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-24
Капуза Н.А.
Проверила:
Андренко К. В.

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.

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

Ход работы:

Код программы:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
# Для 3D-графиков
from mpl_toolkits.mplot3d import Axes3D

# Загрузка и подготовка данных ---
try:
    # Попытка загрузить локальный файл
    df = pd.read_csv('heart_failure_clinical_records_dataset.csv')
except FileNotFoundError:
    # Если файл не найден, загружаем из сети
    print("Локальный файл не найден. Загрузка данных из сети...")
    url = "https://archive.ics.uci.edu/ml/machine-learning-
databases/00519/heart_failure_clinical_records_dataset.csv"
    df = pd.read_csv(url)


print("Первые 5 строк данных:")
print(df.head())
```

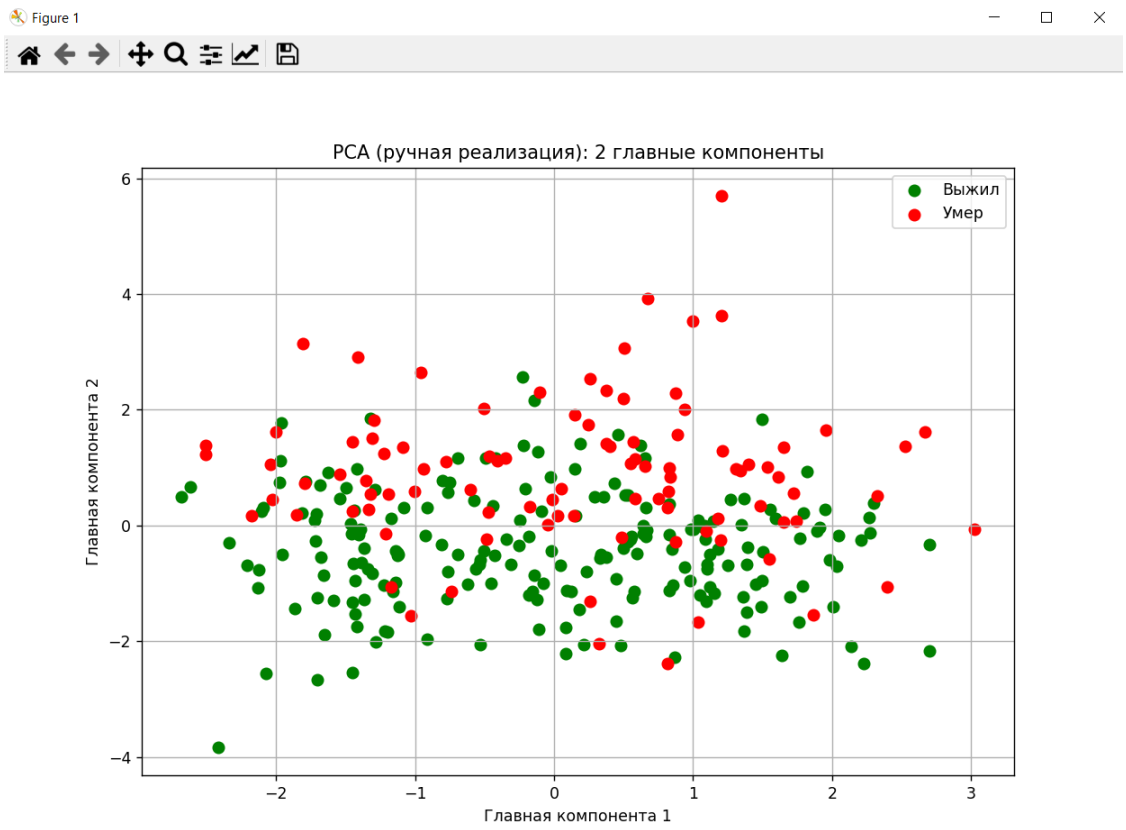
```
print("\nИнформация о данных:")
df.info()

# Проверка на наличие пропущенных значений
print("\nКоличество пропущенных значений в каждом столбце:")
print(df.isnull().sum())
# В данном наборе данных пропущенных значений нет.

# 1. Отделяем признаки (X) от целевой переменной (y)
# Целевая переменная 'DEATH_EVENT' используется только для визуализации
X = df.drop('DEATH_EVENT', axis=1)
y = df['DEATH_EVENT']

# 2. Стандартизация данных
# PCA чувствителен к масштабу признаков, поэтому стандартизация — важный шаг.
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print(f"\nРазмерность исходных данных: {X_scaled.shape}")


# --- Задание 1 (Способ 1): Реализация PCA вручную с помощью NumPy ---

print("\n--- 1. PCA вручную (NumPy) ---")

# 1. Вычисляем ковариационную матрицу
cov_matrix = np.cov(X_scaled.T)

# 2. Находим собственные значения (eigenvalues) и собственные векторы (eigenvectors)
eigen_values, eigen_vectors = np.linalg.eig(cov_matrix)

# 3. Сортируем собственные векторы по убыванию собственных значений
# Создаем пары (собственное значение, собственный вектор)
eigen_pairs = [(np.abs(eigen_values[i]), eigen_vectors[:, i]) for i in range(len(eigen_values))]
eigen_pairs.sort(key=lambda k: k[0], reverse=True)

# 4. Создаем проекционные матрицы для 2 и 3 главных компонент
W_2D = np.hstack((eigen_pairs[0][1][:, np.newaxis],
                  eigen_pairs[1][1][:, np.newaxis]))

W_3D = np.hstack((eigen_pairs[0][1][:, np.newaxis],
                  eigen_pairs[1][1][:, np.newaxis],
                  eigen_pairs[2][1][:, np.newaxis]))

print(f"Размерность проекционной матрицы для 2D: {W_2D.shape}")
print(f"Размерность проекционной матрицы для 3D: {W_3D.shape}")


# 5. Проецируем данные на новое пространство
X_pca_manual_2d = X_scaled.dot(W_2D)
X_pca_manual_3d = X_scaled.dot(W_3D)
```

```
print(f"Размерность данных после проекции на 2 компоненты: {X_pca_manual_2d.shape}")
print(f"Размерность данных после проекции на 3 компоненты: {X_pca_manual_3d.shape}")


# --- Задание 3: Расчет потерь информации ---

print("\n--- 3. Расчет потерь информации ---")

# Общая дисперсия — это сумма всех собственных значений
total_variance = sum(eigen_values)

# Дисперсия, объясненная первыми двумя компонентами
explained_variance_2d = sum(eigen_pairs[i][0] for i in range(2))
explained_variance_ratio_2d = explained_variance_2d / total_variance

# Дисперсия, объясненная первыми тремя компонентами
explained_variance_3d = sum(eigen_pairs[i][0] for i in range(3))
explained_variance_ratio_3d = explained_variance_3d / total_variance

loss_2d = 1 - explained_variance_ratio_2d
loss_3d = 1 - explained_variance_ratio_3d

print(f"Доля объясненной дисперсии (2 компоненты): {explained_variance_ratio_2d:.4f}")
print(f"Потери информации при переходе к 2 компонентам: {loss_2d:.4f} ({loss_2d:.2%})")
print(f"Доля объясненной дисперсии (3 компоненты): {explained_variance_ratio_3d:.4f}")
print(f"Потери информации при переходе к 3 компонентам: {loss_3d:.4f} ({loss_3d:.2%})")

print("\nВыводы:")
print("При проецировании данных на плоскость первых двух главных компонент сохраняется примерно "
    f"{explained_variance_ratio_2d:.2%} дисперсии исходных данных, при этом теряется "
    f"{loss_2d:.2%} информации.")
print("При использовании трех главных компонент сохраняется уже "
    f"{explained_variance_ratio_3d:.2%} дисперсии, а потери сокращаются до "
    f"{loss_3d:.2%}. Это обеспечивает лучшее представление структуры данных.")


# --- Задание 1 (Способ 2): Реализация PCA с помощью scikit-learn ---

print("\n--- 1. PCA с помощью scikit-learn ---")

# PCA на 2 компоненты
pca_sk_2d = PCA(n_components=2)
X_pca_sklearn_2d = pca_sk_2d.fit_transform(X_scaled)

# PCA на 3 компоненты
pca_sk_3d = PCA(n_components=3)
X_pca_sklearn_3d = pca_sk_3d.fit_transform(X_scaled)
```

```
print(f"Размерность данных после проекции на 2 компоненты (sklearn):
{X_pca_sklearn_2d.shape}")
print(f"Размерность данных после проекции на 3 компоненты (sklearn):
{X_pca_sklearn_3d.shape}")

# Сравнение с ручной реализацией (результаты могут отличаться знаком, это нормально)
# Знак векторов может быть инвертирован, но они по-прежнему ортогональны и несут ту же
информацию
print("\nСравнение результатов PCA:")
print(f"Объясненная дисперсия (sklearn, 2D): {sum(pca_sk_2d.explained_variance_ratio_):.4f}")
print(f"Объясненная дисперсия (ручной, 2D):   {explained_variance_ratio_2d:.4f}")


# --- Задание 2: Визуализация главных компонент ---

print("\n--- 2. Визуализация результатов ---")

def plot_2d(X_pca, y, title):
    """Функция для 2D визуализации результатов PCA."""
    plt.figure(figsize=(10, 7))
    targets = [0, 1]
    colors = ['g', 'r']
    labels = ['Выжил', 'Умер']

    for target, color, label in zip(targets, colors, labels):
        indices_to_keep = (y == target)
        plt.scatter(X_pca[indices_to_keep, 0],
                    X_pca[indices_to_keep, 1],
                    c=color,
                    s=50,
                    label=label)

    plt.title(title)
    plt.xlabel('Главная компонента 1')
    plt.ylabel('Главная компонента 2')
    plt.legend()
    plt.grid()
    plt.show()

def plot_3d(X_pca, y, title):
    """Функция для 3D визуализации результатов PCA."""
    fig = plt.figure(figsize=(10, 8))
    ax = fig.add_subplot(111, projection='3d')
    targets = [0, 1]
    colors = ['g', 'r']
    labels = ['Выжил', 'Умер']

    for target, color, label in zip(targets, colors, labels):
        indices_to_keep = (y == target)
        ax.scatter(X_pca[indices_to_keep, 0],
                   X_pca[indices_to_keep, 1],
```

```
                X_pca[indices_to_keep, 2],
                c=color,
                s=50,
                label=label)

    ax.set_title(title)
    ax.set_xlabel('Главная компонента 1')
    ax.set_ylabel('Главная компонента 2')
    ax.set_zlabel('Главная компонента 3')
    ax.legend()
    ax.grid(True)
    plt.show()

# Визуализация для ручной реализации
plot_2d(X_pca_manual_2d, y, 'PCA (ручная реализация): 2 главные компоненты')
plot_3d(X_pca_manual_3d, y, 'PCA (ручная реализация): 3 главные компоненты')

# Визуализация для scikit-learn
plot_2d(X_pca_sklearn_2d, y, 'PCA (scikit-learn): 2 главные компоненты')
plot_3d(X_pca_sklearn_3d, y, 'PCA (scikit-learn): 3 главные компоненты')

print("\nВизуализации сгенерированы. Работа завершена.")
```

Графики:



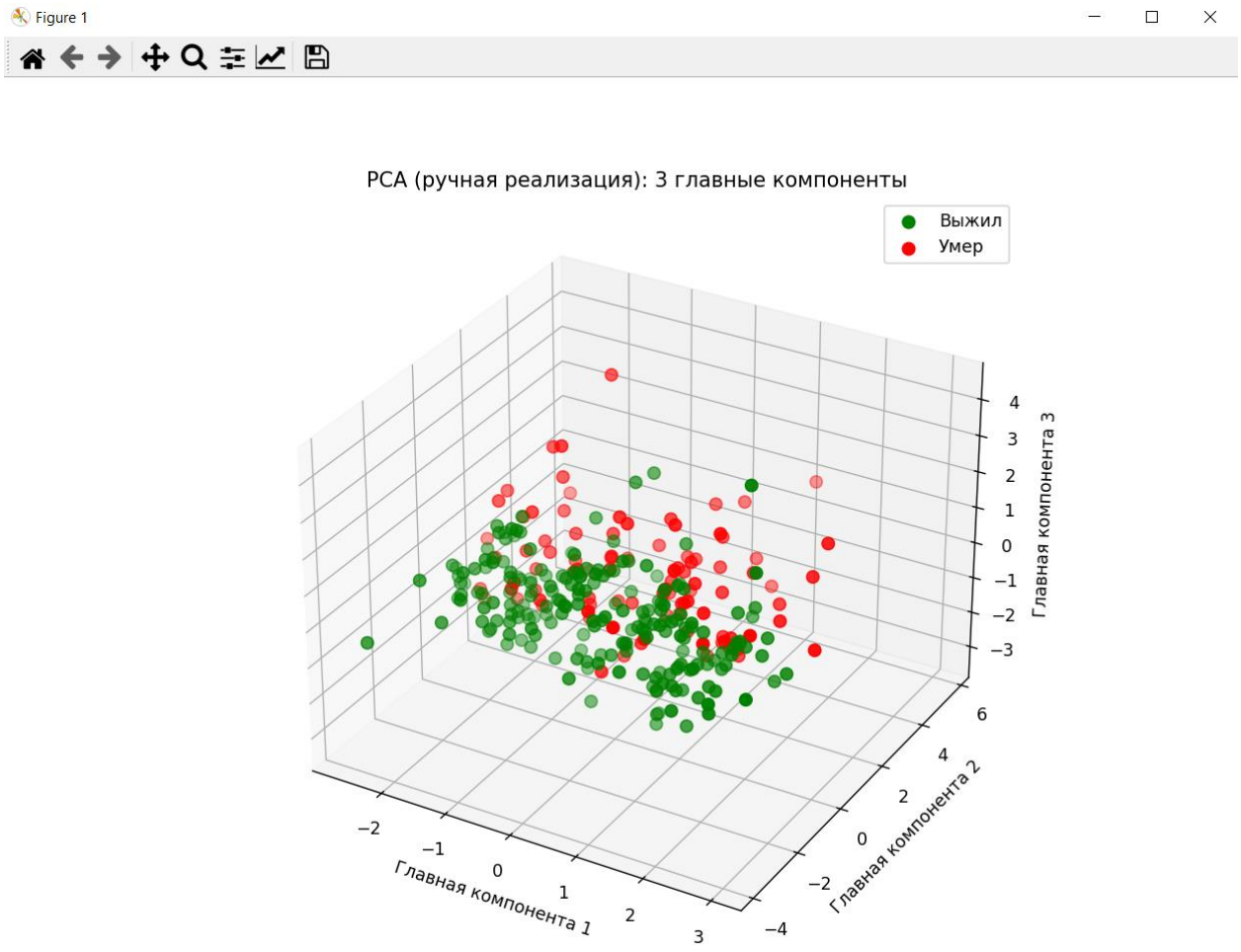


Figure 1
PCA (ручная реализация): 3 главные компоненты
Выжил
Умер
Главная компонента 1
Главная компонента 2
Главная компонента 3

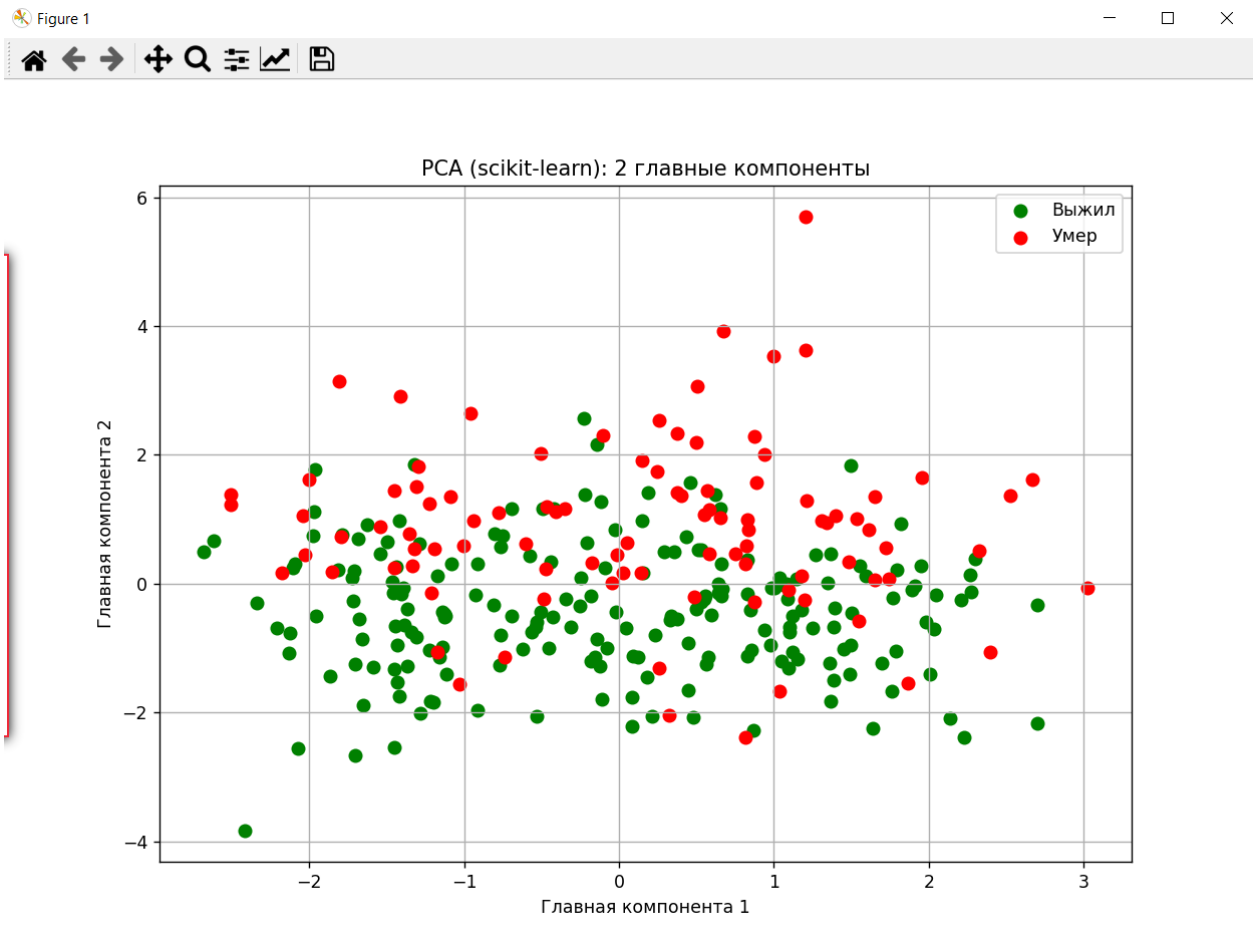
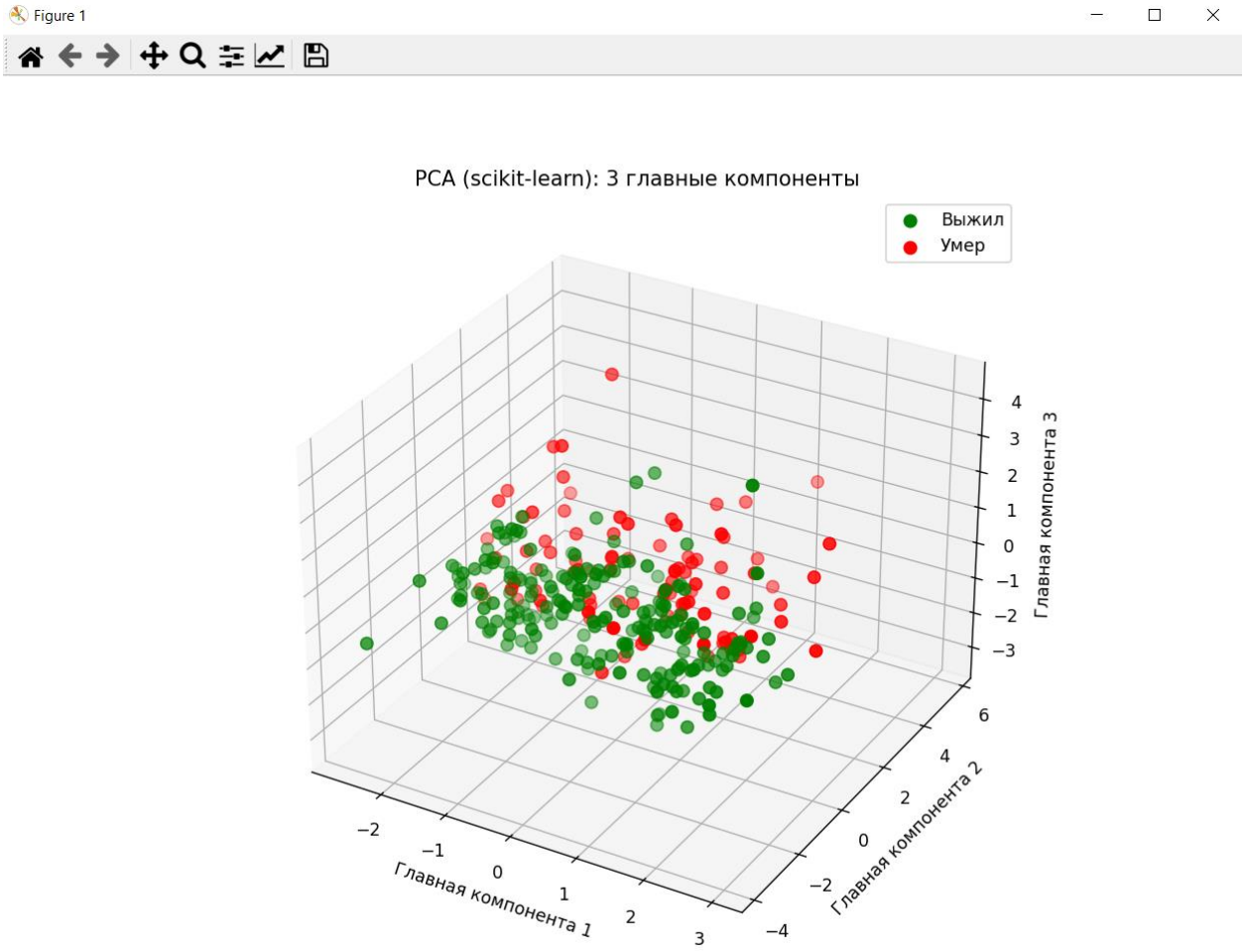


Figure 1
PCA (scikit-learn): 2 главные компоненты
Выжил
Умер
Главная компонента 1
Главная компонента 2



Вывод: Я научился применять метод PCA для осуществления визуализации данных.