Министерство науки и высшего образования

Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования «Рыбинский государственный

авиационный технический университет имени П. А. Соловьева»

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И СИСТЕМЫ УПРАВЛЕНИЯ

Математического и программного обеспечения электронных

вычислительных средств

ОТЧЁТ

по дисциплине:

«Методы и алгоритмы анализа данных»

на тему:

«Кластеризация»

Выполнил: студент группы ИВМ-24 Морозов А. А.

Руководитель: ассистент Вязниковцев Д. А.

Рыбинск 2024

Содержание

[Цель работы 3](#_Toc183892346)

[1 Первичная обработка данных 4](#_Toc183892347)

[2 Снижение размерности 5](#_Toc183892348)

[2.1 Метод главных компонент 5](#_Toc183892349)

[2.2 Метод стохастического вложения соседей с t-распределением 6](#_Toc183892350)

[2.3 Сравнение двух методов 7](#_Toc183892351)

[3 Методы кластеризации 8](#_Toc183892352)

[3.1 Метод *K*-средних 8](#_Toc183892353)

[2.5 Метод *DBSCAN* 11](#_Toc183892354)

[Вывод 13](#_Toc183892355)

# Цель работы

Целями данной лабораторной работы являются:

- изучение теории о кластерном анализе;

- первичная обработка данных;

- работа с методами главных компонент, стохастического вложения соседей с *t*-распределением, *K*-средних и *DBSCAN*;

# 1 Первичная обработка данных

В начале необходимо загрузить датасет *data\_Mar\_64.txt* и вывести его в консоль (рисунок 1.1):

data = pd.read\_csv("data\_Mar\_64.txt", header=None, sep=',', quotechar='"')

print(data)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1.1 – Загруженный датасет

Необходимо разделить датасет на 2. В первом датасете будет храниться целевая переменная, соответствующая столбцу с индексом 0, в закодированном формате, а в другом датасете оставшиеся столбцы:

X, y\_name = data.iloc[:, 1:].values, data.iloc[:, 0].values  
y = LabelEncoder().fit\_transform(y\_name)

# 2 Снижение размерности

# 2.1 Метод главных компонент

Необходимо снизить размерность признакового пространства до двух. После изменения размерности отфильтруем значения целевой переменной по порогу до 14 и выведем получившиеся кластеры на экран (рисунок 1.2):

X\_PCA = PCA(n\_components=2, random\_state=0).fit\_transform(X)

X\_filtered = X\_PCA[y < 15]

y\_filtered = y[y < 15]

print(f”X\_PCA[0]:\n{round(X\_PCA[0][0], 2), round(X\_PCA[0][1], 2)}")

plt\_show(X\_filtered, y\_filtered, "Метод главных компонент")

Изображение выглядит как снимок экрана, диаграмма, График, текст

Автоматически созданное описание

Рисунок 2.1 – Кластеризация методом главных компонент

Значение координат объекта с индексом 0 равно (-0.03, 0.03).

# 2.2 Метод стохастического вложения соседей с t-распределением

Для данного метода воспользуемся функцией *TSNE* из библиотеки *sklearn*:

X\_tsne = TSNE(n\_components=2, random\_state=0).fit\_transform(X)  
X\_filtered = X\_tsne[y < 15]  
plt\_show(X\_filtered, y\_filtered, "Стохастическое вложение соседей с t-распределением")  
print(f"X\_tsne[0]:\n{round(X\_tsne[0][0], 2), round(X\_tsne[0][1], 2)}")

После выполнения кода выше появится график с кластерами (рисунок 2.2).

Изображение выглядит как текст, снимок экрана, диаграмма, График

Автоматически созданное описание

Рисунок 2.2 – Кластеризация методом *TSNE*

Значение координат объекта с индексом 0 равно (-7.26, 15.65).

# 2.3 Сравнение двух методов

Из рисунков 2.1 и 2.2 можно сделать вывод то, что с помощью метода *TSNE* удалось визуализировать объекты на плоскости и объекты разных классов визуально разделимы, а вот про метод главных компонент такое сказать нельзя.

# 3 Методы кластеризации

# 3.1 Метод *K*-средних

Для данного метода реализуем класс *MyKMeans*:

class MyKMeans():

def \_\_init\_\_(self, n\_clusters=3, n\_iters=100):

self.n\_clusters = n\_clusters

self.n\_iters = n\_iters

def fit(self, X):

np.random.seed(0)

self.centers = np.random.uniform(low=X.min(axis = 0),

high=X.max(axis = 0),

size=(self.n\_clusters, X.shape[1]))

for it in range(self.n\_iters):

labels = self.predict(X)

new\_centers = np.array([X[labels == i].mean(axis=0) for i in range(self.n\_clusters)])

if np.all(self.centers == new\_centers):

print(f"Алгоритм сошёлся на {it} итерации")

break

self.centers = new\_centers

def predict(self, X):

labels = pairwise\_distances\_argmin(X, self.centers)

return labels

Сгенерируем данные для кластеризации и кластеризируем их с помощью *MyKMeans* используя гиперпараметры *n\_clusters*=3, *n\_iters*=100:

n\_samples = 1000

noisy\_blobs = datasets.make\_blobs(n\_samples=n\_samples,

cluster\_std=[1.0, 3.0, 0.5],

random\_state=0)

X, y = noisy\_blobs

cluster = MyKMeans()

cluster.fit(X)

print(f"Кластеризация объектов с гиперпараметром n\_iters=100:\n{cluster.predict(X)[:10]}")

print(f"Объект с индексом 1 относится к кластеру с индексом: {cluster.predict(X[1].reshape(1, -1))}")

k\_means100 = cluster.predict(X)

plt\_show(X, k\_means100, "Кластеризация при n\_iters=100")

После выполнения кода выше в консоли выведется информация о кластеризации (рисунок 2.3) и отобразится график на экране (рисунок 2.4).

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 3.1 – Информация о кластеризации методом *K*-средних с параметром *n\_iters* равным 100

Изображение выглядит как текст, снимок экрана, диаграмма, График

Автоматически созданное описание

Рисунок 3.2 – Кластеризация методом *K*-средних с параметром *n\_iters* равным 100

При изменении параметра *n\_iters* со 100 на 5, то кластеризация поменяется. На рисунке 2.5 показана информация о кластеризации, на рисунке 2.6 изображён график кластеризации.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 3.53 – Информация о кластеризации методом *K*-средних с параметром *n\_iters* равным 5

Изображение выглядит как текст, снимок экрана, диаграмма, График

Автоматически созданное описание

Рисунок 3.4 – Кластеризация методом *K*-средних с параметром *n\_iters* равным 5

При изменении параметра *n\_iters* со 100 на 5 у некоторых объектов изменилась и метка предсказываемого кластера и таких объектов было 17.

# 2.5 Метод *DBSCAN*

Для данного метода воспользуемся функцией *DBSCAN* из библиотеки *sklearn*:

clusters = DBSCAN(eps=0.5).fit\_predict(X)

print(f"Объект с индексом 1 принадлежит кластеру: {clusters[1]}")

print(f"Полученное количество кластеров равно {len(set(clusters) - {-1})} (без выбросов)")

print(f"Количество объектов, отнесенных к выбросам: {(clusters == -1).sum()}")

plt\_show(X, clusters, "DBSCAN")

На рисунке 2.7 изображён вывод в консоли, а на рисунке 2.8 изображён вывод графика на экране после выполнения кода.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 3.5 – Информация о кластеризации методом *DBSCAN*

Изображение выглядит как диаграмма, График

Автоматически созданное описание

Рисунок 3.6 – Кластеризация методом *DBSCAN*

# Вывод

В ходе работы были получены результаты разных методов снижения размерности и кластеризации. Метод *TSNE* показал себя лучше, чем метод *PCA*. При кластеризации метод *DBSCAN* самостоятельно определил количество кластеров и убрал шумы, но для него необходимо указать максимальное расстояние между объектами в кластере.