МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Машинное обучение»

Тема: Понижение размерности пространства признаков

Студент гр. 6304 Доброхвалов М. О.

Преподаватель Жангиров Т. Р.

Санкт-Петербург

2020

Цель работы

Ознакомиться с методами понижения размерности данных из библиотеки Scikit Learn

Загрузка данных

1. Скачан и загружен датасет в датафрейм, выделены описательные признаки и признак отображающий класс. Выполнена нормировка данных к интервалу [0 1].

```
df = pd.read_csv('datasets/glass.csv')
var_names = list(df)
labels = df.to_numpy('int')[:,-1]
data = df.to_numpy('float')[:,:-1]
data = preprocessing.minmax_scale(data)
```

2. Выполнено построение диаграмм рассеяния данных(рис. 1) и гистограммы дисперсий признаков.

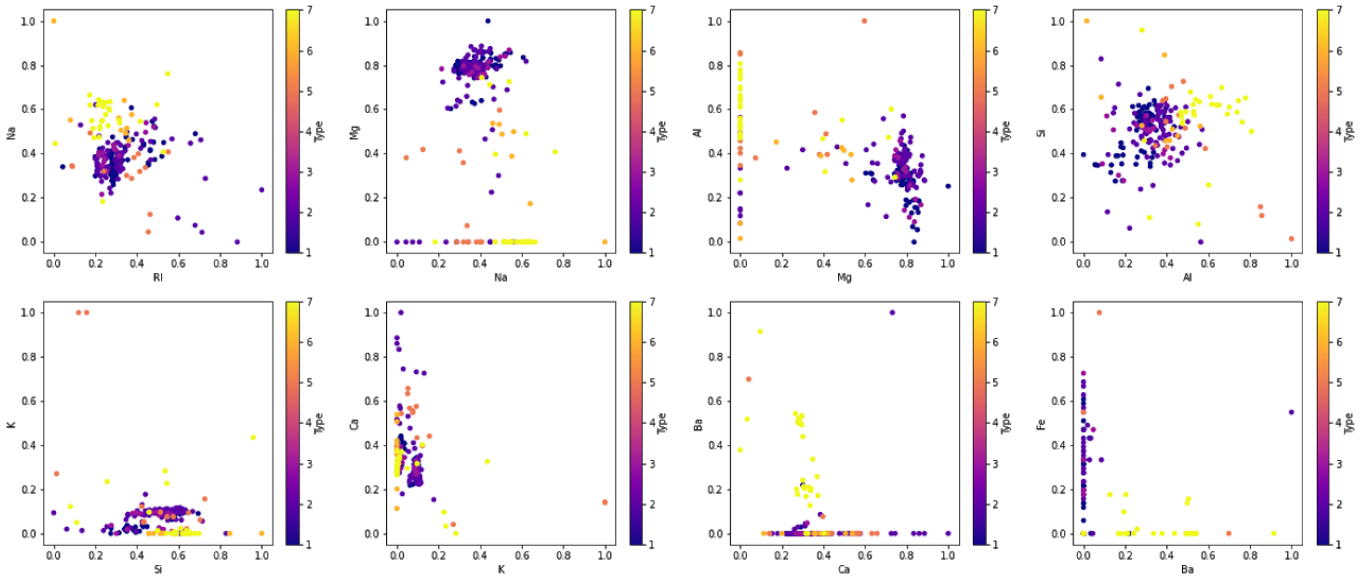

Рис. 1 — Диаграммы рассеяния исходных данных

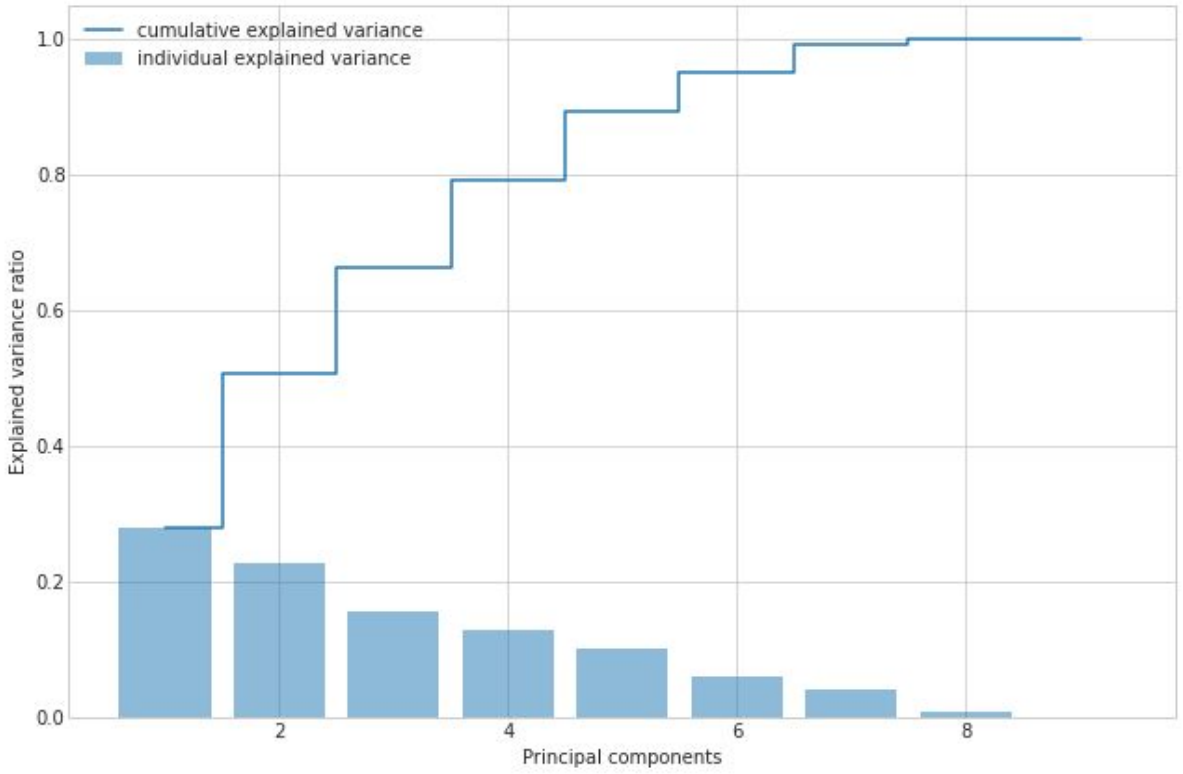

Рис.2 — Гистограмма дисперсии исходных данных

Метод главных компонент

1. Выполнено понижение размерности пространства до размерности 2.

   Была построена диаграмма рассеяния (рис. 3)

   Объясненная дисперсия (рис. 4): [0.454 0.180]

   Собственные числа: [5.105 3.212]

   Данные компоненты объясняют 0.634 общей дисперсии.

На основе результатов можно сделать вывод, что двух компонент недостаточно, чтобы объяснить достаточно большую долю дисперсии. Также можно сделать предположение, что между данными нет явной линейной взаимосвязи.

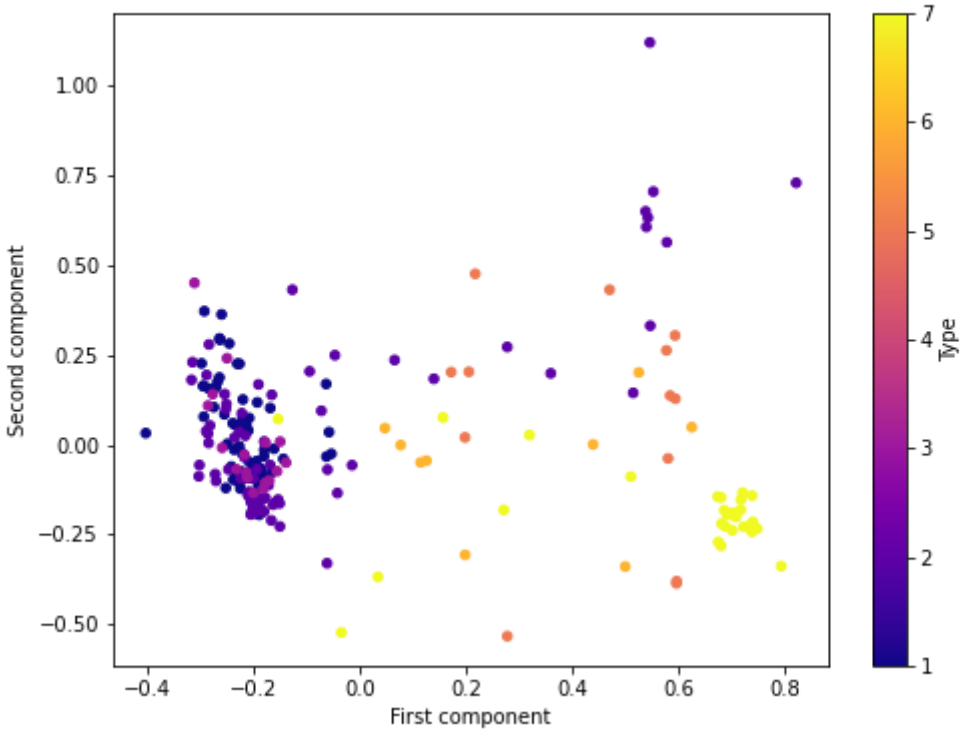

Рис. 3 — Диаграмма рассеяния двух компонент

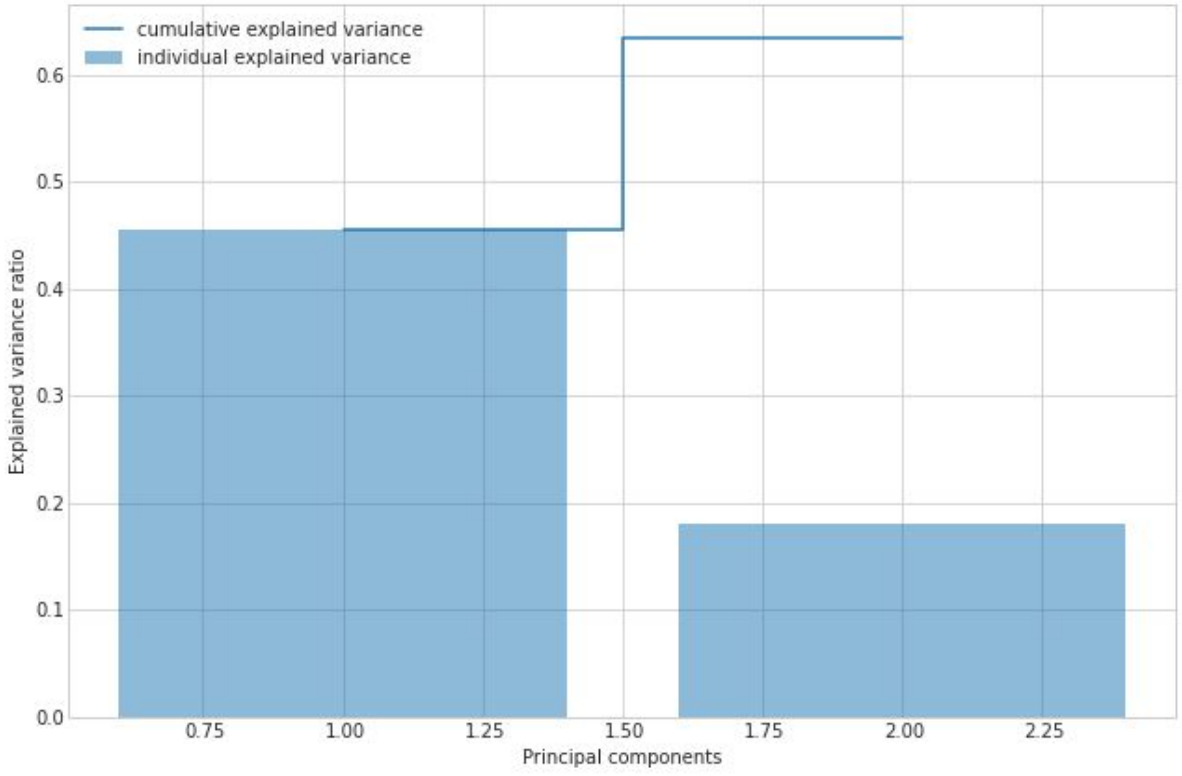

Рис. 4 — Гистограмма объясненной дисперсии 2 компонент PCA

2. Было выполнено исследование зависимости объясненной дисперсии от количества компонент (рис. 5)

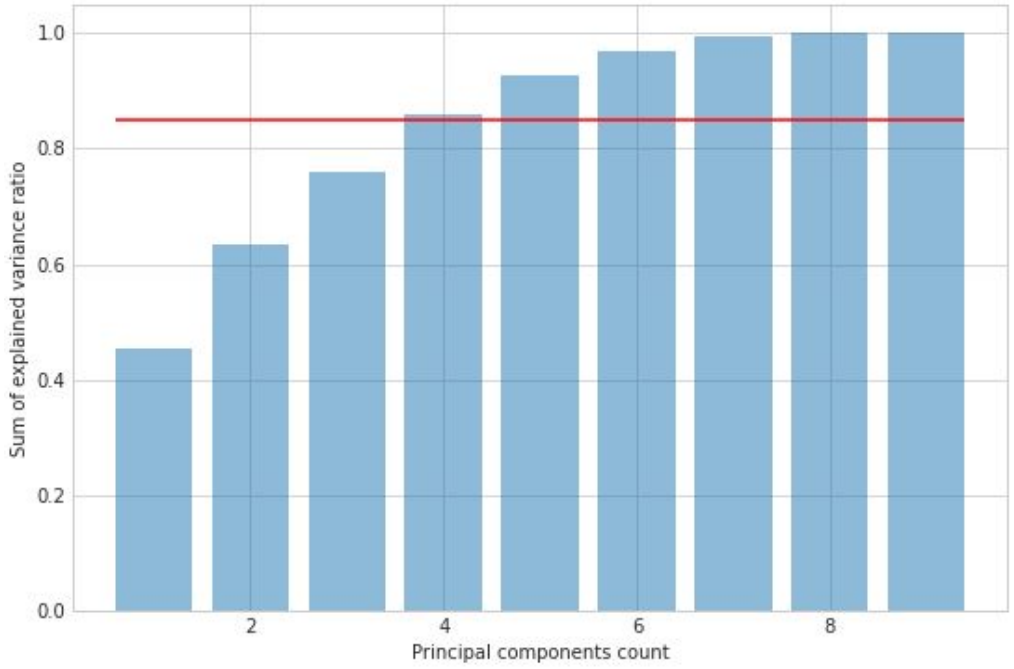

Рис. 5 — Зависимость объясненной дисперсии от количества компонент

Таким образом, минимальное количество компонент, при котором объясненная дисперсия не менее 85% - 4.

3. С помощью метода *inverse_transform* были восстановлены исходные данные. Выполнено сравнение среднего значения и дисперсии (табл. 1). Также построена парная диаграмма рассеивания для признаков (RI, Na) и (RI, Mg) в исходных и восстановленных данных (рис. 6-7)

На основании таблицы можно сделать вывод о том, что средние значение признака восстанавливается полностью. Однако, некоторые признаки при обратном преобразовании потеряли достаточно большую долю СКО - до 58.3%. Вероятно, это связано с выбором главных компонент.

Таблица 1 — Сравнение среднего значения и дисперсии исходных и восстановленных данных

	Среднее			СКО			
	Исходные	Восстановленные	Разность	Исходные	Восстановленные	Разность	Доля потери
RI	0,317	0,317	0,000	0,133	0,130	0,003	0,025
N a	0,403	0,403	0,000	0,123	0,069	0,054	0,440
Mg	0,598	0,598	0,000	0,321	0,321	0,001	0,002
Al	0,360	0,360	0,000	0,156	0,136	0,020	0,126
Si	0,507	0,507	0,000	0,138	0,130	0,009	0,063
K	0,080	0,080	0,000	0,105	0,044	0,061	0,583
Ca	0,328	0,328	0,000	0,132	0,129	0,004	0,027
Ba	0,056	0,056	0,000	0,158	0,132	0,026	0,162
Fe	0,112	0,112	0,000	0,191	0,189	0,002	0,013

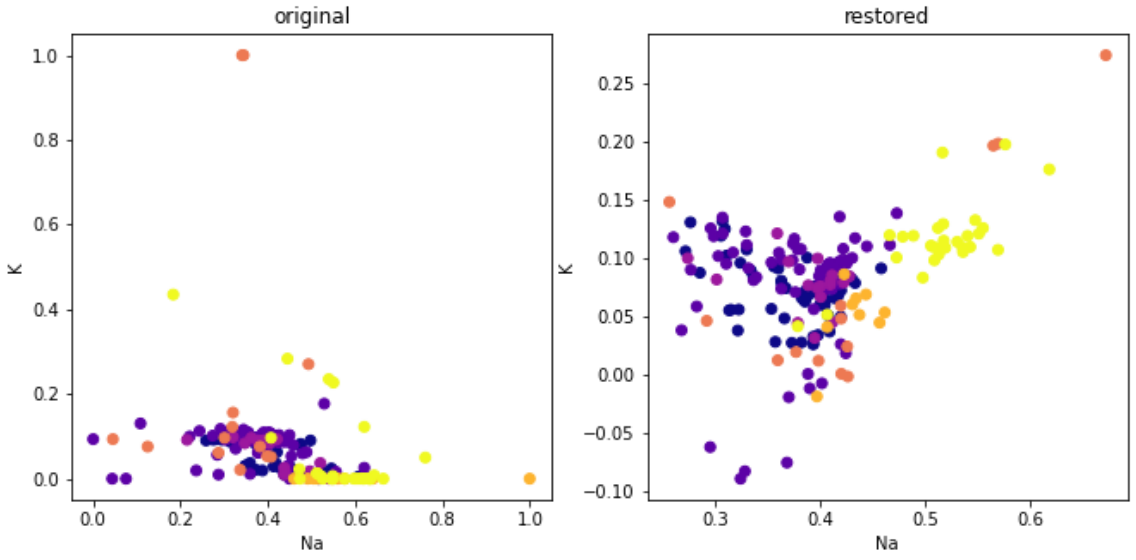

Рис. 6 — Диаграмма рассеяния исходных и восстановленных данных (K, Na)

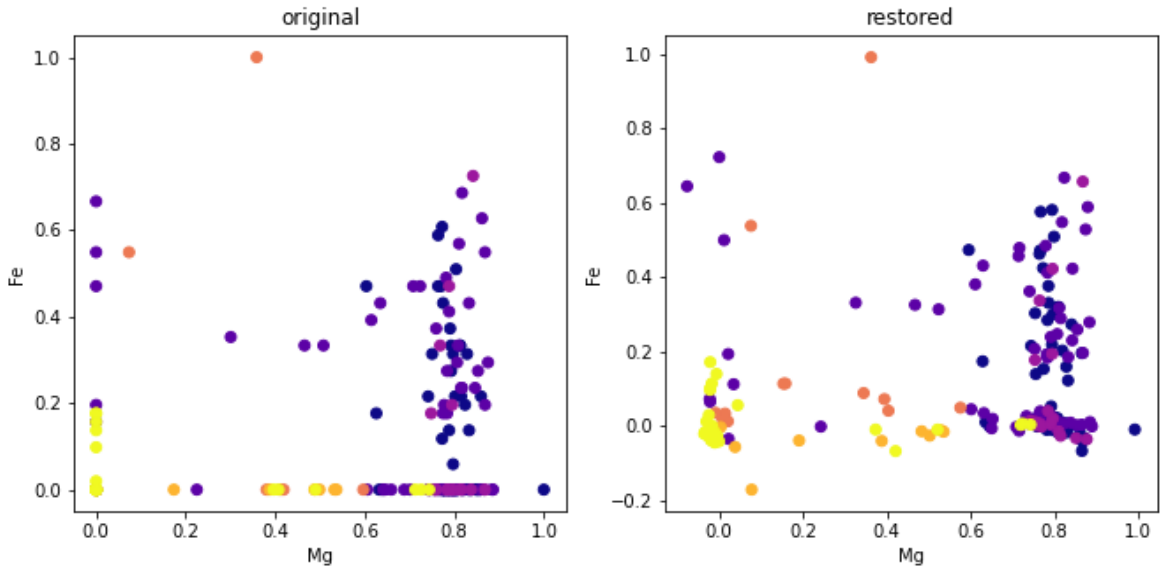

Рис. 7 — Диаграмма рассеяния исходных и восстановленных данных (K, Na)

4. Было выполнено сравнение результатов работы алгоритма анализа главных компонент при различном значении параметра *svd_solver*.

Параметром по умолчанию в реализации sklearn является “auto”. В зависимости от размерности входных данных, а также необходимого количества компонент, которые надо выделить, выбирается один из двух методов: “full” или “randomized”.

При применении различных алгоритмов выбора сингулярных чисел (варьировании параметра *svd_solver*) результирующие значения практически не отличались (рис. 8). Различия можно объяснить точностью представления чисел с плавающей точкой и их арифметикой. Вероятно, сходство связано с небольшим размером выборки.

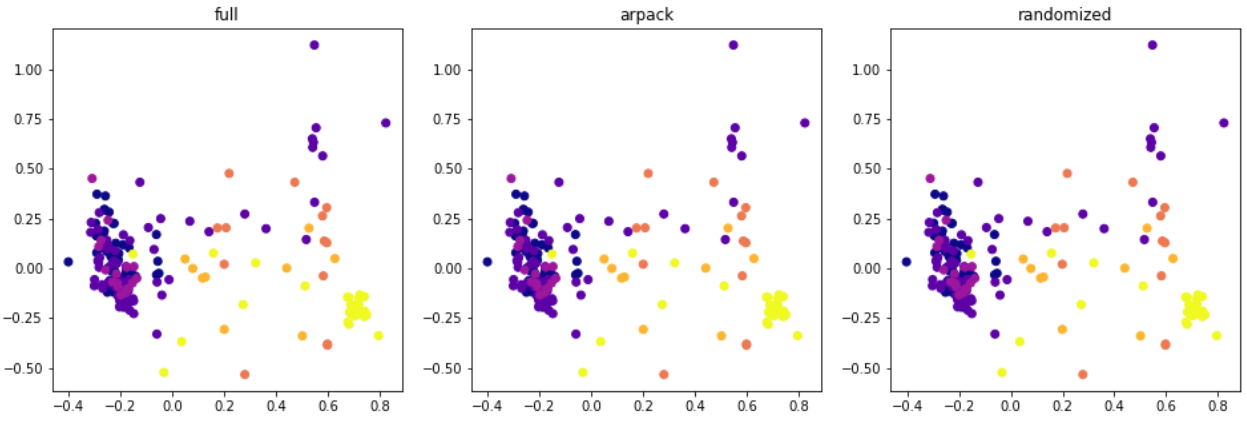

Рис. 8 — Диаграмма рассеяния первых двух главных компонент с различными параметрами svd_solver

Модификации метода главных компонент

KernelPCA пытается описать данные одной из следующих зависимостей: линейная, полиномиальная, функция радиального вида (RBF), сигмоида, косинус (cosine). Также имеется возможность использовать подсчитанное самостоятельно ядро (указав значение *precomputed* параметра *kernel*).

- Линейное ядро:

$$k(x, y) = x^\top y$$

- полиномиальное ядро:

$$k(x, y) = (\gamma x^\top y + c_0)^d$$

- RBF:

$$k(x, y) = \exp(-\gamma \|x - y\|^2)$$

- сигмоида:

$$k(x, y) = \tanh(\gamma x^\top y + c_0)$$

- косинус:

$$k(x, y) = \frac{xy^\top}{\|x\| \|y\|}$$

KernelPCA ведет себя аналогично PCA, если параметр установить kernel “linear”. Таким образом в поле *lambdas_* класса *KernelPCA* пос настройки на данных *data* будут хранится собственные числа

[26.060, 10.320, 7.2560, 5.620],

которые являются квадратами сингулярных чисел, хранящихся в поле *singular_values* класса *KernelPCA*

[5.105, 3.212, 2.694, 2.371]

На основании таблицы 2 можно сделать вывод о том, что на данной выборке объясненная дисперсия примерно одинакова для любого ядра.

Таблица 2 — Сравнение ядер KernelPCA

Ядро	PC1	PC2	PC3	PC4	Сумма	Объясненная дисперсия
linear	26,060	10,320	7,256	5,620	49,257	0.859
poly	10,918	4,319	3,119	2,368	20,724	0.833
rbf	5,351	2,018	1,496	1,111	9,976	0.831
sigmoid	1,006	0,400	0,274	0,216	1,896	0.862
cosine	18,314	6,475	4,696	3,578	33,064	0.860

SparsePCA

Находит набор разреженных компонент, которые могут оптимально восстановить данные. SparsePCA позволяет использовать один из двух методов: наименьшая угловая регрессия (*least angle regression*) или координатный спуск (*coordinate descent*) для решения задачи Lasso (метод регрессионного анализа, который выполняет как выбор переменных, так и регуляризацию, чтобы повысить точность прогнозирования и интерпретируемость создаваемой статистической модели)

Применения разных методов не дает внешне различимых отличий (рис. 9). При этом получаются одинаковые компоненты. Однако варьирование параметра *alpha*, который влияет на разреженность данных

дает видимые различия (рис. 10). Также получаются различные компоненты (таб. 3).

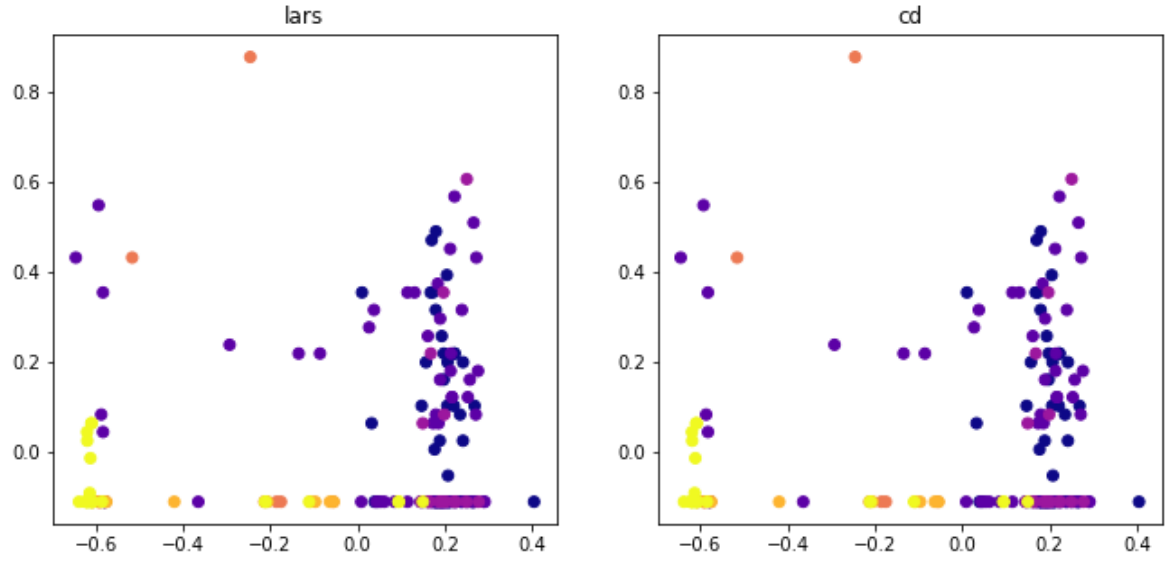

Рис. 9 — Диаграмма рассеяния первых двух главных компонент с использованием различных методов

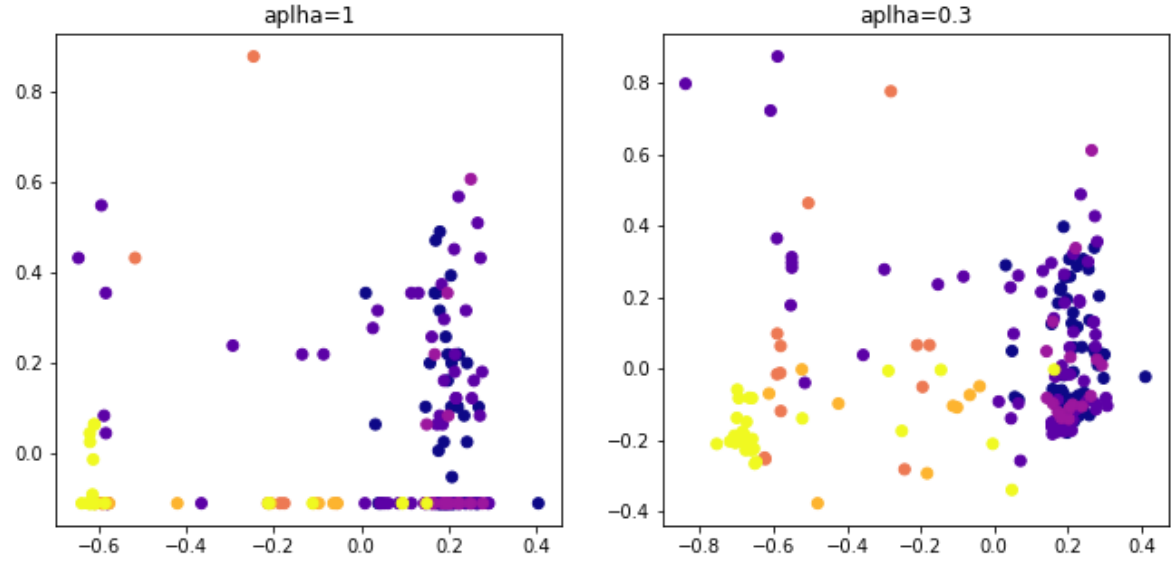

Рис. 10 — Диаграмма рассеяния первых двух главных компонент с варьировании параметра *alpha*

Таблица 3 — Главные компоненты SparsePCA при различных α

α = 1	PC1	0	0	0,998	-0,037	0	0	0	-0,05	0
	PC2	0	0	0	0	0	0	0	0	1
α = 0.3	PC1	0	-0,044	0,946	-0,203	0	0	-0,109	-0,224	0
	PC2	0,367	-0,148	0	-0,163	-0,144	0	0,328	0	0,83

Факторный анализ

1. Было выполнено понижение размерности с использованием факторного анализа

```
transformer = FactorAnalysis(n_components=2)
fa_2 = transformer.fit_transform(data)
```

2. Между результатами PCA и FA есть значимые различия в результирующем виде данных (рис. 11)

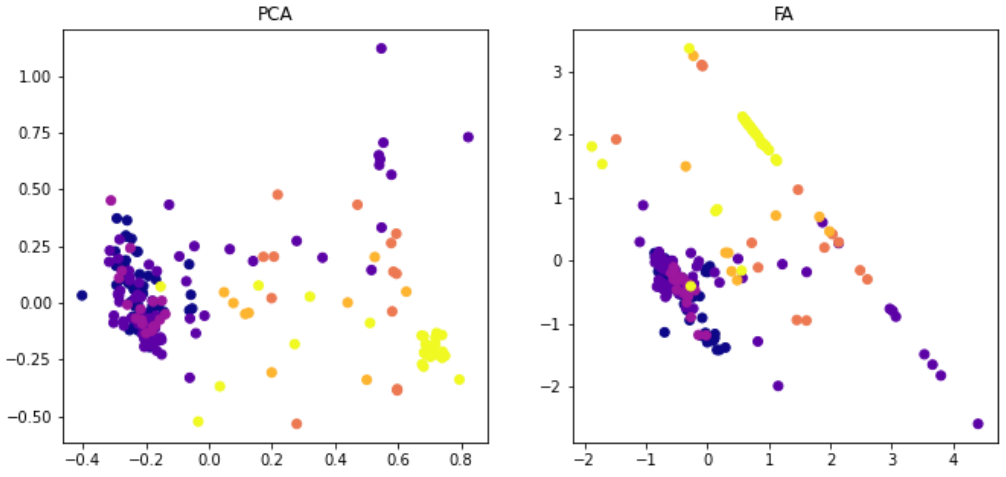

Рис. 11 — Диаграмма рассеяния результатов PCA и FA

3. Различия PCA и FA:
    a. Компоненты PCA объясняют максимальную дисперсию, в то время как факторный анализ объясняет ковариацию данных.
    b. Компоненты PCA полностью ортогональны друг другу, тогда как факторный анализ не требует, чтобы факторы были ортогональными.
    c. Компонент PCA представляет собой линейную комбинацию наблюдаемой переменной, тогда как в FA наблюдаемые переменные представляют собой линейные комбинации ненаблюдаемой переменной или фактора.
    d. PCA - это своего рода метод уменьшения размерности, тогда как факторный анализ - метод поиска скрытых переменных.

Вывод

Были изучены методы понижения размерности данных Factor Analysis и Principal component analysis из библиотеки Scikit Learn.

В ходе работы было выяснено, что разное количество компонент в PCA объясняет разное количество дисперсии данных.

KernelPCA используется для поиска нелинейных зависимостей в данных. Однако на предложенных данных объясненная дисперсия была около 85% независимо от ядра. SparsePCA выгодно использовать на данных большого размера.

Также был изучен факторный анализ и его различия с PCA.