Exercise: Bayesian Learning I
Data Science Specialization (Spring 2025)

Jens Classen
IMT, Roskilde University
jens-classen.net
classen@ruc.dk

26.03.2025

In this exercise, we want to explore the application of the Naive Bayes classifier on some simple text classification tasks such as identifying spam SMS and detecting the sentiments of tweets. In Moodle, you find two corresponding datasets:

- **SMSSpamCollection**
  Source: `https://archive.ics.uci.edu/ml/datasets/sms+spam+collection`
  The first column contains the target ("spam" or "ham"), the second a raw text message.
- **kaggle_tweet_sentiments.csv**
  Source: `https://www.kaggle.com/competitions/tweet-sentiment-extraction`
  The columns contain the ID, raw text, selected text, and the sentiment (target class). For this exercise we won't need the ID and selected text columns.

Also in Moodle, there is a template for a Jupyter notebook that imports (some of) the libraries needed and details the steps.

1 Vectorization

The first problem is that the dataset contains raw text that we have to bring into a form that the algorithm can work with. For this purpose we use *vectorization*. The idea is to split the raw texts such as

- Tweet1: "`How awesome!`"
- Tweet2: "`How very bad.`"

into separate words and count how often and/or if they occur in the different texts:

Word	Tweet1	Tweet2
`How`	1	1
`awesome`	1	0
`bad`	0	1
`apple`	0	0
`very`	0	1

In this example, Tweet1 is represented by a vector $v_1 = [1, 1, 0, 0, 0]$ and Tweet2 by $v_2 = [1, 0, 1, 0, 1]$.

In SciKit-Learn, we can use `sklearn.feature_extraction.text.CountVectorizer` for this purpose. There are multiple options to consider:

- **ngram_range**: In natural language processing (NLP), one distinguishes *unigrams*, *bigrams*, and more generally "n-grams" for arbitrary $n > 0$. Unigrams refer to single words (a unigram representation is often called "bag of words"), bigrams to sequences of two words, etc. Note that the number of possible n-grams grows exponentially in n, so often only unigrams and/or bigrams are used due to memory constraints.
- **binary**: Do we want to *count* how often a word/n-gram is in the text or just distinguish whether or not it occurs.?
- **stop_words**: In NLP one often applies a number of preprocessing steps before working with text. *Stop words* denote words such "the", "a", etc. that are so common that they carry almost no information, and are hence removed.
- **lowercase**: Another preprocessing step is to turn all words to lowercase, so that "`Apple`" is identified with "`apple`".

2 Training

SciKit-Learn comes with implementations of Naive Bayes classification algorithms. In particular, we have

- `sklearn.naive_bayes.MultinomialNB`: Naive Bayes classifier for *multinomial* distributions that use discrete integer features such as word counts.
- `sklearn.naive_bayes.BernoulliNB`: Like `MultinomialNB`, but works with binary (Boolean) features instead of occurence counts.

3 Task

Apply the different classification techniques to the given datasets and evaluate their performance. How good do they perform, and what influence do the various options and preprocessing steps have? How well do they perform on other datasets, e.g. with longer texts? Can other forms of vectorization help, e.g. using TFIDF (*term frequency, inverse document frequency*)?