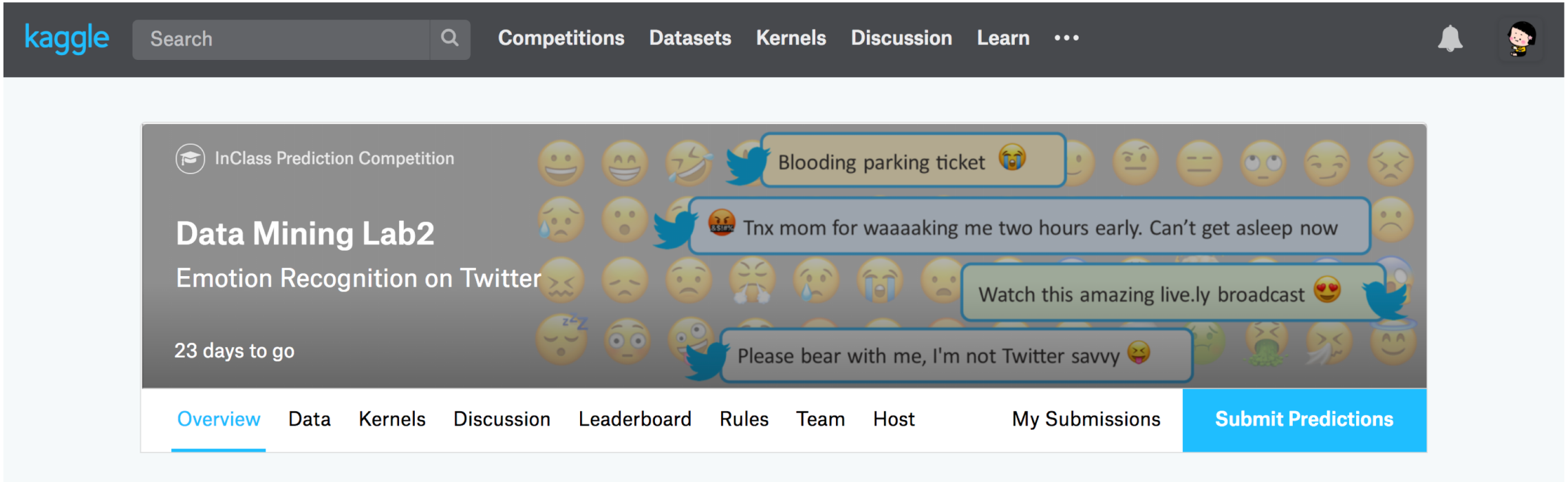
Data Mining Lab 2

Fall 2018

NTHU, ISA

Ssu-rui Lee, Ya-wen Yu

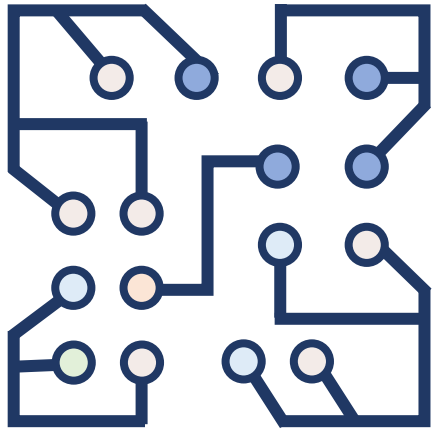
Assignment 2



Task

Model

Text



Emotion

why are people so offended by kendall he ends ...

I'm about to block everyone everywhere posting...

Making my buddy cry !! Bitch wait for revenge 🤗 👌🏻

@KennyCoble @Rosie these horrific situations w...

@BBCNews 😝 scared of their own horror story th...

my husband lost £800 when he booked an apartme...

Food that gets delivered 😍 🙌🏻

Raw data

Raw data

training

joy-ratings-0to1.dev.gold.txt | anger-ratings-0to1.dev.gold.txt | fear-ratings-0to1.dev.gold.txt | sadness-ratings-0to1.dev.gold.txt

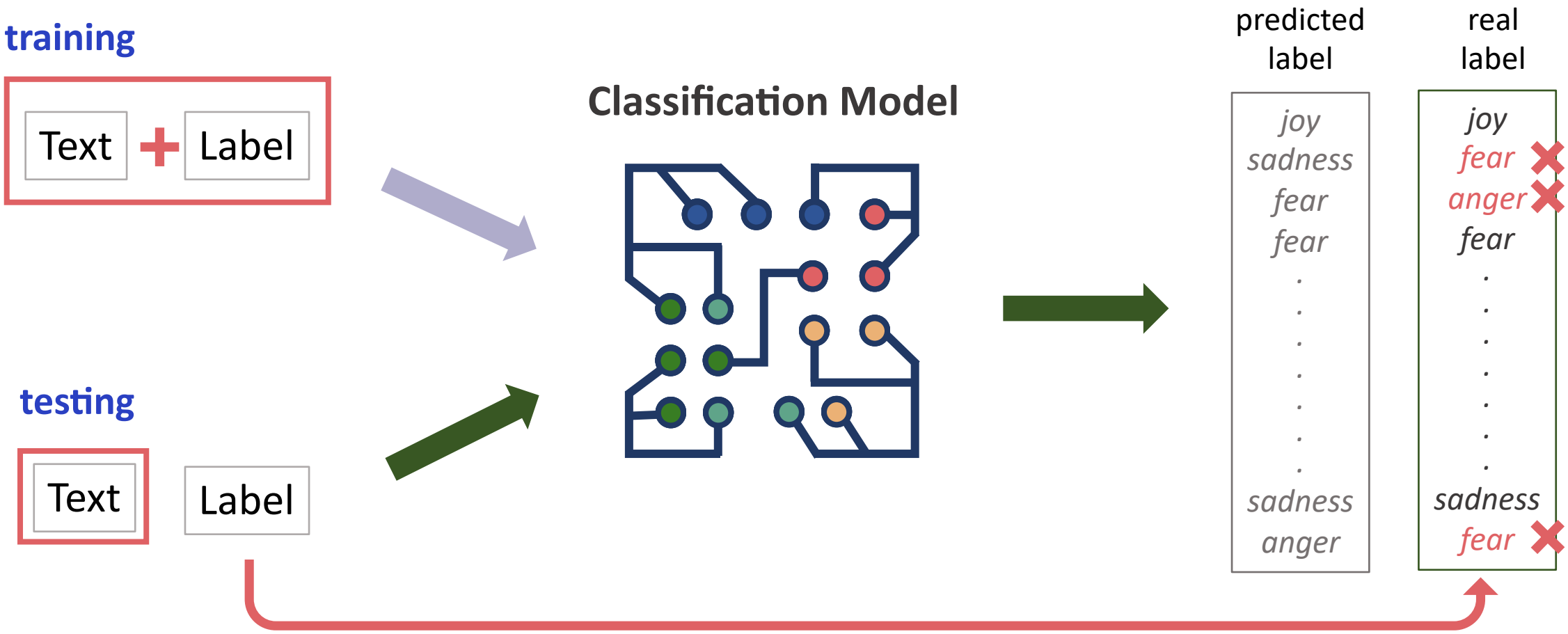
```
30823   @theclobra lol I thought maybe, couldn't decide if there was levity or not  joy 0.312
30824   Nawaz Sharif is getting more funnier than @kapilsharmak9 day by day. #laughter #challenge #kashmir #baloch  joy 0.700
30825   Nawaz Sharif is getting more funnier than @kapilsharmak9 day by day.  #challenge #kashmir #baloch   joy 0.580
30826   @tomderivan73 😁...I'll just people watch and enjoy a rare show of optimism joy 0.438
30827   I love my family so much #lucky #grateful #smartassfamily  #love    joy 0.936
30828   I love my family so much #lucky #grateful #smartassfamily #hilarious #love  joy 0.792
30829   @Casper10666 I assure you there is no laughter, but increasing anger at the costs, and arrogance of Westminster.    joy 0.167
30830   If any trump supporters and Hillary haters wanna chirp some weak minded, pandering liberals just tweet at @EmmyA2 @snickerfritz04   joy 0.100
30831   Google caffeine-an sprightly lengthening into the corridor re seo: WgJ  joy 0.200
30832   This tweet is dedicated to my back pain, which I do not understand because I am youthful and spry. Full of life. Vivacious. joy 0.229
30833   @Bluebelle89 @lsmith855 liking the optimism joy 0.540
30834   Be it a rainy day, be it cheerful sunshine, I am a Prussian, want nothing to be but a Prussian.:|   joy 0.245
30835   @Gronnhair @buryprofs @DittoBistro it was indeed lovely and the team were incredibly attentive and on the ball. Cheese was a lively gesture!    joy 0.646
30836   @Geminiak @LondonNPC you're welcome! #wordgeek \nAlso, good to put a face to the twitter feed, even if it was only a cheery hello! 😊   joy 0.646
30837   ...at your age, the heyday in the blood is tame...' @TheArtofCharm #shakespeareaninsults #hamlet #elizabethan #williamshakespeare   joy 0.260
30838   i was so embarrassed when she saw us i was like knvfkkjg she thinks we're stalkers n then she starts waving all cheerfully inviting us in 😩    joy 0.250
30839   Good #CX most often doesn't require all that much. #smile, #care, #relate and be #helpful. Thanks, Lakis Court Hotel in #cyprus joy 0.583
30840   4-2 Canada final tomorrow #WCH #Predictions #optimism #Canadian 🇨🇦  joy 0.420
30841   I turn 25 in two weeks. I am so happy. 24 was my darkest year yet. I am elated that I survived  joy 0.708
30842   TheNiceBot: IndyMN I thought the holidays could not get any more cheerful, and then I met you. #TheNiceBot #ايچفغل joy 0.769
30843   The smell of freshly cut grass didn't even cheer me up...boy oh boy joy 0.229
```

testing

joy-ratings-0to1.train.txt | anger-ratings-0to1.train.txt | fear-ratings-0to1.train.txt | sadness-ratings-0to1.train.txt

```
30000   Just got back from seeing @GaryDelaney in Burslem. AMAZING!! Face still hurts from laughing so much #hilarious  joy 0.980
30001   Oh dear an evening of absolute hilarity I don't think I have laughed so much in a long time! 😂 joy 0.958
30002   Been waiting all week for this game ❤️❤️❤️ #cheer #friday ❤️   joy 0.940
30003   @gardiner_love : Thank you so much, Gloria! You're so sweet, and thoughtful! You just made my day more joyful! I love you too! 😊💕joy 0.938
30004   I feel so blessed to work with the family that I nanny for ❤️ nothing but love &amp;amp; appreciation, makes me smile.  joy 0.938
30005   Today I reached 1000 subscribers on YT!! , #goodday, #thankful  joy 0.926
30006   @Singaholic121 Good morning, love! Happy first day of fall. Let's make some awesome #autumnmemories #annabailey #laughter #smile    joy 0.924
30007   #BridgetJonesBaby is the best thing I've seen in ages! So funny, I've missed Bridget! #love  #TeamMark  joy 0.922
30008   Just got back from seeing @GaryDelaney in Burslem. AMAZING!! Face still hurts from laughing so much     joy 0.920
30009   @IndyMN I thought the holidays could not get any more cheerful, and then I met you. #TheNiceBot joy 0.917
30010   I'm just still . So happy .\nA blast    joy 0.917
30011   It's meant to be!! #happy #happy    joy 0.917
30012   💥⚖️Yeah‼️ PAUL‼️⚖️💥  #glorious #BB18    joy 0.917
30013   My morning started off amazing!! Hopefully the whole day is going as i want it to go!\n #GreatDay   joy 0.917
30014   😱 @cailamarsai you've had me 😂 😂 the whole time watching @black_ishABC after you've lost your #glasses! It was #hilarious! @mrbabyboogaloo  joy 0.900
30015   @iamTinaDatta love you so much #smile 😊😊 joy 0.896
30016   @WyoWiseGuy @LivingVertical however, REI did offer me the job today as well. Can't believe how exponentially freaking joyous I feel...!!!   joy 0.896
30017   2 days until #GoPackGo and 23 days until #GoGipeGo..... I'm so excited! #smiling    joy 0.880
30018   @TheMandyMoore You are beyond wonderful.  Your singing prowess is phenomenal but damn... I'm just elated to watch you act again. #ThisIsUs 珞   joy 0.879
30019   @luckiiCHARM_ Luckii, I'm changing in so many ways bc of Him!! It's a scary but joyful feeling, making me so strong.    joy 0.877
```

Data – training / testing



Load and prepare data

Load training data

```
import pandas as pd
```

```
### training data
anger_train = pd.read_csv("data/semeval/train/anger-ratings-0to1.train.txt",
                          sep="\t", header=None,names=["id", "text", "emotion", "intensity"])
sadness_train = pd.read_csv("data/semeval/train/sadness-ratings-0to1.train.txt",
                          sep="\t", header=None, names=["id", "text", "emotion", "intensity"])
fear_train = pd.read_csv("data/semeval/train/fear-ratings-0to1.train.txt",
                          sep="\t", header=None, names=["id", "text", "emotion", "intensity"])
joy_train = pd.read_csv("data/semeval/train/joy-ratings-0to1.train.txt",
                          sep="\t", header=None, names=["id", "text", "emotion", "intensity"])
```

	id	text	emotion	intensity
0	10000	How the fu*k! Who the heck! moved my fridge!.....	anger	0.938

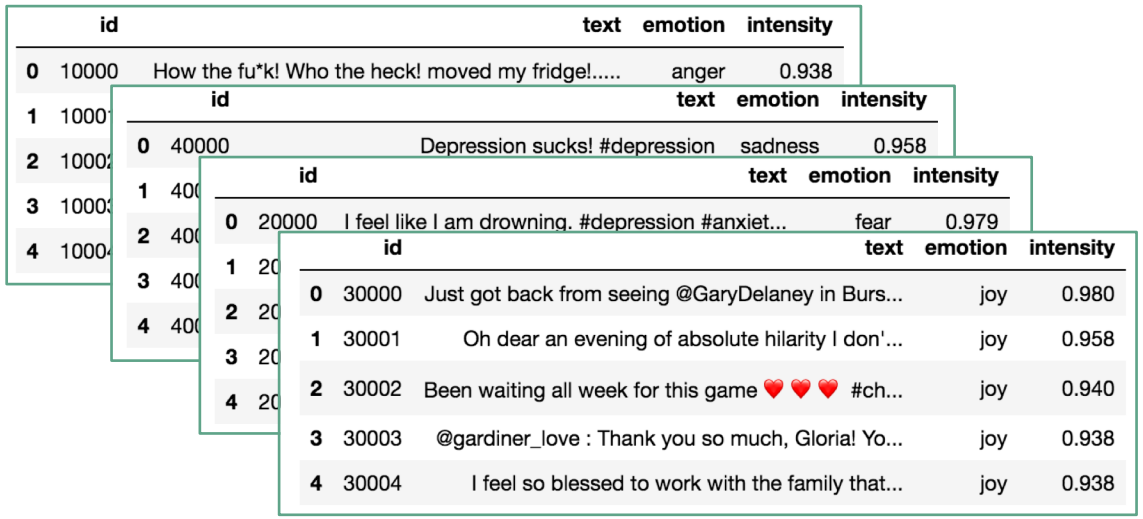
	id	text	emotion	intensity
0	40000	Depression sucks! #depression	sadness	0.958

	id	text	emotion	intensity
0	20000	I feel like I am drowning. #depression #anxiet...	fear	0.979

	id	text	emotion	intensity
0	30000	Just got back from seeing @GaryDelaney in Burs...	joy	0.980
1	30001	Oh dear an evening of absolute hilarity I don'...	joy	0.958
2	30002	Been waiting all week for this game ❤️ ❤️ ❤️ #ch...	joy	0.940
3	30003	@gardiner_love : Thank you so much, Gloria! Yo...	joy	0.938
4	30004	I feel so blessed to work with the family that...	joy	0.938

Prepare data

```
# combine 4 sub-dataset
train_df = pd.concat([anger_train, fear_train, joy_train, sadness_train], ignore_index=True)
```



	id	text	emotion	intensity
0	10000	How the fu*k! Who the heck! moved my fridge!.....	anger	0.938
1	40000	Depression sucks! #depression	sadness	0.958
2	20000	I feel like I am drowning. #depression #anxiet...	fear	0.979
3	30000	Just got back from seeing @GaryDelaney in Burs...	joy	0.980
4	30001	Oh dear an evening of absolute hilarity I don'...	joy	0.958
5	30002	Been waiting all week for this game ❤️❤️❤️ #ch...	joy	0.940
6	30003	@gardiner_love : Thank you so much, Gloria! Yo...	joy	0.938
7	30004	I feel so blessed to work with the family that...	joy	0.938

	id	text	emotion	intensity
0	10000	How the fu*k! Who the heck! moved my fridge!.....	anger	0.938
1	10001	So my Indian Uber driver just called someone t...	anger	0.896
2	10002	@DPD_UK I asked for my parcel to be delivered ...	anger	0.896
3	10003	so ef whichever butt wipe pulled the fire alar...	anger	0.896
4	10004	Don't join @BTCare they put the phone down on ...	anger	0.896
5	10005	My blood is boiling	anger	0.875
6	10006	When you've still got a whole season of Wentwo...	anger	0.875
7	10007	@bt_uk why does tracking show my equipment del...	anger	0.875
8	10008	@TeamShanny legit why i am so furious with him...	anger	0.875
9	10009	How is it suppose to work if you do that? Wtf ...	anger	0.875
10	10010	im so mad about power rangers. im incensed. im...	anger	0.667

Load & prepare testing data

```
### test data
anger_test = pd.read_csv("data/semeval/dev/anger-ratings-0to1.dev.gold.txt",
                         sep="\t", header=None, names=["id", "text", "emotion", "intensity"])
sadness_test = pd.read_csv("data/semeval/dev/sadness-ratings-0to1.dev.gold.txt",
                         sep="\t", header=None, names=["id", "text", "emotion", "intensity"])
fear_test = pd.read_csv("data/semeval/dev/fear-ratings-0to1.dev.gold.txt",
                         sep="\t", header=None, names=["id", "text", "emotion", "intensity"])
joy_test = pd.read_csv("data/semeval/dev/joy-ratings-0to1.dev.gold.txt",
                         sep="\t", header=None, names=["id", "text", "emotion", "intensity"])

# combine 4 sub-dataset
test_df = pd.concat([anger_test, fear_test, joy_test, sadness_test], ignore_index=True)
```

Data preparation

```
print('shape of "training data":', train_df.shape)

train_df
```

```
shape of "training data": (3613, 4)
```

	id	text	emotion	intensity
0	10000	How the fu*k! Who the heck! moved my fridge!.....	anger	0.938
1	10001	So my Indian Uber driver just called someone t...	anger	0.896
2	10002	@DPD_UK I asked for my parcel to be delivered ...	anger	0.896
3	10003	so ef whichever butt wipe pulled the fire alar...	anger	0.896
...	...	...	...	...
3609	40782	Just put the winter duvet on ☃️❄️🌬️☔	sadness	0.104
3610	40783	@SilkInSide @TommyJoeRatliff that's so pretty!...	sadness	0.088
3611	40784	@BluesfestByron second artist announcement loo...	sadness	0.083
3612	40785	I can literally eat creamy pesto pasta topped ...	sadness	0.083

Data preparation

```
print('shape of "training data":', train_df.shape)

train_df
```

```
shape of "training data": (3613, 4)
```

	id	text	emotion	intensity
0	10000	How the fu*k! Who the heck! moved my fridge!.....	anger	0.938
1	10001	So my Indian Uber driver just called someone t...	anger	0.896
2	10002	@DPD_UK I asked for my parcel to be delivered ...	anger	0.896
3	10003	so ef whichever butt wipe pulled the fire alar...	anger	0.896
...	...	...	...	...
3609	40782	Just put the winter duvet on ☃️ ❄️ 🌬️ ☔	sadness	0.104
3610	40783	@SilkInSide @TommyJoeRatliff that's so pretty!...	sadness	0.088
3611	40784	@BluesfestByron second artist announcement loo...	sadness	0.083
3612	40785	I can literally eat creamy pesto pasta topped ...	sadness	0.083

rows / records

Data preparation

```
print('shape of "training data":', train_df.shape)

train_df
```

```
shape of "training data": (3613, 4)
```

	id	text	emotion	intensity
0	10000	How the fu*k! Who the heck! moved my fridge!.....	anger	0.938
1	10001	So my Indian Uber driver just called someone t...	anger	0.896
2	10002	@DPD_UK I asked for my parcel to be delivered ...	anger	0.896
3	10003	so ef whichever butt wipe pulled the fire alar...	anger	0.896
...	...	...	...	...
3609	40782	Just put the winter duvet on ☃️❄️🌬️☔	sadness	0.104
3610	40783	@SilkInSide @TommyJoeRatliff that's so pretty!...	sadness	0.088
3611	40784	@BluesfestByron second artist announcement loo...	sadness	0.083
3612	40785	I can literally eat creamy pesto pasta topped ...	sadness	0.083

columns / attributes

Data preparation

```
# shuffle dataset
train_df = train_df.sample(frac=1)
```

100% of data

id	text	emotion	intensity
10000	How the fu*k! Who the heck! moved my fridge!.....	anger	0.938
10001	So my Indian Uber driver just called someone t...	anger	0.896
10002	@DPD_UK I asked for my parcel to be delivered ...	anger	0.896
10003	so ef whichever butt wipe pulled the fire alar...	anger	0.896
...	...	...	...
40782	Just put the winter duvet on ☃️ ❄️ 🌬️ ☔	sadness	0.104
40783	@SilkInSide @TommyJoeRatliff that's so pretty!...	sadness	0.088
40784	@BluesfestByron second artist announcement loo...	sadness	0.083
40785	I can literally eat creamy pesto pasta topped ...	sadness	0.083

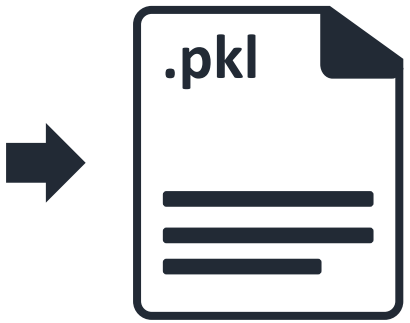
id	text	emotion	intensity
10018	why are people so offended by kendall he ends ...	anger	0.833
10019	I'm about to block everyone everywhere posting...	anger	0.812
10020	Making my buddy cry !! Bitch wait for revenge 🤗 👌	anger	0.812
20171	@KennyCoble @Rosie these horrific situations w...	fear	0.708
20172	@BBCNews 😜 scared of their own horror story th...	fear	0.708
20173	my husband lost £800 when he booked an apartme...	fear	0.708
30123	Food that gets delivered 😍 🙌🏻	joy	0.729
30124	@ArtyBagger With or without cake seeing your w...	joy	0.729
30125	I am a simple human being who just really love...	joy	0.729
40131	@HannahFJames I'm distraught! 😭 Candice and he...	sadness	0.688
40132	@HutchinsonDave I don't know whether to despai...	sadness	0.688
40133	Can I just sulk in peace 😂	sadness	0.688

Save data

pickle file format 

The pickle module implements binary protocols for serializing and de-serializing a Python object structure.

id	text	emotion	intensity
10018	why are people so offended by kendall he ends ...	anger	0.833
10019	I'm about to block everyone everywhere posting...	anger	0.812
10020	Making my buddy cry !! Bitch wait for revenge 🤗 👌🏻	anger	0.812
20171	@KennyCoble @Rosie these horrific situations w...	fear	0.708
20172	@BBCNews 😝 scared of their own horror story th...	fear	0.708
20173	my husband lost £800 when he booked an apartme...	fear	0.708
30123	Food that gets delivered 😍🙌🏻	joy	0.729
30124	@ArtyBagger With or without cake seeing your w...	joy	0.729
30125	I am a simple human being who just really love...	joy	0.729



train_df.pkl

```
8004 9501 0001 0000 0000 008c 1170 616e
6461 732e 636f 7265 2e66 7261 6d65 948c
0944 6174 6146 7261 6d65 9493 9429 8194
7d94 288c 055f 6461 7461 948c 1570 616e
6461 732e 636f 7265 2e69 6e74 6572 6e61
6c73 948c 0c42 6c6f 636b 4d61 6e61 6765
7294 9394 2981 9428 5d94 288c 1870 616e
6461 732e 636f 7265 2e69 6e64 6578 6573
2e62 6173 6594 8c0a 5f6e 6577 5f49 6e64
6578 9493 9468 0b8c 0549 6e64 6578 9493
947d 9428 8c04 6461 7461 948c 156e 756d
7079 2e63 6f72 652e 6d75 6c74 6961 7272
6179 948c 0c5f 7265 636f 6e73 7472 7563
7494 9394 8c05 6e75 6d70 7994 8c07 6e64
6172 7261 7994 9394 4b00 8594 4301 6294
```

pickle file format 

The pickle module implements binary protocols for serializing and de-serializing a Python object structure.

- convenient for cross-platform use (it saves attribute type)
- Save space and loading time

127 million records		
File format:	.csv	.pkl
Size:	18G	7G
Loading time:	600 sec	100 sec

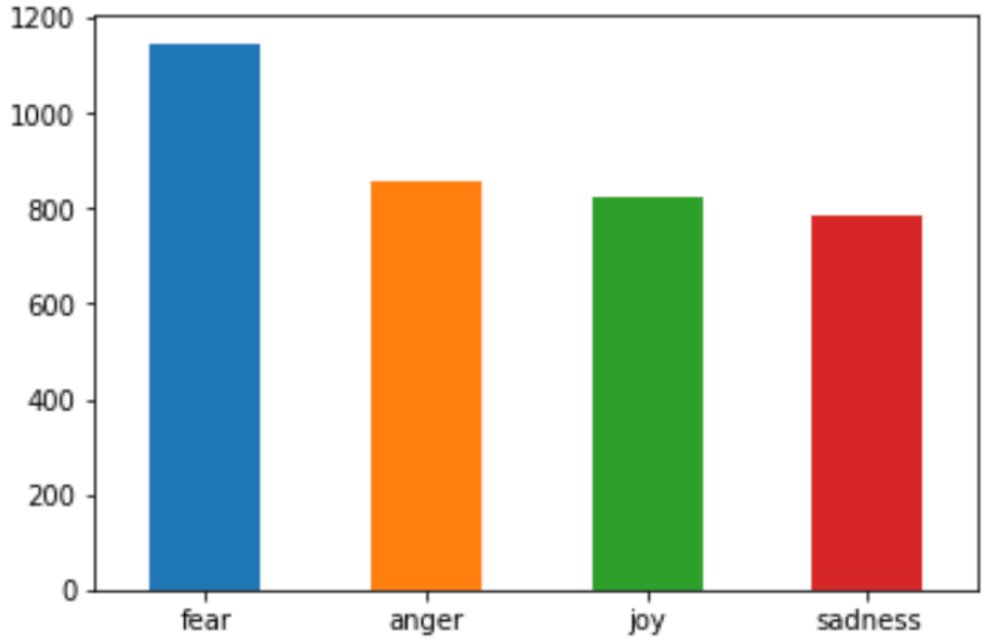
Explore data analysis

Explore data analysis

```
%matplotlib inline

train_df['emotion'].value_counts().plot(kind="bar", rot=0)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x112705908>
```



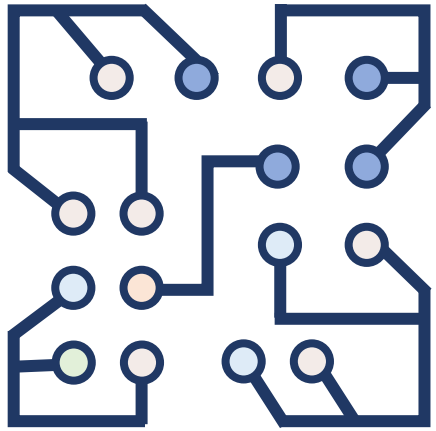
Think & Try

Is this considered an **imbalanced** dataset?

Try to perform some more data exploration and statistical inference as done in lab session 1.

Model

Text

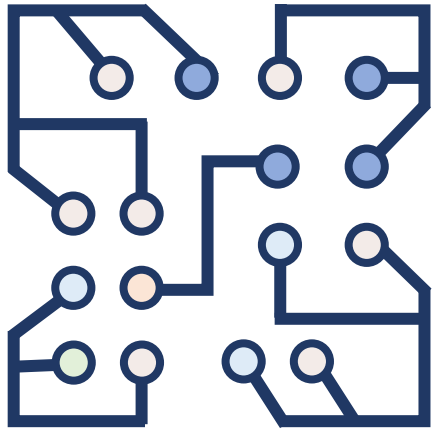


Emotion

why are people so offended by kendall he ends ...
I'm about to block everyone everywhere posting...
Making my buddy cry !! Bitch wait for revenge 🤗 👌🏻
@KennyCoble @Rosie these horrific situations w...
@BBCNews 😜 scared of their own horror story th...
my husband lost £800 when he booked an apartme...
Food that gets delivered 😍🙌🏻
@ArtyBagger With or without cake seeing your w...

Numerical only

Model

Text →  → Emotion

why are people so offended by kendall he ends ...

I'm about to block everyone everywhere posting...

Making my buddy cry !! Bitch wait for revenge 🤗 👌🏻

@KennyCoble @Rosie these horrific situations w...

@BBCNews 😜 scared of their own horror story th...

my husband lost £800 when he booked an apartme...

Food that gets delivered 😍🙌🏻

@ArtyBagger With or without cake seeing your w...

Bag of words →

a	an	buddy	dog	for	from	.....	jump	lazy	men	now	of	over	party
1	1	0	1	0	1	.....	1	1	0	0	0	1	0
2	0	0	1	0	0	.....	0	2	1	1	0	1	2
0	0	0	0	0	0	.....	1	0	0	1	1	1	1

Feature engineering

(Use bag of words)

Feature engineering – Bag of words

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.

Step 1

Feature engineering – Bag of words

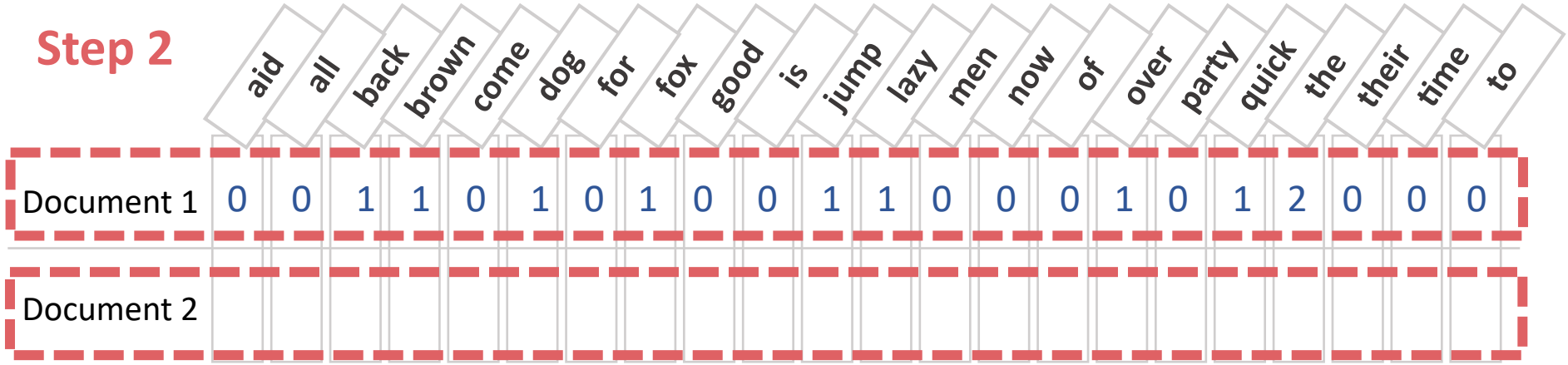
Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.

Step 2



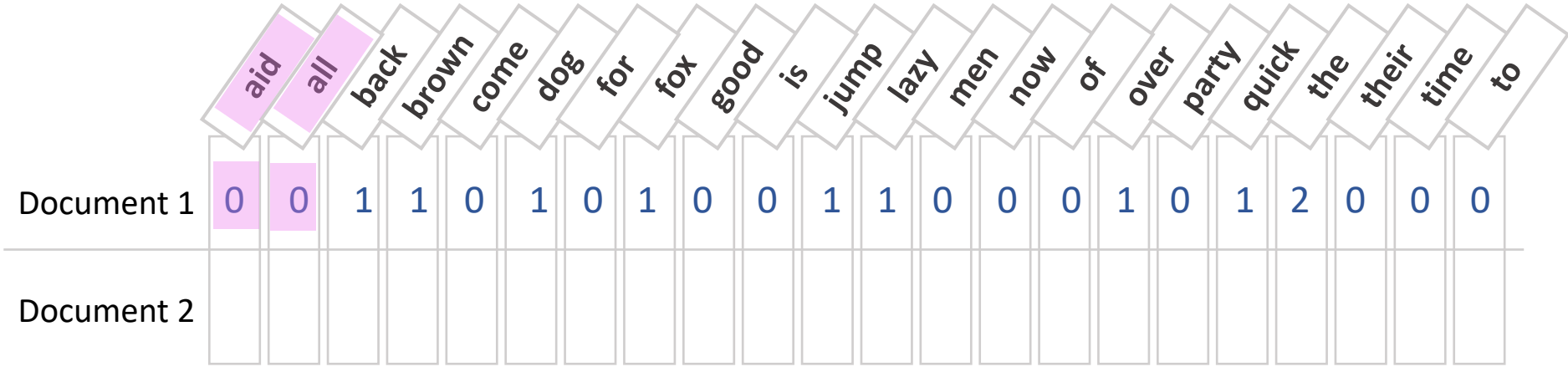
Feature engineering – Bag of words

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.



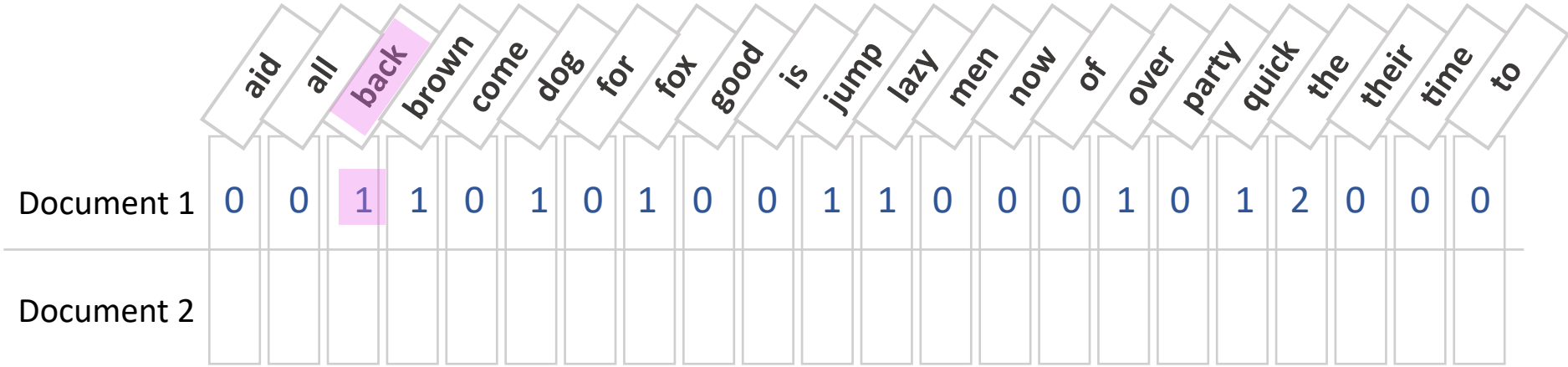
Feature engineering – Bag of words

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.



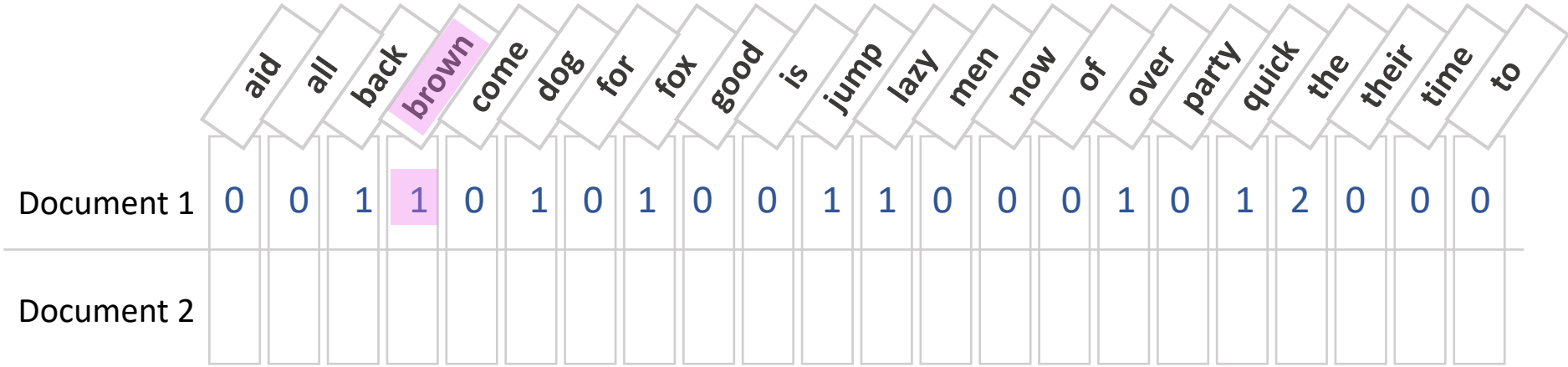
Feature engineering – Bag of words

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.



Feature engineering – Bag of words

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.

Sum up → 10

It should be the same as the word count in the document.

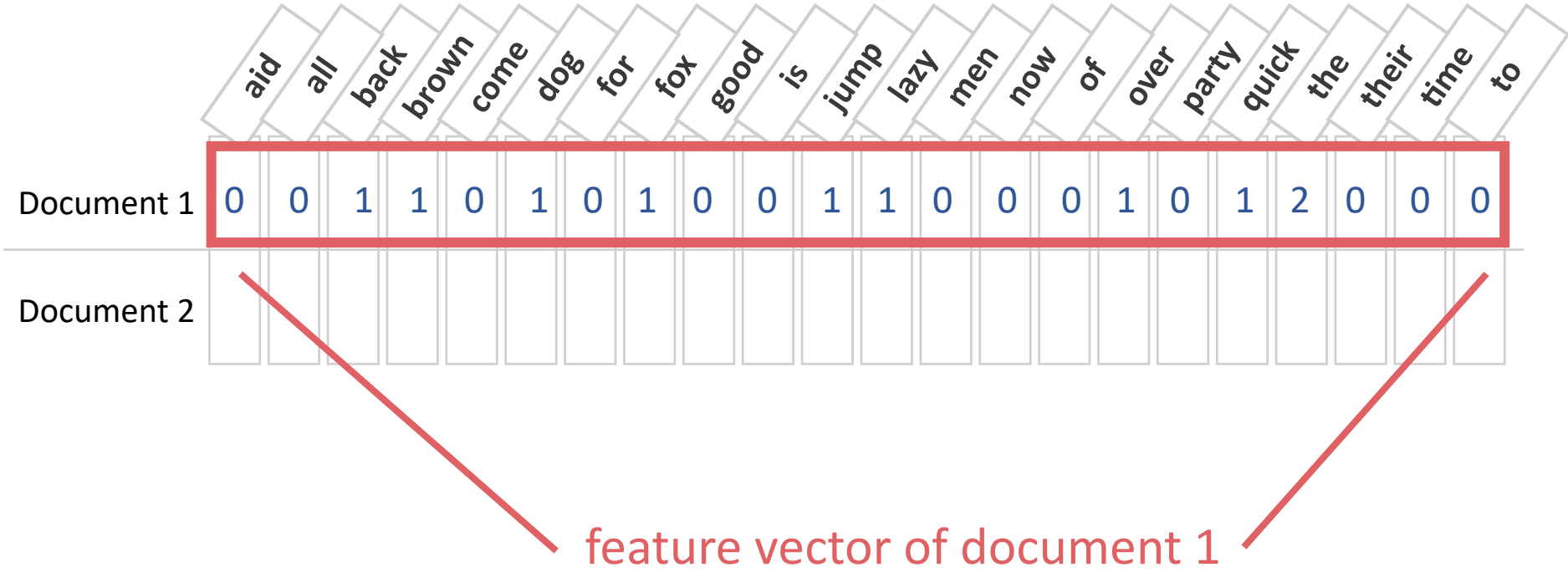
Feature engineering – Bag of words

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.



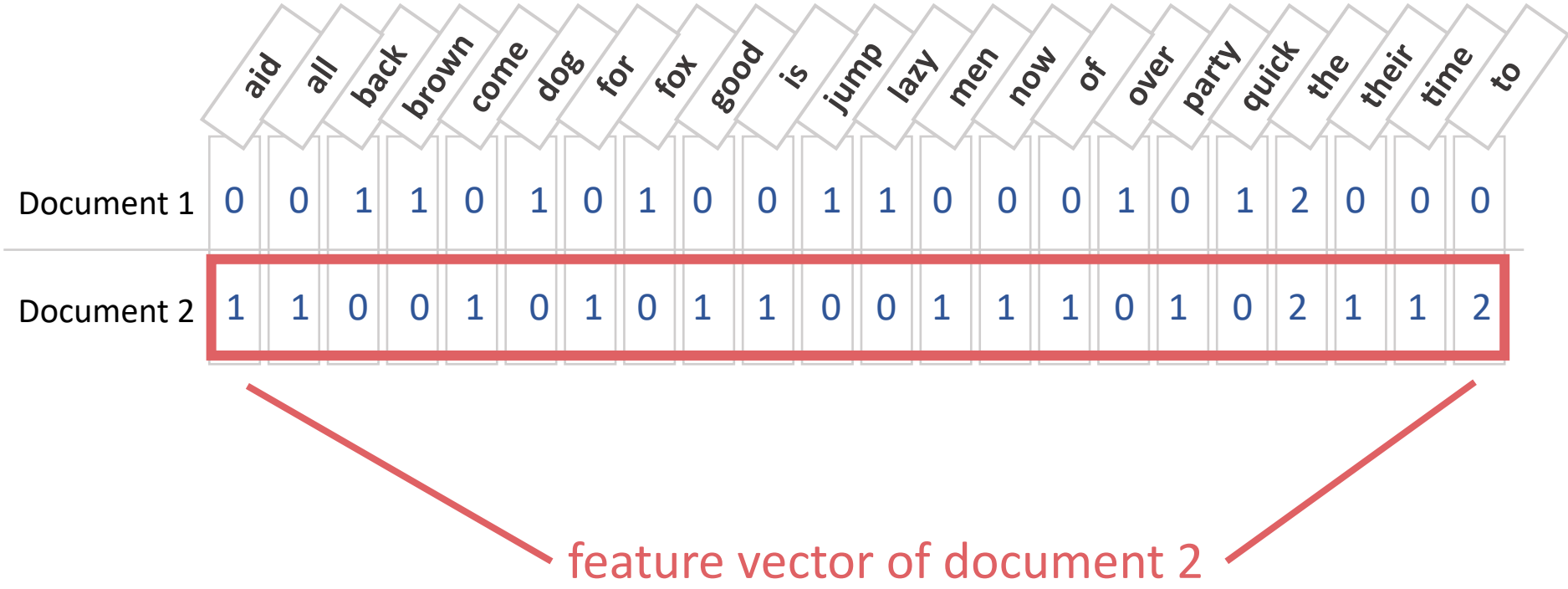
Feature engineering – Bag of words

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.



Feature engineering – Bag of words

Feature extraction

Bag of Words

Using scikit-learn `CountVectorizer` perform word frequency and use these as features to train a model.

```
from sklearn.feature_extraction.text import CountVectorizer
```

Feature engineering – Bag of words

```
# build analyzers (bag-of-words)
BOW_vectorizer = CountVectorizer()
```

Feature engineering – Bag of words

```
# 1. Learn a vocabulary dictionary of all tokens in the raw documents.
BOW_vectorizer.fit(train_df['text'])

# 2. Transform documents to document-term matrix.
train_data_BOW_features = BOW_vectorizer.transform(train_df['text'])
test_data_BOW_features = BOW_vectorizer.transform(test_df['text'])
```

id	text
10018	why are people so offended by kendall he ends ...
10019	I'm about to block everyone everywhere posting...
10020	Making my buddy cry !! Bitch wait for revenge 🤗👌🏻
20171	@KennyCoble @Rosie these horrific situations w...
20172	@BBCNews 😜 scared of their own horror story th...
20173	my husband lost £800 when he booked an apartme...
30123	Food that gets delivered 😍🙌🏻
30124	@ArtyBagger With or without cake seeing your w...
30125	I am a simple human being who just really love...


```
['00', '000', '00pm', '00tiffanyr', ...,'ध_य', 'य_ब', 'सत', 'アニメ']
```

Feature engineering – Bag of words

```
# 1. Learn a vocabulary dictionary of all tokens in the raw documents.
BOW_vectorizer.fit(train_df['text'])

# 2. Transform documents to document-term matrix.
train_data_BOW_features = BOW_vectorizer.transform(train_df['text'])
test_data_BOW_features = BOW_vectorizer.transform(test_df['text'])
```

```
<3613x10115 sparse matrix of type '<class 'numpy.int64'>'
        with 51467 stored elements in Compressed Sparse Row format>
```

	aid	all	back	brown	come	dog	for	fox	good	is	jump	lazy	men	now	of	over	party	quick	the	their	time	to
Document 1	0	0	1	1	0	1	0	1	0	0	1	1	0	0	0	1	0	1	2	0	0	0
Document 2	1	1	0	0	1	0	1	0	1	1	0	0	1	1	1	0	1	0	2	1	1	2

Feature engineering – Bag of words

```
# 1. Learn a vocabulary dictionary of all tokens in the raw documents.
BOW_vectorizer.fit(train_df['text'])

# 2. Transform documents to document-term matrix.
train_data_BOW_features = BOW_vectorizer.transform(train_df['text'])
test_data_BOW_features = BOW_vectorizer.transform(test_df['text'])
```

```
<3613x10115 sparse matrix of type '<class 'numpy.int64'>'
        with 51467 stored elements in Compressed Sparse Row format>
```

```
type(train_data_BOW_features)
```

```
scipy.sparse.csr.csr_matrix
```

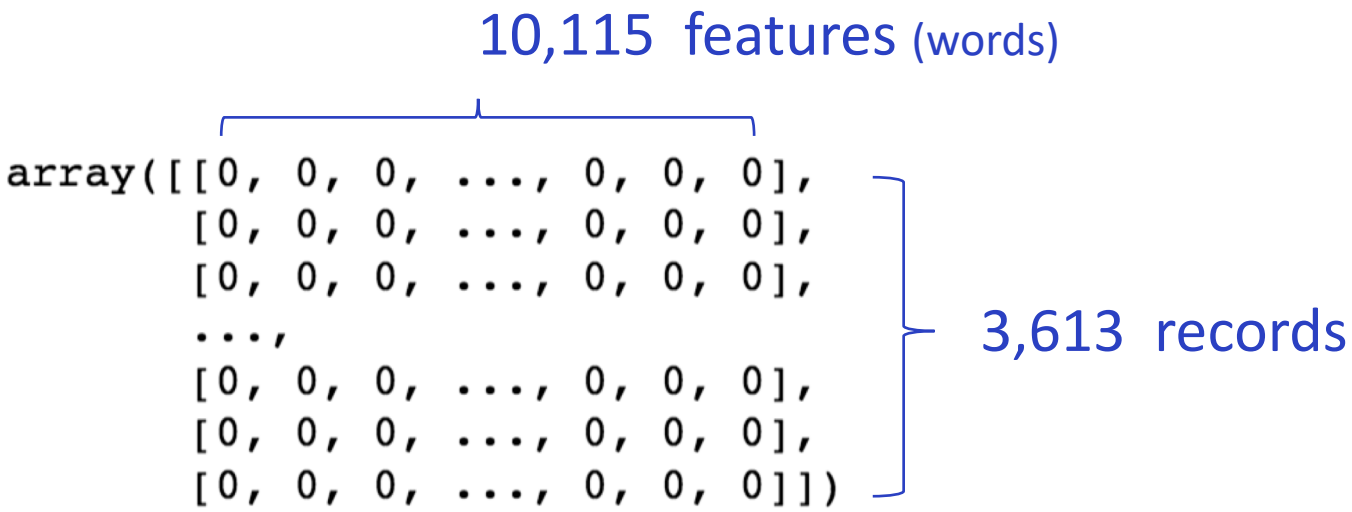
```
train_data_BOW_features.toarray()
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

Feature engineering – Bag of words

```
# check the dimension
train_data_BOW_features.shape
```

```
(3613, 10115)
```



Feature engineering – Bag of words

```
# observe some feature names
feature_names = BOW_vectorizer.get_feature_names()
feature_names[100:110]
```

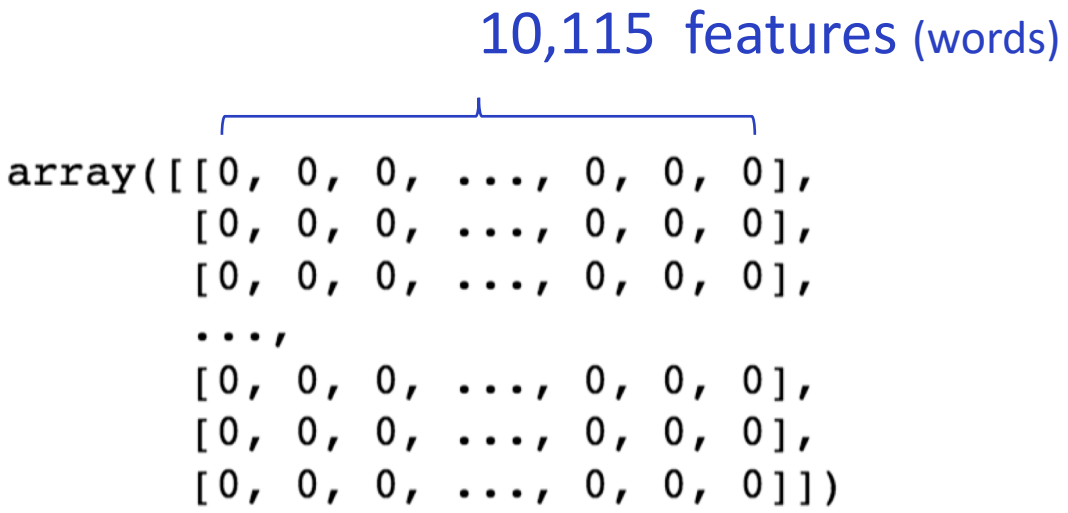
```
['2k17', '2much', '2nd', '30', '300', '301', '30am', '30pm', '30s', '31']
```

Basically, we could put this into classification model now.

Basically, we could put this into classification model now.

However ...

1. curse of dimensionality



Basically, we could put this into classification model now.

However ...

1. curse of dimensionality
2. some important features are ignored

```
"😂 " in feature_names
```

```
False
```

not included here

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

Feature engineering – Bag of words

Adjust the parameters

```
# build analyzers (bag-of-words)
BOW_500 = CountVectorizer(max_features=500, tokenizer=nltk.word_tokenize)
```

```
# build analyzers (bag-of-words)
BOW_vectorizer = CountVectorizer()
```

Feature engineering – Bag of words

Adjust the parameters

```
# build analyzers (bag-of-words)
BOW_500 = CountVectorizer(max_features=500, tokenizer=nltk.word_tokenize)
```

```
# apply analyzer to training data
BOW_500.fit(train_df['text'])

train_data_BOW_features_500 = BOW_500.transform(train_df['text'])

## check dimension
train_data_BOW_features_500.shape
```

```
(3613, 500)
```

500 features

```
array([[1, 0, 0, ..., 0, 0, 0],
       [0, 4, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 1, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

Still 3,613 records

Feature engineering – Bag of words

```
# observe some feature names
feature_names_500 = BOW_500.get_feature_names()
feature_names_500[100:110]
```

```
['cheerful', 'cheering', 'cheery', 'come', 'comes', 'could', 'country', 'cry', 'crying', 'customer']
```

```
# observe some feature names
feature_names = BOW_vectorizer.get_feature_names()
feature_names[100:110]
```

```
['2k17', '2much', '2nd', '30', '300', '301', '30am', '30pm', '30s', '31']
```

Feature engineering – Bag of words

```
# observe some feature names
feature_names_500 = BOW_500.get_feature_names()
feature_names_500[100:110]
```

```
['cheerful', 'cheering', 'cheery', 'come', 'comes', 'could', 'country', 'cry', 'crying', 'customer']
```

```
"😂 " in feature_names
```

```
True
```

included now

```
array([[1, 0, 0, ..., 0, 0, 0],
       [0, 4, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 1, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

Feature engineering – Bag of words

Adjust the parameters

http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

sklearn.feature_extraction.text.CountVectorizer

class sklearn.feature_extraction.text. **CountVectorizer** (*input='content', encoding='utf-8', decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, stop_words=None, token_pattern='(?u)\b\w\w+\b', ngram_range=(1, 1), analyzer='word', max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<class 'numpy.int64'>*) [source]

parameters <default>

Convert a collection of text documents to a matrix of token counts

This implementation produces a sparse representation of the counts using scipy.sparse.csr_matrix.

If you do not provide an a-priori dictionary and you do not use an analyzer that does some kind of feature selection then the number of features will be equal to the vocabulary size found by analyzing the data.

usage

Feature engineering – Bag of words

Parameters:

input : *string {'filename', 'file', 'content'}*

If 'filename', the sequence passed as an argument to fit is expected to be a list of filenames that need reading to fetch the raw content to analyze.

If 'file', the sequence items must have a 'read' method (file-like object) that is called to fetch the bytes in memory.

Otherwise the input is expected to be the sequence strings or bytes items are expected to be analyzed directly.

tokenizer : *callable or None (default)*

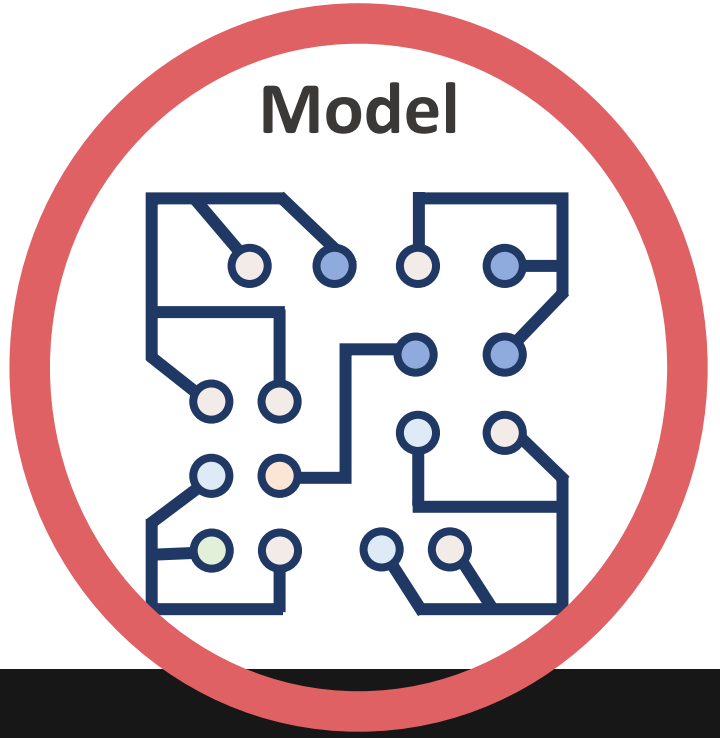
Override the string tokenization step while preserving the preprocessing and n-grams generation steps. Only applies if `analyzer == 'word'`.

max_features : *int or None, default=None*

If not None, build a vocabulary that only consider the top max_features ordered by term frequency across the corpus.

This parameter is ignored if vocabulary is not None.

Text



Emotion

Decision trees

Model

(Use Decision trees)

Model – Decision trees

```
from sklearn.tree import DecisionTreeClassifier
```

Remind: Decision tree is a **supervised learning** algo., it should be trained with labels.

Input: Features & Labels

```
# standardize name (X, y)
X_train = BOW_500.transform(train_df['text'])
y_train = train_df['emotion']

X_test = BOW_500.transform(test_df['text'])
y_test = test_df['emotion']

## check dimension is a good habbit
print('X_train.shape: ', X_train.shape)
print('y_train.shape: ', y_train.shape)
print('X_test.shape: ', X_test.shape)
print('y_test.shape: ', y_test.shape)
```

```
X_train.shape:  (3613, 500)
y_train.shape:  (3613,)
X_test.shape:  (347, 500)
y_test.shape:  (347,)
```

Model – Decision trees

```
## build DecisionTree model
DT_model = DecisionTreeClassifier(random_state=0)
```

class `sklearn.tree.` **`DecisionTreeClassifier`** *(criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False)*

[source]

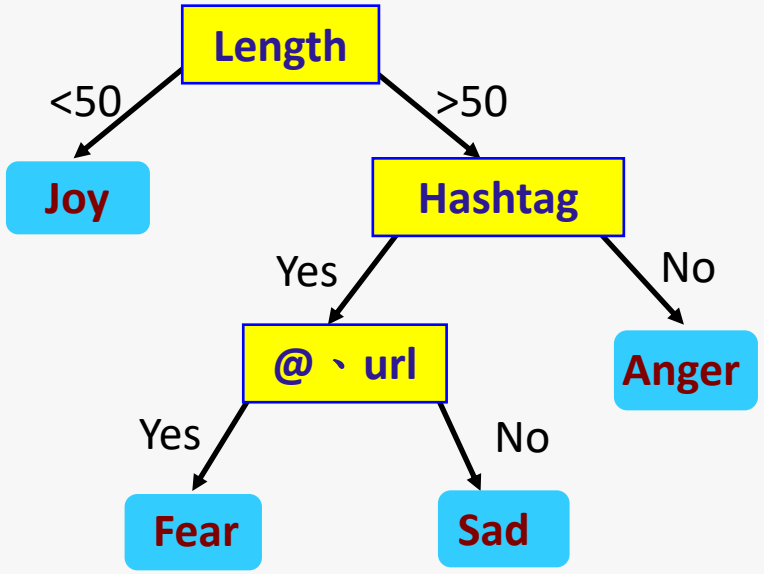
Model – Decision trees

```
## build DecisionTree model
DT_model = DecisionTreeClassifier(random_state=0)

## training!
DT_model = DT_model.fit(X_train, y_train)

## predict!
y_train_pred = DT_model.predict(X_train)
y_test_pred = DT_model.predict(X_test)


## so we get the pred result
y_test_pred[:10]
```



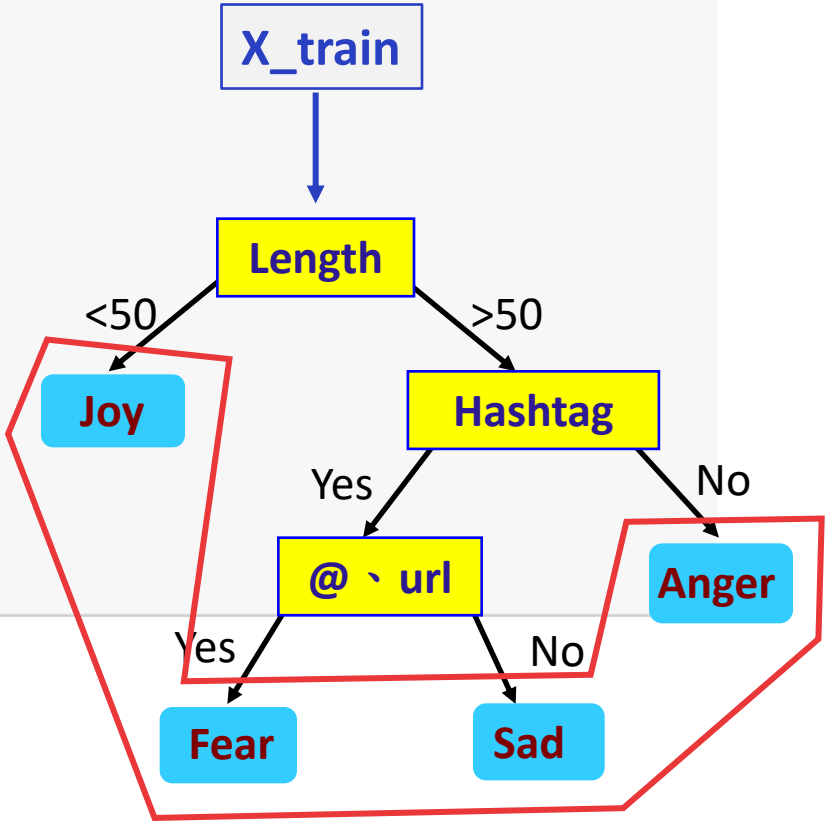
Model – Decision trees

```
## build DecisionTree model
DT_model = DecisionTreeClassifier(random_state=0)

## training!
DT_model = DT_model.fit(X_train, y_train)

## predict!
y_train_pred = DT_model.predict(X_train)
y_test_pred = DT_model.predict(X_test)


## so we get the pred result
y_test_pred[:10]
```



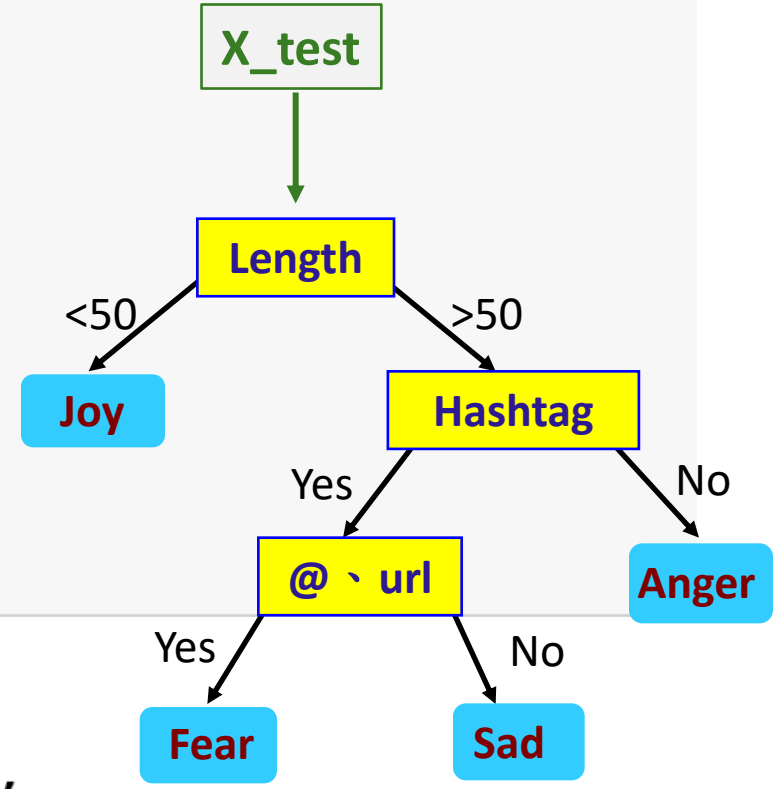
Model – Decision trees

```
## build DecisionTree model
DT_model = DecisionTreeClassifier(random_state=0)

## training!
DT_model = DT_model.fit(X_train, y_train)

## predict!
y_train_pred = DT_model.predict(X_train)
y_test_pred = DT_model.predict(X_test)


## so we get the pred result
y_test_pred[:10]
```




```
array(['anger', 'fear', 'anger', 'joy', 'sadness',
       'sadness', 'anger', 'anger', 'fear', 'sadness'],
      dtype=object)
```

Results evaluation

Model – Statistical results

```
## accuracy
from sklearn.metrics import accuracy_score

acc_train = accuracy_score(y_true=y_train, y_pred=y_train_pred)
acc_test = accuracy_score(y_true=y_test, y_pred=y_test_pred)

print('training accuracy: {}%'.format(round(acc_train, 2)))
print('testing accuracy: {}%'.format(round(acc_test, 2)))
```

```
training accuracy: 0.99

testing accuracy: 0.66
```

Model – Statistical results

```
## classification_report
from sklearn.metrics import classification_report

print(classification_report(y_true=y_test, y_pred=y_test_pred))
```

```
              precision    recall  f1-score   support

       anger       0.70      0.68      0.69        84
        fear       0.68      0.68      0.68       110
         joy       0.62      0.67      0.65        79
     sadness       0.63      0.59      0.61        74

   micro avg       0.66      0.66      0.66       347
   macro avg       0.66      0.66      0.66       347
weighted avg       0.66      0.66      0.66       347
```

Model – Confusion matrix

```
## confusion_matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_true=y_test, y_pred=y_test_pred)
print(cm)
```

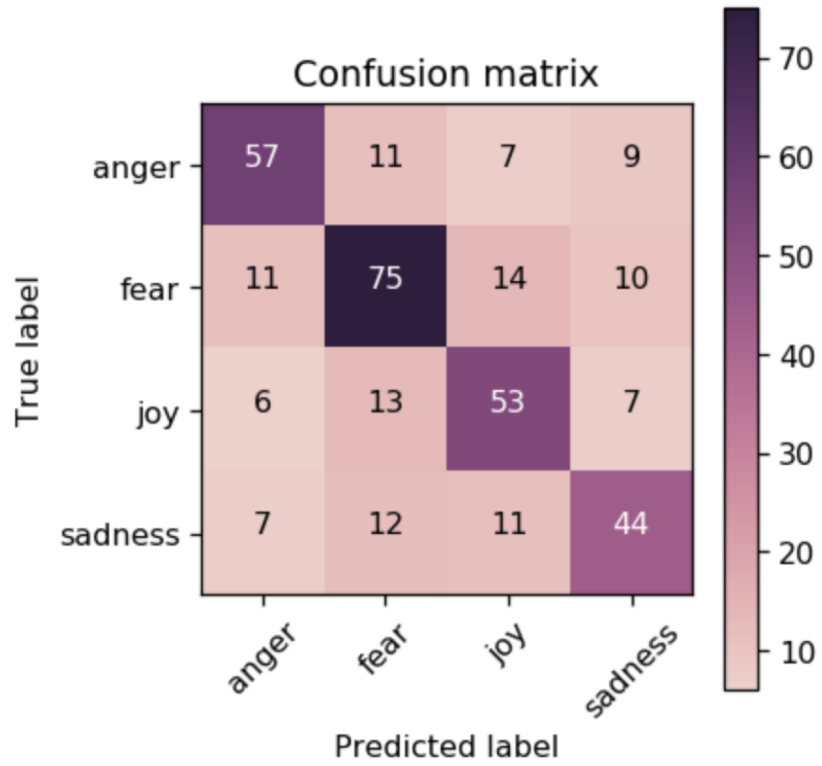
```
[[57 11  7  9]
 [11 75 14 10]
 [ 6 13 53  7]
 [ 7 12 11 44]]
```

```
def plot_confusion_matrix(cm, classes, normalize=False, title='Confusion matrix',
                          cmap=sns.cubehelix_palette(as_cmap=True)):
```

Model – Confusion matrix

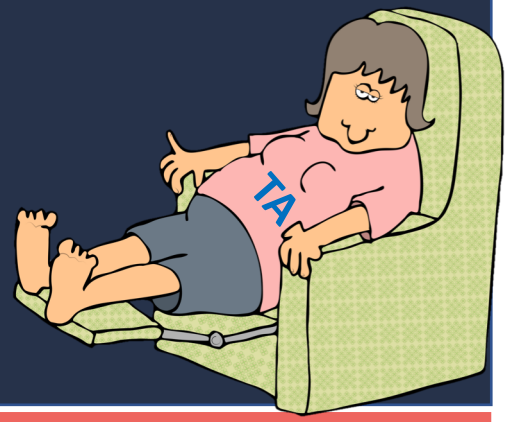
```
# plot here
my_tags = ['anger', 'fear', 'joy', 'sadness']
plot_confusion_matrix(cm, classes=my_tags, title='Confusion matrix')
```

```
Confusion matrix, without normalization
```



Q & A

Your show time now.



Task –

Everyone need to fill in the **8 cells** in google sheet we assigned to you.

Student ID	Name	Bag of words				TF-IDF			
		dimension = 500	dim = 3000	dim = 500	dim = 3000	dim = 500	dim = 3000	dim = 500	dim = 3000
		Decision trees	Decision trees	Naive baysian	Naive baysian	Decision trees	Decision trees	Naive baysian	Naive baysian
example	TA	0.99 / 0.66	0.99 / 0.79	0.79 / 0.71	0.95 / 0.74	0.99 / 0.64	0.99 / 0.73	0.78 / 0.71	0.90 / 0.72

Fill in the corresponding “accuracy” into cells.

The format should be: **training accuracy / testing accuracy**

Task –

And you nee to change the parameters to match our requirements

Note: we assign different requirement for each of you

Student ID	Name	Settings								
		make the features by		stopwords		tokenize by		N-gram		
		word	char	remove English stopwords	retain	nltk.word_tokenize	no tokenize	1-and-2-gram	1-to-3-gram	2-to-4-gram
example	TA	v			v	v		defaul		

Task –

Check the official document for help (we provide the link in jupyter notebook)

sklearn.feature_extraction.text.CountVectorizer

class `sklearn.feature_extraction.text.` **`CountVectorizer`** (*input='content', encoding='utf-8', decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, stop_words=None, token_pattern='(?u)\b\w\w+\b', ngram_range=(1, 1), analyzer='word', max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<class 'numpy.int64'>*) [source]

Parameters:

input : *string {'filename', 'file', 'content'}*

If 'filename', the sequence passed as an argument to fit is expected to be a list of filenames that need reading to fetch the raw content to analyze.

If 'file', the sequence items must have a 'read' method (file-like object) that is called to fetch the bytes in memory.

Otherwise the input is expected to be the sequence strings or bytes items are expected to be analyzed directly.

tokenizer : *callable or None (default)*

Override the string tokenization step while preserving the preprocessing and n-grams generation steps. Only applies if `analyzer == 'word'`.

Task –

- You have **30 minutes** to do this.

  (You are allowed to discuss with your classmate)

- This will count as one preview quiz.

- You could get some bonus for sharing what interesting thing you find on iLMS forum (Forum title: **Lab2 in-class discussion**).

Good luck!