A Practical Introduction to Machine Learning in Python
Day 4 – Thursday
»Supervised Machine Learning«

Rupert Kiddle
Sjoerd Stolwijk

r.t.kiddle@vu.nl, @rptkiddle
s.b.stolwijk@uu.nl

September 19, 2024

Today

- Recap
- Evaluation
- Supervised Machine Learning for Text Classification
  - (Traditional) non-SML approaches
  - Different classification algorithms
  - A note on sample size
  - The implementation
- The problem of overfitting
  - Train/validation/test split

Recap

	Methodological approach		
	Counting and Dictionary	*Supervised Machine Learning*	*Unsupervised Machine Learning*
Typical research interests and content features	visibility analysis sentiment analysis subjectivity analysis	frames topics gender bias	frames topics
Common statistical procedures	string comparisons counting	support vector machines naive Bayes	principal component analysis cluster analysis latent dirichlet allocation semantic network analysis

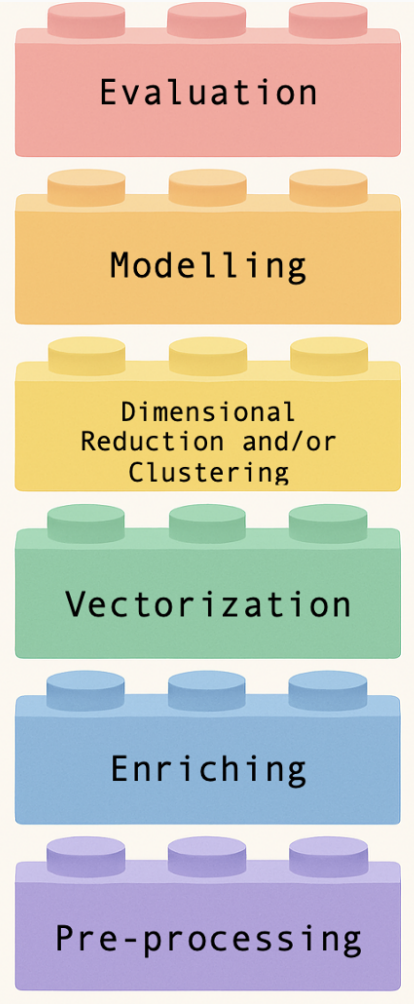
deductive → **inductive**

Boumans and Trilling, 2016

How Humans Learn vs. How Machines Learn

Human Learning	Machine Learning
Example counting (seeing many repetitions)	Rule-based systems / Traditional ML
Pointing out (explicit correction / feedback)	Supervised learning (labels, corrections)
Explaining in other words (paraphrase / abstraction)	Representation learning / Deep Learning
Interpreting context (discourse / world knowledge)	Transformers / Generative AI

Lego pipeline



how to assess the output of your model

how to train a model to describe (UML) or to predict (SML)

how to group features *or* cases

how to turn text → numbers

how to label text data

how to tidy up text data

You have done it before!

Regression

1. Based on your data, you estimate some regression equation:

   $y = \alpha + \beta_1 x_1 + \cdots + \beta_p x_p + \epsilon$

2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$

3. Example: You estimated a regression equation where y is newspaper reading in days/week

   $y = -.8 + .4 \times man + .08 \times age$

4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*

   $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = .12$

   $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
   $y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*:
   $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$
   $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
   $y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*:
   $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$
   $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
   $y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*:
   $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$
   $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week: $y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*: $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$ $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation $y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$
2. Even if you have some *new unseen data*, you can estimate your expected outcome $\hat{y}$!
3. Example: You estimated a regression equation where y is newspaper reading in days/week: $y = -.8 + .4 \times man + .08 \times age$
4. You could now calculate $\hat{y}$ for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*: $\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$ $\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$

This is

Supervised Machine Learning!

. . . but. . .

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding ("labeled data","annotated data" in SML-lingo)
    - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables ("features")
- Typically, IVs are word frequencies (often weighted, e.g. tf×idf) (⇒BOW-representation)

. . . but. . .

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding ("labeled data","annotated data" in SML-lingo)
  - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables ("features")
- Typically, IVs are word frequencies (often weighted, e.g. tf×idf) (⇒BOW-representation)

. . . but. . .

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding ("labeled data","annotated data" in SML-lingo)
  - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables ("features")
- Typically, IVs are word frequencies (often weighted, e.g. tf×idf) (⇒BOW-representation)

. . . but. . .

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding ("labeled data","annotated data" in SML-lingo)
  - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables ("features")
- Typically, IVs are word frequencies (often weighted, e.g. tf$\times$idf) ($\Rightarrow$BOW-representation)

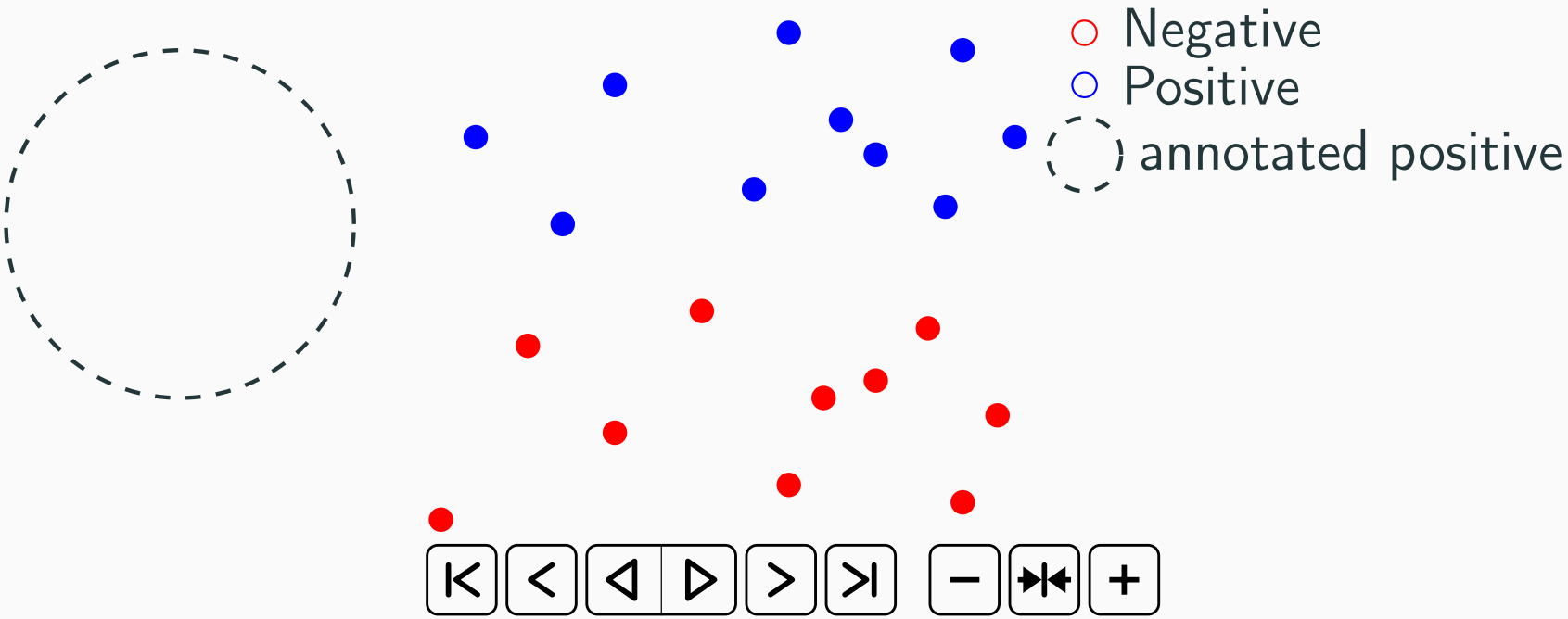
Classification tasks

Typical questions include:

- Is this article about topic A, B, C, D, or E?
- Is this review positive or negative?
- Does this text contain frame F?
- Is this satire?
- Is this misinformation?
- Given past behavior, can I predict the next click?

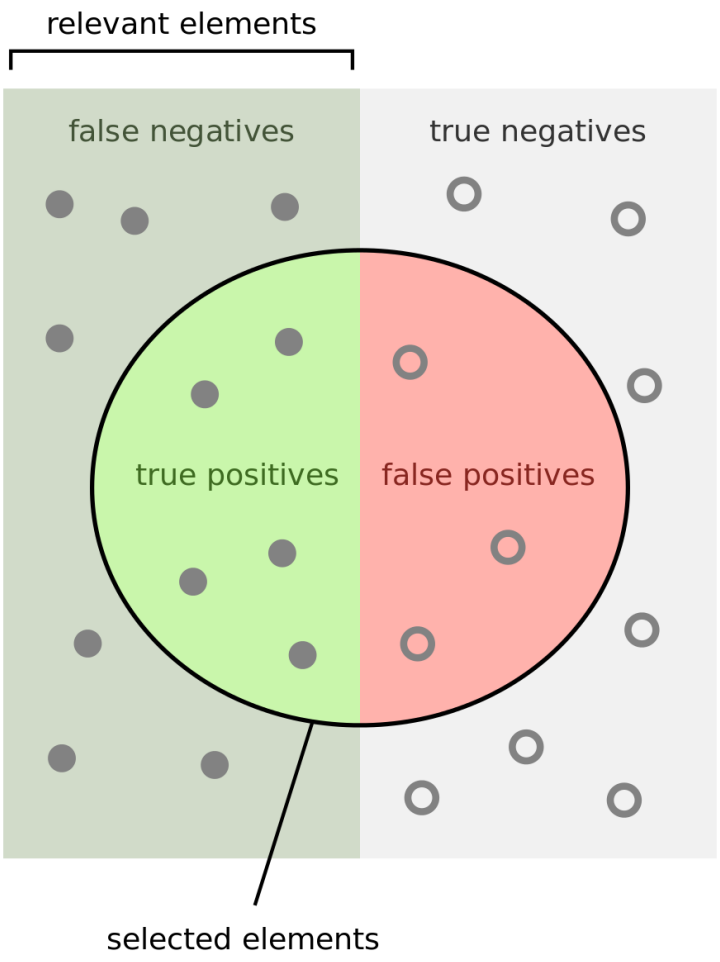
Evaluation

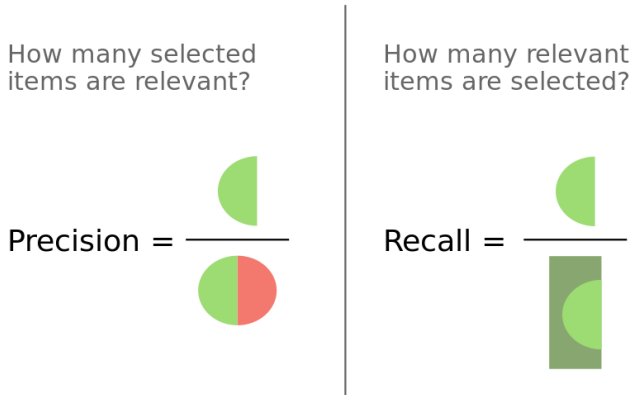
Performance: what do we do and how to evaluate it



Points inside the circle are "annotated positive" by the classifier.
Yellow highlight: annotated points (true/false positives/negatives).

http://commons.wikimedia.org/wiki/File:Precisionrecall.svg





Some measures

- Accuracy
- Recall
- Precision
- $F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

Supervised Machine Learning for Text Classification

Supervised Machine Learning for Text Classification

(Traditional) non-SML approaches

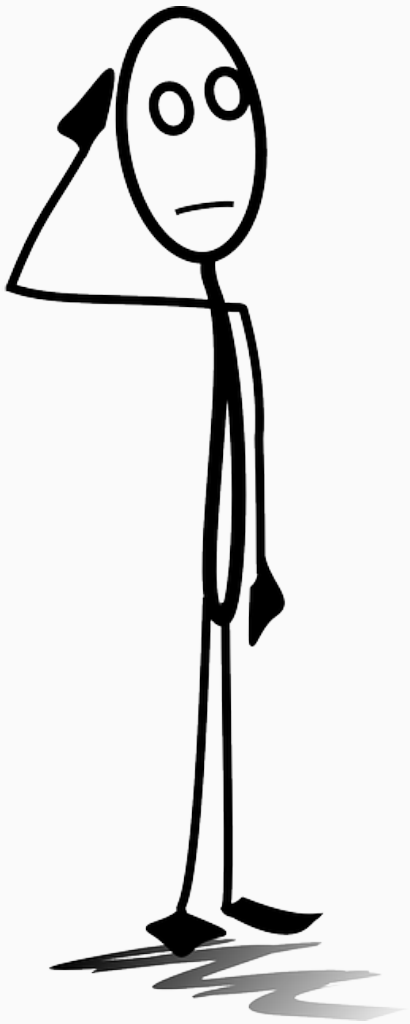
Let's consider three tasks

For a given text (say, a news article, a press release, a review), determine the

sentiment e.g., [positive|neutral|negative]

topic e.g., [sports|economy|politics|entertainment|other]

frames e.g., [economic|human|moral|conflict], or
non-exclusive: economic = [0|1], human = [0|1], ...



Imagine using a dictionary-based (list of keywords, list of regular expressions, or similar) approach to these tasks. How does the design (length, inclusiveness, etc.) of this list influence precision and recall?

Example dictionary (Minqing Hu and Bing Liu, 2004)

9/17/25, 5:41 PM positive-words.txt

```
acclamation
accolade
accolades
accommodative
accomodative
accomplish
accomplished
accomplishment
accomplishments
accurate
accurately
achievable
achievement
achievements
achievible
acumen
adaptable
adaptive
```

https://gist.github.com/mkulakowski2/4289437 4/119

Dictionary-based approaches for text classification

good for

- distinct, manifest things (names of organizations, pronouns, swearwords (?), ...)
- little room for interpretation/misunderstandings etc.
- "must-be-explainable-to-a-five-year-old"

bad for

- latent constructs and concepts
- implicit things

Hence, *not* state-of-the-art for

- topics
- frames
- sentiment

Dictionary-based approaches for text classification

good for

- distinct, manifest things (names of organizations, pronouns, swearwords (?), ...)
- little room for interpretation/misunderstandings etc.
- "must-be-explainable-to-a-five-year-old"

bad for

- latent constructs and concepts
- implicit things

Hence, *not* state-of-the-art for

- topics
- frames
- sentiment

Dictionary-based approaches for text classification

good for

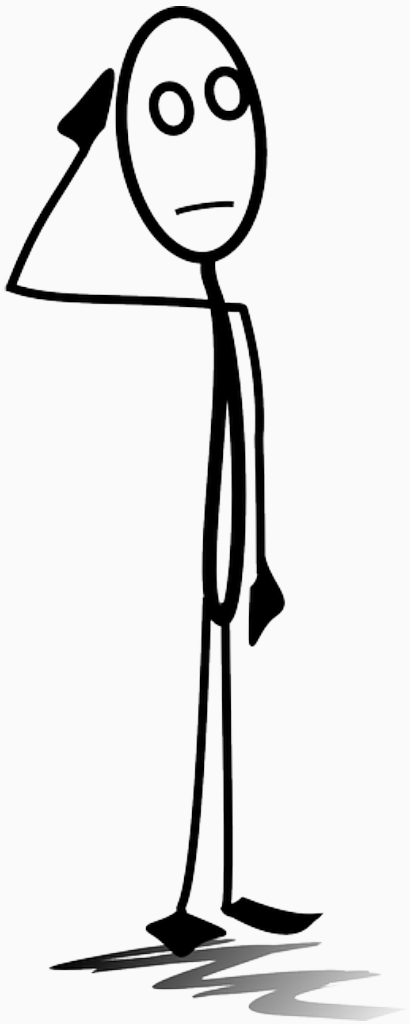
- distinct, manifest things (names of organizations, pronouns, swearwords (?), ...)
- little room for interpretation/misunderstandings etc.
- "must-be-explainable-to-a-five-year-old"

bad for

- latent constructs and concepts
- implicit things

Hence, *not* state-of-the-art for

- topics
- frames
- sentiment



Can we construct a dictionary that, irrespective of the context, gives us a meaningful estimate of sentiment?

Boukes et al., 2020: Sentiment analysis of economic news

Table A1. Correlations between sentiment scores using different methods for headlines (above) and full texts (below).

	Headline							
	Manual coding	Recession	D & B	LIWC	SentiStrength	Pattern	Polyglot	DANEW
Manual coding	1.00 ***							
Recession	-	-						
Damstra and Boukes (2018)	0.16 ***	-	1.00 ***					
LIWC	**0.30 *****	-	0.16 ***	1.00 ***				
SentiStrength	0.24 ***	-	0.08 **	0.26 ***	1.00 ***			
Pattern	0.22 ***	-	0.00	0.30 ***	0.22 ***	1.00 ***		
Polyglot	0.30 ***	-	0.19 ***	0.32 ***	0.37 ***	0.26 ***	1.00 ***	
DANEW	0.24 ***	-	0.04	**0.43 *****	0.33 ***	0.23 ***	0.32 ***	1.00 ***
	Full text							
	Manual coding	Recession	D & B	LIWC	SentiStrength	Pattern	Polyglot	DANEW
Manual coding	1.00 ***							
Recession	−0.06 *	1.00 ***						
Damstra and Boukes (2018)	0.27 ***	−0.16 ***	1.00 ***					
LIWC	0.39 ***	0.02	0.27 ***	1.00 ***				
SentiStrength	0.17 ***	−0.01	0.10 ***	0.18 ***	1.00 ***			
Pattern	0.13 ***	−0.02	0.04	0.28 ***	0.12 ***	1.00 ***		
Polyglot	0.26 ***	0.05	0.17 ***	0.41 ***	0.21 ***	0.30 ***	1.00 ***	
DANEW	0.15 ***	0.06 *	0.05	0.36 ***	0.18 ***	0.29 ***	0.37 ***	1.00 ***

The word "recession" did not occur in headlines of our sample, as such, no correlation coefficient is available for the recession classifier; *** $p < .001$, ** $p < .010$, * $p < .05$.

Boukes et al., 2020: Sentiment analysis of economic news

- Dictionaries have low agreement with each other, and also with human coders
- Even their own dictionary didn't agree
- **This is not because these dictionaries are particularly bad!** Main point: For such a complex and context-dependent task, a dictionary is just not the right tool.

van Atteveldt et al., 2021: Extending Boukes et al., 2020 with SML

"manual coding (using undergraduate students) yields the best results

[. . .] A good second place is taken by crowd coding [. . .]

[. . .] machine learning performs worse than both students' manual coding and crowd coding. Reaching $\alpha = 0.50$ for deep learning (CNN) and slightly worse for classical machine learning (SVM; $\alpha = 0.41$, NB; $\alpha = 0.40$), machine learning still performs significantly better than chance [. . .]

Finally, [. . .] dictionaries [. . .] perform worse than the machine learning results and much worse than manual annotation [. . .] [and] approximate chance agreement"

Vermeer et al., 2019: Satisfaction with brands

Category	Technique	Accuracy	Precision	Recall
Satisfaction ($N = 854$)				
Sentiment analysis	LIWC	0.05	0.06	0.04
	P	0.04	0.04	0.04
	SN	0.07	0.07	0.08
Dictionary-based	D	0.15	0.30	0.10
Machine learning	BNB	0.38	0.44	0.34
	MNB	0.32	0.67	0.21
	LR	0.51	0.38	0.76
	SGD	0.49	0.38	0.69
	SVM	0.52	0.41	0.63
	PA	0.50	0.40	0.68
Neutral ($N = 760$)				
Sentiment analysis	LIWC	0.13	0.16	0.10
	P	0.13	0.13	0.14
	SN	0.19	0.16	0.22
Dictionary-based	D	0.14	0.35	0.09
Machine learning	BNB	0.28	0.25	0.32
	MNB	0.15	0.34	0.10
	LR	0.37	0.25	0.74
	SGD	0.33	0.23	0.60
	SVM	0.36	0.24	0.69
	PA	0.34	0.24	0.60
Dissatisfaction ($N = 267$)				
Sentiment analysis	LIWC	0.20	0.15	0.29
	P	0.19	0.12	0.40
	SN	0.22	0.14	0.54
Dictionary-based	D	0.09	0.41	0.05
Machine learning	BNB	0.26	0.20	0.40
	MNB	0.25	0.48	0.16
	LR	0.35	0.23	0.77
	SGD	0.39	0.32	0.48
	SVM	0.04	0.02	1.00
	PA	0.35	0.23	0.71

Note. LIWC *Linguistic Inquiry and Word Count;* P *Pattern;* SN *Sentiment Net*; D *Dictionary-based;* BN *Bernoulli Naïve Bayes*; MNB *Multinomial Naïve Bayes*; LR *Logistic Regression*; SGD *Stochastic Gradient Descent*; SVM *Support Vector Machine*; and PA *Passive Aggressive*. Performance scores ≥0.60 have been highlighted. Results merely derived from the test set.

SML is no panacea, but it introduces a promising approach to analyzing large quantities of texts. Don't believe off-the-shelf packages that claim to do the work for you. (For small datasets, just do it by hand.)

Supervised Machine Learning for Text Classification

Different classification algorithms

Different classification algorithms

- It is an empirical question which one works best
- We typically try several ones and select the best
- (remember: we have a test dataset that we did *not* use to train the model, so that we can assess how well it predicts the test labels based on the test features)
- To avoid *p*-hacking-like scenarios (which we call "overfitting"), there are techniques available (e.g., cross-validation, which we address this afternoon)

(to make it easier, imagine a binary classification ("positive"/"negative"), but it doesn't really matter whether there are two or more labels)

Different classification algorithms

- It is an empirical question which one works best
- We typically try several ones and select the best
- (remember: we have a test dataset that we did *not* use to train the model, so that we can assess how well it predicts the test labels based on the test features)
- To avoid *p*-hacking-like scenarios (which we call "overfitting"), there are techniques available (e.g., cross-validation, which we address this afternoon)

(to make it easier, imagine a binary classification ("positive"/"negative"), but it doesn't really matter whether there are two or more labels)

Different classification algorithms

- It is an empirical question which one works best
- We typically try several ones and select the best
- (remember: we have a test dataset that we did *not* use to train the model, so that we can assess how well it predicts the test labels based on the test features)
- To avoid *p*-hacking-like scenarios (which we call "overfitting"), there are techniques available (e.g., cross-validation, which we address this afternoon)

(to make it easier, imagine a binary classification ("positive"/"negative"), but it doesn't really matter whether there are two or more labels)

Different classification algorithms

- It is an empirical question which one works best
- We typically try several ones and select the best
- (remember: we have a test dataset that we did *not* use to train the model, so that we can assess how well it predicts the test labels based on the test features)
- To avoid *p*-hacking-like scenarios (which we call "overfitting"), there are techniques available (e.g., cross-validation, which we address this afternoon)

(to make it easier, imagine a binary classification ("positive"/"negative"), but it doesn't really matter whether there are two or more labels)

Naïve Bayes

Bayes' theorem

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)}$$

A = Text is about sports

B = Text contains 'very', 'close', 'game'. Furthermore, we simply multiply the probabilities for the features:

$$P(B) = P(\textit{very close game}) = P(\textit{very}) \times P(\textit{close}) \times P(\textit{game})$$

We can fill in all values by counting how many articles are about sports, and how often the words occur in these texts.

Naïve Bayes

Bayes' theorem

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)}$$

A = Text is about sports

B = Text contains 'very', 'close', 'game'. Furthermore, we simply multiply the probabilities for the features:

$$P(B) = P(very\ close\ game) = P(very) \times P(close) \times P(game)$$

We can fill in all values by counting how many articles are about sports, and how often the words occur in these texts.

Naïve Bayes

Bayes' theorem

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)}$$

A = Text is about sports

B = Text contains 'very', 'close', 'game'. Furthermore, we simply multiply the probabilities for the features:

$$P(B) = P(\textit{very close game}) = P(\textit{very}) \times P(\textit{close}) \times P(\textit{game})$$

We can fill in all values by counting how many articles are about sports, and how often the words occur in these texts.

Naïve Bayes

$$P(\text{label} \mid \text{features}) =$$

$$\frac{P(x_1 \mid \textit{label}) \cdot P(x_2 \mid \text{label}) \ \cdot \mathrm{P}(\mathrm{x}_3 \mid \text{label}) \cdot \mathrm{P}(\text{label})}{P(x_1) \cdot P(x_2) \cdot P(x_3)}$$

.

- Formulas always look intimidating, but we only need to fill in how many documents containing feature x_n have the label, how often the label occurs, and how often each feature occurs
- Also for computers, this is *really easy and fast*
- Weird assumption: features are independent
- Often used as a baseline

Logistic Regression

Probability of a binary outcome in a regression model

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n)}}$$

Just like in OLS regression, we have an intercept and regression coefficients. We use a threshold (default: 0.5) and above, we assign the positive label ('good movie'), below, the negative label ('bad movie').

Logistic Regression

- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- ... or to change the threshold to change our precision/recall-trade off (e.g., 0.7 = higher precision, lower recall)

Logistic Regression

- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- ... or to change the threshold to change our precision/recall-trade off (e.g., 0.7 = higher precision, lower recall)

Logistic Regression

- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- ... or to change the threshold to change our precision/recall-trade off (e.g., 0.7 = higher precision, lower recall)

Logistic Regression

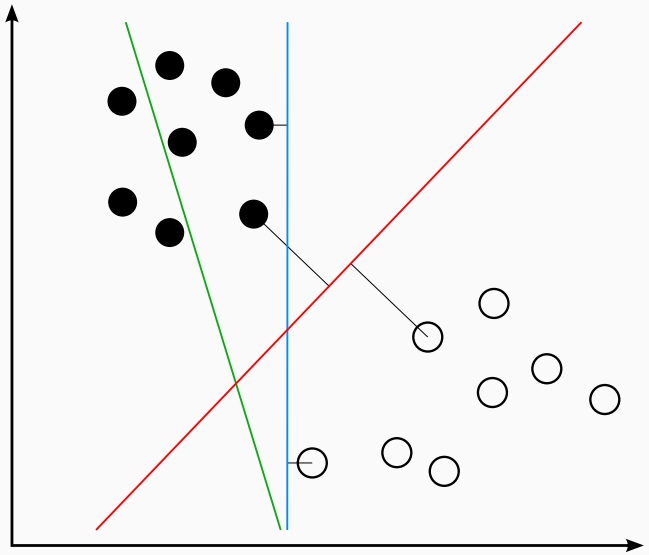
- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- ... or to change the threshold to change our precision/recall-trade off (e.g., 0.7 = higher precision, lower recall)

Logistic Regression

- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- . . . or to change the threshold to change our precision/recall-trade off (e.g., 0.7 = higher precision, lower recall)

Support Vector Machines

- Idea: Find a hyperplane that best separates your cases
- Can be linear, but does not have to be (depends on the so-called kernel you choose)
- Very popular



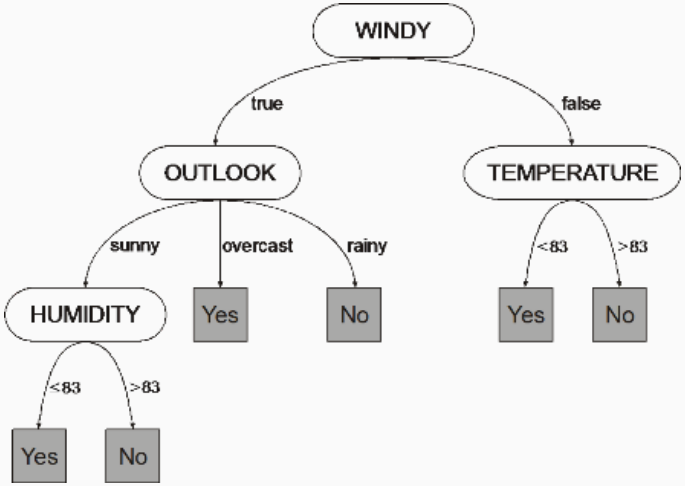
https://upload.wikimedia.org/wikipedia/commons/b/b5/Svm_separating_hyperplanes_%28SVG%29.svg

SVM vs logistic regression

- for *linearly separable* classes not much difference
- with the right hyperparameters, SVM is less sensitive to outliers
- biggest advantage: with the *kernel trick*, data can be transformed that they *become* linearly separable

Decision Trees and Random Forests

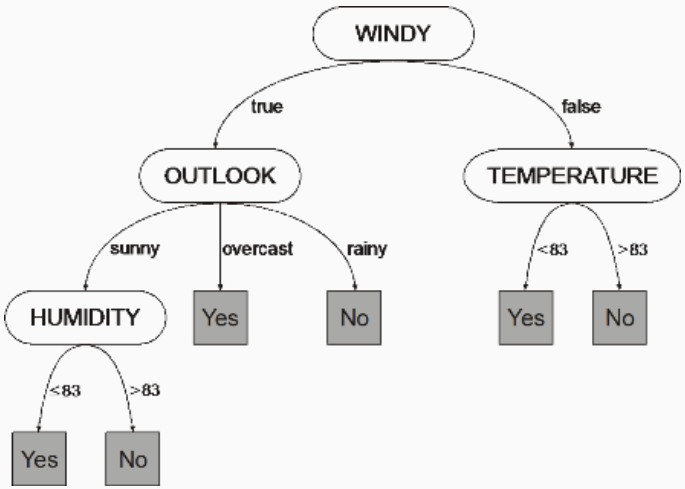
- Model problem as a series of decisions (e.g., if cloudy then . . . if temperature > 30 degrees then . . .)
- Order and cutoff-points are determined by an algorithm
- Big advantage: Model non-linear relationships
- And: They are easy to interpret (!) ("white box")



https://upload.wikimedia.org/wikipedia/en/4/4f/GEP_decision_tree_with_numeric_and_nominal_attributes.png

Decision Trees and Random Forests

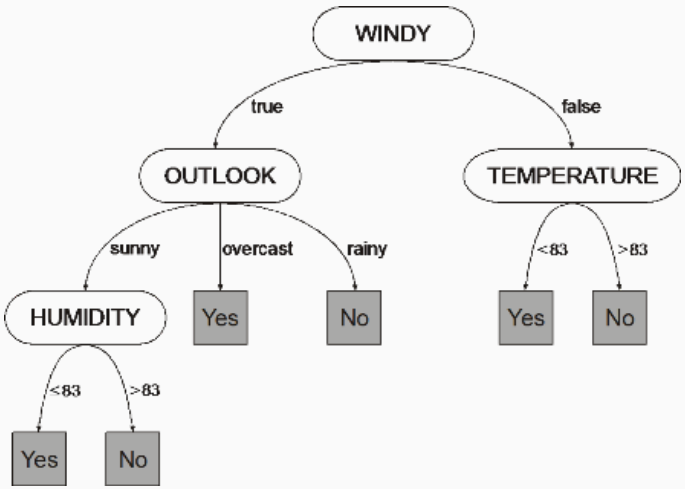
- Model problem as a series of decisions (e.g., if cloudy then . . . if temperature > 30 degrees then . . .)
- Order and cutoff-points are determined by an algorithm
- Big advantage: Model non-linear relationships
- And: They are easy to interpret (!) ("white box")



https://upload.wikimedia.org/wikipedia/en/4/4f/GEP_decision_tree_with_numeric_and_nominal_attributes.png

Decision Trees and Random Forests

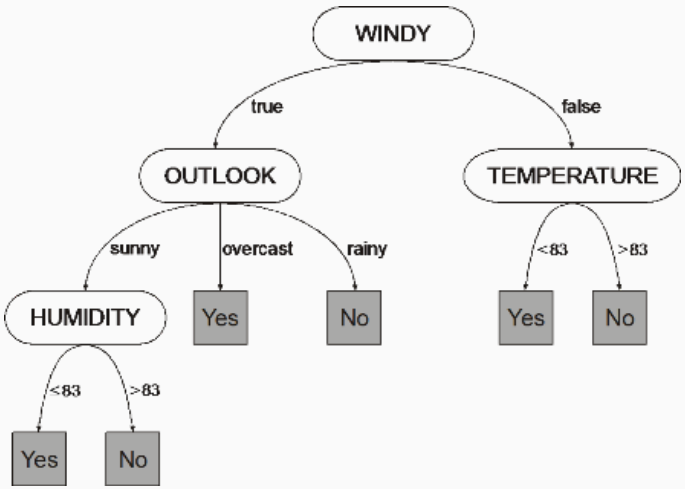
- Model problem as a series of decisions (e.g., if cloudy then . . . if temperature > 30 degrees then . . .)
- Order and cutoff-points are determined by an algorithm
- Big advantage: Model non-linear relationships
- And: They are easy to interpret (!) ("white box")



https://upload.wikimedia.org/wikipedia/en/4/4f/GEP_decision_tree_with_numeric_and_nominal_attributes.png

Decision Trees and Random Forests

- Model problem as a series of decisions (e.g., if cloudy then . . . if temperature > 30 degrees then . . .)
- Order and cutoff-points are determined by an algorithm
- Big advantage: Model non-linear relationships
- And: They are easy to interpret (!) ("white box")



https://upload.wikimedia.org/wikipedia/en/4/4f/GEP_decision_tree_with_numeric_and_nominal_attributes.png

Decision Trees and Random Forests

Disadvantages of decision trees

- comparatively inaccurate
- once you are in the wrong branch, you cannot go 'back up'
- prone to overfitting (e.g., outlier in training data may lead to completely different outcome)

Therefore, nowadays people use *random forests*: Random forests *combine* the predictions of *multiple* trees ⇒ might be a good choice for your non-linear classification problem

Decision Trees and Random Forests

Disadvantages of decision trees

- comparatively inaccurate
- once you are in the wrong branch, you cannot go 'back up'
- prone to overfitting (e.g., outlier in training data may lead to completely different outcome)

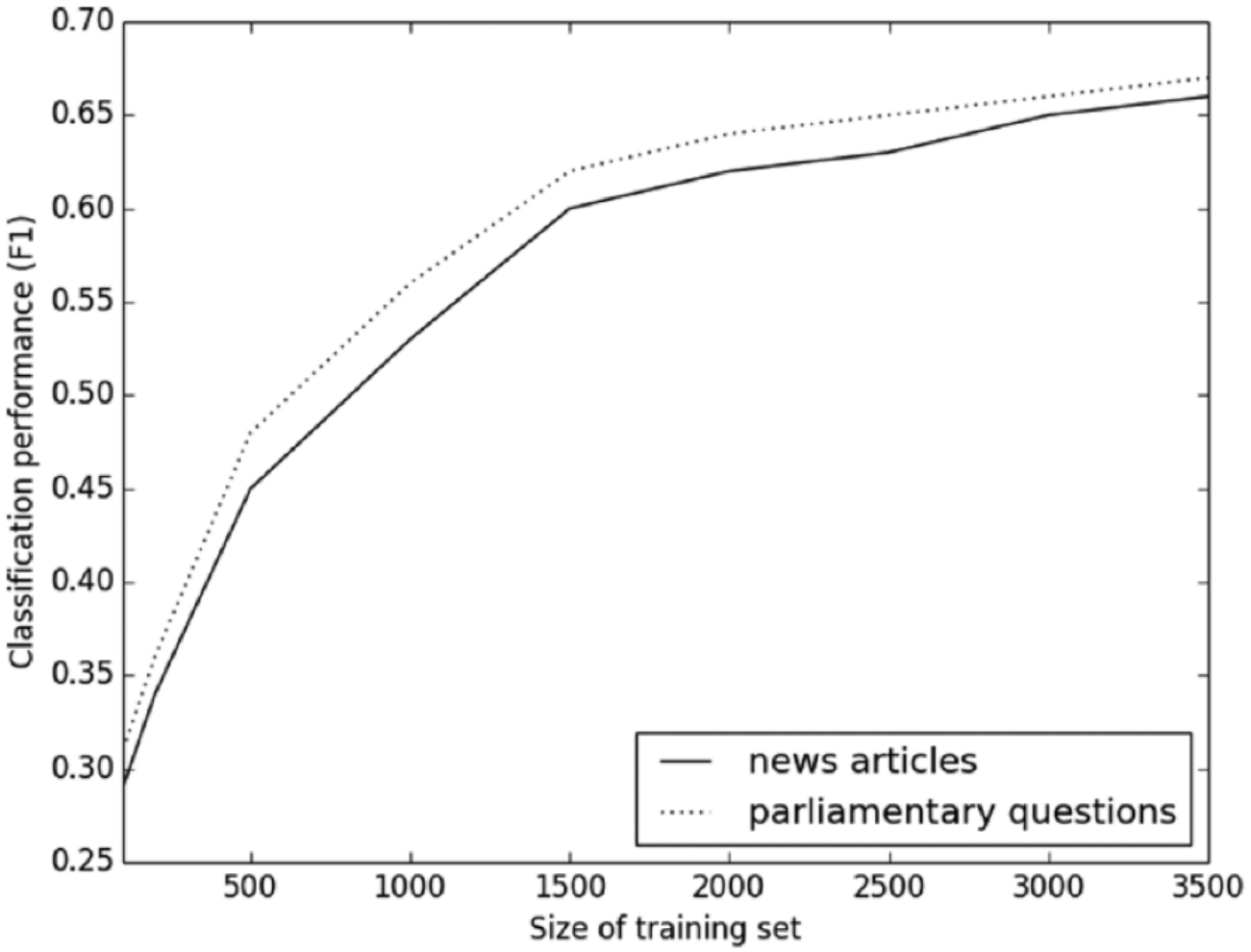
Therefore, nowadays people use *random forests*: Random forests *combine* the predictions of *multiple* trees $\Rightarrow$ might be a good choice for your non-linear classification problem

Supervised Machine Learning for Text Classification

A note on sample size

Training size vs performance Burscher et al., 2015

FIGURE 1
Learning Curves for the Classification of News Articles and PQs



What does this mean for our research?

It we have 2,000 documents with manually coded frames and topics. . .

- we can use them to train a SML classifier
- which can code an unlimited number of new documents
- with an acceptable accuracy (at least for some of them)
- Some easier tasks even need only 500 training documents, see Hopkins and King, 2010

What does this mean for our research?

It we have 2,000 documents with manually coded frames and topics. . .

- we can use them to train a SML classifier
- which can code an unlimited number of new documents
- with an acceptable accuracy (at least for some of them)
- Some easier tasks even need only 500 training documents, see Hopkins and King, 2010

Supervised Machine Learning for Text Classification

The implementation

An implementation

Let's say we have two lists, with movie reviews and their rating:

```
reviews_train = ["This is a great movie", "Bad movie", ... ...]
labels_train = [1,-1, ...]
```

And a second dataset with an identical structure:

```
reviews_test = ["Not that good","Nice film", ... ...]
labels_test = [-1,1, ......]
```

Both are drawn from the same population; it is pure chance whether a specific review is on the one list or the other.

Based on an example from http://blog.dataquest.io/blog/naive-bayes-movies/

Training a A Naïve Bayes Classifier

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
from sklearn import metrics

vectorizer = CountVectorizer(stop_words='english')
features_train = vectorizer.fit_transform(reviews_train)
features_test = vectorizer.transform(reviews_test)

# Fit a naive bayes model to the training data.
nb = MultinomialNB()
nb.fit(features_train, labels_train)

# Now we can use the model to predict classifications for our test features.
predictions = nb.predict(features_test)

print(f"Precision:\t{metrics.precision_score(labels_test, predictions, pos_label=1, labels = [-1,1])}"
print(f"Recall:\t{metrics.recall_score(labels_test, predictions, pos_label=1, labels = [-1,1])}"
```

And it works!

Using 50,000 IMDB movies that are classified as either negative or positive,

- We created a list with 25,000 training tuples and another one with 25,000 test tuples and
- trained a classifier
- with precision and recall values $> .80$

Dataset obtained from http://ai.stanford.edu/~amaas/data/sentiment, Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. *49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*

Playing around with new data

```
newdata=vectorizer.transform(["What a crappy movie! It sucks!", "This is
      awsome. I liked this movie a lot, fantastic actors","I would not
     recomment it to anyone.", "Enjoyed it a lot"])
predictions = nb.predict(newdata)
print(predictions)
```

This returns, as you would expect and hope:

```
[-1  1 -1  1]
```

But we can do even better

We can use different vectorizers and different classifiers.

Different classifiers

Typical options in a nutshell:

- Naïve Bayes
- Logistic Regression
- Support Vector Machine (SVM/SVC)
- Random forests

Different vectorizers

- CountVectorizer
- TfidfVectorizer

Preprocessing:

- Stopword removal
- Pruning
- etc.

Which one would you (not) use for which purpose?

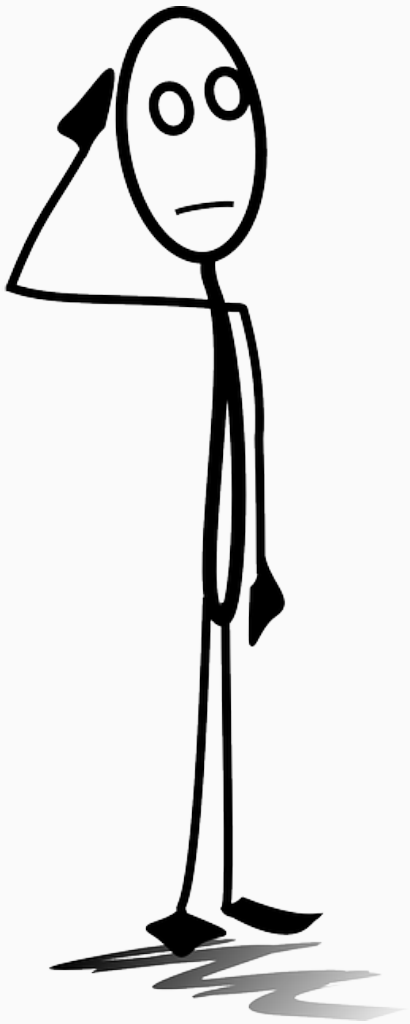
```
NB with Count
                        precision     recall
positive reviews:       0.87          0.77
negative reviews:       0.79          0.88


NB with TfIdf
                        precision     recall
positive reviews:       0.87          0.78
negative reviews:       0.80          0.88


LogReg with Count
                        precision     recall
positive reviews:       0.87          0.85
negative reviews:       0.85          0.87


LogReg with TfIdf
                        precision     recall
positive reviews:       0.89          0.88
negative reviews:       0.88          0.89
```

The problem of overfitting



Isn't training multiple models and then selecting the best one a bit like p-hacking?

Yes. We call this problem "overfitting".

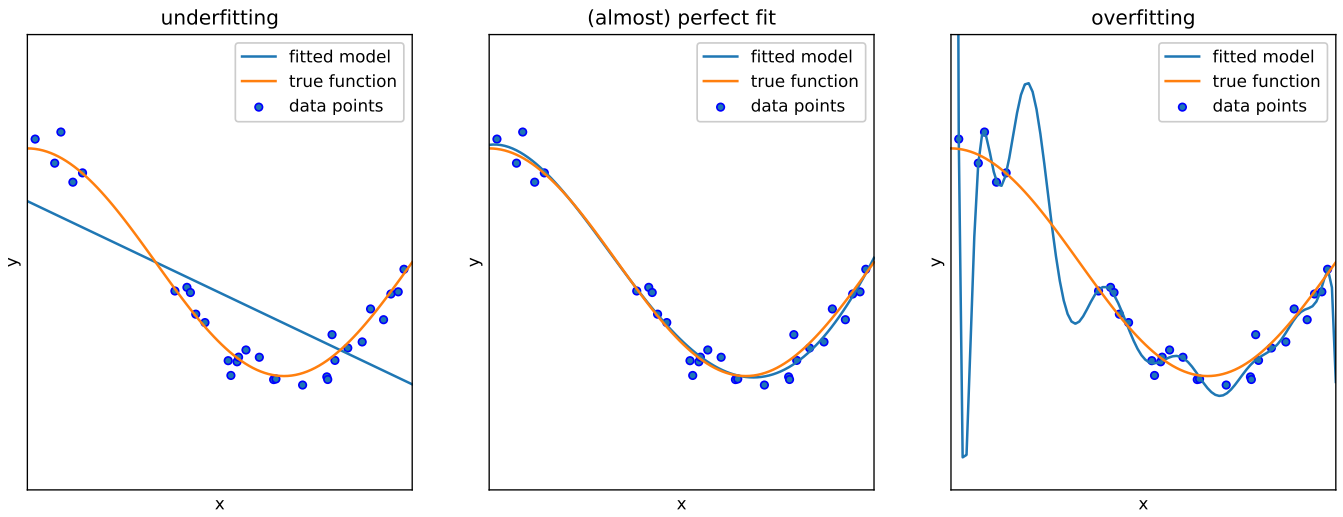


Figure 1: Underfitting and overfitting. Example adapted from https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting

How to avoid overfitting

1. **Train/test split.** We already do this. It avoids overfitting on the training data.[1]
2. **Train/validation/test split.** But maybe we overfit on the test data now? We could set aside *third* dataset.
3. ***k*-fold crossvalidation.** Extending the above such that every case is sometimes $k-1$ times part of the training data and 1 time part of the validation data.
4. **Regularization.** (in combination with the above) We "penalize" models for being too complex.

[1] In classical statistics, the R^2 of an OLS regression is prone to that. Calculating R^2 on a separate test set would be much more conservative.

How to avoid overfitting

1. **Train/test split.** We already do this. It avoids overfitting on the training data.[1]
2. **Train/validation/test split.** But maybe we overfit on the test data now? We could set aside *third* dataset.
3. ***k*-fold crossvalidation.** Extending the above such that every case is sometimes $k-1$ times part of the training data and 1 time part of the validation data.
4. **Regularization.** (in combination with the above) We "penalize" models for being too complex.

[1] In classical statistics, the R^2 of an OLS regression is prone to that. Calculating R^2 on a separate test set would be much more conservative.

How to avoid overfitting

1. **Train/test split.** We already do this. It avoids overfitting on the training data.[1]
2. **Train/validation/test split.** But maybe we overfit on the test data now? We could set aside *third* dataset.
3. ***k*-fold crossvalidation.** Extending the above such that every case is sometimes $k-1$ times part of the training data and 1 time part of the validation data.
4. **Regularization.** (in combination with the above) We "penalize" models for being too complex.

[1] In classical statistics, the R^2 of an OLS regression is prone to that. Calculating R^2 on a separate test set would be much more conservative.

How to avoid overfitting

1. **Train/test split.** We already do this. It avoids overfitting on the training data.[1]
2. **Train/validation/test split.** But maybe we overfit on the test data now? We could set aside *third* dataset.
3. ***k*-fold crossvalidation.** Extending the above such that every case is sometimes $k-1$ times part of the training data and 1 time part of the validation data.
4. **Regularization.** (in combination with the above) We "penalize" models for being too complex.

[1] In classical statistics, the R^2 of an OLS regression is prone to that. Calculating R^2 on a separate test set would be much more conservative.

The problem of overfitting

Train/validation/test split

Train/validation/test split

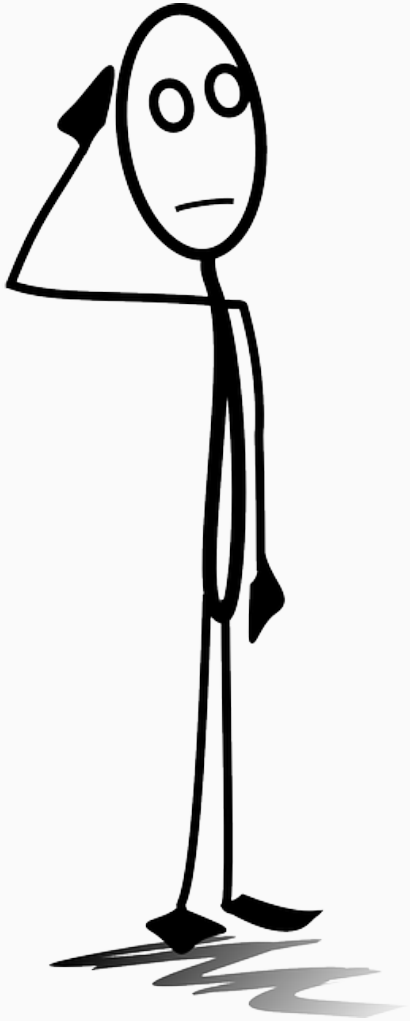
- When you compare a lot of different models (or (hyper-)parameters), you might want to evaluate (compare) them using a third dataset
- e.g., make 80/20 split (train/test); then split first part again 80/20 (train/validation)
- only use the test data *at the very end* to get a final estimate of how good your model is.

In short: Validation data to *select* the best approach; test data to get the accuracy of the approach you chose.

Train/validation/test split

- When you compare a lot of different models (or (hyper-)parameters), you might want to evaluate (compare) them using a third dataset
- e.g., make 80/20 split (train/test); then split first part again 80/20 (train/validation)
- only use the test data *at the very end* to get a final estimate of how good your model is.

In short: Validation data to *select* the best approach; test data to get the accuracy of the approach you chose.



Any questions?

Things to remember

- unsupervised vs supervised
- rough understanding of different techniques and when to use them
- evaluation metrics (e.g., precision, recall)

Let's do an exercise!

References

Boukes, M., van de Velde, B., Araujo, T., & Vliegenthart, R. (2020). **What's the Tone? Easy Doesn't Do It: Analyzing Performance and Agreement Between Off-the-Shelf Sentiment Analysis Tools.** *Communication Methods and Measures*, *14*(2), 83–104. https://doi.org/10.1080/19312458.2019.1671966

Boumans, J. W., & Trilling, D. (2016). **Taking stock of the toolkit: An overview of relevant autmated content analysis approaches and techniques for digital journalism scholars.** *Digital Journalism*, *4*(1), 8–23. https://doi.org/10.1080/21670811.2015.1096598

Burscher, B., Vliegenthart, R., & De Vreese, C. H. (2015). **Using supervised machine learning to code policy issues: Can classifiers generalize across contexts?** *The ANNALS of the American Academy of Political and Social Science*, *659*(1), 122–131. https://doi.org/10.1177/0002716215569441

Hopkins, D. J., & King, G. (2010). **A method of automated nonparametric content analysis for social science.** *American Journal of Political Science*, *54*(1), 229–247.