Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

Кафедра интеллектуально-информационных технологий

Интеллектуальный анализ данных

Лабораторная работа №2

Автоэнкодеры

Выполнила:

студентка 4 курса

группы ИИ-24

Алешко А. В.

Проверила:

Андренко К. В.

Брест-2025

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

**Код программы(вариант 1):**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

from sklearn.datasets import load\_breast\_cancer

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn.manifold import TSNE

import torch

import torch.nn as nn

import torch.optim as optim

from torch.utils.data import DataLoader, TensorDataset

data = load\_breast\_cancer()

X = data.data

y = data.target

feature\_names = data.feature\_names

target\_names = data.target\_names

# Стандартизация данных

scaler = StandardScaler()

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

X\_tensor = torch.tensor(X\_scaled, dtype=torch.float32)

# Автоэнкодер

class Autoencoder(nn.Module):

def \_\_init\_\_(self, input\_dim, latent\_dim):

super(Autoencoder, self).\_\_init\_\_()

self.encoder = nn.Sequential(

nn.Linear(input\_dim, 128),

nn.ReLU(),

nn.Linear(128, latent\_dim)

)

self.decoder = nn.Sequential(

nn.Linear(latent\_dim, 128),

nn.ReLU(),

nn.Linear(128, input\_dim)

)

def forward(self, x):

encoded = self.encoder(x)

decoded = self.decoder(encoded)

return decoded

def encode(self, x):

return self.encoder(x)

def train\_autoencoder(model, data\_loader, epochs=50, lr=0.001):

criterion = nn.MSELoss()

optimizer = optim.Adam(model.parameters(), lr=lr)

for epoch in range(epochs):

model.train()

total\_loss = 0

for batch in data\_loader:

inputs = batch[0]

outputs = model(inputs)

loss = criterion(outputs, inputs)

optimizer.zero\_grad()

loss.backward()

optimizer.step()

total\_loss += loss.item()

print(f"Epoch {epoch+1}/{epochs}, Loss: {total\_loss / len(data\_loader):.4f}")

dataset = TensorDataset(X\_tensor)

data\_loader = DataLoader(dataset, batch\_size=32, shuffle=True)

ae\_2d = Autoencoder(input\_dim=30, latent\_dim=2)

train\_autoencoder(ae\_2d, data\_loader)

X\_ae\_2d = ae\_2d.encode(X\_tensor).detach().numpy()

ae\_3d = Autoencoder(input\_dim=30, latent\_dim=3)

train\_autoencoder(ae\_3d, data\_loader)

X\_ae\_3d = ae\_3d.encode(X\_tensor).detach().numpy()

def plot\_2d(X\_proj, y, title):

plt.figure(figsize=(8, 6))

scatter = plt.scatter(X\_proj[:, 0], X\_proj[:, 1], c=y, cmap='viridis', alpha=0.7)

plt.colorbar(scatter, ticks=[0, 1], format=plt.FuncFormatter(lambda val, loc: target\_names[int(val)]))

plt.title(title)

plt.xlabel('Component 1')

plt.ylabel('Component 2')

plt.show()

def plot\_3d(X\_proj, y, title):

fig = plt.figure(figsize=(10, 8))

ax = fig.add\_subplot(111, projection='3d')

scatter = ax.scatter(X\_proj[:, 0], X\_proj[:, 1], X\_proj[:, 2], c=y, cmap='viridis', alpha=0.7)

fig.colorbar(scatter, ticks=[0, 1], format=plt.FuncFormatter(lambda val, loc: target\_names[int(val)]))

ax.set\_title(title)

ax.set\_xlabel('Component 1')

ax.set\_ylabel('Component 2')

ax.set\_zlabel('Component 3')

plt.show()

plot\_2d(X\_ae\_2d, y, 'Autoencoder 2D Projection')

plot\_3d(X\_ae\_3d, y, 'Autoencoder 3D Projection')

# t-SNE

perplexities = [20, 30, 40, 50, 60]

best\_perplexity = 50

tsne\_2d = TSNE(n\_components=2, perplexity=best\_perplexity, init='pca', random\_state=42)

X\_tsne\_2d = tsne\_2d.fit\_transform(X\_scaled)

plot\_2d(X\_tsne\_2d, y, f't-SNE 2D (perplexity={best\_perplexity})')

tsne\_3d = TSNE(n\_components=3, perplexity=best\_perplexity, init='pca', random\_state=42)

X\_tsne\_3d = tsne\_3d.fit\_transform(X\_scaled)

plot\_3d(X\_tsne\_3d, y, f't-SNE 3D (perplexity={best\_perplexity})')

for perp in perplexities:

tsne\_temp = TSNE(n\_components=2, perplexity=perp, init='pca', random\_state=42)

X\_temp = tsne\_temp.fit\_transform(X\_scaled)

plot\_2d(X\_temp, y, f't-SNE 2D (perplexity={perp})')

# PCA

pca\_2d = PCA(n\_components=2)

X\_pca\_2d = pca\_2d.fit\_transform(X\_scaled)

print(f"Explained variance (2D PCA): {pca\_2d.explained\_variance\_ratio\_}")

plot\_2d(X\_pca\_2d, y, 'PCA 2D Projection')

pca\_3d = PCA(n\_components=3)

X\_pca\_3d = pca\_3d.fit\_transform(X\_scaled)

print(f"Explained variance (3D PCA): {pca\_3d.explained\_variance\_ratio\_}")

plot\_3d(X\_pca\_3d, y, 'PCA 3D Projection')

Результат работы программы:

Epoch 1/50, Loss: 0.9455

Epoch 2/50, Loss: 0.5980

Epoch 3/50, Loss: 0.4413

Epoch 4/50, Loss: 0.3751

Epoch 5/50, Loss: 0.3577

Epoch 6/50, Loss: 0.3432

Epoch 7/50, Loss: 0.3357

Epoch 8/50, Loss: 0.3293

Epoch 9/50, Loss: 0.3278

Epoch 10/50, Loss: 0.3259

Epoch 11/50, Loss: 0.3162

Epoch 12/50, Loss: 0.3151

Epoch 13/50, Loss: 0.3118

Epoch 14/50, Loss: 0.3104

Epoch 15/50, Loss: 0.3082

Epoch 16/50, Loss: 0.3043

Epoch 17/50, Loss: 0.3039

Epoch 18/50, Loss: 0.3048

Epoch 19/50, Loss: 0.3001

Epoch 20/50, Loss: 0.2994

Epoch 21/50, Loss: 0.2975

Epoch 22/50, Loss: 0.2973

Epoch 23/50, Loss: 0.2942

Epoch 24/50, Loss: 0.2949

Epoch 25/50, Loss: 0.2927

Epoch 26/50, Loss: 0.2909

Epoch 27/50, Loss: 0.2896

Epoch 28/50, Loss: 0.2887

Epoch 29/50, Loss: 0.2883

Epoch 30/50, Loss: 0.2871

Epoch 31/50, Loss: 0.2866

Epoch 32/50, Loss: 0.2860

Epoch 33/50, Loss: 0.2852

Epoch 34/50, Loss: 0.2826

Epoch 35/50, Loss: 0.2834

Epoch 36/50, Loss: 0.2829

Epoch 37/50, Loss: 0.2825

Epoch 38/50, Loss: 0.2808

Epoch 39/50, Loss: 0.2814

Epoch 40/50, Loss: 0.2808

Epoch 41/50, Loss: 0.2791

Epoch 42/50, Loss: 0.2784

Epoch 43/50, Loss: 0.2781

Epoch 44/50, Loss: 0.2783

Epoch 45/50, Loss: 0.2775

Epoch 46/50, Loss: 0.2779

Epoch 47/50, Loss: 0.2752

Epoch 48/50, Loss: 0.2761

Epoch 49/50, Loss: 0.2753

Epoch 50/50, Loss: 0.2739

Epoch 1/50, Loss: 0.8834

Epoch 2/50, Loss: 0.5733

Epoch 3/50, Loss: 0.4445

Epoch 4/50, Loss: 0.3347

Epoch 5/50, Loss: 0.3004

Epoch 6/50, Loss: 0.2824

Epoch 7/50, Loss: 0.2745

Epoch 8/50, Loss: 0.2658

Epoch 9/50, Loss: 0.2555

Epoch 10/50, Loss: 0.2504

Epoch 11/50, Loss: 0.2451

Epoch 12/50, Loss: 0.2381

Epoch 13/50, Loss: 0.2319

Epoch 14/50, Loss: 0.2261

Epoch 15/50, Loss: 0.2224

Epoch 16/50, Loss: 0.2203

Epoch 17/50, Loss: 0.2181

Epoch 18/50, Loss: 0.2149

Epoch 19/50, Loss: 0.2125

Epoch 20/50, Loss: 0.2122

Epoch 21/50, Loss: 0.2099

Epoch 22/50, Loss: 0.2102

Epoch 23/50, Loss: 0.2086

Epoch 24/50, Loss: 0.2085

Epoch 25/50, Loss: 0.2065

Epoch 26/50, Loss: 0.2053

Epoch 27/50, Loss: 0.2051

Epoch 28/50, Loss: 0.2042

Epoch 29/50, Loss: 0.2037

Epoch 30/50, Loss: 0.2022

Epoch 31/50, Loss: 0.2014

Epoch 32/50, Loss: 0.2013

Epoch 33/50, Loss: 0.2011

Epoch 34/50, Loss: 0.2000

Epoch 35/50, Loss: 0.2009

Epoch 36/50, Loss: 0.2019

Epoch 37/50, Loss: 0.1986

Epoch 38/50, Loss: 0.1987

Epoch 39/50, Loss: 0.1973

Epoch 40/50, Loss: 0.1969

Epoch 41/50, Loss: 0.1975

Epoch 42/50, Loss: 0.1960

Epoch 43/50, Loss: 0.1948

Epoch 44/50, Loss: 0.1944

Epoch 45/50, Loss: 0.1937

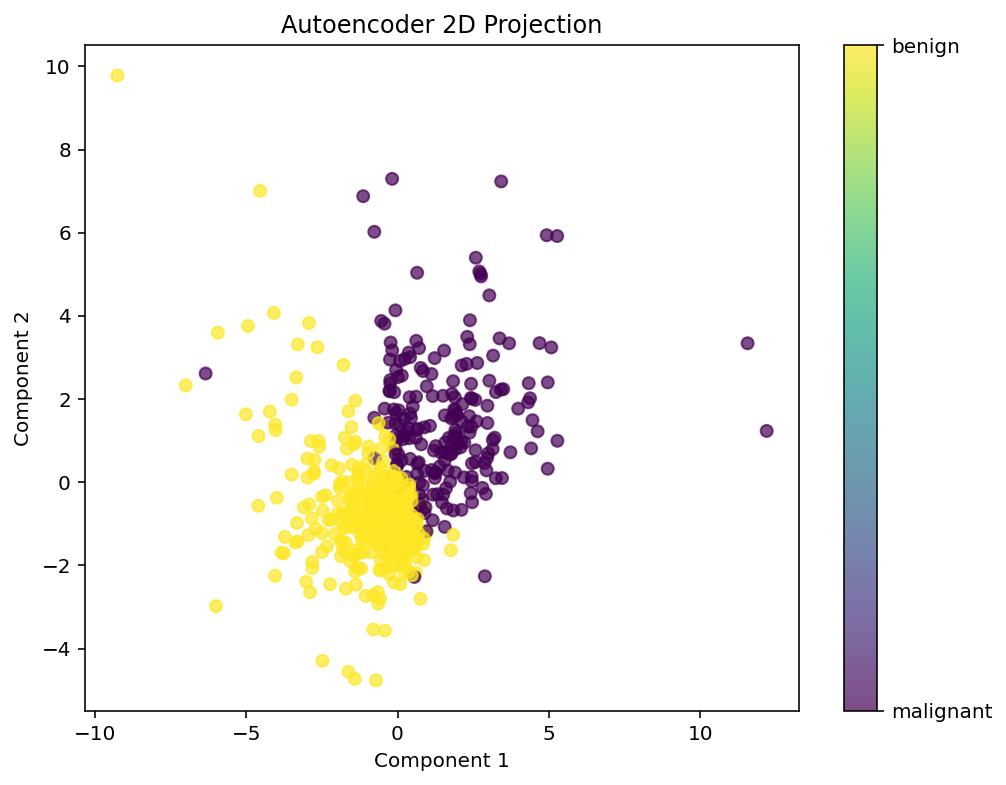
Epoch 46/50, Loss: 0.1929

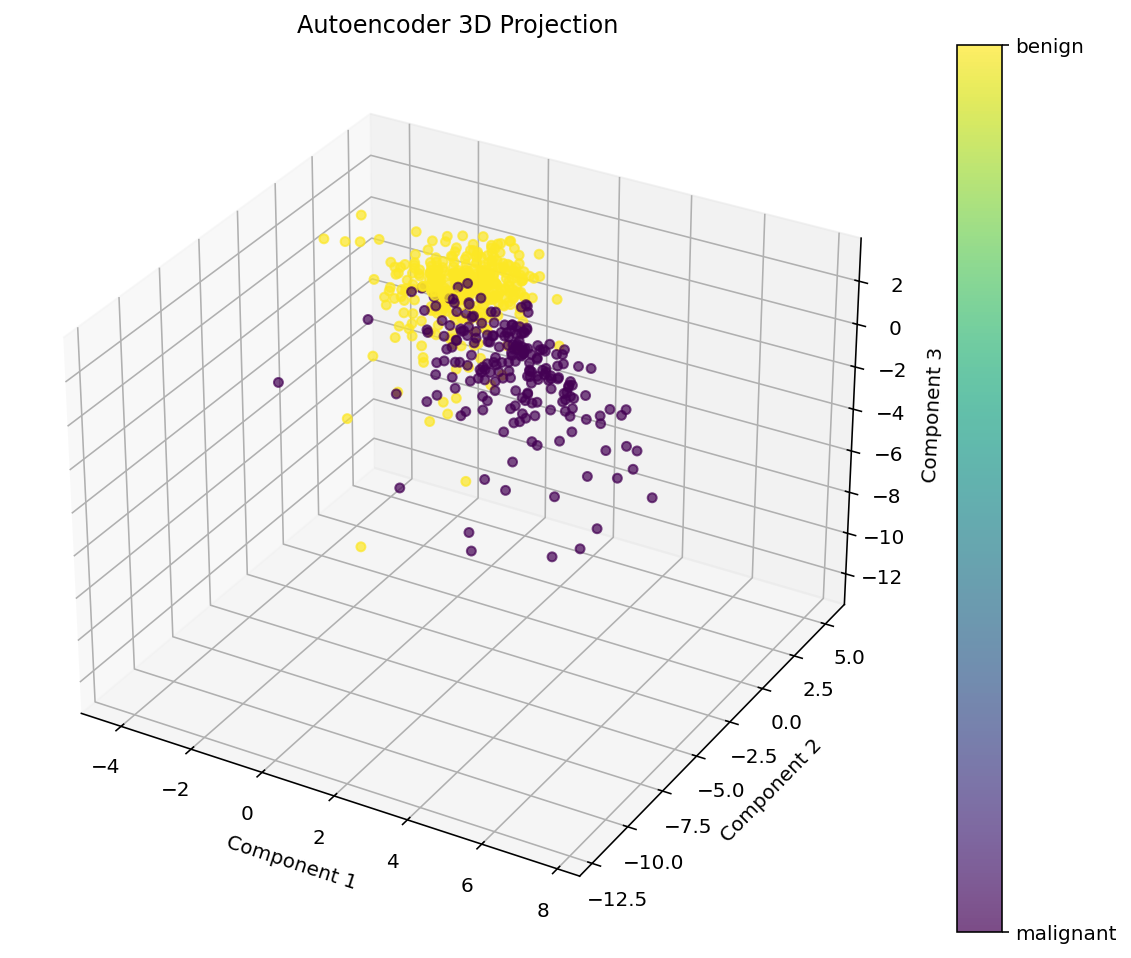
Epoch 47/50, Loss: 0.1933

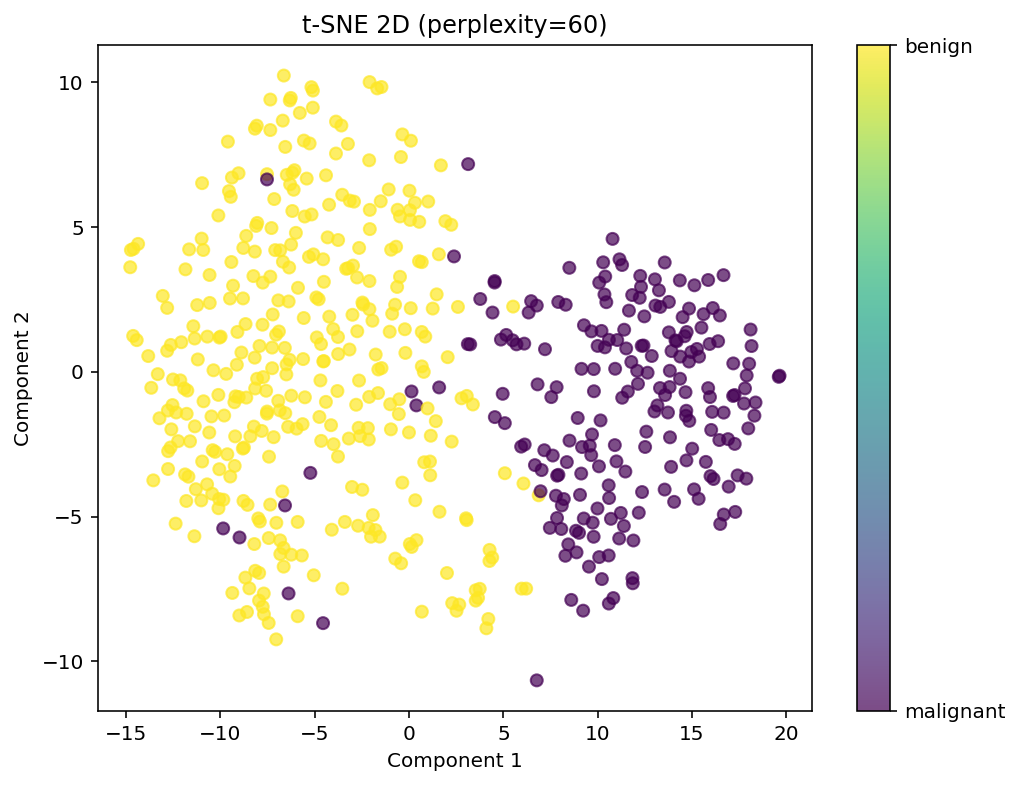
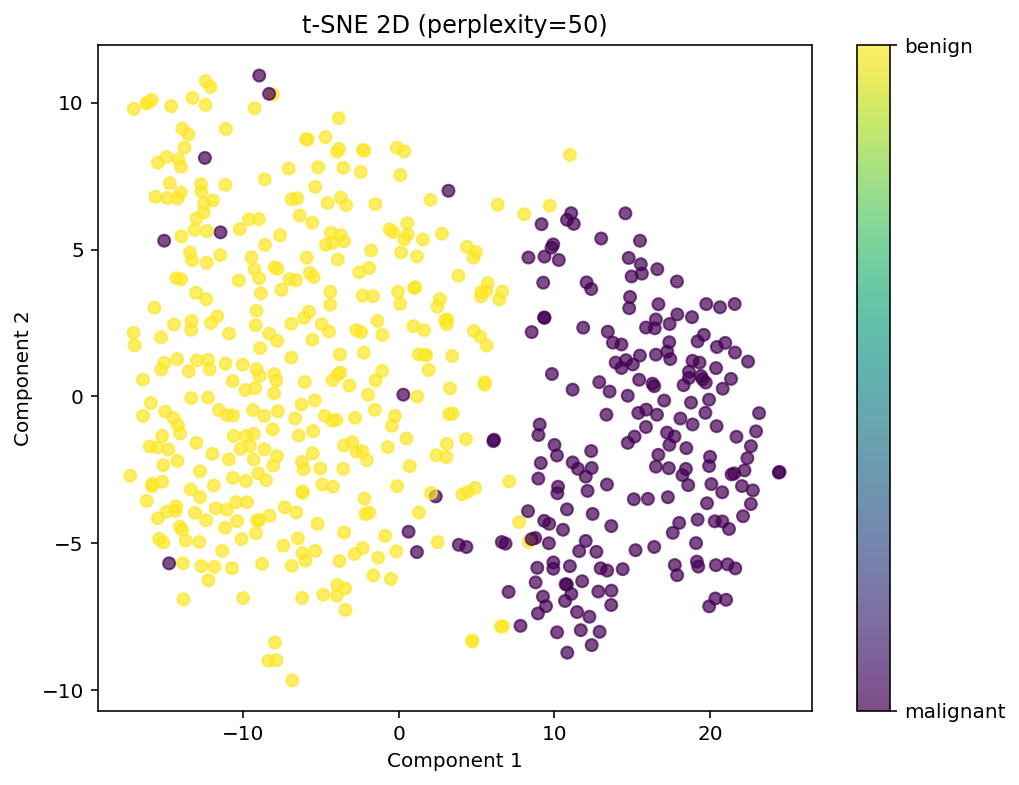
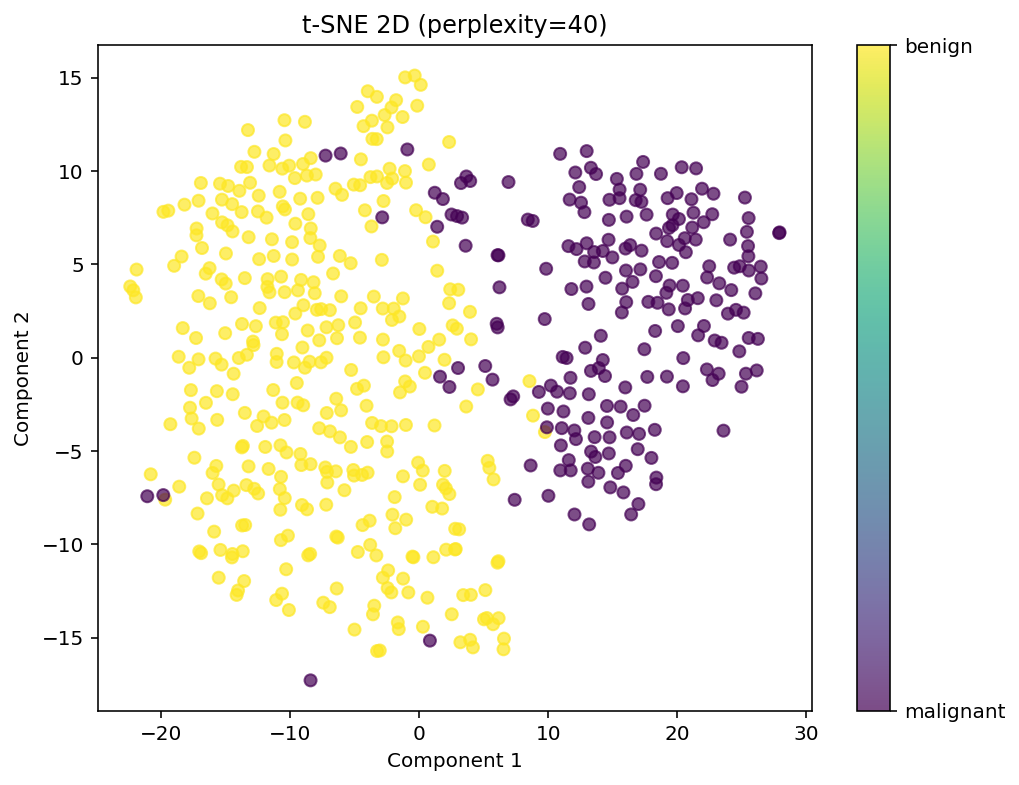
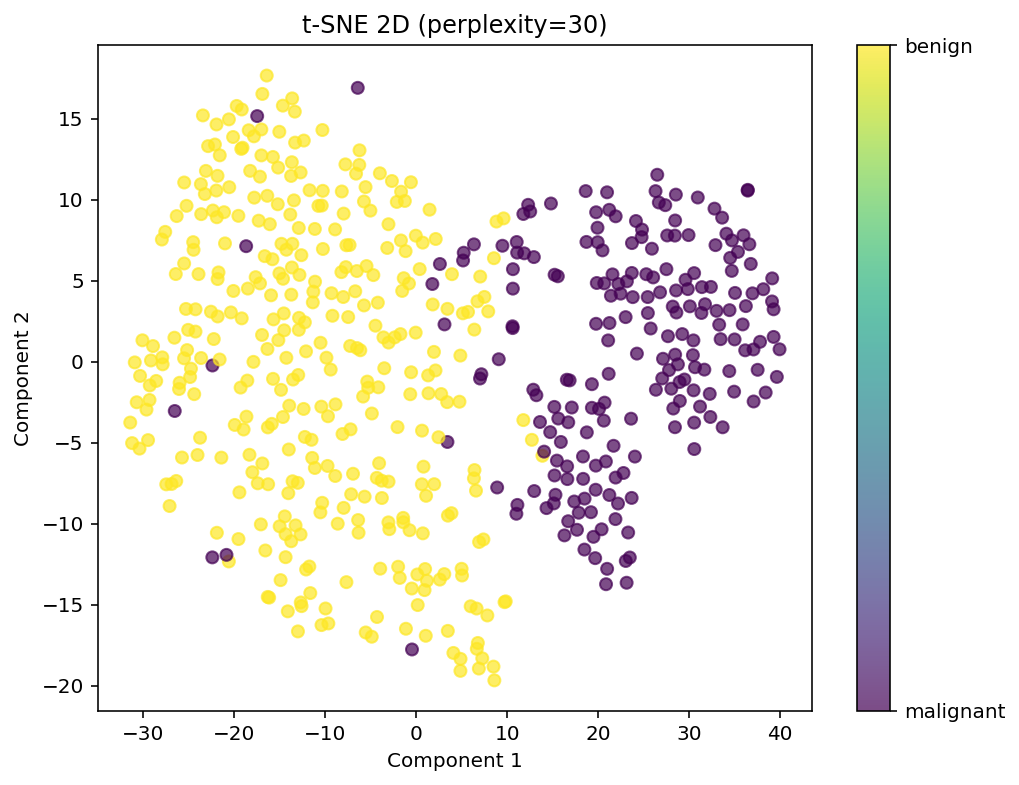
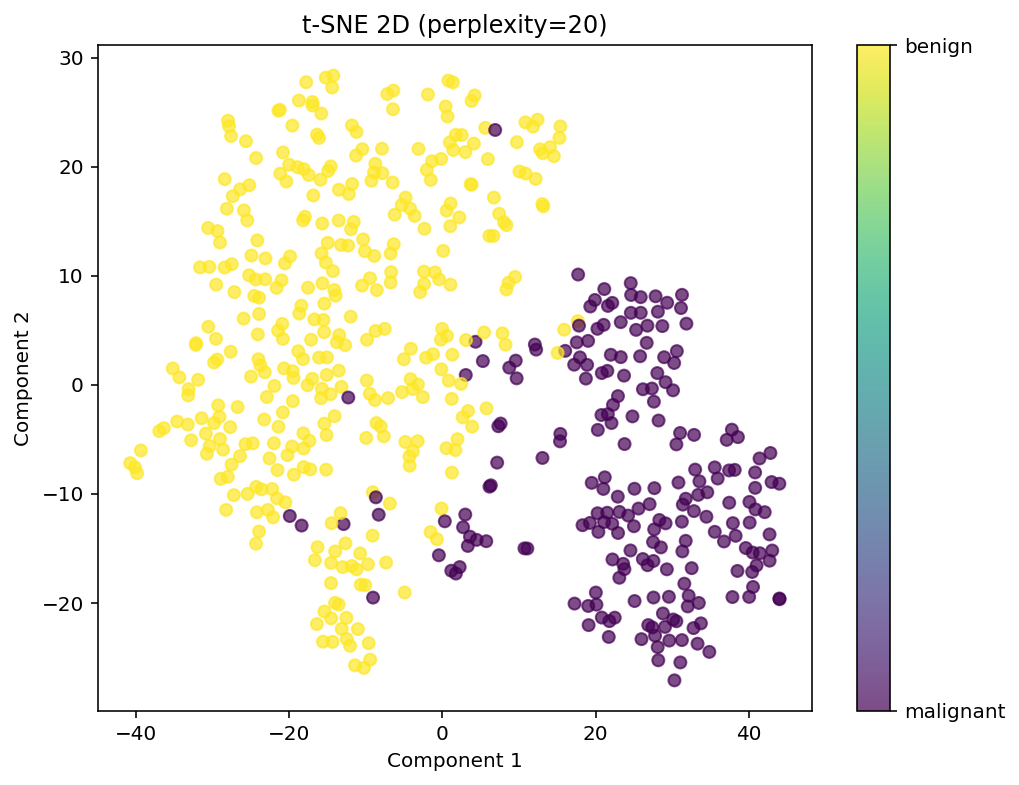
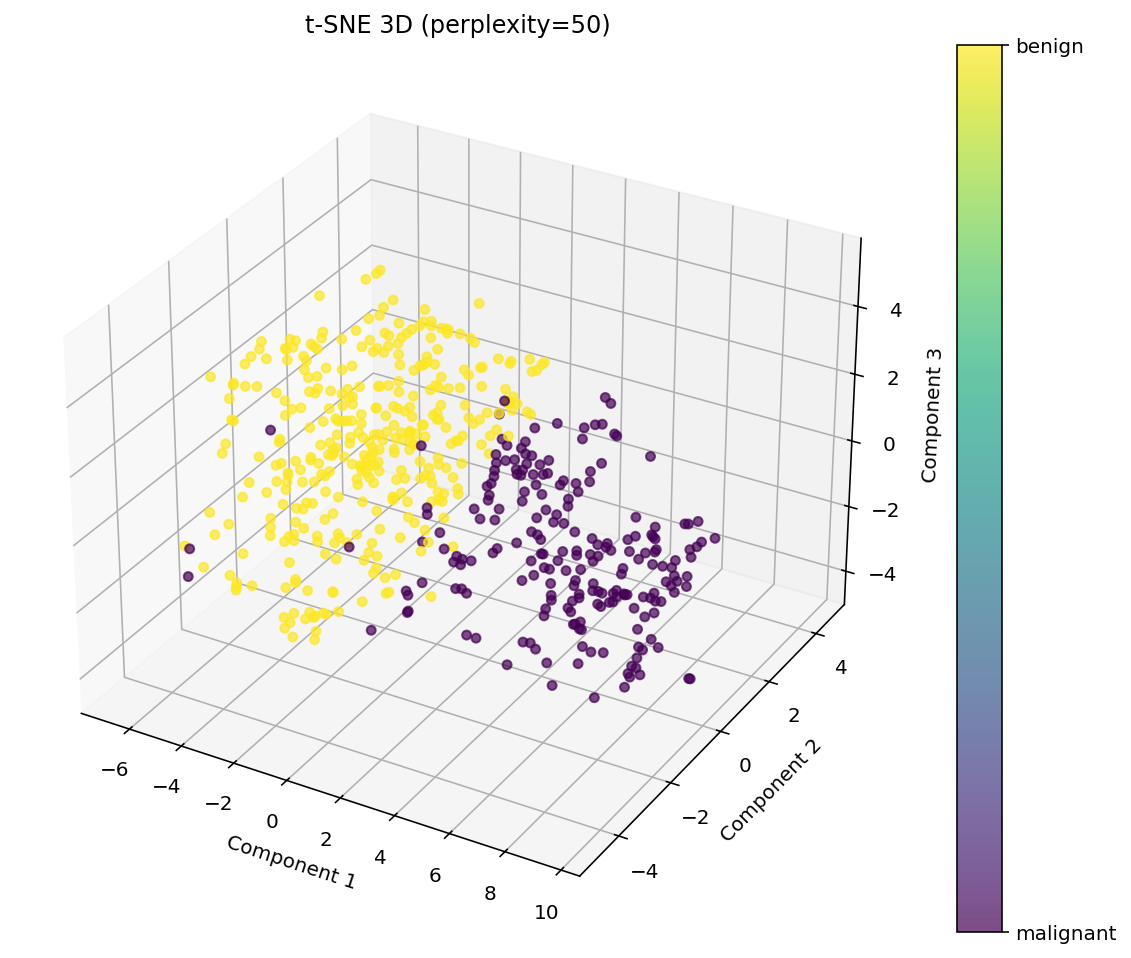
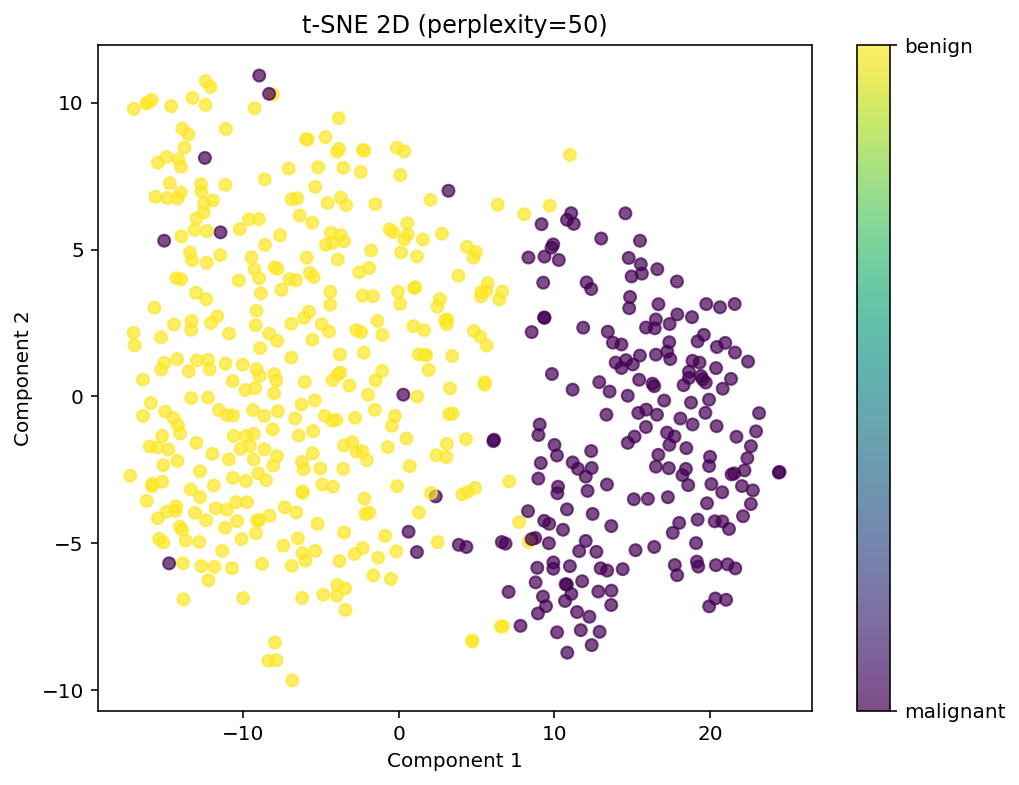
Epoch 48/50, Loss: 0.1925

Epoch 49/50, Loss: 0.1932

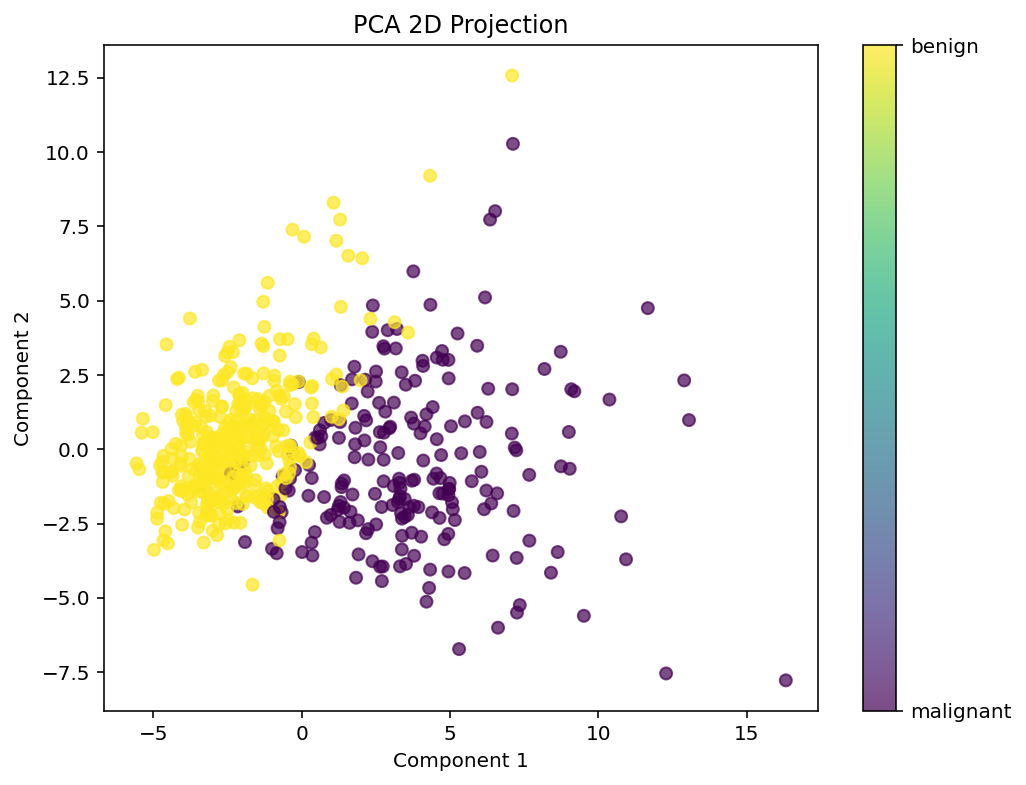
Epoch 50/50, Loss: 0.1923



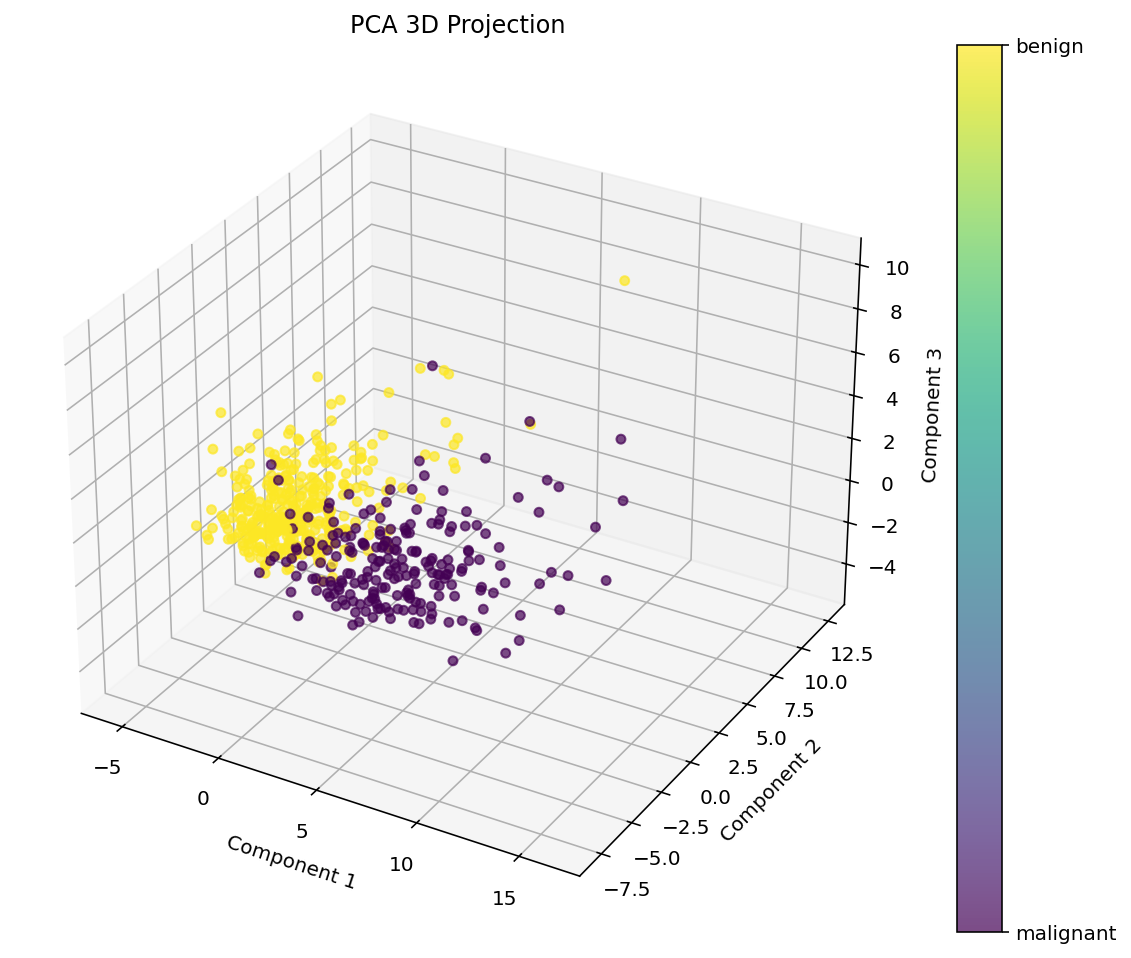




Explained variance (2D PCA): [0.44272026 0.18971182]



Explained variance (3D PCA): [0.44272026 0.18971182 0.09393163]



- PCA: Линейный метод, объясняет ~63% variance для 2D и ~73% для 3D. Хорошая базовая сепарация, но с перекрытиями.

- Autoencoder: Нелинейный, захватывает сложные зависимости. Лучшая сепарация по сравнению с PCA, меньше перекрытий.

- t-SNE: Лучшая визуализация кластеров, особенно при perplexity=50-60. Четкие группы, минимальные перекрытия, полезен для выявления аномалий.

Все методы показывают разделимость классов, но нелинейные (AE, t-SNE) предпочтительны для сложных данных.

} **Вывод:** научилась применять автоэнкодеры для осуществления визуализации данных и их анализа.