МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА - Московский технологический университет»

РТУ МИРЭА

Институт Информационных Технологий
Кафедра Прикладной Математики (ПМ)

ПРАКТИЧЕСКАЯ РАБОТА № 11

Выполнил студент группы ИКБО-08-19

Борисов А.В. (_____)

подпись

Принял Ассистент кафедры ПМ

Высоцкая А.А. (_____)

подпись

Практическая работа выполнена «____» _____2022 г.

«Зачтено» «____» _____2022 г.

Москва 2022

1. ***Найти данные для кластеризации.***

Для кластеризации были выбраны данные о размерах ирисов:

```python
data = load_iris()
df = pd.DataFrame(data=data.data, columns=data.feature_names)
df.head()
```

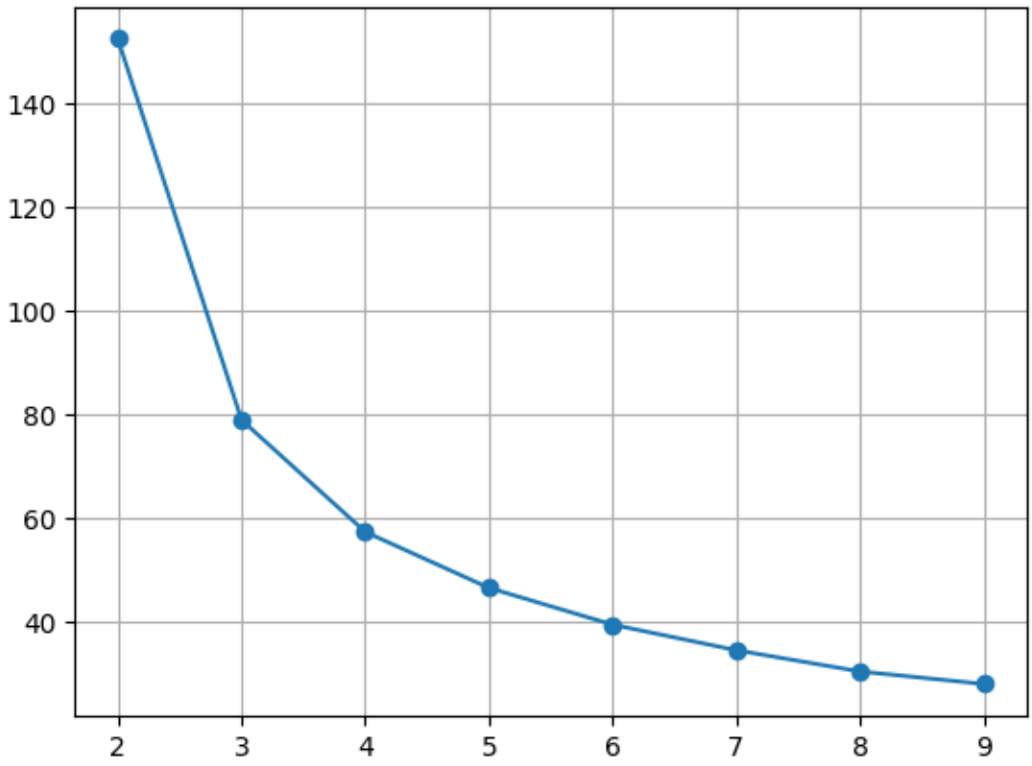
✓ 0.3s Python

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

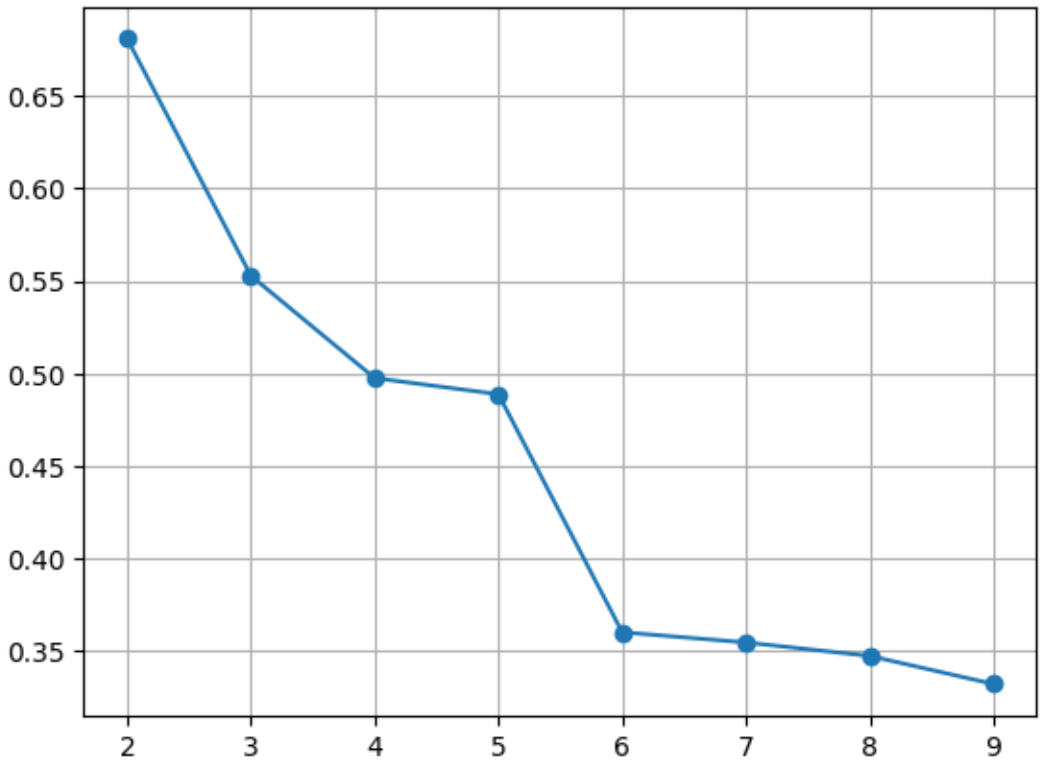
2. ***Провести кластеризацию данных с помощью алгоритма k-means. Использовать «правило локтя» и коэффициент силуэта для поиска оптимального количества кластеров.***

```python
models = []
score1 = []
score2 = []

for i in range(2,10):
    model = KMeans(n_clusters=i, random_state=123, init='k-means++').fit(df)
    models.append(model)
    score1.append(model.inertia_)
    score2.append(silhouette_score(df, model.labels_))
```

✓ 0.2s Python Python Python



Исходя из правила локтя можно сказать, что в данном наборе данных есть 3 отдельных кластера (так оно и есть, класса в нем 3).

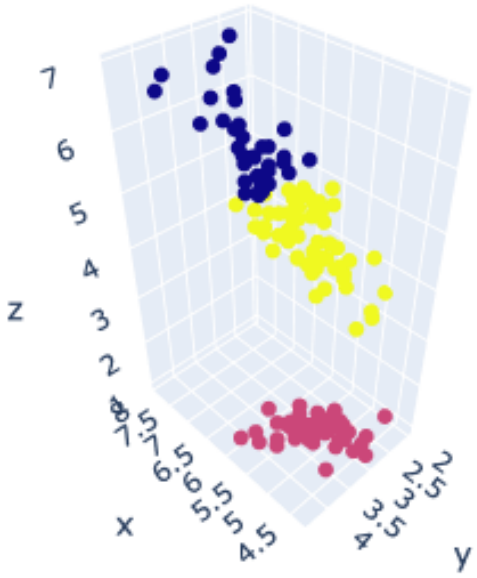


А вот коэффициент силуэта максимален при разделении исходного набора данных на два кластера. Это можно объяснить тем, что два класса в данном наборе довольно схожи (это и подтверждает график далее, где синий и желтый классы довольно близко друг к другу), однако возьмем число известное классов – 3.

```
time_start = time.time()

model1 = KMeans(n_clusters=3, random_state=123, init='k-means++')
model1.fit(df)

time_end = time.time() - time_start

labels = model1.labels_
df['Claster'] = labels

print('\n {:.4f} sec.'.format(time_end))
```

✓ 0.5s Python

После применения алгоритма k-means был построен трехмерный точечный график, который показывает распределение признаков классов по трем осям координат, и на основе данных полученных в результате работы данного алгоритма на графике были отмечены цветами 3 разных класса. Отчетливо видно, что два из трех классов очень схожи по своим признакам, поэтому коэффициент силуэта наибольший при разделении на два кластера.



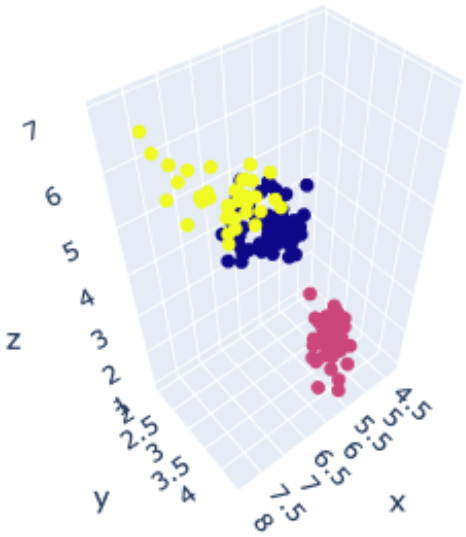
3. ***Провести кластеризацию данных с помощью алгоритма иерархической кластеризации.***

Для реализации алгоритма иерархической кластеризации тоже было выбрано распределение на три класса, после был построен трехмерный график и на нем были отмечены различными цветами классы, распределение которых мы получили после работы алгоритма.

В целом, результат работы очень схож на таковой у предыдущего алгоритма: тоже два очень близких по параметрам класса трудно отличимы друг от друга.

```python
time_start = time.time()

model2 = AgglomerativeClustering(3, compute_distances=True)
clastering = model2.fit(df)

time_end = time.time() - time_start

df['Claster'] = clastering.labels_

print('\n {:.4f} sec.'.format(time_end))
```

✓ 0.3s Python

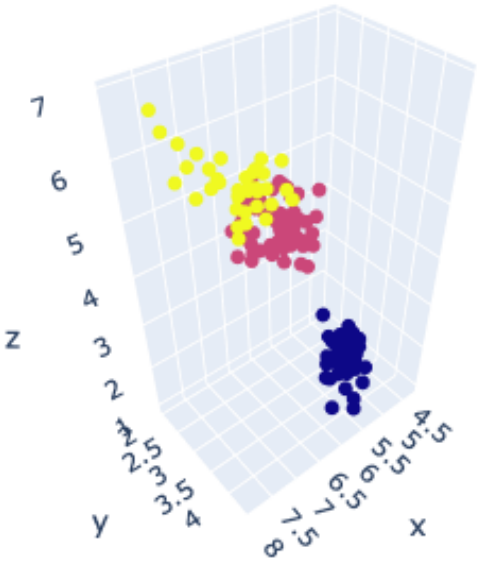


4. *Провести кластеризацию данных с помощью алгоритма DBSCAN.*

Алгоритм DBSCAN отличается от предыдущих тем, что сам находит число предполагаемых кластеров в данных. Эмпирическим путем были подобраны параметры eps и min_samples, при которых распределение на кластеры наиболее схоже с предыдущими результатами (будем считать что они заведомо верны, ведь число классов в исходных данных мы знаем, соответственно количество кластеров, на которые делятся эти данные, тоже).

```python
time_start = time.time()

model3 = DBSCAN(eps=1, min_samples=6).fit(df)

time_end = time.time() - time_start

df['Claster'] = model3.labels_

print('\n {:.4f} sec.'.format(time_end))
```

0.3s Python



5. ***Сравнить скорость работы алгоритмов. Результаты изобразить в виде таблицы.***

Для расчета времени работы алгоритмов была применена стандартная библиотека time. Для сравнения бралось только время работы каждого алгоритма, не учитывая остальные операции, связанные с обработкой данных, полученных при их помощи. Результаты приведены в таблице далее:

Алгоритм	Время (сек.)
k-means	0.0267
Иерарх. класт.	0.044
DBSCAN	0.037

Отчетливо видно, что существенно различается время работы у k-means и остальных алгоритмов. Иерархическая кластеризация и DBSCAN уже не имеют таких больших различий во времени обработки данных (в данном конкретном примере).