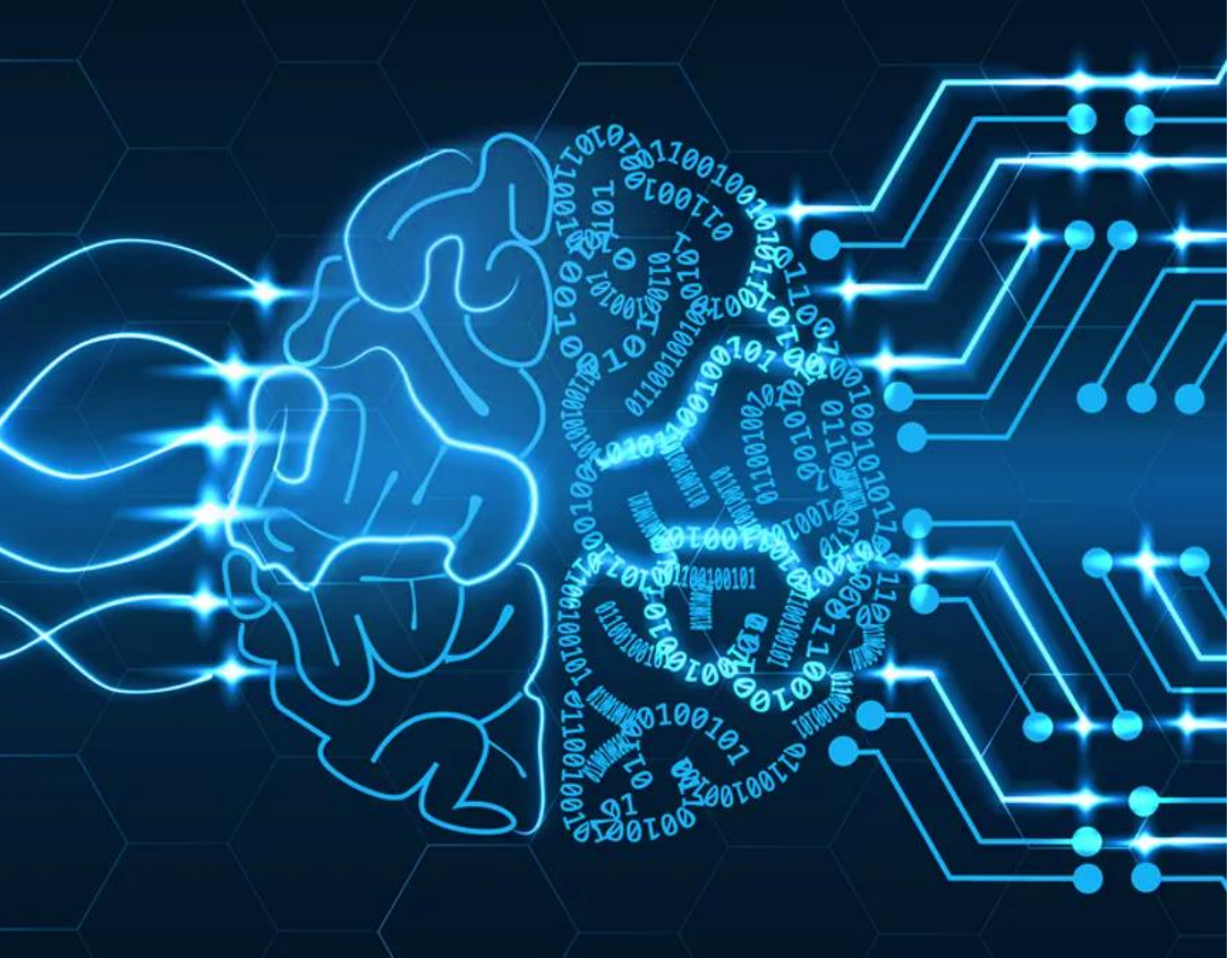


یادگیری ماشین

دکتر مهدی شریف‌زاده

بهمن ۱۴۰۱


بسم الله الرحمن الرحیم

Table of Contents

Goals

- ✓ Acquiring a Comprehensive Insight into Various Machine Learning Techniques and Approaches
- ✓ Acquiring the Competency to Make Informed Decisions Regarding Method Selection For Various Problems, Considering Both Their Advantages and Limitations.
- ✓ Evaluating Various Techniques of Machine Learning to Determine the Most Appropriate Method for a Given Situation
- ✓ Acquiring Familiarity with Open-Source and Proprietary Tools of Machine Learning
- ✓ Acquiring the Proficiency to Execute Machine Learning Tasks Utilizing Cutting-Edge Machine Learning Technologies.

Course Outline

- ✓ Overview and Introduction
- ✓ End-to-End Machine Learning Projects
- ✓ Regression, Logistic Regression
- ✓ Support Vector Machines
- ✓ Tree-Based Models
- ✓ Dimensionality Reduction and Unsupervised Methods
- ✓ Neural Networks: Feedforward, Convolutional, Recurrent Neural Networks
- ✓ Advanced Concepts: Autoencoders, Generative Adversarial Networks
- ✓ Applications(NLP, Recommender Systems) and Case-Studies

End-to-End Machine Learning Projects

- ✓ In this chapter, we cover the entire process of building a machine learning model with a focus on end-to-end projects. We cover subtopics including:
  - ✓ Framing the Problem
  - ✓ Selection of Performance Metric
  - ✓ Splitting Data
  - ✓ Exploratory Data Analysis
  - ✓ Train & Fine-Tune Model
  - ✓ Test & Validation.
- ✓ The goal is to develop a deep understanding of each step and its importance in creating an effective machine learning solution.

Linear Regression

- ✓ Linear Regression: Model relationships between variables with a linear equation. Performance Metric: Evaluate regression models with metrics such as MSE, RMSE, and R-Squared.
- ✓ Normal Equation & Gradient Descent: Find coefficients with closed-form solution or optimization algorithm.
- ✓ Polynomial Regression & Regularization: Model non-linear relationships and prevent overfitting with polynomials and regularization techniques.
- ✓ Logistic Regression: Use for binary classification and evaluate with metrics such as accuracy, confusion matrix, and cost function.

Support Vector Machines

- ✓ Linear SVM (Soft Margin): Use linear SVM for classification with the concept of a soft margin to handle cases where a clear separation between classes does not exist.
- ✓ Kernels (Polynomial, RBF): Enhance the linear SVM model with non-linear transformations through the use of polynomial and radial basis function (RBF) kernels to handle complex and non-linear decision boundaries.
- ✓ Support Vector Regression (SVR): Apply support vector machine to regression problems with SVR to model complex non-linear relationships between variables.

Tree-Based Models

- ✓ Decision Trees: Use decision trees to model and make decisions based on a tree structure that splits data into smaller subsets. Decision trees are popular for their interpretability and ability to handle both categorical and numerical data. They can be used for classification and regression problems.
- ✓ Voting Classifiers & Bagging: Improve the accuracy of decision trees with ensemble methods like voting classifiers and bagging. Voting classifiers combine the predictions of multiple classifiers, while bagging involves creating multiple bootstrapped versions of the same model. Both methods reduce the variance in the model and increase its accuracy.
- ✓ Random Forest: Enhance the decision tree model by building multiple trees and combining their predictions through random subspace sampling. In a random forest, each tree is built on a random subset of the features, which helps reduce the correlation between trees and improve the overall accuracy.

Dimensionality Reduction

- ✓ PCA: Reduce the complexity of high-dimensional data by finding the principal components of the data with PCA. PCA can be used for data visualization, noise reduction, and feature extraction. By finding the principal components, the data can be transformed into a lower-dimensional space while retaining most of the information.
- ✓ Kernel PCA: Enhance the PCA method by using non-linear transformations to handle complex, non-linear relationships in high-dimensional data with Kernel PCA. Non-linear relationships can be captured by transforming the data into a higher-dimensional space, where the relationships can become linear. By applying PCA to this transformed space, the information can be reduced back to a lower-dimensional space, while preserving the non-linear relationships. This can be useful for tasks such as classification, visualization, and data compression.

Unsupervised Techniques

- ✓ Clustering (K-Means): Group similar data points together with the unsupervised learning technique of K-Means clustering to identify underlying patterns or groupings in unlabeled data.

Artificial Neural Networks

- ✓ Perceptron: Start building artificial neural networks with the basic building block of a single neuron: the Perceptron. This linear classifier models the decision boundary between two classes.
- ✓ Multilayer Perceptron & Backpropagation: Enhance the Perceptron model by using multiple layers and the backpropagation algorithm to train the neural network.
- ✓ Activation Functions: Apply non-linear transformations to the inputs in the neural network with activation functions such as sigmoid, tanh, and ReLU.
- ✓ Regression MLP: Use multilayer perceptrons to perform regression tasks by modeling the relationship between inputs and outputs in continuous data.
- ✓ Classification MLP: Use multilayer perceptrons to perform classification tasks by modeling the decision boundary between different classes.

Convolutional Neural Networks

- ✓ Convolutional Layers: We use convolutional layers to extract important features from image data, which is essential for image classification tasks. Our goal is to identify the unique characteristics of an image, such as edges, corners, and textures, to classify the image into its respective class.
- ✓ Pooling: We use pooling layers to reduce the spatial dimensions of the feature maps, which helps prevent overfitting and reduces the computational cost. Our goal is to retain only the most important information from the feature maps and compress it into a smaller, more manageable size.
- ✓ Famous Architectures: By studying famous architectures, we learn how to design and improve our own CNN models for image classification. Our goal is to understand how to apply these architectures to real-world problems and develop models that achieve high accuracy in image classification tasks.

Recurrent Neural Networks

- ✓ Simple RNN: We use Simple RNNs to model sequential data, such as time series, speech signals, and text. Our goal is to understand how to process sequential data and make predictions based on the past values.
- ✓ LSTM and GRU: We use LSTM and GRU models to handle long-term dependencies in sequential data. These models are better suited for tasks that require capturing long-term dependencies compared to Simple RNNs. Our goal is to improve the performance of RNN models on tasks that require capturing long-term dependencies.
- ✓ Encoder-Decoder: We use the Encoder-Decoder model for machine translation and other sequence-to-sequence problems. Our goal is to generate output sequences that are conditioned on the input sequences.

Advanced Concepts

- ✓ Autoencoders (AE): We use autoencoders for dimensionality reduction and unsupervised learning tasks. Autoencoders learn a compact representation of the input data by encoding it into a lower-dimensional space and then decoding it back. Our goal is to learn a compact representation of the input data and use it for other tasks such as classification and clustering.
- ✓ Generative Adversarial Networks (GAN): We use GANs to generate new data samples from a random noise vector. GANs consist of two neural networks: a generator and a discriminator. The generator tries to generate new samples that are indistinguishable from the real data, while the discriminator tries to distinguish between real and fake samples. Our goal is to generate new data samples that are similar to the real data.
- ✓ Hidden Markov Models (HMM): We use Hidden Markov Models to model sequential data with underlying hidden states. HMMs can be used for tasks such as speech recognition, part-of-speech tagging, and sentiment analysis. Our goal is to model sequential data with hidden states and use it for various tasks.

Use-Cases and Applications

- ✓ We explore various use-cases and applications of these advanced concepts to understand how they can be applied to real-world problems. Our goal is to understand the practical applications of these concepts and to gain hands-on experience in using them for solving real-world problems.

What is Machine Learning?

Machine Learning is the science (and art) of programming computers so they can ***learn from data***.

Here is a slightly more general definition:

> *[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.*
> —Arthur Samuel, 1959

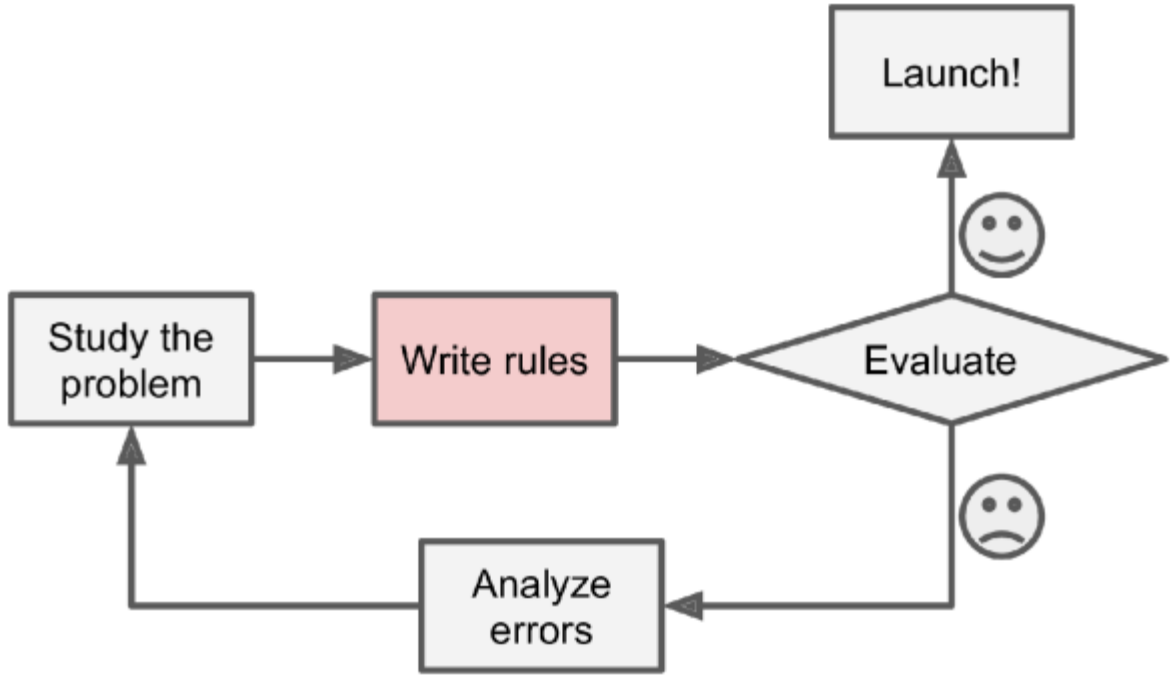
And a more engineering-oriented one:

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

Why Use Machine Learning?

How to write a **spam filter** using traditional programming:

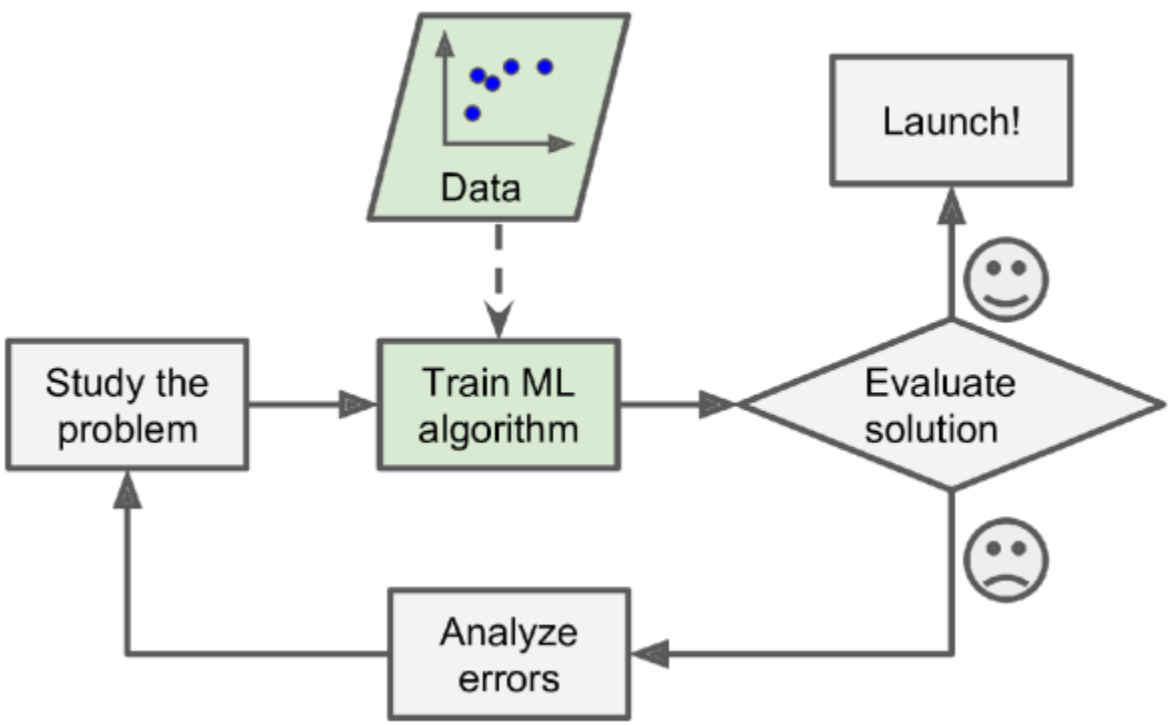
- Identify common features of spam (e.g. certain words, phrases in subject line, sender name, email body)
- Write a detection algorithm for each feature
- Test and repeat until good enough to launch

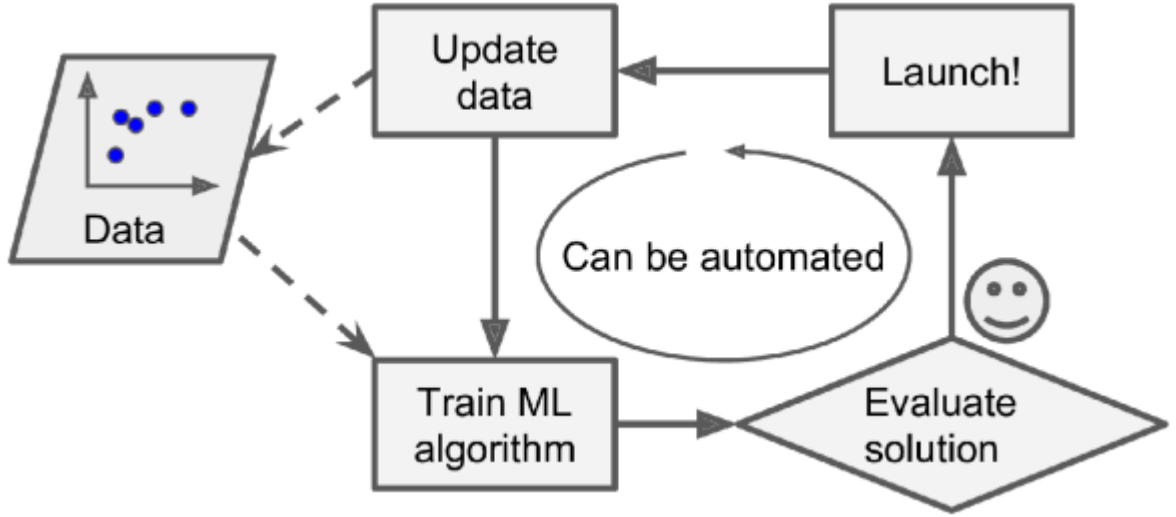


Why Use Machine Learning?

Spam filter comparison: traditional vs ML:

- Traditional: Long list of **complex rules**, hard to maintain
- ML: Shorter, easier to maintain, better accuracy
- Traditional: Spammers work around rules, need constant updates
- ML: Automatically adapts to new spam tactics



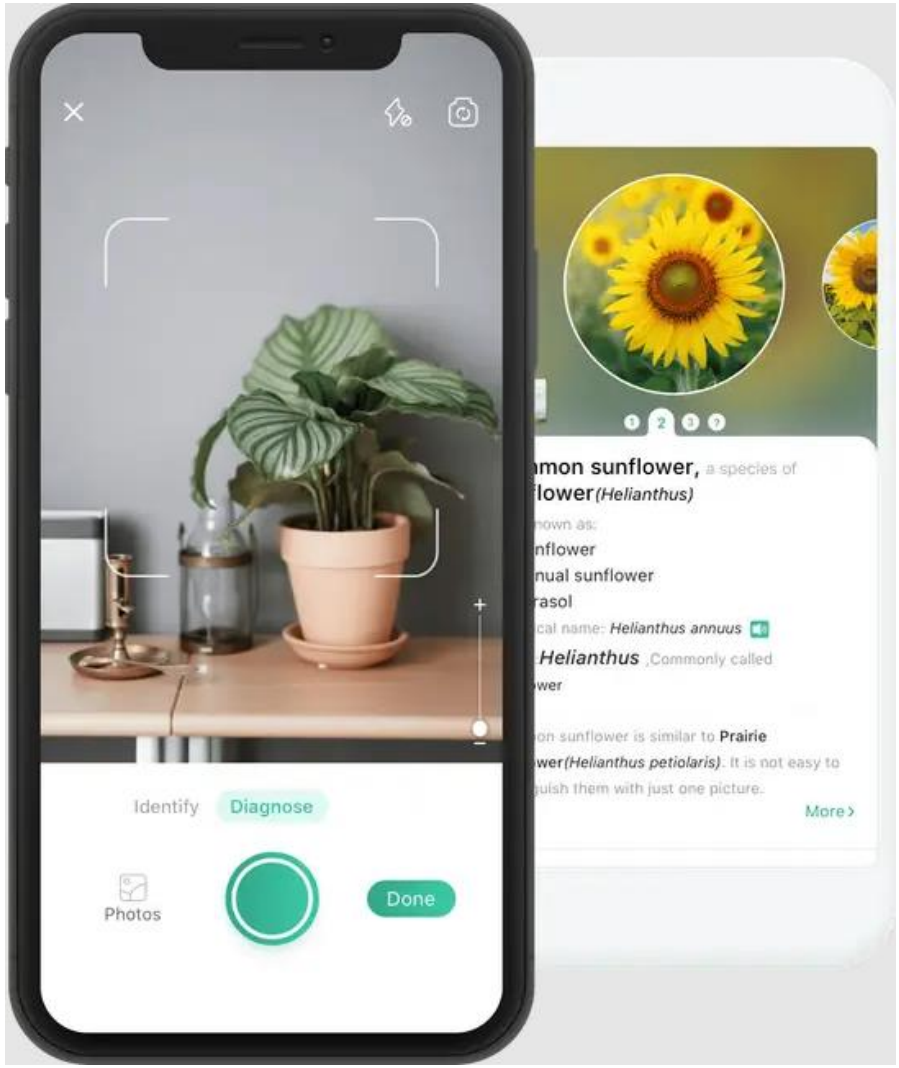


What is Machine Learning?

To summarize, Machine Learning is great for:

- Problems for which existing solutions require **a lot of fine-tuning or long lists of rules**: one Machine Learning algorithm 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 adapt to new data.
- **Getting insights** about complex problems and large amounts of data

Examples of Machine Learning?

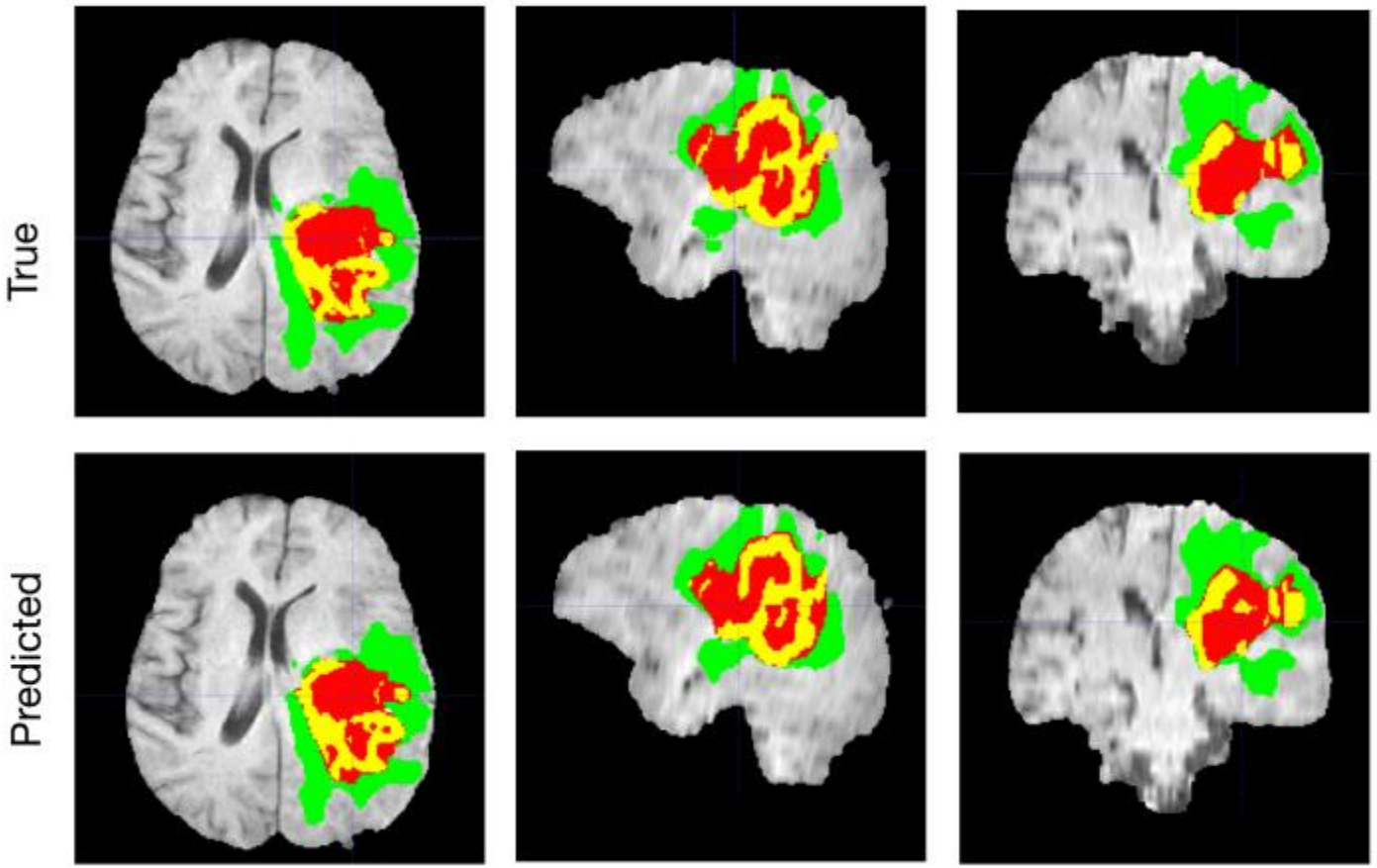


Analyzing images of plants to automatically classify them

This is **image classification**, typically performed using convolutional neural networks

PictureThis App - Online plant encyclopedia and plant identifier (picturethisai.com)

Examples of Machine Learning?



Detecting tumors in brain scans
This is **semantic segmentation**, where each pixel in the image is classified (as we want to determine the exact location and shape of tumors), typically using CNNs as well.

Examples of Machine Learning?

Automatically classifying news articles

This is **natural language processing (NLP)**, and more specifically text classification, which can be tackled using recurrent neural networks (RNNs), CNNs, or Transformers

Examples of Machine Learning?

Forecasting your company's revenue next year, based on many performance metrics

This is a **regression task** (i.e., predicting values) that may be tackled using any regression model, such as a Linear Regression or Polynomial Regression model, a regression SVM, a regression Random Forest, or an artificial neural network. If you want to take into account sequences of past performance metrics, you may want to use RNNs, CNNs, or Transformers

Examples of Machine Learning?

Making your app react to voice commands

This is **speech recognition**, which requires processing audio samples: since they are long and complex sequences, they are typically processed using RNNs, CNNs, or Transformers

Siri - Apple

Amazon Alexa Voice AI | Alexa Developer Official Site

Examples of Machine Learning?

Detecting credit card fraud
This is **anomaly detection**.
Anomaly detection
is identifying data points in data that don't fit the normal patterns. It can be useful to solve many problems including fraud detection, medical diagnosis, etc.

Examples of Machine Learning?

Recommending a product that a client may be interested in, based on past purchases

This is a **recommender system**. One approach is to feed past purchases (and other information about the client) to an artificial neural network, and get it to output the most likely next purchase. This neural net would typically be trained on past sequences of purchases across all clients.

Examples of Machine Learning?

Building an intelligent bot for a game

This is often tackled using **Reinforcement Learning**, which is a branch of Machine Learning that trains agents (such as bots) to pick the actions that will maximize their rewards over time (e.g., a bot may get a reward every time the player loses some life points), within a given environment (such as the game). The famous AlphaGo program that beat the world champion at the game of Go was built using RL.

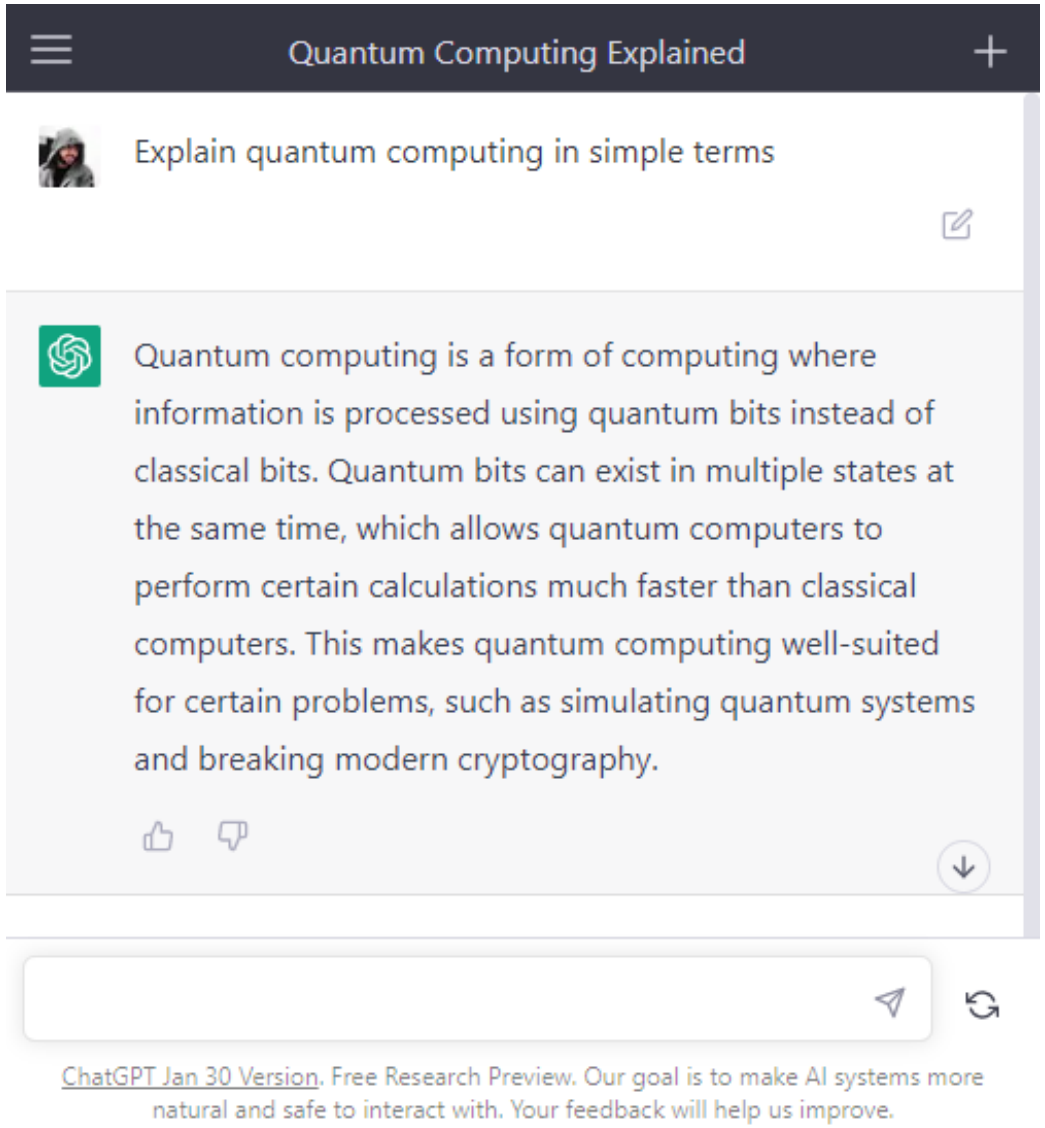
AlphaGo (deepmind.com)

Examples of Machine Learning?

Building an intelligent bot for a game
Another example of **Reinforcement Learning**:
OpenAI Five wins back-to-back games versus Dota 2 world champions OG at Finals, becoming the first AI to beat the world champions in an esports game.

OpenAI Five

Examples of Machine Learning?



Generating human-like text responses
Generally performed using Transformer architecture, Large pre-trained language model (GPT-3), Fine-tuning with additional data.
ChatGPT, a language model developed by OpenAI, used for various applications such as text generation, answering questions, and conversation. ChatGPT has been trained on a large corpus of text data and fine-tuned for specific tasks, allowing it to generate responses that are contextually relevant and semantically coherent.

Supervised, Unsupervised, Reinforcement Learning

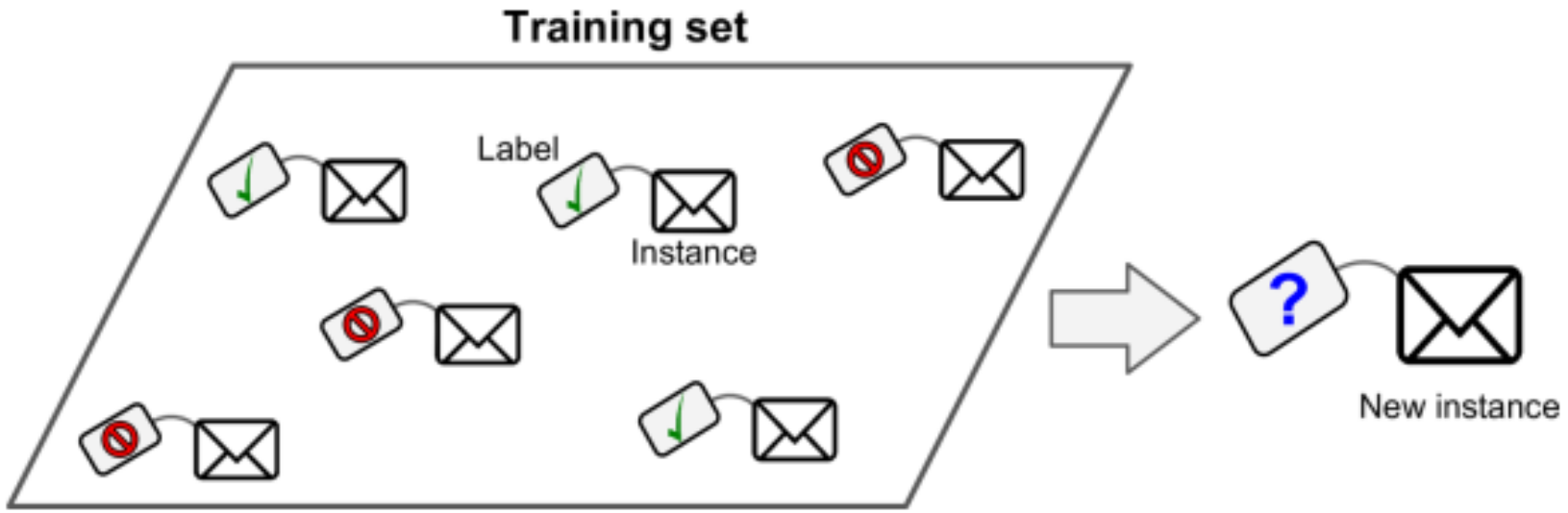
There are so many different types of Machine Learning systems that it is useful to classify them in broad categories, based on the following criteria:

- Whether or not they are trained with human **supervision** (supervised, unsupervised, semisupervised, and Reinforcement Learning)
- Whether or not they can **learn incrementally on the fly** (online versus 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**, much like scientists do (instance-based versus model-based learning)

Supervised Learning

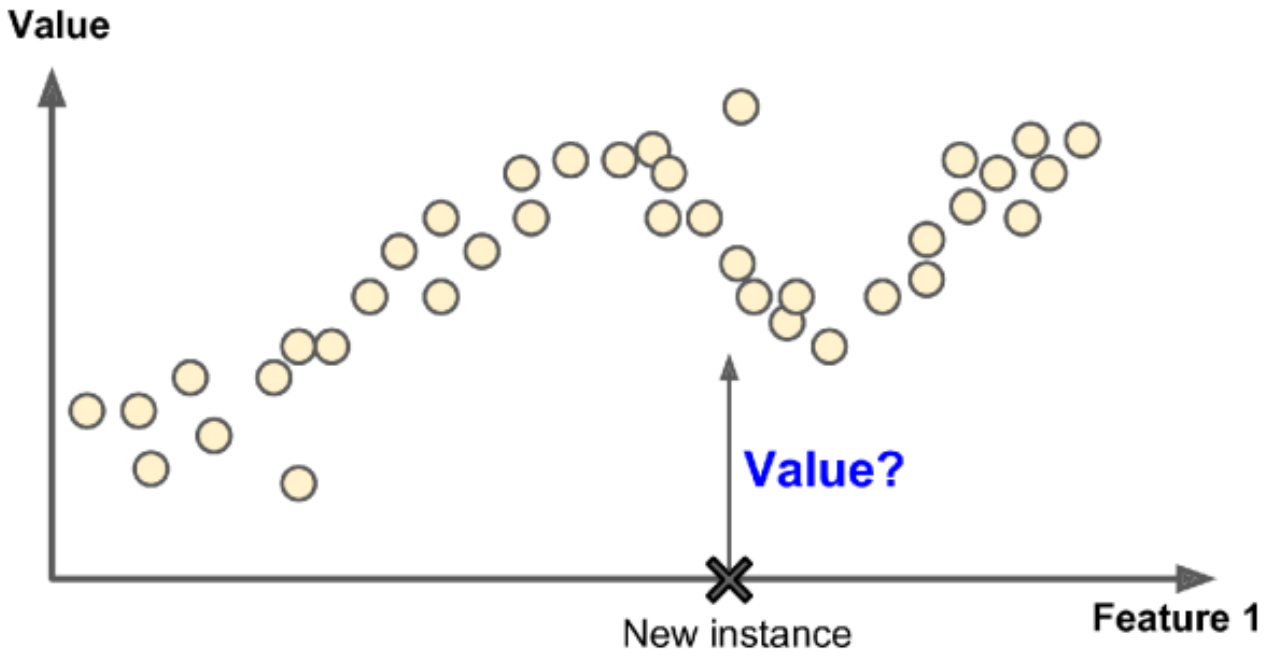
In *supervised learning*, the training set you feed to the algorithm includes the desired solutions, called *labels*

A typical supervised learning task is ***classification***. The spam filter is a good example of this: it is trained with many example emails along with their *class* (spam or ham), and it must learn how to classify new emails.



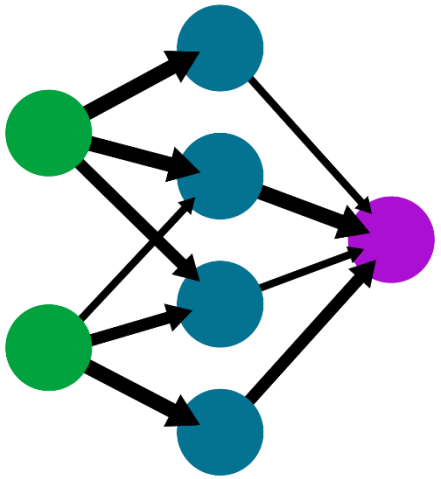
Supervised Learning

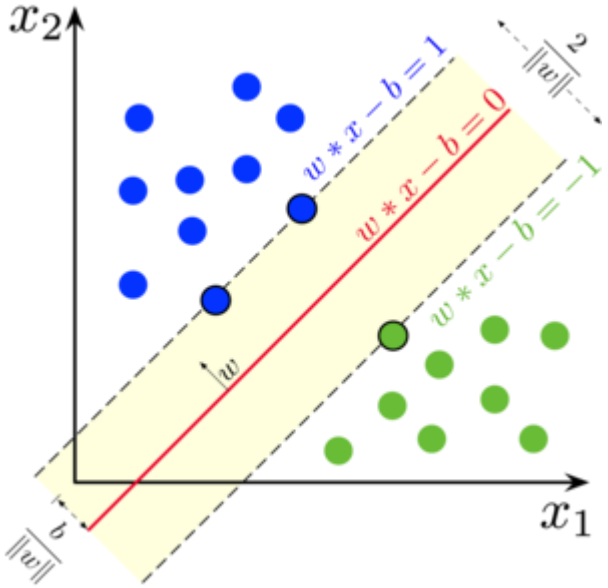
Another typical task is to predict a *target* numeric value, such as the price of a car, given a set of *features* (mileage, age, brand, etc.) called *predictors*. This sort of task is called ***regression***. To train the system, you need to give it many examples of cars, including both their predictors and their labels (i.e., their prices)

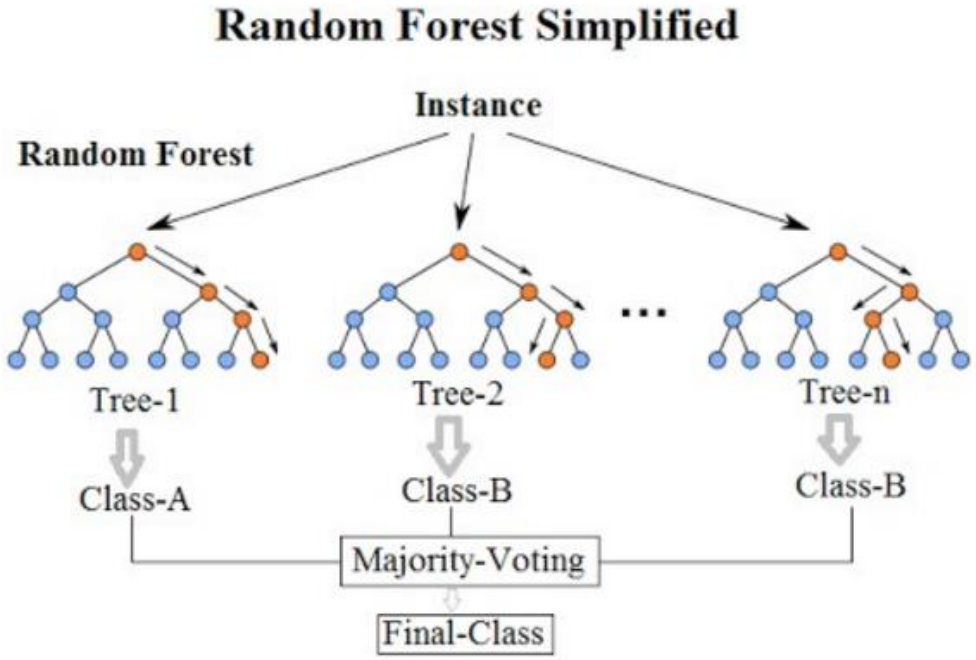


Important Supervised Algorithms

- k-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Neural networks

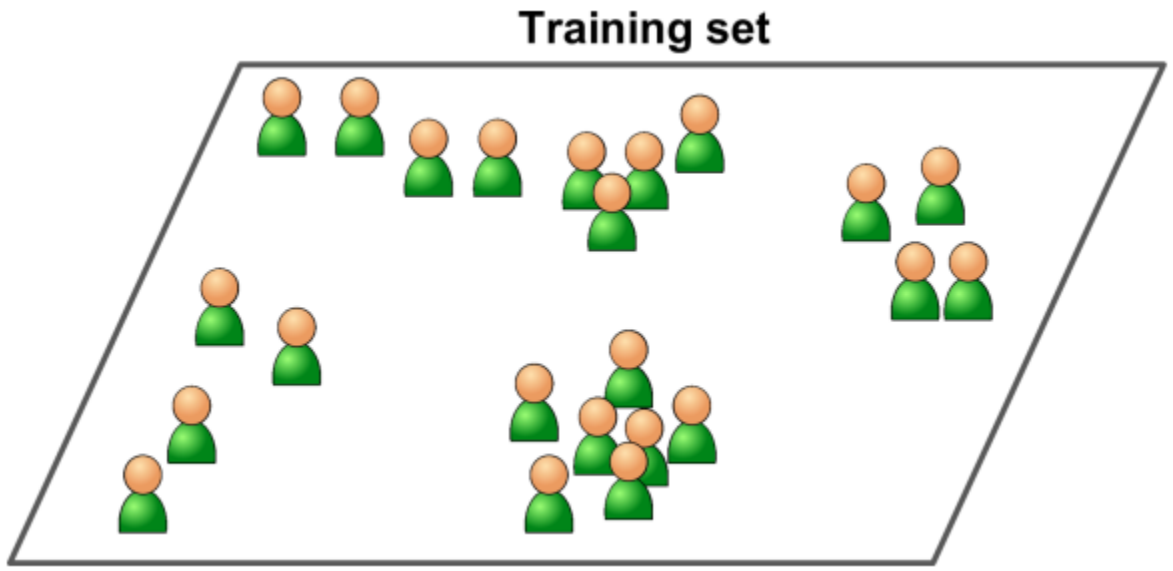






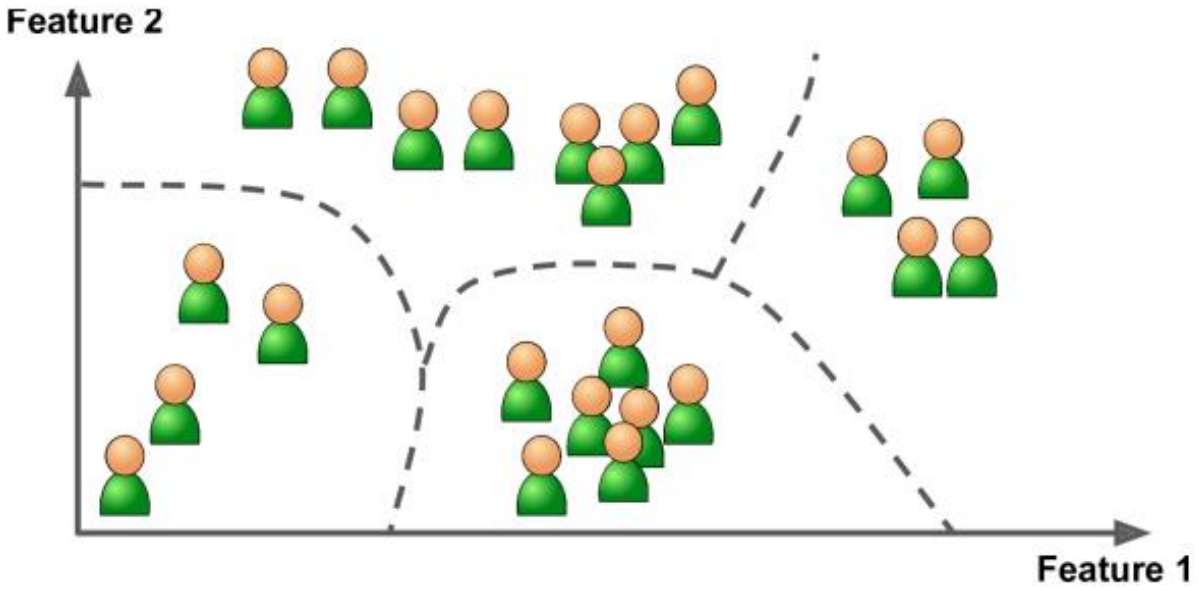
Unsupervised Learning

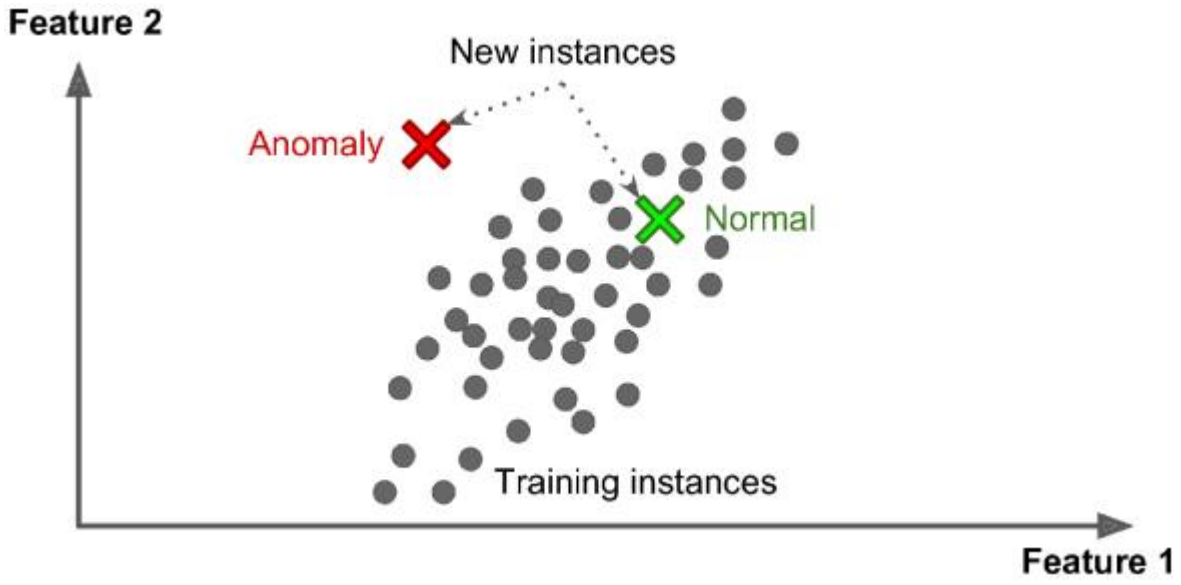
In *unsupervised learning*, as you might guess, the training data is unlabeled. The system tries to learn without a teacher.



Important Unsupervised Learning Algorithms

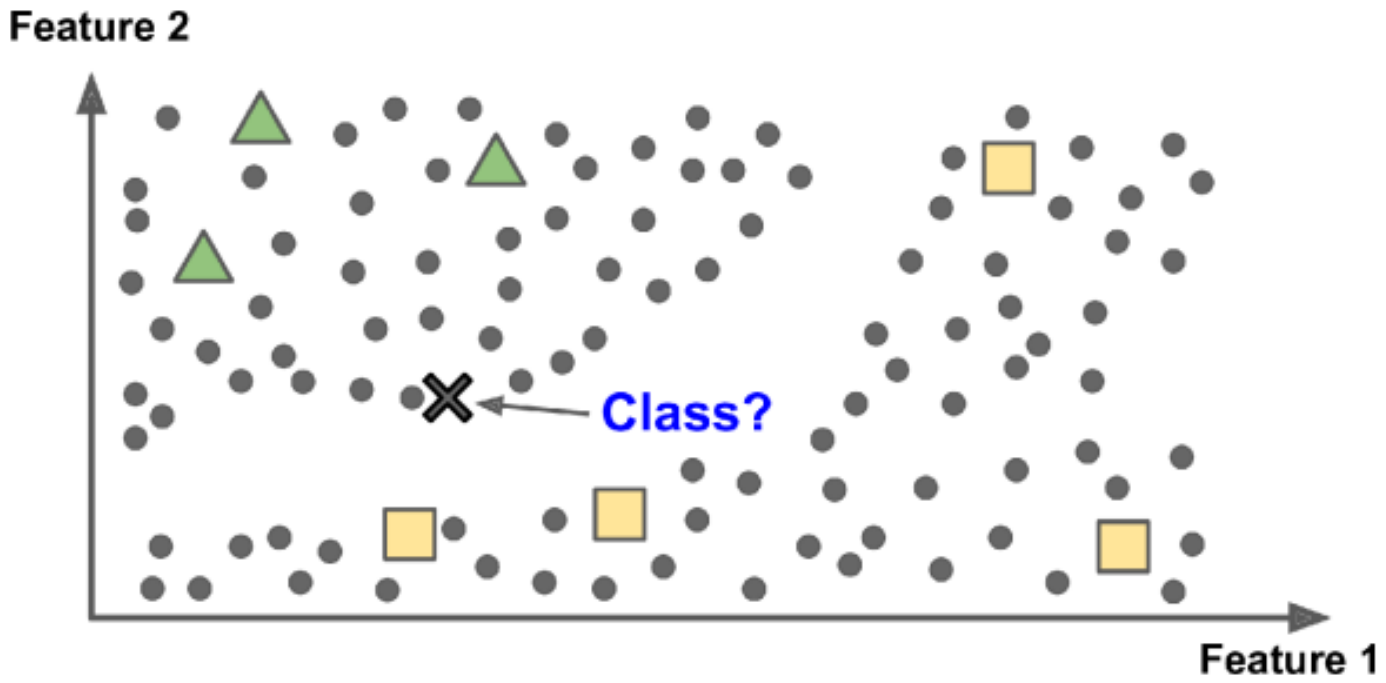
- Clustering (K-Means)
- Anomaly Detection
- Principal Component Analysis (PCA)





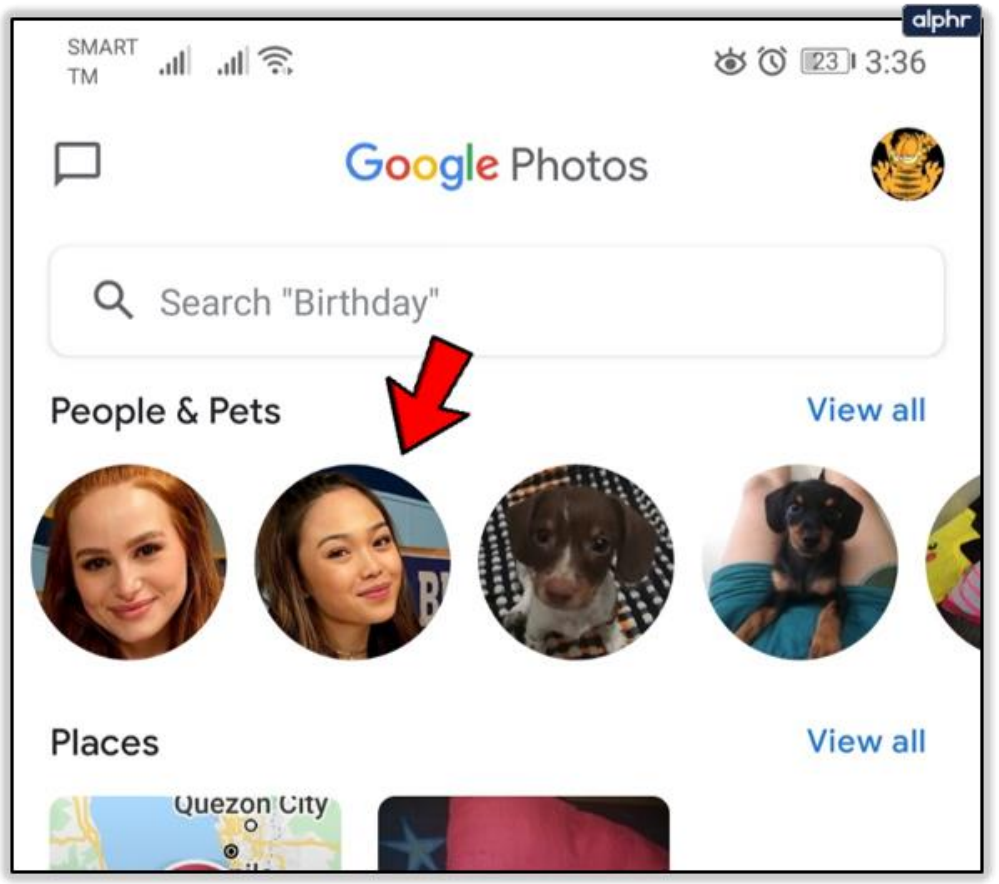
Semisupervised Learning

Since labeling data is usually time-consuming and costly, you will often have plenty of unlabeled instances, and few labeled instances. Some algorithms can deal with data that’s partially labeled.

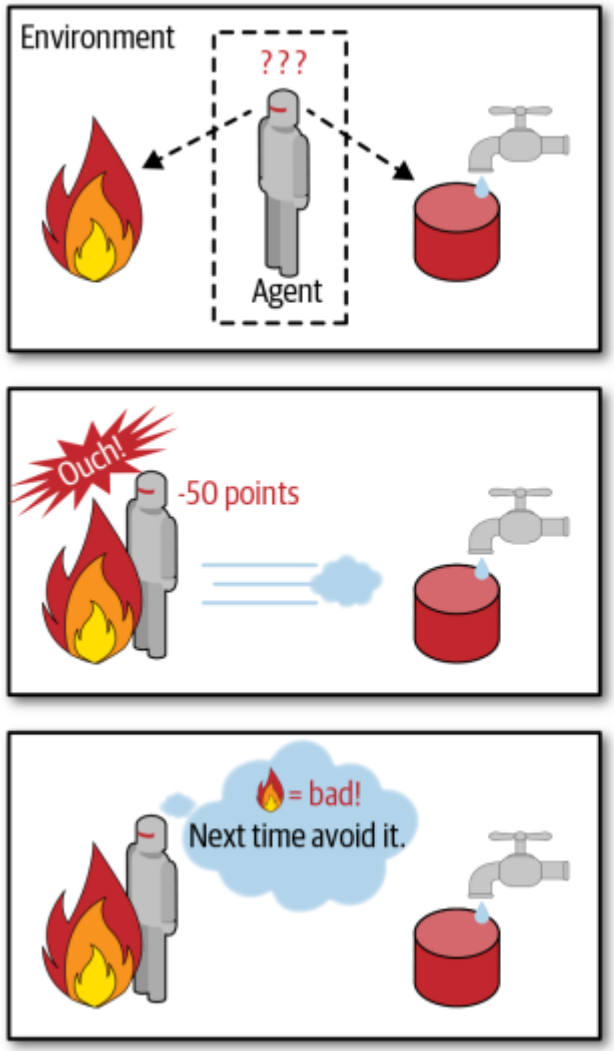


Semisupervised Learning

Some photo-hosting services, such as **Google Photos**, are good examples of this. Once you upload all your family photos to the service, it automatically recognizes that the same person A shows up in photos 1, 5, and 11, while another person B shows up in photos 2, 5, and 7. This is the unsupervised part of the algorithm (clustering). Now all the system needs is for you to tell it who these people are. Just add one label per person and it is able to name everyone in every photo, which is useful for searching photos.



Reinforcement Learning



1. Observe
2. Select action using policy
3. Action!
4. Get reward or penalty
5. Update policy (learning step)
6. Iterate until an optimal policy is found

The learning system, called an ***agent*** in this context, can observe the **environment**, select and perform actions, and get ***rewards*** in return (or ***penalties*** in the form of negative rewards). It must then learn by itself what is the best strategy, called a ***policy***, to get the most reward over time. A policy defines what action the agent should choose when it is in a given situation

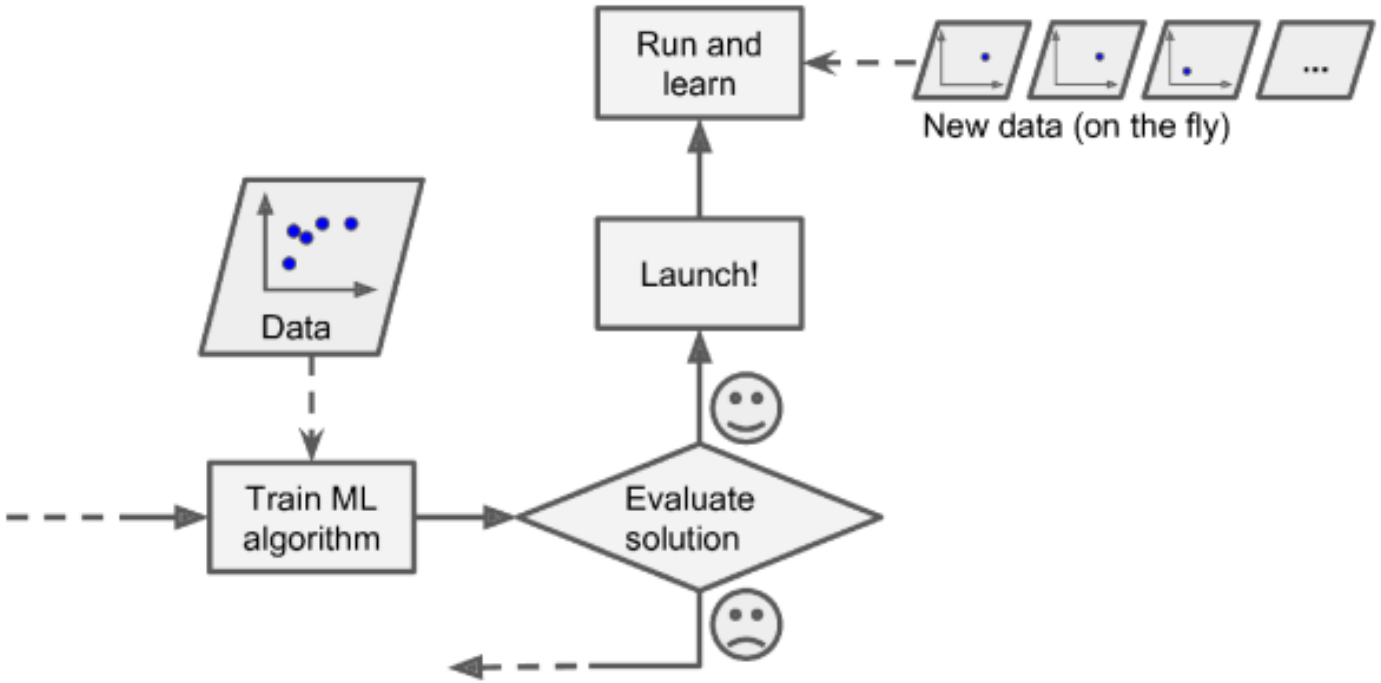
Batch Learning vs Offline Learning

In ***batch learning***, the system is incapable of learning incrementally: it must be trained using all the available data. This will generally take a lot of time and computing resources, so it is typically done offline. First the system is trained, and then it is launched into production and runs without learning anymore; it just applies what it has learned. This is called *offline learning*.

If you want a batch learning system to know about new data (such as a new type of spam), you need to train a new version of the system from scratch on the full dataset (not just the new data, but also the old data), then stop the old system and replace it with the new one. So it may use too much computation power and be very expensive.

Batch Learning vs Offline Learning

In ***online learning***, you train the system incrementally by feeding it data instances sequentially, either individually or in small groups called ***minibatches***. Each learning step is fast and cheap, so the system can learn about new data on the fly, as it arrives



Batch Learning vs Offline Learning

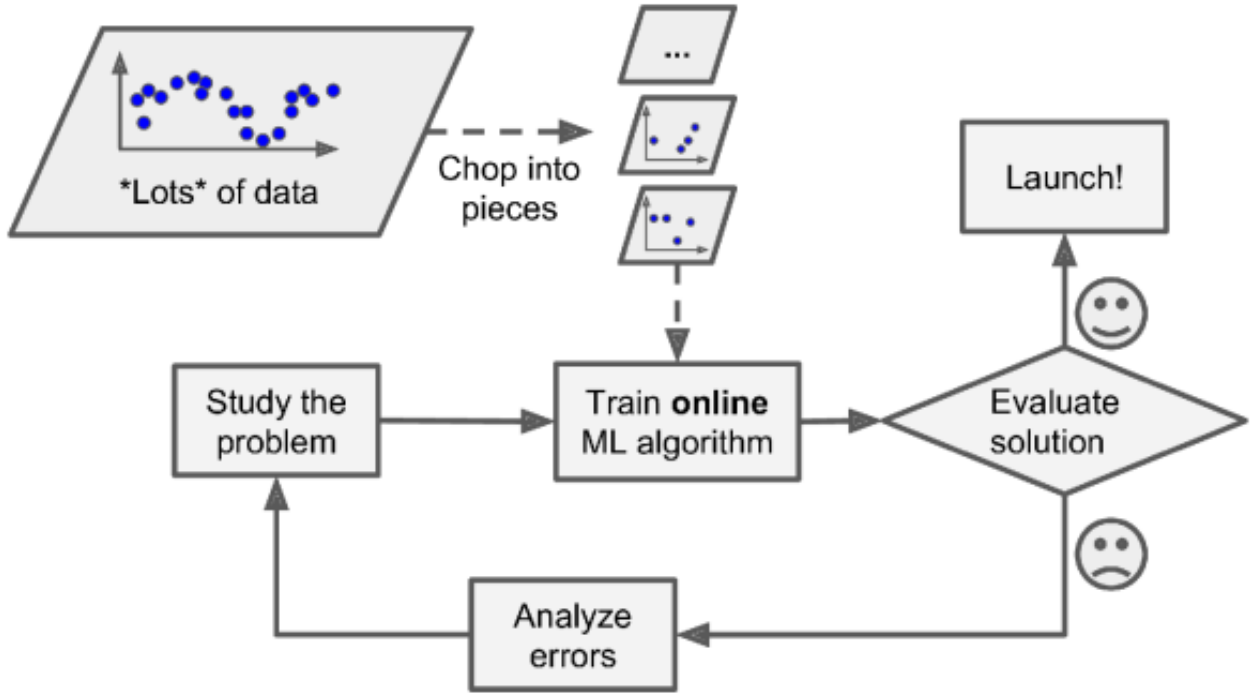
Online Learning great for:

- systems that receive data as a continuous flow (e.g., stock prices)
- if you have limited computing resources

Batch Learning vs Offline Learning

Challenge of online learning:

- Risk of performance decline from bad data fed to system
- Need to monitor system closely and switch off learning if drop in performance detected
- Monitor input data and use anomaly detection to address abnormal data.



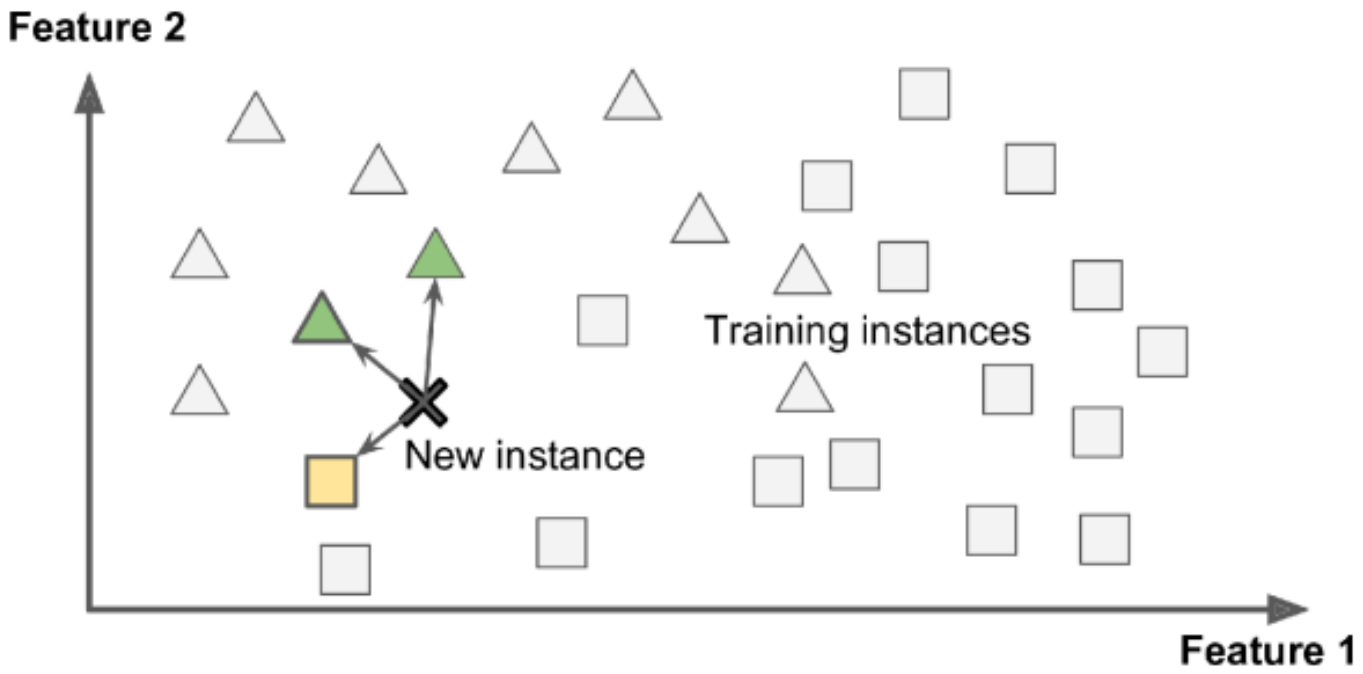
Instance-Based vs Model-Based

Instead of just flagging emails that are identical to known spam emails, your spam filter could be programmed to also flag emails that are very similar to known spam emails. This requires a ***measure of similarity*** between two emails. A (very basic) similarity measure between two emails could be to count the number of words they have in common. The system would flag an email as spam if it has many words in common with a known spam email.

This is called ***instance-based learning***: the system learns the examples by heart, then generalizes to new cases by using a similarity measure to compare them to the learned examples (or a subset of them).

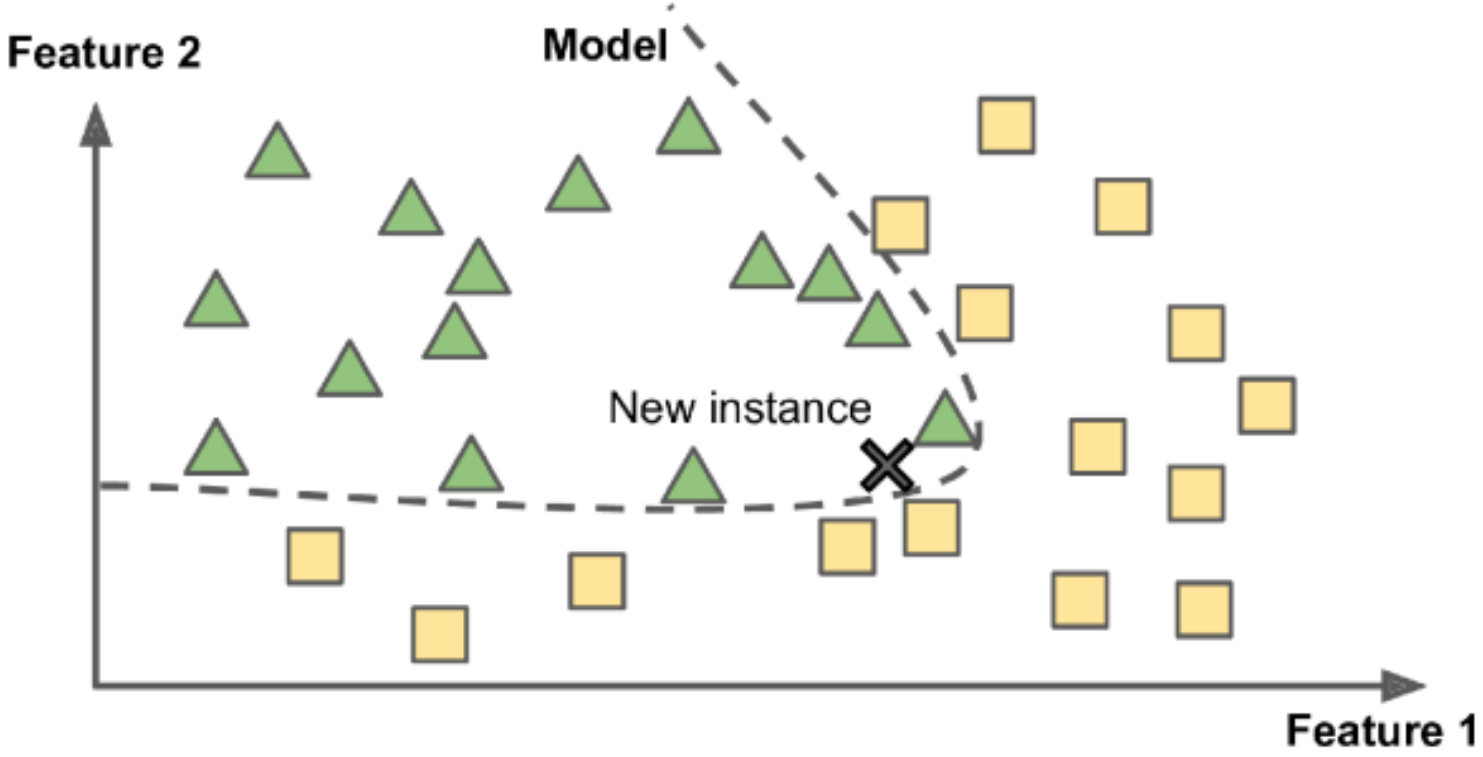
Instance-Based vs Model-Based

The new instance would be classified as a triangle because the majority of the most similar instances belong to that class.



Instance-Based vs Model-Based

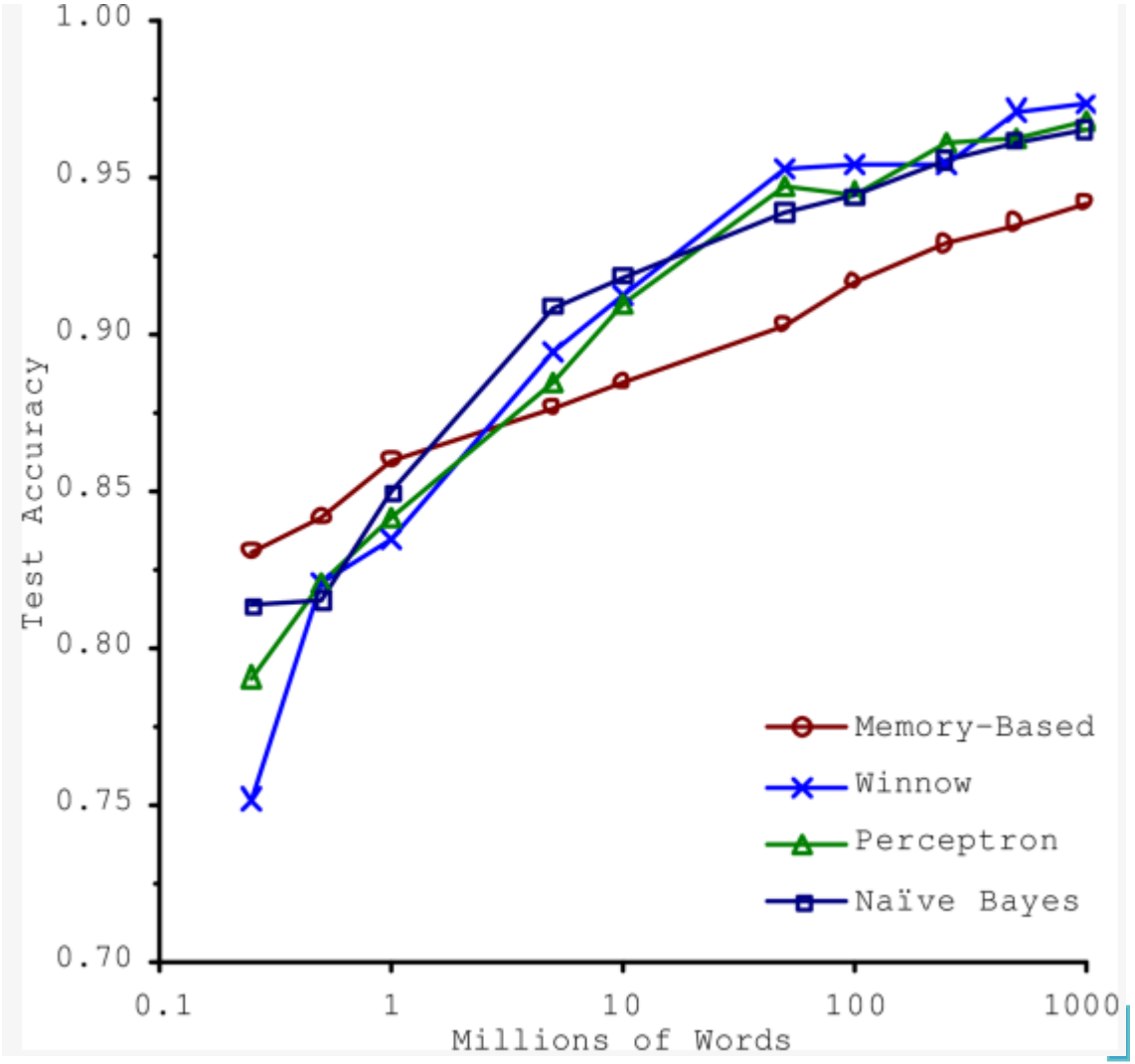
Another way to generalize from a set of examples is to build a model of these examples and then use that model to make *predictions*.



Main Challenges of Machine Learning

Insufficient Quantity of Training Data

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).

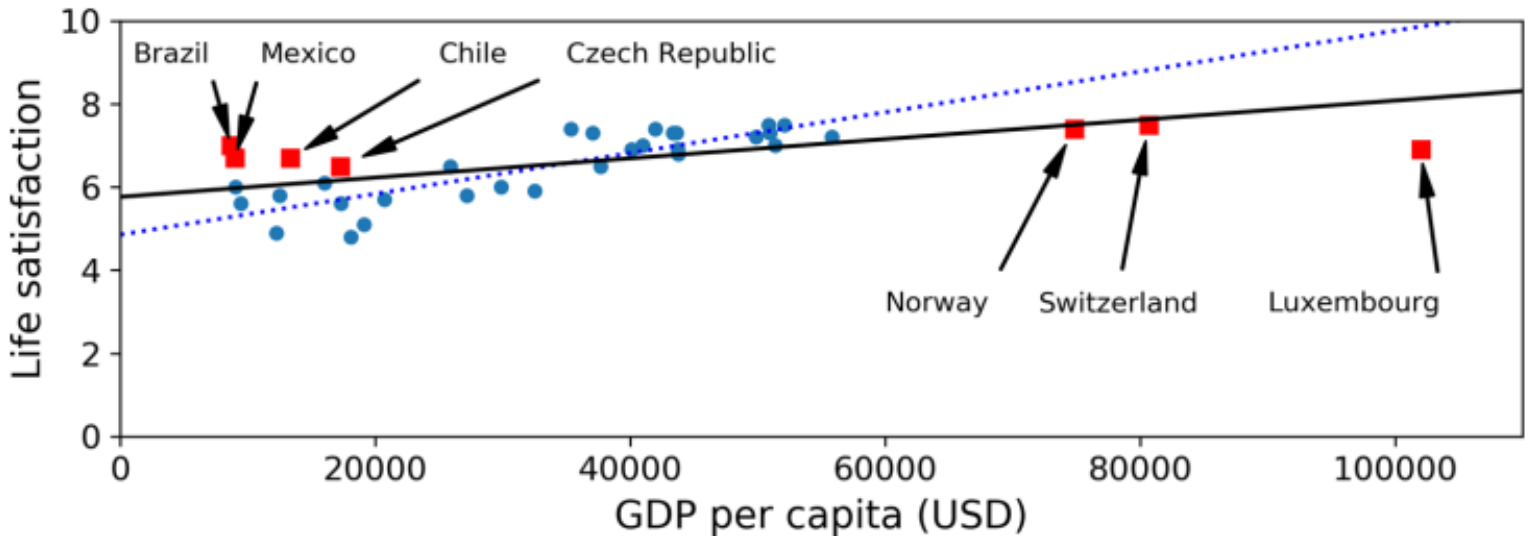


Main Challenges of Machine Learning

Nonrepresentative Training Data

In order to generalize well, it is crucial that your training data be representative of the new cases you want to generalize to. This is true whether you use instance-based learning or model-based learning.

To avoid biases, it's crucial to use a representative training set in your model. The larger the sample size, the less sampling noise you'll have. However, even a large sample can be biased if the sampling method is flawed. To get accurate results, choose a representative sample and avoid sampling bias. The following figure shows how the regressed line changes when you add some new data(solid line)



Main Challenges of Machine Learning

Irrelevant Features

As the saying goes: garbage in, garbage out. Your system will only be capable of learning if the training data contains enough relevant features and not too many irrelevant ones. A critical part of the success of a Machine Learning project is coming up with a good set of features to train on. This process, called *feature engineering*, involves the following steps:

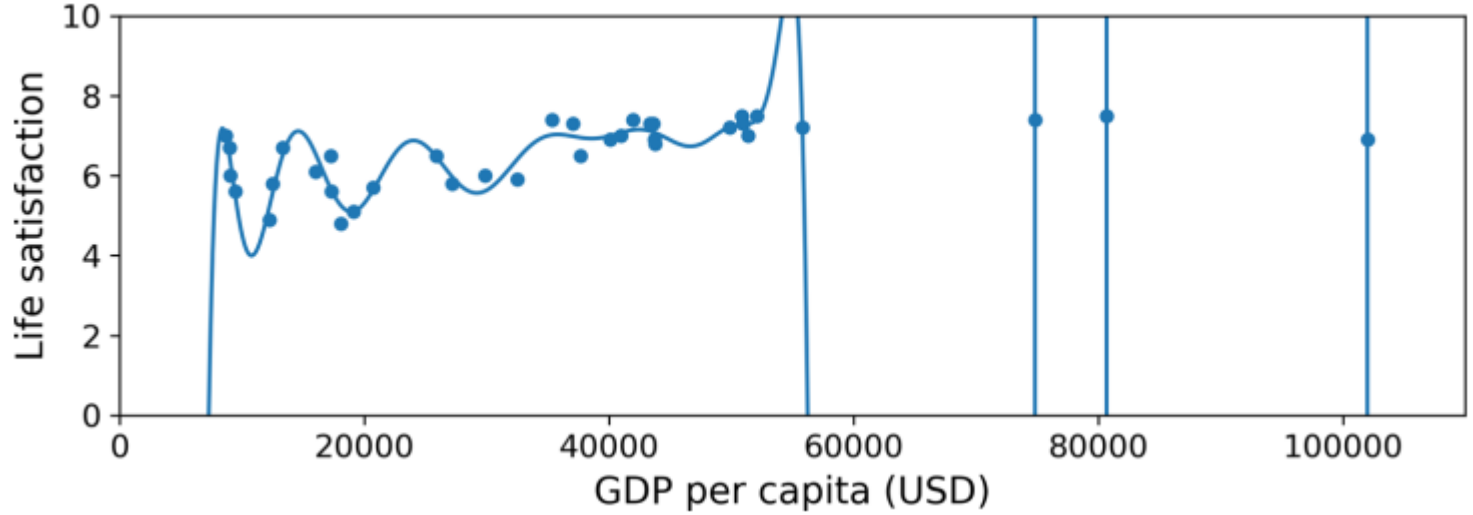
- *Feature selection* (selecting the most useful features to train on among existing features)
- *Feature extraction* (combining existing features to produce a more useful one—as we saw earlier, dimensionality reduction algorithms can help)
- Creating new features by gathering new data

Main Challenges of Machine Learning

Overfitting the Training Data

Say you are visiting a foreign country and the taxi driver rips you off. You might be tempted to say that *all* taxi drivers in that country are thieves. Overgeneralizing is something that we humans do all too often, and unfortunately machines can fall into the same trap if we are not careful. In Machine Learning this is called *overfitting*: it means that the model performs well on the training data, but it does not generalize well.

This figure, shows an example of a high-degree polynomial life satisfaction model that strongly overfits the training data. Even though it performs much better on the training data than the simple linear model, would you really trust its predictions?



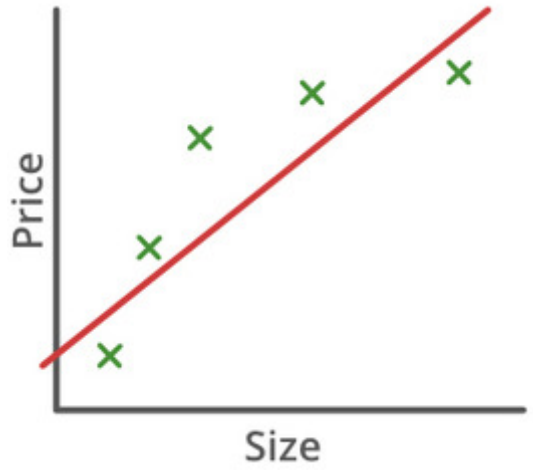
Main Challenges of Machine Learning

Underfitting the Training Data

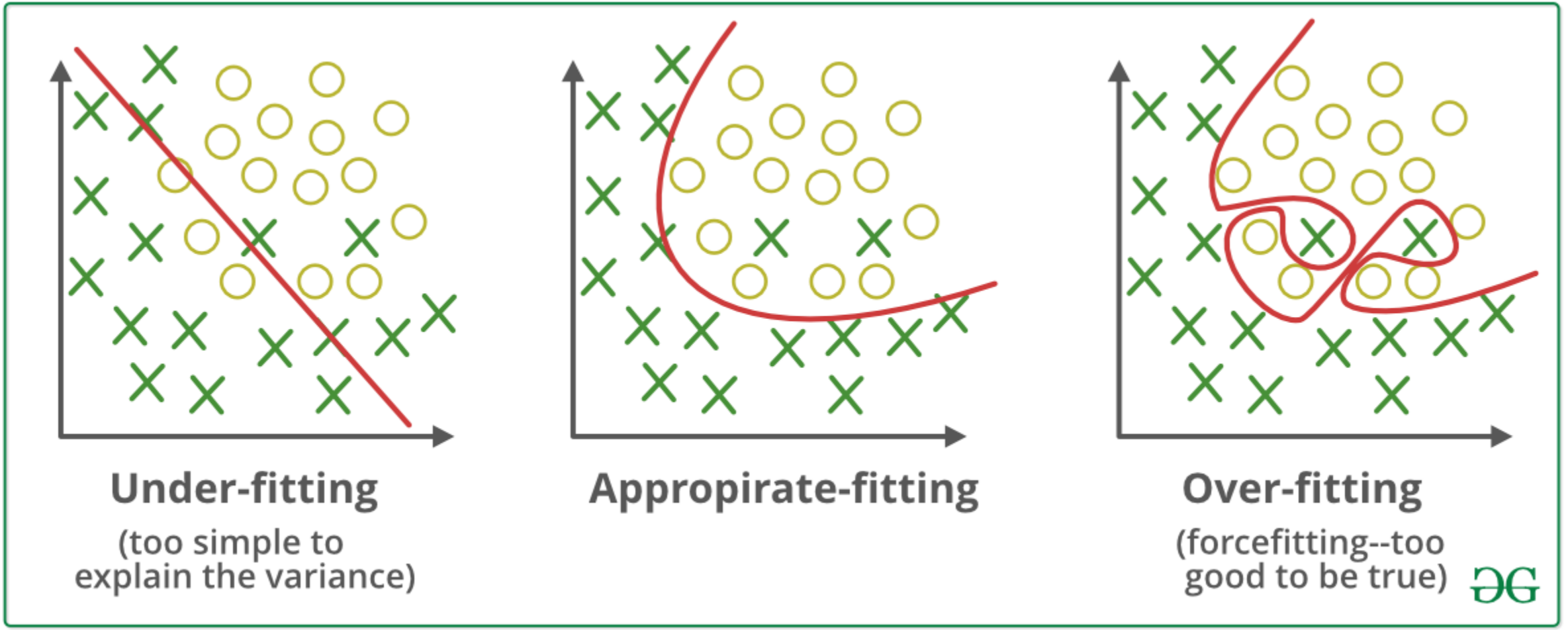
As you might guess, *underfitting* is the opposite of overfitting: it occurs when your model is too simple to learn the underlying structure of the data. For example, a linear model of life satisfaction is prone to underfit; reality is just more complex than the model, so its predictions are bound to be inaccurate, even on the training examples.

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 (e.g., reduce the regularization hyperparameter)



Main Challenges of Machine Learning



ML | Underfitting and Overfitting - GeeksforGeeks

Summary

- Machine Learning allows machines to improve at a task by learning from data instead of explicit coding.
- There are many different types of ML systems: supervised or not, batch or online, instance-based or model-based
- ML project involves gathering data for a training set, which is fed to a learning algorithm. Model-based algorithms tune parameters to fit the model to the training set. Instance-based algorithms learn the examples by heart and generalize to new cases using a similarity measure.
- The success of a machine learning project depends on the quality of the training data and finding the right balance between a model that is too simple and one that is too complex to avoid underfitting or overfitting.