99

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

Институт информационных технологий (ИИТ)

Кафедра Прикладной математики

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ

по дисциплине «Технологии и инструментарий анализа больших данных»

Практическая работа № 2

|  |  |
| --- | --- |
| Студент группы | *ИНБО-07-20, Рыжило А.В.*  (подпись) |
| Преподаватель | *Горячев А.А.*  (подпись) |
|  |
| Отчет представлен | «27» сентября 2023г. |

Москва 2023 г.

**СОДЕРЖАНИЕ**

[Задание 1. 3](#_Toc146673899)

[Задание 2. 3](#_Toc146673900)

[Задание 3. 4](#_Toc146673901)

[Задание 4. 6](#_Toc146673902)

[Задание 5. 7](#_Toc146673903)

[Задание 6. 9](#_Toc146673904)

[Задание 7. 12](#_Toc146673905)

[Вывод 18](#_Toc146673906)

# Задание 1.

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

Был использован датасет Успеваемость студентов на экзаменах. Данный датасет был создан для анализа влияния образования родителей, подготовки к тестам и т.д. на успеваемость учащихся. В датасете 8 столбцов: пол, национальность (в не уточняющихся группах), образование родителей, обед, подготовка к курсу, балл по математике, балл за чтение, балл за письмо.

*Листинг 1 – Задание 1*

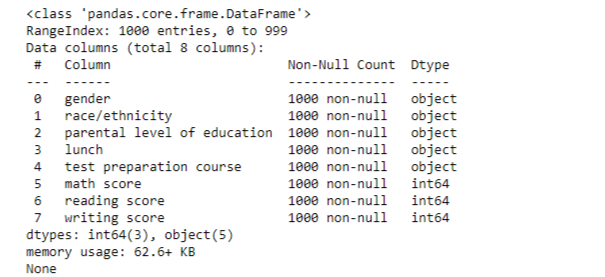
|  |
| --- |
| import pandas as pd  data = pd.read\_csv('StudentsPerformance.csv')  data |

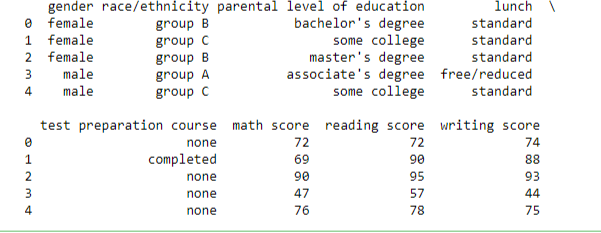
# Задание 2.

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 – Задание 2*

|  |
| --- |
| print(data.info())  print(data.head()) |





# Задание 3.

Построить столбчатую диаграмму (.bar) с использованием модуля graph\_objs из библиотеки Plotly со следующими параметрами:

3.1. По оси Х указать дату или название, по оси У указать количественный показатель.

3.2. Сделать так, чтобы столбец принимал цвет в зависимости от значения показателя (marker=dict(color=признак, coloraxis="coloraxis")).

3.3. Сделать так, чтобы границы каждого столбца были выделены чёрной линией с толщиной равной 2.

3.4. Отобразить заголовок диаграммы, разместив его по центру сверху, с 20 размером текста.

3.5. 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.

3.6. Размер текста меток осей сделать равным 14.

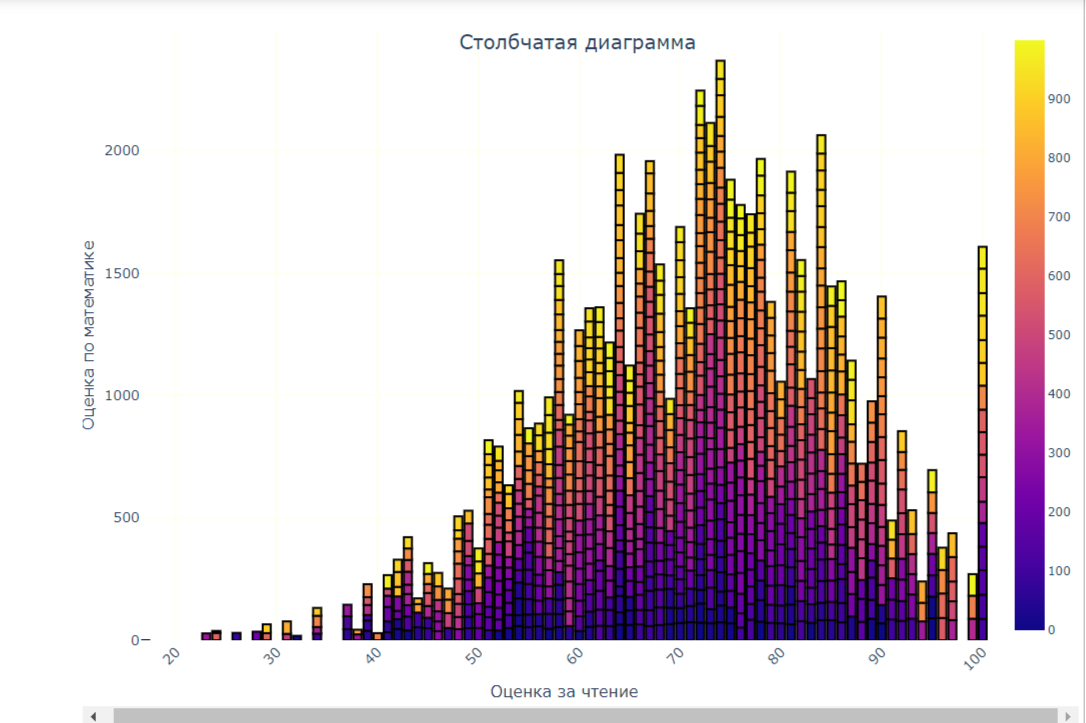
3.7. Расположить график во всю ширину рабочей области и присвоить высоту, равную 700 пикселей.

3.8. Добавить сетку на график, сделать её цвет 'ivory' и толщину равную 2. (Можно сделать это при настройке осей с помощью gridwidth=2, gridcolor='ivory')

3.9. Убрать лишние отступы по краям

*Листинг 3 – Задание 3*

|  |
| --- |
| import plotly.graph\_objs as go  import pandas as pd  datar = pd.read\_csv('StudentsPerformance.csv')  fig = go.Figure(data=go.Bar(x=datar['reading score'], y=datar['math score'],  marker=dict(color=list(range(len(datar['reading score']))), coloraxis="coloraxis"), marker\_line=dict(color='black', width=2)))  fig.update\_layout(  title={  'text': 'Столбчатая диаграмма',  'x': 0.5,  'y': 0.95,  'xanchor': 'center',  'yanchor': 'top',  'font': dict(size=20)  },  xaxis=dict(  title="Оценка за чтение",  tickangle=315,  title\_font=dict(size=16),  tickfont=dict(size=14)  ),  yaxis=dict(  title="Оценка по математике",  title\_font=dict(size=16),  tickfont=dict(size=14)  ),  width=1000,  height=700,  plot\_bgcolor='white',  xaxis\_showgrid=True,  yaxis\_showgrid=True,  xaxis\_gridwidth=2,  yaxis\_gridwidth=2,  xaxis\_gridcolor='ivory',  yaxis\_gridcolor='ivory',  margin=dict(l=0, r=0, t=30, b=0, pad=0)  )  fig.show() |

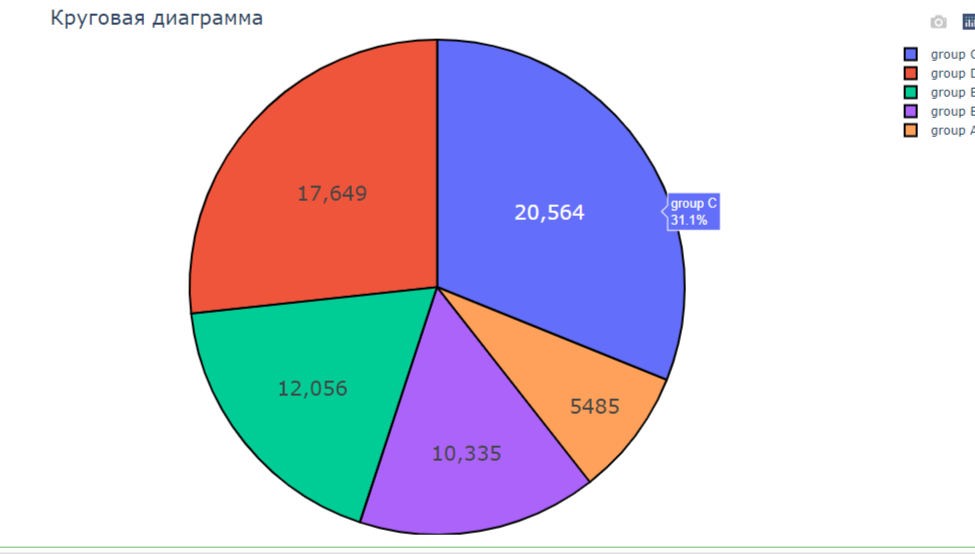


# Задание 4.

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 – Задание 4*

|  |
| --- |
| import plotly.graph\_objs as go  import pandas as pd  datar = pd.read\_csv('StudentsPerformance.csv')  fig = go.Figure(data=[go.Pie(labels=datar['race/ethnicity'],  values=datar['math score'])])  fig.update\_traces(  hoverinfo='label+percent', textinfo='value', textfont\_size=20,  marker=dict(colors=list(range(len(datar['race/ethnicity']))), line=dict(color='#000000', width=2)))  fig.update\_layout(  title='Круговая диаграмма',  title\_font\_size=20,  margin=dict(l=0, r=0, t=30, b=0, pad=0)  )  fig.show() |



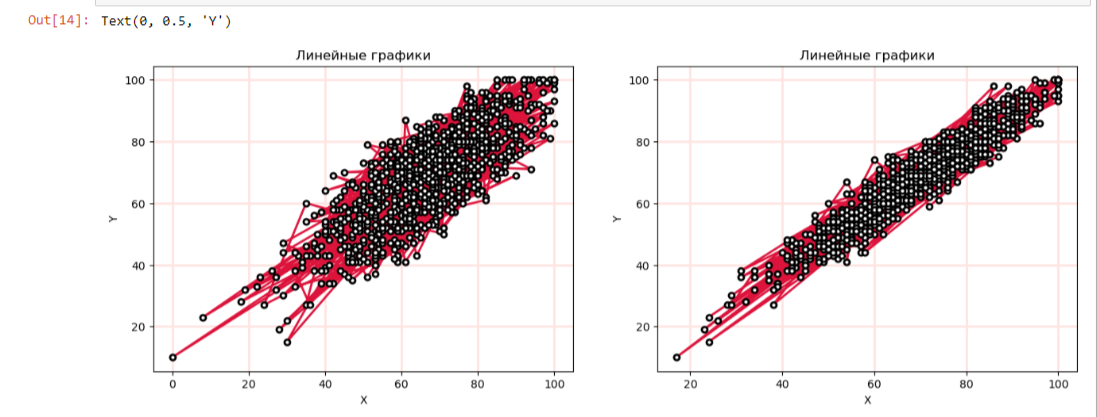
# Задание 5.

Построить линейные графики, взять один из параметров и определить зависимость между другими несколькими (от 2 до 5) показателями с использованием библиотеки matplotlib. Сделать вывод.

5.1. Сделать график с линиями и маркерами, цвет линии 'crimson', цвет точек 'white', цвет границ точек 'black', толщина границ точек равна 2.

5.2. 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 – Задание 6*

|  |
| --- |
| import matplotlib.pyplot as plt  import pandas as pd  datar = pd.read\_csv('StudentsPerformance.csv')  # Создание графика  fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (15,5))  # Линейные графики с маркерами  ax1.plot(datar['math score'], datar['writing score'], color='crimson', linewidth=2, marker='o', markersize=5,  markerfacecolor='white', markeredgewidth=2, markeredgecolor='black')  ax2.plot(datar['reading score'], datar['writing score'], color='crimson', linewidth=2, marker='o', markersize=5,  markerfacecolor='white', markeredgewidth=2, markeredgecolor='black')  # Добавление сетки  ax1.grid(linewidth=2, color='mistyrose')  ax2.grid(linewidth=2, color='mistyrose')  # Настройка внешнего вида графика  ax1.set\_title('Линейные графики')  ax1.set\_xlabel('X')  ax1.set\_ylabel('Y')  ax2.set\_title('Линейные графики')  ax2.set\_xlabel('X')  ax2.set\_ylabel('Y') |



Вывод: оценки по разным направлениям имеют прямую пропорциональную зависимость. Также заметим, что большая часть оценок по математике и письму приблизительно равна 60 баллам, в то время как балл за чтение сильнее разбрасывает оценки по графику.

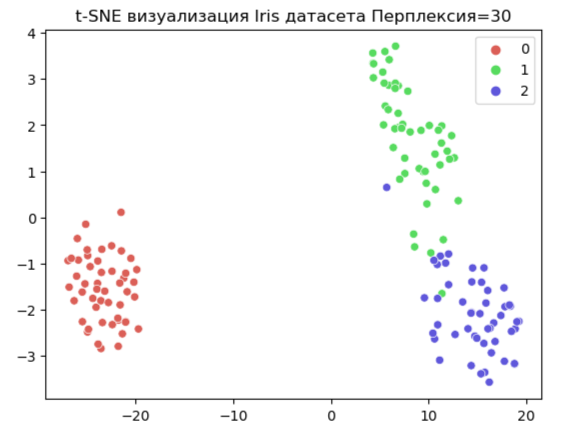
# Задание 6.

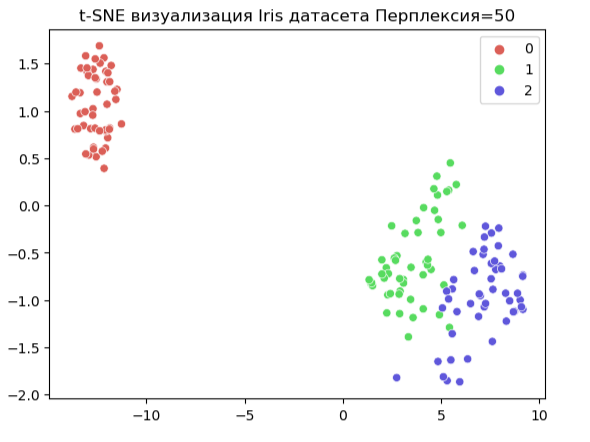
Выполнить визуализацию многомерных данных, используя t-SNE. Необходимо использовать набор данных MNIST или fashion MNIST (можно использовать и другие готовые наборы данных, где можно наблюдать разделение объектов по кластерам). Рассмотреть результаты визуализации для разных значений перплексии.

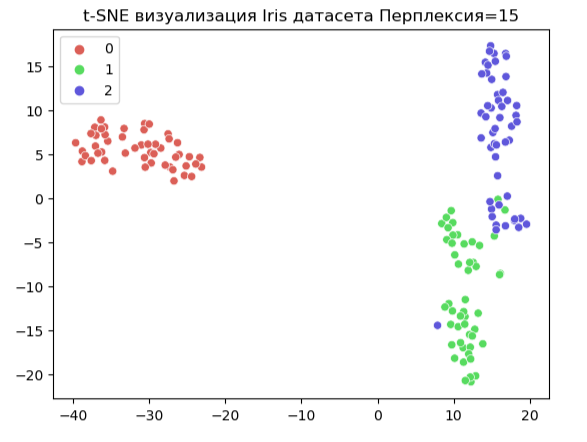
Для выполнения визуализации с использованием t-SNE можно использовать различные готовые наборы данных. Одним из популярных наборов является "Iris", содержащий информацию о различных измерениях цветков ириса и их классификацию на три вида. В данном примере мы загружаем набор данных Iris и разделяем его на матрицу признаков `X` и вектор меток классов `y`. Затем мы применяем t-SNE к данным с помощью `TSNE` из библиотеки `scikit-learn` и устанавливаем значение perplexity равным 30. После этого мы визуализируем результаты с помощью `scatterplot` из библиотеки `seaborn` и устанавливаем цвета точек в соответствии с метками классов. Значение perplexity влияет на результаты визуализации. Оно определяет, насколько близко объект обрабатывает другие объекты во время построения вложения. Более высокое значение perplexity может привести к усреднению кластеров, в то время как более низкое значение может привести к группировке точек вокруг отдельных объектов или малых кластеров.

*Листинг 6 – Задание 6*

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  from sklearn.datasets import load\_iris  from sklearn.manifold import TSNE  # Загрузка набора данных Iris  iris = load\_iris()  X = iris.data  y = iris.target  # Применение t-SNE к данным  tsne = TSNE(n\_components=2, perplexity=30) # Задаем значение perplexity  X\_tsne = tsne.fit\_transform(X)  # Визуализация результатов  sns.scatterplot(x=X\_tsne[:, 0], y=X\_tsne[:, 1], hue=y, palette=sns.color\_palette("hls", 3))  plt.title("t-SNE визуализация Iris датасета Перплексия=30")  plt.show() |







# Задание 7.

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.

С помощью UMAP получили результат, очень похожий на то, что дает t-SNE.

n\_neighbors=5, min\_dist=0.1, time=1.9389050006866455

n\_neighbors=5, min\_dist=0.6, time=1.6654314994812012

n\_neighbors=25, min\_dist=0.1, time=1.8912675380706787

n\_neighbors=25, min\_dist=0.6, time=1.8650519847869873

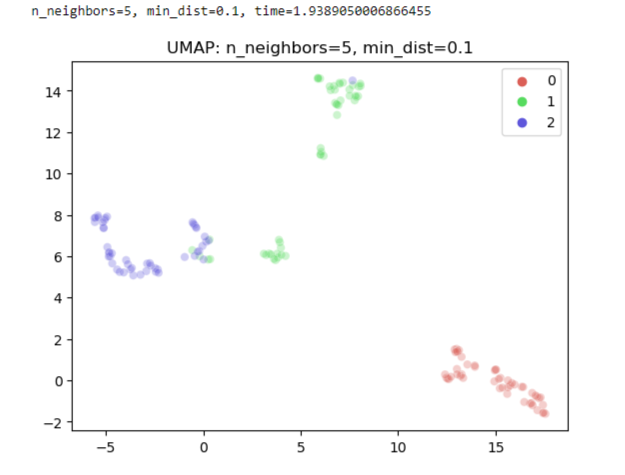
n\_neighbors=50, min\_dist=0.1, time=2.0653607845306396

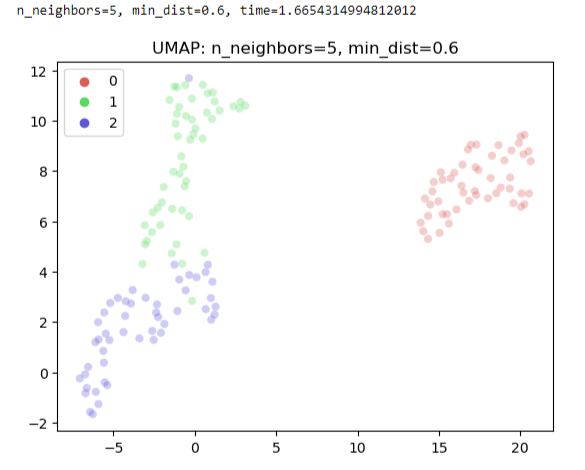
n\_neighbors=50, min\_dist=0.6, time=1.7542290687561035

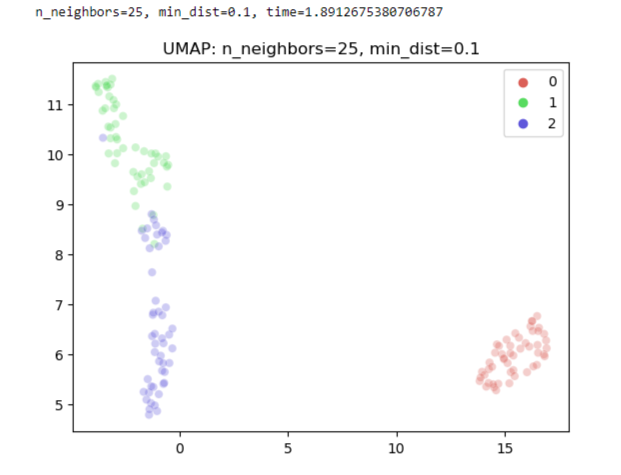
t-SNE time=0.5671863555908203

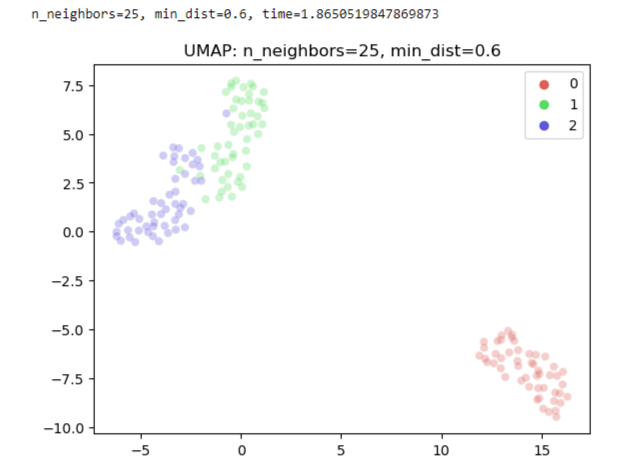
*Листинг 7 – Задание 7*

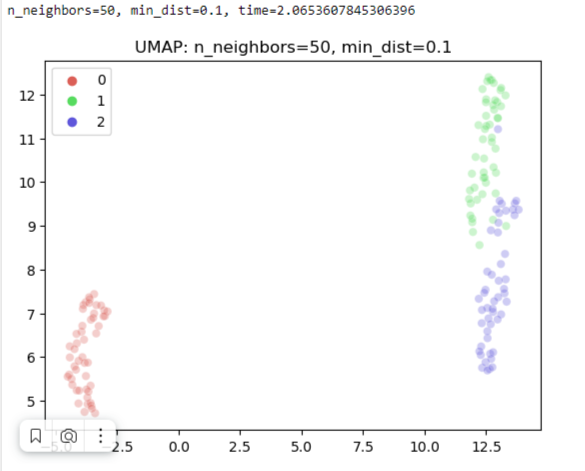
|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  import time  from sklearn.datasets import load\_iris  import umap.umap\_ as umap  # Загрузка набора данных Iris  iris = load\_iris()  X = iris.data  y = iris.target  n\_neighbors = [5, 25, 50]  min\_dist = [0.1, 0.6]  for n in n\_neighbors:  for md in min\_dist:  start\_time = time.time()  embedding = umap.UMAP(n\_neighbors=n, min\_dist=md).fit\_transform(X)  end\_time = time.time()  elapsed\_time = end\_time - start\_time  print(f"n\_neighbors={n}, min\_dist={md}, time={elapsed\_time}")    sns.scatterplot(  x=embedding[:, 0], y=embedding[:, 1],  hue=y,  palette=sns.color\_palette("hls", 3),  legend="full",  alpha=0.3  )  plt.title(f"UMAP: n\_neighbors={n}, min\_dist={md}")  plt.show()    from sklearn.manifold import TSNE  start\_time = time.time()  embedding\_tsne = TSNE(n\_components=2).fit\_transform(X)  end\_time = time.time()  elapsed\_time\_tsne = end\_time - start\_time  print(f"t-SNE time={elapsed\_time\_tsne}")  sns.scatterplot(  x=embedding\_tsne[:, 0], y=embedding\_tsne[:, 1],  hue=y,  palette=sns.color\_palette("hls", 3),  legend="full",  alpha=0.3  )  plt.title(f"t-SNE")  plt.show()) |

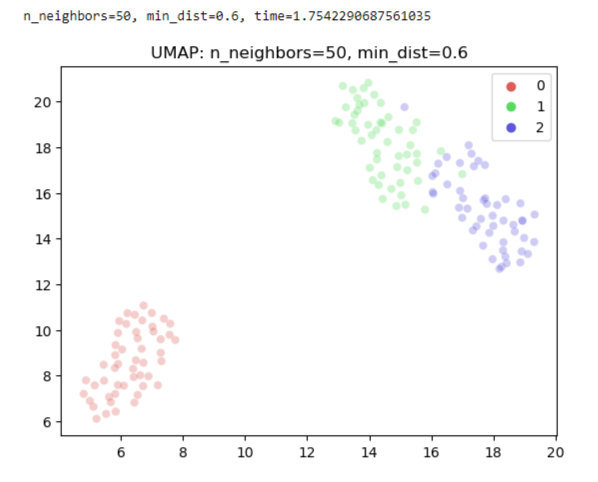


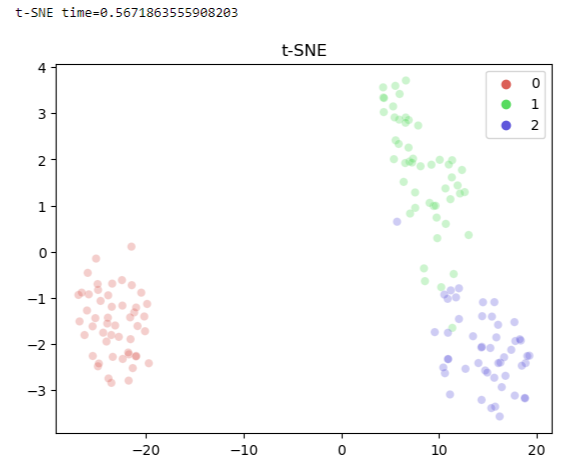












# Вывод

В данной практической работе был использован датасет Успеваемости обучающихся (с данными о поле, этнической принадлежности, оценках по математике, письму, чтению). На основе этих данных были построены: столбчатая диаграмма, круговая диаграмма, линейные графики. С помощью них можно было определить зависимость одни оценок от других, а также количество оценок по математике на этническую группу. При работе с алгоритмами визуализации данных (t-SNE и UMAP) был использован готовый набор данных "Iris". t-SNE и UMAP работают похожим образом - они стремятся сохранить относительные расстояния между точками данных, но при этом учитывают их вероятности или плотность. Это позволило сгруппировать похожие данные ближе друг к другу, а различные данные - дальше друг от друга.