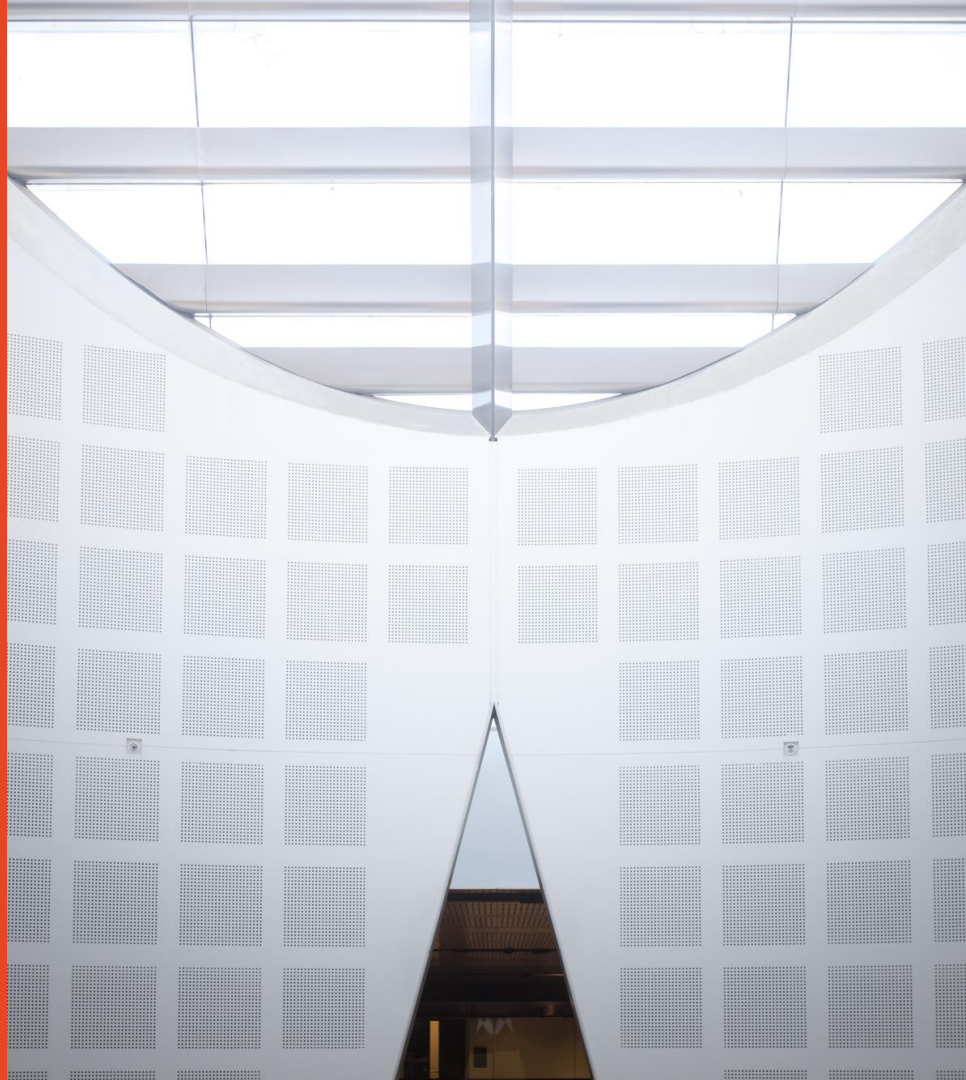
COMP5310: Principles of Data Science

W11: Unstructured Data– Naïve Bayes

Presented by
Maryam Khanian
School of Computer Science

Based on slides by previous lecturers of this unit of study

THE UNIVERSITY OF SYDNEY
The University of Sydney



Last week: Decision trees

Objective

Learn techniques for supervised machine learning, with tools in Python.

Lecture

- Decision tree
- Build a good solution
- Evaluation setup
- Communicate results

Readings

- Data Science from Scratch, Ch. 17

Exercises

- sklearn: decision tree
- sklearn: random forest

Supervised Learning:

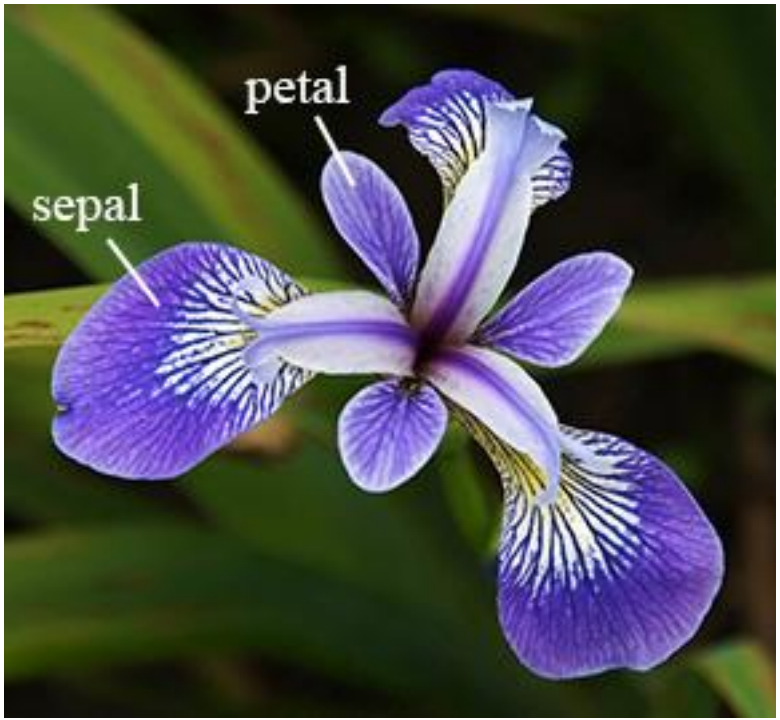
- We'll now focus on supervised machine learning techniques
  - ☑ *Simple linear regression*
  - ☑ *Multiple linear regression*
  - ☑ *Logistic regression*
  - ☑ *Decision tree*
  - **Naïve Bayes**

Unstructured data refers to information that does not have a pre-defined data model.

Unstructured data is typically text-heavy, but may contain dates, numbers and facts as well.

This results in ambiguities that make it more difficult to understand than data in structured databases.

Structured data



- Fielded data
- Stored in databases
- E.g.:
  - Sensor data
  - Financial data
  - Click streams
  - Measurements

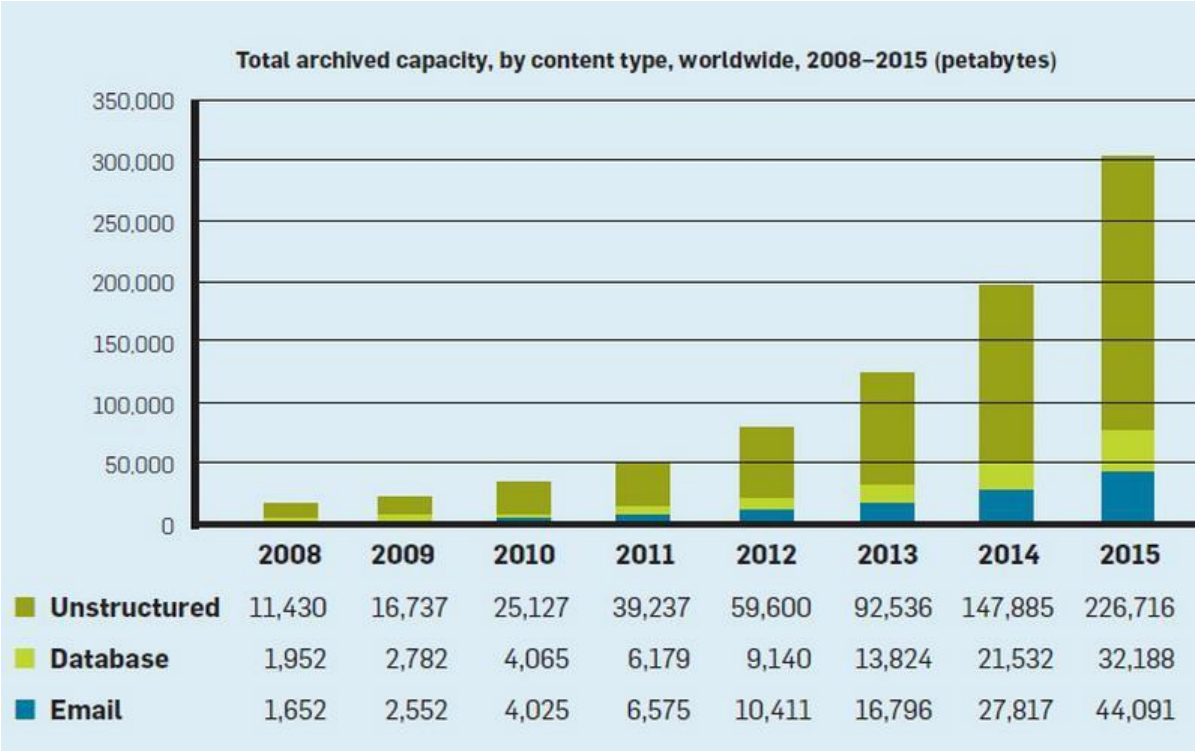
Unstructured data

“In the history of cinematic mustaches, few have been as disgusting as that of Rye Gerhardt (Kieran Culkin), the youngest scion of North Dakota’s reigning crime family and the stray spark that sets off the powder-keg second season of Fargo.”

- 80-90% of all potentially usable business information
- E.g.:
  - Images
  - Video
  - Email
  - Social media

(From a Slate review of Fargo Season 2)

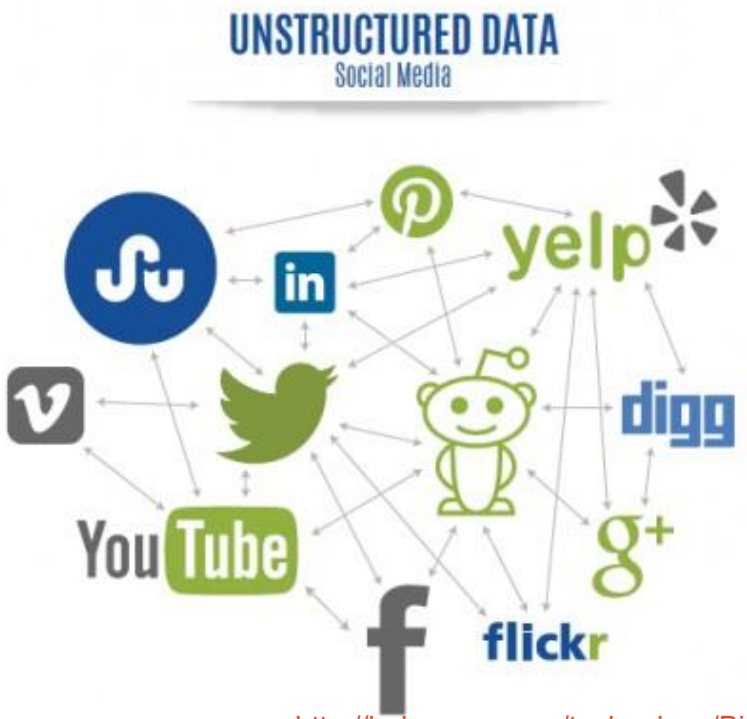
Information overload



	2008	2009	2010	2011	2012	2013	2014	2015
Unstructured	11,430	16,737	25,127	39,237	59,600	92,536	147,885	226,716
Database	1,952	2,782	4,065	6,179	9,140	13,824	21,532	32,188
Email	1,652	2,552	4,025	6,575	10,411	16,796	27,817	44,091

http://bhavnaober.blogspot.com.au/2015_02_01_archive.html

Social media data

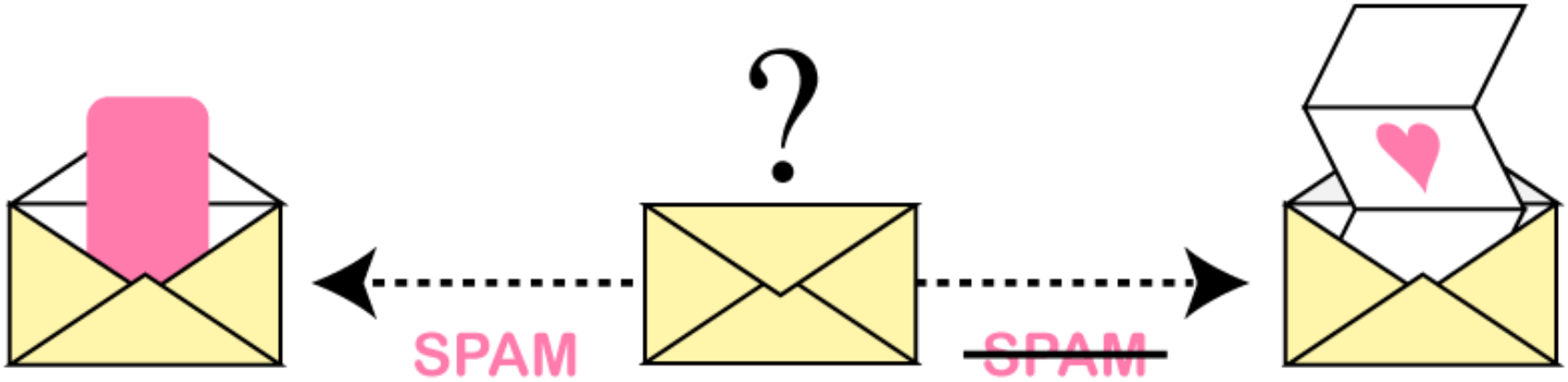


http://hubpages.com/technology/Big-Data-Understanding-New-Insights#

Text Categorization

THE UNIVERSITY OF SYDNEY

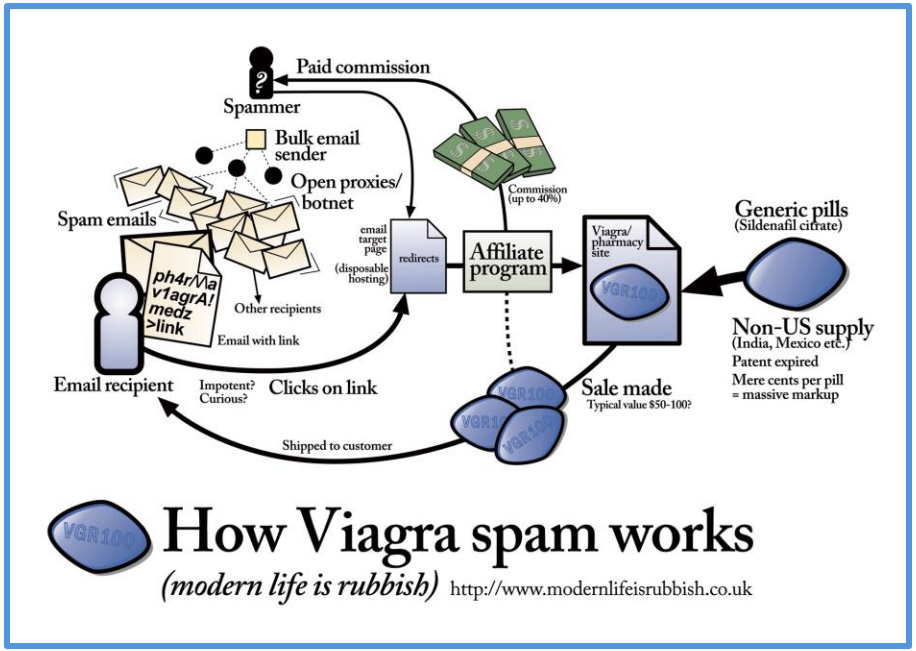
Task: Spam/ham detection



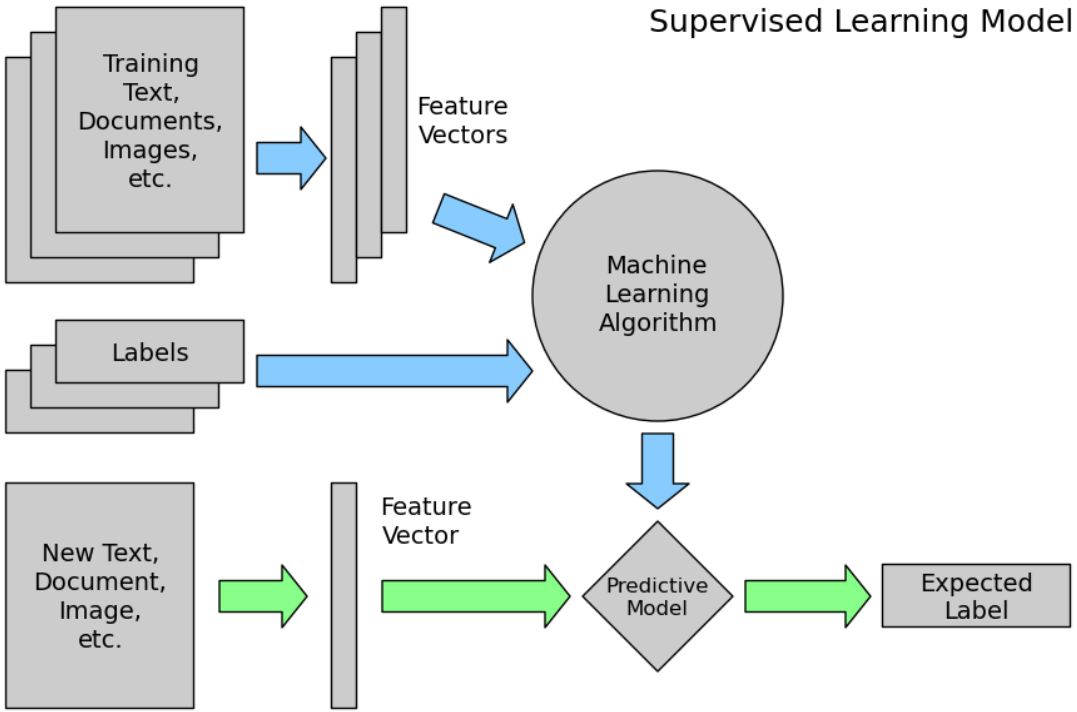
http://blog.dato.com/how-to-evaluate-machine-learning-models-part-2a-classification-metrics

Modelling spam detection

- Input:
  - Emails
  - SMS messages
  - Facebook pages
- Predict:
  - 1 (spam)
  - 0 (ham)



Spam detection as supervised classification



http://www.astroml.org/sklearn_tutorial/general_concepts.html#supervised-learning-model-fit-x-y

Feature vectors from text

- Represent document as a multiset of words
- Keep frequency information
- Disregard grammar and word order

http://www.python-course.eu/text_classification_python.php

Tokenisation

“Friends, Romans, Romans, countrymen”

[“Friends”,
“Romans”,
“Romans”,
“countrymen”]

- Split a string (document) into pieces called tokens
- Possibly remove some characters, e.g., punctuation

Normalisation

[“Friends”,
“Romans”,
“Romans”,
“countrymen”]

[“friend”,
“roman”,
“roman”,
“countrymen”]

- Map similar words to the same token
- Stemming/lemmatisation
  - Avoid grammatical and derivational sparseness
  - E.g., “was” => “be”
- Lower casing, encoding
  - E.g., “Naïve” => “naive”

Indicator features

[“friend”,
“roman”,
“roman”,
“countrymen”]

{“friend”: 1,
“roman”: 1,
“countrymen”: 1}

- Binary indicator feature for each word in a document
- Ignore frequencies

Term frequency weighting

[“friend”,
“roman”,
“roman”,
“countrymen”]

{“friend”: 1,
“roman”: 2,
“countryman”: 1}

- Term frequency
  - Give more weight to terms that are common in document
  - *TF = |occurrences of term in doc|*
- Damping
  - Sometimes want to reduce impact of high counts
  - *TF = log(|occurrences of term in doc|)*

TFIDF Weighting

[“friend”,
“roman”,
“countrymen”]

{“friend”: 0.1,
“roman”: 0.8,
“countrymen”: 0.2}

- Inverse document frequency
  - Give less weight to terms that are common across documents
  - ***IDF = log(| docs | / | docs containing term |)***
- TFIDF
  - ***TFIDF = TF * IDF***

Naïve Bayes Classifier

THE UNIVERSITY OF SYDNEY

Naïve Bayes

- It is a probabilistic classifier based on Bayes' Theorem
- It has a nice intuitive appeal
- It has been successfully used for many purposes, but it works particularly well with natural language processing (NLP) problems.

Prediction Based on Bayes' Theorem

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

- This can be viewed as

  posteriori = likelihood x prior/evidence

Classification is to Derive the Maximum Posteriori

- Let D be a training set of tuples, and each tuple is represented by an n-dimensional attribute vector $X = (x_1, x_2, \ldots, x_n)$
- There are m class labels $C_1, C_2, \ldots, C_m$ associated with D.
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|X)$
- This can be derived from Bayes' theorem $P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$
- Since P(X) is constant for all classes

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes): that's why it is called "naïve"

$$P(\mathbf{X} \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i) = P(x_1 \mid C_i) \times P(x_2 \mid C_i) \times \ldots \times P(x_n \mid C_i)$$

An Example

Class:

C1: Interviewed well = 'False'

C2: Interviewed well = 'True'

Data to be classified:

X = (Level =Senior,

Lang = Python,

Tweets = yes

PhD = No)

14

Level	Lang	Tweets	PhD	Interviewed well
Senior	Java	No	No	False
Senior	Java	No	Yes	False
Mid	Java	No	No	True
Junior	Python	No	No	True
Junior	R	Yes	No	True
Junior	R	Yes	Yes	False
Mid	R	Yes	Yes	True
Senior	Python	No	No	False
Senior	R	Yes	No	True
Junior	Python	Yes	No	True
Senior	Python	Yes	Yes	True
Mid	Python	No	Yes	True
Mid	Java	Yes	No	True
Junior	Python	No	Yes	False

Calculation

X = (Level =Senior, Lang = Python, Tweets = yes, PhD = No)

We need to compute $\mathbf{P(C_i \mid X) = P(X \mid C_i) * P(C_i)}$

$P(C_{True})$: P(Interviewed well = "True") = 9/14 = 0.643

$P(C_{Fasle})$: P(Interviewed well = "False") = 5/14= 0.357

Compute $P(X \mid C_{True})$

P(Level = "Senior" | Interviewed well = "True") = 2/9 = 0.222

P(Lang = "Python" | Interviewed well = "True") = 4/9 = 0.444

P(Tweets = "Yes" | Interviewed well = "True") = 6/9 = 0.667

P(PhD = "No" | Interviewed well = "True") = 6/9 = 0.667

Compute $P(X \mid C_{False})$

P(Level = "Senior" | Interviewed well = "False") = 3/5 = 0.6

P(Lang = "Python" | Interviewed well = "False") = 2/5 = 0.4

P(Tweets = "Yes" | Interviewed well = "False") = 1/5 = 0.2

P(PhD = "No" | Interviewed well = "False") = 2/5 = 0.4

$P(X \mid C_i)$:

P(X | Interviewed well = "True") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

P(X | Interviewed well = "False") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(C_i \mid X) = P(X \mid C_i) * P(C_i)$:

P(X | Interviewed well = "True") * P(Interviewed well = "True") = 0.044 * 0.643 = **0.028**

P(X | Interviewed well = "False") * P(Interviewed well = "False") = 0.019 * 0.357 = 0.007

Since **$P(C_{true} \mid X) > P(C_{false} \mid X)$** , therefore X belongs to class (Interviewed well = "True")

Text Classification

Text Classification

Text	Category
"A great game"	Sports
"the election was over"	Not sports
"Very clean match"	Sports
"A clean but forgettable game"	Sports
"It was a close election"	Not sports

Our goal is to build a Naïve Bayes classifier that will tell us which category the sentence "A very close game" belongs to.

https://medium.com/syncedreview/applying-multinomial-naive-bayes-to-nlp-problems-a-practical-explanation-4f5271768ebf

Text Classification

- In our case, the probability that we wish to calculate can be calculated as:

$$p(\text{Sports} \mid \text{a very close game}) = \frac{p(\text{a very close game} \mid \text{Sports}) \times p(\text{Sports})}{p(\text{a very close game})}$$

$$p(\text{Not sports} \mid \text{a very close game}) = \frac{p(\text{a very close game} \mid \text{Not sports}) \times p(\text{Not sports})}{p(\text{a very close game})}$$

- Because we are only trying to find out which category (Sports or Not Sports) has a higher probability, it makes sense to discard the divisor **P(a very close game)**, and compare only:

$$p(\text{a very close game} \mid \text{Sports}) \times p(\text{Sports})$$

with

$$p(\text{a very close game} \mid \text{Not sports}) \times p(\text{Not sports})$$

Text Classification

- Now, we can write the probability we wish to calculate

$$p(\text{a very close game} \mid \text{Sports}) = p(a \mid \text{Sports}) \times p(\text{very} \mid \text{Sports}) \times p(\text{close} \mid \text{Sports}) \times p(\text{game} \mid \text{Sports})$$

Similarly

$$p(\text{a very close game} \mid \text{Not Sports}) = p(\text{a} \mid \text{Not Sports}) \times p(\text{very} \mid \text{Not Sports}) \times p(\text{close} \mid \text{Not Sports}) \times p(\text{game} \mid \text{Not Sports})$$

- But the word "close" does not exist in the category Sports, thus $p(\text{close} \mid \text{Sports}) = 0$, leading to $p(\text{a very close game} \mid \text{Sports}) = 0$

- Thus loosing the probabilities information of other words.

Laplace smoothing

- $p(w|c) = \frac{\text{count(w,c)}+1}{\text{count(word,c)}+\text{count(word)}}$
  - count(w,c) = Count of word w in class c
  - count(word,c) = Count of all words in class c
  - count(word) = Count of all distinct words in the dataset

- Now we can calculate the probability again :

$$p(\text{close} \mid \text{Sports}) = \frac{(0+1)}{(11+14)}$$

Calculation

p(Sports | a very close game) =?
p(Not Sports | a very close game) =?

w	$p(w \mid \text{Sports})$	$p(w \mid \text{Not Sports})$
a	$\frac{(2+1)}{(11+14)}$	$\frac{(1+1)}{(9+14)}$
very	$\frac{(1+1)}{(11+14)}$	$\frac{(0+1)}{(9+14)}$
close	$\frac{(0+1)}{(11+14)}$	$\frac{(1+1)}{(9+14)}$
game	$\frac{(2+1)}{(11+14)}$	$\frac{(0+1)}{(9+14)}$

Let Z = A Very Close Game ; S = Sport ; NS = Non Sport

Therefore probability of "A Very Close Game" being a Sport => P(S|Z) = P(Z|S) P(S)

P(A|S) x P(V|S) x P(C|S) x P(G|S) x P(S) = 3/25 x 2/25 x 1/25 x 3/25 x 3/5 = 0.0004608 x 0.6 = 0.000027648

And the probability of "A Very Close Game" being a Non Sport => P(NS|Z) = P(Z|NS) P(NS)

P(A|NS) x P(V|NS) x P(C|NS) x P(G|NS) x P(NS) = 2/23 x 1/23 x 2/23 x 1/23 x 2/5 = 0.00001429 x 0.4=0.00000571

Then "a very close game" belongs to the Sports class

SMS spam detection

Label	Message snippet
Ham	Go until jurong point, crazy.. Available only …
Spam	Ok lar… Joking wif u oni...
Ham	U dun say so early hor… U c already then say...
Spam	FreeMsg Hey there darling it's been 3 week's n…

- 425 SMS spam messages from UK Grumbletext web forum
- 3,375 ham randomly chosen from NUS SMS corpus (students)
- 450 ham from somebody's PhD thesis
- 322 spam and 1,002 ham from SMS Spam Corpus

https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection

Use scikit-learn Pipeline to manage cross validation

Scikit-learn Pipelines provide a mechanism for fitting and predicting a sequence of components. This is good practice to avoid data leakage.

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline

# Pipeline for multinomial naive Bayes
mnb = Pipeline([('vect', CountVectorizer(lowercase=False)),
                ('tfidf', TfidfTransformer()),
                ('clf', MultinomialNB())
               ])
mnb.fit(X_train, y_train)
```

1. **Convert string to a bag-of-words token vector.**
2. **Transform vector counts using TFIDF weighting.**
3. **Train/predict using multinomial naïve Bayes.**

Text-driven forecasting

Text-driven forecasting

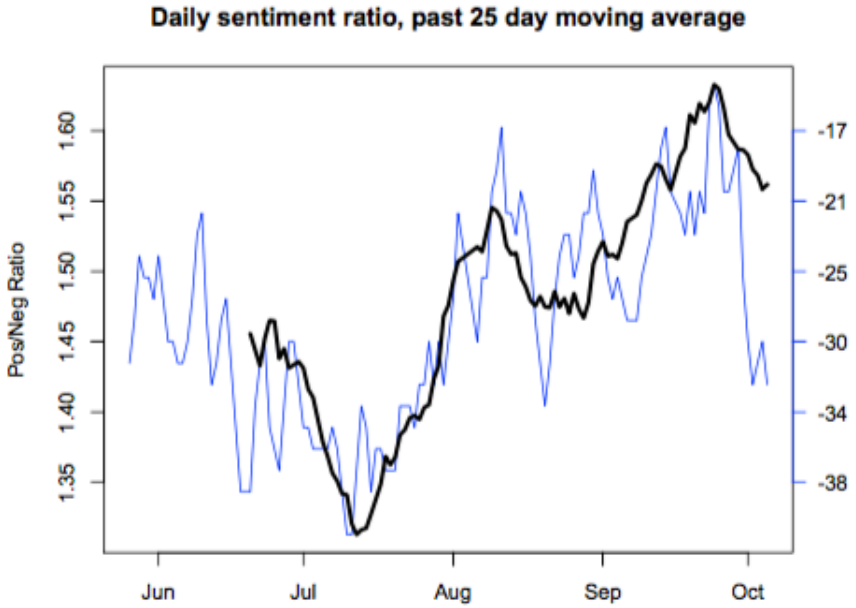
Given a body of text *T* pertinent to a social phenomenon, make a concrete prediction about a measurement *M* of that phenomenon, obtainable only in the future.

http://www.cs.cmu.edu/~nasmith/papers/smith.whitepaper10.pdf

Some text-driven forecasting tasks

- Predict box office gross for films
  - T: description, script, reviews, etc
  - M: how much the film earns at the box office
- Predict volatility of a stock
  - T: annual report, etc
  - M: volatility over the following year
- Predict blog reader behaviour
  - T: political blog posts, etc
  - M: number of reader comments

Predicting public opinion from tweets



- T: tweets mentioning the word "economy"
- M: Gallup's economic confidence index (blue)
- Predictions (black) closely track Gallup's polling data

https://www.aaai.org/ocs/index.php/ICWSM/ICWSM10/paper/viewFile/1536/1842

Review

W11 Review: Unstructured data

Objective

Learn machine learning tools in Python for text categorisation and forecasting.

Lecture

- Naïve Bayes
- Text-driven forecasting
- Structured prediction

Readings

- Data Science from Scratch, Ch. 13

Exercises

- Spam detection
- Predicting box office returns
- Information extraction

Text-driven forecasting (not examinable)

- CMU seminar on text-driven forecasting. http://www.cs.cmu.edu/~nasmith/TDF/
- Smith. Text-driven forecasting (whitepaper). https://www.aaai.org/ocs/index.php/ICWSM/ICWSM10/paper/viewFile/1536/1842
- Henry. Predicting with words (blog post). http://harmony-institute.org/latest/2012/10/19/forecasting-the-influence-of-entertainment/

Natural language processing (not examinable)

- Manning and Schutze. Foundations of statistical NLP.
  http://nlp.stanford.edu/fsnlp/
  Jurafsky and Martin. Speech and language processing.
  https://web.stanford.edu/~jurafsky/slp3/
- Bird et al. Natural language processing with Python.
  http://www.nltk.org/book/