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

Крупич Д.Д.

**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** |
| --- | --- | --- |
| 7 | hcv+data.zip | Category |

**Ход работы:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.decomposition import PCA

import warnings

warnings.simplefilter(*action*='ignore', *category*=FutureWarning)

try:

    data = pd.read\_csv('hcvdat0.csv')

    print("Файл hcvdat0.csv успешно загружен.")

except FileNotFoundError:

    print("Ошибка: Файл hcvdat0.csv не найден.")

    exit()

if 'Unnamed: 0' in data.columns:

    data = data.drop('Unnamed: 0', *axis*=1)

if 'Sex' in data.columns:

    data = pd.get\_dummies(data, *columns*=['Sex'], *drop\_first*=True, *dtype*=float)

for col in data.select\_dtypes(*include*=np.number).columns:

    if data[col].isnull().sum() > 0:

        median\_val = data[col].median()

        data[col] = data[col].fillna(median\_val)

X = data.drop('Category', *axis*=1)

y\_str = data['Category']

label\_encoder = LabelEncoder()

y = label\_encoder.fit\_transform(y\_str)

class\_names = label\_encoder.classes\_

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

cov\_matrix = np.cov(X\_scaled.T)

eigen\_values, eigen\_vectors = np.linalg.eig(cov\_matrix)

sorted\_indices = np.argsort(eigen\_values)[::-1]

sorted\_eigen\_values = eigen\_values[sorted\_indices]

sorted\_eigen\_vectors = eigen\_vectors[:, sorted\_indices]

projection\_matrix\_2d = sorted\_eigen\_vectors[:, :2]

projection\_matrix\_3d = sorted\_eigen\_vectors[:, :3]

X\_pca\_manual\_2d = X\_scaled.dot(projection\_matrix\_2d)

X\_pca\_manual\_3d = X\_scaled.dot(projection\_matrix\_3d)

*# PCA sklearn*

pca\_sklearn\_2d = PCA(*n\_components*=2)

X\_pca\_sklearn\_2d = pca\_sklearn\_2d.fit\_transform(X\_scaled)

pca\_sklearn\_3d = PCA(*n\_components*=3)

X\_pca\_sklearn\_3d = pca\_sklearn\_3d.fit\_transform(X\_scaled)

cmap = plt.get\_cmap('Set1')

plt.figure(*figsize*=(10, 8))

scatter\_sklearn\_2d = plt.scatter(

    X\_pca\_sklearn\_2d[:, 0], X\_pca\_sklearn\_2d[:, 1],

*c*=y, *cmap*=cmap, *alpha*=0.9, *edgecolor*='k', *s*=60

)

plt.title('Скользящая проекция на 2 главные компоненты (sklearn PCA)', *fontsize*=16)

plt.xlabel('Первая главная компонента', *fontsize*=12)

plt.ylabel('Вторая главная компонента', *fontsize*=12)

plt.legend(*handles*=scatter\_sklearn\_2d.legend\_elements()[0], *labels*=list(class\_names), *title*="Категории")

plt.grid(True)

plt.show()

plt.figure(*figsize*=(10, 8))

scatter\_manual\_2d = plt.scatter(

    X\_pca\_manual\_2d[:, 0], X\_pca\_manual\_2d[:, 1],

*c*=y, *cmap*=cmap, *alpha*=0.9, *edgecolor*='k', *s*=60

)

plt.title('Ручное проецирование на 2 главные компоненты (manual PCA)', *fontsize*=16)

plt.xlabel('Первая главная компонента', *fontsize*=12)

plt.ylabel('Вторая главная компонента', *fontsize*=12)

plt.legend(*handles*=scatter\_manual\_2d.legend\_elements()[0], *labels*=list(class\_names), *title*="Категории")

plt.grid(True)

plt.show()

fig = plt.figure(*figsize*=(14, 10))

ax1 = fig.add\_subplot(121, *projection*='3d')

scatter\_sklearn\_3d = ax1.scatter(

    X\_pca\_sklearn\_3d[:, 0], X\_pca\_sklearn\_3d[:, 1], X\_pca\_sklearn\_3d[:, 2],

*c*=y, *cmap*=cmap, *alpha*=0.9, *edgecolor*='k', *s*=60

)

ax1.set\_title('Скользящая проекция на 3 главные компоненты (sklearn PCA)', *fontsize*=14)

ax1.set\_xlabel('Первая главная компонента', *fontsize*=10)

ax1.set\_ylabel('Вторая главная компонента', *fontsize*=10)

ax1.set\_zlabel('Третья главная компонента', *fontsize*=10)

ax1.legend(*handles*=scatter\_sklearn\_3d.legend\_elements()[0], *labels*=list(class\_names), *title*="Категории")

ax2 = fig.add\_subplot(122, *projection*='3d')

scatter\_manual\_3d = ax2.scatter(

    X\_pca\_manual\_3d[:, 0], X\_pca\_manual\_3d[:, 1], X\_pca\_manual\_3d[:, 2],

*c*=y, *cmap*=cmap, *alpha*=0.9, *edgecolor*='k', *s*=60

)

ax2.set\_title('Ручное проецирование на 3 главные компоненты (manual PCA)', *fontsize*=14)

ax2.set\_xlabel('Первая главная компонента', *fontsize*=10)

ax2.set\_ylabel('Вторая главная компонента', *fontsize*=10)

ax2.set\_zlabel('Третья главная компонента', *fontsize*=10)

ax2.legend(*handles*=scatter\_manual\_3d.legend\_elements()[0], *labels*=list(class\_names), *title*="Категории")

plt.tight\_layout()

plt.show()

print("Визуализации отображены")

print("\nАнализ информационных потерь (ручное вычисление)")

total\_variance\_manual = np.sum(sorted\_eigen\_values)

variance\_explained\_2d\_manual = np.sum(sorted\_eigen\_values[:2]) / total\_variance\_manual

variance\_explained\_3d\_manual = np.sum(sorted\_eigen\_values[:3]) / total\_variance\_manual

loss\_2d\_manual = 1 - variance\_explained\_2d\_manual

loss\_3d\_manual = 1 - variance\_explained\_3d\_manual

print(f"Ручное PCA - Сохраненная дисперсия при 2 компонентах: {variance\_explained\_2d\_manual:.2%}")

print(f"Ручное PCA - Потери информации при переходе к 2D: {loss\_2d\_manual:.2%}")

print("-" \* 40)

print(f"Ручное PCA - Сохраненная дисперсия при 3 компонентах: {variance\_explained\_3d\_manual:.2%}")

print(f"Ручное PCA - Потери информации при переходе к 3D: {loss\_3d\_manual:.2%}")

print("\nАнализ информационных потерь (sklearn PCA)")

total\_variance\_sklearn = np.sum(pca\_sklearn\_3d.explained\_variance\_)

variance\_explained\_2d\_sklearn = np.sum(pca\_sklearn\_2d.explained\_variance\_) / total\_variance\_sklearn

variance\_explained\_3d\_sklearn = np.sum(pca\_sklearn\_3d.explained\_variance\_) / total\_variance\_sklearn

loss\_2d\_sklearn = 1 - variance\_explained\_2d\_sklearn

loss\_3d\_sklearn = 1 - variance\_explained\_3d\_sklearn

print(f"sklearn PCA - Сохраненная дисперсия при 2 компонентах: {variance\_explained\_2d\_sklearn:.2%}")

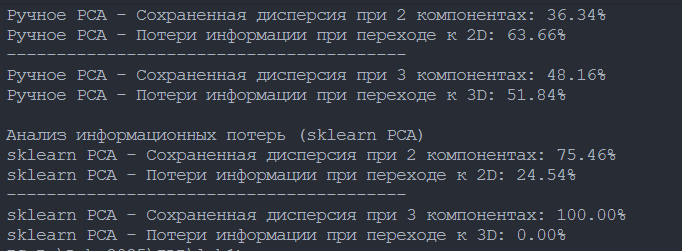
print(f"sklearn PCA - Потери информации при переходе к 2D: {loss\_2d\_sklearn:.2%}")

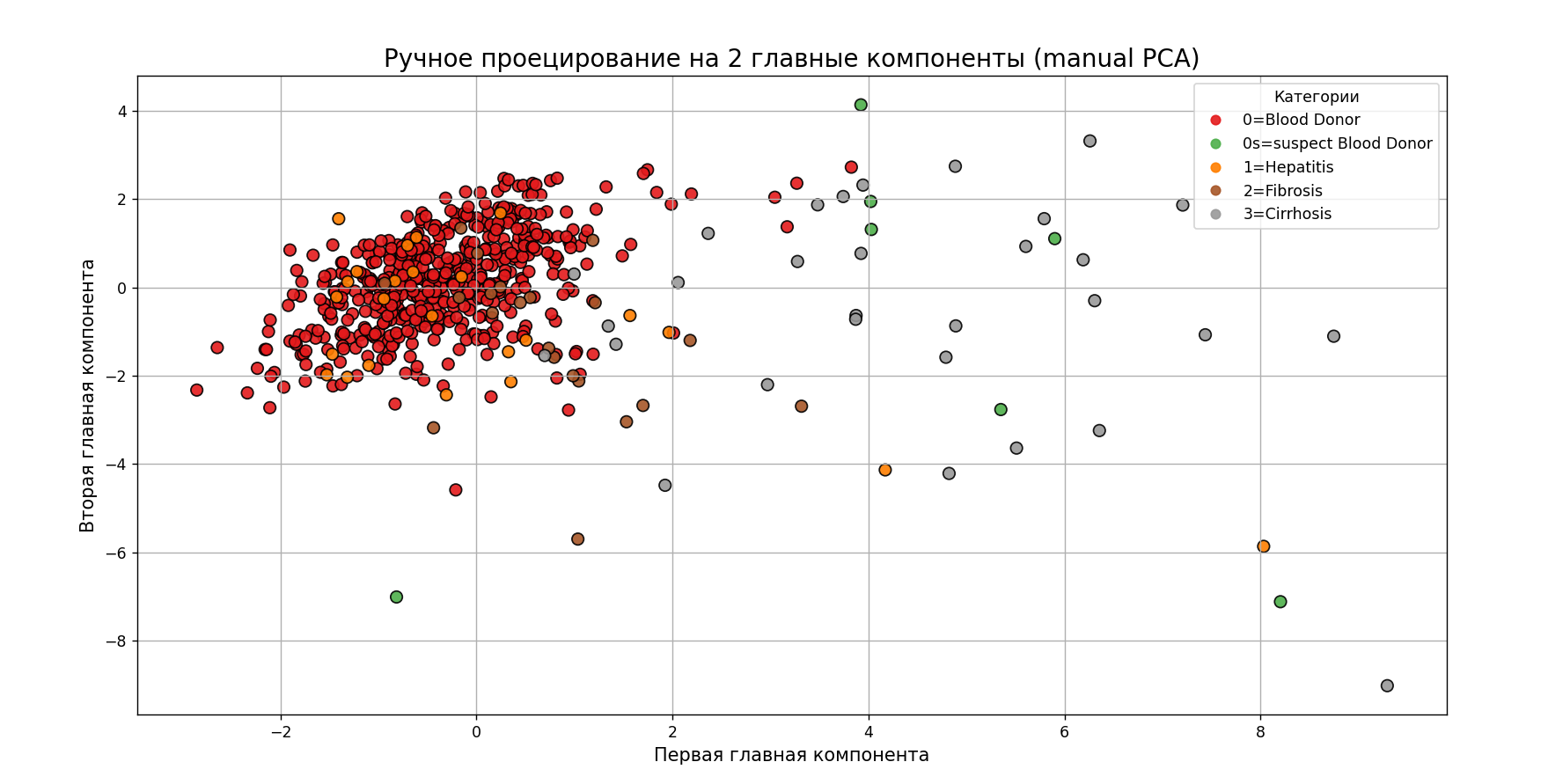
print("-" \* 40)

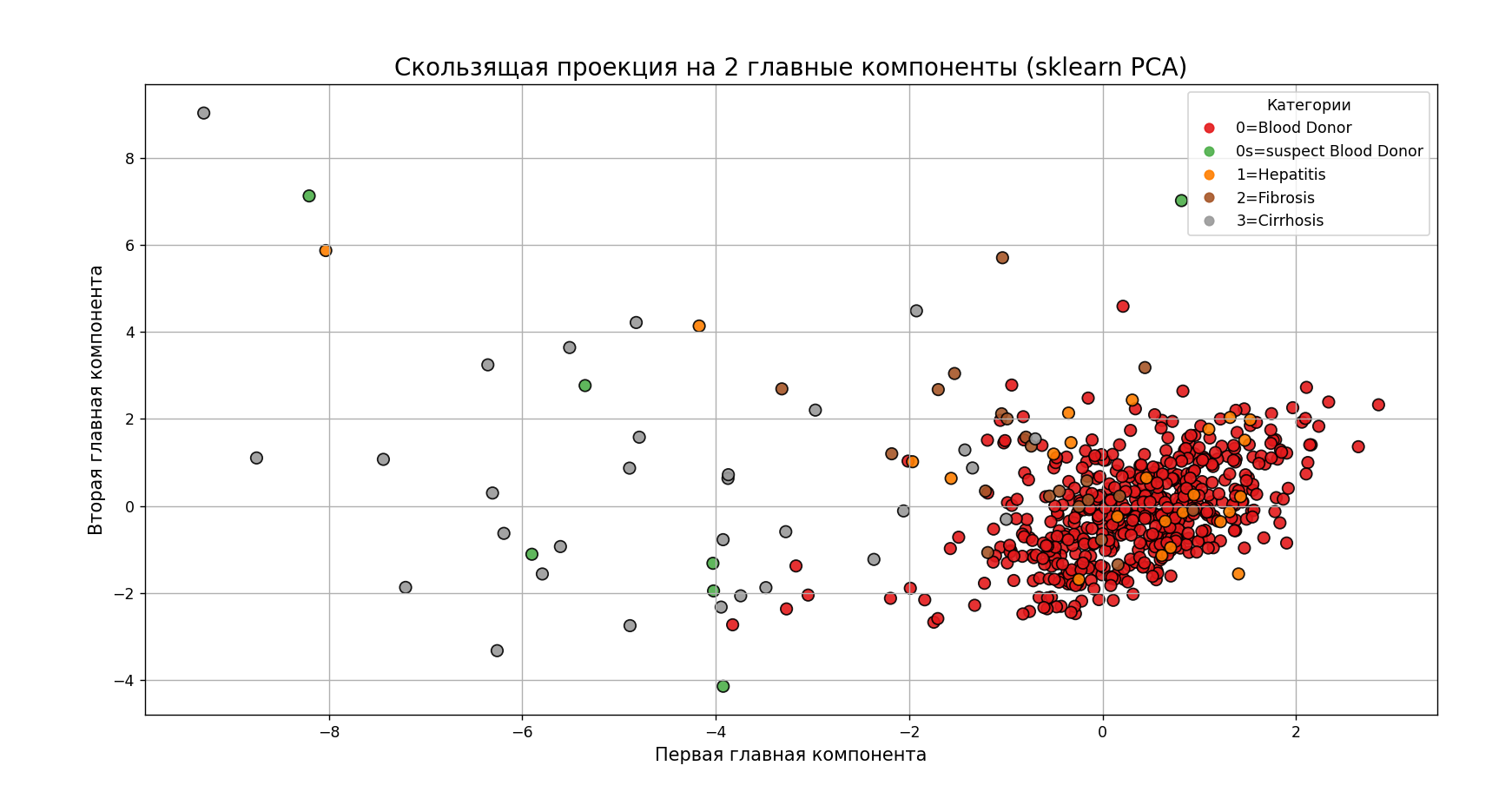
print(f"sklearn PCA - Сохраненная дисперсия при 3 компонентах: {variance\_explained\_3d\_sklearn:.2%}")

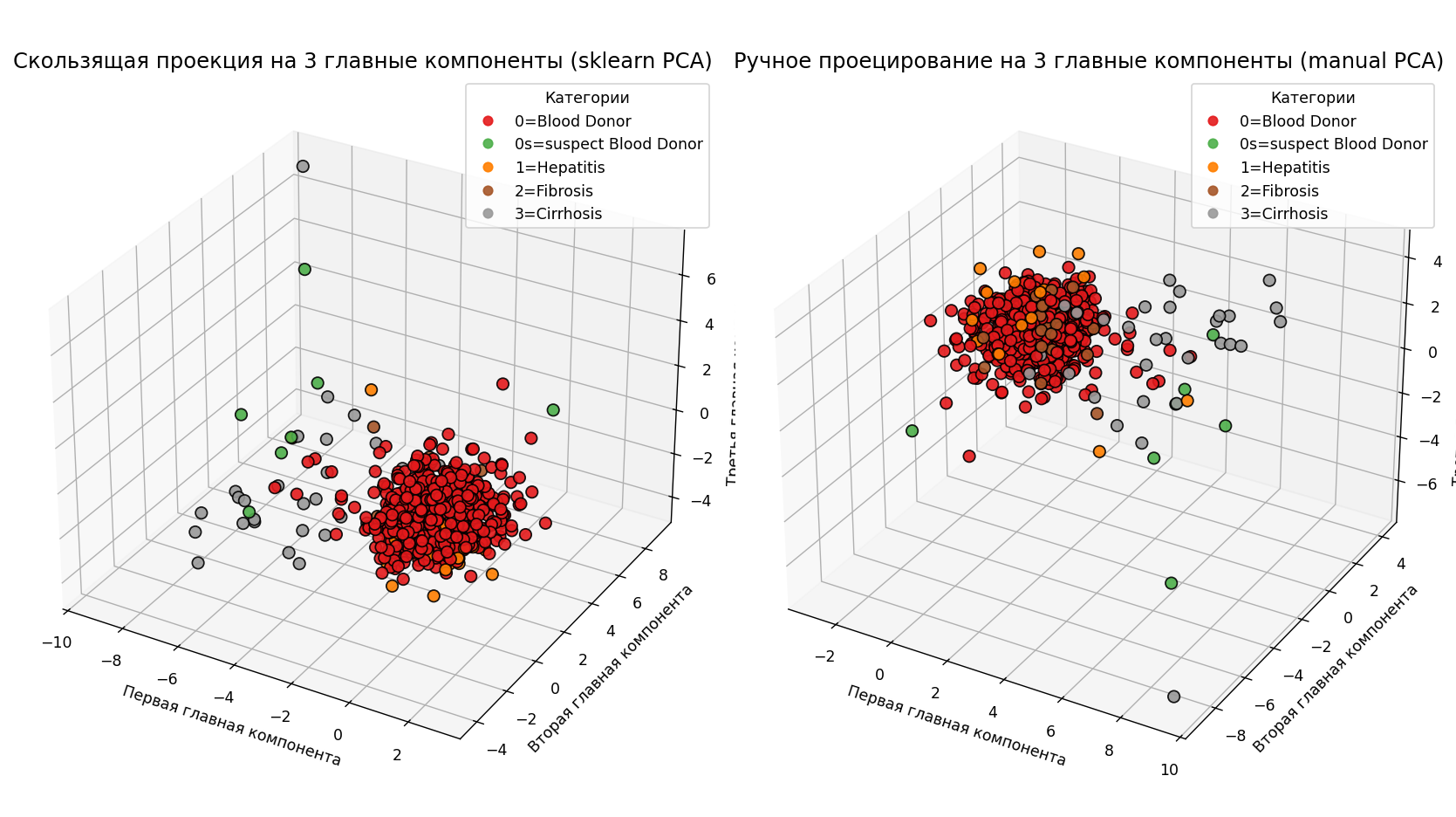
print(f"sklearn PCA - Потери информации при переходе к 3D: {loss\_3d\_sklearn:.2%}")

**Вывод кода:**



****





**Вывод:** Ручной подход полезен для понимания механизма PCA, но для качественного анализа и визуализации данных лучше применять sklearn PCA, так как он обеспечивает значительно лучшие показатели сохранённой дисперсии и меньшие информационные потери.