`

Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Отчет по лабораторной работе 1

Специальность ИИ-23

Выполнил:

Гавришук В.Р.

Студент группы ИИ-23

Проверил:

Андренко К. В.
Преподаватель-стажёр
Кафедры ИИТ,
«___» _____2025 г.

Брест 2025

`

Цель: научиться применять метод PCA для осуществления визуализации данных

Общее задание

1. Используя выборку по варианту, осуществить проецирование данных на плоскость первых двух и трех главных компонент (двумя способами: 1. вручную через использование numpy.linalg.eig для вычисления собственных значений и собственных векторов и 2. с помощью sklearn.decomposition.PCA для непосредственного применения метода PCA – два независимых варианта решения);

2. Выполнить визуализацию полученных главных компонент с использованием средств библиотеки matplotlib, обозначая экземпляры разных классов с использованием разных цветовых маркеров;

3. Используя собственные значения, рассчитанные на этапе 1, вычислить потери, связанные с преобразованием по методу PCA. Сделать выводы;

4. Оформить отчет по выполненной работе, загрузить исходный код и отчет в соответствующий репозиторий на github.

Задание по вариантам

№ варианта	Выборка	Класс
4	heart+failure+clinical+records.zip	death_event

Ход работы:

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

path = "heart_failure_clinical_records_dataset.csv"
df = pd.read_csv(path)

print("Первые строки датасета:")
print(df.head(), "\n")

target_col = None
for c in df.columns:
    if 'death' in c.lower():
        target_col = c
        break
if target_col is None:
```

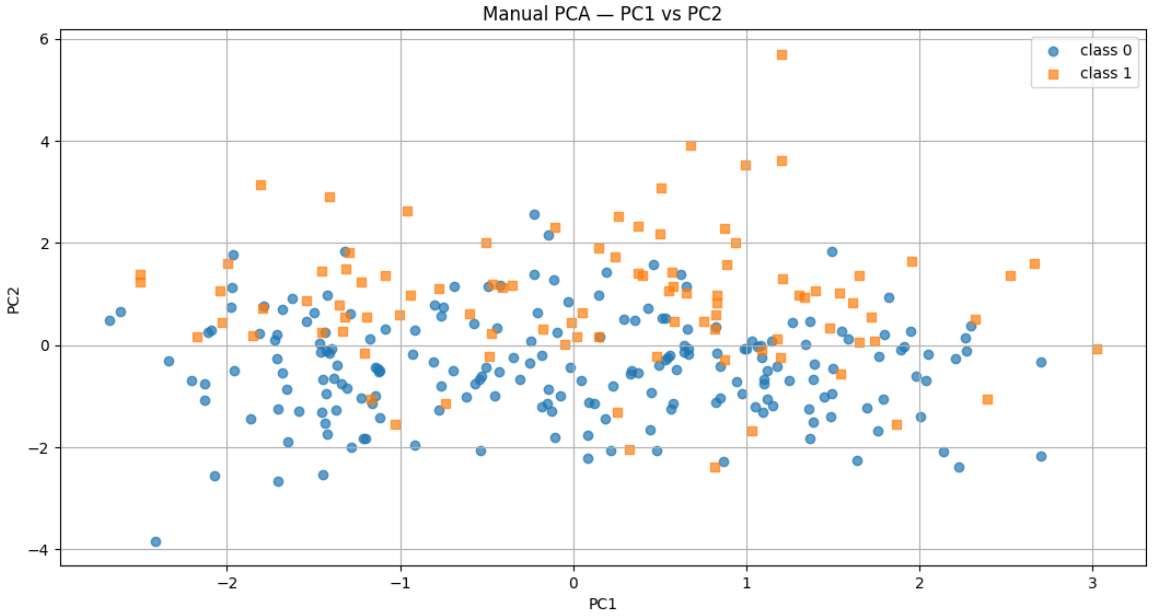
```
    raise ValueError("Не найден столбец, содержащий 'death' в имени.")

print(f"Целевая переменная: {target_col}\n")

y = df[target_col].astype(int).values
X = df.drop(columns=[target_col])

X_num = X.select_dtypes(include=[np.number]).copy()

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

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

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

PC_manual_2 = X_std.dot(eigvecs_sorted[:, :2])
PC_manual_3 = X_std.dot(eigvecs_sorted[:, :3])


pca2 = SKPCA(n_components=3)
PC_sklearn_3 = pca2.fit_transform(X_std)
PC_sklearn_2 = PC_sklearn_3[:, :2]

sk_explained_ratio = pca2.explained_variance_ratio_


total_variance = eigvals_sorted.sum()
loss_2 = eigvals_sorted[2:].sum()
loss_3 = eigvals_sorted[3:].sum()

loss_ratio_2 = loss_2 / total_variance
loss_ratio_3 = loss_3 / total_variance

def reconstruction_mse(X_std, eigvecs_sorted, k):
    top_k = eigvecs_sorted[:, :k]
    projected = X_std.dot(top_k)
    reconstructed = projected.dot(top_k.T)
    return np.mean((X_std - reconstructed) ** 2)

mse_k2 = reconstruction_mse(X_std, eigvecs_sorted, 2)
mse_k3 = reconstruction_mse(X_std, eigvecs_sorted, 3)


print("Число объектов: {}, число признаков: {}".format(*X_std.shape))
print("Собственные значения (по убыванию):", np.round(eigvals_sorted, 4))
print("Суммарная дисперсия:", round(total_variance, 4))
print(f"Потеря при k=2: {loss_ratio_2:.2%}")
print(f"Потеря при k=3: {loss_ratio_3:.2%}")
print(f"MSE реконструкции: k=2 -> {mse_k2:.4f}, k=3 -> {mse_k3:.4f}")
print("\nExplained variance ratio (sklearn):")
```

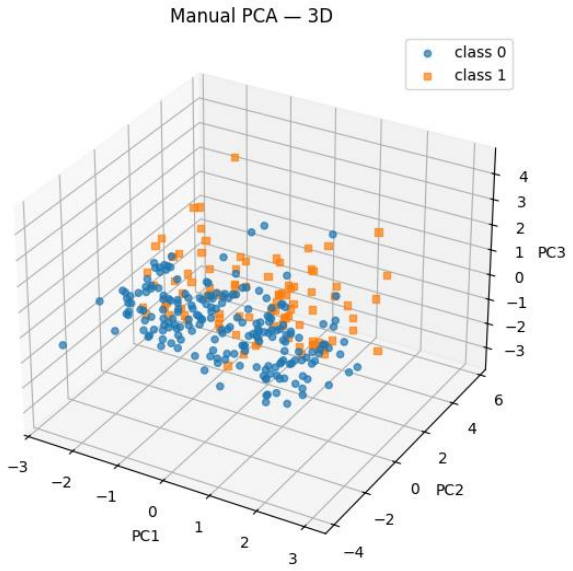
`

```
for i, r in enumerate(sk_explained_ratio, 1):
    print(f"  PC{i}: {r:.2%}")

def plot_2d(pc_scores, title):
    plt.figure(figsize=(7, 6))
    for cls, marker in zip(np.unique(y), ['o', 's']):
        mask = (y == cls)
        plt.scatter(pc_scores[mask, 0], pc_scores[mask, 1],
                    marker=marker, label=f"class {cls}", alpha=0.7)
    plt.xlabel("PC1")
    plt.ylabel("PC2")
    plt.title(title)
    plt.legend()
    plt.grid(True)
    plt.show()

def plot_3d(pc_scores, title):
    fig = plt.figure(figsize=(8, 6))
    ax = fig.add_subplot(111, projection='3d')
    for cls, marker in zip(np.unique(y), ['o', 's']):
        mask = (y == cls)
        ax.scatter(pc_scores[mask, 0], pc_scores[mask, 1], pc_scores[mask, 2],
                   marker=marker, label=f"class {cls}", alpha=0.7)
    ax.set_xlabel("PC1")
    ax.set_ylabel("PC2")
    ax.set_zlabel("PC3")
    ax.set_title(title)
    ax.legend()
    plt.show()

plot_2d(PC_manual_2, "Manual PCA — PC1 vs PC2")
plot_3d(PC_manual_3, "Manual PCA — 3D")
plot_2d(PC_sklearn_2, "sklearn PCA — PC1 vs PC2")
plot_3d(PC_sklearn_3, "sklearn PCA — 3D")
```

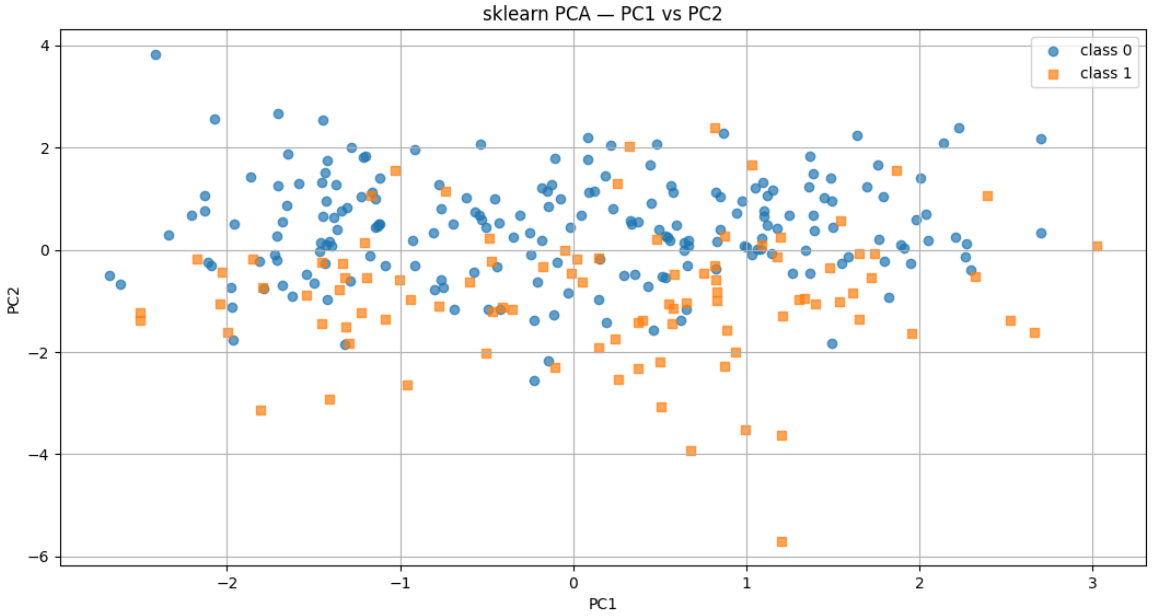
Результат работы программы:

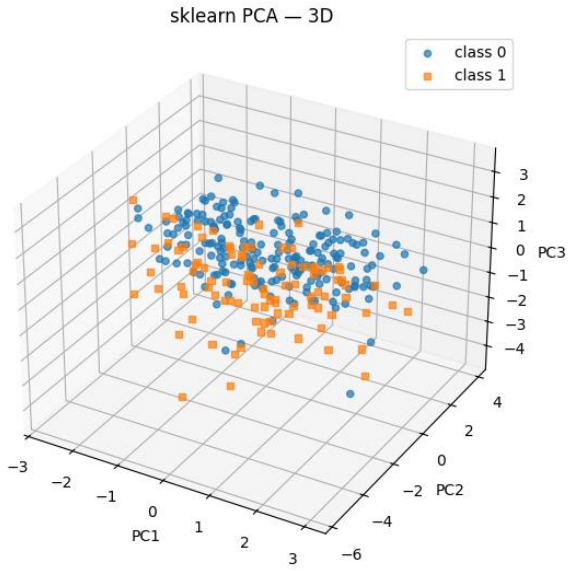
`


Manual PCA — PC1 vs PC2
class 0
class 1
PC1
PC2


Manual PCA — 3D
class 0
class 1
PC1
PC2
PC3

`





Вывод: научился применять метод PCA для осуществления визуализации данных.