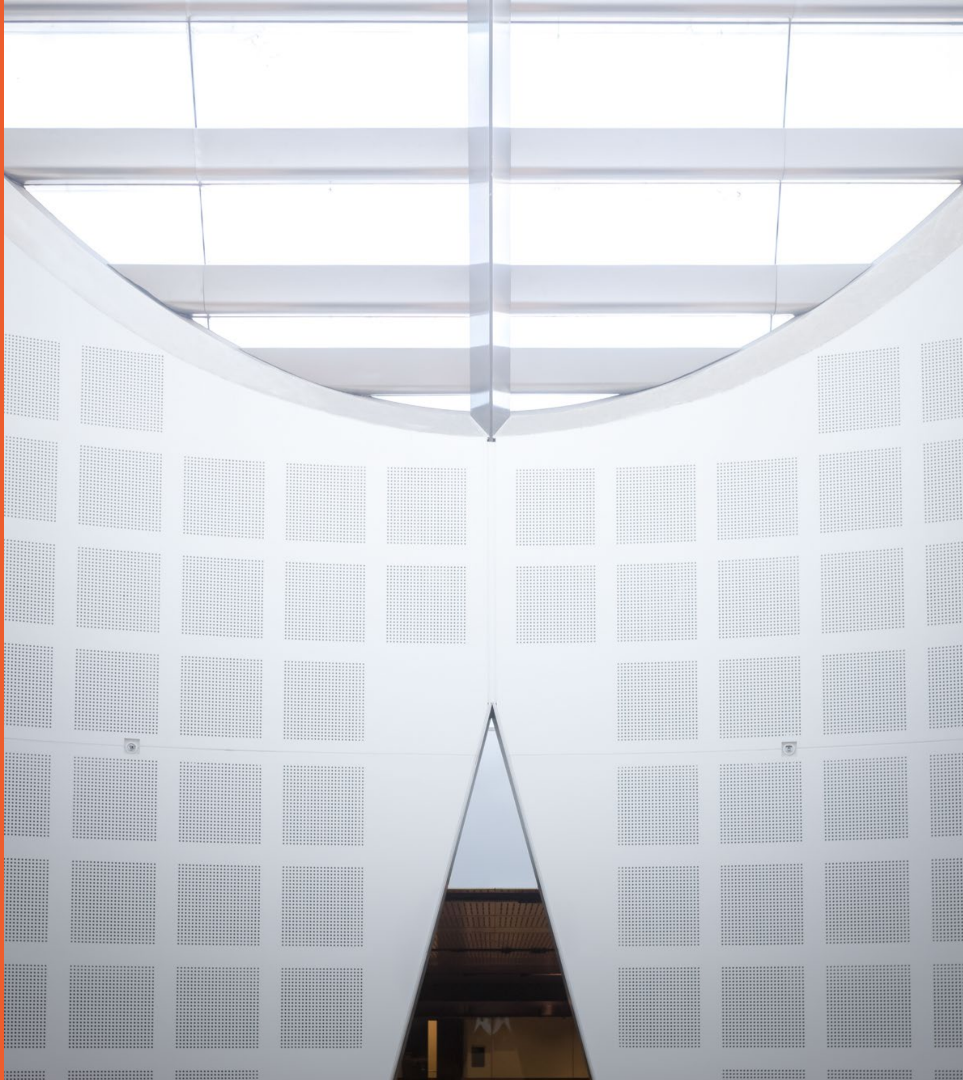
COMP5310: Principles of Data Science

W11: Unstructured Data—Naïve Bayes

Presented by
Ali Anaissi
School of Computer Science

THE UNIVERSITY OF SYDNEY



Today: Unstructured data

Objective

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

Lecture

– Naïve Bayes
– Text-driven forecasting
– Structured prediction

Readings

– Data Science from Scratch, Ch. 13
– Doing Data Science, Ch. 4

Exercises

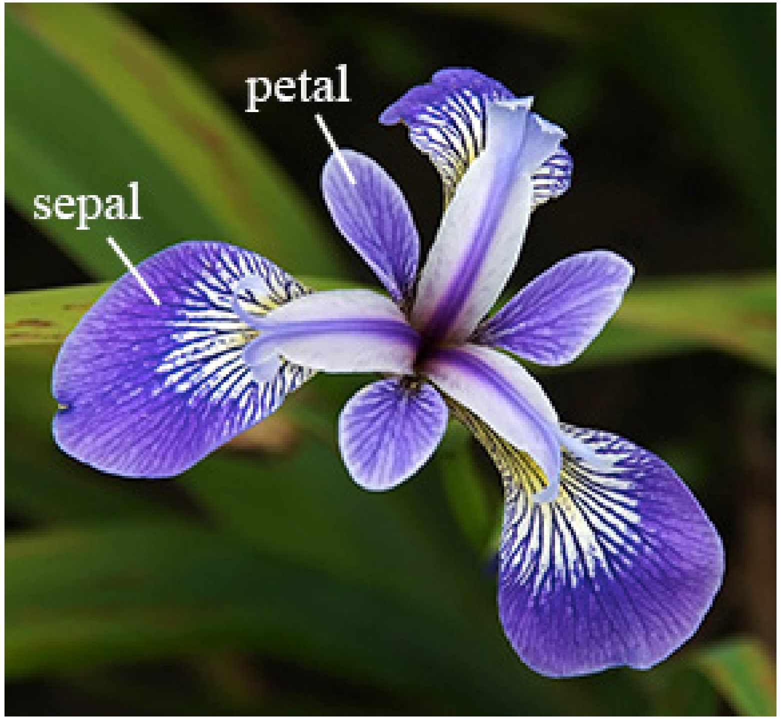
– Spam detection
– Predicting box office returns
– Information extraction

Unstructured data refers to information that either 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

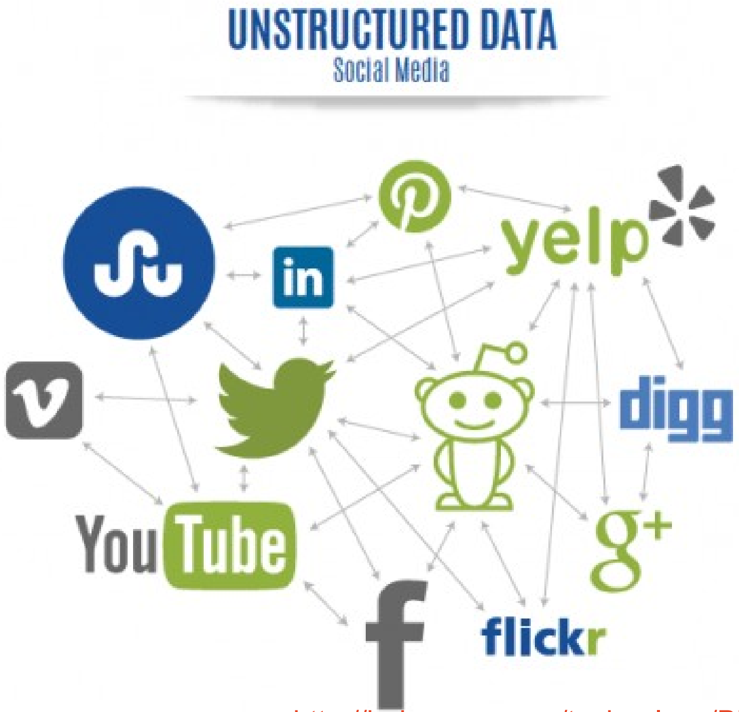
Unstructured data

"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)

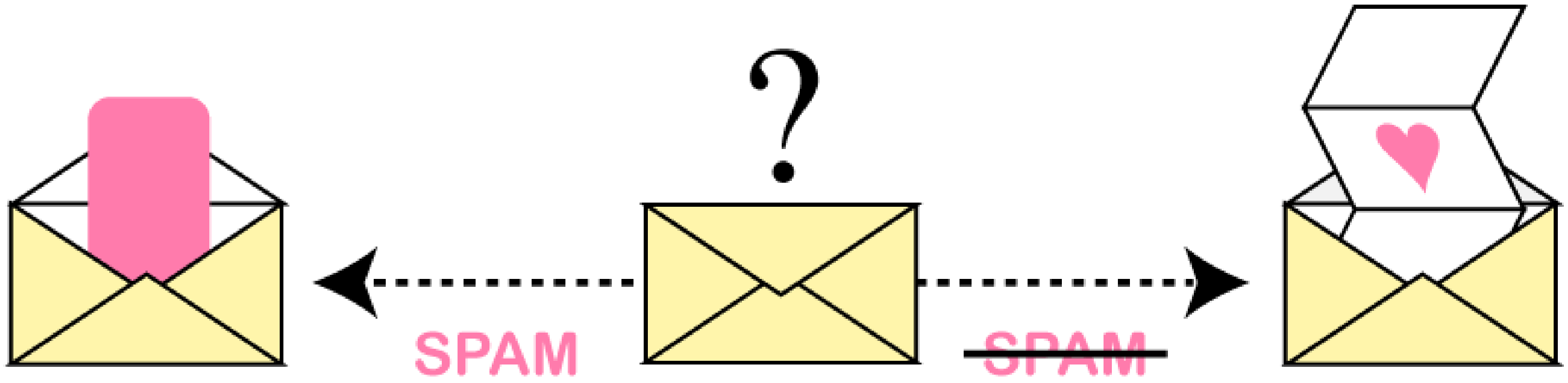
Social media data



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

Text Categorization

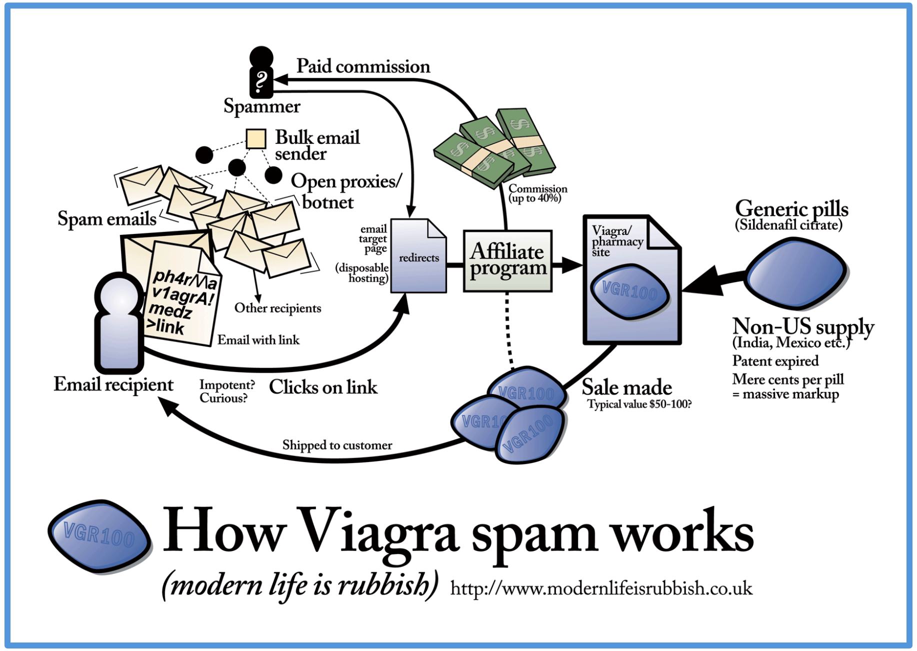
Task: Spam/ham detection



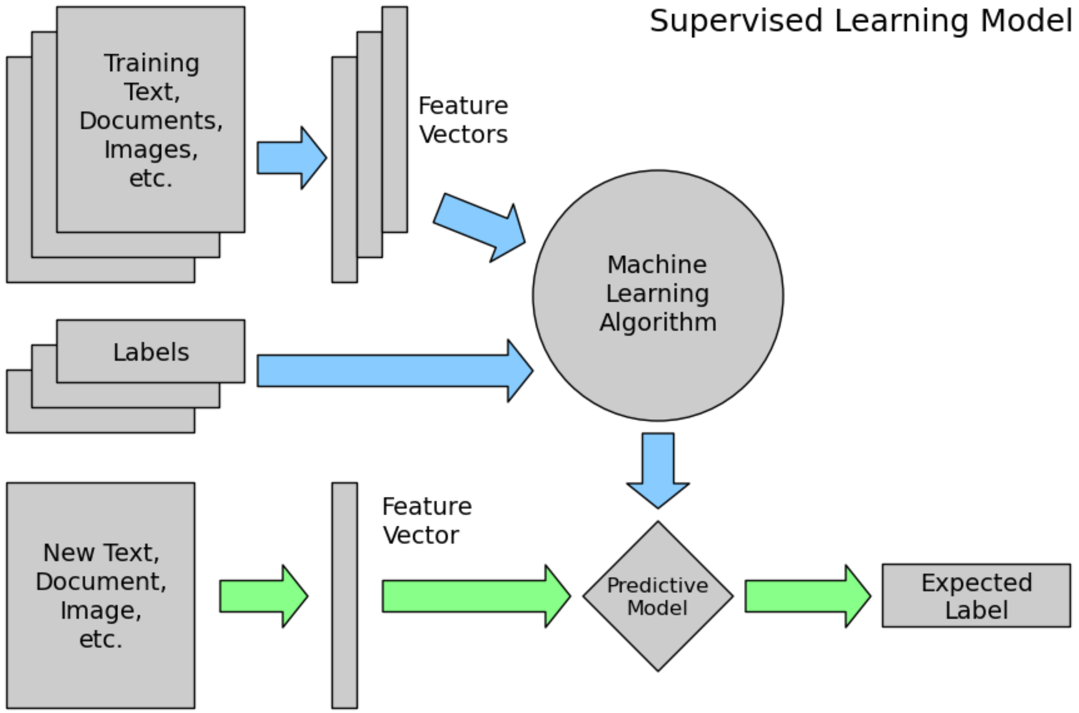
http://blog.dato.com/how-to-evaluate-machine-learning-models-part-2a-classification-metrics

Modelling spam detection

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



Spam detection as supervised classification


Supervised Learning Model
Training Text, Documents, Images, etc.
Feature Vectors
Labels
Machine Learning Algorithm
New Text, Document, Image, etc.
Feature Vector
Predictive Model
Expected Label

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

Feature vectors from text

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

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

Tokenisation

“Friends, Romans, Romans, countrymen”

$\boxed{\downarrow}$

[“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”]
```

$\boxed{\downarrow}$

```
{“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

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-D 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"

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

An Example

Class:

C1: Interviewed well = 'False'

C2: Interviewed well = 'True'

Data to be classified:

X = (Level =Senior,

Lang = Python,

Tweets = yes

PhD = No)

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

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

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

Compute $\mathbf{P(X|C_{True})}$

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 $\mathbf{P(X|C_{False})}$

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

$\mathbf{P(X|C_i)}$ **:**

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

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

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

Text Classification

THE UNIVERSITY OF SYDNEY

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:

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

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

- 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:

$$p(a\ very\ close\ game \,|\, Sports) \times p(Sports)$$

With

$$p(a\ very\ close\ game \,|\, Not\ sports) \times p(Not\ sports)$$

Text Classification

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

$$p(a\ very\ close\ game|\ Sports) = p(a|Sports) \times p(very|Sports) \times p(close|Sports) \times p(game|Sports)$$

Similarly

$$p(a\ very\ close\ game|\ Not\ Sports) = p(a|Not\ Sports) \times p(very|Not\ Sports) \times p(close|Not\ Sports) \times p(game|Not\ Sports)$$

- But the word “close” does not exist in the category Sports, thus $\boldsymbol{p(close|Sports) = 0}$, leading to $\boldsymbol{p(a\ very\ close\ game|\ Sports) = 0}$

- Thus loosing the probabilities information of other words.

Laplace smoothing

– $p(w|c) = \frac{\text{count(w,c)+1}}{\text{count(word,c)+count(word)}}$

count(w,c) = Count of word w in class c

count(word,c) = Count of words in class c.

count(word) = Count all the possible words in the dataset

– **Now we can calculate the probability again :**

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

Calculation

$p(\text{Sports}|\text{a very close game}) = ?$
$p(\text{Not Sports}|\text{a very close game}) = ?$

w	$p(w\|Sports)$	$p(w\|Not\ Sports)$
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/x25] 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” is belong to the Sports class

SMS spam detection

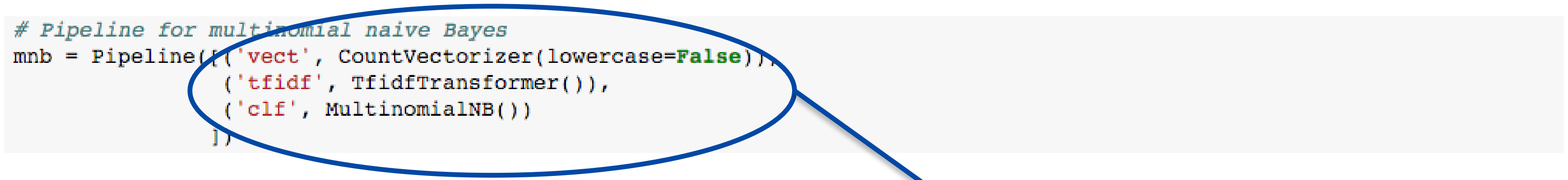
Label	Message snippet
Ham	Go until jurong point, crazy.. Available only …
Spam	Ok lar… Joking wif u oni...
Ham	U dun say so early hor… U c already then say...
Spam	FreeMsg Hey there darling it’s been 3 week’s n…

- 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.



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

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

Exercise: SMS spam filtering

- Build a spam filter
  - ⏯ code cell under “Download data from UCI ML data repo”
  - ⏯ code cell under “Read and profile data using Pandas”
  - ⏯ code cell under “Text feature extraction”
  - ⏯ code cell under “Build pipelines and choose parameters”
- Exercises
  - Which is better: support vector or multinomial naïve Bayes classifier?
  - Handling class imbalance in support vector classifier
  - Generalisation, data size, features

Text-driven forecasting

THE UNIVERSITY OF
SYDNEY

Text-driven forecasting

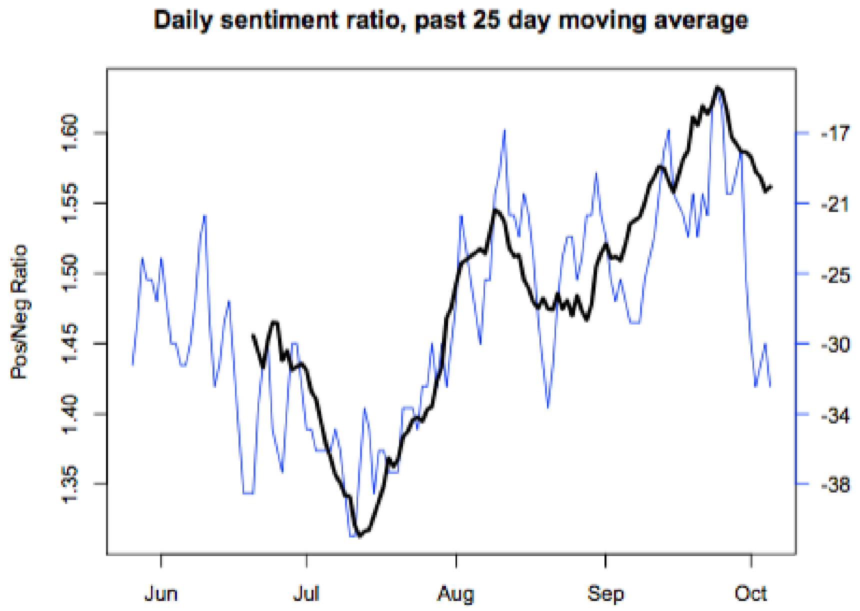
Given a body of text *T* pertinent to a social phenomenon, make a concrete prediction about a measurement *M* of that phenomenon, obtainable only in the future.

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



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

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

Exercise: Predicting box office gross

- Build model to forecast gross
  - ⏯ code cell under “Download reviews and movie gross data”
  - ⏯ code cell under “Select parameters for support vector regression”
- Exercises
  - Which parameters are best?
  - Is the model a good fit?
  - How could we improve the experimental setup?

Review

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
- Doing Data Science, Ch. 4

Exercises

- Spam detection
- Predicting box office returns
- Information extraction

TODO in W11

- Analyse and characterise results

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/

Thank you

THE UNIVERSITY OF
SYDNEY