МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Машинное обучение»

Тема: Понижение размерности пространства признаков

Студент гр. 6307 _____ Новиков Б.М.

Преподаватель _____ Жангиров Т.Р.

2020

1.Загрузить датасет по ссылке: https://www.kaggle.com/uciml/glass . Данные представлены в виде csv таблицы.

2.Создать Python скрипт. Загрузить датасет в датафрейм.

```
df = pd.read_csv('data/glass.csv')
df.head()
```

Out[311]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

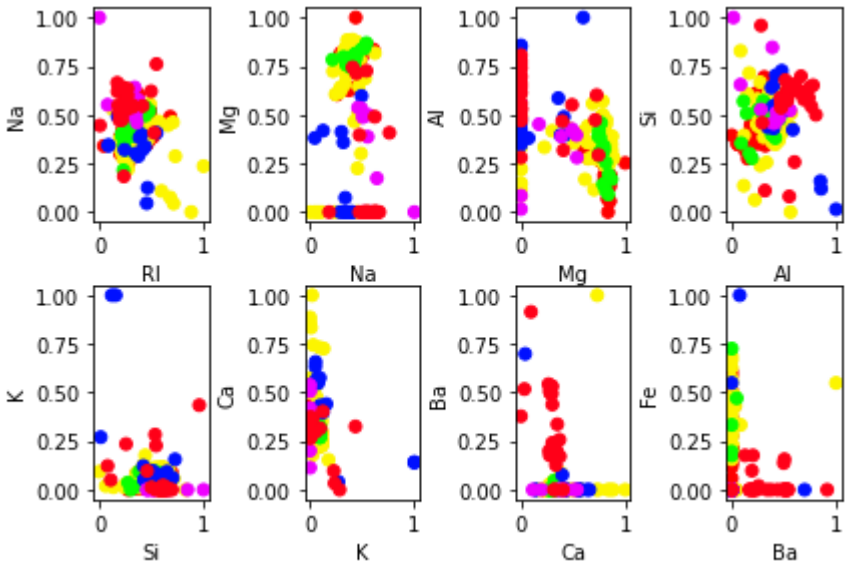
Разделить данные на описательные признаки и признак отображающий класс.

```
var_names = list(df.columns) #получение имен признаков
labels = df.to_numpy('int')[:,-1] #метки классов
data = df.to_numpy('float')[:,:-1] #описательные признаки
```

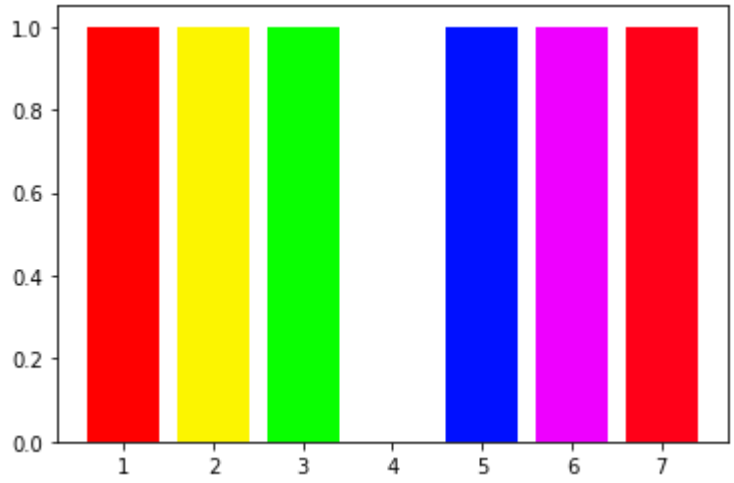
3. Провести нормировку данных к интервалу [0 1]

```
data = preprocessing.minmax_scale(data)
```

4. Построить диаграммы рассеяния для пар признаков. Самостоятельно определите соответствие цвета на диаграмме и класса в датасете



```
my_cmap = cm.get_cmap('hsv')
my_norm = Normalize(vmin=np.min(np.unique(labels)),
                    vmax=np.max(np.unique(labels)))

plt.bar(np.unique(labels), 1,
color=my_cmap(my_norm(np.unique(labels))))
plt.show()
```



Цвет — Класс :

- красный - 1
- желтый - 2
- зеленый - 3
- голубой - 4 (не используется)
- синий - 5
- фиолетовый - 6
- красный - 7

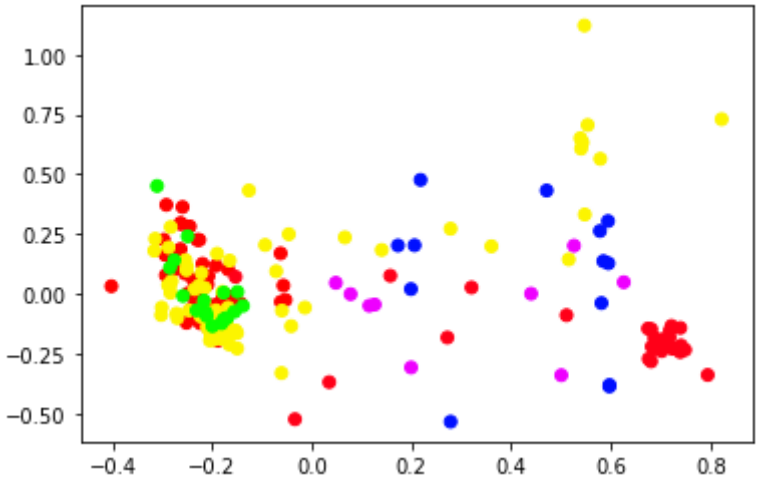
1. Используя метод главных компонент (PCA). Проведите понижение размерности пространства до размерности 2

```
pca = PCA(n_components = 2)
pca_data = pca.fit(data).transform(data)
```

2. Выведите значение объясненной дисперсии в процентах и собственные числа соответствующие компонентам
print(pca_test.explained_variance_ratio_)
`print(pca_test.singular_values_)`

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

3. Постройте диаграмму рассеяния после метода главных компонент



4. Проанализируйте и обоснуйте полученные результаты

Уменьшение линейной размерности с использованием разложения данных по сингулярным значениям для проецирования их в пространство с меньшей размерностью. Входные данные центрируются, но не масштабируются для каждой функции перед применением SVD.

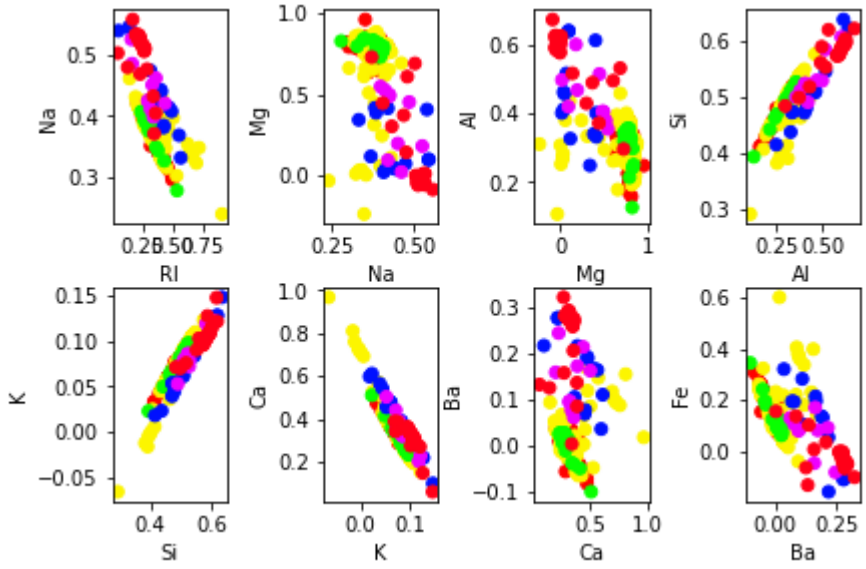
5. Изменяя количество компонент, определите количество при котором компоненты объясняют не менее 85% дисперсии данных

pca_test = PCA(n_components = 4)
`pca_test_data = pca_test.fit(data).transform(data)`print(f"при {pca_test.n_components} объяснено {np.sum(pca_test.explained_variance_ratio_)}")

при 4 объяснено 0.8586697305102717

6. Используя метод inverse_transform восстановите данные, сравните с исходными

prev_data = pca.inverse_transform(pca_data)



Как говорилось ранее, было объяснено только 60% данных, поэтому на графиках мы наблюдаем потери.

7. Исследуйте метод главных компонент при различных параметрах svd_solver

auto: при 2 объяснено 0.6341966621042778 за 0.0008883476257324219
full: при 2 объяснено 0.6341966621042778 за 0.0005629062652587891
arpack: при 2 объяснено 0.634196662104278 за 0.0022330284118652344
randomize: при 2 объяснено 0.6341966621042778 за 0.0017657279968261719

auto: при 3 объяснено 0.7606912558548662 за 0.0004870891571044922
full: при 3 объяснено 0.7606912558548662 за 0.0005779266357421875
arpack: при 3 объяснено 0.7606912558548664 за 0.0014243125915527344
randomize: при 3 объяснено 0.7606912558548665 за 0.0048673152923583984

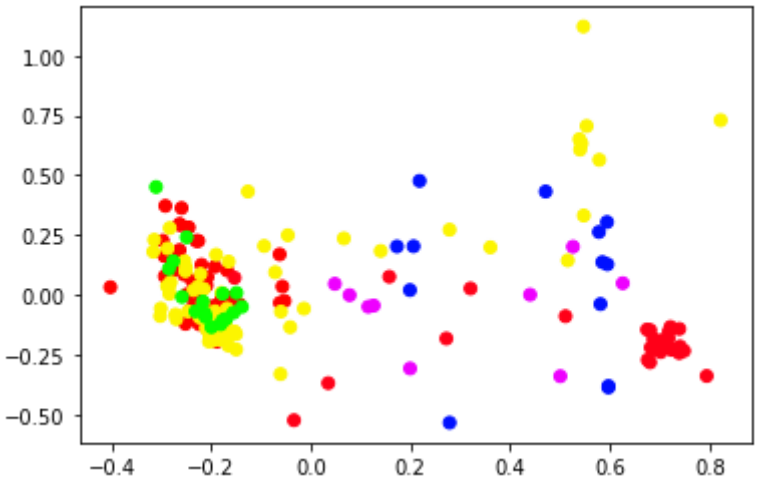
auto: при 4 объяснено 0.8586697305102717 за 0.0006194114685058594
full: при 4 объяснено 0.8586697305102717 за 0.0005605220794677734
arpack: при 4 объяснено 0.8586697305102717 за 0.0010724067687988281
randomize: при 4 объяснено 0.8586697305102716 за 0.001171112060546875

```
auto:            при 5 объяснено 0.9272937149511479 за
0.0005259513854980469
full:            при 5 объяснено 0.9272937149511479 за
0.0004556179046630859 4
arpack:          при 5 объяснено 0.9272937149511477 за
0.0012247562408447266
randomize:       при 5 объяснено 0.9272937149511488 за
0.001383066177368164

auto:            при 6 объяснено 0.9694347221994033 за
0.0004594326019287109 4
full:            при 6 объяснено 0.9694347221994033 за
0.0004661083221435547
arpack:          при 6 объяснено 0.9694347221994032 за
0.0017657279968261719
randomize:       при 6 объяснено 0.969434722199403 за
0.0019042491912841797

auto:            при 7 объяснено 0.9955326243472864 за
0.0005545616149902344
full:            при 7 объяснено 0.9955326243472864 за
0.0005307197570800781
arpack:          при 7 объяснено 0.9955326243472861 за
0.0016689300537109375
randomize:       при 7 объяснено 0.9955326243472872 за
0.002304553985595703

auto:            при 8 объяснено 0.9998605862637865 за
0.0006623268127441406
full:            при 8 объяснено 0.9998605862637865 за
0.0007853507995605469
arpack:          при 8 объяснено 0.9998605862637863 за
0.0019965171813964844
randomize:       при 8 объяснено 0.9998605862637874 за
0.0026693344116210938
```

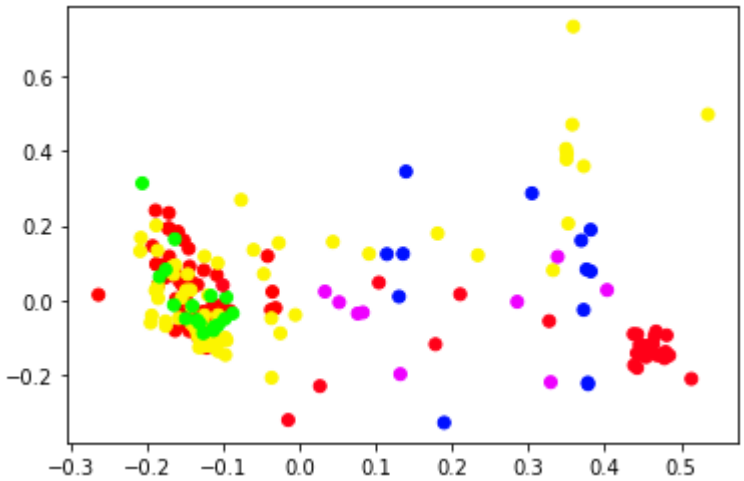
Для данного датасета не наблюдается разницы в общей объясненной дисперсии при любом количестве компонент, но зато есть разница во времени исполнения: лучше всех себя показывает full.

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

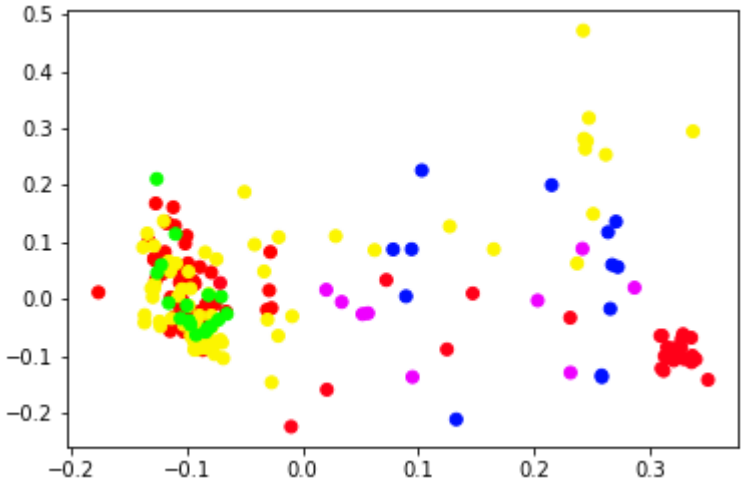
linear



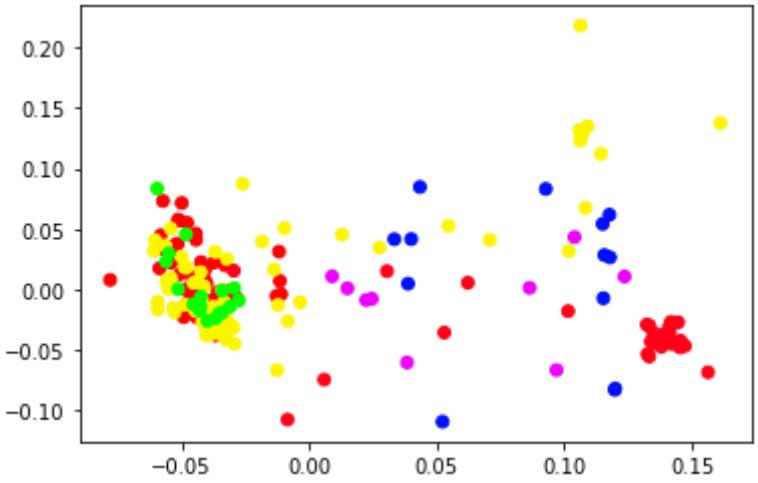
poly



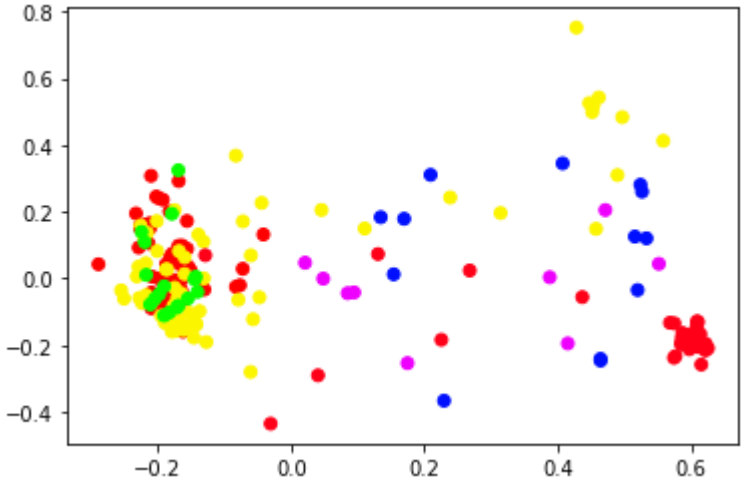
rbf



sigmoid



cosine

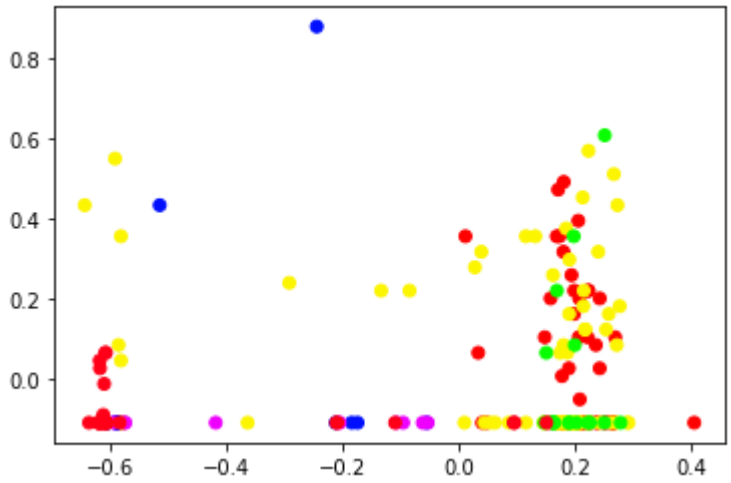


По графикам можно заметить , что данные распределены одинаковым образом, но их масштаб и центр не совпадают.

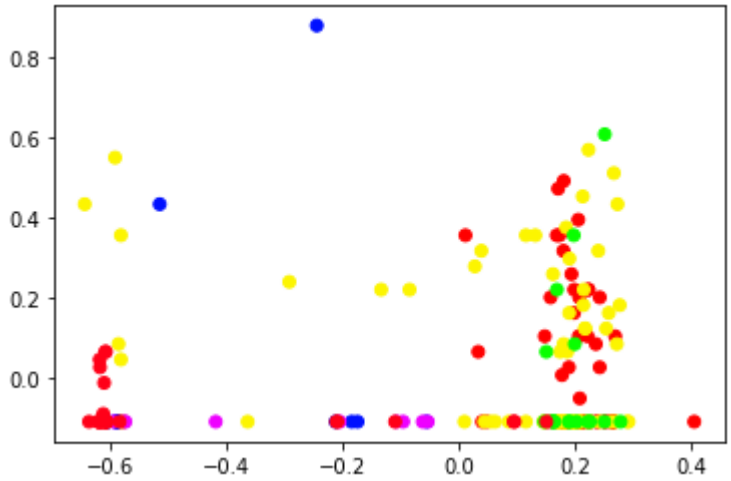
2. Определите, при каких параметрах KernelPCA работает также как PCA

При linear, как видно по графику.

3. Аналогично исследуйте SparcePCA
lars



cd



4. Проанализируйте и обоснуйте полученные результаты

lars: при 2 за 0.021564483642578125
cd: при 2 за 0.008909463882446289

lars: при 3 за 0.025670528411865234
cd: при 3 за 0.010800600051879883

lars: при 4 за 0.028514623641967773
cd: при 4 за 0.013376951217651367

lars: при 5 за 0.03407144546508789
cd: при 5 за 0.011635065078735352

lars: при 6 за 0.08673429489135742
cd: при 6 за 0.035430908203125

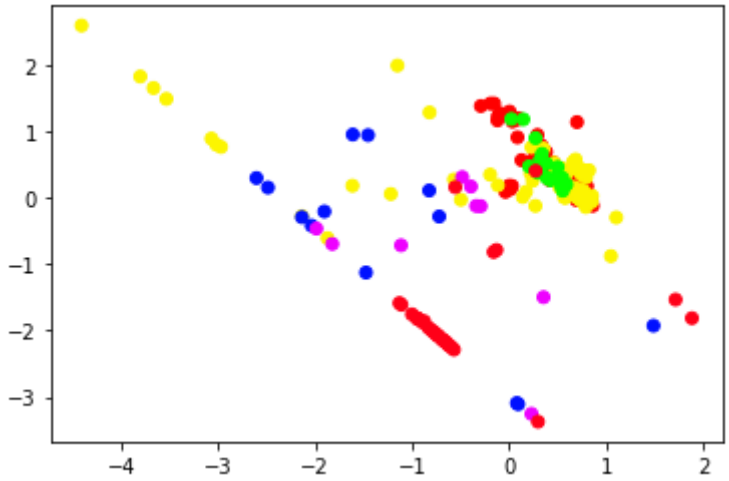
lars: при 7 за 0.09393882751464844

```
cd:               при 7 за 0.034701824188232 42

lars:             при 8 за 0.0957038402557373
cd:               при 8 за 0.03627443313598633
```

Метод cd работает в разы быстрее lars.

1. Проведите понижении размерности используя факторный анализ FactorAnalysis

lapack



randomized