МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙГОСУДАРСТВЕННЫЙТЕХНИЧЕСКИЙУНИВЕРСИТЕТ» ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №1

Специальность ИИ-23

Выполнил А.А. Шурина, студент группы ИИ-23

Проверил К.В. Андренко,

ст. преп. кафедры ИИТ,

« » 2025 г.

Брест 2025

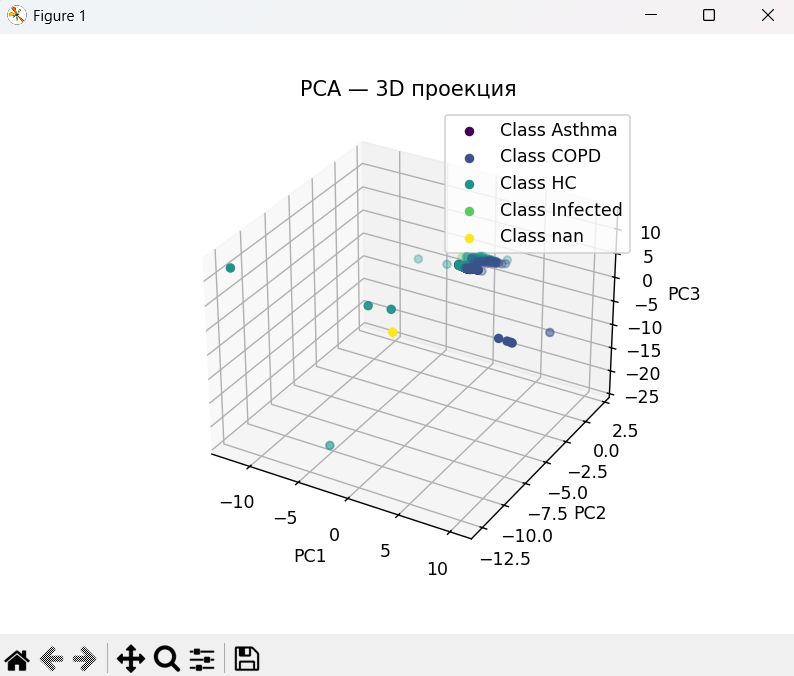
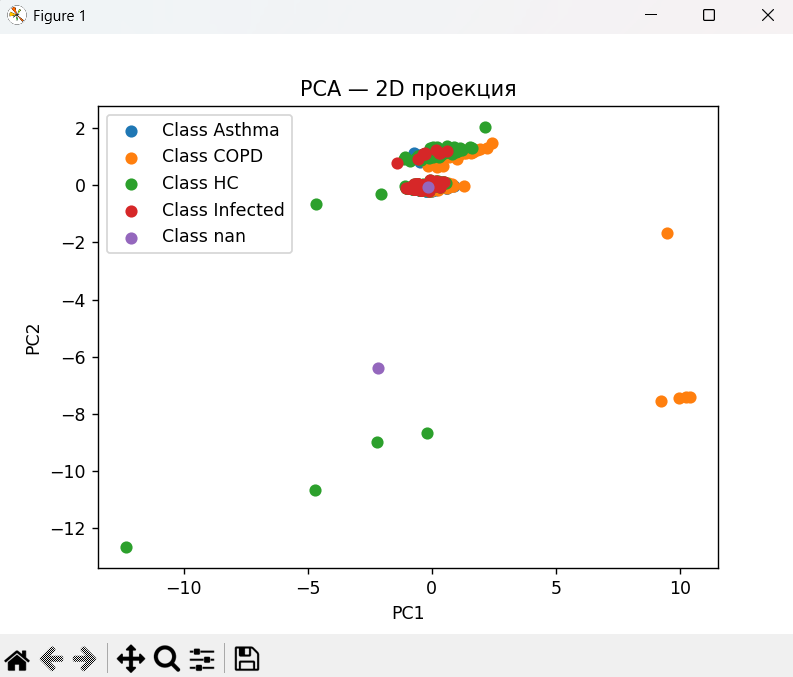
Цель работы: научиться применять метод PCA для осуществления визуализации данных

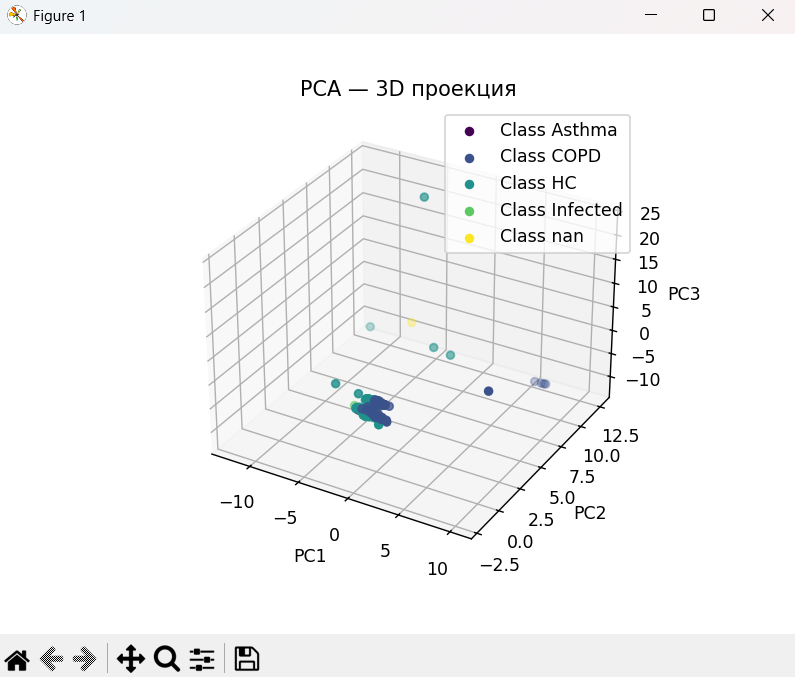
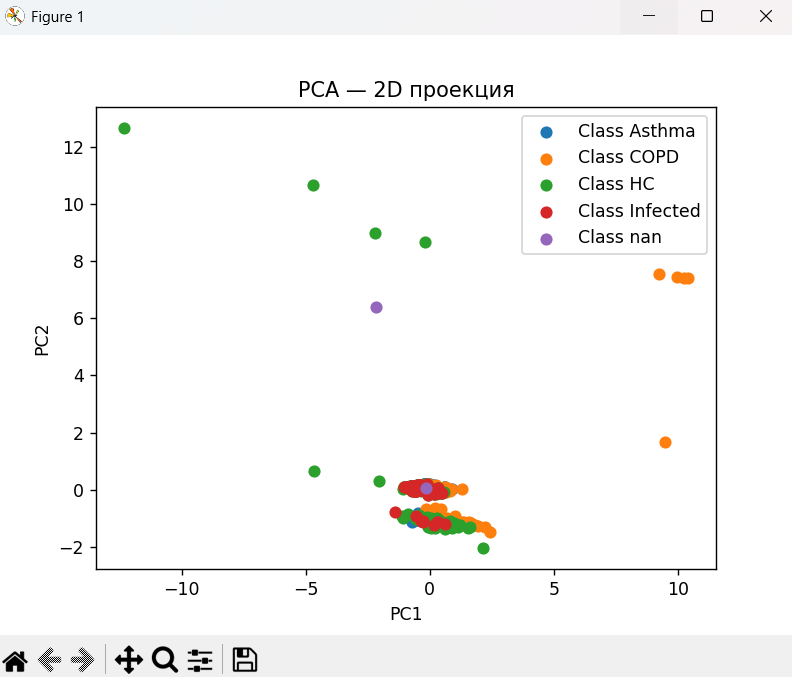
Вариант13



Код программы :

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.decomposition import PCA  
from sklearn.preprocessing import StandardScaler  
  
def plot\_pca(X\_reduced, y):  
 for class\_value in np.unique(y):  
 plt.scatter(X\_reduced[y == class\_value, 0],  
 X\_reduced[y == class\_value, 1],  
 label=f'Class {class\_value}')  
 plt.xlabel('PC1')  
 plt.ylabel('PC2')  
 plt.legend()  
 plt.title('PCA — 2D проекция')  
 plt.show()  
  
def plot\_pca\_3d(X\_reduced\_3, y):  
 fig = plt.figure()  
 ax = fig.add\_subplot(111, projection='3d')  
 unique\_classes = np.unique(y)  
 colors = plt.get\_cmap('viridis', len(unique\_classes))  
 for i, class\_value in enumerate(unique\_classes):  
 ax.scatter(X\_reduced\_3[y == class\_value, 0],  
 X\_reduced\_3[y == class\_value, 1],  
 X\_reduced\_3[y == class\_value, 2],  
 label=f'Class {class\_value}',  
 color=colors(i))  
 ax.set\_xlabel('PC1')  
 ax.set\_ylabel('PC2')  
 ax.set\_zlabel('PC3')  
 ax.legend()  
 plt.title('PCA — 3D проекция')  
 plt.show()  
  
# === Загрузка и очистка данных ===  
data = pd.read\_csv(r"C:\Users\User\Desktop\Studing-7sem\IAD\lab1\Exasens.csv")  
data = data.loc[:, ~data.columns.str.contains('^Unnamed')]  
print("Столбцы после очистки:", data.columns.tolist())  
  
# === Разделение признаков и меток ===  
y = data["Diagnosis"]  
X = data.drop(columns=["Diagnosis", "ID"])  
  
# Преобразуем категориальные переменные  
X = pd.get\_dummies(X, drop\_first=True)  
X = X.astype(float)  
X = X.replace([np.inf, -np.inf], np.nan)  
X = X.fillna(X.mean())  
  
# Нормализация  
from sklearn.preprocessing import StandardScaler  
scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X)  
  
# --- Добавляем это ---  
y = y.astype(str) # <--- Преобразуем метки в строки  
# ----------------------  
  
# PCA вручную  
cov\_matrix = np.cov(X\_scaled, rowvar=False)  
eigen\_values, eigen\_vectors = np.linalg.eig(cov\_matrix)  
  
sorted\_index = np.argsort(eigen\_values)[::-1]  
sorted\_eigenvalue = eigen\_values[sorted\_index]  
sorted\_eigenvectors = eigen\_vectors[:, sorted\_index]  
  
# 2D и 3D проекции  
eigenvector\_subset\_2 = sorted\_eigenvectors[:, 0:2]  
eigenvector\_subset\_3 = sorted\_eigenvectors[:, 0:3]  
  
X\_reduced\_2 = np.dot(X\_scaled, eigenvector\_subset\_2)  
X\_reduced\_3 = np.dot(X\_scaled, eigenvector\_subset\_3)  
  
# Визуализация  
plot\_pca(X\_reduced\_2, y)  
plot\_pca\_3d(X\_reduced\_3, y)  
  
# === PCA через sklearn ===  
pca = PCA(n\_components=2)  
X\_reduced\_sklearn\_2 = pca.fit\_transform(X\_scaled)  
  
pca\_3 = PCA(n\_components=3)  
X\_reduced\_sklearn\_3 = pca\_3.fit\_transform(X\_scaled)  
  
plot\_pca(X\_reduced\_sklearn\_2, y)  
plot\_pca\_3d(X\_reduced\_sklearn\_3, y)  
  
# === Объяснённая дисперсия ===  
explained\_variance\_2 = np.sum(sorted\_eigenvalue[:2]) / np.sum(sorted\_eigenvalue)  
explained\_variance\_3 = np.sum(sorted\_eigenvalue[:3]) / np.sum(sorted\_eigenvalue)  
  
print(f"\nОбъяснённая дисперсия для 2D проекции: {explained\_variance\_2 \* 100:.2f}%")  
print(f"Объяснённая дисперсия для 3D проекции: {explained\_variance\_3 \* 100:.2f}%")

Результаты:  




Вывод: научилась применять метод PCA для осуществления визуализации данных.