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
Андренко К.В,
Преподаватель-стажер 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

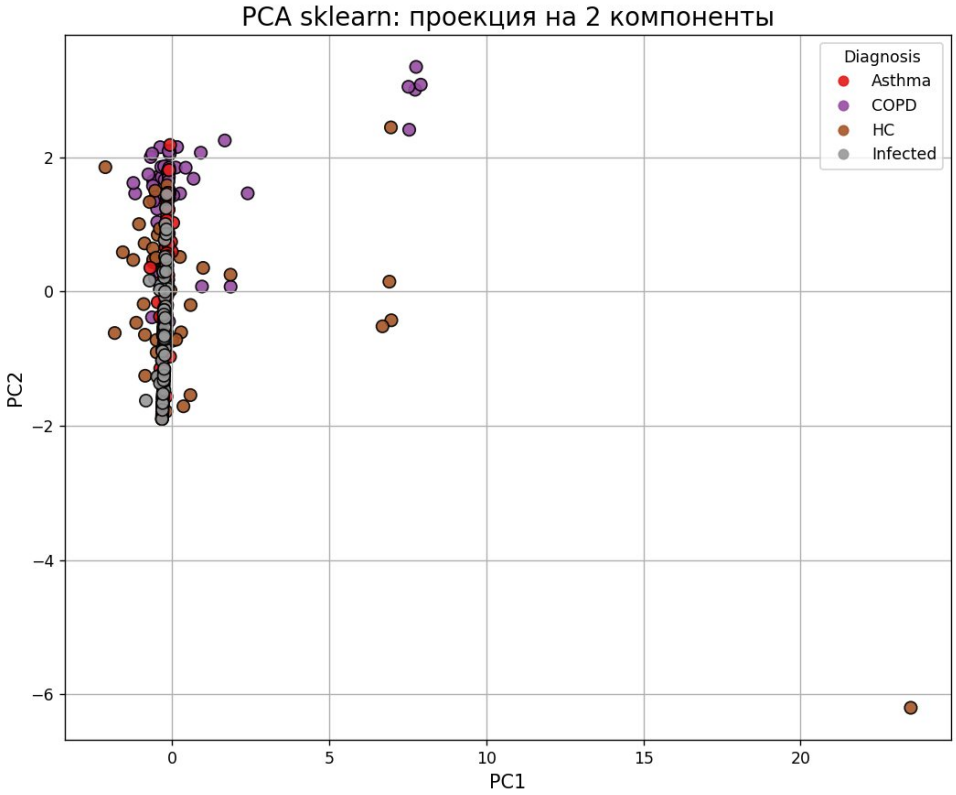
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.decomposition import PCA
import warnings

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())
```

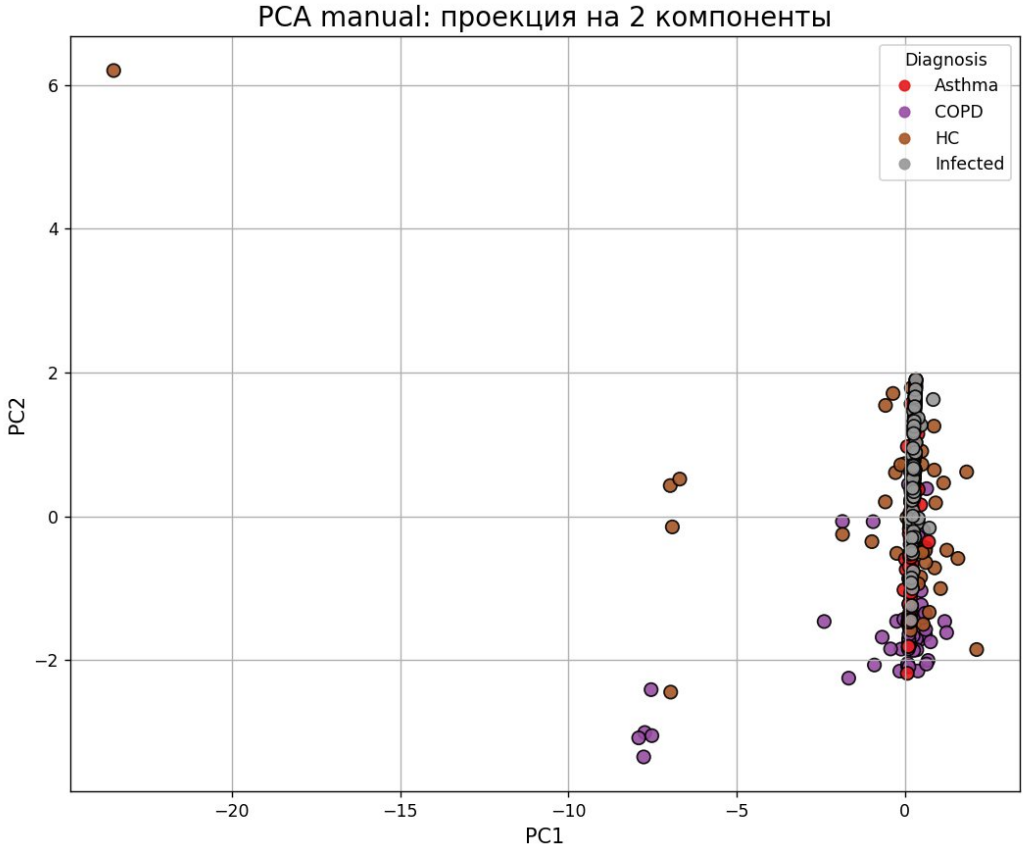
```
# Целевая переменная (Diagnosis)
y_str = data['Diagnosis']
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y_str)
class_names = label_encoder.classes_

# Признаки для PCA
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)

# PCA без ограничения числа компонент (для расчёта полной дисперсии)
pca_full = PCA()
pca_full.fit(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),
```

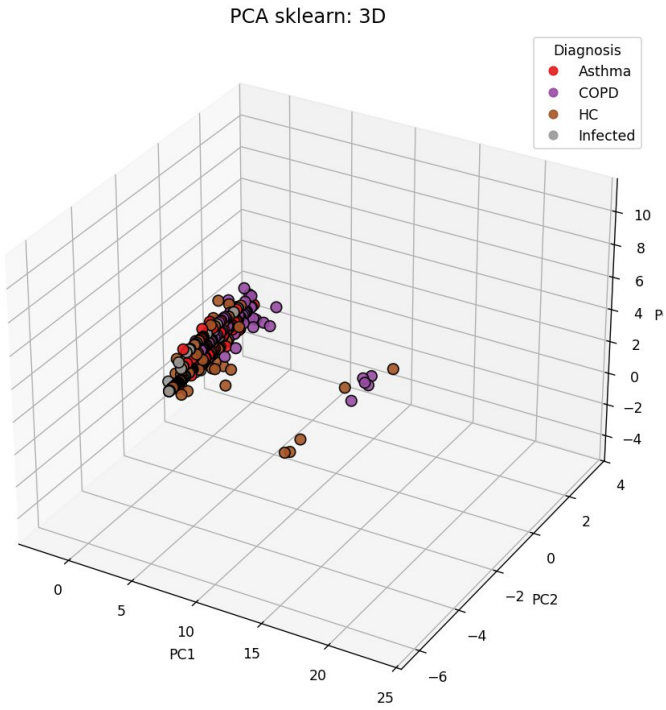
```
title="Diagnosis")
plt.grid(True)
plt.show()
plt.figure(figsize=(10, 8))
scatter_manual_2d = plt.scatter(
    X_pca_manual_2d[:, 0], X_pca_manual_2d[:, 1],
    c=y, cmap=cmap, alpha=0.9, edgecolor='k', s=60
)
plt.title('PCA manual: проекция на 2 компоненты', fontsize=16)
plt.xlabel('PC1', fontsize=12)
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)
```

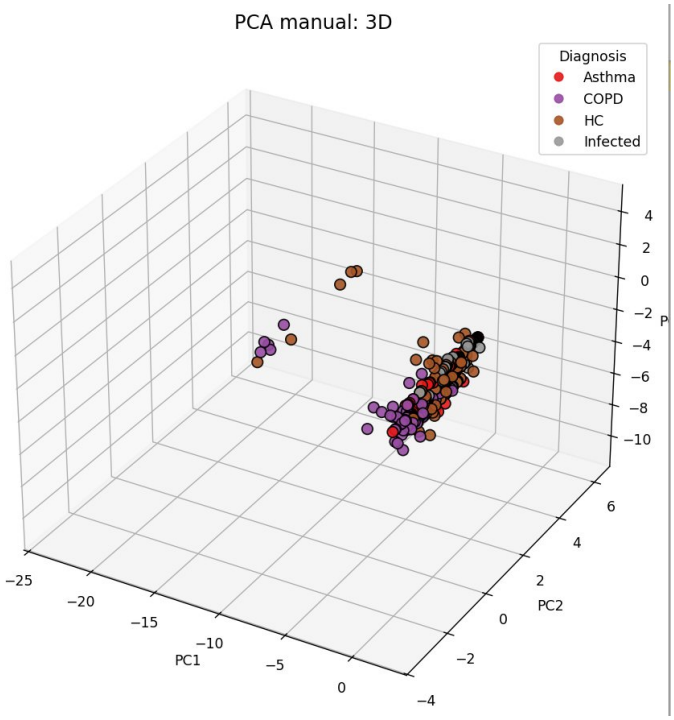
```
variance_explained_2d_manual = np.sum(sorted_eigen_values[:2]) / total_variance_manual
variance_explained_3d_manual = np.sum(sorted_eigen_values[:3]) / total_variance_manual
print(f"Manual PCA - сохраненная дисперсия 2D: {variance_explained_2d_manual:.2%}")
print(f"Manual PCA - потери при 2D: {1 - 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_full.explained_variance_)
variance_explained_2d_sklearn = np.sum(pca_full.explained_variance_[:2]) /
total_variance_sklearn
variance_explained_3d_sklearn = np.sum(pca_full.explained_variance_[:3]) /
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




PCA manual: проекция на 2 компоненты
Diagnosis
Asthma
COPD
HC
Infected
PC1
PC2


PCA sklearn: 3D
Diagnosis
Asthma
COPD
HC
Infected
PC1
PC2


PCA manual: 3D
Diagnosis
Asthma
COPD
HC
Infected
PC1
PC2

```
Анализ информационных потерь (manual PCA)
Manual PCA - сохраненная дисперсия 2D: 58.55%
Manual PCA - потери при 2D: 41.45%
Manual PCA - сохраненная дисперсия 3D: 74.50%
Manual PCA - потери при 3D: 25.50%

Анализ информационных потерь (sklearn PCA)
sklearn PCA - сохраненная дисперсия 2D: 78.58%
sklearn PCA - потери при 2D: 21.42%
sklearn PCA - сохраненная дисперсия 3D: 100.00%
sklearn PCA - потери при 3D: 0.00%
```

Вывод: Я изучил метод PCA.