Федеральное Государственное бюджетное учреждение

высшего профессионального образования

"Заполярный Государственный Университет имени Н. М. Федоровского"

Кафедра «Информационные системы и технологии»

**Многомерный анализ данных**

**Отчет по лабораторной работе №2**

Тема: Кластерный анализ данных

Вариант 10

Выполнили:

студенты группы ИС-21

Сидельников Максим Эдуардович

Соколов Александр Васильевич

Проверил:

Фаддеенков Андрей Владимирович

**Задание 2. Кластерный анализ**

Цель: ознакомиться с основными положениями, понятиями и методами кластерного анализа.

**Порядок выполнения задания**

1. Провести кластеризацию имеющихся данных методом k-средних. Предположить, что имеется 2, 3, 4 кластера. Для каждого числа кластеров выполнить кластеризацию, используя разные меры расстояния:

* Евклидово расстояние
* Манхэттенское расстояние
* Расстояние Чебышева

1. Для каждого выделенного кластера рассчитать (в каждом варианте числа кластеров и при каждой мере расстояния)

* дисперсию кластера,
* радиус кластера.

1. Для всех вариантов кластеризации рассчитать суммарное квадратичное отклонение. Результаты кластеризации представить графически.
2. Указать наилучший вариант кластеризации из полученных.

**Ход работы**

Метод k-средних (k-means) — это алгоритм кластеризации, который используется для разделения набора данных на k различных кластеров. Основная цель алгоритма — минимизировать внутрикластерное расстояние, т.е. сделать так, чтобы точки внутри одного кластера были как можно ближе друг к другу.

**Основные шаги алгоритма k-средних:**

1. **Инициализация центроидов**. Выбираются k начальных центроидов. Это могут быть случайные точки из набора данных или случайно сгенерированные точки в пространстве признаков.
2. **Назначение точек кластерам**. Каждая точка данных назначается ближайшему центроиду. Близость обычно измеряется с использованием евклидова расстояния, но могут использоваться и другие метрики, такие как манхэттенское или чебышевское расстояние.
3. **Обновление центроидов**. После того как все точки назначены кластерам, центроиды пересчитываются как среднее всех точек, принадлежащих каждому кластеру.
4. **Проверка сходимости**. Алгоритм повторяет шаги 2 и 3 до тех пор, пока центроиды не перестанут изменяться (или изменения будут меньше заданного порога), или пока не будет достигнуто максимальное количество итераций.

Согласно вышеописанному алгоритму, был написан код на языке Python (см. Листинг), с помощью библиотеки matplotlib была создана инфографика.

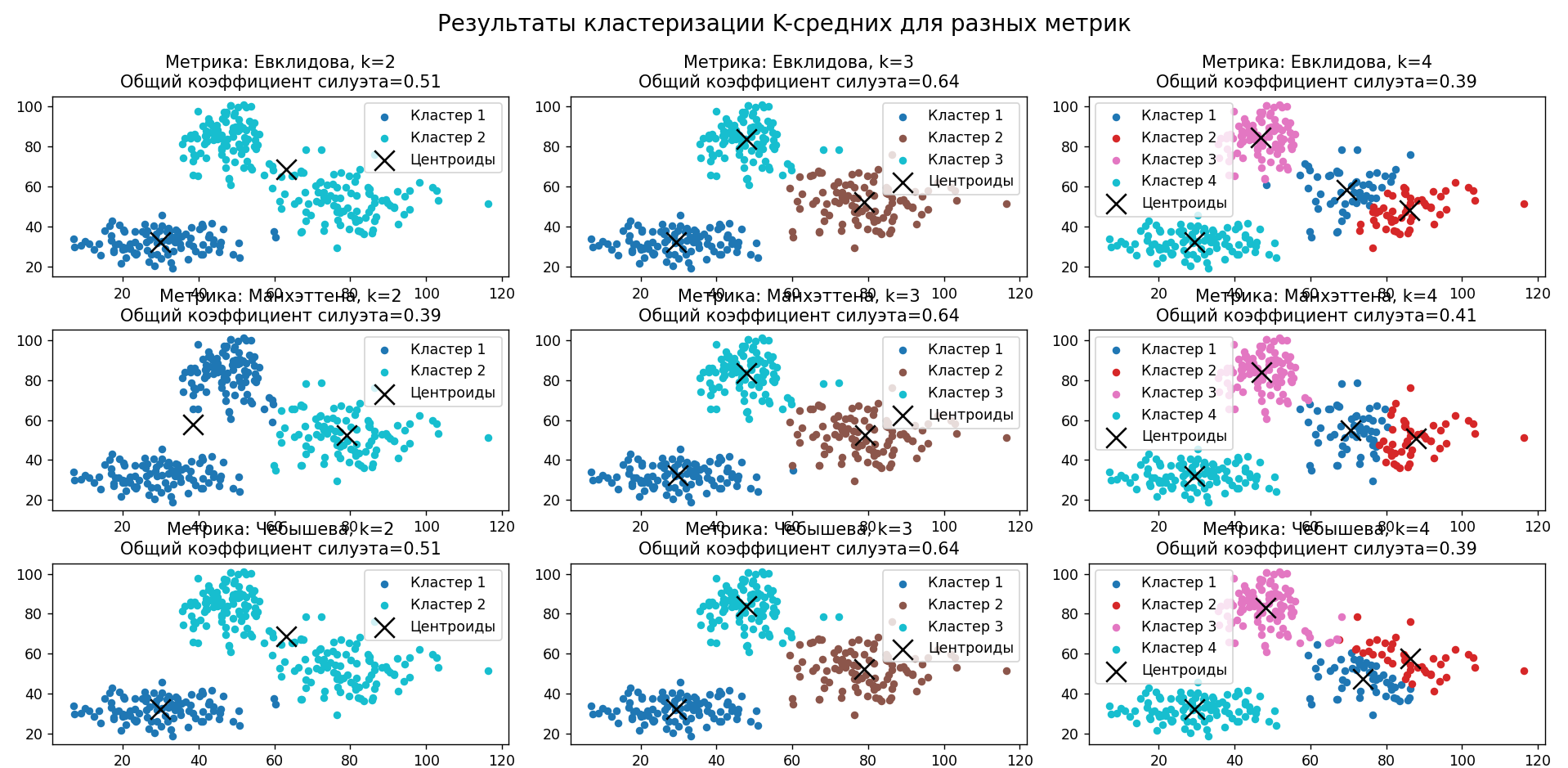


Рисунок 1. Результаты визуализации кластерного анализа

Посмотрев на Рис.1 видно, что как с визуальной точки зрения, так и с точки зрения общего коэффициента силуэта (далее – ОКС), K=3 для числа кластеров является наиболее оптимальным вариантом (учитывая, что чем ближе ОКС к единице – тем лучше были построены кластеры). Однако, имеет смысл также взглянуть на более подробные данные по каждому из графиков.

Получим следующие результаты:

**Евклидова метрика**

1. **K=2**:
   1. Коэффициент силуэта: 0.51
   2. Суммарное квадратичное отклонение: 144351.96
   3. Дисперсия и радиус:
      1. Кластер 1: Дисперсия=152.09, Радиус=30.31
      2. Кластер 2: Дисперсия=650.70, Радиус=55.96
2. **K=3**:
   1. Коэффициент силуэта: 0.64
   2. Суммарное квадратичное отклонение: 46407.63
   3. Дисперсия и радиус:
      1. Кластер 1: Дисперсия=136.39, Радиус=22.95
      2. Кластер 2: Дисперсия=200.23, Радиус=37.38
      3. Кластер 3: Дисперсия=128.88, Радиус=25.07
3. **K=4**:
   1. Коэффициент силуэта: 0.39
   2. Суммарное квадратичное отклонение: 38318.12
   3. Дисперсия и радиус:
      1. Кластер 1: Дисперсия=154.36, Радиус=25.48
      2. Кластер 2: Дисперсия=125.16, Радиус=30.40
      3. Кластер 3: Дисперсия=105.16, Радиус=20.82
      4. Кластер 4: Дисперсия=136.39, Радиус=22.95

**Манхэттенская метрика**

1. **K=2**:
   1. Коэффициент силуэта: 0.39
   2. Суммарное квадратичное отклонение: 313422.17
   3. Дисперсия и радиус:
      1. Кластер 1: Дисперсия=1395.49, Радиус=59.21
      2. Кластер 2: Дисперсия=332.62, Радиус=38.64
2. **K=3**:
   1. Коэффициент силуэта: 0.64
   2. Суммарное квадратичное отклонение: 71950.14
   3. Дисперсия и радиус:
      1. Кластер 1: Дисперсия=214.47, Радиус=33.07
      2. Кластер 2: Дисперсия=308.51, Радиус=38.19
      3. Кластер 3: Дисперсия=199.63, Радиус=29.67
3. **K=4**:
   1. Коэффициент силуэта: 0.41
   2. Суммарное квадратичное отклонение: 60665.76
   3. Дисперсия и радиус:
      1. Кластер 1: Дисперсия=234.92, Радиус=31.23
      2. Кластер 2: Дисперсия=207.65, Радиус=28.90
      3. Кластер 3: Дисперсия=177.51, Радиус=27.05
      4. Кластер 4: Дисперсия=205.63, Радиус=29.51

**Чебышевская метрика**

1. **K=2**:
   1. Коэффициент силуэта: 0.51
   2. Суммарное квадратичное отклонение: 109559.15
   3. Дисперсия и радиус:
      1. Кластер 1: Дисперсия=134.38, Радиус=30.21
      2. Кластер 2: Дисперсия=484.10, Радиус=53.25
2. **K=3**:
   1. Коэффициент силуэта: 0.64
   2. Суммарное квадратичное отклонение: 39248.26
   3. Дисперсия и радиус:
      1. Кластер 1: Дисперсия=118.67, Радиус=22.39
      2. Кластер 2: Дисперсия=165.34, Радиус=37.37
      3. Кластер 3: Дисперсия=109.59, Радиус=24.54
3. **K=4**:
   1. Коэффициент силуэта: 0.39
   2. Суммарное квадратичное отклонение: 33989.98
   3. Дисперсия и радиус:
      1. Кластер 1: Дисперсия=85.24, Радиус=17.82
      2. Кластер 2: Дисперсия=131.31, Радиус=30.00
      3. Кластер 3: Дисперсия=114.71, Радиус=22.26
      4. Кластер 4: Дисперсия=118.67, Радиус=22.39

Взяв во внимание вышеописанные данные, можно сделать следующий **вывод**: несмотря на внешнюю удачность K=3 вариантов, их высокие ОКС, наилучшие результаты по суммарному квадратичному отклонению достигнуты при 𝐾=4 для всех метрик, особенно для чебышевской метрики, где СКО минимально (33989.98). Это указывает на более плотное распределение точек вокруг центроидов, что делает K=4 предпочтительным выбором для кластеризации в данном случае.

**Листинг**

*import* numpy *as* np

*import* pandas *as* pd

*import* matplotlib.pyplot *as* plt

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

*# Загрузка данных*

data = pd.read\_excel("Lab2Data.xlsx")

points = data[['x', 'y']].values

*# Функция вычисления расстояния*

def distance(*a*, *b*, *metric*='euclidean'):

*if* metric == 'euclidean':

*return* np.sqrt(np.sum((a - b)\*\*2))

*elif* metric == 'manhattan':

*return* np.sum(np.abs(a - b))

*elif* metric == 'chebyshev':

*return* np.max(np.abs(a - b))

*else*:

*raise* ValueError("Неизвестная метрика")

*# Алгоритм k-средних*

def k\_means(*points*, *k*, *metric*='euclidean', *max\_iter*=100, *tol*=1e-4):

    np.random.seed(42)

    centers = points[np.random.choice(points.shape[0], k, *replace*=False)]

*for* iteration *in* range(max\_iter):

        clusters = [[] *for* \_ *in* range(k)]

*for* p *in* points:

            dists = [distance(p, c, metric) *for* c *in* centers]

            cluster\_idx = np.argmin(dists)

            clusters[cluster\_idx].append(p)

        clusters = [np.array(c) *for* c *in* clusters *if* len(c) > 0]

*if* len(clusters) < k:

            centers = points[np.random.choice(points.shape[0], k, *replace*=False)]

*continue*

        new\_centers = np.array([np.mean(c, *axis*=0) *for* c *in* clusters])

        shift = np.sqrt(np.sum((centers - new\_centers)\*\*2))

        centers = new\_centers

*if* shift < tol:

*break*

*return* clusters, centers

*# Расчет статистики кластеров*

def cluster\_statistics(*clusters*, *centers*, *metric*='euclidean'):

    total\_variance = 0.0

    cluster\_stats = []

*for* i, cluster *in* enumerate(clusters):

        dists = np.array([distance(p, centers[i], metric) *for* p *in* cluster])

        variance = np.mean(dists\*\*2)

        radius = np.max(dists)

        total\_variance += np.sum(dists\*\*2)

        cluster\_stats.append({

            'cluster': i+1,

            'variance': variance,

            'radius': radius

        })

*return* cluster\_stats, total\_variance

*# Визуализация результатов и расчет метрик*

k\_values = [2, 3, 4]

metrics = ['euclidean', 'manhattan', 'chebyshev']

results = []

*# Сначала выполняем все вычисления и выводим результаты*

*for* i, metric *in* enumerate(metrics):

*for* j, k *in* enumerate(k\_values):

        clusters, centers = k\_means(points, k, *metric*=metric)

        c\_stats, total\_var = cluster\_statistics(clusters, centers, *metric*=metric)

        silhouette = silhouette\_score(points, np.concatenate([[idx] \* len(c) *for* idx, c *in* enumerate(clusters)]))

        results.append({

            'metric': metric,

            'k': k,

            'cluster\_statistics': c\_stats,

            'total\_variance': total\_var,

            'silhouette': silhouette,

            'centers': centers,

            'clusters': clusters  *# Сохраняем кластеры для последующей визуализации*

        })

*# Вывод результатов*

*for* res *in* results:

    print(f"=== Метрика: {res['metric'].capitalize()}, K={res['k']} ===")

    print("Центроиды:")

*for* idx, center *in* enumerate(res['centers']):

        print(f"  Центроид {idx+1}: x={center[0]:.4f}, y={center[1]:.4f}")

*for* c\_stat *in* res['cluster\_statistics']:

        print(f"Кластер {c\_stat['cluster']}: Дисперсия={c\_stat['variance']:.4f}, Радиус={c\_stat['radius']:.4f}")

    print(f"Суммарное квадратичное отклонение={res['total\_variance']:.4f}")

    print(f"Силуэтный коэффициент={res['silhouette']:.4f}\n")

*# Затем создаем визуализацию*

fig, axes = plt.subplots(len(metrics), len(k\_values), *figsize*=(15, 12))

fig.suptitle('Результаты кластеризации K-средних для разных метрик', *fontsize*=16)

*for* idx, res *in* enumerate(results):

    i = idx // len(k\_values)

    j = idx % len(k\_values)

    ax = axes[i, j]

    colors = plt.colormaps.get\_cmap('tab10')(np.linspace(0, 1, res['k']))

*for* cluster\_idx, cluster *in* enumerate(res['clusters']):

        ax.scatter(cluster[:, 0], cluster[:, 1], *s*=20, *color*=colors[cluster\_idx], *label*=f'Кластер {cluster\_idx+1}')

    ax.scatter(res['centers'][:, 0], res['centers'][:, 1], *s*=200, *color*='black', *marker*='x', *label*='Центроиды')

    ax.set\_title(f'Метрика: {"Евклидова" *if* res["metric"] == "euclidean" *else* "Манхэттена" *if* res["metric"] == "manhattan" *else* "Чебышева"}, k={res["k"]}\nОбщий коэффициент силуэта={res["silhouette"]:.2f}')

    ax.legend()

plt.tight\_layout(*rect*=(0, 0, 1, 0.95))

plt.show()