Machine Learning Foundations

Jay Urbain, PhD

Credits:

- Machine Learning on Google Cloud Platform
- Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Editionby Aurélien Géron*Published by O'Reilly Media, Inc., 2019*

What Is Machine Learning?

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

- Machine Learning is a way to use standard algorithms to derive predictive insights from data and make repeated decisions

We will be using production-ready Python frameworks:

- Scikit-Learn: is easy to use ML library with a well defined API. Implements many Machine Learning algorithms efficiently.
- TensorFlow: is a more complex library for distributed numerical computation.
  - It makes it possible to train and run very large neural networks efficiently by distributing the computations across potentially hundreds of multi-GPU servers.
  - TensorFlow was created at Google and supports many of their large-scale Machine Learning applications. It was open-sourced in November 2015.
- Keras is a high level Deep Learning API that makes it very simple to train and run neural networks.
  - It can run on top of either TensorFlow, Theano or Microsoft Cognitive Toolkit (formerly known as CNTK).
  - TensorFlow comes with its own implementation of this API, called *tf.keras*, which provides support for some advanced TensorFlow features (e.g., to efficiently load data).

The main steps in a typical Machine Learning project (part 1):

- Handling, cleaning, and preparing data.
- Selecting and engineering features.
- Learning by fitting a model to data.
- Optimizing a cost function
- Selecting a model and tuning hyperparameters using cross-validation.
- The main challenges of Machine Learning, in particular underfitting and overfitting (the bias/variance tradeoff).
- Reducing the dimensionality of the training data to fight the curse of dimensionality.
- The most common learning algorithms: Linear and Polynomial Regression, Logistic Regression, Decision Trees, Random Forests, and Ensemble methods.

Not covered:

- k-Nearest Neighbors, Support Vector Machines.
- Most unsupervised learning techniques, including clustering, density estimation and anomaly detection. – take Data Science
- Reinforcement learning – take AI course

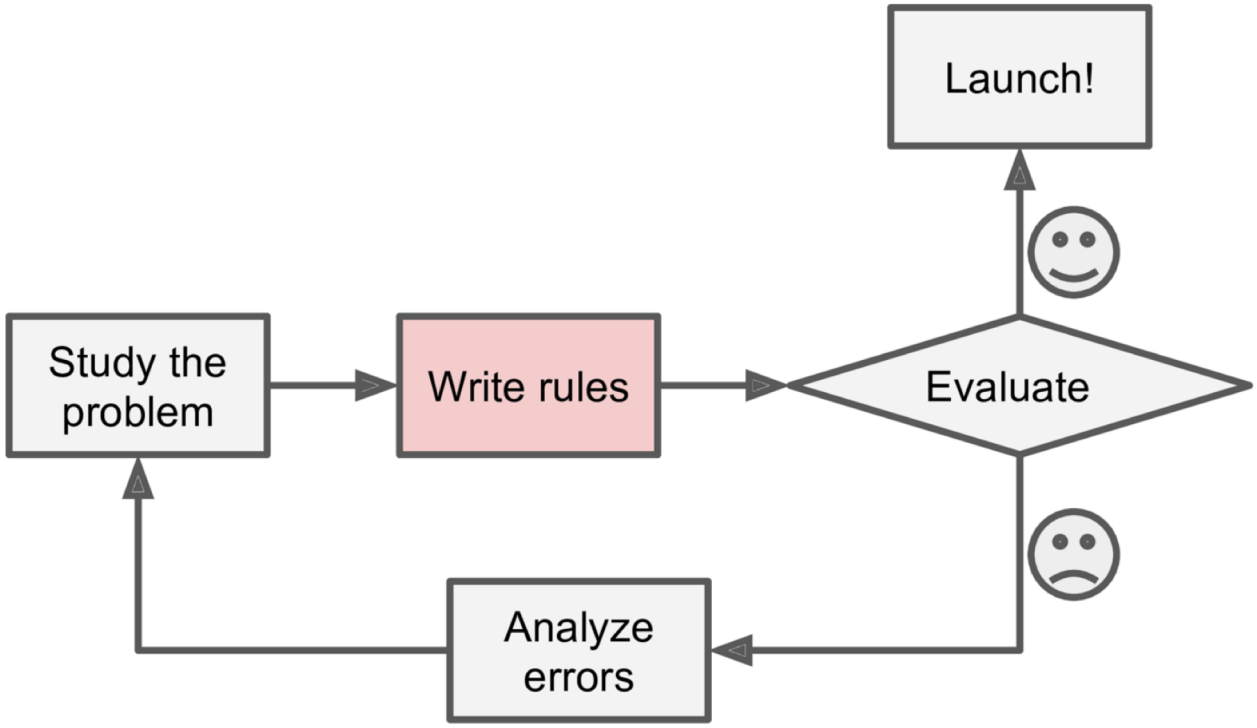
Neural Networks / Deep Learning (part 2)

- What are neural nets? What are they good for?
- Building and training neural nets using TensorFlow and Keras.
- The most important neural net architectures: feedforward neural nets, convolutional nets, recurrent nets, long short-term memory (LSTM) nets, self-attention, autoencoders and generative adversarial networks (GANs).
- Techniques for training deep neural nets.
- Scaling neural networks for large datasets.

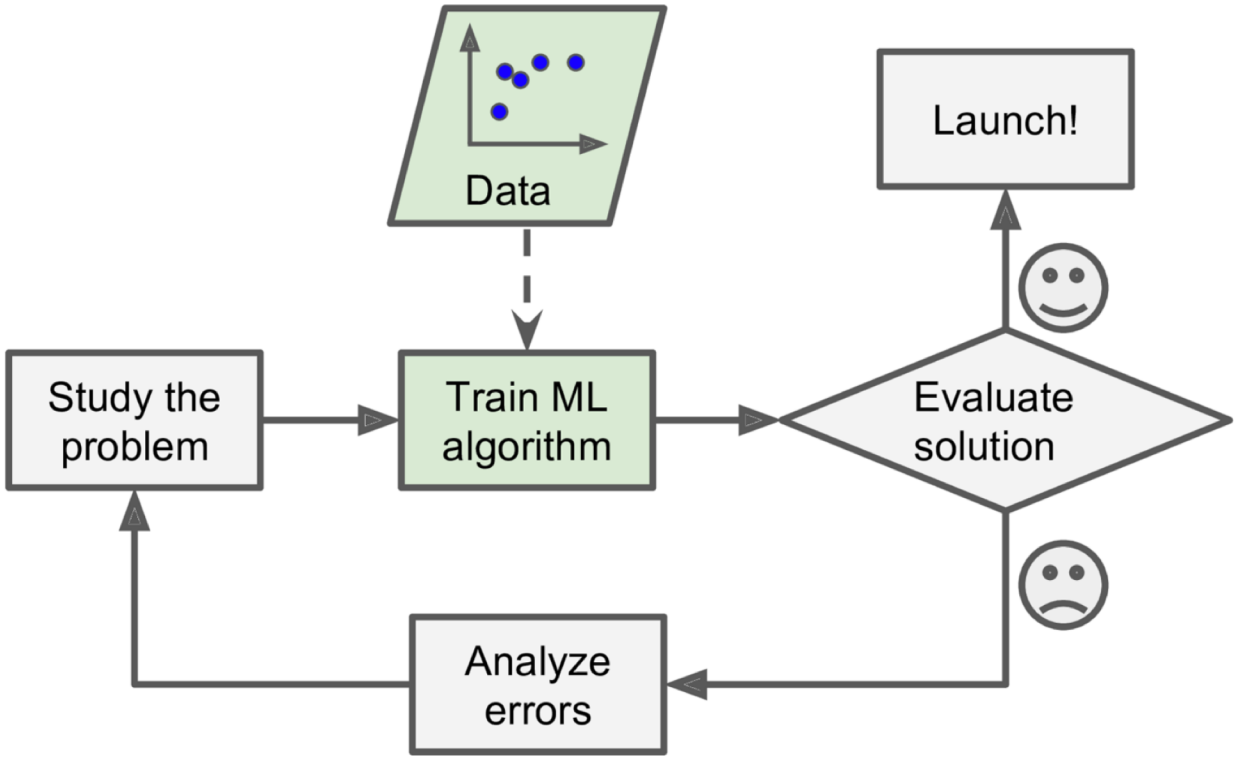
Not covered:

- Learning strategies with Reinforcement Learning - AI
- Handling uncertainty with Bayesian Deep Learning.

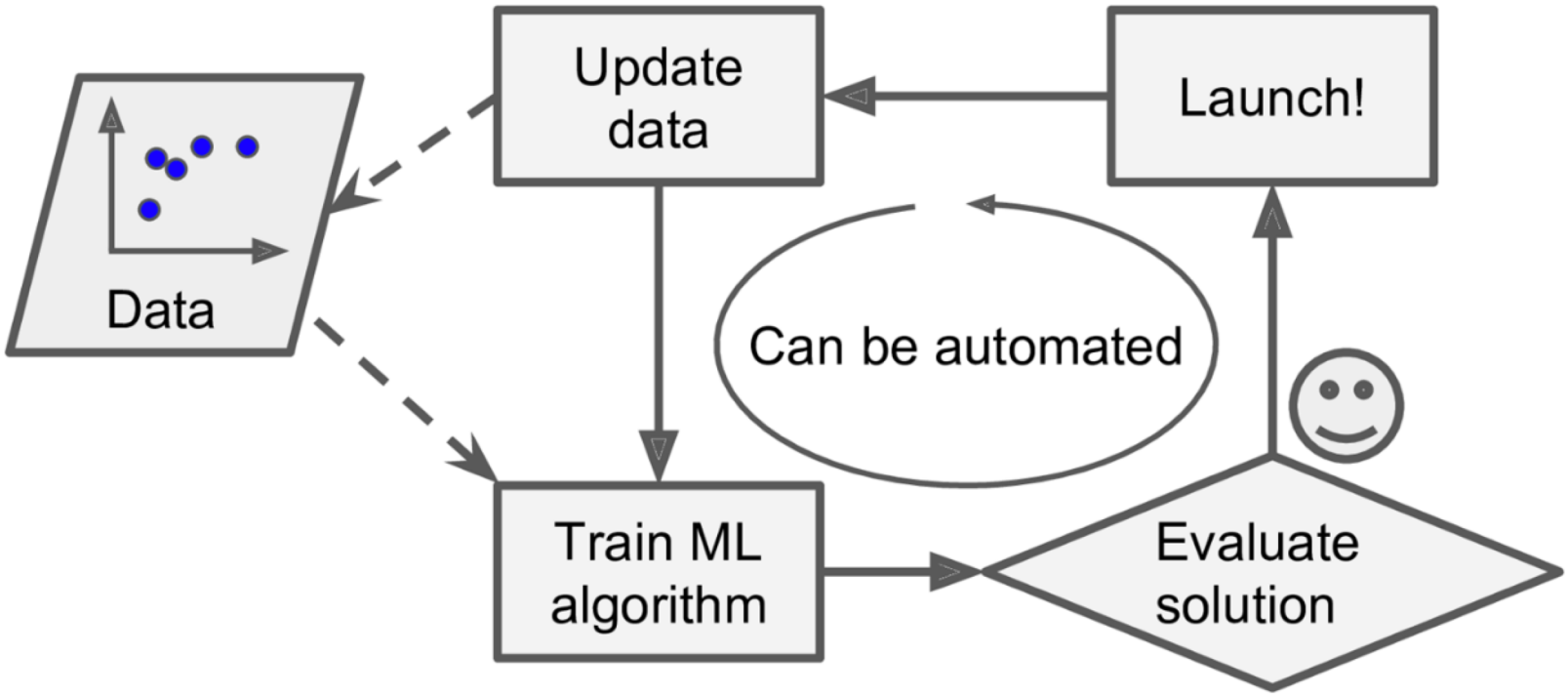
Traditional Development



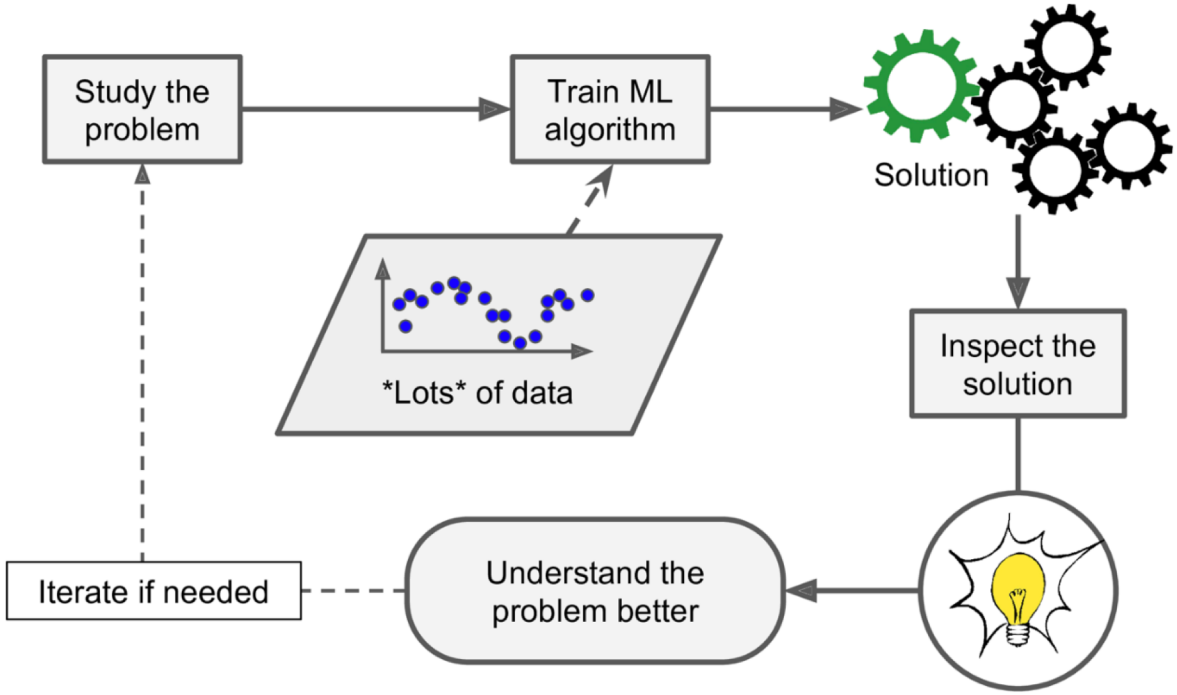
Machine Learning Approach



Automatically Adapting to Change with ML

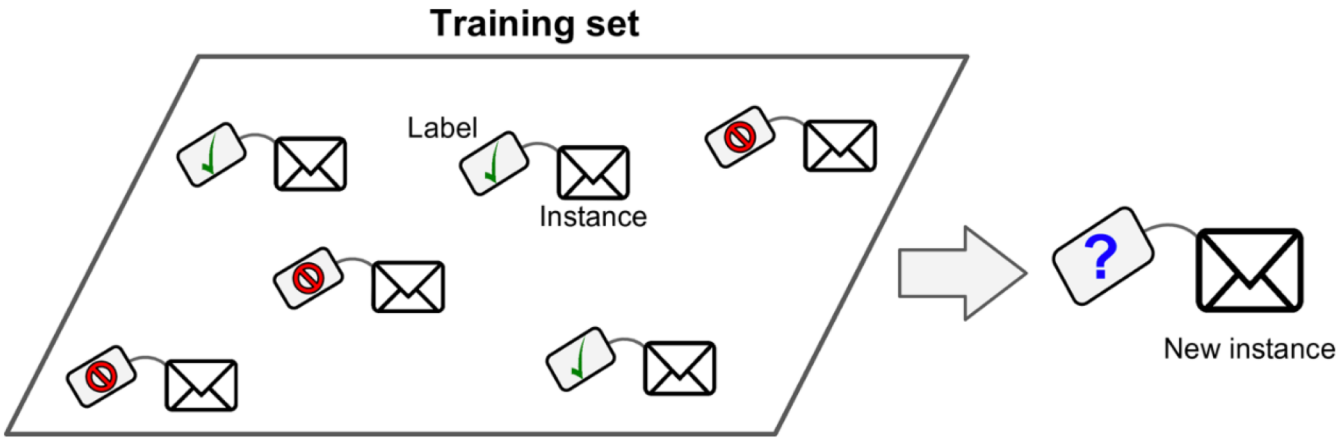


Machine Learning Helping Humans



Supervised Learning

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

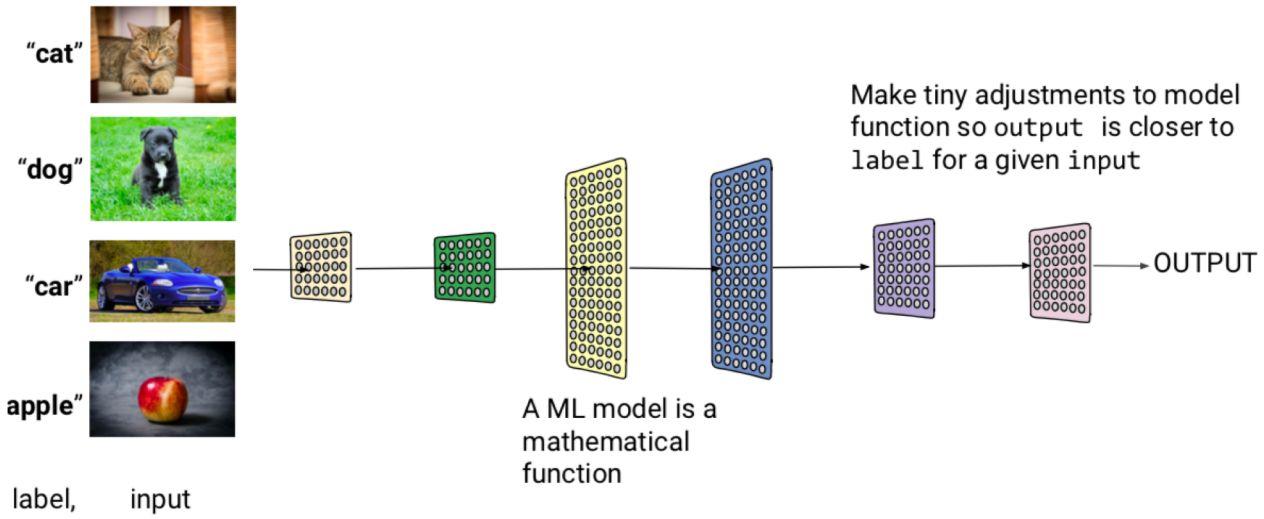


Supervised Learning

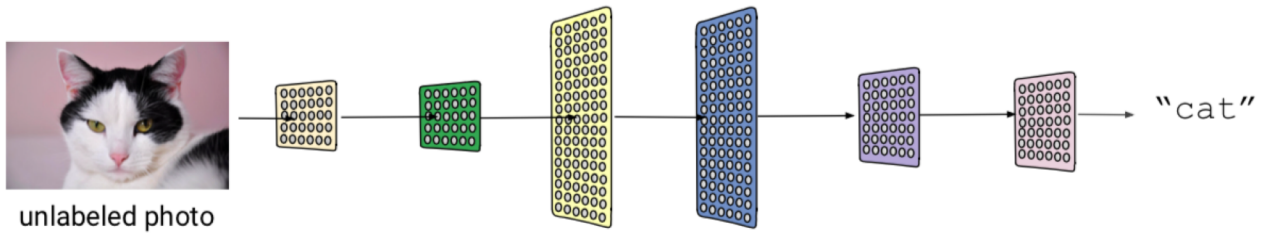
- Here are some of the most important supervised learning algorithms:
- k-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Neural networks
- Probabilistic Graphical Models

Regression

State 1: Train an ML model with examples



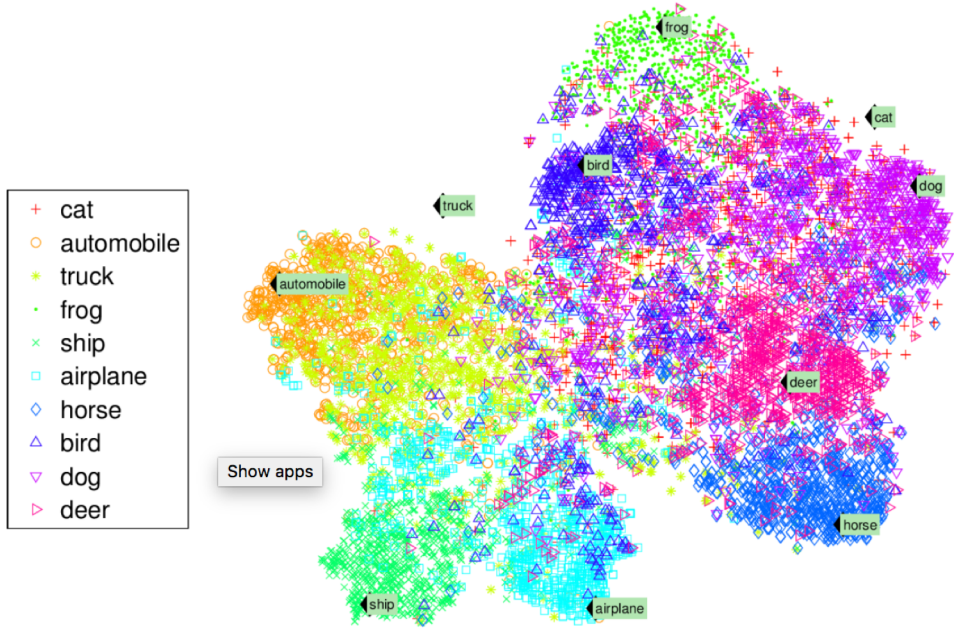
Stage 2: Predict with a trained model



Unsupervised Learning

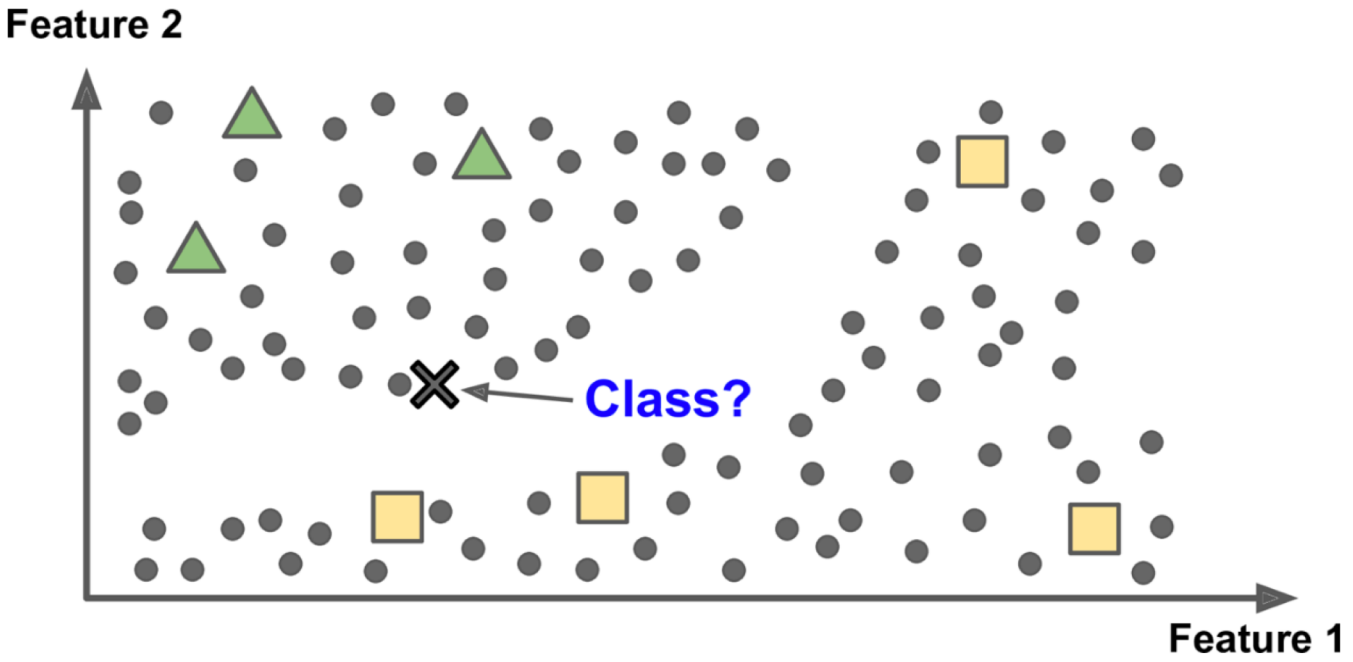
- In *unsupervised learning*, the training data is unlabeled

- Clustering
  - K-Means
  - DBSCAN
  - Hierarchical Cluster Analysis (HCA)
- Anomaly detection and novelty detection
  - One-class SVM
  - Isolation Forest
- Visualization and dimensionality reduction
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Locally-Linear Embedding (LLE)
  - t-distributed Stochastic Neighbor Embedding (t-SNE)
- Association rule learning
  - Apriori
  - Eclat



Semi-supervised Learning

- Some algorithms can deal with partially labeled training data, usually a lot of unlabeled data and a little bit of labeled data.

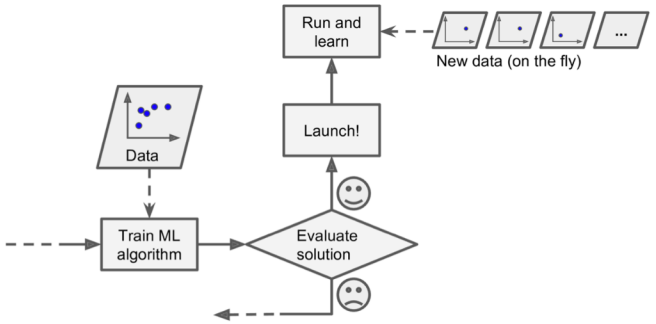


Reinforcement Learning

- *Reinforcement Learning* is a very different.
- 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.

Online Learning

- In *online learning*, you train the system incrementally by feeding it data instances sequentially, either individually or by small groups called *mini-batches*.
- Each learning step is fast and cheap, so the system can learn about new data on the fly.
- Online learning is great for systems that receive data as a continuous flow (e.g., stock prices) and need to adapt to change rapidly or autonomously.
- Good for restricted resources.



Framing a (supervised) ML Problem

Cast this as Machine Learning problem:

- What is being predicted?
- What data is needed?

Cast the ML problem as a software problem:

- What is the API for the problem during prediction?
- Who will use this service? How are they doing it today?

Now, cast it in the framework of a data problem:

- What are some key actions to collect, analyze, predict, and react to the data/predictions (different input features might require different actions)

ML Use Cases

Manufacturing

- Predictive maintenance or condition monitoring
- Warranty reserve estimation
- Propensity to buy
- Demand forecasting
- Process optimization
- Telematics

Retail

- Predictive inventory planning
- Recommendation engines
- Upsell and cross-channel marketing
- Market segmentation and targeting
- Customer ROI and lifetime value

Healthcare and Life Sciences

- Alerts and diagnostics from real-time patient data
- Disease identification and risk satisfaction
- Patient triage optimization
- Proactive health management
- Healthcare provider sentiment analysis

What is the ML problem? What is the software problem? What is the data problem?

ML Use Cases

Travel and Hospitality

- Aircraft scheduling
- Dynamic pricing
- Social media – consumer feedback and interaction analysis
- Customer complaint resolution
- Traffic patterns and congestion management

Financial Services

- Risk analytics and regulation
- Customer Segmentation
- Cross-selling and up-selling
- Sales and marketing campaign management
- Credit worthiness evaluation

Energy, Feedstock and Utilities

- Power usage analytics
- Seismic data processing
- Carbon emissions and trading
- Customer-specific pricing
- Smart grid management
- Energy demand and supply optimization

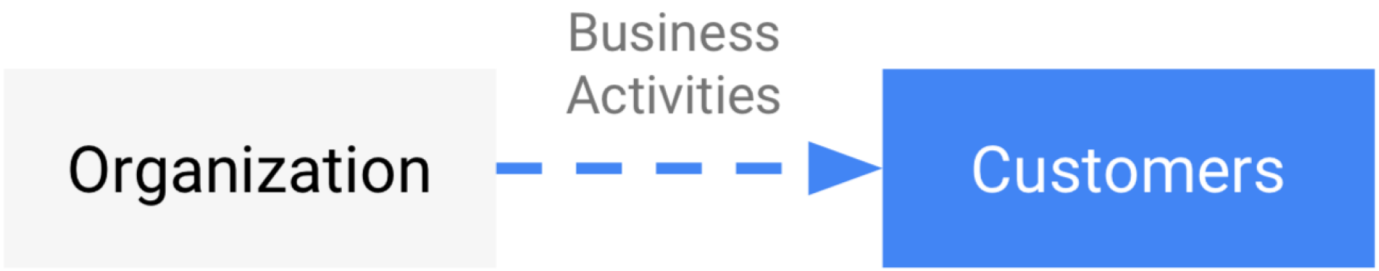
What is the ML problem? What is the software problem? What is the data problem?

Impact of ML

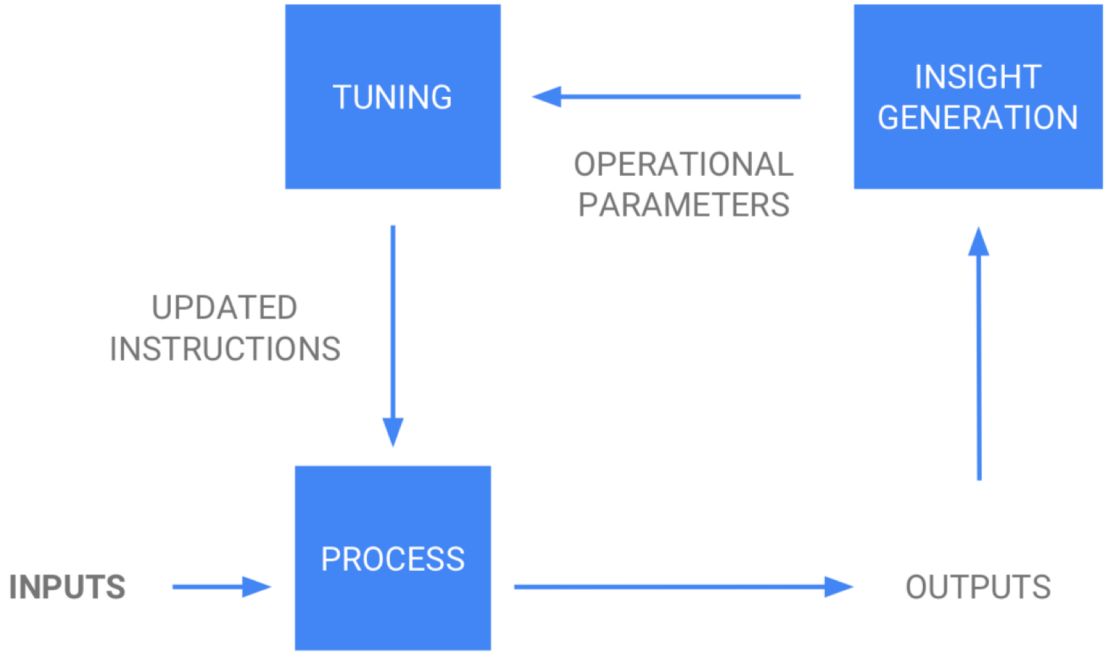
Most of the impact from ML comes along the way to developing a solution:

- Carefully analyzing your problem --> better understanding, less bias in decision making → Engineering an intelligent, scalable solution.

Evolution of a business process



General Feedback Loop



Path to ML

- **Dangers of skipping this step:**
  - Inability to scale
  - Product heads make big, incorrect assumptions that are hard to change later
- **Dangers of lingering too long here:**
  - One person gets skilled and then leaves
  - Fail to scale up the process to meet demand in time

Path to ML

- **Dangers of skipping this step:**
  - Not forced to formalize the process
  - Inherent diversity in human responses become a testbed--great product learning opportunity
  - Great ML systems will need humans in the loop
- **Dangers of lingering too long here:**
  - Paying a high marginal cost to serve each user
  - More voices will say automation isn't possible
  - Organizational lock-in

Path to ML

- **Dangers of skipping this step:**
  - You will always need infrastructure
  - IT project and ML success tied and the whole project will fail if either does

- **Dangers of staying here too long:**
  - Your competitors are collecting data and tuning their offers from these new insights

Path to ML

- **Dangers of skipping this step:**
  - Unclean data means no ML training
  - You can’t measure success

- **Dangers of staying here too long:**
  - Limit the complexity of problems you can solve

Path to ML

1 - Individual contributor

2 - Delegation

3 - Digitization

4 - Big Data and Analytics

5 - Machine learning

Automated
feedback loop
that can outpace
human scale

Evolved business process

How change happens in phases:

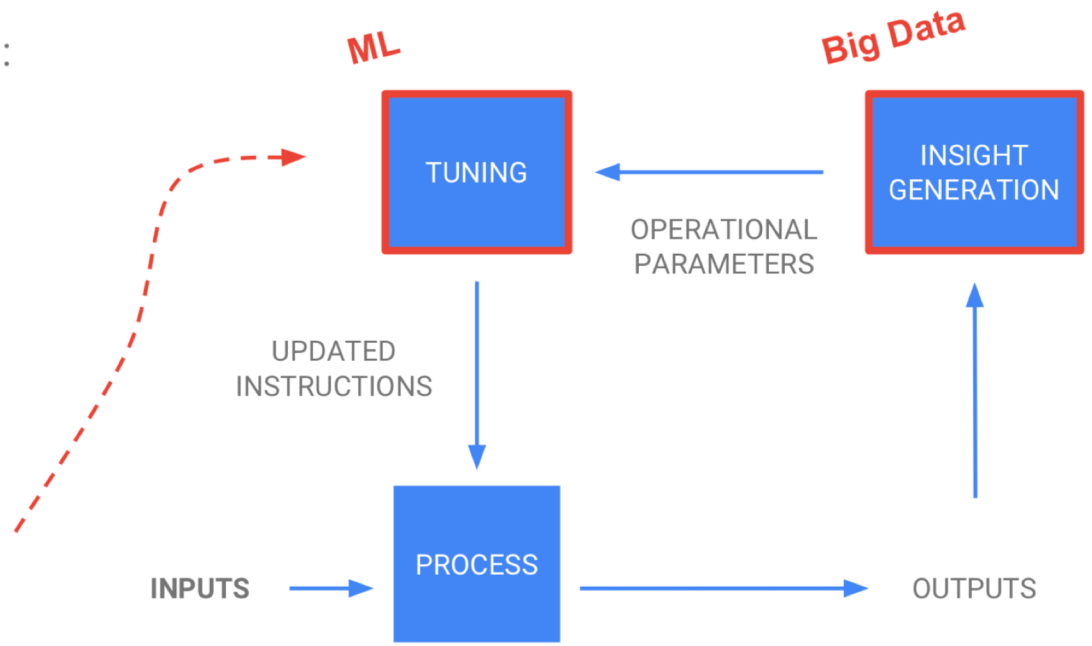
Step 1 - Individual contributor

Step 2 - Delegation

Step 3 - Digitization

Step 4 - Big Data and Analytics

Step 5 - Machine learning

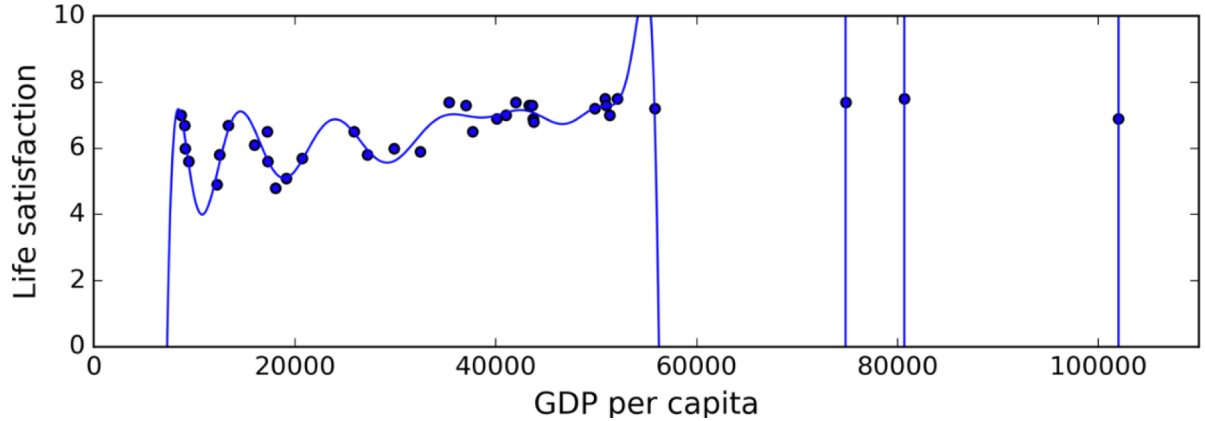


Main Challenges in ML

- Two things that can go wrong: “bad algorithm” and “bad data.”

Bad Data

- Insufficient Quantity of Training Data
- Non-representative Training Data
- Poor-Quality Data
- Irrelevant Features
- Underfitting the Training Data
- Overfitting the Training Data



Bad Algorithm

- The only way to know how well a model will generalize to new cases is to actually try it out on new cases.
- Split your data into two sets: the *training set* and the *test set*.
- Train your model using the training set, and test it using the test set.
- The error rate on new cases is called the *generalization error* (or *out-of-sample error*), and by evaluating your model on the test set, you get an estimate of this error. This value tells you how well your model will perform on instances it has never seen before.
- If the training error is low (i.e., your model makes few mistakes on the training set) but the generalization error is high, it means that your model is overfitting the training data.

Demo walk through ….