Министерство образования Республики Беларусь
Учреждение образования
«Брестский Государственный технический университет»
Кафедра ИИТ

Лабораторная работа №2
По дисциплине «ИАД»
Тема: «Автоэнкодеры»

Выполнил:
Студент 4 курса
Группы ИИ-24
Крейдич А. А.
Проверила:
Андренко К. В.

Брест 2025

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

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.

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

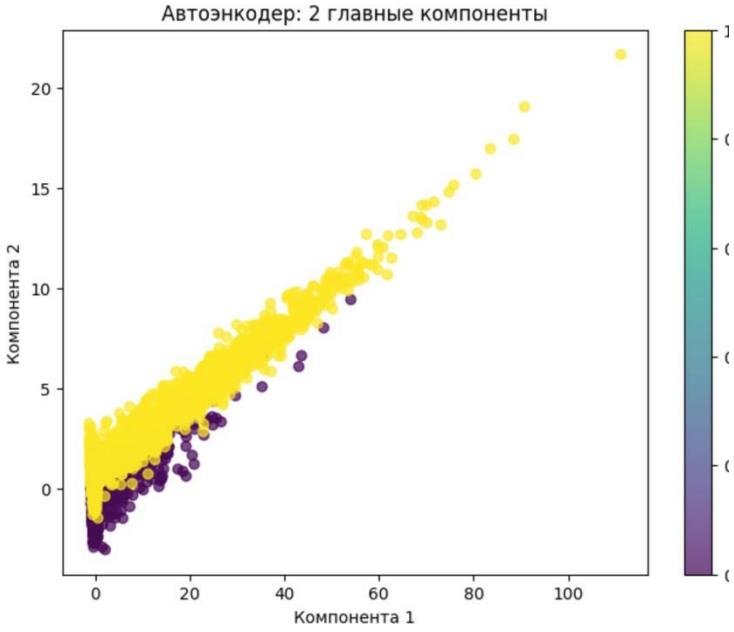
Текст программы :

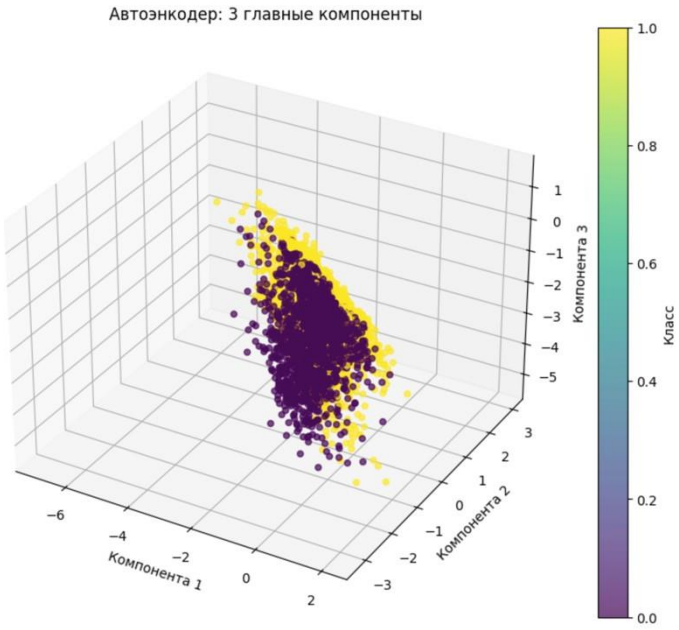
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.optimizers import Adam

from ucimlrepo import fetch_ucirepo


rice = fetch_ucirepo(id=545)

X = rice.data.features
y = rice.data.targets['Class'].map({'Cammeo': 0, 'Osmancik': 1})


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

print("Размерность данных:", X_scaled.shape)


input_dim = X_scaled.shape[1]
encoding_dim = 2

input_layer = Input(shape=(input_dim,))
x = Dense(6, activation='relu')(input_layer)
```

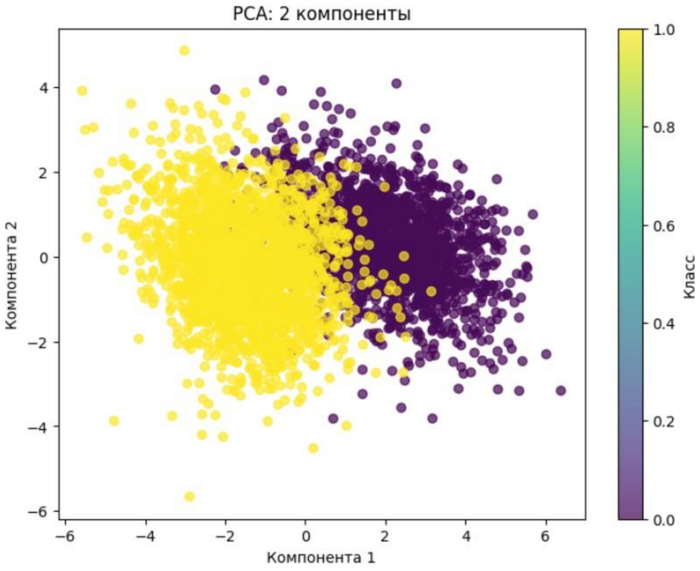
```
bottleneck_2d = Dense(encoding_dim, activation='linear')(x)
x = Dense(6, activation='relu')(bottleneck_2d)
output_layer = Dense(input_dim, activation='linear')(x)

autoencoder_2d = Model(input_layer, output_layer)
encoder_2d = Model(input_layer, bottleneck_2d)

autoencoder_2d.compile(optimizer=Adam(learning_rate=0.01), loss='mse')
autoencoder_2d.fit(
    X_scaled, X_scaled,
    epochs=100,
    batch_size=64,
    verbose=0
)

X_encoded_2d = encoder_2d.predict(X_scaled)

plt.figure(figsize=(8, 6))
plt.scatter(X_encoded_2d[:, 0], X_encoded_2d[:, 1], c=y, cmap='viridis', alpha=0.7)
plt.title('Автоэнкодер: проекция на 2 компоненты')
plt.xlabel('Первая компонента')
plt.ylabel('Вторая компонента')
plt.colorbar(label='Класс')
plt.show()


encoding_dim = 3

input_layer = Input(shape=(input_dim,))
x = Dense(6, activation='relu')(input_layer)
bottleneck_3d = Dense(encoding_dim, activation='linear')(x)
x = Dense(6, activation='relu')(bottleneck_3d)
output_layer = Dense(input_dim, activation='linear')(x)

autoencoder_3d = Model(input_layer, output_layer)
encoder_3d = Model(input_layer, bottleneck_3d)

autoencoder_3d.compile(optimizer=Adam(learning_rate=0.01), loss='mse')
autoencoder_3d.fit(
    X_scaled, X_scaled,
    epochs=100,
    batch_size=64,
    verbose=0
)

X_encoded_3d = encoder_3d.predict(X_scaled)

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
sc = ax.scatter(
    X_encoded_3d[:, 0],
    X_encoded_3d[:, 1],
    X_encoded_3d[:, 2],
    c=y,
    cmap='viridis',
    alpha=0.7
)
ax.set_title('Автоэнкодер: проекция на 3 компоненты')
ax.set_xlabel('Первая компонента')
ax.set_ylabel('Вторая компонента')
ax.set_zlabel('Третья компонента')
fig.colorbar(sc, label='Класс')
plt.show()


# PCA 2D

pca_2d = PCA(n_components=2)
X_pca_2d = pca_2d.fit_transform(X_scaled)

plt.figure(figsize=(8, 6))
plt.scatter(X_pca_2d[:, 0], X_pca_2d[:, 1], c=y, cmap='viridis', alpha=0.7)
```

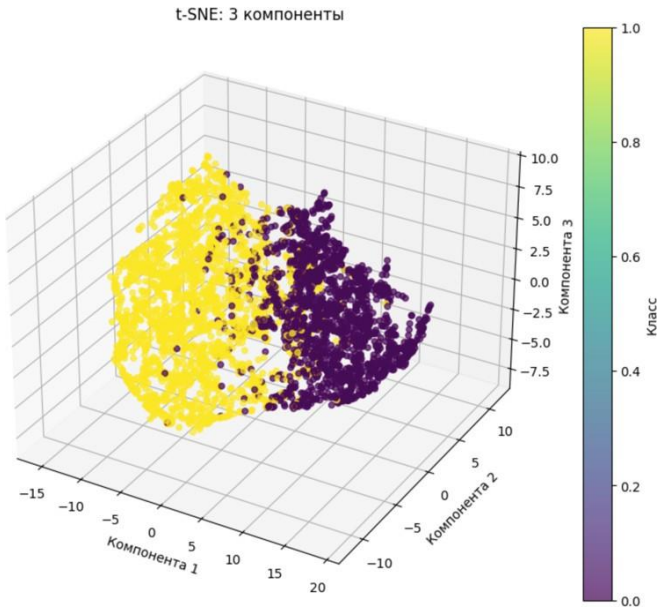
```
plt.title('PCA: проекция на 2 компоненты')
plt.xlabel('Первая главная компонента')
plt.ylabel('Вторая главная компонента')
plt.colorbar(label='Класс')
plt.show()


# PCA 3D

pca_3d = PCA(n_components=3)
X_pca_3d = pca_3d.fit_transform(X_scaled)

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
sc = ax.scatter(
    X_pca_3d[:, 0],
    X_pca_3d[:, 1],
    X_pca_3d[:, 2],
    c=y,
    cmap='viridis',
    alpha=0.7
)
ax.set_title('PCA: проекция на 3 компоненты')
ax.set_xlabel('Первая главная компонента')
ax.set_ylabel('Вторая главная компонента')
ax.set_zlabel('Третья главная компонента')
fig.colorbar(sc, label='Класс')
plt.show()
```

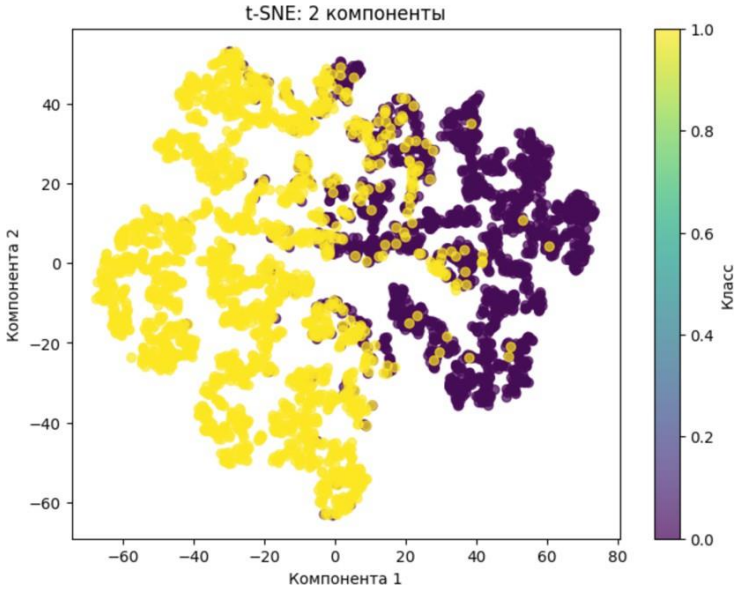
Результат программы:

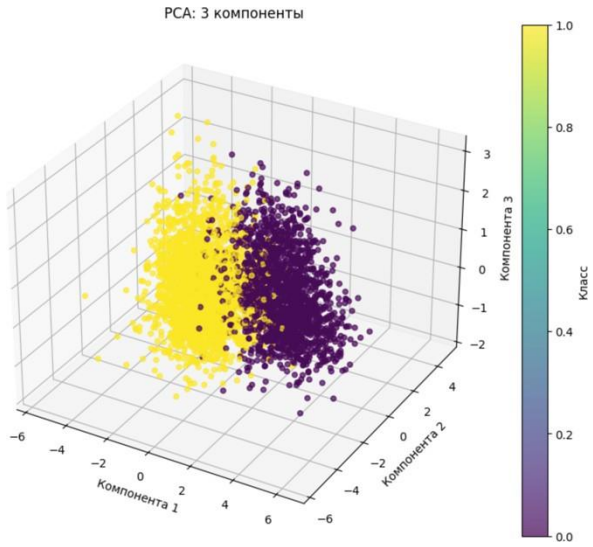












Вывод: научился применять автоэнкодеры для визуализации данных для последующего анализа.