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
Капуза Н.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.

№ варианта	Выборка	Класс
4	Wine Quality (white)	quality

Ход работы:

Код программы:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA

# Для 3D-графиков
from mpl_toolkits.mplot3d import Axes3D

# Для построения автоэнкодера
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model

# Загрузка и подготовка данных
```

```
try:
    # Попытка загрузить локальный файл
    df = pd.read_csv('winequality-white.csv', sep=';')
except FileNotFoundError:
    # Если файл не найден, загружаем из сети
    print("Локальный файл не найден. Загрузка данных из сети...")
    url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-
quality/winequality-white.csv"
    df = pd.read_csv(url, sep=';')

print("Первые 5 строк данных:")
print(df.head())
print("\nИнформация о данных:")
df.info()

# Проверка на наличие пропущенных значений
print("\nКоличество пропущенных значений:")
print(df.isnull().sum())

# 1. Отделяем признаки (X) от целевой переменной (y)
# Целевая переменная 'quality' используется для визуализации
X = df.drop('quality', axis=1)
y = df['quality']

# 2. Стандартизация данных
# Важный шаг для всех трех методов
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print(f"\nРазмерность исходных данных: {X_scaled.shape}")
print(f"Количество уникальных классов (quality): {y.nunique()}")



# --- Задание 1: Проецирование с помощью автоэнкодера ---

print("\n--- 1. Автоэнкодер ---")

# Определяем архитектуру автоэнкодера
input_dim = X_scaled.shape[1]
encoding_dim_2d = 2  # Размерность для 2D визуализации
encoding_dim_3d = 3  # Размерность для 3D визуализации

def create_autoencoder(encoding_dim):
```

```
    # Входной слой
    input_layer = Input(shape=(input_dim,))

    # Слой кодировщика (Encoder)
    # Можно добавить больше слоев для более глубокой модели
    encoded = Dense(10, activation='relu')(input_layer)
    encoded = Dense(encoding_dim, activation='relu')(encoded) # Скрытый слой

    # Слой декодировщика (Decoder)
    decoded = Dense(10, activation='relu')(encoded)
    decoded = Dense(input_dim, activation='linear')(decoded) # Восстановление
исходных данных

    # Модель автоэнкодера
    autoencoder = Model(input_layer, decoded)

    # Модель кодировщика (для получения скрытого представления)
    encoder = Model(input_layer, encoded)

    autoencoder.compile(optimizer='adam', loss='mean_squared_error')
    return autoencoder, encoder

# --- Автоэнкодер для 2D ---
autoencoder_2d, encoder_2d = create_autoencoder(encoding_dim_2d)
print("\nАрхитектура автоэнкодера (2D):")
autoencoder_2d.summary()

# Обучение модели
autoencoder_2d.fit(X_scaled, X_scaled,
                   epochs=50,
                   batch_size=256,
                   shuffle=True,
                   verbose=0, # Отключаем вывод логов обучения
                   validation_split=0.2)

# Получаем скрытое представление (проекцию)
X_autoencoder_2d = encoder_2d.predict(X_scaled)
print(f"Размерность данных после 2D автоэнкодера: {X_autoencoder_2d.shape}")


# --- Автоэнкодер для 3D ---
autoencoder_3d, encoder_3d = create_autoencoder(encoding_dim_3d)
autoencoder_3d.fit(X_scaled, X_scaled, epochs=50, batch_size=256, shuffle=True,
verbose=0, validation_split=0.2)
X_autoencoder_3d = encoder_3d.predict(X_scaled)
```

```
print(f"Размерность данных после 3D автоэнкодера: {X_autoencoder_3d.shape}")


# --- Задание 3: Реализация метода t-SNE ---

print("\n--- 3. t-SNE ---")
# t-SNE может быть медленным для больших наборов данных,
# поэтому для ускорения можно взять случайную подвыборку
sample_size = min(len(X_scaled), 2000) # Используем не более 2000 точек
np.random.seed(42)
indices = np.random.choice(len(X_scaled), sample_size, replace=False)
X_sample = X_scaled[indices]
y_sample = y.iloc[indices]

# --- t-SNE для 2D ---
# perplexity рекомендуется выбирать в диапазоне от 5 до 50
tsne_2d = TSNE(n_components=2, perplexity=30, n_iter=1000, random_state=42)
X_tsne_2d = tsne_2d.fit_transform(X_sample)
print(f"Размерность данных после t-SNE (2D): {X_tsne_2d.shape}")

# --- t-SNE для 3D ---
tsne_3d = TSNE(n_components=3, perplexity=30, n_iter=1000, random_state=42)
X_tsne_3d = tsne_3d.fit_transform(X_sample)
print(f"Размерность данных после t-SNE (3D): {X_tsne_3d.shape}")


# --- Задание 4: Применение метода PCA ---

print("\n--- 4. PCA ---")
# --- PCA для 2D ---
pca_2d = PCA(n_components=2)
X_pca_2d = pca_2d.fit_transform(X_scaled)
print(f"Размерность данных после PCA (2D): {X_pca_2d.shape}")
print(f"Доля объясненной дисперсии (2D):
{sum(pca_2d.explained_variance_ratio_):.4f}")

# --- PCA для 3D ---
pca_3d = PCA(n_components=3)
X_pca_3d = pca_3d.fit_transform(X_scaled)
print(f"Размерность данных после PCA (3D): {X_pca_3d.shape}")
print(f"Доля объясненной дисперсии (3D):
{sum(pca_3d.explained_variance_ratio_):.4f}")


# --- Задание 2: Визуализация результатов ---
```

```
print("\n--- 2. Визуализация ---")

def plot_2d(X_proj, y_proj, title):
    plt.figure(figsize=(10, 8))
    scatter = plt.scatter(X_proj[:, 0], X_proj[:, 1], c=y_proj, cmap='viridis', alpha=0.7)
    plt.title(title, fontsize=16)
    plt.xlabel('Компонента 1')
    plt.ylabel('Компонента 2')
    plt.legend(handles=scatter.legend_elements()[0], labels=sorted(y_proj.unique()),
title="Качество")
    plt.grid(True)
    plt.show()

def plot_3d(X_proj, y_proj, title):
    fig = plt.figure(figsize=(11, 9))
    ax = fig.add_subplot(111, projection='3d')
    scatter = ax.scatter(X_proj[:, 0], X_proj[:, 1], X_proj[:, 2], c=y_proj, cmap='viridis',
alpha=0.7)
    ax.set_title(title, fontsize=16)
    ax.set_xlabel('Компонента 1')
    ax.set_ylabel('Компонента 2')
    ax.set_zlabel('Компонента 3')
    legend = ax.legend(handles=scatter.legend_elements()[0],
labels=sorted(y_proj.unique()), title="Качество")
    plt.show()

# Визуализация для Автоэнкодера
plot_2d(X_autoencoder_2d, y, 'Автоэнкодер: 2D проекция')
plot_3d(X_autoencoder_3d, y, 'Автоэнкодер: 3D проекция')

# Визуализация для t-SNE (используем подвыборку)
plot_2d(X_tsne_2d, y_sample, 't-SNE: 2D проекция (perplexity=30)')
plot_3d(X_tsne_3d, y_sample, 't-SNE: 3D проекция (perplexity=30)')

# Визуализация для PCA
plot_2d(X_pca_2d, y, 'PCA: 2D проекция')
plot_3d(X_pca_3d, y, 'PCA: 3D проекция')

print("\nРабота завершена. Сделайте выводы на основе полученных графиков.")
```

Графики:

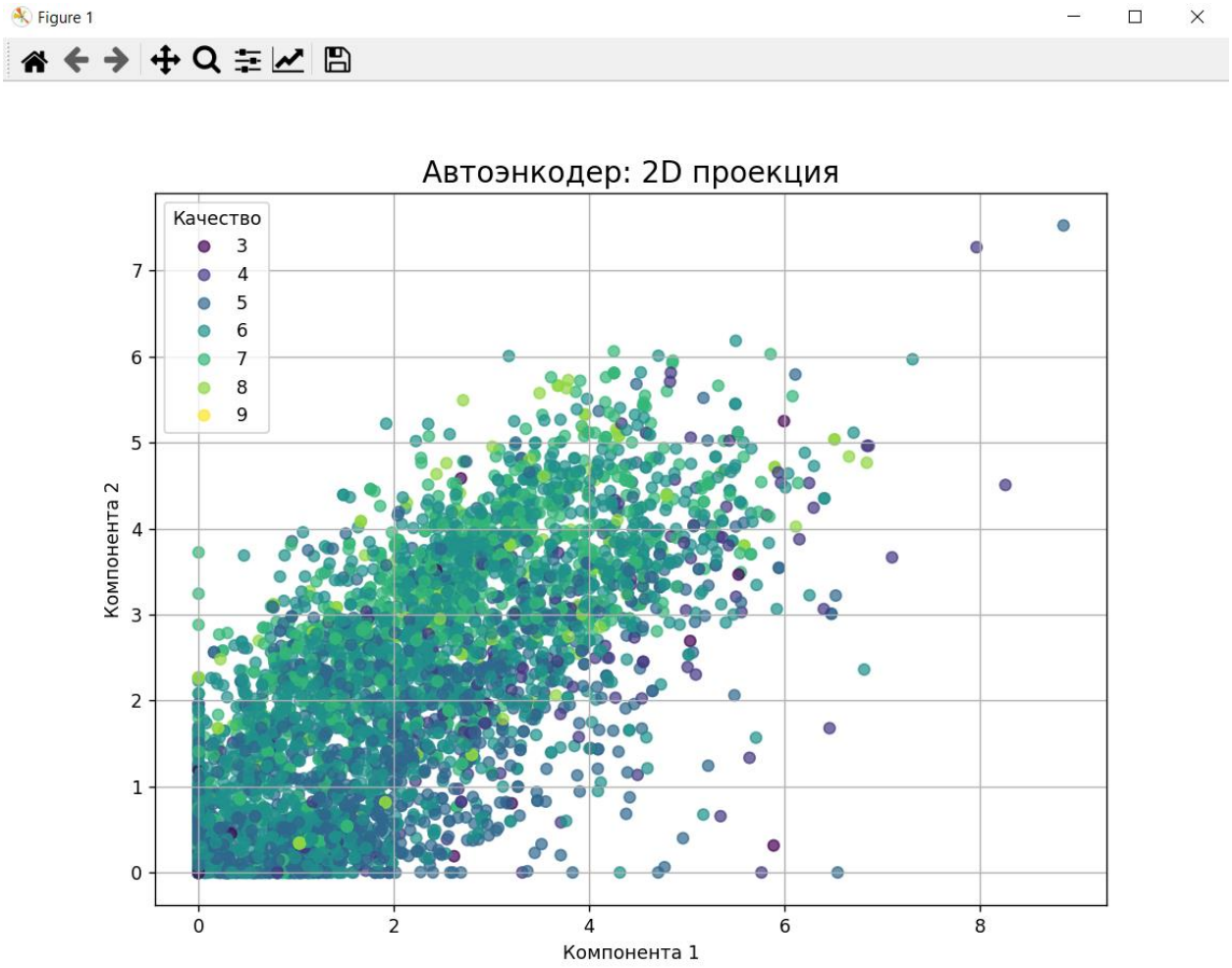


Figure 1
Автоэнкодер: 2D проекция
Качество
3
4
5
6
7
8
9
Компонента 2
Компонента 1
0
1
2
3
4
5
6
7
0
2
4
6
8

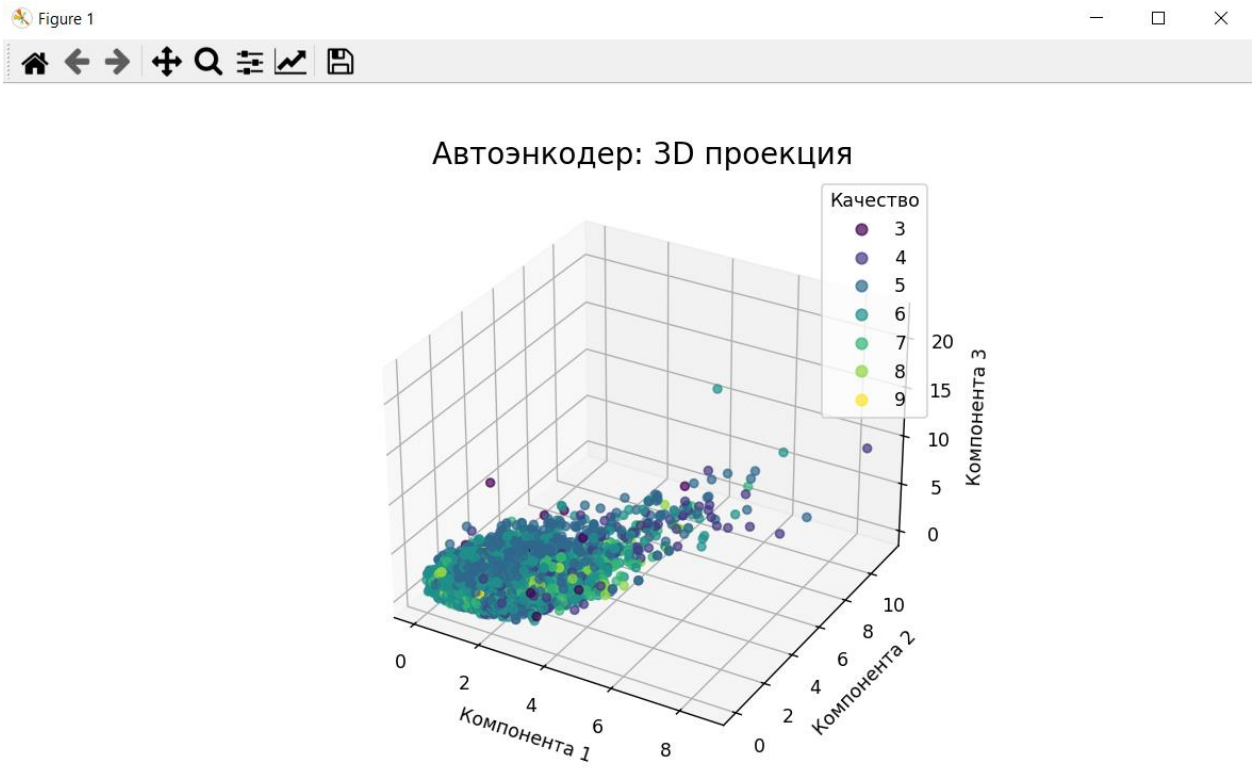


Figure 1
Автоэнкодер: 3D проекция
Качество
3
4
5
6
7
8
9
Компонента 1
Компонента 2
Компонента 3
0
2
4
6
8
0
2
4
6
8
10
0
5
10
15
20

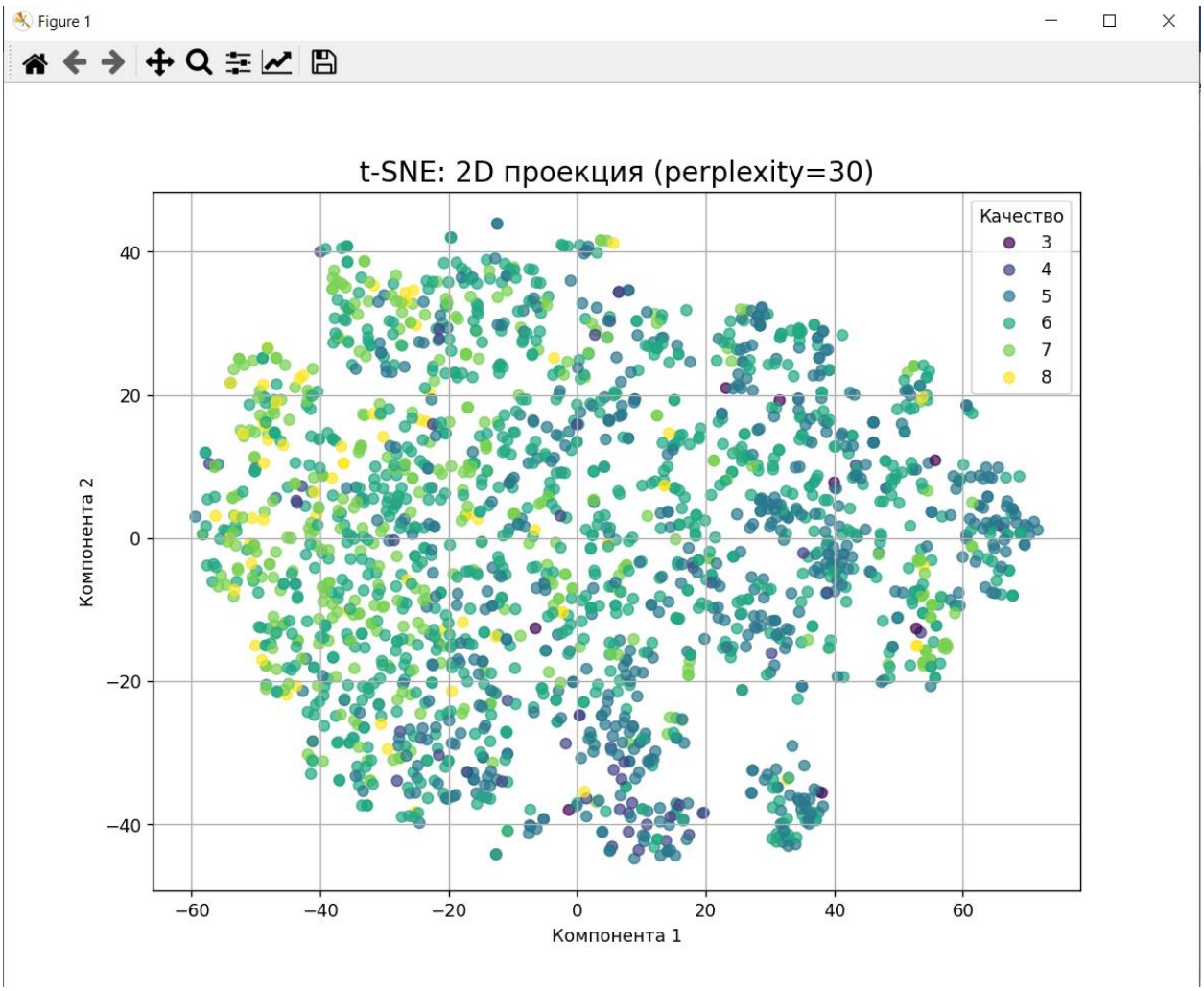


Figure 1
t-SNE: 2D проекция (perplexity=30)
Качество
3
4
5
6
7
8
Компонента 2
Компонента 1
40
20
0
−20
−40
−60
−40
−20
0
20
40
60

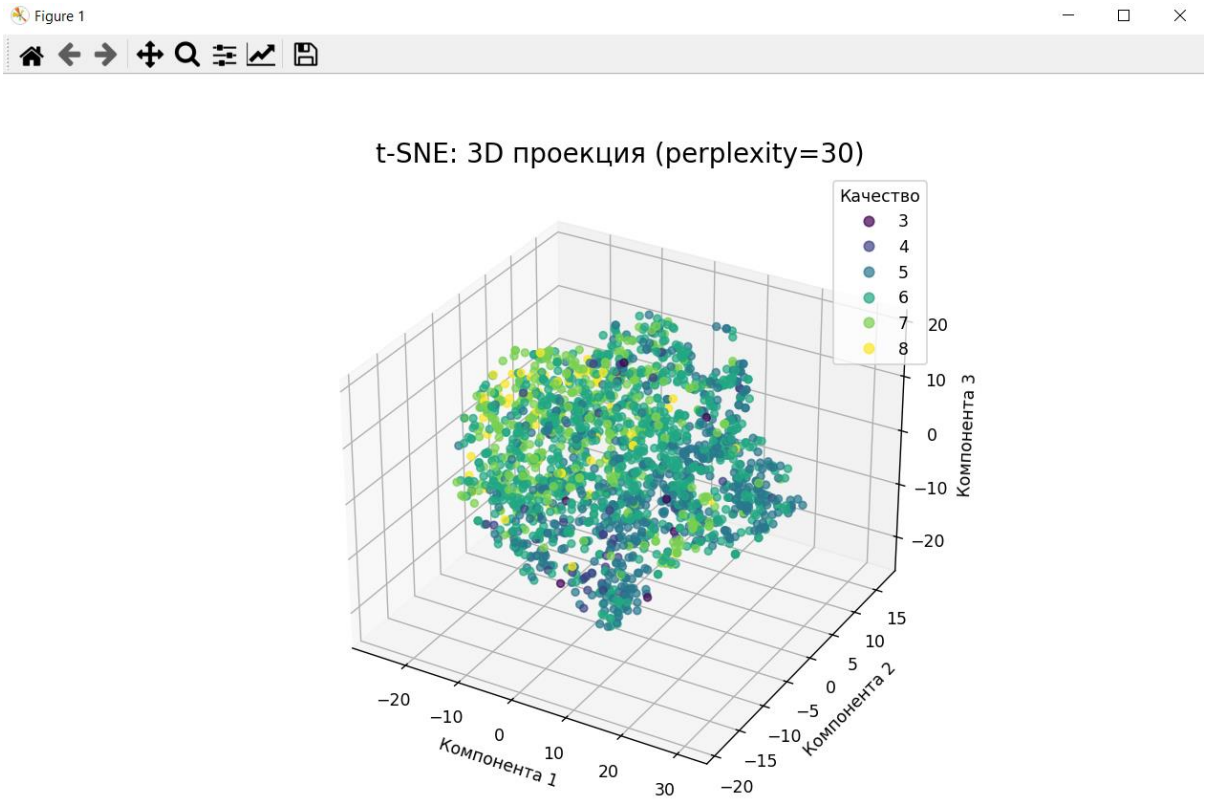


Figure 1
t-SNE: 3D проекция (perplexity=30)
Качество
3
4
5
6
7
8
Компонента 1
Компонента 2
Компонента 3

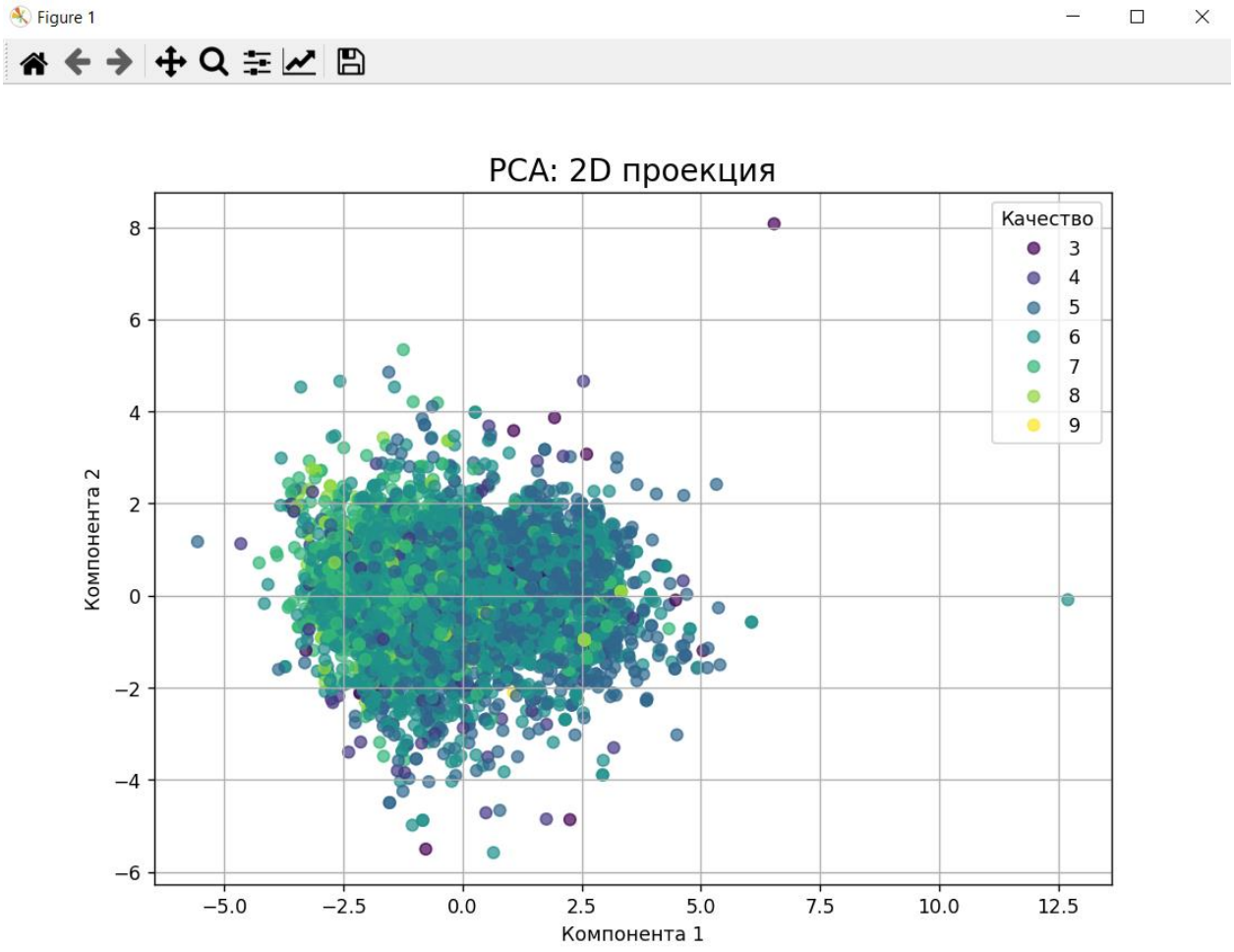
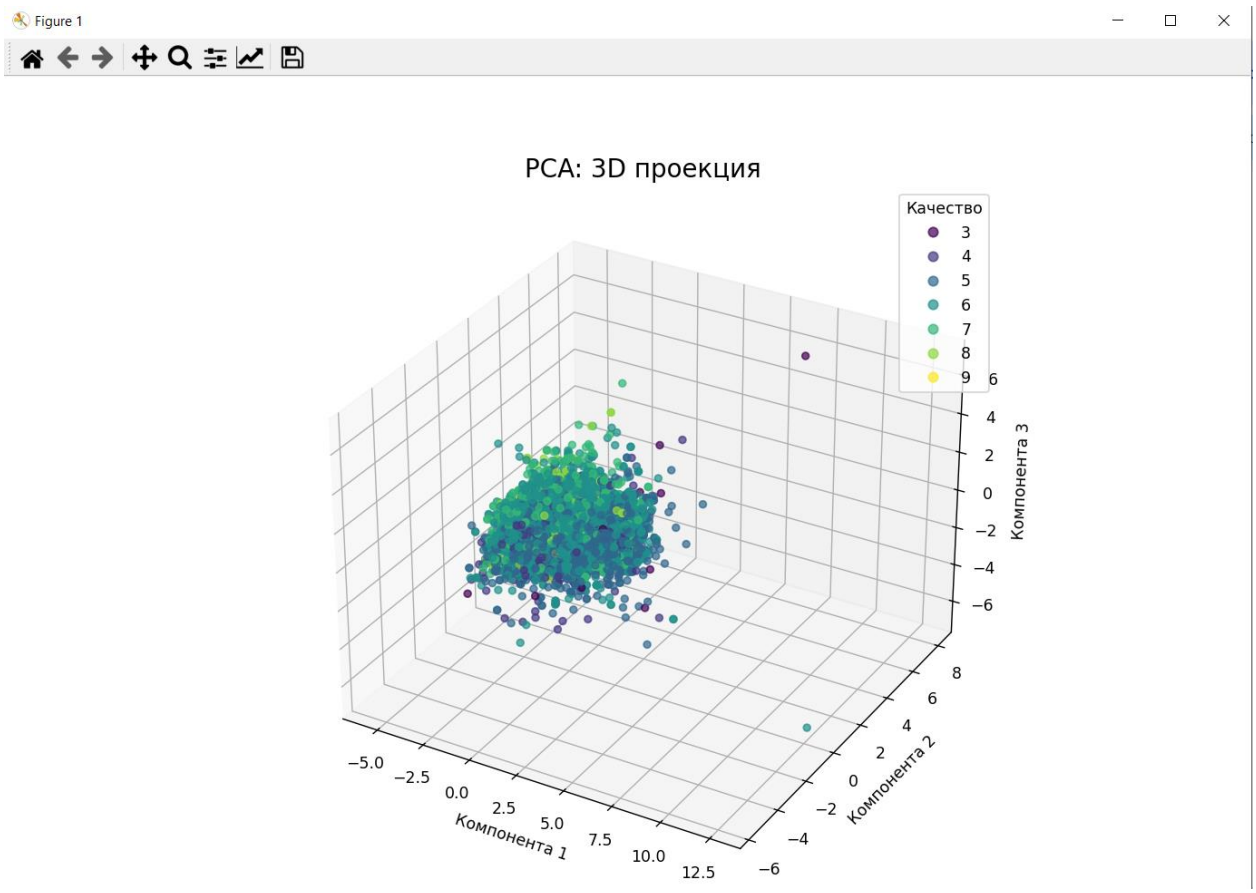


Figure 1
PCA: 2D проекция
Качество
3
4
5
6
7
8
9
Компонента 1
Компонента 2



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