|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА** – **Российский технологический университет»**  **РТУ МИРЭА** | | | | |
| Институт информационных технологий (ИТ) | |
| Кафедра прикладной математики | |

|  |  |
| --- | --- |
| **ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 2** | |
| **по дисциплине «Технологии и инструментарий анализа больших данных»** | |
|  | |
| Выполнил студент группы ИКБО-20-21 | Сидоров С.Д. |
| Проверил ассистент кафедры ПМ ИИТ | Тетерин Н.Н. |

**Задание 1**

Найти и выгрузить многомерные данные (с большим количеством признаков – столбцов) с использованием библиотеки pandas. В отчёте описать найденные данные.   
 Были выбраны данные о домах в Калифорнии, включая медианную стоимость, среднее количество комнат, количество жильцов и другие параметры.

**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.

Код к заданию 2 представлен в листинге 1

Листинг 1 – Код для получения данных о датасете

california\_data = fetch\_california\_housing(as\_frame=True)

data = california\_data.data

target = california\_data.target

data['MedHouseVal'] = target

print("Информация о данных:")

print(data.info())

print("\nПервые строки данных:")

print(data.head())

if data.isnull().sum().sum() > 0:

print("Обнаружены пропущенные значения. Выполняется их удаление...")

data = data.dropna()

else:

print("Пропущенные значения не обнаружены.")

Результат выполнения представлен на рисунке 1.

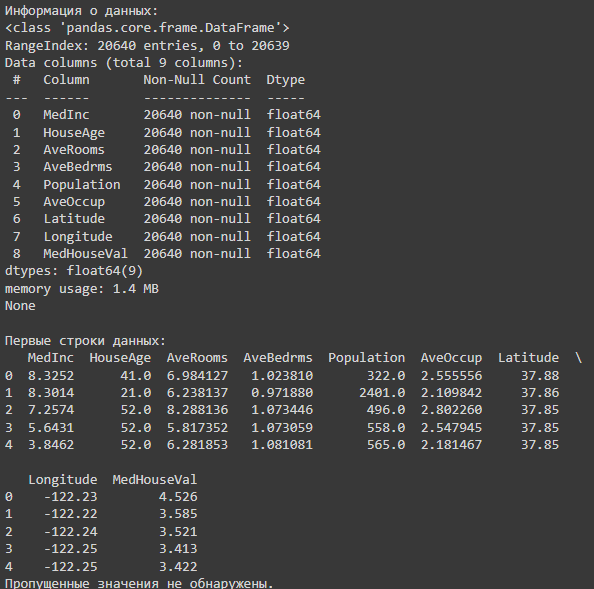


Рисунок 1 – Данные о наборе данных

**Задание 3**

Построить столбчатую диаграмму (.bar) с использованием модуля

graph\_objs из библиотеки Plotly с определенными параметрами.

Код к заданию 3 представлен на листинге 2.

Листинг 2 – Код для создания столбчатой диаграммы

numerical\_columns = data.select\_dtypes(include=[float, int]).columns.difference(['Population'])

bar\_data = data[numerical\_columns].copy()

mean\_values = bar\_data.mean()

plt.figure(figsize=(12, 8))

colors = plt.cm.get\_cmap('tab20', len(mean\_values))

plt.bar(mean\_values.index, mean\_values.values, color=colors(range(len(mean\_values))), edgecolor='black')

plt.title('Средние значения для различных характеристик домов (без Population)', fontsize=20)

plt.xlabel('Характеристика', fontsize=16)

plt.ylabel('Среднее значение', fontsize=16)

plt.xticks(rotation=90, fontsize=12)

plt.grid(axis='y', linestyle='--', linewidth=0.7)

plt.show()

Столбчатая диаграмм представлена на рисунке 2.

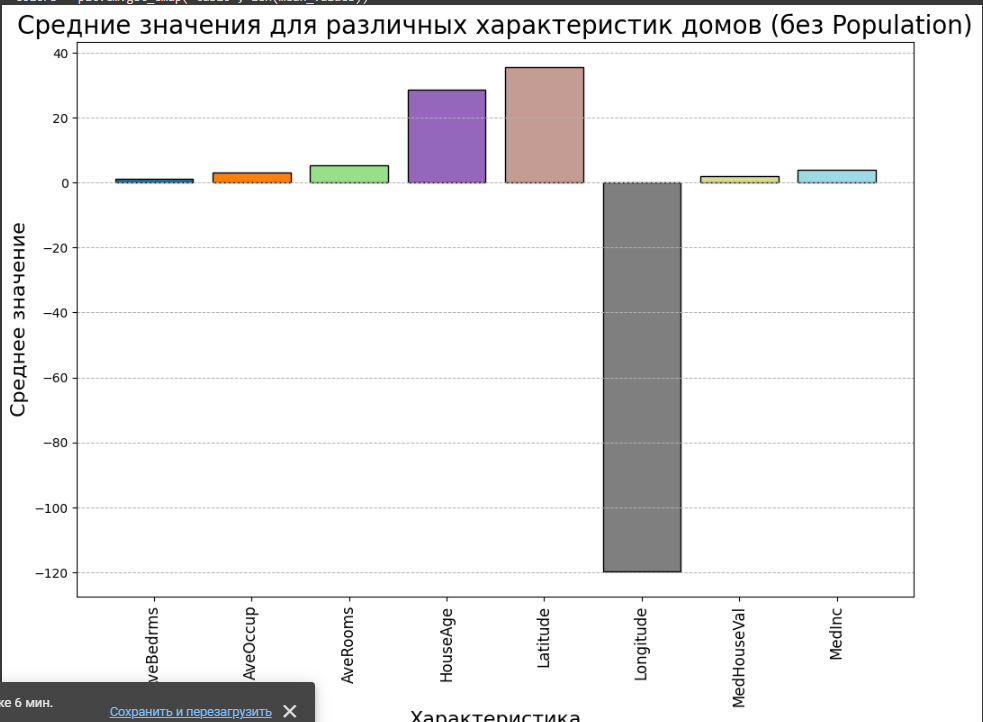


Рисунок 4 – Пример работы

**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 3 – Код для создания круговой диаграммы

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

plt.pie(mean\_values[:5], labels=mean\_values.index[:5], autopct='%1.1f%%', colors=['lightcoral', 'lightskyblue', 'yellowgreen', 'gold', 'violet'],

wedgeprops={'edgecolor': 'black', 'linewidth': 2})

plt.title('Распределение средних значений (топ 5 характеристик домов)', fontsize=20)

plt.show()

Результат работы представлен на рисунке 3.



Рисунок 3 – Круговая диаграмма

**Задание 5**

Построить линейные графики, взять один из параметров и определить

зависимость между другими несколькими (от 2 до 5) показателями с

использованием библиотеки matplotlib.

Код к заданию 5 представлен на в листинге 4.

Листинг 4 – Код для создания линейной диаграммы

plt.figure(figsize=(14, 6))

plt.plot(data['AveRooms'][:100], label='Среднее количество комнат', marker='^', linestyle='-.', color='green', markerfacecolor='white',

markeredgecolor='black', markeredgewidth=2)

plt.plot(data['AveOccup'][:100], label='Среднее количество жильцов', marker='d', linestyle=':', color='purple', markerfacecolor='white',

markeredgecolor='black', markeredgewidth=2)

plt.plot(data['MedHouseVal'][:100], label='Медианная стоимость дома', marker='o', linestyle='-', color='orange', markerfacecolor='white',

markeredgecolor='black', markeredgewidth=2)

plt.plot(data['MedInc'][:100], label='Медианный доход', marker='s', linestyle='--', color='blue', markerfacecolor='white',

markeredgecolor='black', markeredgewidth=2)

plt.grid(True, linewidth=2, color='mistyrose')

plt.title('Линейный график различных показателей дома', fontsize=20)

plt.xlabel('Индекс', fontsize=16)

plt.ylabel('Значение', fontsize=16)

plt.xticks(rotation=315, fontsize=14)

plt.yticks(fontsize=14)

plt.legend()

plt.show()

Результат выполнения представлен на рисунке 4.

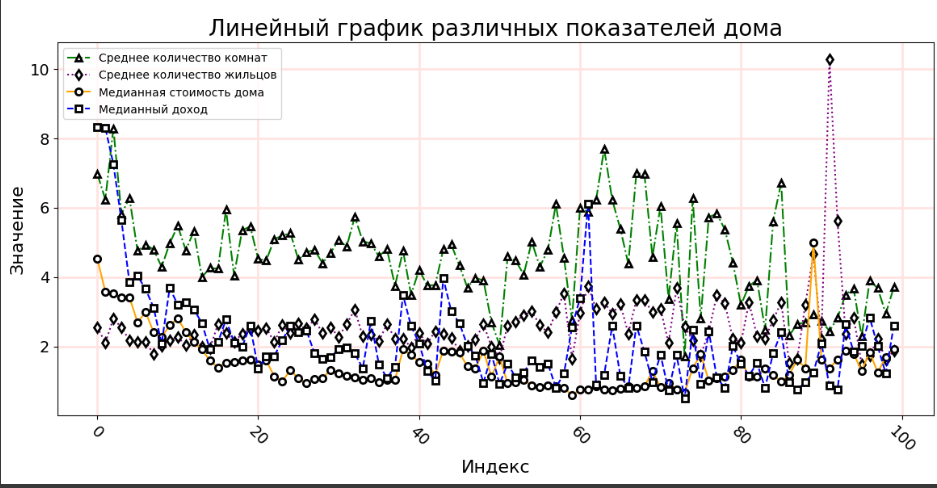


Рисунок 4 – Линейный график

**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.

Код к заданию 6 представлен в листинге 5

Листинг 5 – Визуализация многомерных данных

sample\_size = len(data\_digits)

data\_subset = data\_digits[:sample\_size]

target\_subset = target\_digits[:sample\_size]

tsne\_params = [

{'perplexity': 5, 'learning\_rate': 100, 'n\_iter': 1000},

{'perplexity': 30, 'learning\_rate': 200, 'n\_iter': 2000},

{'perplexity': 50, 'learning\_rate': 300, 'n\_iter': 3000},

{'perplexity': 10, 'learning\_rate': 150, 'n\_iter': 1500},

{'perplexity': 40, 'learning\_rate': 250, 'n\_iter': 2500}

]

for params in tsne\_params:

start\_time\_tsne = time.time()

tsne = TSNE(n\_components=2, perplexity=params['perplexity'], learning\_rate=params['learning\_rate'], n\_iter=params['n\_iter'], random\_state=42)

tsne\_result = tsne.fit\_transform(data\_subset)

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

plt.scatter(tsne\_result[:, 0], tsne\_result[:, 1], c=target\_subset, cmap='viridis', s=5)

plt.title(f"t-SNE Digits: perplexity={params['perplexity']}, learning\_rate={params['learning\_rate']}, n\_iter={params['n\_iter']}")

plt.colorbar(label='Цифра (класс)')

plt.show()

print(f"Время работы t-SNE с параметрами perplexity={params['perplexity']}, learning\_rate={params['learning\_rate']}, n\_iter={params['n\_iter']}: {time.time() - start\_time\_tsne:.4f} секунд")

Результат работы представлен на рисунках 5 - 9.

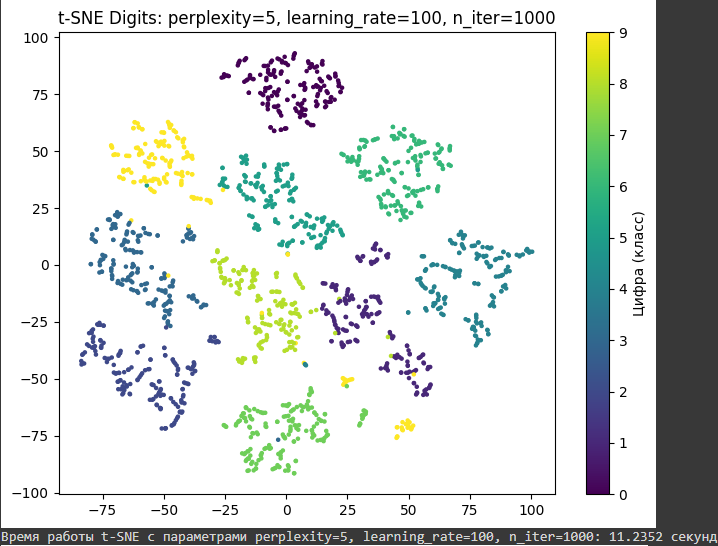


Рисунок 5 –Визуализация t-SNE

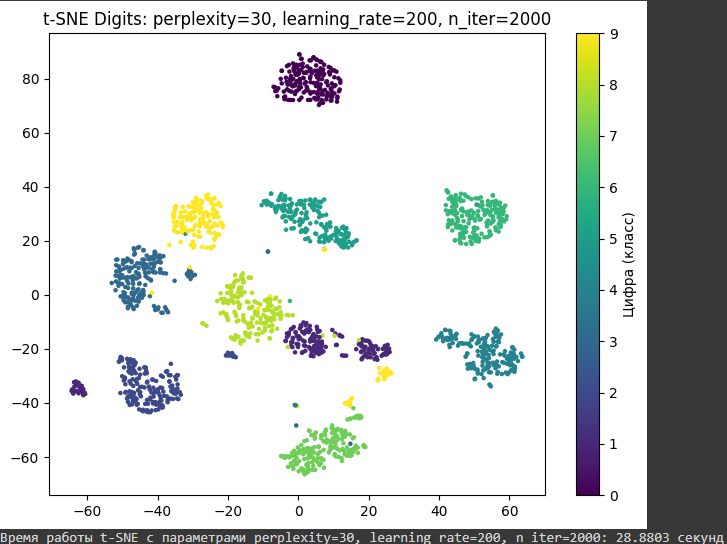


Рисунок 6 -Визуализация t-SNE

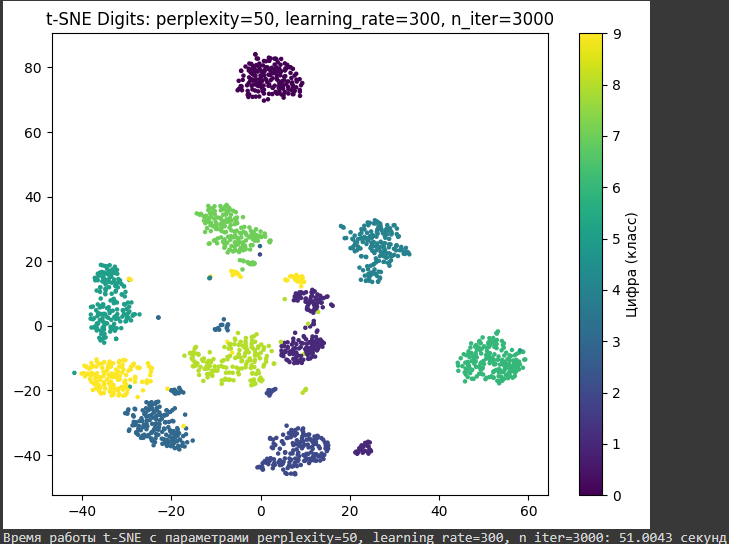


Рисунок 7 - Визуализация t-SNE

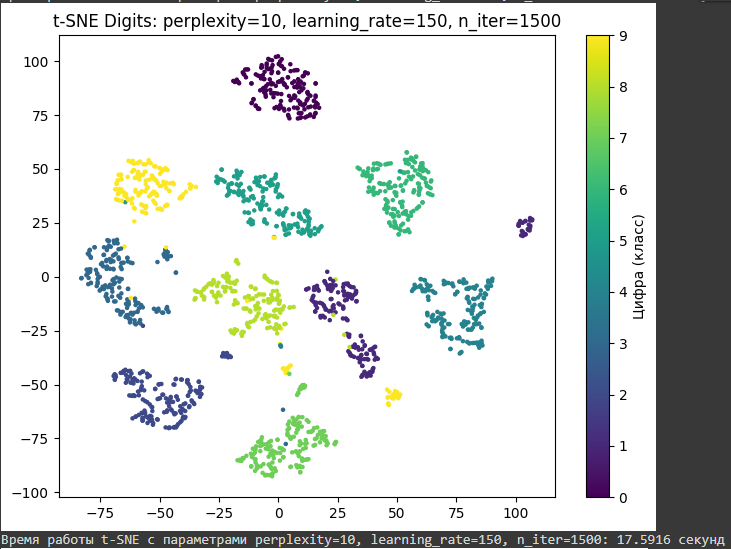


Рисунок 8 - Визуализация t-SNE

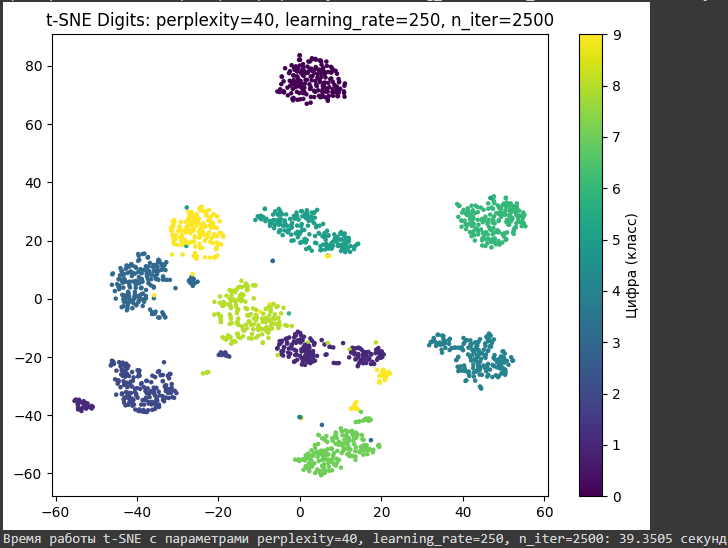


Рисунок 9 - Визуализация t-SNE

**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. Код для выполнения задания представлен в листинге 6.

Листинг 6 – Код для создания визуализации с помощью UMAP

umap\_params = [

{'n\_neighbors': 5, 'min\_dist': 0.1, 'n\_components': 2},

{'n\_neighbors': 15, 'min\_dist': 0.3, 'n\_components': 2},

{'n\_neighbors': 30, 'min\_dist': 0.5, 'n\_components': 2},

{'n\_neighbors': 10, 'min\_dist': 0.2, 'n\_components': 2},

{'n\_neighbors': 20, 'min\_dist': 0.4, 'n\_components': 2}

]

for params in umap\_params:

start\_time\_umap = time.time()

reducer = umap.UMAP(n\_neighbors=params['n\_neighbors'], min\_dist=params['min\_dist'], n\_components=params['n\_components'], random\_state=42)

umap\_result = reducer.fit\_transform(data\_subset)

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

plt.scatter(umap\_result[:, 0], umap\_result[:, 1], c=target\_subset, cmap='viridis', s=5)

plt.title(f"UMAP Digits: n\_neighbors={params['n\_neighbors']}, min\_dist={params['min\_dist']}")

plt.colorbar(label='Цифра (класс)')

plt.show()

print(f"Время работы UMAP с параметрами n\_neighbors={params['n\_neighbors']}, min\_dist={params['min\_dist']}: {time.time() - start\_time\_umap:.4f} секунд")

Результат выполнения представлен на рисунках 10 – 14.

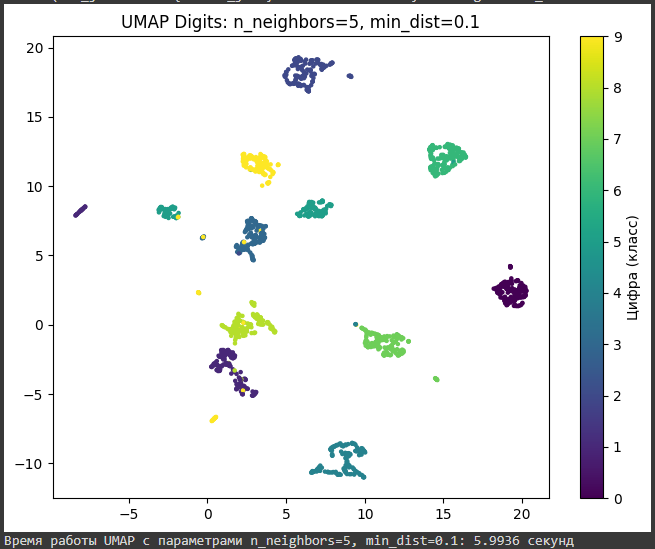


Рисунок 10 – Визуализация UMAP

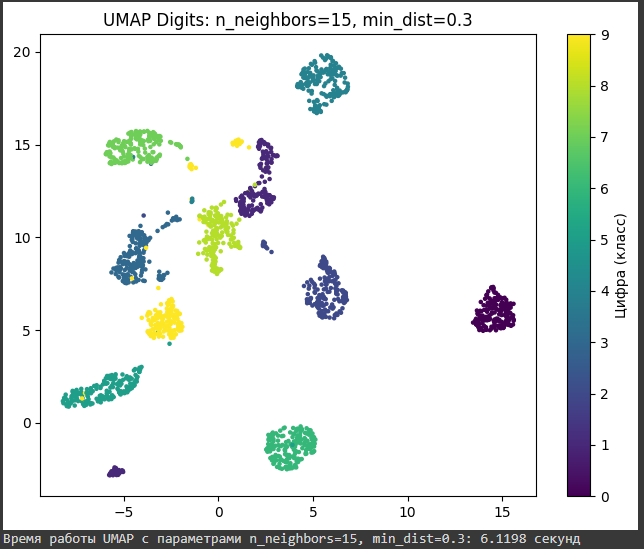


Рисунок 11 – Визуализация UMAP

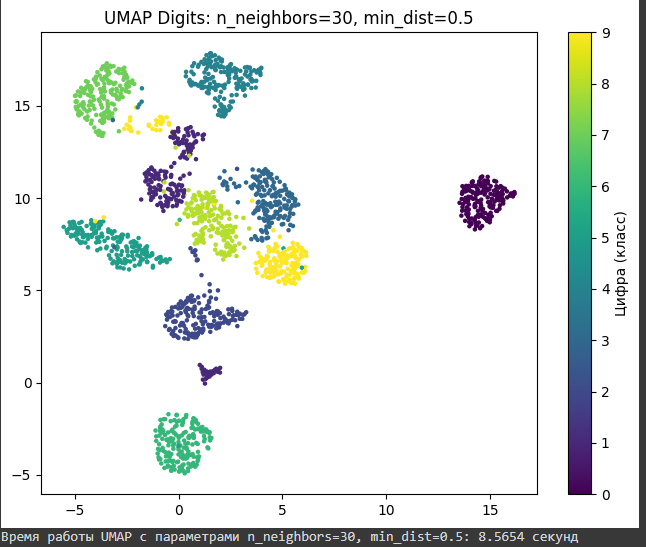


Рисунок 12 – Визуализация UMAP

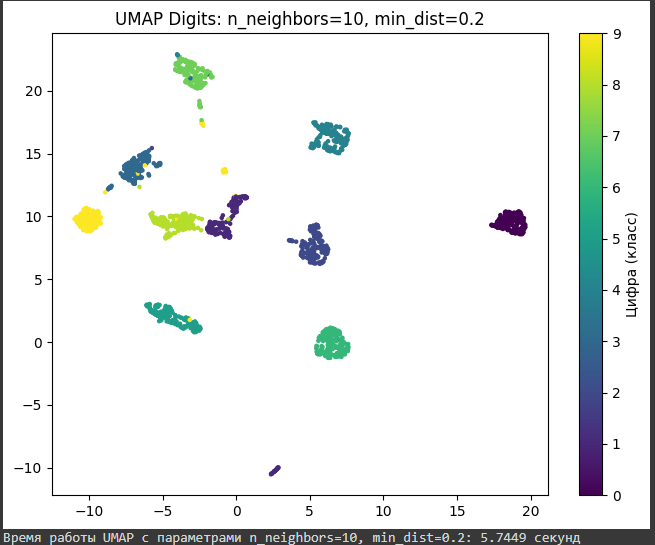


Рисунок 13 – Визуализация UMAP

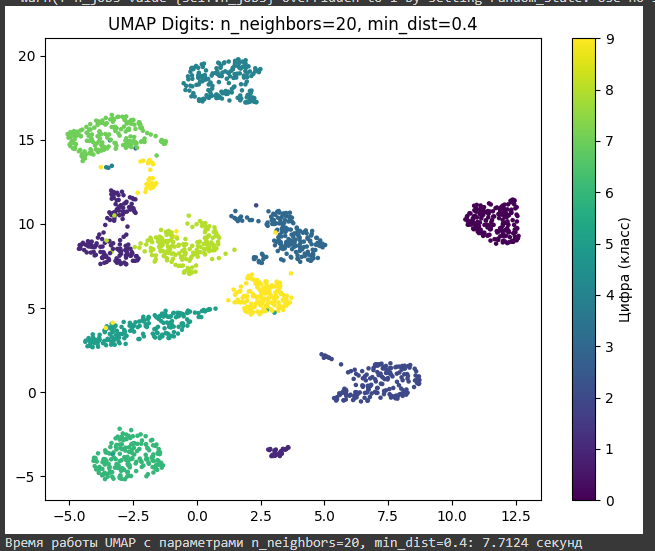


Рисунок 14 – Визуализация UMAP

С точки зрения времени, визуализация UMAP на используемых данных при достаточно схожих итоговых кластерах происходит в не сколько раз быстрее.