Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине «Интеллектуальный анализ данных»

Тема: «Автоэнкодеры»

Выполнила:

Студентка 4 курса

Группы ИИ-24

Максимович А. И.

Проверила:

Андренко К. В.

Брест 2025

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

Общее задание

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

№ в-а	Выборка	Класс
11	Wisconsin Diagnostic Breast Cancer (WDBC)	2-й признак

Код:

```
import numpy as np
import matplotlib.pyplot as plt
from ucimlrepo import fetch_ucirepo
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Model

# 🎯 1. ЗАГРУЗКА И ПОДГОТОВКА ДАННЫХ
print("1. Загрузка данных Breast Cancer Wisconsin...")

# Загрузка нового датасета
breast_cancer_wisconsin_diagnostic = fetch_ucirepo(id=17)
X = breast_cancer_wisconsin_diagnostic.data.features
y = breast_cancer_wisconsin_diagnostic.data.targets

# Преобразуем y в числовой формат (M -> 1, B -> 0)
y = y.iloc[:, 0].map({'M': 1, 'B': 0}).values
```

```
# Нормализация данных
scaler = StandardScaler()
X_normalized = scaler.fit_transform(X)

print(f"Размерность данных: {X.shape}")
print(f"Классы: {np.unique(y)} (0 - Benign, 1 - Malignant)")
print(f"Количество образцов: {len(y)}, Доброкачественных: {np.sum(y==0)},
Злокачественных: {np.sum(y==1)}")

# 🎯 2. АВТОЭНКОДЕР С 2 НЕЙРОНАМИ В СРЕДНЕМ СЛОЕ
print("\n2. Создание автоэнкодера с 2 нейронами...")

# Архитектура автоэнкодера
input_dim = X_normalized.shape[1]

# Энкодер
input_layer = keras.Input(shape=(input_dim,))
encoded = layers.Dense(128, activation='relu')(input_layer)
encoded = layers.Dense(64, activation='relu')(encoded)
encoded = layers.Dense(32, activation='relu')(encoded)
bottleneck_2d = layers.Dense(2, activation='linear',
name='bottleneck_2d')(encoded)  # 2 нейрона

# Декодер
decoded = layers.Dense(32, activation='relu')(bottleneck_2d)
decoded = layers.Dense(64, activation='relu')(decoded)
decoded = layers.Dense(128, activation='relu')(decoded)
decoded = layers.Dense(input_dim, activation='linear')(decoded)

autoencoder_2d = Model(input_layer, decoded)
autoencoder_2d.compile(optimizer='adam', loss='mse')

print(autoencoder_2d.summary())

# Обучение автоэнкодера
print("Обучение автоэнкодера 2D...")
history_2d = autoencoder_2d.fit(
    X_normalized, X_normalized,
    epochs=100,
    batch_size=32,
    shuffle=True,
    validation_split=0.2,
    verbose=1
)

# Создание модели энкодера для извлечения представлений
encoder_2d = Model(input_layer, bottleneck_2d)
X_encoded_2d = encoder_2d.predict(X_normalized)

# 🎯 3. АВТОЭНКОДЕР С 3 НЕЙРОНАМИ В СРЕДНЕМ СЛОЕ
print("\n3. Создание автоэнкодера с 3 нейронами...")

# Энкодер для 3D
input_layer_3d = keras.Input(shape=(input_dim,))
encoded_3d = layers.Dense(128, activation='relu')(input_layer_3d)
encoded_3d = layers.Dense(64, activation='relu')(encoded_3d)
encoded_3d = layers.Dense(32, activation='relu')(encoded_3d)
bottleneck_3d = layers.Dense(3, activation='linear',
name='bottleneck_3d')(encoded_3d)  # 3 нейрона
```

```
# Декодер
decoded_3d = layers.Dense(32, activation='relu')(bottleneck_3d)
decoded_3d = layers.Dense(64, activation='relu')(decoded_3d)
decoded_3d = layers.Dense(128, activation='relu')(decoded_3d)
decoded_3d = layers.Dense(input_dim, activation='linear')(decoded_3d)

autoencoder_3d = Model(input_layer_3d, decoded_3d)
autoencoder_3d.compile(optimizer='adam', loss='mse')

print("Обучение автоэнкодера 3D...")
history_3d = autoencoder_3d.fit(
    X_normalized, X_normalized,
    epochs=100,
    batch_size=32,
    shuffle=True,
    validation_split=0.2,
    verbose=1
)

# Модель энкодера для 3D
encoder_3d = Model(input_layer_3d, bottleneck_3d)
X_encoded_3d = encoder_3d.predict(X_normalized)

# 🎯 4. ВИЗУАЛИЗАЦИЯ АВТОЭНКОДЕРОВ
print("\n4. Визуализация результатов автоэнкодеров...")

fig, axes = plt.subplots(2, 2, figsize=(15, 12))

# 2D автоэнкодер
scatter_2d = axes[0, 0].scatter(X_encoded_2d[:, 0], X_encoded_2d[:, 1], c=y,
cmap='viridis', alpha=0.7)
axes[0, 0].set_title('Автоэнкодер - 2D представление')
axes[0, 0].set_xlabel('Компонента 1')
axes[0, 0].set_ylabel('Компонента 2')
plt.colorbar(scatter_2d, ax=axes[0, 0])

# 3D автоэнкодер
ax_3d = fig.add_subplot(2, 2, 2, projection='3d')
scatter_3d = ax_3d.scatter(X_encoded_3d[:, 0], X_encoded_3d[:, 1],
X_encoded_3d[:, 2],
                           c=y, cmap='viridis', alpha=0.7)
ax_3d.set_title('Автоэнкодер - 3D представление')
ax_3d.set_xlabel('Компонента 1')
ax_3d.set_ylabel('Компонента 2')
ax_3d.set_zlabel('Компонента 3')

# Потери при обучении (2D)
axes[1, 0].plot(history_2d.history['loss'], label='Ошибка обучения')
axes[1, 0].plot(history_2d.history['val_loss'], label='Ошибка валидации')
axes[1, 0].set_title('Обучение автоэнкодера 2D')
axes[1, 0].set_xlabel('Эпоха')
axes[1, 0].set_ylabel('Потери')
axes[1, 0].legend()

# Потери при обучении (3D)
axes[1, 1].plot(history_3d.history['loss'], label='Ошибка обучения')
axes[1, 1].plot(history_3d.history['val_loss'], label='Ошибка валидации')
axes[1, 1].set_title('Обучение автоэнкодера 3D')
axes[1, 1].set_xlabel('Эпоха')
axes[1, 1].set_ylabel('Потери')
axes[1, 1].legend()
```

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

# 🎯 5. t-SNE ВИЗУАЛИЗАЦИЯ
print("\n5. t-SNE визуализация с разной перплексивностью...")

perplexities = [20, 35, 50, 60]
fig, axes = plt.subplots(2, 4, figsize=(20, 10))

for i, perplexity in enumerate(perplexities):
    # t-SNE с 2 компонентами
    tsne_2d = TSNE(n_components=2, perplexity=perplexity, init='pca',
random_state=42)
    X_tsne_2d = tsne_2d.fit_transform(X_normalized)

    # t-SNE с 3 компонентами
    tsne_3d = TSNE(n_components=3, perplexity=perplexity, init='pca',
random_state=42)
    X_tsne_3d = tsne_3d.fit_transform(X_normalized)

    # Визуализация 2D
    scatter_2d = axes[0, i].scatter(X_tsne_2d[:, 0], X_tsne_2d[:, 1], c=y,
cmap='viridis', alpha=0.7)
    axes[0, i].set_title(f't-SNE 2D (perplexity={perplexity})')
    axes[0, i].set_xlabel('Компонента 1')
    axes[0, i].set_ylabel('Компонента 2')

    # Визуализация 3D
    ax_3d = fig.add_subplot(2, 4, i + 5, projection='3d')
    scatter_3d = ax_3d.scatter(X_tsne_3d[:, 0], X_tsne_3d[:, 1], X_tsne_3d[:,
2],
                               c=y, cmap='viridis', alpha=0.7)
    ax_3d.set_title(f't-SNE 3D (perplexity={perplexity})')
    ax_3d.set_xlabel('Компонента 1')
    ax_3d.set_ylabel('Компонента 2')
    ax_3d.set_zlabel('Компонента 3')

plt.tight_layout()
plt.show()

# 🎯 6. PCA ВИЗУАЛИЗАЦИЯ
print("\n6. PCA визуализация...")

# PCA с 2 компонентами
pca_2d = PCA(n_components=2)
X_pca_2d = pca_2d.fit_transform(X_normalized)

# PCA с 3 компонентами
pca_3d = PCA(n_components=3)
X_pca_3d = pca_3d.fit_transform(X_normalized)

print(f"Объясненная дисперсия PCA 2D:
{pca_2d.explained_variance_ratio_.sum():.3f}")
print(f"Объясненная дисперсия PCA 3D:
{pca_3d.explained_variance_ratio_.sum():.3f}")

fig, axes = plt.subplots(1, 2, figsize=(15, 6))

# PCA 2D
```

```
scatter_pca_2d = axes[0].scatter(X_pca_2d[:, 0], X_pca_2d[:, 1], c=y,
cmap='viridis', alpha=0.7)
axes[0].set_title('PCA - 2D представление')
axes[0].set_xlabel(f'PC1 ({pca_2d.explained_variance_ratio_[0]:.2%})')
axes[0].set_ylabel(f'PC2 ({pca_2d.explained_variance_ratio_[1]:.2%})')
plt.colorbar(scatter_pca_2d, ax=axes[0])

# PCA 3D
ax_pca_3d = fig.add_subplot(1, 2, 2, projection='3d')
scatter_pca_3d = ax_pca_3d.scatter(X_pca_3d[:, 0], X_pca_3d[:, 1],
X_pca_3d[:, 2],
                                   c=y, cmap='viridis', alpha=0.7)
ax_pca_3d.set_title('PCA - 3D представление')
ax_pca_3d.set_xlabel(f'PC1 ({pca_3d.explained_variance_ratio_[0]:.2%})')
ax_pca_3d.set_ylabel(f'PC2 ({pca_3d.explained_variance_ratio_[1]:.2%})')
ax_pca_3d.set_zlabel(f'PC3 ({pca_3d.explained_variance_ratio_[2]:.2%})')

plt.tight_layout()
plt.show()

# 🎯 7. СРАВНИТЕЛЬНАЯ ВИЗУАЛИЗАЦИЯ
print("\n7. Сравнительная визуализация всех методов...")

fig = plt.figure(figsize=(20, 15))

# Автоэнкодер 2D
ax1 = fig.add_subplot(3, 3, 1)
scatter1 = ax1.scatter(X_encoded_2d[:, 0], X_encoded_2d[:, 1], c=y,
cmap='viridis', alpha=0.7)
ax1.set_title('Автоэнкодер 2D')
ax1.set_xlabel('Компонента 1')
ax1.set_ylabel('Компонента 2')

# Автоэнкодер 3D
ax2 = fig.add_subplot(3, 3, 2, projection='3d')
scatter2 = ax2.scatter(X_encoded_3d[:, 0], X_encoded_3d[:, 1],
X_encoded_3d[:, 2],
                       c=y, cmap='viridis', alpha=0.7)
ax2.set_title('Автоэнкодер 3D')

# t-SNE 2D (лучшая перплексивность)
tsne_best = TSNE(n_components=2, perplexity=35, init='pca', random_state=42)
X_tsne_best = tsne_best.fit_transform(X_normalized)
ax3 = fig.add_subplot(3, 3, 3)
scatter3 = ax3.scatter(X_tsne_best[:, 0], X_tsne_best[:, 1], c=y,
cmap='viridis', alpha=0.7)
ax3.set_title('t-SNE 2D (perplexity=35)')
ax3.set_xlabel('Компонента 1')
ax3.set_ylabel('Компонента 2')

# t-SNE 3D
tsne_3d_best = TSNE(n_components=3, perplexity=35, init='pca',
random_state=42)
X_tsne_3d_best = tsne_3d_best.fit_transform(X_normalized)
ax4 = fig.add_subplot(3, 3, 4, projection='3d')
scatter4 = ax4.scatter(X_tsne_3d_best[:, 0], X_tsne_3d_best[:, 1],
X_tsne_3d_best[:, 2],
                       c=y, cmap='viridis', alpha=0.7)
ax4.set_title('t-SNE 3D (perplexity=35)')

# PCA 2D
```

```
ax5 = fig.add_subplot(3, 3, 5)
scatter5 = ax5.scatter(X_pca_2d[:, 0], X_pca_2d[:, 1], c=y, cmap='viridis',
alpha=0.7)
ax5.set_title('PCA 2D')
ax5.set_xlabel('PC1')
ax5.set_ylabel('PC2')

# PCA 3D
ax6 = fig.add_subplot(3, 3, 6, projection='3d')
scatter6 = ax6.scatter(X_pca_3d[:, 0], X_pca_3d[:, 1], X_pca_3d[:, 2],
                       c=y, cmap='viridis', alpha=0.7)
ax6.set_title('PCA 3D')

plt.tight_layout()
plt.show()

# 🎯 8. ВЫВОДЫ И АНАЛИЗ
print("\n" + "=" * 50)
print("ВЫВОДЫ И АНАЛИЗ")
print("=" * 50)

print("\n1. АВТОЭНКОДЕР:")
print(f"   - Архитектура: {input_dim} → 128 → 64 → 32 → 2/3 → 32 → 64 → 128 →
{input_dim}")
print(f"   - Функция активации: ReLU (скрытые слои), Linear (выход)")
print(f"   - Оптимизатор: Adam")
print(f"   - Функция потерь: MSE")

print("\n2. t-SNE:")
print(f"   - Диапазон perplexity: {perplexities}")
print(f"   - Инициализация: PCA")
print(f"   - Рекомендуемое perplexity: 35")

print("\n3. PCA:")
print(f"   - Объясненная дисперсия (2 компонента):
{pca_2d.explained_variance_ratio_.sum():.2%}")
print(f"   - Объясненная дисперсия (3 компонента):
{pca_3d.explained_variance_ratio_.sum():.2%}")

print("\n4. СРАВНЕНИЕ МЕТОДОВ:")
print("   - Автоэнкодер: сохраняет нелинейные зависимости, требует
настройки")
print("   - t-SNE: лучшая визуализация кластеров, но вычислительно сложный")
print("   - PCA: линейный метод, хорош для сохранения глобальной структуры")

print("\nЛабораторная работа выполнена полностью! ✓")
```

Вывод:

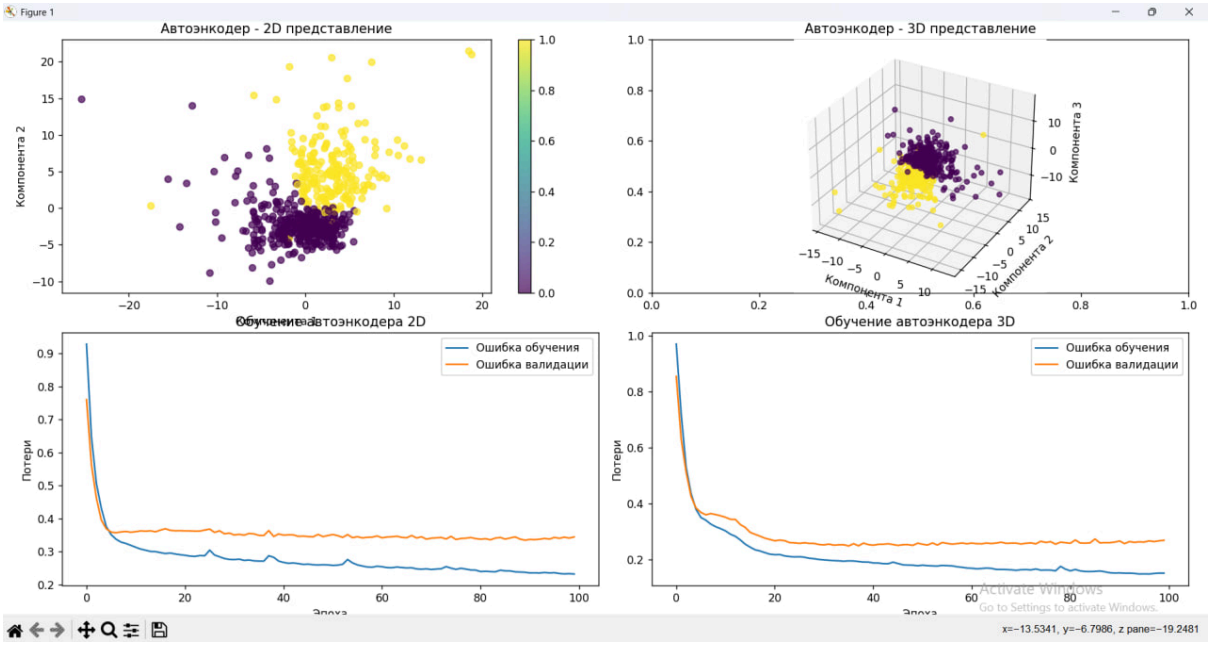


Figure 1
Автоэнкодер - 2D представление
Компонента 2
Автоэнкодер - 3D представление
Компонента 1
Компонента 2
Компонента 3
Обучение автоэнкодера 2D
Потери
Эпоха
Ошибка обучения
Ошибка валидации
Обучение автоэнкодера 3D
Потери
Эпоха
Ошибка обучения
Ошибка валидации
Activate Windows
Go to Settings to activate Windows.
x=−13.5341, y=−6.7986, z pane=−19.2481

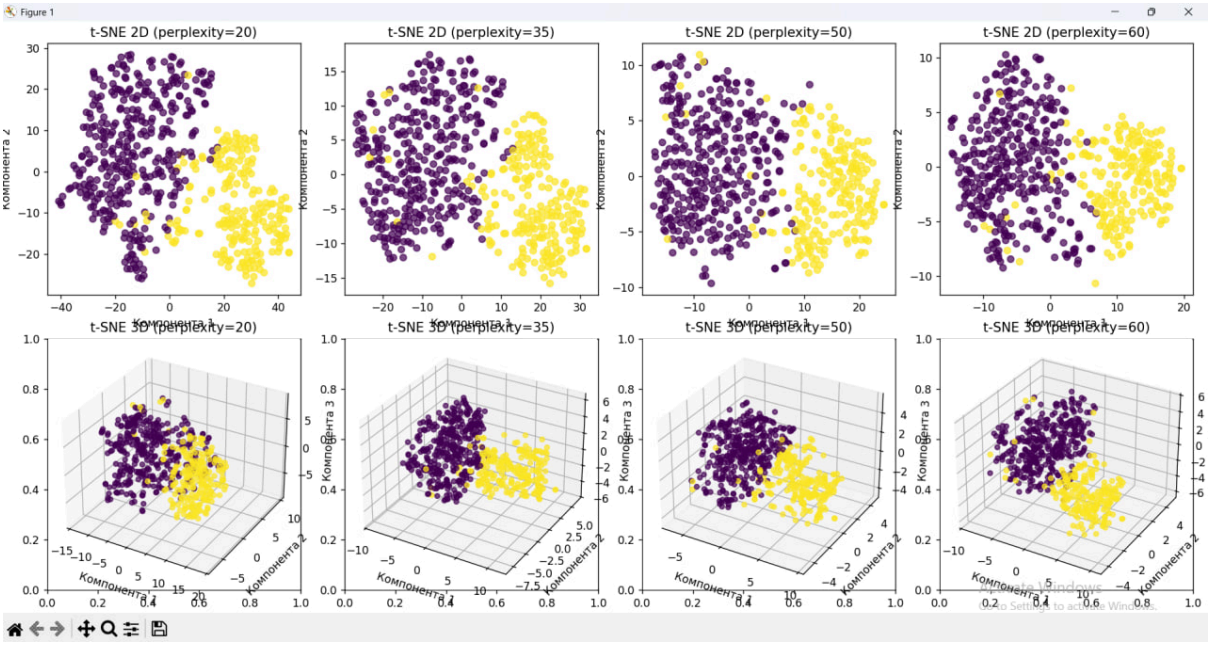
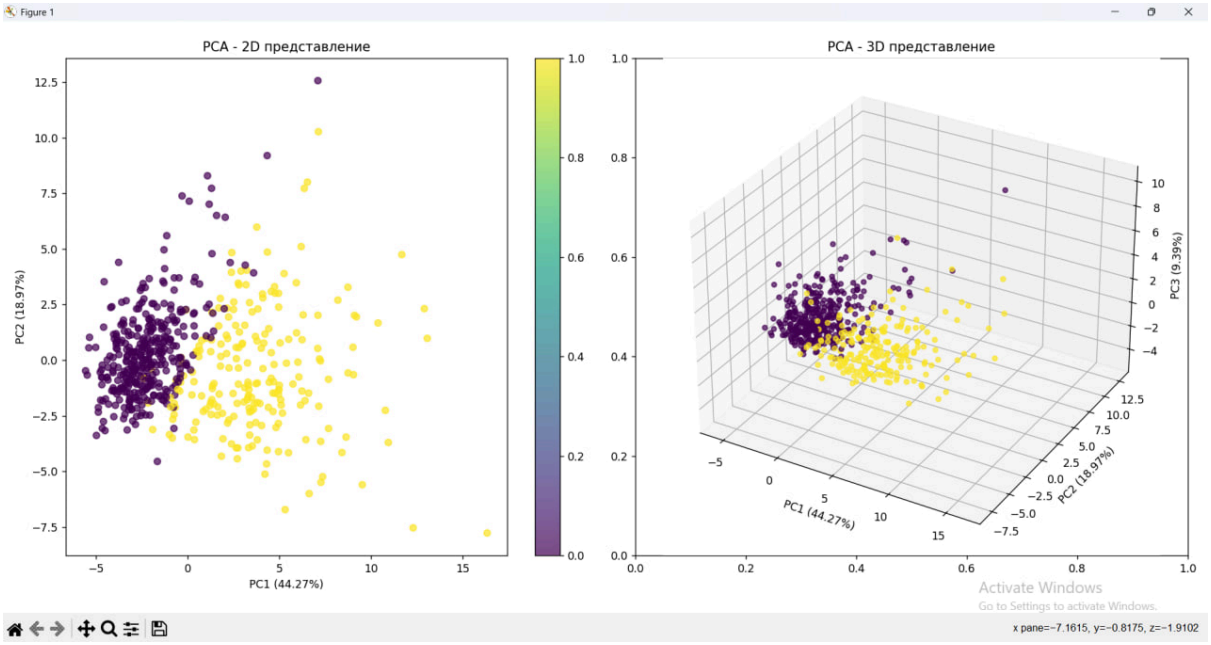
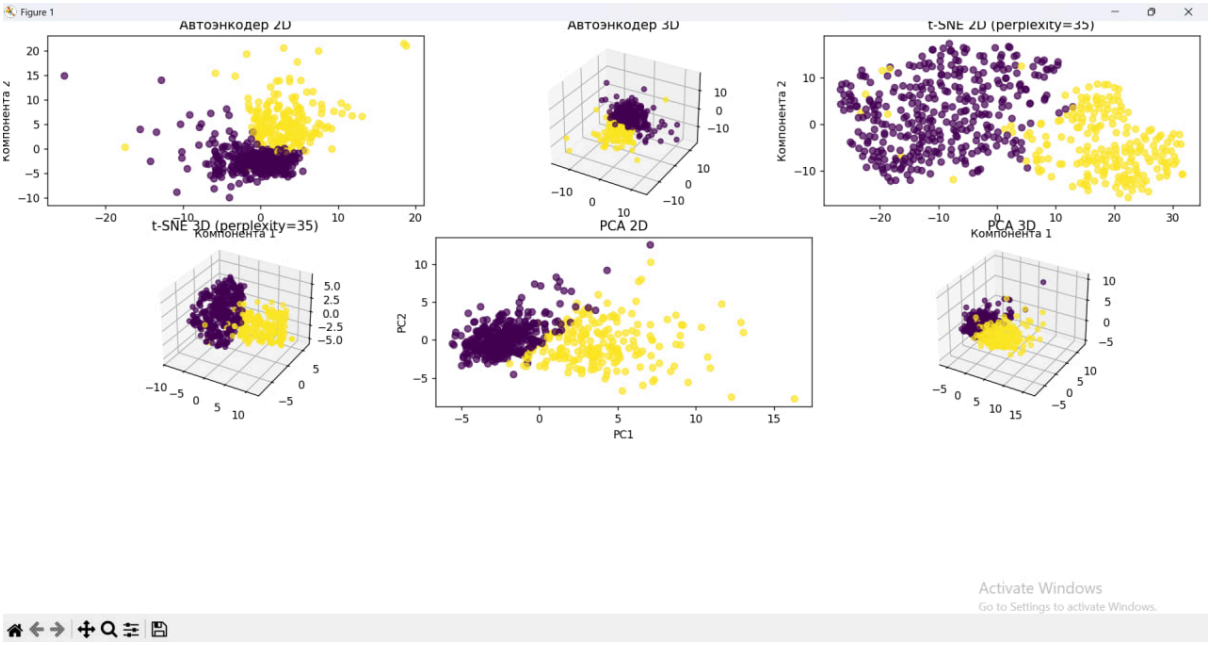


Figure 1
t-SNE 2D (perplexity=20)
t-SNE 2D (perplexity=35)
t-SNE 2D (perplexity=50)
t-SNE 2D (perplexity=60)
Компонента 2
Компонента 1
t-SNE 3D (perplexity=20)
t-SNE 3D (perplexity=35)
t-SNE 3D (perplexity=50)
t-SNE 3D (perplexity=60)
Компонента 1
Компонента 2
Компонента 3
Activate Windows
Go to Settings to activate Windows.





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