K-Means Clustering

CS 780/880 Natural Language Processing Lecture 5

Samuel Carton, University of New Hampshire

Last lecture

Key idea: Classifying text

Concepts

- Supervised learning
    - Training set
    - Development/ validation set
    - Test set
- K-nearest-neighbors
- Classification metrics
    - Accuracy
    - Precision
    - Recall
    - F1

Concepts (continued)

- Model confidence
- Hyperparameters
    - Grid search

Toolkits

- Pandas for reading and manipulating datasets
- Scikit-Learn for model building and evaluation

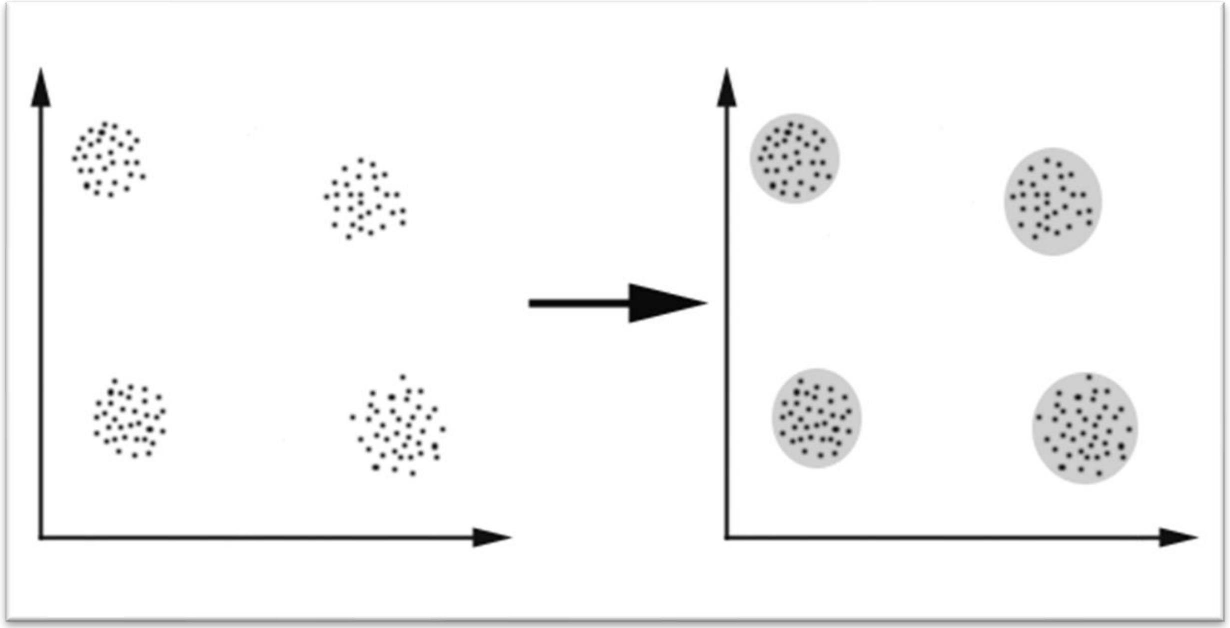
Unsupervised learning for clustering

Clustering: given input text x, cluster x into one of C clusters, along with similar texts

Unsupervised learning: given a dataset X, learn how to predict… something.

- **Clustering**: cluster labels
- **Generation**: new outputs x which belong to the same distribution as the input

Key term: **latent structure**



https://matteucci.faculty.polimi.it/Clustering/tutorial_html/index.html

Clustering for text

When would you want to do clustering on text?

Basically, any time you want to get a high-level understanding of the structure of your corpus.

What are some real-world scenarios where you might want this?

Clustering for text

When would you want to do clustering on text?

Basically, any time you want to get a high-level understanding of the structure of your corpus.

What are some real-world scenarios where you might want this?

- Computational social science/textual analysis
  - E.g. “what are the K basic types of post that exist on this subreddit?”
- Business analytics
  - E.g. “what kinds of things are people saying about my company on Twitter these days?”

Case study: 20-Newsgroups

Classic dataset for text classification and clustering

http://qwone.com/~jason/20Newsgroups/

~20,000 documents, evenly split across 20 newsgroups:

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

20-Newsgroups dataset

```
# certain classic datasets can be downloaded directly from scikit-learn
from sklearn.datasets import fetch_20newsgroups
```

```
# The examples in the 20-Newsgroup dataset are message board posts
# One convenient way of cleaning them is to remove headers, footers and quotes to
# leave just the texts
# and the scikit-learn version of the dataset has convenient functionality for this
ng_dataset = fetch_20newsgroups(remove=('headers', 'footers', 'quotes'),
                                categories=our_categories
                                )
```

```
# There are 20 different groups from which the posts were drawn
ng_dataset.target_names
```

```
['comp.windows.x',
 'misc.forsale',
 'rec.sport.baseball',
 'sci.space',
 'talk.politics.misc']
```

20-Newsgroups dataset

```
1 print(ng_dataset.data[0])
```

```
Its time for a little house cleaning after my PC upgrade.  I have the following
for sale:

Leading Technology PC partner (286) sytsem.  includes
        80286 12mhz intel cpu
        85Mb IDE drive (brand new - canabalized from new system)
        3.5 and 5.24 floppies
        1 Meg ram
        vga congroller
        kb
        5.0 dos on hard drive
need to get $300 for system

AT style kb - $20
Logitech serial trackman with latest drivers $45

Amiga 500 with 2.0 roms installed and 1Mb video ram and 4Mb addon ram
        501 clone (512K ram and clock)
        Roctec addon disk IDE disk controller includes SCSI option
```

20-Newsgroup dataset

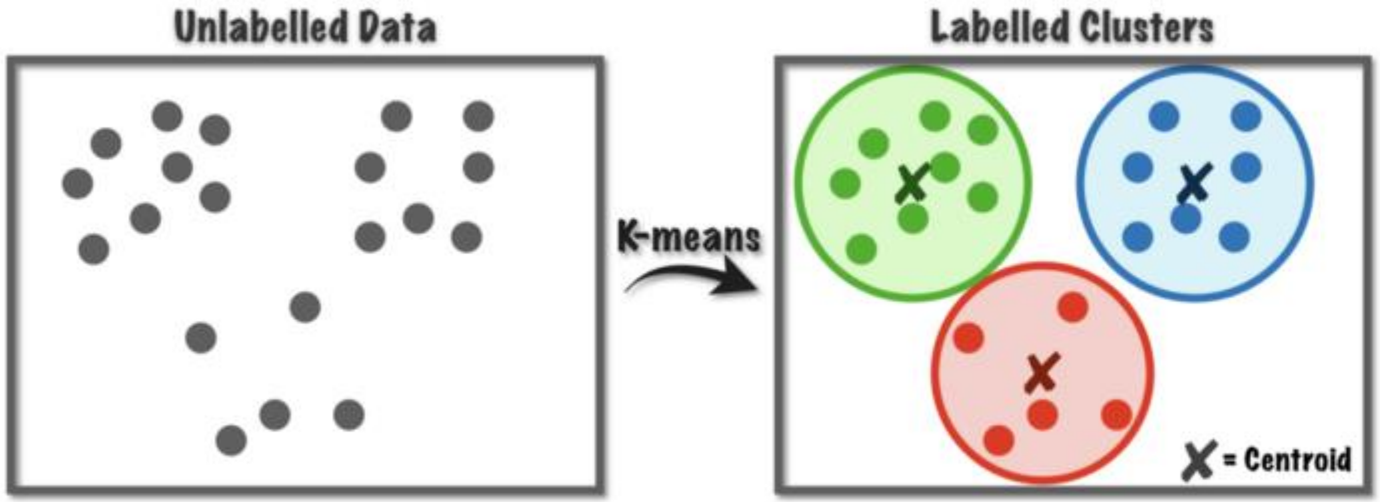
```
# Convert to a dataframe for ease of use/display
ng_df = pd.DataFrame({'text':ng_dataset.data, 'group':ng_dataset.target})
ng_df
```

	text	group
0	Its time for a little house cleaning after my ...	1
1	Bo Bilinsky?\n\n\n	2
2	I have one original SAM (Symantec AntiVirus fo...	1
3	Unless otherwise noted, I am mainly interested...	1
4	Does anyone know what is available in terms of...	0
...	...	...
2828	\nSomeone tell me there's a :-) hidden here so...	2
2829	\nYes, it is -- you could look it up. And spa...	4
2830	\n\nHe's also the one who dubbed it the SR-71 ...	3
2831	record\nhand	2
2832	...	4

2833 rows × 2 columns

K-means clustering

Basic idea: find K points (aka “means”, aka “centroids”) within the data space that represent centers of cohesive clusters



https://towardsdatascience.com/k-means-a-complete-introduction-1702af9cd8c

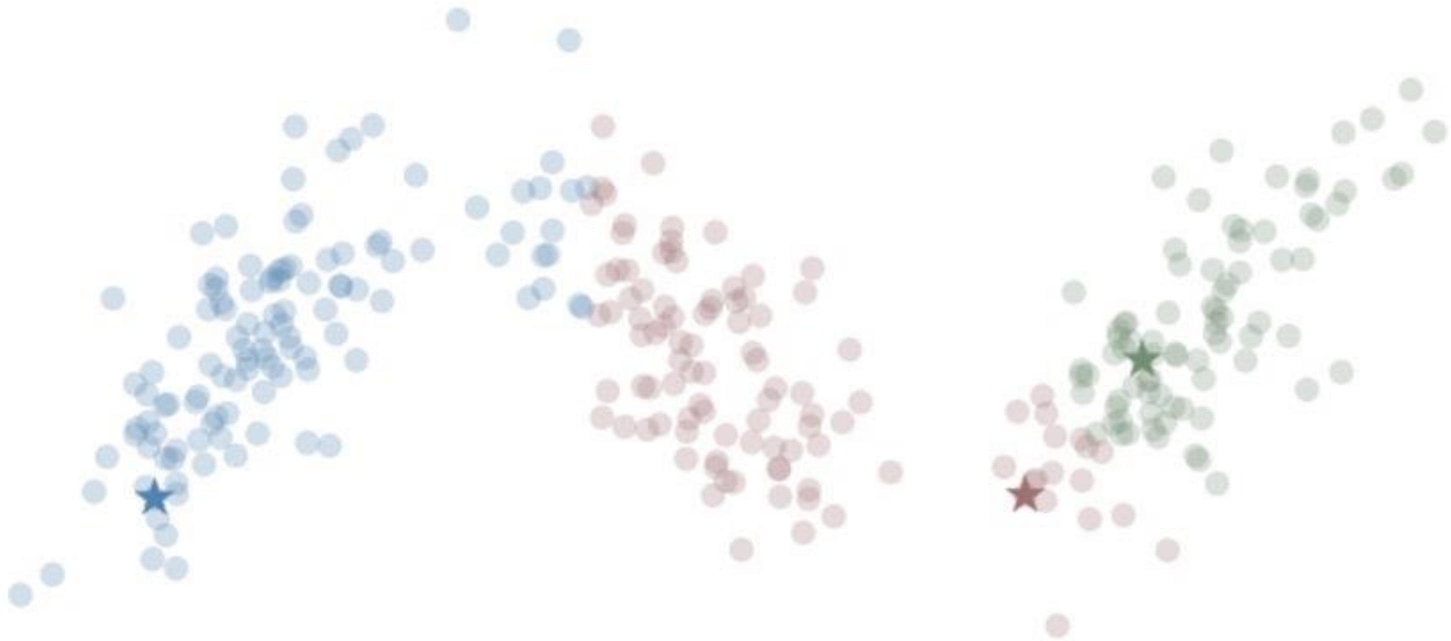
K-means clustering

Algorithm:

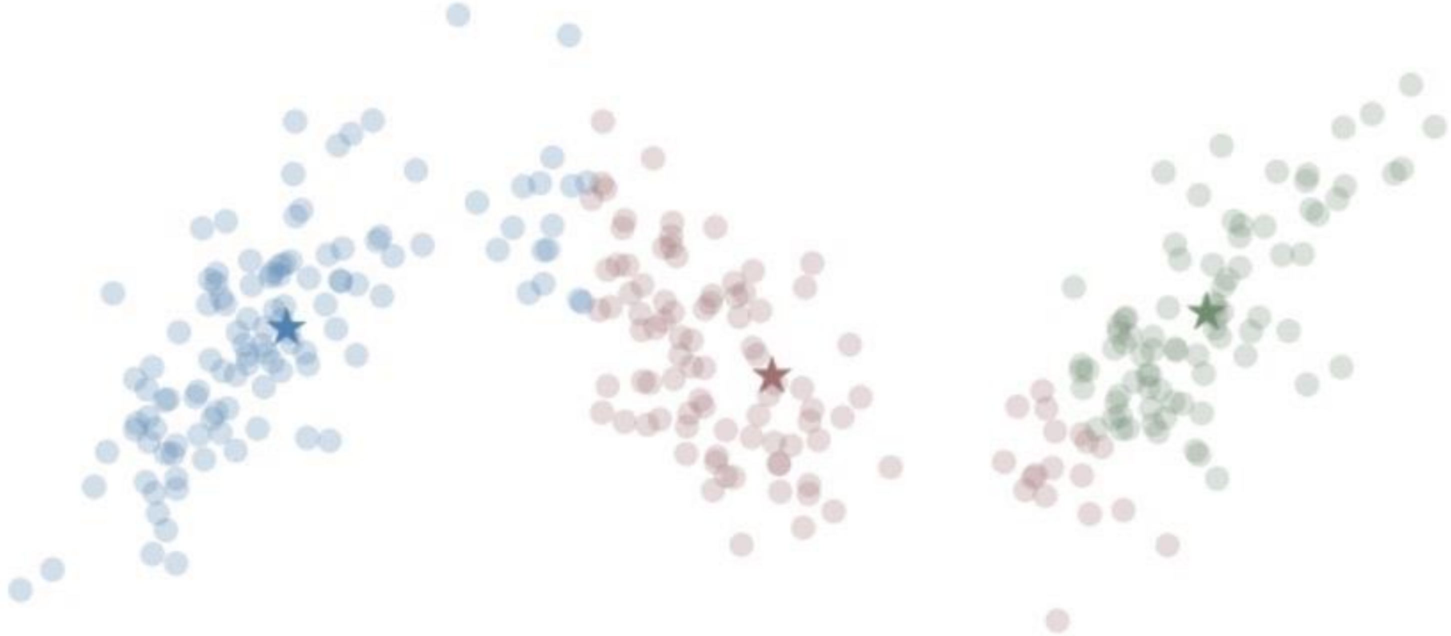
1. Randomly choose K spots within the data space to be initial cluster centers
2. For each point in the data, assign it to the cluster with the closest mean in vector space
   - Implicitly uses Euclidean distance
3. For each cluster center, adjust its position to be the centroid of the data points assigned to it
4. Repeat steps 2 and 3 until some stopping condition is hit
   - Cluster centers stop moving
   - Maximum iterations

K-Means Example

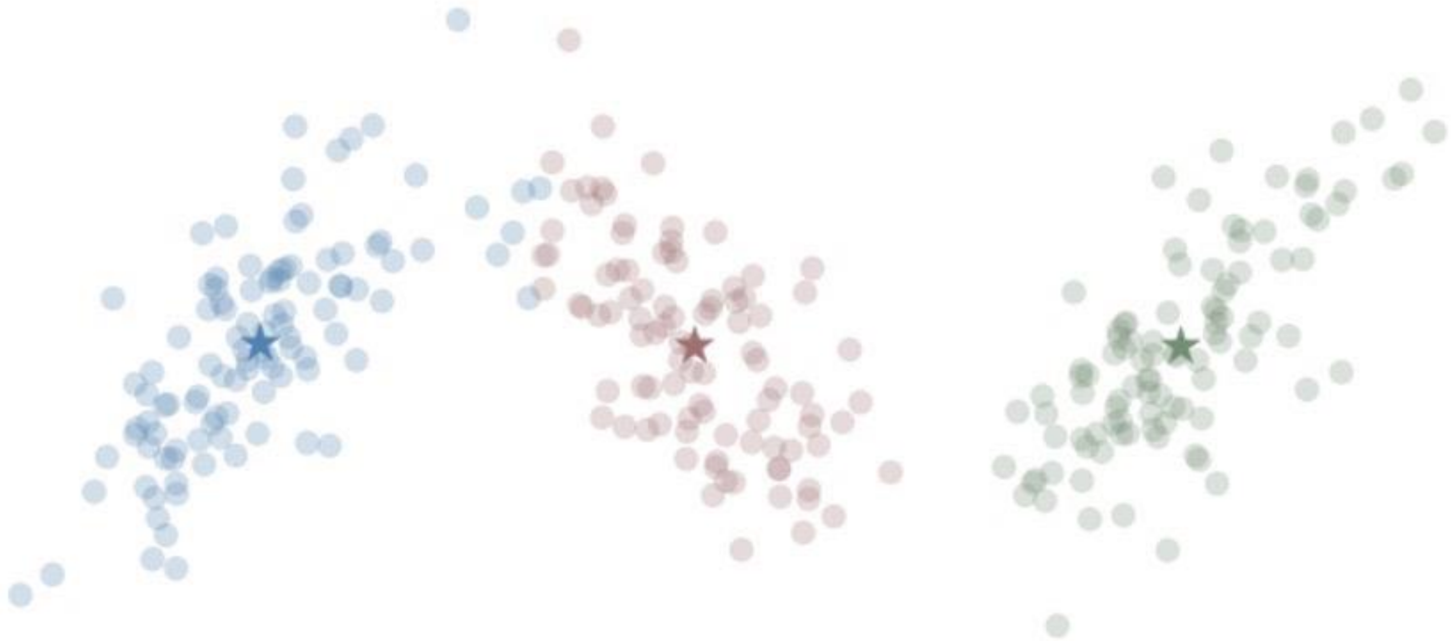
K-Means Example



K-Means Example



K-Means Example



K-Means Example

K-Means Example

K-Means Example

Strengths and weaknesses

Strengths

- Simple
- Intuitive
- Fast

Weaknesses

- Doesn’t work with categorical data
  - Use K-modes instead
- Usually only converges to local minimum
  - Use several random restarts
- Have to determine number of clusters
  - We’ll talk about this in a second
- Can be sensitive to outliers
- Only generates convex clusters

Strengths and weaknesses

Strengths

- Simple
- Intuitive
- Fast

Weaknesses

- Doesn’t work with categorical data
  - Use K-modes instead
- Usually only converges to local minimum
  - Use several random restarts
- Have to determine number of clusters
  - We’ll talk about this in a second
- **Can be sensitive to outliers**
- Only generates convex clusters

Weaknesses - Outlier Sensitivity

Weaknesses - Outlier Sensitivity

Weaknesses - Outlier Sensitivity

Weaknesses - Outlier Sensitivity

Strengths and weaknesses

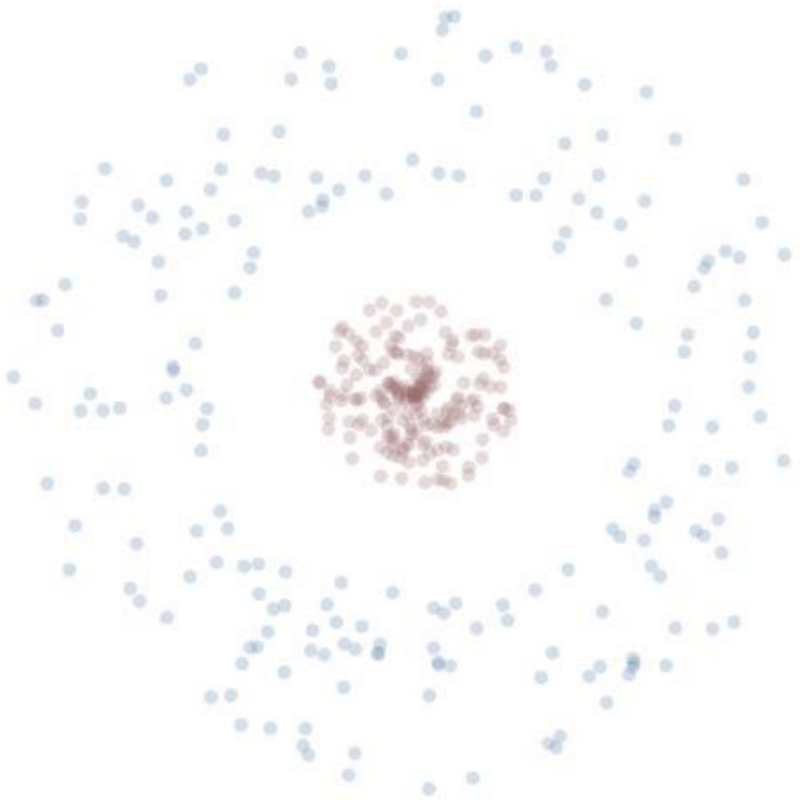
Strengths

- Simple
- Intuitive
- Fast

Weaknesses

- Doesn’t work with categorical data
  - Use K-modes instead
- Usually only converges to local minimum
  - Use several random restarts
- Have to determine number of clusters
  - We’ll talk about this in a second
- Can be sensitive to outliers
- **Only generates convex clusters**

Weaknesses - Convex Clusters



Weaknesses - Convex Clusters

Preprocessing and vectorization

```python
stemmer = PorterStemmer()
def preprocess(s):
    return ' '.join([stemmer.stem(token) for token in word_tokenize(s)])
```

```python
ng_df['preprocessed']= ng_df['text'].apply(preprocess)
```

```python
# I'm cheating a little bit here by ignoring terms that occur in fewer than 1%
# of documents or more than 25% of documents, but I was having trouble getting
# IDFs to outweight TFs and this is a hacky solution

vectorizer = TfidfVectorizer(min_df=0.01, max_df=0.25) # Our old friend TF-IDF
ng_X = vectorizer.fit_transform(ng_df['preprocessed'])
```

```python
ng_X
```

```
<2833x1431 sparse matrix of type '<class 'numpy.float64'>'
	with 136545 stored elements in Compressed Sparse Row format>
```

Building the model

```
model = KMeans(n_clusters=5, random_state=0).fit(ng_X)
```

```
# Putting cluster assignments back into the dataframe as a column so we can look at them side by side
ng_df['cluster'] = model.predict(ng_X)
ng_df
```

	text	group	preprocessed	cluster
0	Its time for a little house cleaning after my ...	1	it time for a littl hous clean after my pc upg...	2
1	Bo Bilinsky?\n\n\n	2	bo bilinski ?	0
2	I have one original SAM (Symantec AntiVirus fo...	1	i have one origin sam (symantec antiviru for ...	2
3	Unless otherwise noted, I am mainly interested...	1	unless otherwis note , i am mainli interest in...	2
4	Does anyone know what is available in terms of...	0	doe anyon know what is avail in term of autom ...	3
...	...	...	...	...
2828	\nSomeone tell me there's a :-) hidden here so...	2	someon tell me there 's a : -) hidden here so...	0
2829	\nYes, it is -- you could look it up. And spa...	4	ye , it is -- you could look it up . and spare...	0
2830	\n\nHe's also the one who dubbed it the SR-71 ...	3	he 's also the one who dub it the sr-71 - it w...	4
2831	record\nhand	2	record hand	0
2832	...	4	not at all . i am not a member of the religi l...	1

2833 rows × 4 columns

Assessing cluster quality

Extrinsic measures: We have some ground-truth clusters to compare with?

- Why can't we just use classification metrics like accuracy, F1, etc?
- **Mutual information**

Intrinsic measures: No ground-trust cluster assignments available.

- What can we even do? Can we do anything?
- **Silhouette coefficient**

https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation

Mutual Information

Basic idea: given two discrete random variables, how much does knowing the value of the one tell you about the other?

$$\mathrm{MI}(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log\left(\frac{P(i,j)}{P(i)P'(j)}\right)$$

https://scikit-learn.org/stable/modules/clustering.html#mutual-info-score

Actually a measure of pairwise **entropy**

Entropy

A measure of **uncertainty** in a discrete probability distribution

$$H(U) = -\sum_{i=1}^{|U|} P(i) \log(P(i))$$

https://scikit-learn.org/stable/modules/clustering.html#mutual-info-score

High value if distribution is spread out over possible outcomes, low value if it is concentrated in one outcome

Example: think about a fair coin $p_{fair} = (0.5{,}0.5)$ versus a trick coin$p_{trick} = (0.9{,}0.1)$

- H(fair) = 0.5*log(0.5)) + 0.5*log(0.5) = -.347 + -.347 = -.693
- H(trick) = 0.9*log(0.9)) + 0.1*log(0.1) = -.094 + -.230 = -.325

Mutual Information

Generalizes entropy to joint distribution of two variables

High value if joint probability is concentrated in one pair of outcomes, low value if it is spread out across pairs

Only defined if we have another variable to compare our clusters to (i.e., when we have ground-truth labels available)

$$\mathrm{MI}(U,V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} P(i,j) \log\left(\frac{P(i,j)}{P(i)P'(j)}\right)$$

https://scikit-learn.org/stable/modules/clustering.html#mutual-info-score

Normalized mutual information

Normalized mutual information normalizes mutual information to fall between 0 (maximum possible pairwise entropy) and 1 (minimum possible pairwise entropy)

Preferable over un-normalized because it can be interpreted visually

$$\mathrm{NMI}(U,V) = \frac{\mathrm{MI}(U,V)}{\mathrm{mean}(H(U), H(V))}$$

https://scikit-learn.org/stable/modules/clustering.html#mutual-info-score

Toy example

Define a few toy cluster assignments

```
v0 = [1,0,0,1]

v1 = [0,1,1,0]

v2 = [0,0,1,1]

v3 = [1,1,0,1]

```

```
# We can see these facts with contingency tables
from sklearn.metrics.cluster import contingency_matrix

print('Contingency matrix between v0 and v1:\n',contingency_matrix(v0, v1))

print('\nContingency matrix between v0 and v2:\n',contingency_matrix(v0,v2))

print('\nContingency matrix between v0 and v3:\n',contingency_matrix(v0,v3))
```

```
Contingency matrix between v0 and v1:
 [[0 2]
 [2 0]]

Contingency matrix between v0 and v2:
 [[1 1]
 [1 1]]

Contingency matrix between v0 and v3:
 [[1 1]
 [0 2]]
```

Toy example

```python
from sklearn.metrics import mutual_info_score, normalized_mutual_info_score
```

```python
#Mutual information captures this notion of discrete correlation

print('Mutual information between v0 and v1:', mutual_info_score(v0, v1))

print('Mutual information between v0 and v2:', mutual_info_score(v0, v2))

print('Mutual information between v0 and v3:', mutual_info_score(v0, v3))


# But due to the nature of how it is calculated, MI is not bounded in a way that
# makes it easy to read or interpret
```

```
Mutual information between v0 and v1: 0.6931471805599453
Mutual information between v0 and v2: 0.0
Mutual information between v0 and v3: 0.21576155433883565
```

Toy example

```python
# Normalized mutual information normalizes MI by its maximum possible value
# given the dimensionality of the dimension, resulting in a [0,1] range

print('Normalized mutual information between v0 and v1:', normalized_mutual_info_score(v0, v1))

print('Normalized mutual information between v0 and v2:', normalized_mutual_info_score(v0, v2))

print('Normalized mutual information between v0 and v3:', normalized_mutual_info_score(v0, v3))

```

```
Normalized mutual information between v0 and v1: 1.0
Normalized mutual information between v0 and v2: 0.0
Normalized mutual information between v0 and v3: 0.3437110184854508
```

Our clustering

```
normalized_mutual_info_score(ng_df['group'], ng_df['cluster'])
```

```
0.43134155661477763
```

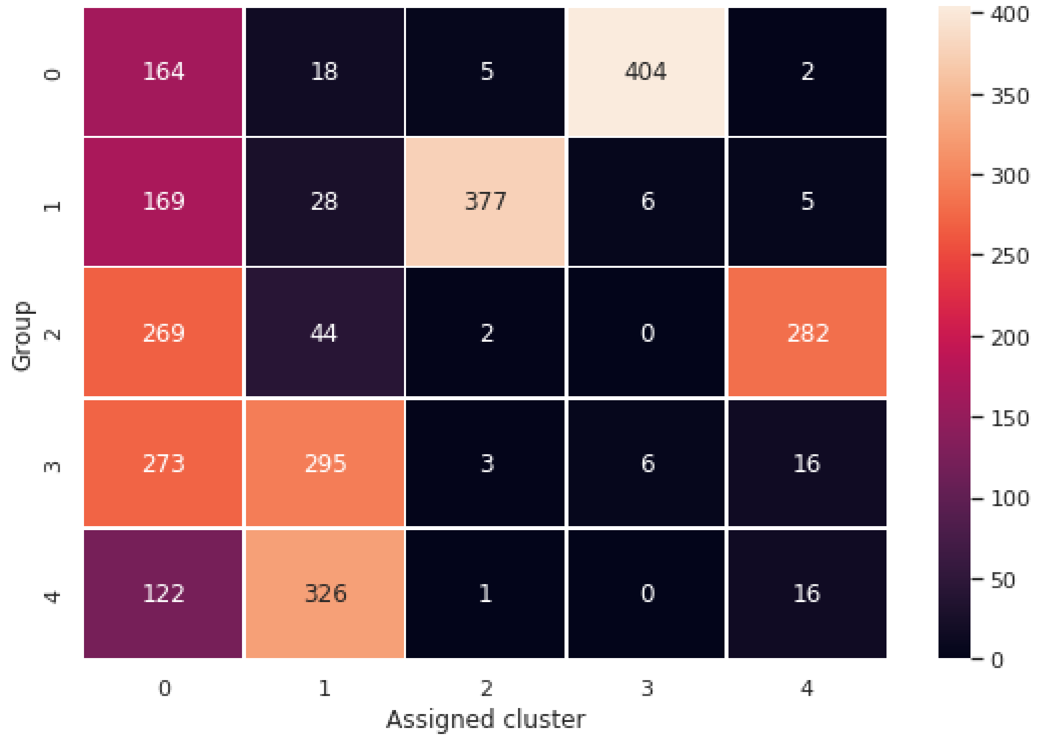
```
# Not great, not terrible. Let's take a look at the contingency matrix
cm = contingency_matrix(ng_df['group']*10, ng_df['cluster'])
print(cm)
```

```
[[164  18   5 404   2]
 [169  28 377   6   5]
 [269  44   2   0 282]
 [273 295   3   6  16]
 [122 326   1   0  16]]
```

Our clustering

```python
# If we don't want to stare at a bunch of numbers like nerds,
# we can create a heatmap

import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme()


# Draw a heatmap with the numeric values in each cell
f, ax = plt.subplots(figsize=(9, 6))
sns.heatmap(cm, annot=True, fmt="d", linewidths=.5, ax=ax)

plt.xlabel('Assigned cluster')
plt.ylabel('Group')

```

```
Text(57.5, 0.5, 'Group')
```



Silhouette coefficient

Intrinsic measure: doesn't require any ground-truth clusters

Basic idea: measures the extent to which data points are **close** to points in the same cluster and **far away** from points in other clusters

- Rewards tight, well-separated clusters

Formula:

$$s = \frac{b - a}{max(a, b)}$$

- **a**: The mean distance between a sample and all other points in the same class.
- **b**: The mean distance between a sample and all other points in the *next nearest cluster*.

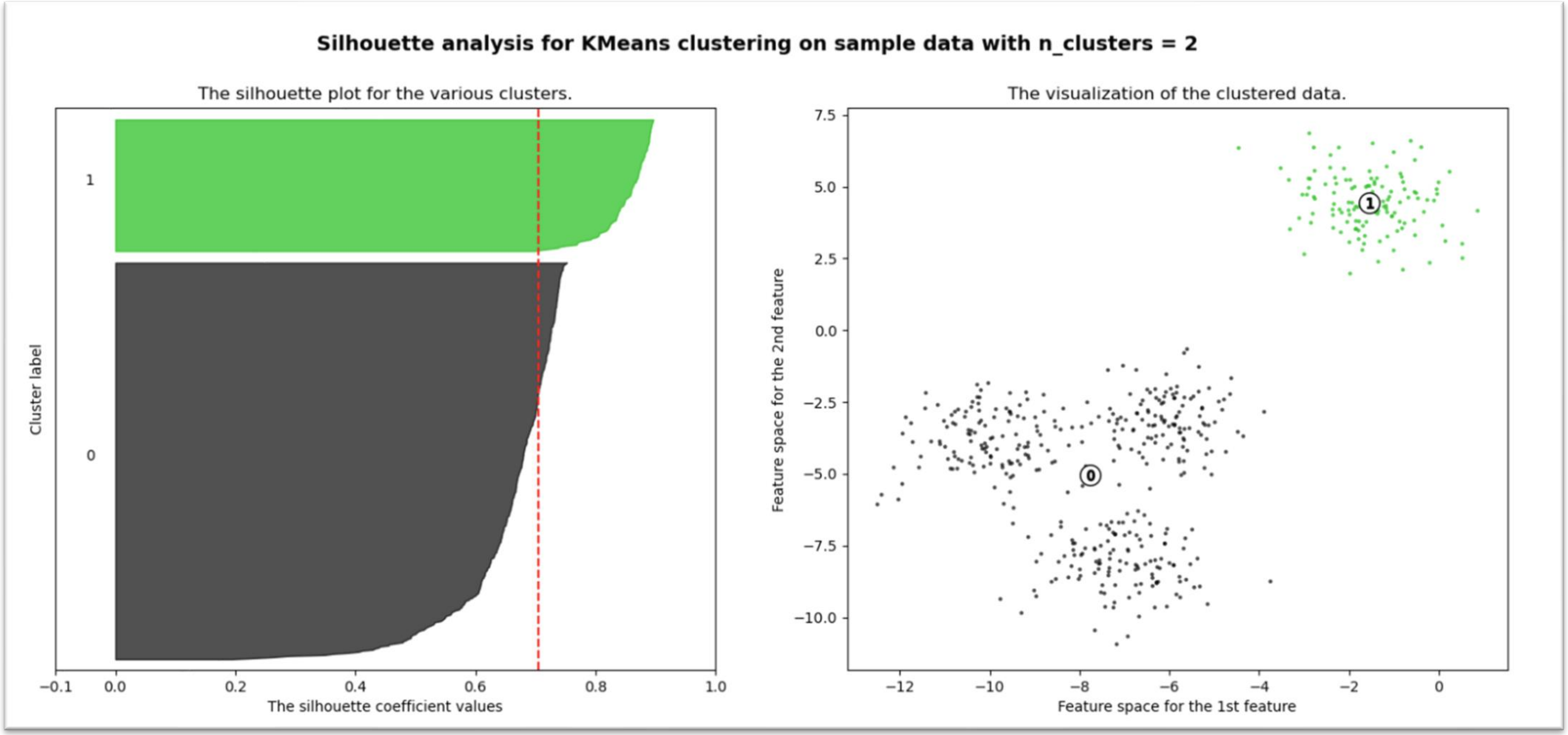
Can be defined for a single sample, or averaged over entire cluster(or entire dataset)

Silhouette analysis to choose K

Basic idea: Visualize silhouette values for each cluster, and choose K such that as many clusters as possible have as many points as possible with high silhouette coefficients.

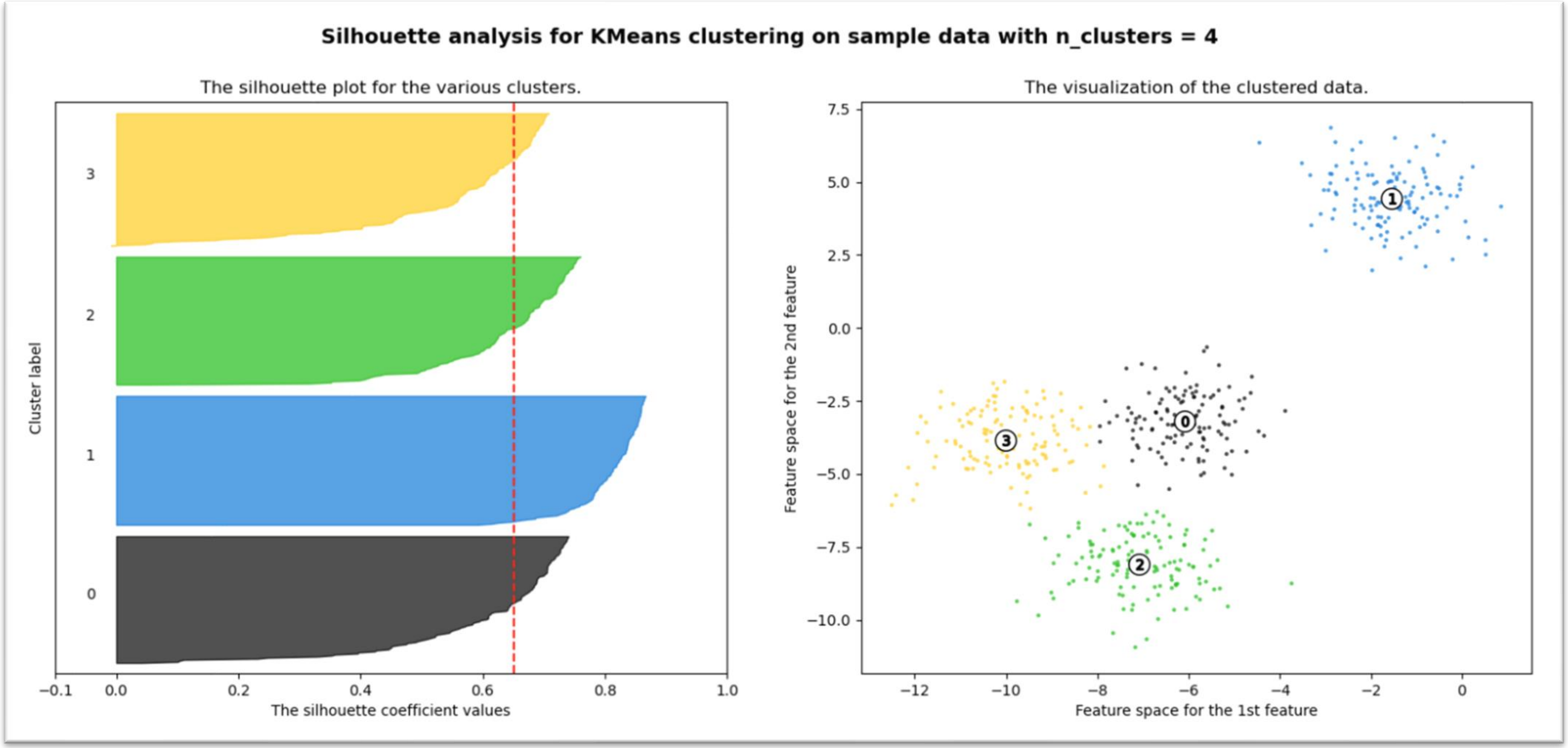
Still a lot of eyeballing involved

Silhouette analysis to choose K



https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Silhouette analysis to choose K



https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Silhouette coefficient

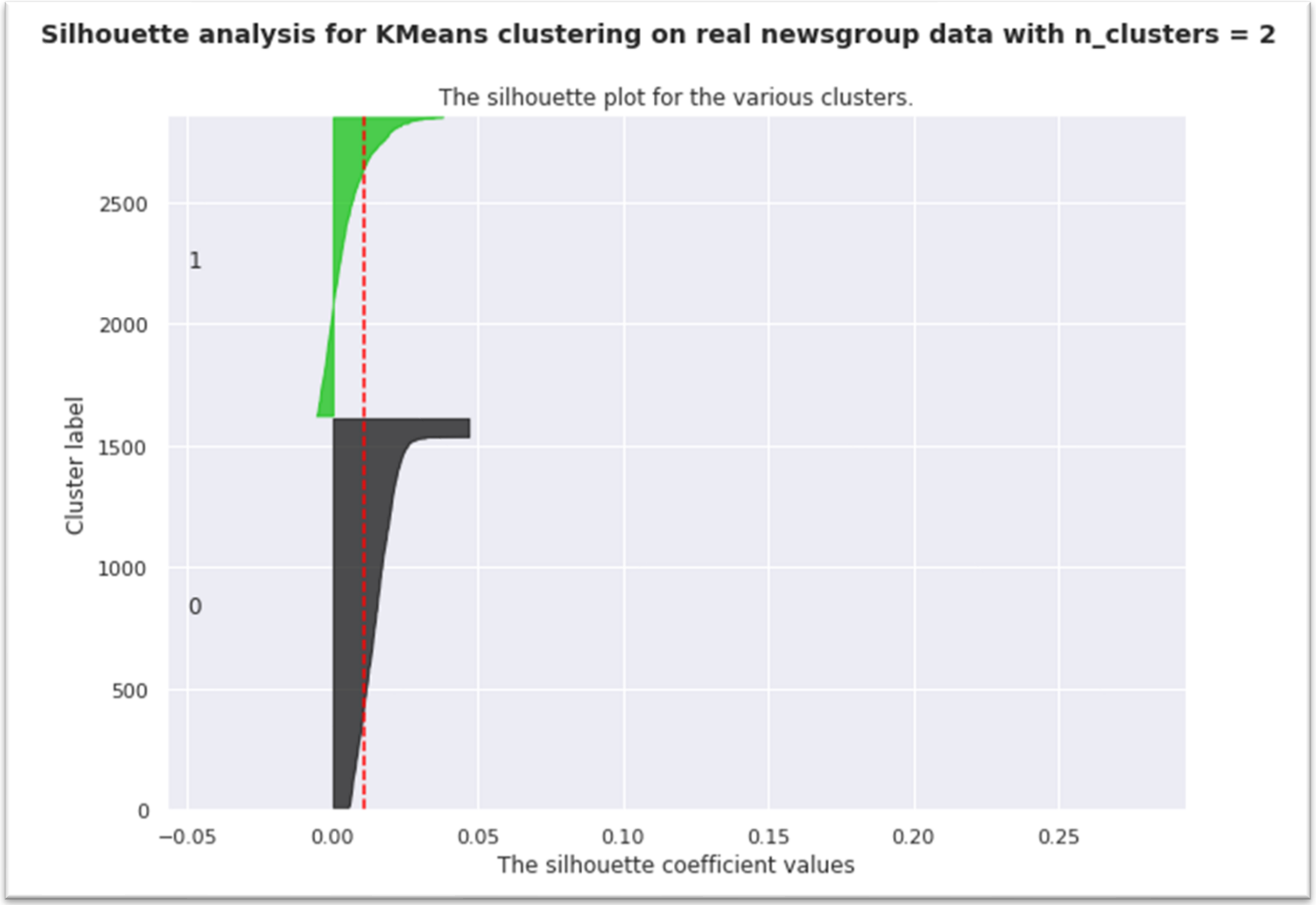
```python
from sklearn.metrics import silhouette_score

# The silhouette coefficient measures how close the items in a cluster are to each
# other relative to how close they are to items in the next closest cluster.

# It rewards tight, well-separated clusters (as most intrinsic metrics do)
# It can be measured for a single cluster, or for a whole dataset
```

```python
silhouette_score(ng_X,
                 ng_df['cluster'].to_numpy().ravel(),
                 metric='euclidean')
```

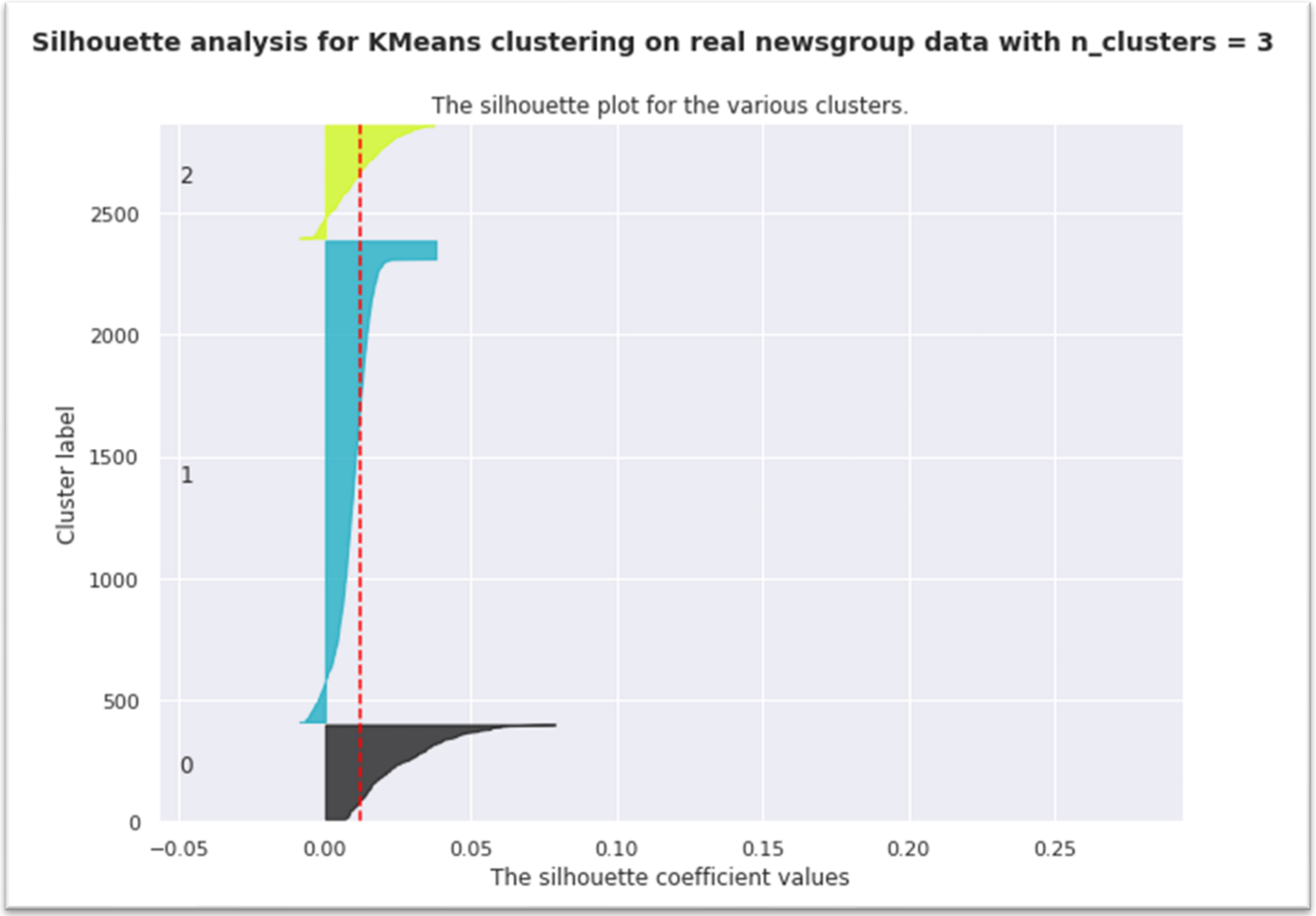
```
0.010642877770122048
```

```python
# That seems pretty low... but how to interpret?
```

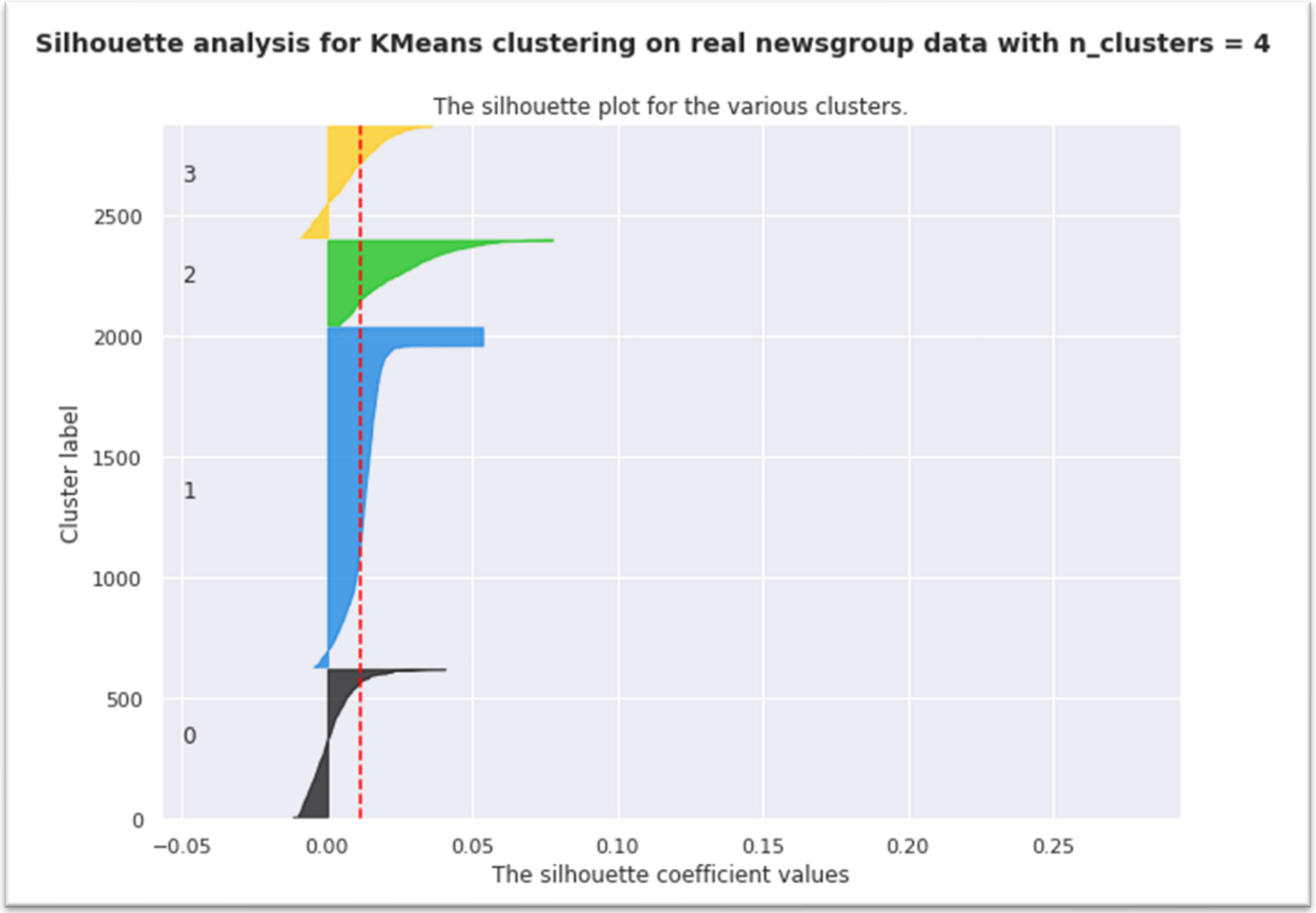
Silhouette analysis



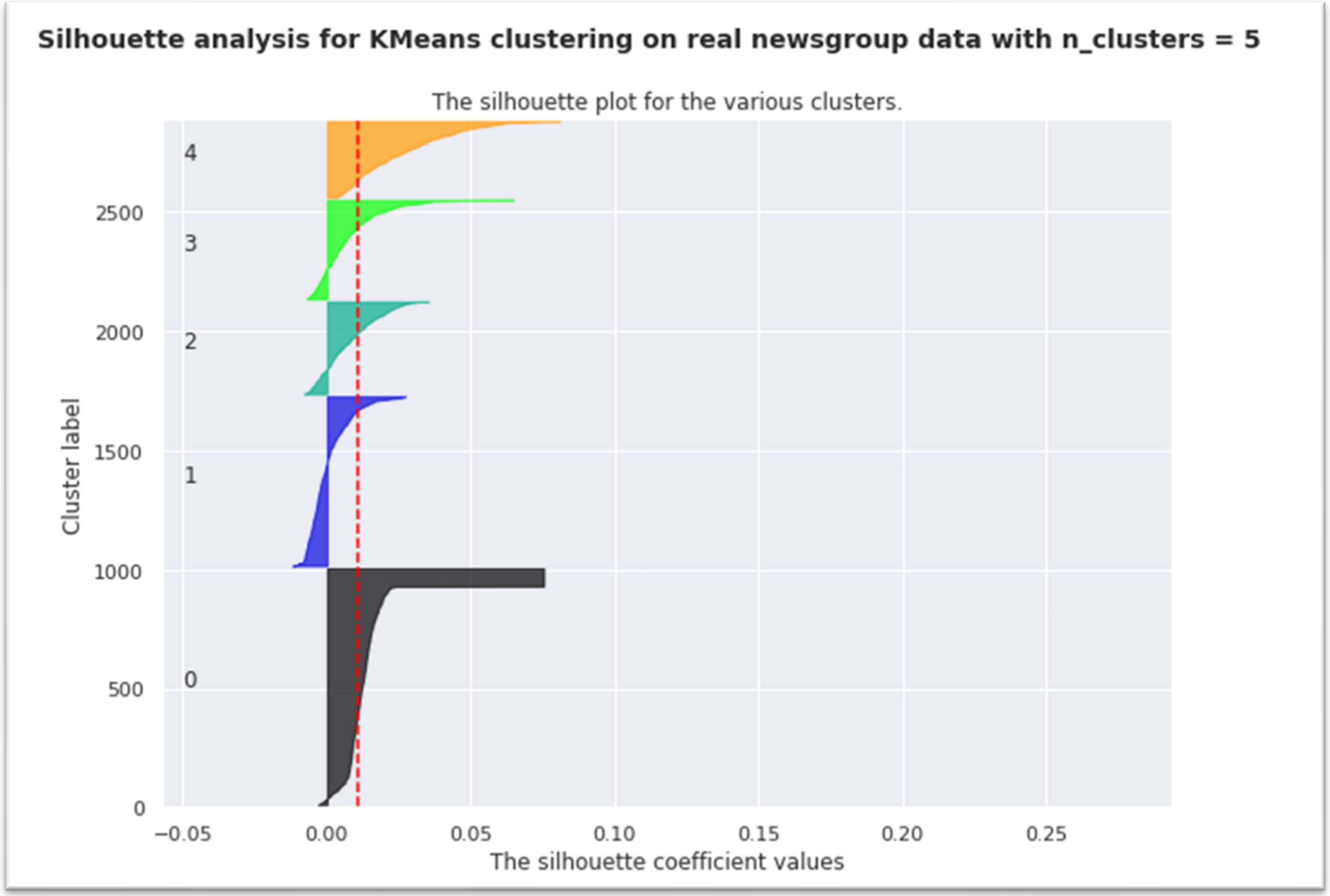
Silhouette analysis



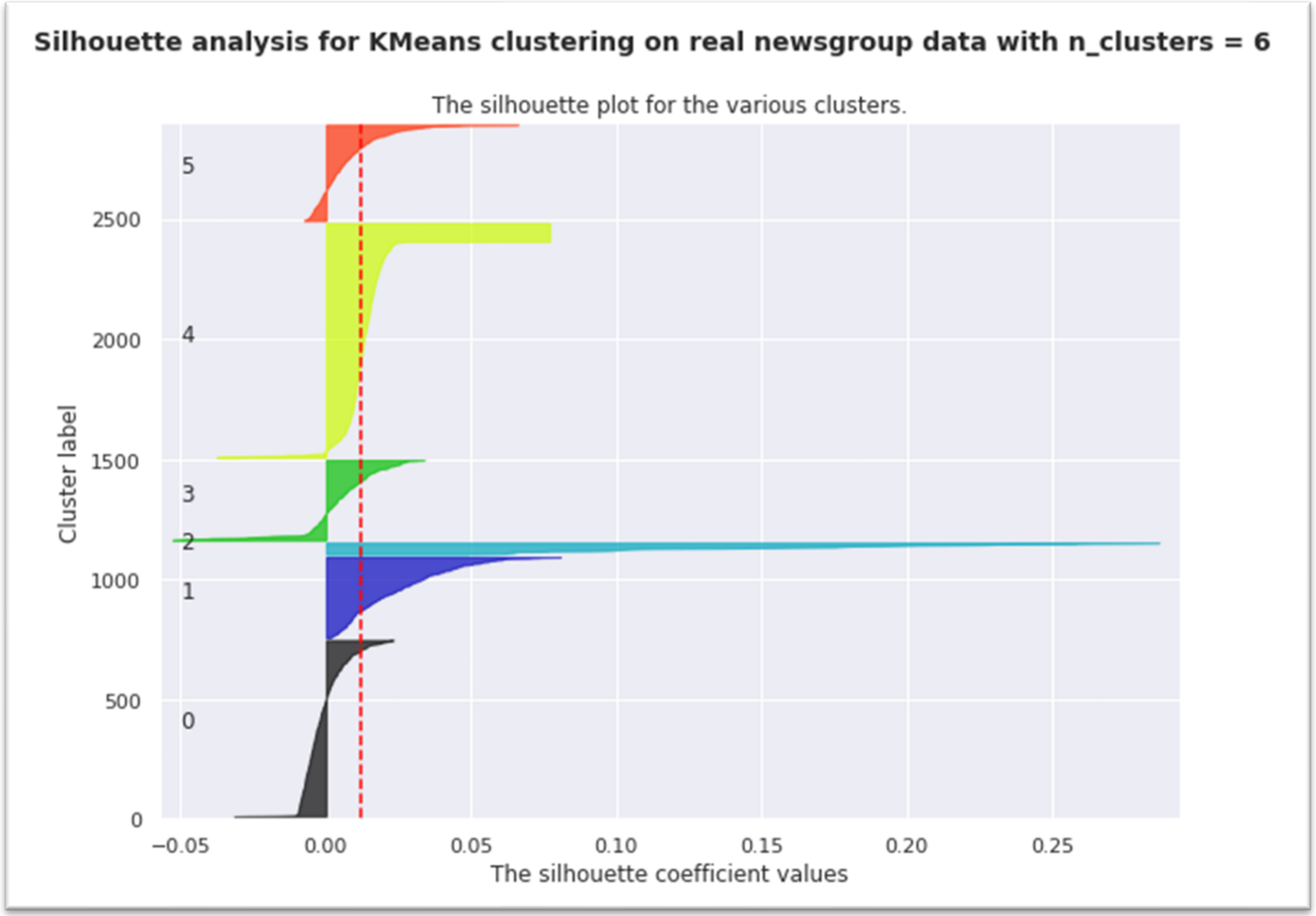
Silhouette analysis



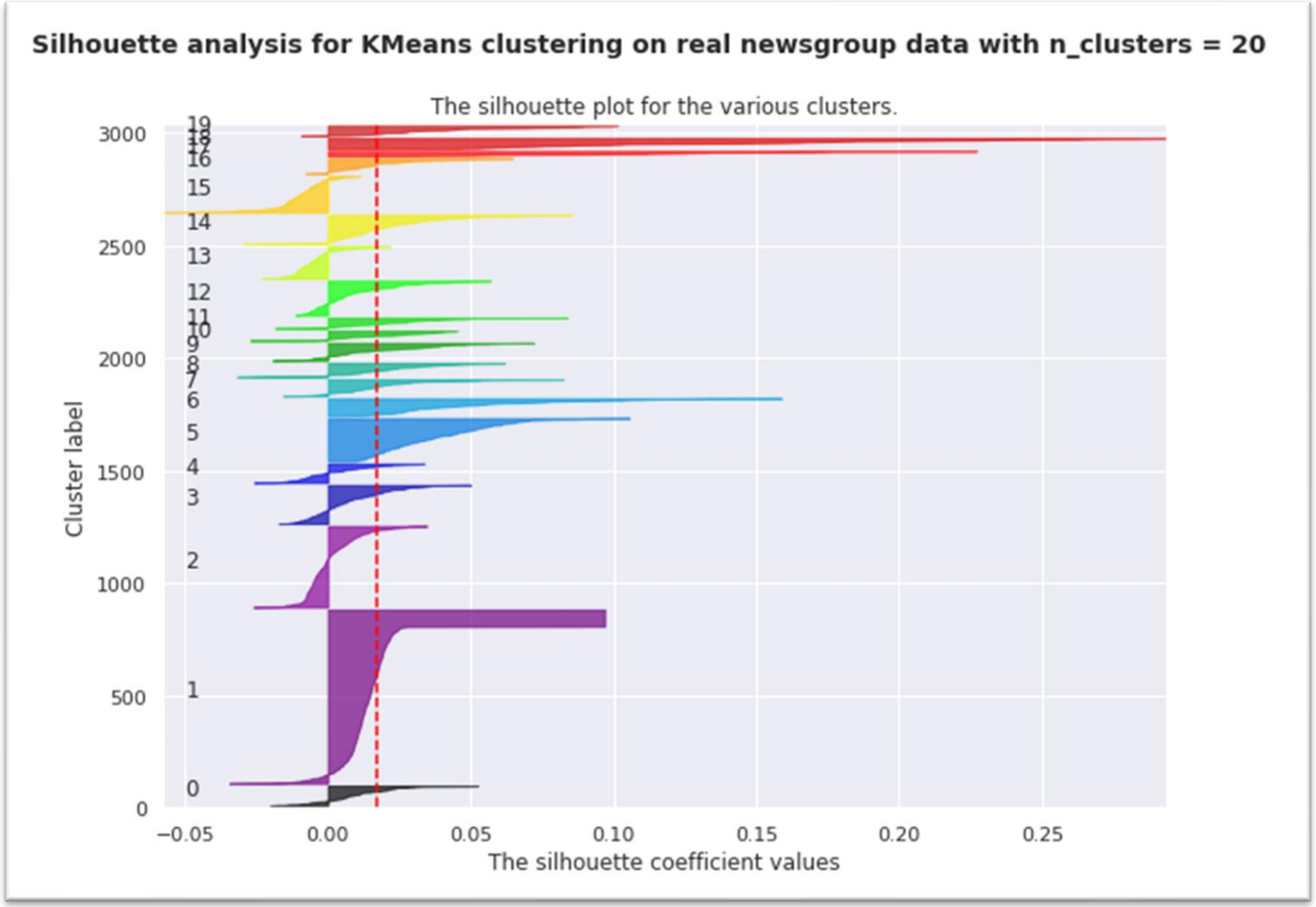
Silhouette analysis



Silhouette analysis



Silhouette analysis



Representing individual text clusters

If we want to represent individual text clusters, how should we do it?

Just pick the top N most frequent words?

- Likely to be stop-words or otherwise uninteresting

TF-IDF to the rescue!

- Treat entire cluster as document, find words that occur frequently in that cluster relative to the corpus as a whole

Displaying clusters

```python
# A simple thing we can do when we want to display the top terms associated with
# a cluster is to treat the whole thing as a document, run it through the vectorizer,
# and choose the top-weighted terms as the most representative

def top_cluster_terms(texts, clusters, cluster, vectorizer, n_words=5, verbose=True):

  text_subset = texts[clusters==cluster]
  combined_text = ' '.join(text_subset) # very inefficient but I don't have time to optimize
  combined_vector = np.array(vectorizer.transform([combined_text]).todense())[0]
  sorted_indices = np.argsort(combined_vector)
  top_indices = sorted_indices[-n_words:]
  vocab = vectorizer.get_feature_names_out()
  top_words = [vocab[index] for index in top_indices]
  top_weights = combined_vector[top_indices]

  if verbose:
    for index in range(n_words-1, -1, -1):
      print(f'\tWord: {top_words[index]} - weight: {top_weights[index]:.3f}')

  return top_words, top_weights
```

Displaying clusters

Top terms associated with the actual groups

```
Top words for group "comp.windows.x"
        Word: file - weight: 0.250
        Word: entri - weight: 0.212
        Word: window - weight: 0.205
        Word: widget - weight: 0.177
        Word: program - weight: 0.163
Top words for group "misc.forsale"
        Word: 00 - weight: 0.450
        Word: 50 - weight: 0.156
        Word: offer - weight: 0.153
        Word: sale - weight: 0.150
        Word: new - weight: 0.138
Top words for group "rec.sport.baseball"
        Word: he - weight: 0.339
        Word: game - weight: 0.186
        Word: year - weight: 0.176
        Word: team - weight: 0.161
        Word: hi - weight: 0.155
Top words for group "sci.space"
        Word: space - weight: 0.342
        Word: launch - weight: 0.185
        Word: orbit - weight: 0.182
        Word: nasa - weight: 0.168
        Word: satellit - weight: 0.153
Top words for group "talk.politics.misc"
        Word: we - weight: 0.272
        Word: mr - weight: 0.196
        Word: presid - weight: 0.186
        Word: he - weight: 0.147
        Word: peopl - weight: 0.143
```

Top terms associated with the clusters we found

```
Top words for cluster 0
        Word: by - weight: 0.135
        Word: 04 - weight: 0.121
        Word: me - weight: 0.116
        Word: mail - weight: 0.114
        Word: edu - weight: 0.111
Top words for cluster 1
        Word: we - weight: 0.227
        Word: space - weight: 0.176
        Word: by - weight: 0.133
        Word: peopl - weight: 0.129
        Word: would - weight: 0.129
Top words for cluster 2
        Word: 00 - weight: 0.598
        Word: 50 - weight: 0.163
        Word: offer - weight: 0.146
        Word: sale - weight: 0.142
        Word: includ - weight: 0.132
Top words for cluster 3
        Word: file - weight: 0.257
        Word: entri - weight: 0.220
        Word: window - weight: 0.215
        Word: widget - weight: 0.183
        Word: program - weight: 0.170
Top words for cluster 4
        Word: he - weight: 0.462
        Word: hi - weight: 0.179
        Word: mr - weight: 0.169
        Word: year - weight: 0.155
        Word: game - weight: 0.147
```

Concluding thoughts

Lots of clustering algorithms out there:
https://scikit-learn.org/stable/modules/clustering.html#clustering

Some methods pick K, others require it as a hyperparameter

Always hard to tell when you have “good” clusters

Combines well with dimension reduction, which we’ll talk about next class

- Especially for visualization purposes