Machine Learning

A few notes from the battlefield

Paris JUG
November 12th, 2013

Thomas Cabrol
Chief Data Scientist at Dataiku
@ThomasCabrol

Data Scientist is the sexiest job of the 21st century ©

??

You'll use Machine Learning to solve cool problems and build (new) products like...

- Find groups of customers who share the same behavior
- Build cross/up sell predictive models
- Reduce churn
- Build recommendation engine
- Personalize search
- Optimize ad placements
- Create credit scoring engines
- Detect fraud
- Predict machine failures
- Design new drugs and look for new cures
- Build Siri or Watson !
- Create self-driving cars !!
- Uncover galaxies !!!

And an endless list of applications with virtually all fields !

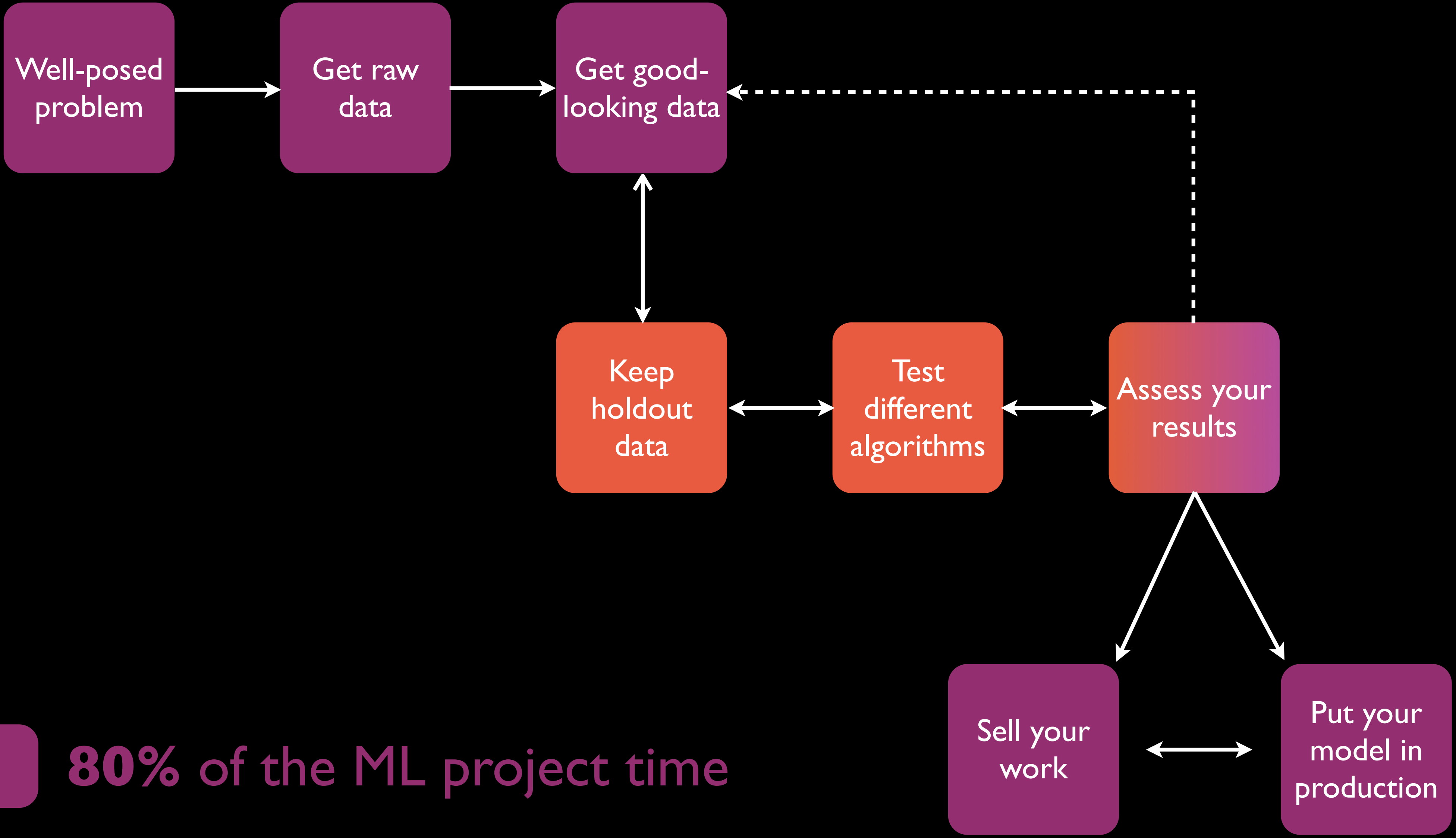
They all share the same idea

Make our computers learn patterns from known data and build systems that can automatically decide what to do with new data.

Machine Learning tasks often fall within one of these 2 categories

- **Supervised** machine learning
  - model p(y|x) => given a set of «features», learn to predict the value of x (i.e given the size, location and number of rooms of a flat, predict its price)
  - Classification is for the case when you want to predict a discrete value (Yes/No), and Regression when you want to precit a numeric value.
- **Unsupervised** machine learning
  - model p(x) => there is no labeled data to predict, only a set of features from which you want to uncover underlying structure
  - Clustering (find groups of observations that share similar patterns) dimensionality reduction (reduce the set of initial features into a smaller subset of components)

So what does it look like in practice ?



🟪 **80%** of the ML project time

🟧 **20%** of the ML project time

Well-posed problem

- This is tough, and highly important for the following steps, including the type of approach to use (supervised / unsupervised).

- You need to get to a very precise question you want to adress using ML
  - otherwise you'll never know or you'll never be able to say STOP

- Unless you are fully autonomous on your project, make sure you have everyone on-board at this step (your boss, your colleagues, the Other Team, your client...)

Get raw data

- You know it, they can be anywhere and issues can be endless
  - in poorly formatted logs on cloud FS
  - in a «limited access» drive or directory
  - in a slave (or better production) SQL DB that doesn't like to be queried
  - in a KV or document store
  - in HDFS
  - in an Excel spreadsheet, hopefully in a wonderful format (merged / hidden cells, pivoted columns, split tables... Open data anyone ?)
  - at the end of an undocumented API
- Just don't underestimate this step, this can be tricky and time-consuming
  - Usually, start with what you have at hand first and add more data later

Create good-looking data

- Most of the time you'll need to build a fully denormalized, «flat» data structure to run the ML algorithms

User ID	User Age	User Email Domain	User Postal Code	Postal Code mean revenue	Nb searches last months	Nb Clicks last week	Nb Purchases lifetime	Nb Searches last day	**Did purchase last week**
1	19	Gmail.com	57280	18657.45	1	1	3	0	0
2	32	Facebook.com	6293	23952.5	39	32	8	1	1

  - a matrix storing in columns all the attributes / inputs (the Features vector) of a given record (a sample or an observation), and optionnally another column storing the value you want to predict (the Label)

- **Features Engineering is key**
  - ultimately this is what will make the difference
  - be ready to be creative
  - be ready for a time and resource consuming session (large scale joins and agregations...), this is where frameworks such as Hadoop can come into play.

Better data > clever maths ©

- Did I mention that Features Engineering is key ?
- If you miss the really important features in your dataset, you'll probably never reach your goal to build a good model, and using fancy algorithms won't help that much.
- This is where Big Data (and one of its 3, 4, more ? V's) come into play: Variety.
  - Using large datasets with granular data coming from various sources will allow us to build a richer set of features

Get to know your data

- Your dataset will be messy, so once you have a first working version, it is very important to understand it first, by:
  - creating statistics for each of your features (distribution ...)
  - creating joint statistics between each feature and the labels in case of supervised learning
  - looking at correlations between features
  - see your dataset as a whole using techniques such a PCA
  - data visualisation helps a lot at this step !

- This will also allow to understand how much messy is the data, and eventually deal with theses cases:
  - missing values
  - long-tailed distribution
  - outlying values...

And if this didn't make you nut, you can start ML

Be ready to fight overfitting

- What we are loking for is a model that has a good generalization property, i.e, how confident we are in its capacity to deal with new, unknown data.
  - A model could be very good at training time, but fail miserably to predict new observations: it overfitted (i.e it learned to well on a dataset and now is unable to predict a new one).

- Always keep at least one portion of your initial dataset in a separate sample that will not be used to train the model, but only to test its real accuracy and capacity of generalization.
  - This is **Cross Validation** !
  - Or at least just one approach to CV, since many strategies exist, sharing all the same principle of keeping an holdout sample.

Build a baseline model quick

- Go fast with the creation of your first model that will be used to benchmark your future iterations.
    - simple in terms of algorithm (it can be a simple but fast to train linear model, or even simple logic like a pure randomness in prediction)
    - simple in terms of features included

- This way you get a taste of where you are heading to, for instance in you obtain very poor performances it might be a good idea to rethink the Features Engineering process.

Try several algorithms

- There are literally hundreds of algorithms, both for supervised and unsupervised, and (I guess) nobody has yet the answer to «what is the best algorithm ? »
  - it is highly dependant on your dataset
  - but be aware that each method has its own limitations
- A bit of name dropping:

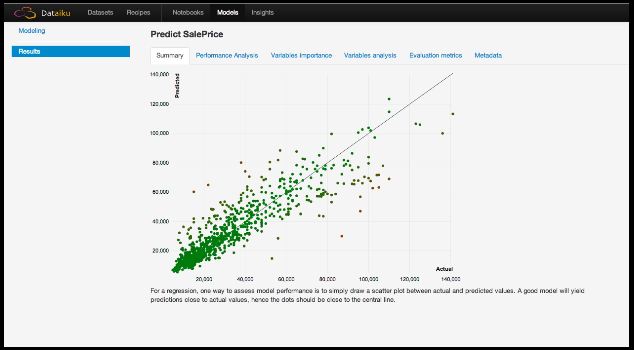
Unsupervised	Supervised
K-Means Hierarchical Clustering Gaussian Mixtures DBSCAN Mean Shift...	Linear Models (OLS, Logistic, Ridge, Lasso) Support Vector Machines KNN Naive Bayes Decision Trees Neural Nets Tree-based ensemble (Random Forest, Gradient Boosted Trees...)

Measure carefully your results

- Make sure you use your holdout sample to measure the accuracy of your model (for supervised learning).
- Several metrics are available and depends on your task. Common metrics for classification for instance:

		Predicted	
		Negative	Positive
Actual	Negative	TN	FP
	Positive	FN	TP

  - Accuracy: % of correct predictions (TN + TP) / size of the sample
  - Recall: % of positive cases catched (TP / (TP+FN)) and Precision: % of positive cases correct (TP / (TP +FP))
  - Beware, that can be highly misleading for unbalanced samples ! (see other methods like ROC curve or gain charts)
- You may also want to use graphical outputs to assess your model performance



Predict SalePrice

For a regression, one way to assess model performance is to simply draw a scatter plot between actual and predicted values. A good model will yield predictions close to actual values, hence the dots should be close to the central line.

Don’t reinvent the wheel (at first)

- You can use existing software, at least to begin with, when doing a machine learning project to get faster to results.
  - The most common language is R, and while the learning curve might be steep and the performances not very high (but that highly depends on implementations), basically all methods are ported to R and there are several nice plotting libraries.
  - Python is also gaining momentum, notably because of the very nice Scikit-learn library.
- When the scale of your data requires it, you can use MR based frameworks such as Mahout (Hadoop) or MLBase (Spark), or deploy the algorithms within your DB if it supports it (MADLib, or dedicated SDK’s for MPP db’s - Vertica, Greenplum, Netezza...)
  - but are your sure you really need a distributed version of the algorithm ?

Communicate the results

- This is especially true when you're not building a new piece of software, but rather a prelimiary study or proof-of-concept.

- You'll need to get the buy-in of «business» users by :
  - share frequently the results of your iterations on the model
  - show visual outputs
  - be transparent with the assumptions you made (data used, features build, models used...)
  - build simple apps to let the users «play» with your model themselves, if applicable
  - assess or help them assess the potential impact on the business (some sense of ROI)

- Don't forget that correlations do not imply causation

Productionalize

- Cool, your ML project has been adopted !

- Now what ?

  - I hope you didn't forget to serialize your model somewhere or to make it reusable !
  - Not as easy as it seems: how will it be used ? Real-time (aka a production application gets live predictions from an API), batch (you can produce the predictions every month/week/hour... for the whole base) ?
  - You'll need to be able to reproduce the whole modeling steps, including the preprocessing steps to make sure each new «scored» observation is inline with your ML project.
  - You'll probably need to keep track and monitor how the model behaves over time, and know when it's time to retrain it again.

Data Scientist is the sexiest job of the 21st century ©

Yes, but that's no free lunch :)

Dataiku

http://dataiku.com/

contact@dataiku.com