Text Analysis with NLTK

Week 2

13 November 2020

CDCS Python Course Series

with Lucy Havens

Course Topics

Text analysis

Introduction to tools in Natural Language Toolkit (NLTK)

Structuring code around a research question

Using GitHub

Recap: Research with NLTK

The building blocks:

- Word tokenization
- Sentence tokenization
- Casefolding - *normalize text by lowercasing all words*
- Part-of-speech tags

Working with Contractions

Whitespace	Isn't	Ahab,	Ahab?	;)				
Treebank	Is	n't	Ahab	,	Ahab	?	;	)
Tweet	Isn't	Ahab	,	Ahab	?	;)		
TokTok (Dehdari, 2014)	Isn	'	t	Ahab	,	Ahab	?	;)

Figure 4.1: The output of four NLTK tokenizers, applied to the string *Isn't Ahab, Ahab? ;)*

Reference: Jacob Eisenstein (2018) Chapter 4, Natural Language Processing.

Similar trade-offs with reducing to roots

Original	The	Williams	sisters	are	leaving	this	tennis	centre
Porter stemmer	the	william	sister	are	leav	thi	tenni	centr
Lancaster stemmer	the	william	sist	ar	leav	thi	ten	cent
WordNet lemmatizer	The	Williams	**sister**	are	leaving	this	tennis	centre

Figure 4.2: Sample outputs of the Porter (1980) and Lancaster (Paice, 1990) stemmers, and the WORDNET lemmatizer

Reference: Jacob Eisenstein (2018) Chapter 4, Natural Language Processing.

Recap: Research with NLTK

Going further:

- Syntactic structure
  - For example: sentence structures represented as trees
- Shallow semantics
  - For example: named entity recognition, coreference resolution
- Dialogue and discourse
  - For example: dialog act tags, rhetorical structure

For more detail: https://www.nltk.org/book/ch06.html

Recap: Research Example

To identify **places** named in a text:

1. Tokenize words
2. Tokenize sentences
3. Tag parts of speech
4. Chunk or label to annotate named entities
5. Compare named places with places in a gazetteer

For more:
https://programminghistorian.org/en/lessons/geoparsing-text-with-edinburgh
https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da

Git and GitHub

Version control system

Code on your own computer

Local

Merge code with others in an online repository (“repo”)

Upstream

Reference: https://guides.github.com/introduction/git-handbook/

Assignment

Helpful Resources - **for reference**, not required reading

Pick your own text to analyze! (ideas provided in case you're stuck)

Go Further - **optional** tutorials

What datasets did you use?

What questions did you ask?

Was it easier, harder, or about the same as you expected?

So these out-of-the-box tools….

These **machine learning (ML) algorithms** or **models**

Examples:

- Punkt Tokenizer
- Lancaster Stemmer
- Porter Stemmer
- WordNet Lematizer

…how do they work?

Classification

Text analysis with machine learning (ML)

Multi-class, open-class, or sequence

Supervised, semi-supervised, unsupervised

Reference: https://www.nltk.org/book/ch06.html

Classification

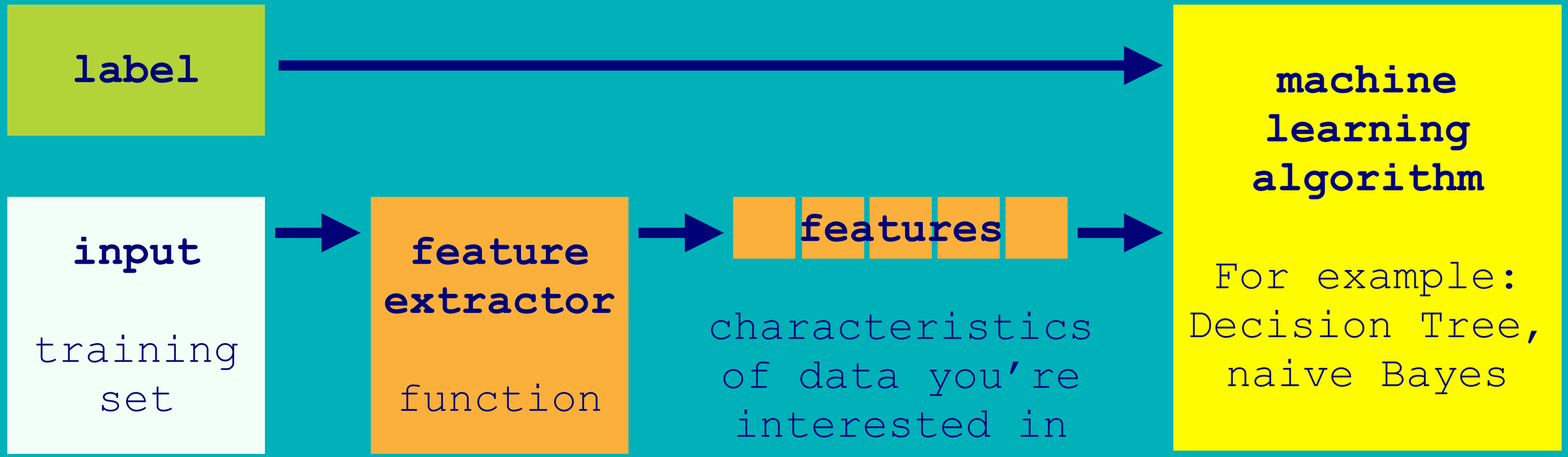
Text analysis with machine learning (ML)

Multi-class, open-class, or sequence

Supervised, semi-supervised, unsupervised

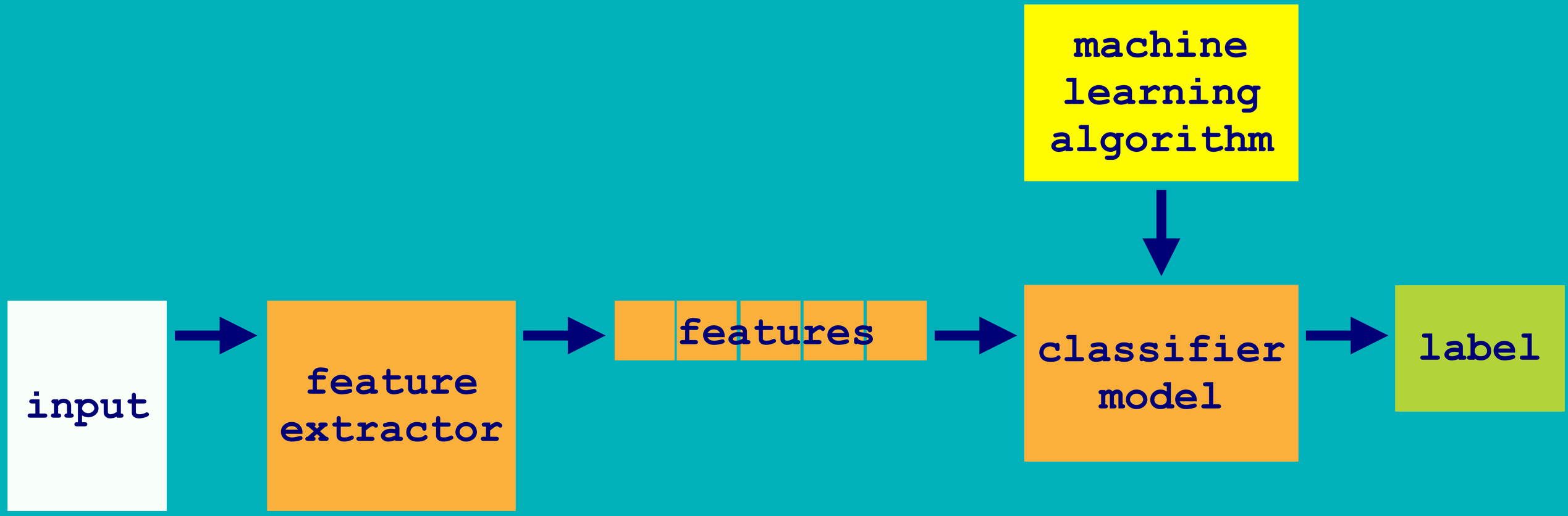
Reference: https://www.nltk.org/book/ch06.html

How a Classifier Works: **Training**



Reference: https://www.nltk.org/book/ch06.html

How a Classifier Works: **Prediction**



Reference: https://www.nltk.org/book/ch06.html

Classification Example: Sentiment Analysis

Can be binary

- Positive/Negative

Can be categorical

- Emotions - surprise, happiness, fear, disgust, sadness, etc.
- Stances - for/against

Can be scored

- Number indicating how strongly sentiment/emotion expressed

Reference: Daniel Jurafsky, James H. Martin. (2019). Chapter 21. Speech and Language Processing. 3rd Edition

Building a Supervised Classifier

Step 1: Choose **features**

Step 2: Define a **feature extractor** (a function)

Step 3: Prepare a list of labeled examples

Step 4: Process data with the feature extractor

Step 5: Divide the output of step 4 into a training set and test set

Step 6: Train a **classifier** on the training set

Step 7: Examine the **likelihood ratios**

How do you choose a classifier?

Step 1: Choose **features**

Step 2: Define a **feature extractor** (a function)

Step 3: Prepare a list of labeled examples

Step 4: Process data with the feature extractor

Step 5: Divide the output of step 4 into a training set and test set

Step 6: Train a **classifier** on the training set

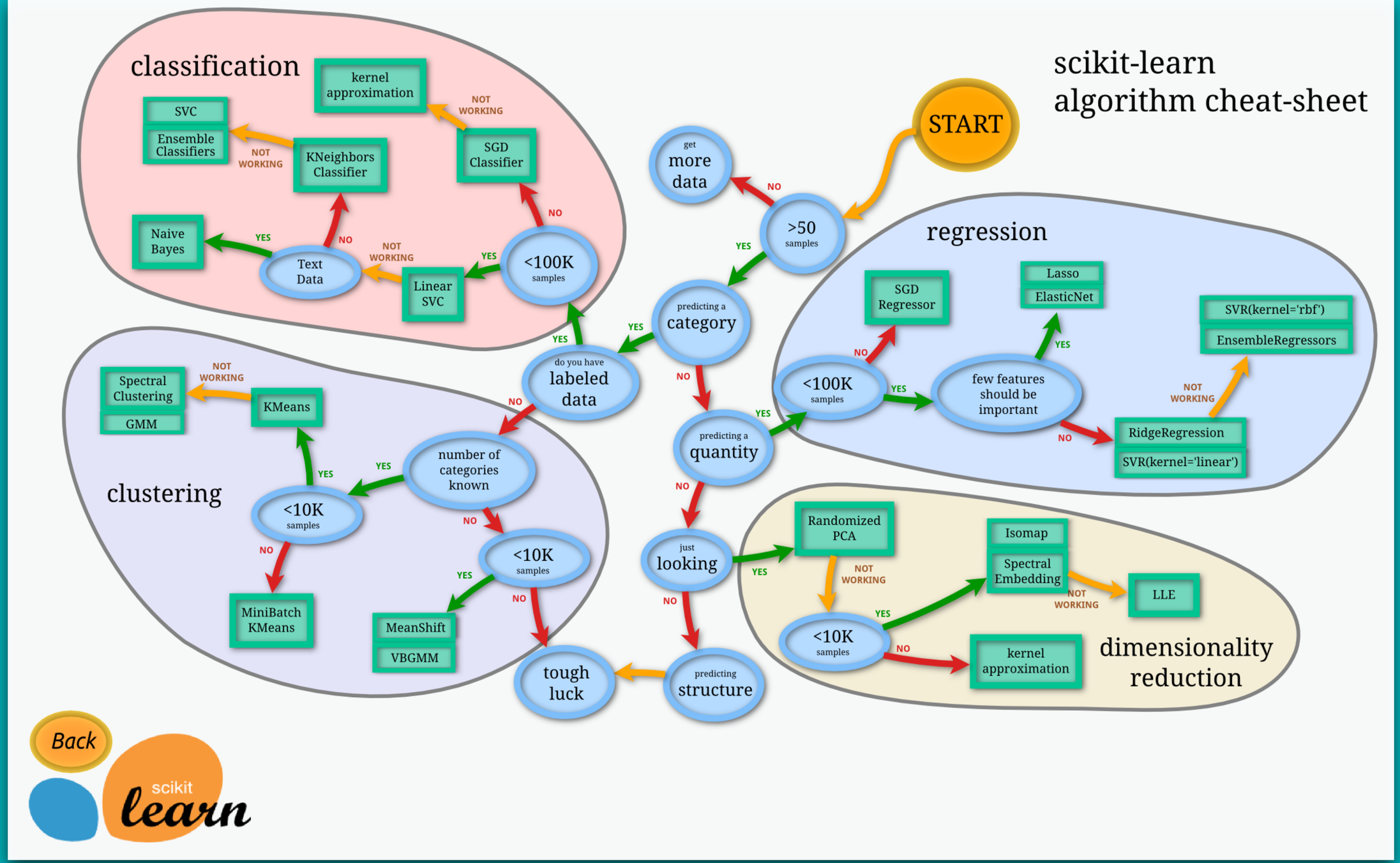
Step 7: Examine the likelihood ratios

How do you choose a classifier?

Consider what kind of text you're using and what kind of text a classifier was trained on

Consider what type of classification you're looking for (binary, categorical)

Start simple - **there are a lot of out-of-the-box tools!**



Reference: https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

DEMO

Evaluating Classification Models

Run the model on a test set and then measure…

- **Accuracy**: percentage of test set inputs that the model correctly labels

$$\frac{\texttt{correct predictions (labels given)}}{\texttt{total number of data points (inputs)}}$$

Evaluating Classification Models

- **Precision**: items model gave label A that should have label A

$$\frac{\texttt{true positives}}{\texttt{true positives + false positives}}$$

- **Recall**: items that should have label A that model gave label A

$$\frac{\texttt{true positives}}{\texttt{true positives + false negatives}}$$

Evaluating Classification Models

- **F-score** or **F-measure**: harmonic mean of precision & recall

$$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Feedback please!

Thank you!

Looking to learn more Python? Try the upcoming course on network analysis and data visualization!

30th November - 11th December