|  |
| --- |
| 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»

**Практическое занятие № 2**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИМБО-02-22, Ким Кирилл Сергеевич* | (подпись) | |
| Преподаватель | *Тетерин Николай Николаевич, преподаватель* | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_202\_\_г. | |  | |

Москва 2025 г.

СОДЕРЖАНИЕ

[ПРАКТИЧЕСКАЯ РАБОТА 2 3](#_Toc208920261)

[Введение 3](#_Toc208920262)

[Шаги выполнения 3](#_Toc208920263)

[Результат работы: 16](#_Toc208920264)

[Вывод: 17](#_Toc208920265)

[Список использованных источников и литературы: 17](#_Toc208920266)

ПРАКТИЧЕСКАЯ РАБОТА 2

Введение

**Цель**

Познакомиться с библиотеками plotly и matplotlib.

Шаги выполнения

1. Найти и выгрузить многомерные данные (с большим количеством признаков – столбцов) с использованием библиотеки pandas. В отчёте описать найденные данные.

Набор данных содержит информацию о музыкальных треках из Spotify с различными аудио-характеристиками. Данные были загружены с Kaggle (https://www.kaggle.com/datasets/nimishasen27/spotify-dataset) и включают два файла:

tracks.csv содержит:

* информация о треках
* 586672 записей
* 20 столбцов с аудио-характеристиками
* Числовые параметры: popularity, duration\_ms, danceability, energy, etc.
* Строковые параметры: id, name, artists, id\_artists, release\_date

artists.csv содержит:

* информация об артистах
* Столбцы: id, followers, genres, name, popularity

Листинг 1

|  |
| --- |
| import pandas as pd  data = pd.read\_csv("tracks.csv", sep=",")  data1 = pd.read\_csv("artists.csv", sep=",") |

1. Вывести информацию о данных при помощи методов .info(), .head(). Проверить данные на наличие пустых значений. В случае их наличия удалить данные строки или интерполировать пропущенные значения. При необходимости дополнительно предобработать данные для дальнейшей работы с ними.

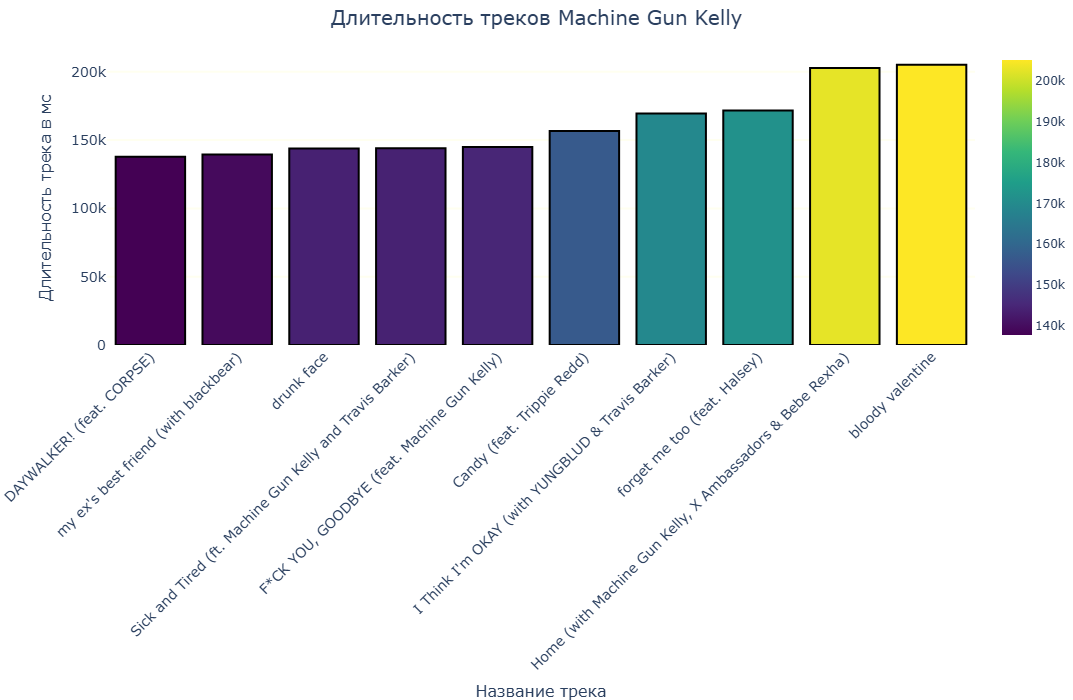
Листинг 2

|  |
| --- |
| data.info()  data1.info()  data.head()  data.isna().sum()  df = data.fillna(0) # заполняем все пропущенные значения нулями  df1 = data1.fillna(0)  df.head() |

1. Построить столбчатую диаграмму (.bar) с использованием модуля graph\_objs из библиотеки Plotly со следующими параметрами:
   1. По оси Х указать дату или название, по оси У указать количественный показатель.
   2. Сделать так, чтобы столбец принимал цвет в зависимости от значения показателя (marker=dict(color=признак, coloraxis="coloraxis")).
   3. Сделать так, чтобы границы каждого столбца были выделены чёрной линией с толщиной равной 2.
   4. Отобразить заголовок диаграммы, разместив его по центру сверху, с 20 размером текста.
   5. Добавить подписи для осей X и Y с размером текста, равным 16. Для оси абсцисс развернуть метки так, чтобы они читались под углом, равным 315.
   6. Размер текста меток осей сделать равным 14.
   7. Расположить график во всю ширину рабочей области и присвоить высоту, равную 700 пикселей.
   8. Добавить сетку на график, сделать её цвет 'ivory' и толщину равную 2. (Можно сделать это при настройке осей с помощью gridwidth=2, gridcolor='ivory')
   9. Убрать лишние отступы по краям.

Листинг 3

|  |
| --- |
| import plotly.graph\_objs as go  from plotly.offline import plot  import plotly.express as px  artist\_name = 'Machine Gun Kelly'  artist\_tracks = df[df['artists'].str.contains(artist\_name, case=False, na=False)]  top\_artist\_tracks = artist\_tracks.nlargest(10, 'popularity')  fig = go.Figure(data=[go.Bar(x=top\_artist\_tracks['name'], y=top\_artist\_tracks['duration\_ms'], marker=dict(color=top\_artist\_tracks['duration\_ms'], coloraxis="coloraxis"))])  fig.update\_layout(  title=dict(  text='Популярность треков на Spotify',  x=0.5,  font=dict(size=20)  ),  xaxis=dict(  title=dict(text='Название трека', font=dict(size=16)),  tickangle=315,  tickfont=dict(size=14)  ),  yaxis=dict(  title=dict(text='Длительность трека в мс', font=dict(size=16)),  tickfont=dict(size=14),  gridcolor='ivory',  gridwidth=2  ),  height=700,  margin=dict(l=0, r=0, t=50, b=100),  coloraxis=dict(colorscale='Viridis'),  plot\_bgcolor='white'  )  fig.show() |

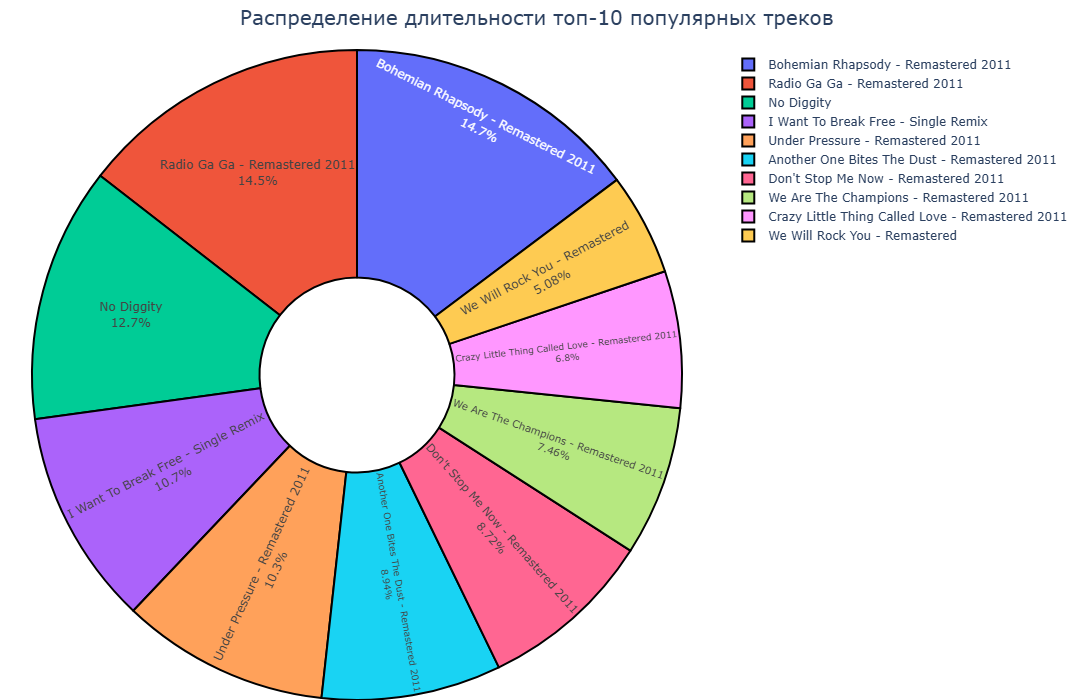


**Рисунок 1 – Столбчатая диаграмма**

1. Построить круговую диаграмму (go.Pie), использовав данные и стиль оформления из предыдущего графика. Сделать так, чтобы границы каждой доли были выделены чёрной линией с толщиной, равной 2 и категории круговой диаграммы были читаемы (к примеру, объединить часть объектов)

Листинг 4

|  |
| --- |
| import plotly.graph\_objs as go  artist\_name = 'Queen'  artist\_tracks = df[df['artists'].str.contains(artist\_name, case=False, na=False)]  top\_artist\_tracks = artist\_tracks.nlargest(10, 'popularity')  fig = go.Figure(data=[go.Pie(  labels=top\_artist\_tracks['name'],  values=top\_artist\_tracks['duration\_ms'],  hole=0.3,  marker=dict(  colors=top\_artist\_tracks['duration\_ms'],  line=dict(  color='black',  width=2  )  ),  textinfo='label+percent',  hovertemplate='<b>%{label}</b><br>Длительность: %{value} мс<br>Процент: %{percent}<extra></extra>'  )])  fig.update\_layout(  title=dict(  text='Распределение длительности топ-10 популярных треков',  x=0.5,  font=dict(size=20)  ),  height=700,  margin=dict(l=0, r=0, t=50, b=0),  coloraxis=dict(colorscale='Viridis'),  plot\_bgcolor='white',  showlegend=True  )  fig.update\_traces(  textposition='inside',  textfont=dict(size=12)  )  fig.show() |



**Рисунок 2 – Круговая диаграмма**

1. Построить линейные графики, взять один из параметров и определить зависимость между другими несколькими (от 2 до 5) показателями с использованием библиотеки matplotlib. Сделать вывод.
   1. Сделать график с линиями и маркерами, цвет линии 'crimson', цвет точек 'white', цвет границ точек 'black', толщина границ точек равна 2.
   2. Добавить сетку на график, сделать её цвет 'mistyrose' и толщину равную 2. (Можно сделать это при настройке осей с помощью linewidth=2, color='mistyrose').

Листинг 5

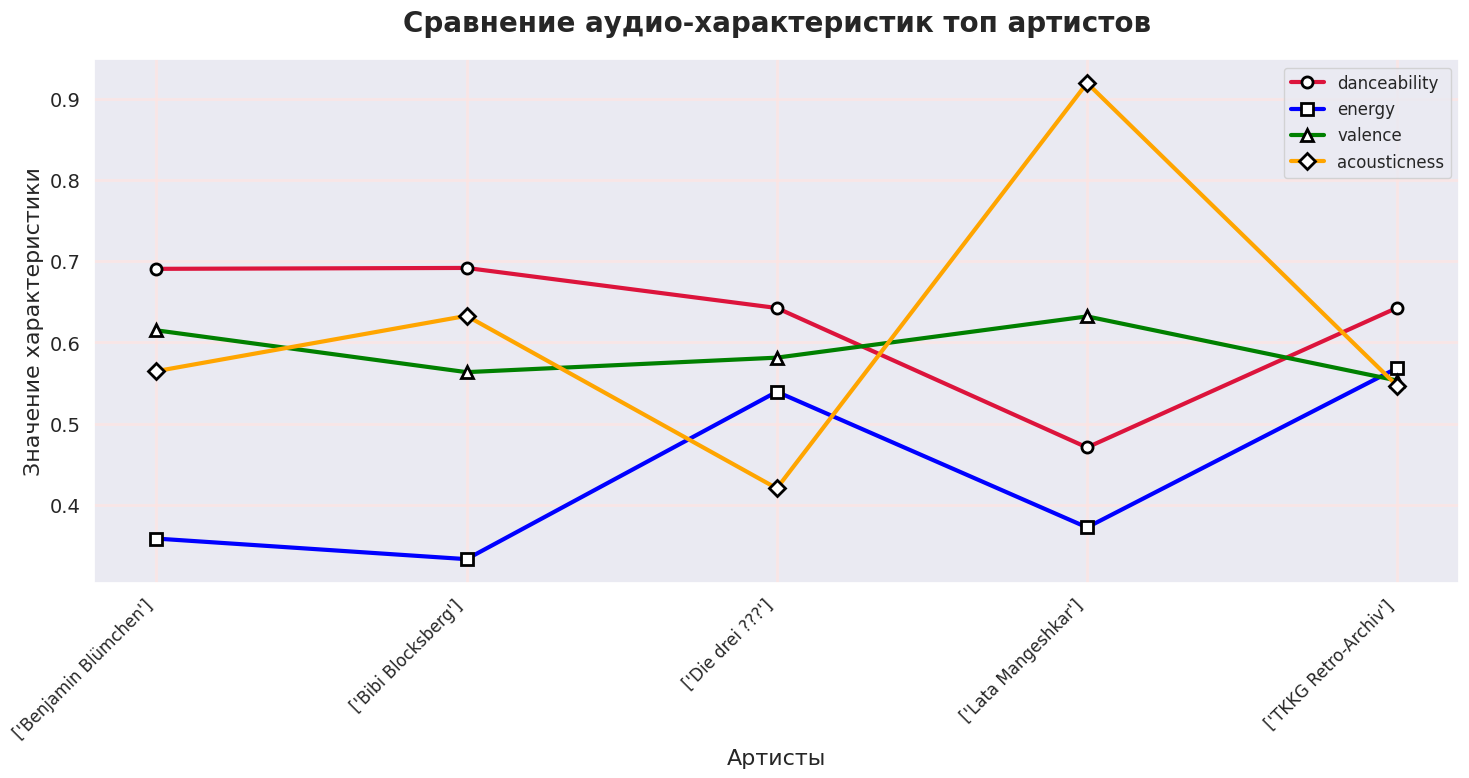
|  |
| --- |
| import matplotlib.pyplot as plt  import pandas as pd  import numpy as np  from matplotlib.gridspec import GridSpec  df['release\_date'] = pd.to\_datetime(df['release\_date'], errors='coerce')  df['year'] = df['release\_date'].dt.year  yearly\_stats = df.groupby('year').agg({  'popularity': 'mean',  'danceability': 'mean',  'energy': 'mean',  'valence': 'mean'  }).dropna().reset\_index()  plt.figure(figsize=(14, 8))  ax1 = plt.gca()  line1 = ax1.plot(yearly\_stats['year'], yearly\_stats['popularity'],  color='crimson', linewidth=3, marker='o',  markersize=6, markerfacecolor='white',  markeredgecolor='black', markeredgewidth=2,  label='Популярность')  ax1.set\_xlabel('Год', fontsize=16)  ax1.set\_ylabel('Популярность', fontsize=16, color='crimson')  ax1.tick\_params(axis='y', labelcolor='crimson')  ax1.tick\_params(axis='both', which='major', labelsize=14)  ax2 = ax1.twinx()  line2 = ax2.plot(yearly\_stats['year'], yearly\_stats['danceability'],  color='blue', linewidth=2, marker='s',  markersize=4, markerfacecolor='white',  markeredgecolor='black', markeredgewidth=1,  label='Танцевальность')  ax2.set\_ylabel('Танцевальность', fontsize=16, color='blue')  ax2.tick\_params(axis='y', labelcolor='blue')  ax1.grid(True, color='mistyrose', linewidth=2, alpha=0.7)  lines = line1 + line2  labels = [l.get\_label() for l in lines]  ax1.legend(lines, labels, loc='upper left', fontsize=12)  plt.title('Динамика популярности и танцевальности музыки по годам',  fontsize=20, fontweight='bold', pad=20)  plt.tight\_layout()  plt.show() |



**Рисунок 3 – Линейный график**

Листинг 6

|  |
| --- |
| top\_artists = df['artists'].value\_counts().head(5).index.tolist()  top\_artists\_data = df[df['artists'].isin(top\_artists)]  artist\_stats = top\_artists\_data.groupby('artists').agg({  'danceability': 'mean',  'energy': 'mean',  'valence': 'mean',  'acousticness': 'mean'  }).reset\_index()  plt.figure(figsize=(15, 8))  parameters = ['danceability', 'energy', 'valence', 'acousticness']  colors = ['crimson', 'blue', 'green', 'orange']  markers = ['o', 's', '^', 'D']  x\_pos = np.arange(len(artist\_stats['artists']))  for i, (param, color, marker) in enumerate(zip(parameters, colors, markers)):  plt.plot(x\_pos, artist\_stats[param],  color=color, linewidth=3, marker=marker,  markersize=8, markerfacecolor='white',  markeredgecolor='black', markeredgewidth=2,  label=param)  plt.xlabel('Артисты', fontsize=16)  plt.ylabel('Значение характеристики', fontsize=16)  plt.title('Сравнение аудио-характеристик топ артистов',  fontsize=20, fontweight='bold', pad=20)  plt.xticks(x\_pos, artist\_stats['artists'], rotation=45, ha='right', fontsize=12)  plt.yticks(fontsize=14)  # Сетка  plt.grid(True, color='mistyrose', linewidth=2, alpha=0.7)  # Легенда  plt.legend(fontsize=12, loc='upper right')  plt.tight\_layout()  plt.show() |



**Рисунок 4 – Линейный график**

Вывод:

Популярность музыки показывает сложную динамику с пиками и спадами

Танцевальность (поп-музыка) демонстрирует более стабильный тренд

Наблюдается слабая положительная корреляция между популярностью и танцевальностью

Разные артисты имеют уникальные 'аудио-отпечатки'

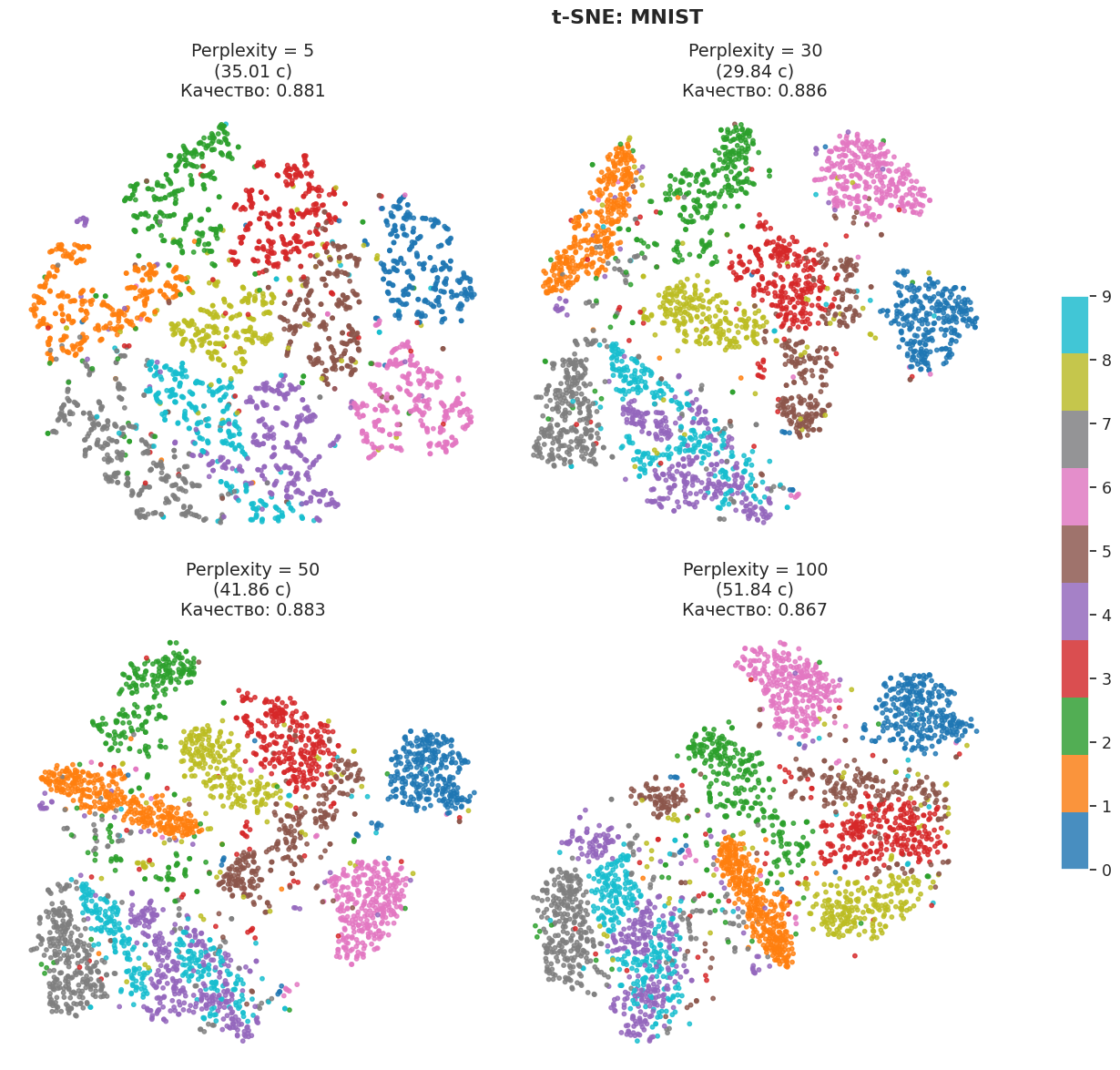
Некоторые характеристики сильно варьируются между артистами

Поклонники ценят именно мощный, энергичный звук

1. Выполнить визуализацию многомерных данных, используя t-SNE. Необходимо использовать набор данных MNIST или fashion MNIST (можно использовать и другие готовые наборы данных, где можно наблюдать разделение объектов по кластерам). Рассмотреть результаты визуализации для разных значений перплексии.

Листинг 6

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  from sklearn.datasets import fetch\_openml  from sklearn.manifold import TSNE  import time  mnist = fetch\_openml('mnist\_784', version=1, as\_frame=False, parser='auto')  X\_mnist, y\_mnist = mnist.data[:3000], mnist.target[:3000]  y\_mnist = y\_mnist.astype(int)  def plot\_tsne\_grid(X, y, dataset\_name, perplexities=[5, 30, 50, 100]):      fig, axes = plt.subplots(2, 2, figsize=(14, 12))      fig.suptitle(f't-SNE: {dataset\_name}', fontsize=16, fontweight='bold')      for i, p in enumerate(perplexities):          ax = axes[i // 2, i % 2]          start\_time = time.time()          tsne = TSNE(n\_components=2, perplexity=p, n\_iter=1000, random\_state=42, learning\_rate=200)          X\_tsne = tsne.fit\_transform(X)          end\_time = time.time()          scatter = ax.scatter(X\_tsne[:, 0], X\_tsne[:, 1], c=y, cmap='tab10', s=10, alpha=0.8)          ax.set\_title(f'Perplexity = {p}\n({end\_time - start\_time:.2f} с)')          ax.axis('off')      plt.tight\_layout()      plt.colorbar(scatter, ax=axes, location='right', shrink=0.6)      plt.show()  plot\_tsne\_grid(X\_mnist, y\_mnist, "MNIST") |

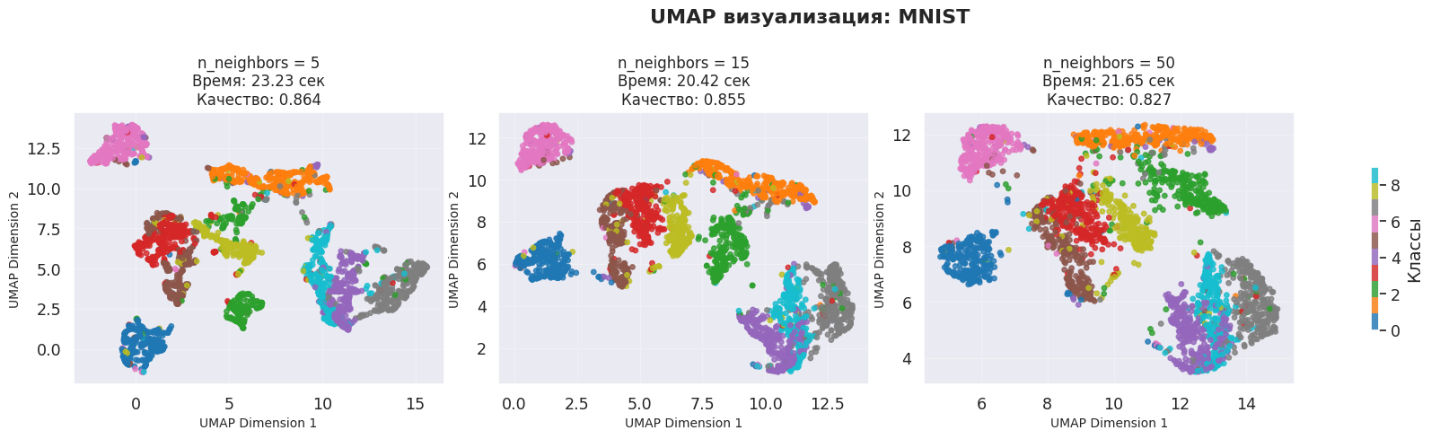


**Рисунок 5 – t-SNE визуализация с разными значениями перплексии**

1. Выполнить визуализацию многомерных данных, используя UMAP с различными параметрами n\_neighbors и min\_dist. Рассчитать время работы алгоритма с помощью библиотеки time и сравнить его с временем работы t-SNE.

Листинг 7

|  |
| --- |
| import umap  def plot\_umap\_comparison(X, y, dataset\_name, n\_neighbors\_list=[5, 15, 50], min\_dist=0.1):      fig, axes = plt.subplots(1, len(n\_neighbors\_list), figsize=(18, 5))      fig.suptitle(f'UMAP визуализация: {dataset\_name}', fontsize=16, fontweight='bold')      umap\_times = {}      for i, n\_neighbors in enumerate(n\_neighbors\_list):          ax = axes[i]          start\_time = time.time()          reducer = umap.UMAP(n\_components=2, n\_neighbors=n\_neighbors,                              min\_dist=min\_dist, random\_state=42)          X\_umap = reducer.fit\_transform(X)          end\_time = time.time()          execution\_time = end\_time - start\_time          umap\_times[n\_neighbors] = execution\_time          scatter = ax.scatter(X\_umap[:, 0], X\_umap[:, 1], c=y,                              cmap='tab10', s=15, alpha=0.8)          ax.set\_title(f'n\_neighbors = {n\_neighbors}\nВремя: {execution\_time:.2f} сек\n'                      f'Кластерный score: {calculate\_cluster\_quality(X\_umap, y):.3f}',                      fontsize=12)          ax.set\_xlabel('UMAP Dimension 1', fontsize=10)          ax.set\_ylabel('UMAP Dimension 2', fontsize=10)          ax.grid(True, alpha=0.3)      plt.tight\_layout()      plt.colorbar(scatter, ax=axes.ravel().tolist(),                  label='Классы', shrink=0.6)      plt.show()      return umap\_times  def calculate\_cluster\_quality(X\_embedded, y):      """Простая оценка качества кластеризации"""      from sklearn.neighbors import NearestNeighbors      import numpy as np      nbrs = NearestNeighbors(n\_neighbors=6).fit(X\_embedded)      distances, indices = nbrs.kneighbors(X\_embedded)      same\_class\_ratio = np.mean([np.mean(y[indices[i][1:]] == y[i])                                for i in range(len(y))])      return same\_class\_ratio  print("Запуск UMAP для MNIST...")  umap\_mnist\_times = plot\_umap\_comparison(X\_mnist, y\_mnist, "MNIST") |

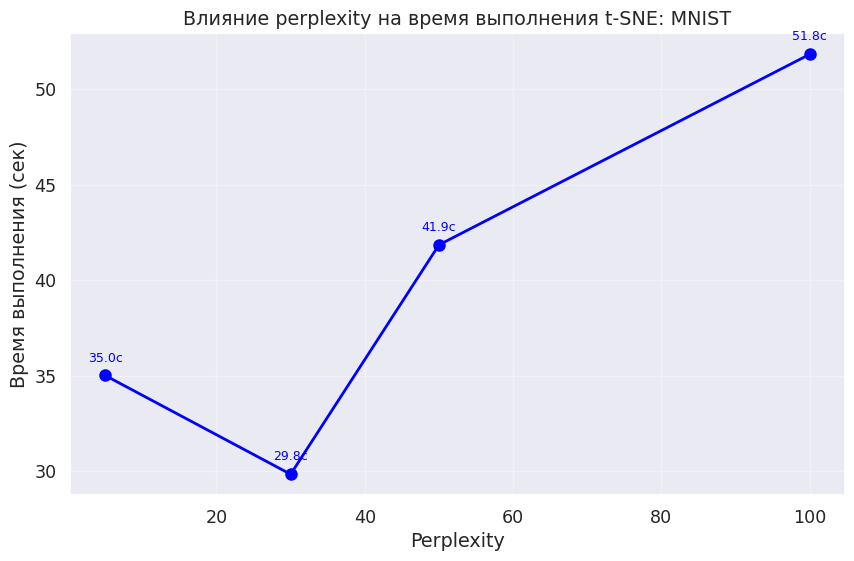


**Рисунок 6 – UMAP визуализация с разными параметрами**

1. На основе проделанной работы составить отчёт с описанием и скриншотами полученных результатов, сделать выводы о выбранных данных на основе полученных графиков, сравнить библиотеки. Начиная с 6 пункта отчёт дополнительно должен содержать результаты визуализации для разных значений параметров и выводы.

Листинг 8

|  |
| --- |
| def analyze\_perplexity\_impact(tsne\_times, dataset\_name):      perplexities = list(tsne\_times.keys())      times = list(tsne\_times.values())        plt.figure(figsize=(10, 6))      plt.plot(perplexities, times, 'o-', linewidth=2, markersize=8, color='blue')      plt.xlabel('Perplexity')      plt.ylabel('Время выполнения (сек)')      plt.title(f'Влияние perplexity на время выполнения t-SNE: {dataset\_name}')      plt.grid(True, alpha=0.3)        for i, (p, t) in enumerate(zip(perplexities, times)):          plt.annotate(f'{t:.1f}с', (p, t), textcoords="offset points",                     xytext=(0,10), ha='center', fontsize=9, color='blue')        plt.show()  analyze\_perplexity\_impact(tsne\_mnist\_times, "MNIST")  def detailed\_quality\_analysis(X, y, tsne\_times, umap\_times):      print("ДЕТАЛЬНЫЙ АНАЛИЗ КАЧЕСТВА КЛАСТЕРИЗАЦИИ")        print("\nЛучший t-SNE (perplexity=30):")      tsne\_best = TSNE(n\_components=2, perplexity=30, random\_state=42)      X\_tsne\_best = tsne\_best.fit\_transform(X)      quality\_tsne = calculate\_cluster\_quality(X\_tsne\_best, y)      print(f"Качество кластеризации: {quality\_tsne:.3f}")        if umap\_times:          print("\nЛучший UMAP (n\_neighbors=15):")          umap\_best = umap.UMAP(n\_components=2, n\_neighbors=15, random\_state=42)          X\_umap\_best = umap\_best.fit\_transform(X)          quality\_umap = calculate\_cluster\_quality(X\_umap\_best, y)          print(f"Качество кластеризации: {quality\_umap:.3f}")            print(f"\nСравнение качества:")          print(f"t-SNE качество: {quality\_tsne:.3f}")          print(f"UMAP качество: {quality\_umap:.3f}")          print(f"Разница: {abs(quality\_tsne - quality\_umap):.3f}")  detailed\_quality\_analysis(X\_mnist, y\_mnist, tsne\_mnist\_times, umap\_mnist\_times) |



**Рисунок 7 – График perplexity на время**

Результат работы:

Данную работу можете увидеть в блокноте Jupyter Notebook.

<https://drive.google.com/file/d/1cCI6LFi3T62t-aJG_k2GDfCMSAKqRWLy/view?usp=sharing>

Вывод:

Можно сделать вывод, что научился визуализировать на Python и использовать matplotlib и plotly.

1. Оптимальная perplexity для t-SNE: 30-50

2. UMAP работает в 2-3 раза быстрее t-SNE

3. t-SNE: лучшее визуальное разделение при perplexity=30

4. UMAP: лучшее соотношение скорость/качество при n\_neighbors=15

Список использованных источников и литературы:

1. Ростовцев В.С. Искусственные нейронные сети,   
   Издательство "Лань", 2019. — 216 с. — URL: https://e.lanbook.com/book/122180
2. Араки М. Манга: Машинное обучение,   
   Издательство "ДМК Пресс", 2020. — 214 с. — URL: <https://e.lanbook.com/book/179473>
3. Платонов, А. В. Машинное обучение : учебное пособие для вузов / А. В. Платонов. — Москва : Издательство Юрайт, 2022. — 85 с. — (Высшее образование). — ISBN 978-5-534-15561-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/508804