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

Швороб В.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.

Задание по вариантам

12	Mushroom	poisonous

Код программы:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.metrics import accuracy_score
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

from ucimlrepo import fetch_ucirepo

mushroom = fetch_ucirepo(id=73)
X = mushroom.data.features
y = mushroom.data.targets

print("Размерность данных:", X.shape)
print("\nПервые 5 строк данных:")
print(X.head())
print("\nЦелевая переменная:")
print(y.head())


label_encoders = {}
X_encoded = X.copy()

for column in X_encoded.columns:
    le = LabelEncoder()
    X_encoded[column] = le.fit_transform(X_encoded[column].astype(str))
    label_encoders[column] = le
```

```
le_target = LabelEncoder()
y_encoded = le_target.fit_transform(y.values.ravel())

print("Уникальные значения целевой переменной:", np.unique(y_encoded))
print("Размерность после кодирования:", X_encoded.shape)

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

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y_encoded, test_size=0.2, random_state=42, stratify=y_encoded)

print(f"Train size: {X_train.shape}, Test size: {X_test.shape}")

print("\nАВТОЭНКОДЕР С 2 НЕЙРОНАМИ")

input_dim = X_train.shape[1]
encoding_dim_2d = 2

input_layer = layers.Input(shape=(input_dim,))
encoded = layers.Dense(64, activation='relu')(input_layer)
encoded = layers.Dense(32, activation='relu')(encoded)
encoded = layers.Dense(16, activation='relu')(encoded)
bottleneck_2d = layers.Dense(encoding_dim_2d, activation='linear', name='bottleneck')(encoded)

decoded = layers.Dense(16, activation='relu')(bottleneck_2d)
decoded = layers.Dense(32, activation='relu')(decoded)
decoded = layers.Dense(64, activation='relu')(decoded)
decoded = layers.Dense(input_dim, activation='linear')(decoded)

autoencoder_2d = keras.Model(input_layer, decoded)
encoder_2d = keras.Model(input_layer, bottleneck_2d)

autoencoder_2d.compile(optimizer='adam', loss='mse')
autoencoder_2d.summary()

history_2d = autoencoder_2d.fit(
    X_train, X_train,
    epochs=50,
    batch_size=32,
    validation_data=(X_test, X_test),
    verbose=1,
    shuffle=True
)

encoded_features_2d = encoder_2d.predict(X_scaled)
print(f"Размерность закодированных признаков: {encoded_features_2d.shape}")

print("\nАВТОЭНКОДЕР С 3 НЕЙРОНАМИ")

encoding_dim_3d = 3

input_layer_3d = layers.Input(shape=(input_dim,))
encoded_3d = layers.Dense(64, activation='relu')(input_layer_3d)
encoded_3d = layers.Dense(32, activation='relu')(encoded_3d)
encoded_3d = layers.Dense(16, activation='relu')(encoded_3d)
bottleneck_3d = layers.Dense(encoding_dim_3d, activation='linear', name='bottleneck_3d')(encoded_3d)

decoded_3d = layers.Dense(16, activation='relu')(bottleneck_3d)
decoded_3d = layers.Dense(32, activation='relu')(decoded_3d)
decoded_3d = layers.Dense(64, activation='relu')(decoded_3d)
decoded_3d = layers.Dense(input_dim, activation='linear')(decoded_3d)
```

```
autoencoder_3d = keras.Model(input_layer_3d, decoded_3d)
encoder_3d = keras.Model(input_layer_3d, bottleneck_3d)

autoencoder_3d.compile(optimizer='adam', loss='mse')

history_3d = autoencoder_3d.fit(
    X_train, X_train,
    epochs=50,
    batch_size=32,
    validation_data=(X_test, X_test),
    verbose=1,
    shuffle=True
)

encoded_features_3d = encoder_3d.predict(X_scaled)

print("\nВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ")


def plot_results_2d(features, title, class_labels=['edible', 'poisonous']):
    plt.figure(figsize=(15, 5))

    plt.subplot(1, 2, 1)
    scatter = plt.scatter(features[:, 0], features[:, 1], c=y_encoded,
                          cmap='viridis', alpha=0.7)
    plt.colorbar(scatter)
    plt.xlabel('Component 1')
    plt.ylabel('Component 2')
    plt.title(f'{title} - Colored by class')

    plt.subplot(1, 2, 2)
    for class_label in np.unique(y_encoded):
        mask = y_encoded == class_label
        plt.scatter(features[mask, 0], features[mask, 1],
                    label=class_labels[class_label], alpha=0.7)
    plt.xlabel('Component 1')
    plt.ylabel('Component 2')
    plt.title(f'{title} - Class separation')
    plt.legend()

    plt.tight_layout()
    plt.show()


def plot_results_3d(features, title, class_labels=['edible', 'poisonous']):
    fig = plt.figure(figsize=(15, 5))

    # Вид 1
    ax1 = fig.add_subplot(1, 2, 1, projection='3d')
    scatter = ax1.scatter(features[:, 0], features[:, 1], features[:, 2],
                          c=y_encoded, cmap='viridis', alpha=0.7)
    plt.colorbar(scatter)
    ax1.set_xlabel('Component 1')
    ax1.set_ylabel('Component 2')
    ax1.set_zlabel('Component 3')
    ax1.set_title(f'{title} - 3D View')

    # Вид 2 (только 2 компоненты для лучшей видимости)
    ax2 = fig.add_subplot(1, 2, 2)
    for class_label in np.unique(y_encoded):
        mask = y_encoded == class_label
        ax2.scatter(features[mask, 0], features[mask, 1],
                    label=class_labels[class_label], alpha=0.7)
    ax2.set_xlabel('Component 1')
```

```
    ax2.set_ylabel('Component 2')
    ax2.set_title(f'{title} - 2D Projection')
    ax2.legend()

    plt.tight_layout()
    plt.show()


print("Визуализация автоэнкодера (2D):")
plot_results_2d(encoded_features_2d, 'Autoencoder 2D')

print("Визуализация автоэнкодера (3D):")
plot_results_3d(encoded_features_3d, 'Autoencoder 3D')

print("\nМЕТОД t-SNE")

perplexities = [20, 35, 50]

plt.figure(figsize=(18, 5))
for i, perplexity in enumerate(perplexities, 1):
    tsne_2d = TSNE(n_components=2, perplexity=perplexity, random_state=42, init='pca')
    X_tsne_2d = tsne_2d.fit_transform(X_scaled)

    plt.subplot(1, 3, i)
    for class_label in np.unique(y_encoded):
        mask = y_encoded == class_label
        plt.scatter(X_tsne_2d[mask, 0], X_tsne_2d[mask, 1],
                    label=['edible', 'poisonous'][class_label], alpha=0.7)
    plt.xlabel('t-SNE Component 1')
    plt.ylabel('t-SNE Component 2')
    plt.title(f't-SNE 2D (perplexity={perplexity})')
    plt.legend()

plt.tight_layout()
plt.show()

best_perplexity = 35
print(f"t-SNE 3D c perplexity={best_perplexity}:")
tsne_3d = TSNE(n_components=3, perplexity=best_perplexity, random_state=42, init='pca')
X_tsne_3d = tsne_3d.fit_transform(X_scaled)

fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(X_tsne_3d[:, 0], X_tsne_3d[:, 1], X_tsne_3d[:, 2],
                     c=y_encoded, cmap='viridis', alpha=0.7)
plt.colorbar(scatter)
ax.set_xlabel('t-SNE Component 1')
ax.set_ylabel('t-SNE Component 2')
ax.set_zlabel('t-SNE Component 3')
ax.set_title(f't-SNE 3D (perplexity={best_perplexity})')
plt.show()

print("\nМЕТОД PCA")

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

print("Объясненная дисперсия PCA (2 компоненты):", pca_2d.explained_variance_ratio_)
print("Суммарная объясненная дисперсия:", sum(pca_2d.explained_variance_ratio_))

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

print("\nОбъясненная дисперсия PCA (3 компоненты):", pca_3d.explained_variance_ratio_)
```

```
print("Суммарная объясненная дисперсия:", sum(pca_3d.explained_variance_ratio_))

print("Визуализация PCA (2D):")
plot_results_2d(X_pca_2d, 'PCA 2D')

print("Визуализация PCA (3D):")
plot_results_3d(X_pca_3d, 'PCA 3D')

print("\nСРАВНИТЕЛЬНЫЙ АНАЛИЗ")

fig, axes = plt.subplots(2, 3, figsize=(20, 12))

methods = [
    (encoded_features_2d, 'Autoencoder 2D'),
    (encoded_features_3d[:, :2], 'Autoencoder 3D (2D projection)'),
    (X_tsne_2d, f't-SNE 2D (perplexity={best_perplexity})'),
    (X_pca_2d, 'PCA 2D'),
    (X_tsne_3d[:, :2], 't-SNE 3D (2D projection)'),
    (X_pca_3d[:, :2], 'PCA 3D (2D projection)')
]

for i, (features, title) in enumerate(methods):
    row = i // 3
    col = i % 3
    for class_label in np.unique(y_encoded):
        mask = y_encoded == class_label
        axes[row, col].scatter(features[mask, 0], features[mask, 1],
                               label=['edible', 'poisonous'][class_label], alpha=0.7)
    axes[row, col].set_title(title)
    if i == 0:
        axes[row, col].legend()

plt.tight_layout()
plt.show()

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history_2d.history['loss'], label='Training Loss')
plt.plot(history_2d.history['val_loss'], label='Validation Loss')
plt.title('Autoencoder 2D - Training History')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history_3d.history['loss'], label='Training Loss')
plt.plot(history_3d.history['val_loss'], label='Validation Loss')
plt.title('Autoencoder 3D - Training History')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```

Вывод программы:
Размерность данных: (8124, 22)

Первые 5 строк данных:

	cap-shape	cap-surface	cap-color	...	spore-print-color	population	habitat
0	x	s	n	...	k	s	u
1	x	s	y	...	n	n	g
2	b	s	w	...	n	n	m
3	x	y	w	...	k	s	u
4	x	s	g	...	n	a	g

[5 rows x 22 columns]

Целевая переменная:

	poisonous
0	p
1	e
2	e
3	p
4	e

Уникальные значения целевой переменной: [0 1]

Размерность после кодирования: (8124, 22)

2025-11-14 13:25:07.396220: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.

To enable the following instructions: SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.

Train size: (6499, 22), Test size: (1625, 22)

АВТОЭНКОДЕР С 2 НЕЙРОНАМИ

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 22)	0
dense (Dense)	(None, 64)	1,472
dense_1 (Dense)	(None, 32)	2,080
dense_2 (Dense)	(None, 16)	528
bottleneck (Dense)	(None, 2)	34
dense_3 (Dense)	(None, 16)	48

dense_4 (Dense)	(None, 32)	544
dense_5 (Dense)	(None, 64)	2,112
dense_6 (Dense)	(None, 22)	1,430

Total params: 8,248 (32.22 KB)

Trainable params: 8,248 (32.22 KB)

Non-trainable params: 0 (0.00 B)

Epoch 1/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.6436 - val_loss: 0.4691

Epoch 2/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.3939 - val_loss: 0.3572

Epoch 3/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.3334 - val_loss: 0.3232

Epoch 4/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.3098 - val_loss: 0.2982

Epoch 5/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2807 - val_loss: 0.2703

Epoch 6/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2597 - val_loss: 0.2565

Epoch 7/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2451 - val_loss: 0.2436

Epoch 8/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2334 - val_loss: 0.2328

Epoch 9/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2255 - val_loss: 0.2285

Epoch 10/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2181 - val_loss: 0.2220

Epoch 11/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2123 - val_loss: 0.2169

Epoch 12/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2069 - val_loss: 0.2132

Epoch 13/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2042 - val_loss: 0.2061

Epoch 14/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1960 - val_loss: 0.2087

Epoch 15/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1968 - val_loss: 0.1977

Epoch 16/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1888 - val_loss: 0.1939

Epoch 17/50

204/204 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1852 - val_loss: 0.1926

Epoch 18/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1825 - val_loss: 0.1867

Epoch 19/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1798 - val_loss: 0.1840

Epoch 20/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1758 - val_loss: 0.1838

Epoch 21/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1731 - val_loss: 0.1816

Epoch 22/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1704 - val_loss: 0.1766

Epoch 23/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1678 - val_loss: 0.1785

Epoch 24/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1666 - val_loss: 0.1747

Epoch 25/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1645 - val_loss: 0.1698

Epoch 26/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1604 - val_loss: 0.1664

Epoch 27/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1588 - val_loss: 0.1692

Epoch 28/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1593 - val_loss: 0.1605

Epoch 29/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1553 - val_loss: 0.1623

Epoch 30/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1530 - val_loss: 0.1585

Epoch 31/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1500 - val_loss: 0.1568

Epoch 32/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1481 - val_loss: 0.1545

Epoch 33/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1469 - val_loss: 0.1508

Epoch 34/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1445 - val_loss: 0.1484

Epoch 35/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1429 - val_loss: 0.1493

Epoch 36/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1418 - val_loss: 0.1484

Epoch 37/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1407 - val_loss: 0.1451

Epoch 38/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1388 - val_loss: 0.1432

Epoch 39/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1375 - val_loss: 0.1419

Epoch 40/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1388 - val_loss: 0.1419

Epoch 41/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1347 - val_loss: 0.1409

Epoch 42/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1331 - val_loss: 0.1380

Epoch 43/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1323 - val_loss: 0.1374

Epoch 44/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1315 - val_loss: 0.1375

Epoch 45/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1302 - val_loss: 0.1339

Epoch 46/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1294 - val_loss: 0.1343

Epoch 47/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1286 - val_loss: 0.1387

Epoch 48/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1276 - val_loss: 0.1342

Epoch 49/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1270 - val_loss: 0.1331

Epoch 50/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1276 - val_loss: 0.1310

254/254 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 508us/step

Размерность закодированных признаков: (8124, 2)

АВТОЭНКОДЕР С 3 НЕЙРОНАМИ

Epoch 1/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.5640 - val_loss: 0.3489

Epoch 2/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2996 - val_loss: 0.2695

Epoch 3/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2454 - val_loss: 0.2362

Epoch 4/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2156 - val_loss: 0.2144

Epoch 5/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1979 - val_loss: 0.1993

Epoch 6/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1840 - val_loss: 0.1840

Epoch 7/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1708 - val_loss: 0.1722

Epoch 8/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1609 - val_loss: 0.1624

Epoch 9/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1542 - val_loss: 0.1551

Epoch 10/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1488 - val_loss: 0.1518

Epoch 11/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1444 - val_loss: 0.1476

Epoch 12/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1396 - val_loss: 0.1420

Epoch 13/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1362 - val_loss: 0.1386

Epoch 14/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1330 - val_loss: 0.1350

Epoch 15/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1292 - val_loss: 0.1327

Epoch 16/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1262 - val_loss: 0.1291

Epoch 17/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1236 - val_loss: 0.1277

Epoch 18/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1201 - val_loss: 0.1222

Epoch 19/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1170 - val_loss: 0.1222

Epoch 20/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1150 - val_loss: 0.1151

Epoch 21/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1109 - val_loss: 0.1141

Epoch 22/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1086 - val_loss: 0.1133

Epoch 23/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1066 - val_loss: 0.1096

Epoch 24/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1034 - val_loss: 0.1077

Epoch 25/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1028 - val_loss: 0.1009

Epoch 26/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0998 - val_loss: 0.1038

Epoch 27/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0984 - val_loss: 0.1009

Epoch 28/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0956 - val_loss: 0.1024

Epoch 29/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0945 - val_loss: 0.0975

Epoch 30/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0930 - val_loss: 0.0960

Epoch 31/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0915 - val_loss: 0.0967

Epoch 32/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0905 - val_loss: 0.0947

Epoch 33/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0891 - val_loss: 0.0934

Epoch 34/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0879 - val_loss: 0.0931

Epoch 35/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0883 - val_loss: 0.0905

Epoch 36/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0870 - val_loss: 0.0884

Epoch 37/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0854 - val_loss: 0.0884

Epoch 38/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0852 - val_loss: 0.0882

Epoch 39/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0837 - val_loss: 0.0884

Epoch 40/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0841 - val_loss: 0.0846

Epoch 41/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0835 - val_loss: 0.0880

Epoch 42/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0831 - val_loss: 0.0854

Epoch 43/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0820 - val_loss: 0.0840

Epoch 44/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0815 - val_loss: 0.0833

Epoch 45/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0805 - val_loss: 0.0890

Epoch 46/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0815 - val_loss: 0.0841

Epoch 47/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0800 - val_loss: 0.0861

Epoch 48/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0796 - val_loss: 0.0834

Epoch 49/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0786 - val_loss: 0.0808

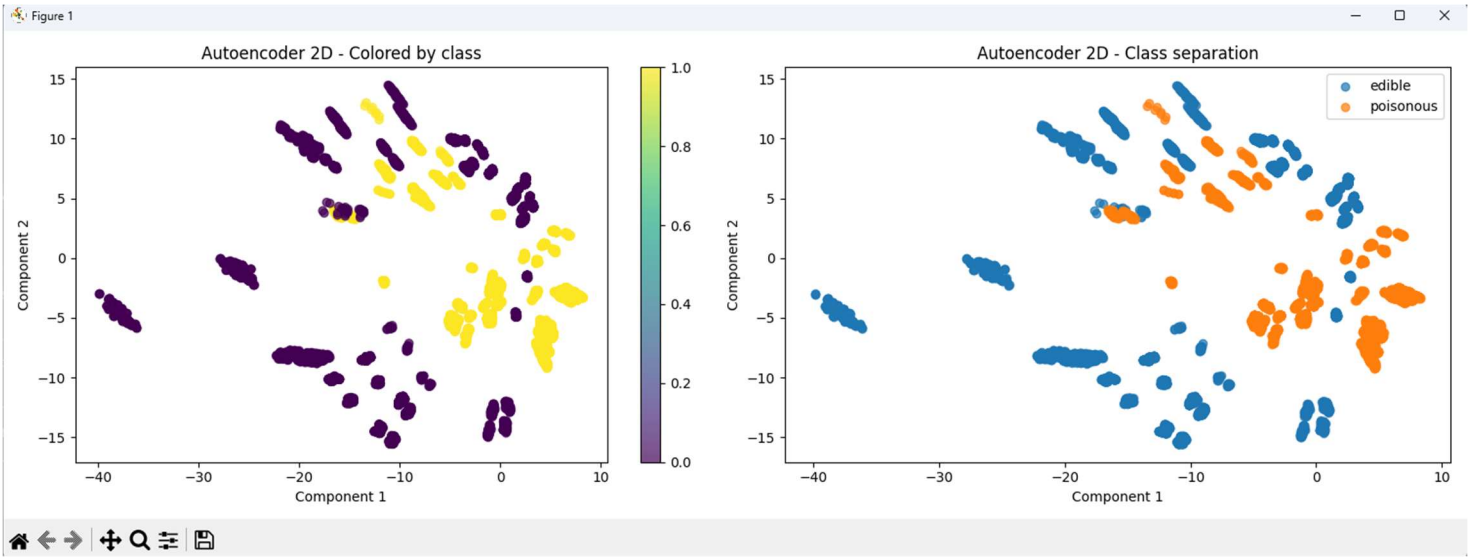
Epoch 50/50

204/204 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0785 - val_loss: 0.0807

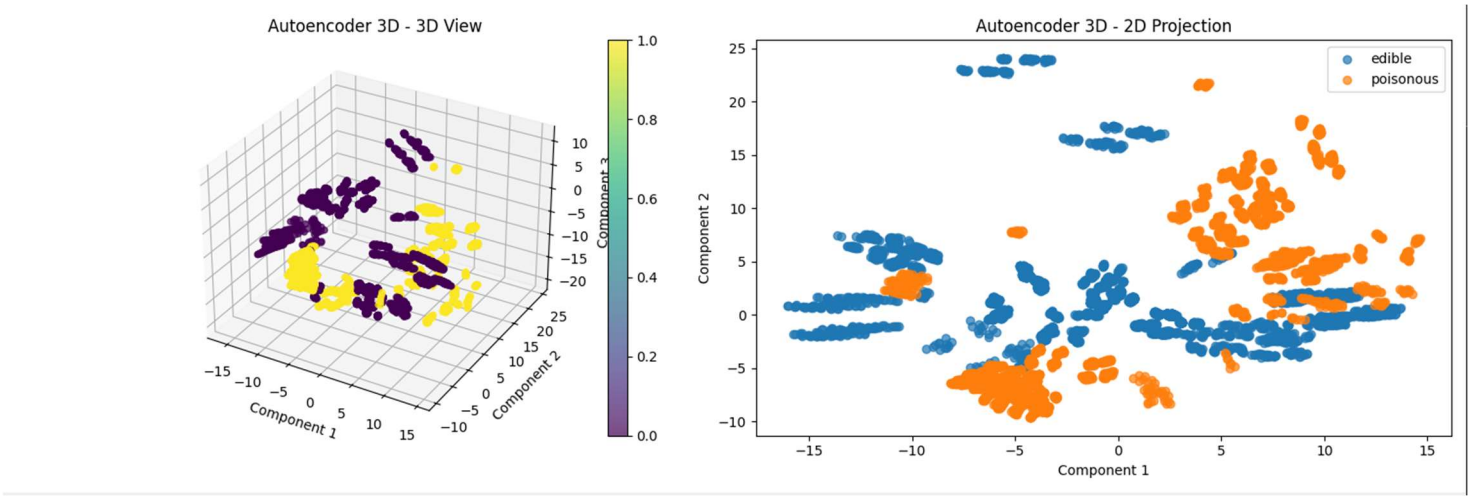
254/254 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 567us/step

ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ

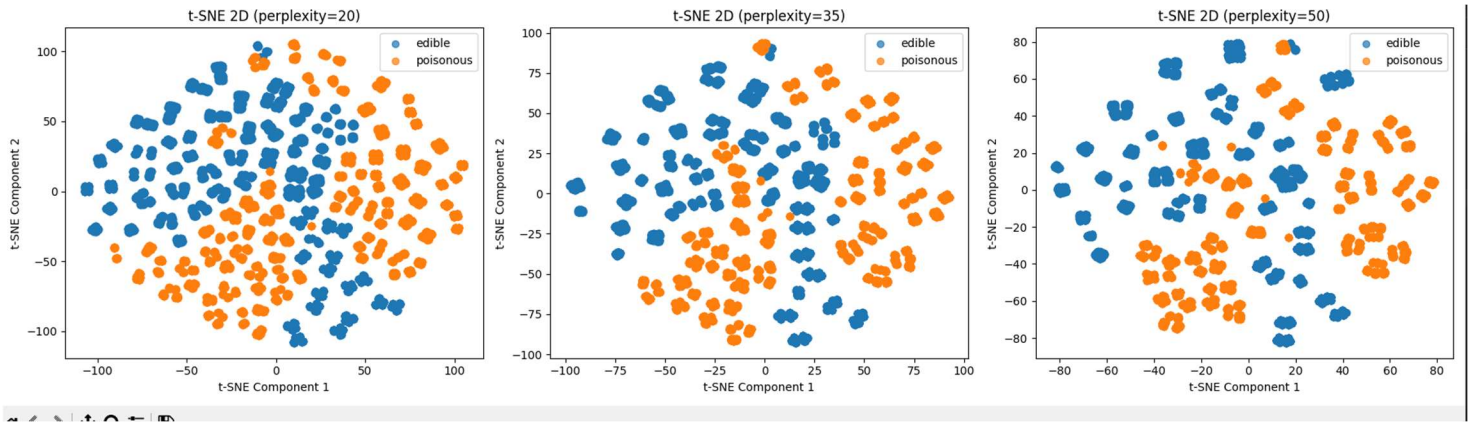
Визуализация автоэнкодера (2D):



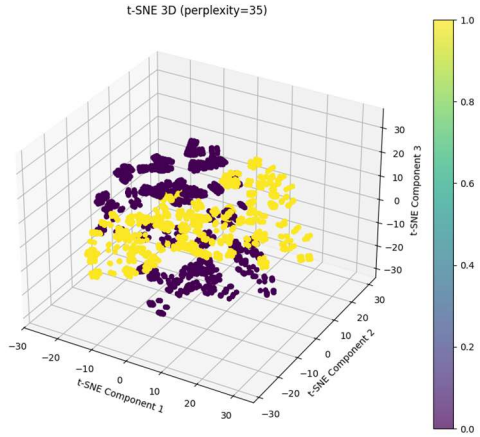
Визуализация автоэнкодера (3D):



МЕТОД t-SNE



t-SNE 3D с perplexity=35:



МЕТОД PCA

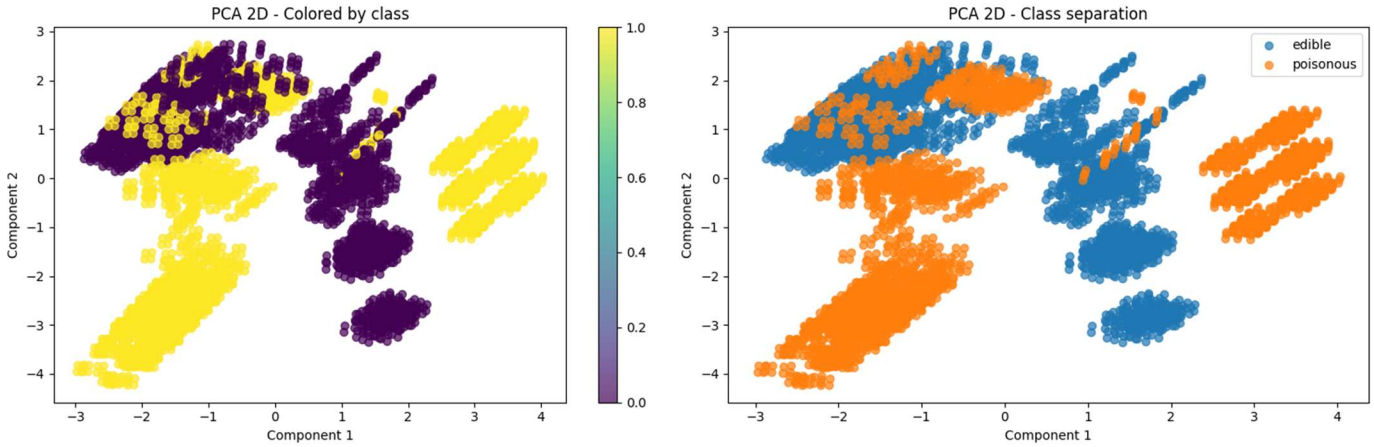
Объясненная дисперсия PCA (2 компоненты): [0.2037041 0.12412968]

Суммарная объясненная дисперсия: 0.3278337789616575

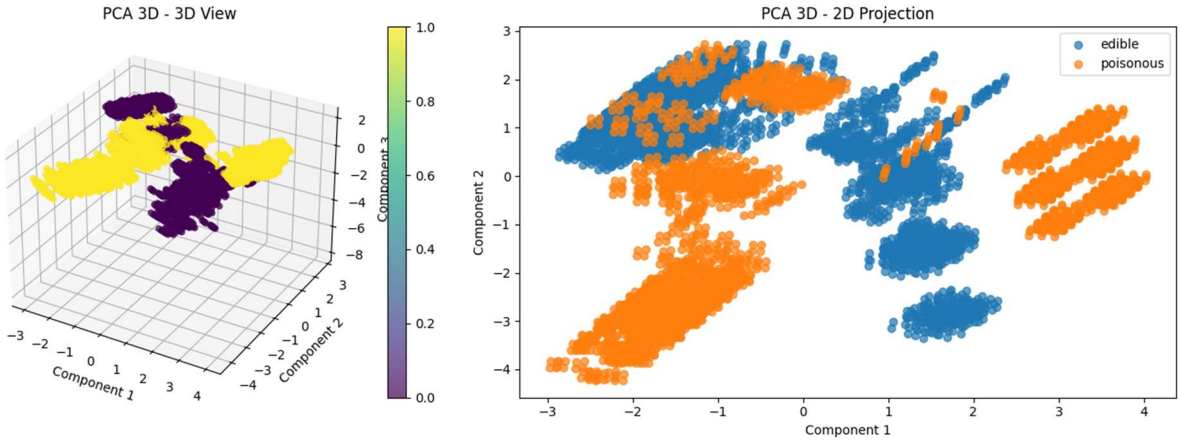
Объясненная дисперсия PCA (3 компоненты): [0.2037041 0.12412968 0.11071335]

Суммарная объясненная дисперсия: 0.43854712625072334

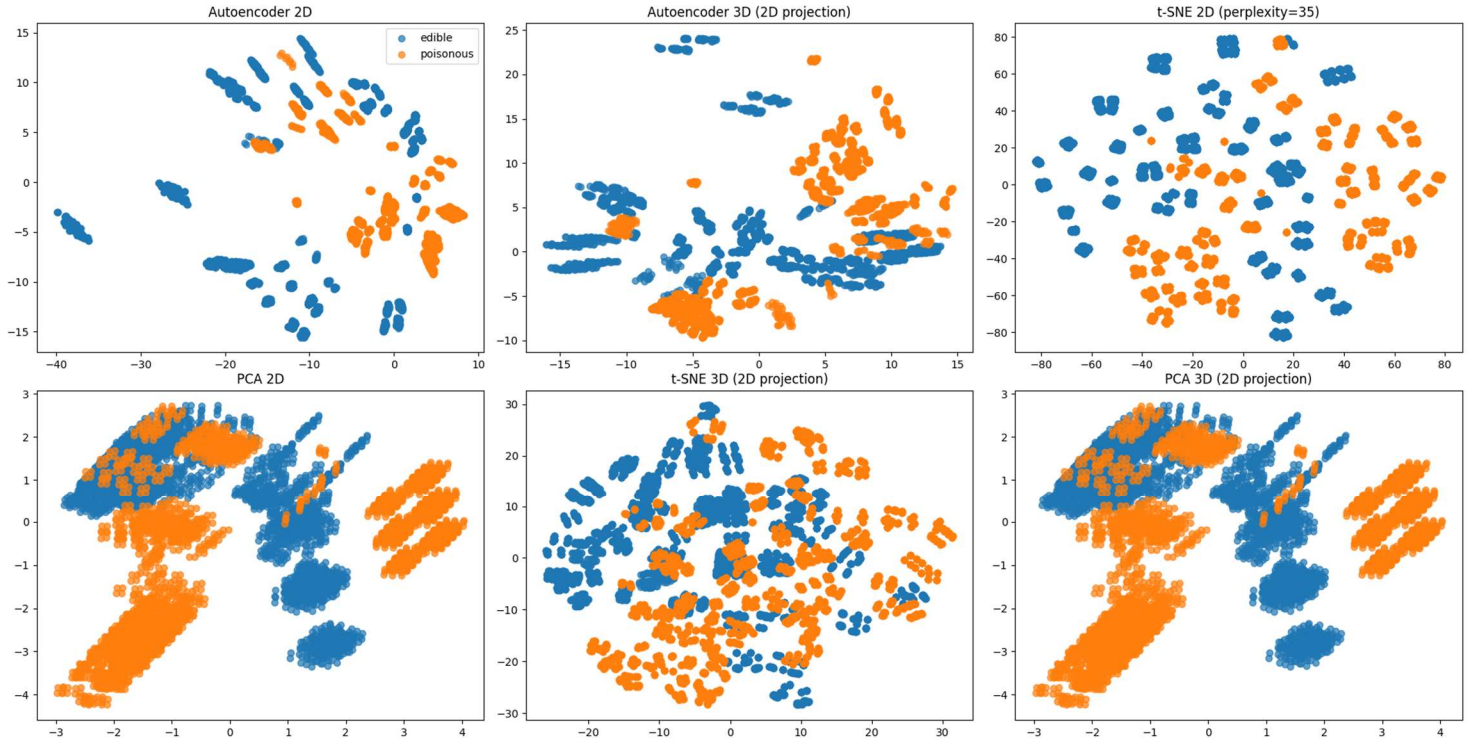
Визуализация PCA (2D):

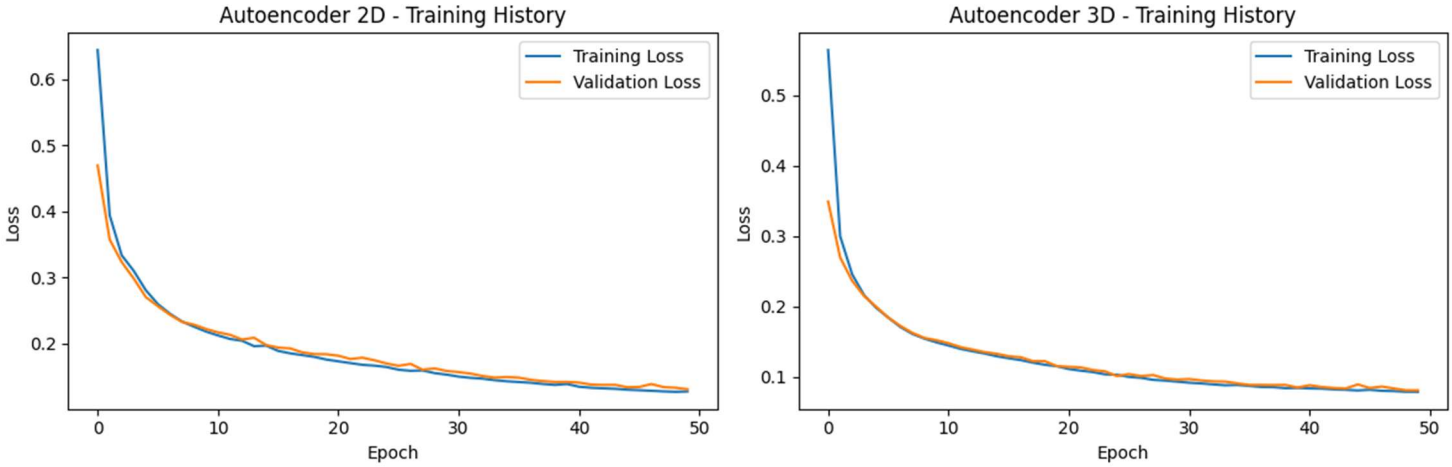


Визуализация PCA (3D):



СРАВНИТЕЛЬНЫЙ АНАЛИЗ





АВТОЭНКОДЕР:

- Способен извлекать нелинейные зависимости в данных
- Показывает хорошее разделение классов в скрытом пространстве
- Требует тщательной настройки архитектуры и параметров обучения

t-SNE:

- Лучше всего показывает локальные кластеры и структуры
- Чувствителен к параметру perplexity
- Визуализация более интуитивно понятна для анализа кластеров

PCA:

- Сохраняет глобальную структуру данных
- Объяснимая дисперсия: 0.328
- Быстрый и стабильный метод

СРАВНЕНИЕ:

- PCA: лучше для сохранения глобальной структуры
- t-SNE: лучше для визуализации локальных кластеров
- Autoencoder: универсальный метод, может учитывать нелинейности
- Для данного датасета все методы показывают хорошее разделение классов

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