САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ ИНФОРМАТИКИ

Отчет по работе
по курсу «Современные инструменты анализа данных»
Тема: Кластеризация данных

Выполнила:
Гусейнова М. Э.

Проверила:
Добренко Н.В.

Санкт-Петербург
2024 г.

Цель работы: освоить практическое применение методов кластеризации данных, в частности алгоритмов K-средних и DBSCAN, на различных наборах данных.

Задание:

- Изучите теоретический материал и примеры в Jupyter Notebook по ссылке;
- Используйте метод K-средних и метод DBSCAN на самостоятельно сгенерированной выборке:
  - Создайте выборку с помощью функции make_blobs, задав количество кластеров не менее 4 через параметр centers;
  - Примените оба метода к полученным данным;
  - Визуализируйте результаты.
- Примените методы K-средних и DBSCAN к датасету Mall_Customers:
  - Загрузите датасет с Kaggle (Moll_Customers);
  - Проведите предварительный анализ данных;
  - Примените оба метода кластеризации;
  - Визуализируйте и интерпретируйте результаты.
- Для каждого метода постройте график и проанализируйте результаты:
  - Для K-средних используйте метод локтя для определения оптимального числа кластеров;
  - Для DBSCAN поэкспериментируйте с параметрами eps и min_samples.

Метод K-средних – это итеративный алгоритм, который разбивает набор данных на K предопределенных непересекающихся подгрупп (кластеров), где каждая точка данных принадлежит только одному кластеру.

Метод DBSCAN (Density-Based Spatial Clustering of Applications with Noise) – это алгоритм кластеризации, основанный на плотности. Он группирует вместе точки, которые находятся близко друг к другу в областях высокой плотности, отмечая точки в областях с низкой плотностью как выбросы.

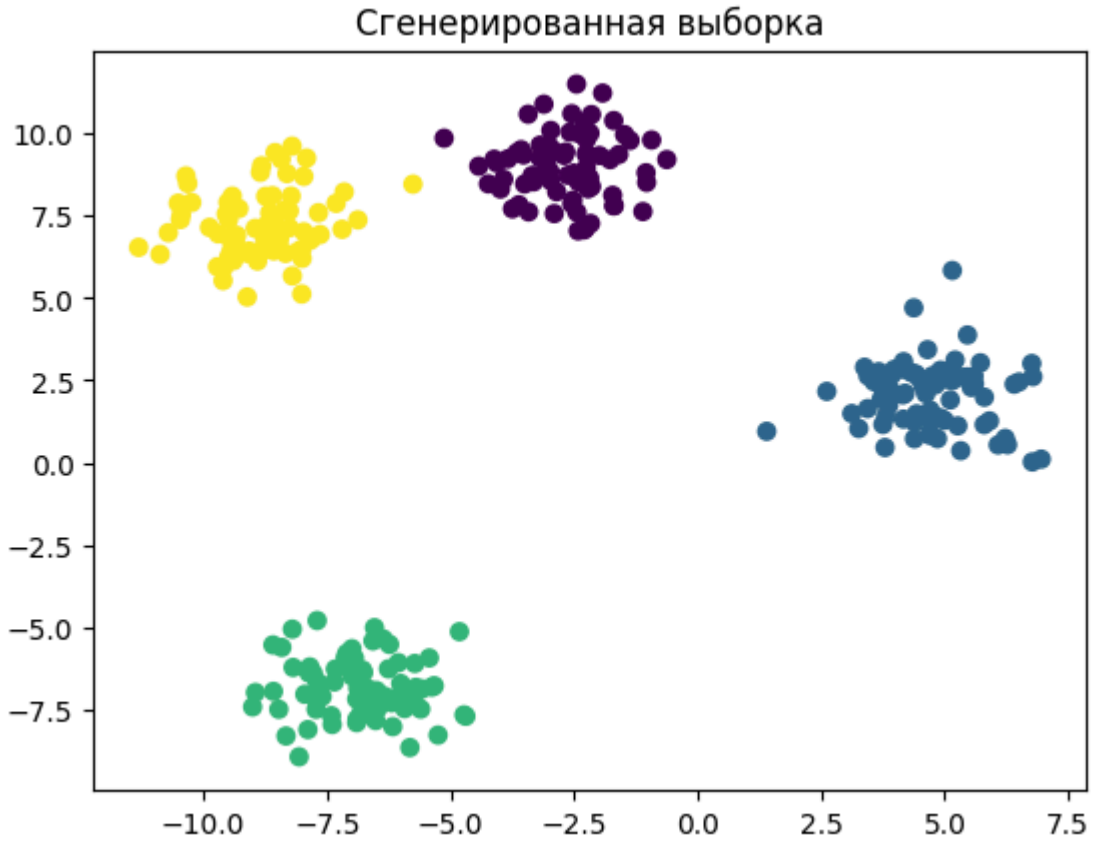
Ход работы:

Часть 1. Сгенерированная выборка.

1. **Генерация выборки.**

Воспользуемся библиотекой sklearn и функцией make_blobs. Сгенерируем 300 объектов, разделенные на 4 кластера и визуализируем данные:

```
X, y = make_blobs(n_samples=300, centers=4, random_state=42, cluster_std=1.0)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis')
plt.title("Сгенерированная выборка")
plt.show()
```



2. **Метод K-средних.**

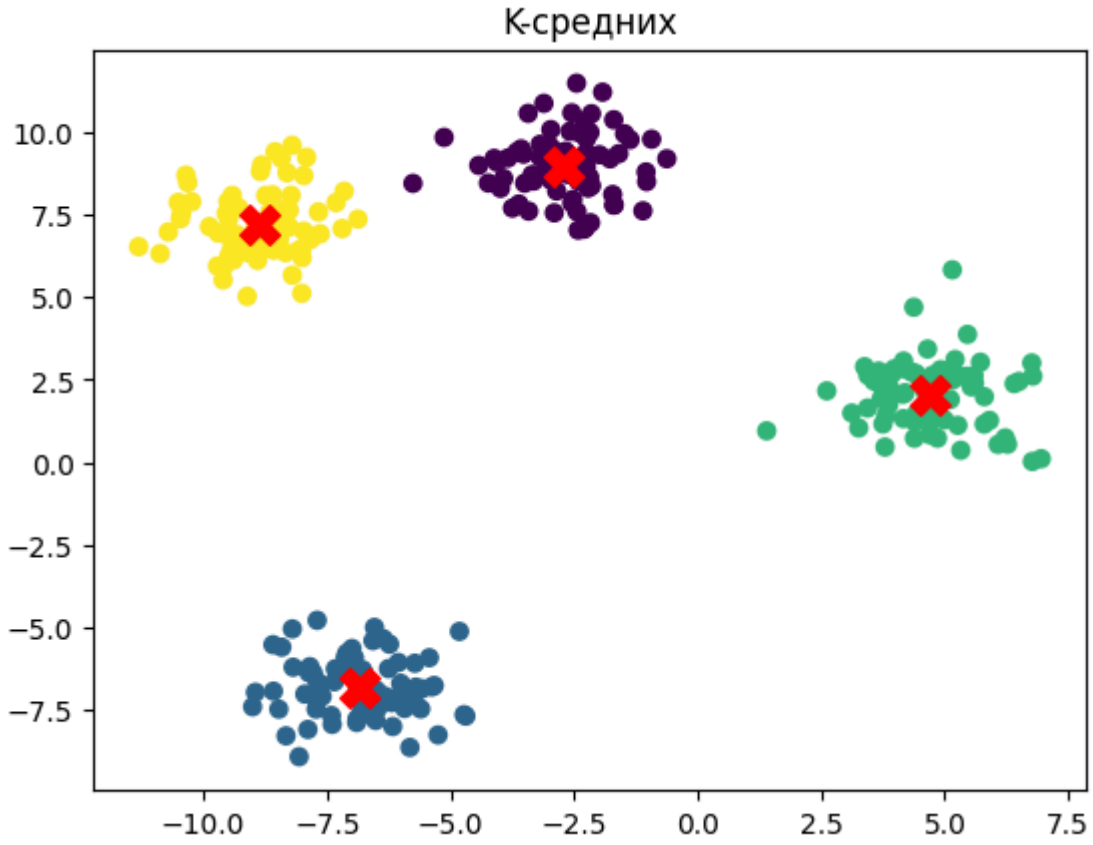
Воспользуемся библиотечным алгоритмом k-средних. `fit_predict(X)` обучает модель на данных X и возвращает метки кластеров.

```
# Метод K-средних
kmeans = KMeans(n_clusters=4, random_state=42)
kmeans_labels = kmeans.fit_predict(X)
```

Теперь визуализируем данные.

```
# Визуализация результатов K-средних
plt.scatter(X[:, 0], X[:, 1], c=kmeans_labels, cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], marker='X', s=200, color='red')
plt.title("K-средних")
plt.show()
```

Центры кластеров помечены красным крестом.



3. Метод DBSCAN

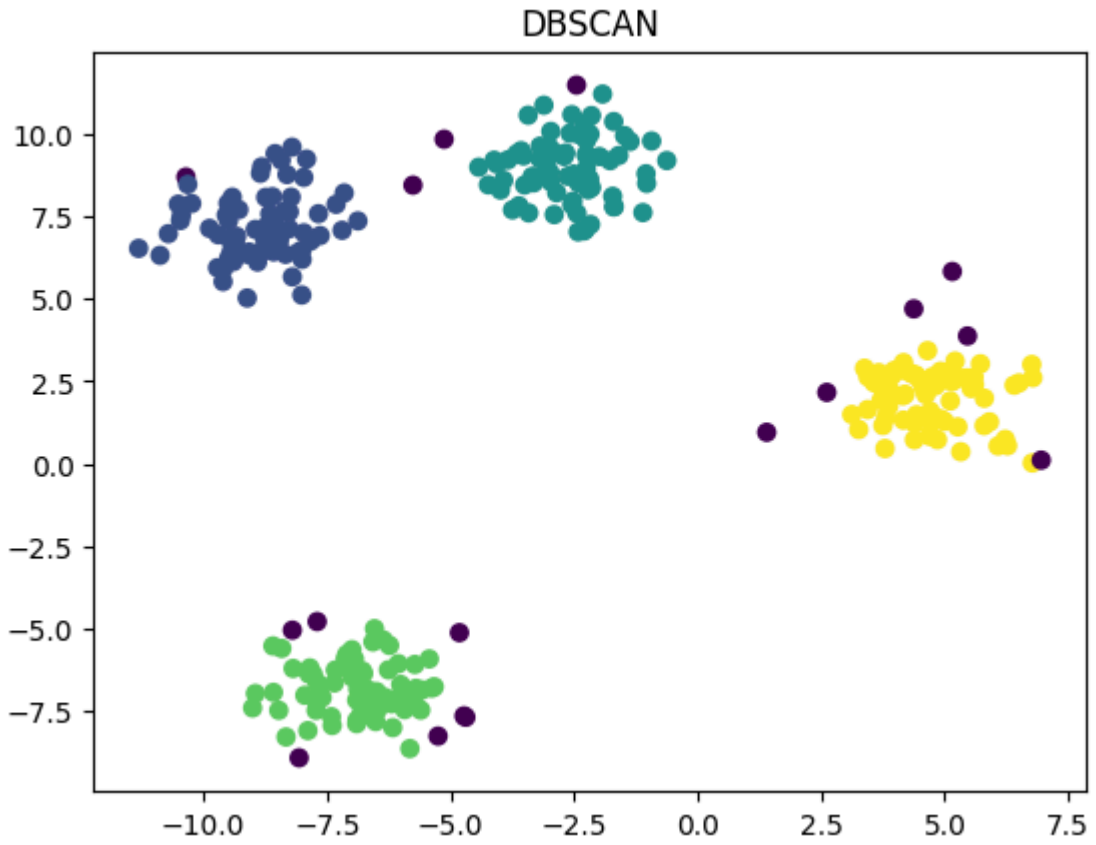
Теперь к нашей выборке применим метод DBSCAN.

```
# Метод DBSCAN
clustering = DBSCAN(eps=0.8, min_samples=5)
dbscan_labels = clustering.fit_predict(X)
```

Здесь радиус для объединения точек в кластеры равен 0.8, минимальное количество точек для образования кластера равно 5.

Теперь визуализируем полученные данные.

```
plt.scatter(X[:, 0], X[:, 1], c=dbscan_labels, cmap='viridis')
plt.title("DBSCAN")
plt.show()
```



Анализ результатов:

- Метод K-средних хорошо разделил выборку на четыре кластера, визуально кластеры выглядят компактными и хорошо определенными.
- Метод DBSCAN тоже хорошо справился с кластеризацией, но результаты могут зависеть от выбранных параметров eps и min_samples. В методе DBSCAN точки, не принадлежащие явно к какому-либо кластеру, могут быть помечены меткой -1 (выбросы), что и произошло.

Часть 2. Датасет Mall_Customers.

1. **Предварительный анализ.**

Загружаем датасет с Kaggle и проверяем данные:

```
file_path = 'Mall_Customers.csv'
mall_customers = pd.read_csv(file_path)
```

```
mall_customers
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

200 rows × 5 columns

Проверим, в порядке ли наши данные (например, есть ли пустые значения или значения других типов):

```
mall_customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

Вроде все хорошо. Для полной уверенности проверим, что в столбце Gender нет лишних данных:

```
mall_customers['Gender'].unique()
```

```
array(['Male', 'Female'], dtype=object)
```

Убедились, что все хорошо. Я выбрала для кластеризации признаки “Annual Income (k$)” и “Spending Score (1-100)”. Теперь стандартизируем данные:

```
features = ['Annual Income (k$)', 'Spending Score (1-100)']
X = mall_customers[features].values

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Теперь можем приступать к применению методов кластеризации.

2. **Метод k-средних.**

Воспользуемся методом локтя для определения оптимального числа кластеров.

```
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)
```

Визуализируем:

```
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Метод локтя для K-средних')
plt.xlabel('Количество кластеров')
plt.ylabel('Внутрикластерное расстояние')
plt.show()
```

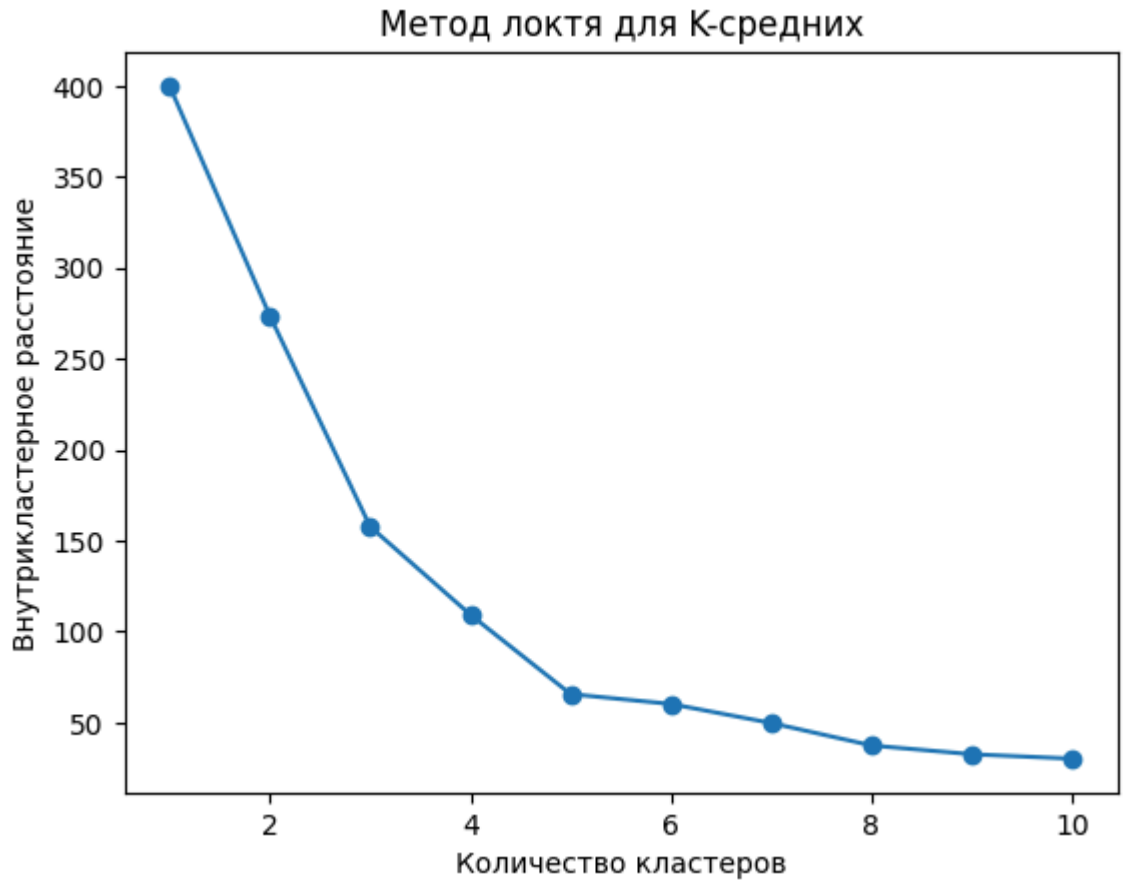
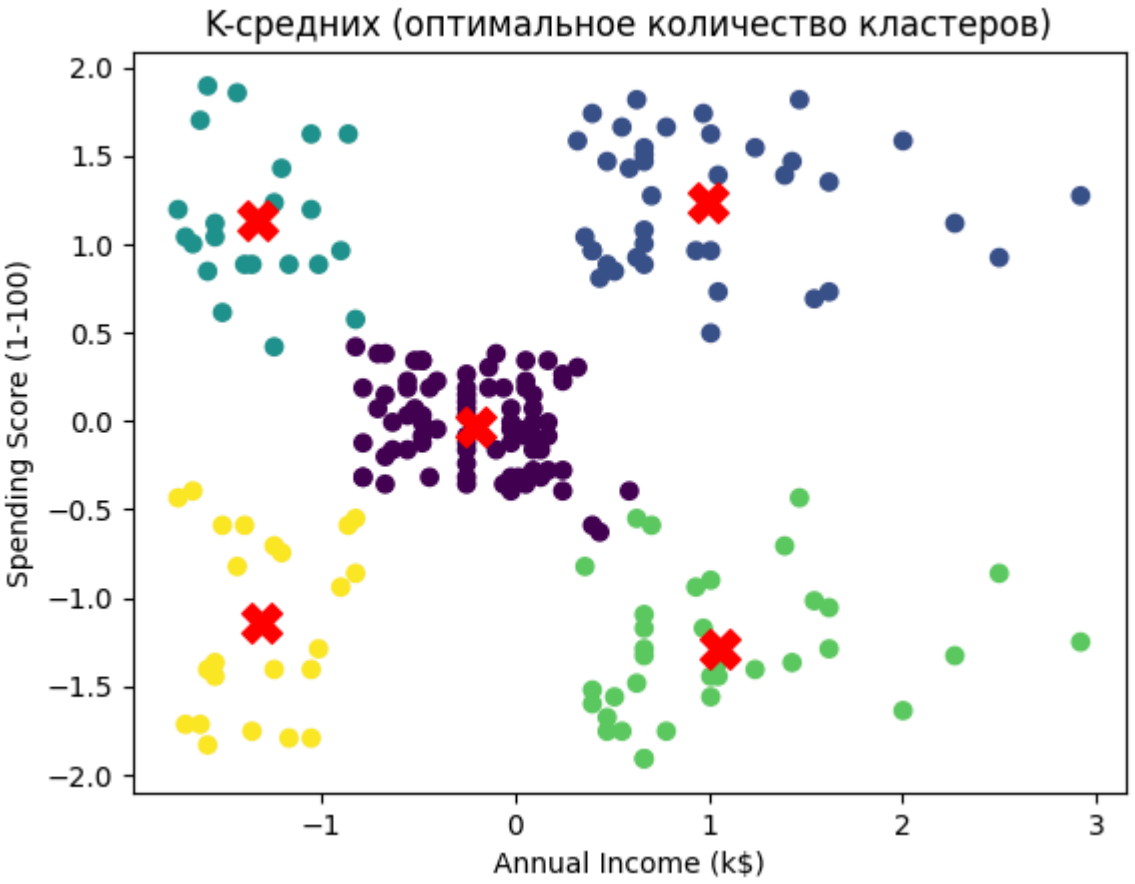


График показывает "локоть" (точку изменения крутизны графика), которая часто считается оптимальным количеством кластеров. В данном случае, локоть расположен примерно в районе **5** кластеров.
Применим метод K-средних с учетом этой информации:

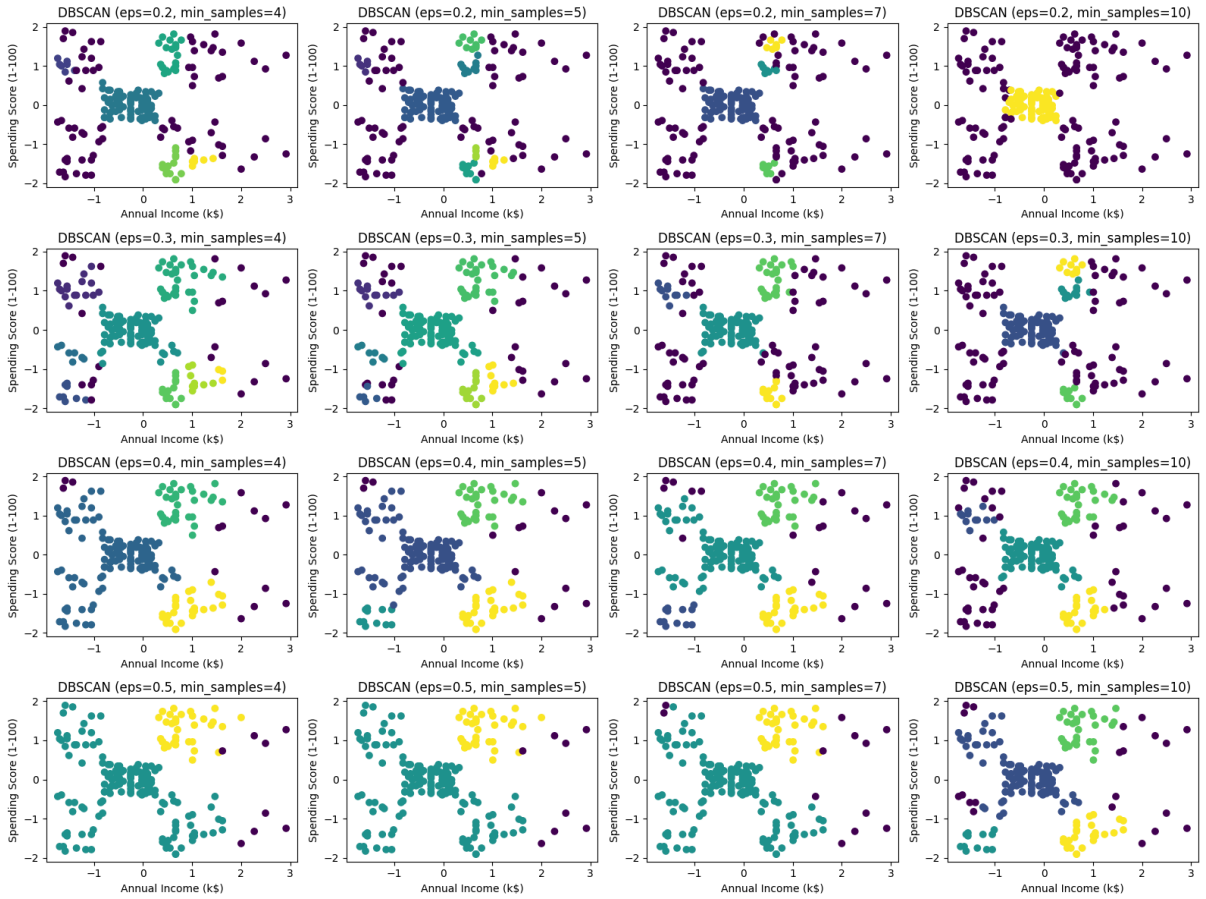
```
# Выбор оптимального числа кластеров для K-средних
optimal_clusters_kmeans = 5

# Метод K-средних с оптимальным числом кластеров
kmeans_optimal = KMeans(n_clusters=optimal_clusters_kmeans, random_state=42)
kmeans_labels_optimal = kmeans_optimal.fit_predict(X_scaled)
```

Теперь визуализируем**.** Каждый кластер обозначен уникальным цветом, а центры кластеров выделены красными крестами.



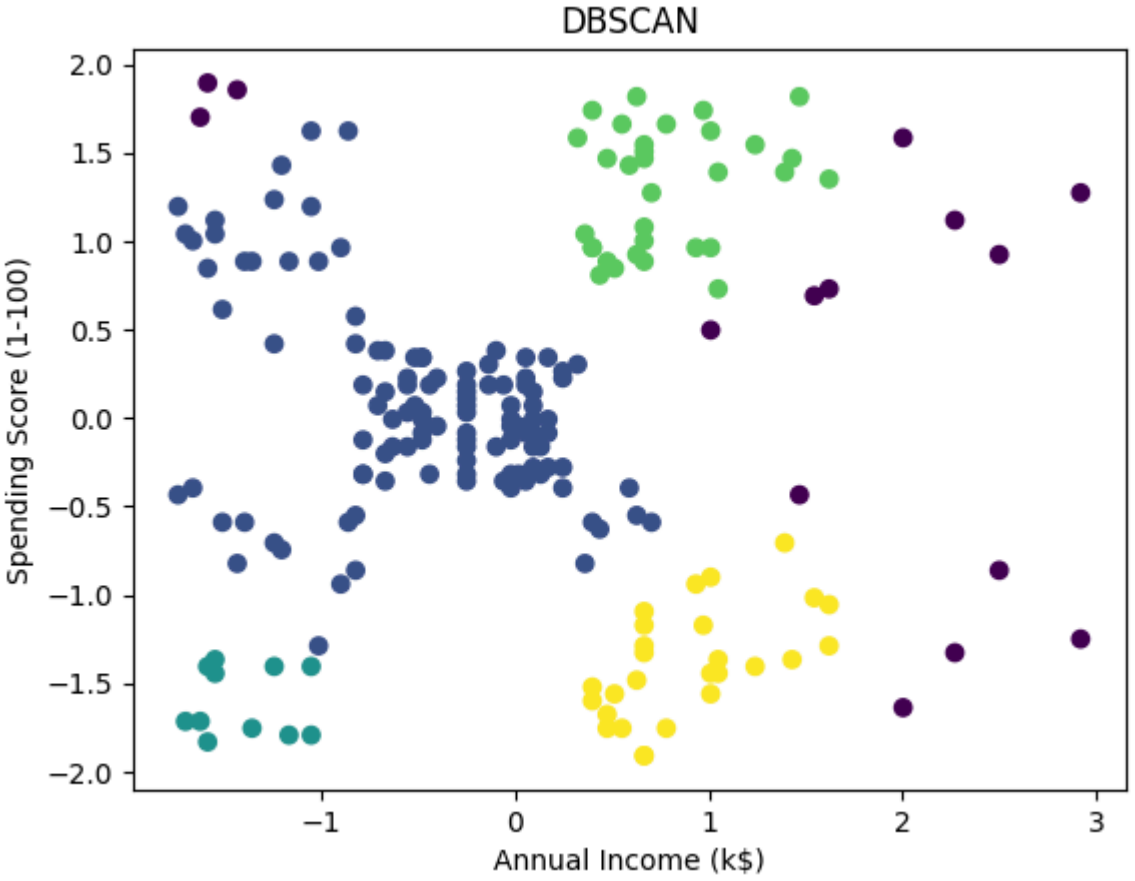
3. Метод DBSCAN

Попробуем разные параметры eps и min_samples.



Мне не удалось достичь кластеризации как в методе К-средних. Как мне кажется, один из лучших результатов у eps = 0.4 и min_samples = 5.

```
dbscan = DBSCAN(eps=0.4, min_samples=5)
dbscan_labels = dbscan.fit_predict(X_scaled)

# Визуализация результатов DBSCAN
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=dbscan_labels, cmap='viridis', marker='o')
plt.title("DBSCAN")
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.show()
```



Анализ результатов:

- Метод K-средних показал четкую кластеризацию, хорошо справился с работой.
- Метод DBSCAN выделил выбросы, но не так хорошо справился с задачей, кластеры получились не такими четкими и симметричными, как в методе K-средних.