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-24

Бузель С.Д.

Проверил:

Андренко К. В.

Брест 2025

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

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.

№	Выборка	Класс
3	Rice (Cammeo and Osmancik)	Class

Код программы:

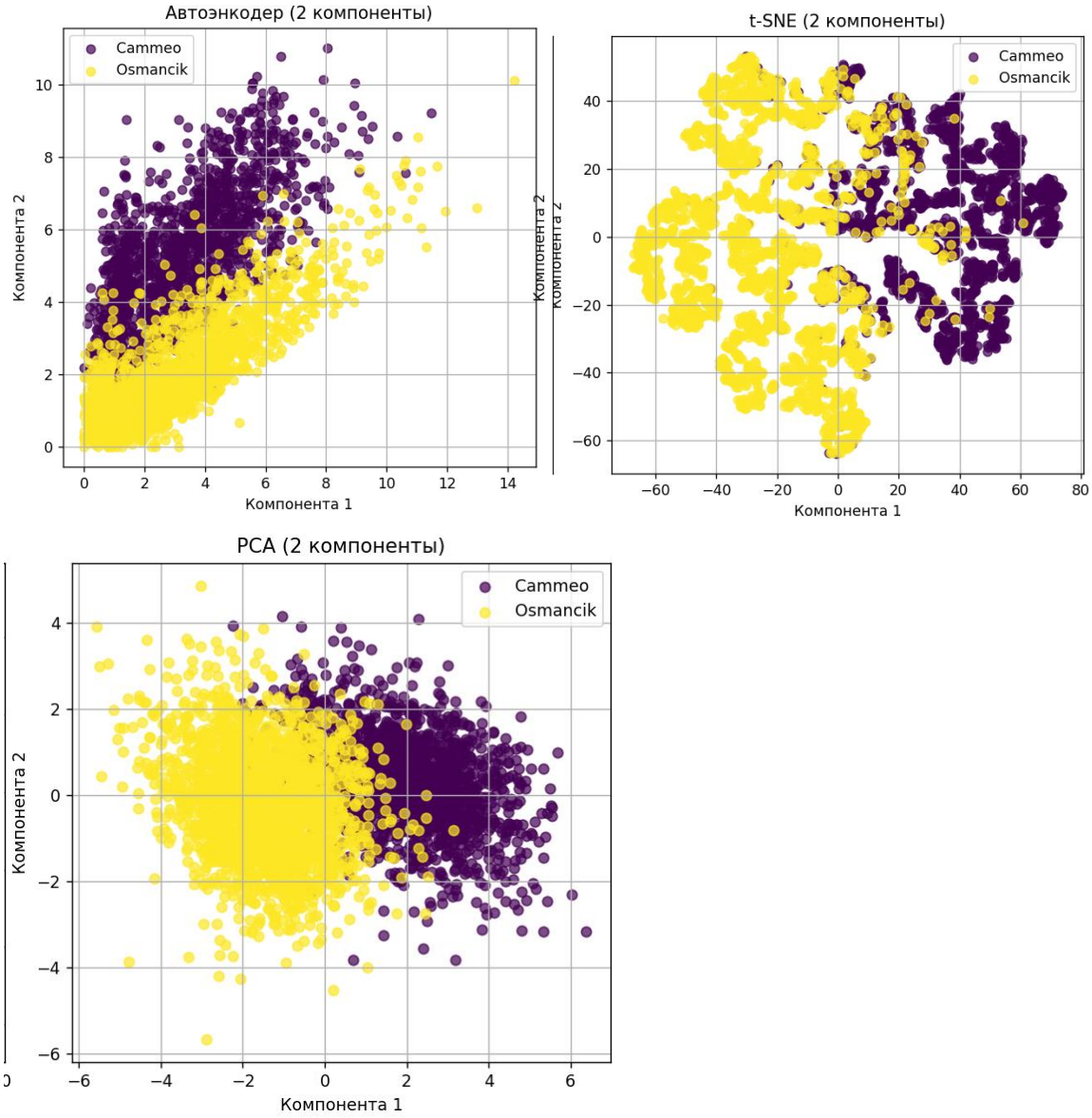
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import arff
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE

# Импорты для нейросети (Автоэнкодер)
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense

# --- 1. ЗАГРУЗКА И ПОДГОТОВКА ДАННЫХ ---

file_path = 'D:/ОИИС/lab2/Rice_Cammeo_Osmancik.arff'

try:
    # Загружаем .arff файл
    data_arff, meta = arff.loadarff(file_path)

    # Преобразуем его в Pandas DataFrame
    data = pd.DataFrame(data_arff)

    data['Class'] = data['Class'].apply(lambda x: x.decode('utf-8'))
```

```
    print("Данные успешно загружены. Первые 5 строк:")
    print(data.head())

    # Отделяем признаки X от целевого класса y
    X = data.drop('Class', axis=1)
    y = data['Class']

    # Стандартизация признаков
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    n_features = X_scaled.shape[1]

except FileNotFoundError:
    print(f"Ошибка: Файл не найден по пути '{file_path}'.")
    data = None

if data is not None:
    # --- 2. РЕАЛИЗАЦИЯ МЕТОДОВ СНИЖЕНИЯ РАЗМЕРНОСТИ ---

    # --- Метод 1: Автоэнкодер ---
    def run_autoencoder(data, n_components):
        print(f"\nОбучение автоэнкодера для {n_components} компонент...")

        # Архитектура нейросети
        input_layer = Input(shape=(n_features,))
        # Слой кодировщика
        encoded = Dense(128, activation='relu')(input_layer)
        encoded = Dense(64, activation='relu')(encoded)
        # Скрытый слой с нужной размерностью
        bottleneck = Dense(n_components, activation='relu')(encoded)
        # Слой декодировщика
        decoded = Dense(64, activation='relu')(bottleneck)
        decoded = Dense(128, activation='relu')(decoded)
        # Выходной слой (размерность как у входа)
        output_layer = Dense(n_features, activation='linear')(decoded)

        # Модель автоэнкодера (для обучения)
        autoencoder = Model(input_layer, output_layer)
        autoencoder.compile(optimizer='adam', loss='mse')

        # Обучаем автоэнкодер восстанавливать исходные данные
        autoencoder.fit(data, data, epochs=50, batch_size=256, shuffle=True, verbose=0)

        # Модель кодировщика (для получения сжатых данных)
        encoder = Model(input_layer, bottleneck)

        # Получаем данные в новой размерности
        projected_data = encoder.predict(data)
        print("Обучение завершено.")
        return projected_data

    X_ae_2d = run_autoencoder(X_scaled, n_components=2)
    X_ae_3d = run_autoencoder(X_scaled, n_components=3)

    # --- Метод 2: t-SNE ---
    print("\nПрименение t-SNE для 2 компонент...")
    tsne_2d = TSNE(n_components=2, perplexity=30, random_state=42)
```

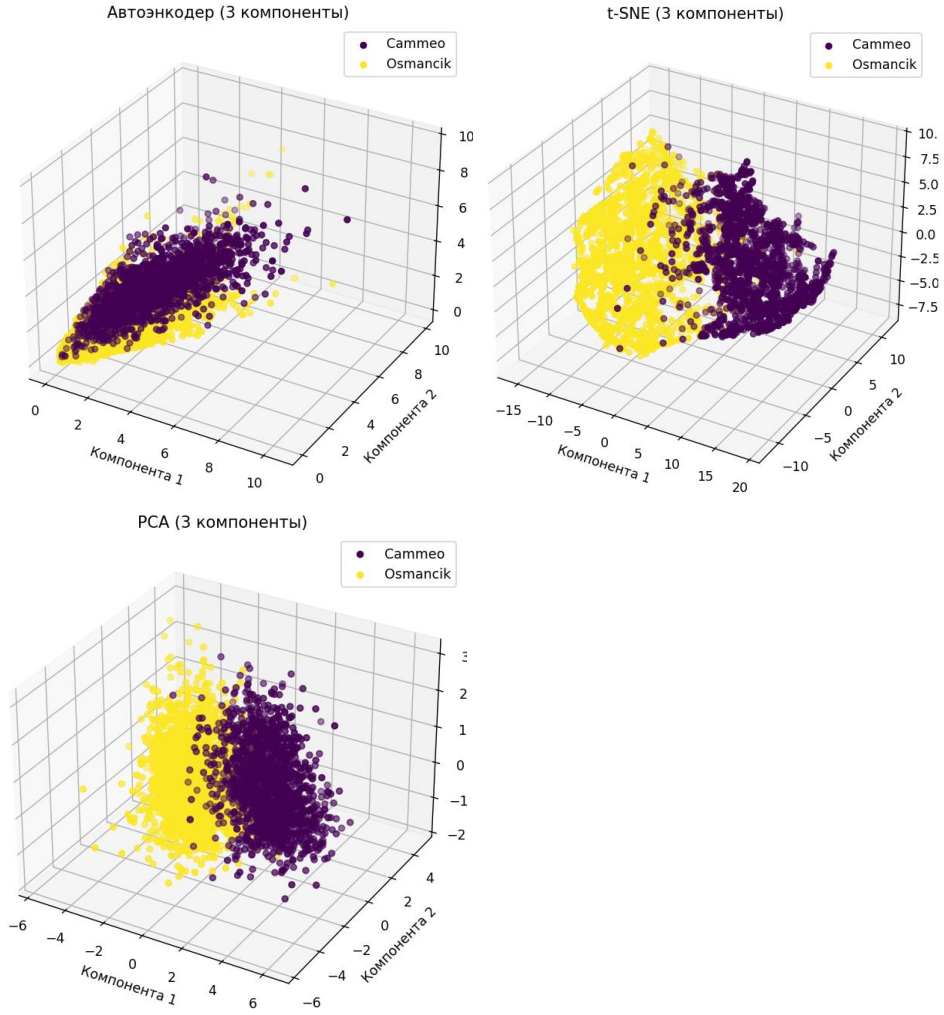
```
X_tsne_2d = tsne_2d.fit_transform(X_scaled)

print("Применение t-SNE для 3 компонент...")
tsne_3d = TSNE(n_components=3, perplexity=30, random_state=42)
X_tsne_3d = tsne_3d.fit_transform(X_scaled)
print("t-SNE завершен.")

# --- Метод 3: PCA (из Лаб 1) ---
print("\nПрименение PCA...")
pca_2d = PCA(n_components=2)
X_pca_2d = pca_2d.fit_transform(X_scaled)

pca_3d = PCA(n_components=3)
X_pca_3d = pca_3d.fit_transform(X_scaled)
print("PCA завершен.")

# --- 3. ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ ---

unique_classes = y.unique()
colors = plt.cm.get_cmap('viridis', len(unique_classes))

# --- Визуализация 2D ---
fig, axes = plt.subplots(1, 3, figsize=(20, 6))

plots_2d = {
    'Автоэнкодер (2 компоненты)': X_ae_2d,
    't-SNE (2 компоненты)': X_tsne_2d,
    'PCA (2 компоненты)': X_pca_2d
}

for i, (title, data_proj) in enumerate(plots_2d.items()):
    ax = axes[i]
    for j, cls in enumerate(unique_classes):
        indices = y == cls
        ax.scatter(data_proj[indices, 0], data_proj[indices, 1], color=colors(j), label=cls, alpha=0.7)
    ax.set_title(title)
    ax.set_xlabel('Компонента 1')
    ax.set_ylabel('Компонента 2')
    ax.legend()
    ax.grid(True)

plt.suptitle('Сравнение методов снижения размерности (2D)', fontsize=16)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()

# --- Визуализация 3D ---
fig = plt.figure(figsize=(21, 7))
plots_3d = {
    'Автоэнкодер (3 компоненты)': X_ae_3d,
    't-SNE (3 компоненты)': X_tsne_3d,
    'PCA (3 компоненты)': X_pca_3d
}

for i, (title, data_proj) in enumerate(plots_3d.items()):
    ax = fig.add_subplot(1, 3, i + 1, projection='3d')
    for j, cls in enumerate(unique_classes):
        indices = y == cls
        ax.scatter(data_proj[indices, 0], data_proj[indices, 1], data_proj[indices, 2], color=colors(j), label=cls)
```

```
    ax.set_title(title)
    ax.set_xlabel('Компонента 1')
    ax.set_ylabel('Компонента 2')
    ax.set_zlabel('Компонента 3')
    ax.legend()

plt.suptitle('Сравнение методов снижения размерности (3D)', fontsize=16)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

Сравнение методов снижения размерности (2D):



Сравнение методов снижения размерности (3D):



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