Table of Contents


```
(setq-local org-image-actual-width '(1024))
(setq-local org-html-htmlize-output-type 'css)
```

1 Выполнение

1.1 Загрузка данных

```python
import pandas as pd
import numpy as np

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

var_names = list(df.columns)

labels = df.to_numpy('int')[:, -1]
data = df.to_numpy('float')[:,:-1]
df
```

```
        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  0.00  0.0     1
```

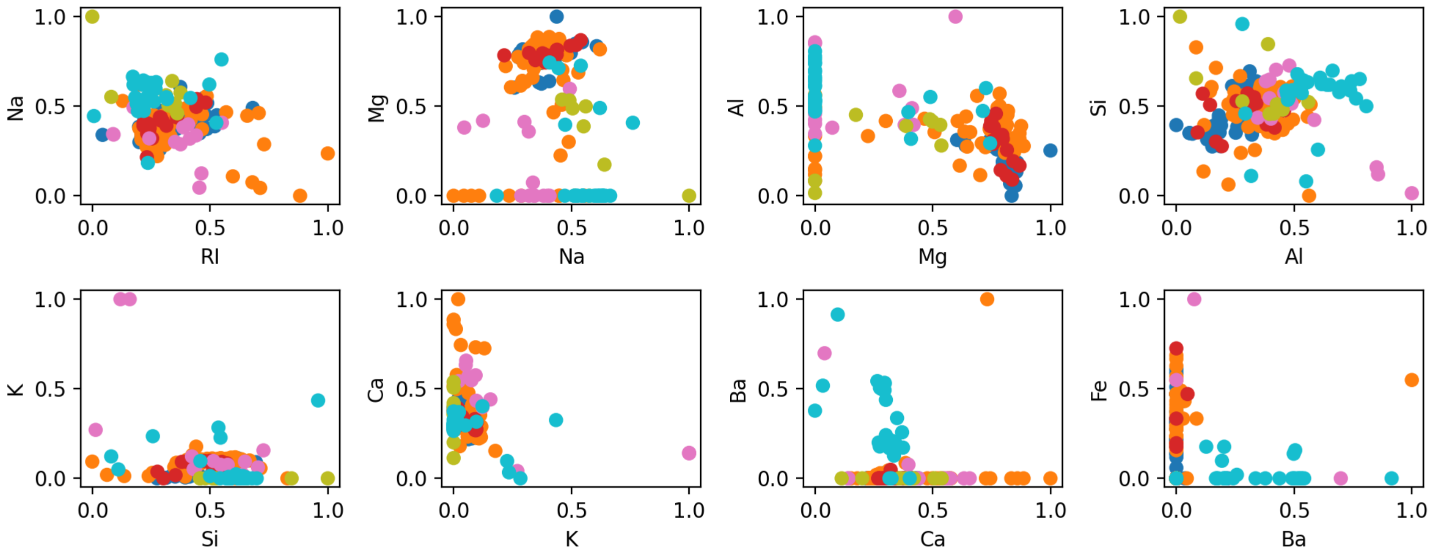
```
1    1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.00  0.0     1
2    1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.00  0.0     1
3    1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.00  0.0     1
4    1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.00  0.0     1
..       ...    ...   ...   ...    ...   ...   ...   ...  ...   ...
209  1.51623  14.14  0.00  2.88  72.61  0.08  9.18  1.06  0.0     7
210  1.51685  14.92  0.00  1.99  73.06  0.00  8.40  1.59  0.0     7
211  1.52065  14.36  0.00  2.02  73.42  0.00  8.44  1.64  0.0     7
212  1.51651  14.38  0.00  1.94  73.61  0.00  8.48  1.57  0.0     7
213  1.51711  14.23  0.00  2.08  73.36  0.00  8.62  1.67  0.0     7

[214 rows x 10 columns]
```

1.2 Нормировка данных

```python
from sklearn import preprocessing

data = preprocessing.minmax_scale(data)
```

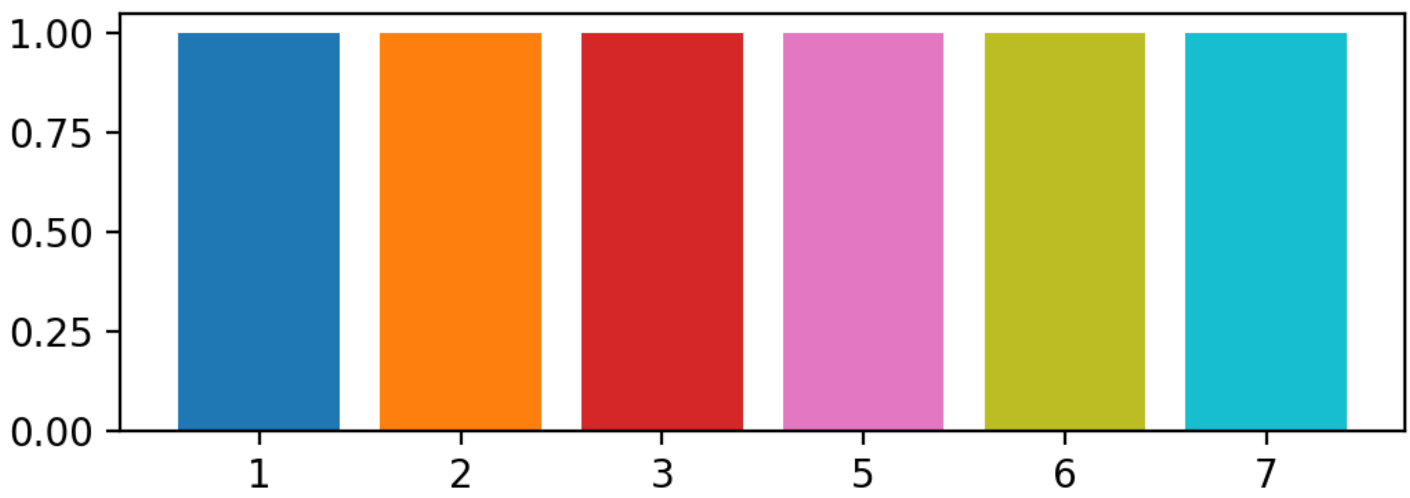
1.3 Диаграмма рассеяния

```python
from matplotlib import pyplot as plt
import matplotlib as mpl

mpl.rcParams['figure.dpi'] = 200
mpl.rcParams['figure.facecolor'] = '1'

def plot_scatter(data):
    fig, axes = plt.subplots(2, 4, figsize=(10, 4))

    for i in range(data.shape[1] - 1):
        axes[i // 4, i % 4].scatter(data[:,i], data[:,(i+1)], c=labels, cmap='tab10')

        axes[i // 4, i % 4].set_xlabel(var_names[i])
        axes[i // 4, i % 4].set_ylabel(var_names[i+1])
        # axes[i // 4, i % 4].legend()

    return fig

fig = plot_scatter(data)
fig.tight_layout()
```



1.4 Соответствие цвета и класса

```python
from matplotlib import cm, colors

norm = colors.Normalize(vmin=min(labels), vmax=max(labels))
cmap = cm.get_cmap('tab10')
unique_labels = list(set(labels))
label_colors = [cmap(norm(label)) for label in unique_labels]
fig, ax = plt.subplots(figsize=(6, 2))

ax.bar(range(len(unique_labels)), 1, color=label_colors)
ax.set_xticklabels([0, *unique_labels])
pass
```



1.5 Метод главных компонент

```python
from sklearn.decomposition import PCA
```

```
pca = PCA(n_components=2)
pca_data = pca.fit_transform(data)
print(data.shape, pca_data.shape)
```

```
(214, 9) (214, 2)
```

1.5.1 Значение объясненной дисперсии и собственные числа

Python

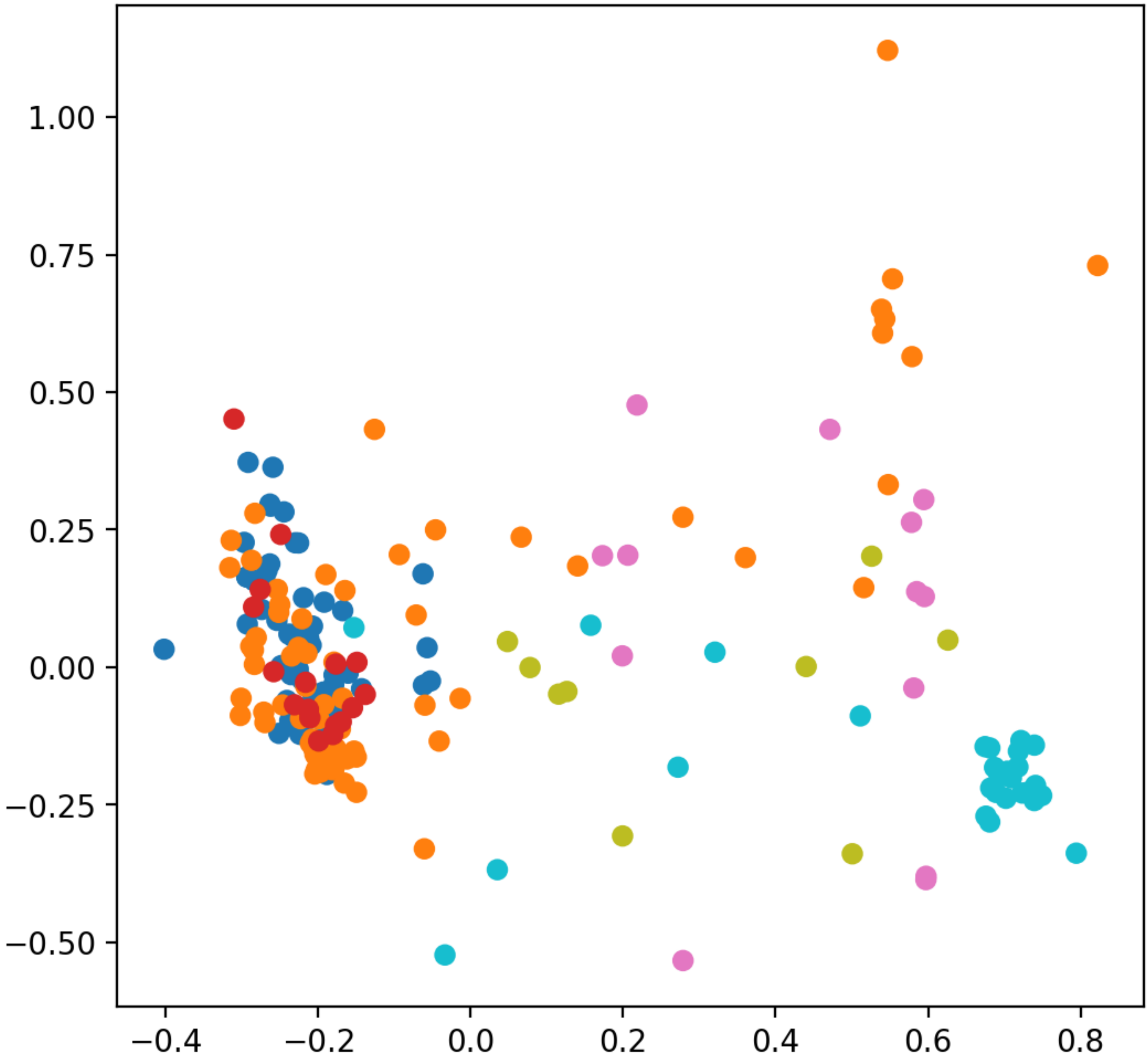
```
print(pca.explained_variance_ratio_)
print(pca.singular_values_)

with open('./output/file1.tex', 'w') as f:
    f.write(f'explained_variance_ratio_: {pca.explained_variance_ratio_}\n')
    f.write(f'singular_values: {pca.singular_values_}\n')
```

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

1.5.2 Диаграмма рассеяния после метода главных компонент

Python

```
fig, ax = plt.subplots(figsize=(6, 6))
ax.scatter(pca_data[:,0], pca_data[:,1], c=labels, cmap='tab10')

pass
```



1.5.3 Количество компонент для 85% дисперсии

```python
from tabulate import tabulate

def get_variances(threshold, get_pca, max_n=9):
    n_components = 2
    variances = []

    pca_ret, pca_data_ret = None, None

    while n_components <= max_n:
        pca_2 = get_pca(n_components=n_components)
        pca_data_2 = pca_2.fit_transform(data)
        variance = np.sum(pca_2.explained_variance_ratio_)
        variances.append((n_components, variance))
        if variance > threshold and pca_ret is None:
            pca_ret = pca_2

            pca_data_ret = pca_data_2
```

```python
        n_components += 1

    return np.array(variances), pca_ret, pca_data_ret

variances, pca_2, pca_data_2 = get_variances(0.85, lambda n_components: PCA(n_components=n_components))

with open('./output/table_dispersion.tex', 'w') as f:
    f.write(tabulate(variances, headers=['n_components', 'variance'], tablefmt='latex_booktabs'))

print(tabulate(variances, headers=['n_components', 'variance'], tablefmt='orgtbl'))
```

```
|   n_components |   variance |
|----------------+------------|
|              2 |   0.634197 |
|              3 |   0.760691 |
|              4 |   0.85867  |
|              5 |   0.927294 |
|              6 |   0.969435 |
|              7 |   0.995533 |
|              8 |   0.999861 |
|              9 |   1        |
```

1.5.4 Обратное преобразование

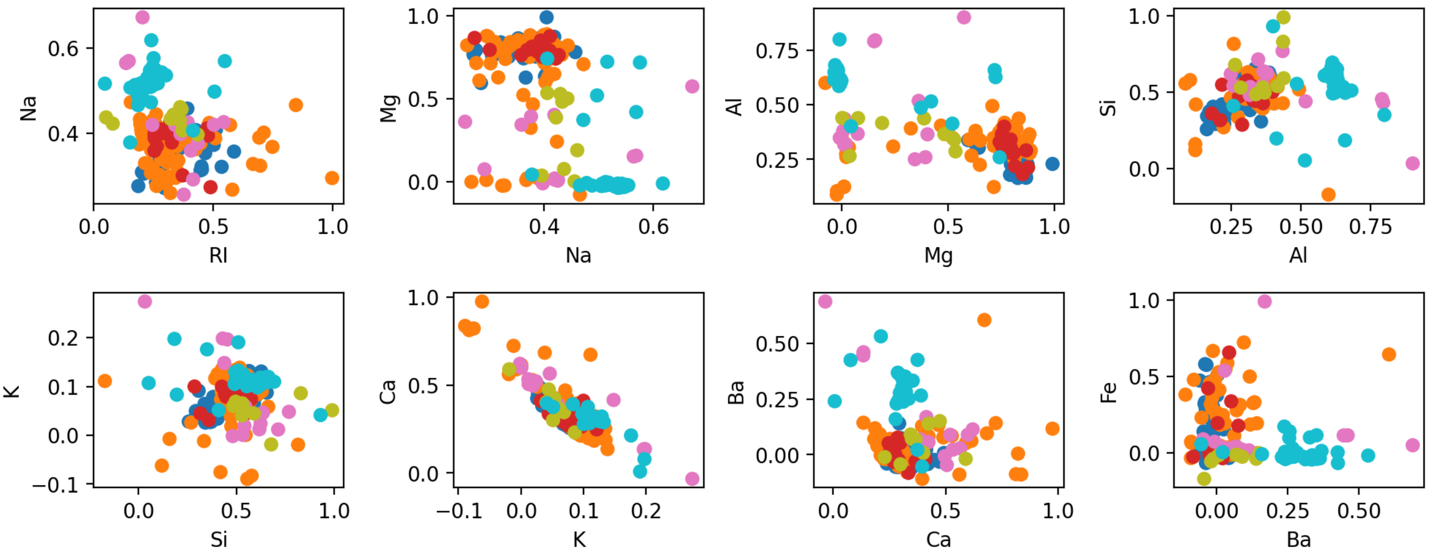
Python
```python
inverse_data = pca_2.inverse_transform(pca_data_2)
print(pca_data_2.shape, inverse_data.shape)
```

```
(214, 4) (214, 9)
```

Python
```python
def calculate_error(data, reverse_data):
    mse = (np.square(data - reverse_data)).mean(axis=None)
    return mse

print(calculate_error(data, inverse_data))
```

```
0.0042093981609607495
```

Python
```python
fig = plot_scatter(inverse_data)
fig.tight_layout()
```



1.5.5 Параметр `svd_solver`

Стандартное значение - `auto`

```
Python
results = []
options = 'auto', 'full', 'arpack', 'randomized'
for svd_solver in options:
    variances, _, _ = get_variances(0.85, lambda n_components: PCA(n_components=n_components, svd_solver
    if len(results) == 0:
        results.append(variances[:,0])
    results.append(variances[:,1])

results = np.array(results).T

print(tabulate(results, headers=['n_components', *options], tablefmt='orgtbl'))
with open('./output/table_solver.tex', 'w') as f:
    f.write(tabulate(results, headers=['n_components', *options], tablefmt='latex_booktabs'))
```

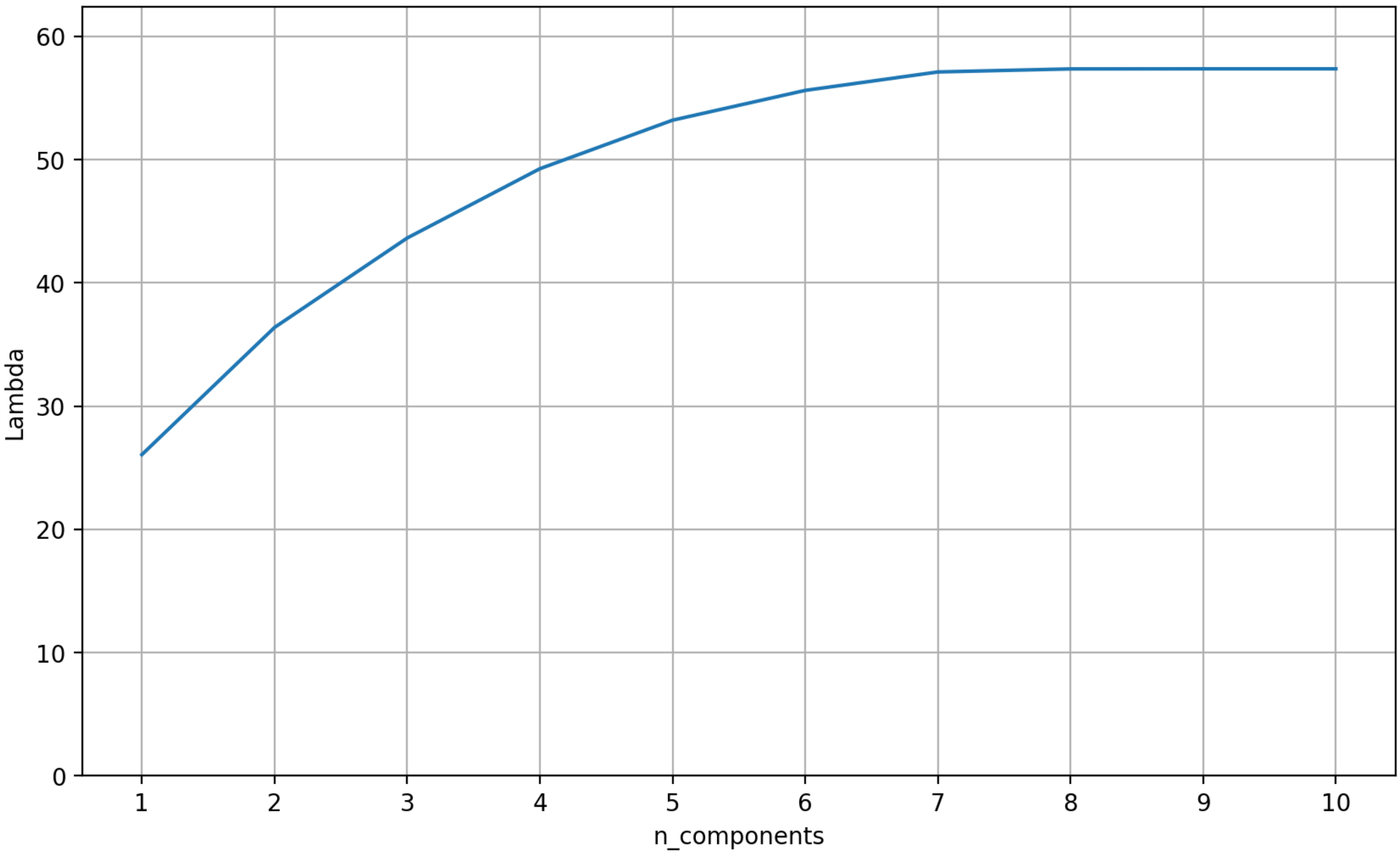
```
|   n_components |     auto |     full |   arpack |   randomized |
|----------------+----------+----------+----------+--------------|
|              2 | 0.634197 | 0.634197 | 0.634197 |     0.634197 |
|              3 | 0.760691 | 0.760691 | 0.760691 |     0.760691 |
|              4 | 0.85867  | 0.85867  | 0.85867  |     0.85867  |
|              5 | 0.927294 | 0.927294 | 0.927294 |     0.927294 |
|              6 | 0.969435 | 0.969435 | 0.969435 |     0.969435 |
|              7 | 0.995533 | 0.995533 | 0.995533 |     0.995533 |
|              8 | 0.999861 | 0.999861 | 0.999861 |     0.999861 |
```

1.6 KernelPCA

1.6.1 `KernelPCA` с линейным ядром

```
Python
from sklearn.decomposition import KernelPCA
```

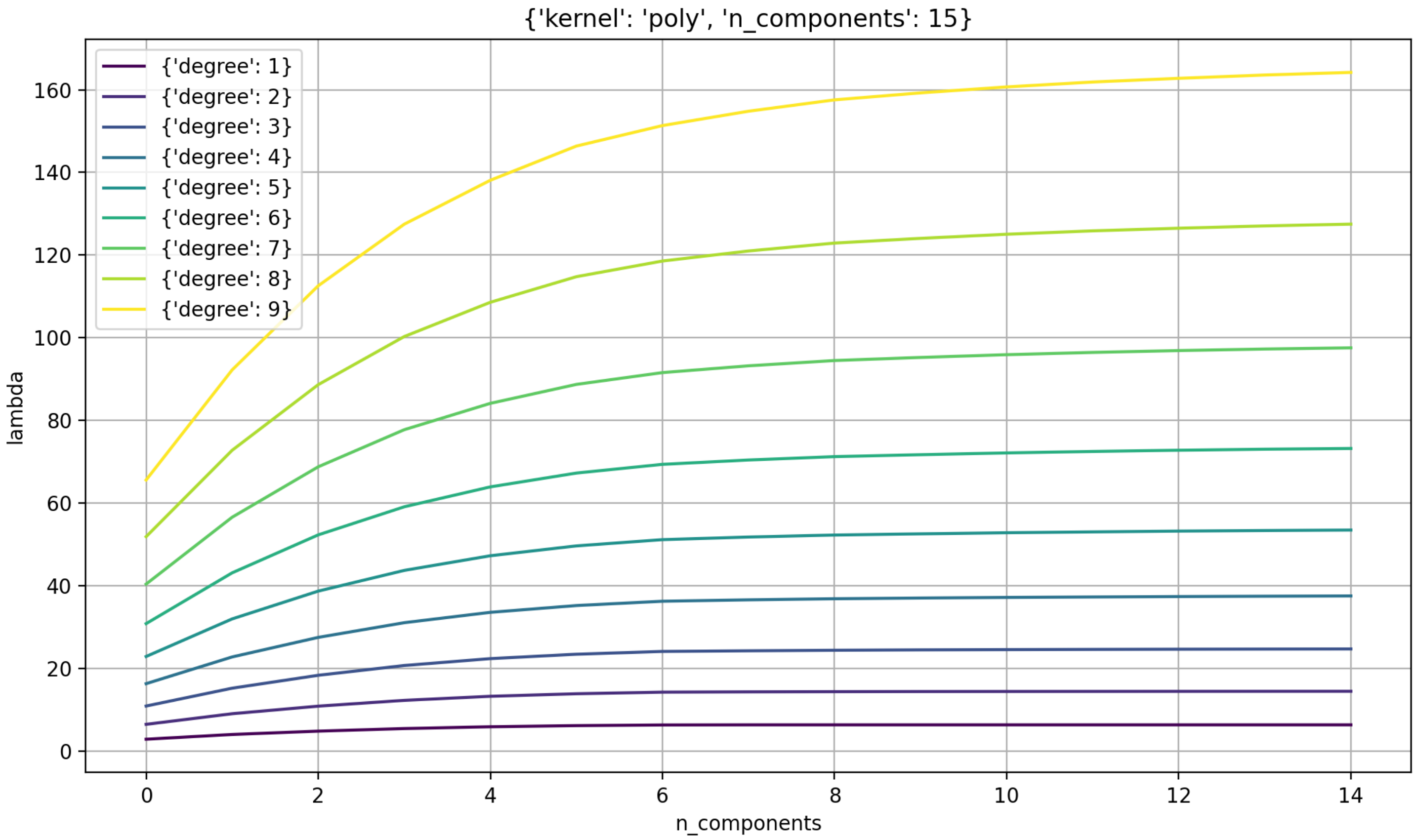
```python
kernelpca = KernelPCA(n_components=10, kernel='linear')
kernelpca_data = kernelpca.fit_transform(data)

cum_lambas = np.cumsum(kernelpca.lambdas_)

fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(cum_lambas)
ax.grid(True)
ax.set_ylim(0, max(cum_lambas) + 5)
ax.set_xlabel('n_components')
ax.set_ylabel('Lambda')
ax.set_xticklabels(range(1, 11))
ax.set_xticks(range(0, 10))

pass
```



1.6.2 Полиномиальное ядро

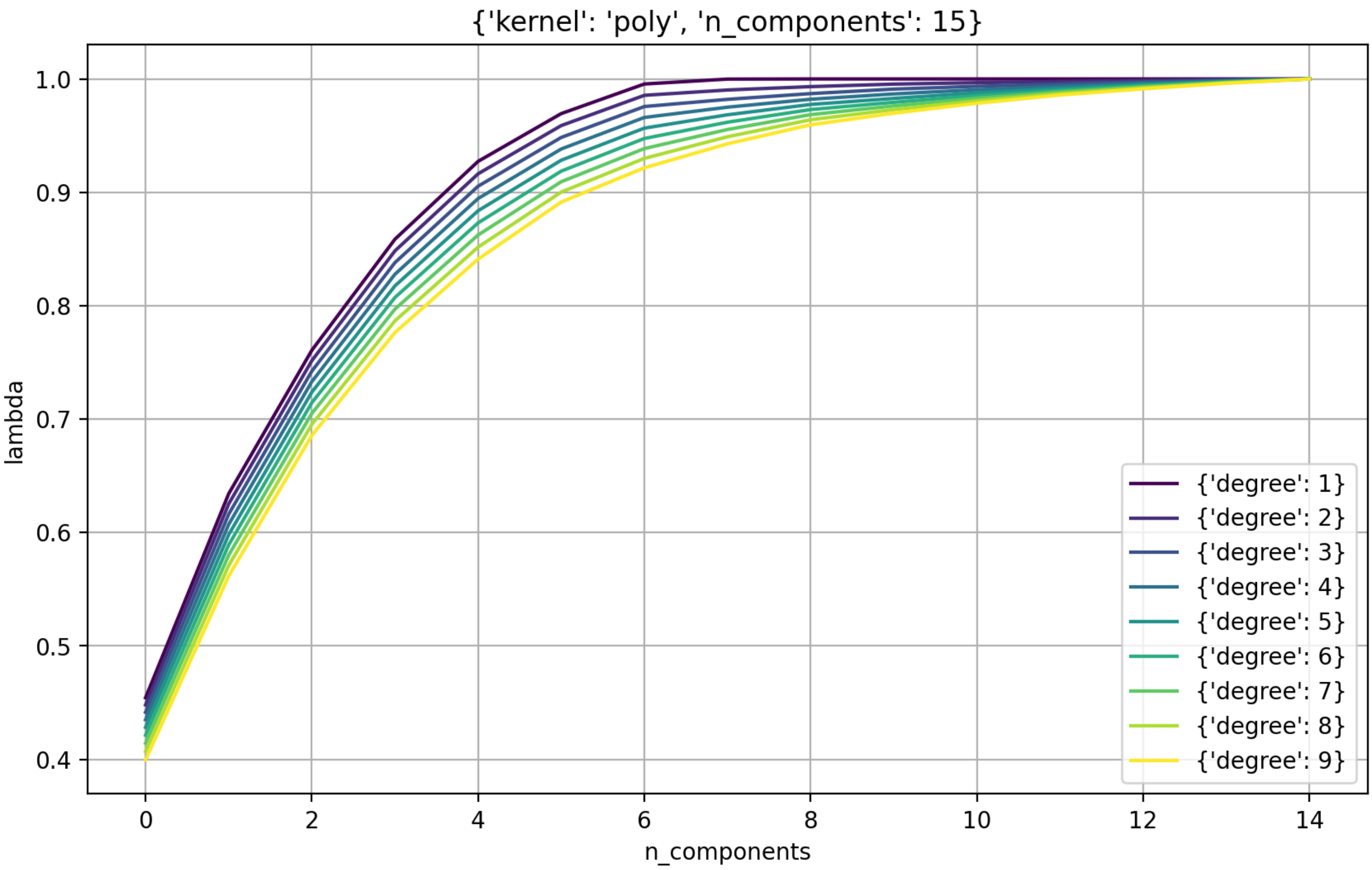
```python
def plot_labmdas(params, xlabel='lambda', norm=False, **kwargs):
    fig, ax = plt.subplots(figsize=(10, 6))
    colors = plt.cm.viridis(np.linspace(0, 1, len(params)))

    ax.set_title(str(kwargs))

    for i, param in enumerate(params):
        kpca = KernelPCA(**param, **kwargs)
        kpca.fit_transform(data)
        plot_data = np.cumsum(kpca.lambdas_)
        if norm:
```

```
            plot_data = plot_data / np.max(plot_data)
        ax.plot(plot_data, label=str(param), color=colors[i])

    ax.legend()
    ax.grid()
    ax.set_xlabel('n_components')
    ax.set_ylabel(xlabel)
    return fig

fig = plot_labmdas([{ 'degree': i } for i in range(1, 10)], kernel='poly', n_components=15)

fig.tight_layout()
```



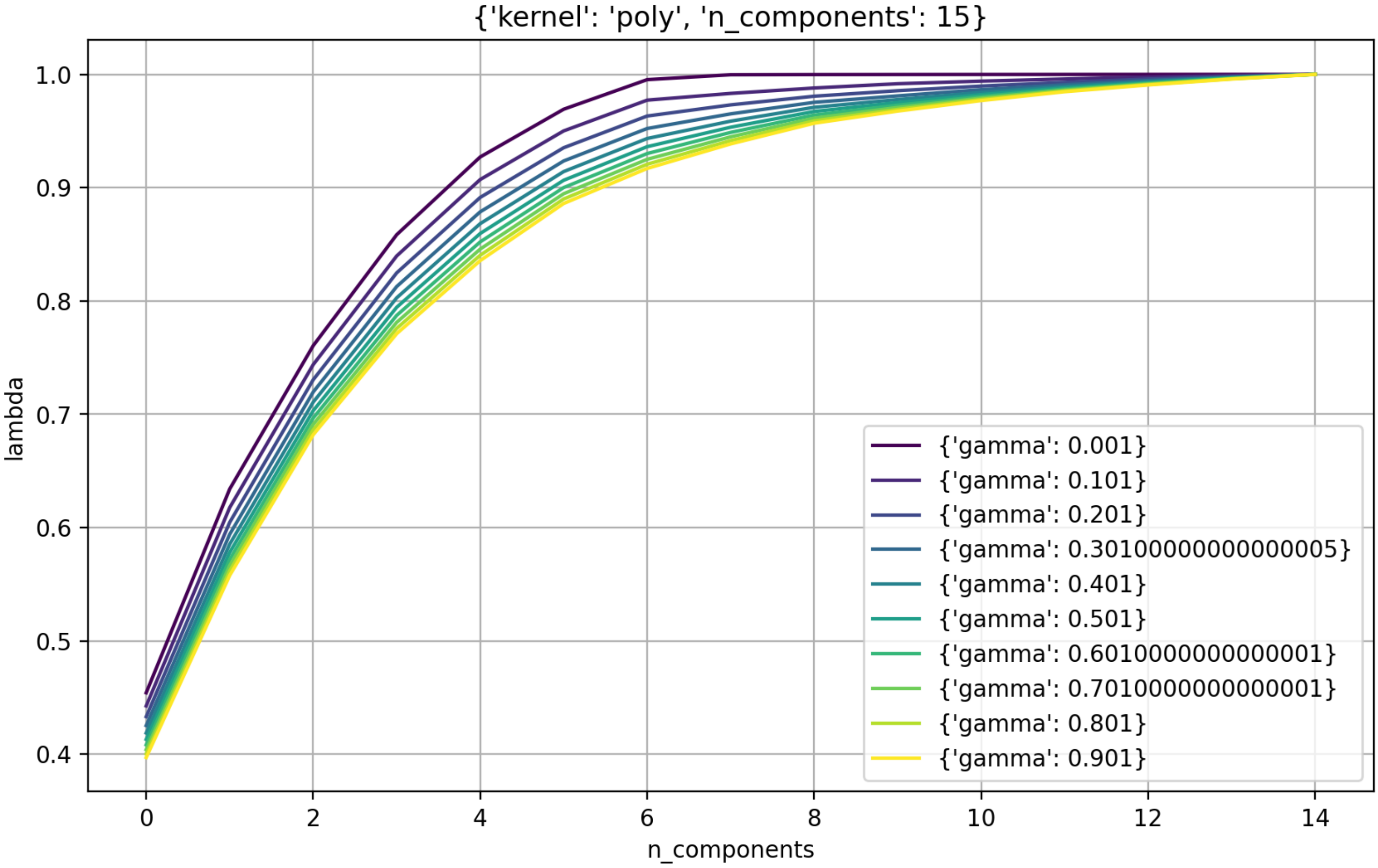
1.6.3 Черт знает что

```python
fig = plot_labmdas([{ 'degree': i } for i in range(1, 10)], norm=True, kernel='poly', n_components=15)
```



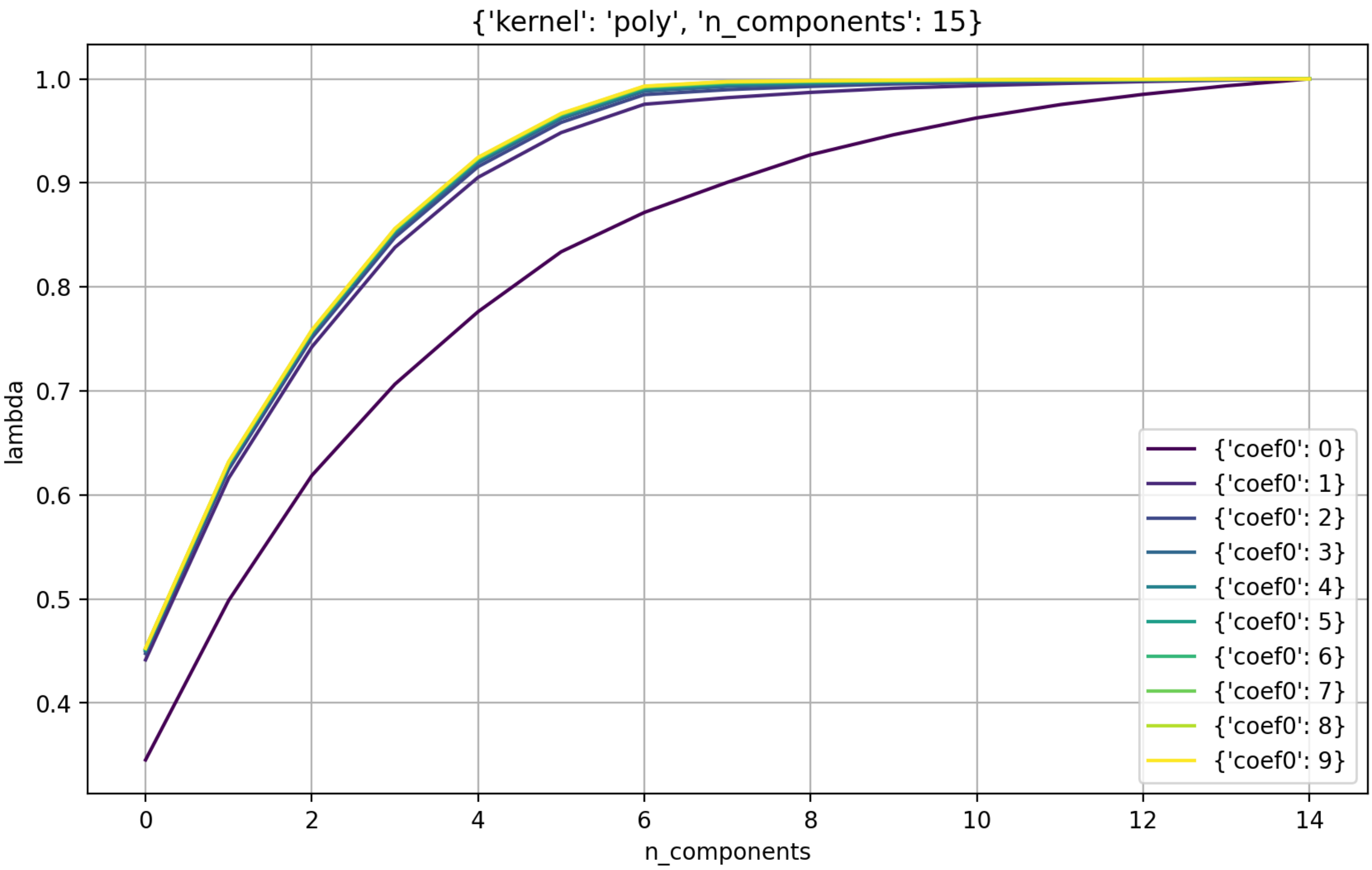
1.6.4 Полиномиальное ядро и gamma

```python
fig = plot_labmdas([{ 'gamma': i } for i in np.arange(1/1000, 1, 1/10)], norm=True, kernel='poly', n_com
```



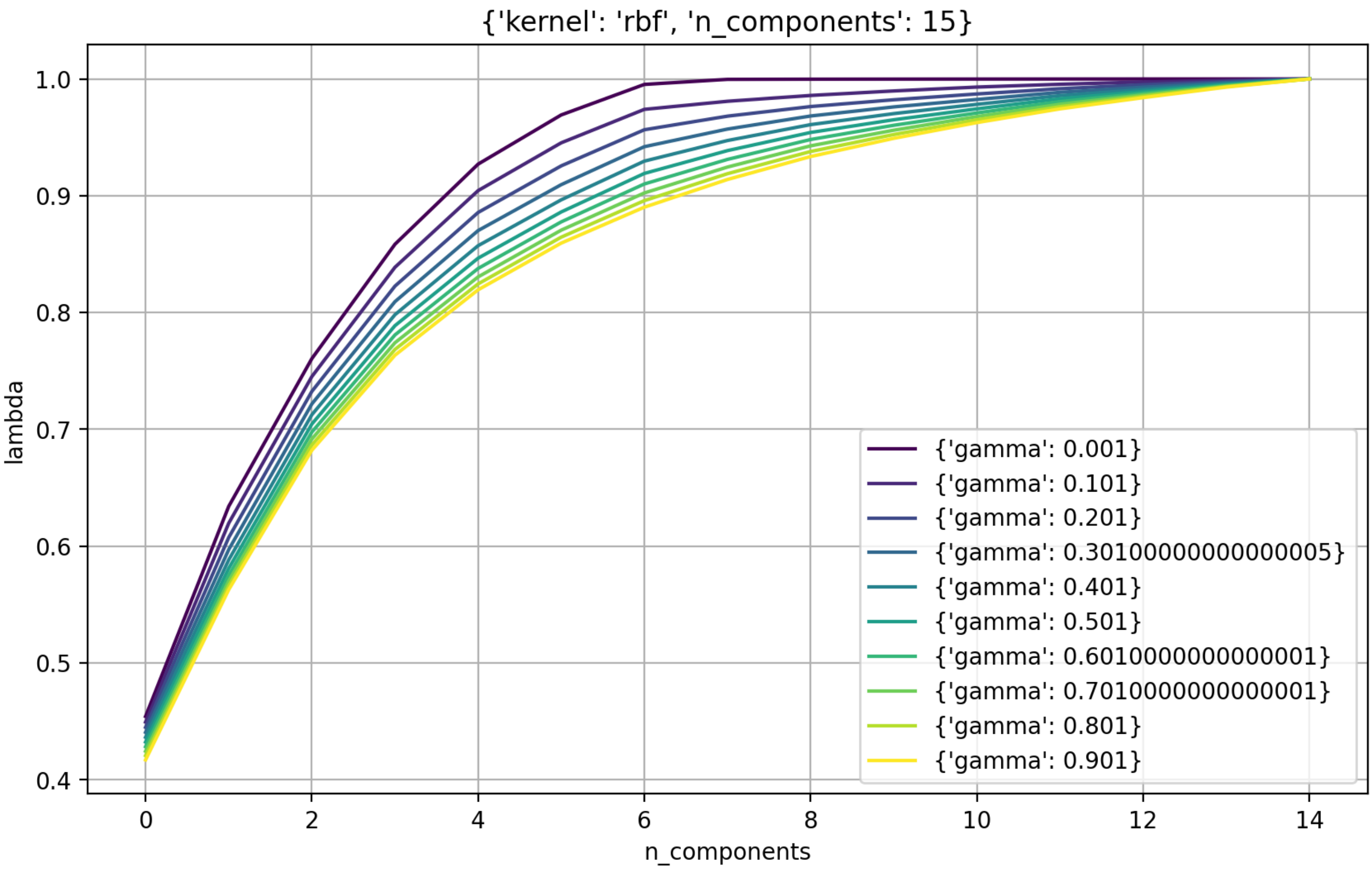
1.6.5 Полиномиальное ядро и coef0

```python
fig = plot_labmdas([{ 'coef0': i } for i in range(0, 10)], norm=True, kernel='poly', n_components=15)
```



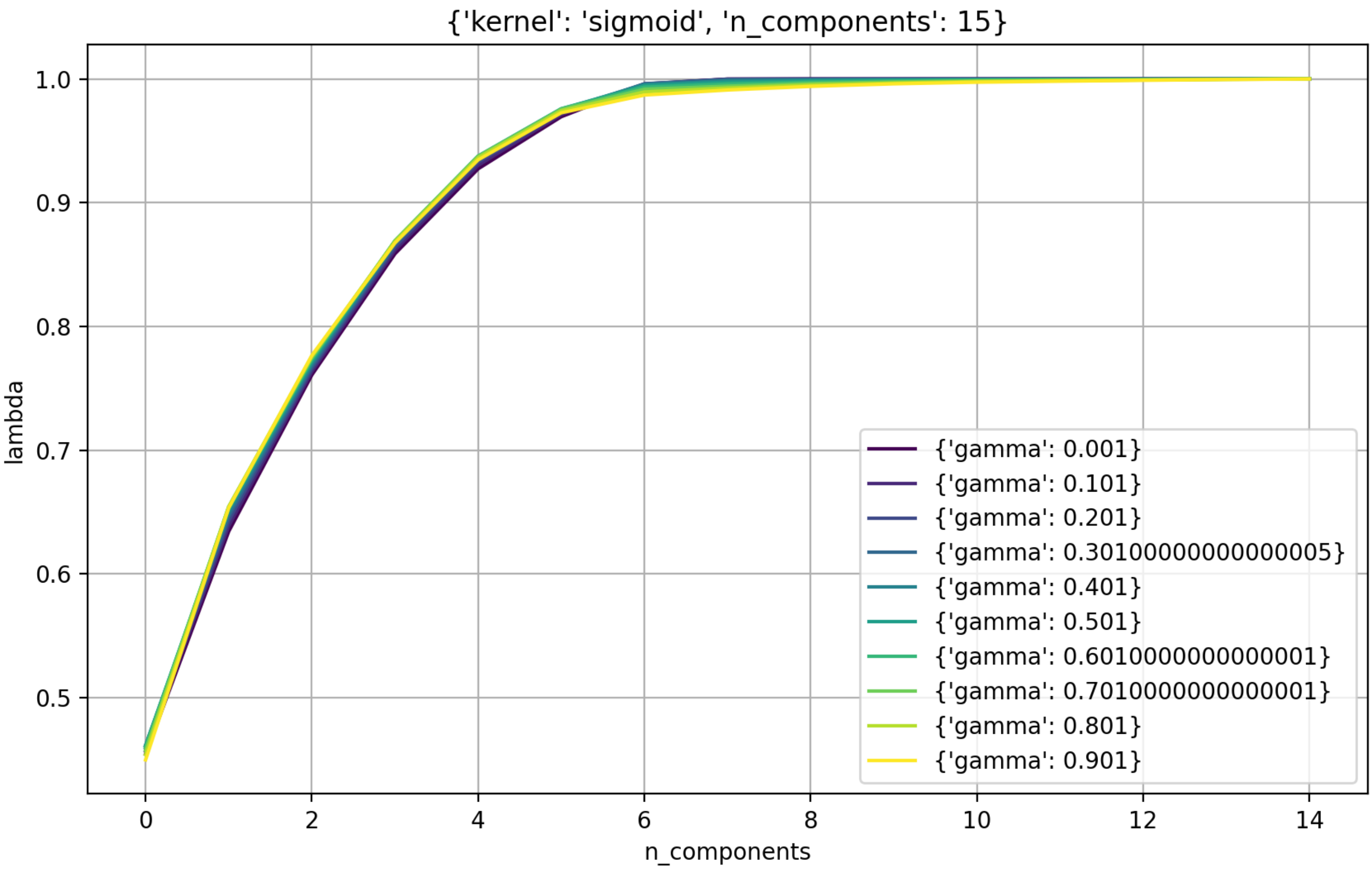
1.6.6 RBF и gamma

```python
fig = plot_labmdas([{ 'gamma': i } for i in np.arange(1/1000, 1, 1/10)], norm=True, kernel='rbf', n_comp
```



1.6.7 Сигмоидальное ядро и gamma

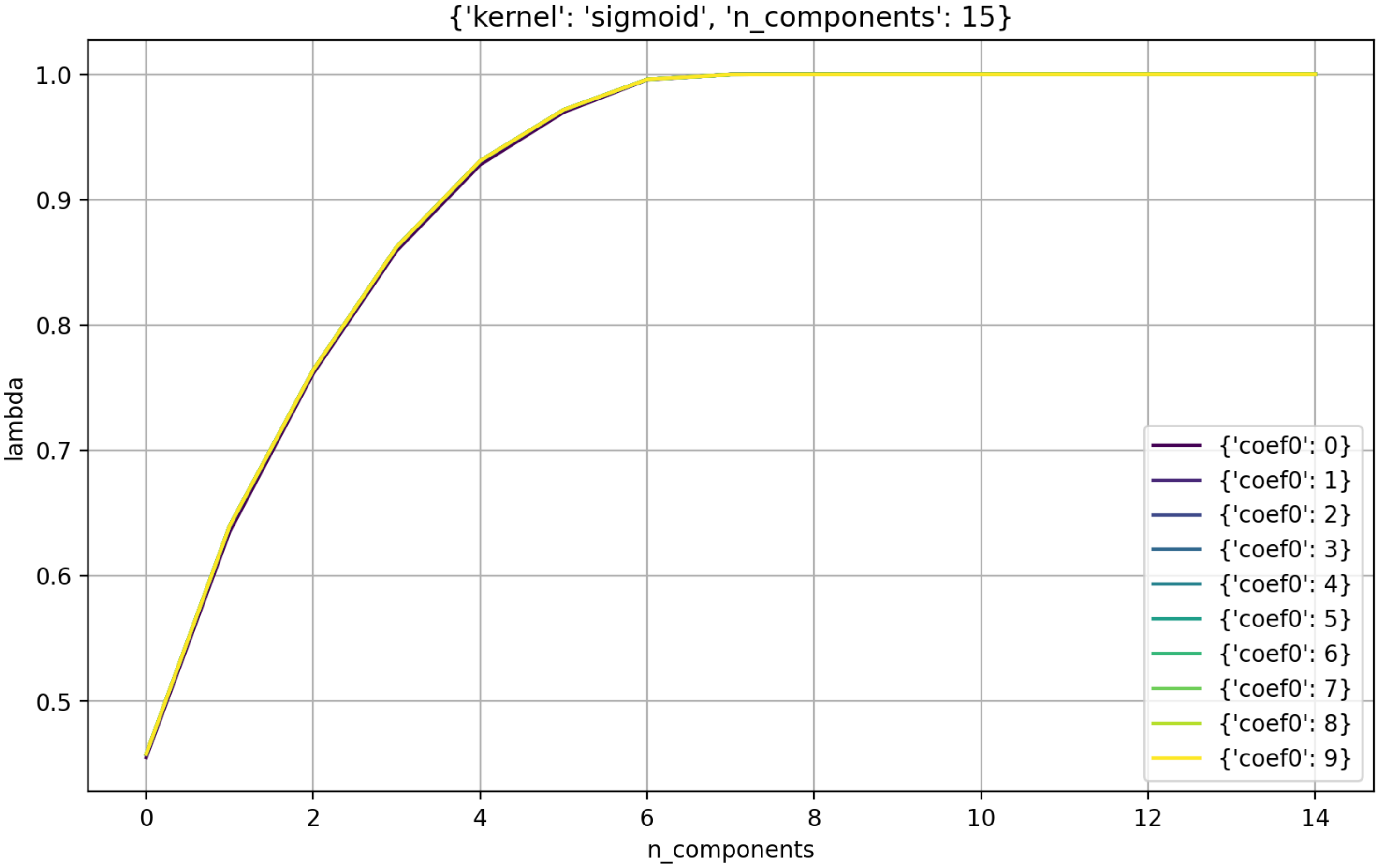
```python
fig = plot_labmdas([{ 'gamma': i } for i in np.arange(1/1000, 1, 1/10)], norm=True, kernel='sigmoid', n_
```



1.6.8 Сигмоидальное ядро и coef0

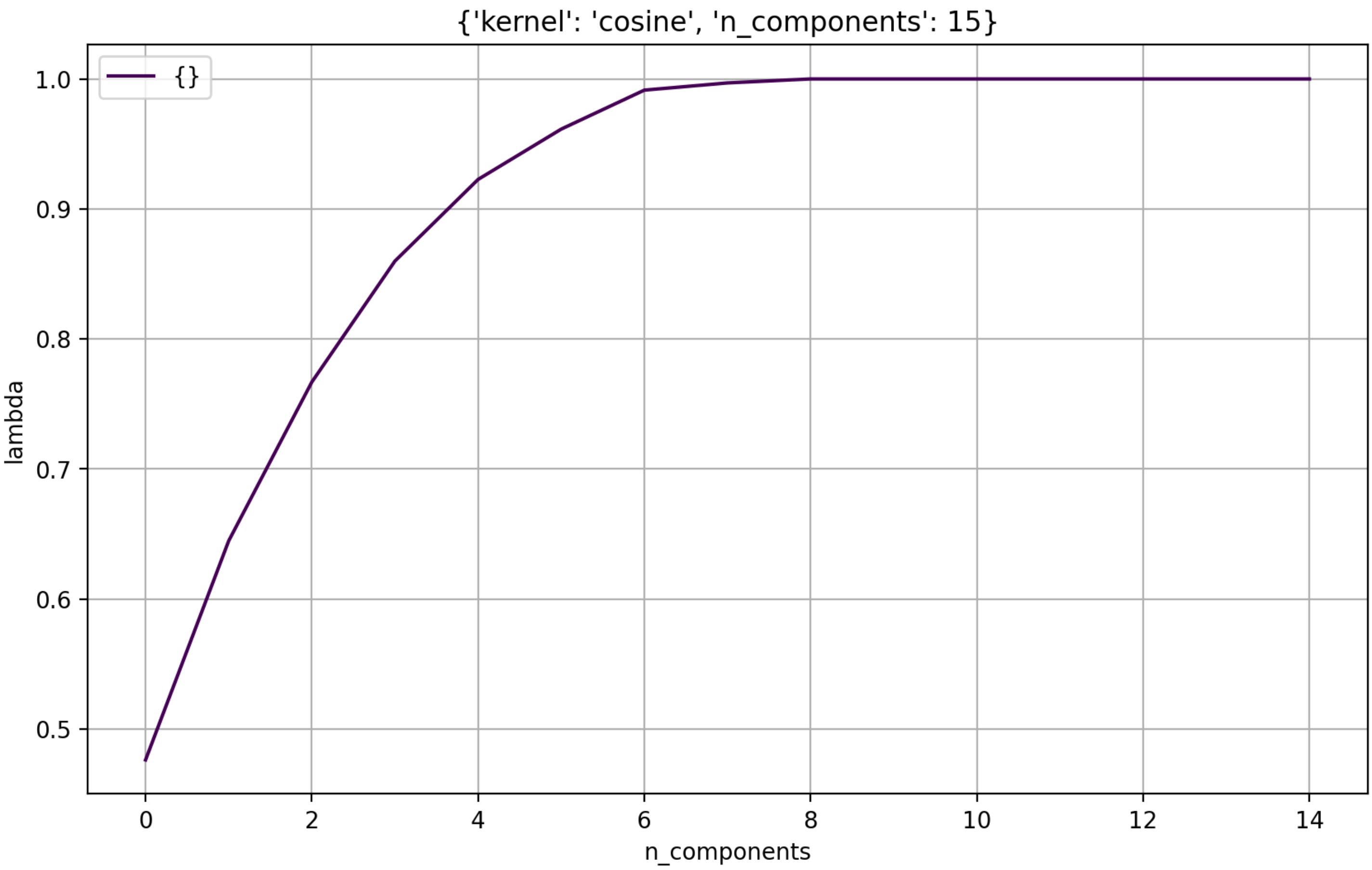
Python

```
fig = plot_labmdas([{ 'coef0': i } for i in range(0, 10)], norm=True, kernel='sigmoid', n_components=15)
```



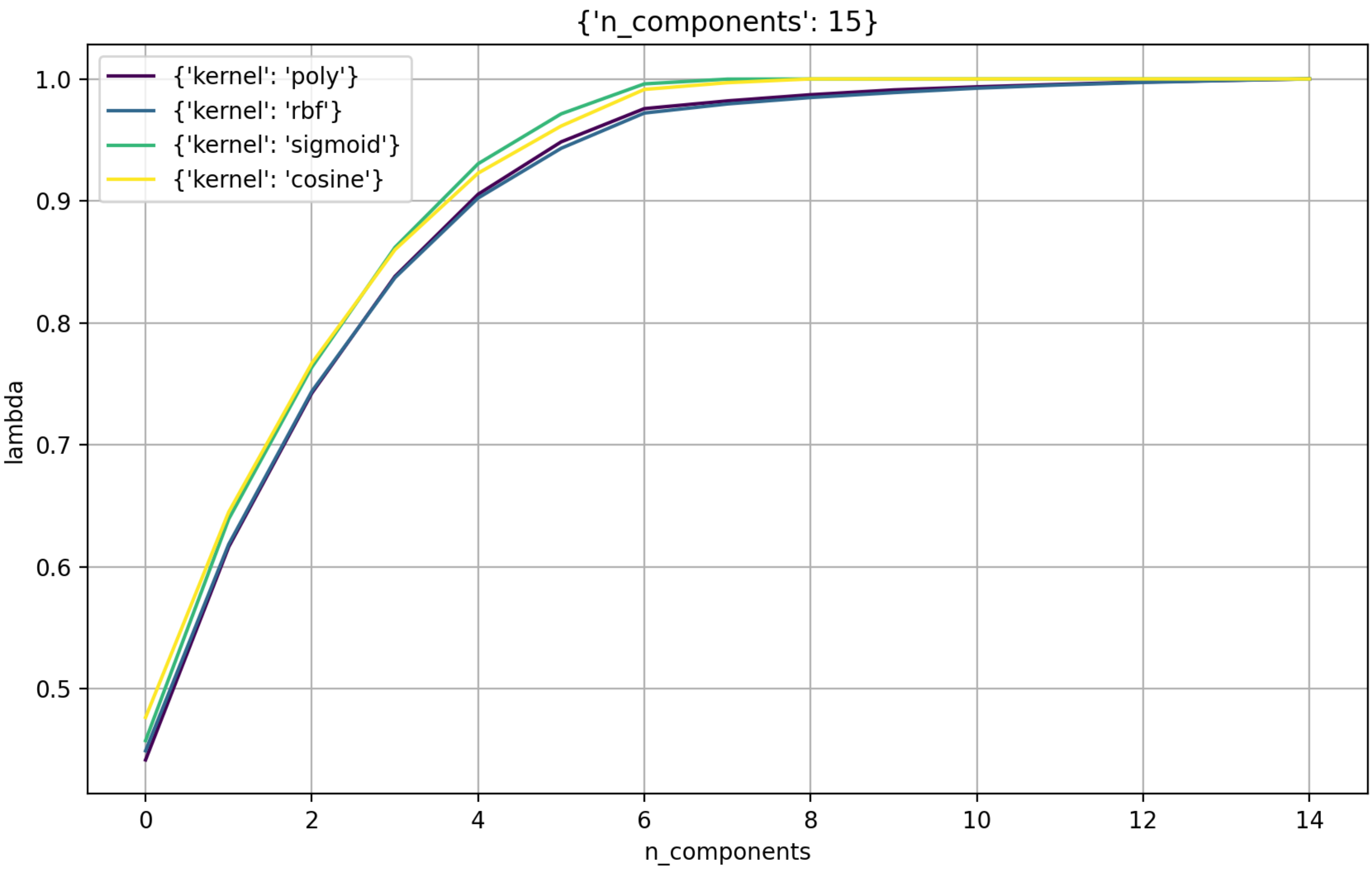
1.6.9 cosine

```python
fig = plot_labmdas([{}], norm=True, kernel='cosine', n_components=15)
```



1.6.10 Сравнение всего

```python
fig = plot_labmdas([
    { 'kernel': 'poly' },
    { 'kernel': 'rbf' },
    { 'kernel': 'sigmoid' },
    { 'kernel': 'cosine' },
], norm=True, n_components=15)
```



1.7 SparsePCA

```python
from sklearn.decomposition import SparsePCA

sparce_pca = SparsePCA(n_components=2)

spca_data = sparce_pca.fit_transform(data)
print(data.shape, pca_data.shape)
```

```
(214, 9) (214, 2)
```

1.7.1 Сравнение

```python
print(np.mean(sparce_pca.components_ == 0))
print(np.mean(pca.components_ == 0))

print(tabulate(sparce_pca.components_, tablefmt='fancy_grid', floatfmt='.4f'))
print(tabulate(pca.components_, tablefmt='fancy_grid', floatfmt='.4f'))

with open('./output/sparsepca_data.txt', 'w') as f:
    f.write('Компоненты SparcePCA:\n')
    f.write(tabulate(sparce_pca.components_, tablefmt='simple', floatfmt=".3f"))

    f.write('\nКомпоненты PCA:\n')
```

```python
    f.write(tabulate(pca.components_, tablefmt='simple', floatfmt=".3f"))
```

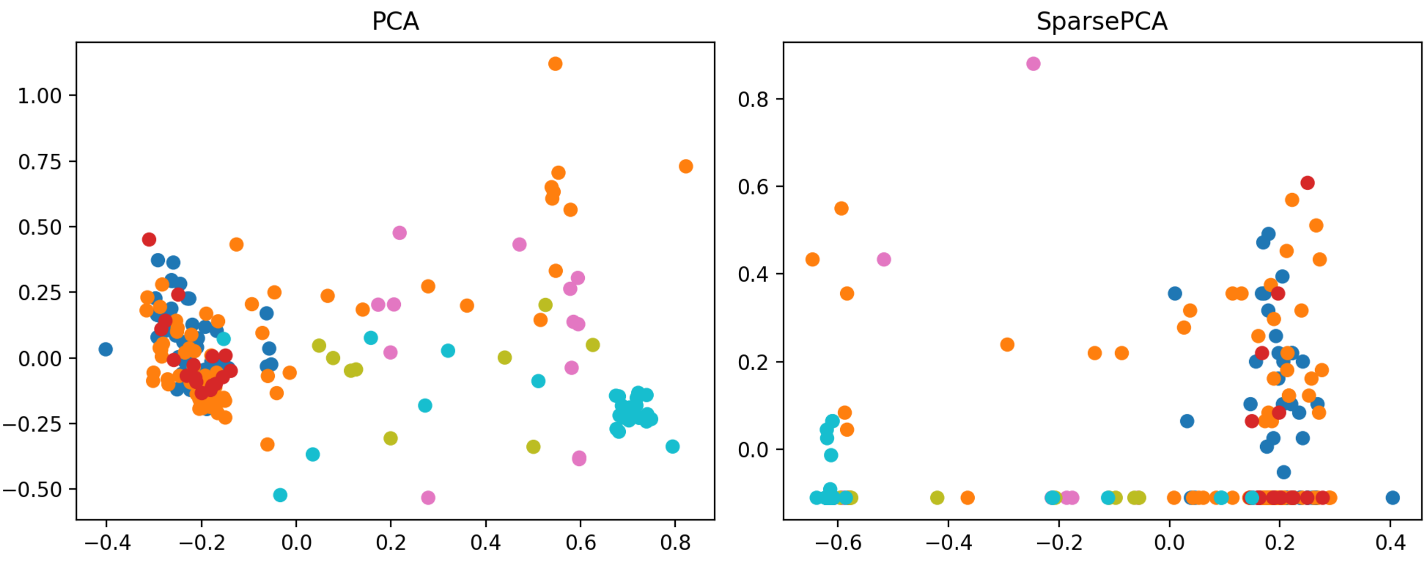
```
0.7777777777777778
0.0
```

0.0000	0.0000	0.9980	-0.0372	0.0000	0.0000	0.0000	-0.0503	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

0.0342	0.1104	-0.9090	0.2490	0.0508	-0.0027	0.1409	0.2668	-0.0
0.5133	-0.1987	-0.1171	-0.3474	-0.2164	-0.1293	0.5023	-0.1643	0.4

1.7.2 Диаграмма рассеяния

```python
fig, [ax1, ax2] = plt.subplots(1, 2, figsize=(10, 4))
ax1.scatter(pca_data[:,0], pca_data[:,1], c=labels, cmap='tab10')
ax2.scatter(spca_data[:,0], spca_data[:,1], c=labels, cmap='tab10')
ax1.set_title('PCA')
ax2.set_title('SparsePCA')
fig.tight_layout()
pass
```



1.7.3 Параметры

```python
# fig, ax = plt.subplots(1, 1, figsize=(10, 6))

results = []


for alpha in 0, 0.01, 0.1, 1, 10:
```

```
    spca = SparsePCA(n_components=2, alpha=alpha)
    spca_data_2 = spca.fit_transform(data)

    for i, component in enumerate(spca.components_):
        results.append([alpha, i + 1, *component])

print(tabulate(results, headers=['alpha', 'i', *range(data.shape[1])], tablefmt='orgtbl'))

with open('./output/spca_alpha.tex', 'w') as f:
    f.write(tabulate(results, headers=['alpha', 'i', *range(data.shape[1])], tablefmt='latex_booktabs',
```

```
|   alpha |   i |           0 |           1 |          2 |           3 |           4 |
|---------+-----+-------------+-------------+------------+-------------+-------------+--
|    0    |   1 | -0.0342095  | -0.110442   |  0.909035  | -0.24902    | -0.0507955  |
|    0    |   2 |  0.513273   | -0.19867    | -0.1171    | -0.347363   | -0.216426   | -
|    0.01 |   1 | -0.100348   | -0.0811734  |  0.917807  | -0.199658   | -0.019656   |
|    0.01 |   2 |  0.504528   | -0.210305   |  0         | -0.375515   | -0.220352   | -
|    0.1  |   1 | -0.0665449  | -0.0714373  |  0.92832   | -0.202591   | -0.00954142 |
|    0.1  |   2 |  0.504693   | -0.199356   |  0         | -0.364529   | -0.2123     | -
|    1    |   1 |  0          |  0          |  0.998042  | -0.0371835  |  0          |
|    1    |   2 |  0          |  0          |  0         |  0          |  0          |
|   10    |   1 |  0          |  0          |  0         |  0          |  0          |
|   10    |   2 |  0          |  0          |  0         |  0          |  0          |
```

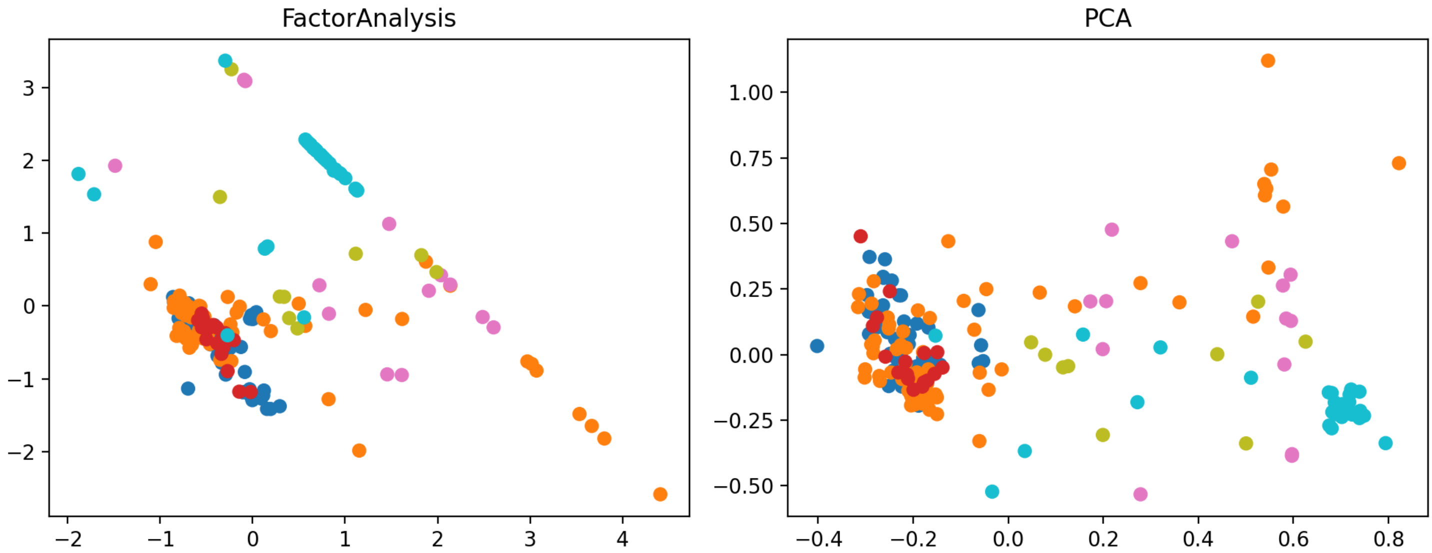
1.8 Факторный анализ

```python
from sklearn.decomposition import FactorAnalysis

fa = FactorAnalysis(n_components=2)
fa_data = fa.fit_transform(data)

print(data.shape, fa_data.shape)
```

```
(214, 9) (214, 2)
```

1.8.1 Диаграмма рассеяния

```python
fig, [ax1, ax2] = plt.subplots(1, 2, figsize=(10, 4))
ax1.scatter(fa_data[:,0], fa_data[:,1], c=labels, cmap='tab10')
ax2.scatter(pca_data[:,0], pca_data[:,1], c=labels, cmap='tab10')
ax1.set_title('FactorAnalysis')
ax2.set_title('PCA')
fig.tight_layout()
pass
```



Author: Pavel

Created: 2020-11-04 Ср 22:26

Validate