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:
```

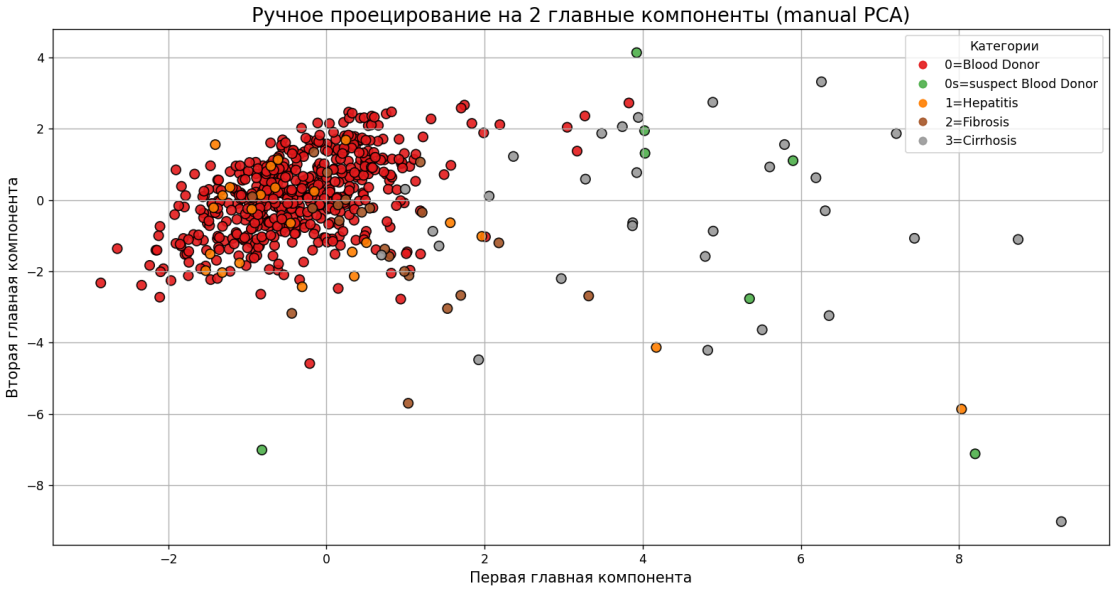
```
    data = pd.get_dummies(data, columns=['Sex'], drop_first=True, dtype=float)

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)
```

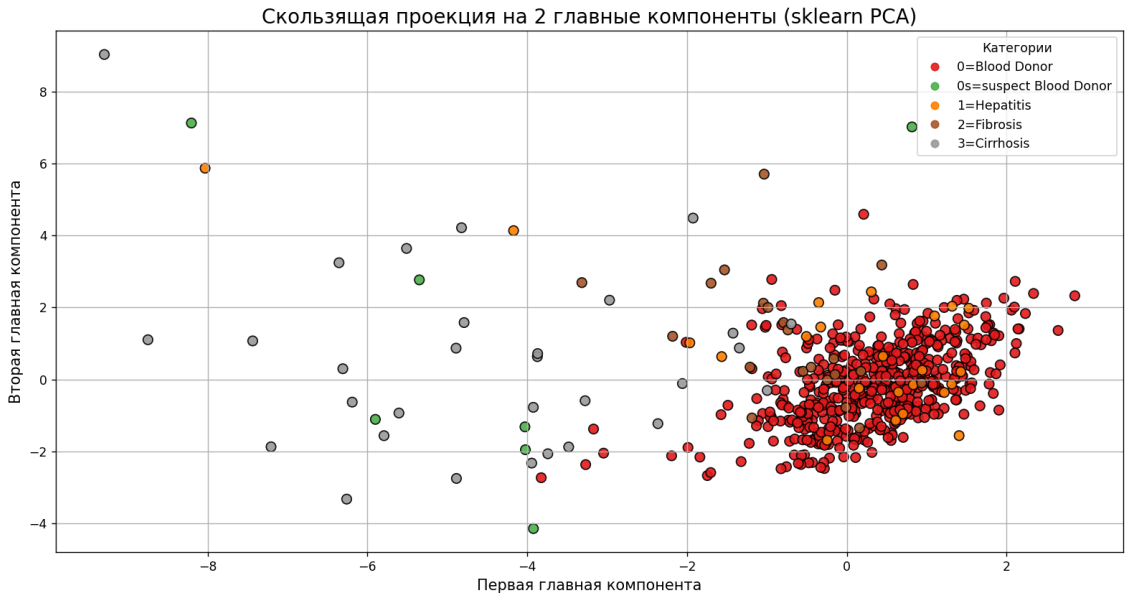
```
plt.ylabel('Вторая главная компонента', fontsize=12)
plt.legend(handles=scatter_sklearn_2d.legend_elements()[0], labels=list(class_names),
title="Категории")
plt.grid(True)
plt.show()

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()
```

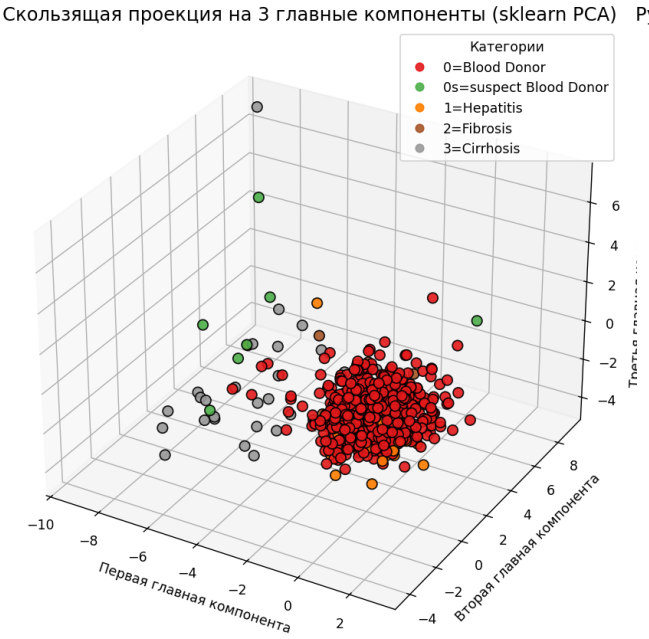
```
print("Визуализации отображены")

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)")

variance_explained_2d_sklearn = np.sum(pca_sklearn_2d.explained_variance_ratio_)
variance_explained_3d_sklearn = np.sum(pca_sklearn_3d.explained_variance_ratio_)

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%}")
```

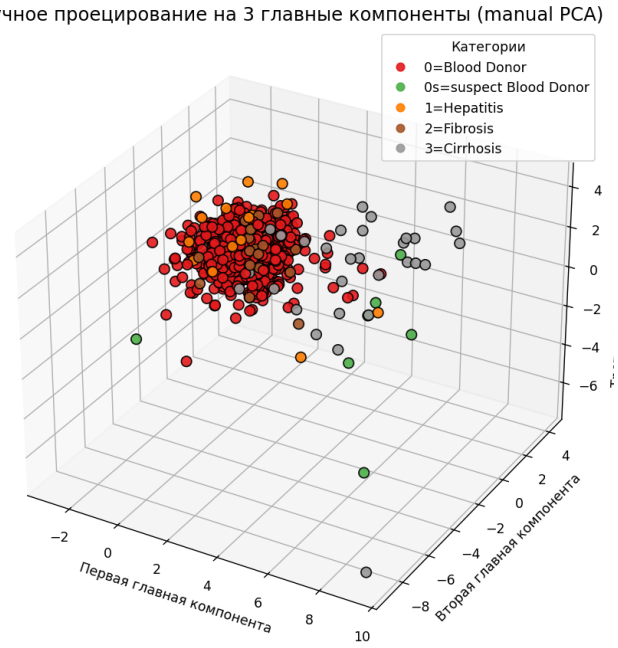
Вывод кода:

```
Анализ информационных потерь (ручное вычисление)
Ручное PCA - Сохраненная дисперсия при 2 компонентах: 36.34%
Ручное PCA - Потери информации при переходе к 2D: 63.66%
----------------------------------------
Ручное PCA - Сохраненная дисперсия при 3 компонентах: 48.16%
Ручное PCA - Потери информации при переходе к 3D: 51.84%

Анализ информационных потерь (sklearn PCA)
sklearn PCA - Сохраненная дисперсия при 2 компонентах: 36.34%
sklearn PCA - Потери информации при переходе к 2D: 63.66%
----------------------------------------
sklearn PCA - Сохраненная дисперсия при 3 компонентах: 48.16%
sklearn PCA - Потери информации при переходе к 3D: 51.84%
```









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