Университет ИТМО

Факультет: Программная инженерия

Факультет: Безопасность информационных технологий

ОТЧЁТ

по лабораторной работе №1
«Применение методов кластеризации с помощью Python»

Студенты группы СИА 3.2:

Кулинич Ярослав Вадимович P3213
Кириллова Надежда Сергеевна P3213
Тараненко Софья Сергеевна N3250
Гутник Дмитрий Вячеславович N3249

Преподаватель:

Добренко Наталья Викторовна

Санкт-Петербург

2020 г.

Задание:

1. Используйте метод K-средних и метод DBSCAN на самостоятельно сгенерированной выборке с количеством кластеров не менее 4. Для увеличения числа кластеров при генерации можно задать количество центров в функции make_blobs через параметр centers.
2. Используйте эти же два метода на датасете Mall Customers.
3. Для каждого метода необходимо построить график.

Выполнение:

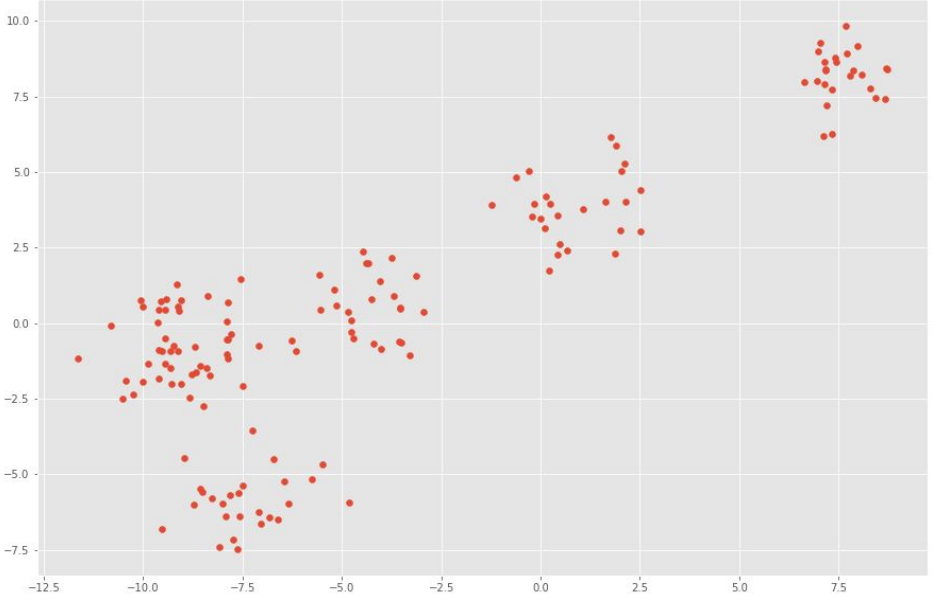
1. Для начала работы мы выполняем импорт и настройку рабочего окружения:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('ggplot')
plt.rcParams['figure.figsize']=(15,10)
```

Далее генерируем 150 точек с 6 центрами:

```
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
X,y = make_blobs(n_samples=150, centers=6, random_state=3)
plt.scatter(X[:,0], X[:,1])
```

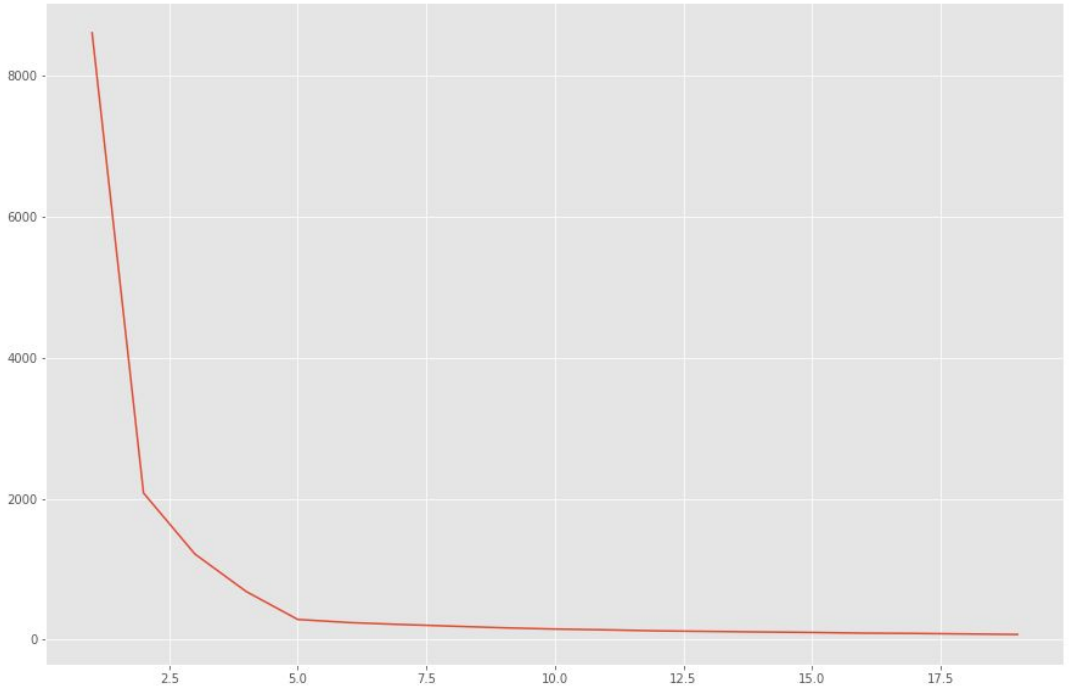
и получаем:



С помощью метода локтя находим оптимальное количество кластеров для метода к-средних:

```
criteries = []
for k in range(1,20):
  kmeansModel=KMeans(n_clusters=k, random_state=3)
  kmeansModel.fit(X)
  criteries.append(kmeansModel.inertia_)
print(criteries)
plt.plot(range(1,20), criteries)
```

и получаем график убывающей функции:



Мы нашли такое k, начиная с которого значение критерия k-means будет убывать не слишком быстро. Итак, оптимальное число кластеров равняется 5-ти.

Далее обучаем модель и строим график, на котором четко видны 5 кластеров:

```
kmeansModel=KMeans(n_clusters=5, random_state=1)
kmeansModel.fit(X)
labels = kmeansModel.labels_
plt.scatter(X[:,0], X[:,1], c=labels)
```

Следующим этапом будет метод DBSCAN. Методом подбора переменных eps (радиус окрестности) и min_samples (минимальное количество точек, образующих кластер) мы нашли их оптимальные значения для образования кластеров и визуализировали полученные данные:

```
from sklearn.cluster import DBSCAN

clustering = DBSCAN(eps=1, min_samples=6).fit_predict(X)
print(clustering)
plt.scatter(X[:,0], X[:,1], c=clustering)
```

```
[ 0  0  0  0  1 -1  0  1  2  2  1  3  3  1  1 -1  3  2 -1  3  3  3  2  3
  3  3  3  3 -1  3  3  3  2  0 -1  2  3  3  4  1 -1  3  1  4  0  3  0  4
  0  3 -1 -1  2  2 -1  1  4  1  3 -1  2  3  3  4  0  4  4 -1  2  0  3  4
  3  4  1  4  3  4  2 -1  4  3  3  4  4  3  4 -1  3  0 -1  4  2  3  3  3
  3  2  2  3  0  3  4  3 -1  3  3  3  2  1  3 -1 -1  3  3  3  0  1  2  0
  2  2  3  3  2  4 -1 -1 -1  1  4  0  3  0  2  0  0  0  1  4  0  0  2  3
  3  0  2 -1  0  2]
```

По данной таблице можно судить о количестве кластеров и шумов.

Кластеры обозначены цифрами 0, 1, 2, 3, 4, а шум обозначен значением -1.

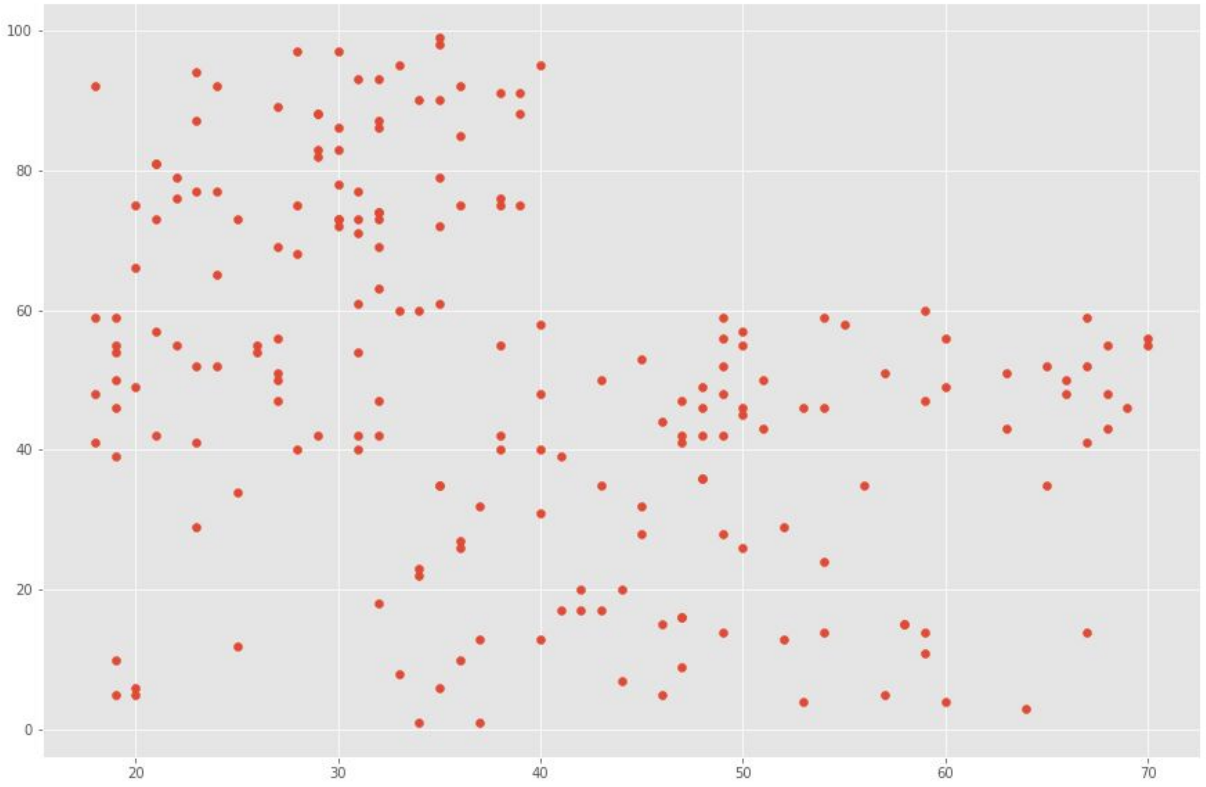
На графике ниже можно судить о том, что кластеры были образованы и их количество 5.

2. Второе задание предполагает использование методов k-means и DBSCAN на выделенном датасете Mall_Customers.
   Первым делом нам нужно импортировать файл в Google Colab для дальнейшей работы:

```
import io
from google.colab import files
uploaded = files.upload()
```

Построим распределение данных, полученных из файла Mall Customers:

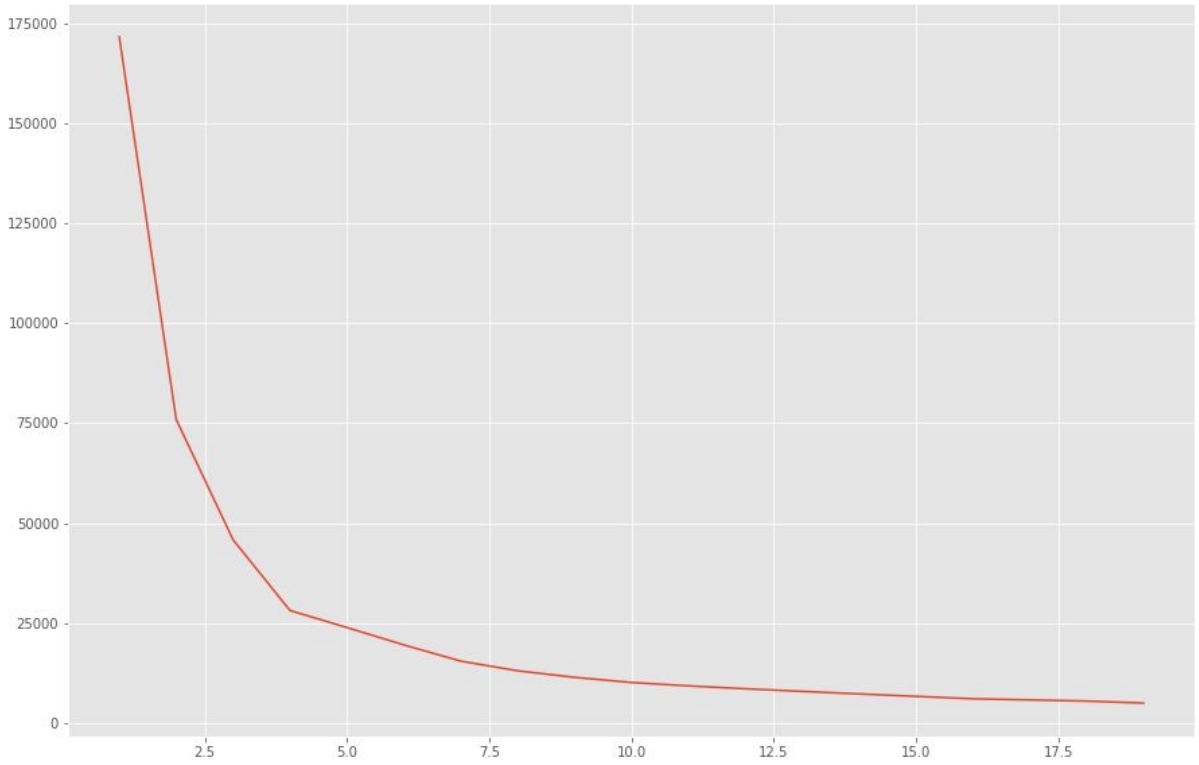
```
data =
pd.read_csv(io.StringIO(uploaded["Mall_Customers.csv"].decode('utf
-8')))
csvData = data[['Age', 'Spending Score (1-100)']]
plt.scatter(csvData.iloc[:,0], csvData.iloc[:,1])
```



С помощью метода локтя находим оптимальное количество кластеров для метода к-средних:

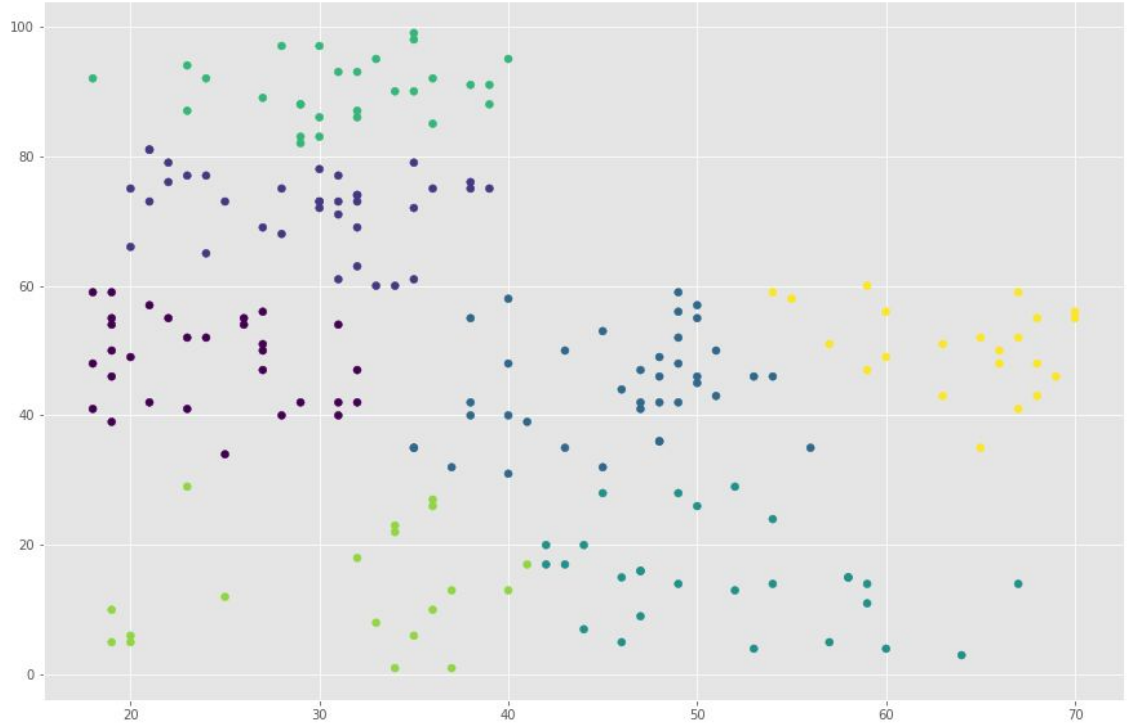
```
criteries = []
for k in range(1,20):
  kmeansModel=KMeans(n_clusters=k, random_state=3)
  kmeansModel.fit(csvData)
  criteries.append(kmeansModel.inertia_)
print(criteries)
plt.plot(range(1,20), criteries)
```

На графике мы можем наблюдать, что конкретной точки, с которой функция начинает медленно убывать, нет. Подбираем оптимальное значение, которое будет равняться 7-ми.



Строим модель распределения на кластеры методом k-means:

```
kmeansModel=KMeans(n_clusters=7, random_state=1)
kmeansModel.fit(csvData)
labels = kmeansModel.labels_
plt.scatter(csvData.iloc[:,0], csvData.iloc[:,1], c=labels)
```



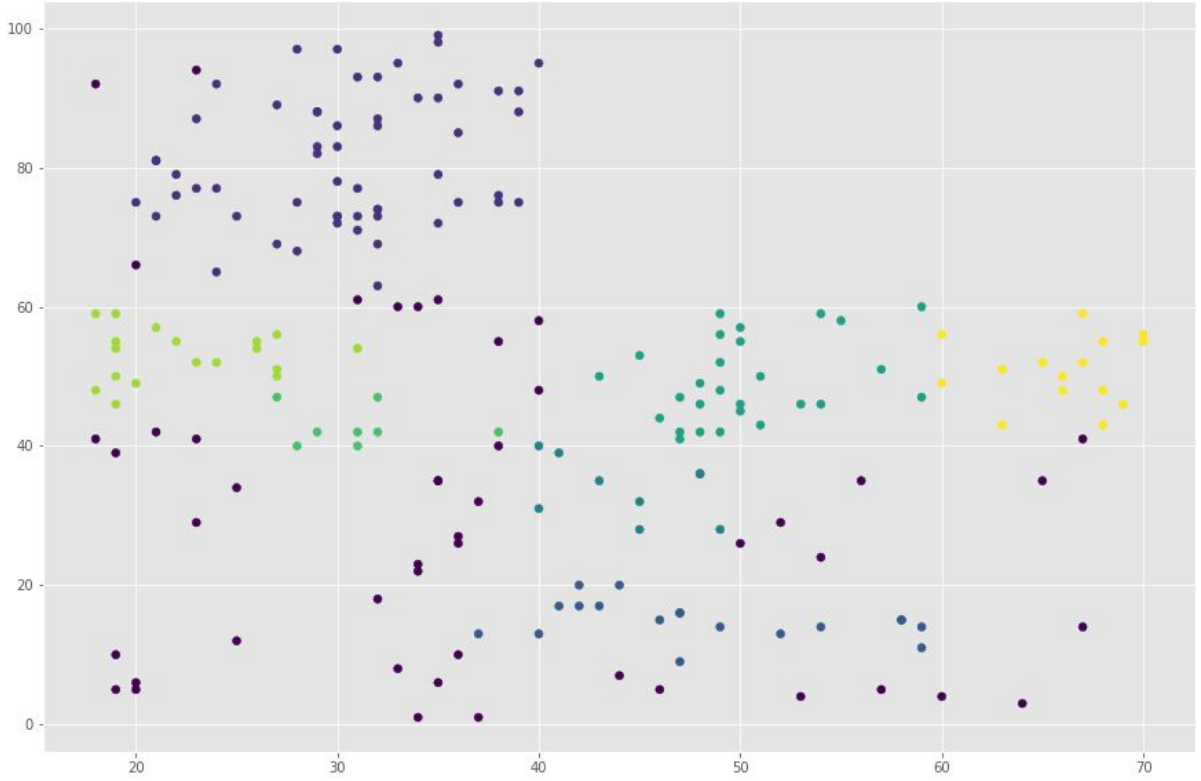
Методом DBSCAN мы образовали 7 кластеров и визуализировали их:

```
from sklearn.cluster import DBSCAN

clustering = DBSCAN(eps=6, min_samples=7).fit_predict(csvData)
print(clustering)
plt.scatter(csvData.iloc[:,0], csvData.iloc[:,1], c=clustering)
```

```
[-1  0 -1  0  4  0 -1 -1 -1  0 -1  0  1  0  1  0 -1 -1 -1  0 -1  0 -1  0
  1  0  2 -1  2  0 -1  0 -1 -1  1  0  1  0 -1  0 -1  0  2 -1  2  0  3  4
  4  4  3 -1  5  3  3  3  3  6  5  3  6  5  6  3  6  5  3  6  5  4  6  3
  6  6  3  5  3 -1  5  3  3 -1 -1  3  5  3  3  5 -1  3  6 -1  3  2  4  5
  3  5  3  5 -1  3  6  5  3 -1  6  3  6  6  6  5  4  5  5  5  6  3  3  3
  5 -1 -1  0 -1  0  2  0  1  0  1  0 -1  0 -1  0 -1  0 -1  0 -1  0  4  0
 -1  0  2  0 -1  0  1  0  1  0  1  0 -1  0 -1  0 -1  0 -1  0 -1  0  1  0
 -1  0  1  0 -1  0  1  0  1  0  1  0 -1  0  1  0  2  0 -1  0  1  0 -1  0
 -1  0  1  0  2  0 -1  0]
```

По таблице также можно судить о количестве кластеров. Они обозначены значениями от 0 до 6, шум обозначен значением -1. Количество кластеров на графике - 7:



Ссылка на Colab с готовой работой: ссылка на работу

Вывод:

При выполнении данной лабораторной работы были получены базовые знания о кластеризации данных с помощью языка Python. Были освоены метода k-means и DBscan, а также реализованы с помощью Python на сгенерированном и готовом наборах данных.