**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. 8304 |  | Кирьянов Д.И. |
| Преподаватель |  | Жангиров Т. Р. |

Санкт-Петербург

2021

**Цель работы**

Ознакомиться с методами понижения размерности данных из библиотеки Scikit Learn.

**Ход работы**

**1. Загрузка данных**

Загружен датасет в датафрейм и разделены данные на описательные признаки и признак отображающий класс. Проведена нормировка данных к интервалу [0, 1]. Построены диаграммы рассеяния для пар признаков.

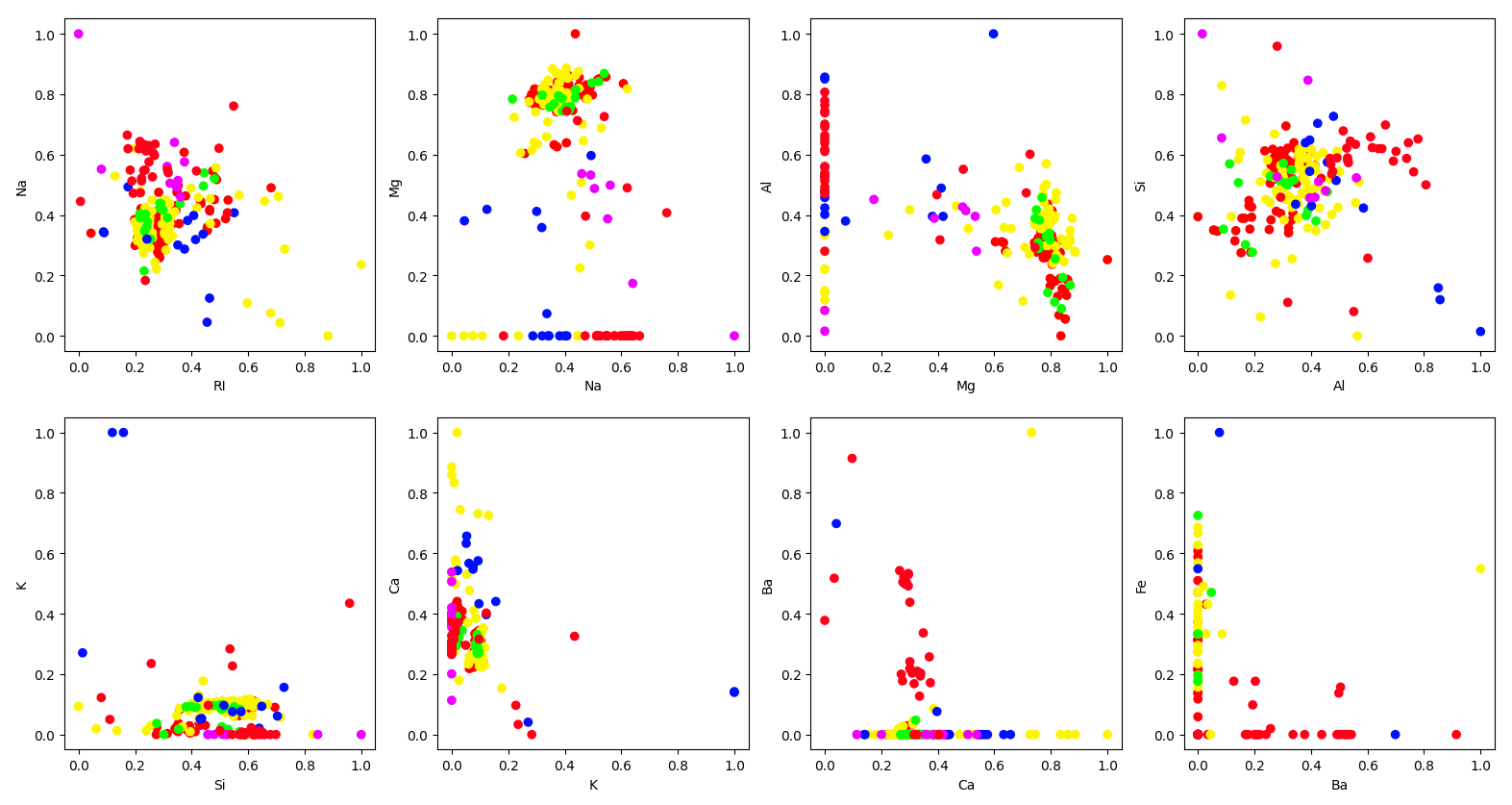


Рисунок 1 – Диаграммы рассеяния.

Цвета соответствуют HSV colormap, таким образом: 1 – красный, 2 – желтый, 3 – зеленый, 5 – синий, 6 – розовый, 7 – красный.

**2. Метод главных компонент**

Используя метод главных компонент (PCA) проведено понижение размерности пространства до размерности 2.

Выведены значения объясненной дисперсии в процентах и собственные числа, соответствующие компонентам.



Рисунок 2 – Полученные значения.

Построена диаграмма рассеяния после метода главных компонент.

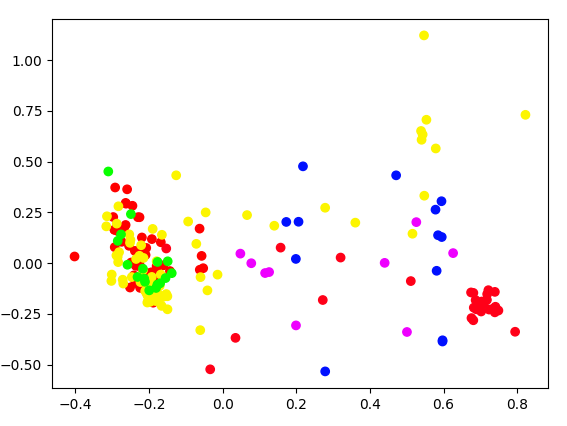


Рисунок 3 – Диаграмма рассеяния после метода главных компонент.

Изменяя количество компонент, определено количество, при котором компоненты объясняют не менее 85% дисперсии данных. Полученный результат 4 и более.

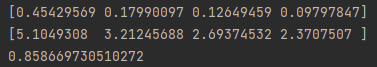


Рисунок 4 – Полученные значения.

Используя метод inverse\_transform восстановлены данные, построены диаграммы рассеяния.

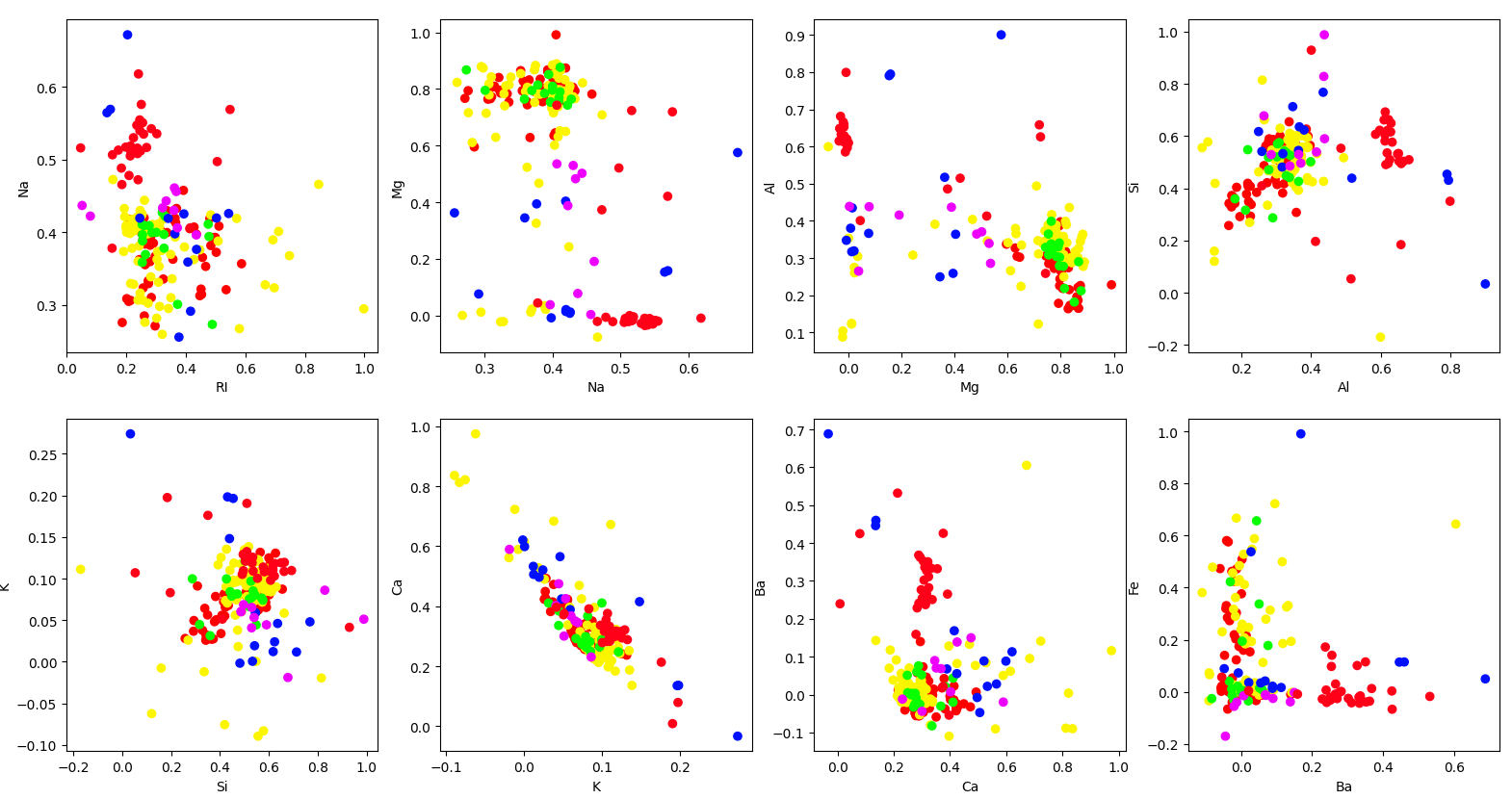


Рисунок 5 – Диаграммы рассеяния после восстановления данных.

Различия объясняются тем, что восстанавливается в нашем случае 85% данных.

Исследован метод главных компонент при различных параметрах svd\_solver.

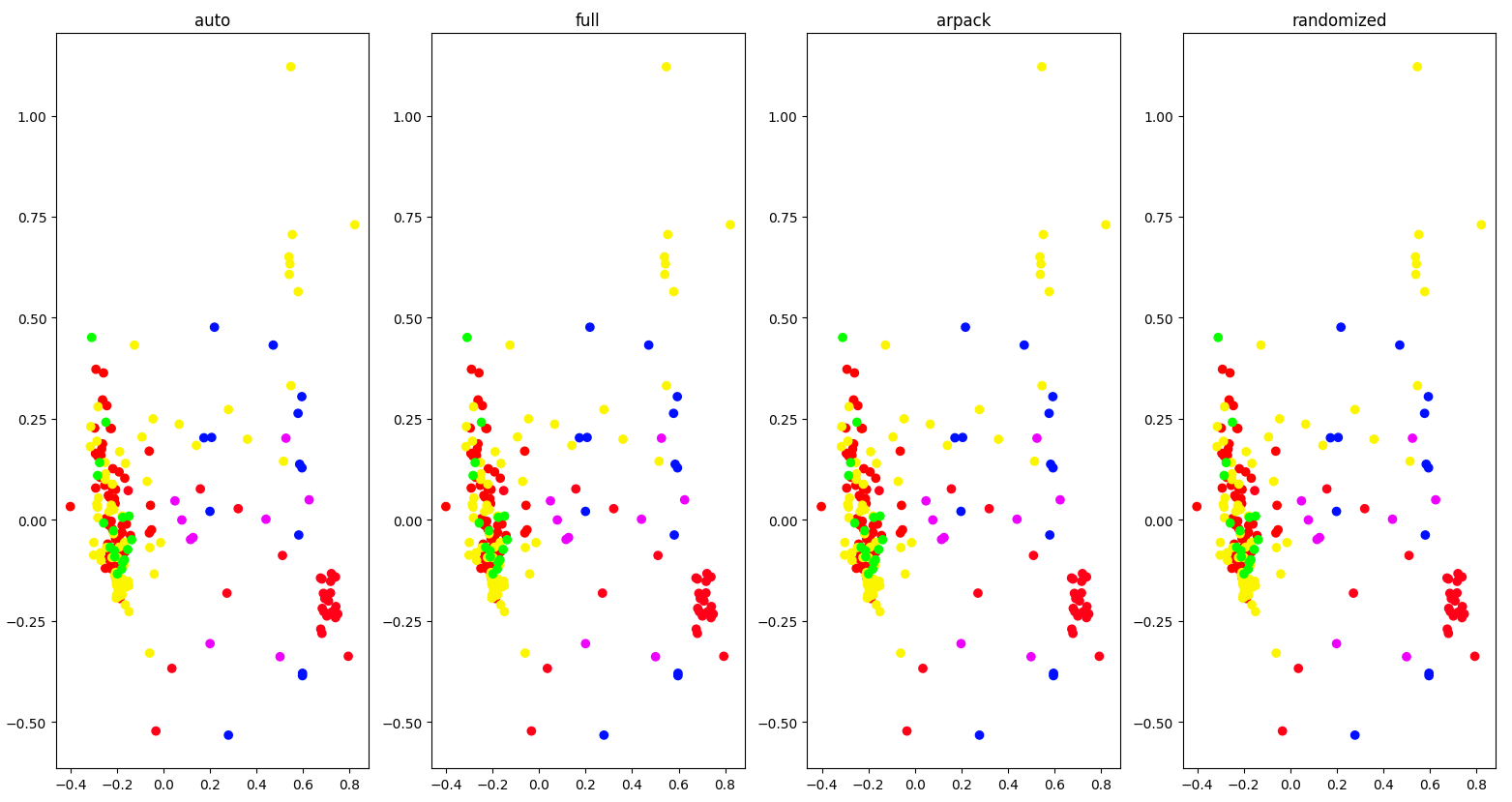


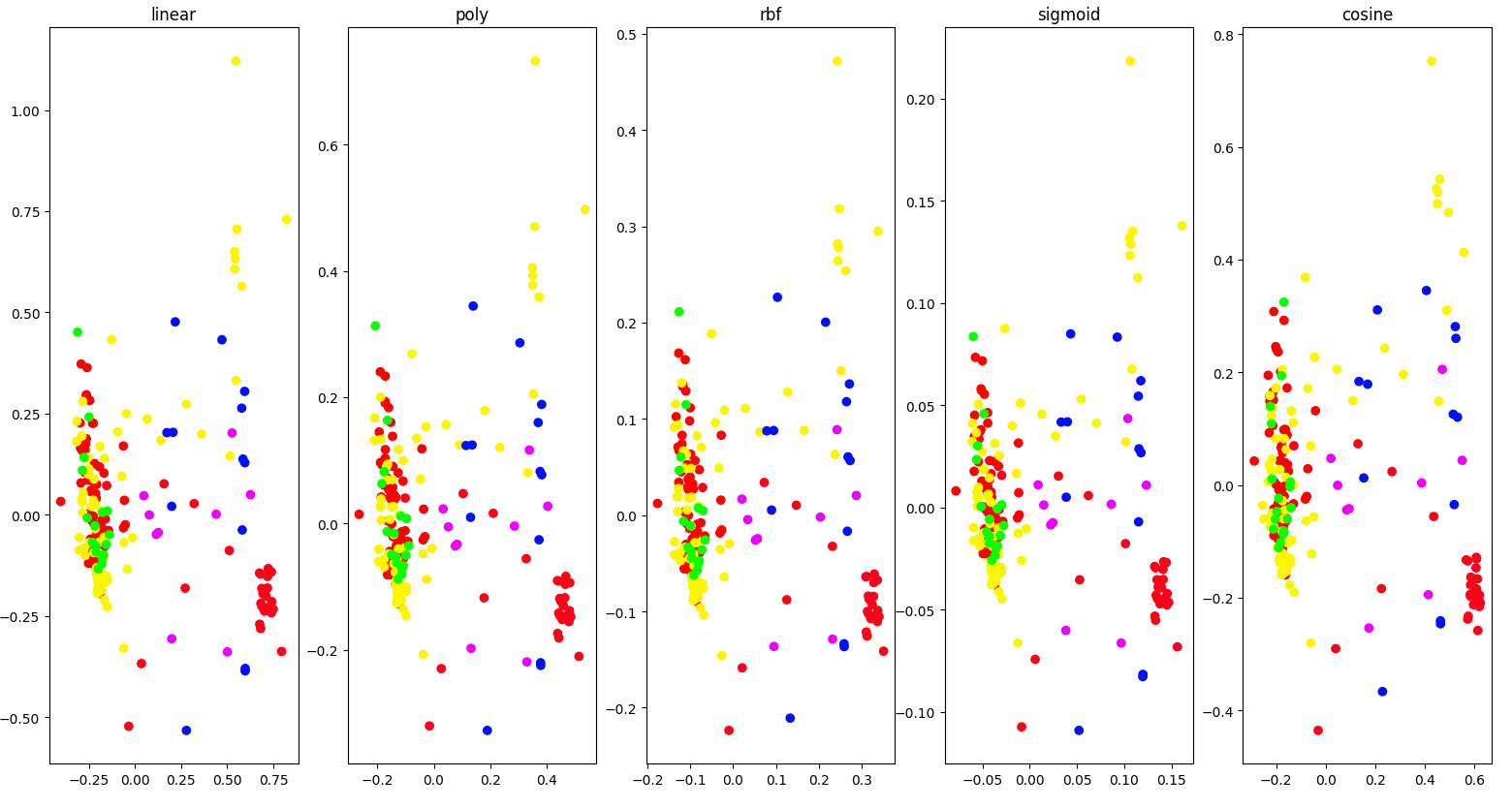
Рисунок 6 – Диаграммы рассеивания при различных значениях svd\_solver.

*If auto :  
 The solver is selected by a default policy based on `X.shape` and  
 `n\_components`: if the input data is larger than 500x500 and the  
 number of components to extract is lower than 80% of the smallest  
 dimension of the data, then the more efficient 'randomized'  
 method is enabled. Otherwise the exact full SVD is computed and  
 optionally truncated afterwards.  
If full :  
 run exact full SVD calling the standard LAPACK solver via  
 `scipy.linalg.svd` and select the components by postprocessing  
If arpack :  
 run SVD truncated to n\_components calling ARPACK solver via  
 `scipy.sparse.linalg.svds`. It requires strictly  
 0 < n\_components < min(X.shape)  
If randomized :  
 run randomized SVD by the method of Halko et al.*

Для рассмотренных данных параметр svd\_solver не имеет значения.

**3. Модификации метода главных компонент**

По аналогии с PCA исследован KernelPCA для различных параметров kernel.



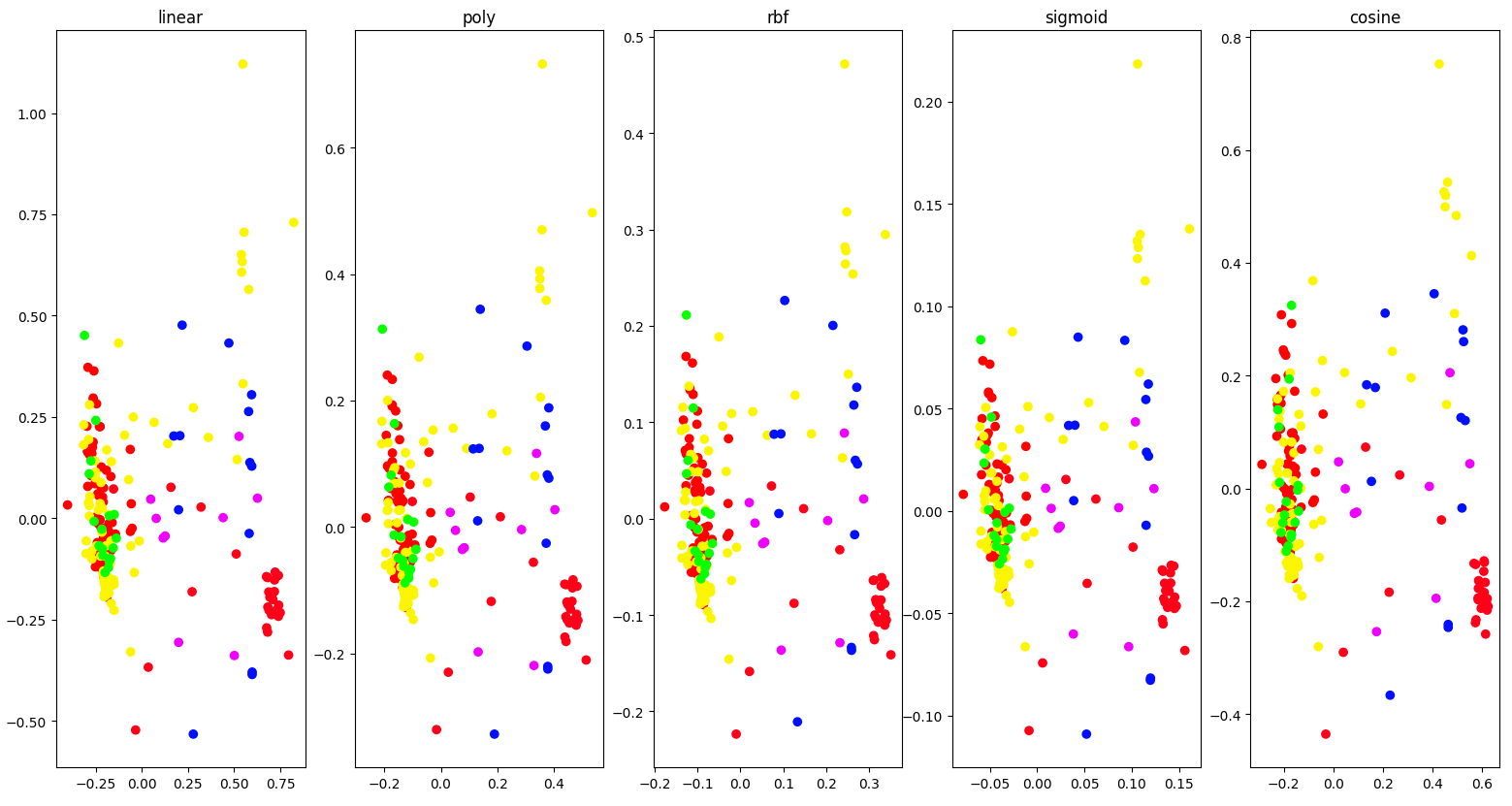


Рисунок 7 – Исследование параметра kernel

При параметре kernel = linear KernelPCA работает также, как PCA.

Аналогично исследован SparcePCA для различных параметров alpha.

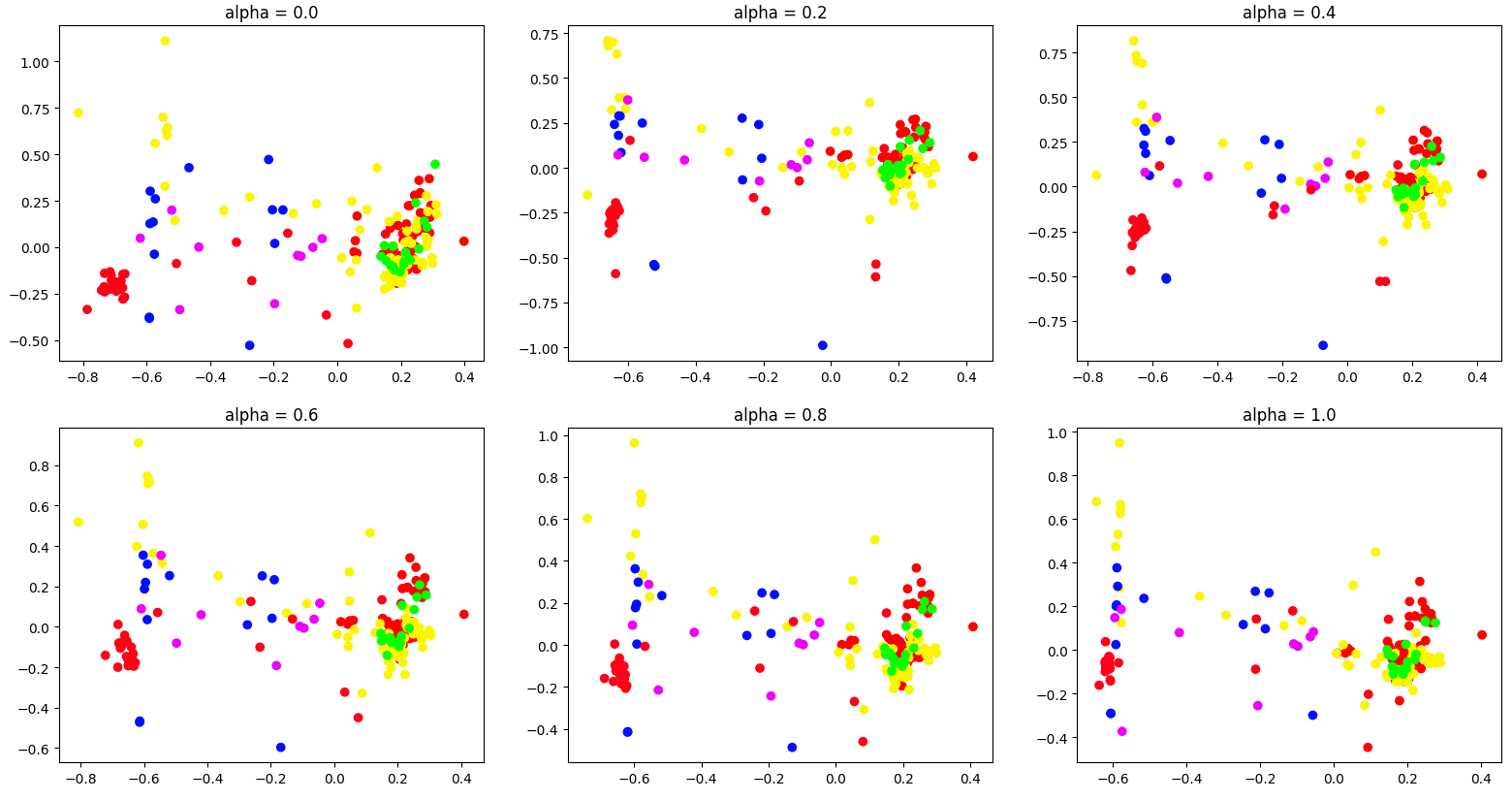


Рисунок 8 – Исследование параметра alpha

При параметре alpha = 0 SparcePCA работает также, как PCA.

**4. Факторный анализ**

Проведено понижении размерности используя факторный анализ FactorAnalysis.

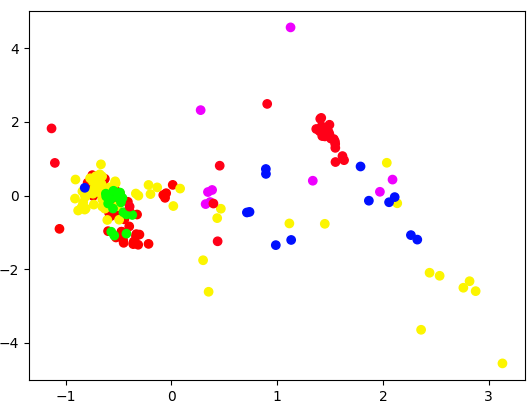


Рисунок 9 - Диаграммы рассеивания факторного анализа

Компонент - это производная новая переменная, так что переменные линейно независимы друг от друга. Фактор - это общий элемент, с которым коррелируют несколько других переменных. В PCA компоненты вычисляются как линейные комбинации исходных переменных. В факторном анализе исходные переменные определяются как линейные комбинации факторов. PCA стремится идентифицировать измерения, которые являются составными частями наблюдаемых переменных. Факторный анализ явно предполагает наличие факторов в данных.

**ПРИЛОЖЕНИЕ А**

**Исходный код программы**

import pandas as pd  
import numpy as np  
from sklearn import preprocessing  
import matplotlib.pyplot as plt  
from sklearn.decomposition import PCA, KernelPCA, SparsePCA, FactorAnalysis  
  
df = pd.read\_csv('glass.csv')  
var\_names = list(df.columns) #получение имен признаков  
labels = df.to\_numpy('int')[:, -1] #метки классов  
data = df.to\_numpy('float')[:, :-1] #описательные признаки  
  
data = preprocessing.minmax\_scale(data)  
  
fig, axs = plt.subplots(2, 4)  
for i in range(data.shape[1]-1):  
 axs[i // 4, i % 4].scatter(data[:, i], data[:, (i+1)], c=labels, cmap='hsv')  
 axs[i // 4, i % 4].set\_xlabel(var\_names[i])  
 axs[i // 4, i % 4].set\_ylabel(var\_names[i+1])  
  
plt.show()  
  
svd\_solver = ["auto", "full", "arpack", "randomized"]  
fig, axs = plt.subplots(1, 4)  
for i in range(len(svd\_solver)):  
 pca = PCA(n\_components=4, svd\_solver=svd\_solver[i])  
 pca\_data = pca.fit(data).transform(data)  
 axs[i].scatter(pca\_data[:, 0], pca\_data[:, 1], c=labels, cmap='hsv')  
 axs[i].set\_title(svd\_solver[i])  
  
  
print(pca.explained\_variance\_ratio\_)  
print(pca.singular\_values\_)  
print(sum(pca.explained\_variance\_ratio\_))  
plt.show()  
  
inv\_data = pca.inverse\_transform(pca\_data)  
  
fig, axs = plt.subplots(2, 4)  
for i in range(inv\_data.shape[1]-1):  
 axs[i // 4, i % 4].scatter(inv\_data[:, i], inv\_data[:, (i+1)], c=labels, cmap='hsv')  
 axs[i // 4, i % 4].set\_xlabel(var\_names[i])  
 axs[i // 4, i % 4].set\_ylabel(var\_names[i+1])  
  
plt.show()  
  
pars = ["linear", "poly", "rbf", "sigmoid", "cosine"]  
fig, axs = plt.subplots(1, 5)  
for i in range(len(pars)):  
 data\_kernel = KernelPCA(n\_components=4, kernel=pars[i]).fit\_transform(data)  
 axs[i].scatter(data\_kernel[:, 0], data\_kernel[:, 1], c=labels, cmap='hsv')  
 axs[i].set\_title(pars[i])  
  
plt.show()  
  
fig, axs = plt.subplots(2, 3)  
for i in range(0, 11, 2):  
 data\_sparse = SparsePCA(n\_components=4, alpha=i/10).fit\_transform(data)  
 axs[i // 6, (i % 6)//2].scatter(data\_sparse[:, 0], data\_sparse[:, 1], c=labels, cmap='hsv')  
 axs[i // 6, (i % 6)//2].set\_title(f"alpha = {i/10}")  
  
plt.show()  
  
pca = FactorAnalysis(n\_components=4)  
data\_factor = pca.fit\_transform(data)  
  
plt.scatter(data\_factor[:, 0], data\_factor[:, 1], c=labels, cmap='hsv')  
plt.show()