Supervised Learning Methods

Computational Text Analysis

Theresa Gessler

University of Zurich | http://theresagessler.eu/ | @th_ges

2022-05-09

Program

- **machine learning / supervised methods**
  - general idea
- **Machine Learning in Quanteda / work flow**
  - splitting your data
  - classifier implementation in quanteda and caret
  - evaluation of classifiers
- **substantive uses of machine learning**
  - feature scores
  - prediction accuracy
- **getting better**
  - text data
  - other algorithms
  - cross-validation and sampling
  - more evaluation metrics

Machine Learning

- a new answer to the (old) classification problem
  - e.g. How do we know if a text is positive or negative?
  - e.g. How do we know which topic a text speaks about?

human decision on features that are important (e.g. words in dictionary)

↕

automation of decision

- particularly attractive when you already have 'labelled data'
  - e.g. a set of speeches where we know the topic
  - e.g. when we use data coded by other researchers

overcoming difficulty of defining words ↔ decision-making as (somewhat) a black box

...still, the results are based on human coding decisions and share our biases!

Machine Learning

General idea

Tom Mitchell, *Machine Learning*, 1997

"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"

→ Classification as **task T**

→ pre-classified texts as **experience E**

→ correctly predicted new texts base for **Performance measures P**

Assumption: relation in data we know → relation in unknown data

Machine Learning

General idea

- we learn which text features predict categories of interest
- most classifier work similar to regressions: How much does a feature predict outcome
  - **regressions**: do resources predict conflict?
  - **dictionary**: does the word 'army' predict conflict?
  - **classifier**: does the word 'army' predict conflict? does the word 'is' predict conflict? does the word...?

Generalisation and Overfitting

1. **Generalisation**: A classifier learns to correctly predict output from given inputs not only in previously seen samples but **also in previously unseen samples**
2. **Overfitting**: A classifier learns to correctly predict output from given inputs in previously seen samples but **fails to do so in previously unseen samples**. This causes poor prediction/generalisation.

→ **overfitting**: predicting too close to existing data

- We train classifiers on existing data
  - trained to maximize in-sample performance
- BUT: applications typically on new data

→ we counter this with a specific work flow

Machine Learning

Workflow

- **(hand-coded) data** (*gold standard*), to be split into two parts:
  - **training set** - from which we learn
  - **test set** - on which we validate our classifier
- method to learn from hand-coded data: **classification algorithm**
  - how do we translate features into categories?
  - e.g. Naive Bayes, regularized regression, SVM, k-nearest neighbours
  - potentially combined with **cross-validation**
- method to **evaluate classifier**
  - performance metrics: confusion matrix, accuracy, precision, recall, F1 scores

Machine Learning in Quanteda

Machine Learning in Quanteda

Packages

- useful for getting started: `quanteda` (and `quanteda.textmodels`)
  - implementation of a few standard classification models (naiveBayes, SVM, regularized regression)
  - works directly on dfm
  - incredibly fast for text data
- package for machine learning: `caret` (**c**lassification **a**nd **re**gression **t**raining)
  - more complex but suited for a variety of uses
  - unifies the usage of machine learning algorithms from different R packages
    - currently 238 different classification models
  - tools for evaluation

However, caret is not focused on text data → useful for other ML applications but only takes data.frame version of dfm

Machine Learning in Quanteda

Data

Movie review dataset

A corpus object containing 2,000 movie reviews classified by positive or negative sentiment.

```
library(quanteda)
library(quanteda.textmodels)
reviews_corp ← data_corpus_moviereviews
reviews_dfm ← dfm(data_corpus_moviereviews,remove_punct=T)
```

Splitting

- splitting data into training and test set
  - build classifier on training set
  - evaluate classifier on test set
- in quanteda: `corpus_sample()` or `dfm_sample()`

```
reviews_train <- dfm_sample(reviews_dfm,0.8*ndoc(reviews_corp))
reviews_test <- dfm_subset(reviews_dfm,
  !(docnames(reviews_dfm) %in% docnames(reviews_train)))
```

Data: Separating test and training sample

```
head(reviews_train,3)
```

```
## Document-feature matrix of: 3 documents, 48,339 features (99.11% sparse) and 3 docvars.
##                   features
## docs                plot two teen couples go to  a church party drink
##   cv675_22871.txt    0   0    0       0  0 14 26      0     0     0
##   cv663_13019.txt    0   0    0       0  0 23 33      0     0     0
##   cv303_27520.txt    0   0    0       0  0 15 29      0     1     0
## [ reached max_nfeat ... 48,329 more features ]
```

```
head(reviews_test,3)
```

```
## Document-feature matrix of: 3 documents, 48,339 features (99.40% sparse) and 3 docvars.
##                   features
## docs                plot two teen couples go to  a church party drink
##   cv002_17424.txt    2   1    0       0  2  6 10      0     0     0
##   cv007_4992.txt     1   0    0       0  0  8 23      0     0     0
##   cv012_29411.txt    0   0    0       0  0 13  9      0     0     0
## [ reached max_nfeat ... 48,329 more features ]
```

Data: Adjusting training and test sample

Training and test sample have to be fully separated

- only use features contained in training set by trimming dfm_train (`dfm_trim()`)
- `dfm_match()` both **pads missing features** and **removes features** not contained in training data.

```
reviews_train ← reviews_train %>% dfm_trim(1)
reviews_test ← dfm_match(reviews_test, featnames(reviews_train))

head(reviews_test,3)
```

```
## Document-feature matrix of: 3 documents, 43,681 features (99.37% sparse) and 3
docvars.
##                    features
## docs                plot two teen couples go to  a church party drink
##   cv002_17424.txt    2   1    0       0  2  6 10      0     0     0
##   cv007_4992.txt     1   0    0       0  0  8 23      0     0     0
##   cv012_29411.txt    0   0    0       0  0 13  9      0     0     0
## [ reached max_nfeat ... 43,671 more features ]
```

Classification Algorithm: Naive Bayes (NB)

- Bayes Theorem: probability of event based on conditions

Intuition: If we observe the term "fantastic" in a text, how likely is this text a positive review?

1. Determine frequency of term in positive and negative reviews (prior).
2. Assess probability of features given a particular class.
3. Get probability of a document belonging to each class (posterior).
4. Which posterior is highest?

Classification Algorithm: Naive Bayes (NB)

Advantages

- Simple, fast, effective
- Relatively small training set required for good results (with reasonably balanced classes)
- Easy to obtain probabilities

Disadvantages

- Strong assumption of conditional independence ('naive') is problematic
- If feature is not in training set, it is disregarded for the classification ('irrelevant words')

Classification Algorithm: Training the model

- fast implementation in `textmodel_nb()`

```
nb_model<-textmodel_nb(reviews_train,docvars(reviews_train,
  "sentiment"))
nb_model
```

```
##
## Call:
## textmodel_nb.dfm(x = reviews_train, y = docvars(reviews_train,
##     "sentiment"))
##
##  Distribution: multinomial ; priors: 0.5 0.5 ; smoothing value: 1 ; 1600 training
documents;  fitted features.
```

Should be done within seconds:

```
## Time difference of 0.5411482 secs
```

Classification Algorithm: Predicting test data

- prediction of new data with `predict()`, model and new dfm

```
test_predictions<-predict(nb_model,
  newdata=reviews_test)
head(test_predictions,5)
```

```
## cv002_17424.txt  cv007_4992.txt cv012_29411.txt cv013_10494.txt  cv016_4348.txt
##             neg             neg             neg             neg             neg
## Levels: neg pos
```

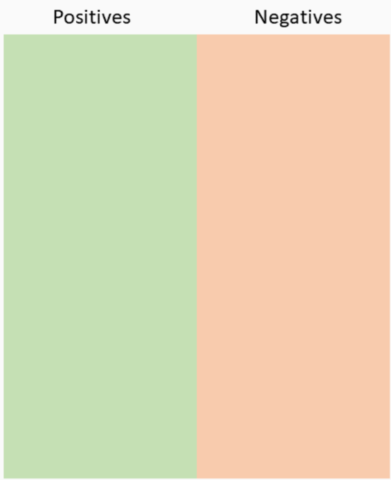
→ **How well did we do?**

```
table(docvars(reviews_test,"sentiment"),test_predictions)
```

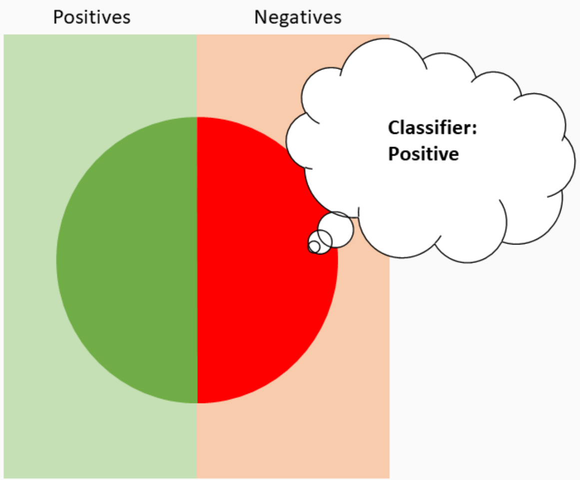
```
##      test_predictions
##       neg pos
##   neg 167  34
##   pos  32 167
```

Machine Learning in Quanteda

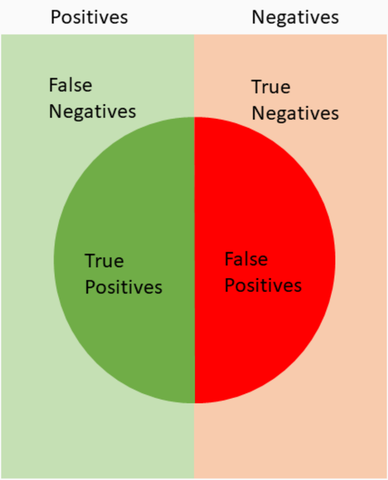
Evaluation



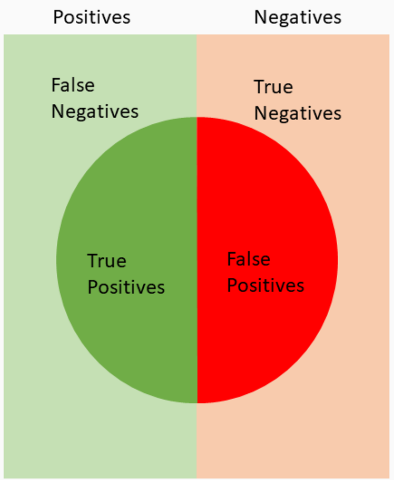
Evaluation

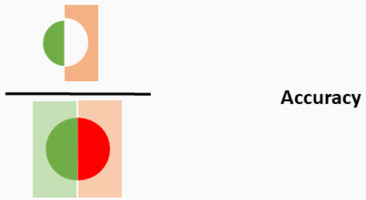


Evaluation


Positives
Negatives
False
Negatives
True
Negatives
True
Positives
False
Positives

Evaluation





- **Accuracy**: How many cases did we classify correctly?
    - *How many reviews did we correctly evaluate?*
- $\frac{Correctly\ classified}{Total\ number\ of\ cases} = \frac{true\ positives + true\ negatives}{Total\ number\ of\ cases}$

Substantive uses of ML

Substantive uses of ML

When we classify texts into outcomes we already know, classification results tell us

- the distribution of features across groups (Beltran et. al. 2020)
- our ability to accurately predict groups (Peterson & Spirling 2018)

→ We can use text classification for answering substantive questions

Substantive uses of ML

Feature scores

- many classifiers provide us with a score for each feature that quantifies **how predictive a feature is** of the outcome
  - e.g. *how predictive a word is of author gender*
  - *which words are most predictive of progressive / conservative ideology*
- this is based on the **differential use of this feature** across outcomes
  - sometimes combined with its **frequency**

- **application**: prediction with a classifier that contains feature weights, extracting those weights for best-fitted model

- Beltran, Javier, Aina Gallego, Alba Huidobro, Enrique Romero, and Lluís Padró. "Male and Female Politicians on Twitter: A Machine Learning Approach." European Journal of Political Research.

However, we might equally use a different metric - e.g. Keyness - to identify features

Substantive uses of ML

Prediction Accuracy

- **classification accuracy also "provides an informative instrument for the degree of aggregate polarization"** (120)
    - easily distinguishable parties are polarized, parties that speak similarly are not
- **application**: training per legislature, evaluation on held-out test set → accuracy (correctly predicted classes) as measure of polarization
- **limitation**: works best when parties use different words to discuss the same issue not raise different subjects → best for debates constrained by an agenda
    - ↔ issue competition
- Peterson, Andrew, and Arthur Spirling (2018). “Classification Accuracy as a Substantive Quantity of Interest: Measuring Polarization in Westminster Systems.” Political Analysis 26, no. 1: 120–28. https://doi.org/10.1017/pan.2017.39.
    - also: *intra-party polarization*, Goet, Niels D (2019). “The Politics of Procedural Choice: Regulating Legislative Debate in the UK House of Commons, 1811–2015.” British Journal of Political Science

Substantive uses of ML

Practicing Predictions

- `04_classifyingparliament.rmd`
  - classification accuracy in the British parliament
- At home: `04_thesisabstracts.rmd`
  - which features are most telling for each EUI department

Getting better

Getting better

Machine learning and text analysis

- machine learning is widely used for non-text problems
  - we use the same machine learning algorithms for text as for other data
- however, with text data **dimensionality** is a challenge
  - **number of features** is very high
  - **sparsity**: most features are very rare
  - we usually do not have **enough texts** to learn from

We optimize with several goals

→ **improve speed & computational costs**

→ **improve performance** (in unseen data)

→ **improve interpretability**

Getting better

Feature Engineering

- **reducing the number of features**
  - stemming and lemmatization can unify similar features
  - dimensionality reduction techniques (e.g. principal component analysis)
  - → **gain**: speed, computational costs
- **selecting meaningful features**
  - very infrequent features are unlikely to help with classification
  - overly frequent features are equally unlikely to help with classification
  - stopword removal
  - feature weighting
  - → **gain**: (speed, computational costs and) accuracy

Getting better

Feature Engineering

```
reviews_train_small ← dfm_trim(reviews_train,5)
```

- training model with 43681 respectively 13388 features

```
start ← Sys.time()
nb_model←textmodel_nb(reviews_train,docvars(reviews_train,
  "sentiment"))
end ← Sys.time()

start5 ← Sys.time()
nb_model←textmodel_nb(reviews_train_small,docvars(reviews_train,
  "sentiment"))
end5 ← Sys.time()
```

```
## [1] "Model with  43681  features:  0.034008026123046"
```

```
## [1] "Model with  13388  features:  0.03000807762146"
```

Getting better

Model building

- **obtaining more data**
  - new data generation methods like crowd-coding
  - → **gain**: accuracy
- **trying different models and specification**
  - e.g. from `caret`
  - → **gain**: accuracy, potentially speed for repeated runs
- **cross-validation** / **sampling**
  - preventing over-fitting
  - → **gain**: accuracy (on new data)
- **optimizing depending on task**
  - different evaluation standards for different tasks

Getting better: Algorithms

Getting better: Algorithms

Support vector machine (SVM)

- combines all data points
  - draw hyperplanes into multidimensional space to separate classes
  - no independence assumptions
- while NB is generative, SVM is **discriminative**
  - what is most likely classification, given text
- works better for large datasets (compared to NB)

```
svm_model<-textmodel_svm(reviews_train,docvars(reviews_train, "sentiment"))
svm_model
```

```
##
## Call:
## textmodel_svm.dfm(x = reviews_train, y = docvars(reviews_train,
##     "sentiment"))
##
## 1,600 training documents; 43,682 fitted features.
## Method: L2-regularized L2-loss support vector classification dual
(L2R_L2LOSS_SVC_DUAL)
```

Getting better: Algorithms

Support vector machine (SVM)

May take minimally longer than NB:

```
## Time difference of 7.264851 secs
```

Getting better: Algorithms

Regularized Regression

- outcome regressed on text features
- implementation in caret: LASSO (*Least Absolute Shrinkage and Selection Operator*)
  - penalty that biases estimates towards zero
  - in effect, LASSO performs feature / variable selection
- intuitive understanding of feature scores as regression results
  - focus on important features

Online Tutorial - including penalty estimation

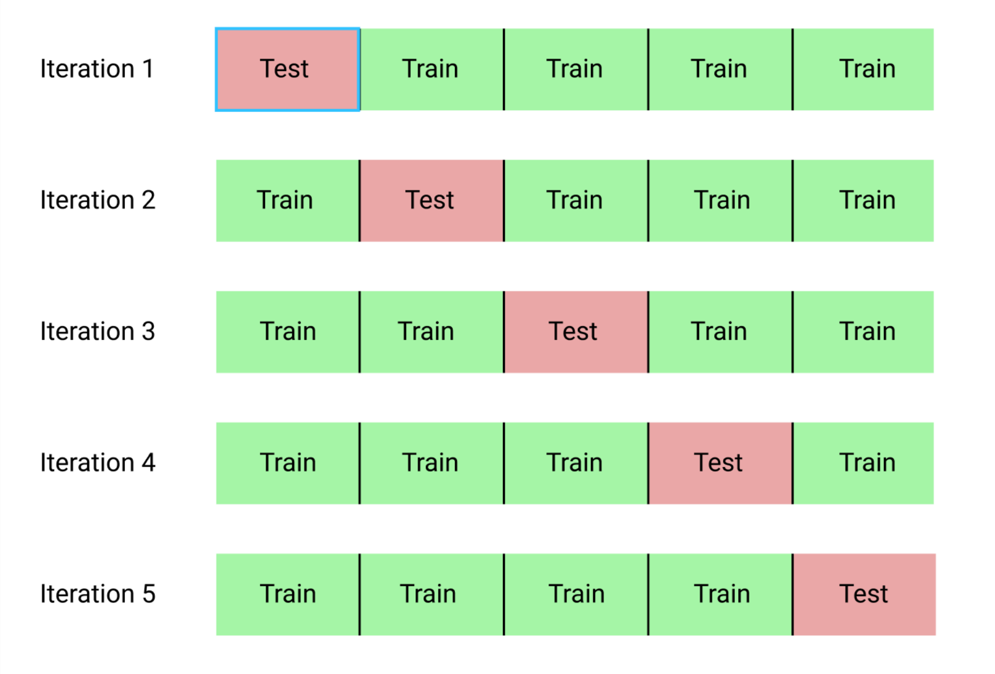
→ *this model, like many other models, requires caret or other external packages*

Getting better: Sampling

Getting better: Sampling

Cross-Validation

- Create K training and test sets ("folds") within training set
- For each k in K, run classifier and estimate performance in test set within fold



Getting better: Sampling

Cross-Validation and general advice

- **cross-validation**: implementation so far only in `quanteda.classifiers` (crossval), `caret` (documentation) or by hand
  - caret also has other sampling strategies like upsampling and downsampling

- Important: **model complexity**
  - decreases error on training set: adaptation to specifics of data set
  - likely increases error in test set

→ **simple models are often preferable, also for interpretability**

- whether you choose to do a simple train-test split or use k-fold cross-validation: **final evaluation should be done on test-sample!**
- *Splitting our data is like pre-registering a survey:*
  - you can try whatever you want during pre-testing (on the training set)
  - once you decide on a model, you are 'stuck' when going to test set

Getting better: Evaluation

Getting better: Evaluation

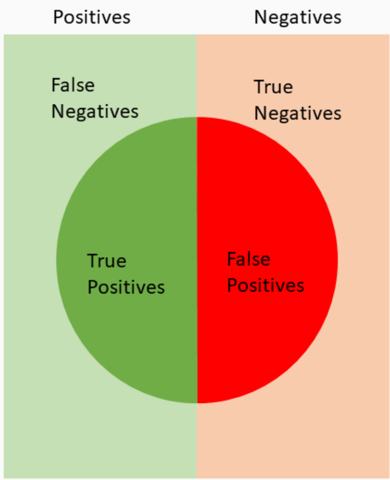
The confusion matrix

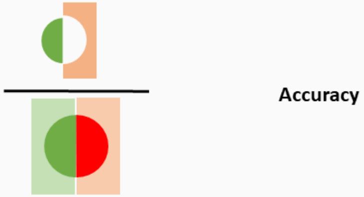
- more detailed evaluation with `confusionMatrix()` function from `caret`

```
confusionMatrix(as.factor(docvars(reviews_test,"sentiment")), test_predictions)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 167  34
##        pos  32 167
##
##                Accuracy : 0.835
##                  95% CI : (0.7949, 0.87)
##     No Information Rate : 0.5025
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.67
##
##  Mcnemar's Test P-Value : 0.902
##
##             Sensitivity : 0.8392
##             Specificity : 0.8308
##          Pos Pred Value : 0.8308
##          Neg Pred Value : 0.8392
##              Prevalence : 0.4975
##          Detection Rate : 0.4175
##    Detection Prevalence : 0.5025
##       Balanced Accuracy : 0.8350
##
```

Accuracy





- **Accuracy**: How many cases did we classify correctly?
  - *How many reviews did we correctly evaluate?*
- $\frac{Correctly\ classified}{Total\ number\ of\ cases} = \frac{true\ positives + true\ negatives}{Total\ number\ of\ cases}$

Getting better: Evaluation

What's wrong with accuracy?

- imagine, we are interested in a very rare outcome - here: B
  - rare classes are often under-predicted
  - consider these - fabricated - predictions to see the effect

```
true <- factor(c("B","B","B","B","B",rep("A",35)))
pred <- factor(c("B",rep("A",39)))
caret::confusionMatrix(pred,true,"B")$table
```

```
##            Reference
## Prediction  A  B
##           A 35  4
##           B  0  1
```

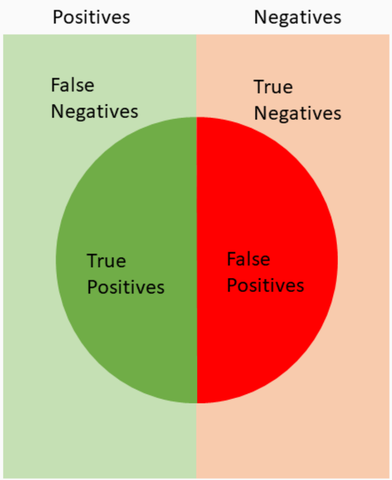
```
caret::confusionMatrix(pred,true,"B")$overall['Accuracy']
```

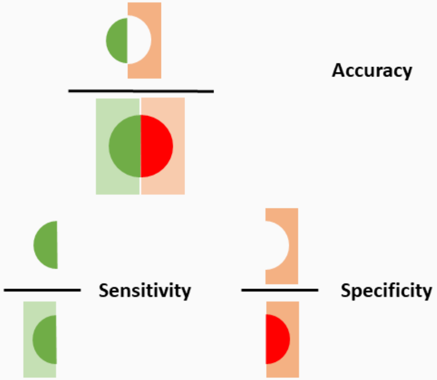
```
## Accuracy
##      0.9
```

→ **Overall accuracy is a bad measure when classes are imbalanced**

Getting better: Evaluation

Accuracy, Sensitivity, Specificity

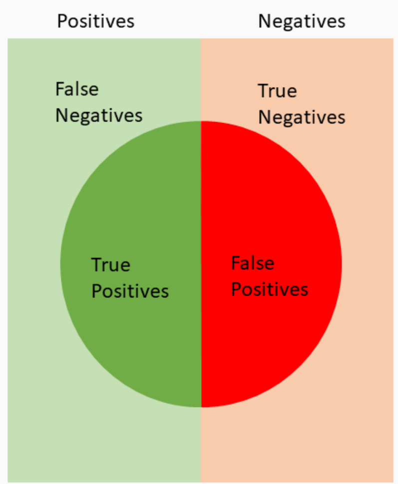


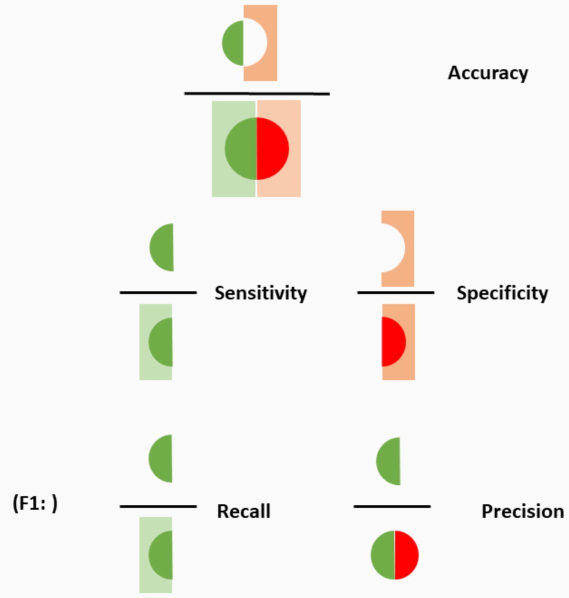


- **Accuracy** = (TP+TN)/(TP+TN+FP+FN)
  - **sensitivity: true positive rate**
  - **specificity: true negative rate**

Getting better: Evaluation

Precision, Recall, F1 scores





- **recall: TP / (TP+FN)**
- **precision: TP / (TP+FP)**
- **F1 score**: harmonic mean of precision and recall (=sensitivity)
- typically calculated by Class

What's wrong with accuracy?

```
true <- factor(c("B","B","B","B","B",rep("A",35)))
pred <- factor(c("B",rep("A",39)))
caret::confusionMatrix(pred,true,positive="B")$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.2000000            1.0000000            1.0000000
##       Neg Pred Value            Precision               Recall
##            0.8974359            1.0000000            0.2000000
##                   F1           Prevalence       Detection Rate
##            0.3333333            0.1250000            0.0250000
## Detection Prevalence    Balanced Accuracy
##            0.0250000            0.6000000
```

Tip: specify the positve class

→ Our fake-predictions have a low Recall, low Sensitivity, bad F1 score

Getting better: Evaluation

Which metric matters?

Depending on the task, we optimize precision, recall or other metrics:

- **interested in multiple categories** → F1 score, accuracy
  - e.g. multiple newspaper topics, disease detection for chronic diseases
- **finding the needle in the hay stack** → recall
  - e.g. hate speech to be checked by human evaluators, disease detection for Covid-19
- **finding only what we need** → precision
  - e.g. content to be banned

- Generally: Measuring performance is a whole science in itself

After the break

After the break

Unsupervised classification

- Unsupervised methods scale documents based on patterns of similarity from the document-feature matrix, without requiring a training step
- Examples
  - Wordfish
  - topic models
- **Relative advantage**: You do not have to know the dimension being scaled (also a disadvantage!)

After the break

Homework

- **complete & hand in:**
  - **03_classification.rmd**
  - **03_classifyingparliament.rmd**
  - EUI Thesis abstracts: **03_thesisabstracts.rmd**
- if you want, try a different parliament

Thank you! - Questions?