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 Фомичев Р.А.

Проверил ассистент кафедры ПМ ИИТ Тетерин Н.Н.

Москва 2024

Задание 1

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

В качестве многомерных данных будем использовать набор данных о самых прослушиваемых треках в Spotify за 2023 год. Данные можно получить по ссылке: https://www.kaggle.com/datasets/nelgiriyewithana/top-spotify-songs-2023.

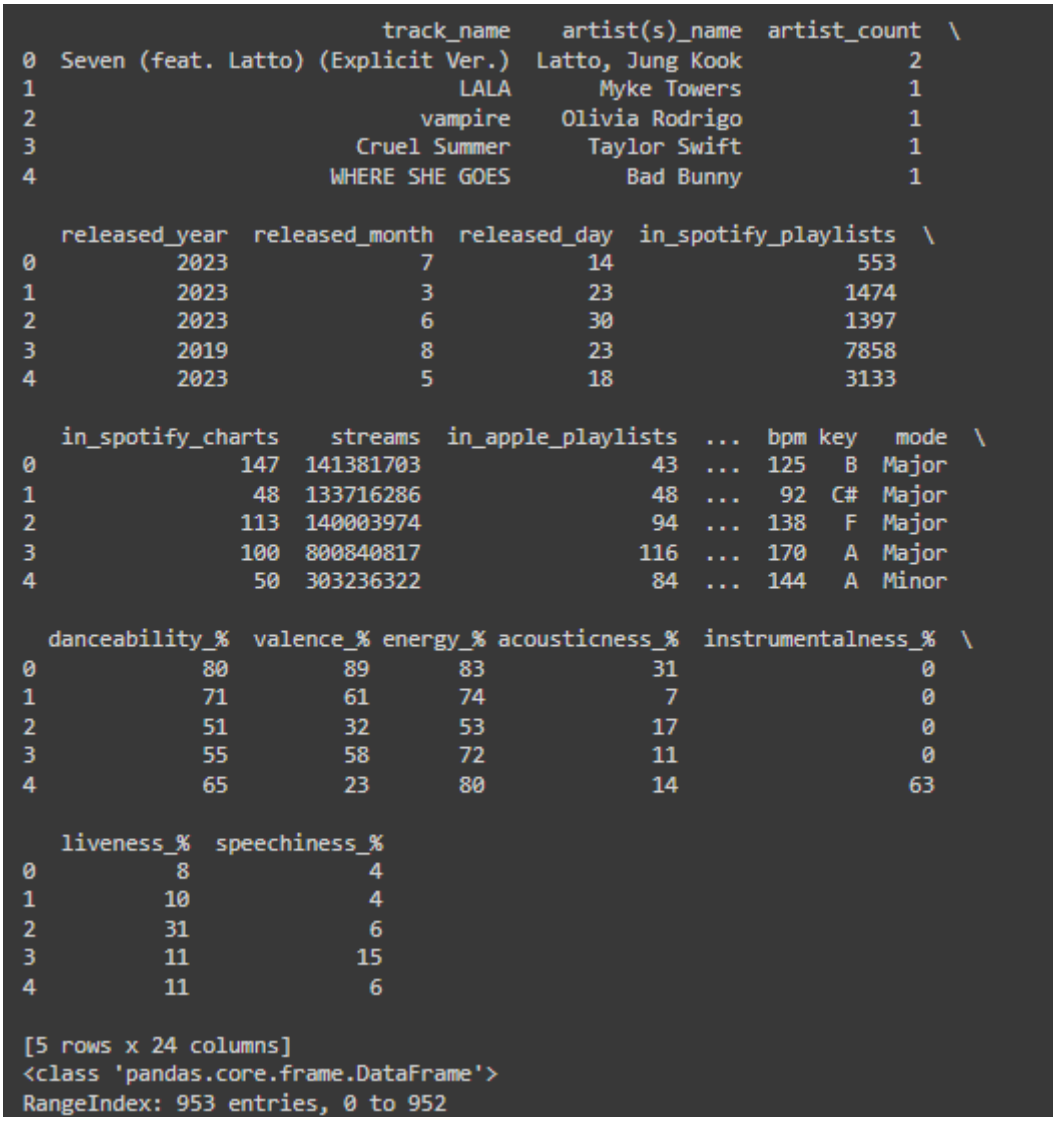
Задание 2

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 представлен на рисунке 1.

```
import pandas as pd

# Загрузка данных
data = pd.read_csv('sample_data/spotify-2023.csv', encoding='latin-1')

# Вывести первые 5 строк данных
print(data.head())

# Показать общую информацию о данных, включая пропущенные значения
print(data.info())

# Проверка наличия пропущенных значений
print(data.isnull().sum())

# Предобработка данных:
data.dropna(inplace=True)
```

Рисунок 1 – Код

Результат работы программы представлен на рисунках 2-3.



```
                          track_name    artist(s)_name  artist_count  \
0  Seven (feat. Latto) (Explicit Ver.)  Latto, Jung Kook             2
1                                 LALA       Myke Towers             1
2                              vampire    Olivia Rodrigo             1
3                          Cruel Summer      Taylor Swift             1
4                       WHERE SHE GOES         Bad Bunny             1

   released_year  released_month  released_day  in_spotify_playlists  \
0           2023               7            14                   553
1           2023               3            23                  1474
2           2023               6            30                  1397
3           2019               8            23                  7858
4           2023               5            18                  3133

   in_spotify_charts    streams  in_apple_playlists  ...  bpm key   mode  \
0                147  141381703                  43  ...  125   B  Major
1                 48  133716286                  48  ...   92  C#  Major
2                113  140003974                  94  ...  138   F  Major
3                100  800840817                 116  ...  170   A  Major
4                 50  303236322                  84  ...  144   A  Minor

  danceability_%  valence_% energy_% acousticness_%  instrumentalness_%  \
0             80         89       83             31                   0
1             71         61       74              7                   0
2             51         32       53             17                   0
3             55         58       72             11                   0
4             65         23       80             14                  63

   liveness_%  speechiness_%
0           8              4
1          10              4
2          31              6
3          11             15
4          11              6

[5 rows x 24 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 953 entries, 0 to 952
```

Рисунок 2 – Результат работы программы

```
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   track_name            953 non-null    object
 1   artist(s)_name        953 non-null    object
 2   artist_count          953 non-null    int64
 3   released_year         953 non-null    int64
 4   released_month        953 non-null    int64
 5   released_day          953 non-null    int64
 6   in_spotify_playlists  953 non-null    int64
 7   in_spotify_charts     953 non-null    int64
 8   streams               953 non-null    object
 9   in_apple_playlists    953 non-null    int64
 10  in_apple_charts       953 non-null    int64
 11  in_deezer_playlists   953 non-null    object
 12  in_deezer_charts      953 non-null    int64
 13  in_shazam_charts      903 non-null    object
 14  bpm                   953 non-null    int64
 15  key                   858 non-null    object
 16  mode                  953 non-null    object
 17  danceability_%        953 non-null    int64
 18  valence_%             953 non-null    int64
 19  energy_%              953 non-null    int64
 20  acousticness_%        953 non-null    int64
 21  instrumentalness_%    953 non-null    int64
 22  liveness_%            953 non-null    int64
 23  speechiness_%         953 non-null    int64
dtypes: int64(17), object(7)
memory usage: 178.8+ KB
None
track_name               0
artist(s)_name           0
artist_count             0
released_year            0
released_month           0
released_day             0
in_spotify_playlists     0
in_spotify_charts        0
streams                  0
in_apple_playlists       0
in_apple_charts          0
in_deezer_playlists      0
in_deezer_charts         0
in_shazam_charts        50
bpm                      0
key                     95
mode                     0
danceability_%           0
valence_%                0
energy_%                 0
acousticness_%           0
instrumentalness_%       0
liveness_%               0
speechiness_%            0
dtype: int64
```

Рисунок 3 – Результат работы программы

Задание 3

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

- По оси X указать дату или название, по оси У указать количественный показатель.
- Сделать так, чтобы столбец принимал цвет в зависимости от значения показателя (marker=dict(color=признак, coloraxis="coloraxis")).
- Сделать так, чтобы границы каждого столбца были выделены чёрной линией с толщиной равной 2.
- Отобразить заголовок диаграммы, разместив его по центру сверху, с 20 размером текста.
- 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.
- Размер текста меток осей сделать равным 14.
- Расположить график во всю ширину рабочей области и присвоить высоту, равную 700 пикселей.
- Добавить сетку на график, сделать её цвет 'ivory' и толщину равную 2. (Можно сделать это при настройке осей с помощью gridwidth=2, gridcolor='ivory')
- Убрать лишние отступы по краям.

Код представлен на рисунке 4.

```
df = pd.read_csv("sample_data/spotify-2023.csv", encoding='latin-1')

print(df.info())
print(df.head())

grouped_df = df.groupby('released_year', as_index=False)['track_name'].count()
grouped_df.columns = ['year', 'count']

fig = go.Figure(
    go.Bar(
        x=grouped_df['year'],
        y=grouped_df['count'],
        marker=dict(color=grouped_df['count'], coloraxis="coloraxis")
    )
)

fig.update_layout(
    title={
        'text': "Total number of tracks released by year",
        'y': 0.95,
        'x': 0.5,
        'xanchor': 'center',
        'yanchor': 'top',
        'font': dict(size=20, color='white')
    },
    xaxis={
        'title': 'Year',
        'tickangle': 315,
        'titlefont': dict(size=16, color='white'),
        'tickfont': dict(size=14, color='white')
    },
    yaxis={
        'title': 'Number of tracks',
        'titlefont': dict(size=16, color='white'),
        'tickfont': dict(size=14, color='white'),
        'gridwidth': 2,
        'gridcolor': 'ivory'
    },
    height=700,
    plot_bgcolor='rgba(0,0,0,0)',
    paper_bgcolor='rgba(0,0,0,0)',
    margin=dict(l=50, r=50, t=100, b=50)
)
fig.show()
```

Рисунок 4 – Код

Результат работы программы представлен на рисунке 5.

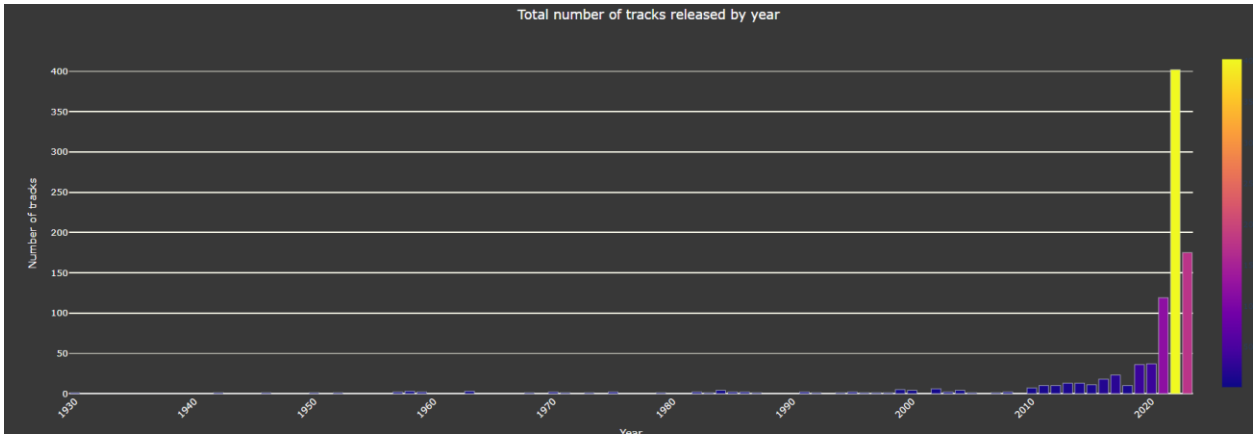


Рисунок 5 – Результат работы программы

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)

Напишем код, который отражает количество треков, находящихся в плейлистах на платформе Spotify (начиная от 25 000). Код представлен на рисунке 6.

```python
import pandas as pd
import plotly.graph_objs as go

df = pd.read_csv("sample_data/spotify-2023.csv", encoding='latin-1')
import plotly.express as px

# Группировка по нахождению в плейлистах Spotify
fat_grouped = df.groupby(['in_spotify_playlists'], as_index=False)['track_name'].count()

# Объединение всех треков с плейлистами >= 25 000
fat10_index = fat_grouped[fat_grouped['in_spotify_playlists'] >= 25000].index
print(fat10_index)

pie_df = pd.DataFrame(columns=['in_spotify_playlists', 'track_name'])

pie_df.loc[0] = ['0', fat_grouped[fat_grouped['in_spotify_playlists'] == 0]['track_name'].sum()]

for i in range(len(fat10_index)):
    in_spotify_playlists = str(fat_grouped.iloc[fat10_index[i]]['in_spotify_playlists']) + '+'
    track_name = fat_grouped.iloc[fat10_index[i]:]['track_name'].sum()

    pie_df.loc[i+1] = [in_spotify_playlists, track_name]

# Построение круговой диаграммы
fig = px.pie(
    pie_df,
    values='track_name',
    names='in_spotify_playlists',
    hole=.5,
    color='in_spotify_playlists',
    color_discrete_sequence=px.colors.sequential.RdBu,
    labels=None
)

fig.update_traces(
    textposition='inside',
    marker_line_width=2,
    marker_line_color='black'
)

fig.update_layout(
    title=dict(
        text='Число треков в зависимости от нахождения в плейлистах Spotify',
        x=0.5,
        font=dict(size=20)
    ),
    margin=dict(l=0, r=0, t=50, b=0)
)

fig.show()
```

Рисунок 6 – Код

Результат работы программы представлен на рисунке 7.

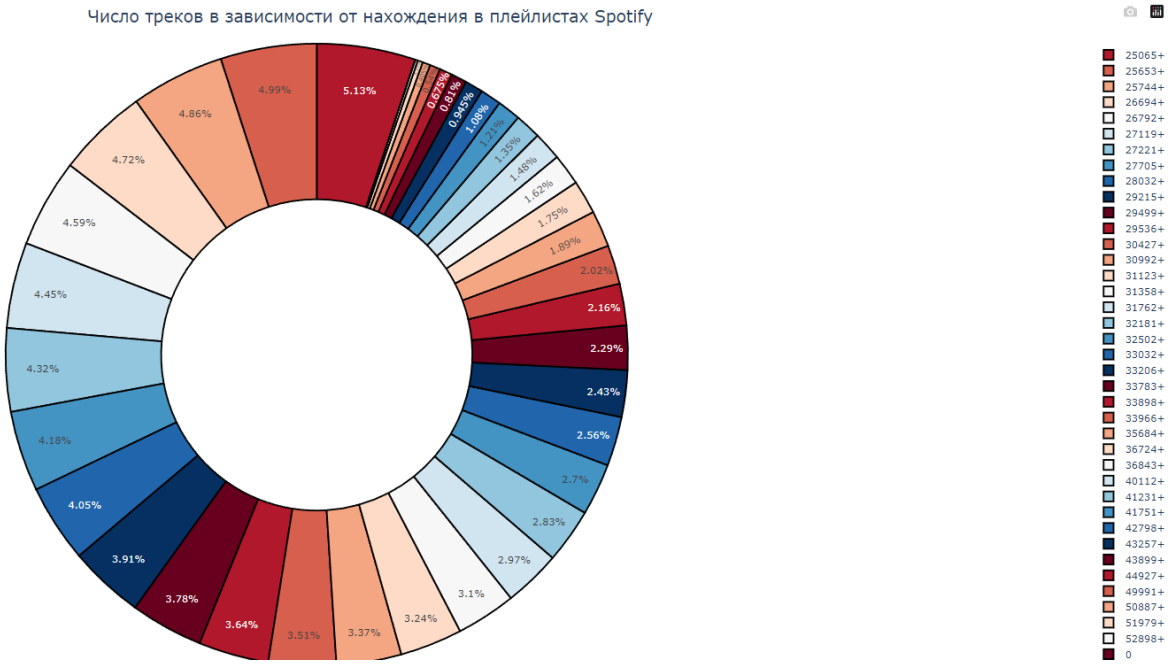


Рисунок 7 – результат работы программа

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.

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

Построим линейный график, чтобы узнать зависимость числа релизов от года. Код представлен на рисунке 8.

```python
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("spotify-2023.csv", encoding='latin-1')


# Группировка по годам для подсчета числа
year_grouped = df.groupby(['released_year'], as_index=False)['track_name'].count()

# Группировка по годам и чартам
year_chart_grouped = df.groupby(['released_year', 'in_spotify_charts'], as_index=False)['track_name'].count()

# Выбор информации за последние 20 лет
years = year_grouped.iloc[-20:]['released_year']
tracks = year_grouped.iloc[-20:]['track_name']

# Подписи осей
plt.xlabel('Год')
plt.ylabel('Число треков')

# График
plt.plot(years, tracks,
         color='crimson',
         linewidth=2,
         marker='o',
         markersize=10,
         markerfacecolor='white',
         markeredgecolor='black',
         markeredgewidth=2)

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

# Отображение графика
plt.show()
```

Рисунок 8 – Код

Результат работы программы представлен на рисунке 9.

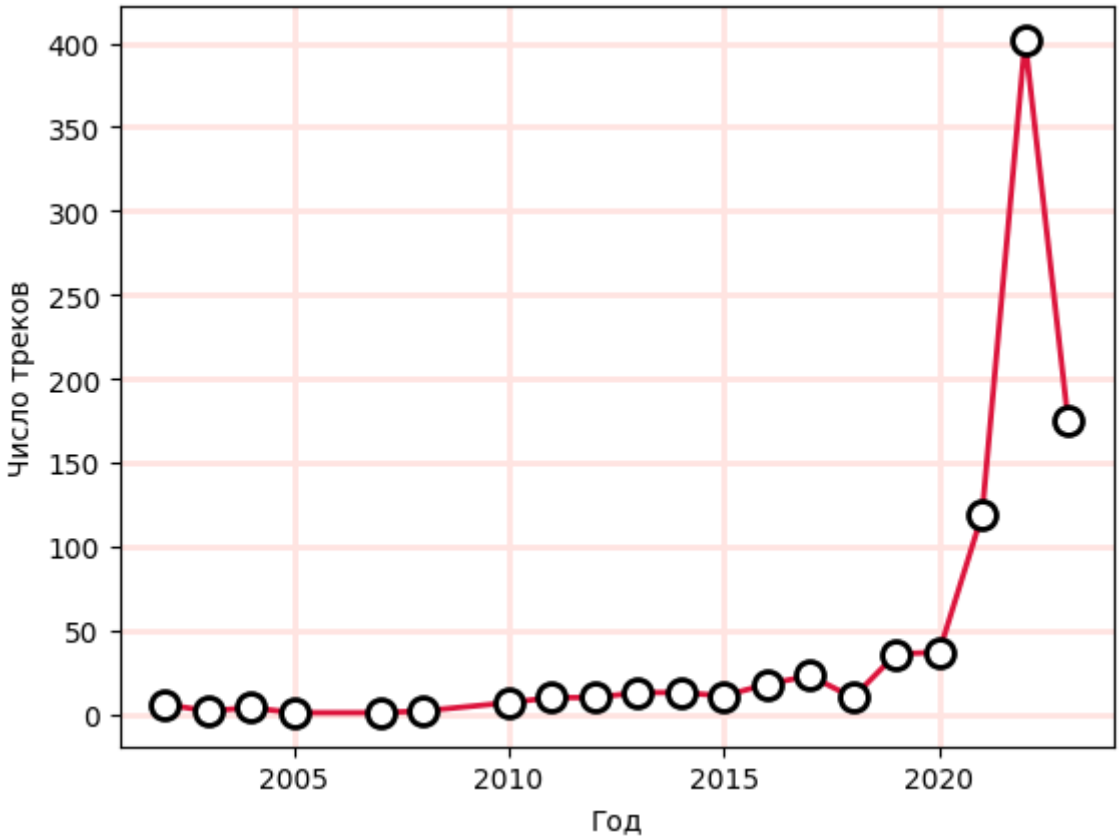


Рисунок 9 – Результат работы программы

Проанализировав получившийся график, можно сделать вывод, что рост количества, выпущенных треков происходит после 2020 года.

Построим еще один линейный график, чтобы узнать зависимость числа релизов треков от месяца. Код представлен на рисунке 10.

```python
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("spotify-2023.csv", encoding='latin-1')

# Группировка по месяцам для подсчета числа
year_grouped = df.groupby(['released_month'], as_index=False)['track_name'].count()

# Сортировка данных по месяцам
year_grouped = year_grouped.sort_values(by='released_month', key=lambda x: x.map(
    {'January': 1, 'February': 2, 'March': 3, 'April': 4, 'May': 5, 'June': 6, 'July': 7, 'August': 8,
     'September': 9, 'October': 10, 'November': 11, 'December': 12}))

month = year_grouped['released_month']
tracks = year_grouped['track_name']

# Подписи осей
plt.xlabel('Месяц')
plt.ylabel('Число треков')

# График
plt.plot(month, tracks,
         color='crimson',
         linewidth=2,
         marker='o',
         markersize=10,
         markerfacecolor='white',
         markeredgecolor='black',
         markeredgewidth=2)

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

# Отображение графика
plt.show()
```

Рисунок 10 – Код

Результат работы программы представлен на рисунке 11.

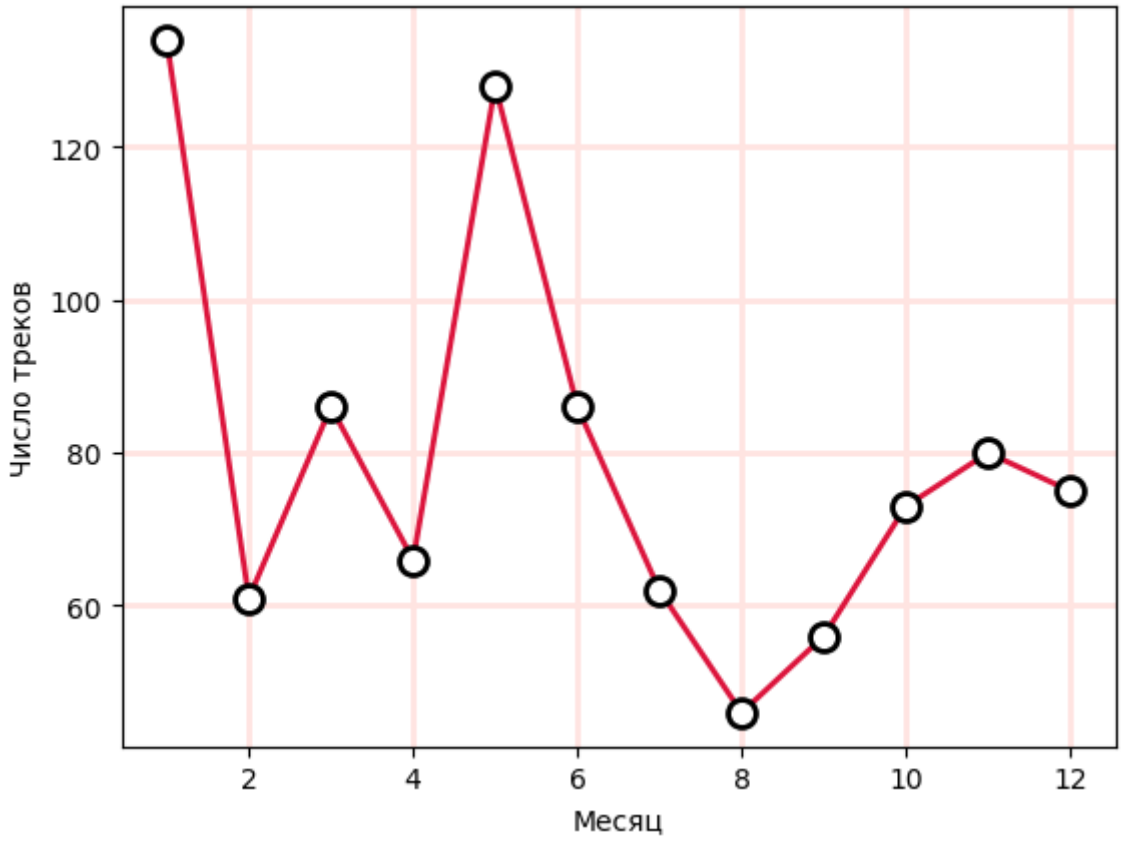


Рисунок 11 – Результат работы программы

Проанализировав получившийся график, можно сделать вывод, что самые популярные песни в большинстве пишутся либо в январе, либо в мае.

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.

Напишем код, который строит график набора данных MNIST, содержащий изображения рукописных цифр от 0 до 9. Код представлен на рисунке 12.

```python
from sklearn import datasets
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
import time

# Загрузка данных MNIST
t0 = time.time()
digits = datasets.load_digits()

# Применение t-SNE для визуализации данных в 2D-пространство
tsne = TSNE(n_components=2, random_state=0, perplexity=5)
digits_tsne = tsne.fit_transform(digits.data)

# Отображение в 2D-координатах
plt.figure(figsize=(10, 8))
plt.scatter(digits_tsne[:, 0], digits_tsne[:, 1], c=digits.target, cmap=plt.cm.get_cmap('jet', 10))
plt.colorbar(ticks=range(10))
plt.clim(-0.5, 9.5)
t1 = time.time()
plt.show()
print(f"Время обработки для модели 1: {t1-t0} секунд")
```

Рисунок 12 – Код

Результат работы программы представлен на рисунке 13.

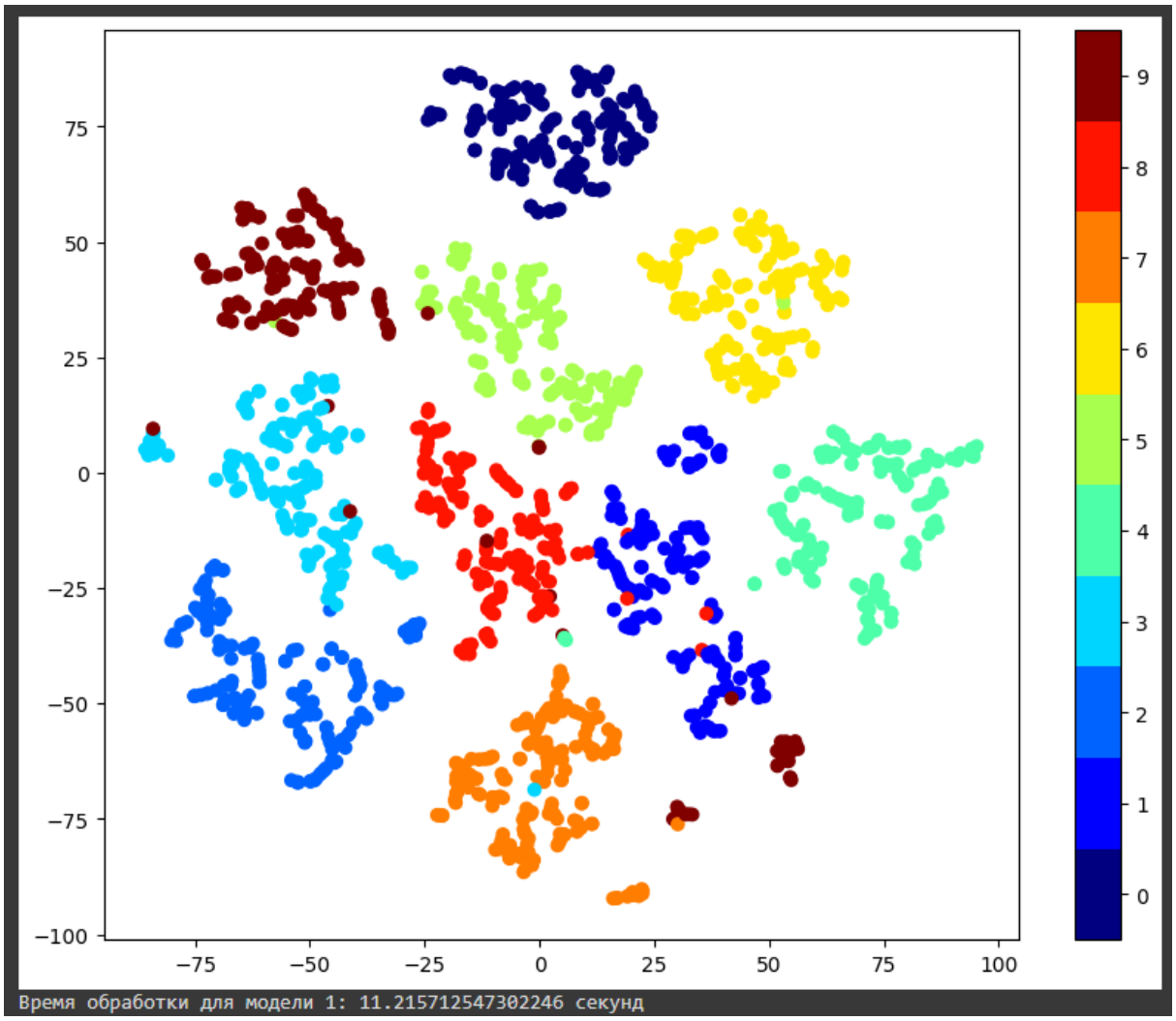


Рисунок 13 – Результат работы программы

Изменим значение перплексии с 5 на 15. Результат работы программы представлен на рисунке 14.

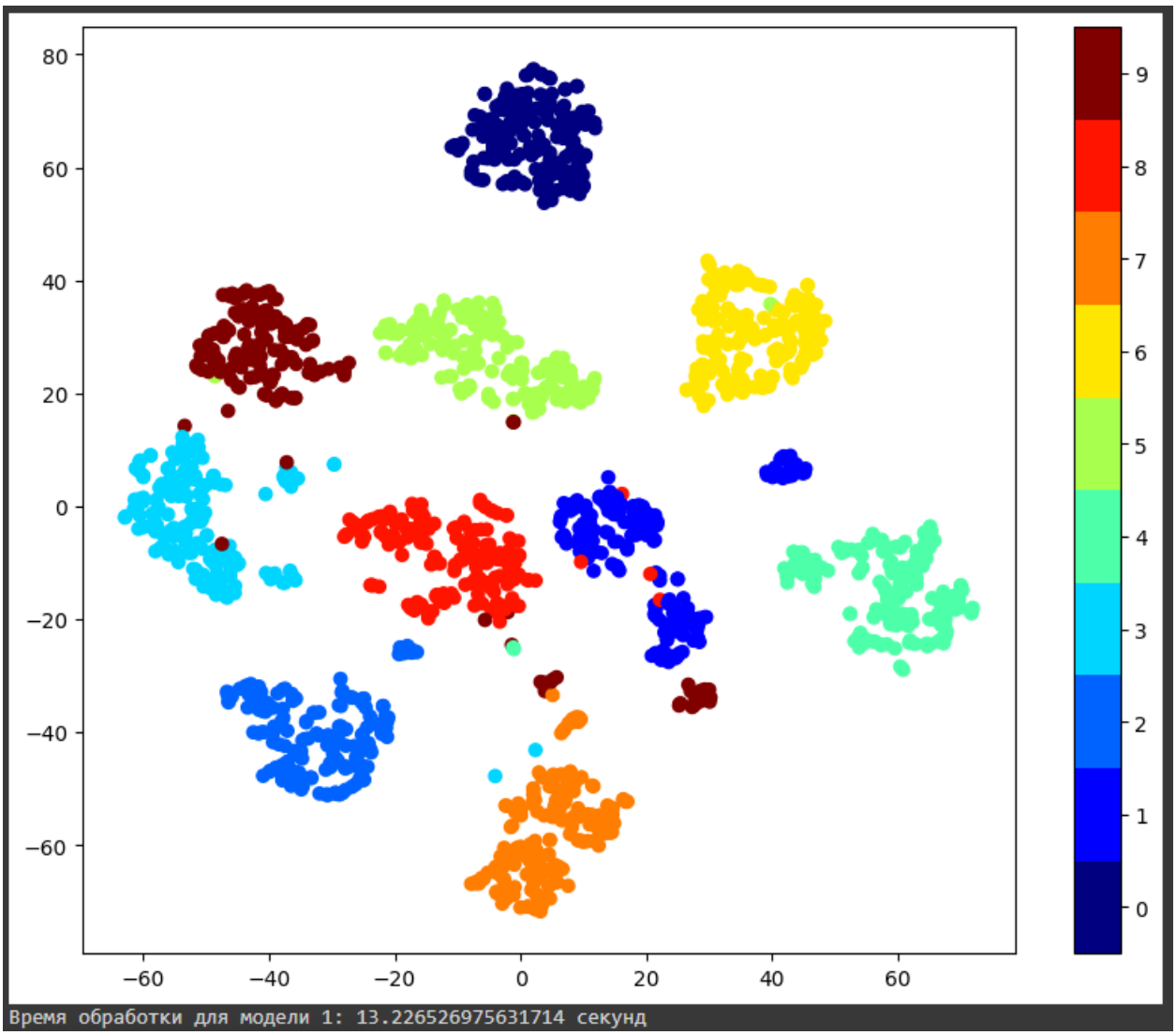


Рисунок 14 – Результат работы программы

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.

Напишем код, который строит график набора данных MNIST, содержащий изображения рукописных цифр от 0 до 9, с помощью библиотеки umap. Код представлен на рисунке 15.

```python
import umap.umap_ as umap
from sklearn import datasets
import matplotlib.pyplot as plt

import time

# Загрузка данных MNIST
digits = datasets.load_digits()
# Параметры алгоритма
n_neighbors_list = [5, 10, 15]
min_dist_list = [0.1, 0.25, 0.5]

for n_neighbors in n_neighbors_list:
    for min_dist in min_dist_list:
        # Измерение времени работы алгоритма
        start_time = time.time()
        # Применение UMAP для визуализации данных в 2D-пространство
        reducer = umap.UMAP(n_neighbors=n_neighbors, min_dist=min_dist)
        digits_umap = reducer.fit_transform(digits.data)
        # Отображение в 2D-координатах
        plt.figure(figsize=(8, 6))
        plt.scatter(digits_umap[:, 0], digits_umap[:, 1], c=digits.target, cmap=plt.cm.get_cmap('jet', 10))
        plt.colorbar(boundaries=range(11))
        plt.clim(-0.5, 9.5)
        plt.title('n_neighbors = ' + str(n_neighbors) + ', min_dist = ' + str(min_dist))

        # Время работы алгоритма
        print('n_neighbors =', n_neighbors, ', min_dist =', min_dist,
              ', время работы: %.2f сек' % (time.time() - start_time))

        plt.show()
```

Рисунок 15 – Код

Результат работы программы представлен на рисунках 16 - 24.

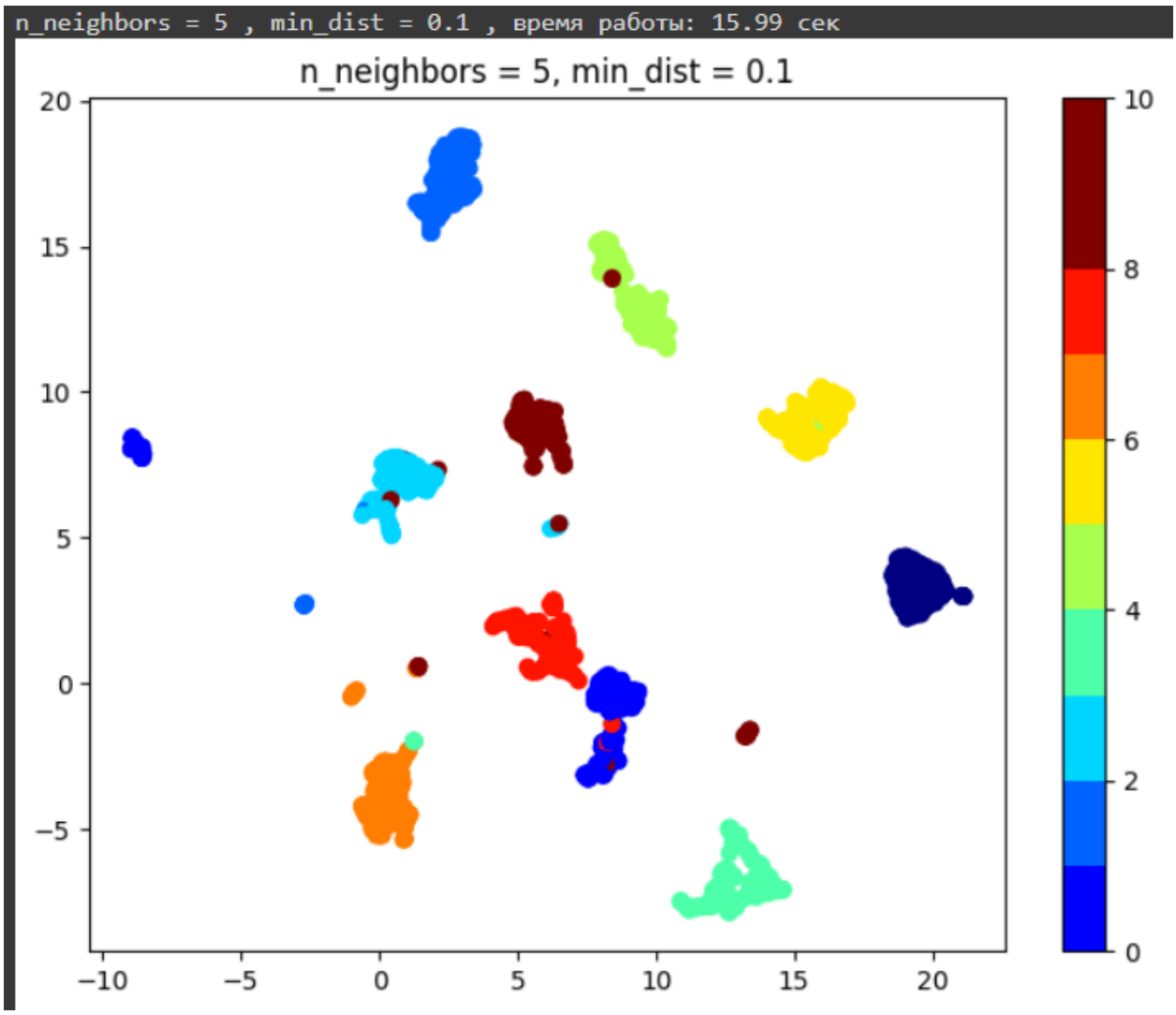


Рисунок 16 – Результат работы программы

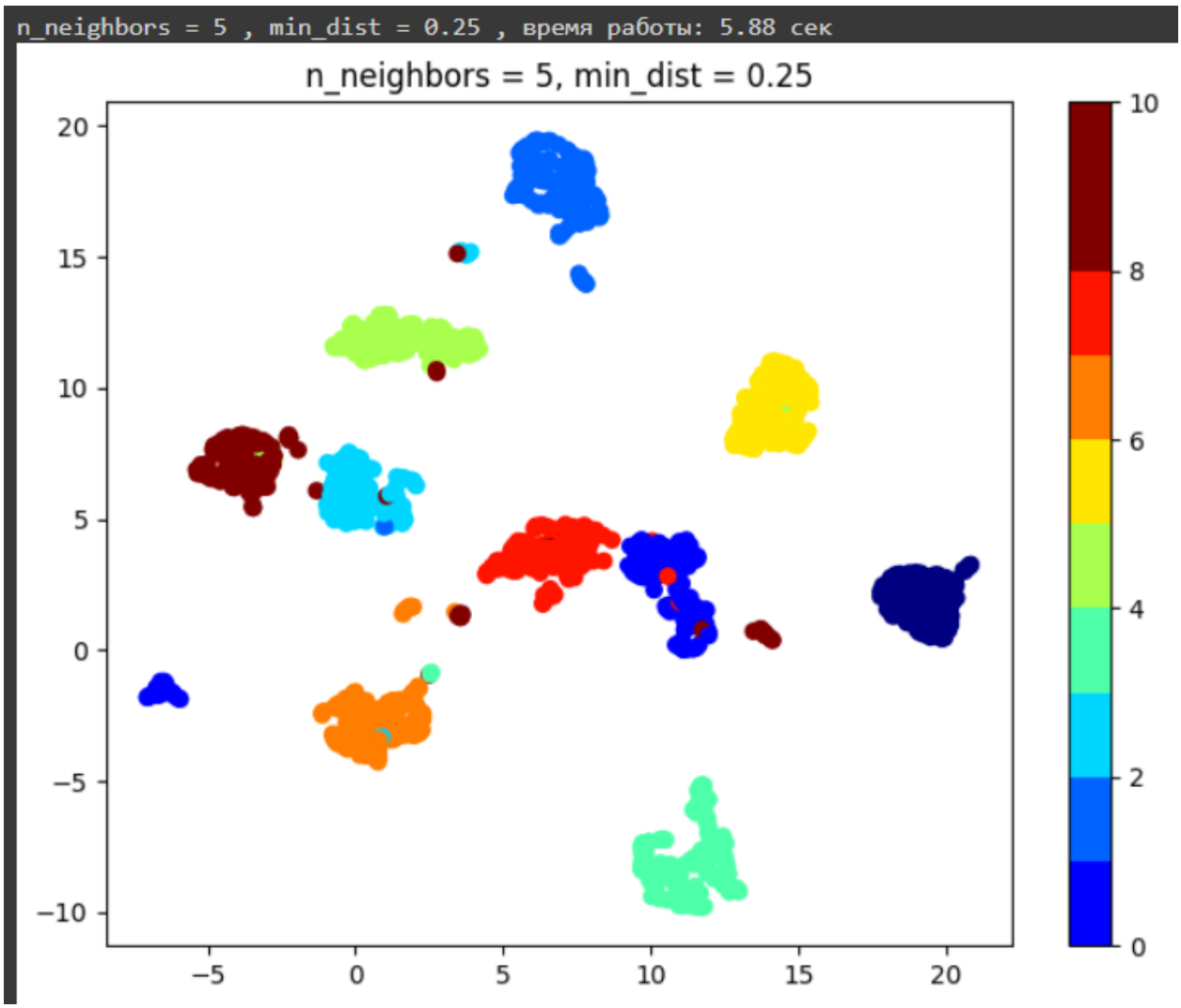


Рисунок 17 – Результат работы программы

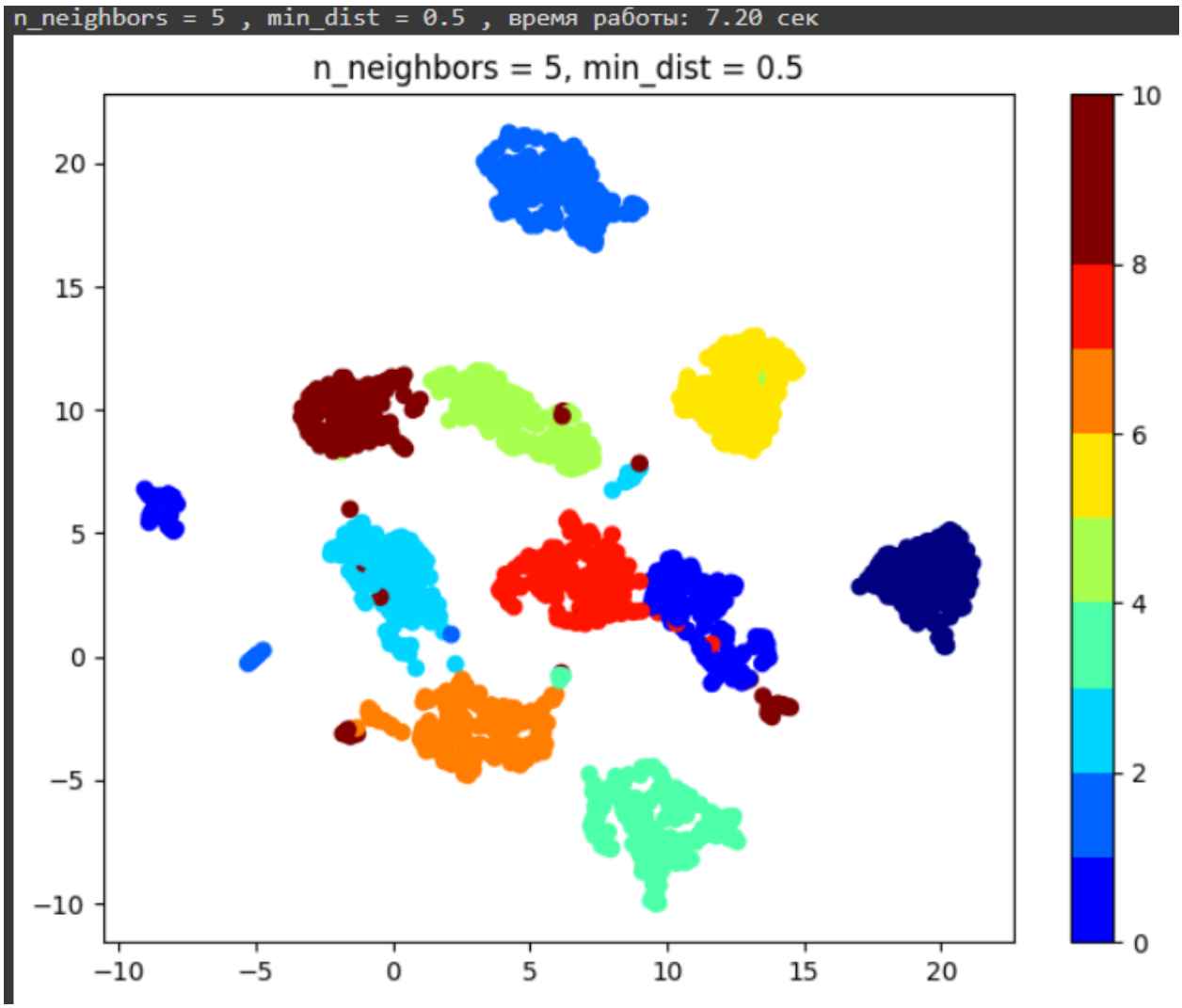


Рисунок 18 – Результат работы программы

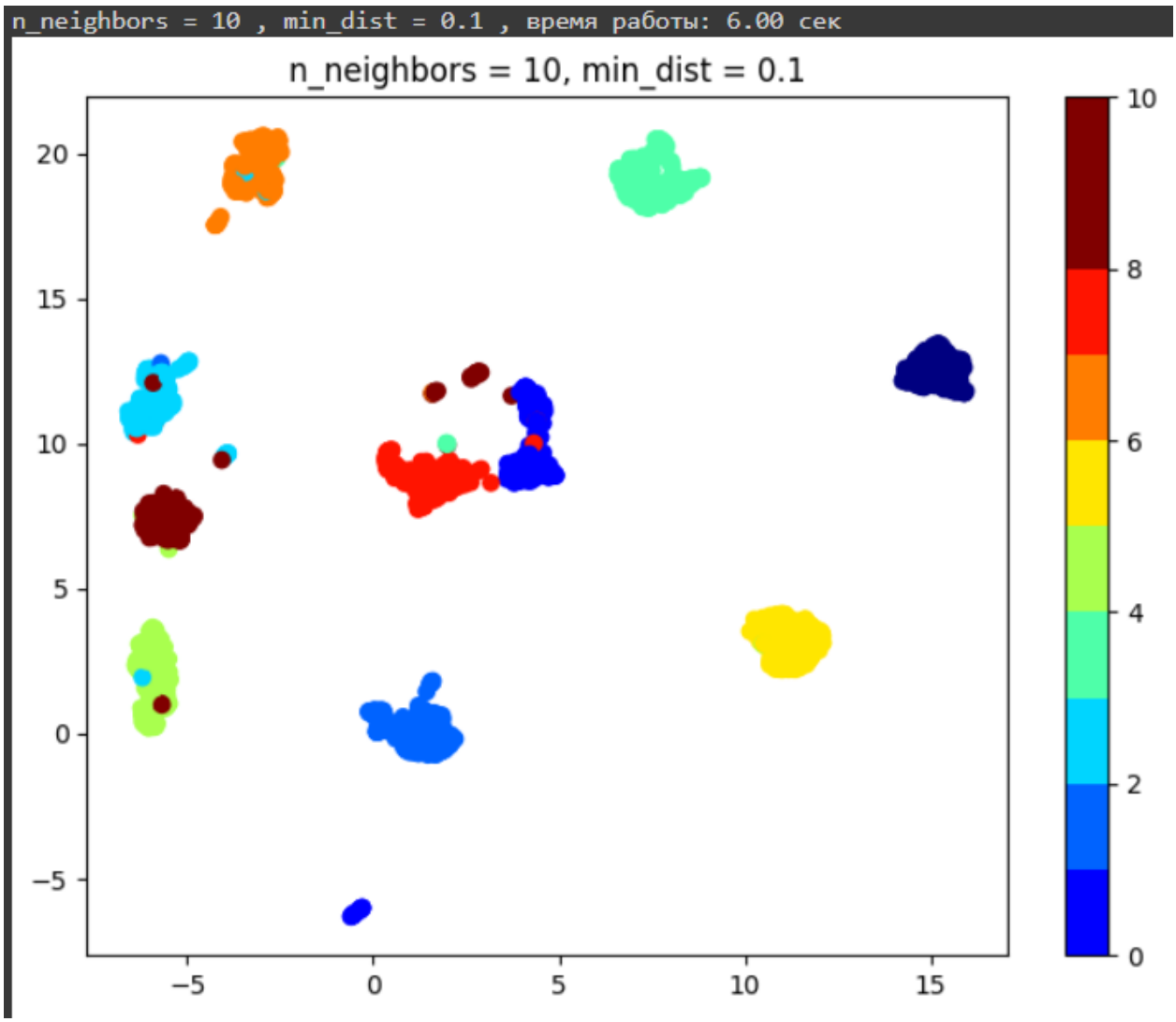


Рисунок 19 – Результат работы программы

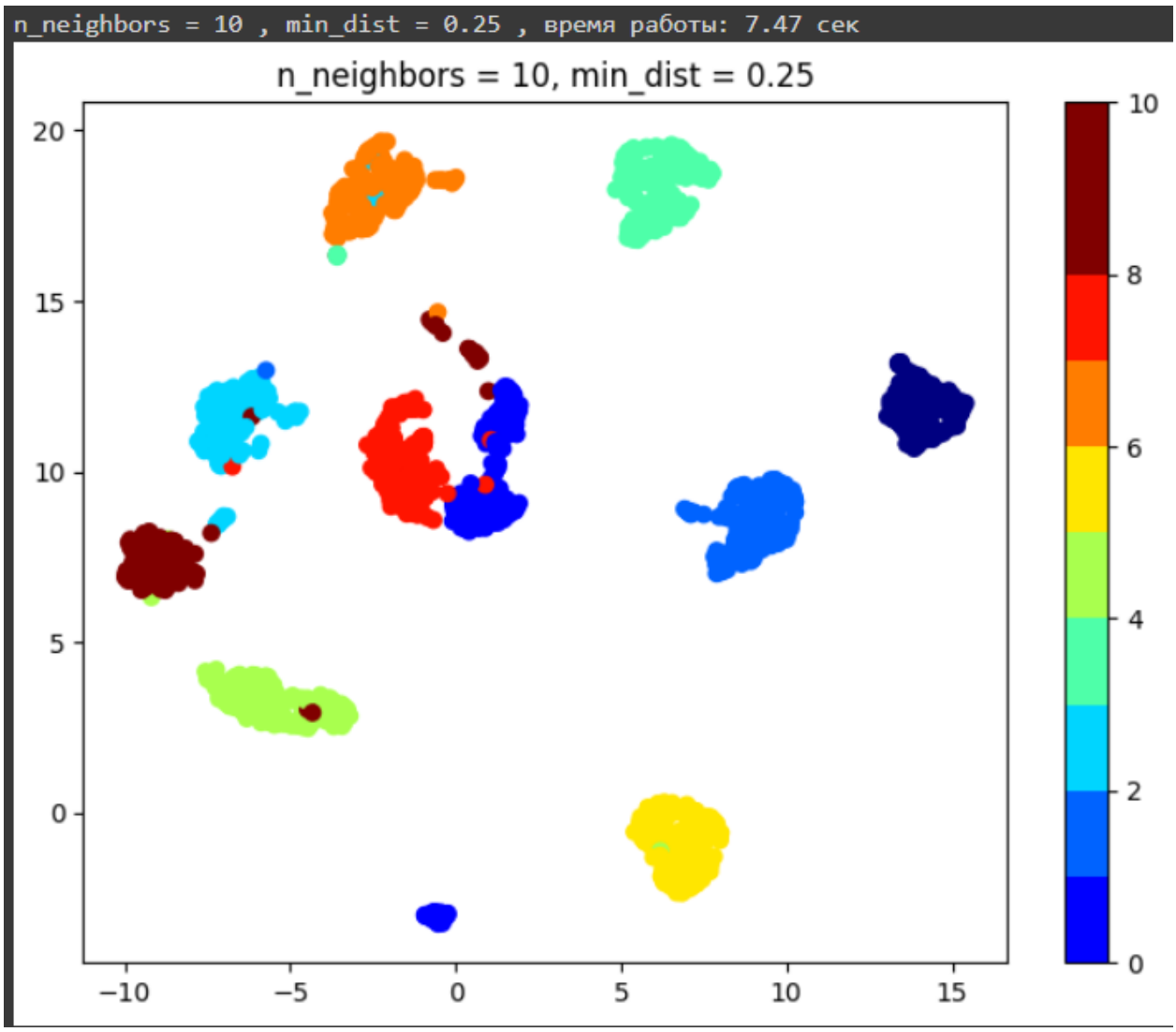


Рисунок 20 – Результат работы программы

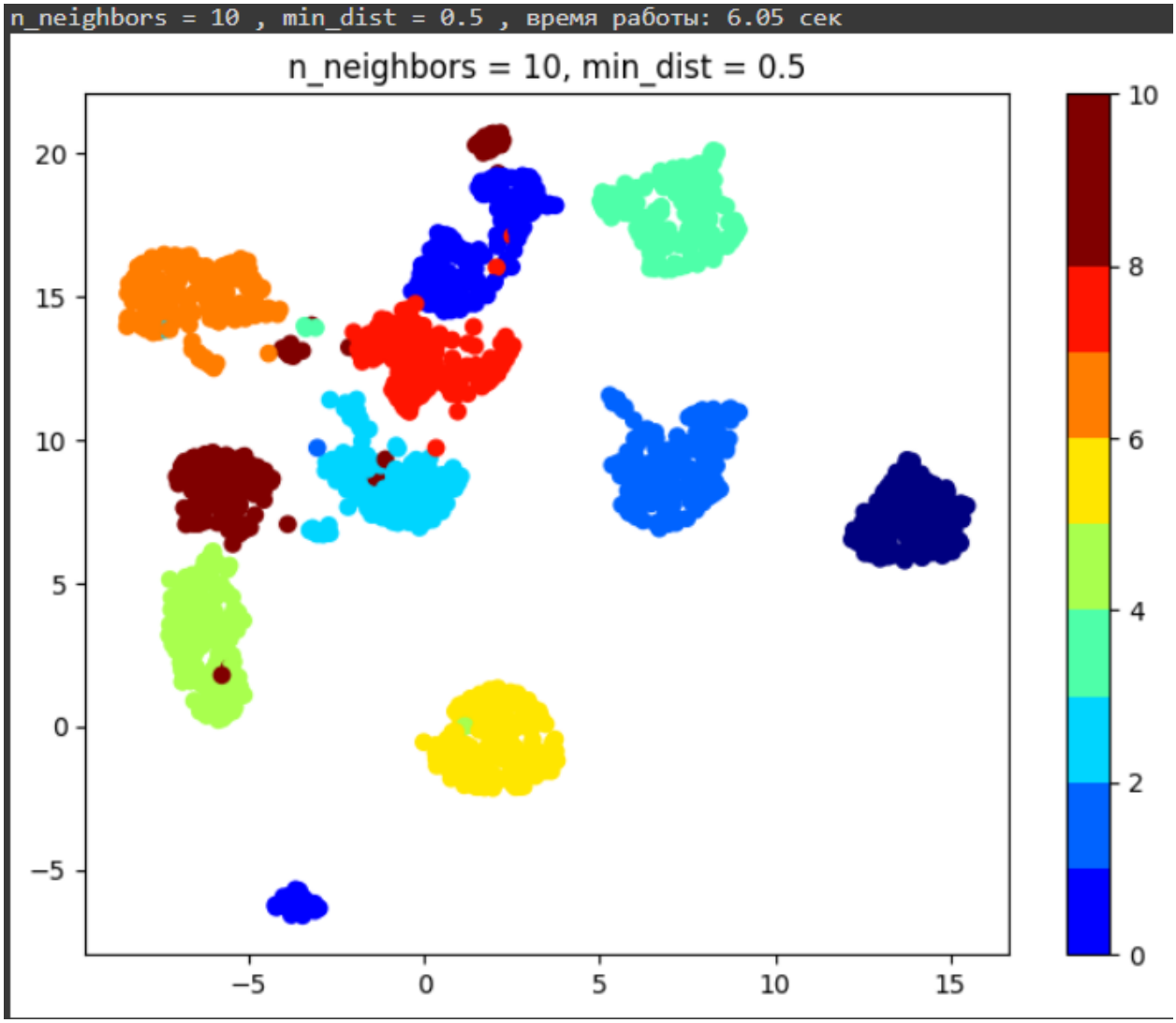


Рисунок 21 – Результат работы программы

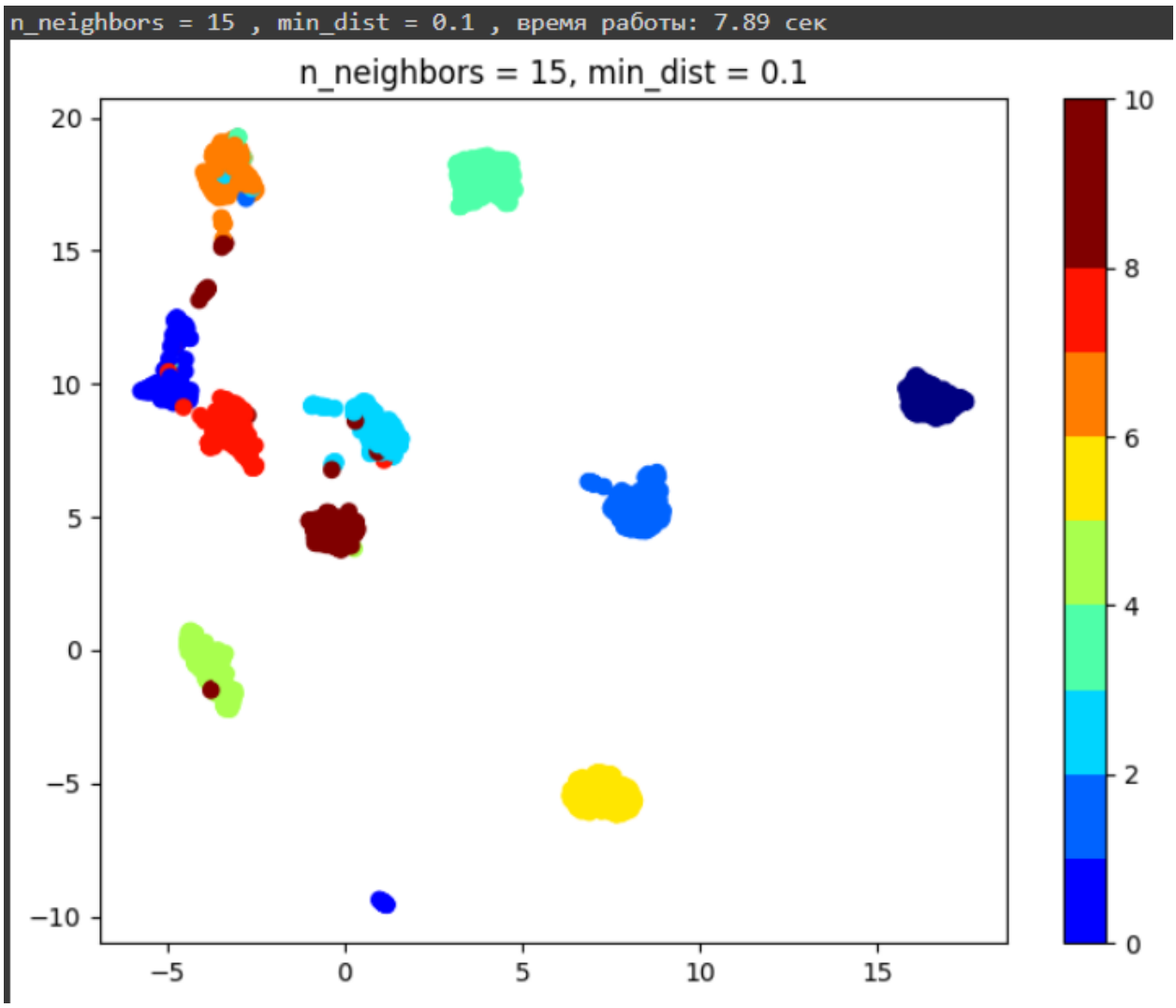


Рисунок 22 – Результат работы программы

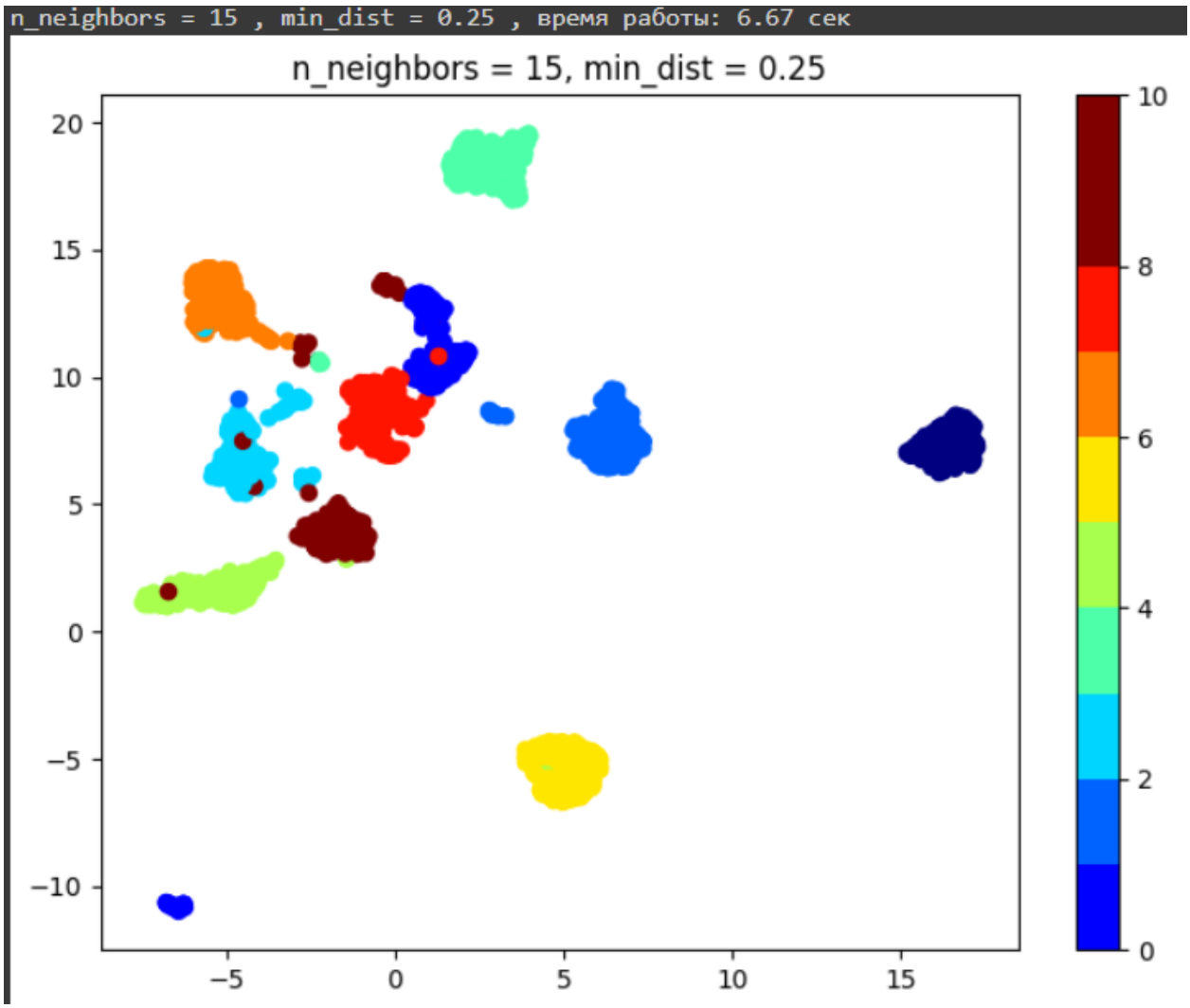


Рисунок 23 – Результат работы программы

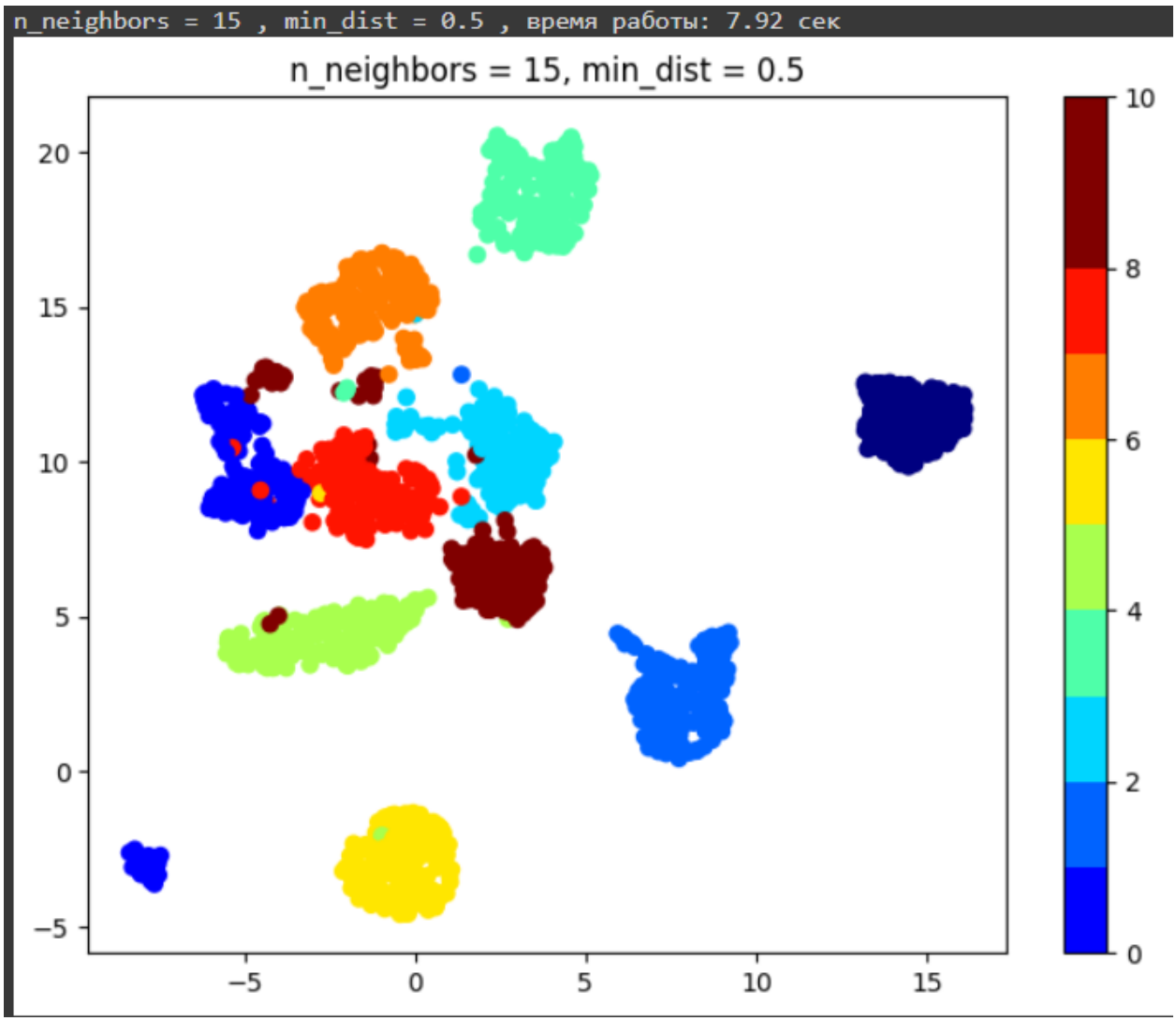


Рисунок 24 – Результат работы программы

Можно заметить, что время отрисовки каждого графика umap меньше, чем время отрисовки графика t-SNE.