WQD7001 Machine Learning

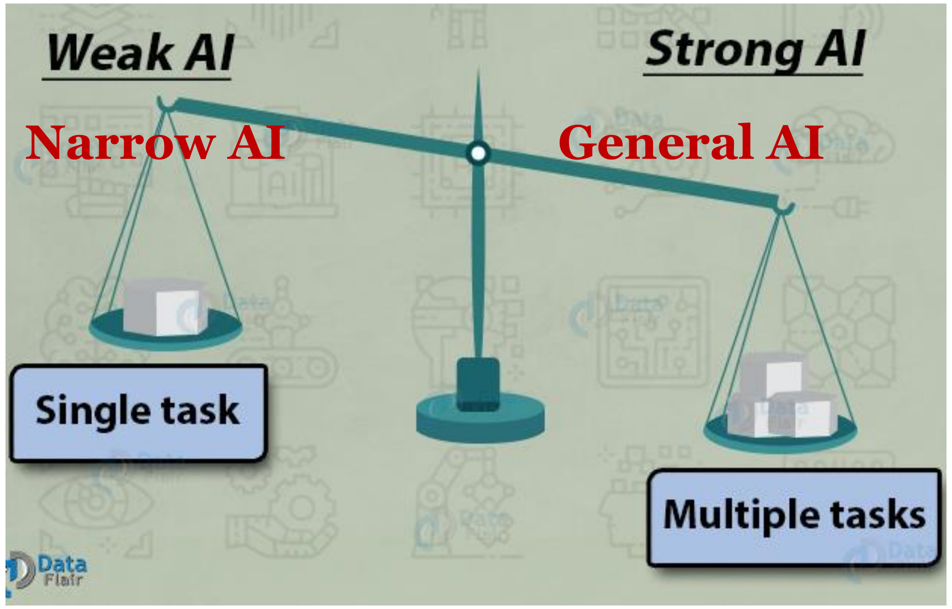
By Dr. Salimah M

Learning Objectives:

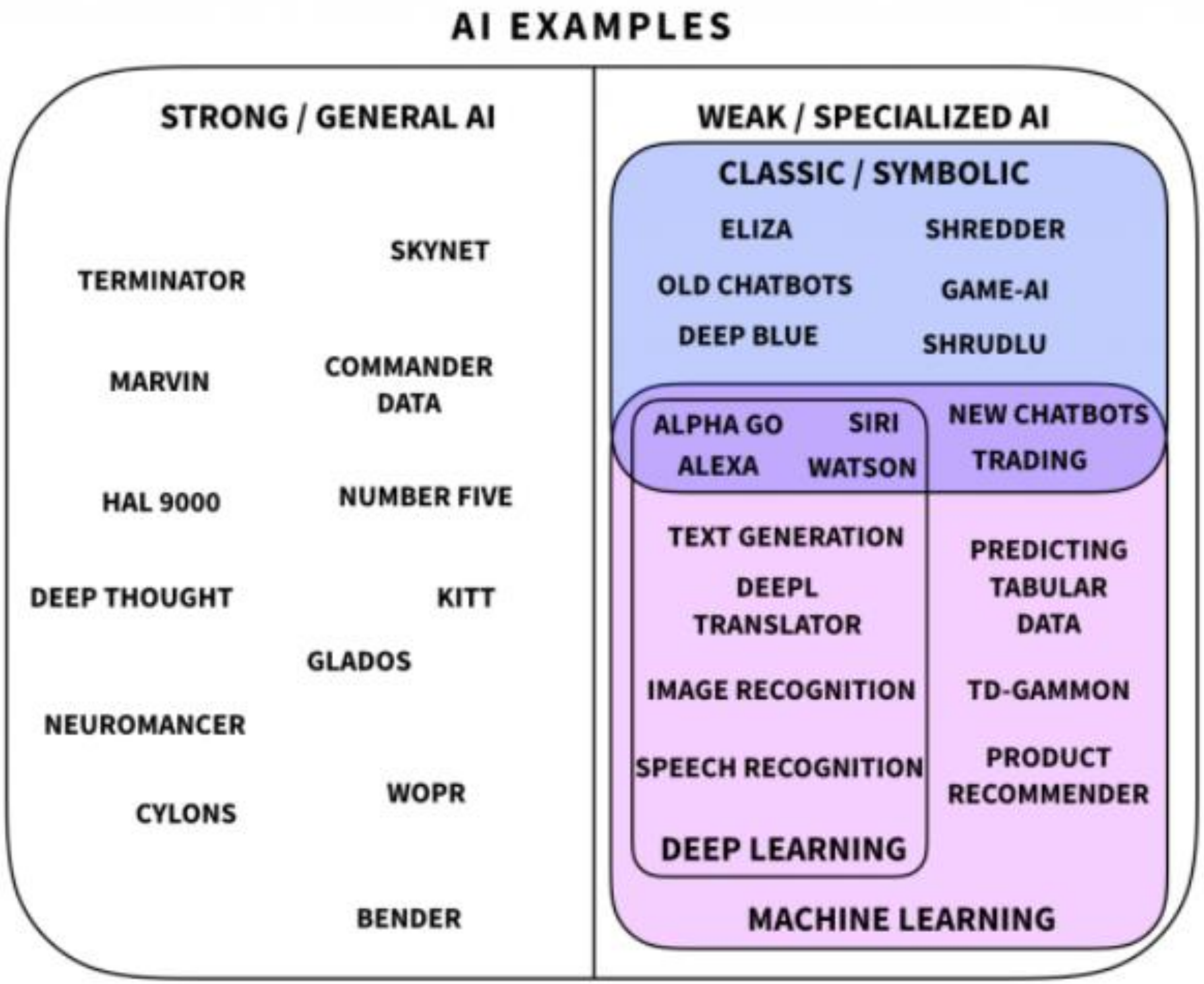
1. To review AI and introduce ML
2. To present examples and application of ML
3. To categorize types of ML algorithms
4. To explain inductive learning
5. To describe the Applied ML process
6. To discuss several important ML concepts.

What is Artificial Intelligence (AI)?

AI is the imitation or cloning of human intelligence, that allows machines, more specifically computer systems, to perform in an extremely intelligent manner.



Strong AI (artificial general intelligence)	Weak AI
complex physical, biological and social world knowledge	no knowledge at all or enormously specific, database knowledge
self-awareness, reflectivity	no self-awareness whatsoever
Transfer of learning: found solutions can be applied to similar problems	able to only solve only one (or very few) tasks
permanently learning and adapting	learning (or programming) and application strictly separated
Science Fiction	State of the art technology



https://divis.io/en/2019/03/ai-for-laymen-part-2-symbolic-ai-neural-networks-and-deep-learning/#gref

What is ML?

A branch of **AI,** concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data.

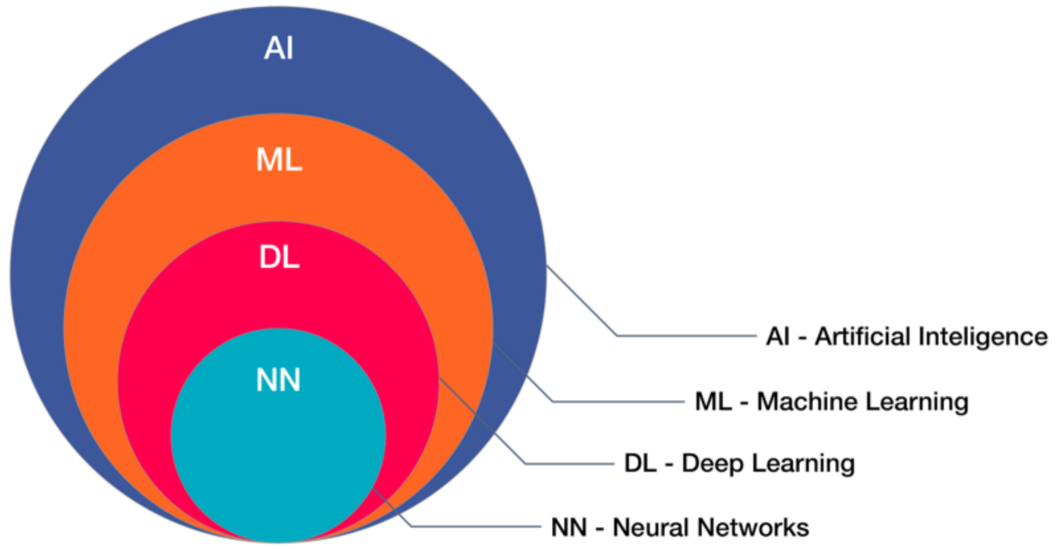
“**Machine Learning** is the field of study that gives computers the ability to learn without being explicitly programmed.”
[Arthur Samuel (1959)]

Machine learning is about **learning** to do better in the future based on what was experienced in the past.

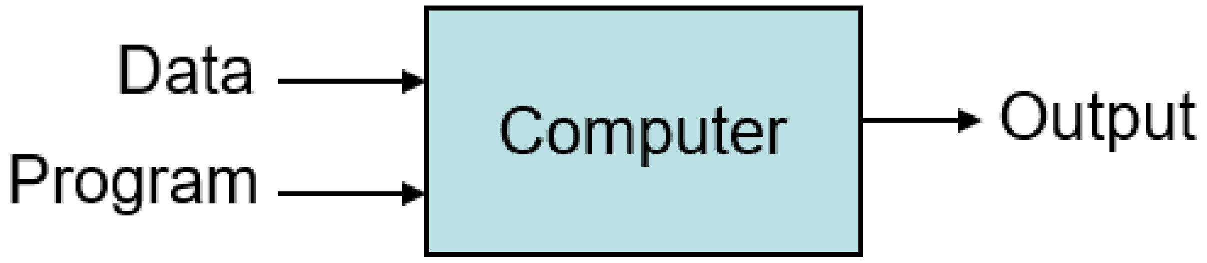
The **goal** is to devise learning algorithms that do the learning automatically without human intervention or assistance.

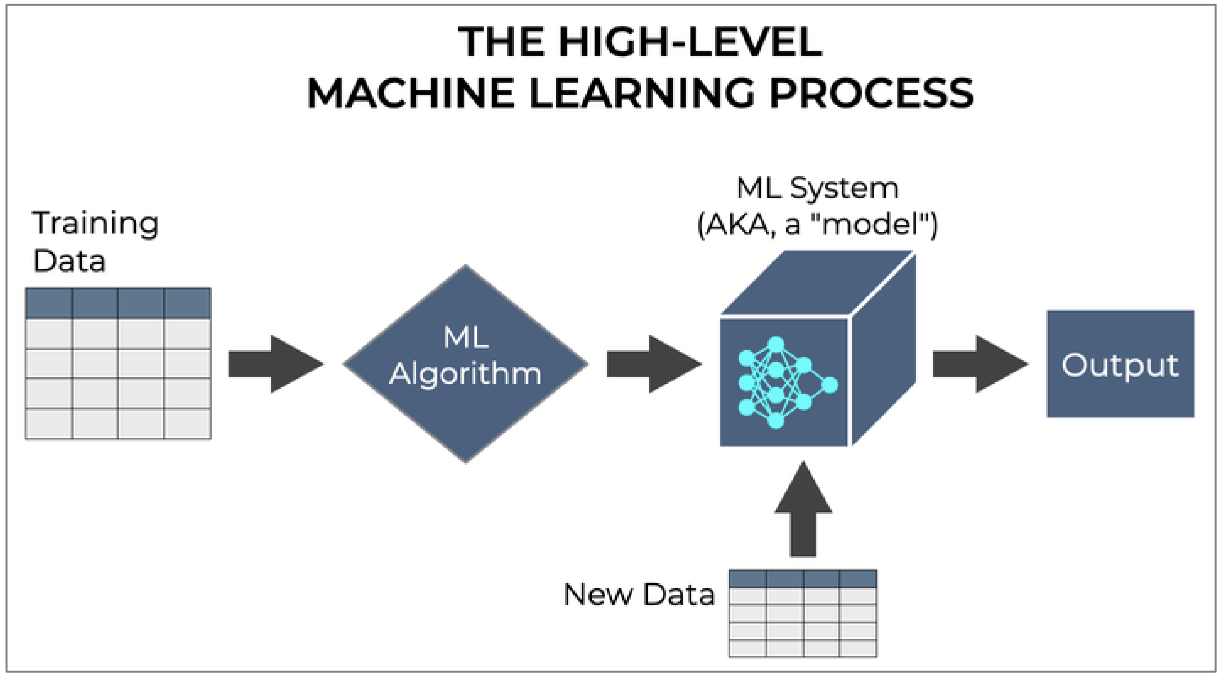
- Automating automation.
- Getting computers to program themselves.
- Let the data do the work instead of people!

Optimize a performance criterion using example data or past experience.

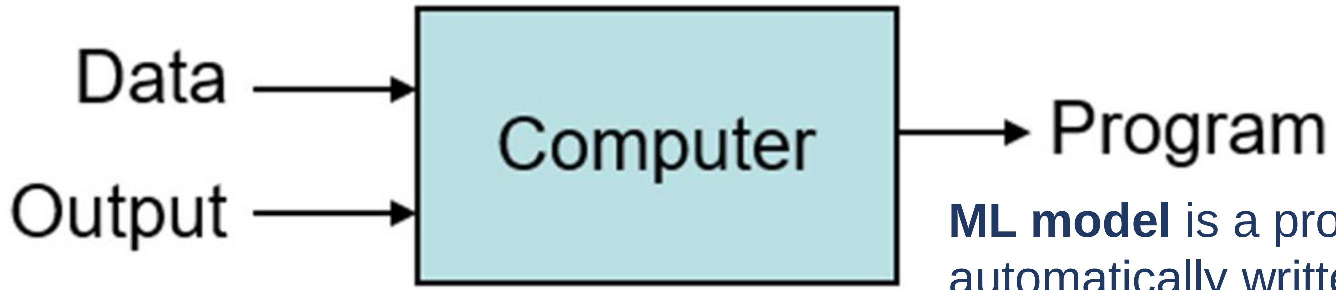


Traditional Programming





Machine Learning

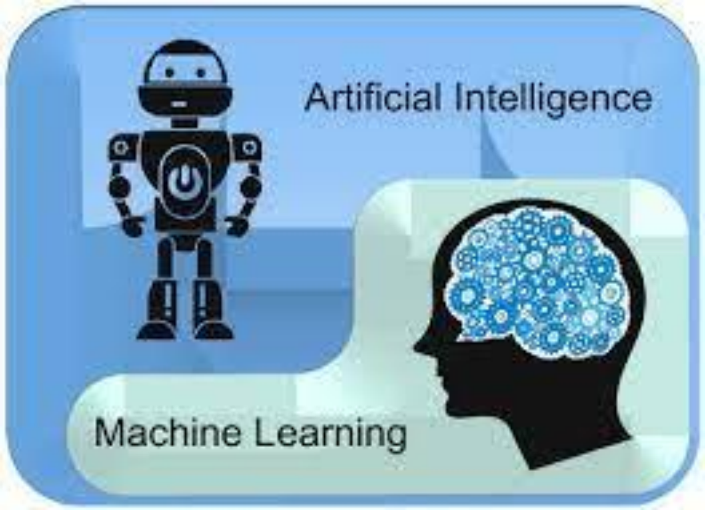


ML model is a program automatically written or created or learned by the **machine learning algorithm** to solve our problem.

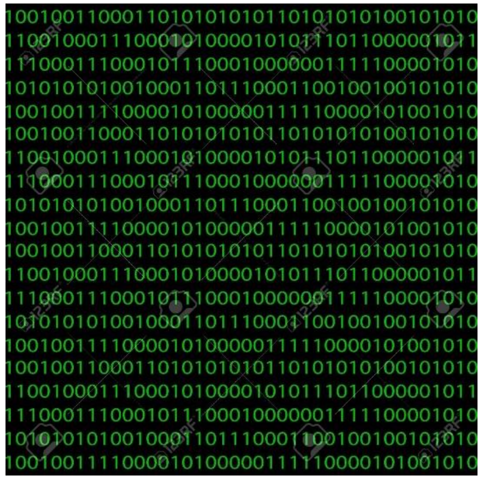
https://machinelearningmastery.com/difference-between-algorithm-and-model-in-machine-learning/

Learn from experiences

Human teach machine to learn by performing repetitive tasks with high accuracy. How?

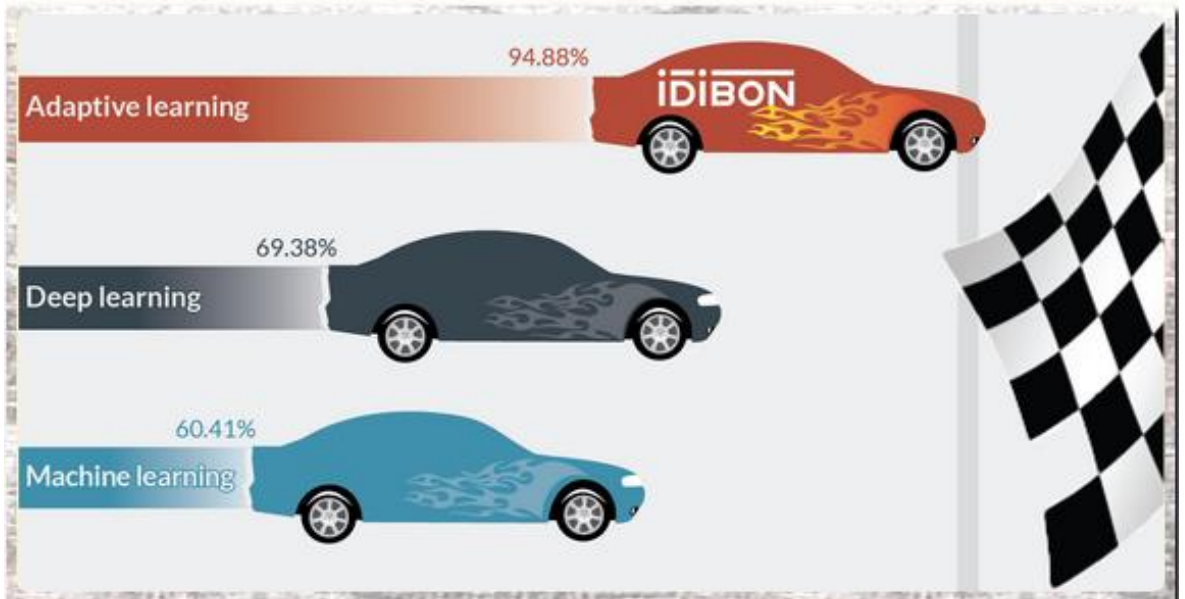


Learn from ~~experiences~~ data

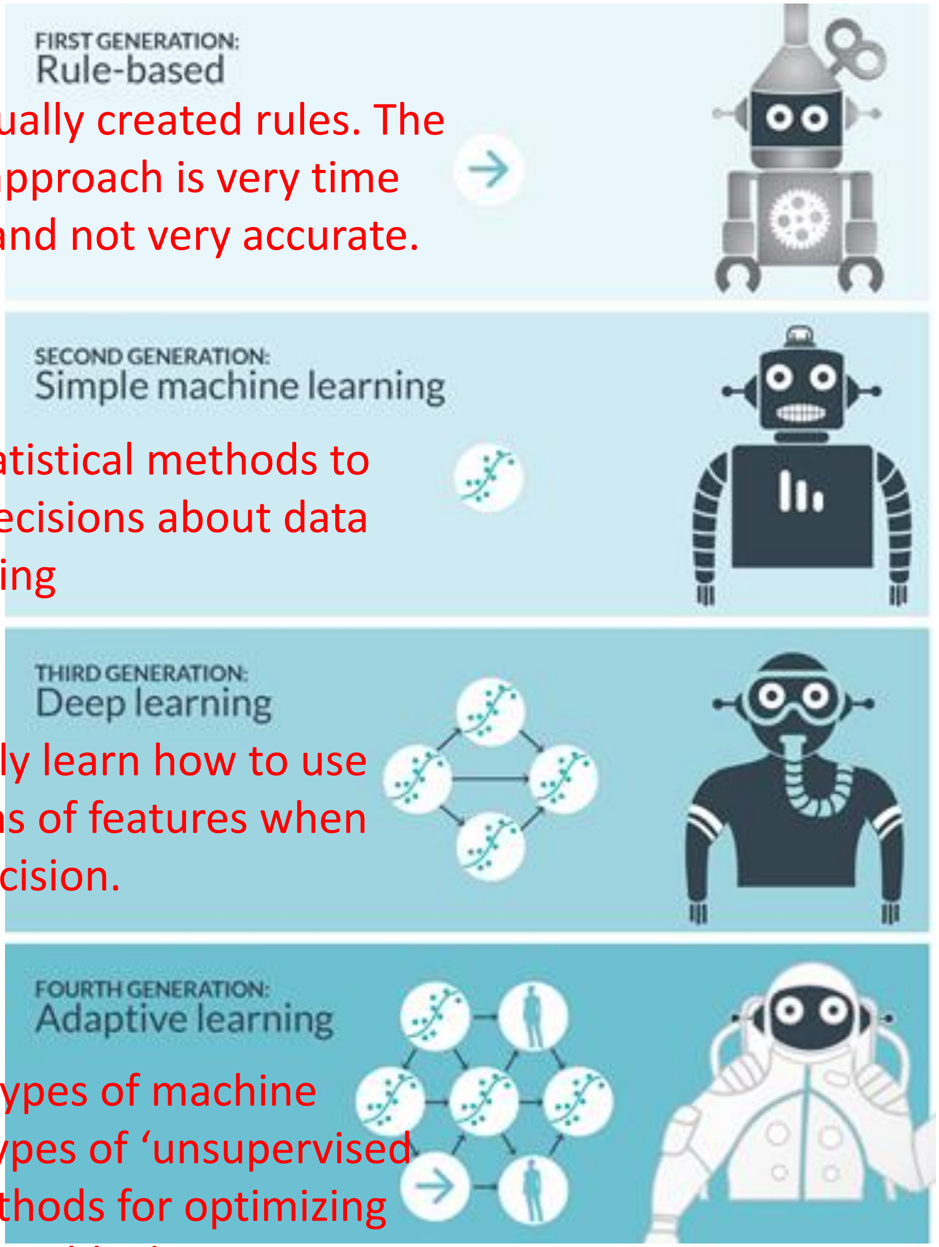

Follow instructions given by human

Four generations of machine intelligence

RESULTS: Compared to deep learning and simple machine learning systems from some of the most prominent machine intelligence companies in text analytics.



Accuracy in sentiment analysis about purchasing cars: Idibon and competitor systems



People manually created rules. The rule-based approach is very time consuming and not very accurate.

Uses statistical methods to make decisions about data processing

Automatically learn how to use combinations of features when making a decision.

Combine the three other types of machine intelligence, adding new types of ‘unsupervised machine learning’ and methods for optimizing the input from multiple, possibly disagreeing, humans.

FOR FURTHER INFORMATION, REFER TO SOURCE:
http://www.junglelightspeed.com/the-fourth-generation-of-machine-learning-adaptive-learning/

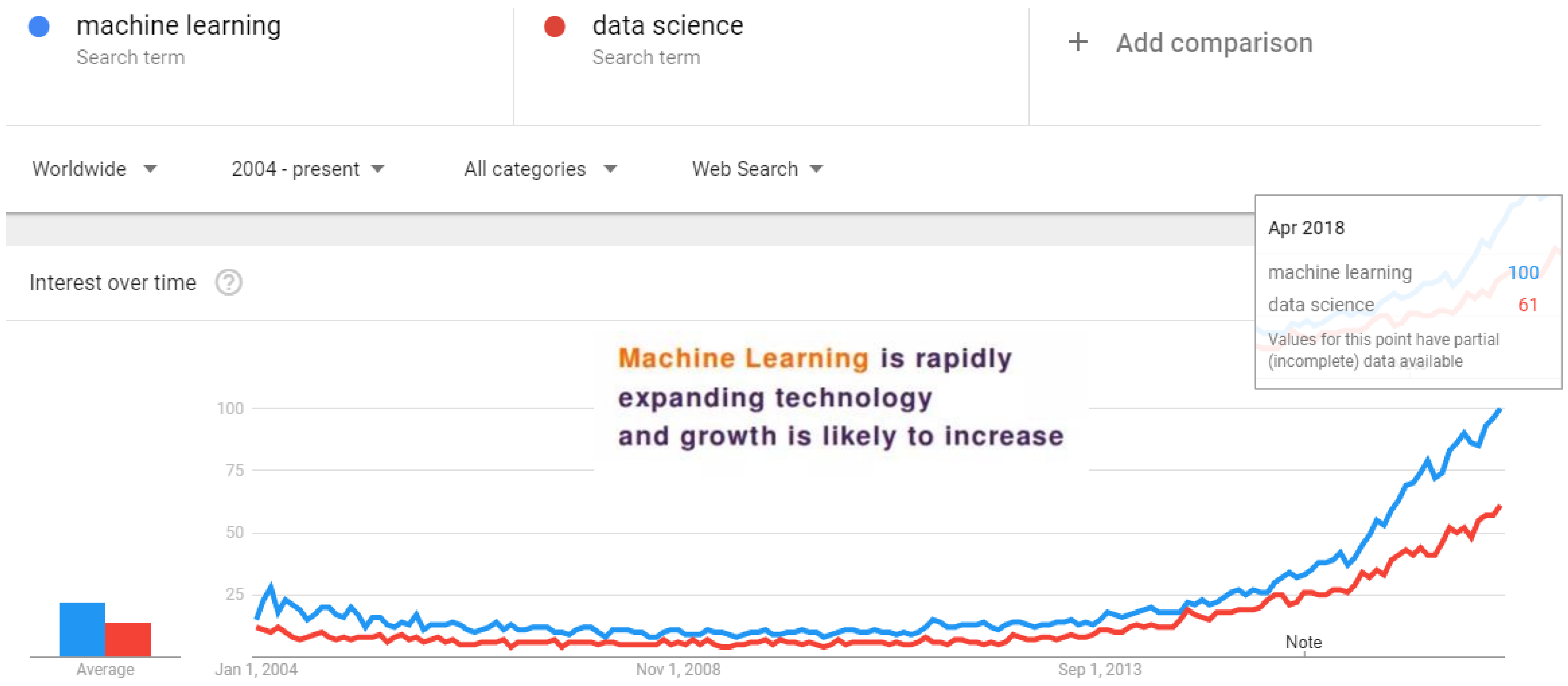
Why is machine learning popular right now?

1. **Computational speed** of technology is rapidly advancing. *GPUs* (graphical processing units) have allowed computations to be parallelized - more calculations can be done together at the same time instead of one after the other.

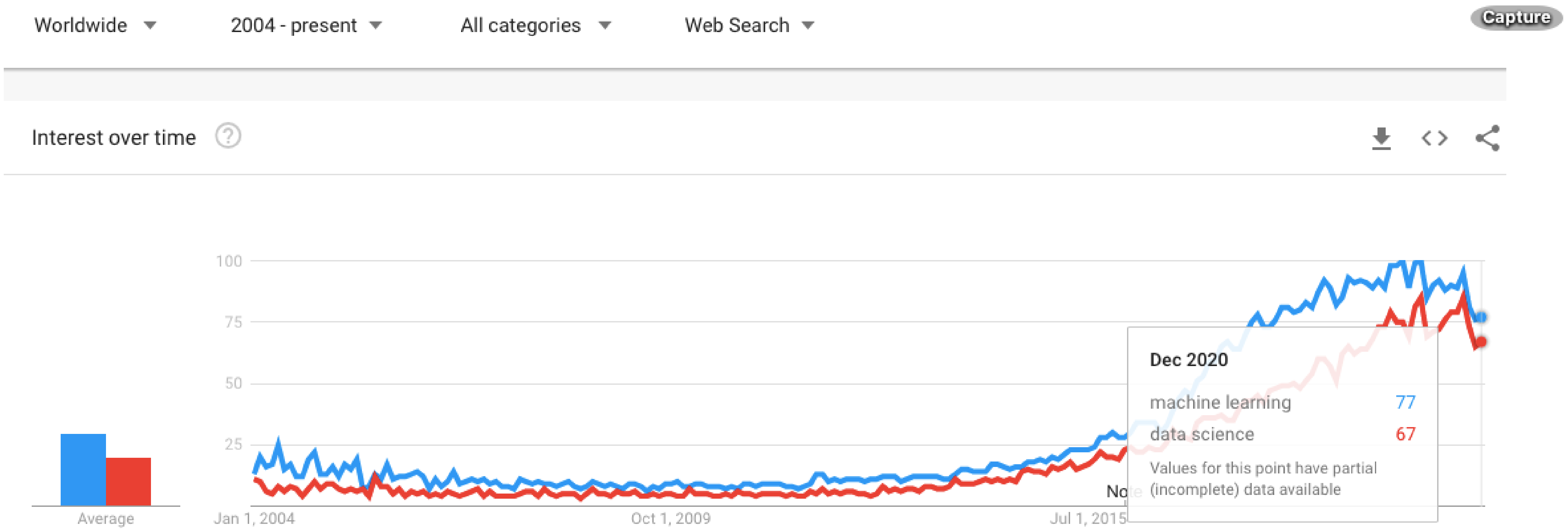
2. There have been significant advances in **algorithms**. The deep learning frameworks or architectures have improved from the likes of Google, Facebook, the research community, and emerging individuals in the open-source communities.

3. The exponential increase in **data** available within industries, the web, and businesses.

From Google Trends



Machine Learning is rapidly expanding technology and growth is likely to increase



Dec 2020

machine learning 77

data science 67

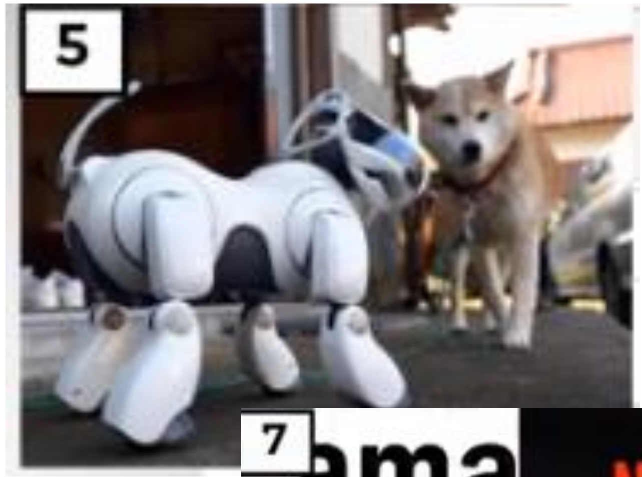
Values for this point have partial (incomplete) data available

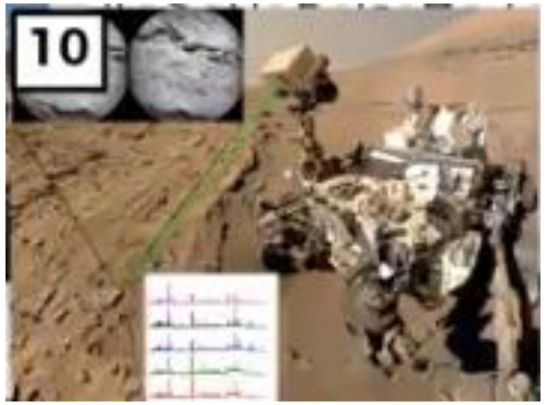
Examples of ML Applications

Here are a few widely publicized examples of machine learning applications you may be familiar with:

- The heavily hyped, **self-driving Google car**? The essence of machine learning.
- **Online recommendation** offers such as those from Amazon and Netflix? Machine learning applications for everyday life.
- Knowing **what customers are saying** about you on Twitter? Machine learning combined with linguistic rule creation.
- **Fraud detection**? One of the more obvious, important uses in our world today.

Applications of ML





EXAMPLE: SPAM CLASSIFICATION

Perhaps the most canonical example of how machine learning is used in everyday life is the spam filter for your email.

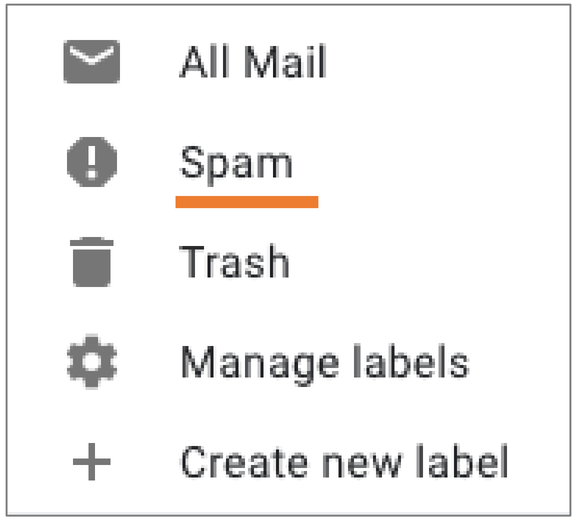
Almost all modern email clients, such as Google's Gmail, have a spam filter.

The spam filter automatically evaluates incoming email messages and attempts to categorize any "junk" mail as "spam," after which, it's sent to the spam folder so you don't have to see it.

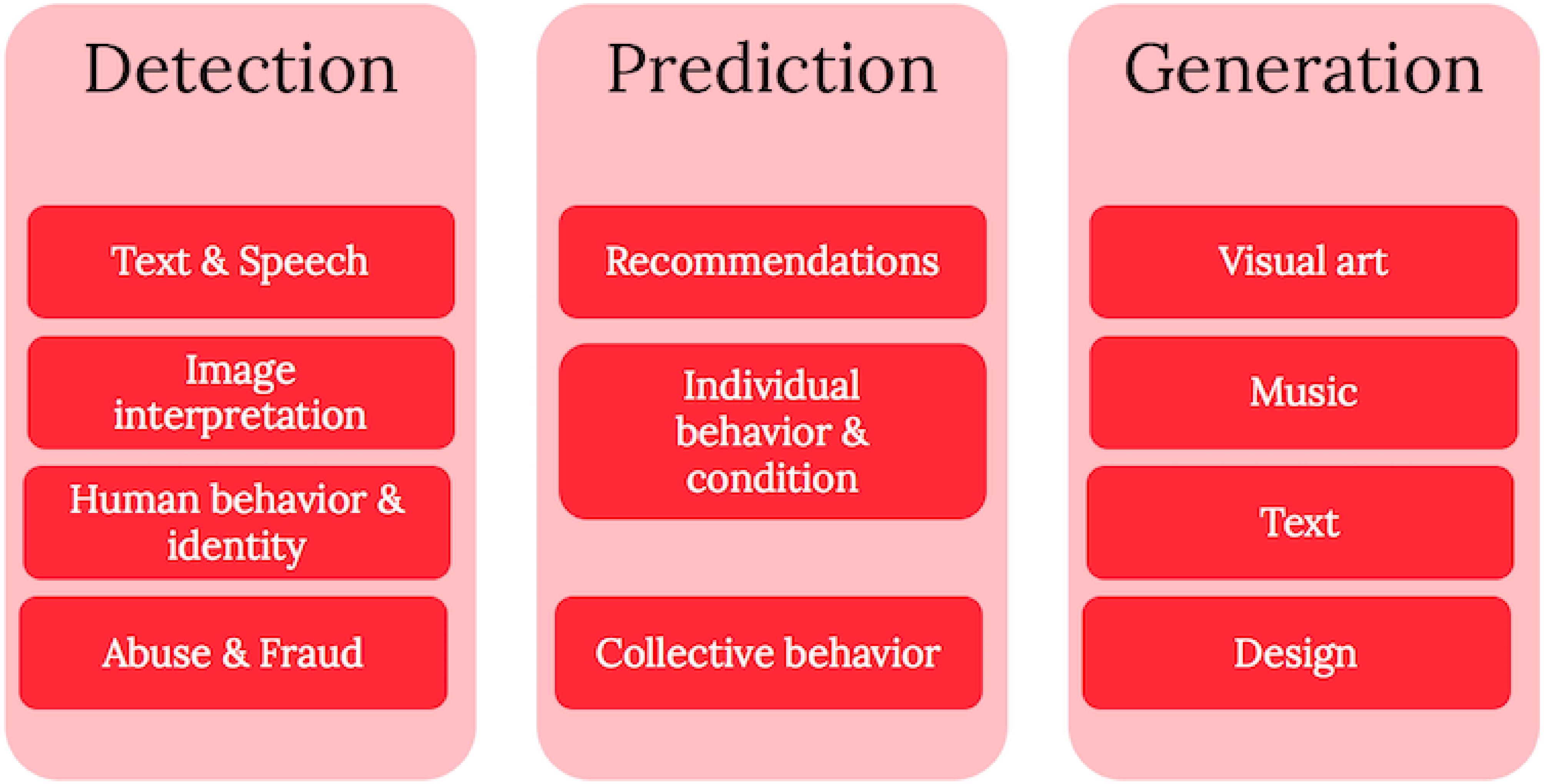
This too is built with machine learning.

The contents of an email – things like words, grammar, titles, senders – can all be considered forms of data. Email companies have used historical email data to "train" machine learning systems. These systems have "learned" to categorize and identify "spam" messages based on the email contents.

Now, after training these spam classification systems, email services can use them to classify new incoming messages so your inbox stays relatively free of junk email.



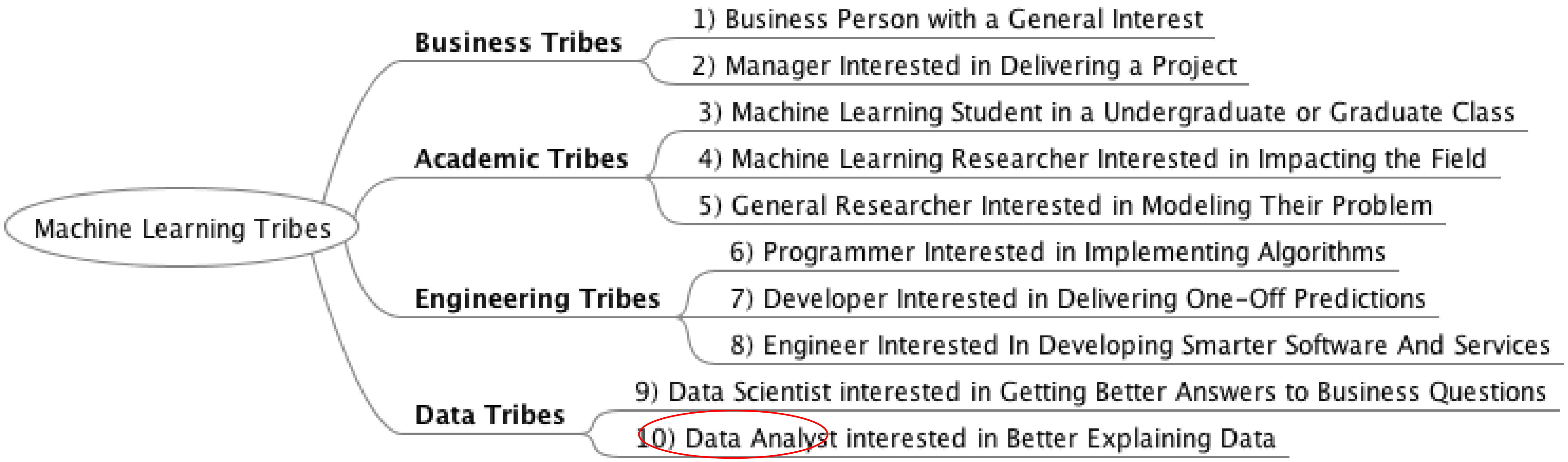
3 Main ML Categories - Applications & Common Use Cases



Detection	Prediction	Generation
Text & Speech	Recommendations	Visual art
Image interpretation	Individual behavior & condition	Music
Human behavior & identity		Text
Abuse & Fraud	Collective behavior	Design

https://sc5.io/posts/three-main-uses-machine-learning/#gref

ML Tribes


Machine Learning Tribes
Business Tribes
1) Business Person with a General Interest
2) Manager Interested in Delivering a Project
Academic Tribes
3) Machine Learning Student in a Undergraduate or Graduate Class
4) Machine Learning Researcher Interested in Impacting the Field
5) General Researcher Interested in Modeling Their Problem
Engineering Tribes
6) Programmer Interested in Implementing Algorithms
7) Developer Interested in Delivering One-Off Predictions
8) Engineer Interested In Developing Smarter Software And Services
Data Tribes
9) Data Scientist interested in Getting Better Answers to Business Questions
10) Data Analyst interested in Better Explaining Data

Source: https://machinelearningmastery.com/machine-learning-tribe/

Can ML Solve All Problems?

ML is **NOT** a solution for every type of problem. There are certain cases where robust solutions can be developed without using ML techniques.

- ML is not needed if you can determine a target value by using simple rules, computations, or predetermined steps that can be programmed without needing any data-driven learning.

Use machine learning for the following situations:

✓ **You cannot code the rules:** Many human tasks (such as recognizing whether an email is spam or not spam) cannot be adequately solved using a simple (deterministic), rule-based solution. A large number of factors could influence the answer. **When rules depend on too many factors** and many of these rules overlap or need to be tuned very finely, it soon becomes **difficult for a human to accurately code the rules**. You can use ML to effectively solve this problem.

✓ **You cannot scale:** You might be able to manually recognize a few hundred emails and decide whether they are spam or not. However, this task becomes tedious for millions of emails. ML solutions are effective at **handling large-scale problems**.

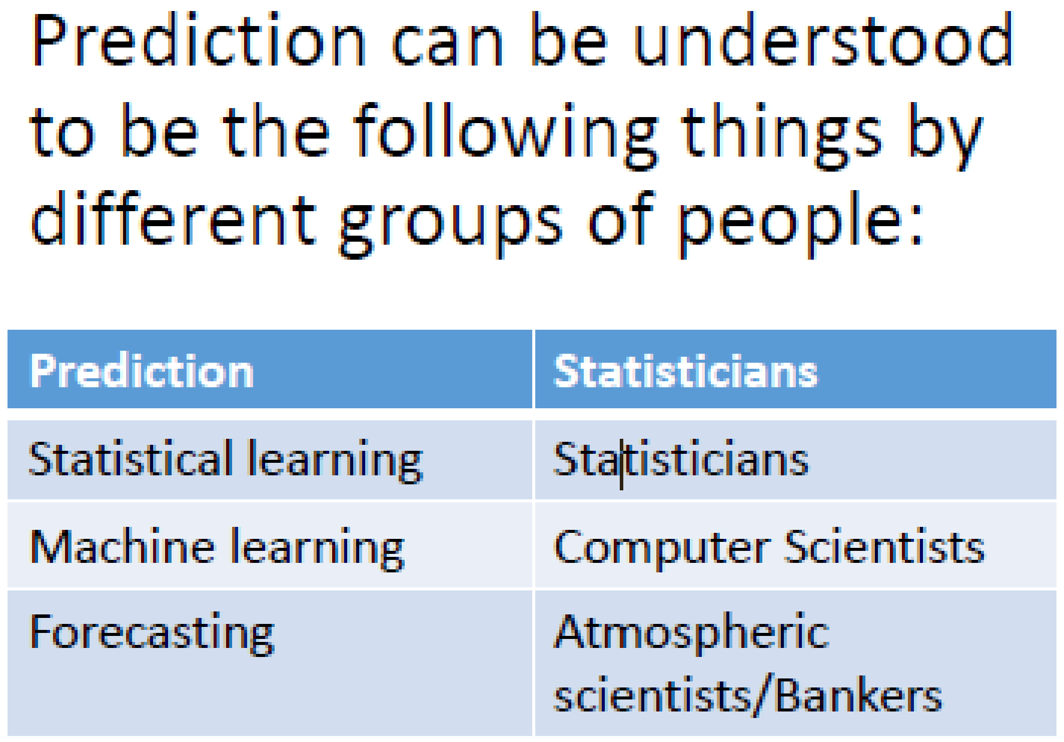
Solve problems where rules can’t be written by hand

Roles in ML?

Role of **Statistics**: Inference from a sample

Role of **Computer science**: Efficient algorithms to

- Solve the optimization problem.
- Representing and evaluating the model for inference.



Prediction can be understood to be the following things by different groups of people:

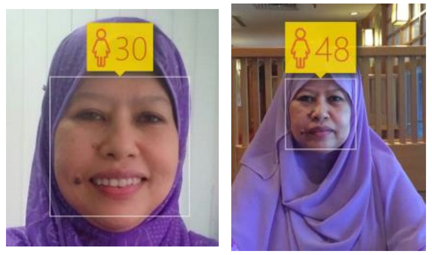
Prediction	Statisticians
Statistical learning	Statisticians
Machine learning	Computer Scientists
Forecasting	Atmospheric scientists/Bankers

Goal of ML → A **Good Model** [not the one that gives perfect predictions on the known data or training data but the one which gives **good (almost accurate) predictions** on the new data and avoids overfitting and underfitting]

5 Qs ML Help Answer

- Is this A or B?

Classification algorithm



- Is this weird?

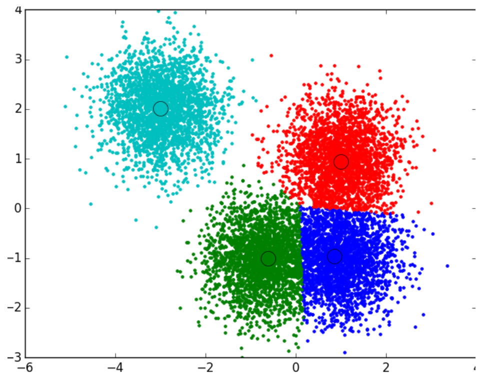
Anomaly detection algorithms

- How much / many?

Regression algorithms

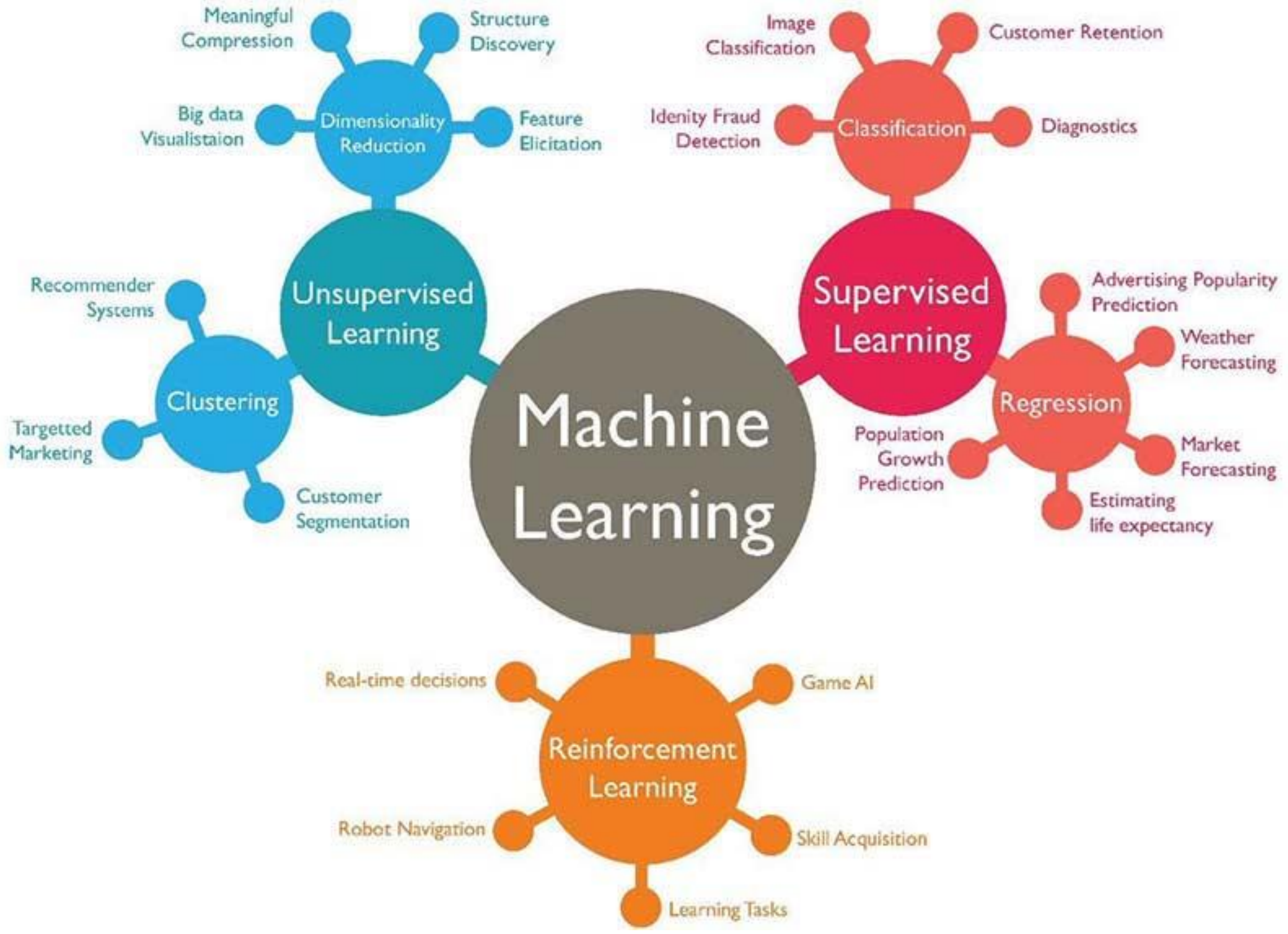
- How is this organized?

Clustering algorithms

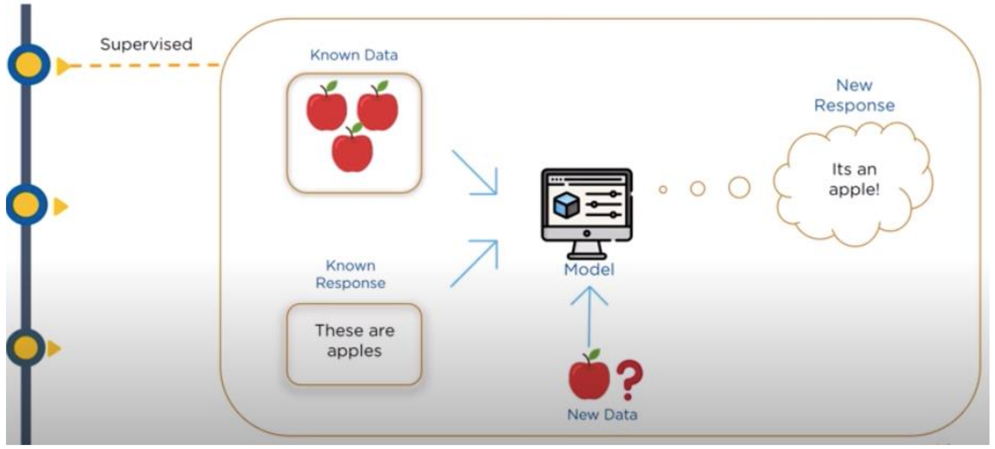


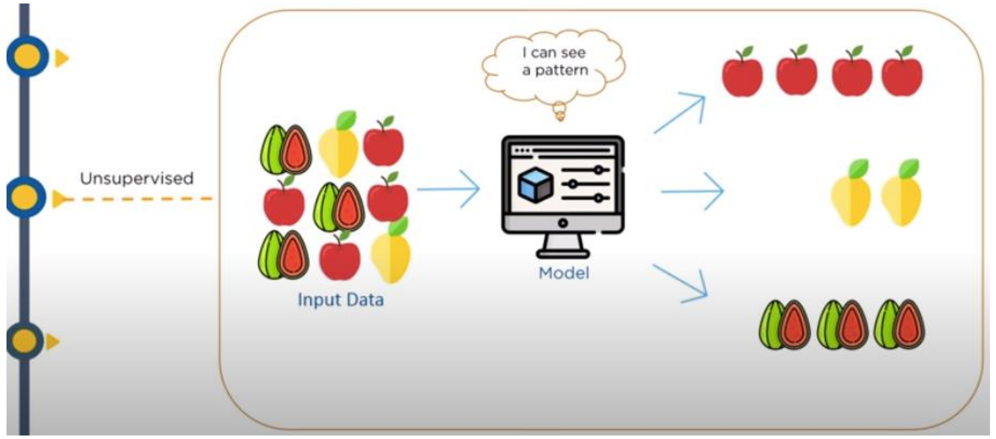
- What should I do now?

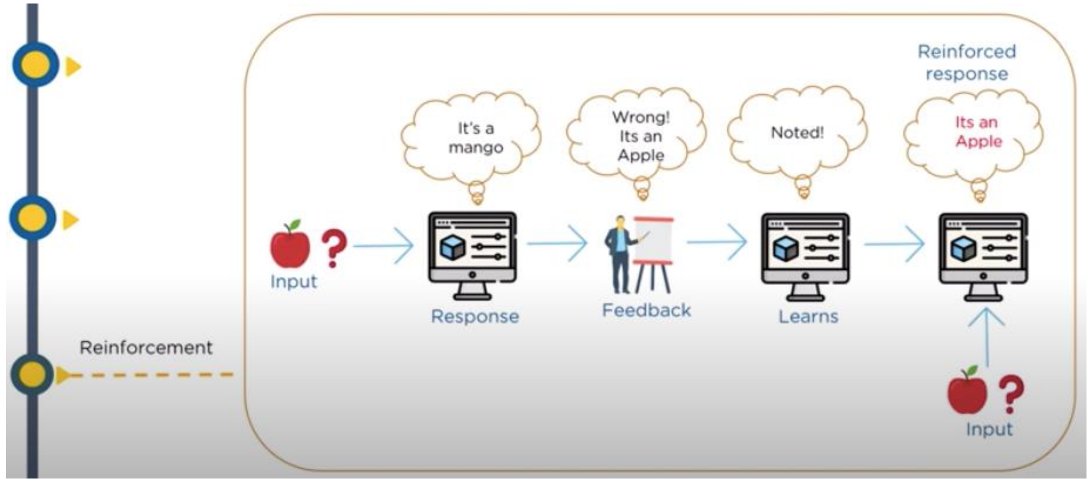
Reinforcement learning algorithms

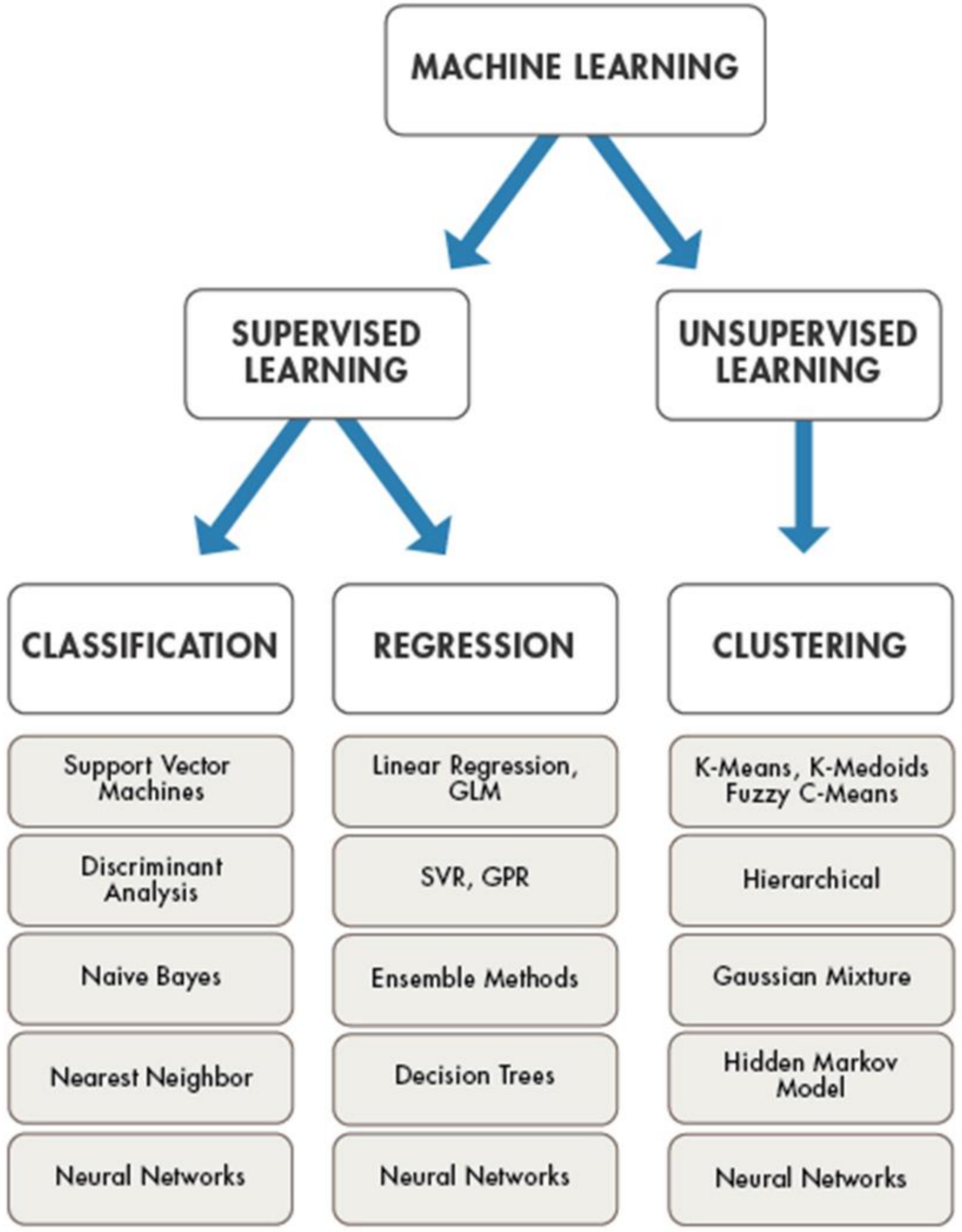


https://www.techleer.com/articles/203-machine-learning-algorithm-backbone-of-emerging-technologies/

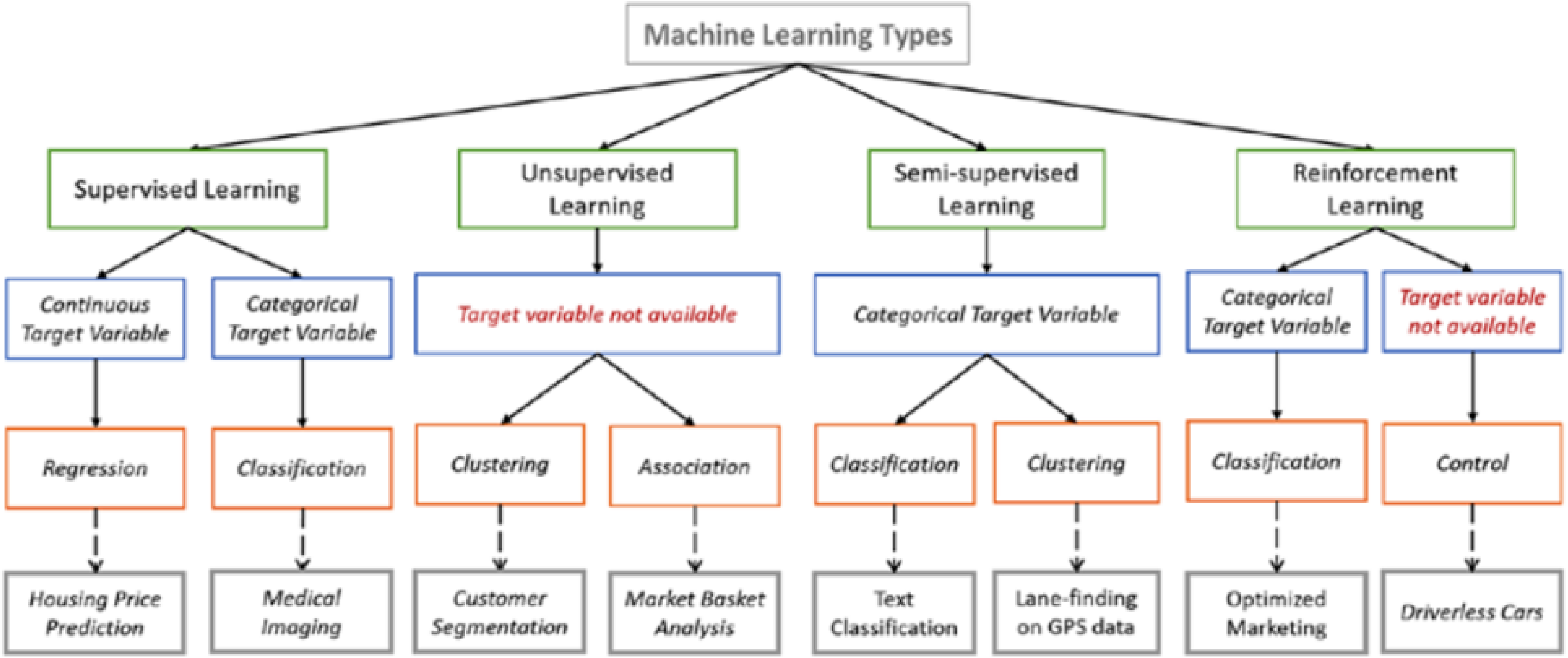








https://www.youtube.com/watch?v=G7fPB4OHkys


Machine Learning Types
Supervised Learning
Unsupervised Learning
Semi-supervised Learning
Reinforcement Learning
Continuous Target Variable
Categorical Target Variable
Target variable not available
Categorical Target Variable
Categorical Target Variable
Target variable not available
Regression
Classification
Clustering
Association
Classification
Clustering
Classification
Control
Housing Price Prediction
Medical Imaging
Customer Segmentation
Market Basket Analysis
Text Classification
Lane-finding on GPS data
Optimized Marketing
Driverless Cars

What is ML Algorithm?

- ML algorithms are **programs** (math and logic) that adjust themselves to perform better as they are exposed to more data.
- The "**learning**" part of machine learning means that those programs change how they process data over time, much as humans change how they process data by learning.
- So a ML algorithm is a program with a specific way to adjusting its own **parameters**, given **feedback** on its previous performance in making **predictions** about a dataset.
- ML uses algorithms (the ML engine) to turn a data set into a model.

$$Y = \text{function } (X)$$

Function is **Algorithm** in ML, where Y is **Predicted Output** for **Input** X

Components of ML Algorithm

- Tens of thousands of machine learning algorithms.
- Hundreds new algorithms every year.
- Every ML algorithm has **THREE (3)** components:
    - i. **Representation** **-** data structure or language / grammar that represents the model that is built through an ML algorithm.
    - ii. **Evaluation** **-** refers to a specific performance metric, the most basic being accuracy.
    - iii. **Optimization** **-** computational method used to fit the representation to the observed data.
- All ML algorithms are combinations of these three components.
- A framework for understanding all algorithms.

ML Algorithms = Representation + Evaluation + Optimization

A Few Useful Things to Know About Machine Learning
https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

Representation - How to represent knowledge

(What model you choose to represent the data?)

Representation: The model landscape

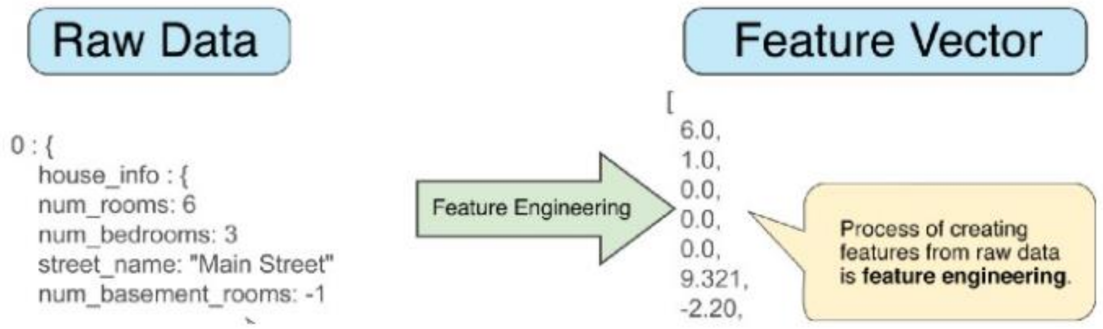
Representation is basically the space of allowed models (the hypothesis space), but also takes into account the fact that we are expressing models in some formal language.

Think what are the variables, how they are structured in space, do they take discrete or continues values, are they nodes in a directed graph or elements of vectors

Examples include:

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles

Feature Engineering: is a representation problem that turns the inputs *x* into 'things' the algorithm can understand.


Raw Data
Feature Vector
Feature Engineering
Process of creating features from raw data is **feature engineering**.

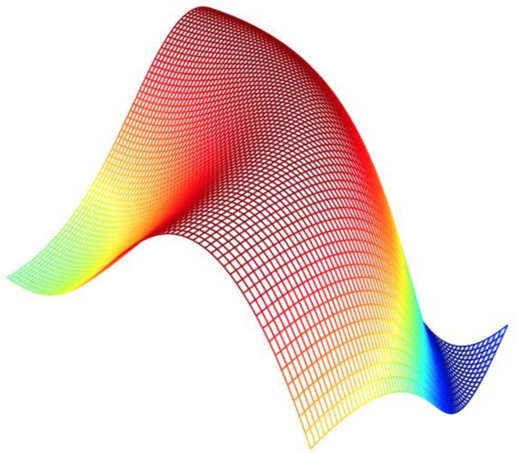
```
0 : {
  house_info : {
  num_rooms: 6
  num_bedrooms: 3
  street_name: "Main Street"
  num_basement_rooms: -1
```

```
[
  6.0,
  1.0,
  0.0,
  0.0,
  0.0,
  9.321,
  -2.20,
```

Feature engineering, adapted from Machine Learning Crash course.

https://annisap.medium.com/learning-representation-evaluation-optimization-c73fce64281t

Evaluation



Evaluate the loss function given different values for the parameters

Evaluation is essentially how you judge or prefer one model vs. another; it's what you might have seen as a utility function, loss function, scoring function, or fitness function in other contexts.

The **loss function** is also called a utility, scoring, or a reward function. *An objective function is a loss function that desired to be **maximized**.*

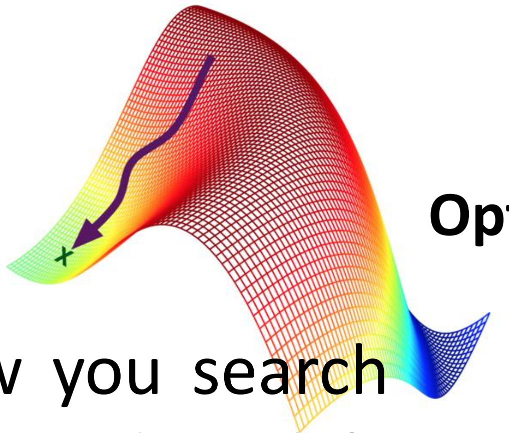
The evaluation function of the external model is typically called a **performance metric**.

Evaluation: Preferences over the landscape

Examples include:

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence

Optimization



Optimization: Strategy for fulfilling preferences

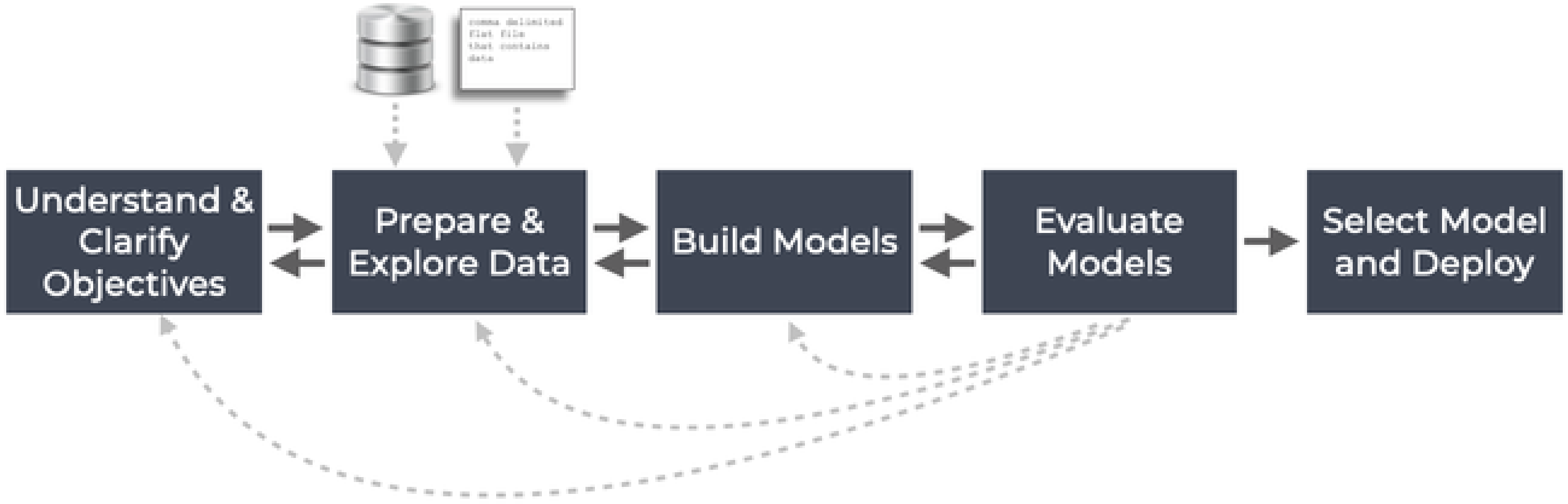
Optimization is how you search the space of represented models to obtain better evaluations.
This is the way you expect to traverse the landscape to find the promised land of ideal models; the strategy of getting to where you want to go.

Examples include:

- Combinatorial optimization
  - E.g.: Greedy search
- Convex optimization
  - E.g.: Gradient descent
- Constrained optimization
  - E.g.: Linear programming

In practice, defining the evaluation function is often the first step of this process, before selecting a representation or optimization.

THE MODEL BUILDING PROCESS


comma delimited flat file that contains data
Understand & Clarify Objectives
Prepare & Explore Data
Build Models
Evaluate Models
Select Model and Deploy

It simplifies things quite a bit, but it gives you a rough idea of what happens when you build a machine learning system.

CLARIFY OBJECTIVES

Start by clarifying the objectives of the machine learning system.
In a business setting, this frequently involves talking to team members and business partners to generate system requirements and outcomes.

GET AND CLEAN DATA

Then get the data and clean the data. Data wrangling is 80% of the job in data science.
It's true that data preparation and exploration is a big part of the task when creating a new machine learning model.

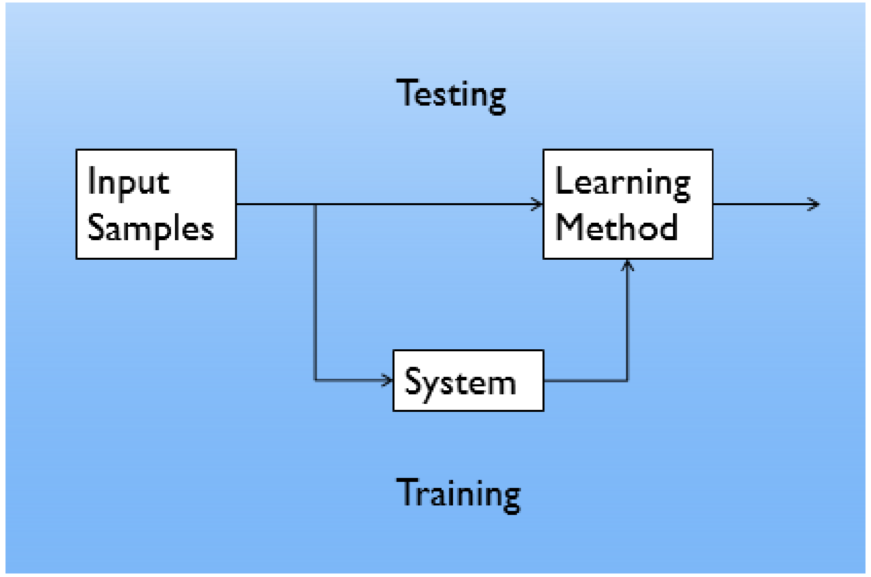
BUILD MODELS

Need to build several different models.
This means building models that are very similar but with slight differences that change or modify the performance.
Can also mean using multiple different machine learning techniques to build different types of machine learning systems.
For example, for a project, you might build a decision tree model, a support vector machine, and a logistic regression model, just to see how each different model type performs.

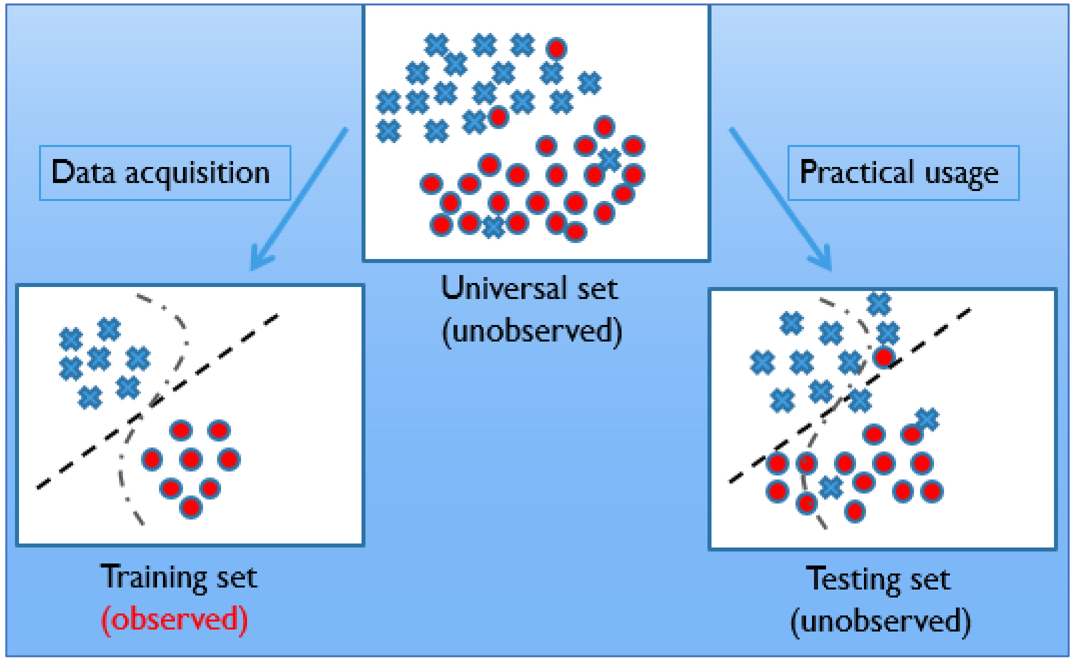
EVALUATE MODELS AND SELECT

Finally, once several models have been built, evaluate them, select the "best" model relative to the project requirements. Once one of the models have been selected, finalize it and then deploy the model.

Learning System Model



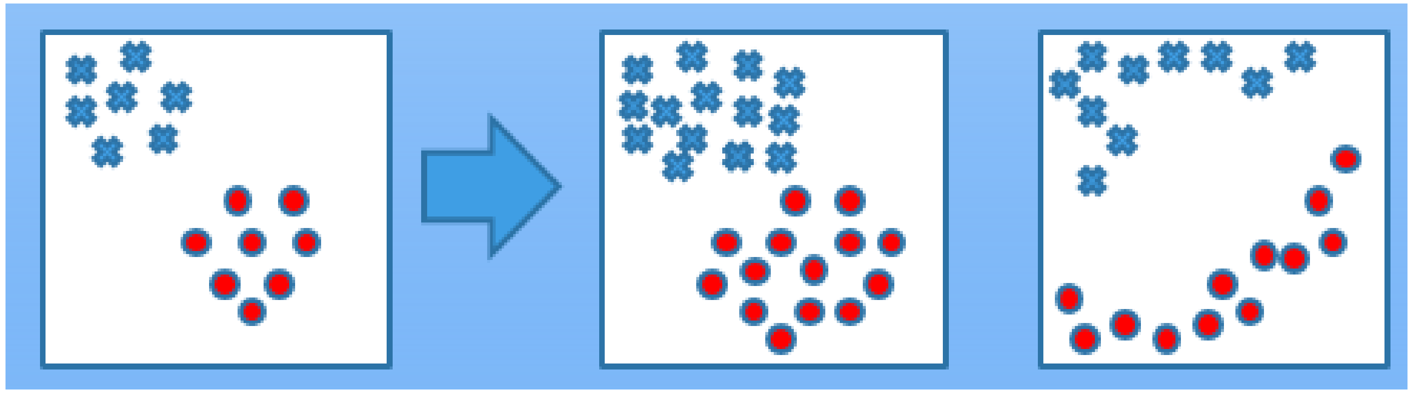
Training & Testing



- **Training** is the process of making the system able to learn.

No free lunch rule:

- **Training set** and **testing set** come from the same distribution.
- Need to make some assumptions or bias.



Inductive Learning

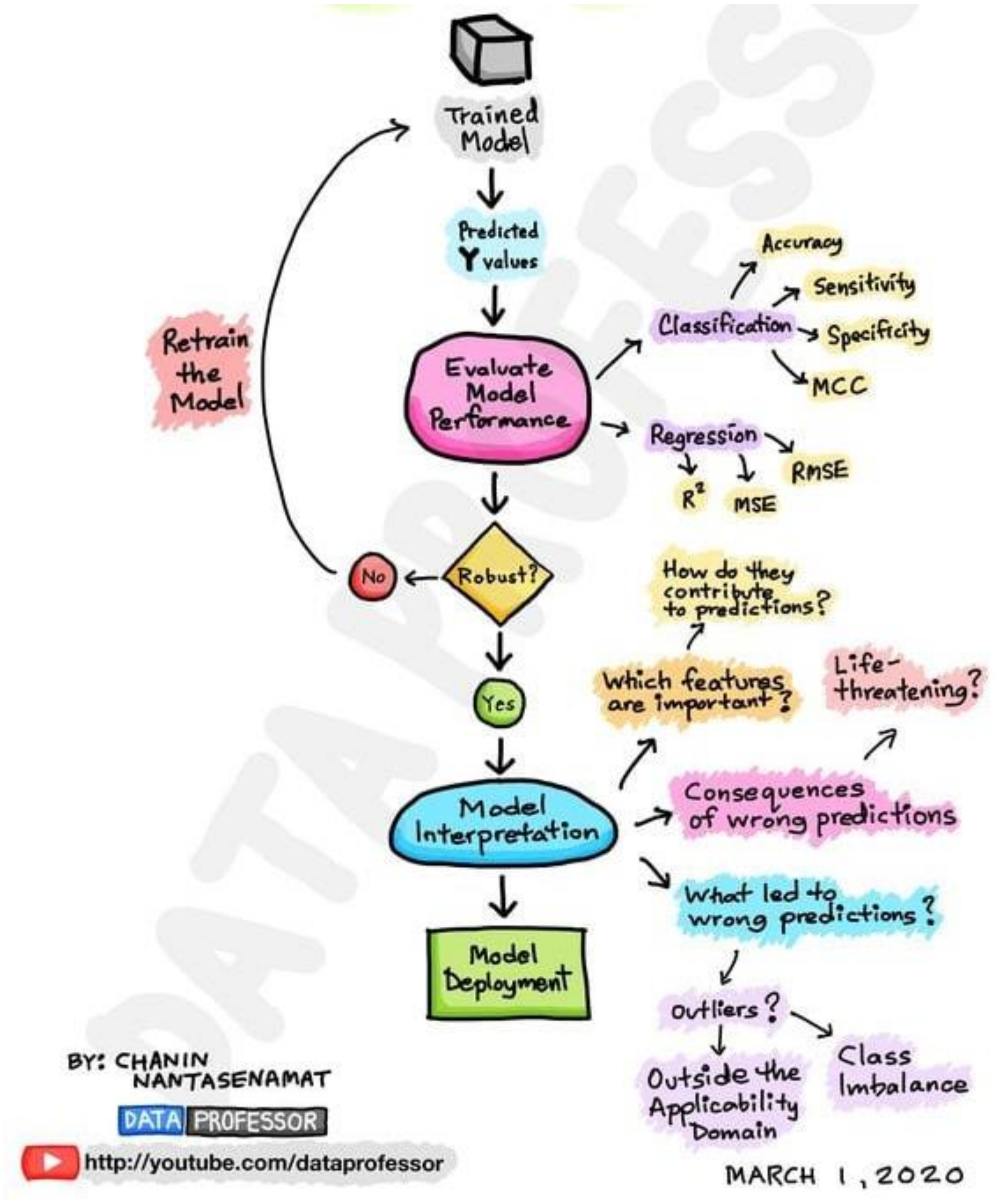
Given input samples (x) and output samples (f(x)) and the problem is to estimate the function (f).

The problem is to generalize from the samples and the mapping to be useful to estimate the output for new samples in the future.

Some practical examples of induction are:

- ❖**Credit risk assessment**
  - ✓The x is the properties of the customer.
  - ✓The f(x) is credit approved or not.
- ❖**Disease diagnosis**
  - ✓The x are the properties of the patient.
  - ✓The f(x) is the disease they suffer from.
- ❖**Face recognition**
  - ✓The x are bitmaps of peoples faces.
  - ✓The f(x) is to assign a name to the face.
- ❖**Automatic steering**
  - ✓The x are bitmap images from a camera in front of the car.
  - ✓The f(x) is the degree the steering wheel should be turned.

Model Interpretability



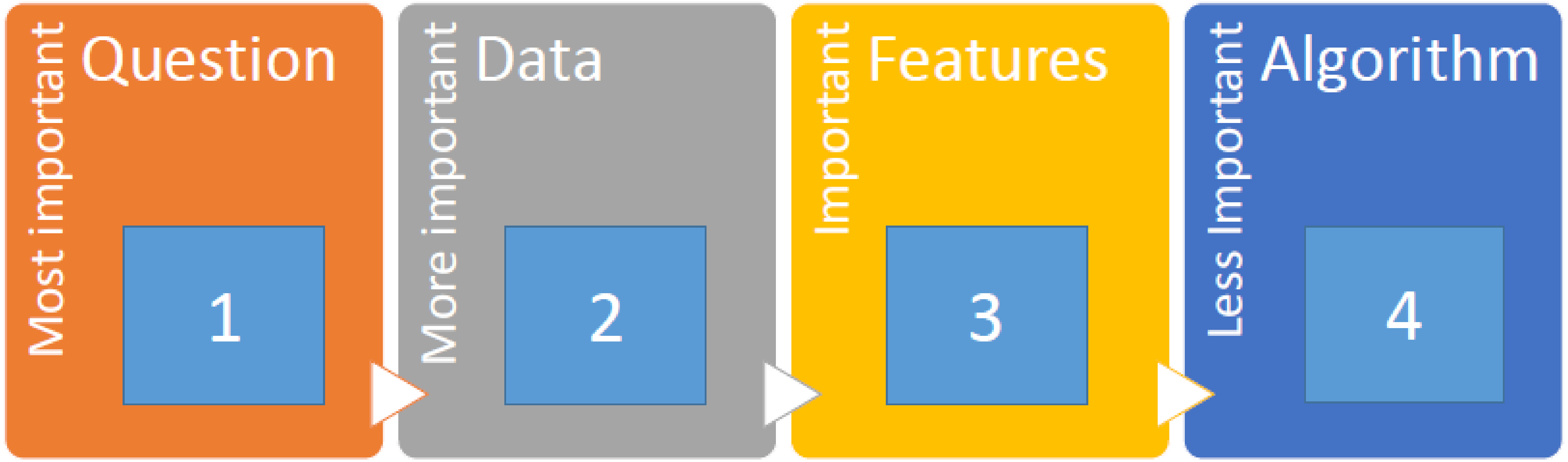
Question to Reflect

How would you write a program to distinguish a picture of yourself from a picture of someone else?

4 Problems where Inductive Learning Might be a Good Idea:

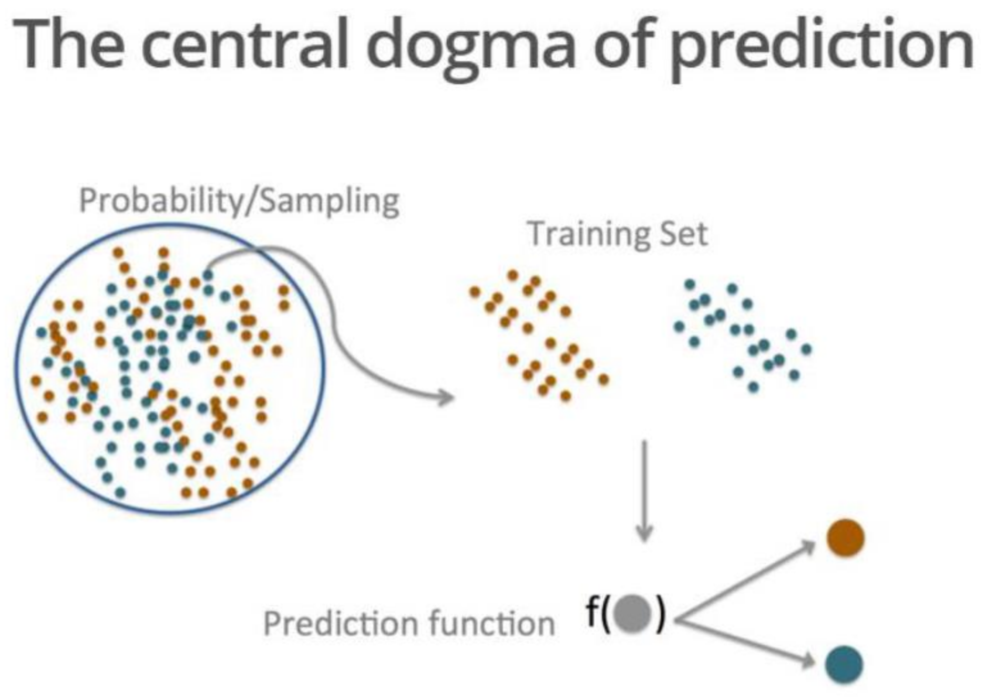
- Problems where there is **no human expert**. If people do not know the answer they cannot write a program to solve it. These are areas of true discovery.
- Humans can perform the task but **no one can describe how to do it**. There are problems where humans can do things that computer cannot do or do well. Examples include riding a bike or driving a car.
- Problems where the desired **function changes frequently**. Humans could describe it and they could write a program to do it, but the problem changes too often. It is not cost effective. Examples include the stock market.
- Problems where each user **needs a custom function**. It is not cost effective to write a custom program for each user. Example is recommendations of movies or books on Netflix or Amazon.

Which Components are More Important?

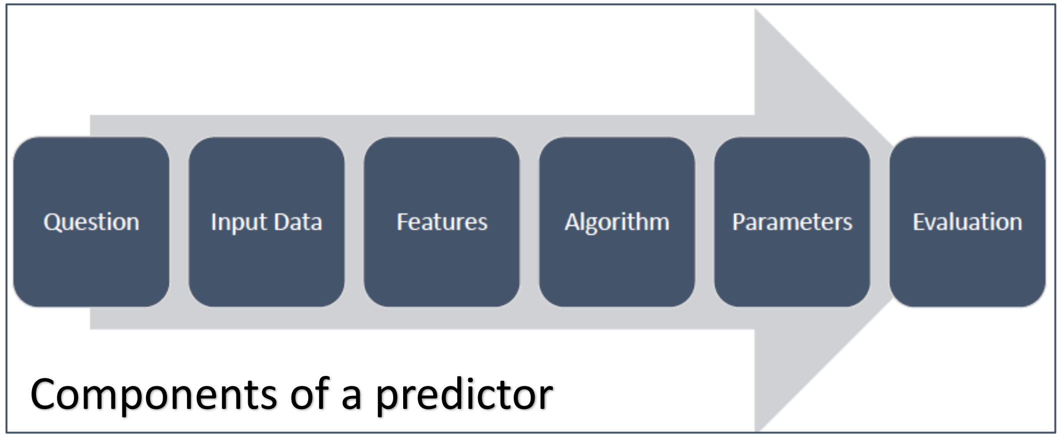


Applied Machine Learning Process

1. Define the Problem
2. Prepare Data
3. Spot Check Algorithms
4. Improve Results
5. Present Results



SPAM example



Components of a predictor

COMPONENTS	ELABORATION
Question	Can I use quantitative attributes of the email to classify the email as belonging to the SPAM or non-SPAM group?
Input data	Is the data perfect or sufficient? If not what do you do about it?
Features	In this context, can be a word count : in this example the count of the word ‘you’ is evaluated as a feature through.
Algorithm	Here, a kernel density function of the frequency of the word ‘your’ in the email is used as the prediction function.
Parameter	The cut-off point of 0.5 is used to differentiate spam emails from non-spam email.
Evaluation	The prediction accuracy is measured in terms of percentage of correctly predicted labels.

1. Define the Problem

Problem Definition Framework

Step 1: What is the problem?

Describe the problem informally and formally and list assumptions and similar problems.

Step 2: Why does the problem need to be solved?

List your motivation for solving the problem, the benefits a solution provides and how the solution will be used.

Step 3: How would I solve the problem?

Describe how the problem would be solved manually to flush domain knowledge.

Step 1.1: What is the Problem

Defining the problem.

- **Informal description** - Describe the problem as though you were describing it to a friend or colleague. It provides the basis for a one sentence description you can use to share your understanding of the problem.
  - For example: I need a program that will tell me which tweets will get retweets.
- **Formalism** - Tom Mitchell's (1998) machine learning formalism.
  - *A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**.*

  Use this formalism to define the T, P, and E for your problem.

  For example:

  - Task (T): Classify a tweet that has not been published as going to get retweets or not.
  - Experience (E): A corpus of tweets for an account where some have retweets and some do not.
  - Performance (P): Classification accuracy, the number of tweets predicted correctly out of all tweets considered as a percentage.

Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam.
Define the T, P, and E for this spam filter.

- ❑ Watching you label emails as spam or not spam.
- ❑ The number of emails correctly classified as spam/not spam.
- ❑ Classifying emails as spam or not spam.

Step 1.1: What is the Problem

Assumptions

- Create a **list of assumptions** about the problem and it’s phrasing.
    - May be **rules of thumb** and **domain specific information**
- It can be useful to highlight questions that can be tested against real data because breakthroughs and innovation occur when assumptions and best practice are demonstrated to be wrong in the face of real data.
- It can also be useful to highlight areas of the problem specification that may need to be challenged, relaxed or tightened.

For example:

- The specific words used in the tweet matter to the model.
- The specific user that retweets does not matter to the model.
- The number of retweets may matter to the model.
- Older tweets are less predictive than more recent tweets.

Step 1.2: Why does the problem need to be solved?

Motivation

- What need will be fulfilled when the problem is solved?
- As a learning exercise or as part of a duty at work.

Solution Benefits

- What capabilities does it enable?
- Be clear on the benefits of the problem being solved to ensure that you capitalize on them.

Solution Use

- How the solution to the problem will be used and what type of lifetime you expect the solution to have? As programmers we often think the work is done as soon as the program is written, but really the project is just beginning it’s maintenance lifetime.
- The way the solution will be used will influence the nature and requirements of the solution you adopt.

Step 1.3: How would I solve the problem?

- **List out step-by-step** what data to collect, how to prepare it and how to design a program to solve the problem.
  - May include prototypes and experiments needed to perform & highlight questions and uncertainties about the domain that could be explored.
- This is a powerful tool - can highlight problems that actually can be solved satisfactorily using a manually implemented solution.
  - Also flushes out important domain knowledge that has been trapped up until now like where the data is actually stored, what types of features would be useful and many other details.
- Collect all of these details as they occur and update the previous sections of the problem definition. Especially the assumptions and rules of thumb.

2. Prepare Data

The actual data preparation process is 3 step as follows:

Step 1: Data Selection: Consider what data is available, what data is missing and what data can be removed.

Step 2: Data Pre-processing: Organize your selected data by formatting, cleaning and sampling from it.

Step 3: Data Transformation: Transform pre-processed data ready for machine learning by engineering features using scaling, attribute decomposition and attribute aggregation.

More on Data Preparation → https://machinelearningmastery.com/how-to-prepare-data-for-machine-learning/

3. Spot Check Algorithms

Once you have **defined your problem** and **prepared your data** you need to **apply machine learning algorithms** to the data in order to solve your problem.

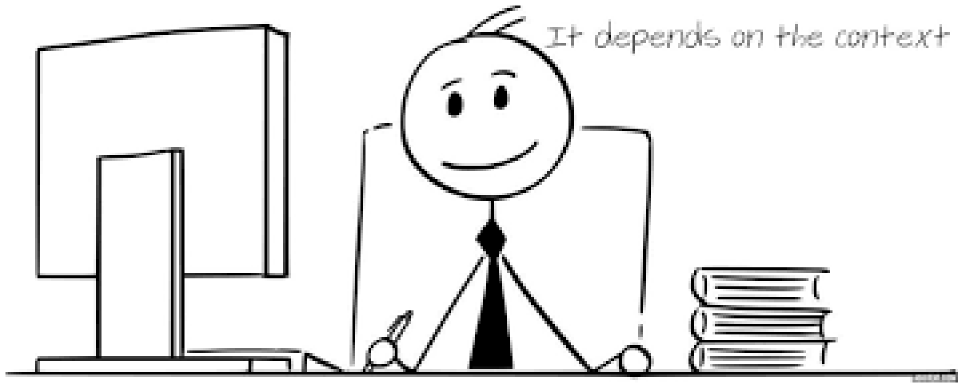
Spot-checking algorithms - a quick assessment of a bunch of different algorithms on your machine learning problem so that you know what algorithms to focus on and what to discard.

Test harness - the data you will train and test an algorithm against and the performance measure you will use to assess its performance. It is important to define your test harness well so that you can focus on evaluating different algorithms and thinking deeply about the problem.

Goal of test harness - to quickly and consistently test algorithms against a fair representation of the problem being solved.

- The outcome of testing multiple algorithms against the harness will be an estimation of how a variety of algorithms perform on the problem against a chosen performance measure.

Which ML Algorithm to Choose?



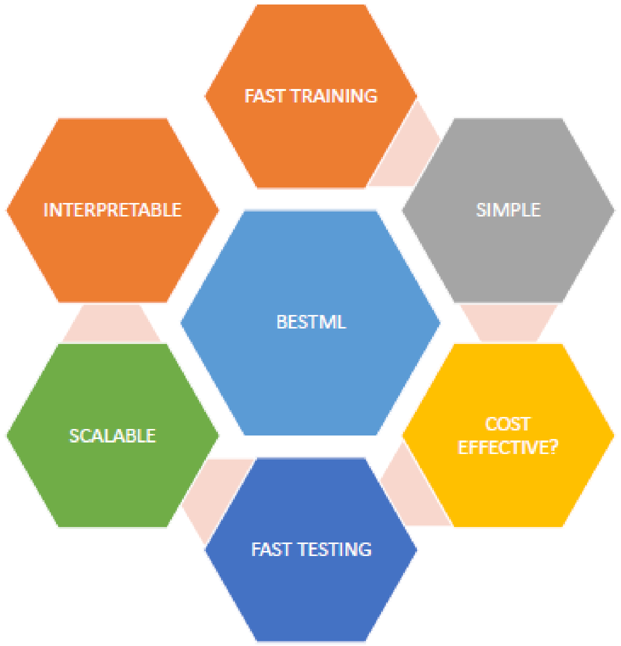
Which Algorithms to be used for the right ML solution?

The problem statement / What you want to do with the data

The size, quality and nature of data

The available computational time

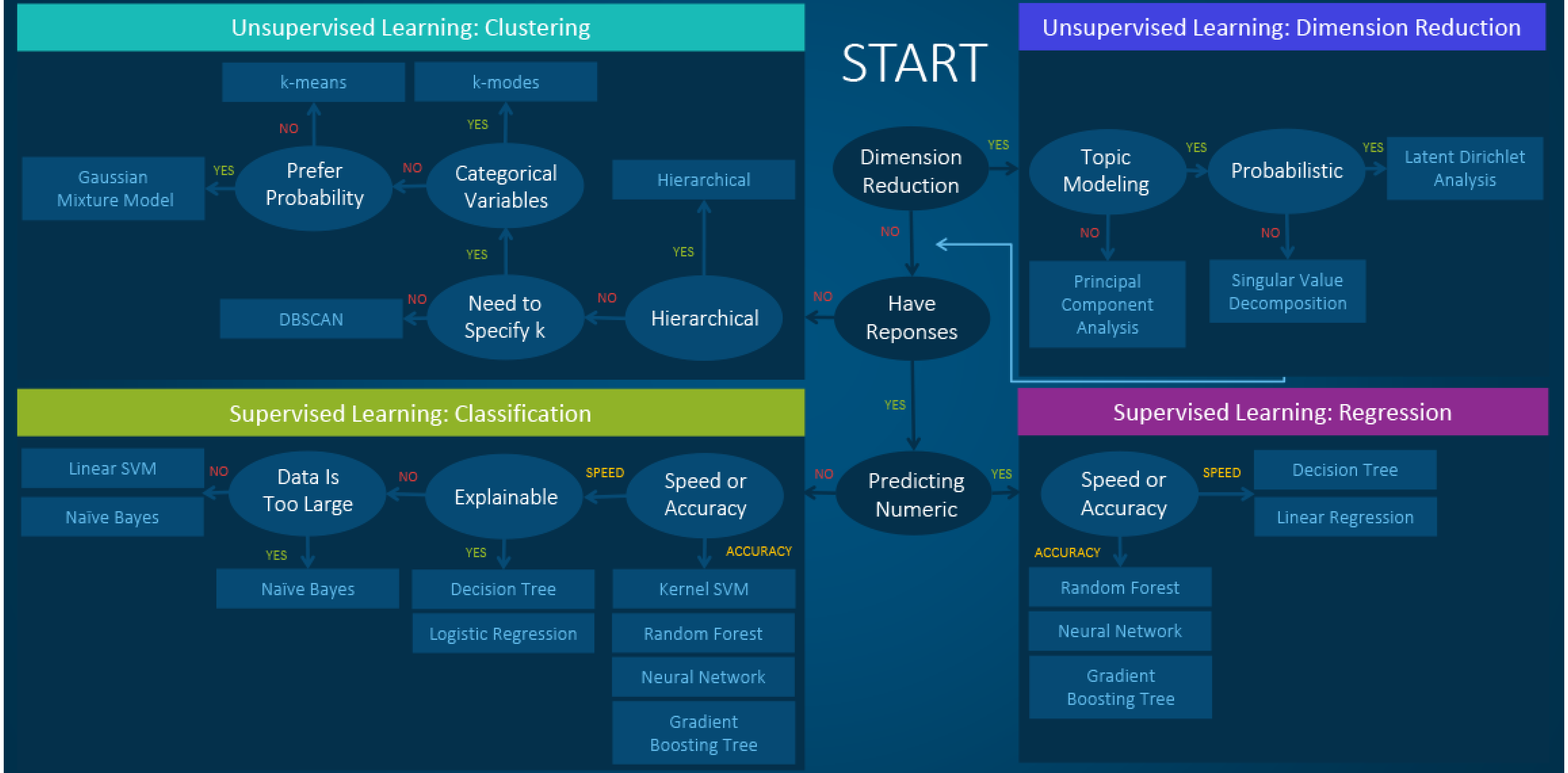
The urgency of the task

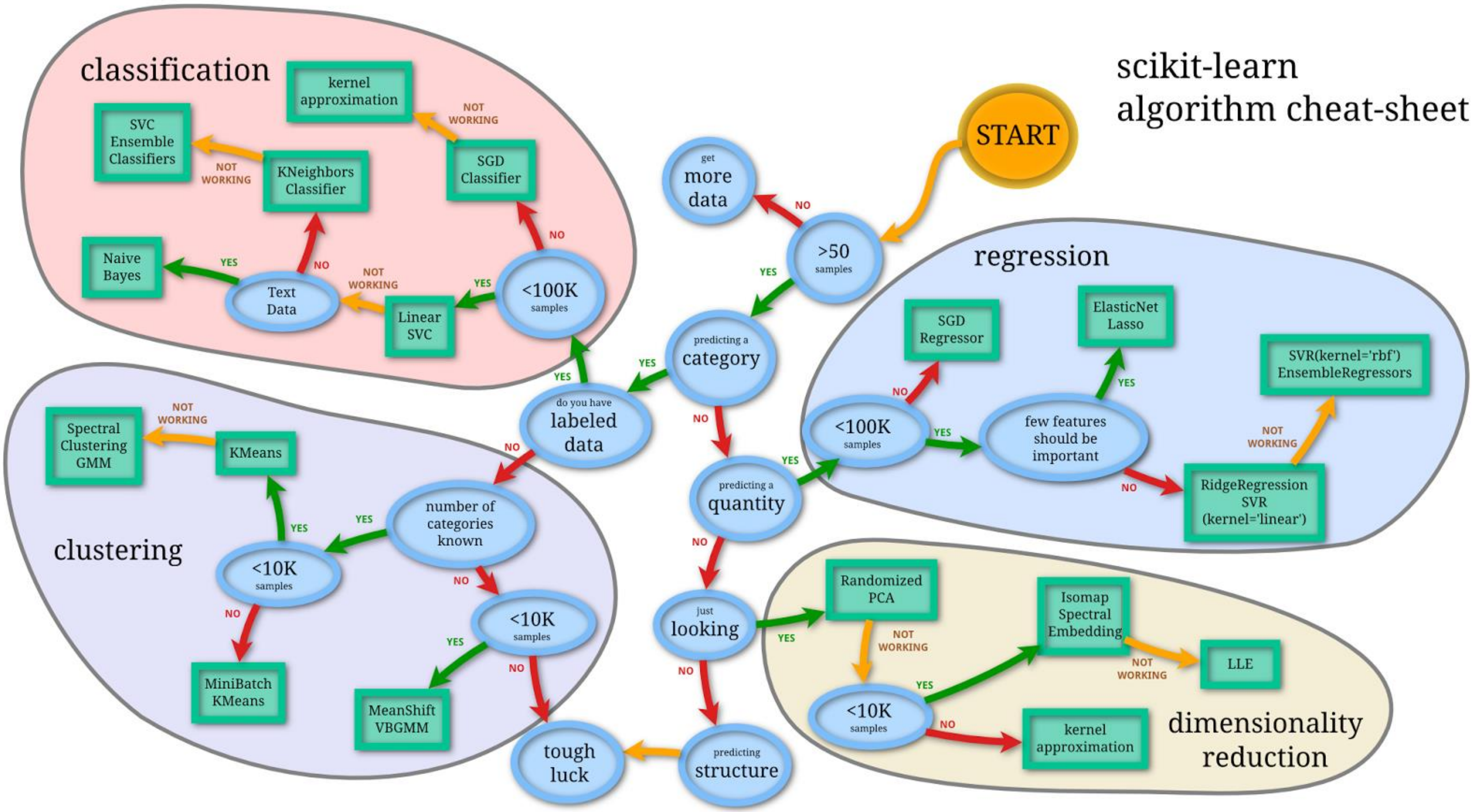


Even an experienced data scientist cannot tell which algorithm will perform the best before trying different algorithms.

If you want to perform dimension reduction then use principal component analysis.
If you need a numeric prediction quickly, use decision trees or logistic regression.
If you need a hierarchical result, use hierarchical clustering.

Machine Learning Algorithms Cheat Sheet




scikit-learn
algorithm cheat-sheet
START
>50 samples
get more data
predicting a category
do you have labeled data
<100K samples
predicting a quantity
just looking
predicting structure
tough luck
classification
kernel approximation
SVC Ensemble Classifiers
KNeighbors Classifier
SGD Classifier
Naive Bayes
Text Data
Linear SVC
clustering
Spectral Clustering GMM
KMeans
number of categories known
<10K samples
MiniBatch KMeans
MeanShift VBGMM
regression
SGD Regressor
ElasticNet Lasso
SVR(kernel='rbf') EnsembleRegressors
few features should be important
RidgeRegression SVR (kernel='linear')
dimensionality reduction
Randomized PCA
Isomap Spectral Embedding
LLE
kernel approximation
YES
NO
NOT WORKING

The **performance measure** is the way you want to evaluate a solution to the problem. It is the measurement you will make of the predictions made by a trained model on the test dataset.

- **Test and Train Datasets**

  From the transformed data, you will need to select a test set and a training set. An algorithm will be trained on the training dataset and will be evaluated against the test set. This may be as simple as selecting a random split of data (66% for training, 34% for testing) or may involve more complicated sampling methods.

- **Cross Validation**

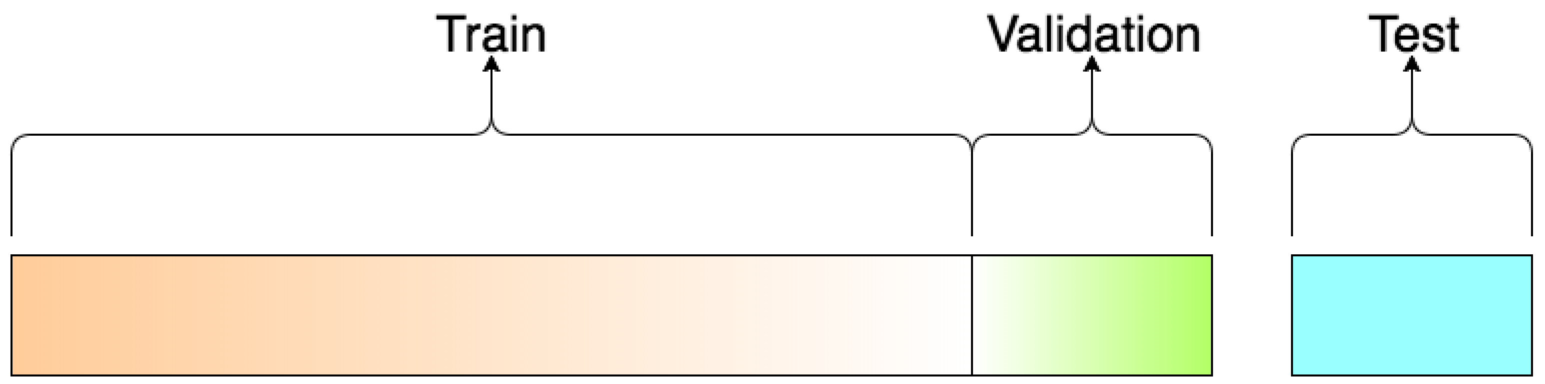
  A more sophisticated approach than using a test and train dataset is to use the entire transformed dataset to train and test a given algorithm. A method you could use in your test harness that does this is called cross validation.

- **Testing Algorithms**

  The best first algorithm to spot check is a random. Plug in a random number generator to generate predictions in the appropriate range. This should be the worst “algorithm result” you achieve and will be the measure by which all improvements can be assessed.

  - Select 5-10 standard algorithms that are appropriate for your problem and run them through your test harness.

Training set vs. Test set vs. Validation set



Training set: The sample of data used to train the model, i.e. fitting the parameters (e.g., neural network weights and biases etc.).

Validation set: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters.
The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.

Test set: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

The only purpose of the test set is to evaluate the final model.

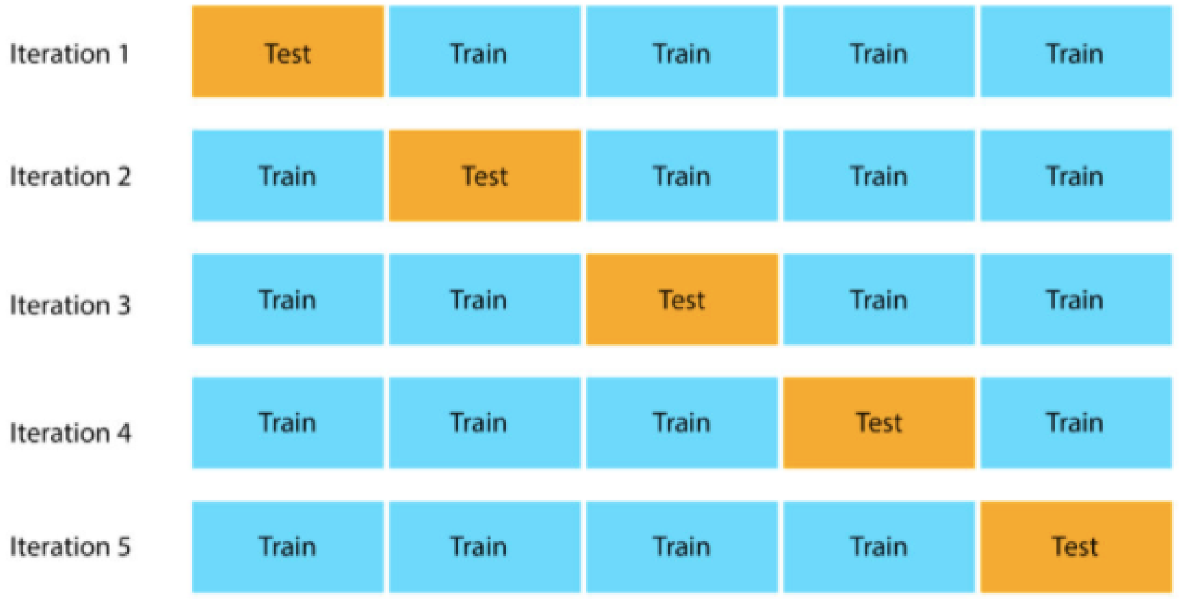
A **model parameter** is a variable that is **internal to the model**. The value of a parameter is estimated from training data.
A **hyperparameter** is a variable that is **external to the model**. The value cannot be estimated from data, and they are commonly used to estimate model parameters.

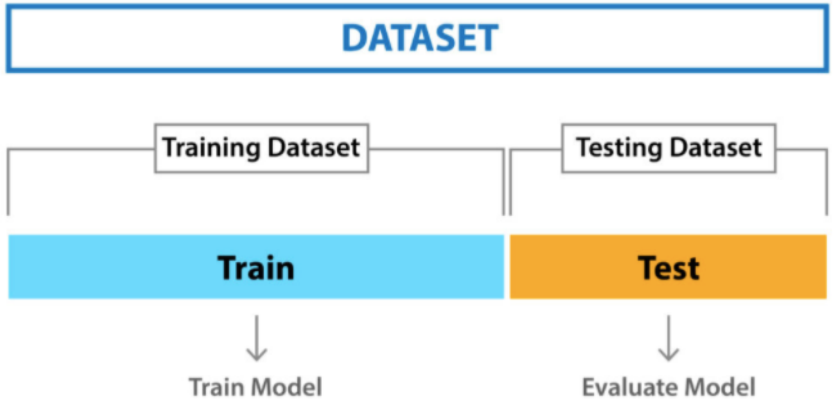
The **validation set** is different from the test set in that it is **used in the model building process for hyperparameter selection and to avoid overfitting**. It can therefore be regarded as a part of the training set.

Cross Validation

Cross-validation is a technique used to determine how the results of a machine learning model could be generalized to new, unseen data.

In cross-validation, you make a fixed number of folds (or partitions) of the data, run the analysis on each fold, and then average the overall error estimate.





Holdout Method - good to use when dataset is very large.

K-fold cross validation is one way to improve the holdout method when dataset not too large.
The data set is divided into **k** number of subsets and the holdout method is repeated k number of times.
The average of the k recorded accuracy is called the **cross-validation accuracy** and will serve as the **performance metric** for the model.

4. Improve Results

The process of improving results involves:

- **Algorithm Tuning:** where discovering the best models is treated like a search problem through model parameter space.
- **Ensemble Methods**: where the predictions made by multiple models are combined.
- **Extreme Feature Engineering**: where the attribute decomposition and aggregation seen in data preparation is pushed to the limits.

  - ✓ Run an automated sensitivity analysis on the parameters of the top performing algorithms.
  - ✓ Design and run experiments using standard ensemble methods of the top performing algorithms.
  - ✓ Statistical significance of results is critical here. It is so easy to focus on the methods and play with algorithm configurations. The results are only meaningful if they are significant and all configuration are already thought out and the experiments are executed in batch.

5. Present Results

There are two main facets to making use of the results of your machine learning endeavor:

- Report the results
- Operationalize the system

(i) Report the results

- **Context (Why):** Define the environment in which the problem exists and set up the motivation for the research question.
- **Problem (Question):** Concisely describe the problem as a question that you went out and answered.
- **Solution (Answer):** Concisely describe the solution as an answer to the question you posed in the previous section. Be specific.
- **Findings:** Bulleted lists of discoveries you made along the way that interests the audience. They may be discoveries in the data, methods that did or did not work or the model performance benefits you achieved along your journey.
- **Limitations**: Consider where the model does not work or questions that the model does not answer. Do not shy away from these questions, defining where the model excels is more trusted if you can define where it does not excel.
- **Conclusions (Why + Question + Answer):** Revisit the “why”, research question and the answer you discovered in a tight little package that is easy to remember and repeat for yourself and others.

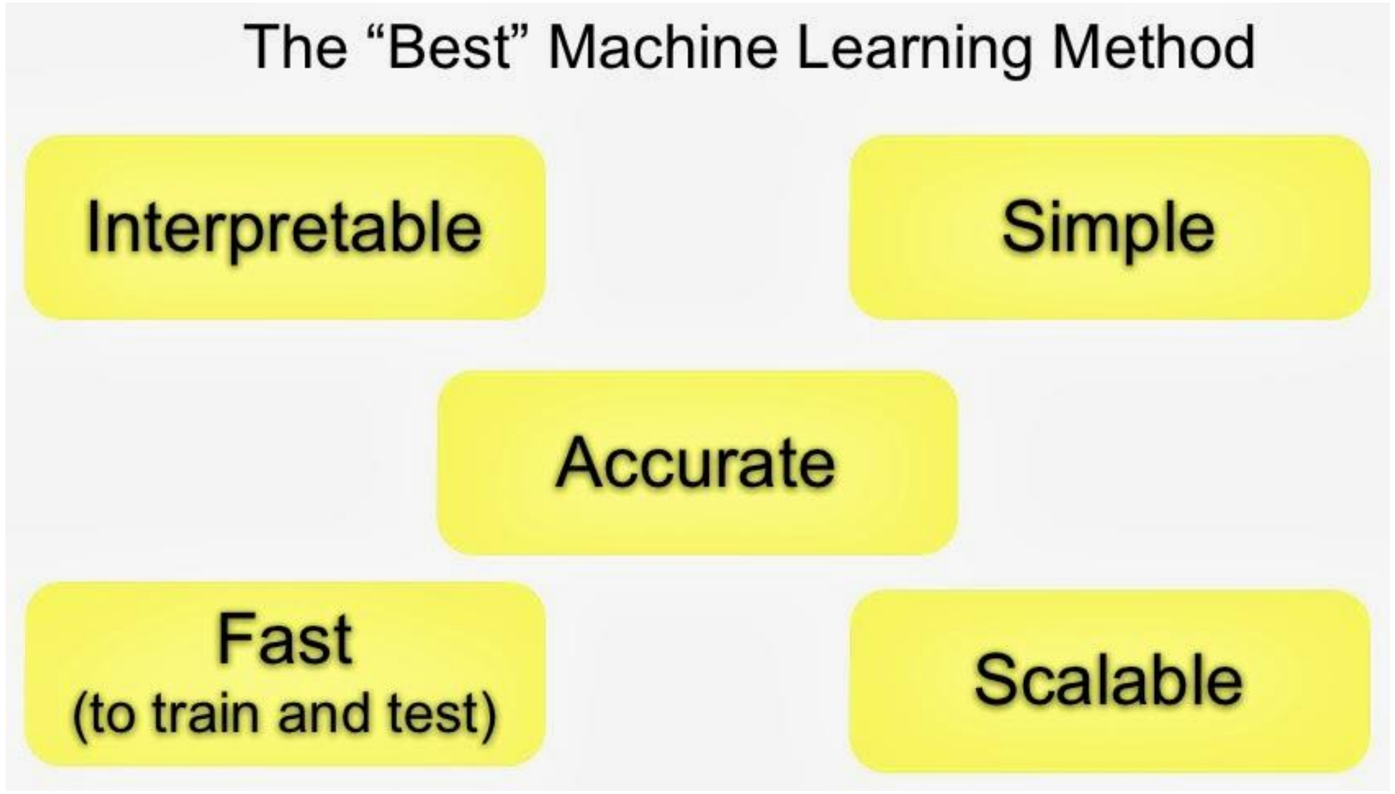
(ii) Operationalize the System

- You have found a model that is good enough at addressing the problem you face that you would like to put it into production.
- This may be an operational installation on your workstation in the case of a fun side project, all the way up to integrating the model into an existing enterprise application.
- Three key aspects to operationalizing a model to consider carefully before putting a system into production.
  - the algorithm implementation,
  - the automated testing of the model
  - the tracking of the models performance through time.
- These three issues will very likely influence the type of model you choose.

Important ML Concepts

Model Selection: Accuracy & other considerations

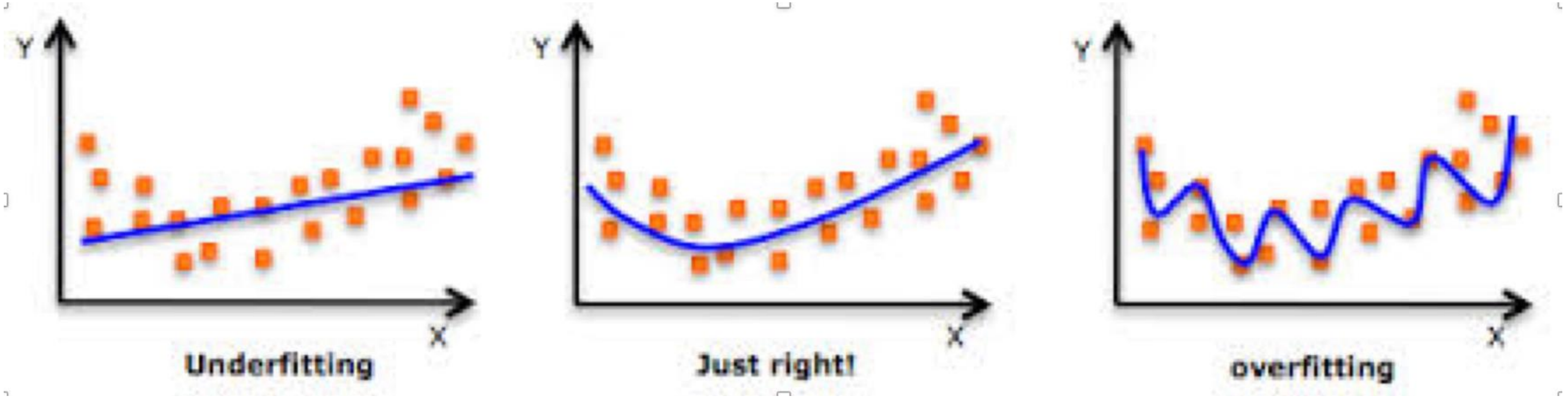
Accuracy is the main objective, but in practice tradeoffs have to be made in model selection. Find the **correct balance** between accuracy vs interpretability vs speed vs simplicity vs scalability.



- ***Speed*** (to train / score) is important if the model is to be used in production.
- ***Interpretability*** is critical if a model has to be explained for transparency reasons.
- ***Simplicity*** is important for practical reasons.
- ***Scalability*** means the learning algorithm can deal with any amount of data, without consuming ever growing amounts of resources like memory.

Overfitting and Underfitting

- **Overfitting** is making a model too **perfectly suited** to the **sample data**. It lacks the generalized relationship between input and output and thus holds **no predictive power**.
- **Underfitting** is when the model is **unable to describe** the **sample data well**. In a perfect world, a model is able to describe the input/output relationship in such a way that the logical relationship between input and the corresponding outcome holds true on new data.



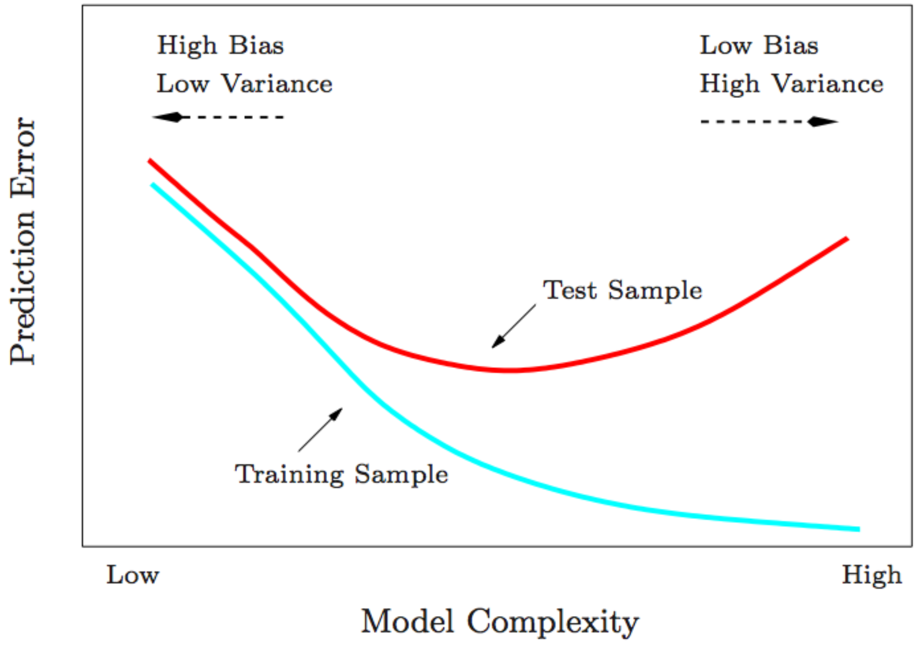
Bias (how well a model fits data) refers to errors due to inaccurate or simplistic assumptions in your ML algorithm, which leads to overfitting.

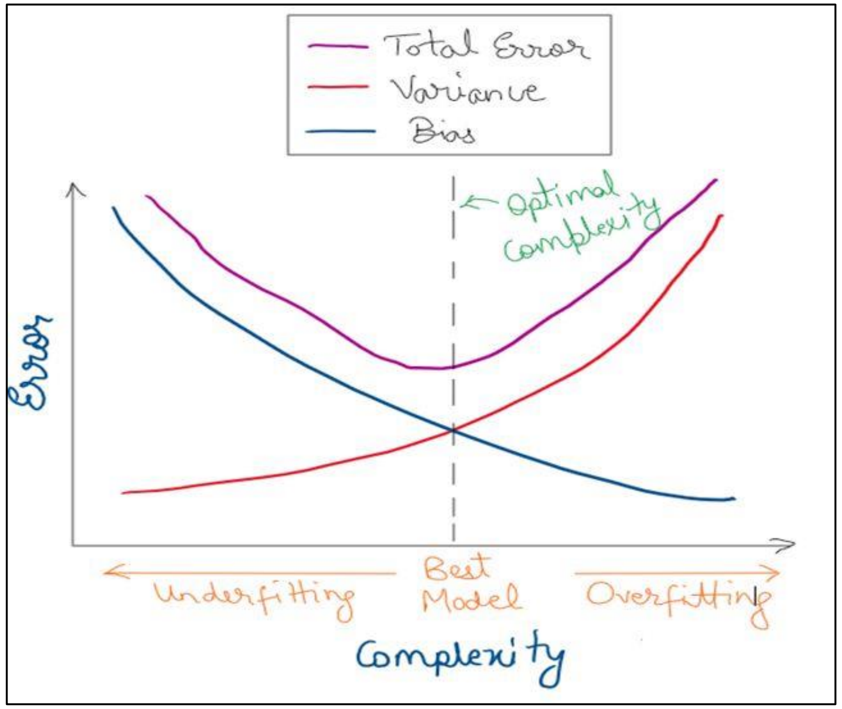
Variance (how much a model changes based on inputs) refers to errors due to complexity in your ML algorithm, which generates sensitivity to high levels of variation in training data and overfitting.

In other words, **simple models** are stable (low variance) but highly biased.

Complex models are prone to overfitting but express the truth of the model (low bias).

The optimal reduction of error requires a **tradeoff** of bias and variance to avoid both high variance and high bias.





In Sample VS. Out of Sample Errors

- **In Sample Error** : the error rate you get on the same data set you used to build your predictor, aka **resubstitution error.**
- **Out of sample error**: the error rate you get on a new data set, aka **generalization error.**

This is what we should care about, due to overfitting errors (in other words, **out of sample error** is more important than in sample error)

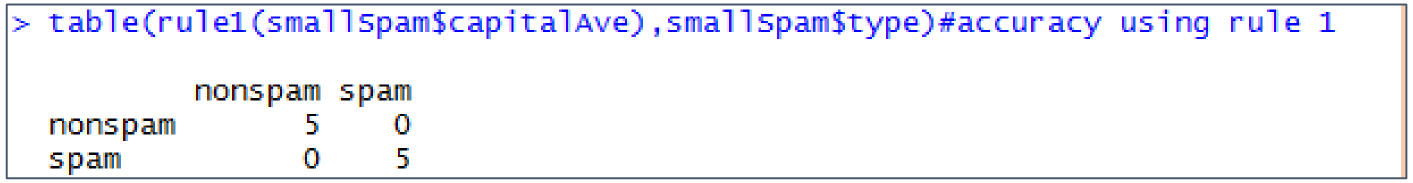
In Sample Vs Out Sample Errors Discussion

A subset of the SPAM data is taken, and called smallSpam–this consists of 10 observations selected at random

- A set of prediction rules is build around the feature capitalAve(average length uninterrupted sequence of capital letters)

Prediction rule 1:

- capitalAve> 2.7 = 'spam'
- capitalAve< 2.4 = 'nonspam'
- between 2.4 and 2.45 = spam
- between 2.45 and 2.7 = nonspam

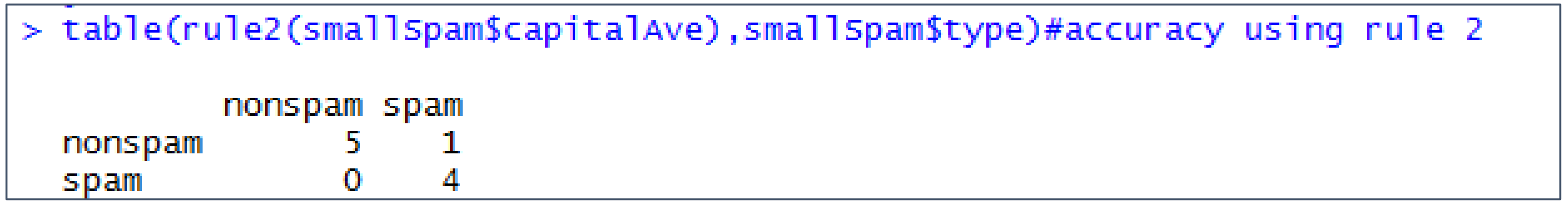


```
> table(rule1(smallSpam$capitalAve),smallSpam$type)#accuracy using rule 1

          nonspam spam
  nonspam       5    0
  spam          0    5
```

- This gives **perfect results**!

In Sample Vs Out Sample Errors Discussion

Suppose we introduce Rule 2, which states that capitalAve> 2.4 = spam, else nonspam



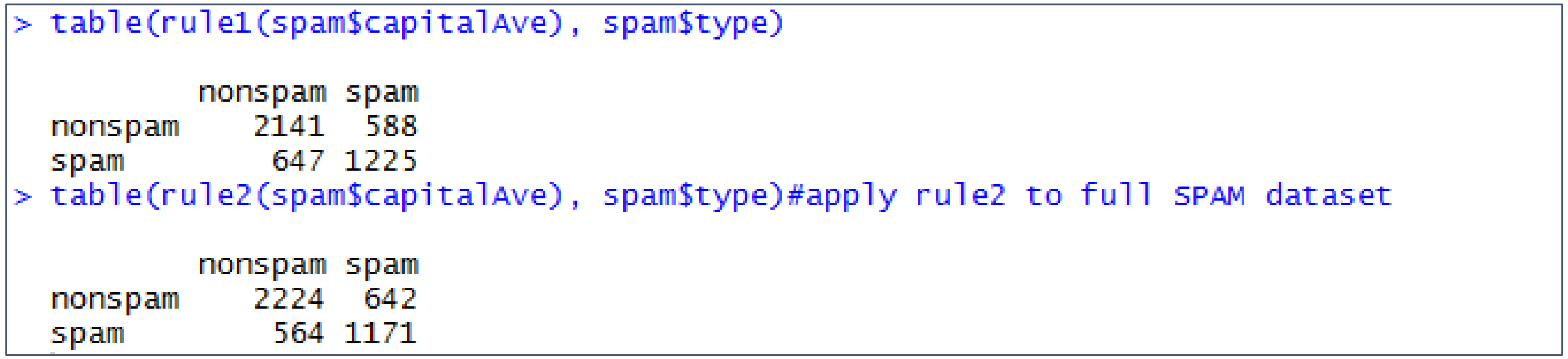
```
> table(rule2(smallSpam$capitalAve),smallSpam$type)#accuracy using rule 2

            nonspam spam
  nonspam         5    1
  spam            0    4
```

This results in 1 error where a non-spam is classified as a spam

Which Rule is better? Rule 1 or Rule 2?

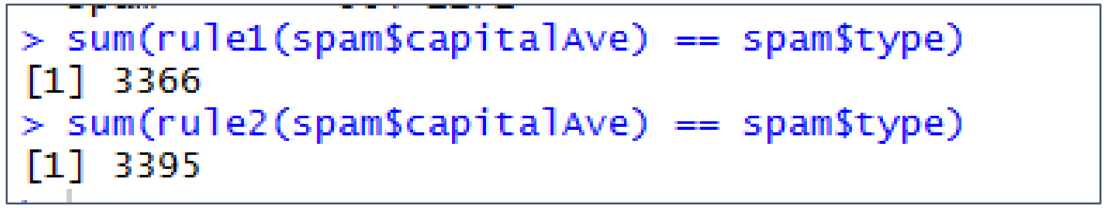
- At this point we have considered Rule 1 and Rule 2 on a subset of the SPAM dataset.
- In this case, for the subset that we have considered , in sample error seems acceptable
- What about Out Sample Errors?
- Test Rule 1 and Rule 2 on the entire SPAM dataset.

In Sample Vs Out Sample Errors Discussion



```
> table(rule1(spam$capitalAve), spam$type)

          nonspam spam
  nonspam    2141  588
  spam        647 1225
> table(rule2(spam$capitalAve), spam$type)#apply rule2 to full SPAM dataset

          nonspam spam
  nonspam    2224  642
  spam        564 1171
```

Rule 1 and Rule 2’s accuracy for the whole dataset is examined. Which is better?



```
> sum(rule1(spam$capitalAve) == spam$type)
[1] 3366
> sum(rule2(spam$capitalAve) == spam$type)
[1] 3395
```

What caused Rule 1 to do well on the small SPAM data set but not so on the full SPAM data set?

- OVERFITTING has occured
- Data have two parts: signal, noise.
- The goal of a predictor is to find signal.
- You can always design a perfect in-sample predictor.
- You capture both signal and noise when you do that.
- Predictor won't perform as well on new samples.

Tips on minimizing In and Out of Sample Error

- Define your error rate.
- Split data into: training, testing, validation (optional).
- On the training set, pick features. use cross-validation.
- On the training set, pick prediction function. use cross-validation.
- If no validation: apply once to test set (don't want to use test set to train the model).
- If there is validation: apply to test set and refine, and apply once to validation.
- Know the benchmarks

Study design:

- E.g. Netflix, test may be 'hold out set', split to 3: probe (labels provided), quiz, test (so you would train on training, and apply to the probe. netflixapplied model to quiz set and you could see results. Test was for ultimate test at end of competition.)
- Kaggle: used by professionals

- Avoid small sample sizes, especially for test set.

Tips on minimizing In and Out of Sample Error

Rules of thumb for prediction study design

- If you have a **large sample size**: 60% training, 20% test, 20% validation.
- **Medium sample size**: 60% training, 40% testing.
- **Small**: do **cross validation**, report caveat of small sample size, or even reconsider whether to do.

Some **principles to remember**

- Set the test validation set aside and don't look at it.
- In general, randomly sample training and test.
- Your data sets must reflect structure of the problem -if predictions evolve with time, split train/test in time chunks (called backtesting in finance).
- All subsets should reflect as much diversity as possible: random assignment does this; you can also try to balance by features, but this is tricky.

Measuring Error in ML Model
Common Error Measures

In **regression**, your goal is to predict the value of an output value given one or more input values.

- R-squared
- Root mean squared error (RMSE), aka standard error

In **binary classification**, you aim to predict which of two classes an observation will fall.

- Accuracy
- Precision
- Recall (aka sensitivity)
- F1 (aka F-Score)

Multiclass classification is the task of determining which of three or more classes a specific observation belongs to.

- Confusion matrix
- Cohen’s Kappa
- Informedness (aka Powers’ Kappa)

Confusion Matrix

A **confusion matrix** - describe the performance of a classification model on a set of test data for which the true values are known.

		Predicted class	
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

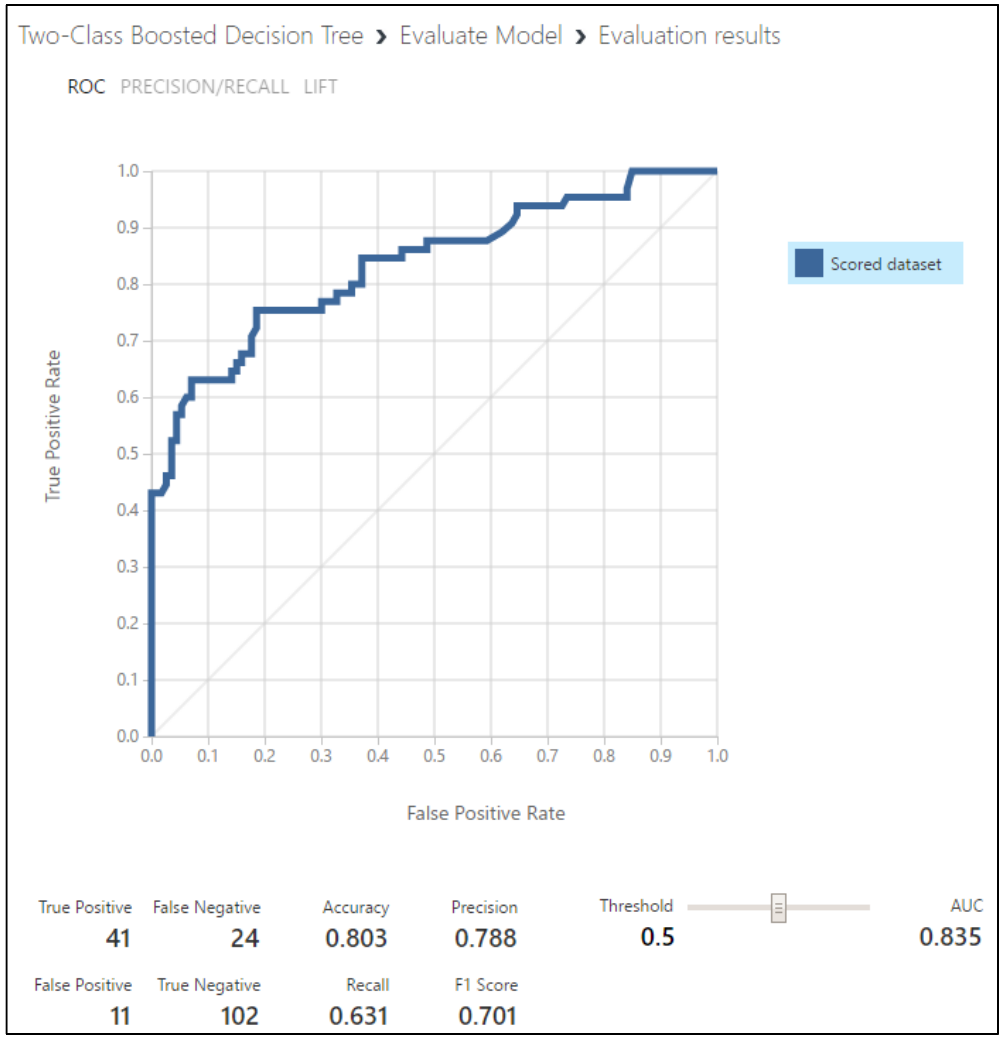
True positive and **true negatives** are the observations that are correctly predicted.
We want to minimize false positives and false negatives

True Positives (TP) - These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes.

True Negatives (TN) - These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no.

False positives and **false negatives,** these values occur when your actual class contradicts with the predicted class.
False Positives (FP) – When actual class is no and predicted class is yes.
False Negatives (FN) – When actual class is yes but predicted class in no.

Once you understand these four parameters then we can calculate Accuracy, Precision, Recall and F1 score.



Two-Class Boosted Decision Tree › Evaluate Model › Evaluation results

ROC PRECISION/RECALL LIFT

True Positive	False Negative	Accuracy	Precision	Threshold	AUC
41	24	0.803	0.788	0.5	0.835
False Positive	**True Negative**	**Recall**	**F1 Score**		
11	102	0.631	0.701		

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.
From the model, we have got 0.803 which means our model is approx. 80% accurate.

Accuracy = TP+TN/TP+FP+FN+TN

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

Precision = TP/TP+FP

Recall (Sensitivity) is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

Recall = TP/TP+FN

F1 Score is the weighted average of Precision and Recall.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Type I and Type II Error

Type I Error

- This is the **incorrect rejection** of a true Null Hypothesis (Ho).
- Null Hypothesis is a statement that is by default true unless proven otherwise.
- Type I Error leads to False Positive (FP) - (claiming something has happened when it hasn’t)

Type II Error

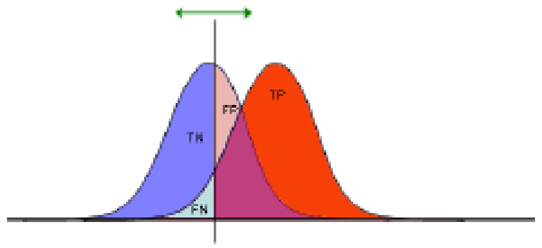
- This is the **incorrect retaining** of a false Null Hypothesis (Ho).
- This is synonymous to when the system ignore the possibility that it might not have retrieved some document which are relevant.
- This type of error leads to False Negative (FN) - (claiming nothing has happened when it actually has)

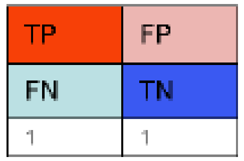
False Positives and False Negatives are the two unique characteristics of Precision and Recall respectively.
To decrease one means increasing the other as P α 1/R

Receiver Operating Characteristics (ROC)

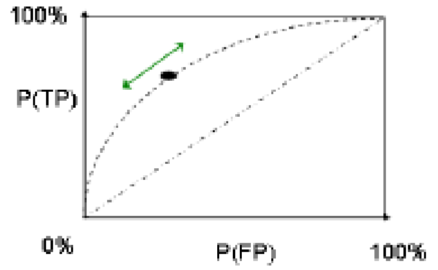
- **ROC** is a measure of the quality of the goodness of a prediction algorithm
- ROC exists as a curve because:
  - In binary classification, you are predicting one of two categories: alive/dead, click on ad/do not click.
  - But predictions are often quantitative: P(alive), prediction on a scale from 1-10
  - The cutoff you choose gives different results

The ROC curve is a **graphical representation** of the performance of a classification model at all thresholds. It has two thresholds: true positive rate & false positive rate.



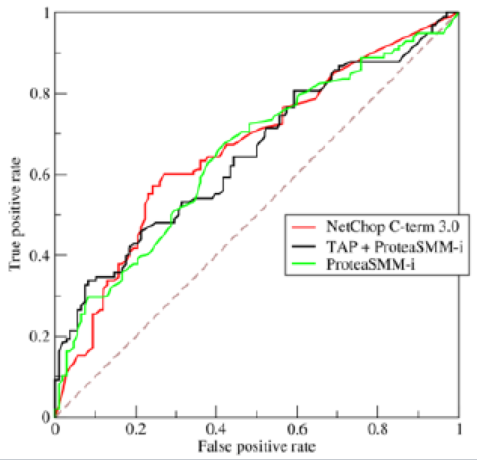


TP	FP
FN	TN
1	1



Description of a ROC curve

- x axis = 1-specificity (P(FP) = false positive rate)
- y axis = sensitivity (P(TP) = true positive rate)
- Both axes go 0 to 1, with a concave down curve



Evaluating Area Under ROC Curve (AUC)

AUC (Area Under the ROC Curve) is, simply, the area under the ROC curve.
AUC measures the two-dimensional area underneath the ROC curve from (0,0) to (1,1).
It used as a performance metric for evaluating binary classification models.

- AUC = 0.5: random guessing (45 degree line basically)
- AUC < 0.5 means you did worse
- AUC = 1: perfect classifier
- In general, AUC above 0.8 is considered 'good'

Conclusion

The goal of ML is never to make "perfect" guesses, because ML deals in domains where there is no such thing. The goal is to make guesses that are **good enough** to be useful.

Tutorial 9

Go the following link, study the code and try out to see how ML model is built in R. (No need to submit)

Building Spam Filter Using ML Model

- https://rpubs.com/Seun/455974
- https://www.rpubs.com/hoakevinquach/SMS-Spam-or-Ham-Text

A **machine learning model** can only begin to add value to an organization when that **model's** insights routinely become available to the users for which it was built.

Find an example on how a ML model is deployed.

Share on **group WhatsApp**.

Acknowledgement & Credit

- Jeffery Leek Coursera course at John Hopkins
- Ray Mooney's ML course
- Jason Brownlee of machinelearningmastery.com
- SAS Blog
- http://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/