МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №1

Специальность ИИ(з)

Выполнил

А. Ю. Кураш,

студент группы ИИ-24

Проверил

А. А. Крощенко,

ст. преп. кафедры ИИТ,

« k 2025 г.

Брест 2025

Цель работы:

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;

Выполнение:

# **Код программы**

# 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)

# # Загружаем и очищаем датасет

# file\_path = "Exasens.csv"

# data\_raw = pd.read\_csv(file\_path)

# # Удаляем первые 2 строки с метаданными

# data = data\_raw.iloc[2:].reset\_index(drop=True)

# # Удаляем пустые столбцы

# cols\_to\_drop = [col for col in data.columns if data[col].isnull().sum() == len(data[col])]

# data = data.drop(columns=cols\_to\_drop)

# # Преобразуем числовые колонки

# numeric\_cols = ['Imaginary Part', 'Unnamed: 3', 'Real Part', 'Unnamed: 5', 'Gender', 'Age', 'Smoking']

# for col in numeric\_cols:

# data[col] = pd.to\_numeric(data[col], errors='coerce')

# # Заполняем пропуски медианами

# for col in numeric\_cols:

# if data[col].isnull().sum() > 0:

# data[col] = data[col].fillna(data[col].median())

# # Целевая переменная (Diagnosis)

# y\_str = data['Diagnosis']

# label\_encoder = LabelEncoder()

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

# class\_names = label\_encoder.classes\_

# X = data[numeric\_cols]

# scaler = StandardScaler()

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

# # --- Ручное PCA ---

# 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)

# # --- Визуализация ---

# 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('PCA sklearn: проекция на 2 компоненты', fontsize=16)

# plt.xlabel('PC1', fontsize=12)

# plt.ylabel('PC2', fontsize=12)

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

# 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('PCA manual: проекция на 2 компоненты', fontsize=16)

# plt.xlabel('PC1', fontsize=12)

# plt.ylabel('PC2', fontsize=12)

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

# 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('PCA sklearn: 3D', fontsize=14)

# ax1.set\_xlabel('PC1')

# ax1.set\_ylabel('PC2')

# ax1.set\_zlabel('PC3')

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

# 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('PCA manual: 3D', fontsize=14)

# ax2.set\_xlabel('PC1')

# ax2.set\_ylabel('PC2')

# ax2.set\_zlabel('PC3')

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

# plt.tight\_layout()

# plt.show()

# # --- Анализ потерь информации ---

# print("\nАнализ информационных потерь (manual PCA)")

# 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

# print(f"Manual PCA - сохраненная дисперсия 2D: {variance\_explained\_2d\_manual:.2%}")

# print(f"Manual PCA - потери при 2D: {1 - variance\_explained\_2d\_manual:.2%}")

# print(f"Manual PCA - сохраненная дисперсия 3D: {variance\_explained\_3d\_manual:.2%}")

# print(f"Manual PCA - потери при 3D: {1 - variance\_explained\_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

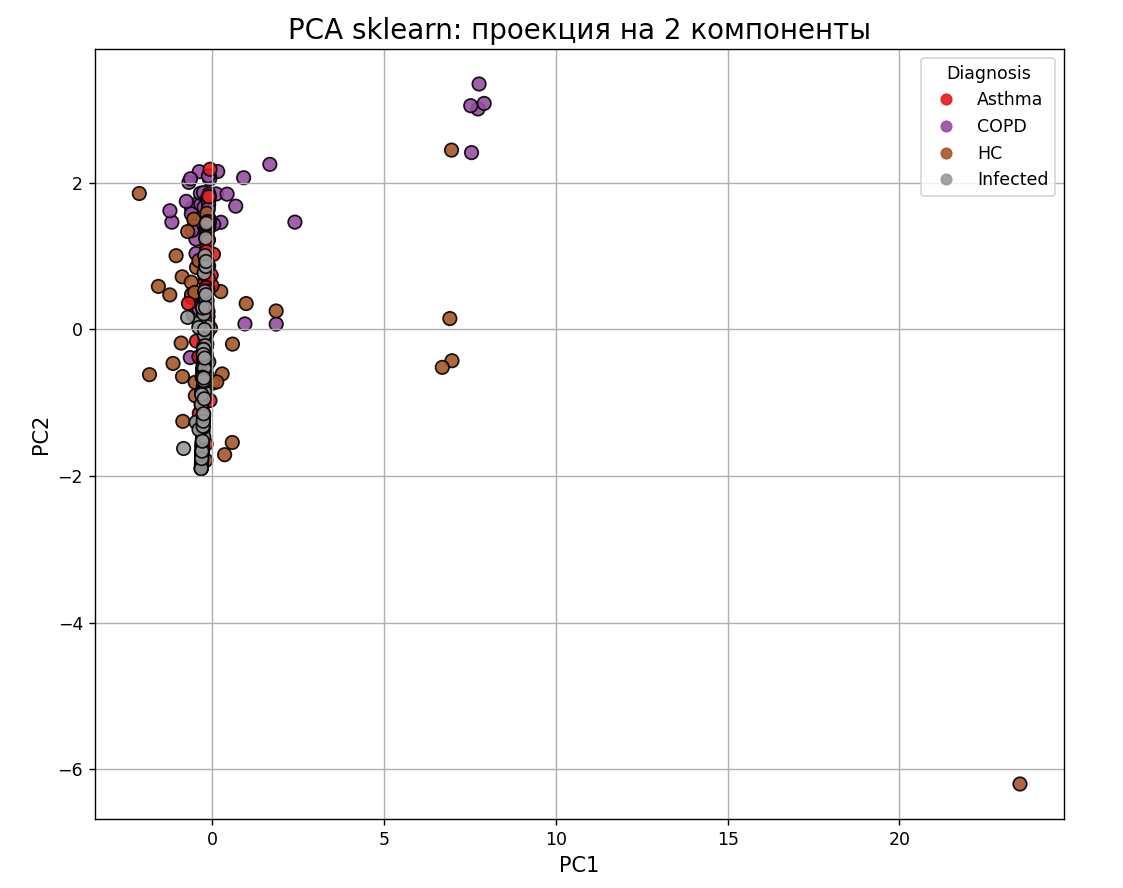
# print(f"sklearn PCA - сохраненная дисперсия 2D: {variance\_explained\_2d\_sklearn:.2%}")

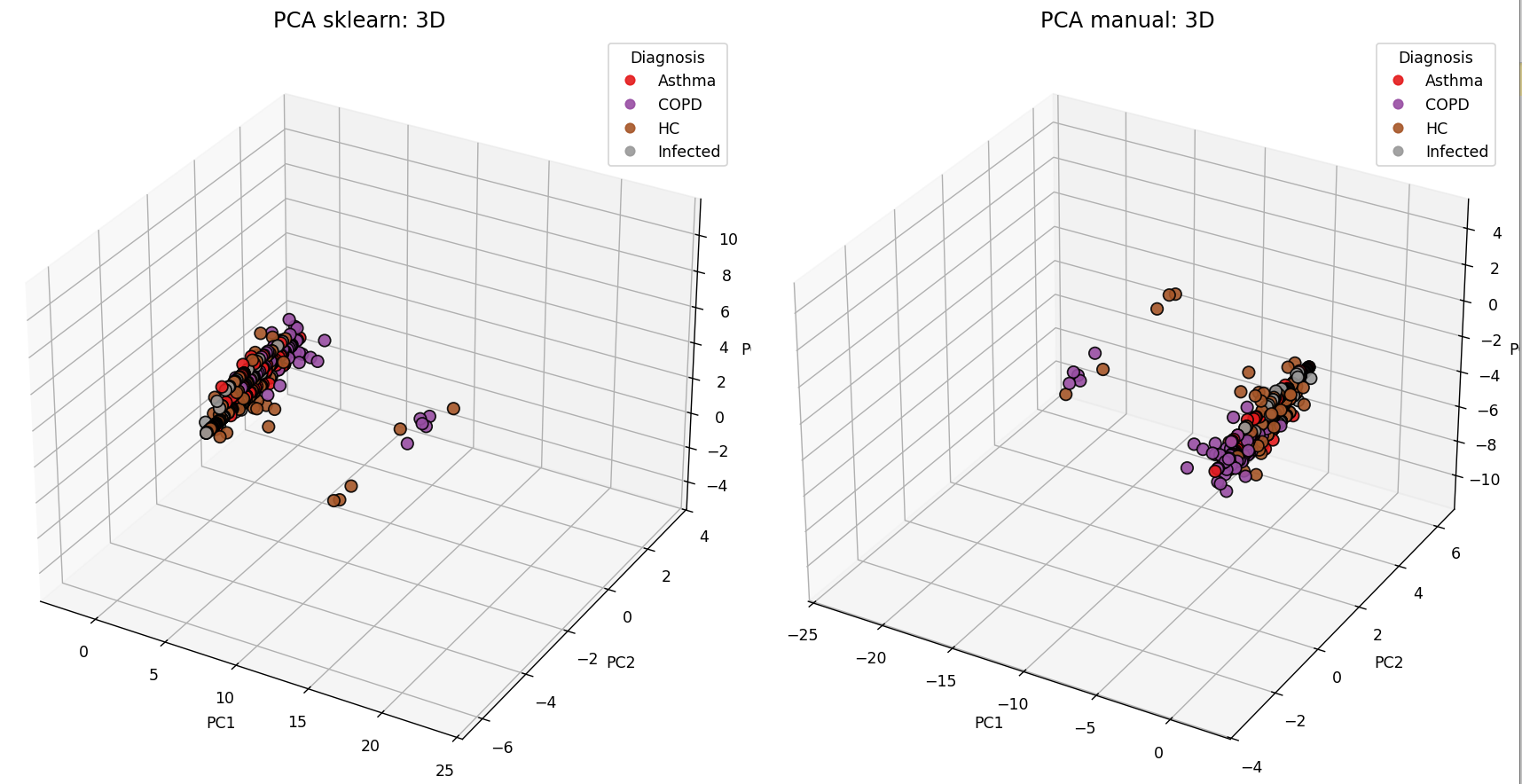
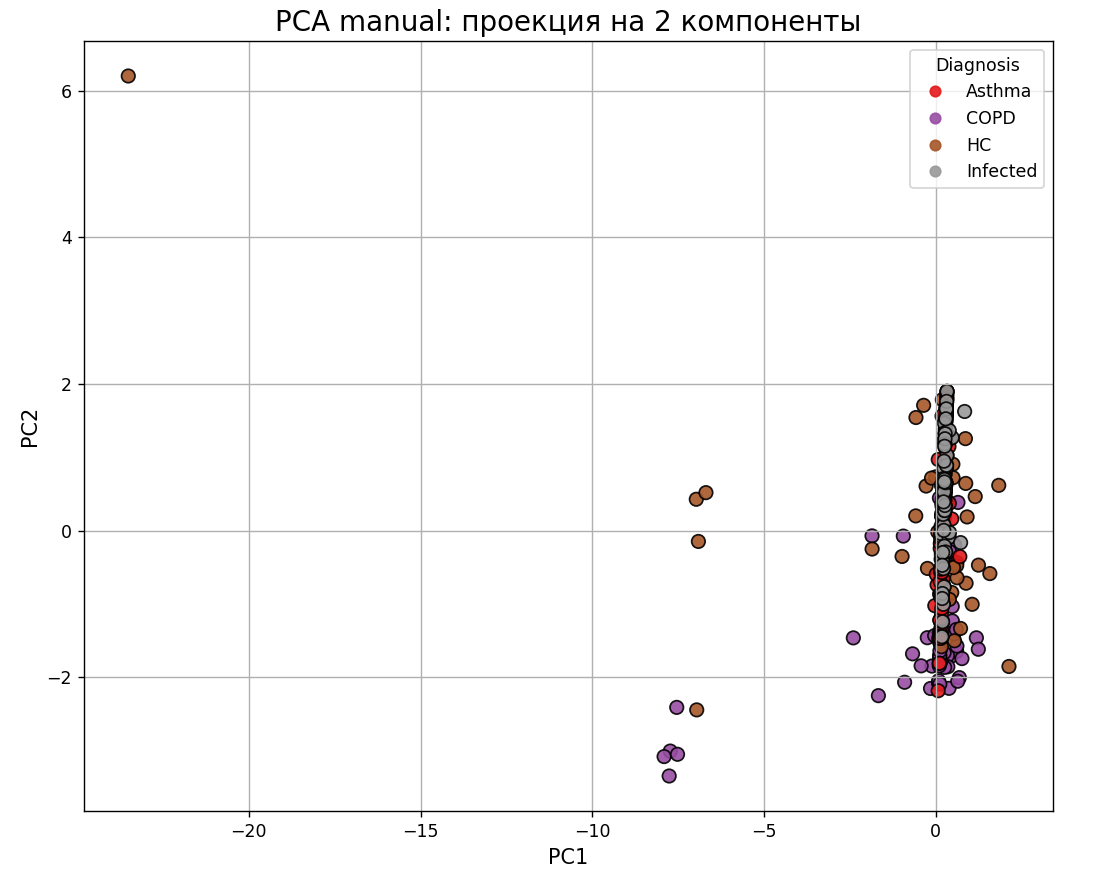
# print(f"sklearn PCA - потери при 2D: {1 - variance\_explained\_2d\_sklearn:.2%}")

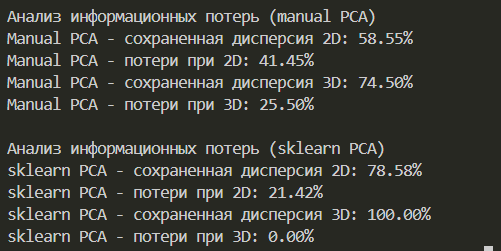
# print(f"sklearn PCA - сохраненная дисперсия 3D: {variance\_explained\_3d\_sklearn:.2%}")

# print(f"sklearn PCA - потери при 3D: {1 - variance\_explained\_3d\_sklearn:.2%}")

# Рисунки с результатами работы программы







**Вывод:** Янаучился работать с нейросетями и изучил метод PCA.