Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине «Интеллектуальный анализ данных»

Тема: «Автоэнкодеры»

Выполнил:

Студент 4 курса

Группы ИИ-23

Вышинский А. С.

Проверила:

Андренко К. В.

Брест 2025

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

Общее задание

Общее задание

1. Используя выборку по варианту, осуществить проецирование данных на плоскость первых двух и трех главных компонент с использованием нейросетевой модели автоэнкодера (с двумя и тремя нейронами в среднем слое);

2. Выполнить визуализацию полученных главных компонент с использованием средств библиотеки matplotlib, обозначая экземпляры разных классов с использованием разных цветовых маркеров;

3. Реализовать метод t-SNE для визуализации данных (использовать также 2 и 3 компонента), построить соответствующую визуализацию;

4. Применить к данным метод PCA (2 и 3 компонента), реализованный в ЛР №1, сделать выводы;

5. Оформить отчет по выполненной работе, загрузить исходный код и отчет в соответствующий репозиторий на github.

Задание по вариантам

№ в-а	Выборка	Класс
3	Rice (Cammeo and Osmancik)	Class

Код:

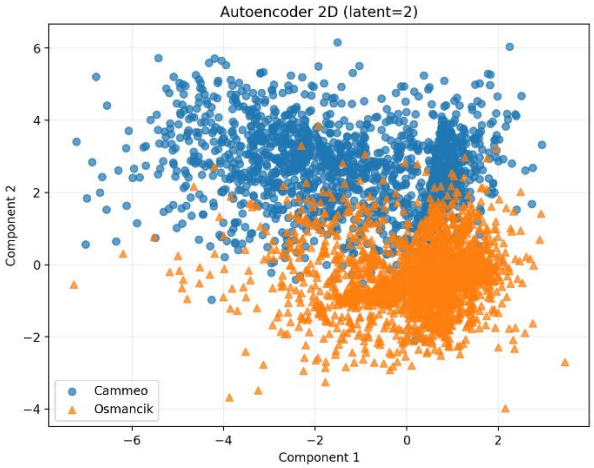
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
from ucimlrepo import fetch_ucirepo


SAVE_FIGS = True
FIG_PATH = "./figs/"


rice_data = fetch_ucirepo(id=545)
X = rice_data.data.features
```

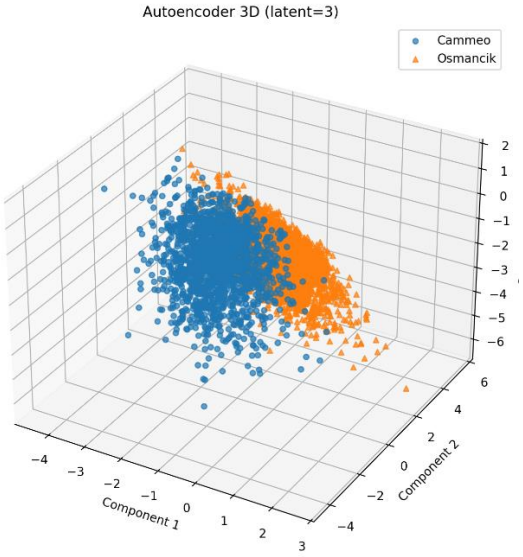
```
y = rice_data.data.targets.values  # shape (n_samples, 1)


le = LabelEncoder()
y_encoded = le.fit_transform(y.ravel())  # ravel(): (n,1) -> (n,)
target_names = le.classes_.tolist()  # ['Cammeo', 'Osmancik']


scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_tensor = torch.tensor(X_scaled, dtype=torch.float32)


class Autoencoder(nn.Module):
    def __init__(self, input_dim, latent_dim):
        super(Autoencoder, self).__init__()
        hidden_dim = max(32, input_dim // 2)
        self.encoder = nn.Sequential(
            nn.Linear(input_dim, hidden_dim),
            nn.ReLU(),
            nn.Linear(hidden_dim, latent_dim)
        )
        self.decoder = nn.Sequential(
            nn.Linear(latent_dim, hidden_dim),
            nn.ReLU(),
            nn.Linear(hidden_dim, 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,
device="cpu"):
    model.to(device)
    criterion = nn.MSELoss()
    optimizer = optim.Adam(model.parameters(), lr=lr)
    for epoch in range(epochs):
        model.train()
        total_loss = 0.0
        for batch in data_loader:
            inputs = batch[0].to(device)
            outputs = model(inputs)
            loss = criterion(outputs, inputs)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
        avg = total_loss / len(data_loader)
        print(f"Epoch {epoch + 1}/{epochs}, Loss: {avg:.6f}")
    model.to("cpu")
    return model

dataset = TensorDataset(X_tensor)
data_loader = DataLoader(dataset, batch_size=32, shuffle=True)
```

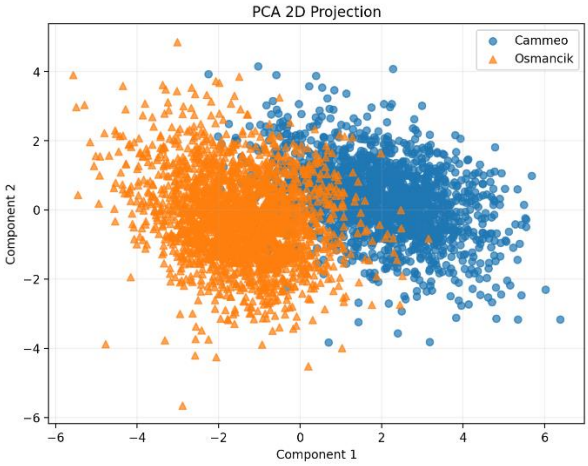
```
ae_2d = Autoencoder(input_dim=X.shape[1], latent_dim=2)
train_autoencoder(ae_2d, data_loader, epochs=50, lr=1e-3)
X_ae_2d = ae_2d.encode(X_tensor).detach().numpy()

ae_3d = Autoencoder(input_dim=X.shape[1], latent_dim=3)
train_autoencoder(ae_3d, data_loader, epochs=50, lr=1e-3)
X_ae_3d = ae_3d.encode(X_tensor).detach().numpy()


def plot_2d_markers(X_proj, y, title, save_as=None):
    plt.figure(figsize=(8,6))
    markers = ['o', '^', 's', 'P', 'X', 'D']  # на случай >2 классов
    unique_labels = np.unique(y)
    for i, lab in enumerate(unique_labels):
        idx = (y == lab)
        plt.scatter(X_proj[idx,0], X_proj[idx,1],
                    label=str(target_names[int(lab)]),
                    marker=markers[i % len(markers)],
                    alpha=0.7)
    plt.title(title)
    plt.xlabel('Component 1')
    plt.ylabel('Component 2')
    plt.legend()
    plt.grid(alpha=0.2)
    if save_as and SAVE_FIGS:
        plt.savefig(save_as, dpi=200, bbox_inches='tight')
    plt.show()

def plot_3d_markers(X_proj, y, title, save_as=None):
    fig = plt.figure(figsize=(10,8))
    ax = fig.add_subplot(111, projection='3d')
    markers = ['o', '^', 's', 'P', 'X', 'D']
    unique_labels = np.unique(y)
    for i, lab in enumerate(unique_labels):
        idx = (y == lab)
        ax.scatter(X_proj[idx,0], X_proj[idx,1], X_proj[idx,2],
                   label=str(target_names[int(lab)]),
                   marker=markers[i % len(markers)],
                   alpha=0.7)
    ax.set_title(title)
    ax.set_xlabel('Component 1')
    ax.set_ylabel('Component 2')
    ax.set_zlabel('Component 3')
    ax.legend()
    if save_as and SAVE_FIGS:
        plt.savefig(save_as, dpi=200, bbox_inches='tight')
    plt.show()


plot_2d_markers(X_ae_2d, y_encoded, 'Autoencoder 2D (latent=2)',
save_as=FIG_PATH+"ae_2d.png")
plot_3d_markers(X_ae_3d, y_encoded, 'Autoencoder 3D (latent=3)',
save_as=FIG_PATH+"ae_3d.png")


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)
```

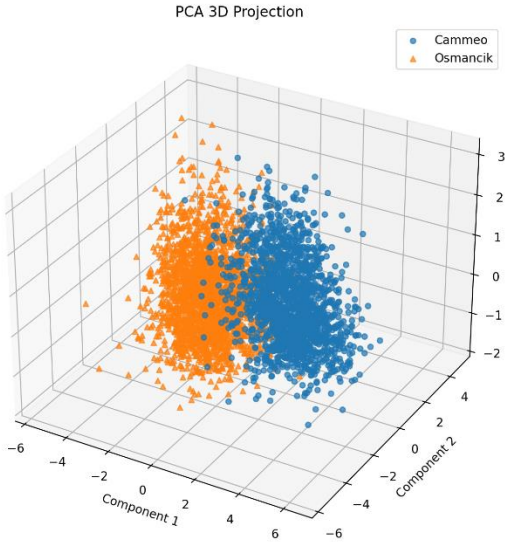
```
plot_2d_markers(X_tsne_2d, y_encoded, f"t-SNE 2D
(perplexity={best_perplexity})",
save_as=FIG_PATH+f"tsne2d_p{best_perplexity}.png")

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_markers(X_tsne_3d, y_encoded, f"t-SNE 3D
(perplexity={best_perplexity})",
save_as=FIG_PATH+f"tsne3d_p{best_perplexity}.png")

for perp in perplexities:
    X_temp = TSNE(n_components=2, perplexity=perp, init='pca',
random_state=42).fit_transform(X_scaled)
    plot_2d_markers(X_temp, y_encoded, f"t-SNE 2D (perplexity={perp})",
save_as=FIG_PATH+f"tsne2d_p{perp}.png")


pca_2d = PCA(n_components=2)
X_pca_2d = pca_2d.fit_transform(X_scaled)
print("Explained variance ratio (2D PCA):",
pca_2d.explained_variance_ratio_)
plot_2d_markers(X_pca_2d, y_encoded, "PCA 2D Projection",
save_as=FIG_PATH+"pca2d.png")

pca_3d = PCA(n_components=3)
X_pca_3d = pca_3d.fit_transform(X_scaled)
print("Explained variance ratio (3D PCA):",
pca_3d.explained_variance_ratio_)
plot_3d_markers(X_pca_3d, y_encoded, "PCA 3D Projection",
save_as=FIG_PATH+"pca3d.png")
```

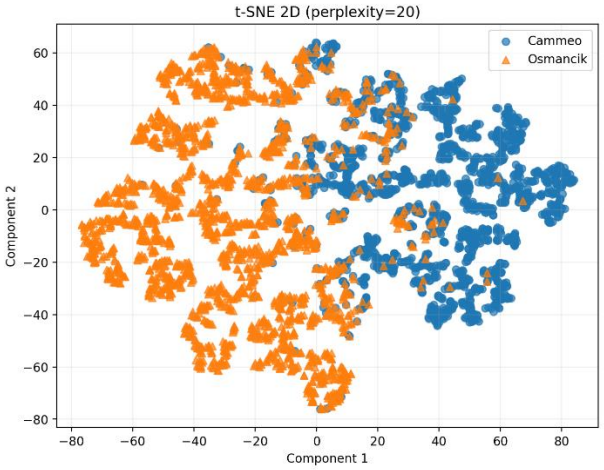
Вывод:

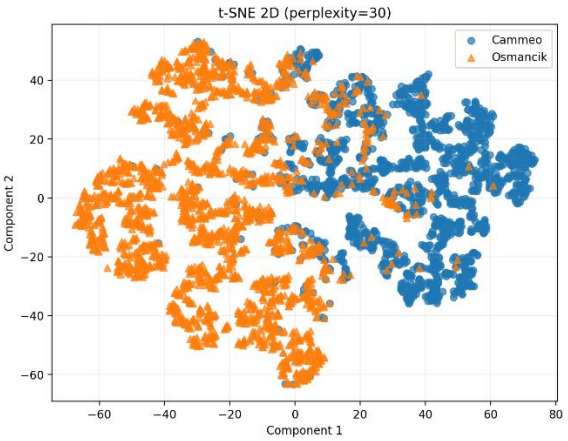

Autoencoder 2D (latent=2)
Cammeo
Osmancik
Component 1
Component 2

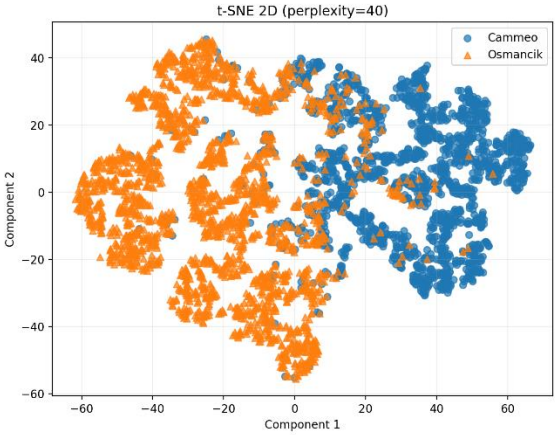

Autoencoder 3D (latent=3)
Cammeo
Osmancik
Component 1
Component 2

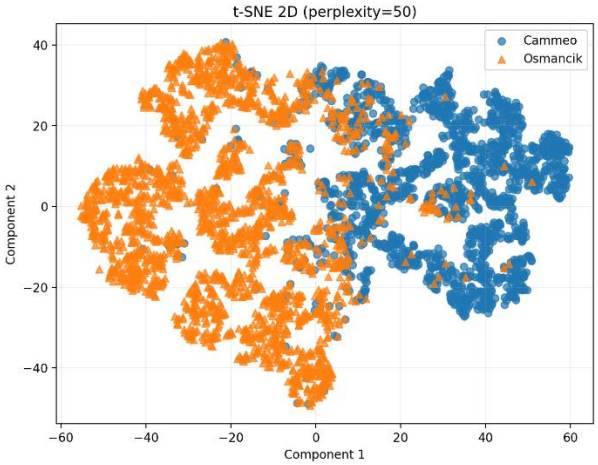

PCA 2D Projection
Cammeo
Osmancik
Component 1
Component 2

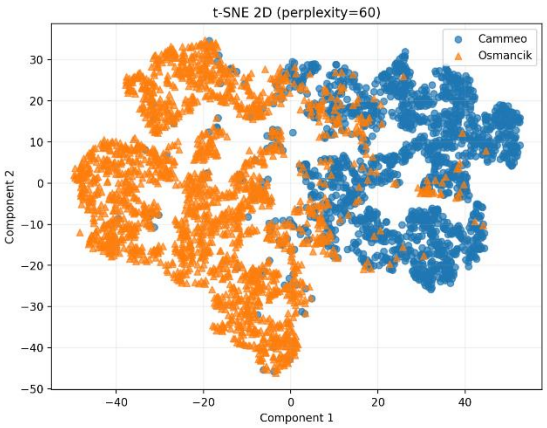

PCA 3D Projection
Cammeo
Osmancik
Component 1
Component 2

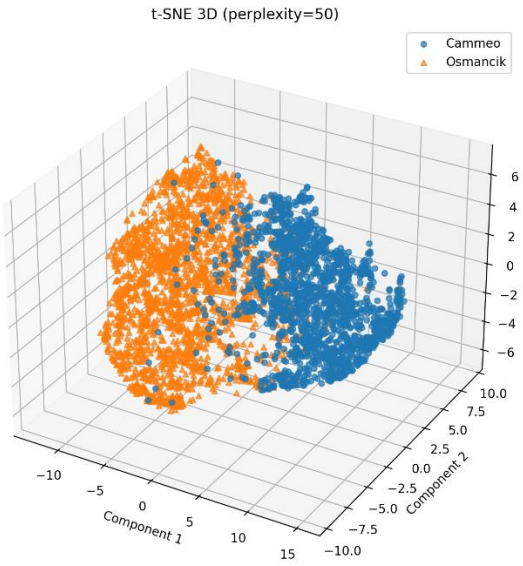

t-SNE 2D (perplexity=20)
Cammeo
Osmancik
Component 1
Component 2


t-SNE 2D (perplexity=30)
Cammeo
Osmancik
Component 1
Component 2









C:\Users\arcio\PycharmProjects\IAD\laba2\.venv\Scripts\python.exe
C:\Users\arcio\PycharmProjects\IAD\laba2\main.py

Epoch 1/50, Loss: 0.686580

Epoch 2/50, Loss: 0.166787

Epoch 3/50, Loss: 0.135932

Epoch 4/50, Loss: 0.130325

Epoch 5/50, Loss: 0.127600

Epoch 6/50, Loss: 0.125266

Epoch 7/50, Loss: 0.122868

Epoch 8/50, Loss: 0.121105

Epoch 9/50, Loss: 0.120844

Epoch 10/50, Loss: 0.117529

Epoch 11/50, Loss: 0.116487

Epoch 12/50, Loss: 0.115020

Epoch 13/50, Loss: 0.113622

Epoch 14/50, Loss: 0.112064

Epoch 15/50, Loss: 0.111730

Epoch 16/50, Loss: 0.111891

Epoch 17/50, Loss: 0.109662

Epoch 18/50, Loss: 0.109796

Epoch 19/50, Loss: 0.108009

Epoch 20/50, Loss: 0.106856

Epoch 21/50, Loss: 0.106087

Epoch 22/50, Loss: 0.104694

Epoch 23/50, Loss: 0.104572

Epoch 24/50, Loss: 0.105191

Epoch 25/50, Loss: 0.105422

Epoch 26/50, Loss: 0.102887

Epoch 27/50, Loss: 0.102331

Epoch 28/50, Loss: 0.101433

Epoch 29/50, Loss: 0.099541

Epoch 30/50, Loss: 0.099433

Epoch 31/50, Loss: 0.098639

Epoch 32/50, Loss: 0.098377

Epoch 33/50, Loss: 0.097235

Epoch 34/50, Loss: 0.097048

Epoch 35/50, Loss: 0.096293

Epoch 36/50, Loss: 0.095677

Epoch 37/50, Loss: 0.095429

Epoch 38/50, Loss: 0.095098

Epoch 39/50, Loss: 0.094935

Epoch 40/50, Loss: 0.095087

Epoch 41/50, Loss: 0.094076

Epoch 42/50, Loss: 0.093395

Epoch 43/50, Loss: 0.094697

Epoch 44/50, Loss: 0.092995

Epoch 45/50, Loss: 0.091786

Epoch 46/50, Loss: 0.091199

Epoch 47/50, Loss: 0.093272

Epoch 48/50, Loss: 0.092646

Epoch 49/50, Loss: 0.090923

Epoch 50/50, Loss: 0.090961

Epoch 1/50, Loss: 0.622595

Epoch 2/50, Loss: 0.114343

Epoch 3/50, Loss: 0.019431

Epoch 4/50, Loss: 0.008998

Epoch 5/50, Loss: 0.006389

Epoch 6/50, Loss: 0.004850

Epoch 7/50, Loss: 0.003968

Epoch 8/50, Loss: 0.003448

Epoch 9/50, Loss: 0.003064

Epoch 10/50, Loss: 0.002822

Epoch 11/50, Loss: 0.002618

Epoch 12/50, Loss: 0.002511

Epoch 13/50, Loss: 0.002358

Epoch 14/50, Loss: 0.002265

Epoch 15/50, Loss: 0.002148

Epoch 16/50, Loss: 0.002084

Epoch 17/50, Loss: 0.002008

Epoch 18/50, Loss: 0.001951

Epoch 19/50, Loss: 0.001939

Epoch 20/50, Loss: 0.001919

Epoch 21/50, Loss: 0.001867

Epoch 22/50, Loss: 0.001839

Epoch 23/50, Loss: 0.001810

Epoch 24/50, Loss: 0.001818

Epoch 25/50, Loss: 0.001757

Epoch 26/50, Loss: 0.001757

Epoch 27/50, Loss: 0.001725

Epoch 28/50, Loss: 0.001720

Epoch 29/50, Loss: 0.001702

Epoch 30/50, Loss: 0.001723

Epoch 31/50, Loss: 0.001675

Epoch 32/50, Loss: 0.001724

Epoch 33/50, Loss: 0.001677

Epoch 34/50, Loss: 0.001670

Epoch 35/50, Loss: 0.001640

Epoch 36/50, Loss: 0.001652

Epoch 37/50, Loss: 0.001637

Epoch 38/50, Loss: 0.001657

Epoch 39/50, Loss: 0.001634

Epoch 40/50, Loss: 0.001620

Epoch 41/50, Loss: 0.001602

Epoch 42/50, Loss: 0.001592

Epoch 43/50, Loss: 0.001616

Epoch 44/50, Loss: 0.001600

Epoch 45/50, Loss: 0.001586

Epoch 46/50, Loss: 0.001584

Epoch 47/50, Loss: 0.001581

Epoch 48/50, Loss: 0.001599

Epoch 49/50, Loss: 0.001547

Epoch 50/50, Loss: 0.001561

Explained variance ratio (2D PCA): [0.65413989 0.21425124]

Explained variance ratio (3D PCA): [0.65413989 0.21425124 0.12868649]

Во время обучения наблюдалось устойчивое снижение функции потерь (Loss), что говорит о корректной работе модели и успешном восстановлении входных данных после сжатия.

Для модели с 2 нейронами ошибка снизилась с 0.6866 до 0.0909.

Для модели с 3 нейронами — с 0.6226 до 0.00156, что демонстрирует более точную реконструкцию данных при большей размерности скрытого слоя.

Таким образом, автоэнкодер эффективно извлёк основные признаки данных и осуществил их компактное представление без существенных потерь информации.

Визуализация показала, что автоэнкодер способен выявлять скрытую структуру данных, аналогично PCA, но при этом лучше передаёт нелинейные зависимости.

Метод t-SNE продемонстрировал наиболее отчётливое разделение классов, что делает его полезным для визуального анализа, но не для реконструкции данных.

PCA, будучи линейным методом, уступает автоэнкодеру и t-SNE в выраженности кластеров, однако сохраняет интерпретируемость компонент и высокую долю объяснённой дисперсии.

В целом, результаты показывают, что: PCA — оптимален для быстрого анализа и уменьшения размерности; Автоэнкодер — мощный инструмент для извлечения нелинейных признаков; t-SNE — лучший выбор для визуализации сложных многомерных данных.

Вывод: научился применять автоэнкодеры для осуществления визуализации данных и их анализа