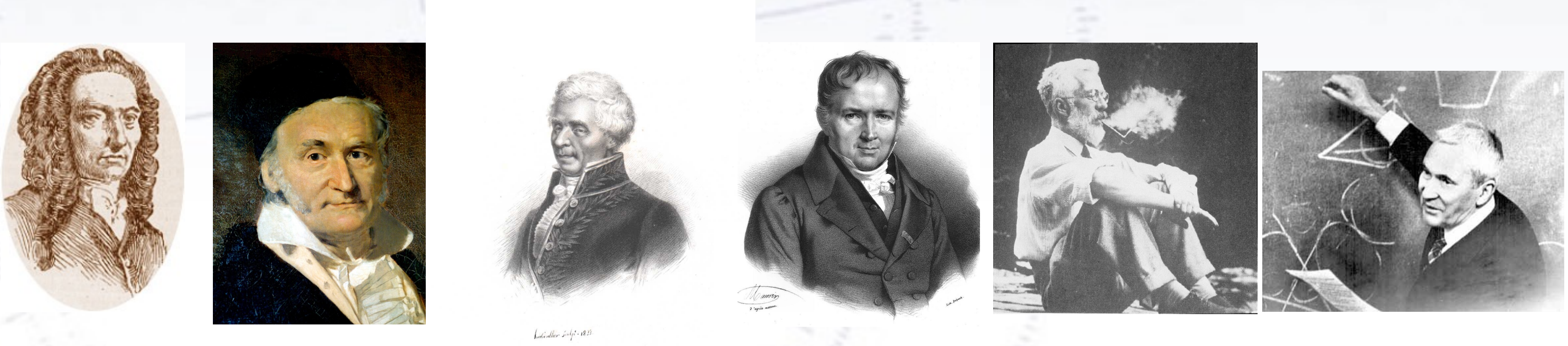
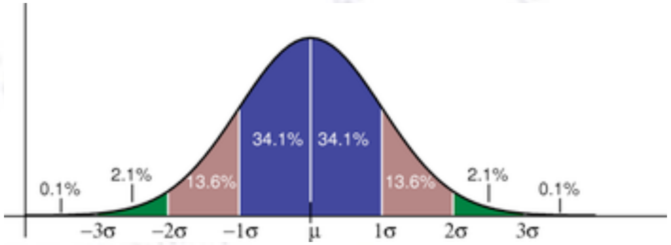
Applied ML

Natural Language Processing (NLP)



Troels C. Petersen (NBI)



“Statistics is merely a quantisation of common sense - Machine Learning is a sharpening of it!”

Housing Prices

“Inside” vs “outside” variables

Problem with all the variables until now has been that all of the variables are “outside” variables.

(square meter size, distance to X, floor number etc).

We need ”inside” variables for extra information

(kitchen condition, bathroom condition etc.)

So we use the descriptions of the houses!

Text Analysis

Natural Language Processing

How to use descriptions:

Flot delevenlig ejerlejlighed på Frederiksbjerg - tilbagetrukket fra vejen i hyggeligt baghus!

Køkkenet fremstår pænt og velholdt, og har hårde hvidevarer i stål.

Det sidste værelse i lejligheden er helt sit eget, og har en spændende indretning.

Text Analysis

Natural Language Processing

Multiple documents:

1)	*"The villa is big."*	3,000,000 DKK
2)	*"The apartment needs to be renovated."*	1,000,000 DKK
3)	*"With a big, newly renovated kitchen."*	2,000,000 DKK

Remove punctuation

Stop words: *the, is, a, to, be*

Text Analysis

Natural Language Processing

1) *“villa big”*
2) *“apartment needs renovated”*
3) *“with big newly renovated kitchen”*

Text Analysis

Natural Language Processing

1) *"villa big"*
2) *"apartment needs renovated"*
3) *"with big newly renovated kitchen"*

Bag of words:

	apartment	big	kitchen	needs	newly	renovated	villa	with
1)	0	1	0	0	0	0	1	0

Text Analysis

Natural Language Processing

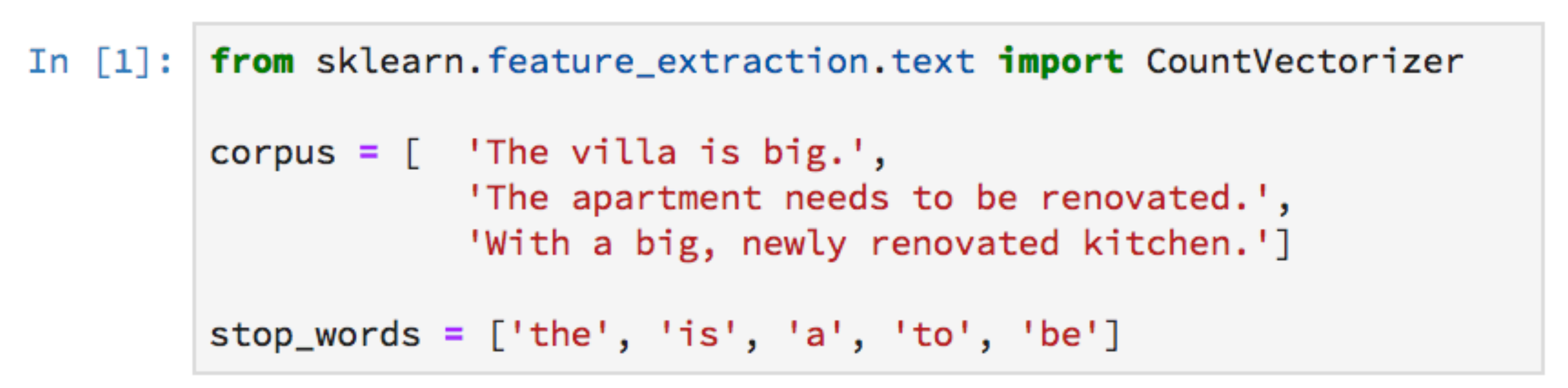
1) *"villa big"*
2) *"apartment needs renovated"*
3) *"with big newly renovated kitchen"*

Bag of words:

	apartment	big	kitchen	needs	newly	renovated	villa	with
1)	0	1	0	0	0	0	1	0
2)	1	0	0	1	0	1	0	0
3)	0	1	1	0	1	1	0	1

Text Analysis

Natural Language Processing

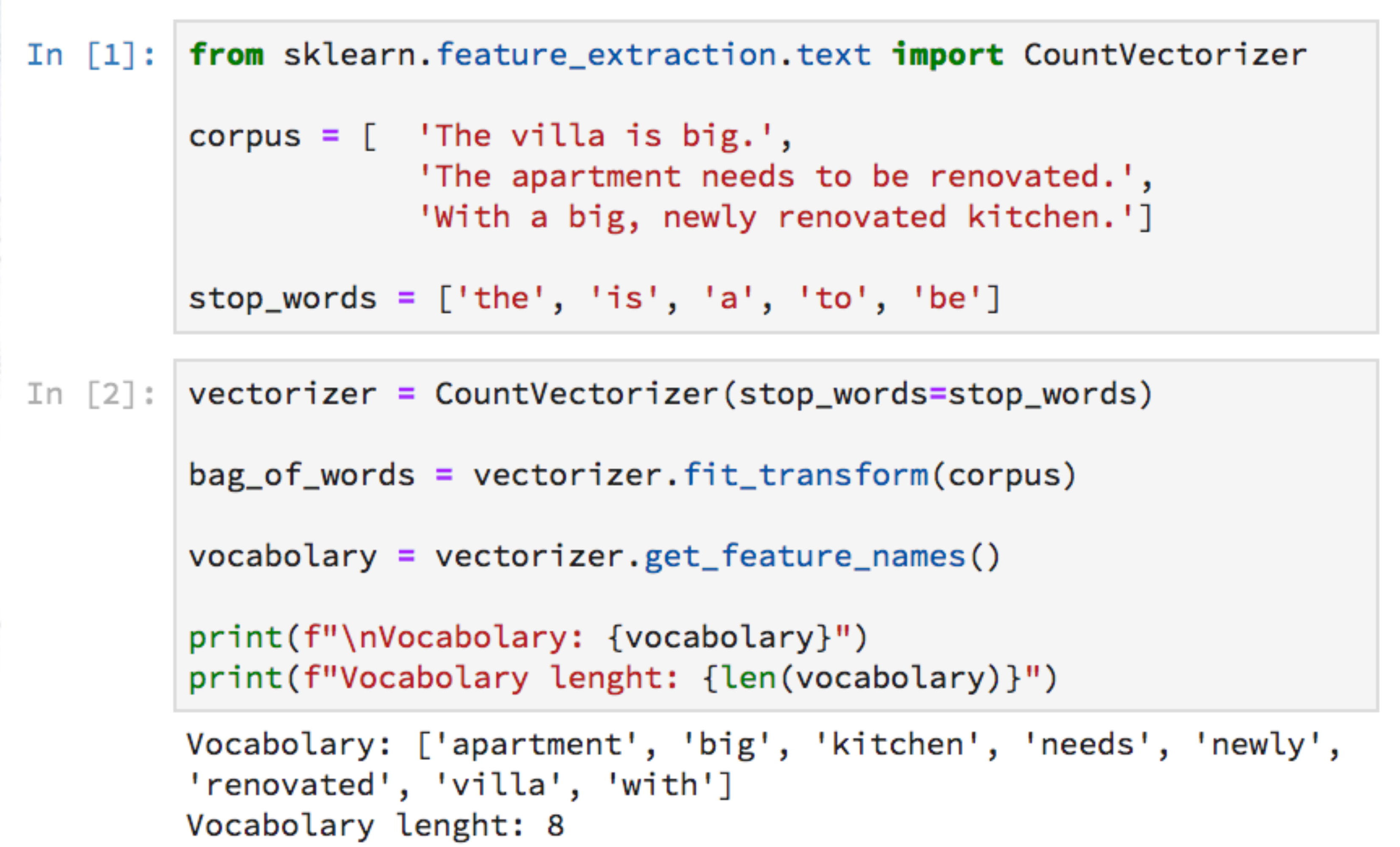


In [1]:

```
from sklearn.feature_extraction.text import CountVectorizer

corpus = [  'The villa is big.',
            'The apartment needs to be renovated.',
            'With a big, newly renovated kitchen.']

stop_words = ['the', 'is', 'a', 'to', 'be']
```

Text Analysis

Natural Language Processing



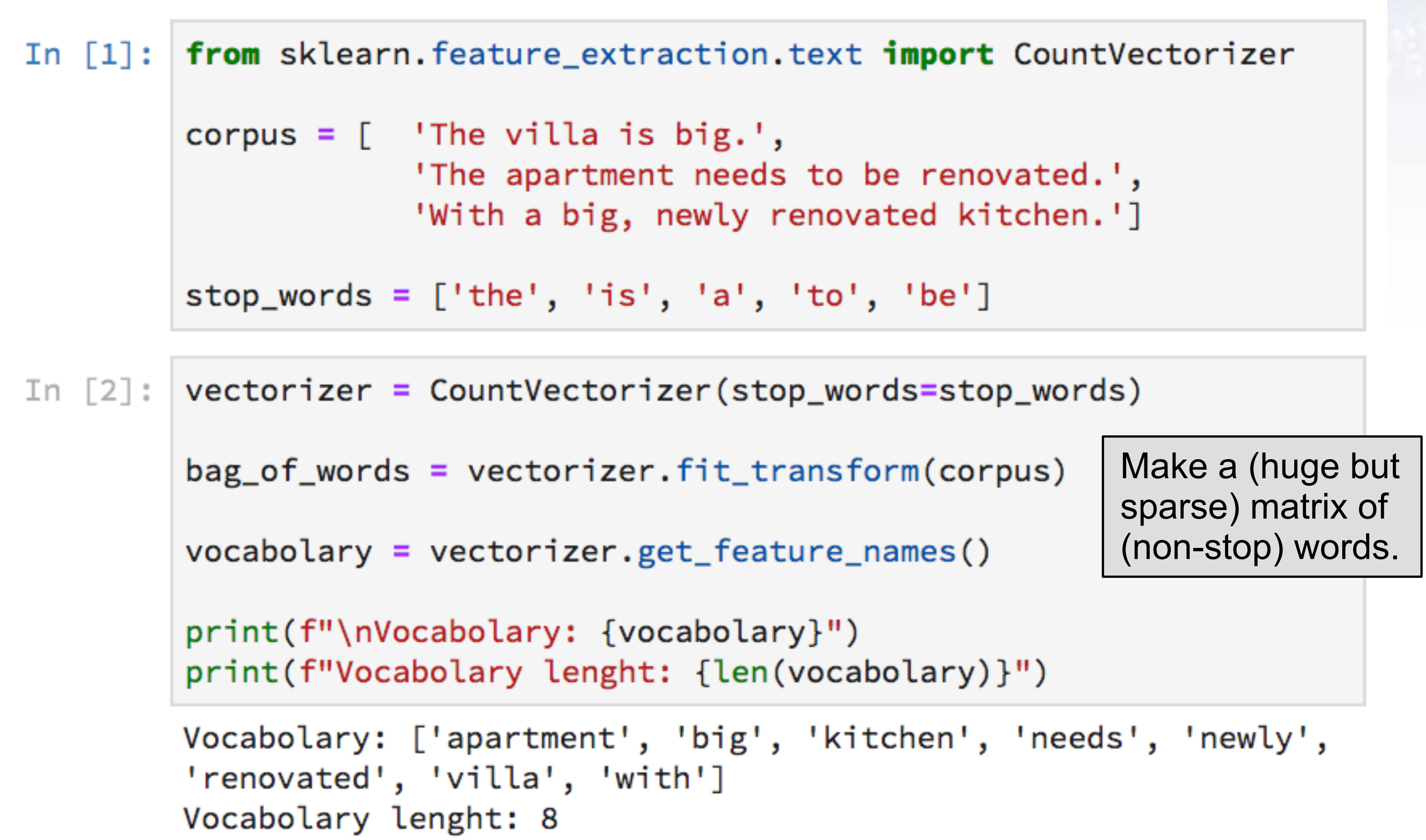
```
In [1]: from sklearn.feature_extraction.text import CountVectorizer

        corpus = [  'The villa is big.',
                    'The apartment needs to be renovated.',
                    'With a big, newly renovated kitchen.']

        stop_words = ['the', 'is', 'a', 'to', 'be']
```

```
In [2]: vectorizer = CountVectorizer(stop_words=stop_words)

        bag_of_words = vectorizer.fit_transform(corpus)

        vocabolary = vectorizer.get_feature_names()

        print(f"\nVocabolary: {vocabolary}")
        print(f"Vocabolary lenght: {len(vocabolary)}")
```

```
Vocabolary: ['apartment', 'big', 'kitchen', 'needs', 'newly',
'renovated', 'villa', 'with']
Vocabolary lenght: 8
```

Text Analysis

Natural Language Processing



```
In [1]: from sklearn.feature_extraction.text import CountVectorizer

        corpus = [  'The villa is big.',
                    'The apartment needs to be renovated.',
                    'With a big, newly renovated kitchen.']

        stop_words = ['the', 'is', 'a', 'to', 'be']
```

```
In [2]: vectorizer = CountVectorizer(stop_words=stop_words)

        bag_of_words = vectorizer.fit_transform(corpus)

        vocabolary = vectorizer.get_feature_names()

        print(f"\nVocabolary: {vocabolary}")
        print(f"Vocabolary lenght: {len(vocabolary)}")
```

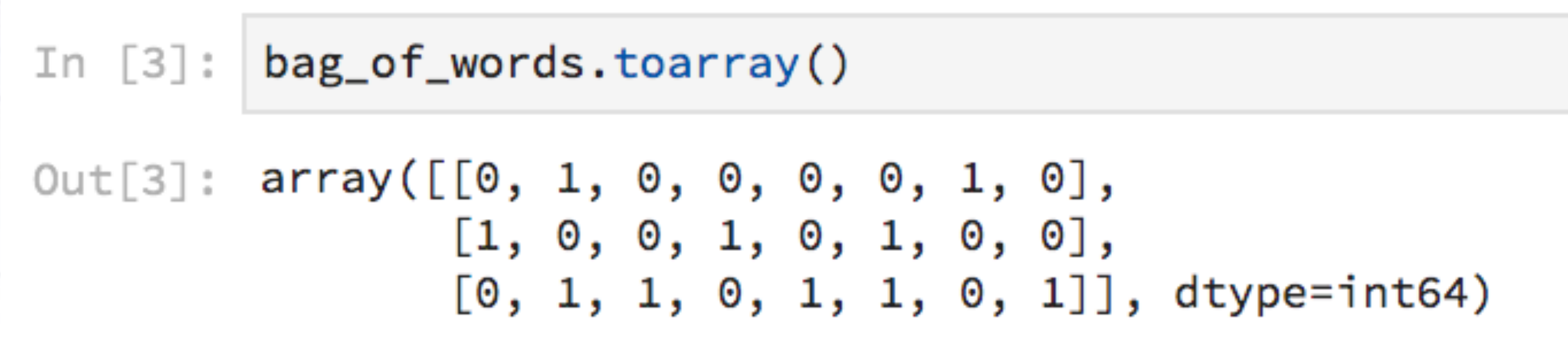
```
Vocabolary: ['apartment', 'big', 'kitchen', 'needs', 'newly',
'renovated', 'villa', 'with']
Vocabolary lenght: 8
```

Make a (huge but sparse) matrix of (non-stop) words.

Text Analysis

Natural Language Processing

Show the matrix/array.



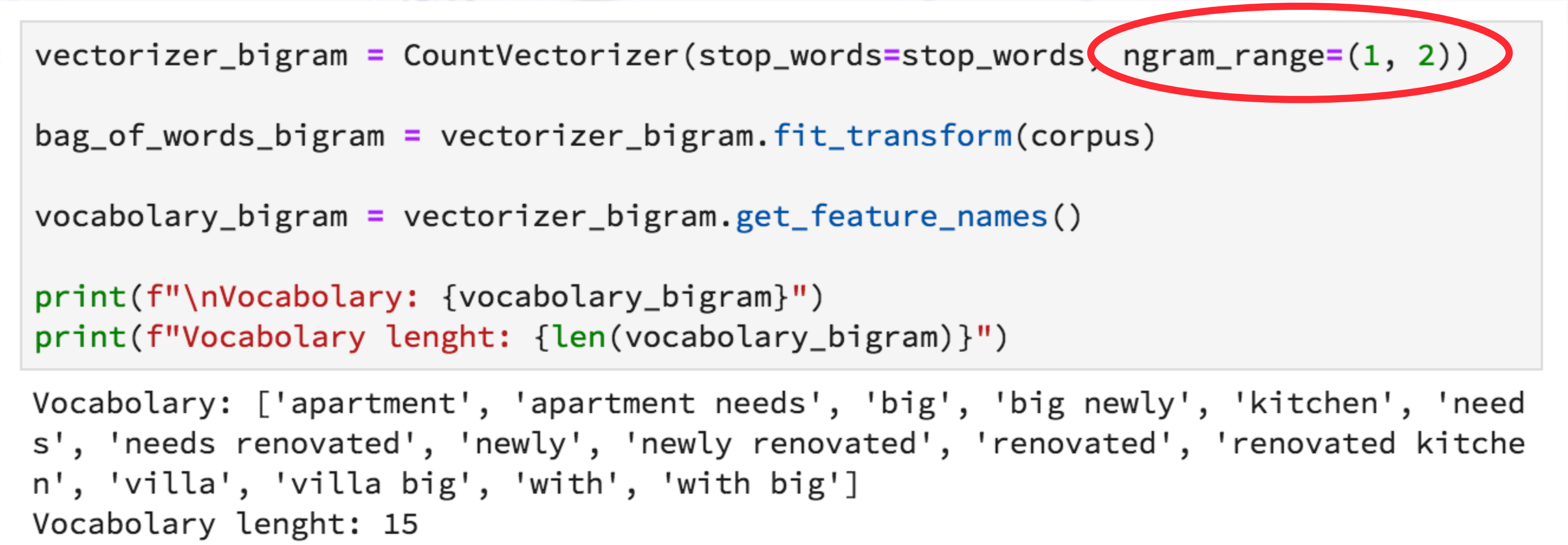
```
In [3]: bag_of_words.toarray()

Out[3]: array([[0, 1, 0, 0, 0, 0, 1, 0],
               [1, 0, 0, 1, 0, 1, 0, 0],
               [0, 1, 1, 0, 1, 1, 0, 1]], dtype=int64)
```

	apartment	big	kitchen	needs	newly	renovated	villa	with
1)	0	1	0	0	0	0	1	0
2)	1	0	0	1	0	1	0	0
3)	0	1	1	0	1	1	0	1

Text Analysis

Natural Language Processing

Bi-grams:



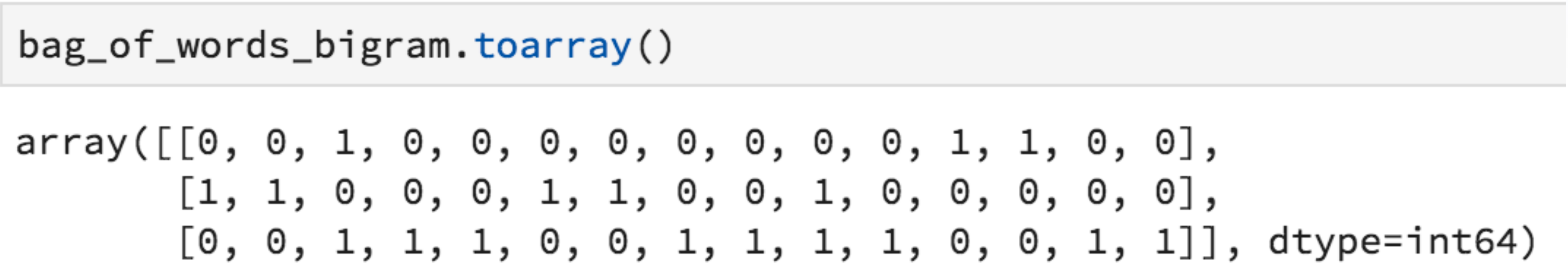
```
vectorizer_bigram = CountVectorizer(stop_words=stop_words, ngram_range=(1, 2))

bag_of_words_bigram = vectorizer_bigram.fit_transform(corpus)

vocabolary_bigram = vectorizer_bigram.get_feature_names()

print(f"\nVocabolary: {vocabolary_bigram}")
print(f"Vocabolary lenght: {len(vocabolary_bigram)}")
```

```
Vocabolary: ['apartment', 'apartment needs', 'big', 'big newly', 'kitchen', 'need
s', 'needs renovated', 'newly', 'newly renovated', 'renovated', 'renovated kitche
n', 'villa', 'villa big', 'with', 'with big']
Vocabolary lenght: 15
```

Text Analysis

Natural Language Processing

Bi-grams:



```
bag_of_words_bigram.toarray()
```

```
array([[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0],
       [1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1]], dtype=int64)
```

Text Analysis

Natural Language Processing

Term Frequency - Inverse Document Frequency: *TF-IDF*

Natural weighting of words

CountVectorizer, TfidfVectorizer

Assign a weight to each word, according to its frequency of use. weight_IDF = log(N_{all} / $N_{appearances}$)

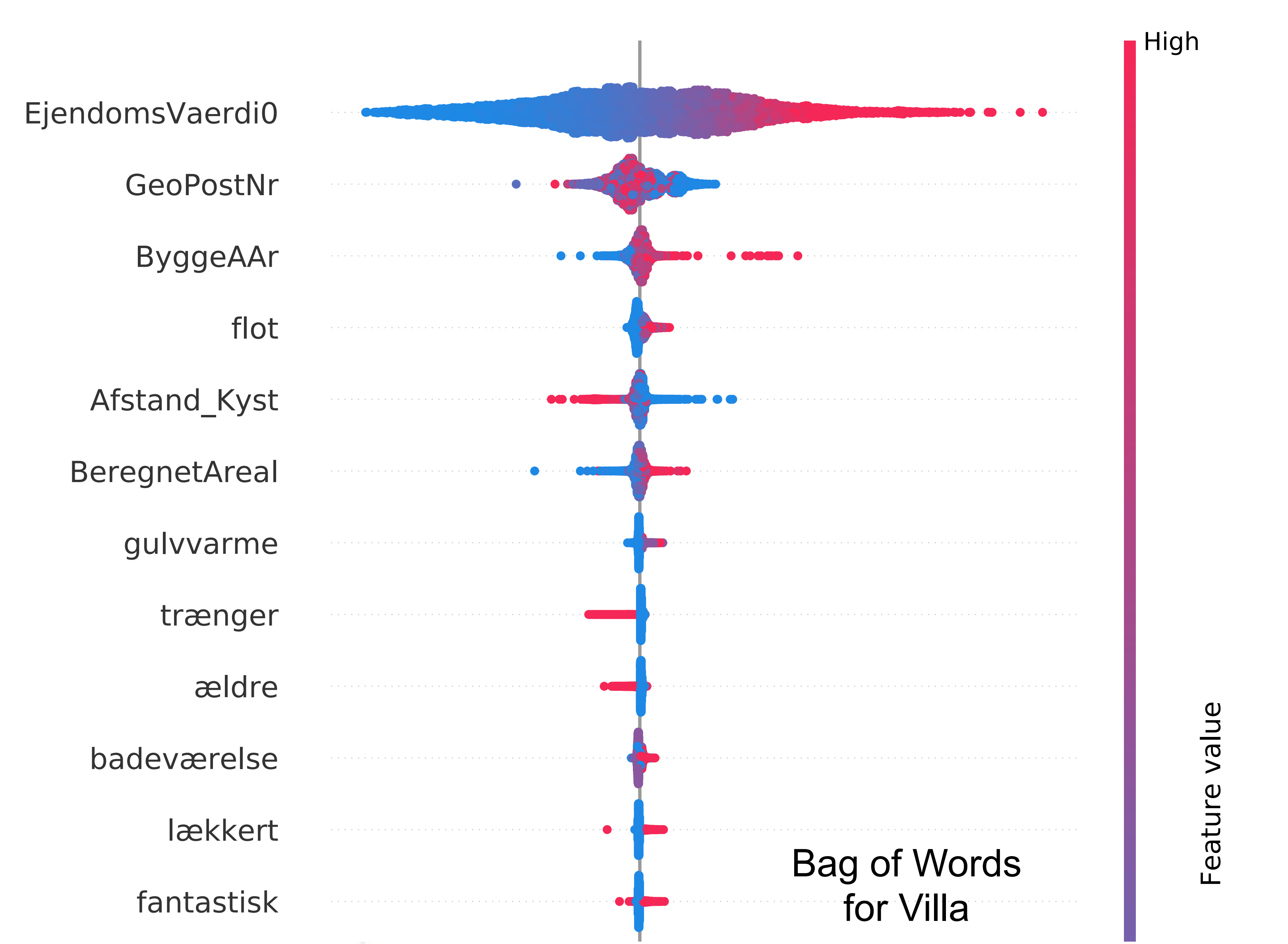
MAD(XGB, numerics only) = 0.165

MAD(XGB, text only, BOW) = 0.254

MAD(XGB, combined) = 0.147

(Numerics: GeoPostNr, BeregnetAreal, ByggeAAr, EjendomsVaerdi0, Afstand_Kyst)

Text Analysis

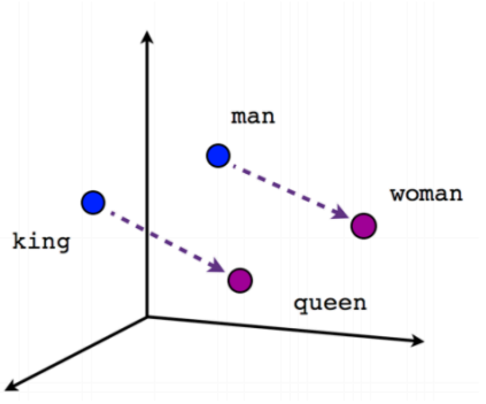


Bag of Words
for Villa

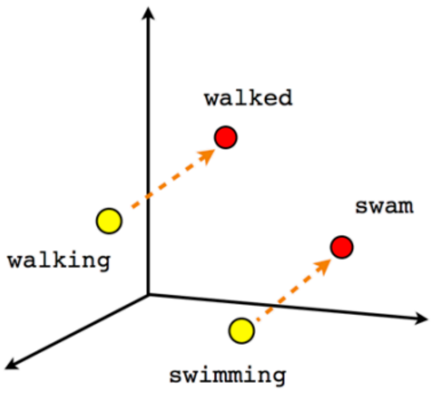
Text Analysis

More advanced methods:

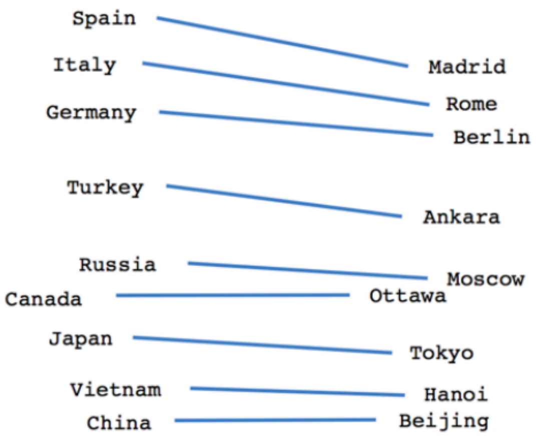
- Latent Dirichlet Allocation, *LDA*
- Word Vectors / Word Embeddings, *word2vec, GloVe, FastText*



Male-Female



Verb tense



Country-Capital