Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №1

По дисциплине: «Интеллектуальный анализ данных»

Тема: «PCA»

Выполнил:

Студент 4 курса

Группы ИИ-23

Бусень А.Д.

Проверила:

Андренко К.В.

Брест 2025

Цель работы: научиться применять метод PCA для осуществления визуализации данных.

Вариант 1.

№ варианта	Выборка	Класс
1	seeds.zip	Последняя колонка

Код программы:

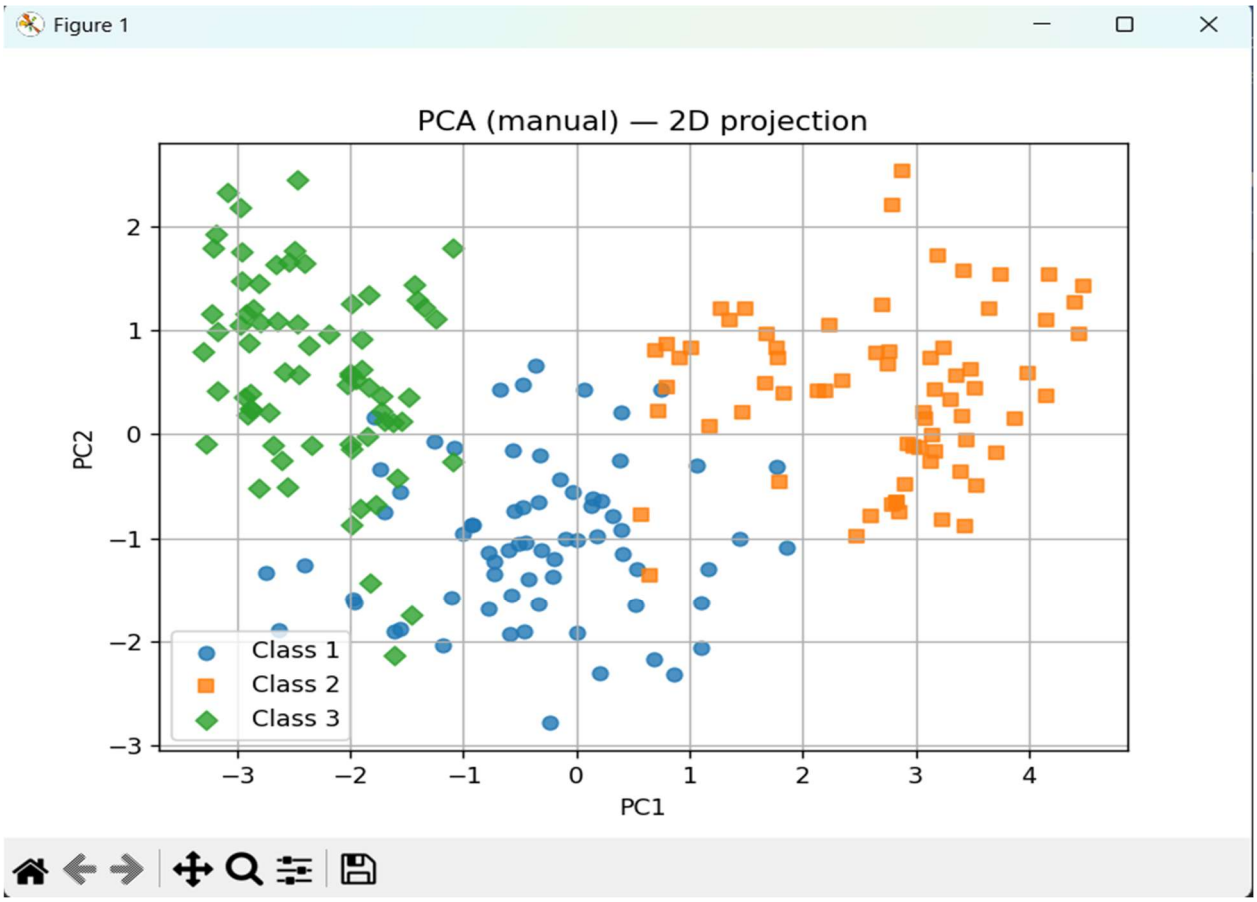
```
import os
import io
import zipfile
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA as SKPCA

local_zip_path = 'seeds.zip'

def load_from_zip(zip_path):
    with zipfile.ZipFile(zip_path, 'r') as z:
        names = [n for n in z.namelist() if n.lower().endswith(('.csv','.txt','.data'))]
        if not names:
            raise FileNotFoundError("В архиве нет табличных файлов (.csv/.txt/.data).")
        name = names[0]
        raw = z.read(name)
        df = pd.read_csv(io.BytesIO(raw), sep=r'\s+', engine='python', header=None)
        print(f"Загружен файл {name} из архива.")
        return df

if not os.path.exists(local_zip_path):
    raise FileNotFoundError(f"Файл {local_zip_path} не найден на ПК!")

df = load_from_zip(local_zip_path)
pd.set_option("display.max_rows", None)
pd.set_option("display.max_columns", None)
print(df)

n_cols = df.shape[1]
X = df.iloc[:, :-1].astype(float).copy()
y = df.iloc[:, -1].copy()

scaler = StandardScaler()
X_std = scaler.fit_transform(X)

cov = np.cov(X_std, rowvar=False)
eigvals, eigvecs = np.linalg.eig(cov)
eigvals = eigvals.real
eigvecs = eigvecs.real
```

```
order = np.argsort(eigvals)[::-1]
eigvals_sorted = eigvals[order]
eigvecs_sorted = eigvecs[:, order]
explained_ratio = eigvals_sorted / eigvals_sorted.sum()

proj2_manual = X_std.dot(eigvecs_sorted[:, :2])
proj3_manual = X_std.dot(eigvecs_sorted[:, :3])

pca2 = SKPCA(n_components=2)
proj2_sklearn = pca2.fit_transform(X_std)
pca3 = SKPCA(n_components=3)
proj3_sklearn = pca3.fit_transform(X_std)

unique_classes = sorted(np.unique(y))
markers = ['o','s','D','^','v','P','*']

plt.figure(figsize=(7,5))
for i, cls in enumerate(unique_classes):
   mask = (y == cls)
   plt.scatter(proj2_manual[mask,0], proj2_manual[mask,1], label=f'Class {cls}',
marker=markers[i%len(markers)], alpha=0.8)
plt.xlabel('PC1'); plt.ylabel('PC2'); plt.title('PCA (manual) — 2D projection'); plt.legend(); plt.grid(True)
plt.show()

plt.figure(figsize=(7,5))
for i, cls in enumerate(unique_classes):
   mask = (y == cls)
   plt.scatter(proj2_sklearn[mask,0], proj2_sklearn[mask,1], label=f'Class {cls}',
marker=markers[i%len(markers)], alpha=0.8)
plt.xlabel('PC1'); plt.ylabel('PC2'); plt.title('PCA (sklearn) — 2D projection'); plt.legend(); plt.grid(True)
plt.show()

fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(111, projection='3d')
for i, cls in enumerate(unique_classes):
   mask = (y == cls)
   ax.scatter(proj3_manual[mask,0], proj3_manual[mask,1], proj3_manual[mask,2], label=f'Class {cls}',
marker=markers[i%len(markers)], alpha=0.8)
ax.set_xlabel('PC1'); ax.set_ylabel('PC2'); ax.set_zlabel('PC3')
ax.set_title('PCA (manual) — 3D projection'); ax.legend()
plt.show()

fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(111, projection='3d')
for i, cls in enumerate(unique_classes):
   mask = (y == cls)
   ax.scatter(proj3_sklearn[mask,0], proj3_sklearn[mask,1], proj3_sklearn[mask,2], label=f'Class {cls}',
marker=markers[i%len(markers)], alpha=0.8)
ax.set_xlabel('PC1'); ax.set_ylabel('PC2'); ax.set_zlabel('PC3')
ax.set_title('PCA (sklearn) — 3D projection'); ax.legend()
plt.show()
```

```
total_var = eigvals_sorted.sum()
loss_2 = 1 - eigvals_sorted[:2].sum() / total_var
loss_3 = 1 - eigvals_sorted[:3].sum() / total_var

def reconstruct(X_std, eigvecs_sorted, k):
    Wk = eigvecs_sorted[:, :k]
    Z = X_std.dot(Wk)
    X_rec = Z.dot(Wk.T)
    return X_rec

X_rec_2 = reconstruct(X_std, eigvecs_sorted, 2)
X_rec_3 = reconstruct(X_std, eigvecs_sorted, 3)
mse2 = np.mean((X_std - X_rec_2)**2)
mse3 = np.mean((X_std - X_rec_3)**2)

print("Eigenvalues (sorted desc):")
for i,val in enumerate(eigvals_sorted):
    print(f"PC{i+1}: eigenvalue={val:.6f}, explained_ratio={explained_ratio[i]:.6f}")
print()
print(f"Доля дисперсии, потерянная при хранении 2 ПК: {loss_2:.6f} ({loss_2*100:.2f}%)")
print(f"Доля дисперсии, потерянная при хранении 3 ПК: {loss_3:.6f} ({loss_3*100:.2f}%)")
print()
print(f"MSE реконструкции (стандартизованные признаки): 2 ПК = {mse2:.6f}, 3 ПК = {mse3:.6f}")

out_dir = 'pca_results'
os.makedirs(out_dir, exist_ok=True)
pd.DataFrame(proj2_manual, columns=['PC1','PC2']).assign(Class=y.values) \
    .to_csv(os.path.join(out_dir, 'proj2_manual.csv'), index=False)
pd.DataFrame(proj3_manual, columns=['PC1','PC2','PC3']).assign(Class=y.values) \
    .to_csv(os.path.join(out_dir, 'proj3_manual.csv'), index=False)
pd.DataFrame({
    'PC': [f'PC{i+1}' for i in range(len(eigvals_sorted))],
    'Eigenvalue': eigvals_sorted,
    'ExplainedVarRatio': explained_ratio,
    'Cumulative': np.cumsum(explained_ratio)
}).to_csv(os.path.join(out_dir, 'explained_variance.csv'), index=False)
print("Результаты сохранены в папке:", out_dir)
```

Результат работы программы:

```
Eigenvalues (sorted desc):
PC1: eigenvalue=5.055274, explained_ratio=0.718743
PC2: eigenvalue=1.203303, explained_ratio=0.171082
PC3: eigenvalue=0.681247, explained_ratio=0.096858
PC4: eigenvalue=0.068692, explained_ratio=0.009766
PC5: eigenvalue=0.018803, explained_ratio=0.002673
PC6: eigenvalue=0.005358, explained_ratio=0.000762
PC7: eigenvalue=0.000816, explained_ratio=0.000116

Доля дисперсии, потерянная при хранении 2 ПК: 0.110175 (11.02%)
Доля дисперсии, потерянная при хранении 3 ПК: 0.013318 (1.33%)

MSE реконструкции (стандартизованные признаки): 2 ПК = 0.110175, 3 ПК = 0.013318
Результаты сохранены в папке: pca_results
```



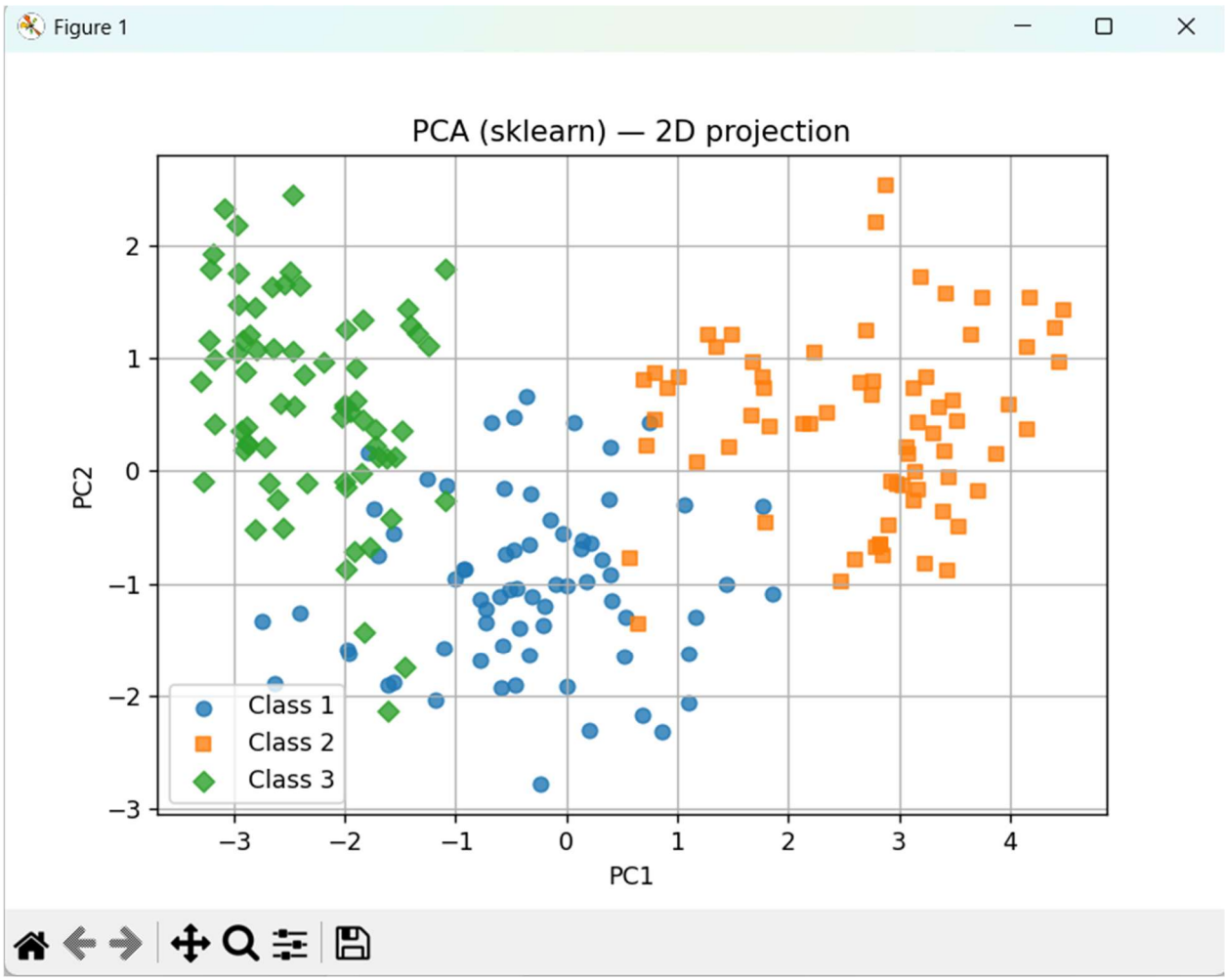


Figure 1
PCA (sklearn) — 2D projection
PC2
PC1
Class 1
Class 2
Class 3

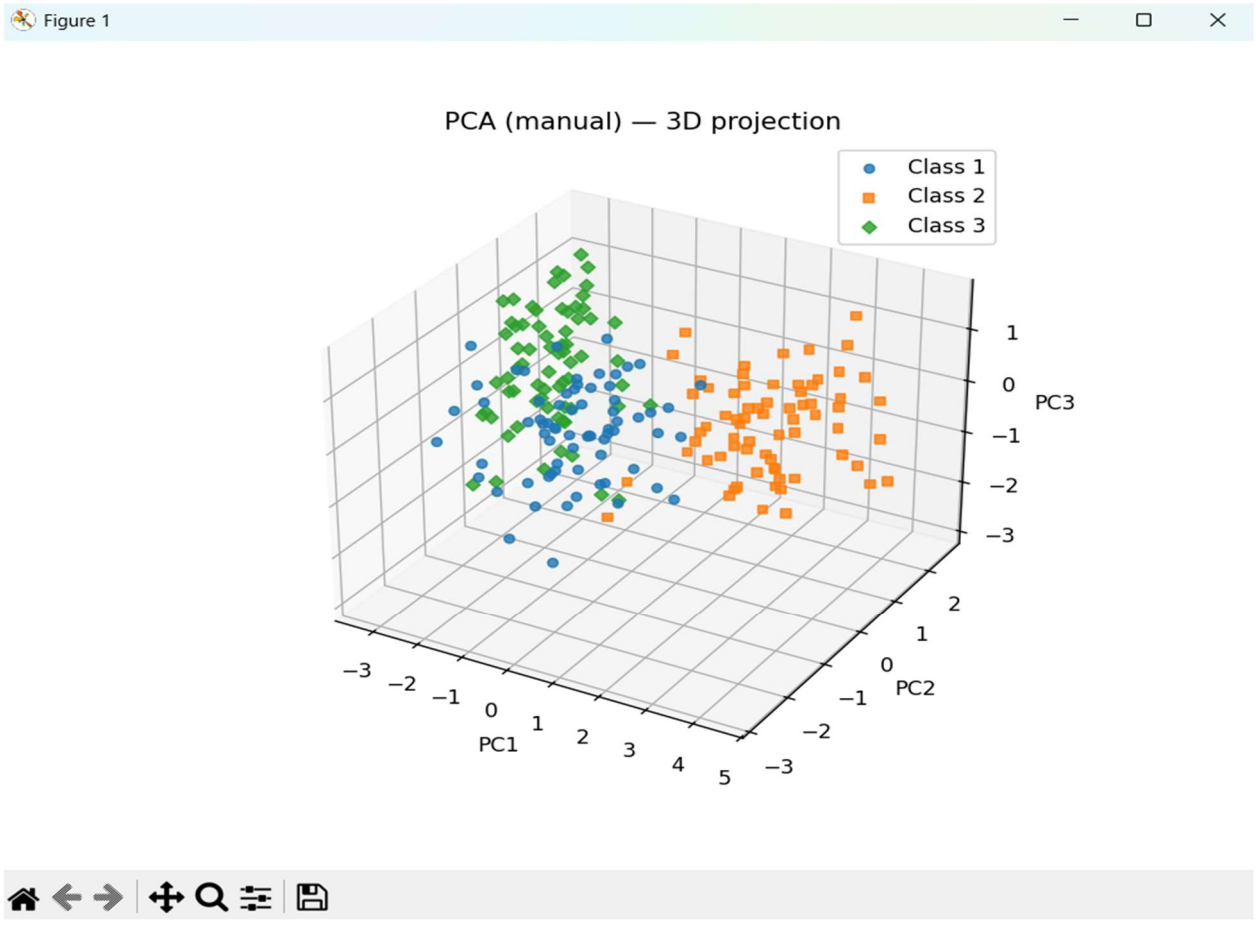
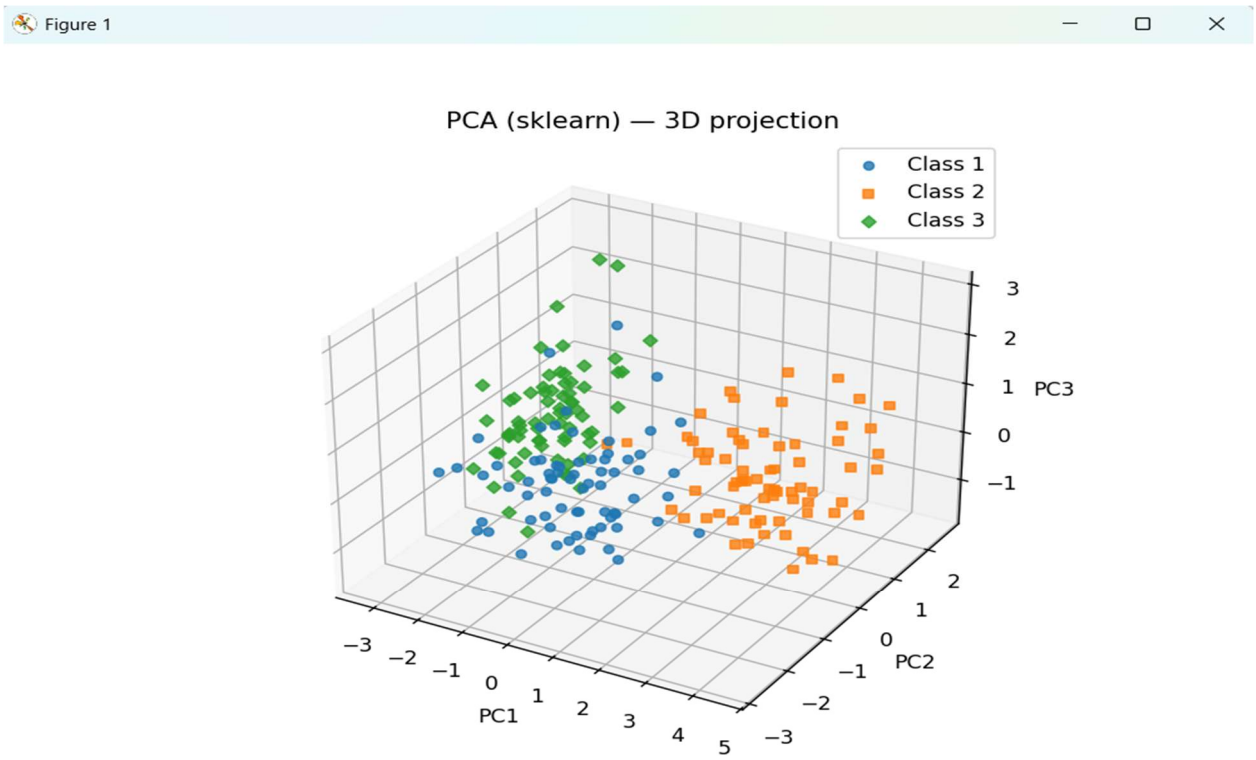


Figure 1
PCA (manual) — 3D projection
Class 1
Class 2
Class 3
PC1
PC2
PC3




```
def compare_projections(manual, sklearn_proj, k):
    diffs = []
    for i in range(k):
        # Корреляция между компонентами (модуль, чтобы не зависеть от инверсии знака)
        corr = np.corrcoef(manual[:, i], sklearn_proj[:, i])[0, 1]
        diff_percent = (1 - abs(corr)) * 100
        diffs.append(diff_percent)
    return np.mean(diffs), np.max(diffs), diffs

mean_diff_2, max_diff_2, diffs_2 = compare_projections(proj2_manual, proj2_sklearn, 2)
mean_diff_3, max_diff_3, diffs_3 = compare_projections(proj3_manual, proj3_sklearn, 3)

print("\nПроверка согласованности PCA (ручной vs sklearn)")
for i, d in enumerate(diffs_2, start=1):
    print(f"2D: Компонента PC{i}: расхождение {d:.4f}%")
print(f"Среднее расхождение (2 ПК): {mean_diff_2:.4f}%  |  Максимальное: {max_diff_2:.4f}%")

for i, d in enumerate(diffs_3, start=1):
    print(f"3D: Компонента PC{i}: расхождение {d:.4f}%")
print(f"Среднее расхождение (3 ПК): {mean_diff_3:.4f}%  |  Максимальное: {max_diff_3:.4f}%")

# Проверим разницу в объяснённой дисперсии
explained_ratio_sklearn = np.concatenate([pca3.explained_variance_ratio_,
                                          np.zeros(len(eigvals_sorted)-3)])
diff_explained = np.abs(explained_ratio - explained_ratio_sklearn[:len(explained_ratio)]) * 100

print("\nСравнение объяснённой дисперсии")
```

```python
for i, (man, skl, diff) in enumerate(zip(explained_ratio, explained_ratio_sklearn, diff_explained)):
    if i < 3:
        print(f"PC{i+1}: manual={man:.6f}, sklearn={skl:.6f}, расхождение={diff:.4f}%")

mean_diff_var = np.mean(diff_explained[:3])
print(f"Среднее различие в объяснённой дисперсии (топ-3 ПК): {mean_diff_var:.4f}%")

# Краткий вывод
if mean_diff_2 < 0.5 and mean_diff_var < 0.1:
    print("\n Методы PCA полностью синхронизированы")
else:
    print("\n Обнаружены небольшие различия")
```

```
Проверка согласованности PCA (ручной vs sklearn)
2D: Компонента PC1: расхождение 0.0000%
2D: Компонента PC2: расхождение 0.0000%
Среднее расхождение (2 ПК): 0.0000%  |  Максимальное: 0.0000%
3D: Компонента PC1: расхождение 0.0000%
3D: Компонента PC2: расхождение 0.0000%
3D: Компонента PC3: расхождение 0.0000%
Среднее расхождение (3 ПК): 0.0000%  |  Максимальное: 0.0000%

Сравнение объяснённой дисперсии
PC1: manual=0.718743, sklearn=0.718743, расхождение=0.0000%
PC2: manual=0.171082, sklearn=0.171082, расхождение=0.0000%
PC3: manual=0.096858, sklearn=0.096858, расхождение=0.0000%
Среднее различие в объяснённой дисперсии (топ-3 ПК): 0.0000%

 Методы PCA полностью синхронизированы
```

Вывод: научился применять метод PCA для визуализации данных.