DS2500

4/8-Tues

Admin

- second chance (4/11 9pm)
- project deadline (4/15 9pm)
  - ↳ group: report, abstract
    indv: reflection

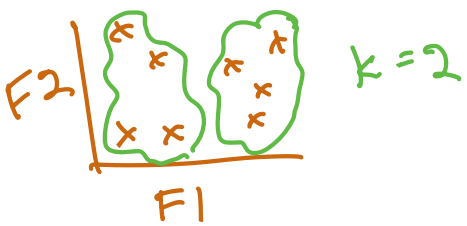
- XC sign-ups 4/18, 4/23, 4/24 8-10AM

Agenda

1. Clustering w/text
2. Principal Component Analysis (PCA)
3. Python

} Colab, sklearn, aliens.json

1. K-means Clustering, w/text



last time: Shiny 0-10
kittens 0-10

today: catch phrase

goal: group similar aliens based on catchphrase

still need: euclidean distance
represent aliens w/ numbers

text → numeric

① sentiment analysis
text → -1 to +1
same score if vibes are similar, but words can differ

② TF-IDF
similar score if words are similar
content based

Clustering text:
- reviews, descriptions, restaurants
  chatgpt responses, etc.

↳ part of Natural Language Processing (NLP)

- document — one instance of text — (ex) catchphrase
- corpus — all the documents

- bag of words — order doesn't matter
- TF — term frequency → $\frac{\#\text{ occurrences of a word}}{\#\text{ words in the document}}$
- IDF — inverse document frequency → $\log\left(\frac{\#\text{ documents in corpus}}{\#\text{ documents w/ the word}}\right)$

TF.IDF = TF * IDF

↳ one value per (alien, word in corpus)


```
{
  "results": [
    {
      "name": "stitch",
      "slimy": 1,
      "kill humans": 2,
      "catchphrase": "ohana means family family means no one is left behind or forgotten"
    },
    {
      "name": "E.T.",
      "slimy": 2,
      "kill humans": 1,
      "catchphrase": "E.T. phone home!"
    },
    {
      "name": "Harvester",
      "slimy": 9,
      "kill humans": 10,
      "catchphrase": "I would like to conquer your planet"
    },
    {
      "name": "kang",
      "slimy": 10,
      "kill humans": 8,
      "catchphrase": "Go ahead! Throw your vote away!"
    },
    {
      "name": "kodos",
      "slimy": 10,
      "kill humans": 7,
      "catchphrase": "Eat, Simpsons! Grow large with food!"
    },
    {
      "name": "Audrey II",
      "slimy": 5,
      "kill humans": 9,
      "catchphrase": "I'm just a mean green mother from outer space and I'm mad"
    },
    {
      "name": "Marvin",
      "slimy": 3,
      "kill humans": 6,
      "catchphrase": "Where's the kaboom? There was supposed to be an Earth-shattering kaboom!"
    },
```


→ today's dataset

cat? no

Simpsons? no

grow? no

large? no

ex) alien = stitch

catchphrase = ... family ...

times stitch says "family" = 3

words in stitch's phrase = 100

docs in corpus = 1000

docs w/ "family" = 10

goal: TF.IDF(stitch, family)

$TF = \frac{3}{100} = .03$

$IDF = \log\left(\frac{1000}{10}\right) = \log(100) = 2$

$TF.IDF(\text{stitch, fam}) = .03 \cdot 2 = .06$

In python: TfidfVectorizer()

fit_transform()

2. PCA

PCA helps us w/ this:

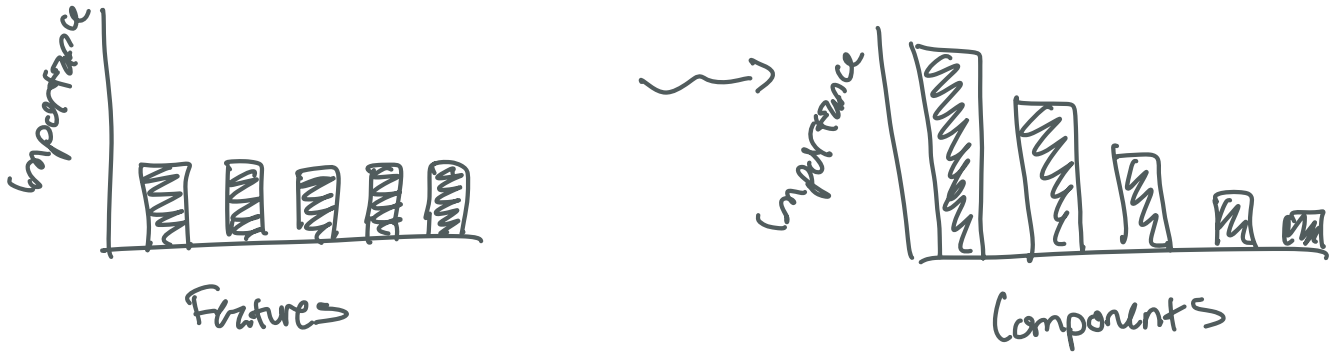
	w1	w2	w3	...	w200
stitch	~	~	~		~
robocop	~	~	~		~
Avatar	~	~	~		~
⋮					

- TF-IDF on every alien-word pair
- lots of cols/features
  - ↳ high dimensionality
- lots of 0s
  - ↳ sparse

PCA ⇝ reduce dimensionality!

- squish features into components
- lose some info, but not too much
- keep most important components

5 features → 5 components



- value in feature is TF-IDF
- keep top 2-3 components
- values have no units

Python: `PCA()` `fit_transform()`