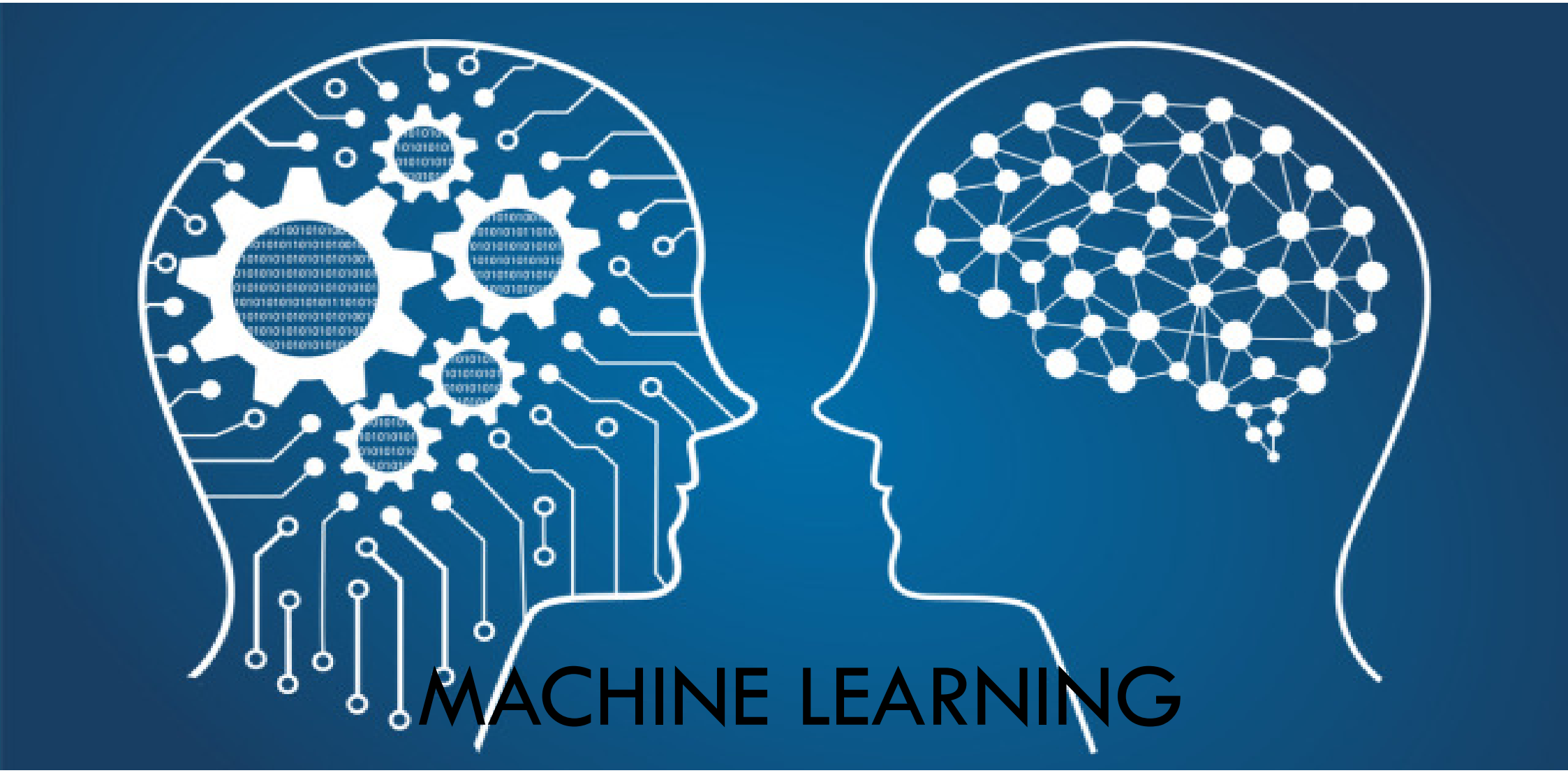


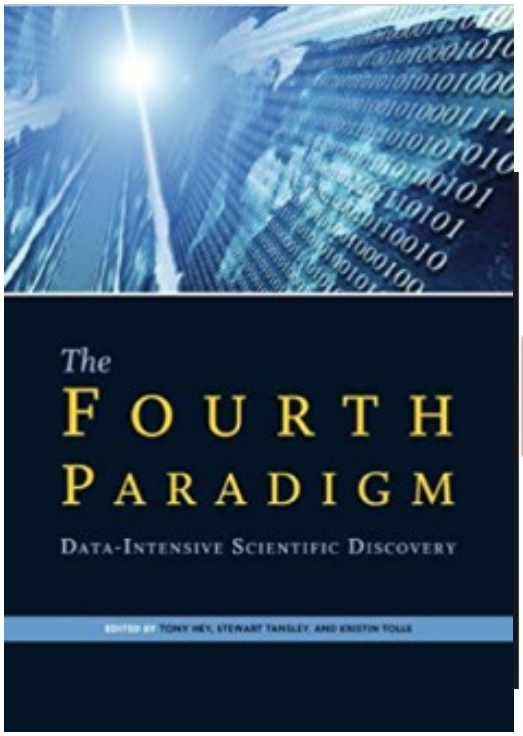
MACHINE LEARNING

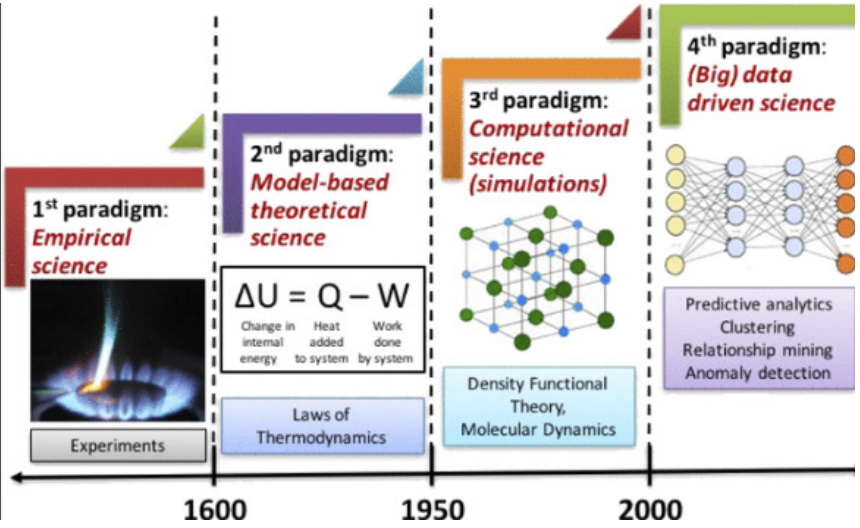
Pr. BEN LAHMAR EL Habib

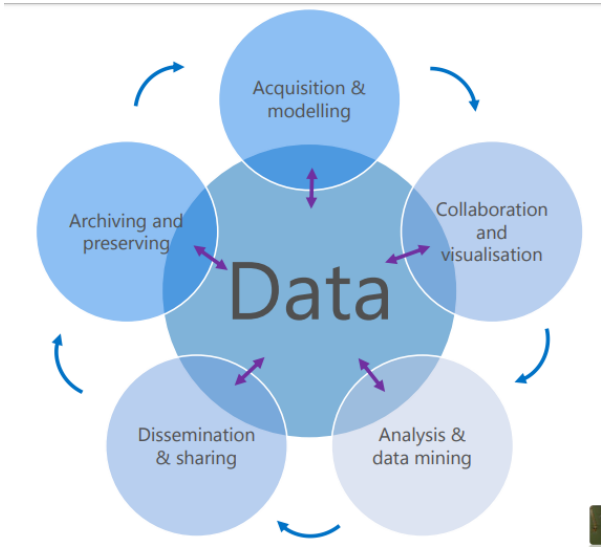
What is maching learning?

- Algorithms types
- Learning techniques: an overview
- Data Preparation (Data pre-processing)

Fourth Paradigm





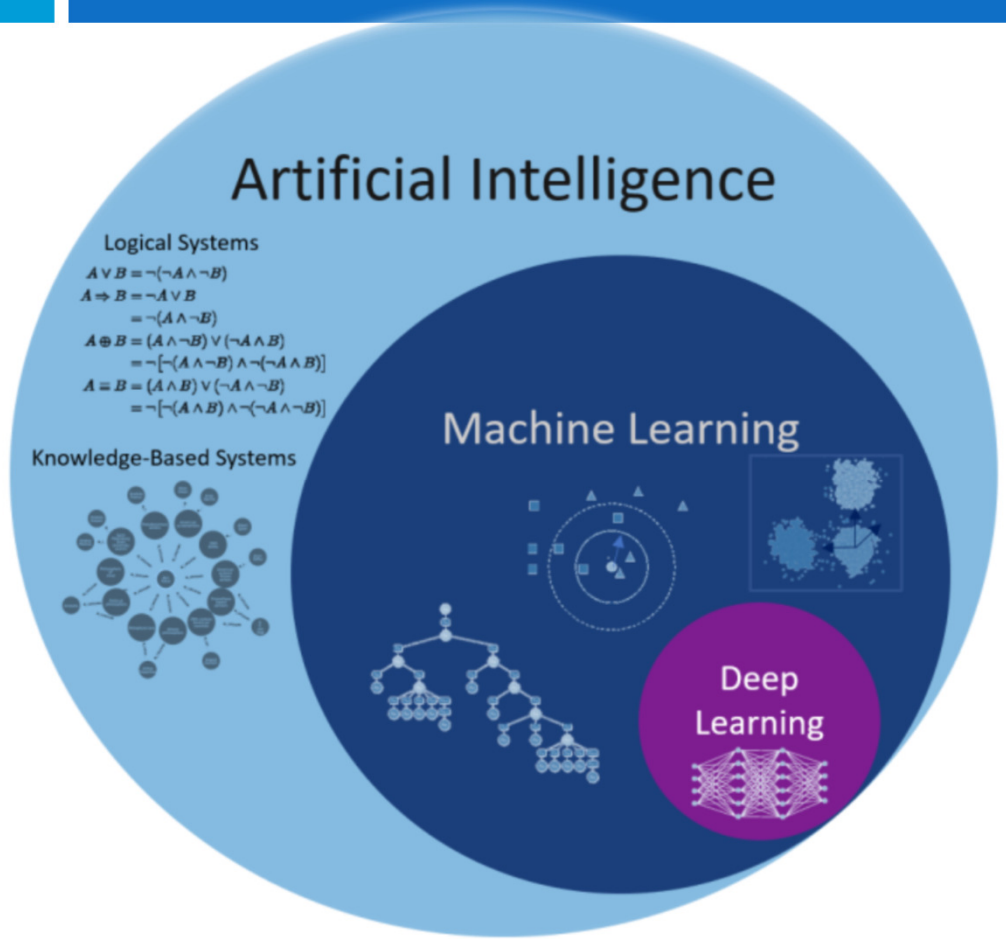


What We Talk About When We Talk About"Learning"

- Learning general models from a data of particular examples
- Data is cheap and abundant (data warehouses, data marts); knowledge is expensive and scarce.
- Example in retail: Customer transactions to consumer behavior:

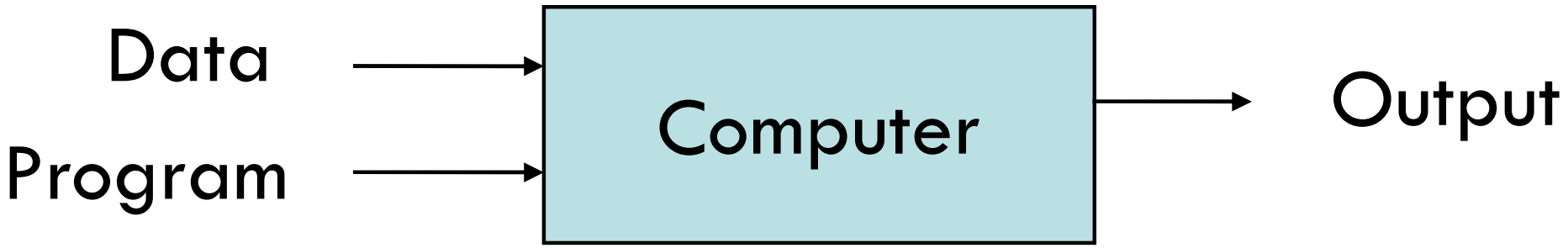
  People who bought "Da Vinci Code" also bought "The Five People You Meet in Heaven" (www.amazon.com)
- Build a model that is *a good and useful approximation* to the data.

What is machine learning?

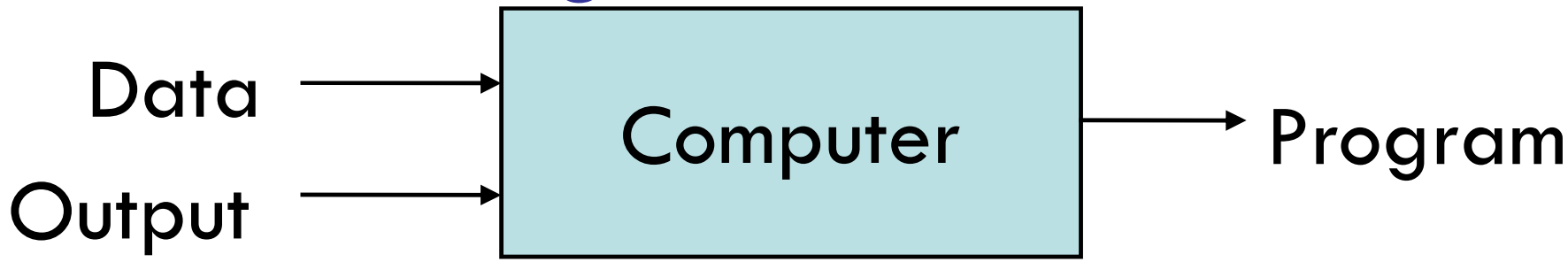


- A branch of **artificial intelligence**, concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data.
- As intelligence requires knowledge, it is necessary for the computers to acquire knowledge.

Traditional Programming



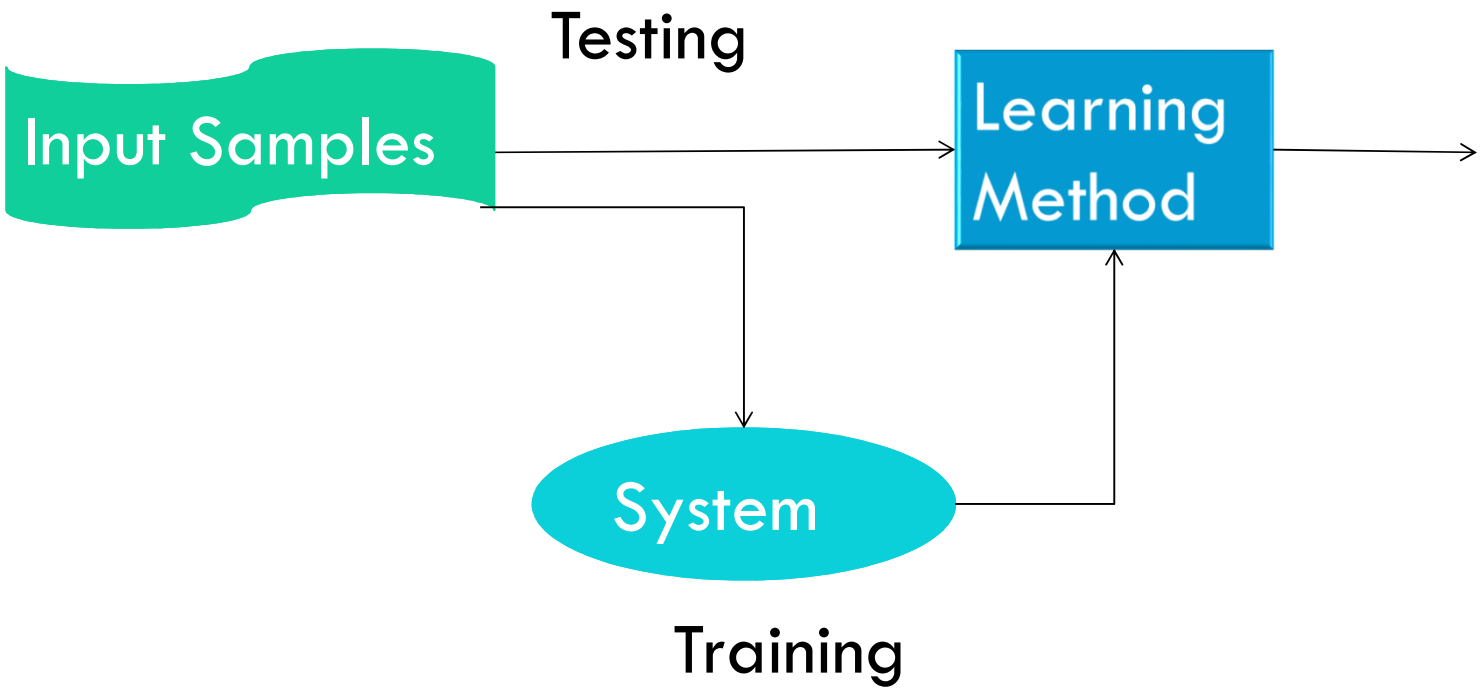
Machine Learning



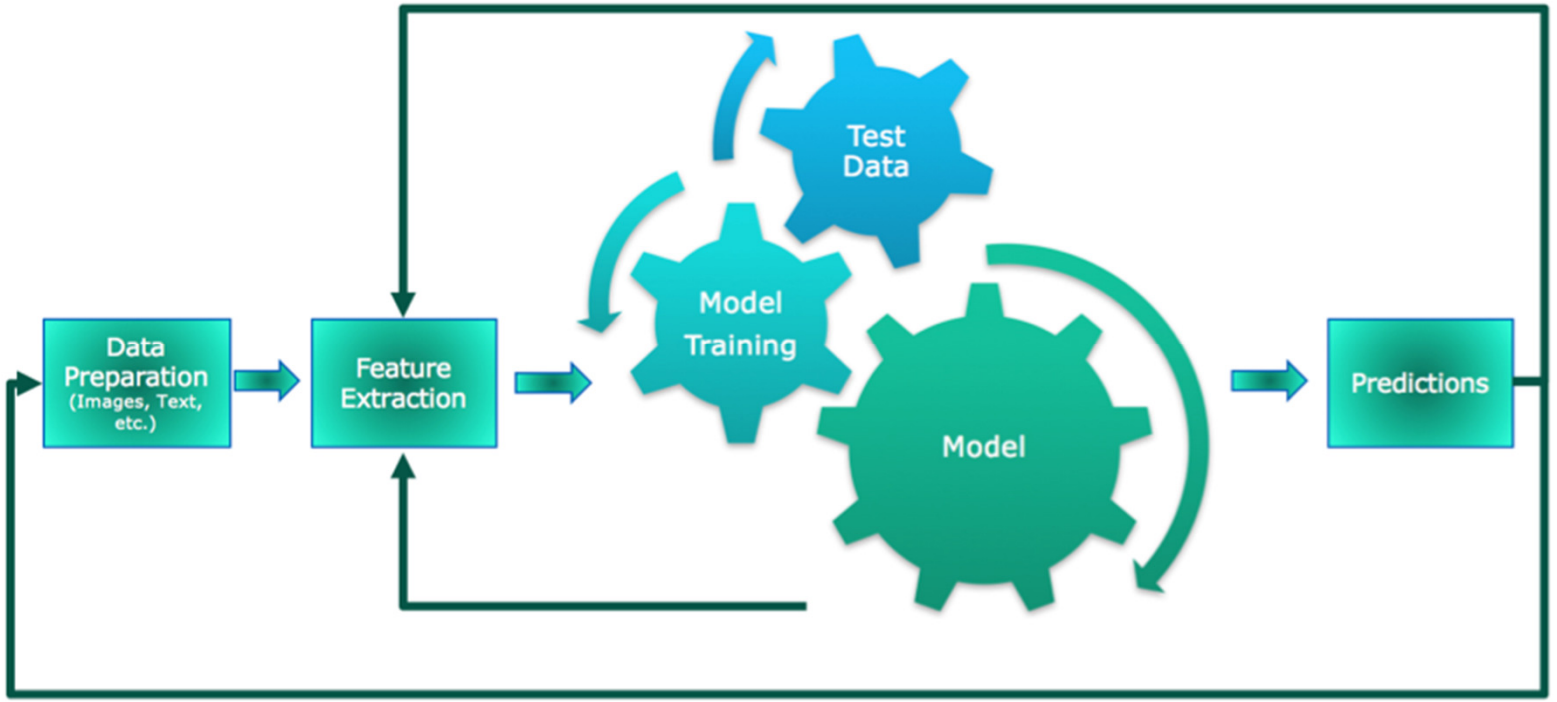
What is machine learning?

- Machine Learning
  - Study of algorithms that
  - improve their performance
  - at some task
  - with experience
- Optimize a performance criterion using example data or past experience.
- 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

Learning system model

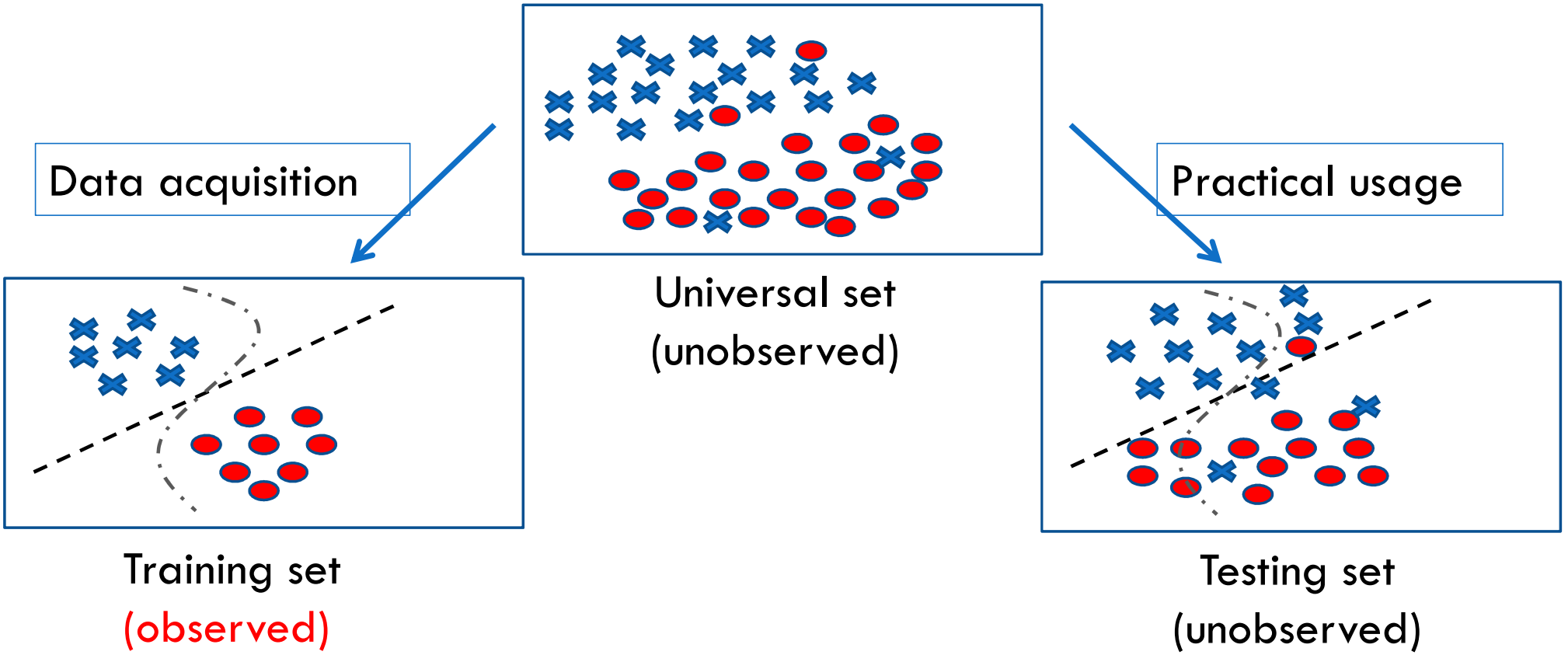


A standard Machine learning pipline



- *Feature extraction is critical for machine learning pipelines (Courtesy: Western Digital)*

Training and testing

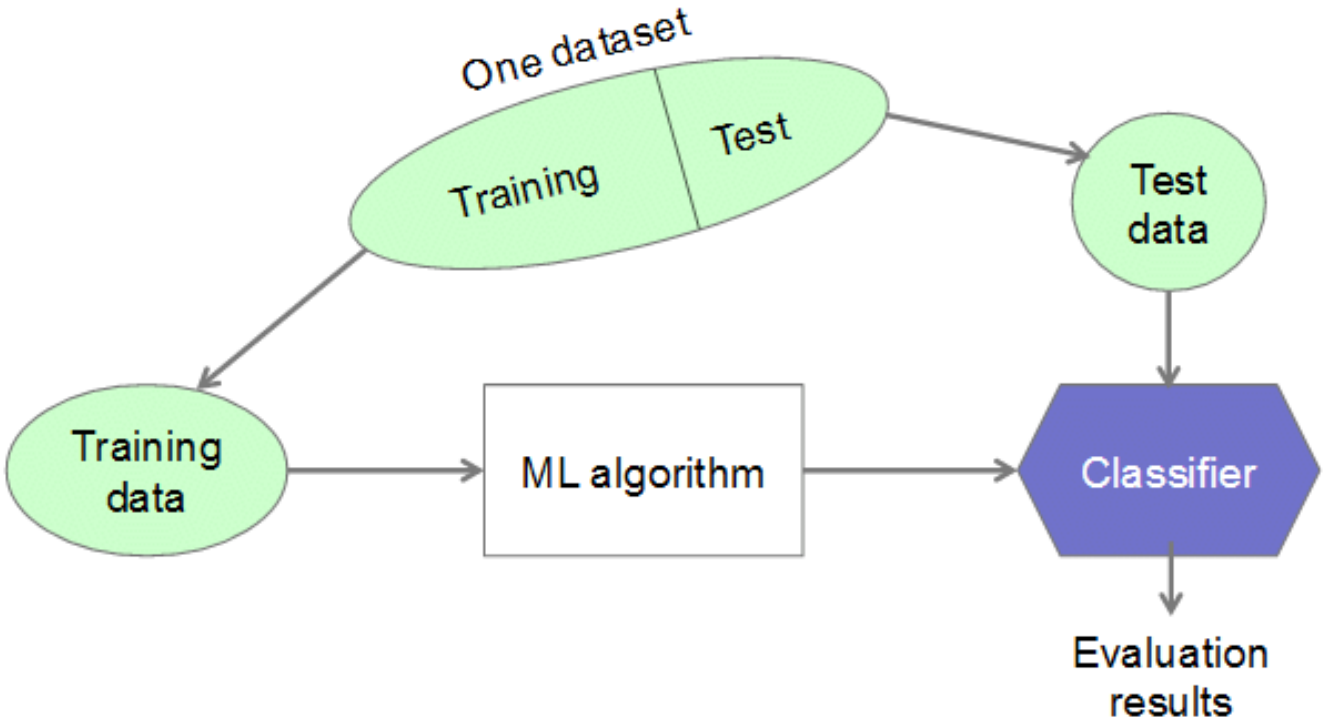


Training and testing

- 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

Training and testing



Performance

- There are several factors affecting the performance:
  - **Types of training** provided
  - The form and extent of any initial **background knowledge**
  - The **type of feedback** provided
  - The **learning algorithms** used

- Two important factors:
  - Modeling
  - Optimization

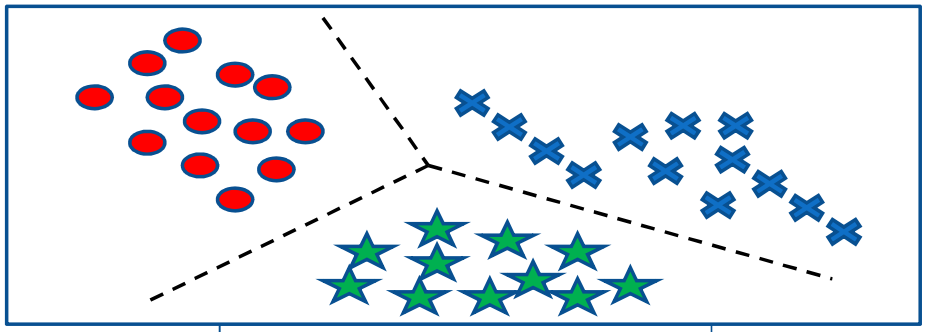
Algorithms

- The success of machine learning system also depends on the algorithms.
- The algorithms control the search to find and build the knowledge structures.
- The learning algorithms should extract useful information from training examples.

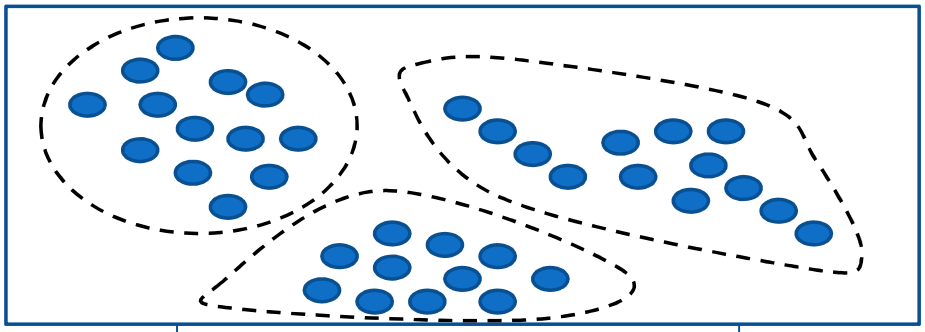
Algorithms

- **Supervised learning** ($\{x_n \in R^d, y_n \in R\}_{n=1}^N$)
  - Prediction
  - Classification (discrete labels), Regression (real values)
  - Training data includes desired outputs
- **Unsupervised learning** ($\{x_n \in R^d\}_{n=1}^N$)
  - Clustering
  - Probability distribution estimation
  - Finding association (in features)
  - Dimension reduction
  - Training data does not include desired outputs
- **Semi-supervised learning**
  - Training data includes a few desired outputs
- **Reinforcement learning**
  - Decision making (robot, chess machine)
  - Rewards from sequence of actions

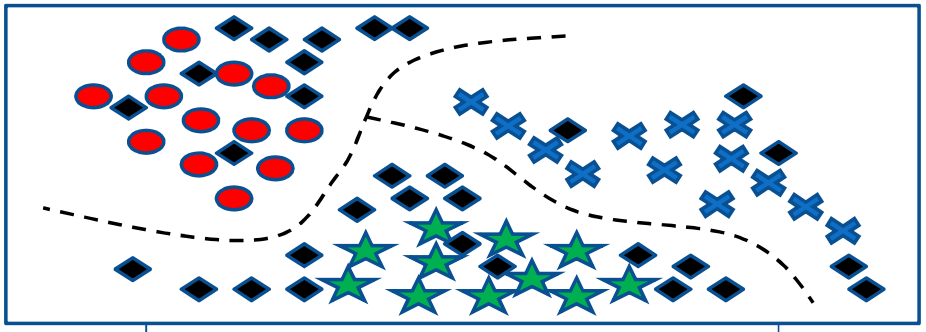
Algorithms



Supervised learning



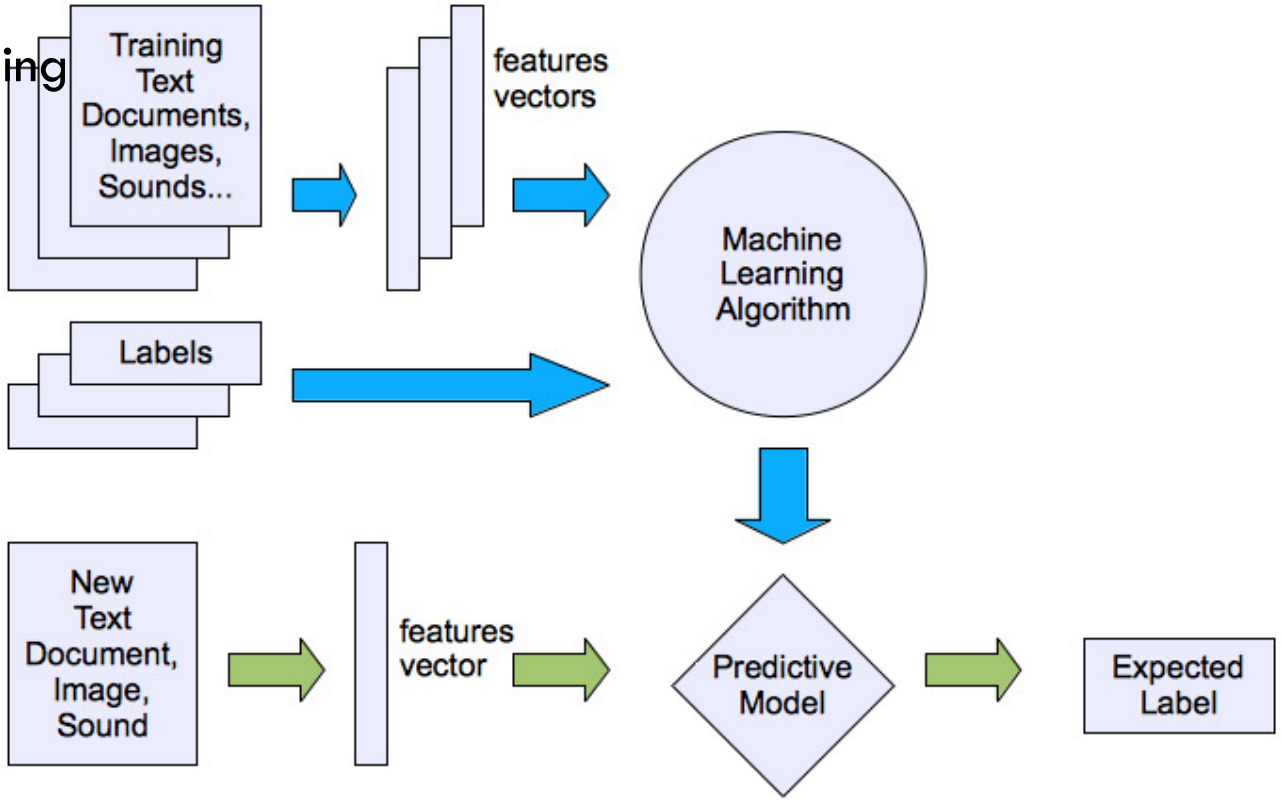
Unsupervised learning



Semi-supervised learning

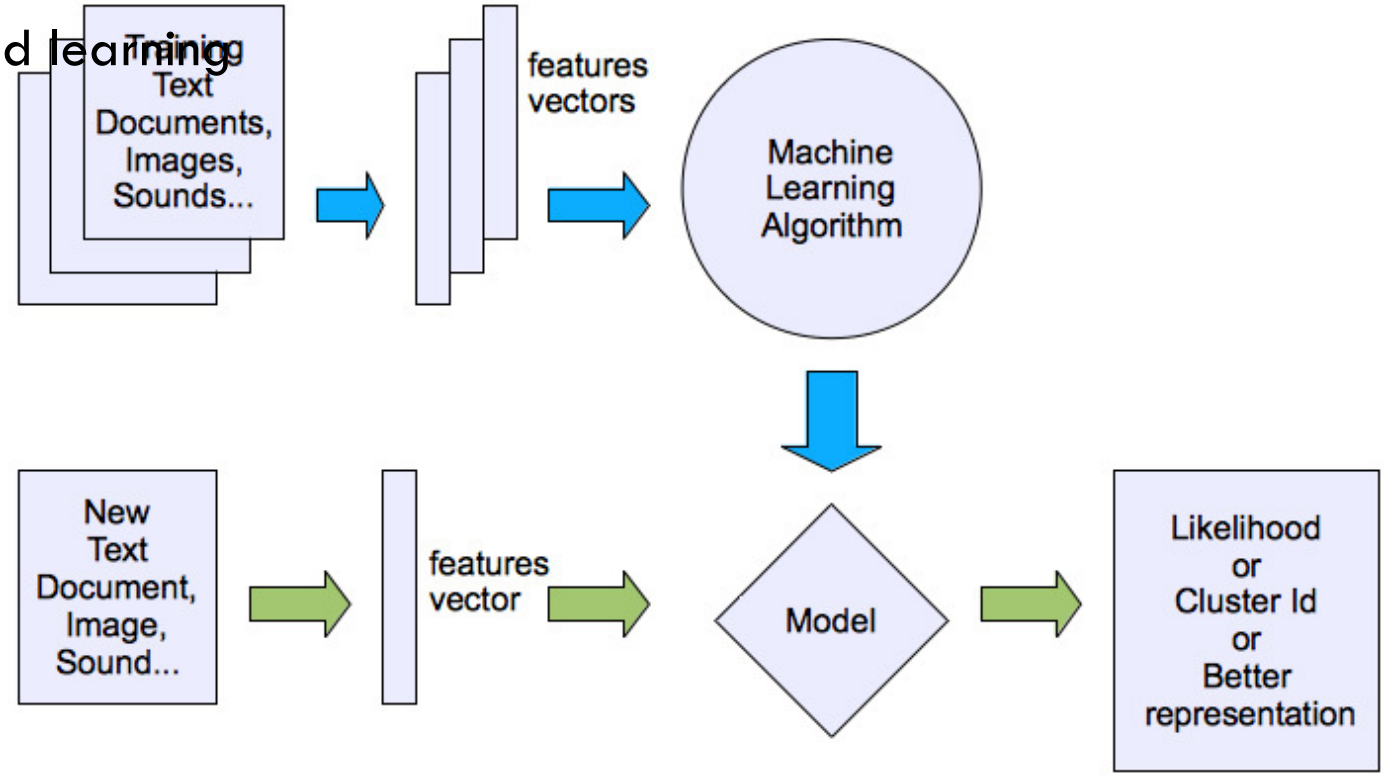
Machine learning structure

Supervised learning



Machine learning structure

Unsupervised learning



What are we seeking?

- Supervised: Low E-out or maximize probabilistic terms

$$error = \frac{1}{N}\sum_{n=1}^{N}[y_n \neq g(x_n)]$$

$$Eout(g) \leq Ein(g) \pm O\left(\sqrt{\frac{d_{VC}}{N} ln\, N}\right)$$

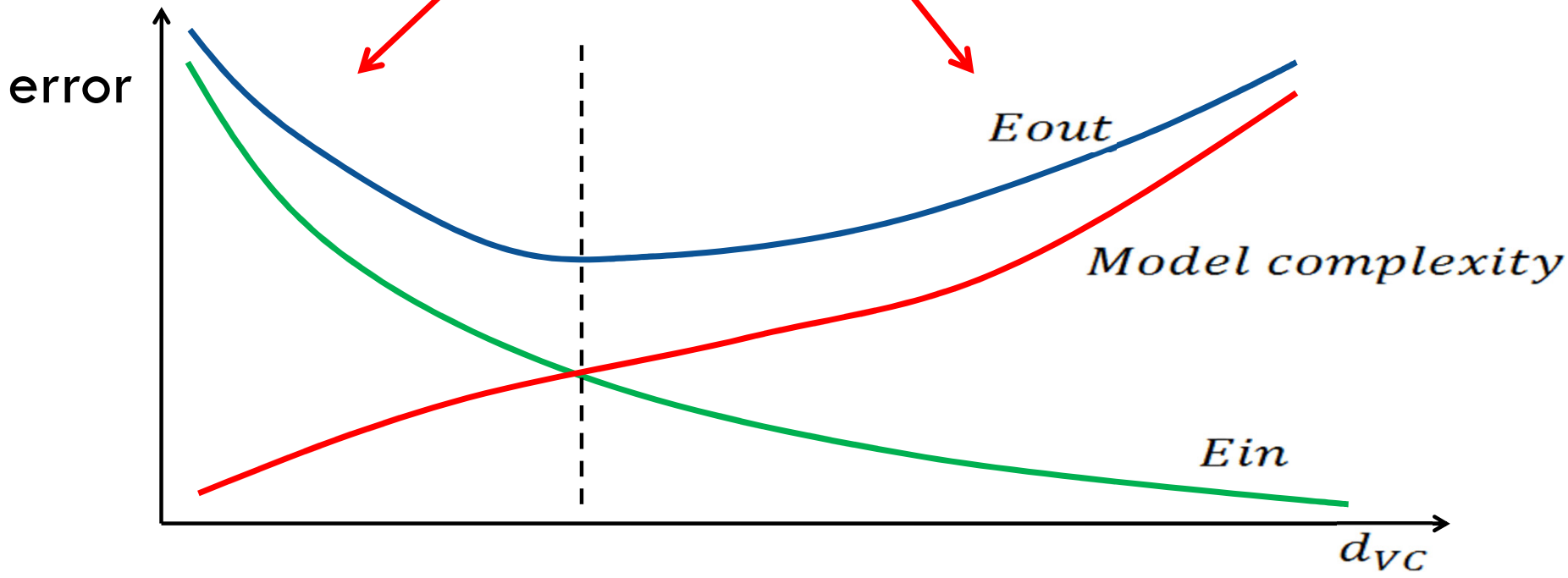
E-in: for training set

E-out: for testing set

- Unsupervised: Minimum quantization error, Minimum distance, MAP, MLE(maximum likelihood estimation)

What are we seeking?

Under-fitting VS. Over-fitting (fixed *N*)



Learning techniques

- Supervised learning categories and techniques
  - **Linear classifier** (numerical functions)
  - **Parametric** (Probabilistic functions)
    - Naïve Bayes, Gaussian discriminant analysis (GDA), Hidden Markov models (HMM), Probabilistic graphical models
  - **Non-parametric** (Instance-based functions)
    - *K*-nearest neighbors, Kernel regression, Kernel density estimation, Local regression
  - **Non-metric** (Symbolic functions)
    - Classification and regression tree (CART), decision tree
  - **Aggregation**
    - Bagging (bootstrap + aggregation), Adaboost, Random forest

Learning techniques

- Unsupervised learning categories and techniques
  - **Clustering**
    - K-means clustering
    - Spectral clustering
  - **Density Estimation**
    - Gaussian mixture model (GMM)
    - Graphical models
  - **Dimensionality reduction**
    - Principal component analysis (PCA)
    - Factor analysis

Reinforcement Learning

- Topics:
  - Policies: what actions should an agent take in a particular situation
  - Utility estimation: how good is a state (→used by policy)
- No supervised output but delayed reward
- Credit assignment problem (what was responsible for the outcome)
- Applications:
  - Game playing
  - Robot in a maze
  - Multiple agents, partial observability, ...

ML in a Nutshell

- Tens of thousands of machine learning algorithms
- Hundreds new every year
- Every machine learning algorithm has three components:
  - **Representation**
  - **Evaluation**
  - **Optimization**

Representation

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

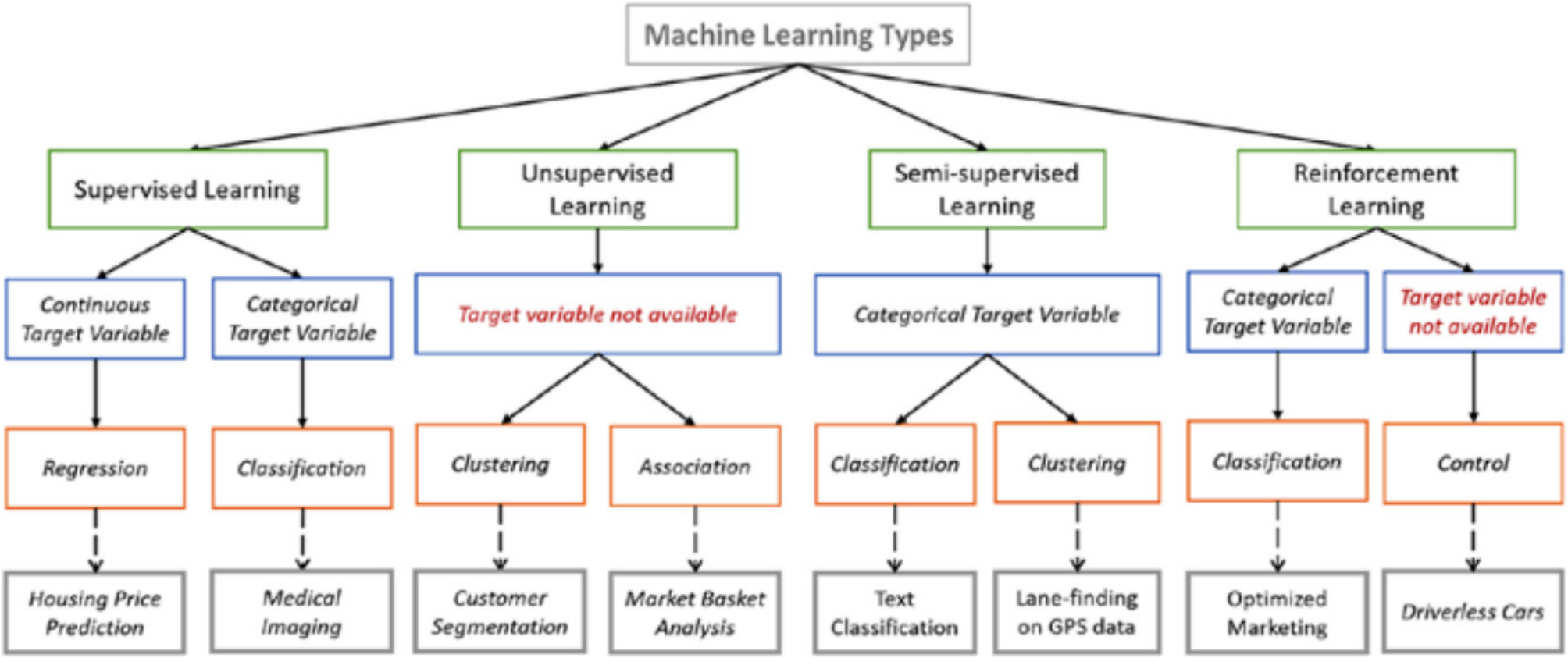
Evaluation

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

Optimization

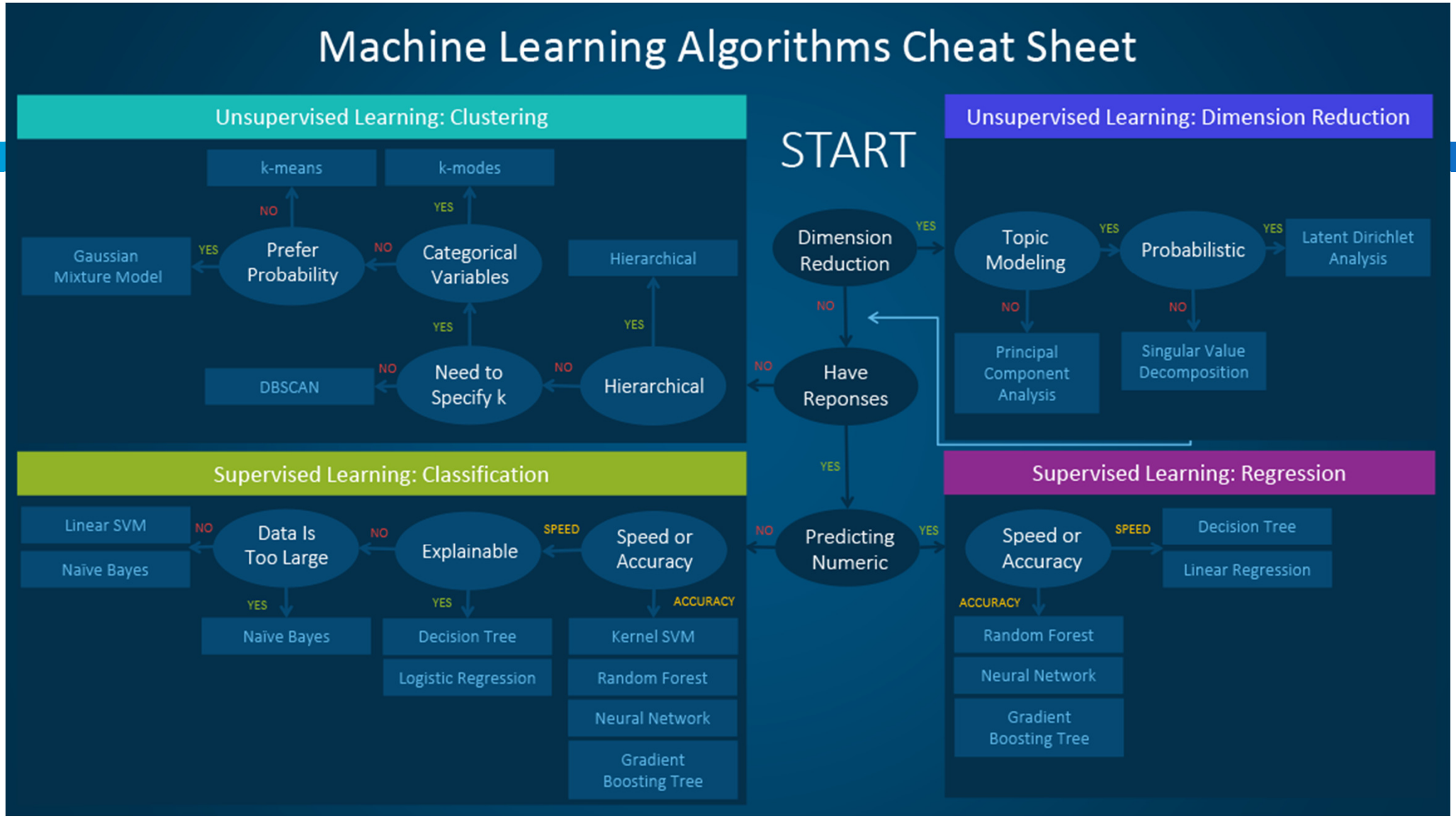
- Combinatorial optimization
  - E.g.: Greedy search
- Convex optimization
  - E.g.: Gradient descent
- Constrained optimization
  - E.g.: Linear programming

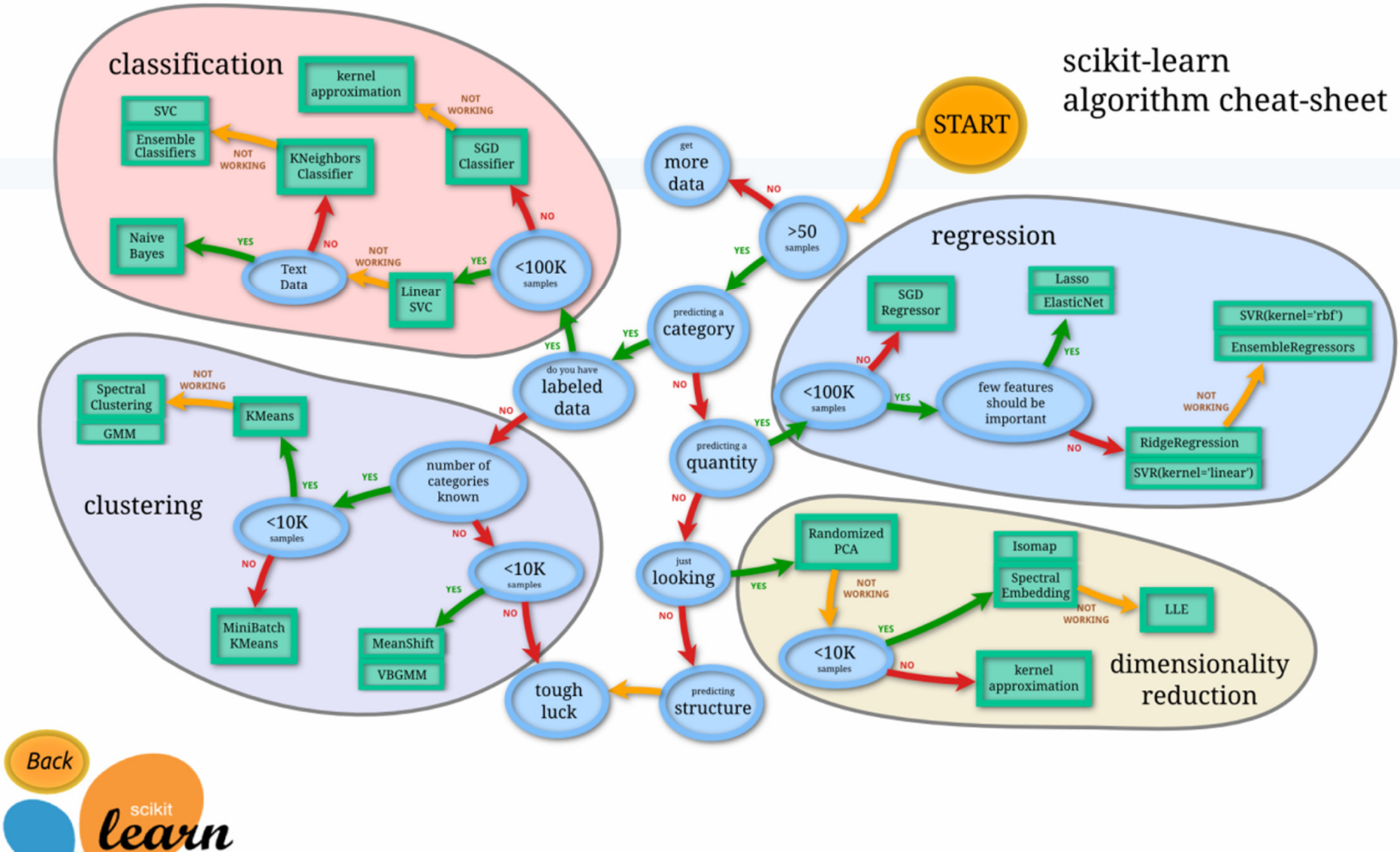
Tech. Overview



Steps in developing a machine learning application

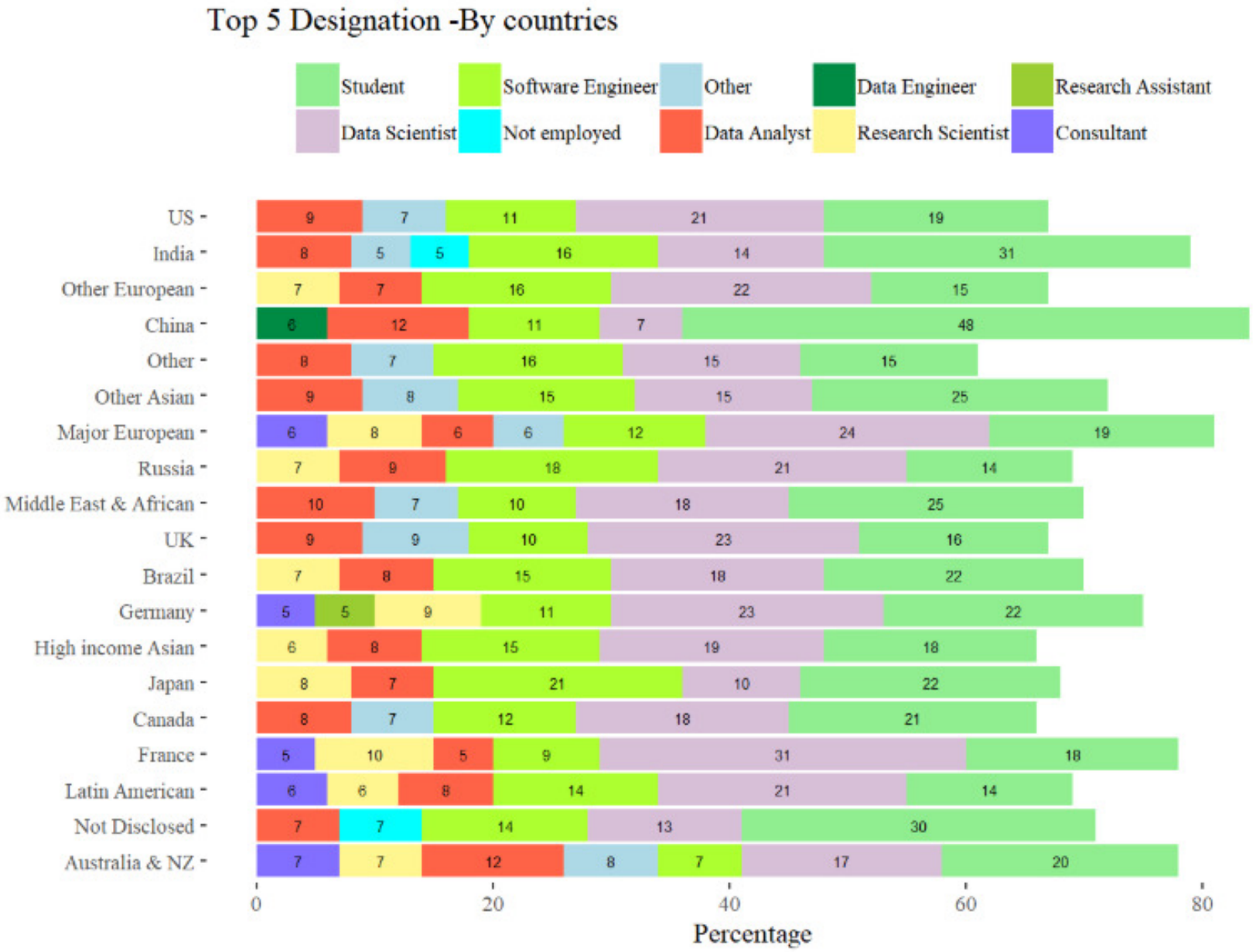
1. Collect data.
2. Prepare the input data.
3. Analyze the input data.
4. Filter garbage
5. Train the algorithm.
6. Test the algorithm.
7. Use it.

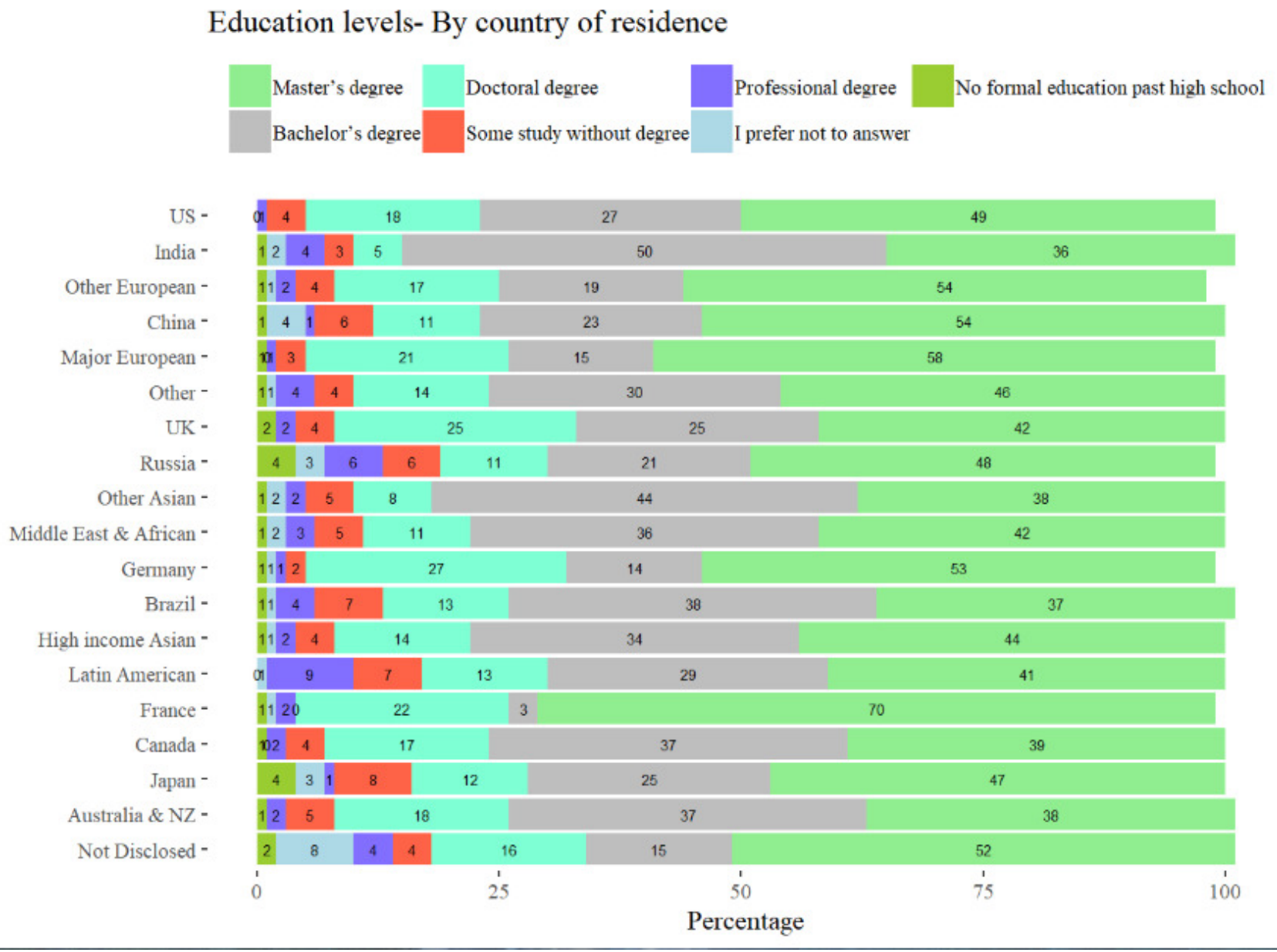

Machine Learning Algorithms Cheat Sheet
START
Unsupervised Learning: Clustering
k-means
k-modes
NO
YES
Gaussian Mixture Model
YES
Prefer Probability
NO
Categorical Variables
Hierarchical
YES
YES
DBSCAN
NO
Need to Specify k
NO
Hierarchical
Unsupervised Learning: Dimension Reduction
Dimension Reduction
YES
Topic Modeling
YES
Probabilistic
YES
Latent Dirichlet Analysis
NO
NO
NO
Principal Component Analysis
Singular Value Decomposition
NO
Have Reponses
YES
Supervised Learning: Classification
Linear SVM
NO
Data Is Too Large
NO
Explainable
SPEED
Speed or Accuracy
Naïve Bayes
YES
YES
ACCURACY
Naïve Bayes
Decision Tree
Logistic Regression
Kernel SVM
Random Forest
Neural Network
Gradient Boosting Tree
NO
Predicting Numeric
YES
Supervised Learning: Regression
Speed or Accuracy
SPEED
Decision Tree
Linear Regression
ACCURACY
Random Forest
Neural Network
Gradient Boosting Tree


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

State of art

- Global report on state of Data Science & Machine Learning-2018 Based on Kaggle Survey
  - https://rpubs.com/cvrajesh/kagglesurvey2018

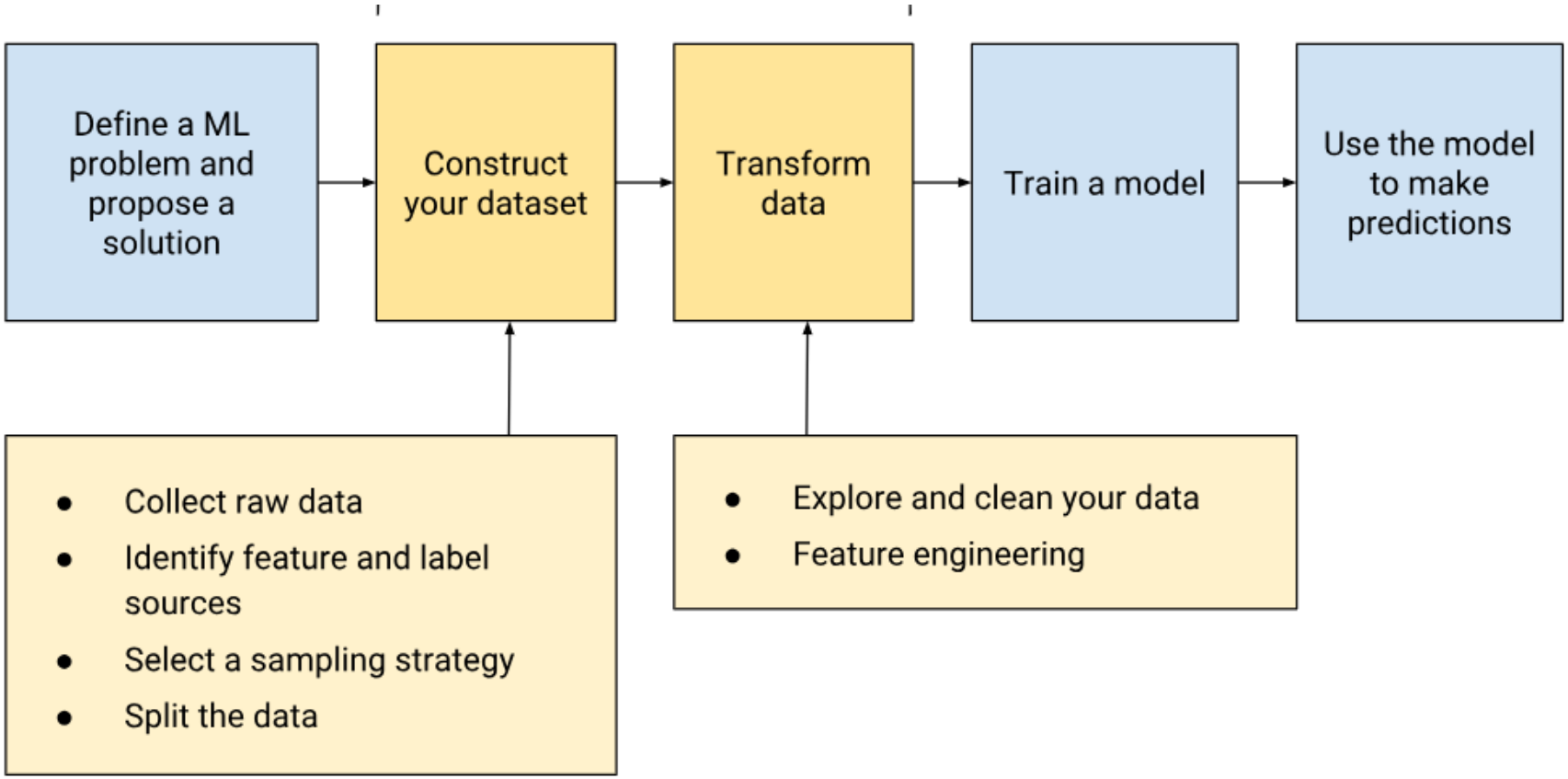

Top 5 Designation -By countries
Student
Software Engineer
Other
Data Engineer
Research Assistant
Data Scientist
Not employed
Data Analyst
Research Scientist
Consultant
US
India
Other European
China
Other
Other Asian
Major European
Russia
Middle East & African
UK
Brazil
Germany
High income Asian
Japan
Canada
France
Latin American
Not Disclosed
Australia & NZ
0
20
40
60
80
Percentage


Education levels- By country of residence
Master's degree
Doctoral degree
Professional degree
No formal education past high school
Bachelor's degree
Some study without degree
I prefer not to answer
US
India
Other European
China
Major European
Other
UK
Russia
Other Asian
Middle East & African
Germany
Brazil
High income Asian
Latin American
France
Canada
Japan
Australia & NZ
Not Disclosed
0
25
50
75
100
Percentage

Data

Data Preparation (Data pre-processing)

The Process for Data Preparation and Feature Engineering



Steps to Constructing Your Dataset

- To construct your dataset (and before doing data transformation), you should:
  1. Collect the raw data.
  2. Identify feature and label sources.
  3. Select a sampling strategy.
  4. Split the data.
- These steps depend a lot on how you've framed your ML problem. Use the self-check below to refresh your memory about problem framing and to check your assumptions about data collection.

Self-check of Problem Framing and Data Collection Concepts

- You're on a brand new machine learning project, about to select your first features. How many features should you pick?
  - Pick as many features as you can, so you can start observing which features have the strongest predictive power.

Self-check of Problem Framing and Data Collection Concepts

Pick as many features as you can, so you can start observing which features have the strongest predictive power.

- Start smaller. Every new feature adds a new dimension to your training data set. When the dimensionality increases, the volume of the space increases so fast that the available training data become sparse. The sparser your data, the harder it is for a model to learn the relationship between the features that actually matter and the label. This phenomenon is called "the curse of dimensionality."

Pick 4-6 features that seem to have strong predictive power.

- You might eventually use this many features, but it's still better to start with fewer. Fewer features usually means fewer unnecessary complications.

Pick 1-3 features that seem to have strong predictive power.

- It's best for your data collection pipeline to start with only one or two features. This will help you confirm that the ML model works as intended. Also, when you build a baseline from a couple of features, you'll feel like you're making progress!

Self-check of Problem Framing and Data Collection Concepts

- Your friend Sam is excited about the initial results of his statistical analysis. He says that the data show a positive correlation between the number of app downloads and the number of app review impressions. But he's not sure whether they would have downloaded it anyway without seeing the review. What response would be most helpful to Sam?

Self-check of Problem Framing and Data Collection Concepts

You could run an experiment to compare the behavior of users who didn't see the review with similar users who did.

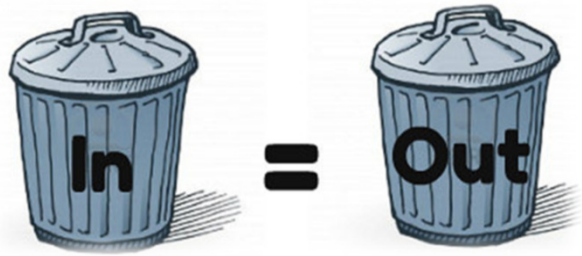
- Correct! If Sam observes that users who saw the positive review were more likely to download the app than those who didn't, then he has reasonable evidence to suggest that the positive review is encouraging people to get the app.

Trust the data. It's clear that that great review is the reason users are downloading the app.

- Incorrect. This response wouldn't lead Sam in the right direction. You can't determine causation from only observational data. Sam is seeing a correlation (that is, a statistical dependency between the numbers) that may or may not indicate causation. Don't let your analyses join the ranks of spurious correlations.

The Size and Quality of a Data Set

- "Garbage in, garbage out"



- After all, your model is only as good as your data.
- But how do you measure your data set's quality and improve it? And how much data do you need to get useful results? The answers depend on the type of problem you're solving.

The Size of a Data Set

- As a rough rule of thumb, your model should train on at least an order of magnitude more examples than trainable parameters. Simple models on large data sets generally beat fancy models on small data sets.

Data set	Size (number of examples)
Iris flower data set	150 (total set)
MovieLens (the 20M data set)	20,000,263 (total set)
Google Gmail SmartReply	238,000,000 (training set)
Google Books Ngram	468,000,000,000 (total set)
Google Translate	trillions

The Quality of a Data Set

- It's no use having a lot of data if it's bad data; quality matters, too.
  - But what counts as "quality"?
    - It's a fuzzy term.
- Consider taking an empirical approach and picking the option that produces the best outcome. With that mindset, a quality data set is one that lets you succeed with the business problem you care about. In other words, the data is *good* if it accomplishes its intended task.

The Quality of a Data Set

- However, while collecting data, it's helpful to have a more concrete definition of quality. Certain aspects of quality tend to correspond to better-performing models:
  - reliability
  - feature representation
  - minimizing skew

Reliability

- **Reliability** refers to the degree to which you can *trust* your data. A model trained on a reliable data set is more likely to yield useful predictions than a model trained on unreliable data. In measuring reliability, you must determine:
  1. How common are label errors? For example, if your data is labeled by humans, sometimes humans make mistakes.
  2. Are your features noisy? For example, GPS measurements fluctuate. Some noise is okay. You'll never purge your data set of *all* noise. You can collect more examples too.
  3. Is the data properly filtered for your problem? For example, should your data set include search queries from bots? If you're building a spam-detection system, then likely the answer is yes, but if you're trying to improve search results for humans, then no.

Reliability

- What makes data unreliable?
- many examples in data sets are unreliable due to one or more of the following:
  - Omitted values. For instance, a person forgot to enter a value for a house's age.
  - Duplicate examples. For example, a server mistakenly uploaded the same logs twice.
  - Bad labels. For instance, a person mislabeled a picture of an oak tree as a maple.
  - Bad feature values. For example, someone typed an extra digit, or a thermometer was left out in the sun.

Feature Representation

- Representation is the mapping of data to useful features. You'll want to consider the following questions:
  - How is data shown to the model?
  - Should you normalize numeric values?
  - How should you handle outliers?

Training versus Prediction

- Let's say you get great results offline. Then in your live experiment, those results don't hold up. What could be happening?
- This problem suggests training/serving skew—that is, different results are computed for your metrics at training time vs. serving time.
- Causes of skew can be subtle but have deadly effects on your results. Always consider what data is available to your model at prediction time. During training, use only the features that you'll have available in serving, and make sure your training set is representative of your serving traffic.

Identifying Labels and Sources

Direct vs. Derived Labels

- Machine learning is easier when your labels are well-defined. The best label is a **direct label** of what you want to predict. For example, if you want to predict whether a user is a Taylor Swift fan, a direct label would be "User is a Taylor Swift fan."
- A simpler test of fanhood might be whether the user has watched a Taylor Swift video on YouTube. The label "user has watched a Taylor Swift video on YouTube" is a **derived label** because it does not directly measure what you want to predict. Is this derived label a reliable indicator that the user likes Taylor Swift? Your model will only be as good as the connection between your derived label and your desired prediction.

Label Sources

- The output of your model could be either an Event or an Attribute. This results in the following two types of labels:
- **Direct label for Events,** such as “Did the user click the top search result?”
- **Direct label for Attributes,** such as “Will the advertiser spend more than $X in the next week?”

Why Use Human Labeled Data?

- There are advantages and disadvantages to using human-labeled data.
- +
  - Human raters can perform a wide range of tasks.
  - The data forces you to have a clear problem definition.
- -
  - The data is expensive for certain domains.
  - Good data typically requires multiple iterations.

Sampling and Splitting Data

- It's often a struggle to gather enough data for a machine learning project. Sometimes, however, there is *too much* data, and you must select a subset of examples for training.

Sampling and Splitting Data

- How do you select that subset?
- As an example, consider Google Search. At what granularity would you sample its massive amounts of data? Would you use random queries? Random sessions? Random users?

Sampling and Splitting Data

- Ultimately, the answer depends on the problem: what do we want to predict, and what features do we want?
- To use the feature *previous query*, you need to sample at the session level, because sessions contain a sequence of queries.
- To use the feature *user behavior from previous days*, you need to sample at the user level.

Imbalanced Data

- A classification data set with skewed class proportions is called **imbalanced**. Classes that make up a large proportion of the data set are called **majority classes**. Those that make up a smaller proportion are **minority classes**.

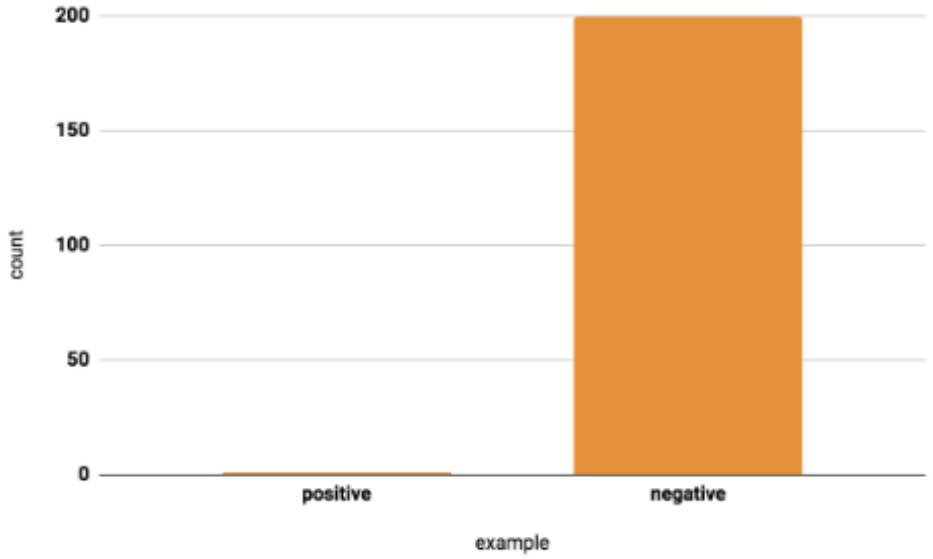
Imbalanced Data

- What counts as imbalanced? The answer could range from mild to extreme, as the table below shows.

Degree of imbalance	Proportion of Minority Class
Mild	20-40% of the data set
Moderate	1-20% of the data set
Extreme	<1% of the data set

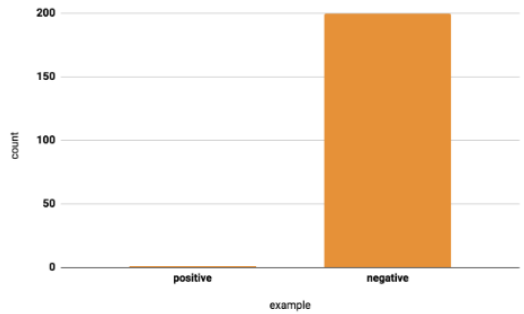
Why look out for imbalanced data?

- Consider the following example of a model that detects fraud. Instances of fraud happen once per 200 transactions in this data set, so in the true distribution, about 0.5% of the data is positive.



Why look out for imbalanced data?

- Why would this be problematic?
- With so few positives relative to negatives, the training model will spend most of its time on negative examples and not learn enough from positive ones.
- For example, if your batch size is 128, many batches will have no positive examples, so the gradients will be less informative.



Why look out for imbalanced data?

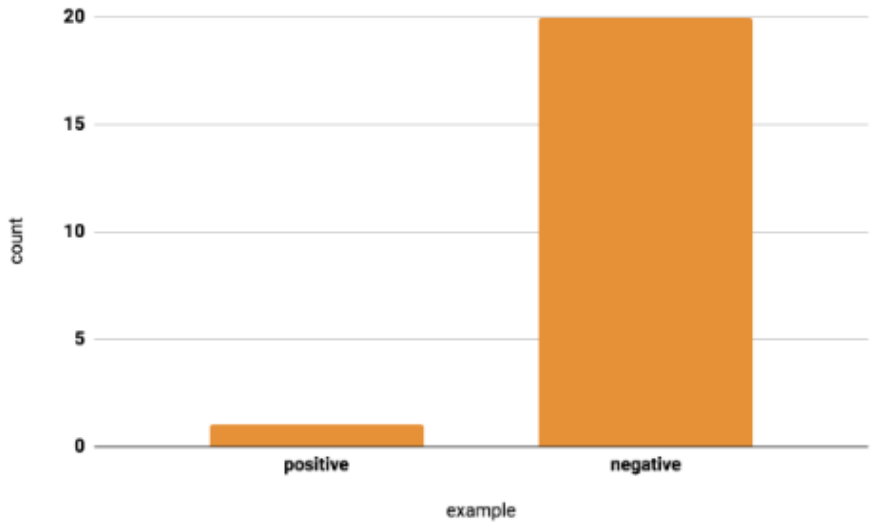
- If you have an imbalanced data set, **first try training on the true distribution.** If the model works well and generalizes, you're done! If not, try the following downsampling and upweighting technique.

Downsampling and Upweighting

- An effective way to handle imbalanced data is to downsample and upweight the majority class. Let's start by defining those two new terms:
  - **Downsampling** (in this context) means training on a disproportionately low subset of the majority class examples.
  - **Upweighting** means adding an example weight to the downsampled class equal to the factor by which you downsampled.

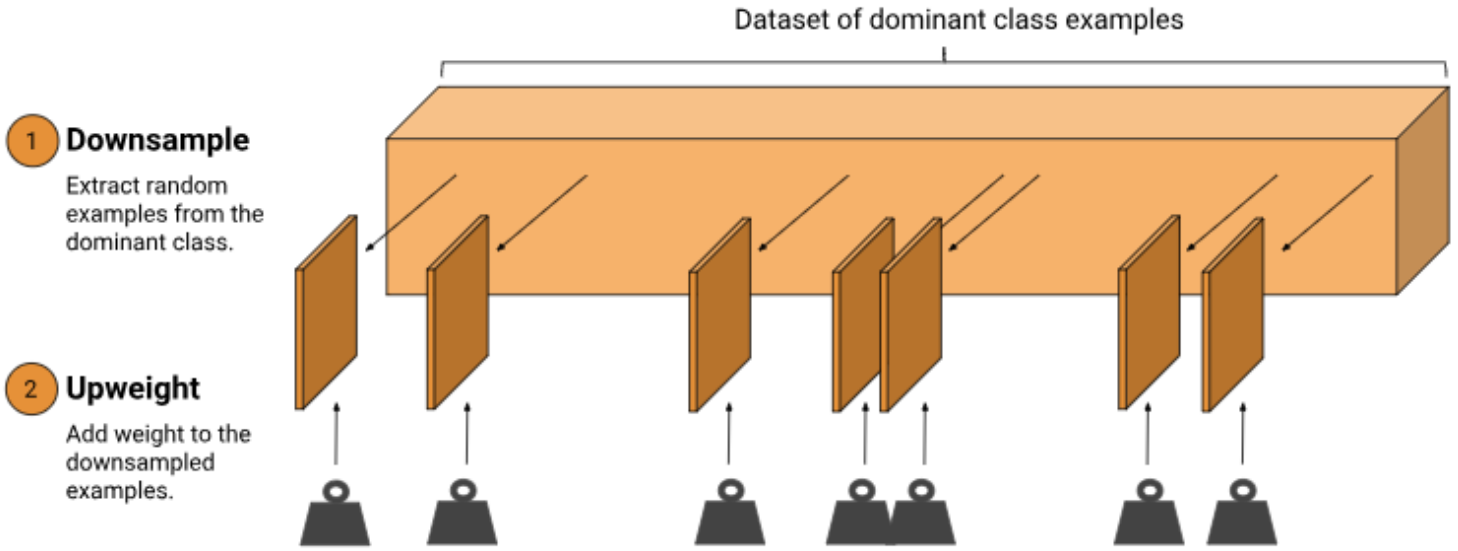
Downsample the majority class

- **Step 1: Downsample the majority class.** Consider again our example of the fraud data set, with 1 positive to 200 negatives. We can downsample by a factor of 20, taking 1/10 negatives. Now about 10% of our data is positive, which will be much better for training our model.



Upweight the downsampled class

- **Step 2: Upweight the downsampled class:** The last step is to add example weights to the downsampled class. Since we downsampled by a factor of 20, the example weight should be 20.



weights

- Here we're talking about *example weights*, which means counting an individual example more importantly during training. An example weight of 10 means the model treats the example as 10 times as important (when computing loss) as it would an example of weight 1.
- The weight should be equal to the factor you used to downsample:

$$\{\textit{example weight}\} = \{\textit{original example weight}\} \times \{\textit{downsampling factor}\}$$

Why Downsample and Upweight?

- It may seem odd to add example weights after downsampling. We were trying to make our model improve on the minority class -- why would we upweight the majority? These are the resulting changes:
  - **Faster convergence**: During training, we see the minority class more often, which will help the model converge faster.
  - **Disk space**: By consolidating the majority class into fewer examples with larger weights, we spend less disk space storing them. This savings allows more disk space for the minority class, so we can collect a greater number and a wider range of examples from that class.
  - **Calibration**: Upweighting ensures our model is still calibrated; the outputs can still be interpreted as probabilities.

Data Split Example

- After collecting your data and sampling where needed, the next step is to split your data into **training sets**,**validation sets**, and **testing sets**.

When Random Splitting isn't the Best Approach

- While random splitting is the best approach for many ML problems, it isn't always the right solution. For example, consider data sets in which the examples are naturally clustered into similar examples.

When Random Splitting isn't the Best Approach

Figure 1. News Stories are Clustered.

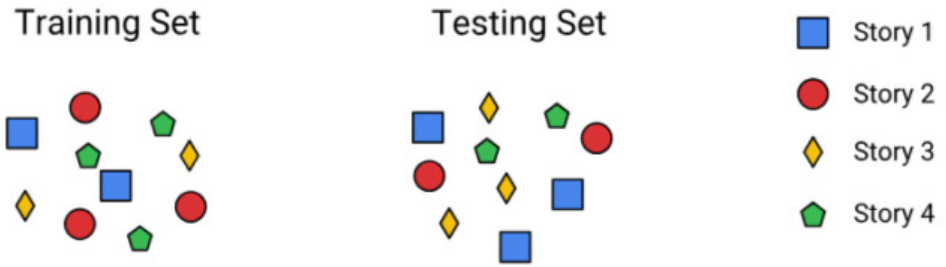


Figure 2. A random split will split a cluster across sets, causing skew.

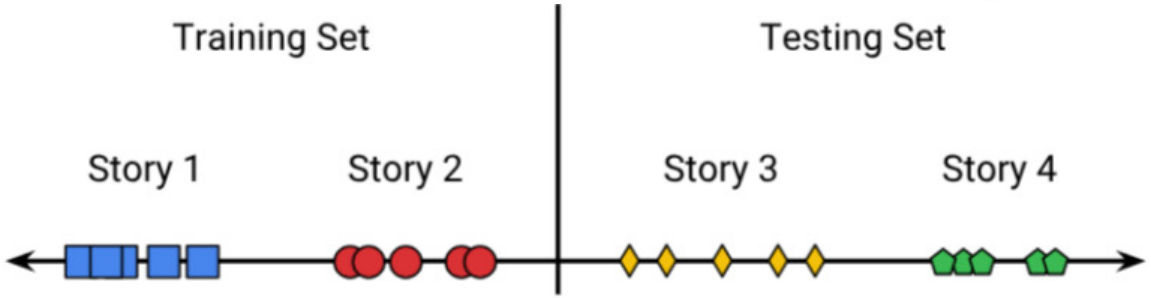


Figure 3. Splitting on time allows the clusters to mostly end up in the same set.

Why Our model perform poor sometime?

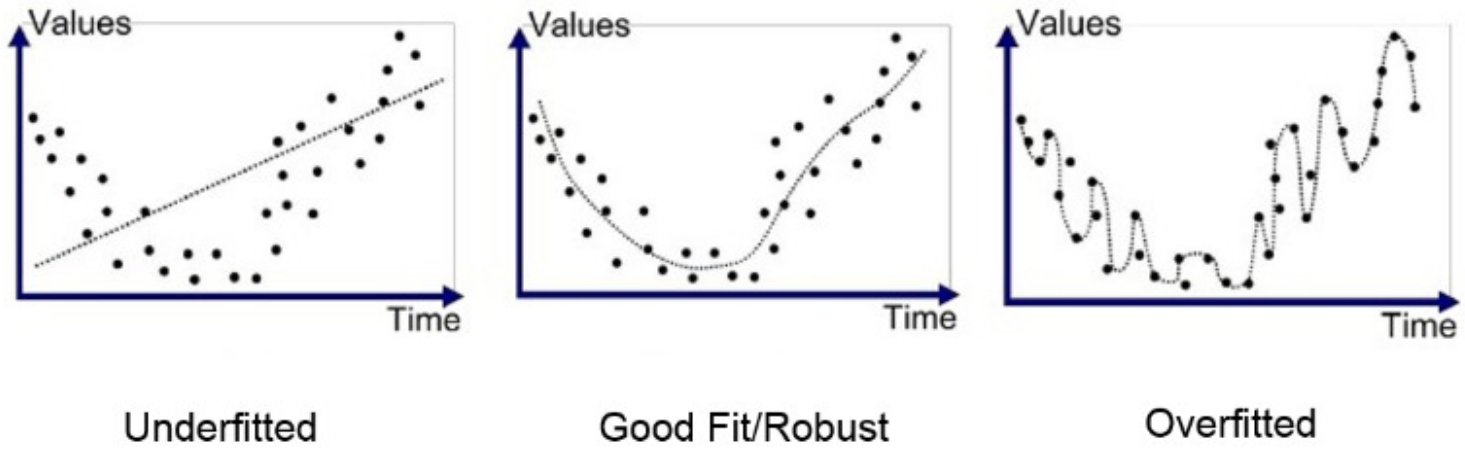
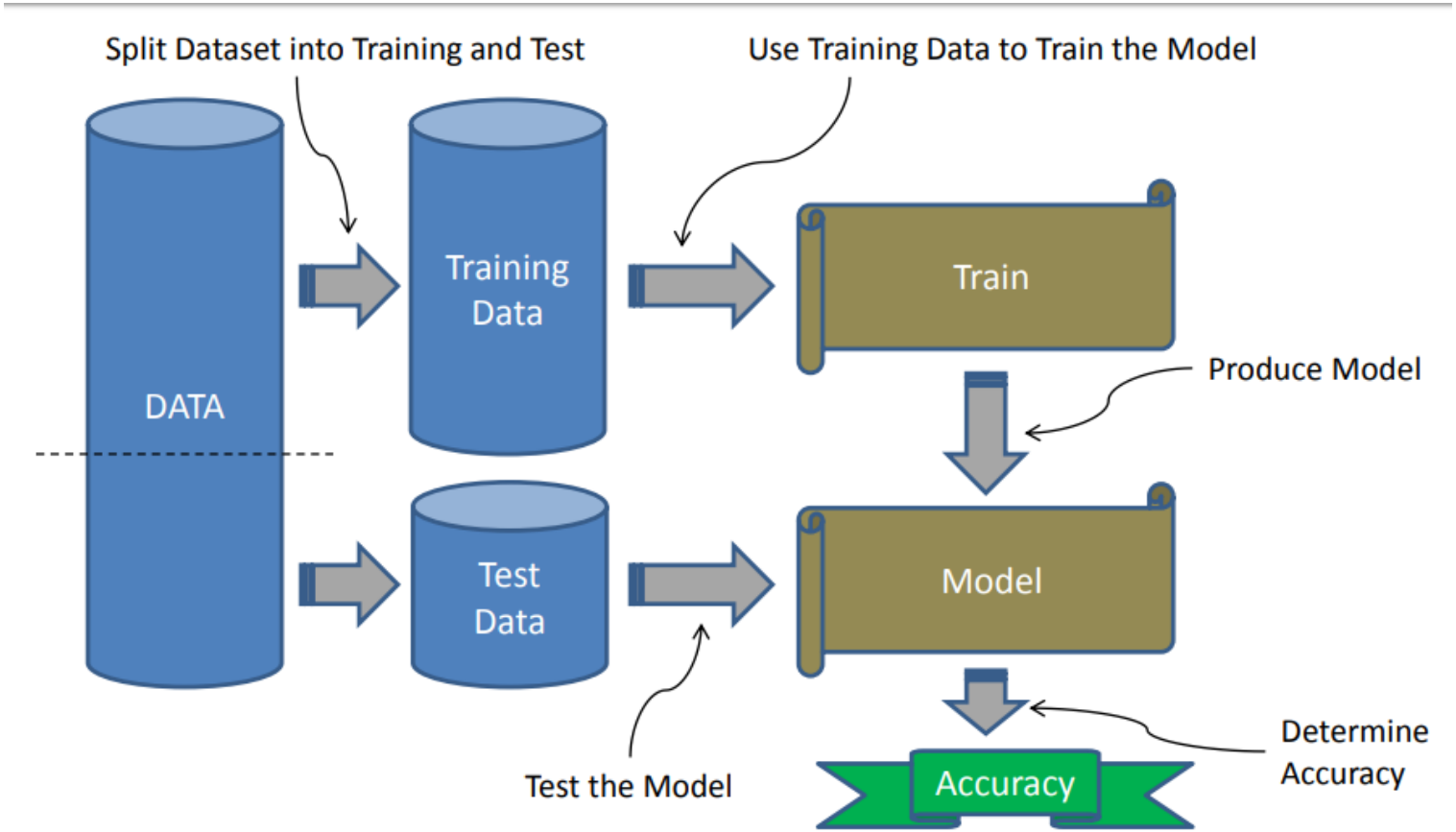


Image Source: *http://blog.algotrading101.com/design-theories/what-is-curve-fitting-overfitting-in-trading/*

Recall …


Split Dataset into Training and Test
Use Training Data to Train the Model
DATA
Training Data
Train
Produce Model
Test Data
Model
Test the Model
Accuracy
Determine Accuracy

imbalanced data - overfitting

- If the training data is overly unbalanced, then the model will predict a non-meaningful result.
  - For example, if the model is a binary classifier (e.g., cat vs. dog), and nearly all the samples are of the same label (e.g., cat), then the model will simply learn that everything is a that label (cat).
- This is called overfitting. To prevent overfitting, there needs to be a fairly equal distribution of training samples for each classification, or range if label is a real value.

Overfitting

- In data science courses, an overfit model is explained as having high variance and low bias on the training set which leads to poor generalization on new testing data.

How To Limit Overfitting

- Both overfitting and underfitting can lead to poor model performance. But by far the most common problem in applied machine learning is overfitting.
- Overfitting is such a problem because the evaluation of machine learning algorithms on training data is different from the evaluation we actually care the most about, namely how well the algorithm performs on unseen data.

How To Limit Overfitting

- There are two important techniques that you can use when evaluating machine learning algorithms to limit overfitting:
  1. Use a resampling technique to estimate model accuracy.
  2. Hold back a validation dataset.

Underfitting

- Underfitting refers to a model that can neither model the training data nor generalize to new data.
- An underfit machine learning model is not a suitable model and will be obvious as it will have poor performance on the training data.

Basic

- Data science may seem complex but it is really built out of a series of basic building blocks:
  - **Overfitting:** too much reliance on the training data
  - **Underfitting:** a failure to learn the relationships in the training data
  - **High Variance:** model changes significantly based on training data
  - **High Bias:** assumptions about model lead to ignoring training data
  - Overfitting and underfitting cause poor **generalization** on the test set
  - A **validation set** for model tuning can prevent under and overfitting
  - …

Web Scraping

API

- **Beautiful Soup**
  - is a tool which help programmer quickly extract valid data from web pages
- **Scrapy**
  - is a web crawling framework

Example