Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине: «Автоэнкодеры»

Тема: «PCA»

Выполнил:

Студент 4 курса

Группы ИИ-23

Бусень А.Д.

Проверила:

Андренко К.В.

Брест 2025

Цель работы: научиться применять автоэнкодеры для осуществления визуализации данных и их анализа.

Вариант 1

№	Выборка	Класс
1	Wisconsin Diagnostic Breast Cancer (WDBC)	2-й признак

Код программы:

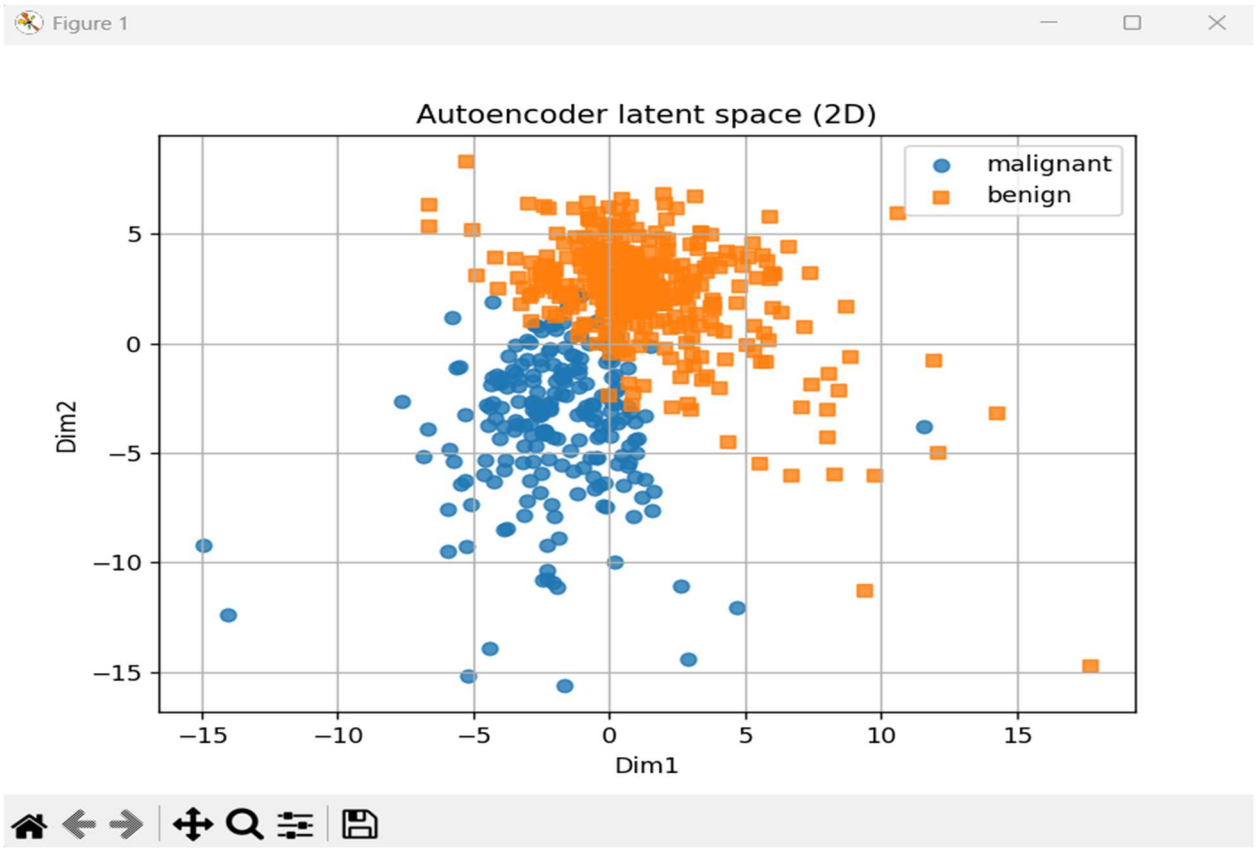
```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA as SKPCA
from sklearn.manifold import TSNE
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras import layers, models, callbacks, optimizers

data = load_breast_cancer()
X = data.data.copy()
y = data.target.copy()
feature_names = data.feature_names
class_names = data.target_names

print("Dataset:", data.DESCR.splitlines()[0])
print("Shape X:", X.shape, "Classes:", np.unique(y))

scaler = StandardScaler()
X_std = scaler.fit_transform(X)

X_train, X_val = train_test_split(X_std, test_size=0.2, random_state=42, stratify=y)

markers = ['o','s','D','^','v','P','*']
colors = ['C0','C1','C2','C3','C4']

def build_autoencoder(input_dim, latent_dim, hidden_units=[64,32], activation='relu', lr=1e-3):
    inp = layers.Input(shape=(input_dim,))
    x = inp
    for h in hidden_units:
        x = layers.Dense(h, activation=activation)(x)
    z = layers.Dense(latent_dim, activation=None, name='bottleneck')(x)  # линейный для легкой интерпретации
    x = z
    for h in hidden_units[::-1]:
        x = layers.Dense(h, activation=activation)(x)
    out = layers.Dense(input_dim, activation=None)(x)  # регрессия на стандартизованные признаки

    auto = models.Model(inp, out, name=f'AE_{latent_dim}d')
    enc = models.Model(inp, z, name=f'Encoder_{latent_dim}d')
```

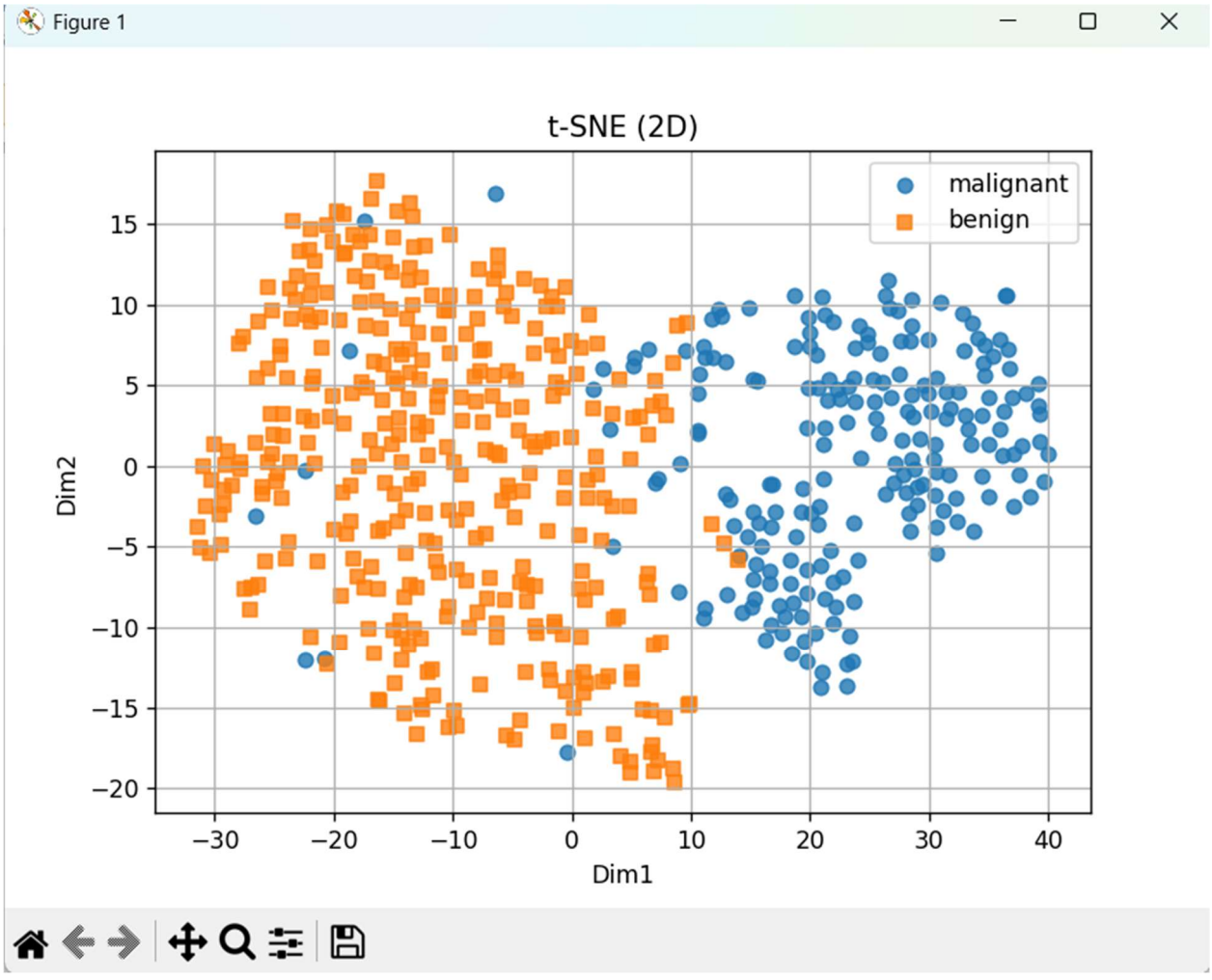
```
    auto.compile(optimizer=optimizers.Adam(lr), loss='mse')
    return auto, enc

ae_results = {}
for latent in (2, 3):
    auto, enc = build_autoencoder(input_dim=X_std.shape[1], latent_dim=latent,
                                  hidden_units=[64, 32], activation='relu', lr=1e-3)
    cb = [callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True,
verbose=0)]
    history = auto.fit(X_train, X_train,
                       validation_data=(X_val, X_val),
                       epochs=200,
                       batch_size=32,
                       callbacks=cb,
                       verbose=0)
    Z = enc.predict(X_std)
    recon = auto.predict(X_std)
    mse = np.mean((X_std - recon) ** 2)
    ae_results[latent] = {'auto': auto, 'enc': enc, 'Z': Z, 'mse': mse, 'history': history}
    print(f"Autoencoder {latent}D trained. Reconstruction MSE (std features): {mse:.6f}")

print("\nComputing t-SNE (may take some time)...")
tsne_2 = TSNE(n_components=2, perplexity=30, random_state=42, init='pca')
Z_tsne2 = tsne_2.fit_transform(X_std)

tsne_3 = TSNE(n_components=3, perplexity=30, random_state=42, init='pca')
Z_tsne3 = tsne_3.fit_transform(X_std)

cov = np.cov(X_std, rowvar=False)
eigvals, eigvecs = np.linalg.eig(cov)
eigvals = eigvals.real
eigvecs = eigvecs.real
order = np.argsort(eigvals)[::-1]
eigvals_sorted = eigvals[order]
eigvecs_sorted = eigvecs[:, order]
explained_ratio_manual = eigvals_sorted / eigvals_sorted.sum()
proj2_manual = X_std.dot(eigvecs_sorted[:, :2])
proj3_manual = X_std.dot(eigvecs_sorted[:, :3])

pca2 = SKPCA(n_components=2)
proj2_sklearn = pca2.fit_transform(X_std)
pca3 = SKPCA(n_components=3)
proj3_sklearn = pca3.fit_transform(X_std)

explained_sklearn = pca3.explained_variance_ratio_

out_dir = 'pca_tsne_ae_results'
os.makedirs(out_dir, exist_ok=True)

pd.DataFrame(proj2_manual,
columns=['PC1','PC2']).assign(Class=y).to_csv(os.path.join(out_dir,'proj2_manual.csv'), index=False)
```

```
pd.DataFrame(proj3_manual,
columns=['PC1','PC2','PC3']).assign(Class=y).to_csv(os.path.join(out_dir,'proj3_manual.csv'),
index=False)
pd.DataFrame(ae_results[2]['Z'],
columns=['Z1','Z2']).assign(Class=y).to_csv(os.path.join(out_dir,'ae2_proj.csv'), index=False)
pd.DataFrame(ae_results[3]['Z'],
columns=['Z1','Z2','Z3']).assign(Class=y).to_csv(os.path.join(out_dir,'ae3_proj.csv'), index=False)
pd.DataFrame(Z_tsne2,
columns=['TSNE1','TSNE2']).assign(Class=y).to_csv(os.path.join(out_dir,'tsne2_proj.csv'), index=False)
pd.DataFrame(Z_tsne3,
columns=['TSNE1','TSNE2','TSNE3']).assign(Class=y).to_csv(os.path.join(out_dir,'tsne3_proj.csv'),
index=False)

pd.DataFrame({
    'PC': [f'PC{i+1}' for i in range(len(eigvals_sorted))],
    'Eigenvalue': eigvals_sorted,
    'ExplainedVarRatio_manual': explained_ratio_manual,
    'Cumulative_manual': np.cumsum(explained_ratio_manual)
}).to_csv(os.path.join(out_dir,'explained_variance_manual.csv'), index=False)

print("Проекции сохранены в папке:", out_dir)

def plot_2d(Z, labels, title, filename=None):
    plt.figure(figsize=(7,5))
    unique = np.unique(labels)
    for i, cls in enumerate(unique):
        mask = (labels == cls)
        plt.scatter(Z[mask,0], Z[mask,1], label=f'{class_names[cls]}', marker=markers[i%len(markers)],
alpha=0.8)
    plt.xlabel('Dim1'); plt.ylabel('Dim2'); plt.title(title); plt.legend(); plt.grid(True)
    if filename:
        plt.savefig(filename, bbox_inches='tight', dpi=150)
    plt.show()

def plot_3d(Z, labels, title, filename=None):
    fig = plt.figure(figsize=(8,6))
    ax = fig.add_subplot(111, projection='3d')
    unique = np.unique(labels)
    for i, cls in enumerate(unique):
        mask = (labels == cls)
        ax.scatter(Z[mask,0], Z[mask,1], Z[mask,2], label=f'{class_names[cls]}',
marker=markers[i%len(markers)], alpha=0.8)
    ax.set_xlabel('Dim1'); ax.set_ylabel('Dim2'); ax.set_zlabel('Dim3')
    ax.set_title(title); ax.legend()
    if filename:
        plt.savefig(filename, bbox_inches='tight', dpi=150)
    plt.show()

plot_2d(ae_results[2]['Z'], y, 'Autoencoder latent space (2D)', os.path.join(out_dir,'ae2_2d.png'))
plot_3d(ae_results[3]['Z'], y, 'Autoencoder latent space (3D)', os.path.join(out_dir,'ae3_3d.png'))
```

```
plot_2d(Z_tsne2, y, 't-SNE (2D)', os.path.join(out_dir,'tsne2_2d.png'))
plot_3d(Z_tsne3, y, 't-SNE (3D)', os.path.join(out_dir,'tsne3_3d.png'))

plot_2d(proj2_manual, y, 'PCA manual (2D)', os.path.join(out_dir,'pca_manual_2d.png'))
plot_2d(proj2_sklearn, y, 'PCA sklearn (2D)', os.path.join(out_dir,'pca_sklearn_2d.png'))
plot_3d(proj3_manual, y, 'PCA manual (3D)', os.path.join(out_dir,'pca_manual_3d.png'))
plot_3d(proj3_sklearn, y, 'PCA sklearn (3D)', os.path.join(out_dir,'pca_sklearn_3d.png'))

from scipy.stats import pearsonr

def compare_latent_spaces(A, B, top_k=None):
    if top_k is None:
        top_k = min(A.shape[1], B.shape[1])
    diffs = []
    for i in range(top_k):
        corr, _ = pearsonr(A[:, i], B[:, i])
        diff_percent = (1 - abs(corr)) * 100.0
        diffs.append(diff_percent)
    return diffs

diffs_ae2_pca2 = compare_latent_spaces(ae_results[2]['Z'], proj2_manual, top_k=2)
diffs_ae3_pca3 = compare_latent_spaces(ae_results[3]['Z'], proj3_manual, top_k=3)

diffs_pca_manual_sklearn_2 = compare_latent_spaces(proj2_manual, proj2_sklearn, top_k=2)
diffs_pca_manual_sklearn_3 = compare_latent_spaces(proj3_manual, proj3_sklearn, top_k=3)

expl_manual_top3 = explained_ratio_manual[:3]
expl_sklearn_top3 = np.concatenate([explained_sklearn, np.zeros(max(0, len(expl_manual_top3)-
len(explained_sklearn)))])[:3]
expl_diff_pct = np.abs(expl_manual_top3 - expl_sklearn_top3) * 100.0

print("\n=== Сравнение представлений (в процентах расхождения = (1-|corr|)*100) ===")
print("AE(2) vs PCA_manual(2):", [f"{d:.4f}%" for d in diffs_ae2_pca2])
print("AE(3) vs PCA_manual(3):", [f"{d:.4f}%" for d in diffs_ae3_pca3])
print("PCA_manual vs PCA_sklearn (2D):", [f"{d:.6f}%" for d in diffs_pca_manual_sklearn_2])
print("PCA_manual vs PCA_sklearn (3D):", [f"{d:.6f}%" for d in diffs_pca_manual_sklearn_3])

print("\n=== Сравнение объяснённой дисперсии (top-3), в %-пунктах ===")
for i,(m,s,d) in enumerate(zip(expl_manual_top3, expl_sklearn_top3, expl_diff_pct), start=1):
    print(f"PC{i}: manual={m:.6f}, sklearn={s:.6f}, diff={d:.6f} percentage points")

print("\n=== MSE реконструкции автоэнкодеров (на стандартизованных признаках) ===")
for latent in (2,3):
    print(f"AE {latent}D MSE = {ae_results[latent]['mse']:.6f}")
```

Результат работы программы:





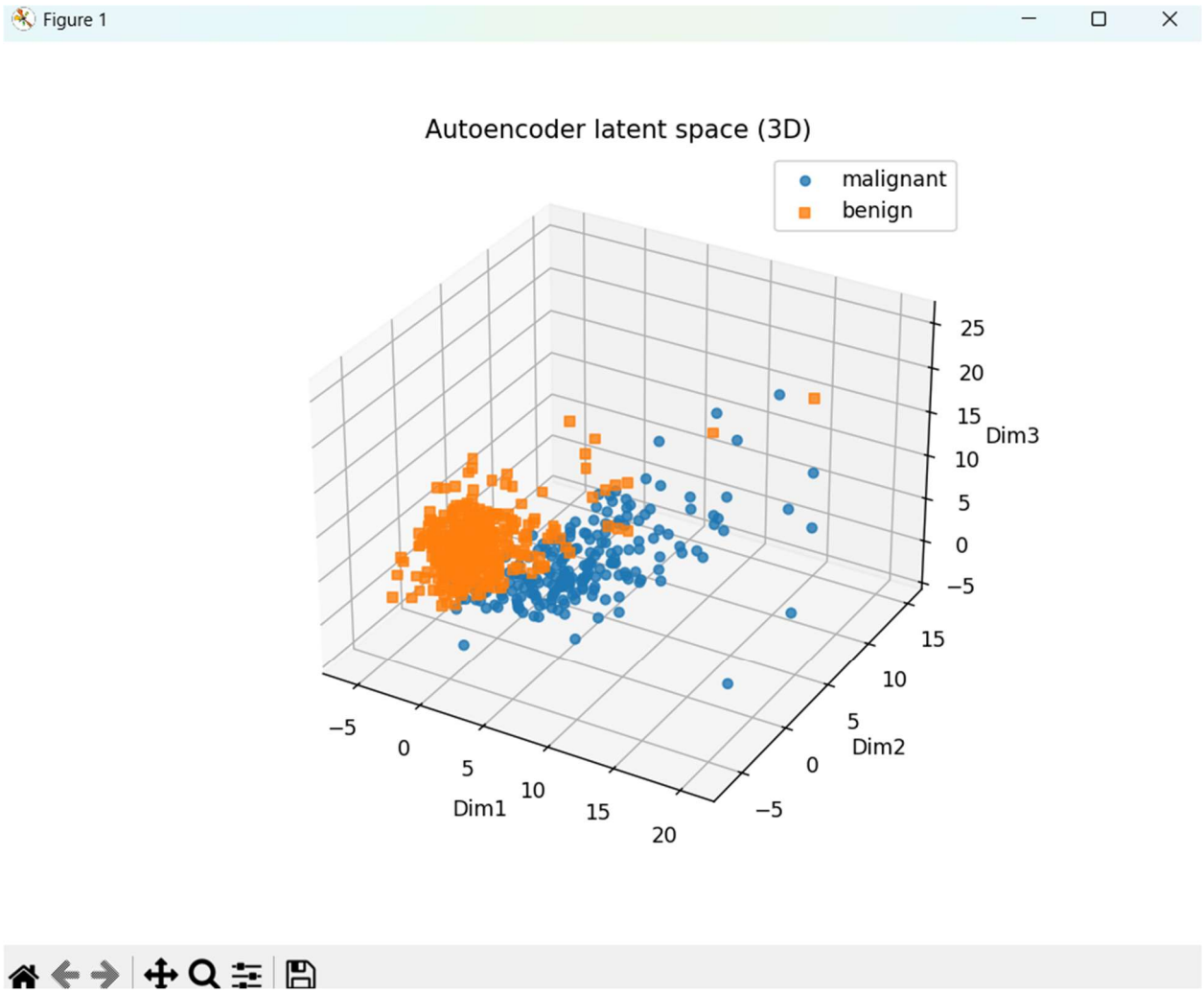


Figure 1
Autoencoder latent space (3D)
malignant
benign
Dim1
Dim2
Dim3

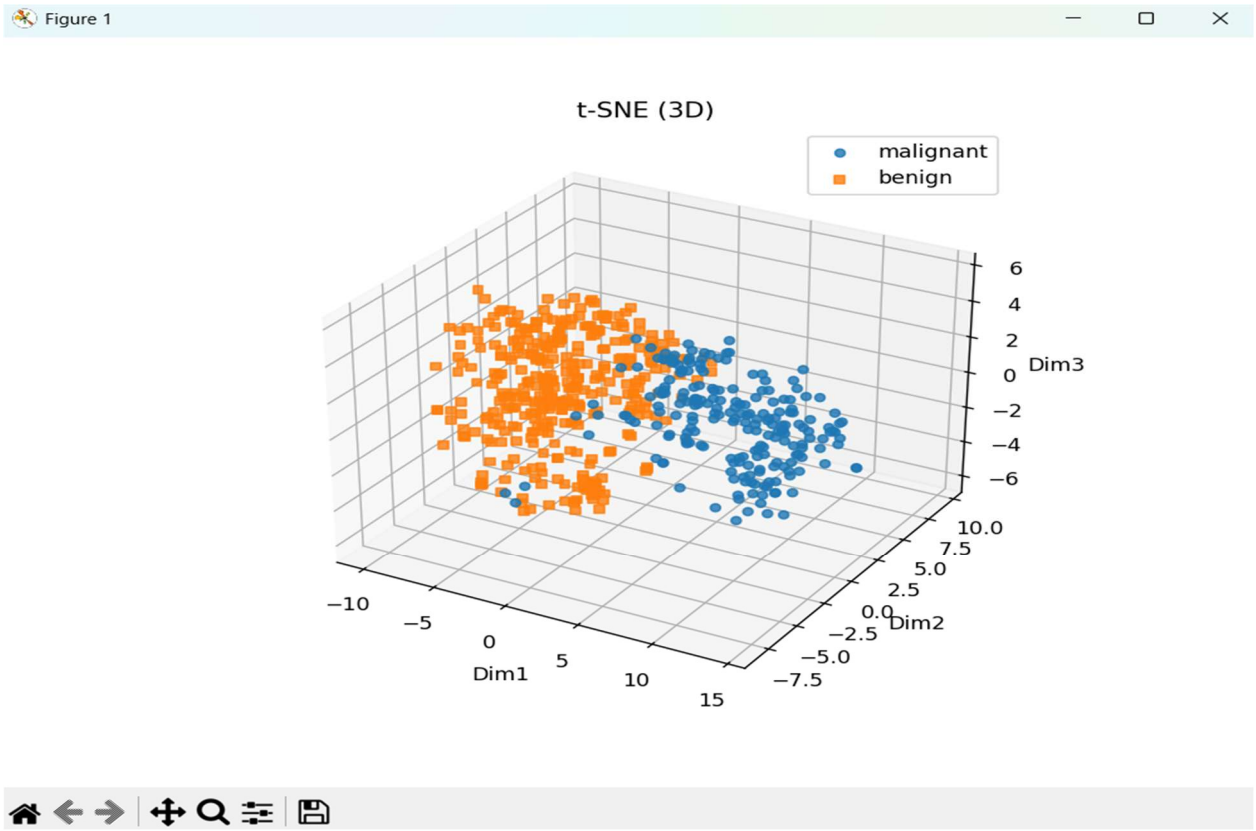


Figure 1
t-SNE (3D)
malignant
benign
Dim1
Dim2
Dim3

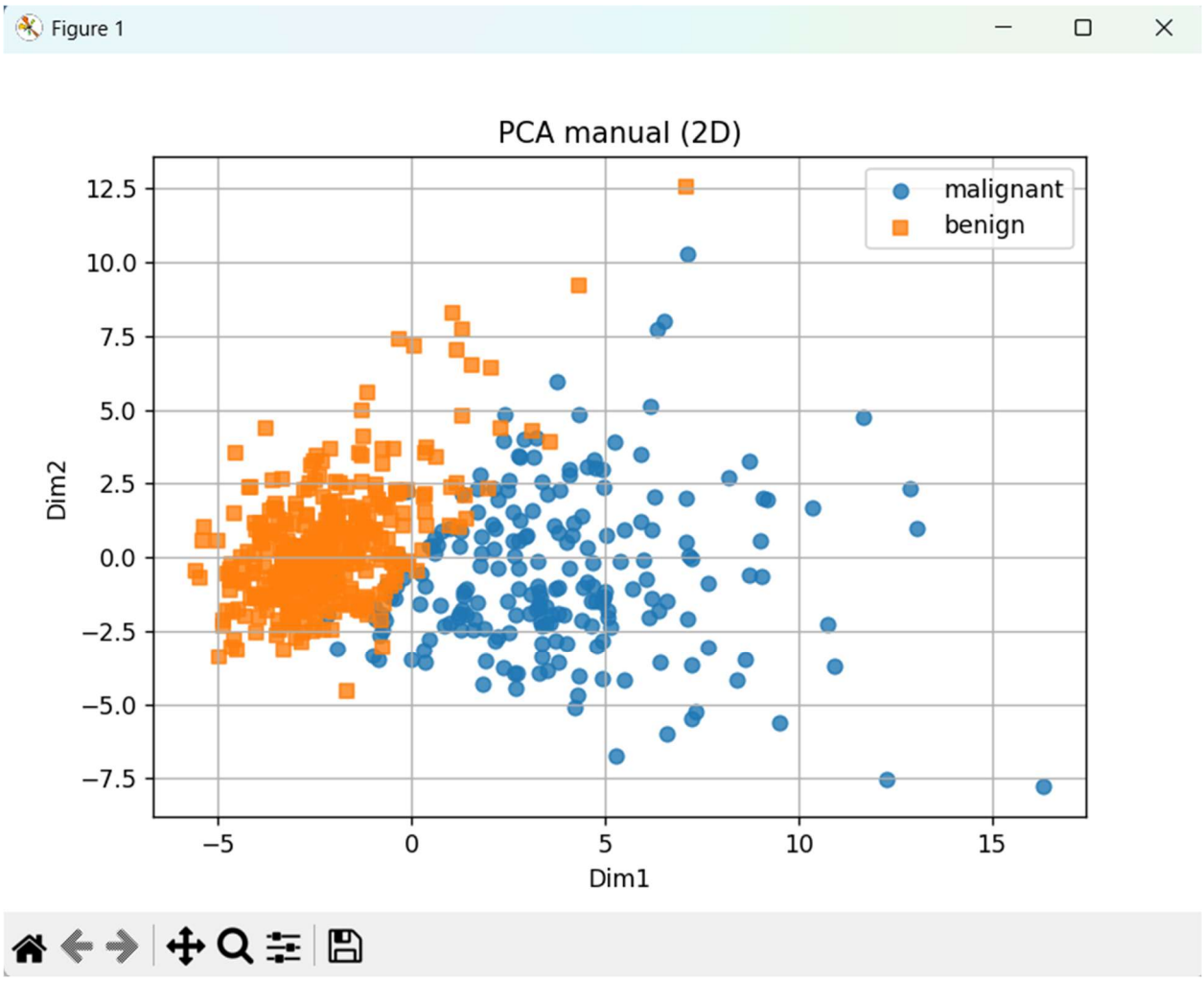


Figure 1
PCA manual (2D)
malignant
benign
Dim2
Dim1
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−5
0
5
10
15

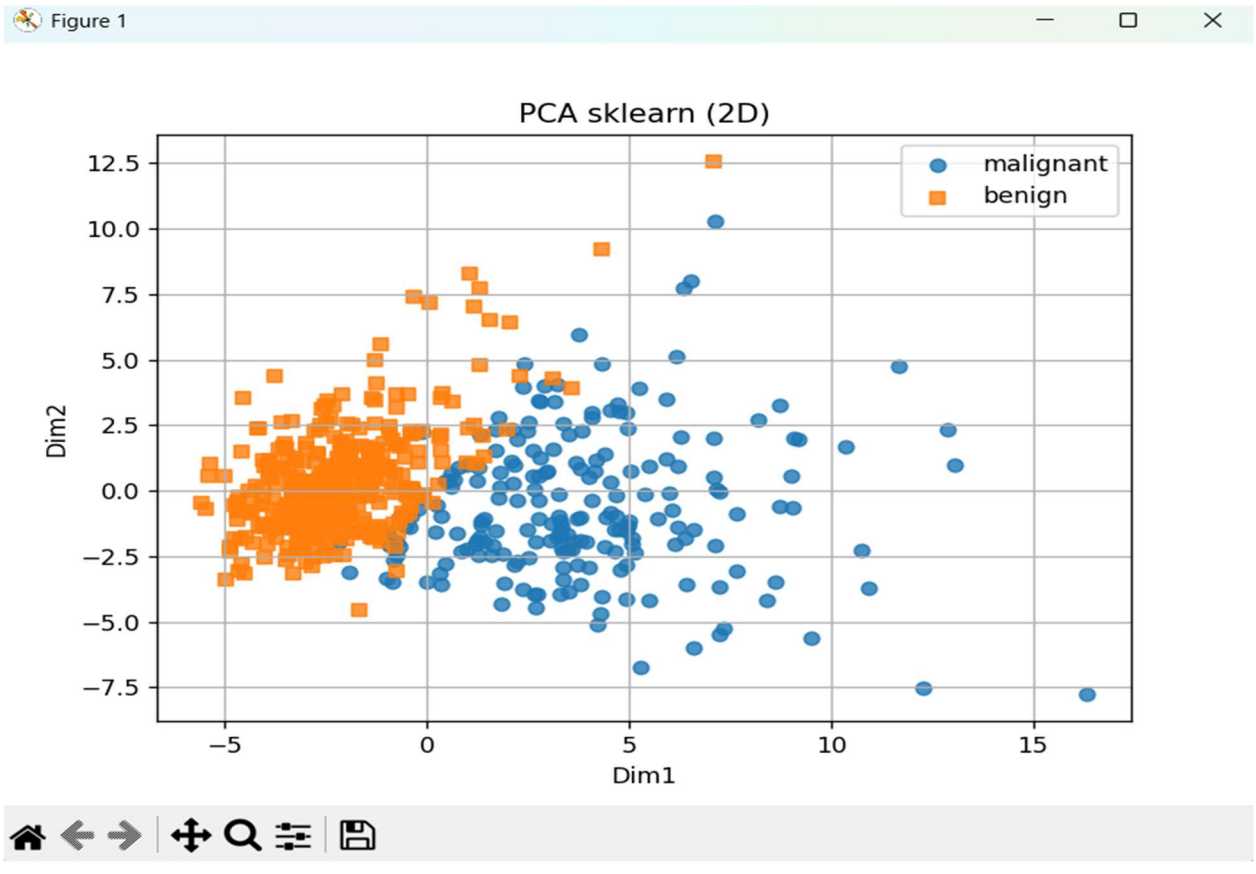


Figure 1
PCA sklearn (2D)
malignant
benign
Dim2
Dim1
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−5
0
5
10
15

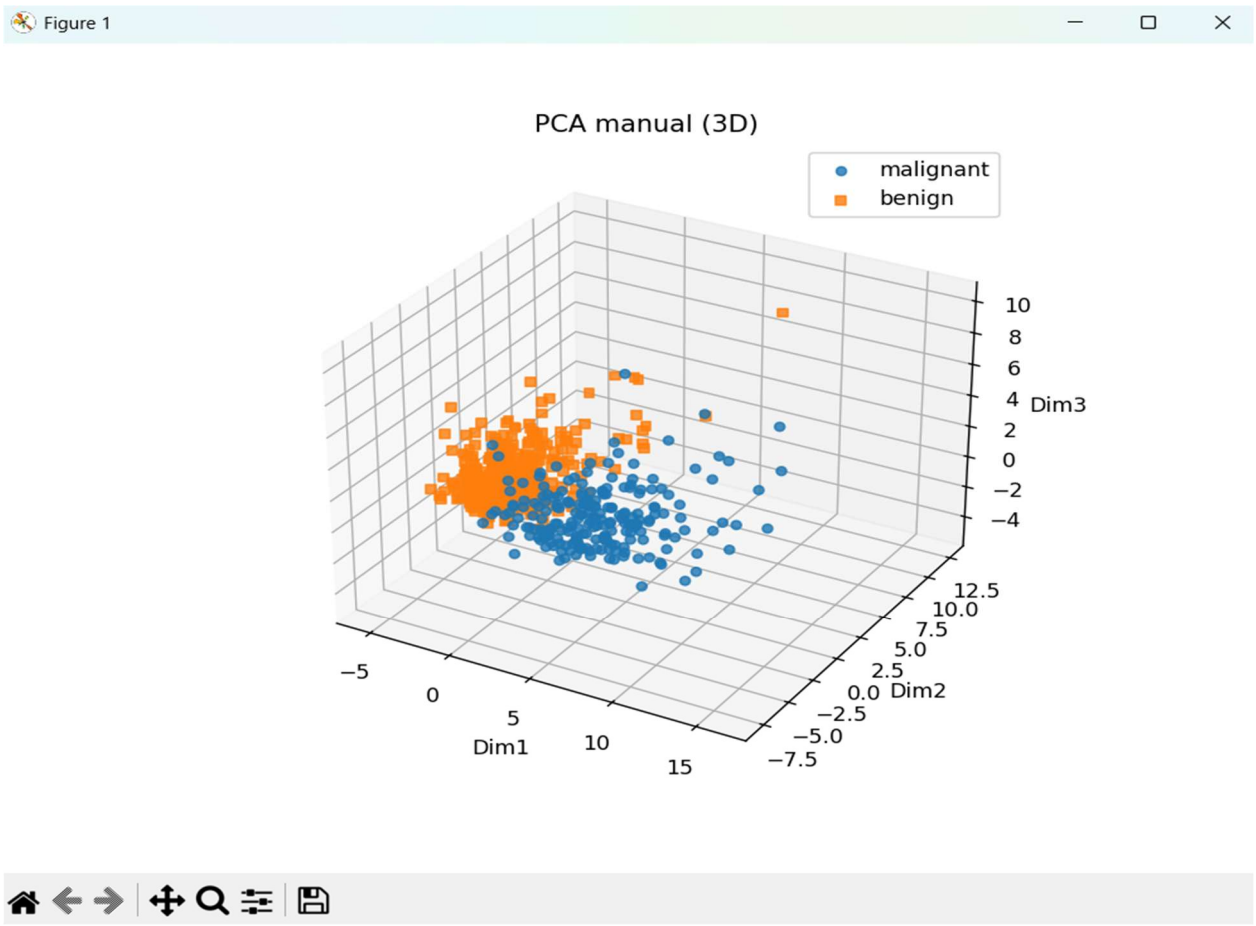


Figure 1
PCA manual (3D)
malignant
benign
Dim1
Dim2
Dim3

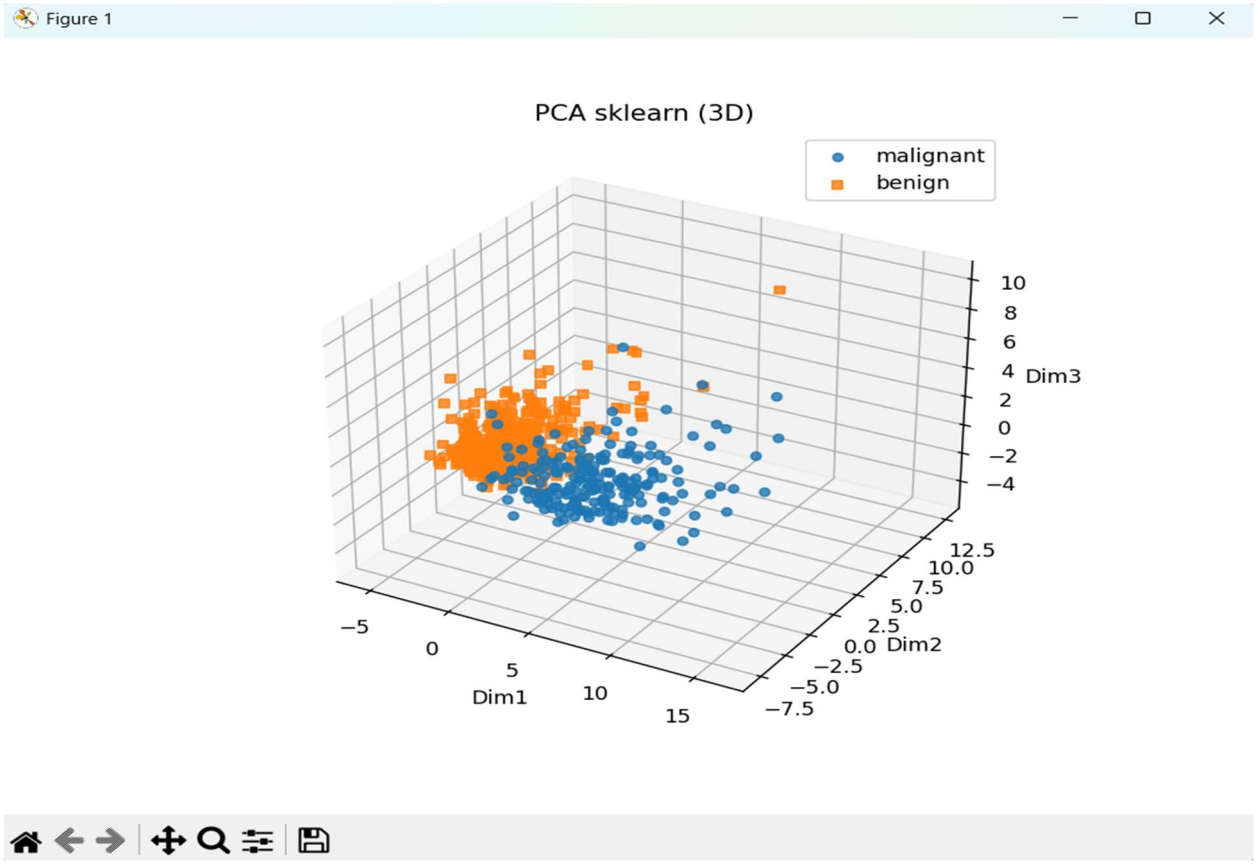


Figure 1
PCA sklearn (3D)
malignant
benign
Dim1
Dim2
Dim3

```
=== Сравнение представлений (в процентах расхождения = (1-|corr|)*100) ===
AE(2) vs PCA_manual(2): ['28.6083%', '46.2127%']
AE(3) vs PCA_manual(3): ['1.5544%', '18.8955%', '12.7622%']
PCA_manual vs PCA_sklearn (2D): ['0.000000%', '0.000000%']
PCA_manual vs PCA_sklearn (3D): ['0.000000%', '0.000000%', '0.000000%']

=== Сравнение объяснённой дисперсии (top-3), в %-пунктах ===
PC1: manual=0.442720, sklearn=0.442720, diff=0.000000 percentage points
PC2: manual=0.189712, sklearn=0.189712, diff=0.000000 percentage points
PC3: manual=0.093932, sklearn=0.093932, diff=0.000000 percentage points

=== MSE реконструкции автоэнкодеров (на стандартизованных признаках) ===
AE 2D MSE = 0.266967
AE 3D MSE = 0.210366
```

Вывод из результатов:

1) PCA (manual vs sklearn) — очень хорошее совпадение (малые различия), т.к. оба решают одну и ту же линейную задачу и стандартизация одинаковая.

2) Автоэнкодер (AE) обучается минимизировать MSE реконструкции через нелинейную сеть. Его латентные координаты

не обязаны совпадать с линейными главными компонентами: AE может захватывать нелинейные структуры и распределения.

3) t-SNE — метод нелинейной несохранённой глобальной структуры (он оптимизирует локальные соседства).

Он превосходен для визуализации кластеров, но не даёт линейно-интерпретируемых компонент.

4) Практически: если AE(2/3) показывает лучшее разделение классов на плоскости (визуально), это говорит о том, что

нелинейная компрессия полезна для отделения классов; если же PCA даёт столь же хорошее разделение — данные преимущественно линейно разделимы.

5) Различия в процентах (корреляции) между AE и PCA обычно существенны (> несколько процентов), это нормально.

Вывод: научился применять метод автоэнкодеров для визуализации данных.