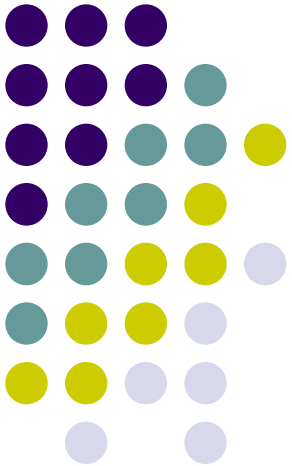
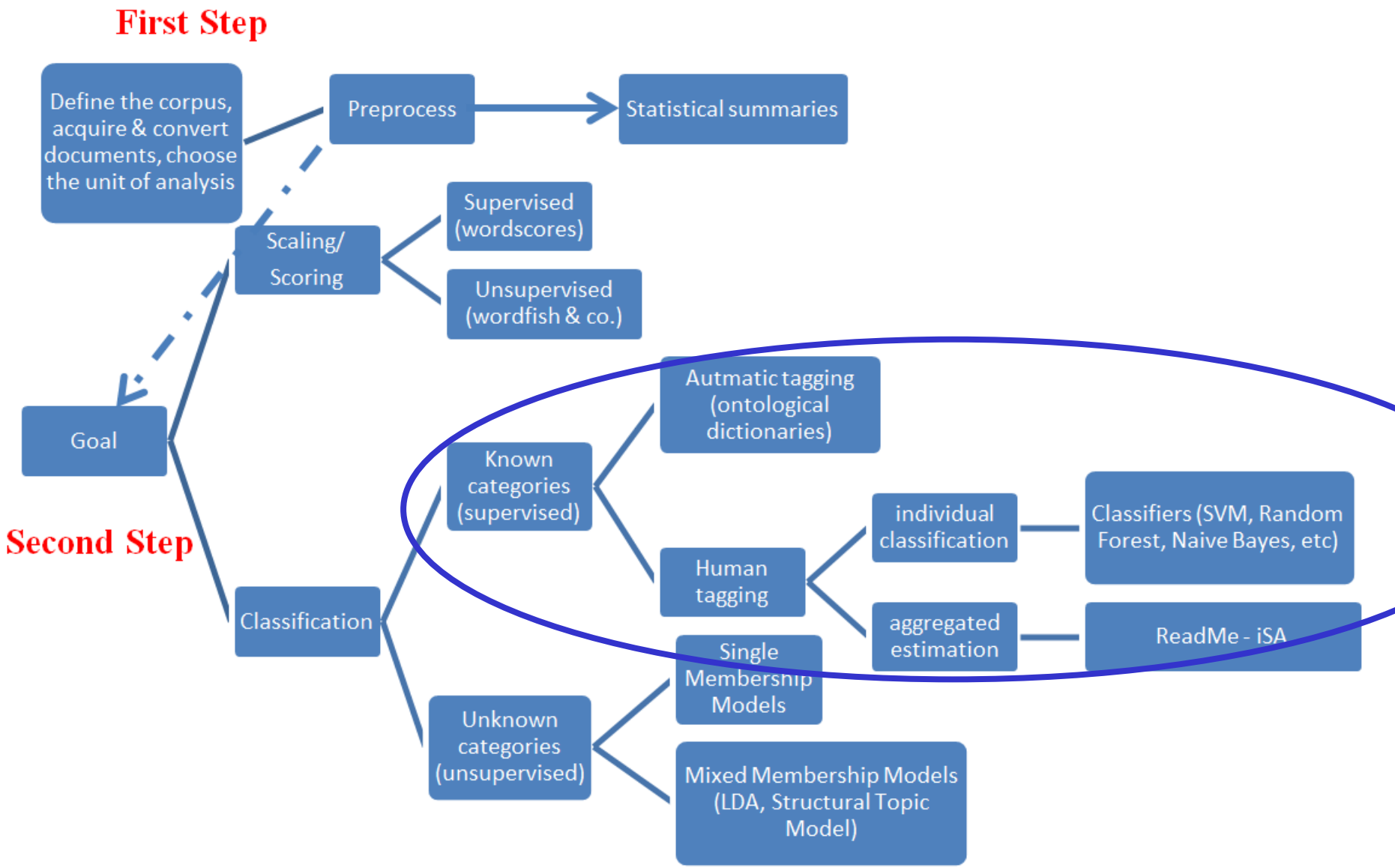
Applied Scaling & Classification Techniques in Political Science

Lecture 8

Cross-Validation



Our Course Map



Reference

- ✓ Grimmer, Justin, and Stewart, Brandon M. (2013). Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts. *Political Analysis*, 21(3): 267-297
- ✓ Curini, Luigi, and Robert Fahey (2020). Sentiment Analysis and Social Media. In Luigi Curini and Robert Franzese (eds.), *SAGE Handbook of Research Methods is Political Science & International Relations*, London, Sage, chapter 29

Validation


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Validation

Once labels are applied to the training set and a supervised learning method generalizes from this training set to the test set, only **validation** of the model remains

Supervised methods are designed to automate the hand coding of documents into categories or measuring the proportion of documents in categories as we have already noticed

If a method is performing well, it will **directly replicate the hand coding**. If it performs poorly, it will fail to replicate the coding – instead introducing serious errors

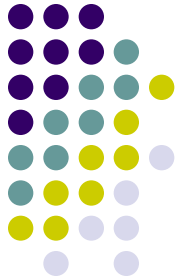
This clear objective implies a **clear standard for evaluation**: comparing the output of machine coding to the output of hand coding

Validation

The *ideal validation* procedure would divide the data into **three subsets**

1. Initial model fitting would be performed on the training-set
2. Once a final model is chosen, *a second set of hand-coded* documents - the validation set - would be used to assess the performance of the model
3. The final model would then be applied to the test to complete the classification

Validation



This approach to validation is difficult to apply in most settings. But **cross-validation** (also called: **K-fold validation**) can be used to replicate this ideal procedure

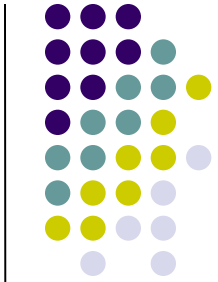
In K-fold cross-validation, the training set is randomly partitioned into some groups (say two: K1 and K2)

Validation

For each group, the first model is trained on K1, then applied to the K2 to assess performance; similarly a model is trained on the K2 and then applied to K1 to assess performance

Then you take the average across the results you get in the two scenarios

Validation



And if you want to run a K-fold cross-validation with K larger than 2?

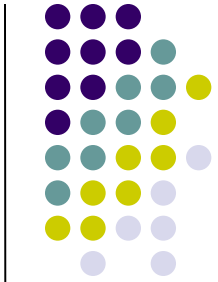
The algorithm is as follow:

1. Randomly split the data set into k-subsets (or k-fold) (for example 5 subsets)
2. Reserve one subset and train the model on all other subsets
3. Test the model on the reserved subset and record the prediction error
4. Repeat this process until each of the k subsets has served as the test set
5. Compute the average of the k recorded errors. This is called the cross-validation error serving as the performance metric for the model

Validation

So, for example, with K-fold cross-validation=5…

Validation



How to choose the “correct” value of k?

Lower value of K is more biased and hence undesirable. On the other hand, higher value of K is less biased, but can suffer from large variability

In practice, one typically performs k-fold cross-validation using k = 5 or k = 10, as these values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance

Validation

When you run a ML algorithm on the test-set, there are no available statistics to control for the **goodness-of-fit of your prediction** (by definition the “true” values of the test-set are unknown!)

That is why **cross-validation** is so important! This is the **only way** to control if the ML algorithm you are using is doing a good job or not (unless you are ready to believe in that by fiat)!

Moreover, cross-validation **avoids overfitting** by focusing on **out-of-sample prediction** and **selects the best model** for the underlying data from a set of candidate models

Validation

Which statistics (or **performance metrics**) should we us to assess model performance?

There are several of them, but we are going to focus on three metrics for individual classifiers with text-analysis

Validation

Accuracy: proportion of correctly classified documents

While of course we want this score to be as high as possible, it can also be important to look at the two components which make up that score, known as **recall** and **precision**

Validation

Recall or **Sensitivity** (for a category k) is a measure of what proportion of instances of a given category the algorithm correctly identified

For example, if there were 10 instances of the category "positive" in the data set, and the algorithm correctly identified 8 of them, we would say that this algorithm has "recall of 0.8 for the category *positive*" (i.e., given that a human coder labels a document as belonging to category k, what is the chance the machine identifies the document?)

Validation

Precision or **Positive Predictive Value** (for a category k) on the other hand is a measure of how many of the times the algorithm identified a category were actually correct, as against how many times were false positives

In the above example, where the algorithm correctly identified 8 of the 10 instances of *positive*, perhaps the algorithm also miss-identified 4 other documents as *positive* - so 8 out of its 12 *positive* classifications were correct, allowing us to say that it has a "precision of 0.667 for the category *positive*" (i.e., given that the machine guesses category k, what is the probability that the machine made the right guess?)

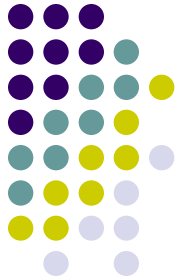
Validation

Finding market differences between **recall and precision** (for example, with a recall rate >> precision) implies that your algorithm guesses too often that a document belongs to category k

The result is that it labels a large portion of the human coder's as *k* correctly (and so you have a **high recall rate**). But it also includes several documents that humans label differently (and so you have a **low precision**)

This sometimes applies when the original *k* category, compared to the other k_{n-1} categories, is (by far) the most relevant category in the training-set (remember our previous discussion on ML vs. Proportional Algorithm…)

Validation



The aggregate of the recall and precision scores for a category is known as the **f1 score**

More precisely, the traditional F-measure or balanced F-score (**f1**) is the harmonic mean of precision and recall:

f1 <- (2 * precision * recall) / (precision + recall)…

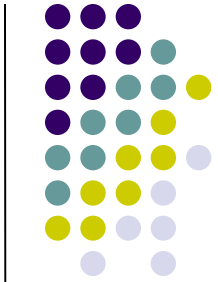
…where the highest level of performance is equal to 1 and the lowest 0

Validation

The **average of the f1 scores for all the categories** is a reasonable rough measurement of the performance of the algorithm (more than accuracy alone!)

However, before using the algorithm for any serious analysis work, it is advisable also to take a look at the precision and recall scores for individual categories - you may find that a category you are planning to use in your analysis actually has very high rate of false-positive or false-negative identifications, which could cause serious problems for your results

Performance metrics



Confusion matrix

	Actual label	
Classification (algorithm)	Black	White
Black	True Black	False Black
White	False White	True White

$$\text{Accuracy} = \frac{\text{TrueBlack} + \text{TrueWhite}}{\text{TrueBlack} + \text{TrueWhite} + \text{FalseBlack} + \text{FalseWhite}}$$

$$\text{Precision}_{\text{Black}} = \frac{\text{TrueB}}{\text{TrueB} + \text{FalseB}}$$

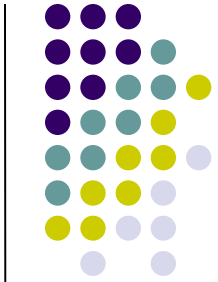
Think horizontally!

$$\text{Recall}_{\text{Black}} = \frac{\text{TrueB}}{\text{TrueB} + \text{FalseW}}$$

Think vertically!

$$f_{1\text{Black}} = \frac{2 * \text{precisionB} * \text{recallB}}{\text{precisionB} + \text{recallB}}$$

Performance metrics: an example



	Actual label	
Classification (algorithm)	Black	White
Black	800	100
White	50	50

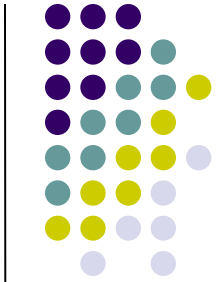
$$\text{Accuracy} = \frac{800 + 50}{800 + 50 + 100 + 50} = 0.85$$

$$\text{Precision}_{\text{Black}} = \frac{800}{800 + 100} = 0.88$$

$$\text{Recall}_{\text{Black}} = \frac{800}{800 + 50} = 0.94$$

$$f_{1\ \text{Black}} = \frac{2 * .88 * .94}{.88 + .94} = 0.91$$

Performance metrics



Confusion matrix

	Actual label	
Classification (algorithm)	Black	White
Black	True Black	False Black
White	False White	True White

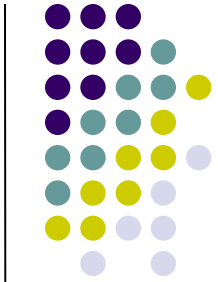
$$\text{Accuracy} = \frac{\text{TrueBlack} + \text{TrueWhite}}{\text{TrueBlack} + \text{TrueWhite} + \text{FalseBlack} + \text{FalseWhite}}$$

$$\text{Precision}_{\text{White}} = \frac{\text{TrueW}}{\text{TrueW} + \text{FalseW}}$$ **Think horizontally!**

$$\text{Recall}_{\text{White}} = \frac{\text{TrueW}}{\text{TrueW} + \text{FalseB}}$$ **Think vertically!**

$$f_{1\text{White}} = \frac{2 * \text{precisionW} * \text{recallW}}{\text{precisionW} + \text{recallW}}$$

Performance metrics: an example



	Actual label	
Classification (algorithm)	Black	White
Black	800	100
White	50	50

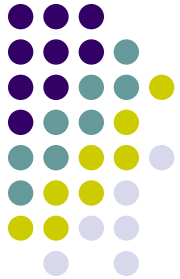
$$\text{Accuracy} = \frac{800 + 50}{800 + 50 + 100 + 50} = 0.85$$

$$\text{Precision}_{\text{White}} = \frac{50}{50 + 50} = 0.5$$

$$\text{Recall}_{\text{White}} = \frac{50}{50 + 100} = 0.33$$

$$f_{1\ \text{White}} = \frac{2 * .5 * .33}{.5 + .33} = 0.39$$

Performance metrics



In this example you have a single Accuracy value=0.85

Then you could take the average of the F1 scores for the classes as another (and more reliable) measure of the performance of the algorithm

In our case: (.91+.39)/2=.65

You see the difference here between Accuracy and the averaged F1 score!

This difference is due to the fact that we are doing quite well with the Black class, but relative poorly with the White class

Validation

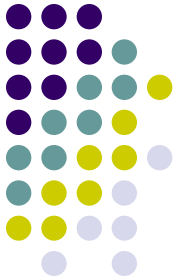
As already underlined, in some given circumstances

Accuracy is not a reliable metric for the real performance of a classifier, because it will yield misleading results if the **data set is unbalanced** (that is, when the numbers of observations in different classes vary greatly)

For example, if there were 95 cats and only 5 dogs in the data, a particular classifier **might classify all the observations as cats**

The overall accuracy would be then…how much?

Performance metrics: an example



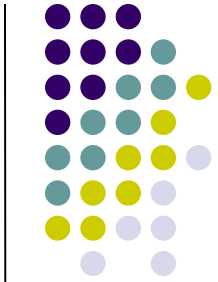
	Actual label	
Classification (algorithm)	Cats	Dogs
Cats	95	5
Dogs	0	0

$$\text{Accuracy} = \frac{95 + 0}{95 + 5 + 0 + 0} = 0.95$$

Validation

However, in this same circumstance, the classifier would have a **recall rate (sensitivity)** for the *dog class* equals to…? And for the cats equals to…?

Performance metrics: an example



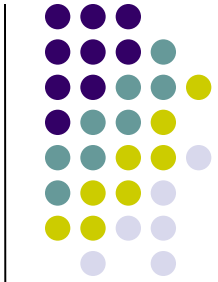
	Actual label	
Classification (algorithm)	Cats	Dogs
Cats	95	5
Dogs	0	0

$$\text{Accuracy} = \frac{95+0}{95+5+0+0} = 0.95$$

$$\text{Recall}_{\text{Dogs}} = \frac{0}{0+5} = 0$$

$$\text{Recall}_{\text{Cats}} = \frac{95}{95+0} = 1$$

Validation: another example with 3 categories



		Actual class		
		Cat	**Dog**	**Rabbit**
Predicted class	**Cat**	**5**	2	0
	Dog	3	**3**	2
	Rabbit	0	1	**11**

Here **Accuracy** is equal to the ratio between the sum of the diagonal (i.e., the sum of «True Positive») and the total number of observations, i.e., (5+3+11)/(5+2+3+3+2+1+11)=0.704

Validation

For each category *k* we can move from here

		Actual class		
		Cat	**Dog**	**Rabbit**
Predicted class	**Cat**	**5**	2	0
	Dog	3	**3**	2
	Rabbit	0	1	**11**

to here (example for the “cat” category)

		Actual class	
		Cat	**Non-cat**
Predicted class	**Cat**	5 True Positives	2 False Positives
	Non-cat	3 False Negatives	17 True Negatives

Validation

Then:

In the case, Precision for the cat class is: 5/(5+2)=0.71

Recall for the cat class is: 5/(5+3)=0.625

f1 for the cat class is: 2*(0.625*071)/(0.625+071)=0.66

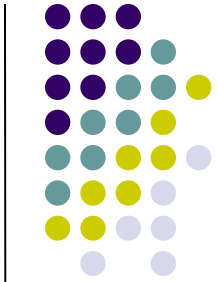
You can do the same thing for the dog and the rabbit cases, and then averaging across values to have a sense of the overall performance of your model

Validation

Depending on the application, scholars may conclude that the supervised method is able to sufficiently replicate human coders

Or, additional steps can be taken to improve accuracy, including: applying other methods…or switching the **quantity of interest to proportions**!!!!

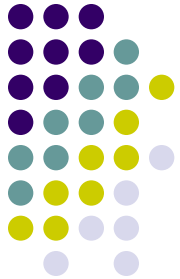
Validation



Most algorithms also have a range of "**hyper-parameters**" – assumptions and modifiers which are used to fine-tune the model and which can be set to different values prior to training – that can significantly impact performance (remember about C in SVM!)

Finding the **right set of hyper-parameters** for a certain task and a specific data set is also largely a case of trial and error, and it can only be done, once again, via cross-validation!

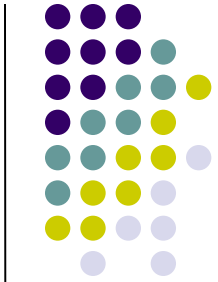
Validation



This process can take a lot of time – often in the order of several hours for algorithms with complex sets of parameters – but often yields better performance than the default parameter set

However, some packages in R (such as `Caret`) provide ways to automate this task; this is known as a "*grid search*", allowing researchers to exhaustively search through every combination of a set of hyper-parameters to find the best performing model

Validation



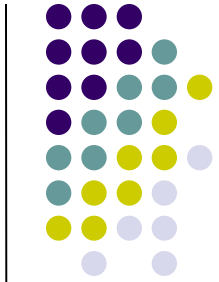
Remember: the purpose of cross-validation is **model checking**, not **model building**!

Accordingly, cross-validation allows also **to select among different machine-learning algorithms**!

Which is the machine-learning algorithms to prefer given your specific training-test?!? The one that fares better in cross-validation!!!

So trust this latter one, when you want to classify the unknown and unlabeled test-set!

Validation: a summary



Two possible routes in this regard according to how you want to deal with the **hyper-parameters**:

First route (to success…)

a) You keep the default hyper-parameters of your ML algorithms;

b) you run a CV on each of such ML algorithms

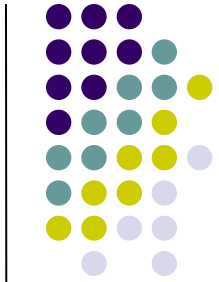
c) you select the one (or two) with the best performance

d) you fine-tune the hyper-parameters on such model(s)

e) you re-run CV just on them

f) you keep the ML algorithm that performs better in the CV

Validation: a summary



Two possible routes in this regard according to how you want to deal with the **hyper-parameters**:

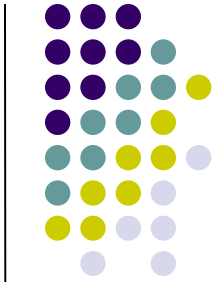
Second route (to success…)

a) You fine-tune the hyper-parameters on each of the your ML algorithms you want to test

b) you run CV on each of them

c) you keep the ML algorithm that performs better in the CV

Validation when measuring proportions

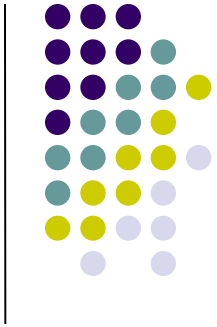


How to run a cross-validation given that you do not make any individual classification in this case?

Well, you can still run a **cross-validation** procedure on your training-set (but ONLY at the aggregate level)!

For example, what you can do is estimating the **MAE** (mean average error) across categories

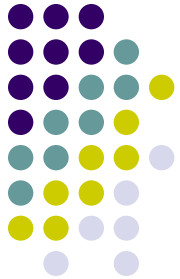
Validation when measuring proportions: estimating MAE



		Actual class		
		Cat	**Dog**	**Rabbit**
Predicted class	**Cat**	**5**	2	0
	Dog	3	**3**	2
	Rabbit	0	1	**11**

A *True # classes*	*B* *% True*	*C* *Predicted # classes*	*D* *% Predicted*	*Absolute difference B-D*	
8	29.6%	7	25.9%	3.7%	
6	22.2%	8	29.6%	7.4%	
13	48.1%	12	44.4%	3.7%	
				4.9%	***MAE***

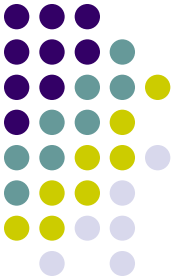
Some more words on Machine learning “fever”



Three points highly inter-connected….

FIRST: If a **huge amount of training data** are available, then the choice of classifier probably has little effect on your results, and we should be guided by the principle of **parsimony** to prefer more **transparent and simpler models** over more opaque and complex ones, even at the cost of small tradeoffs in performance

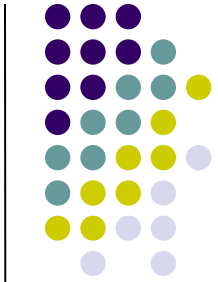
Some more words on Machine learning “fever”



SECOND: Just as our concern in social science is explanation rather than prediction, we generally prefer model specification based on theory and isolating the effect of specific explanatory factors, rather than attempting to include every possible variable to maximise variance explained

That is, while “classic” statistical methods focus primarily on theory testing and input variables, machine learning approaches are mostly concerned with explaining outputs (e.g. whether a group is considered prominent or not); which ‘leads researchers to be more concerned with **prediction accuracy** and less concerned with **explanation**’

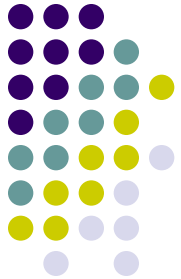
Some more words on Machine learning “fever”



This difference MATTERS a lot!

Because the goals of explanation or measurement differ from the (typical machine learning) objective of prediction, it is worth reminding ourselves of this preference

Some more words on Machine learning “fever”

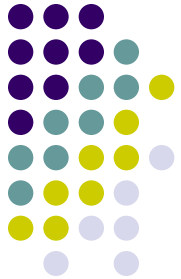


For a long time, those developing machine learning approaches sought in fact only to predict outcomes rather than understand the phenomena of interest

This is problematic for at least two reasons

First, many ML algorithms, methods, and models rely on black box techniques. As a result, interpreting the learned relationship between inputs and outcomes becomes much more difficult than with classic techniques used within political science

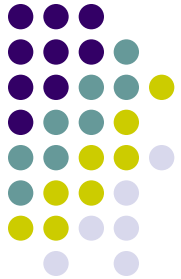
Some more words on Machine learning “fever”



For example, while SVMs are fantastic tools for accurate classification, they do not offer an easy way of determining which features are most predictive. This implies that their use for theory evaluation is extremely limited

Second, and partly as a result of the complex problems they tackle, ML approaches tend not to be robust to technical decisions made by researchers

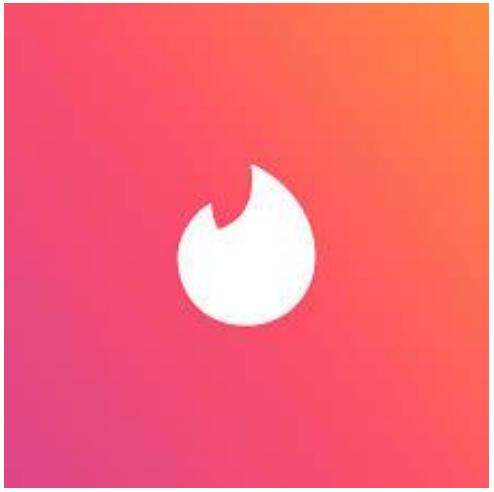
Some more words on Machine learning “fever”

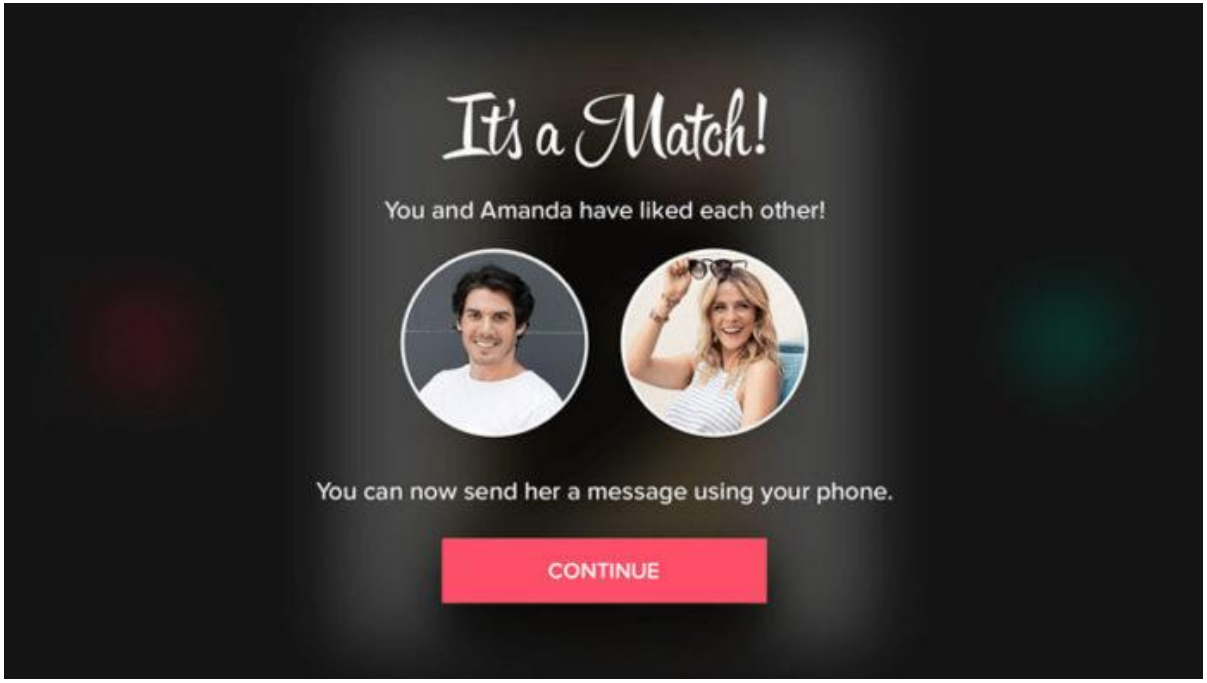


THIRD: ML algorithms tend to replicate what they already know, i.e., what they learnt in the training set, but they do not know which are the reasons behind (they are **predictive NOT explanatory models**!) and therefore usually they cannot innovate so easily!

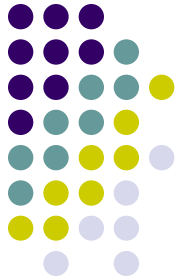
Some more words on Machine learning “fever”

The example: the Tinder ML algorithm for matching!





Some more words on Machine learning “fever”



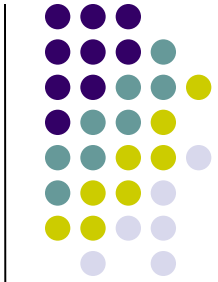
Therefore…OK with the fever! But

a) remember you remain a social scientist!

b) remember the four principles of text analysis!

c) do not play with the scripts! First of all, learn what you are (really) doing to your (poor…) data!

One final sentence (or better two…)



ONLY More GOOD (and well understood)

Data

is better than less

And this is true for **methods**

as well!!!