CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Machine Learning

Machine Learning

Lecture: Bayesian Classification

Ted Scully

Pre-processing for Document Classification using Naïve Bayes

- Quite often a range of pre-processing activities can be used to clean the data prior to it's usage by Naïve Bayes.

- These pre-processing steps can include very basic steps such as removal of **punctuation**, **URLs** and **lower-casing** all words. The objective of many of these techniques is reducing the number of features (words) in the dataset.

- However, there is a host of more advanced techniques that we can also apply and may improve classification accuracy. Many of these techniques are available in Python's NLTK.

Stemming and Lemmatization

- Stemming and Lemmatization
- Commonly documents are going to use different forms of the same word, such as organize, organizes, and organizing or fishes, fishing, fish .
- In addition we also have words that are highly related such as with democracy and democratic.

- Stemming and lemmatization attempt to truncate words to their stem or root word.
  - A stemmer for English, for example, should identify the string "fishing", "fished", and "fisher" to the root word, "fish“, and "stemmer", "stemming", "stemmed" as based on "stem".
  - Porter’s Stemming Algorithm (There are stemmers available from the natural language toolkit in Python)
  - Typically a stemming algorithm will truncate existing words to form the root.
  - In contrast lemmatization attempts to do this by using a vocabulary.

Stemming Example

- 'cats cacti geese rocks python wolves‘

cats -- cat
cacti -- cacti
geese -- gees
rocks -- rock
python -- python
wolves -- wolv

Lemmatization Example

- 'cats cacti geese rocks python wolves'”

cats -- cat
cacti -- cactus
geese -- goose
rocks -- rock
python -- python
wolves -- wolf

Emoticons, Stop-Words, Misspelled Words

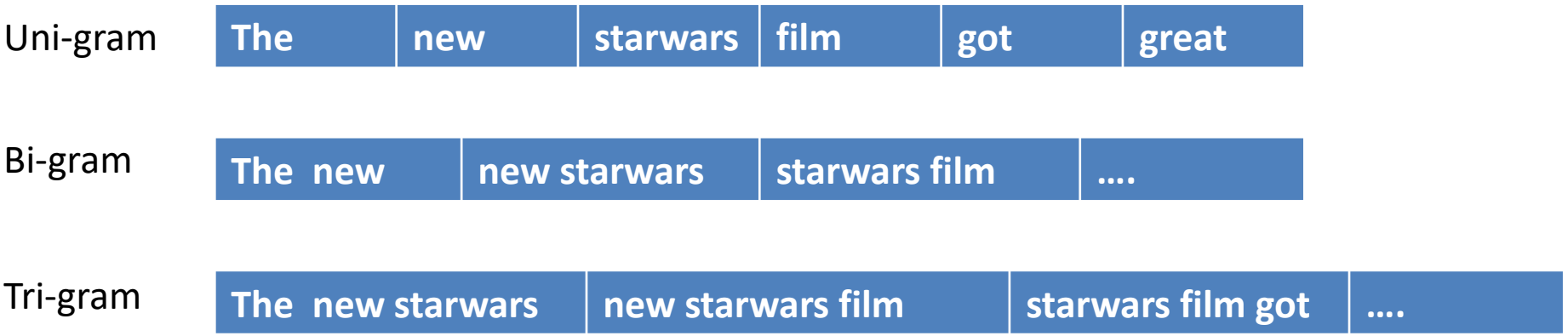
- It is important in the process of sentiment analysis to identify the graphical cues for sentiment as represented by **emoticons**
  - One common approach is to use a dictionary that has emoticons labelled according to their emotional state.
  - For example, ":)" is labelled as positive whereas ":-(" is labelled as negative. Commonly each emoticon is given one of the following labels
    - Extremely-positive, Extremely-negative, Positive, Negative, Neural

- Another common parsing techniques is the **removal of stop-words**. There are freely available dictionaries of stopwords (http://xpo6.com/list-of-english-stop-words/) NLTK also provides a stopword dictionary.

- Detection and correction of **misspelled words** using a dictionary (using tool such as PyEnchant).

N Grams

- In the n-gram model, a token can be defined as a sequence of n items.
- The simplest case is the so-called **unigram (1-gram)** where each **token** consists of exactly **one word**.
- Everything that we have looked at so far has been uni-gram.
- In a bi-gram (2-gram) each token consists of **two adjacent works**, in a tri-gram (3-gram) it consists of **three adjacent words**.
- N-grams can often have a positive impact on accuracy but also significant increase the number of features (hence the size of your vocabulary)



Dealing with Continuous Variables

- So far we dealt only with categorical features and to calculate the probability of an event, we have just counted how often the event occurred and divided this by how often the event could have occurred.

- Clearly adopting the above approach is not practical for a continuous features because it can have an **infinite number of values in it's domain.**

- One common approach to dealing with this issue is binning.

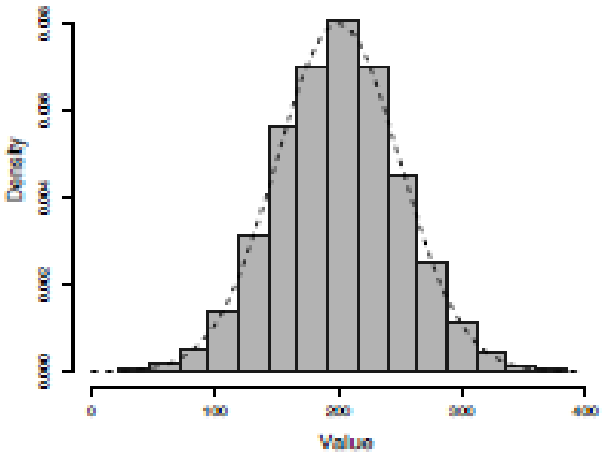
Headach	Fever	Vomit	Meningitis
True	True	False	False
False	True	False	False
True	False	True	False
True	False	True	False
False	True	False	True
True	False	True	False
True	False	True	False
True	False	True	True
False	True	False	False
True	False	True	True

Binning Continuous Variables

- A alternative approach to dealing with continuous features is to convert them into categorical variables using binning.

- To perform binning, we define a series of ranges (called bins) for the continuous feature that correspond to the levels of the new categorical feature we are creating.

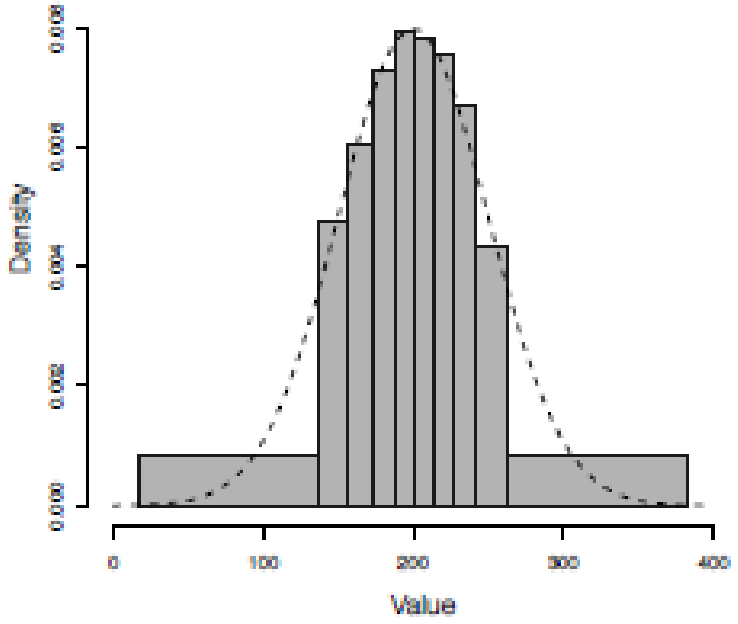
- **Equal-width binning** - The equal-width binning algorithm splits the range of the feature values into b bins each of size range/b.



Binning Continuous Variables

- **Equal-frequency binning** first sorts the continuous feature values into ascending order and then places an equal number of instances into each bin, starting with bin 1.

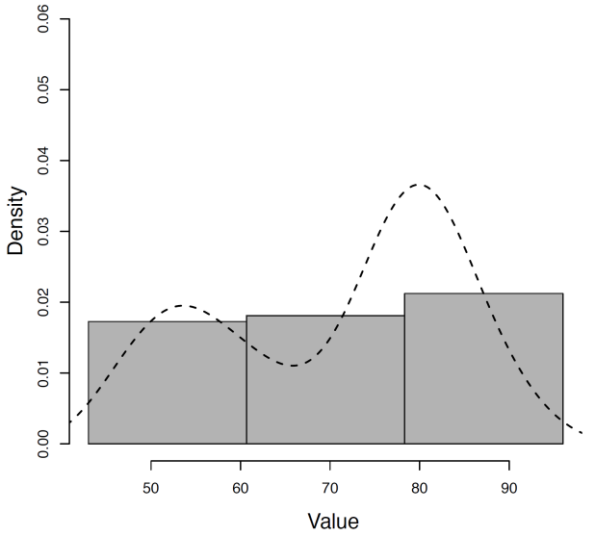
- The number of instances placed in each bin is simply the total number of instances divided by the number of bins, b..

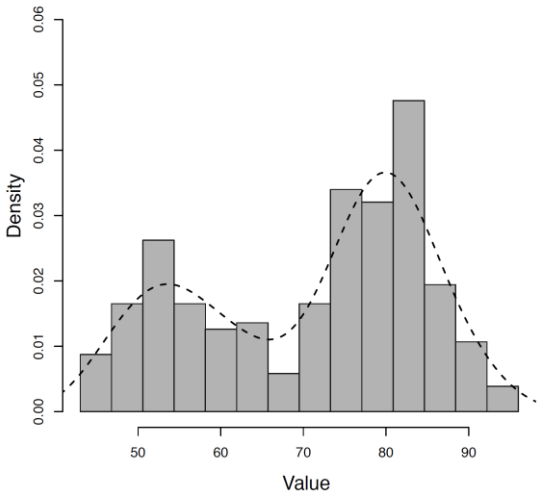


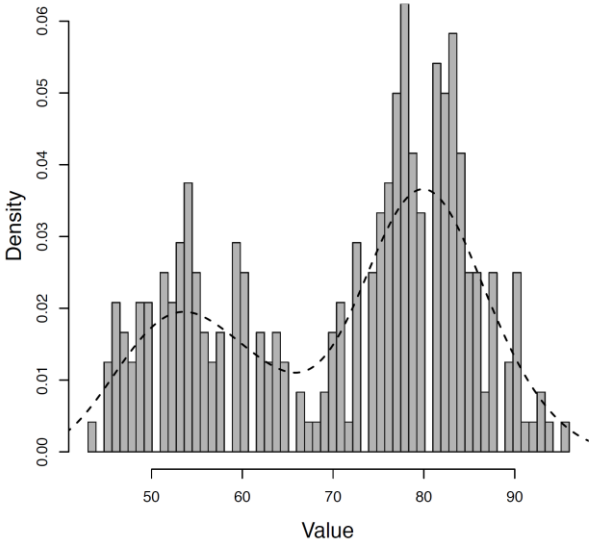
Selecting the Number of Bins

- If we select the number of bin **very low** then we **lose useful information** about the distribution of the feature. However, each bin has a large number of instances.

- If we select a value that is too high then the bins will provide a better representation of the distribution of the feature but each of the individual bins contain fewer instances.

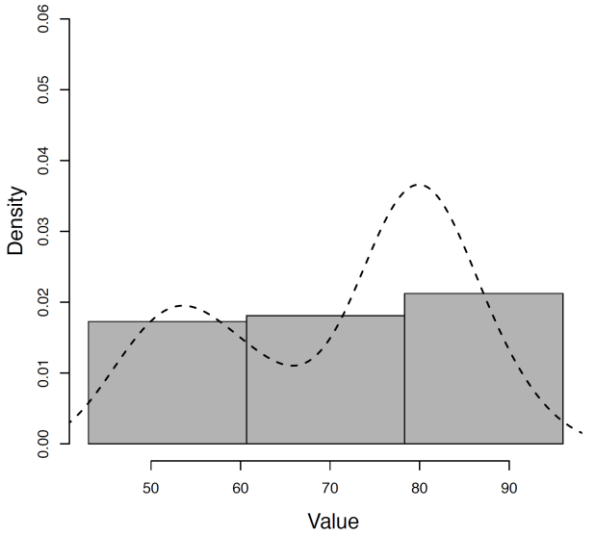


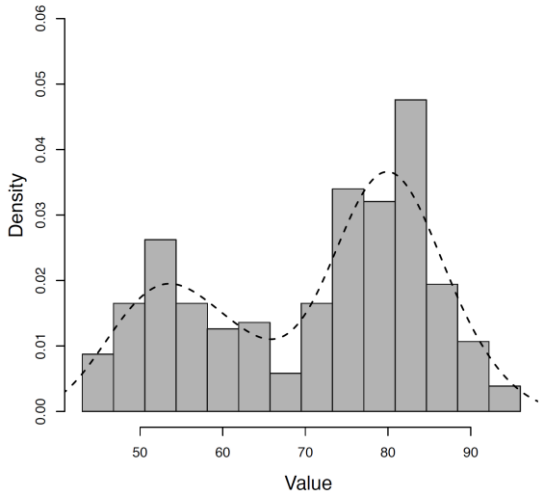


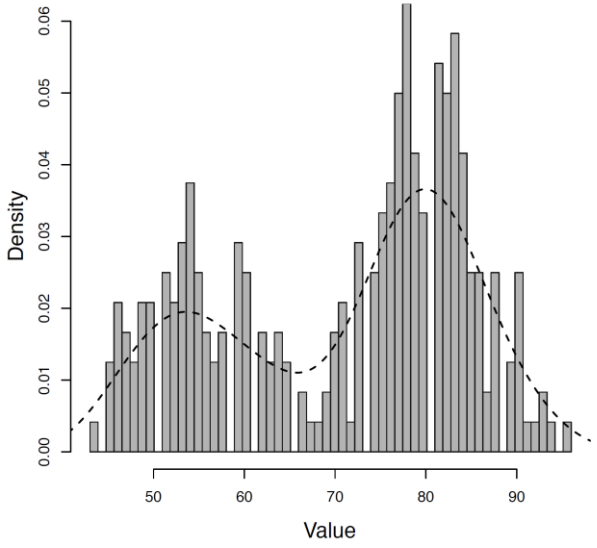


Selecting the Number of Bins

- Unfortunately there is no way of guaranteed method of finding the optimal number of bins to represent a continuous valued feature.
- It is often treated as a parameter that can be adjusted through experimentation.







Strengths of Naïve Bayes

- Training Set Size and Speed
  - Naïve Bayes is a **very fast algorithm**
    - The process of calculating the probabilities is the only potentially time consuming component.

  - Another advantage of Naïve Bayes is that it is a probabilistic classifier so unlike many other algorithms it provides **some degree of certainty** in it's conclusions.
    - For example, we may only wish to classify the polarity of a tweet if we are more than 75% confident that the tweet is positive or negative (see previous slides).

Results for Twitter Sentiment Analysis

		Confidence Prediction		
		50%	**70%**	**90%**
Baseline Naïve Bayes	% Accuracy	73.4	81.4	86.2
	% Predicted	100	75.3	42.5
Bayes with Pre-processing	% Accuracy	76.5	84.2	90.2
	% Predicted	100	73.2	43.5

Strengths of Naïve Bayes

- Naïve Bayes is **less sensitive to irrelevant features**...
  - Suppose we are trying to classify a persons sex based on several features, including eye colour (Of course, eye colour is completely irrelevant to a persons gender)
  - How would Naïve Bayes deal with such an irrelevant features.

Strengths of Naïve Bayes

- Naïve Bayes is **less sensitive to irrelevant features**...
  - Suppose we are trying to classify a persons gender based on several features, including eye colour (Of course, eye colour is completely irrelevant to a persons gender)
  - How would Naïve Bayes deal with such an irrelevant attribute.

p(eye = brown | female) * p(wears_dress= yes | female) * …..

p(eye = brown | male) * p(wears_dress= yes | male) * …..

p(eye = brown | female) * p(wears_dress= yes | female) * …..
=> 5000/10000 * 9,975/10000

p(eye = brown | male) * p(wears_dress= yes | male) * …..
=> 5000/10000 * 25/10000

Weakness of Naïve Bayes

- Naïve Bayes is primarily a classification algorithm. While studies have adapted NB as for **regression** problems it's performance on such problems has been generally poor.

- The "Naive" term comes from the fact that the model **assumes that all features are fully independent** given the class, which in real problems they almost never are.

- In practice this approach still works reasonably well for many real-world problems.
- However, we can adopt a more realistic approach that will incorporate certain dependencies amongst the variables in our domain using Bayesian Networks.