МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙГОСУДАРСТВЕННЫЙТЕХНИЧЕСКИЙУНИВЕРСИТЕТ» ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №2

Специальность ИИ-23

Выполнил А.А. Шурина, студент группы ИИ-23

Проверил К.В. Андренко,

ст. преп. кафедры ИИТ,

« » 2025 г.

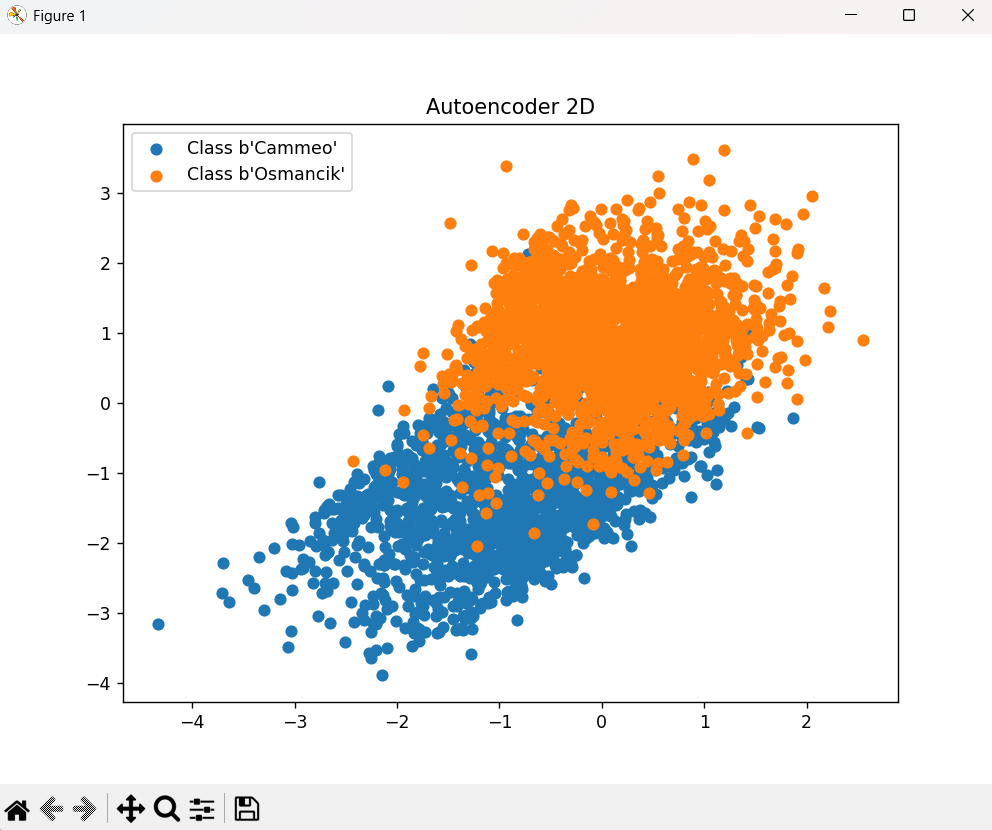
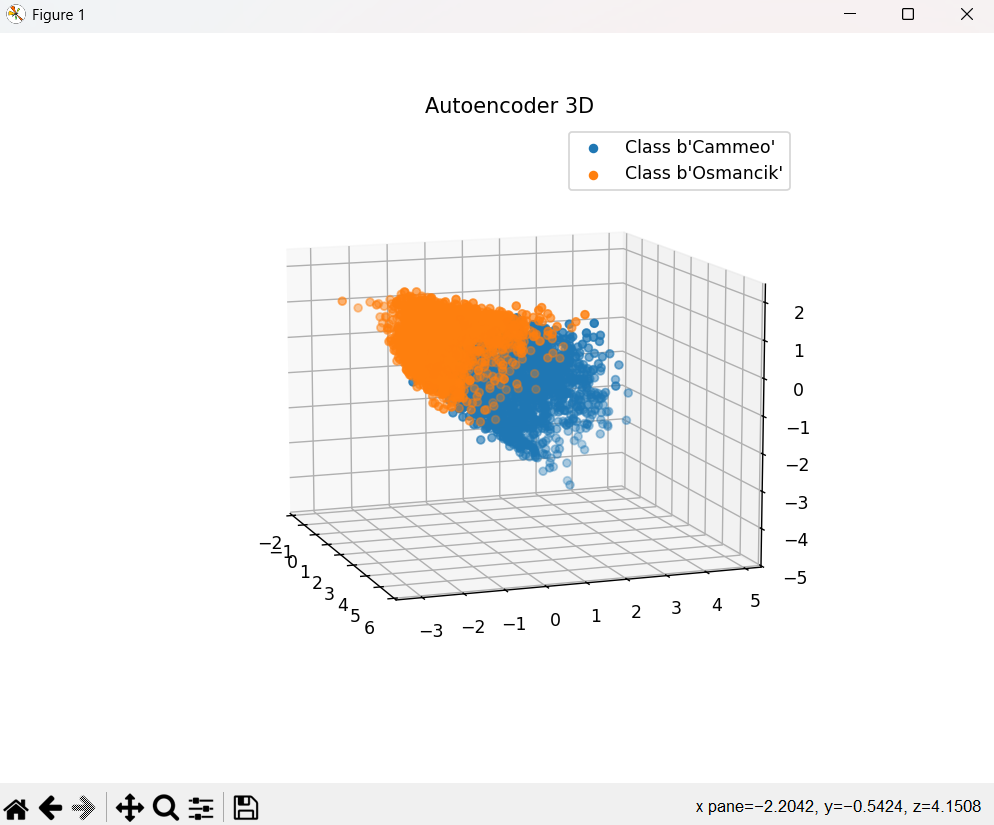
Брест 2025

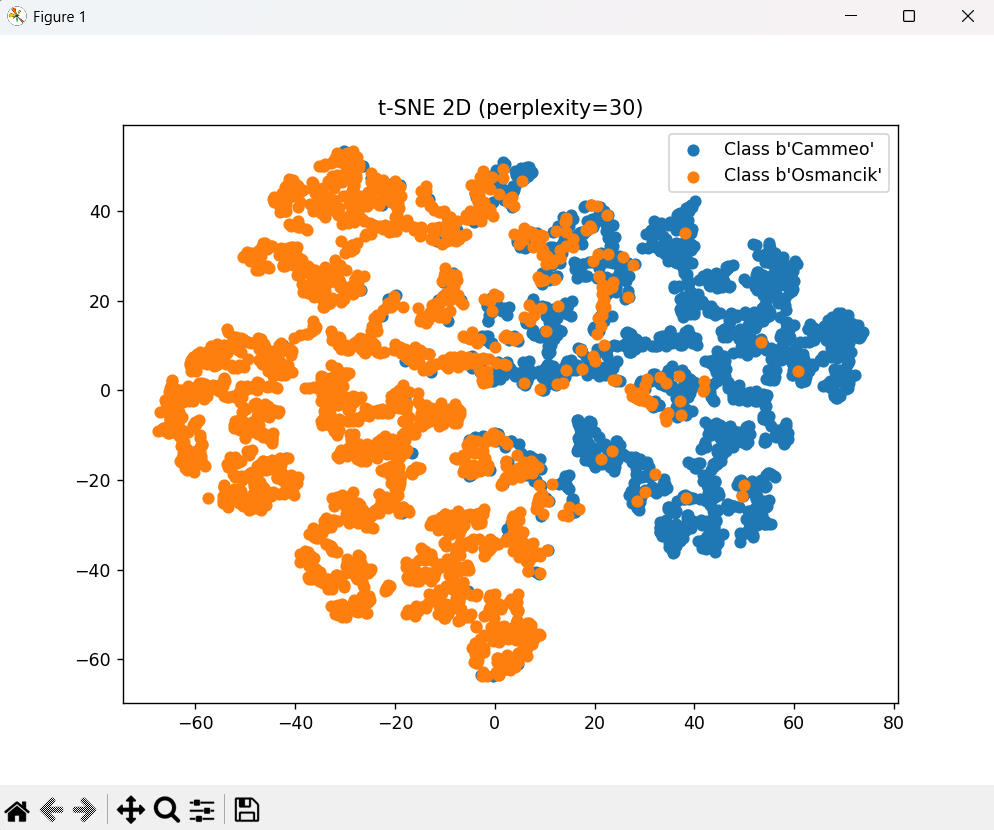
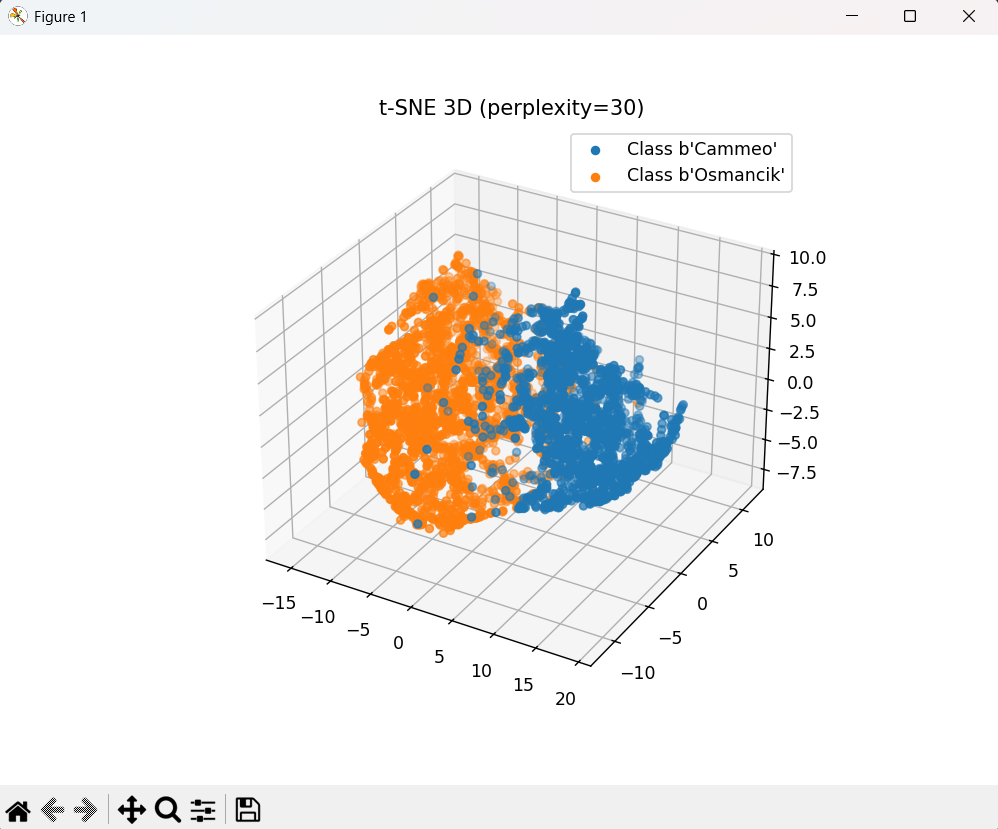
Цель работы: научиться применять автоэнкодеры для осуществления визуализации данных и их анализа

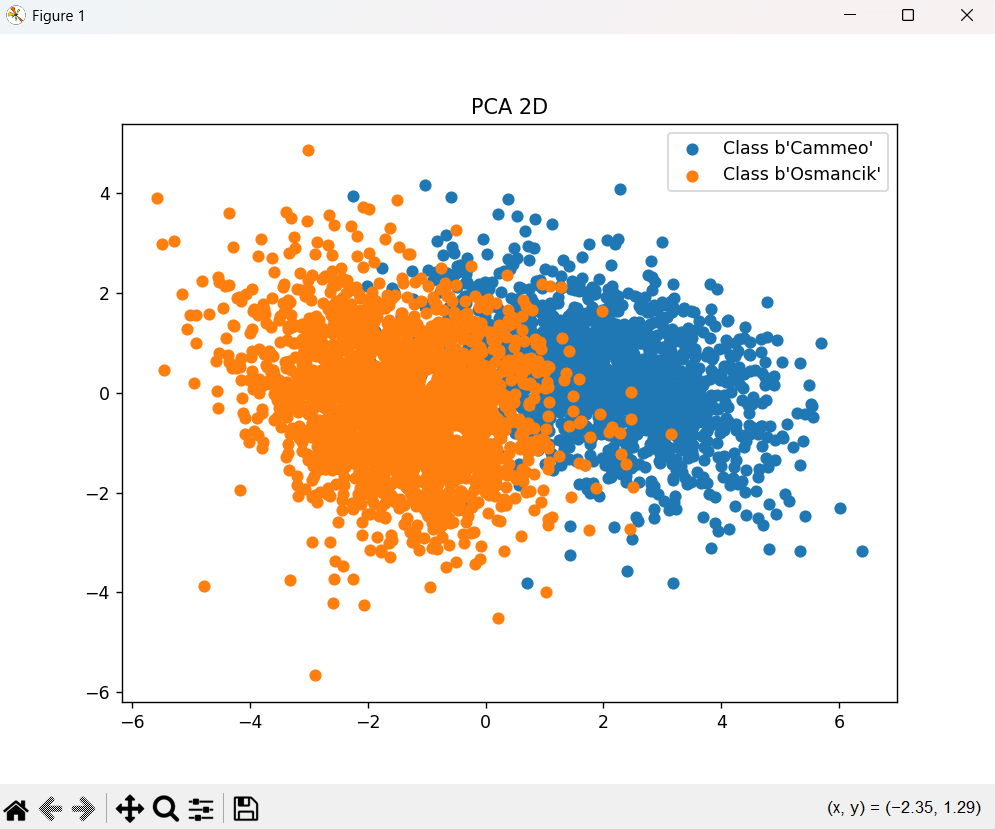
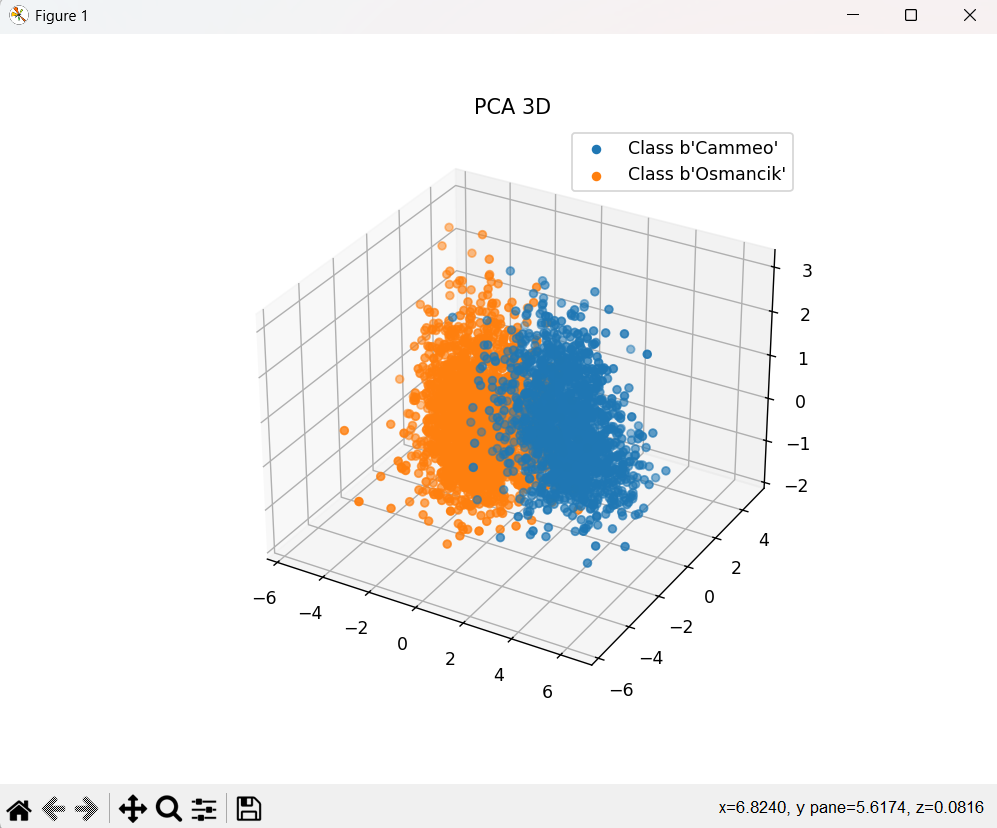
Вариант13  


Код программы :

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
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 scipy.io import arff  
  
# -------------------------------  
# 1. Загрузка данных из .arff  
# -------------------------------  
data\_path = r"C:\Users\User\Desktop\Studing-7sem\IAD\lab2\Rice\_Cammeo\_Osmancik.arff"  
data\_arff = arff.loadarff(data\_path)  
data\_df = pd.DataFrame(data\_arff[0])  
  
# Предполагаем, что последняя колонка — это класс, возможно в формате bytes  
X = data\_df.iloc[:, :-1].values  
y = data\_df.iloc[:, -1].values  
  
# Если классы в байтах, конвертируем в строки  
if y.dtype.kind == 'S':  
 y = np.array([val.decode('utf-8') for val in y])  
  
scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X)  
  
# -------------------------------  
# 2. Автоэнкодер  
# -------------------------------  
class Autoencoder(nn.Module):  
 def \_\_init\_\_(self, input\_size, bottleneck\_size):  
 super(Autoencoder, self).\_\_init\_\_()  
 self.encoder = nn.Sequential(  
 nn.Linear(input\_size, 64),  
 nn.ReLU(),  
 nn.Linear(64, bottleneck\_size)  
 )  
 self.decoder = nn.Sequential(  
 nn.Linear(bottleneck\_size, 64),  
 nn.ReLU(),  
 nn.Linear(64, input\_size)  
 )  
  
 def forward(self, x):  
 encoded = self.encoder(x)  
 decoded = self.decoder(encoded)  
 return encoded, decoded  
  
def train\_autoencoder(X\_scaled, bottleneck\_size=2, epochs=100, lr=0.001):  
 input\_size = X\_scaled.shape[1]  
 model = Autoencoder(input\_size, bottleneck\_size)  
 criterion = nn.MSELoss()  
 optimizer = optim.Adam(model.parameters(), lr=lr)  
  
 X\_tensor = torch.FloatTensor(X\_scaled)  
 for epoch in range(epochs):  
 optimizer.zero\_grad()  
 encoded, decoded = model(X\_tensor)  
 loss = criterion(decoded, X\_tensor)  
 loss.backward()  
 optimizer.step()  
 if epoch % 10 == 0:  
 print(f'Epoch {epoch}, Loss: {loss.item():.4f}')  
 return model  
  
# Пример для 2D и 3D узкого слоя  
ae\_2d = train\_autoencoder(X\_scaled, bottleneck\_size=2)  
ae\_3d = train\_autoencoder(X\_scaled, bottleneck\_size=3)  
  
# -------------------------------  
# 3. Визуализация кодировки автоэнкодера  
# -------------------------------  
def plot\_encoded(encoded\_data, labels, title="Autoencoder 2D", dim=2):  
 encoded\_data = encoded\_data.detach().numpy()  
 if dim == 2:  
 plt.figure(figsize=(8,6))  
 for cls in np.unique(labels):  
 plt.scatter(encoded\_data[labels==cls, 0], encoded\_data[labels==cls, 1], label=f'Class {cls}')  
 plt.legend()  
 plt.title(title)  
 plt.show()  
 elif dim == 3:  
 from mpl\_toolkits.mplot3d import Axes3D  
 fig = plt.figure(figsize=(8,6))  
 ax = fig.add\_subplot(111, projection='3d')  
 for cls in np.unique(labels):  
 ax.scatter(encoded\_data[labels==cls, 0], encoded\_data[labels==cls, 1],  
 encoded\_data[labels==cls, 2], label=f'Class {cls}')  
 ax.set\_title(title)  
 ax.legend()  
 plt.show()  
  
encoded\_2d, \_ = ae\_2d(torch.FloatTensor(X\_scaled))  
encoded\_3d, \_ = ae\_3d(torch.FloatTensor(X\_scaled))  
  
plot\_encoded(encoded\_2d, y, title="Autoencoder 2D", dim=2)  
plot\_encoded(encoded\_3d, y, title="Autoencoder 3D", dim=3)  
  
# -------------------------------  
# 4. t-SNE (2D и 3D)  
# -------------------------------  
def plot\_tsne(X\_scaled, labels, dim=2, perplexity=30):  
 tsne = TSNE(n\_components=dim, perplexity=perplexity, init='pca', random\_state=42)  
 X\_tsne = tsne.fit\_transform(X\_scaled)  
 if dim == 2:  
 plt.figure(figsize=(8,6))  
 for cls in np.unique(labels):  
 plt.scatter(X\_tsne[labels==cls, 0], X\_tsne[labels==cls, 1], label=f'Class {cls}')  
 plt.legend()  
 plt.title(f"t-SNE 2D (perplexity={perplexity})")  
 plt.show()  
 elif dim == 3:  
 from mpl\_toolkits.mplot3d import Axes3D  
 fig = plt.figure(figsize=(8,6))  
 ax = fig.add\_subplot(111, projection='3d')  
 for cls in np.unique(labels):  
 ax.scatter(X\_tsne[labels==cls, 0], X\_tsne[labels==cls, 1], X\_tsne[labels==cls, 2], label=f'Class {cls}')  
 ax.set\_title(f"t-SNE 3D (perplexity={perplexity})")  
 ax.legend()  
 plt.show()  
  
plot\_tsne(X\_scaled, y, dim=2, perplexity=30)  
plot\_tsne(X\_scaled, y, dim=3, perplexity=30)  
  
# -------------------------------  
# 5. PCA (2D и 3D)  
# -------------------------------  
def plot\_pca(X\_scaled, labels, dim=2):  
 pca = PCA(n\_components=dim)  
 X\_pca = pca.fit\_transform(X\_scaled)  
 if dim == 2:  
 plt.figure(figsize=(8,6))  
 for cls in np.unique(labels):  
 plt.scatter(X\_pca[labels==cls, 0], X\_pca[labels==cls, 1], label=f'Class {cls}')  
 plt.legend()  
 plt.title("PCA 2D")  
 plt.show()  
 elif dim == 3:  
 from mpl\_toolkits.mplot3d import Axes3D  
 fig = plt.figure(figsize=(8,6))  
 ax = fig.add\_subplot(111, projection='3d')  
 for cls in np.unique(labels):  
 ax.scatter(X\_pca[labels==cls, 0], X\_pca[labels==cls, 1], X\_pca[labels==cls, 2], label=f'Class {cls}')  
 ax.set\_title("PCA 3D")  
 ax.legend()  
 plt.show()  
  
plot\_pca(X\_scaled, y, dim=2)  
plot\_pca(X\_scaled, y, dim=3)

Результаты:  
 

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