|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра прикладной математики (ПМ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Технологии и инструментарий анализа больших данных»

**Практическая работа № 2**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИКБО-37-22, Гущин К. А.* | (подпись) | |
| Проверил | *Солтан А.А.* | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_2025 г. | |  | |

Москва 2025 г.

1.

import pandas as pd  
data = pd.read\_csv('pyramids.csv')

2.

print(data.info())  
print(data.head())  
print(data.isnull().sum())  
data = data.dropna()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 62 entries, 0 to 61

Data columns (total 16 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Pharaoh 62 non-null object

1 Ancient name 62 non-null object

2 Modern name 62 non-null object

3 Dynasty 62 non-null int64

4 Site 62 non-null object

5 Base1 (m) 62 non-null float64

6 Base2 (m) 62 non-null float64

7 Height (m) 53 non-null float64

8 Slope (dec degr) 43 non-null float64

9 Volume (cu.m) 33 non-null object

10 Latitude 62 non-null float64

11 Longitude 62 non-null float64

12 Type 62 non-null object

13 Lepsius 44 non-null object

14 Material 61 non-null object

15 Comment 21 non-null object

dtypes: float64(6), int64(1), object(9)

memory usage: 7.9+ KB

None

Pharaoh Ancient name Modern name Dynasty Site \

0 Djoser Hiemlender Step Pyramid of Djoser 3 Saqqara

1 Sekhemkhet - Buried Pyramid 3 Saqqara

2 Khaba (?) - Layer Pyramid 3 Zawyet el'Aryan

3 Huni - Pyramid Lepsius I 3 Abu Rawash

4 Huni (?) - Pyramid of Athribis 3 Athribis

Base1 (m) Base2 (m) Height (m) Slope (dec degr) Volume (cu.m) \

0 121.0 109.0 60.0 NaN 330.4

1 120.0 120.0 70.0 NaN 33.6

2 84.0 84.0 40.0 NaN 47.04

3 215.0 215.0 145.0 NaN NaN

4 20.0 20.0 16.0 50.0 NaN

Latitude Longitude Type Lepsius Material Comment

0 29.871215 31.216644 Step XXXII Mudbrick NaN

1 29.866000 31.213000 Step NaN Mudbrick (unfinished)

2 29.932820 31.161262 Step XIV Mudbrick (unfinished)

3 30.040833 31.094444 Step I Mudbrick totally ruined

4 30.470556 31.188056 Step NaN Mudbrick NaN

Pharaoh 0

Ancient name 0

Modern name 0

Dynasty 0

Site 0

Base1 (m) 0

Base2 (m) 0

Height (m) 9

Slope (dec degr) 19

Volume (cu.m) 29

Latitude 0

Longitude 0

Type 0

Lepsius 18

Material 1

Comment 41

dtype: int64

3.

|  |
| --- |
| import plotly.graph\_objects as go import plotly.express as px  fig = go.Figure(  go.Bar(  x=data['Pharaoh'],  y=data['Height (m)'],  marker=dict(  color=data['Height (m)'],  coloraxis="coloraxis"  )  ) ) fig.update\_layout(  title=dict(text='Высота пирамид по фараонам', x=0.5, font=dicimport plotly.graph\_objects as go import plotly.express as px  fig = go.Figure(  go.Bar(  x=data['Pharaoh'],  y=data['Height (m)'],  marker=dict(  color=data['Height (m)'],  coloraxis="coloraxis"  )  ) ) fig.update\_layout(  title=dict(text='Высота пирамид по фараонам', x=0.5, font=dict(size=20)),  xaxis=dict(  title=dict(text='Фараон', font=dict(size=16)),  tickangle=315,  tickfont=dict(size=14),  gridcolor='ivory',  gridwidth=2  ),  yaxis=dict(  title=dict(text='Высота (м)', font=dict(size=16)),  tickfont=dict(size=14),  gridcolor='ivory',  gridwidth=2  ),  coloraxis=dict(colorscale='Viridis'),  width=1200,  height=700,  margin=dict(l=0, r=0, t=60, b=0) ) fig.show() |

Изображение выглядит как снимок экрана, текст, диаграмма, График

Контент, сгенерированный ИИ, может содержать ошибки.

4.

fig\_pie = go.Figure(  
 go. fig\_pie = go.Figure(  
 go.Pie(  
 labels=data['Dynasty'],  
 values=data['Height (m)'],  
 marker=dict(line=dict(color='black', width=2)),  
 textinfo='label+percent+value', # Добавляем подписи: название + процент + значение  
 hovertemplate='<b>%{label}</b><br>Высота: %{value} м<br>Доля: %{percent}<extra></extra>',  
 textposition='outside', # Размещаем подписи вне секторов  
 textfont=dict(size=12, color='black'), # Настройка шрифта подписей  
 insidetextorientation='horizontal' # Ориентация текста внутри секторов  
 )  
)  
fig\_pie.update\_layout(  
 title=dict(text='Распределение высот пирамид по династиям', x=0.5, font=dict(size=20)),  
 width=1200,  
 height=700,  
 showlegend=True # Показываем легенду для дополнительной информации  
)  
fig\_pie.show()

Изображение выглядит как текст, снимок экрана, диаграмма, круг

Контент, сгенерированный ИИ, может содержать ошибки.

5.

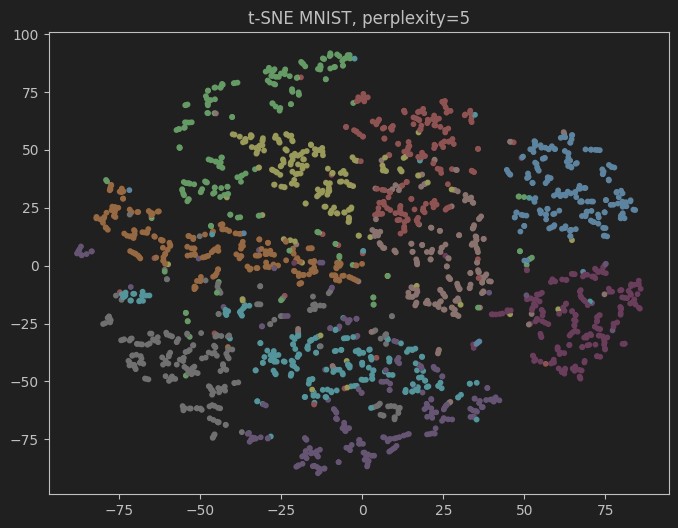
|  |
| --- |
| import matplotlib.pyplot as plt  x = data['Dynasty'] y1 = data['Height (m)'] y2 = data['Slope (dec degr)'] y3 = data['Volume (cu.m)']  plt.figure(figsize=(12, 7)) plt.plot(x, y1, label='Высота (м)', color='crimson', marker='o', markerfacecolor='white', markeredgecolor='black', markeredgewidth=2) plt.plot(x, y2, label='Уклон (град)', color='crimson', marker='s', markerfacecolor='white', markeredgecolor='black', markeredgewidth=2) plt.plot(x, y3, label='Объем (куб.м)', color='crimson', marker='^', markerfacecolor='white', markeredgecolor='black', markeredgewidth=2) plt.xlabel('Династия', fontsize=16) plt.ylabel('Параметры пирамид', fontsize=16) plt.xticks(fontsize=14) plt.yticks(fontsize=14) plt.grid(color='mistyrose', linewidth=2) plt.legend() plt.show() |

Изображение выглядит как снимок экрана, текст, линия, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

6.

|  |
| --- |
| from sklearn.manifold import TSNE from sklearn.datasets import fetch\_openml import matplotlib.pyplot as plt  mnist = fetch\_openml('mnist\_784', version=1) X = mnist.data[:2000] y = mnist.target[:2000]  for perplexity in [5, 30, 50]:  tsne = TSNE(n\_components=2, perplexity=perplexity, random\_state=42)  X\_embedded = tsne.fit\_transform(X)  plt.figure(figsize=(8, 6))  plt.scatter(X\_embedded[:, 0], X\_embedded[:, 1], c=y.astype(int), cmap='tab10', s=10)  plt.title(f't-SNE MNIST, perplexity={perplexity}')  plt.show() |



Изображение выглядит как снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как снимок экрана

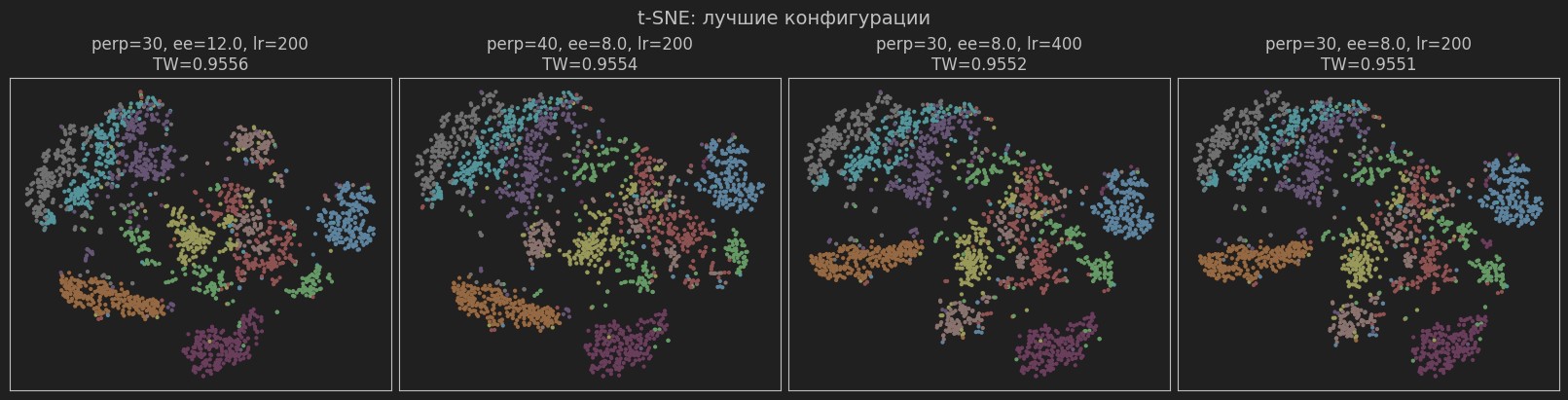
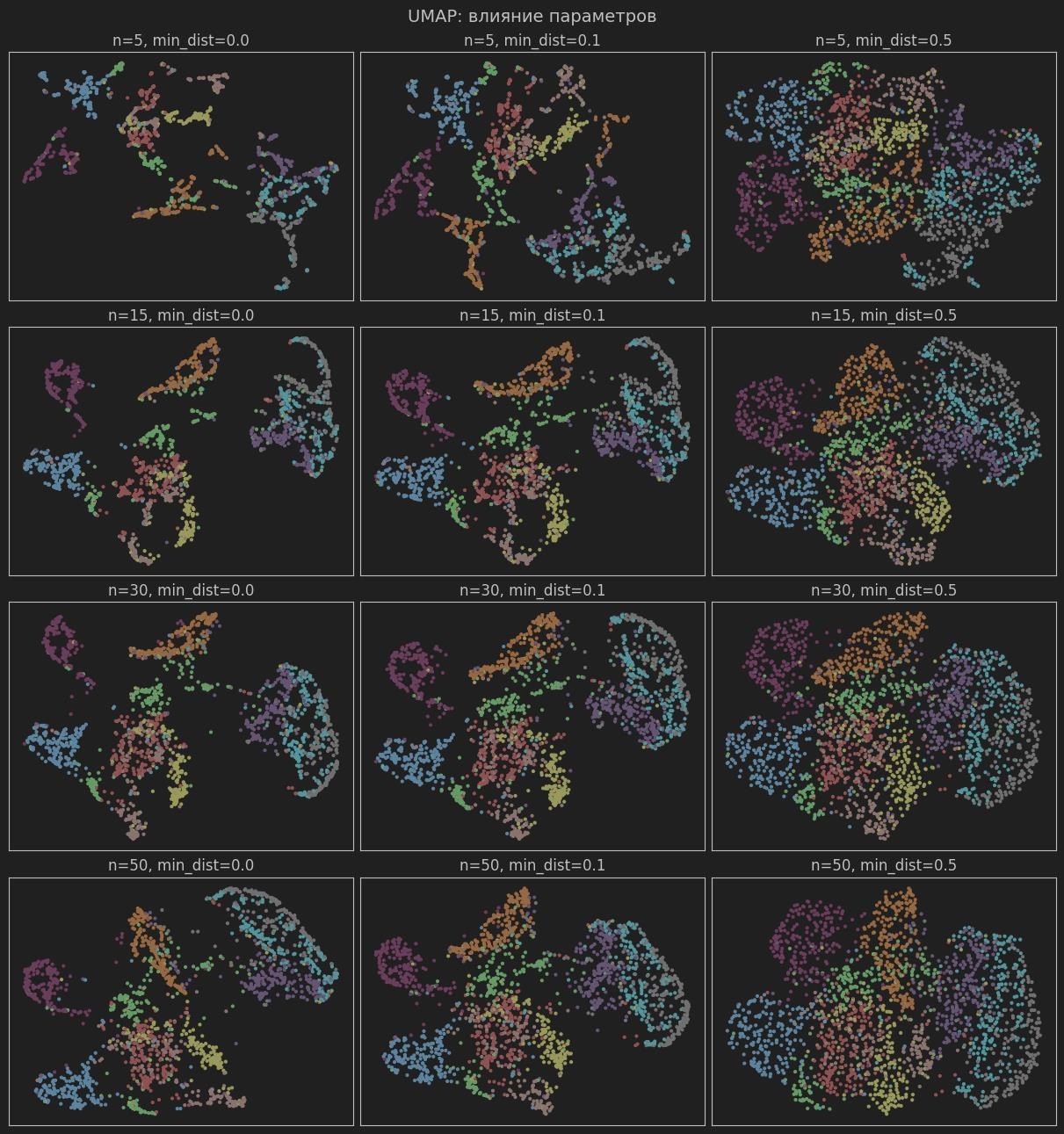
Контент, сгенерированный ИИ, может содержать ошибки.

7.

|  |
| --- |
| import time import itertools import numpy as np import pandas as pd  import matplotlib.pyplot as plt    import umap  from sklearn.manifold import TSNE, trustworthiness from sklearn.preprocessing import StandardScaler from sklearn.decomposition import PCA from sklearn.datasets import fetch\_openml    RANDOM\_STATE = 42  UMAP\_N\_NEIGHBORS = [5, 15, 30, 50]  UMAP\_MIN\_DIST = [0.0, 0.1, 0.5]    TSNE\_PERPLEXITIES = [20, 30, 40, 50]  TSNE\_EE = [8.0, 12.0]  TSNE\_LR = [200, 400, 800] |

|  |
| --- |
| TSNE\_MAX\_ITER = 3500  TSNE\_N\_ITER\_WO\_PROGRESS = 700  TRUST\_K = 15  PCA\_COMPONENTS = 50    N\_SAMPLES = 2000  X = X.iloc[:N\_SAMPLES].to\_numpy() y = y.iloc[:N\_SAMPLES]    if hasattr(y, "cat"):  y\_plot = y.cat.codes.to\_numpy() else:  y\_plot = pd.Series(y).astype(str).astype(int).to\_numpy()    scaler = StandardScaler() X\_std = scaler.fit\_transform(X)    # PCA для t-SNE  pca = PCA(n\_components=min(PCA\_COMPONENTS, X\_std.shape[1]), random\_state=RANDOM\_STATE)  X\_pca = pca.fit\_transform(X\_std)    umap\_results = [] umap\_embeddings = {}    for n\_nb, m\_dist in itertools.product(UMAP\_N\_NEIGHBORS, UMAP\_MIN\_DIST): t0 = time.perf\_counter() emb = umap.UMAP( n\_neighbors=n\_nb, min\_dist=m\_dist, n\_components=2,  metric='euclidean',  random\_state=RANDOM\_STATE  ).fit\_transform(X\_std) dt = time.perf\_counter() - t0  tw = trustworthiness(X\_std, emb, n\_neighbors=TRUST\_K) umap\_results.append({  'method': 'UMAP',  'params': f"n={n\_nb}, min\_dist={m\_dist}",  'time': dt,  'tw': tw  })  umap\_embeddings[(n\_nb, m\_dist)] = emb    tsne\_results = [] tsne\_embeddings = {} |

|  |
| --- |
| for perp, ee, lr in itertools.product(TSNE\_PERPLEXITIES, TSNE\_EE, TSNE\_LR): if perp >= X\_pca.shape[0]:  continue  label = f"perp={perp}, ee={ee}, lr={lr}" t0 = time.perf\_counter() tsne = TSNE( n\_components=2, perplexity=perp, early\_exaggeration=ee, learning\_rate=lr,  max\_iter=TSNE\_MAX\_ITER,  n\_iter\_without\_progress=TSNE\_N\_ITER\_WO\_PROGRESS, init='pca',  random\_state=RANDOM\_STATE, method='barnes\_hut', angle=0.5, verbose=0  )  emb = tsne.fit\_transform(X\_pca)  dt = time.perf\_counter() - t0  tw = trustworthiness(X\_pca, emb, n\_neighbors=TRUST\_K) tsne\_results.append({  'method': 't-SNE',  'params': label,  'time': dt,  'tw': tw  })  tsne\_embeddings[label] = emb    all\_results = umap\_results + tsne\_results  all\_results\_sorted = sorted(all\_results, key=lambda d: d['tw'], reverse=True)    print("Сводная таблица (sorted by TW desc):") for r in all\_results\_sorted:  print(f"{r['method']:5s} {r['params']:<28} time={r['time']:.2f}s TW={r['tw']:.4f}")    best\_umap = max(umap\_results, key=lambda x: x['tw']) fast\_umap = min(umap\_results, key=lambda x: x['time']) best\_tsne = max(tsne\_results, key=lambda x: x['tw']) fast\_tsne = min(tsne\_results, key=lambda x: x['time'])    print("\nИтог:")  print(f"Лучший UMAP: {best\_umap['params']} TW={best\_umap['tw']:.4f} time={best\_umap['time']:.2f}s")  print(f"Быстрый UMAP: {fast\_umap['params']} TW={fast\_umap['tw']:.4f} time={fast\_umap['time']:.2f}s")  print(f"Лучший t-SNE: {best\_tsne['params']} TW={best\_tsne['tw']:.4f} |
| time={best\_tsne['time']:.2f}s")  print(f"Быстрый t-SNE: {fast\_tsne['params']} TW={fast\_tsne['tw']:.4f} time={fast\_tsne['time']:.2f}s")    rows = len(UMAP\_N\_NEIGHBORS) cols = len(UMAP\_MIN\_DIST)  fig, axes = plt.subplots(rows, cols, figsize=(4 \* cols, 3.2 \* rows), constrained\_layout=True) for i, n\_nb in enumerate(UMAP\_N\_NEIGHBORS): for j, m\_dist in enumerate(UMAP\_MIN\_DIST): ax = axes[i, j]  emb = umap\_embeddings[(n\_nb, m\_dist)]  ax.scatter(emb[:, 0], emb[:, 1], c=y\_plot, cmap='tab10', s=4) ax.set\_title(f"n={n\_nb}, min\_dist={m\_dist}") ax.set\_xticks([]); ax.set\_yticks([])  fig.suptitle("UMAP: влияние параметров", fontsize=14) plt.show()    top\_k = 4  top\_tsne = sorted(tsne\_results, key=lambda x: x['tw'], reverse=True)[:top\_k] fig, axes = plt.subplots(1, len(top\_tsne), figsize=(4 \* len(top\_tsne), 4), constrained\_layout=True) if len(top\_tsne) == 1: axes = [axes] for ax, rec in zip(axes, top\_tsne):  emb = tsne\_embeddings[rec['params']]  ax.scatter(emb[:, 0], emb[:, 1], c=y\_plot, cmap='tab10', s=4) ax.set\_title(f"{rec['params']}\nTW={rec['tw']:.4f}") ax.set\_xticks([]); ax.set\_yticks([])  fig.suptitle("t-SNE: лучшие конфигурации", fontsize=14) plt.show() |



Сводная таблица (sorted by TW desc):

t-SNE perp=30, ee=12.0, lr=200 time=15.79s TW=0.9556

t-SNE perp=40, ee=8.0, lr=200 time=17.81s TW=0.9554 t-SNE perp=30, ee=8.0, lr=400 time=15.72s TW=0.9552 t-SNE perp=30, ee=8.0, lr=200 time=16.03s TW=0.9551 t-SNE perp=20, ee=8.0, lr=800 time=13.54s TW=0.9550 t-SNE perp=40, ee=8.0, lr=800 time=18.66s TW=0.9550 t-SNE perp=50, ee=12.0, lr=800 time=19.92s TW=0.9548 t-SNE perp=50, ee=12.0, lr=400 time=20.38s TW=0.9547 t-SNE perp=50, ee=12.0, lr=200 time=20.06s TW=0.9546 t-SNE perp=50, ee=8.0, lr=800 time=20.45s TW=0.9544 t-SNE perp=30, ee=12.0, lr=400 time=15.87s TW=0.9541 t-SNE perp=40, ee=8.0, lr=400 time=18.72s TW=0.9540 t-SNE perp=20, ee=8.0, lr=400 time=13.50s TW=0.9539 t-SNE perp=20, ee=12.0, lr=400 time=14.14s TW=0.9538 t-SNE perp=30, ee=12.0, lr=800 time=16.14s TW=0.9537 t-SNE perp=20, ee=12.0, lr=200 time=13.43s TW=0.9536 t-SNE perp=20, ee=12.0, lr=800 time=13.91s TW=0.9535 t-SNE perp=30, ee=8.0, lr=800 time=16.18s TW=0.9532 t-SNE perp=40, ee=12.0, lr=200 time=18.39s TW=0.9530 t-SNE perp=40, ee=12.0, lr=800 time=18.59s TW=0.9526 t-SNE perp=20, ee=8.0, lr=200 time=13.57s TW=0.9526 t-SNE perp=40, ee=12.0, lr=400 time=18.32s TW=0.9525 t-SNE perp=50, ee=8.0, lr=200 time=20.63s TW=0.9523 t-SNE perp=50, ee=8.0, lr=400 time=19.94s TW=0.9514 UMAP n=5, min\_dist=0.1 time=3.33s TW=0.8855

UMAP n=5, min\_dist=0.0 time=3.41s TW=0.8842

UMAP n=15, min\_dist=0.0 time=3.91s TW=0.8838

UMAP n=15, min\_dist=0.1 time=3.91s TW=0.8806

UMAP n=30, min\_dist=0.0 time=4.44s TW=0.8785

UMAP n=30, min\_dist=0.1 time=4.63s TW=0.8713 UMAP n=50, min\_dist=0.0 time=4.95s TW=0.8690

UMAP n=50, min\_dist=0.1 time=4.93s TW=0.8645

UMAP n=5, min\_dist=0.5 time=3.26s TW=0.8550

UMAP n=15, min\_dist=0.5 time=3.94s TW=0.8549

UMAP n=30, min\_dist=0.5 time=4.58s TW=0.8434

UMAP n=50, min\_dist=0.5 time=5.00s TW=0.8297

Итог:

Лучший UMAP: n=5, min\_dist=0.1 TW=0.8855 time=3.33s

Быстрый UMAP: n=5, min\_dist=0.5 TW=0.8550 time=3.26s

Лучший t-SNE: perp=30, ee=12.0, lr=200 TW=0.9556 time=15.79s

Быстрый t-SNE: perp=20, ee=12.0, lr=200 TW=0.9536 time=13.43s