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
Мшар В.В.
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
12	hcv+data.zip	Category

Ход работы:

Код программы:

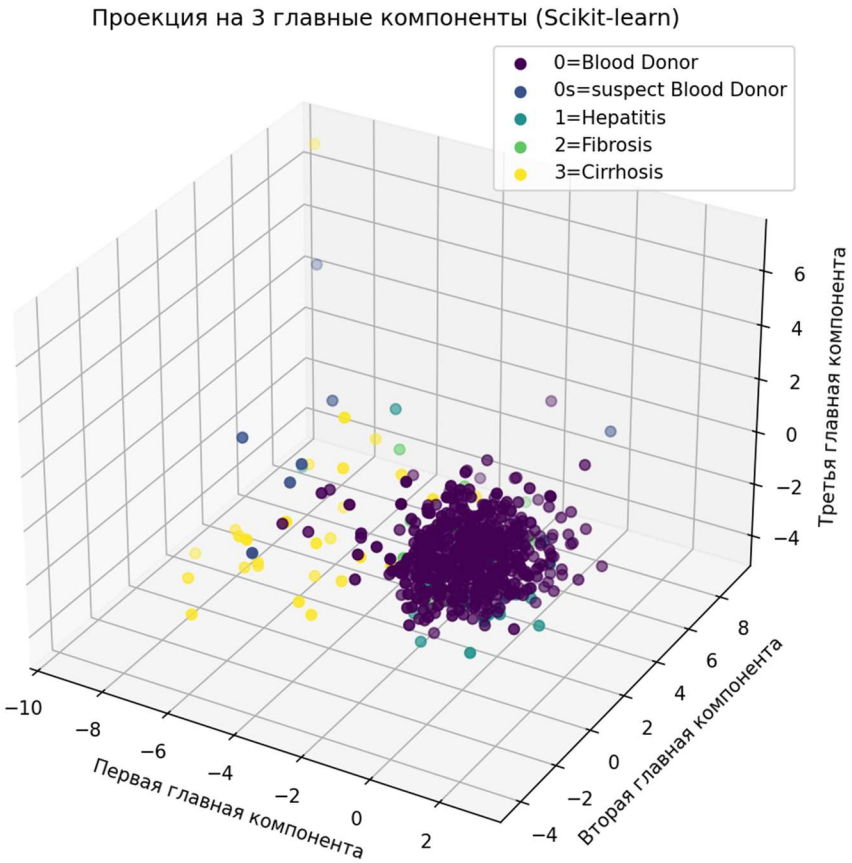
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from mpl_toolkits.mplot3d import Axes3D

# --- 1. Загрузка и предварительная обработка данных ---

# Загрузка данных из файла
try:
    df = pd.read_csv('hcvdat0.csv', index_col=0)
except FileNotFoundError:
    print("Файл hcvdat0.csv не найден. Пожалуйста, убедитесь, что он находится в той же директории.")
    exit()

# Обработка пропущенных значений (NA)
# Сначала заменим строковые 'NA' на numpy.nan
df.replace('NA', np.nan, inplace=True)
# Заполним пропущенные значения средним значением по каждому столбцу
for col in df.select_dtypes(include=np.number).columns:
```

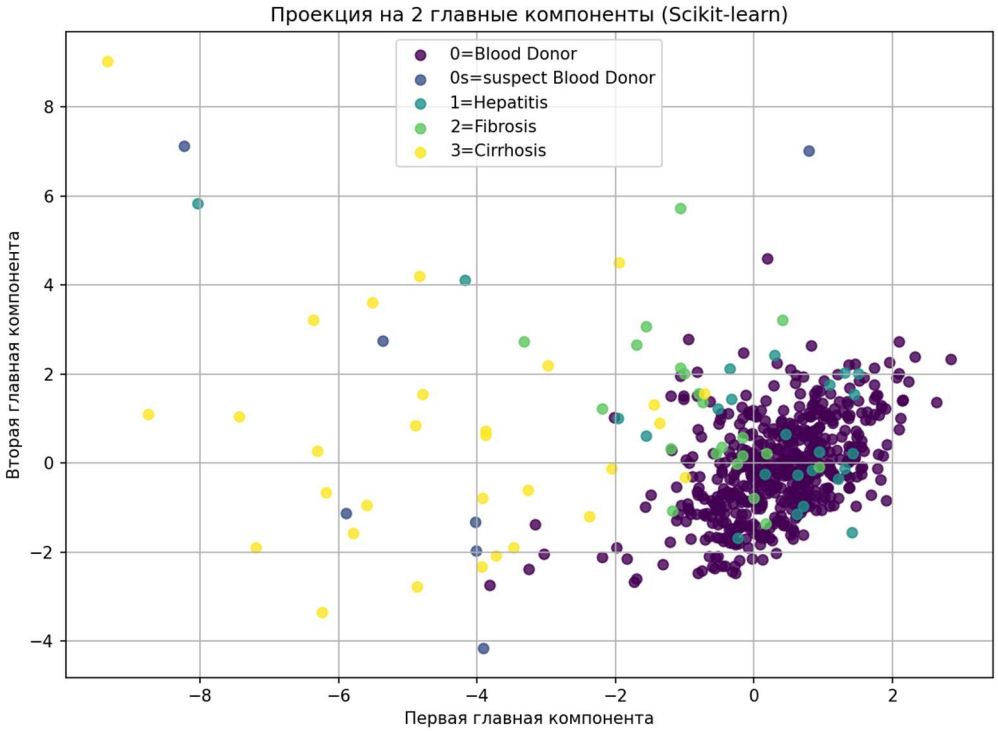
```
    df[col].fillna(df[col].mean(), inplace=True)

# Преобразование категориальной переменной 'Sex' в числовую с помощью One-Hot Encoding
df = pd.get_dummies(df, columns=['Sex'], drop_first=True)

# Разделение данных на признаки (X) и метки классов (y)
X = df.drop('Category', axis=1)
y = df['Category']

# Стандартизация (масштабирование) данных
# Это важный шаг для PCA, так как метод чувствителен к масштабу признаков
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# --- 2. PCA: Ручная реализация ---

print("--- Ручная реализация PCA ---")
# Шаг 1: Вычисление ковариационной матрицы
cov_matrix = np.cov(X_scaled, rowvar=False)

# Шаг 2: Вычисление собственных значений и собственных векторов
eigen_values, eigen_vectors = np.linalg.eig(cov_matrix)

# Шаг 3: Сортировка собственных векторов по убыванию собственных значений
sorted_indices = np.argsort(eigen_values)[::-1]
sorted_eigen_values = eigen_values[sorted_indices]
sorted_eigen_vectors = eigen_vectors[:, sorted_indices]

# Шаг 4: Проецирование на 2 главные компоненты
W_2d = sorted_eigen_vectors[:, :2]
X_pca_manual_2d = X_scaled.dot(W_2d)

# Шаг 5: Проецирование на 3 главные компоненты
W_3d = sorted_eigen_vectors[:, :3]
X_pca_manual_3d = X_scaled.dot(W_3d)

print("Проекция на 2 компоненты (ручная) создана. Размер:", X_pca_manual_2d.shape)
print("Проекция на 3 компоненты (ручная) создана. Размер:", X_pca_manual_3d.shape)


# --- 3. PCA: Реализация с помощью Scikit-learn ---

print("\n--- Реализация PCA с помощью Scikit-learn ---")
# Проецирование на 2 главные компоненты
pca_2 = PCA(n_components=2)
X_pca_sklearn_2d = pca_2.fit_transform(X_scaled)

# Проецирование на 3 главные компоненты
pca_3 = PCA(n_components=3)
X_pca_sklearn_3d = pca_3.fit_transform(X_scaled)
```

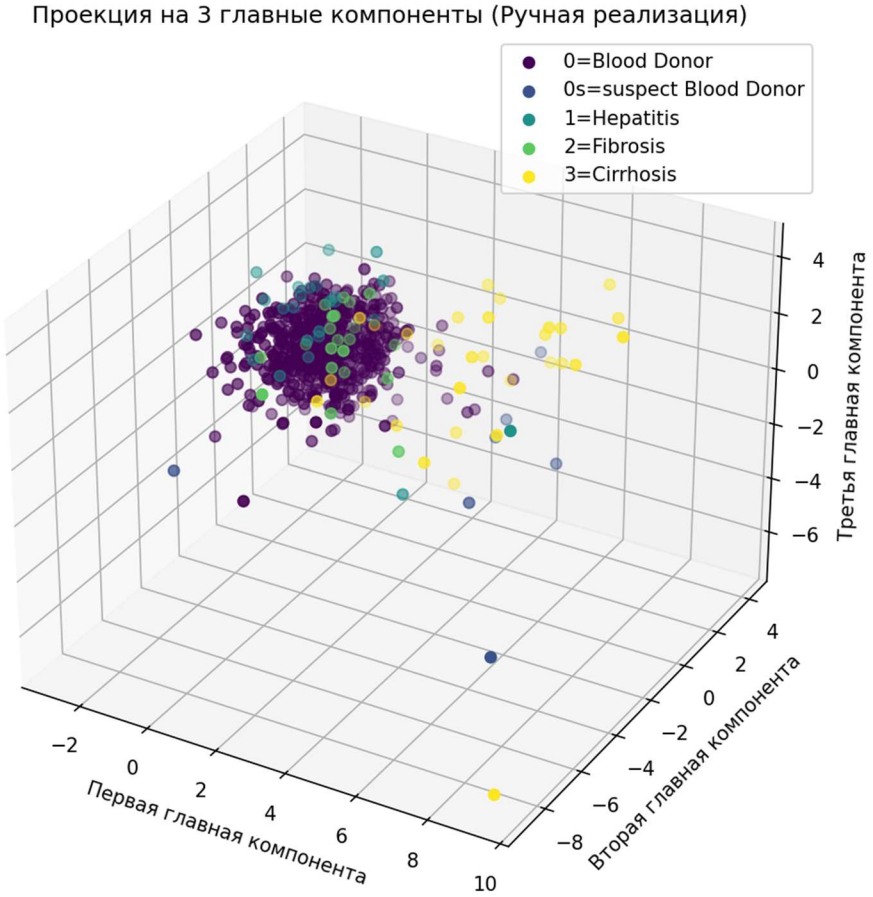
```
print("Проекция на 2 компоненты (sklearn) создана. Размер:", X_pca_sklearn_2d.shape)
print("Проекция на 3 компоненты (sklearn) создана. Размер:", X_pca_sklearn_3d.shape)

# --- 4. Визуализация результатов ---

def plot_pca(X_pca, y, title, is_3d=False):
    """Функция для визуализации результатов PCA."""
    unique_categories = y.unique()
    # Используем цветовую карту, которая хорошо различима
    colors = plt.cm.get_cmap('viridis', len(unique_categories))

    if is_3d:
        fig = plt.figure(figsize=(10, 8))
        ax = fig.add_subplot(111, projection='3d')
        for i, category in enumerate(unique_categories):
            indices = y == category
            ax.scatter(X_pca[indices, 0], X_pca[indices, 1], X_pca[indices, 2],
                       color=colors(i), label=category, s=30)
        ax.set_xlabel('Первая главная компонента')
        ax.set_ylabel('Вторая главная компонента')
        ax.set_zlabel('Третья главная компонента')
    else:
        plt.figure(figsize=(10, 7))
        for i, category in enumerate(unique_categories):
            indices = y == category
            plt.scatter(X_pca[indices, 0], X_pca[indices, 1],
                        color=colors(i), label=category, alpha=0.8)
        plt.xlabel('Первая главная компонента')
        plt.ylabel('Вторая главная компонента')

    plt.title(title)
    plt.legend()
    plt.grid(True)
    # plt.show() # Мы будем вызывать show() в конце, чтобы показать все графики сразу

# Визуализация для ручной реализации
print("\nОтображение графиков для ручной реализации PCA...")
plot_pca(X_pca_manual_2d, y, 'Проекция на 2 главные компоненты (Ручная реализация)')
plot_pca(X_pca_manual_3d, y, 'Проекция на 3 главные компоненты (Ручная реализация)',
is_3d=True)

# Визуализация для реализации Scikit-learn
print("Отображение графиков для реализации PCA с помощью Scikit-learn...")
plot_pca(X_pca_sklearn_2d, y, 'Проекция на 2 главные компоненты (Scikit-learn)')
plot_pca(X_pca_sklearn_3d, y, 'Проекция на 3 главные компоненты (Scikit-learn)', is_3d=True)

# --- 5. Расчет потерь и выводы ---

print("\n--- Расчет информационных потерь ---")
# Сумма всех собственных значений равна общей дисперсии
total_variance = np.sum(sorted_eigen_values)
```

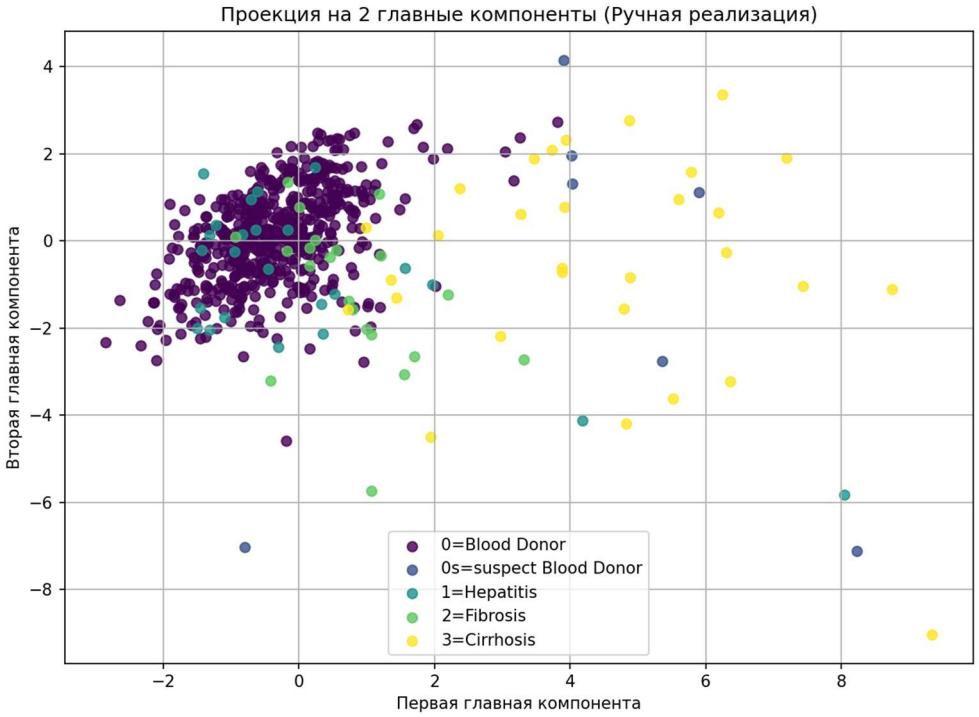
```
# Объясненная дисперсия и потери для 2 компонент
explained_variance_2d = np.sum(sorted_eigen_values[:2]) / total_variance
loss_2d = 1 - explained_variance_2d
print(f"Доля объясненной дисперсии (2 компоненты): {explained_variance_2d:.4f}")
print(f"Информационные потери (2 компоненты): {loss_2d:.4f}")

# Объясненная дисперсия и потери для 3 компонент
explained_variance_3d = np.sum(sorted_eigen_values[:3]) / total_variance
loss_3d = 1 - explained_variance_3d
print(f"Доля объясненной дисперсии (3 компоненты): {explained_variance_3d:.4f}")
print(f"Информационные потери (3 компоненты): {loss_3d:.4f}")

# Показываем все созданные графики
plt.show()
```

Графики:




Проекция на 2 главные компоненты (Scikit-learn)
0=Blood Donor
0s=suspect Blood Donor
1=Hepatitis
2=Fibrosis
3=Cirrhosis
Вторая главная компонента
Первая главная компонента
8
6
4
2
0
−2
−4
−8
−6
−4
−2
0
2


Проекция на 3 главные компоненты (Ручная реализация)
0=Blood Donor
0s=suspect Blood Donor
1=Hepatitis
2=Fibrosis
3=Cirrhosis
Первая главная компонента
Вторая главная компонента
Третья главная компонента
−2
0
2
4
6
8
10
4
2
0
−2
−4
−6
−8
4
2
0
−2
−4
−6



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