**Лабораторная работа 12.**

**Коэффициент силуэта. Практика**

В задачах используется датасет из прошлого модуля:

import numpy as np

import matplotlib.pyplot as plt

from sklearn import datasets

n\_samples = 1500

dataset = datasets.make\_blobs(n\_samples=n\_samples, centers=2, center\_box=(-7.0, 7.5),

cluster\_std=[1.4, 1.7],

random\_state=42)

X\_2, \_ = datasets.make\_blobs(n\_samples=n\_samples, random\_state=170, centers=[[-4, -3]], cluster\_std=[1.9])

transformation = [[1.2, -0.8], [-0.4, 1.7]]

X\_2 = np.dot(X\_2, transformation)

X, y = np.concatenate((dataset[0], X\_2)), np.concatenate((dataset[1], np.array([2] \* len(X\_2))))

**Коэффициент силуэта** можно посчитать при помощи реализации из библиотеки **sklearn**:

**from** sklearn.cluster **import** KMeans

**from** sklearn.metrics **import** silhouette\_score

# сначала получим предсказанные кластеры при помощи метода кластеризации

kmeans = KMeans(n\_clusters=3, random\_state=42)

kmeans.fit(X)

kmeans\_pred = kmeans.labels\_

# теперь посчитаем коэффициент силуэта

silhouette\_score(X=X, labels=kmeans\_pred, metric='euclidean')

В качестве параметров в**функции silhouette\_score** используются:

* + **X** — массив признаков объектов выборки или массив попарных расстояний между объектами;
  + **Y** — массив предсказанных кластеров для объектов выборки;
  + **metric** — метрика, используемая для вычисления расстояния между объектами, мы будем использовать **euclidean** (Евклидово расстояние), полный список названий метрик можно увидеть [здесь](https://scikit-learn.org/0.19/modules/generated/sklearn.metrics.pairwise.pairwise_distances.html).

Полное описание функции можно прочитать в [документации](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html) (англ.).

При решении заданий модуля стандартизация должна быть выключена.

**Задание 7.11.1**

Обучите **модель GaussianMixture** с параметрами n\_components=3 и random\_state=42 на признаках исходного датасета. Посчитайте коэффициент силуэта для получившейся кластеризации.

**Задание 7.11.2**

Сравните результаты кластеризации четырёх рассмотренных алгоритмов на исходном датасете при помощи коэффициента силуэта, инициализируйте алгоритмы со следующими параметрами:

* + - * K-means — n\_clusters=3, random\_state=42
      * EM-алгоритм (GaussianMixture) — n\_components=3, random\_state=42
      * Агломеративная кластеризация – n\_clusters=3
      * DBSCAN – eps=0.9, min\_samples=35

Укажите максимальное значение коэффициента силуэта, полученное при помощи данных моделей.

**Задание 7.11.3**

Подберите оптимальное количество кластеров с помощью **коэффициента силуэта**. Для этого найдите такое число кластеров, при котором значение коэффициента будет максимальным.

В трёх из рассмотренных нами алгоритмов необходимо задать число кластеров при инициализации: K-means, EM-алгоритм и агломеративная кластеризация.

Найдите значение коэффициента силуэта для данных алгоритмов при числе кластеров от 2 до 10 включительно. Для K-means и EM-алгоритма установите значение **random\_state=42**.

В качестве ответа через пробел введите **число кластеров**, при котором значение коэффициента силуэта для результатов кластеризации было наибольшим для каждого из алгоритмов. Вводите в следующем порядке: K-means, EM-алгоритм, агломеративная кластеризация.

**Однородность. Практика**

**Однородность** можно посчитать при помощи реализации из библиотеки **sklearn**:

**from** sklearn.cluster **import** KMeans

**from** sklearn.metrics.cluster **import** homogeneity\_score

**from** sklearn.preprocessing **import** StandardScaler

# сначала получим предсказанные кластеры при помощи метода кластеризации

kmeans = KMeans(n\_clusters=3, random\_state=42)

X = StandardScaler().fit\_transform(X)

kmeans.fit(X)kmeans\_pred = kmeans.labels\_ # теперь посчитаем однородностьhomogeneity\_score(labels\_true=y, labels\_pred=kmeans\_pred)

Полное описание функции можно прочитать в [документации](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.homogeneity_score.html) (англ.).

**Задание 7.12.1**

Сравните результаты кластеризации алгоритмов k-means, GaussianMixture, AgglomerativeClustering и DBSCAN на исходном датасете при помощи однородности, инициализируйте алгоритмы со следующими параметрами:

* + - * *k-means* — n\_clusters=3, random\_state=42
      * *GaussianMixture* — n\_components=3, random\_state=42
      * *AgglomerativeClustering* — n\_clusters=3
      * *DBSCAN* — eps=0.9, min\_samples=35

В качестве ответа укажите **максимальное значение однородности**, полученное при помощи данных моделей.

При решении задания модуля стандартизация должна быть включена.

**Полнота. Практика**

**Полноту** можно посчитать при помощи реализации из библиотеки **sklearn**:

**from** sklearn.cluster **import** KMeans

**from** sklearn.metrics.cluster **import** completeness\_score

# сначала получим предсказанные кластеры при помощи метода кластеризации

kmeans = KMeans(n\_clusters=3, random\_state=42)

kmeans.fit(X)

kmeans\_pred = kmeans.labels\_

# теперь посчитаем полноту

completeness\_score(labels\_true=y, labels\_pred=kmeans\_pred)

Полное описание функции можно прочитать в [документации](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.completeness_score.html) (англ.).

**Задание 7.13.1**

Обучите **модель GaussianMixture** с параметрами n\_components=3 и random\_state=42 на признаках исходного датасета. Посчитайте **полноту** для получившейся кластеризации.

Подсказка: При решении задания модуля стандартизация должна быть включена.

**Задание 7.13.2**

Сравните результаты кластеризации алгоритмов k-means, GaussianMixture, AgglomerativeClustering и DBSCAN на исходном датасете при помощи полноты, инициализируйте алгоритмы со следующими параметрами:

* + - * *k-means* — n\_clusters=3, random\_state=42
      * *GaussianMixture* — n\_components=3, random\_state=42
      * *AgglomerativeClustering* — n\_clusters=3
      * *DBSCAN* — eps=0.9, min\_samples=35

В качестве ответа укажите **максимальное значение полноты**, полученное при помощи данных моделей.

Подсказка: При решении задания модуля стандартизация должна быть включена.

**V-мера. Практика**

**V-меру** можно посчитать при помощи реализации из библиотеки **sklearn**:

**from** sklearn.cluster **import** KMeans

**from** sklearn.metrics.cluster **import** v\_measure\_score

# сначала получим предсказанные кластеры при помощи метода кластеризации

kmeans = KMeans(n\_clusters=3, random\_state=42)

kmeans.fit(X)

kmeans\_pred = kmeans.labels\_

# теперь посчитаем полноту

v\_measure\_score(labels\_true=y, labels\_pred=kmeans\_pred)

Полное описание функции можно прочитать в [документации](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_measure_score.html) (англ.).

**Задание 7.14.1**

Обучите **модель GaussianMixture** с параметрами n\_components=3 и random\_state=42 на признаках исходного датасета. Посчитайте **v-меру** для получившейся кластеризации.

Подсказка: При решении задания модуля стандартизация должна быть включена.

**Задание 7.14.2**

Сравните результаты кластеризации алгоритмов k-means, GaussianMixture, AgglomerativeClustering и DBSCAN на исходном датасете при помощи v-меры, инициализируйте алгоритмы со следующими параметрами:

* + - * *k-means* — n\_clusters=3, random\_state=42
      * *GaussianMixture* — n\_components=3, random\_state=42
      * *AgglomerativeClustering* — n\_clusters=3
      * *DBSCAN* — eps=0.9, min\_samples=35

В качестве ответа укажите **максимальное значение v-меры**, полученное при помощи данных моделей.

**Задание 7.14.3**

Сравним модификации K-means с использованием случайной инициализации **центроид** и с использованием алгоритма **K-means++** для инициализации центроид.

Для этого обучим на исходном датасете 2 модели k-means со следующими параметрами:

* + - * n\_clusters=3, init='k-means++', n\_init=1, random\_state=42
      * n\_clusters=3, init='random', n\_init=1, random\_state=42

В качестве ответа укажите **максимальное значение v-меры**, полученное при помощи данных моделей.

Подсказка: При решении задания модуля стандартизация должна быть включена.

**Задание 7.14.4**

Теперь сравним k-means с ещё одной модификацией — K-means mini batch. Воспользоваться реализацией K-means mini batch в библиотеке *sklearn* можно следующим образом:

from sklearn.cluster **import** MiniBatchKMeans

kmeans\_mini\_batch = MiniBatchKMeans(n\_clusters=

3, random\_state=

42)

kmeans\_mini\_batch.fit(X)

kmeans\_mini\_batch\_pred = kmeans\_mini\_batch.labels\_

Механизм кластеризации версии K-means mini batch схож с оригинальной версией алгоритма. Обучите на исходном датасете две модели:

* + - * *k-means* с параметрами n\_clusters=3, n\_init=1, random\_state=42
      * *MiniBatchKMeans* с параметрами n\_clusters=3, n\_init=1, random\_state=42

В качестве ответа введите **максимальное значение v-меры**, полученное при помощи данных моделей. В задании может понадобиться, а может не понадобиться нормализация и это нужно проверить во время решения задания.

**Задание 7.14.5**

Рассмотрим агломеративную кластеризацию. Сравним, как влияет на качество кластеризации разный тип расстояния между кластерами.

Обучите на исходном датасете четыре **модели AgglomerativeClustering** с параметром n\_clusters=3, меняя параметр linkage.

В качестве ответа укажите **максимальное значение v-меры**, полученное при помощи данных моделей. В задании может понадобиться, а может не понадобиться нормализация и это нужно проверить во время решения задания.

**Задание 7.14.6**

Сравним, как влияет предварительный расчёт матрицы смежности на качество агломеративной кластеризации.

Обучите на исходном датасете две **модели AgglomerativeClustering**:

* + - * с параметром n\_clusters=3
      * с параметром n\_clusters=3 и предварительно посчитанной матрицей смежности для объектов датасета

Построить матрицу смежности можно с помощью кода:

fromsklearn.neighborsimport kneighbors\_graph

connectivity = kneighbors\_graph(X, n\_neighbors=

6, include\_self=

False)

connectivity = 0.5 \* (connectivity + connectivity.T)

В качестве ответа введите **максимальное значение v-меры**, полученное при помощи данных моделей. В задании может понадобиться, а может не понадобиться нормализация и это нужно проверить во время решения задания.

**Задание 7.14.7**

Алгоритм DBSCAN очень требователен к параметрам: небольшое изменение в параметре eps или min\_samples может изменить результат и качество кластеризации.

Обучите на исходном датасете две модели *DBSCAN*:

* + - * с параметрами eps=0.9, min\_samples=35
      * с параметрами eps=0.8, min\_samples=35

В качестве ответа укажите **максимальное значение v-меры**, полученное при помощи данных моделей. В задании может понадобиться, а может не понадобиться нормализация и это нужно проверить во время решения задания.

**Задание 7.14.8**

Особенностью алгоритма DBSCAN является то, что помимо кластеризации этот алгоритм определяет **выбросы в выборке**. Посмотрим на качество кластеризации без учёта таких объектов.

Обучите на исходном датасете модель DBSCAN с параметрами eps=0.9, min\_samples=35. Посчитайте **значение v-меры** только для основных и граничных объектов выборки, то есть для объектов, что не являются выбросами. Ответ округлите до сотых и запишите с точкой.

Задание 7.14.9

В курсе мы рассмотрели две метода нормализации данных:

* + - * MinMax нормализация — приведение данных к масштабу между  0 и 1 .
      * Стандартная нормализация — данные имеют среднее 0 и стандартное отклонение 1.

Проверим, влияет ли предобработка данных на результат кластеризации. Обучите две **модели AgglomerativeClustering** с параметрами n\_clusters=3:

* + - * на признаках исходного датасета,
      * предварительно трансформируйте признаки при помощи стандартной нормализации.

Посчитайте **v-меру** для получившихся результатов, в качестве ответа введите наибольшее значение.

Задание 7.14.10

Обучите две модели AgglomerativeClustering с параметрами n\_clusters=3:

* + - * на признаках исходного датасета,
      * предварительно трансформируйте признаки при помощи MinMax нормализации.

Посчитайте **v-меру** для получившихся результатов, в качестве ответа введите наибольшее значение.

**Кластеризация текстов**

**Используйте юпитер ноутбук TextClustering**

Рассмотрим пример кластеризации текстов на стандартном наборе данных, новостях, выберем в наборе 4 из 20 категорий.

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**from** sklearn **import** metrics

**from** sklearn.cluster **import** **KMeans**

**from** sklearn.datasets **import** fetch\_20newsgroups

**from** sklearn.feature\_extraction.text **import** **CountVectorizer**

# Выбираем 4 категории новостей для легковесности примера

categories = [

'rec.sport.hockey', # хоккей

'talk.politics.mideast', # политические новости о Ближнем Востоке

'comp.graphics', # компьютерная графика

'sci.crypt' # криптография

]

# Скачиваем набор данных

dataset = fetch\_20newsgroups(subset='all', categories=categories,

shuffle=True, random\_state=42)

print("%d документов" % len(dataset.data))

print("%d категории" % len(dataset.target\_names))

# Записываем значения категорий для каждой новости

labels = dataset.target

Тексты новостей нужно преобразовать в числа и почистить от посторонних символов. Для этого воспользуемся инструментом из библиотеки sklearn, он построит **токенизированные данные**.

**Примечание:** Токенизация — это разделение текста на компоненты. Например, токенизация по предложениям — это процесс разделения письменного языка на предложения-компоненты. А токенизация по словам — это разделение предложений на слова-компоненты.

# Создаём объект, который будет токенизировать данные

analyzer = CountVectorizer(stop\_words='english').build\_analyzer()

# Токенизируем набор данных

docs = []

**for** document **in** dataset.data:

docs.append(analyzer(document.replace('\_', '')))

Каждый текст превратился в отдельный набор слов. Перейдём к **векторизации текстов**: воспользуемся моделью **Word2Vec**, которая для каждого слова строит числовой вектор, в нашем случае каждое слово будет **кодироваться** 50 числами. При этом вектора будем делать для тех слов, которые встречаются больше 20 раз во всех текстах документов. Также сделаем средний вектор для всех слов, таким образом получим вектор для текста в целом.

**Примечание:** более подробно с моделью Word2vec можно познакомиться в [этой статье](https://habr.com/ru/post/446530/).

**from** gensim.models **import** Word2Vec

# Обучаем модель векторайзера на нашем наборе данных

# На выходе мы получим вектор признаков для каждого слова

model = Word2Vec(docs, min\_count=20, size=50)

# Наивный подход к созданию единого эмбеддинга для документа – средний эмбеддинг по словам

**def** **doc\_vectorizer**(doc, model):

doc\_vector = []

num\_words = 0

**for** word **in** doc:

**try**:

**if** num\_words == 0:

doc\_vector = model[word]

**else**:

doc\_vector = np.add(doc\_vector, model[word])

num\_words += 1

**except**:

**pass**

**return** np.asarray(doc\_vector) / num\_words

# Составляем эмбеддинги для наших документов

X = []

**for** doc **in** docs:

X.append(doc\_vectorizer(doc, model))

Наша кластеризация будет работать лучше, если снизить **размерность**. Для этого воспользуемся методом TSNE.

**Примечание:** TSNE — это техника нелинейного снижения размерности и визуализации многомерных признаков, почитать о ней можно, например, [здесь](https://habr.com/ru/post/267041/).

При понижении размерности мы сохраняем близость элементов, то есть если элементы были близки при size=50, то они останутся близки и при size=2.

# t-SNE – метод понижения размерности

from sklearn.manifold import TSNE

# Создаём объект для выполнения t-SNE

tsne = TSNE(n\_components=2, random\_state=0)

# Преобразуем наши данные, понизив размерность с 50 до 2

X = tsne.fit\_transform(X)

Создадим кластеризатор **KMeans**и обучим на подготовленных данных. Мы можем посмотреть получившиеся центроиды и предсказанные кластеры.

# Создаём KMeans кластеризатор

kmeans = KMeans(n\_clusters=4)

# Обучаем кластеризатор на подготовленных данных

kmeans.fit(X)

# Получаем предсказанные кластеры

y\_pred = kmeans.labels\_.astype(np.int)

# Координаты полученных центроидов

print ("Координаты центроидов:\n", kmeans.cluster\_centers\_)

Какие у нас **метрики**? Коэффициент силуэта, однородность, полнота и V-мера говорят о том, что кластеры довольно цельные, высокие совпадения между истинными значениями и теми, что мы получили.