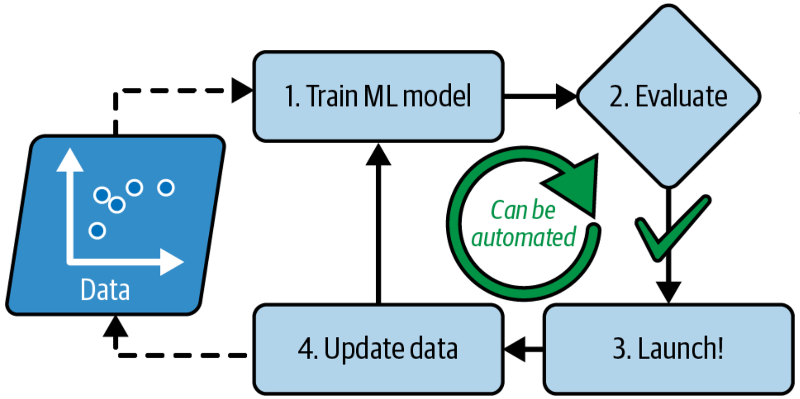
4. If good: launch

4.1. If bad: analyse errors and restart from 1.



• What motivates the machine learning approach?

→One reason to use ML is that it has a great ability to adapt to a situation that is changing constantly ("Fluctuating environment").



Screenshot: ML adapting to change

→Also, in situations that require long lists of instructions for many cases, here ML can perform better and present more elegant, readable code.

• When will I use one approach and when will I use the other, think about an example.

→ A traditional programming approach can be better in situations that do not change, like an automated electric door opening system. If the sensor detects motion, it activates the engine which opens the door. The system is cheaper to build and more robust. Obviously, this is useful if no particular security is required etc.…

→A ML approach is suited for a case where the input is not constant like face recognition. It would be impossibly overwhelming to write code to recognize each face, in a multitude of different light / angle / colour conditions. Here ML would shine and be much more efficient.

- What other strengths does a machine learning approach have? Try using the terms “fluctuating environments” and “data mining”.

→ Another advantage is that sometimes by training a ML model for one task, we gain insight into huge amounts of data and find new patterns, which leads to a better understanding of the problem. Digging into data to find patterns is called “data mining”.

→ ML can be used for complex problems where traditional approaches have failed. For example ML models trained to play board games like chess or “Go!”, have performed new moves / strategies which were counterintuitive and unknown to humans.

To summarize, machine learning is great for:

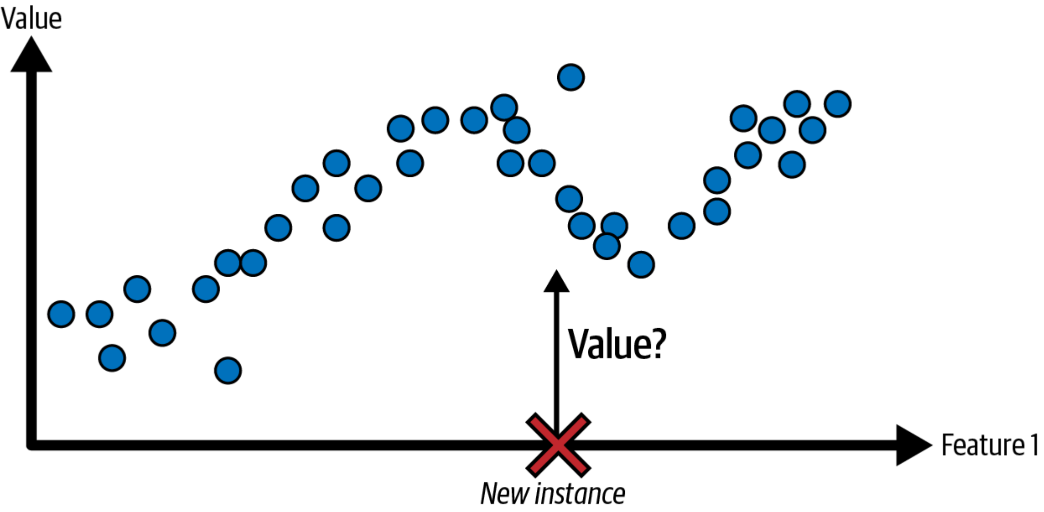
- Problems for which existing solutions require a lot of fine-tuning or long lists of rules (a machine learning model can often simplify code and perform better than the traditional approach)
- Complex problems for which using a traditional approach yields no good solution (the best machine learning techniques can perhaps find a solution)
- Fluctuating environments (a machine learning system can easily be retrained on new data, always keeping it up to date)
- Getting insights about complex problems and large amounts of data

Screenshot from book: advantages of ML

2. **Types of Machine Learning Systems**

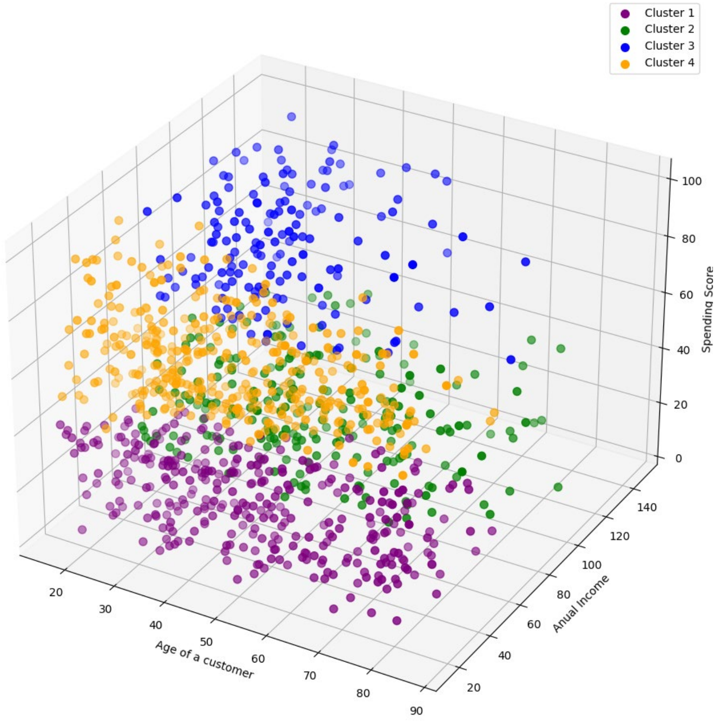
- Try to summarize all five training supervisions (supervised learning, unsupervised learning, self-supervised learning, semi-supervised learning, and reinforcement learning).

→ Supervised: The training set includes “labels” for each instance. Used for classification and prediction as “regression” models (“label” = “target”).
New elements are presented without labels and the model tries to categorise or predict a value.



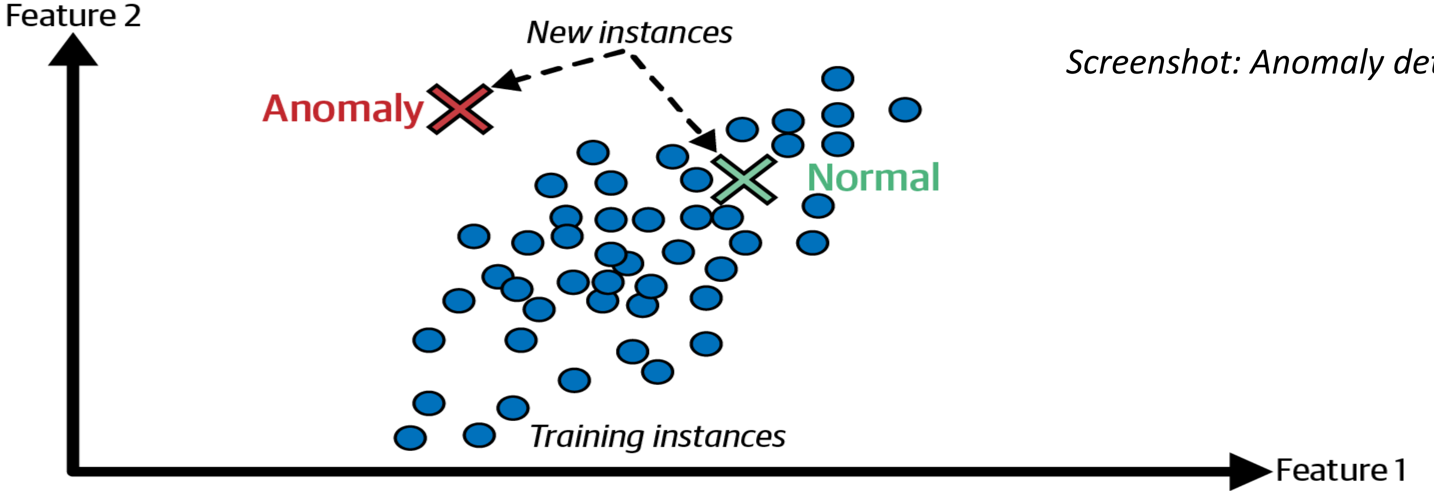
Screenshot: A regression problem: predict a value, given an input feature

→<u>Unsupervised</u>: Training data is not labelled, for example clustering algorithms like K-means.
Here the training set is not labelled, the algorithm places ("clusters") data together according to measures of similarity or difference.
For example by minimizing intra-group differences and maximizing inter-group differences.
Another example is visualization algorithms which try to visualize a lot of complex unstructured data in 2D or 3D, such that it is easily readable by humans.



Screenshot: Representation of clusters of supermarket customers from unstructured data (source)

→ Another important unstructured task is <u>detecting anomalies</u>, e.g. for financial transactions.

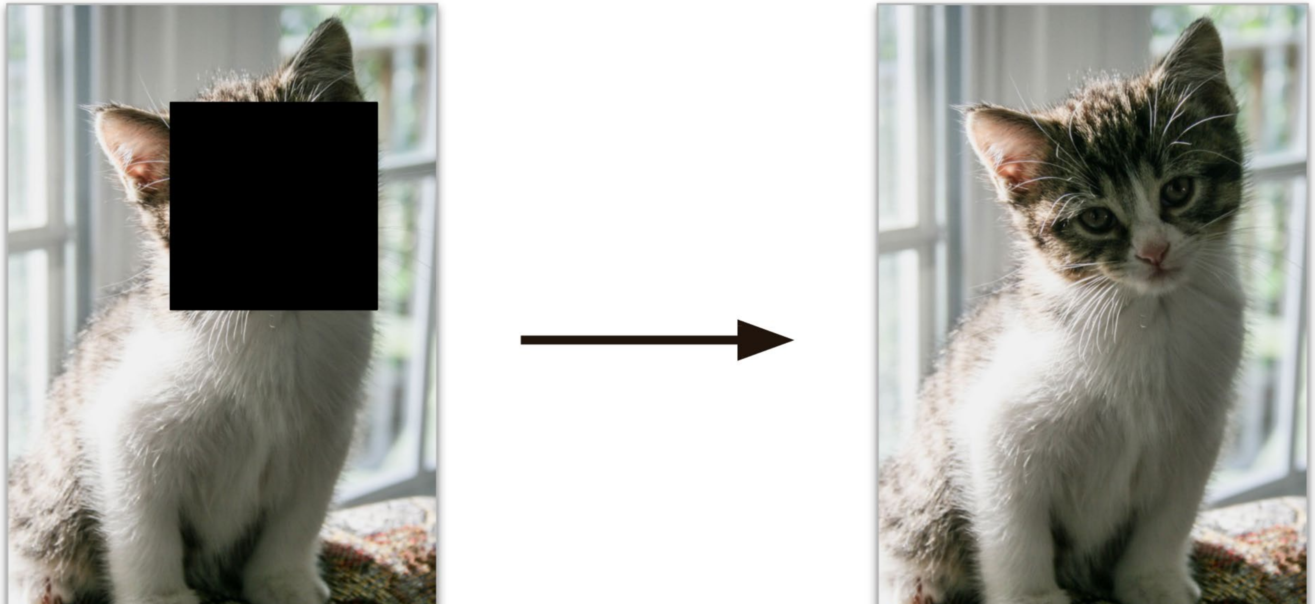


Screenshot: Anomaly detection

→<u>Semi-supervised learning</u>: As the name suggests, this type of algorithms can deal with data that is partially labeled. This is often the case because labeling is time/cost intensive. They often combine an unsupervised and a supervised part. For example, a clustering algorithm can cluster unlabeled data and then label the unlabeled training instances according to the most common label in the cluster it has been put into.

→<u>Self-supervised learning:</u> This method generates a labeled training set from an unlabeled one. Then a supervised algorithm can be used.
For example, a model can be trained to generate a missing or damaged part of an image. Here a part of a complete image is hidden and the model trained to regenerate it. The image with the masked part is the input and the complete image is the label:

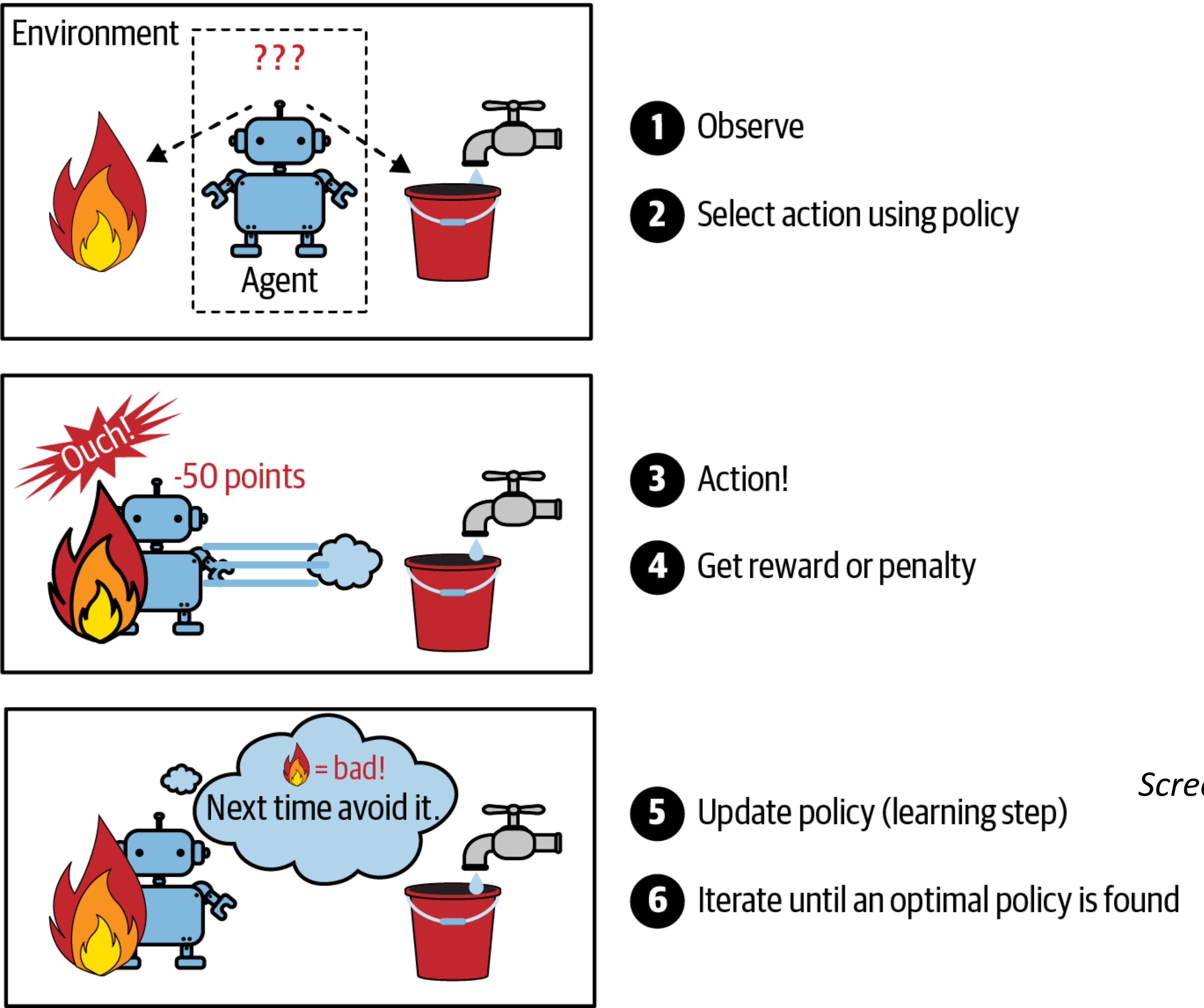
„*Transfer learning*" → Often, models using self-supervised learning are an intermediate step to a final goal, like training a model to repair images, can later be used to recognize images; as to repair images, it has to know which parts should be placed in an image, and which not.



Screenshot:
Self-supervised learning example: input (left), target (right)

→Reinforcment learning: Is a different approach, the system („*agent*“) is able to observe the environment and make choices. Then a reward / punishment is given to it according to how well it performs. The systems adapts ist choices and develops the best „*policy*“ to maximize reward and minimize punishments.

→A famous example is Google‘s Alpha Go model, which analyzed many games and played against itself, until it was very good at minimizing the cost function – avoiding punishments, chasing rewards.



Screenshot from book: agent in reinforcment learning

- What is the difference between batch and online learning? What is the disadvantage of batch learning?

→<u>Batch learning</u>:
In this mode, a model <u>is incapable of incrementally learning new things</u>. It is trained on the training set and then performs updating.

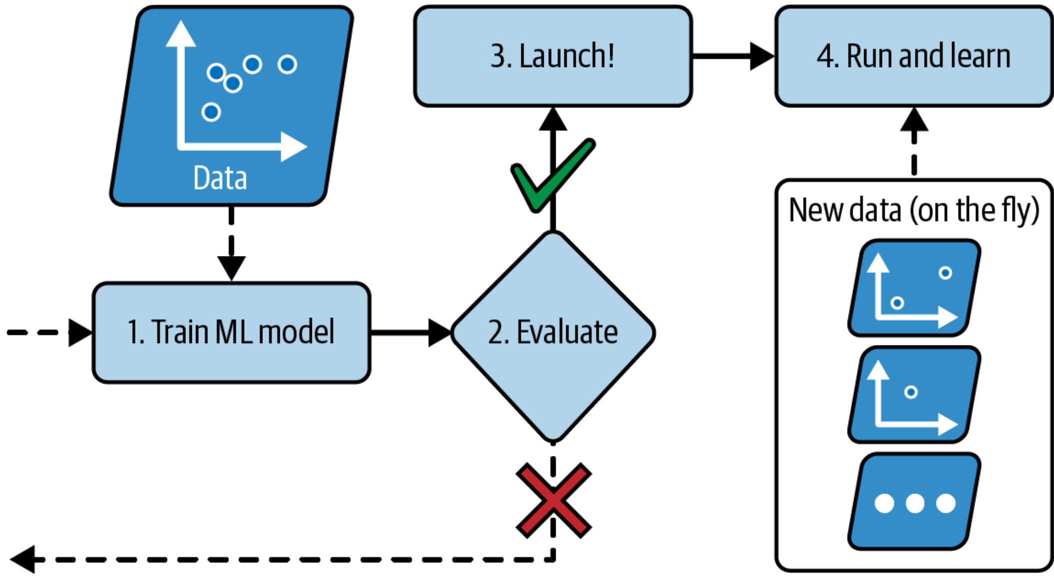
The <u>main disadvantage</u> is that in the real world, its performance worsens *(“<u>model rot”, ”data drift</u>”)* – as the world advances – without it.
It is not a big issue in cases where the environment changes slowly, like species recognition. (Although recording technologies like cameras likely change faster than evolution)

→ <u>Online learning:</u>
In this mode, a system is capable of adding new knowledge *incrementally* to its pre-existing one.

Online learning is useful for systems that need to adapt to change extremely rapidly (e.g., to detect new patterns in the stock market). It is also a good option if you have limited computing resources; for example, if the model is trained on a mobile device.

Additionally, online learning algorithms can be used to train models on huge datasets that cannot fit in one machine's main memory (this is called *out-of-core* learning).

Screenshot from book



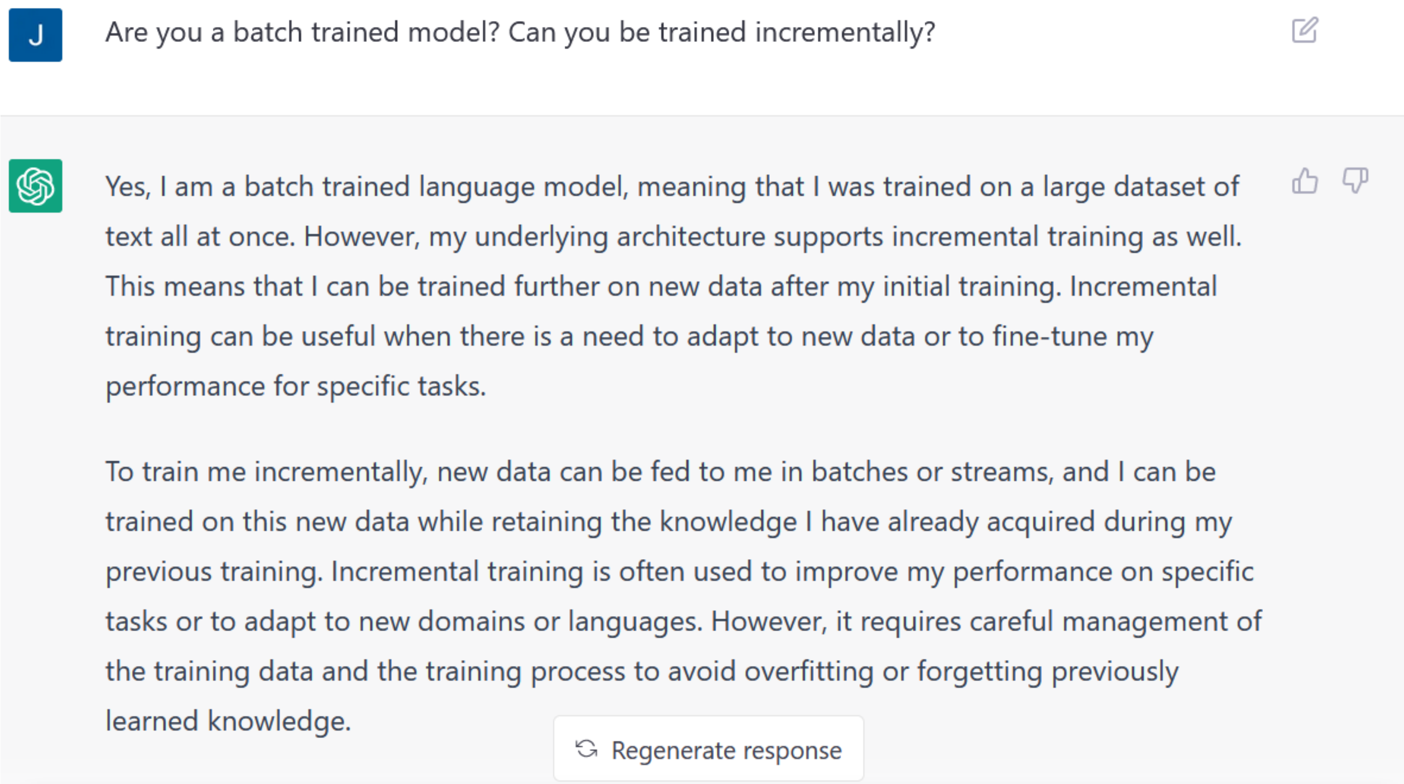
Screenshot: in online learning a model keeps learning as data comes in

- Do you already know of models that definitely do batch learning?

→ When ChatGPT came out it was trained on data until 2021 beforehand. However, as it says itself, it is also capable of incremental learning, by tuning the parameters of its Deep Neural Network.
This is a special case of a batch-trained model that has the ability to incrementally train.

→ Another example of a batch-trained model is BERT: huggingface.co/docs/transformers/model_doc/bert

Screenshot: ChatGPT interface explaining itself (chat.openai.com)



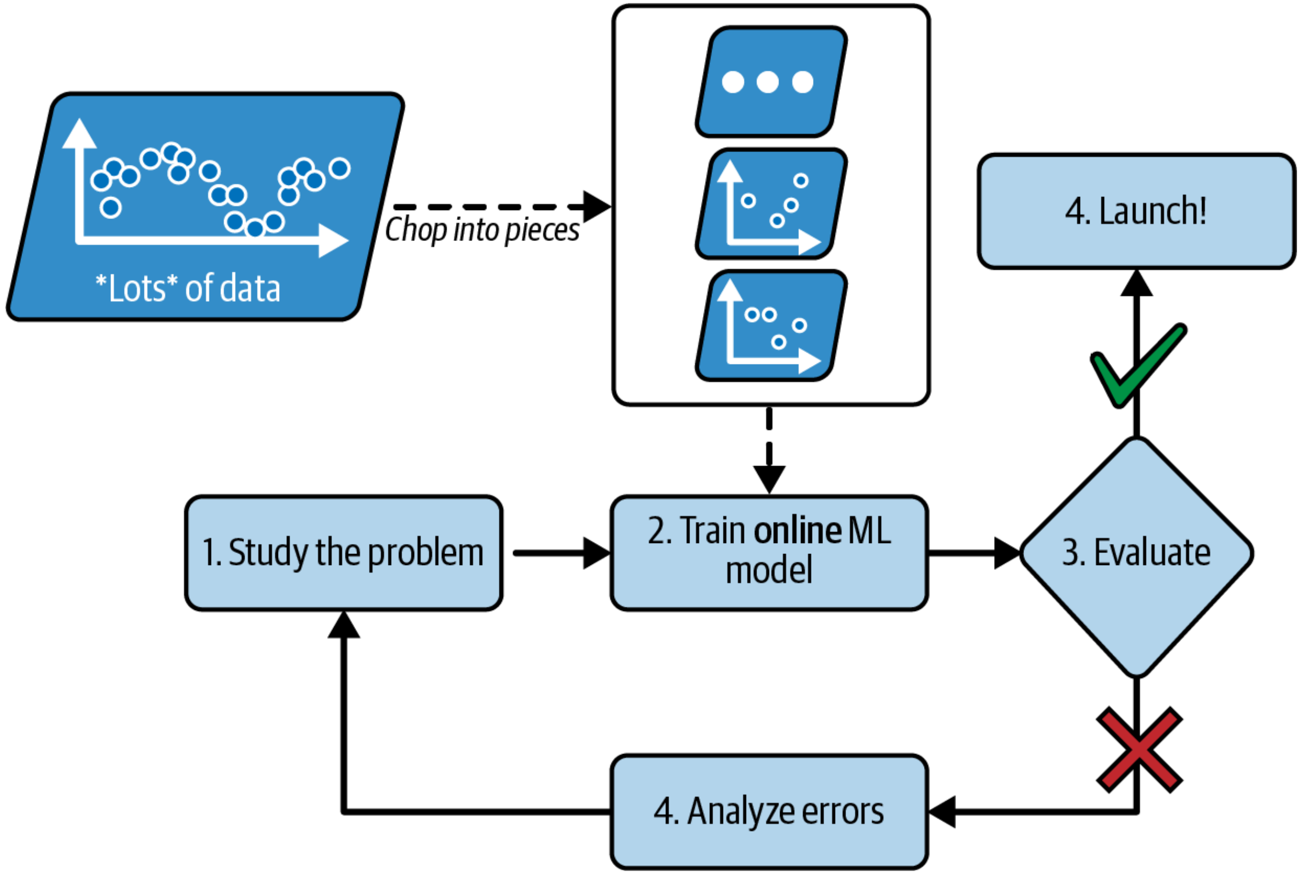
- What does the term “out-of-core learning” mean?

→It is a technique, used if the resources are limited compared to the dataset, that might be huge. If the training set does not fit into memory, it is broken into smaller chunks and these are processed sequentially until the whole dataset has been worked through training.

- What are challenges in online learning?

→ A big challenge is that is if bad data is given to a system, it learns the wrong things and its performance can drop. To fight against it, the system should be monitored and learning disabled if performance is dropping. Also, the input data can be monitored using E.g. an *anomaly detection* algorithm

→ Another important challenge is selecting the right “*learning rate”* – how fast the system should adapt to new data while forgetting old data (high *learning rate*); or how slowly the system should learn (low *learning rate*) and be less sensitive to noise/outliers.



Screenshot: out-of-core learning process

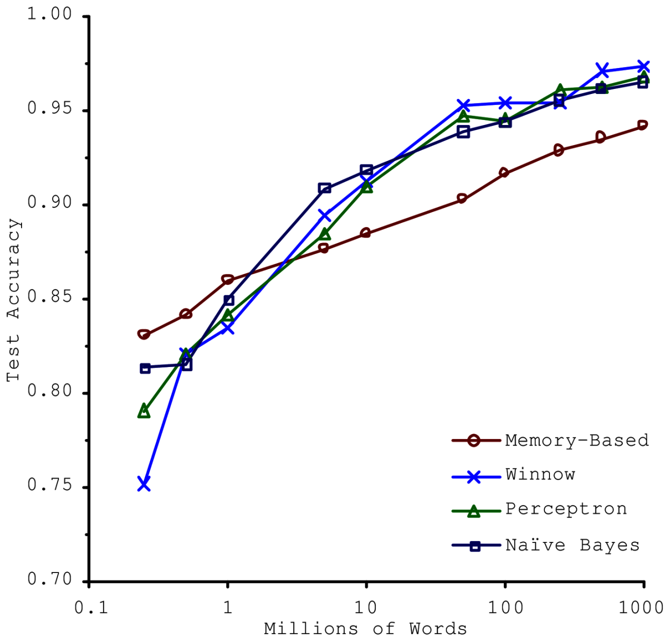
3. **The main challenges in Machine Learning**

- Briefly name all the challenges we encounter in Machine Learning

→They can be summarized in 2 categories: Bad model & bad data

→Bad Data:
  - →Insufficient quantity, small sample
  - →Nonrepresentative data
    - → Sampling noise
    - → Sampling bias

Machine learning is not quite there yet; it takes a lot of data for most machine learning algorithms to work properly. Even for very simple problems you typically need thousands of examples, and for complex problems such as image or speech recognition you may need millions of examples (unless you can reuse parts of an existing model).



Screenshot: performance increasing with enough data
"The Unreasonable Effectiveness of Data", IEEE Intelligent Systems 24, no. 2 (2009)

- What do we mean by sampling noise and sampling bias?

It is crucial to use a training set that is representative of the cases you want to generalize to. This is often harder than it sounds: if the sample is too small, you will have *sampling noise* (i.e., nonrepresentative data as a result of chance), but even very large samples can be nonrepresentative if the sampling method is flawed. This is called *sampling bias*.

Screenshot: Nonrepresentative data, sampling - noise, sampling - bias

- Further challanges with data:

→Poor quality data

- If some instances are clearly outliers, it may help to simply discard them or try to fix the errors manually.
- If some instances are missing a few features (e.g., 5% of your customers did not specify their age), you must decide whether you want to ignore this attribute altogether, ignore these instances, fill in the missing values (e.g., with the median age), or train one model with the feature and one model without it.

- What are features?

→Features are attributes of the data, used to train the model, for example, "*age*". They need to be relevant to make meaningful predictions

→Irrelevant features: Important to select relevant features → "*feature engineering*":

- *Feature selection* (selecting the most useful features to train on among existing features)
- *Feature extraction* (combining existing features to produce a more useful one—as we saw earlier, dimensionality reduction algorithms can help)
- Creating new features by gathering new data

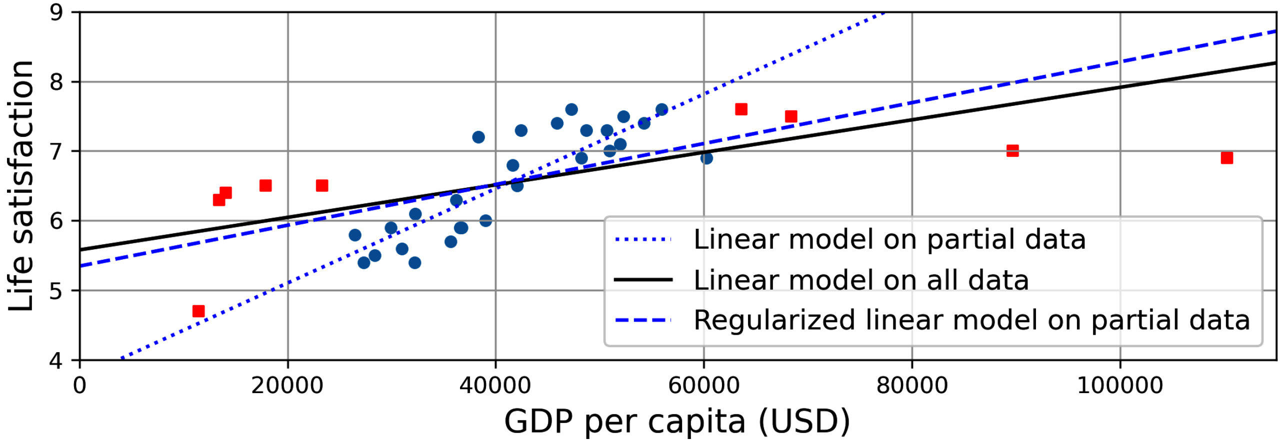
Use a simple ML model and describe what is meant by feature selection and feature extraction?
→If we build a model to predict the weight of a person, *feature selection* is the process of selecting relevant attributes for the prediction like "sex", "age", "ethnicity" and discarding irrelevant ones such as "name".

→*Feature extraction:* would be synthetizing attributes into a more meaningful one, in the case of humans it could be something like body-mass-index or athleticism level.

→Examples of bad model (algorithm):

- What is and what favours overfitting in Machine Learning and how can we prevent it?

→Overfitting: Model too complex, fitting the training set too tightly, wrongfully detecting patterns in noise



Screenshot: constrained model (dashed) generalizing better than non-constrained one (dotted)

Overfitting happens when the model is too complex relative to the amount and noisiness of the training data. Here are possible solutions:

- Simplify the model by selecting one with fewer parameters (e.g., a linear model rather than a high-degree polynomial model), by reducing the number of attributes in the training data, or by constraining the model.
- Gather more training data.
- Reduce the noise in the training data (e.g., fix data errors and remove outliers).

Screenshot from book: Overfitting & solutions

→ Important to tune hyperparameters to right value of *regularization*

- What is and what favors underfitting in Machine Learning and how can we prevent it?

→ Underfitting: Model is too simple to describe reality or structure in data

Here are the main options for fixing 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 (for example by reducing the regularization hyperparameter).

3. **Testing and Validating**

- What is the motivation of holdout validation? Using Figure 1-25, try to explain what is the test set, the training set, and the validation set?

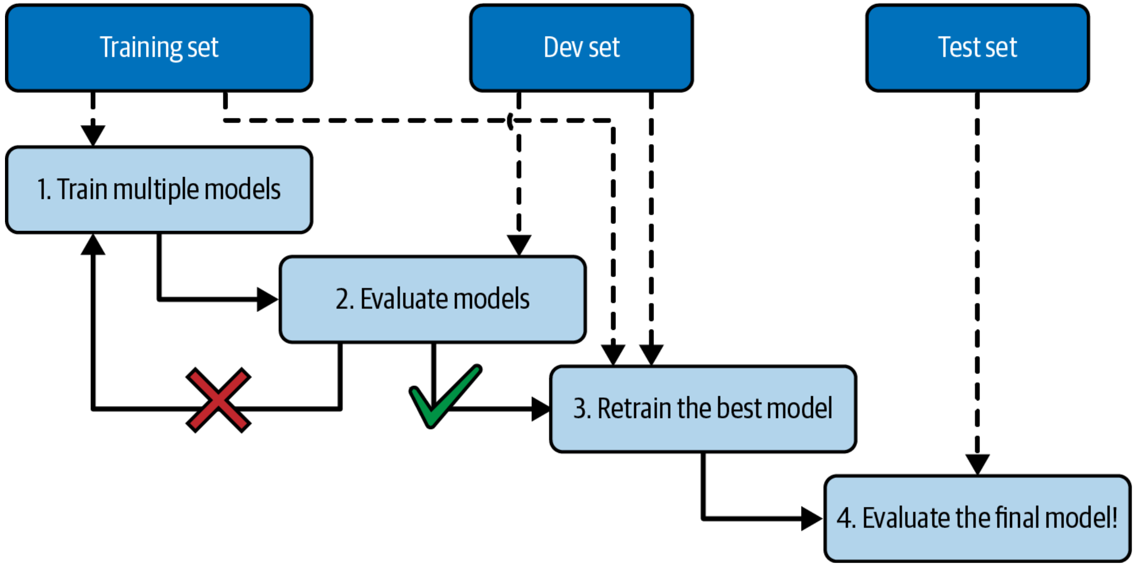
→It is an approach to avoid having the model and its hyperparameters *overfitting* only to the training set and not generalizing well to real-life predictions.

1. Split the data into a *training set* used to train multiple models with different hyperparameters.

2. Use part of the remaining data as a *validation (dev) set*, onto which we test the different models.

3. Select the model with the best performance on the *validation set,* train it on *training* + *validation sets* joined together.

4. This gives us the best model (hyperparameters) which we test on the remaining *test set* to get an estimate of the *generalization error*



Screenshot: holdout validation

Sources

- Hands-on machine Learning with Scikit Learn Keras and TensorFlow Third Edition, 1st chapter.
- https://uibkacat-my.sharepoint.com/:p:/g/personal/lars_hofmann_student_uibk_ac_at/EYcCueCPQc9Gv0UrPnGFaOoBI3AZTbmXzOKaPlMgG14nng?rtime=9CkJiBIi20g
- chat.openai.com (Slide 10)
- huggingface.co/docs/transformers/model_doc/bert
- www.alphagomovie.com