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.

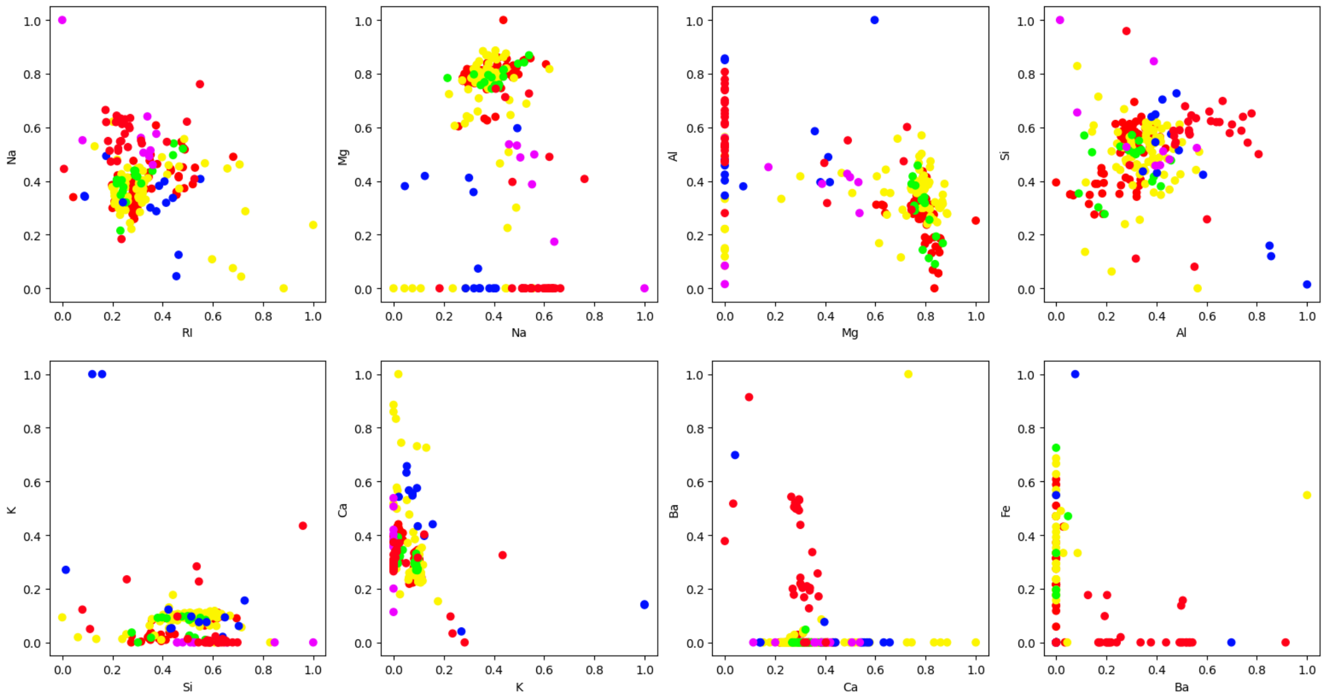


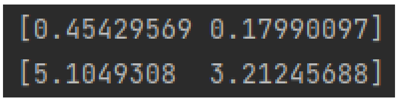
Рисунок 1 – Диаграммы рассеяния.

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.



```
[0.45429569 0.17990097]
[5.1049308  3.21245688]
```

Рисунок 2 – Полученные значения.

Построена диаграмма рассеяния после метода главных компонент.

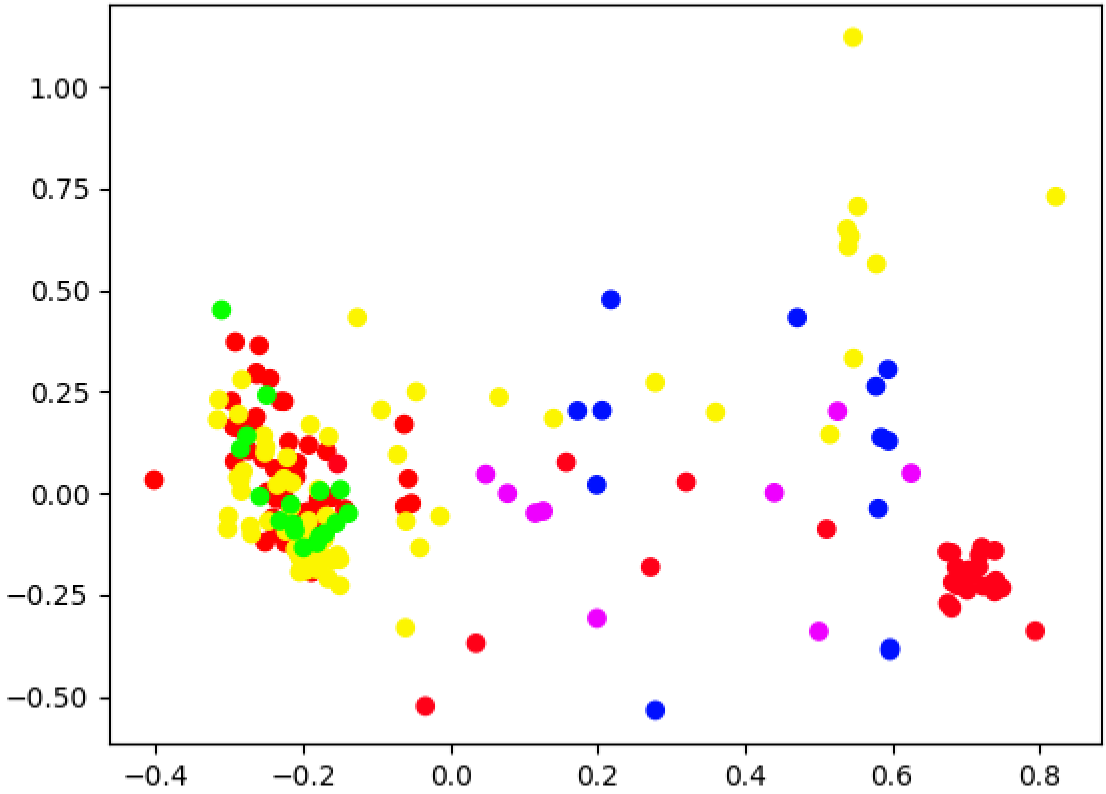


Рисунок 3 – Диаграмма рассеяния после метода главных компонент.

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

```
[0.45429569 0.17990097 0.12649459 0.09797847]
[5.1049308  3.21245688 2.69374532 2.3707507 ]
0.858669730510272
```

Рисунок 4 – Полученные значения.

Используя метод inverse_transform восстановлены данные, построены диаграммы рассеяния.

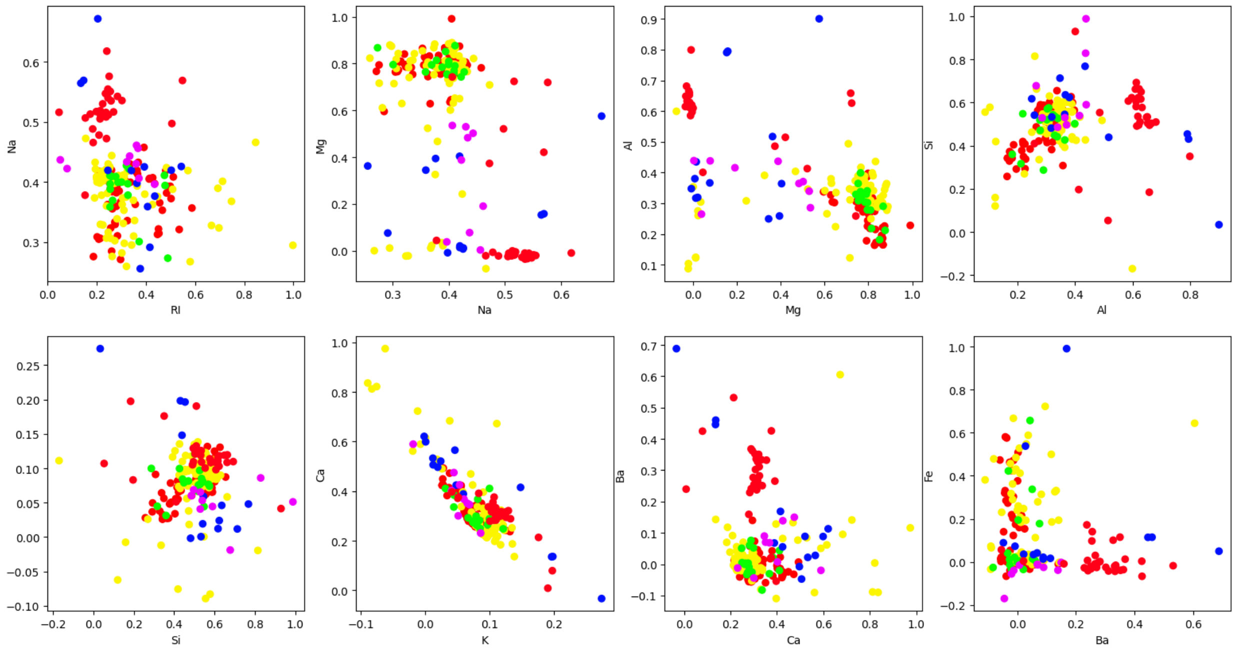


Рисунок 5 – Диаграммы рассеяния после восстановления данных.

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

Исследован метод главных компонент при различных параметрах svd_solver.

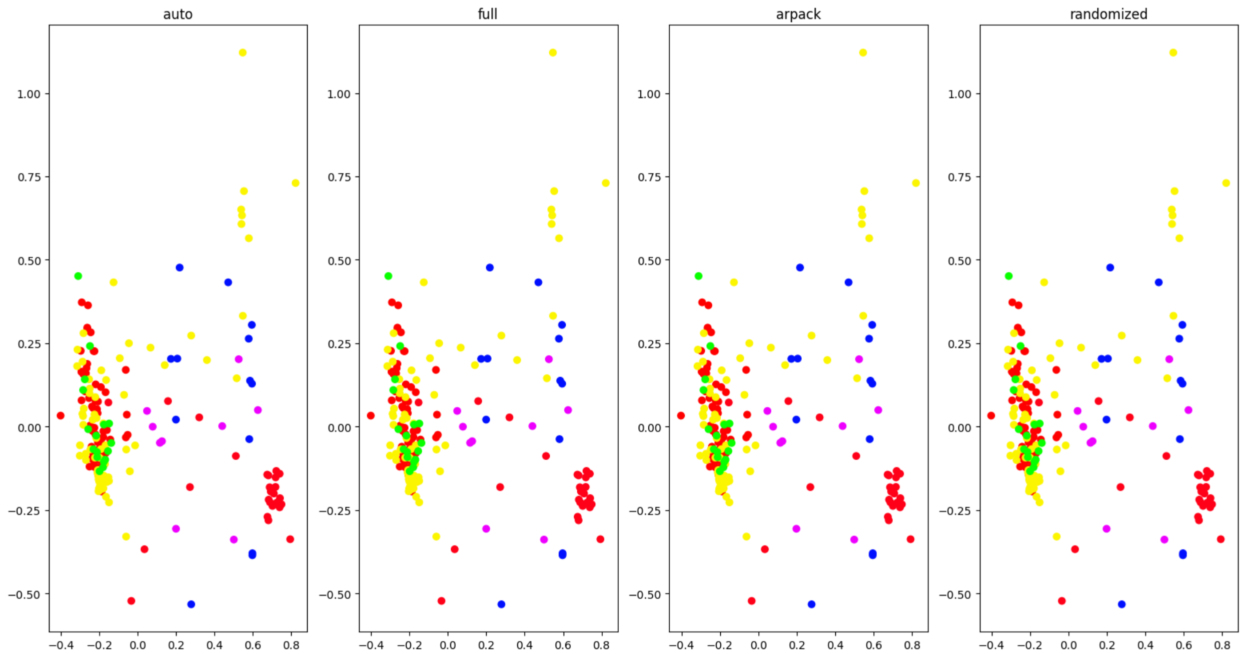


Рисунок 6 – Диаграммы рассеивания при различных значениях svd_solver.

```
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.

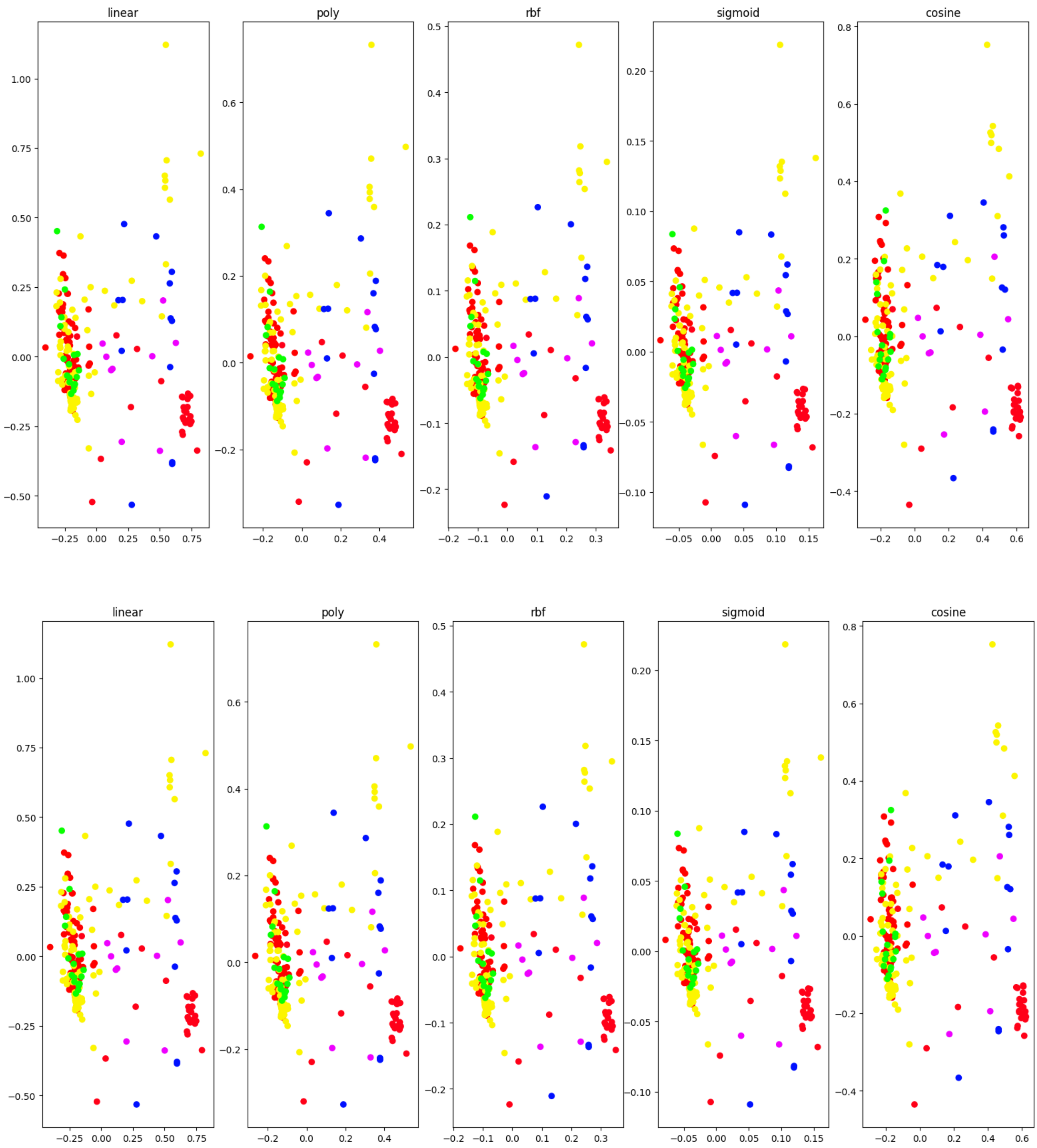


Рисунок 7 – Исследование параметра kernel

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

Аналогично исследован SparcePCA для различных параметров alpha.

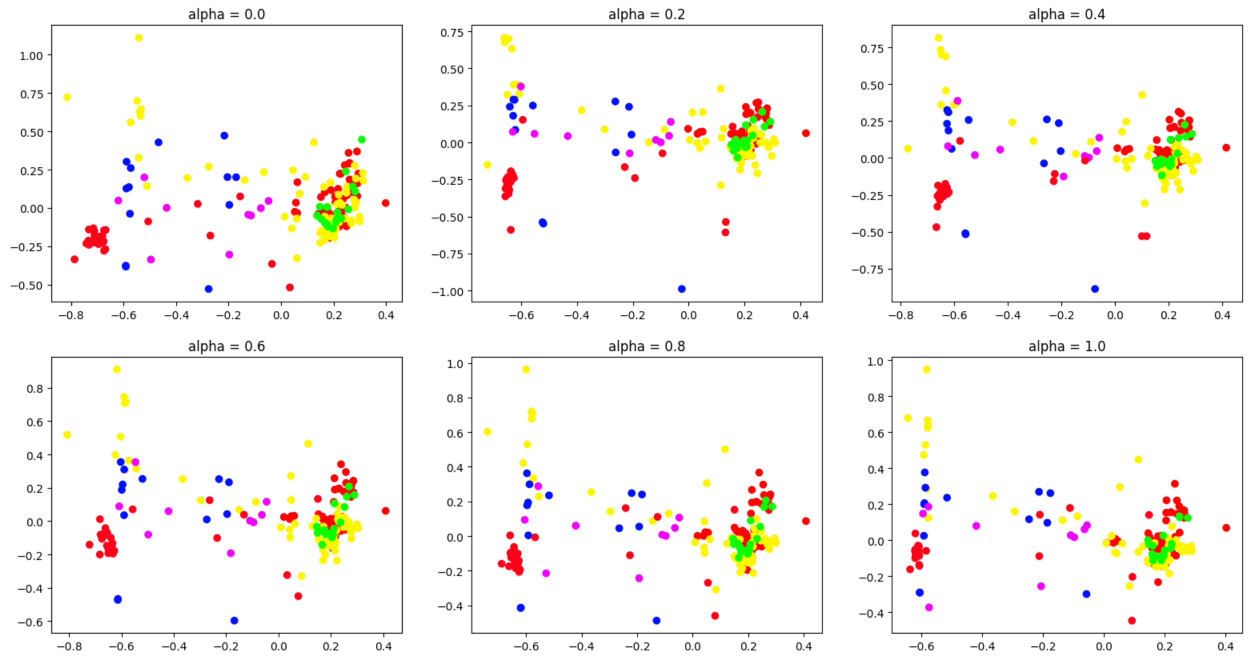


Рисунок 8 – Исследование параметра alpha

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

4. Факторный анализ

Проведено понижении размерности используя факторный анализ FactorAnalysis.

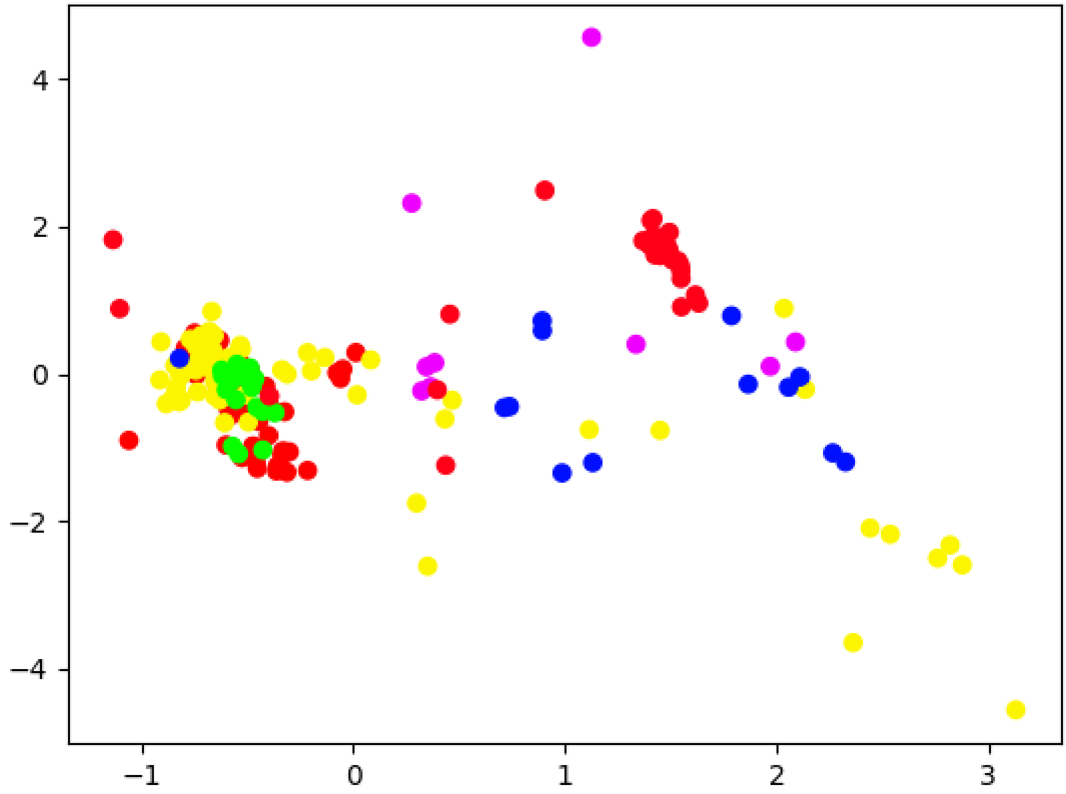


Рисунок 9 - Диаграммы рассеивания факторного анализа

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()
```