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. 6307 _____ Давыдова Н. П.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2020

Содержание

Загрузка данных

1. Загрузила датасет.
2. Загрузила датасет в датафрейм, и разделила данные на описательные признаки и признак отображающий класс, провела нормировку данных к интервалу [0 1]

```
 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)
print(df.describe())
```

3. Построила диаграммы рассеяния для пар признаков. Определила соответствие цвета на диаграмме и класса в датасете.

```
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()
```

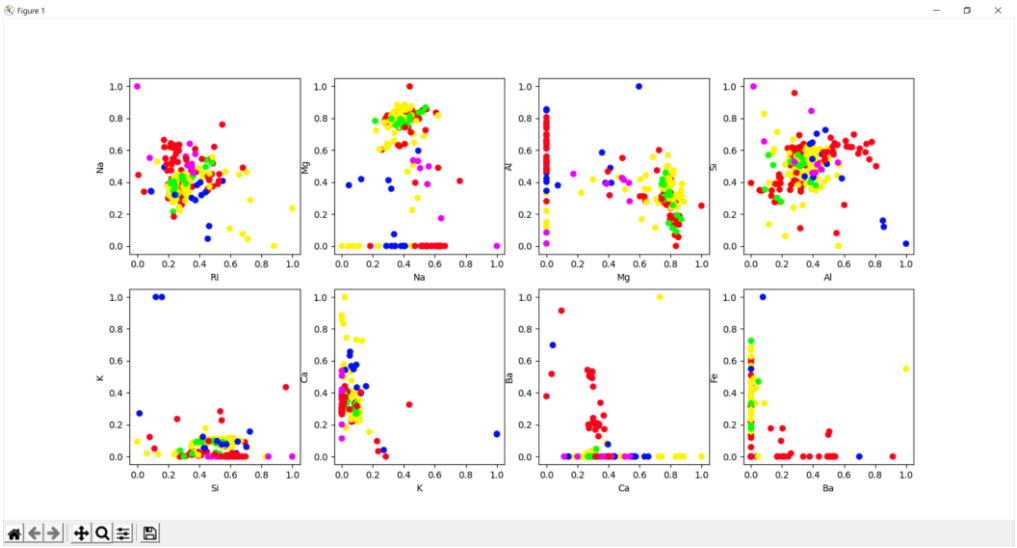


Рис. 1 диаграмма рассеивания для пар признаков

```
fig, axs = plt.subplots(1, 1)
plt.scatter(labels, [1]*214, c = labels, cmap = 'hsv')
plt.show()
```

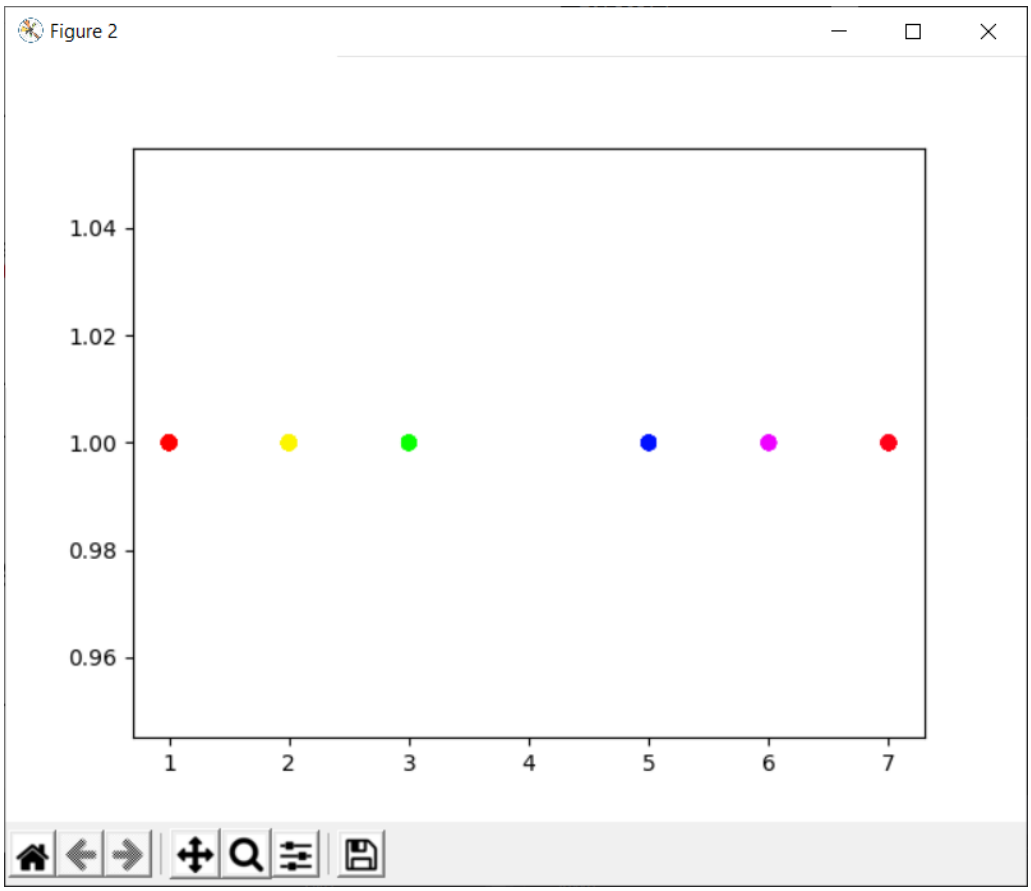


Рис. 2 Соответствие цветов и классов

Метод главных компонент

1. Используя метод главных компонент (PCA) провела понижение размерности пространства до 2.
   Вывела значение объясненной дисперсии в процентах
   и собственные числа, соответствующие компонентам

```
#метод главных компонент
pca = PCA(n_components = 2)
pca_data = pca.fit(data).transform(data)

#значение объясненной дисперсии в процентах и собственные числа
соответствующие компонентам
print(pca.explained_variance_ratio_)
print(pca.singular_values_)
```

[0.45429569 0.17990097] 45%, 18%
[5.1049308 3.21245688]

2. Построила диаграмму рассеяния после метода главных компонент

```
plt.scatter(pca_data[:,0],pca_data[:,1],c=labels,cmap='hsv')
plt.show()
```

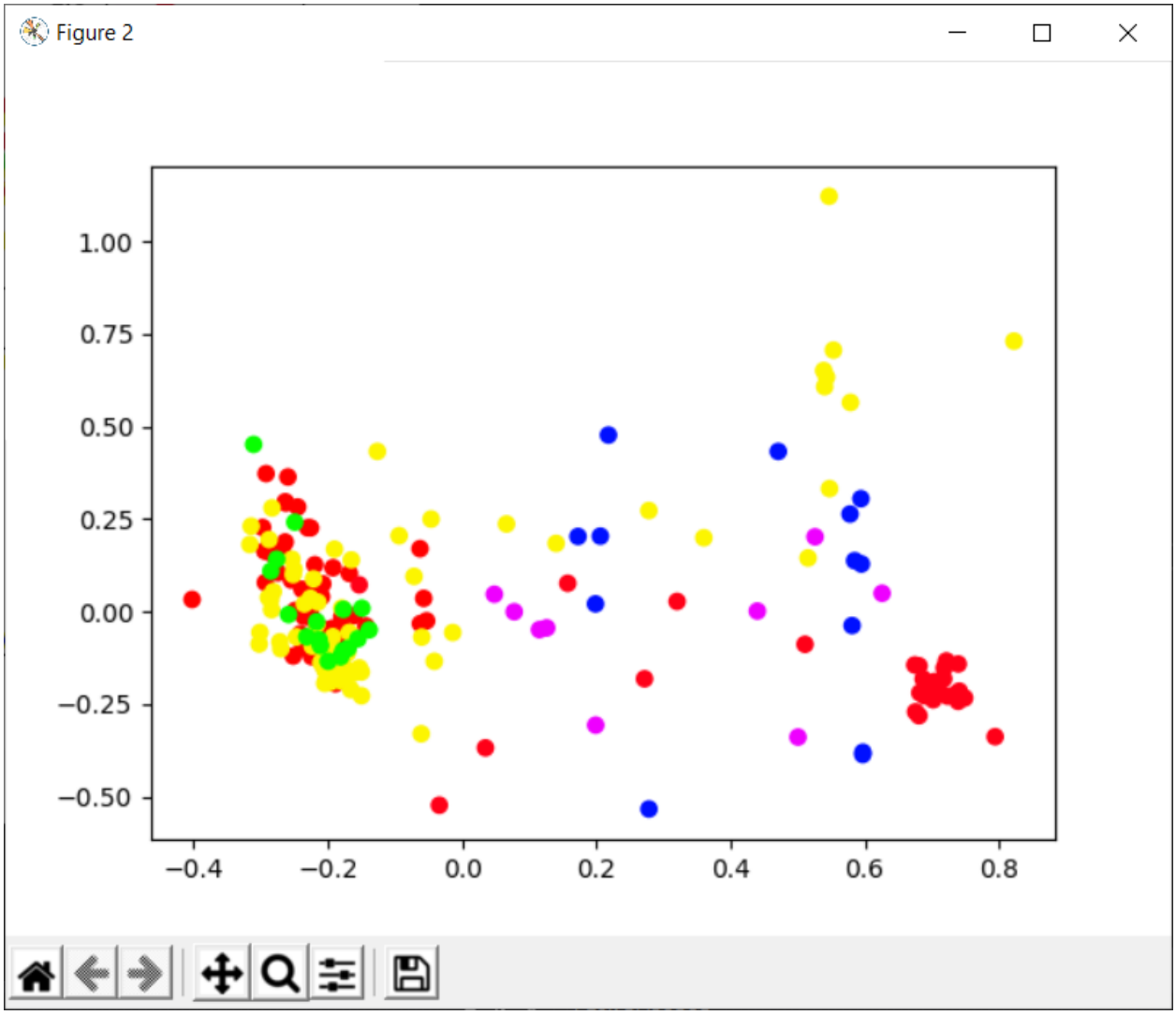


Рис. 3 Диаграмма рассеивания после метода главных компонент

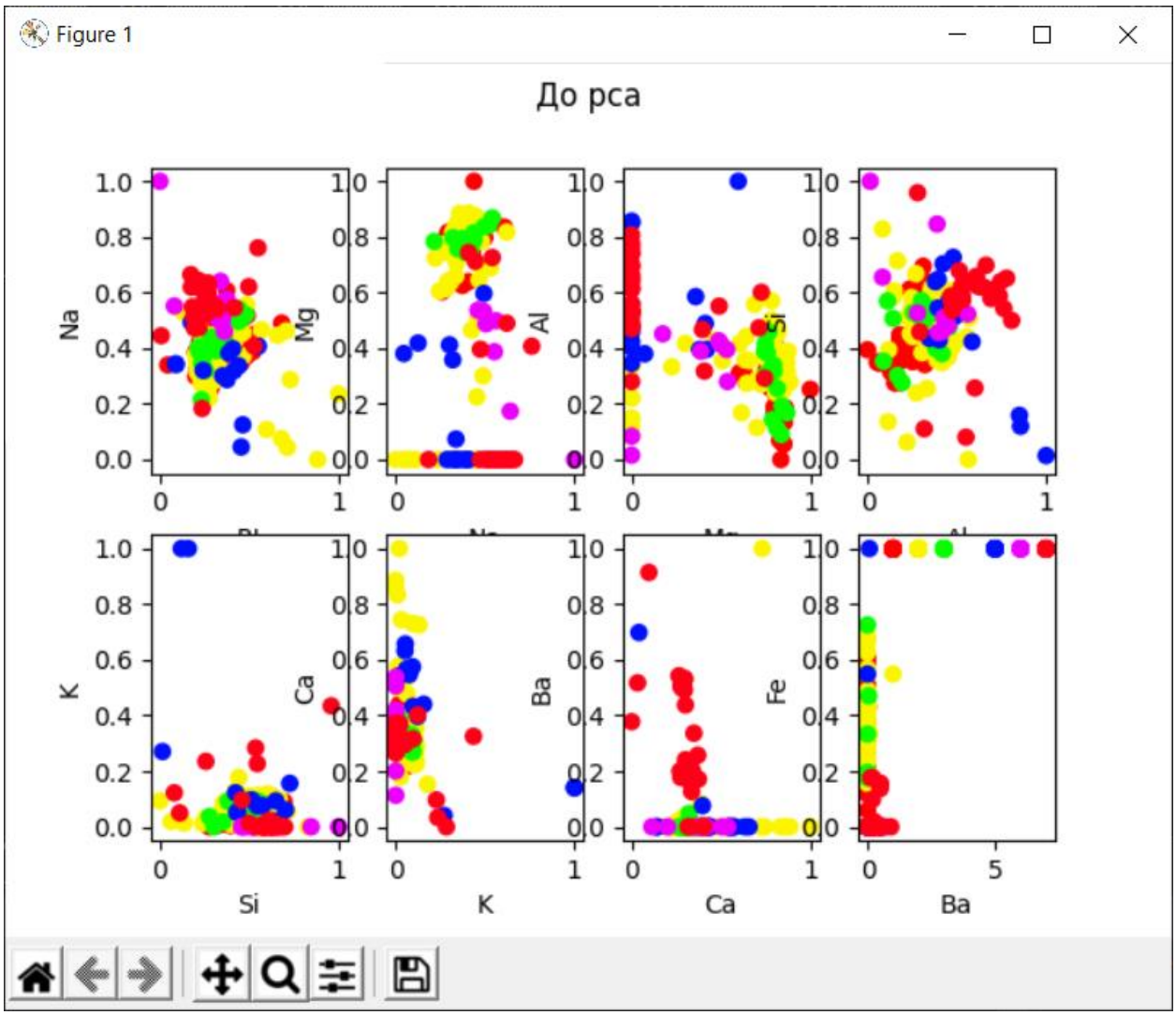


Figure 1
До pca
Na
Mg
Al
Si
K
Ca
Ba
Fe
Si
K
Ca
Ba

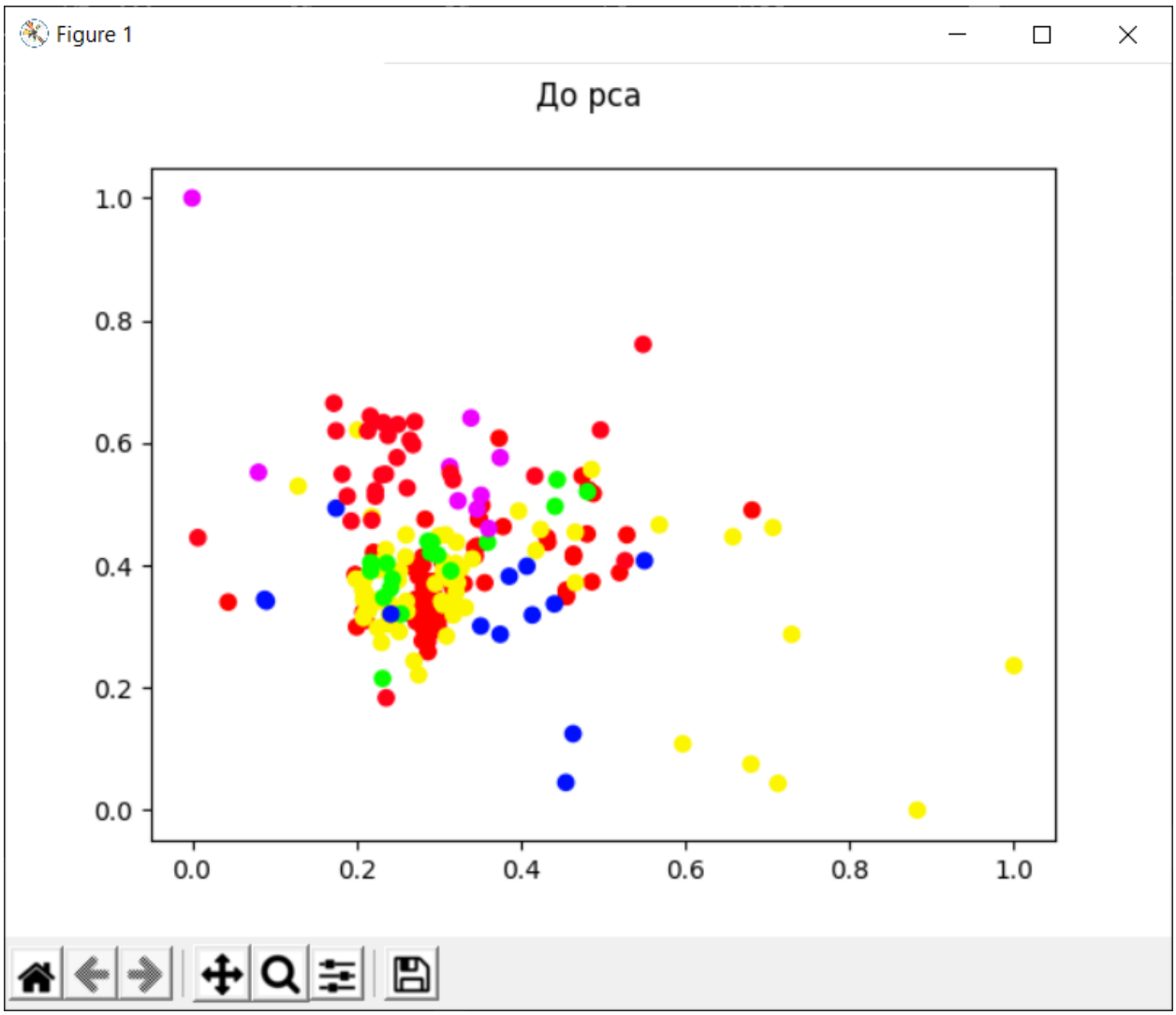


Рис. 4 Диаграмма рассеивания до метода главных компонент

Дисперсия по компонентам охватывает только 63% дисперсии, центрирует данные.

3. Изменила количество компонент на 4, чтобы дисперсия покрывала 85%.

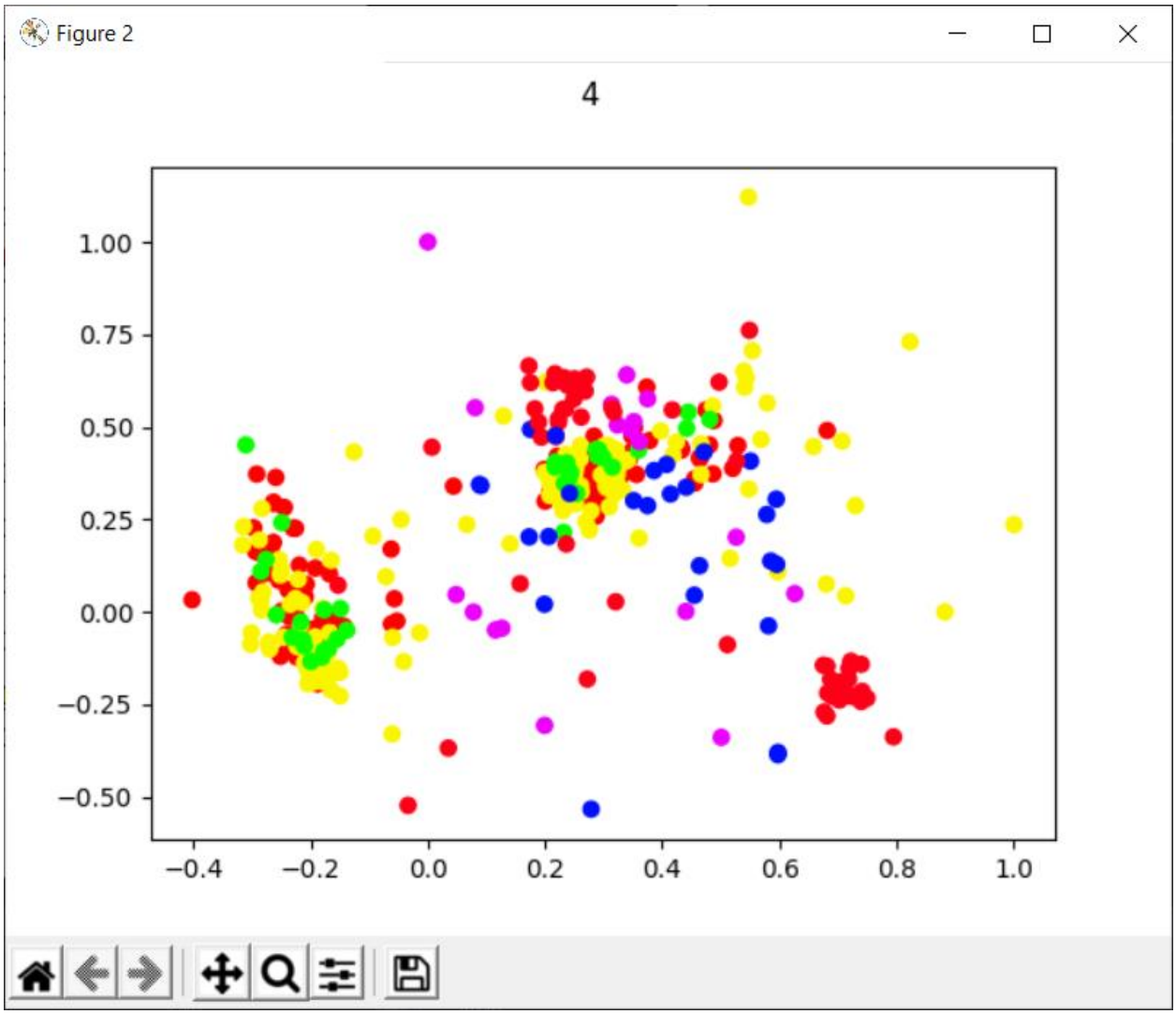


Рис. 5 Диаграмма рассеивания по 4 компонентам

Дисперсия в сумме ~85.5%:

[0.45429569 0.17990097 0.12649459 0.09797847]

4. Используя метод inverse_transform восстановила данные

```
inversed_data = pca.inverse_transform(pca_data)
```

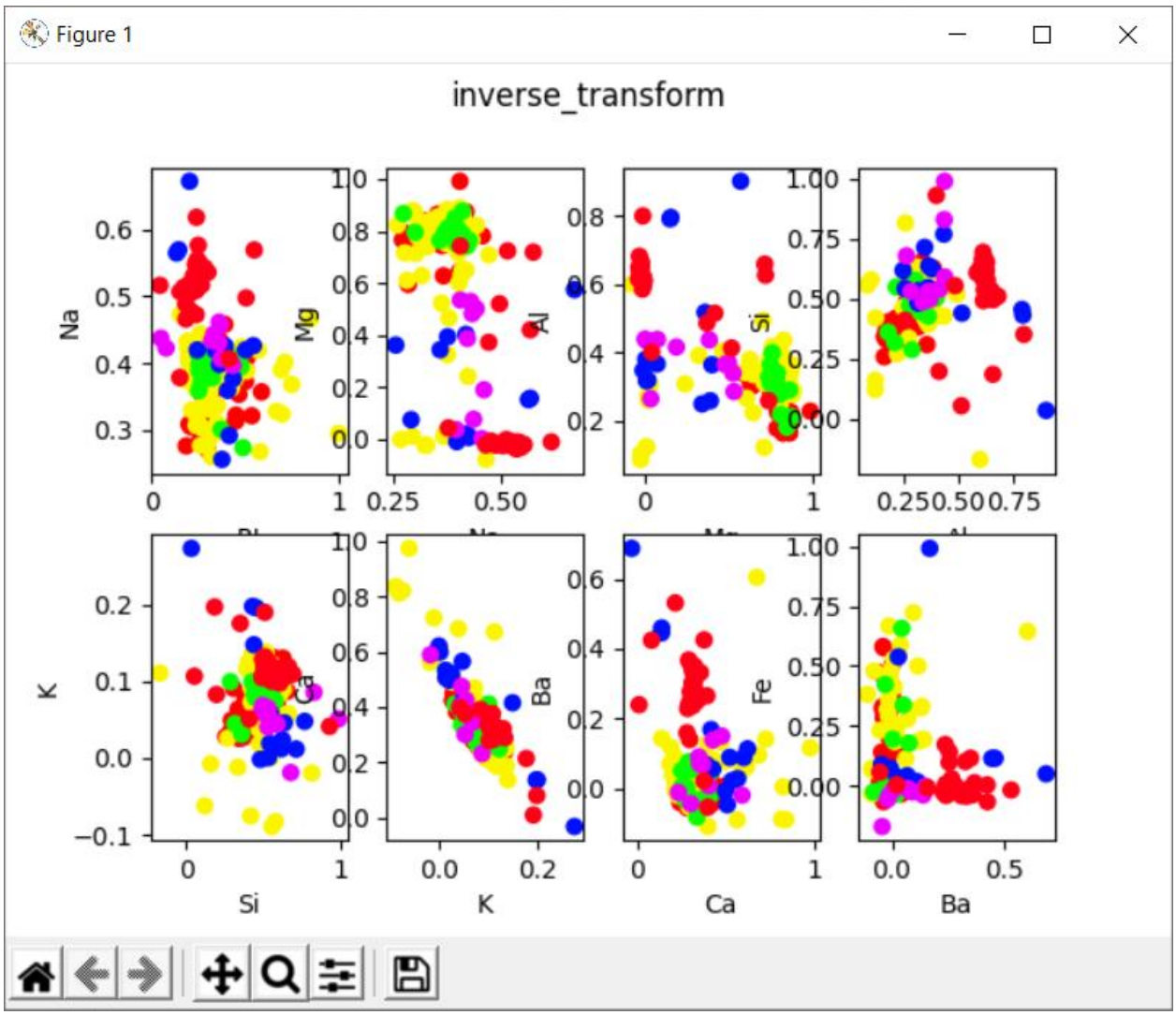


Рис. 6 inverse_transform

```
print(inversed_data.var())
```

Дисперсия стала равна ~58%,

т.е. ~27.5% потеряно, часть значений не восстановлена.

При рса с 6 компонентами при таких же действиях дисперсия была

~96.9 -> 61.1

Стоит отметить, что исходное среднее значение и восстановленное – совпадают. (0.3066213794472578 и 0.30662137944725787)

Модификации метода главных компонент

1. По аналогии с PCA исследуйте KernelPCA для различных параметров kernel и различных параметрах для ядра

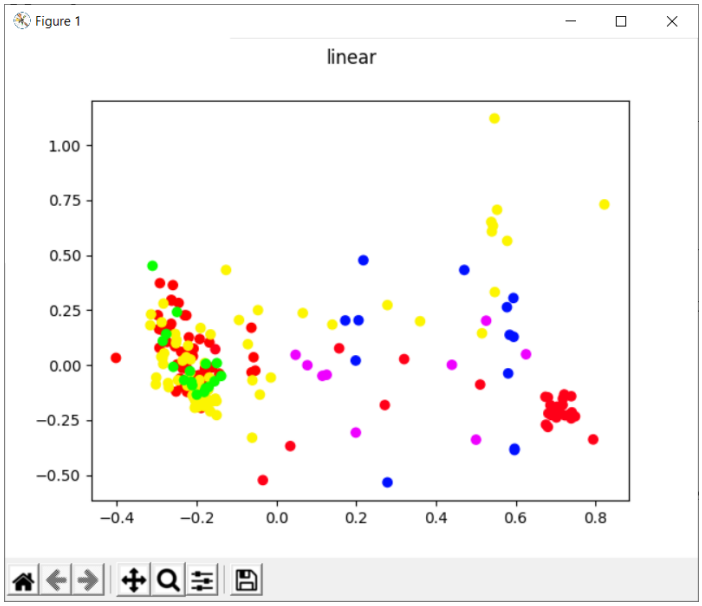
```
kernels = ["linear", "poly", "rbf", "sigmoid", "cosine"]

plt.tight_layout()

for i in range(len(kernels)):
    kernels_pca = KernelPCA(n_components=4, kernel=kernels[i])
    kernels_pca_data = kernels_pca.fit(data).transform(data)
    print(kernels_pca.lambdas_)
    plt.scatter(kernels_pca_data[:,0], kernels_pca_data[:,1], c=labels, cmap='hsv')
    plt.suptitle(kernels[i])
    plt.show()
```

Linear	[26.06031845 10.31987923 7.25626387 5.6204589]
Poly	[10.9181964 4.31937695 3.1188508 2.36791674]
Rbf	[5.35145251 2.0180542 1.4957381 1.11090453]
Sigmoid	[1.00618101 0.39983752 0.27409853 0.2161195]
Cosine	[18.31403041 6.47538495 4.6959991 3.57812492]

Судя по собственным числам, при параметре linear KernelPCA соответствует PCA



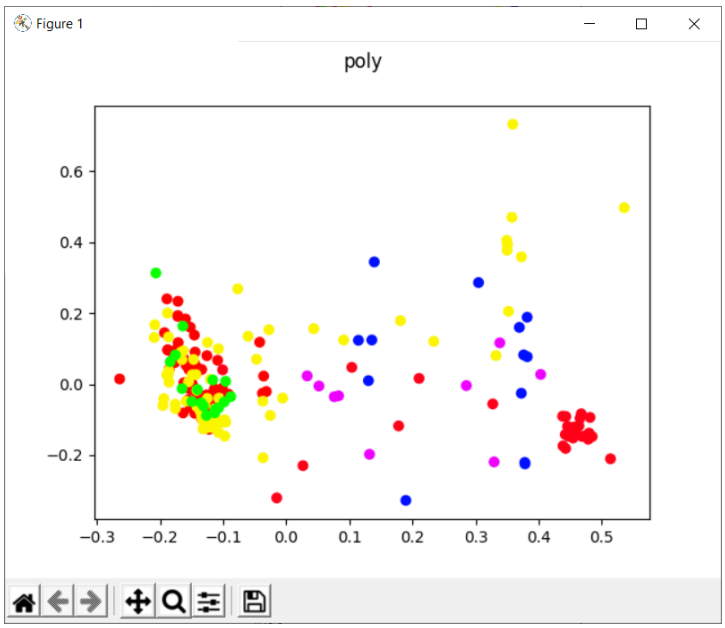


Figure 1
poly
0.6
0.4
0.2
0.0
−0.2
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
0.5

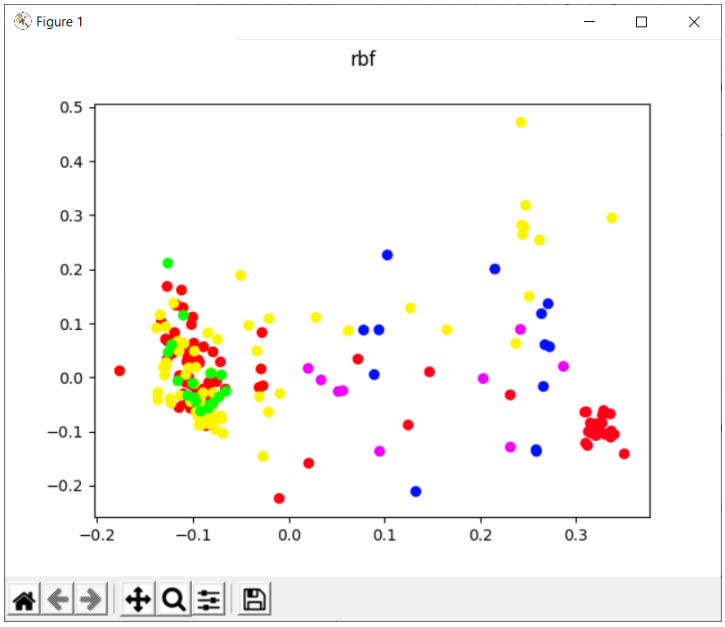


Figure 1
rbf
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.2
−0.1
0.0
0.1
0.2
0.3

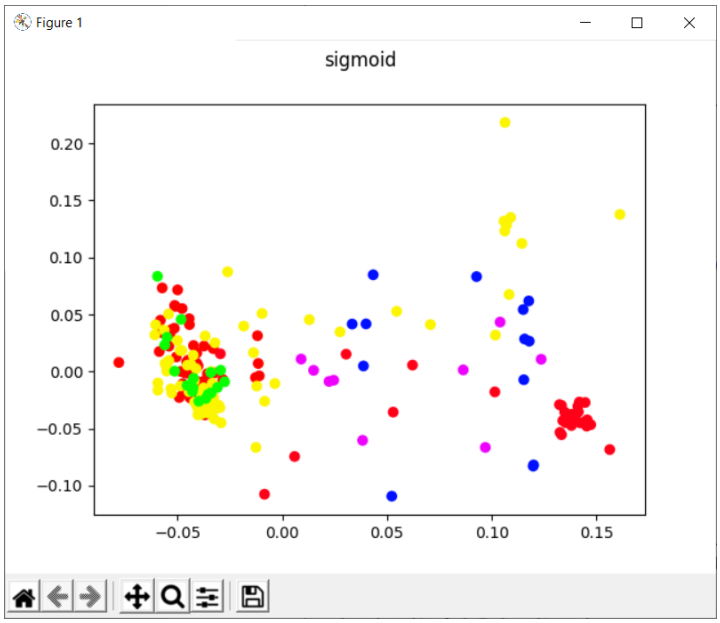
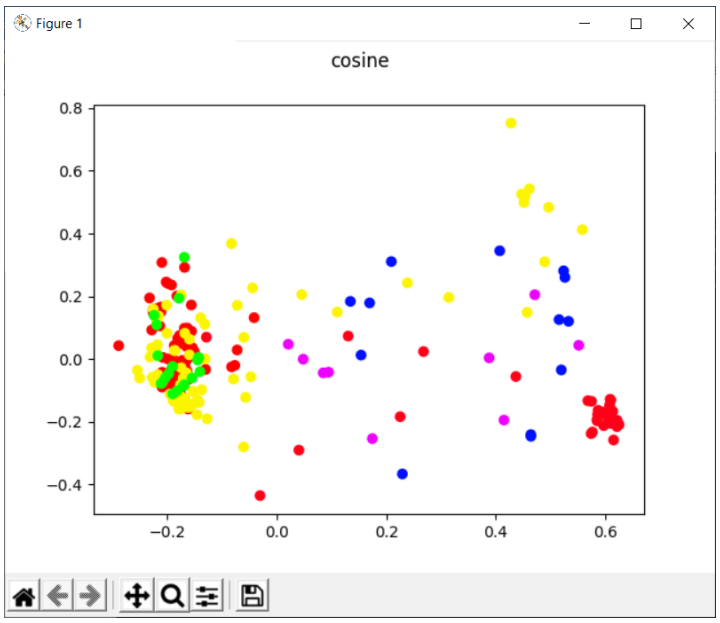


Figure 1
sigmoid
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.05
0.00
0.05
0.10
0.15

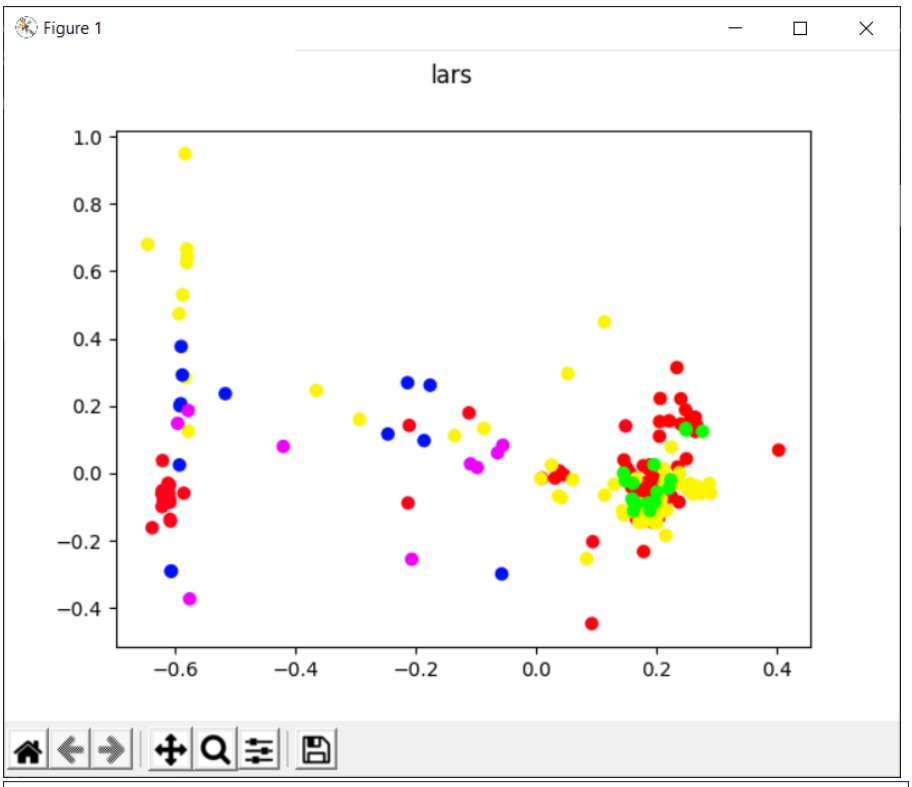


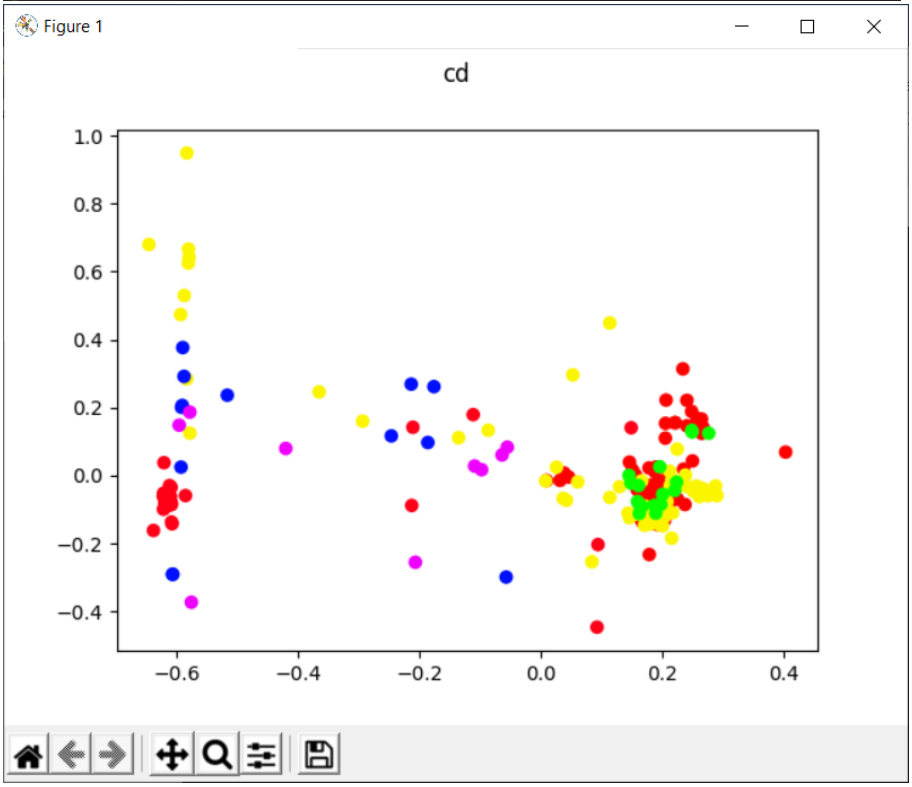
Графики почти не отличаются кроме диапозона.

2. Аналогично исследуйте SparcePCA

```
sparse = ["lars", "cd"]
for i in range(len(sparse)):
    sparse_pca = SparsePCA(n_components=4, method=sparse[i])
    sparse_pca_data = sparse_pca.fit(data).transform(data)

    plt.scatter(sparse_pca_data[:,0], sparse_pca_data[:,1], c=labels,
cmap='hsv')
    plt.suptitle(sparse[i])
    plt.show()
```





При изменении cd и lars изменения не заметны

Если изменить параметр alpha (разреженность), то получаются различные компоненты:

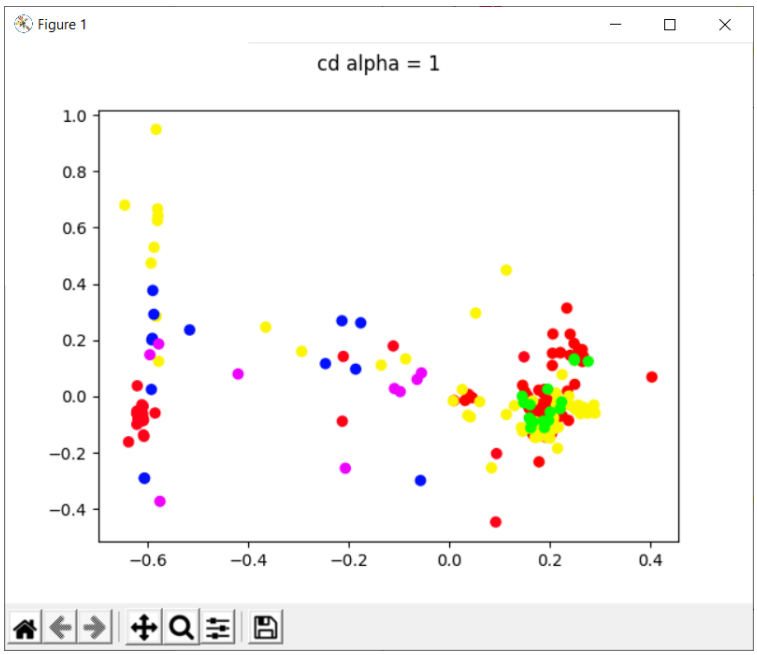


Figure 1
cd alpha = 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.4
−0.2
0.0
0.2
0.4

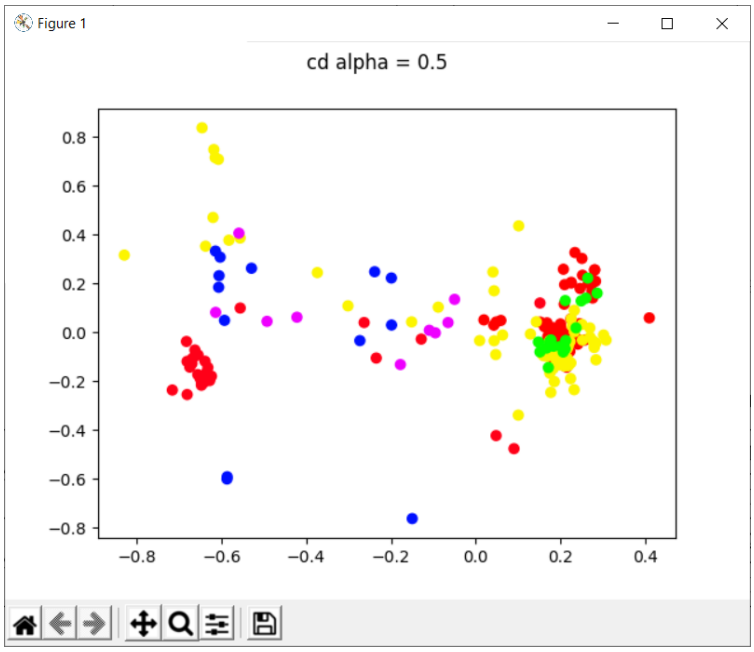
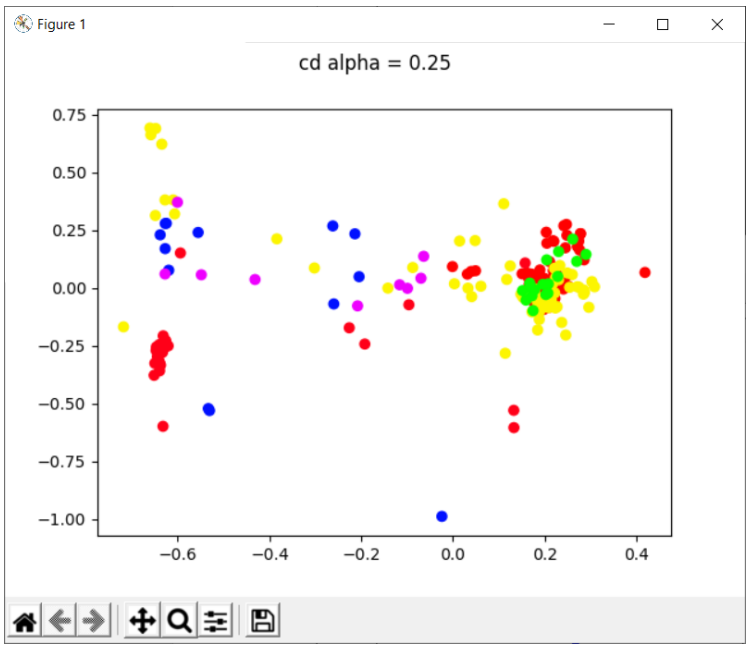


Figure 1
cd alpha = 0.5
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4



Факторный анализ

1. Проведите понижении размерности используя факторный анализ FactorAnalysis

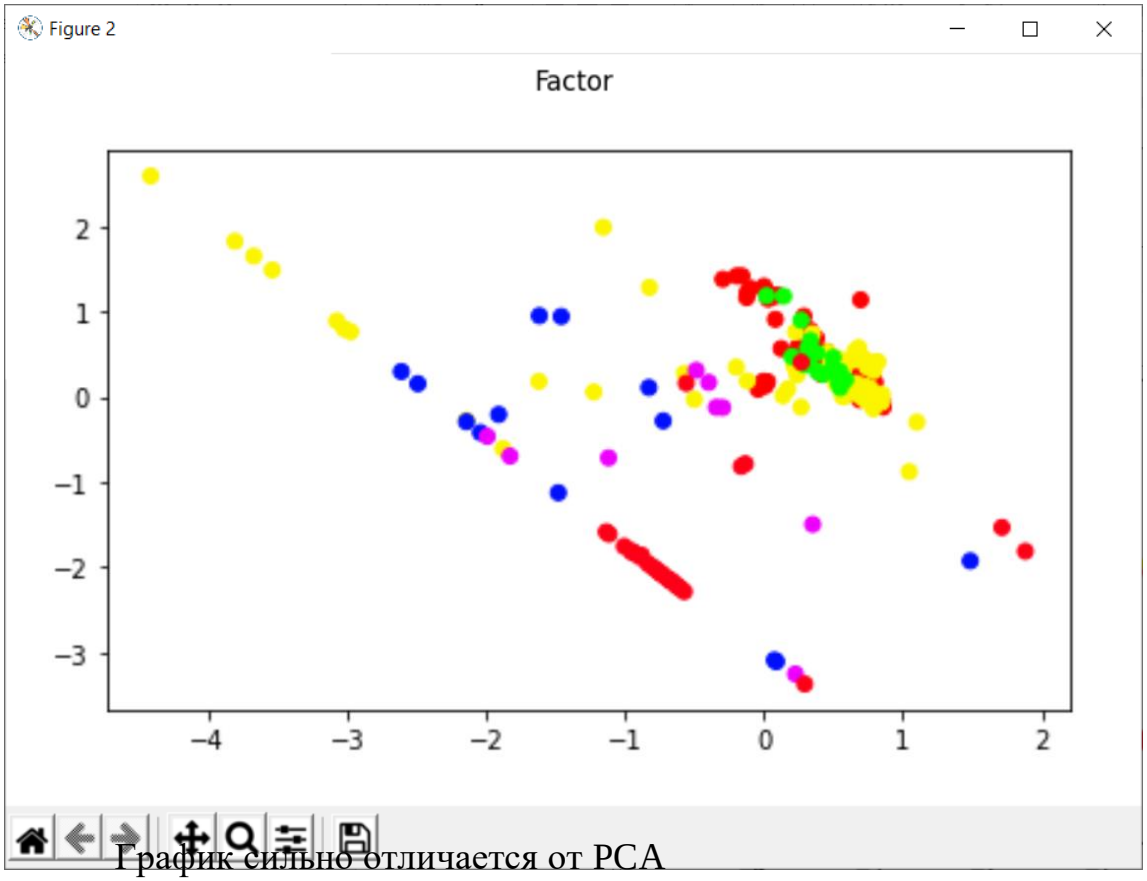


График сильно отличается от PCA

Он так же нужен для сокращения количества данных, необходимых для анализа.
Разница в создании компонент.

Вывод:

Ознакомилась с методами понижения размерности данных из библиотеки Scikit Learn.