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

Бузель С.Д.

Проверил:

Андренко К. В.

Брест 2025

Цель: научиться применять метод PCA для осуществления визуализации данных.

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
3	exasens.zip	Diagnosis ID

Код программы:

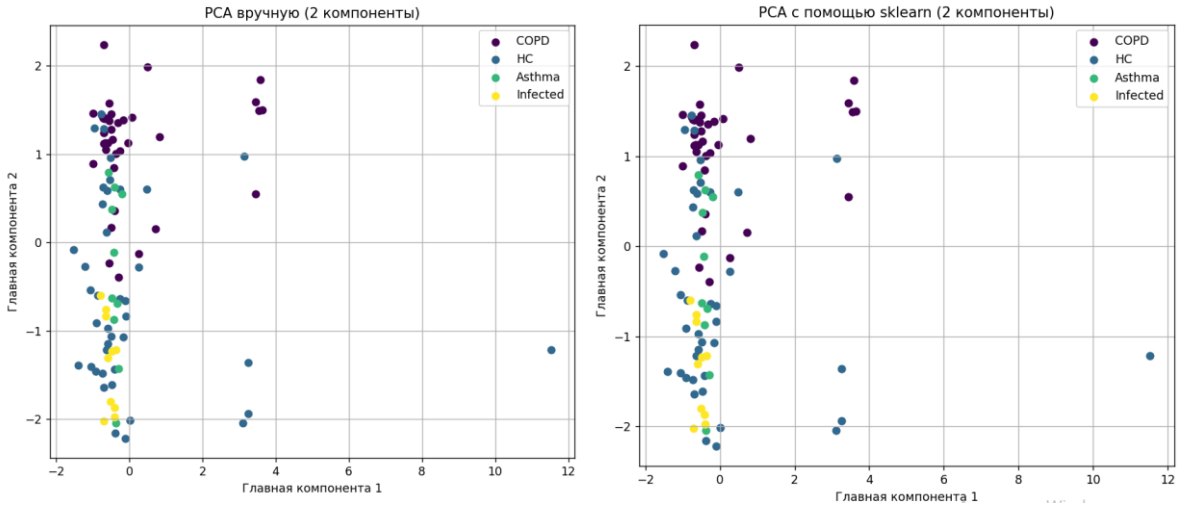
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from mpl_toolkits.mplot3d import Axes3D # Для 3D-графика

# --- Шаг 1: Загрузка и подготовка данных ---
file_path = 'D:/ОИИС/Exasens_cleaned.csv'

try:
    # Напоминание: Pandas по умолчанию использует запятую как разделитель
    data = pd.read_csv(file_path)

    print("Файл успешно загружен. Первые 5 строк данных:")
    print(data.head())
    print("\nИнформация о типах данных:")
    data.info()

except FileNotFoundError:
    print(f"Ошибка: Файл не найден по пути '{file_path}'.")
    print("Пожалуйста, убедитесь, что файл существует и путь указан верно.")
    data = pd.DataFrame()

if not data.empty:

    # 1. Отделение признаков X от целевой переменной y
```

```
    # y это столбец 'Diagnosis ID', который нужен для раскраски графика.
    # X все остальные столбцы с числовыми признаками.
    X = data.drop('Diagnosis ID', axis=1)
    y = data['Diagnosis ID']

    print("\nПризнаки, используемые для PCA:")
    print(X.columns.tolist())

    # 'Gender': 1=Male, 0=Female
    # 'Smoking': 1=Non-smoker, 2=Ex-smoker, 3=Active-smoker.

    # 2. Стандартизация данных
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # --- Шаг 2: Проецирование данных (два способа) ---

    # Способ 1: вручную через использование numpy.linalg.eig для вычисления собственных значений и
собственных векторов
    print("\nВыполнение PCA вручную с помощью NumPy...")
    # Ковариационная матрица
    cov_matrix = np.cov(X_scaled, rowvar=False)
    # Собственные значения и векторы
    eigen_values, eigen_vectors = np.linalg.eig(cov_matrix)
    # Сортировка
    sorted_indices = np.argsort(eigen_values)[::-1]
    sorted_eigen_values = eigen_values[sorted_indices]
    sorted_eigen_vectors = eigen_vectors[:, sorted_indices]
    # Проецирование на 2 и 3 компоненты
    X_projected_manual_2d = np.dot(X_scaled, sorted_eigen_vectors[:, 0:2])
    X_projected_manual_3d = np.dot(X_scaled, sorted_eigen_vectors[:, 0:3])
    print("Ручной метод завершен.")

    # Способ 2: с помощью sklearn.decomposition.PCA
    print("Выполнение PCA с помощью scikit-learn...")
    # Для 2 компонент
    pca_2d = PCA(n_components=2)
    X_projected_sklearn_2d = pca_2d.fit_transform(X_scaled)
    # Для 3 компонент
    pca_3d = PCA(n_components=3)
    X_projected_sklearn_3d = pca_3d.fit_transform(X_scaled)
    print("Метод scikit-learn завершен.")

    # --- Шаг 3: Визуализация главных компонент ---

    print("Создание визуализаций...")

    unique_classes = y.unique()
    colors = plt.cm.get_cmap('viridis', len(unique_classes))

    # Визуализация 2D проекции
    plt.figure(figsize=(14, 7))

    # График для ручного метода
    plt.subplot(1, 2, 1)
    for i, diagnosis_class in enumerate(unique_classes):
        indices = y == diagnosis_class
        plt.scatter(
```

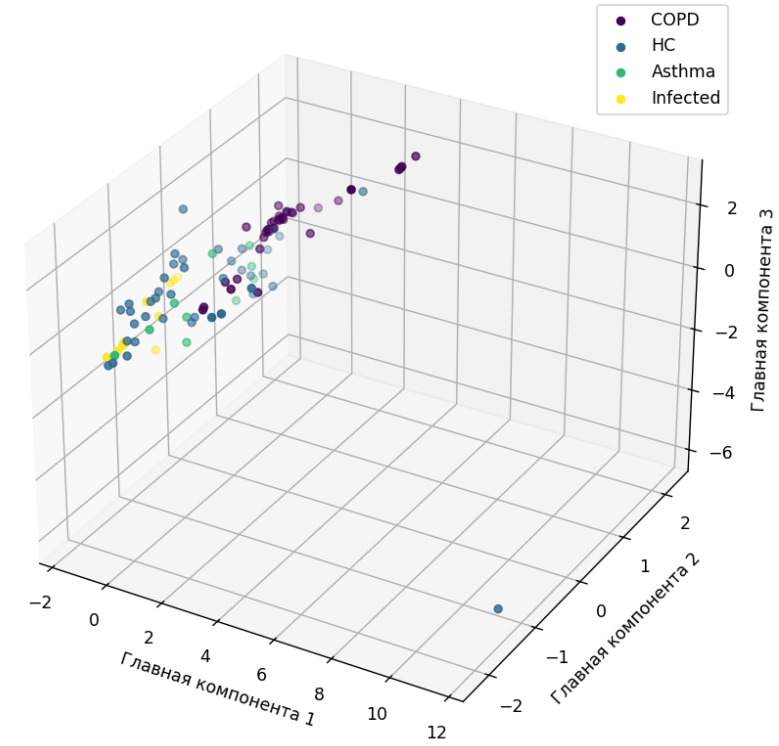
```
        X_projected_manual_2d[indices, 0],
        X_projected_manual_2d[indices, 1],
        color=colors(i),
        label=diagnosis_class
    )
plt.title('PCA вручную (2 компоненты)')
plt.xlabel('Главная компонента 1')
plt.ylabel('Главная компонента 2')
plt.legend()
plt.grid(True)

# График для метода sklearn
plt.subplot(1, 2, 2)
for i, diagnosis_class in enumerate(unique_classes):
    indices = y == diagnosis_class
    plt.scatter(
        X_projected_sklearn_2d[indices, 0],
        X_projected_sklearn_2d[indices, 1],
        color=colors(i),
        label=diagnosis_class
    )
plt.title('PCA с помощью sklearn (2 компоненты)')
plt.xlabel('Главная компонента 1')
plt.ylabel('Главная компонента 2')
plt.legend()
plt.grid(True)

plt.suptitle('Проекция данных на 2 главные компоненты')
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()

# Визуализация 3D проекции
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

for i, diagnosis_class in enumerate(unique_classes):
    indices = y == diagnosis_class
    ax.scatter(
        X_projected_sklearn_3d[indices, 0],
        X_projected_sklearn_3d[indices, 1],
        X_projected_sklearn_3d[indices, 2],
        color=colors(i),
        label=diagnosis_class
    )

ax.set_title('Проекция данных на 3 главные компоненты (sklearn)')
ax.set_xlabel('Главная компонента 1')
ax.set_ylabel('Главная компонента 2')
ax.set_zlabel('Главная компонента 3')
ax.legend()
plt.show()

# --- Шаг 4: Вычисление потерь и выводы ---

total_variance = np.sum(sorted_eigen_values)

# Доля объясненной дисперсии для 2 компонент
explained_variance_2d_ratio = np.sum(sorted_eigen_values[:2]) / total_variance
```

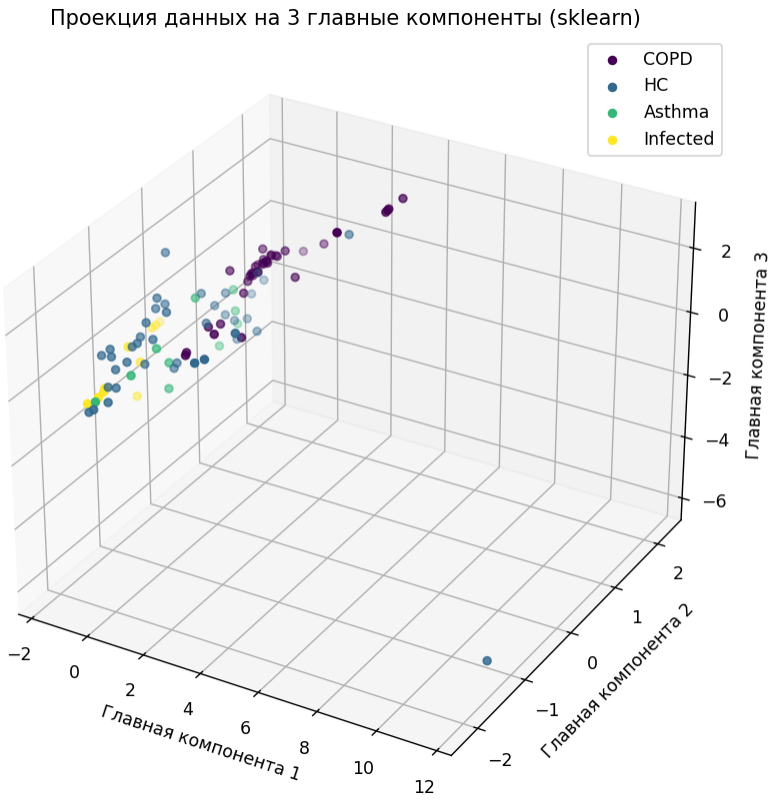
```
loss_2d = 1 - explained_variance_2d_ratio

# Доля объясненной дисперсии для 3 компонент
explained_variance_3d_ratio = np.sum(sorted_eigen_values[:3]) / total_variance
loss_3d = 1 - explained_variance_3d_ratio

print("\n--- Анализ потерь информации ---")
print(f"Доля информации, сохраненная 2 компонентами: {explained_variance_2d_ratio:.2%}")
print(f"Потери при проецировании на 2 компоненты: {loss_2d:.2%}")

print(f"\nДоля информации, сохраненная 3 компонентами: {explained_variance_3d_ratio:.2%}")
print(f"Потери при проецировании на 3 компоненты: {loss_3d:.2%}")
```

Проекция данных на 2 главные компоненты:



Проекция данных на 3 главные компоненты (ручная реализация)





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