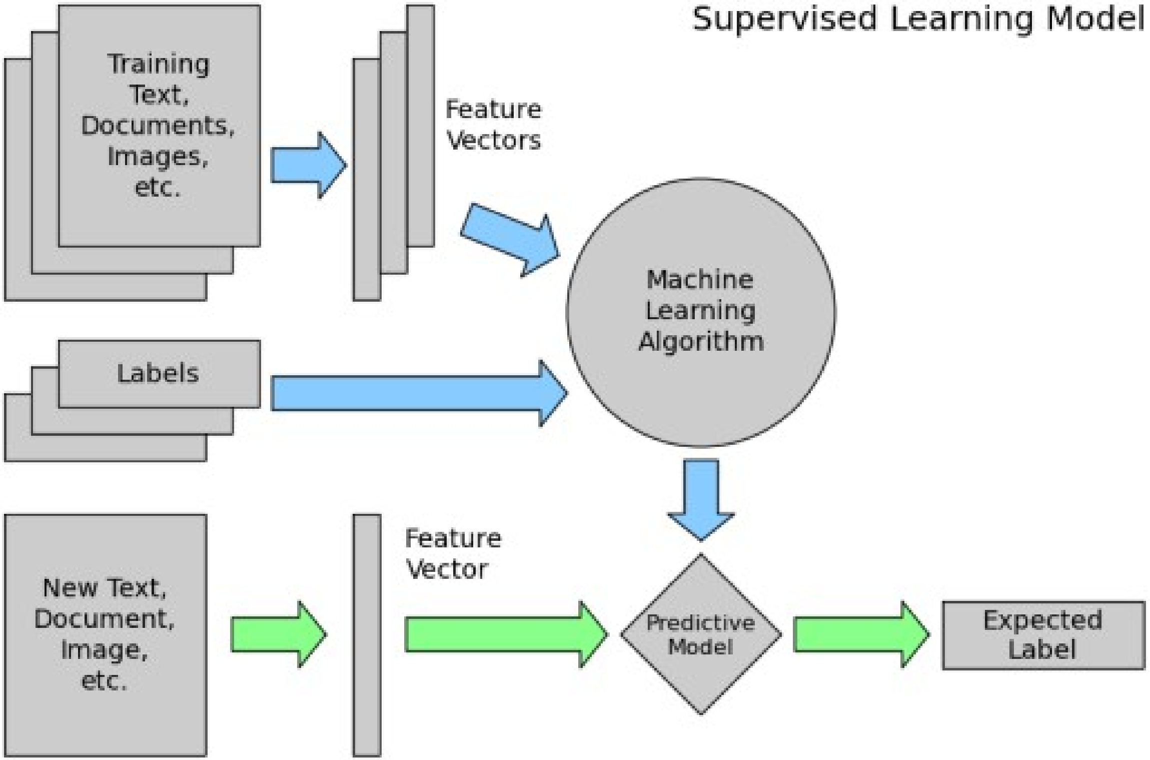
Lecture 10

Learning from Text

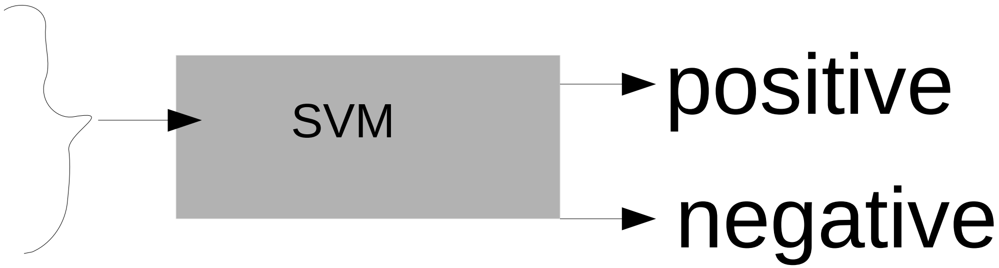
By Nazerke Sultanova

Working with text:


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

Example:

- Nice day
- A very nice day



SVM

positive

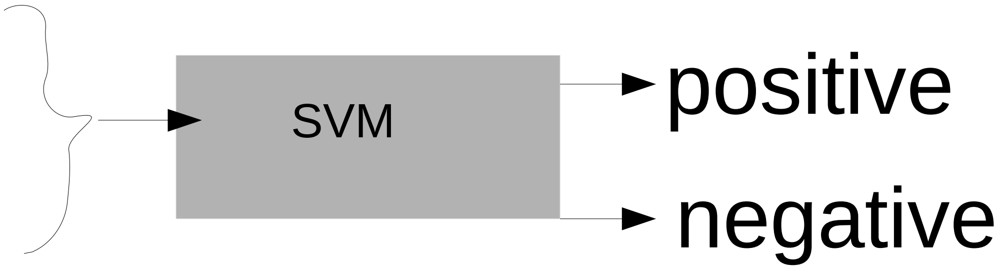
negative

- How many features needed?

A) 1 B) 2 C) 3 D) 5 E) hard to tell

Example:

- Nice day
- A very nice day



SVM

positive

negative

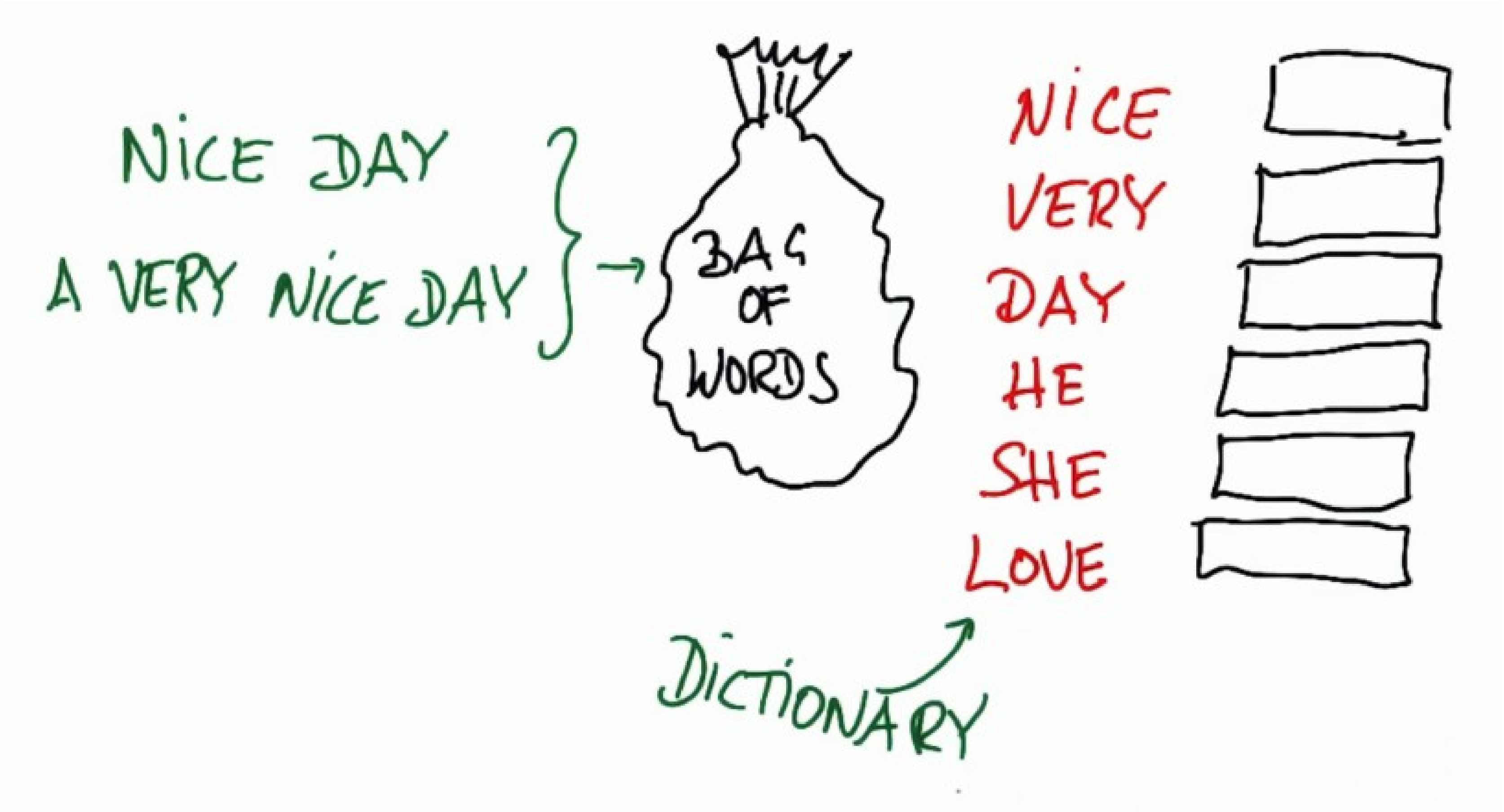
- How many features needed?

A) 1 B) 2 C) 3 D) 5 E) hard to tell

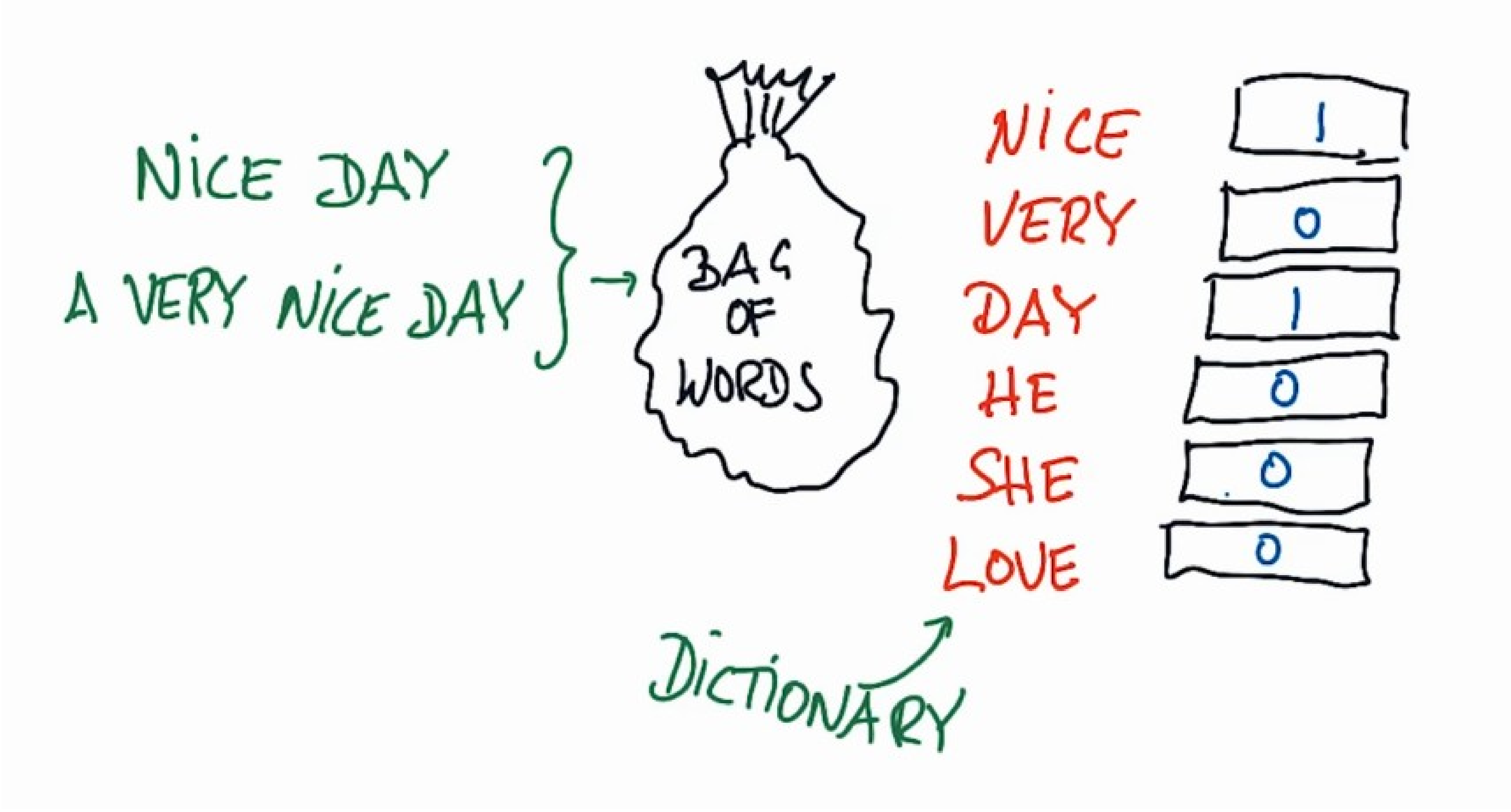
Learning from text - frequency

NICE DAY
A VERY NICE DAY
BAG
OF
WORDS

- Fill in frequency vector for a first message:

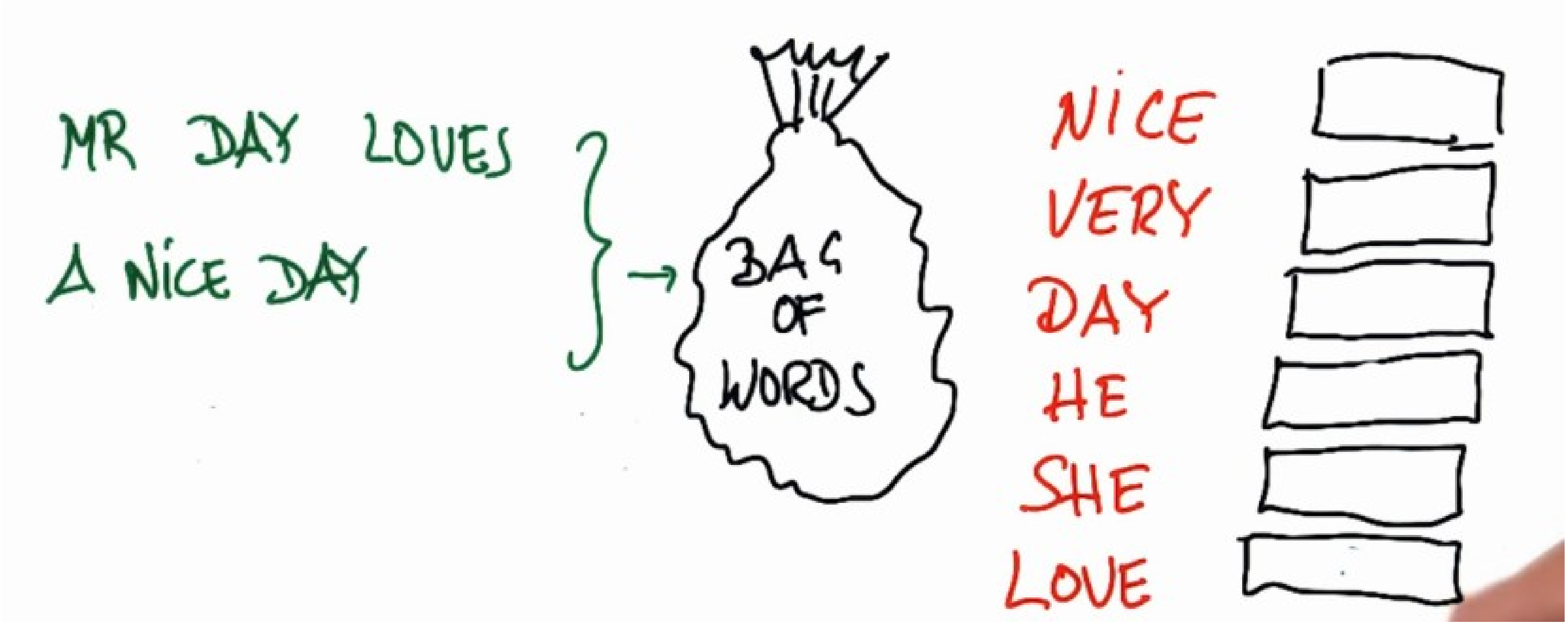


- Fill in frequency vector for a first message:



NICE	1
VERY	0
DAY	1
HE	0
SHE	0
LOVE	0

- BAG of WORDS works for any text



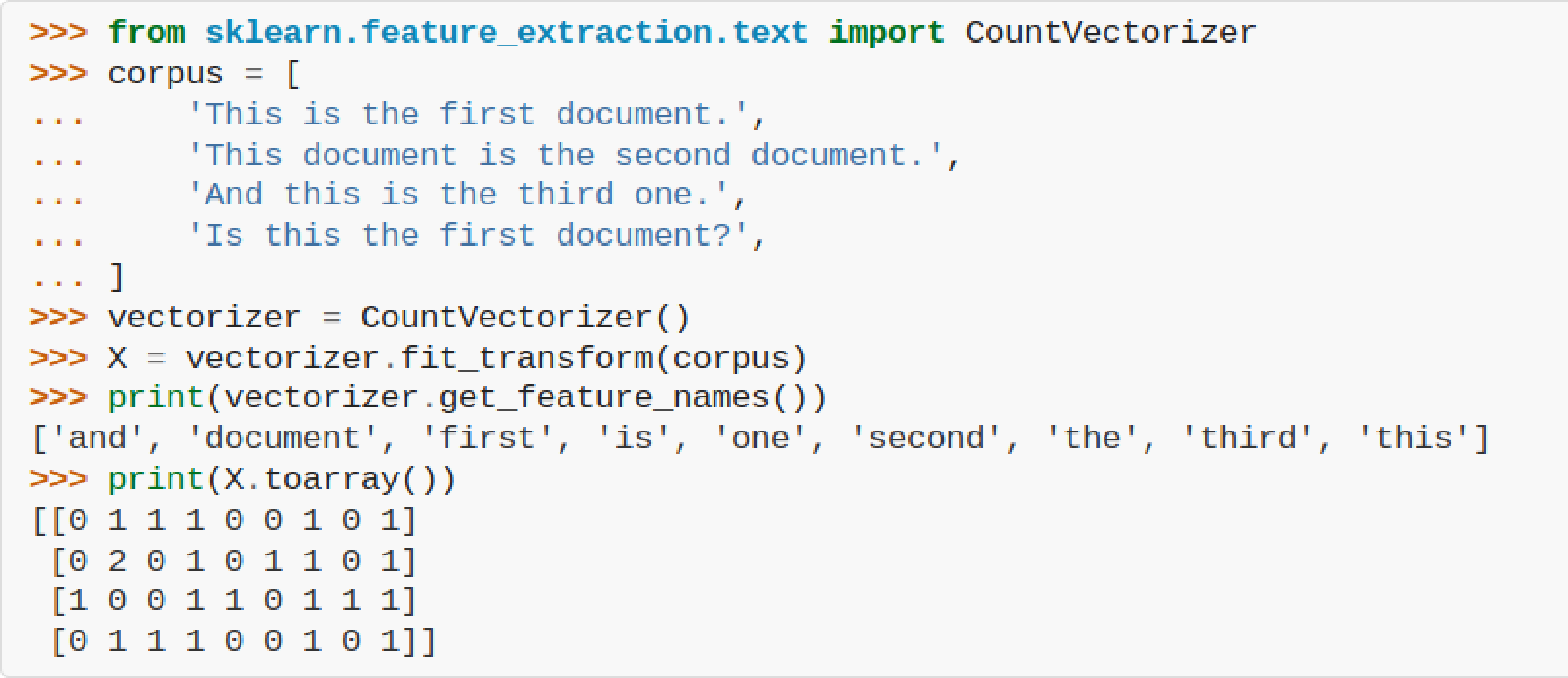
BAG OF WORDS properties:

- Does the word order matter?
- Do long phrases give different input vectors?
- Can we handle complex phrases?
  - Ex: “Chicago Bulls”

BAG OF WORDS properties:

- Does the word order matter? No
- Do long phrases give different input vectors? Yes
- Can we handle complex phrases? No
  - Ex: “Chicago Bulls”

- BAG OF WORDS in sklearn - CountVectorizer



```
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> corpus = [
...     'This is the first document.',
...     'This document is the second document.',
...     'And this is the third one.',
...     'Is this the first document?',
... ]
>>> vectorizer = CountVectorizer()
>>> X = vectorizer.fit_transform(corpus)
>>> print(vectorizer.get_feature_names())
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
>>> print(X.toarray())
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```

Vocabulary: not all words are equal

- Some words contain more information than others
- Find the low information words:

  [‘hi’, ‘Katie’, ‘will’, ‘the’, ‘driving’, ‘machine’, ‘learning’, ‘great’]

Vocabulary: not all words are equal

- Some words contain more information than others
- Find the low information words:

  [‘hi’, ‘Katie’, ‘will’, ‘the’, ‘driving’, ‘machine’, ‘learning’, ‘great’]

Vocabulary: not all words are equal

- Some words contain more information than others
- Stopwords – low information words that occur highly frequently
- Examles: and, the, I, have, you..
-

Remove stopwords before doing any analysis with text!

Example:

- Stopwords: [the, in, for, you, will, have, be]
- How many words will be removed when we remove stopwords from:

  ‘Hi Katie, the machine learning class will be great’

Example:

- Stopwords: [the, in, for, you, will, have, be]
- How many words will be removed when we remove stopwords from:

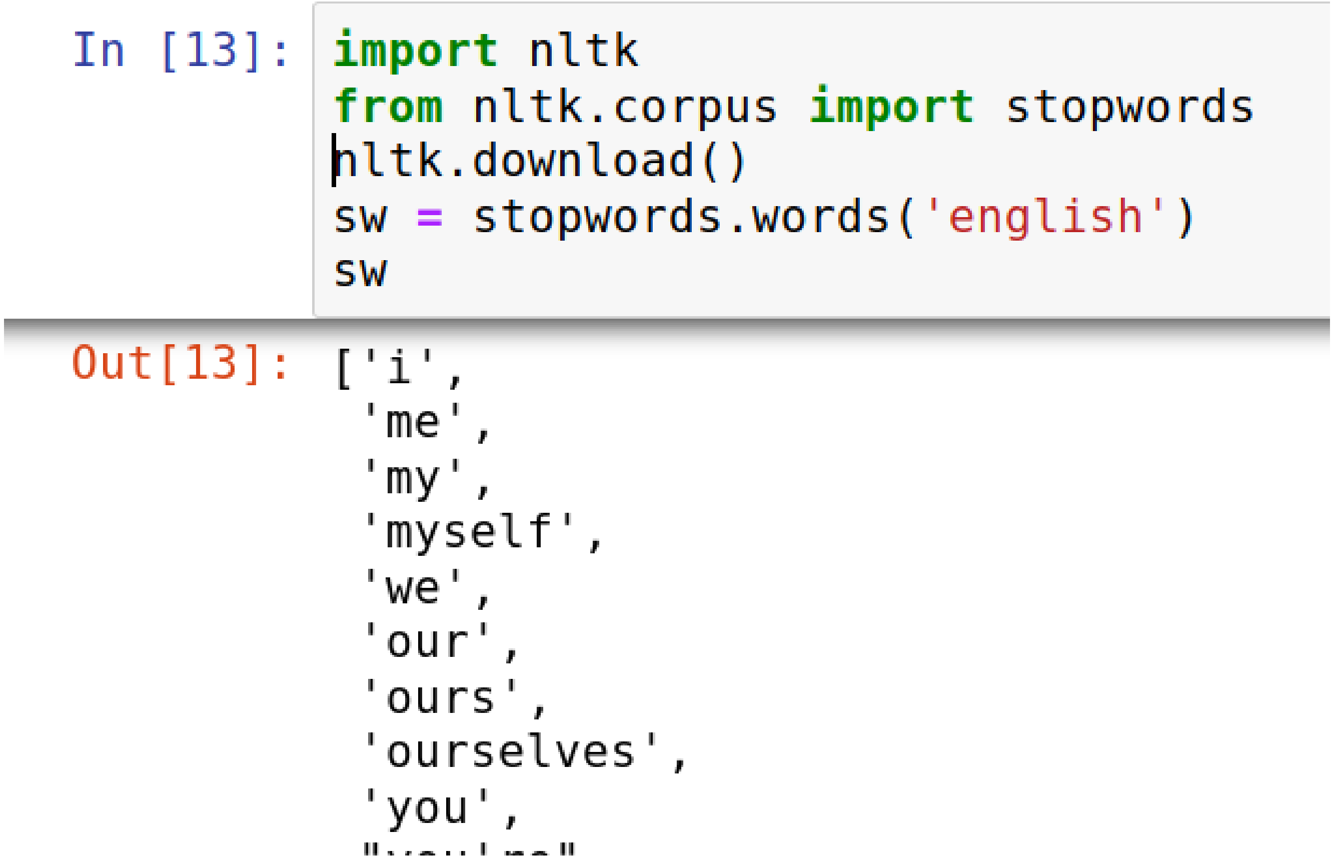
  ‘Hi Katie, the machine learning class will be great’

- Answer: 3

How to get a list of stopwords?

- Python package NLTK
- Natural Language Toolkit (NLTK) is a leading platform for building Python programs to work with human language data

Importing nltk



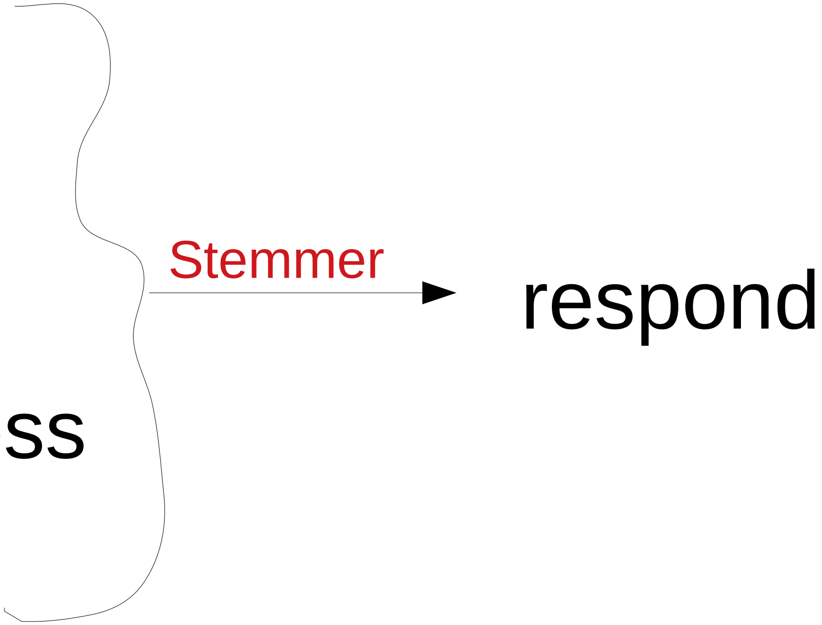
```
In [13]: import nltk
         from nltk.corpus import stopwords
         nltk.download()
         sw = stopwords.words('english')
         sw

Out[13]: ['i',
          'me',
          'my',
          'myself',
          'we',
          'our',
          'ours',
          'ourselves',
          'you',
          "you're"
```

Vocabulary –
different words with one root

- Unresponsive
- Response
- Responsivity
- Responsiveness
- Respond

Vocabulary – different words with one root

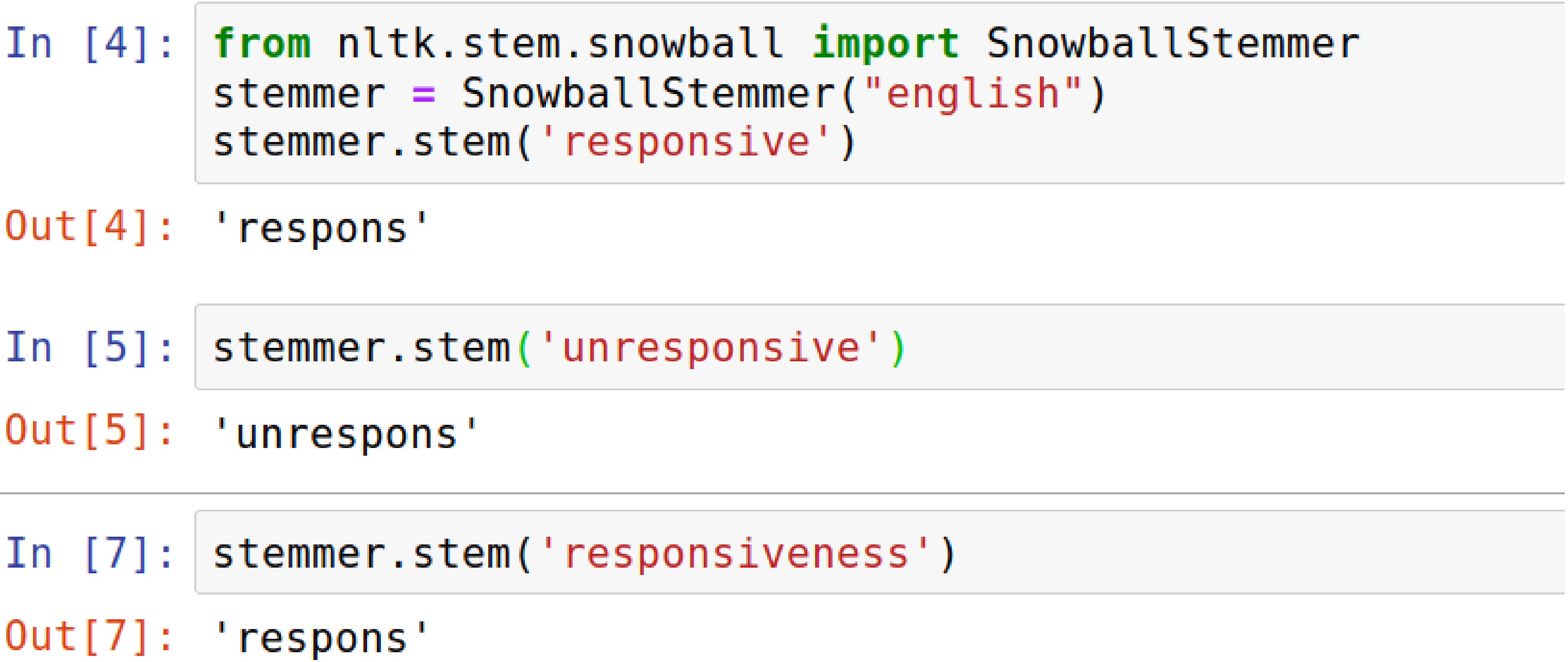
- Unresponsive
- Response
- Responsivity
- Responsiveness
- Respond



Stemmer

respond

Example of stemming - **nltk**



```
In [4]: from nltk.stem.snowball import SnowballStemmer
        stemmer = SnowballStemmer("english")
        stemmer.stem('responsive')

Out[4]: 'respons'

In [5]: stemmer.stem('unresponsive')

Out[5]: 'unrespons'

In [7]: stemmer.stem('responsiveness')

Out[7]: 'respons'
```

Order of operations in text processing

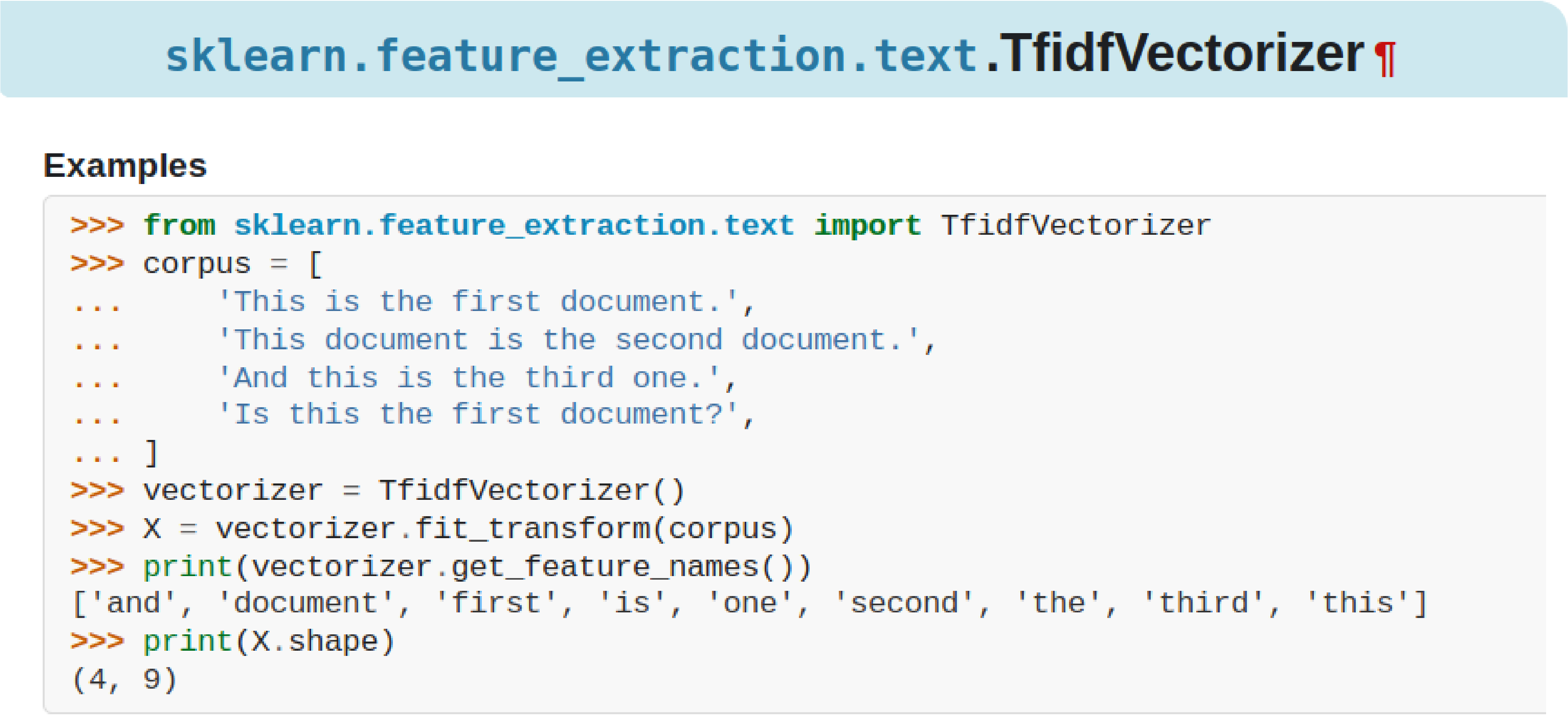
- Which one do we need to do first?
  - Bag of words
  - Stemming
- Why?

Order of operations in text processing

- Which one do we need to do first?
  - Bag of words
  - Stemming

TfIdf Representation

- Tf – term frequency – like bag of words
- Idf – inverse document frequency – weighting by how often word occur in a corpus.
  - Rare words weight higher
  - Help to distinguish different messages



sklearn.feature_extraction.text.TfidfVectorizer ¶

Examples

```
>>> from sklearn.feature_extraction.text import TfidfVectorizer
>>> corpus = [
...     'This is the first document.',
...     'This document is the second document.',
...     'And this is the third one.',
...     'Is this the first document?',
... ]
>>> vectorizer = TfidfVectorizer()
>>> X = vectorizer.fit_transform(corpus)
>>> print(vectorizer.get_feature_names())
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
>>> print(X.shape)
(4, 9)
```