

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

Лабораторная работа №2  
По дисциплине: «Языковые процессы интеллектуальных систем»  
Тема: «Автоэнкодеры»

Выполнил:  
Студент 4 курса  
Группы ИИ-23  
Романюк А. П.  
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**

9	<u>Wine Quality (white)</u>	quality
---	-----------------------------	---------

**Код программы:**

```
import pandas as pd

import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Dense, Dropout, BatchNormalization
from tensorflow.keras.models import Model
from sklearn.preprocessing import MinMaxScaler

df = pd.read_csv('winequality-white.csv', sep=";")

labels = df['quality']
data = df.drop(columns=['quality'])

scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)

input_dim = data_scaled.shape[1]
encoding_dim = 2

input_layer = Input(shape=(input_dim,))
x = Dense(256, activation='relu')(input_layer)
x = BatchNormalization()(x)
x = Dropout(0.3)(x)

x = Dense(128, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.2)(x)

x = Dense(64, activation='relu')(x)
```

```

encoded = Dense(encoding_dim, activation='linear')(x)

x = Dense(64, activation='relu')(encoded) x
= BatchNormalization()(x)

x = Dense(128, activation='relu')(x) x
= Dropout(0.2)(x)

x = Dense(256, activation='relu')(x)
decoded = Dense(input_dim, activation='sigmoid')(x)

autoencoder_2d = Model(input_layer, decoded)
encoder_2d = Model(input_layer, encoded)

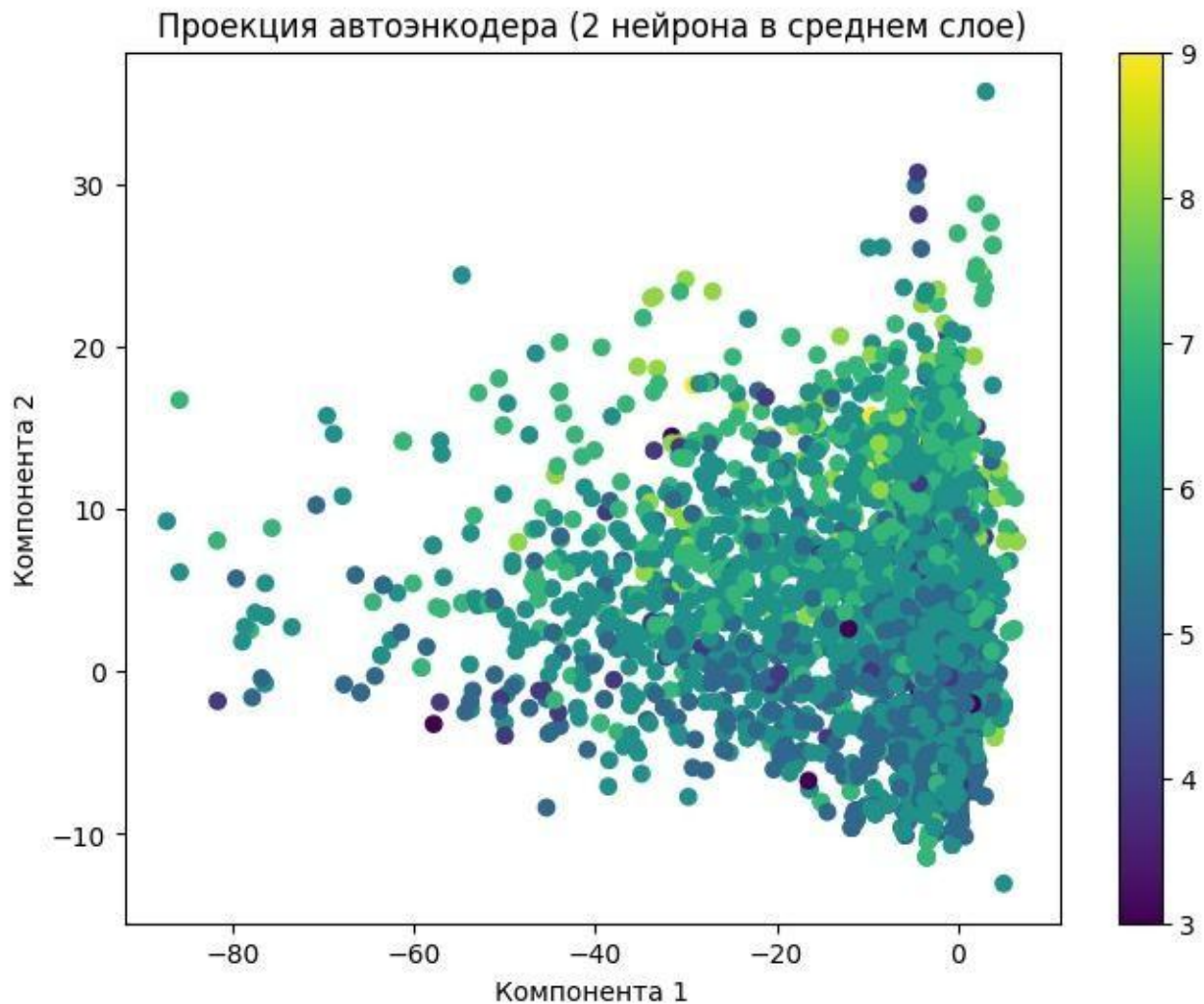
autoencoder_2d.compile(optimizer='adam', loss='mse')

history = autoencoder_2d.fit(
    data_scaled, data_scaled,
    epochs=100,
    batch_size=32,
    shuffle=True, verbose=1,
    validation_split=0.1
)

encoded_data_2d = encoder_2d.predict(data_scaled)

plt.figure(figsize=(8, 6))
scatter = plt.scatter(encoded_data_2d[:, 0], encoded_data_2d[:, 1], c=labels, cmap='viridis')
plt.colorbar(scatter) plt.xlabel('Компонента 1') plt.ylabel('Компонента 2')
plt.title('Проекция автоэнкодера (2 нейрона в среднем слое)')
plt.show()

```



```

input_dim = data_scaled.shape[1] encoding_dim = 3
input_layer = Input(shape=(input_dim,)) x =
Dense(256, activation='relu')(input_layer) x =
BatchNormalization()(x) x = Dropout(0.3)(x)

x = Dense(128, activation='relu')(x)
x = BatchNormalization()(x) x =
Dropout(0.2)(x)

x = Dense(64, activation='relu')(x)
encoded = Dense(encoding_dim, activation='linear')(x)

x = Dense(64, activation='relu')(encoded) x
= BatchNormalization()(x)

x = Dense(128, activation='relu')(x) x
= Dropout(0.2)(x)

x = Dense(256, activation='relu')(x)
decoded = Dense(input_dim, activation='sigmoid')(x)

autoencoder = Model(input_layer, decoded)
encoder = Model(input_layer, encoded)

```

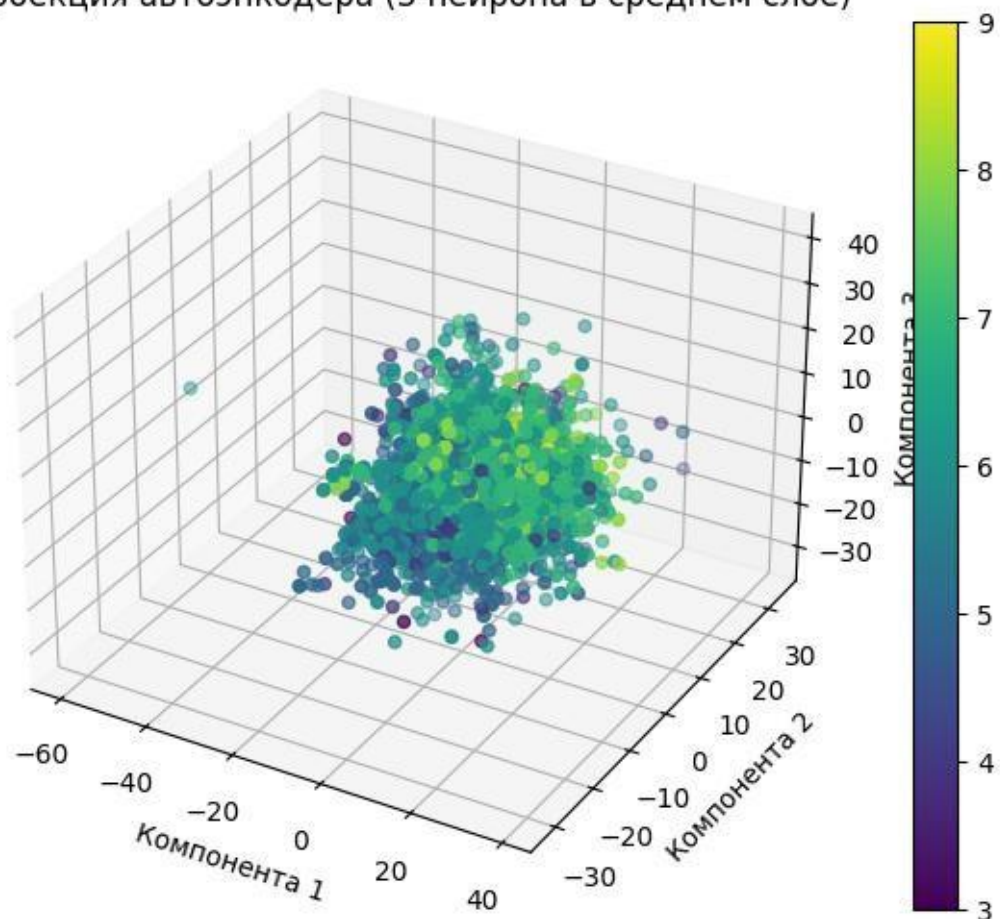
```
autoencoder.compile(optimizer='adam', loss='mse')
```

```
history = autoencoder.fit(  
    data_scaled, data_scaled,  
    epochs=100,  
    batch_size=32,  
    shuffle=True, verbose=1,  
    validation_split=0.1  
)
```

```
encoded_data_3d = encoder.predict(data_scaled)
```

```
fig = plt.figure(figsize=(8, 6)) ax =  
fig.add_subplot(111, projection='3d')  
scatter = ax.scatter(encoded_data_3d[:, 0], encoded_data_3d[:, 1], encoded_data_3d[:, 2], c=labels,  
    cmap='viridis') fig.colorbar(scatter) ax.set_xlabel('Компонента 1') ax.set_ylabel('Компонента 2')  
ax.set_zlabel('Компонента 3')  
ax.set_title('Проекция автоэнкодера (3 нейрона в среднем слое)')  
plt.show()
```

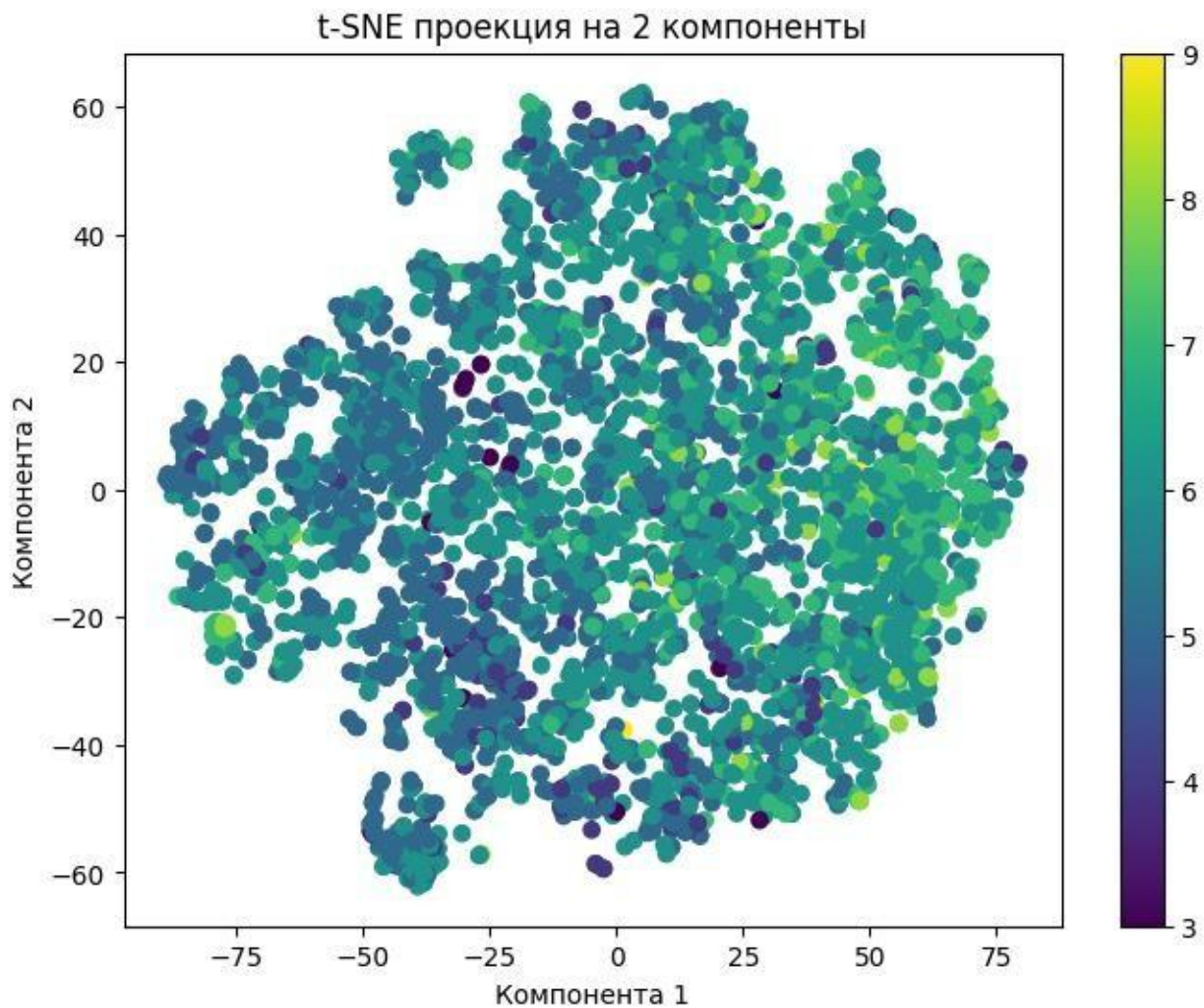
Проекция автоэнкодера (3 нейрона в среднем слое)



from sklearn.manifold import TSNE

```
tsne_2d = TSNE(n_components=2, perplexity=40)  
tsne_proj_2d = tsne_2d.fit_transform(data_scaled)
```

```
plt.figure(figsize=(8, 6))
scatter = plt.scatter(tsne_proj_2d[:, 0], tsne_proj_2d[:, 1], c=labels, cmap='viridis')
plt.colorbar(scatter) plt.xlabel('Компонента 1') plt.ylabel('Компонента 2')
plt.title('t-SNE проекция на 2 компоненты') plt.show()
```

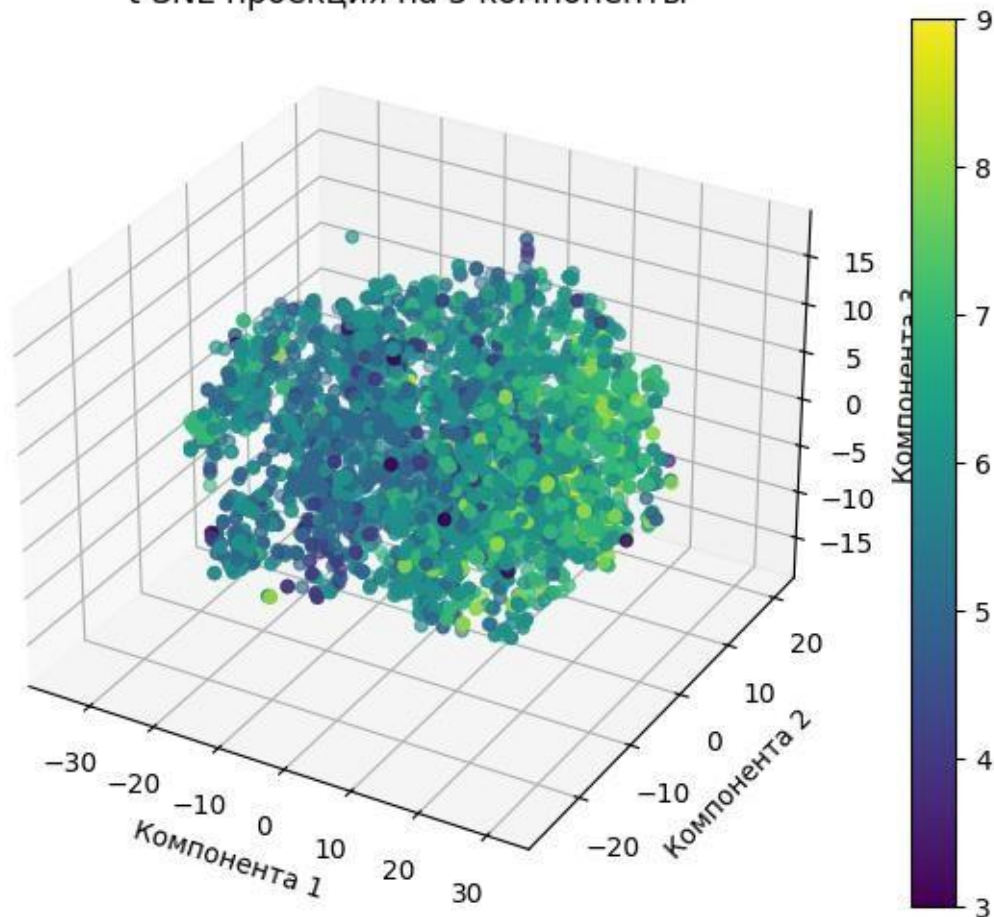


```
tsne_3d = TSNE(n_components=3, perplexity=40) tsne_proj_3d = tsne_3d.fit_transform(data_scaled)
```

```
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(tsne_proj_3d[:, 0], tsne_proj_3d[:, 1], tsne_proj_3d[:, 2], c=labels, cmap='viridis')
fig.colorbar(scatter) ax.set_xlabel('Компонента 1') ax.set_ylabel('Компонента 2')
ax.set_zlabel('Компонента 3')
ax.set_title('t-SNE проекция на 3 компоненты') plt.show()
```



t-SNE проекция на 3 компоненты



```
data_centred = data_scaled - data_scaled.mean() cov_matrix = np.cov(data_centred,
rowvar=False) eig_values, eig_vectors = np.linalg.eig(cov_matrix) idx =
np.argsort(eig_values)[::-1]
eig_vectors = eig_vectors[:,idx]
```

```
data_2d = data_centred.dot(eig_vectors[:, :2]) data_3d
= data_centred.dot(eig_vectors[:, :3])
```

```
plt.figure(figsize=(8, 6))
scatter = plt.scatter(data_2d[:, 0], data_2d[:, 1], c=labels, cmap='viridis')
plt.colorbar(scatter)
plt.xlabel('Первая главная компонента') plt.ylabel('Вторая
главная компонента')
plt.title('PCA проекция на плоскость первых двух компонент') plt.show()
```

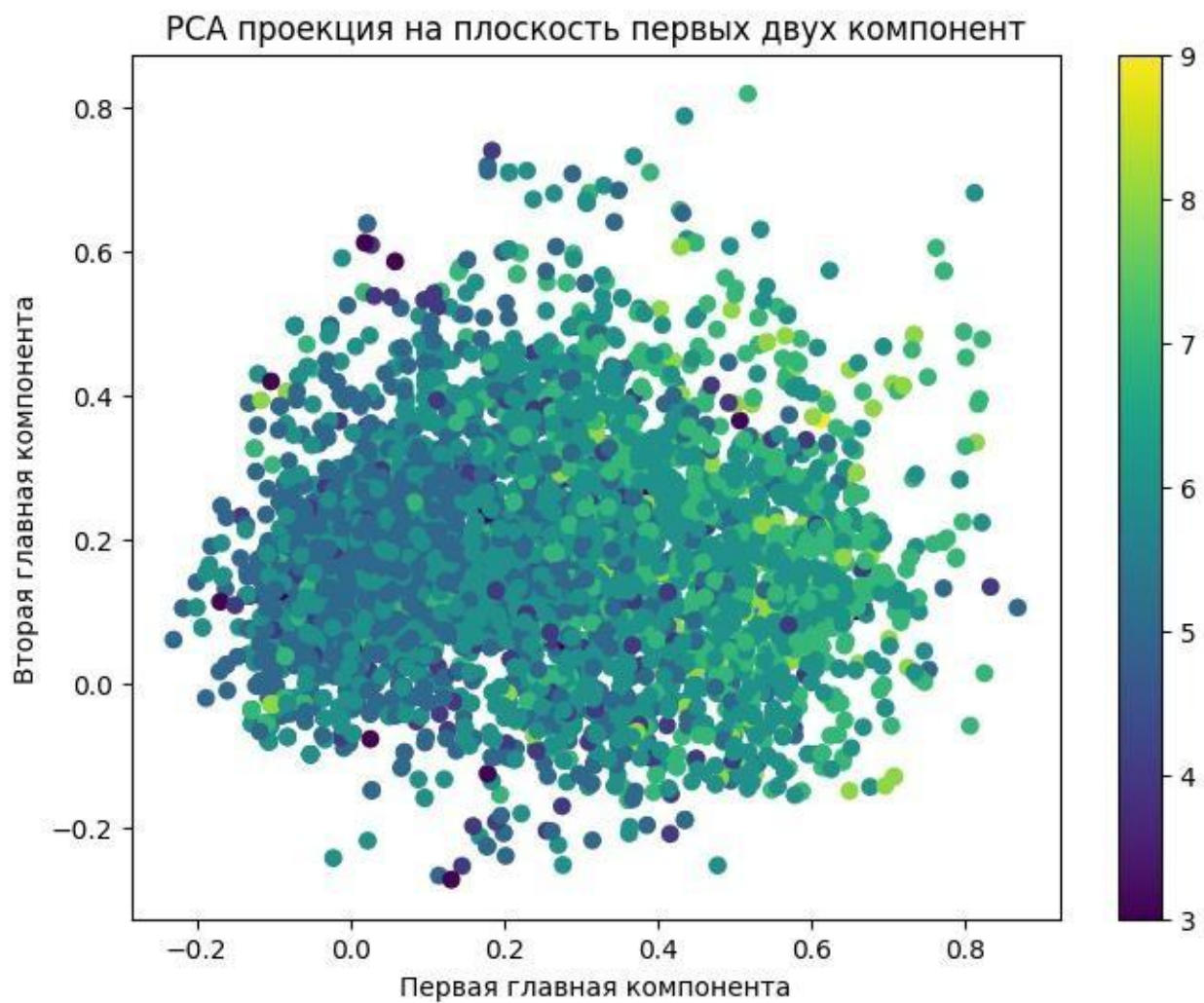
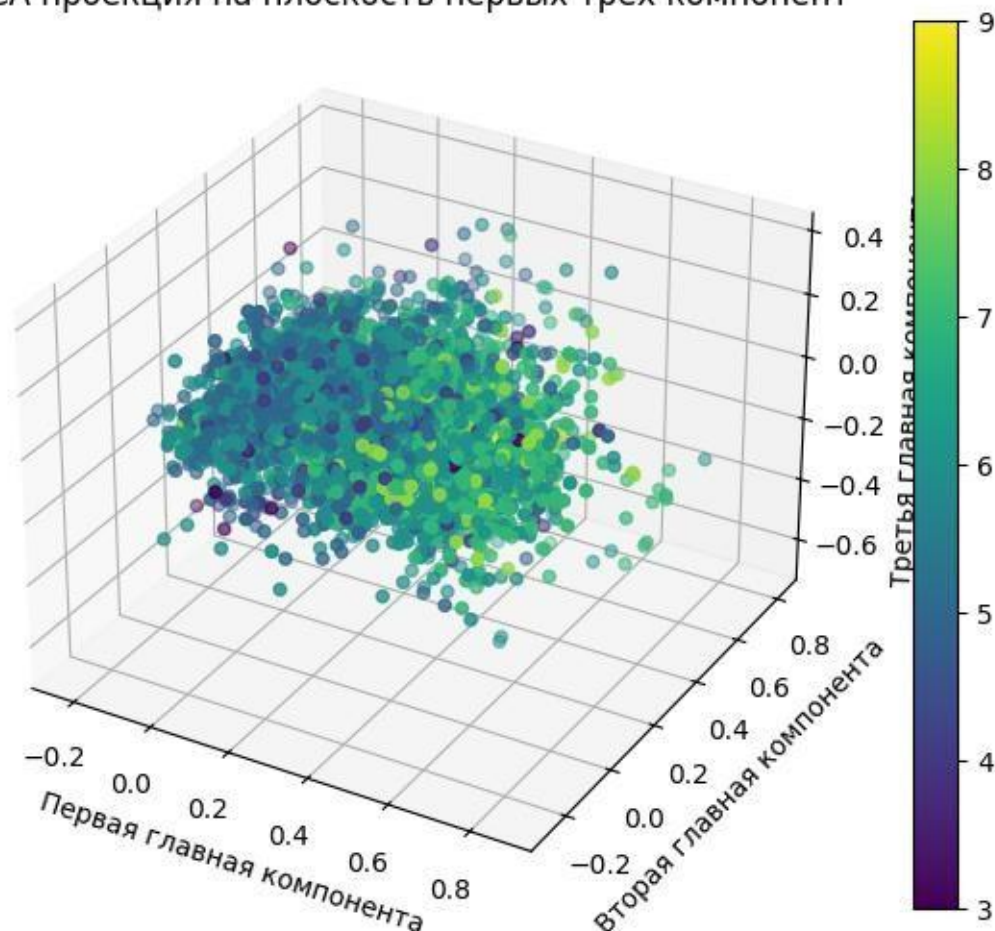


fig = plt.figure(figsize=(8, 6))

```
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(data_3d[:, 0], data_3d[:, 1], data_3d[:, 2], c=labels, cmap='viridis') fig.colorbar(scatter)
ax.set_xlabel('Первая главная компонента')
ax.set_ylabel('Вторая главная компонента') ax.set_zlabel('Третья
главная компонента')
ax.set_title('PCA проекция на плоскость первых трёх компонент')
plt.show()
```



## РСА проекция на плоскость первых трёх компонент



По результатам визуализации видно, что метод **t-SNE** наиболее эффективно выделяет кластеры экземпляров разных классов качества. Автоэнкодер частично справляется с разделением, однако границы между классами остаются размытыми. Метод **РСА** даёт наименее выраженное разделение, так как он линейный и учитывает только максимальную дисперсию данных, а не их классовую структуру.

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