

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ
ИМ. ПЕТРА ВЕЛИКОГО

ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ И МЕХАНИКИ
КАФЕДРА "ПРИКЛАДНАЯ МАТЕМАТИКА"

ОТЧЁТ ПО
КУРСОВОЙ РАБОТЕ
ПО ДИСЦИПЛИНЕ
"МАТЕМАТИЧЕСКАЯ СТАТИСТИКА"

ВЫПОЛНИЛ СТУДЕНТ:
МАЛЬЦОВ ДМИТРИЙ ДМИТРИЕВИЧ
ГРУППА: 3630102/70401

ПРОВЕРИЛ:
К.Ф-М.Н., ДОЦЕНТ
БАЖЕНОВ АЛЕКСАНДР НИКОЛАЕВИЧ

САНКТ-ПЕТЕРБУРГ
2020 год

Содержание

	Стр.
1. Постановка задачи	4
2. Подготовка данных	4
3. Метод главных компонент	4
3.1. Описание метода	4
3.2. Использование сторонних пакетов	5
3.3. Сингулярное разложение матрицы	5
3.4. Реализация сингулярного разложения	6
3.5. Проверка сингулярного разложения	6
4. Кластеризация полученных данных	7
4.1. Используемые методы	7
4.2. Результаты кластеризации	7
5. Вывод	9
6. Список литературы	10
7. Приложение	10

Список иллюстраций

1	Пример исходных и очищенных данных	4
2	Доля объясненной дисперсии для каждой компоненты	5
3	Первые 10 элементов матриц главных компонент. Слева-полученные с помощью sklearn, справа-полученные через сингулярное разложение	6
4	Распределение данных полученных с помощью встроенного PCA и с помощью сингулярного разложения	7
5	Результат кластеризации с помощью KMeans	7
6	Результат кластеризации с помощью DBSCAN	8
7	Распределение исходных данных	9

1 Постановка задачи

Имеется датасет проб, содержащий информацию о концентрации определенных химических соединений в каждой из проб.

Цель работы: Отпределить, возможно ли разделить данные пробы на две группы.

Задачи: Построить метод главных компонент, кластеризовать исходные данные.

2 Подготовка данных

Данные представлены в виде двух .xlsx файлов.

Объединим эти файлы в один, предварительно удалив из них лишнюю информацию.

№ п.п.	проба	метан	этан	этилен	пропан	пропилен	н-бутан	бутен-1	н-бутилени-пентан	н-пентан			
		C _{H4}	C _{2H6}	C _{2H4}	C _{3H8}	C _{3H6}	i-C _{4H10}	n-C _{4H10}	i-C _{4H8}	i-C _{5H12}	n-C _{5H12}		
		концентрация, мг/л											
1	038	220.126	17.28648		6.289029	8.79385	3.733841	0.822172	3.092925	1.711174	1.875641	1.240593	1.177349
2	048	281.8113	35.96388	10.72783	15.47563	7.3475	1.03425	5.00975	2.706125	2.00225	1.955875	2.06375	
3	058	139.423	8.153831	2.553068	3.515057	1.253461	0.29306	1.409902	0.510188	1.07331	0.914508	0.483854	
4	068	258.4819	4.68786	1.223232	1.927446	0.292974	0.237912	0.906522	0.276552	1.04673	0.886374	0.206586	
5	088	155.6503	8.258578	2.511838	3.745818	1.398482	0.308424	1.304554	0.698214	1.163548	1.016862	0.478836	
6	098	151.7737	18.19258	5.1198	8.3286	3.09668	0.62832	2.85012	1.26868	1.33266	1.22962	1.28828	
7	108	113.2304	5.85057	1.661715	2.946075	1.307955	0.252945	0.98373	1.44408	0	2.122395	0.598475	
8	118	121.1258	6.46668	1.696035	3.25512	1.300035	0.386925	1.11639	1.09527	0	1.76022	0.36993	
9	138	166.4957	6.44355	1.7301	2.9334	1.29285	0.23175	1.0701	0.5886	1.2492	1.35255	0.3144	
10	148	192.9288	4.443108	1.16846	2.03426	0.985976	0.277796	0.739704	0.28046	0.791208	0.626336	0.316572	
11	158	188.8445	4.01643	1.05336	1.947165	0.854205	0.220275	0.701085	0.2409	1.263735	1.166955	0.464145	
12	178	757.0353	39.09136	10.27642	17.59156	7.84042	1.12374	5.03598	1.50766	2.5564	2.877	1.50066	
13	18-038_0	61.99882	2.0629	0.67892	0.41608	0.10038	0.21196	0.20062	0.05838	1.51424	1.28646	0.5411	
14	198	179.0472	4.3778	1.172675	2.1469	0.957775	0.343425	0.958125	0.566125	1.811075	1.691725	0.67585	
15	208	145.7293	3.569137	0.431574	1.953666	0.428428	0.250965	1.109108	0.64922	1.12827	0.722436	0.23738	
16	218	167.5794	4.81812	1.175042	1.909818	0.816804	0.194382	0.64719	0.429462	1.167372	1.458	0.394794	
17	228	65.64781	5.18703	1.33829	2.436635	1.01195	0.23384	1.004735	0.76294	1.78229	1.174565	0.38184	
18	238	67.56704	4.87616	1.35632	2.18784	0.8232	0.19904	0.76624	0.43048	1.47936	1.56384	0.25776	
19	248	66.9596	4.742	1.2136	2.2058	0.9268	0.389	1.158	0.6128	2.9144	2.5962	0.5612	
20	258	285.0981	20.10242	3.594354	6.58681	2.735332	0.254988	2.508246	1.069532	1.645272	2.37006	1.144692	

Рис. 1: Пример исходных и очищенных данных

После этого можно считать данные с помощью пакета pandas в Python.

3 Метод главных компонент

3.1 Описание метода

Principal component analysis, PCA - один из основных способов уменьшить размерность данных, потеряв наименьшее количество информации. PCA применяется к данным, записанным в виде матрицы. Суть метода заключается в том, что из исходной матрицы мы получаем матрицу главных компонент меньшего размера, где каждый столбец(главная компонента) объясняет дисперсию исходных данных. Чем больше главных компонент мы берем, тем меньше данных мы теряем, и наоборот.

Перед использованием метода данные необходимо отцентрировать и отнормировать. В результате применения этого метода мы приходим от большого количества переменных к новому представлению, размерность которого значительно меньше.

Метод сводится к вычислению сингулярного разложения исходной матрицы либо к вычислению собственных векторов и собственных значений ковариационной матрицы исходных данных.

В данной работе было реализовано сингулярное разложение матрицы, затем правильность разложения проверилось с помощью метода PCA, реализованным в программном пакете sklearn.decomposition в python

3.2 Использование сторонних пакетов

Применим РСА, для двух главных компонент. Построим график долей объясненной дисперсии для каждой компоненты.

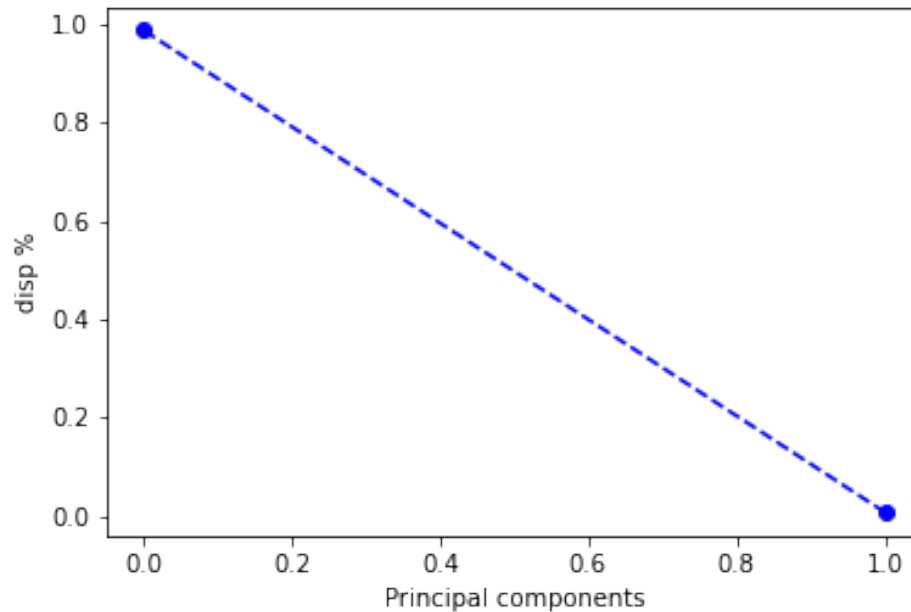


Рис. 2: Доля объясненной дисперсии для каждой компоненты

Из графика видно, что первая компонента практически полностью объясняет дисперсию исходных данных, поэтому брать больше двух компонент не имеет никакого смысла. Ниже, при реализации МГК через сингулярное разложение мы тоже будем брать первые две главные компоненты.

3.3 Сингулярное разложение матрицы

Пусть матрица M порядка $m \times n$ состоит из элементов из поля K , где K — поле вещественных чисел.

Неотрицательное вещественное число σ называется сингулярным числом матрицы M тогда и только тогда, когда существуют два вектора единичной длины $u \in K^m$ и $v \in K^n$ такие, что: $Mv = \sigma u$ и $M^*u = \sigma v$. Такие векторы u и v называются левым сингулярным вектором и правым сингулярным вектором, соответствующим сингулярному числу σ . Сингулярным разложением матрицы M порядка $m \times n$ является разложение следующего вида: $M = U\Sigma V^*$, где Σ — матрица размера $m \times n$ с неотрицательными элементами, у которой элементы, лежащие на главной диагонали, являются нулевыми), а матрицы U (порядка m) и V (порядка n) — это две унитарные матрицы, состоящие из левых и правых сингулярных векторов соответственно (а V^* — это сопряжённо-транспонированная матрица к V).

3.4 Реализация сингулярного разложения

Для начала нормализуем исходные данные, при применении PCA из `sklearn.decomposition` мы этого не делали, так как в данном методе нормализация уже реализована.

Затем проведем полное сингулярное разложение. Полное в том смысле, что матрица vt содержит все столбцы исходной матрицы.

Далее проверим правильность разложения: для этого поэлементно вычтем из матрицы $X_{svd} = U\Sigma V^*$ элементы исходной матрицы, возведем в квадрат и подсчитаем сумму полученных элементов. В итоге мы получили число близкое к нулю, значит разложение было произведено верно.

3.5 Проверка сингулярного разложения

Теперь проверим полученную матрицу главных компонент: для этого умножим исходную нормализованную матрицу данных на транспонированную матрицу двух первых столбцов матрицы главных компонент V^* и сравним со значениями, полученными при применении PCA из `sklearn.decomposition`.

		probe	
array([[-123.23513302, 1.53928913], [-60.11642204, 16.64336107], [-204.58894327, -3.20480501], [-86.24523388, -16.82728194], [-188.39762025, -4.3040104], [-151.46841413, 4.1445747], [-230.87958675, -3.32280025], [-222.95797761, -3.32237075], [-177.75207393, -7.19491622], [-151.59266909, -11.64327682]])		03B	-123.235133 1.539289
		04B	-60.116422 16.643361
		05B	-204.588943 -3.204805
		06B	-86.245234 -16.827282
		08B	-188.397620 -4.304010
		09B	-151.468414 4.144575
		10B	-230.879587 -3.322800
		11B	-222.957978 -3.322371
		13B	-177.752074 -7.194916
		14B	-151.592669 -11.643277

Рис. 3: Первые 10 элементов матриц главных компонент. Слева-полученные с помощью `sklearn`, справа-полученные через сингулярное разложение

Также построим полученные матрицы и сравним визуально. Видим, что матрицы и графики совпали, значит разложение произведено верно и МГК действительно сводится к сингулярному разложению.

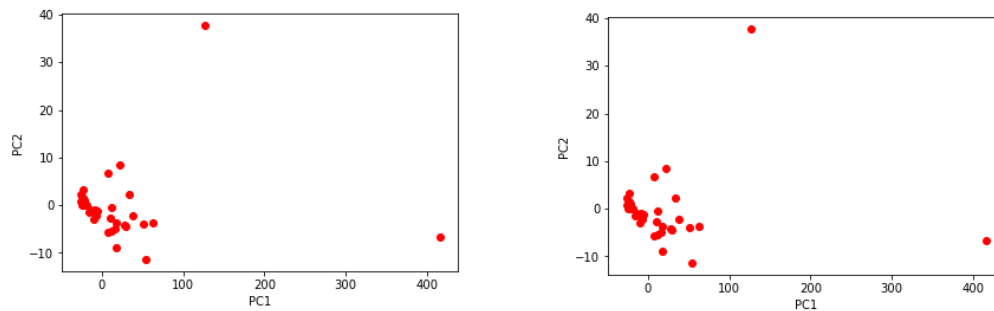


Рис. 4: Распределение данных полученных с помощью встроенного PCA и с помощью сингулярного разложения

4 Кластеризация полученных данных

4.1 Используемые методы

Для кластеризации данных используем два метода: классический KMeans и DBSCAN из `sklearn.cluster`

4.2 Результаты кластеризации

KMeans сумел кластеризовать данные на две группы, в то время как DBSCAN не смог разделить данные. произведено верно и МГК действительно сводится к сингулярному разложению.

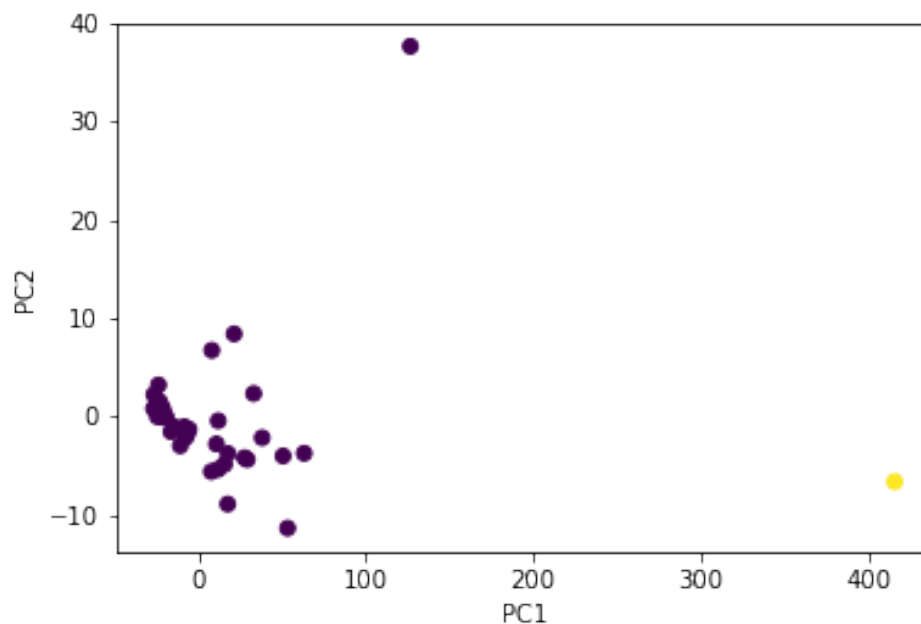


Рис. 5: Результат кластеризации с помощью KMeans

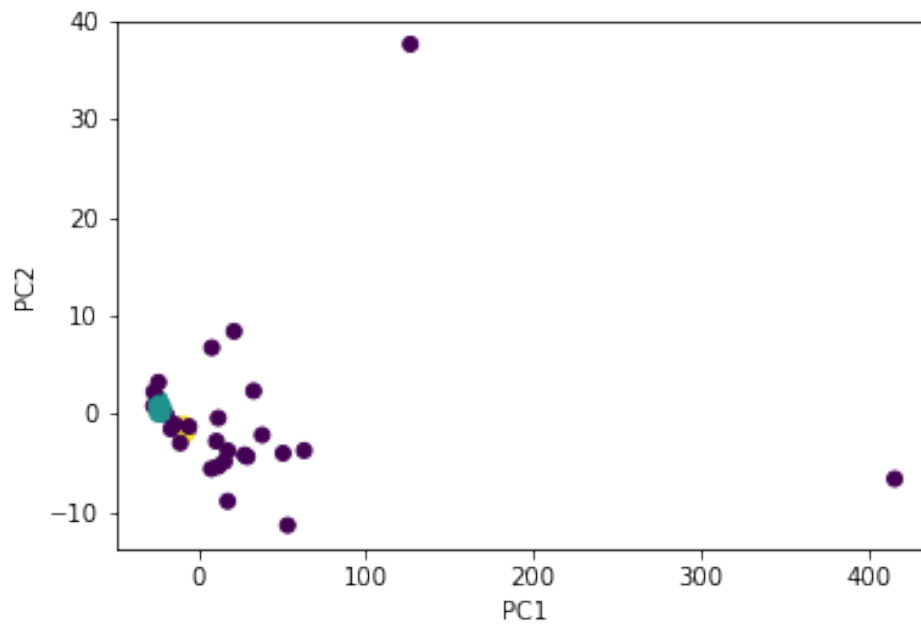


Рис. 6: Результат кластеризации с помощью DBSCAN

Значения второй группы, полученной с помощью KMeans можно скорее отнести к выбросам, чем к отдельному кластеру. Таким образом, делаем вывод, что исходные данные не получилось разделить по двум группам.

Действительно, для более наглядной демонстрации невозможности разделения данных на две группы построим все графики зависимостей параметров исходных данных друг от друга.

Видим, что нигде нет и намека на разделение на группы:

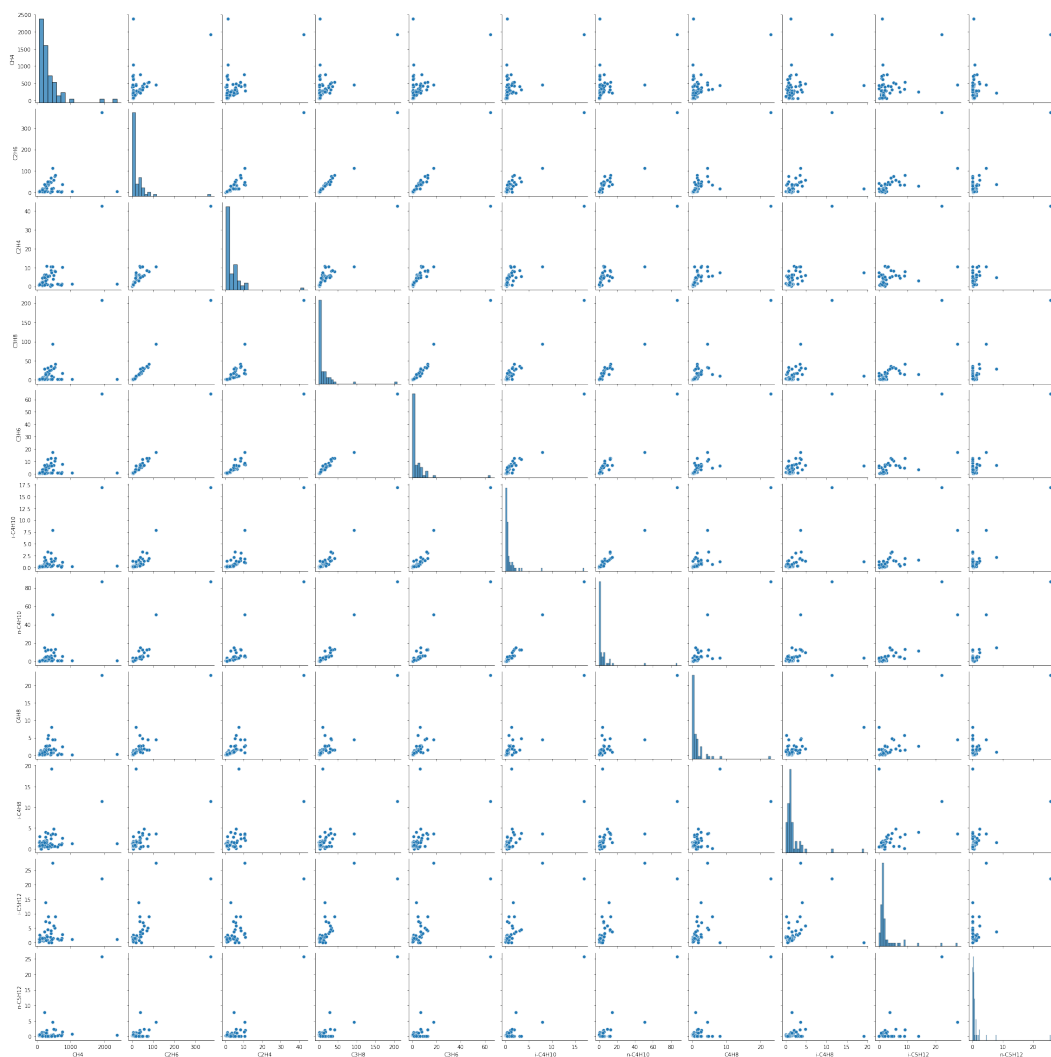


Рис. 7: Распределение исходных данных

5 Вывод

При помощи PCA удалось уменьшить размерность до двумерной, так как было обнаружено, что основной вклад в дисперсию исходных данных оказывают первые две компоненты.

Было продемонстрировано, что МГК сводится к сингулярному разложению.

Тем ни менее, исходные данные оказались слишком похожи, и их не удалось кластеризовать на две группы ни одним из использованных алгоритмов кластеризации.

6 Список литературы

- [1]Метод главных компонент
- [2]Модуль numpy
- [3]Модуль matplotlib
- [4]Документация sklearn PCA
- [5]Нормализация и стандартизация в Python
- [6]Принцип работы МГК

7 Приложение

Код работы