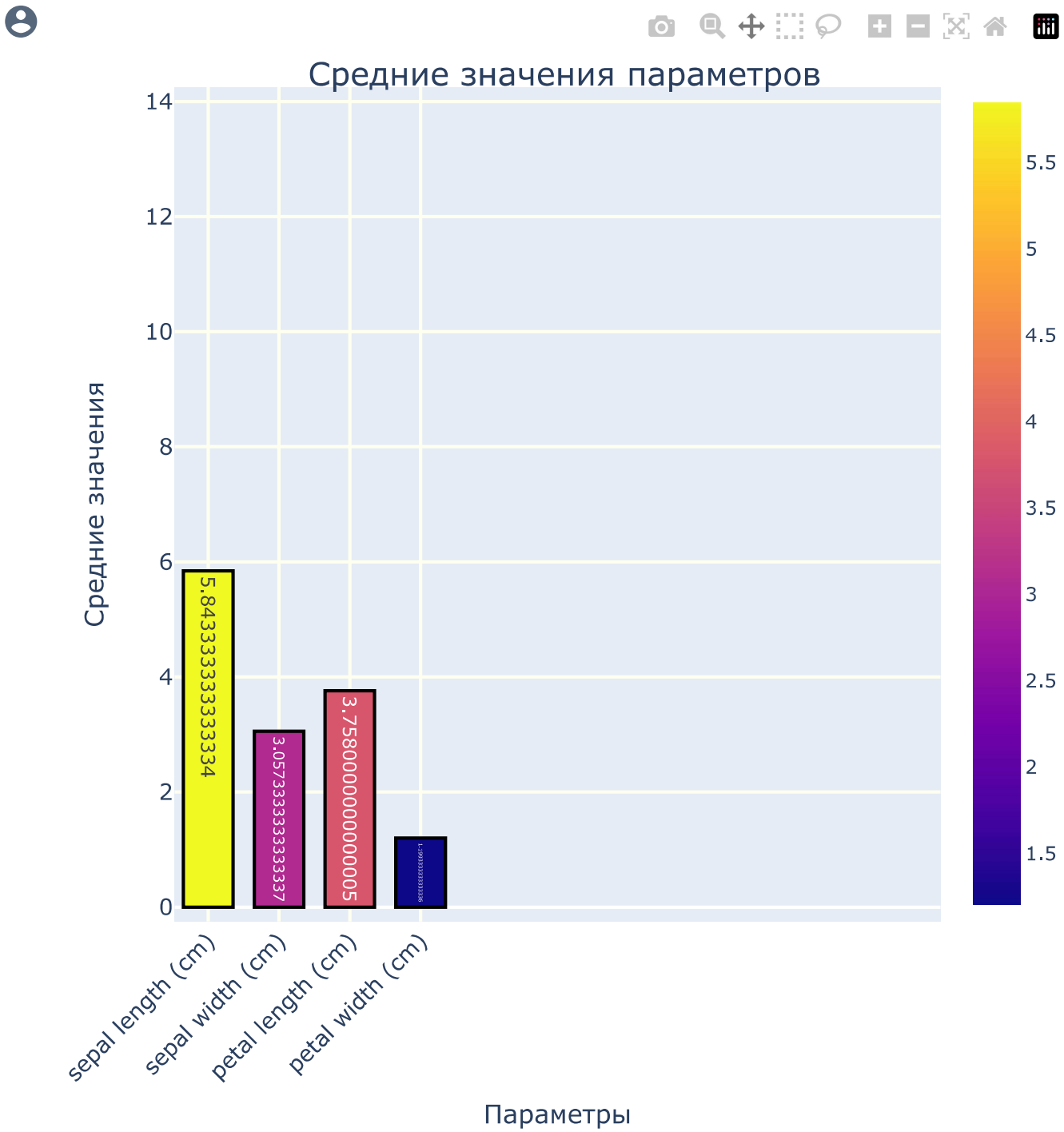
```
import pandas as pd

dataset_url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
dataset = pd.read_csv(dataset_url, header=None, names=["sepal_length", "sepal_width", "pet

# выведем первые пять строк
print("\nhead")
print(dataset.head())

# выведем последние пять строк
print("\ntail")
print(dataset.tail())

# выведем информацию о всех столбцах
print("\ninfo")
print(dataset.info())

# выведем описательную статистику для всех числовых столбцов
print("\ndescribe")
print(dataset.describe())
```

```
head
   sepal_length  sepal_width  petal_length  petal_width        class
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa

tail
     sepal_length  sepal_width  petal_length  petal_width           class
145           6.7          3.0           5.2          2.3  Iris-virginica
146           6.3          2.5           5.0          1.9  Iris-virginica
147           6.5          3.0           5.2          2.0  Iris-virginica
148           6.2          3.4           5.4          2.3  Iris-virginica
149           5.9          3.0           5.1          1.8  Iris-virginica

info
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   class         150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None

describe
       sepal_length  sepal_width  petal_length  petal_width
```

```
       count    150.000000   150.000000    150.000000   150.000000
       mean       5.843333     3.054000      3.758667     1.198667
       std        0.828066     0.433594      1.764420     0.763161
       min        4.300000     2.000000      1.000000     0.100000
       25%        5.100000     2.800000      1.600000     0.300000
       50%        5.800000     3.000000      4.350000     1.300000
       75%        6.400000     3.300000      5.100000     1.800000
       max        7.900000     4.400000      6.900000     2.500000
```

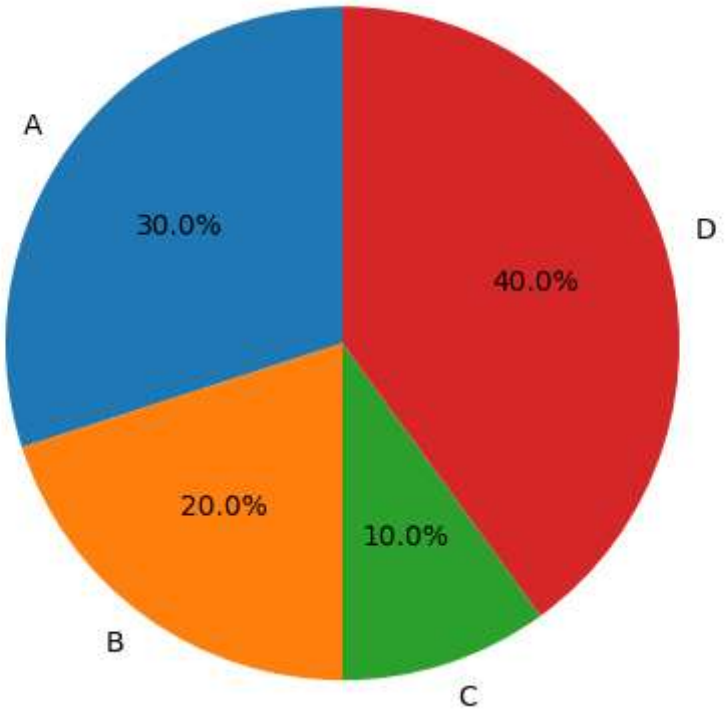
```
import plotly.graph_objs as go
import plotly.io as pio
pio.renderers.default = 'colab'
import pandas as pd
from sklearn.datasets import load_iris

# загрузка датасета
iris = load_iris()

# создание переменной dataset с именами колонок в качестве названий параметров
dataset = pd.DataFrame(iris.data, columns=iris.feature_names)

# задаем данные для столбчатой диаграммы
x_axis_labels = ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal widt
y_axis_values = dataset.describe().loc['mean', :'petal width (cm)'].tolist()

# устанавливаем цвета столбцов в зависимости от значения показателя
colors = ['red' if val < 3 else 'green' for val in y_axis_values]

# создаем объект для столбчатой диаграммы
fig = go.Figure(data=[go.Bar(
    x=x_axis_labels,  # по оси X указываем названия параметров
    y=y_axis_values,  # по оси Y указываем значения параметров
    marker=dict(color=y_axis_values, coloraxis="coloraxis"),  # устанавливаем цвета столбц
    text=y_axis_values,
    textposition='auto',
    # урамляем дополнительные параметры для улучшения внешнего вида графика
    width=0.7,
    marker_line_width=2,
    marker_line_color='black',
)])

# установим заголовок графика
fig.update_layout(title=dict(text="Средние значения параметров", y=0.95, x=0.5, xanchor='c

# установим подписи для осей X и Y
fig.update_layout(xaxis=dict(title=dict(text="Параметры", font=dict(size=16)), tickangle=-
                  yaxis=dict(title=dict(text="Средние значения", font=dict(size=16)), tick

# установим высоту графика
fig.update_layout(height=700)

# установим сетку
fig.update_xaxes(showgrid=True, gridwidth=2, gridcolor='ivory')
fig.update_yaxes(showgrid=True, gridwidth=2, gridcolor='ivory')
```

```
# уберем лишние отступы по краям
fig.update_layout(margin=dict(l=0, r=0, t=50, b=0))

fig.show()
```



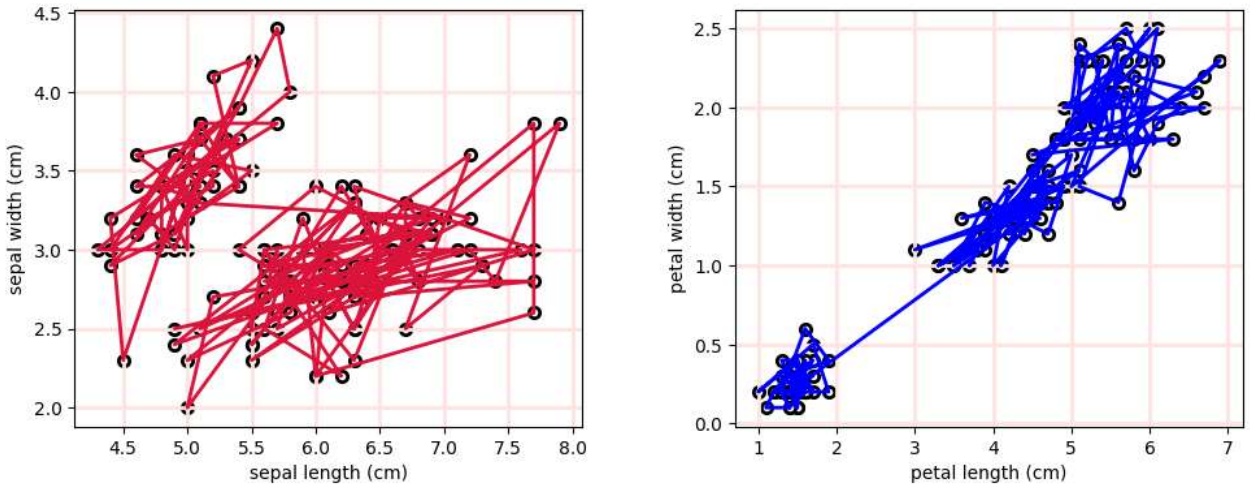

```
import pandas as pd
import matplotlib.pyplot as plt

# Define your data for the chart
x_axis_labels = ['A', 'B', 'C', 'D']
y_axis_values = [30, 20, 10, 40]

# Create a DataFrame with data for a pie chart
pie_df = pd.DataFrame({'Параметры': x_axis_labels, 'Значения': y_axis_values})
```

```
# Group values in column 'Значения' based on condition '>=3' and 'меньше 3', create column
pie_df['Группы'] = pie_df['Значения'].apply(lambda x: '>=3' if x >= 3 else 'меньше 3')

# Create a pie chart
fig, ax = plt.subplots()
ax.pie(pie_df['Значения'], labels=pie_df['Параметры'], autopct='%1.1f%%', startangle=90)
ax.axis('equal')  # Equal aspect ratio ensures the pie chart is circular

# Display the chart
plt.show()
```




```
import pandas as pd
import seaborn as sns
from sklearn import datasets
import matplotlib.pyplot as plt
%matplotlib inline

# Загружаем набор данных ирисов
iris = datasets.load_iris()

# Конвертируем его в DataFrame
dataset = pd.DataFrame(data=iris.data, columns=iris.feature_names)
dataset["species"] = iris.target_names[iris.target]

# разделим датафрейм на две группы для построения линейных графиков
group1 = dataset[['sepal length (cm)', 'sepal width (cm)']]
group2 = dataset[['petal length (cm)', 'petal width (cm)']]

# создаем список названий параметров
names = ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

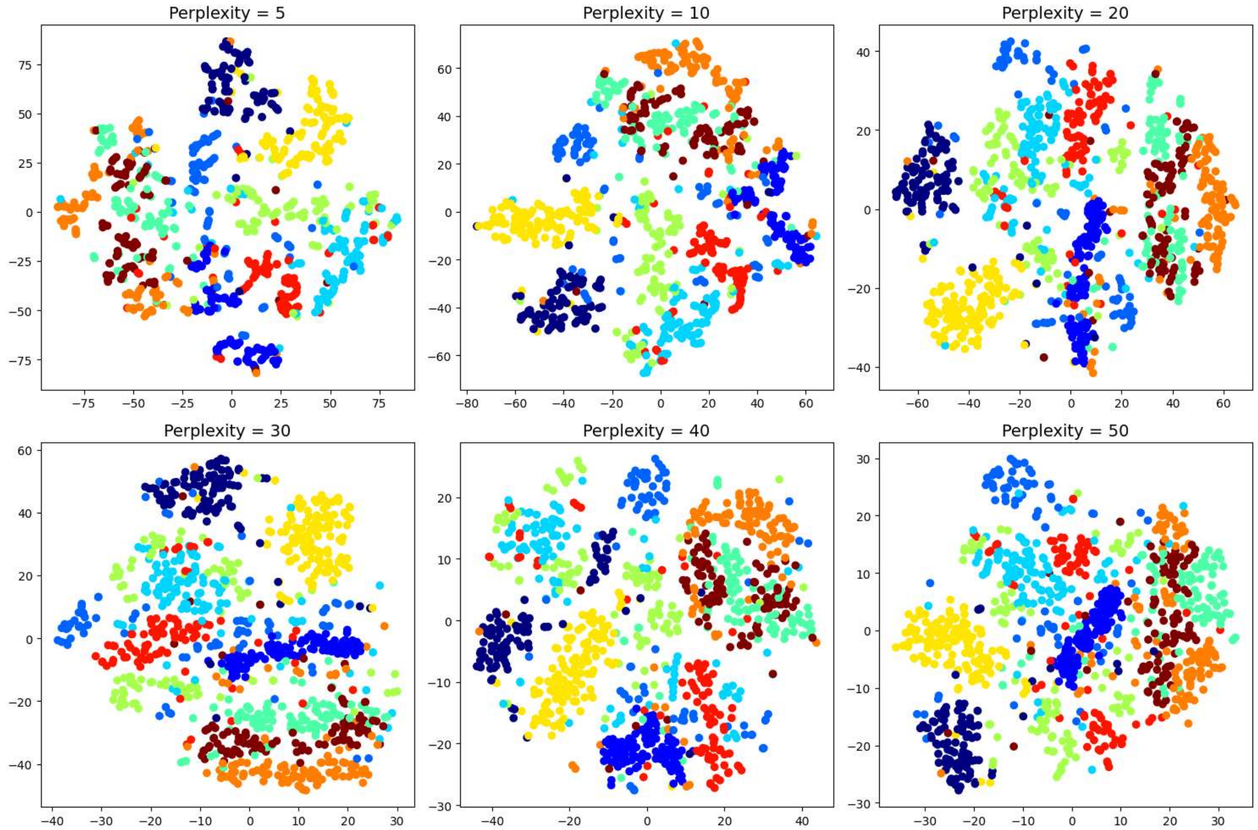
```
# создаем словарь цветов для линий и точек
colors = {'crimson': names[:2], 'blue': names[2:]}

# строим графики
fig, axs = plt.subplots(ncols=2, figsize=(10,4))

# рисуем график для первой группы
axs[0].plot(group1[names[0]], group1[names[1]], color='crimson', linewidth=2)
axs[0].scatter(group1[names[0]], group1[names[1]], color='white', edgecolors='black', line
axs[0].set_xlabel(names[0])
axs[0].set_ylabel(names[1])

# рисуем график для второй группы
axs[1].plot(group2[names[2]], group2[names[3]], color='blue', linewidth=2)
axs[1].scatter(group2[names[2]], group2[names[3]], color='white', edgecolors='black', line
axs[1].set_xlabel(names[2])
axs[1].set_ylabel(names[3])

# устанавливаем сетку
axs[0].grid(linewidth=2, color='mistyrose')
axs[1].grid(linewidth=2, color='mistyrose')

# уберем лишние отступы по краям
plt.subplots_adjust(left=0.05, bottom=0.1, right=0.95, top=0.9, wspace=0.3, hspace=0.2)
plt.show()
```




```
import numpy as np
import matplotlib.pyplot as plt

# загружаем набор данных MNIST
from sklearn.datasets import fetch_openml
```

```
mnist = fetch_openml('mnist_784')

# подготовим данные
X = mnist.data / 255.0
y = mnist.target.astype('int')

# выберем случайную подвыборку из 1000 изображений
indices = np.random.choice(X.shape[0], 1000, replace=False)
X_sample = np.take(X, indices, axis=0)
y_sample = np.take(y, indices)

from sklearn.manifold import TSNE

# задаем значения перплексии
perplexities = [5, 10, 20, 30, 40, 50]

# задаем размер графиков
fig, axs = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))

# проходим в цикле по значениям перплексии и строим графики
for i, perplexity in enumerate(perplexities):
    row = i // 3
    col = i % 3

    # применяем алгоритм t-SNE и строим график
    tsne = TSNE(n_components=2, learning_rate=200.0, perplexity=perplexity, early_exaggera
    # где n_components - размерность пространства(в данном случае используется 2х мерное)
    # perplexity - равномерность распределения объектов early_exaggeration - начальное рас
    # (чтобы генерировать одинаковую последовательность случайных чисел)
    X_tsne = tsne.fit_transform(X_sample)
    axs[row][col].scatter(X_tsne[:, 0], X_tsne[:, 1], c=y_sample, cmap=plt.cm.get_cmap('je
    axs[row][col].set_title("Perplexity = {}".format(perplexity), fontsize=14)

plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/datasets/_openml.py:968: FutureWar
  warn(
<ipython-input-2-31c9e045aafc>:33: MatplotlibDeprecationWarning: The get_cmap func
  axs[row][col].scatter(X_tsne[:, 0], X_tsne[:, 1], c=y_sample, cmap=plt.cm.get_cm
<ipython-input-2-31c9e045aafc>:33: MatplotlibDeprecationWarning: The get_cmap func
  axs[row][col].scatter(X_tsne[:, 0], X_tsne[:, 1], c=y_sample, cmap=plt.cm.get_cm
<ipython-input-2-31c9e045aafc>:33: MatplotlibDeprecationWarning: The get_cmap func
  axs[row][col].scatter(X_tsne[:, 0], X_tsne[:, 1], c=y_sample, cmap=plt.cm.get_cm
<ipython-input-2-31c9e045aafc>:33: MatplotlibDeprecationWarning: The get_cmap func
  axs[row][col].scatter(X_tsne[:, 0], X_tsne[:, 1], c=y_sample, cmap=plt.cm.get_cm
<ipython-input-2-31c9e045aafc>:33: MatplotlibDeprecationWarning: The get_cmap func
  axs[row][col].scatter(X_tsne[:, 0], X_tsne[:, 1], c=y_sample, cmap=plt.cm.get_cm
<ipython-input-2-31c9e045aafc>:33: MatplotlibDeprecationWarning: The get_cmap func
  axs[row][col].scatter(X_tsne[:, 0], X_tsne[:, 1], c=y_sample, cmap=plt.cm.get_cm
```



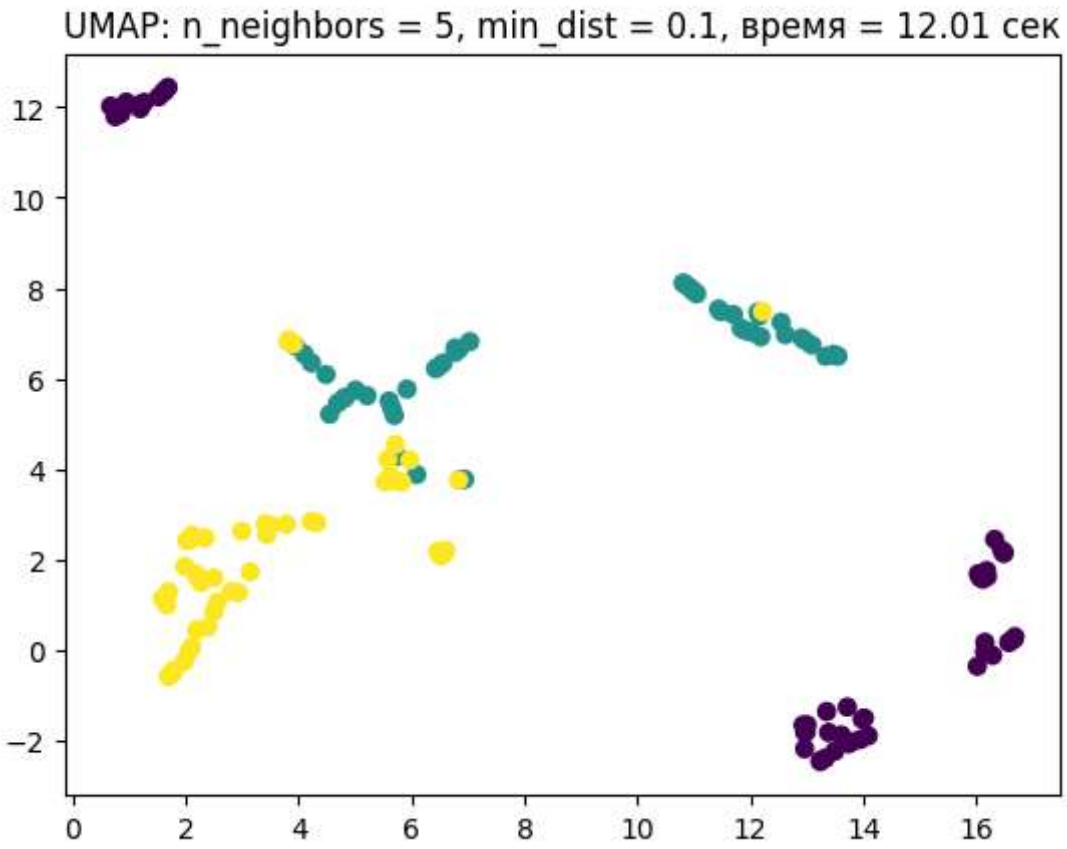

```
pip install umap-learn
```

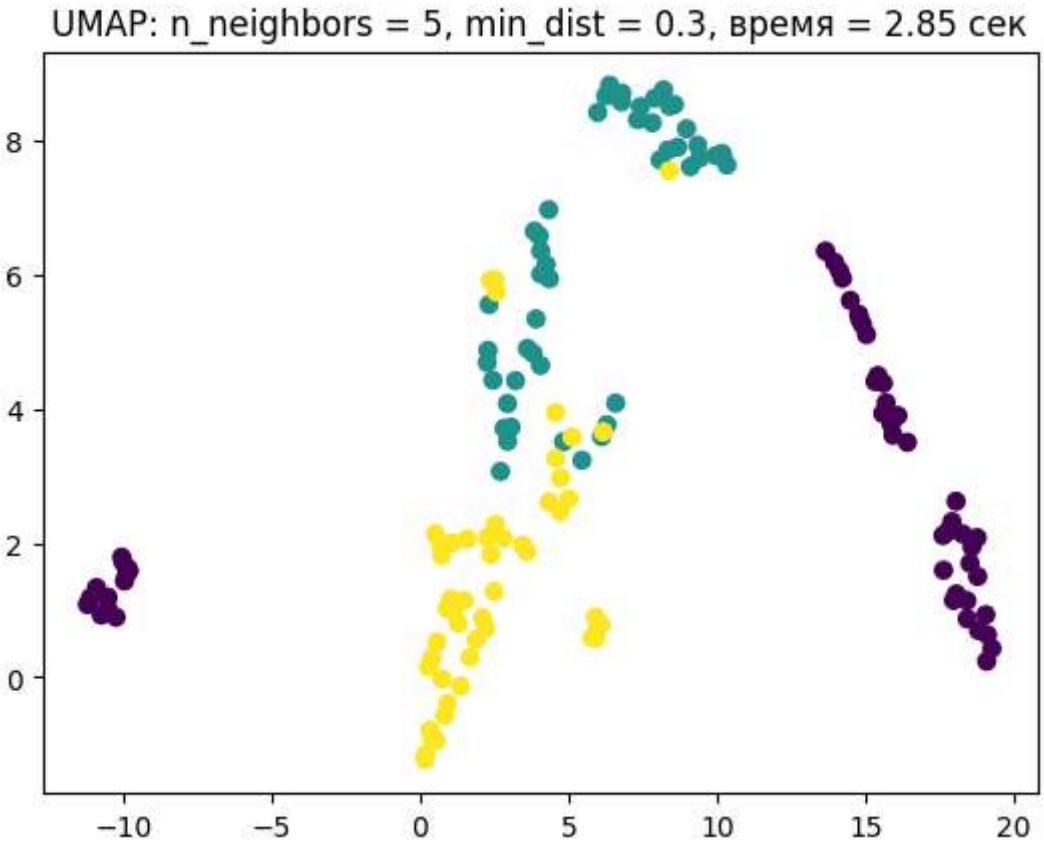
```
Collecting umap-learn
  Downloading umap-learn-0.5.4.tar.gz (90 kB)
```

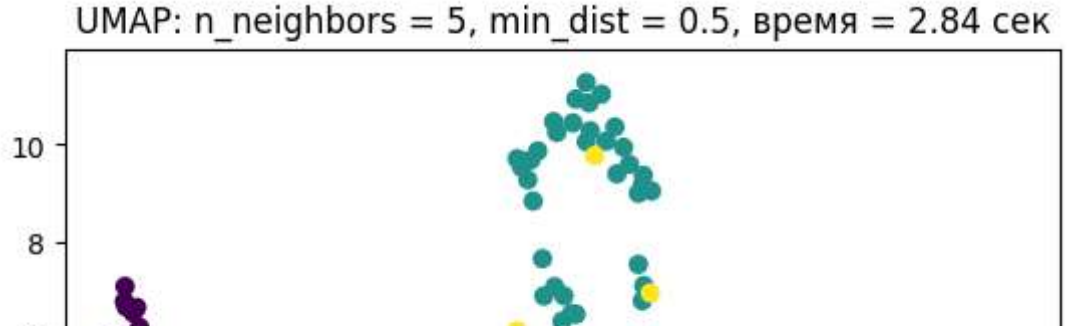
```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 90.8/90.8 kB 1.8 MB/s eta 0:00:00
       Preparing metadata (setup.py) ... done
     Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages
     Requirement already satisfied: scipy>=1.3.1 in /usr/local/lib/python3.10/dist-package
     Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-p
     Requirement already satisfied: numba>=0.51.2 in /usr/local/lib/python3.10/dist-packag
     Collecting pynndescent>=0.5 (from umap-learn)
       Downloading pynndescent-0.5.10.tar.gz (1.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 22.0 MB/s eta 0:00:00
       Preparing metadata (setup.py) ... done
     Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from
     Requirement already satisfied: tbb>=2019.0 in /usr/local/lib/python3.10/dist-packages
     Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/python3.1
     Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages
     Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-package
     Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist
     Building wheels for collected packages: umap-learn, pynndescent
       Building wheel for umap-learn (setup.py) ... done
       Created wheel for umap-learn: filename=umap_learn-0.5.4-py3-none-any.whl size=86776
       Stored in directory: /root/.cache/pip/wheels/fb/66/29/199acf5784d0f7b8add6d466175ab
       Building wheel for pynndescent (setup.py) ... done
       Created wheel for pynndescent: filename=pynndescent-0.5.10-py3-none-any.whl size=55
       Stored in directory: /root/.cache/pip/wheels/4a/38/5d/f60a40a66a9512b7e5e83517ebc2c
     Successfully built umap-learn pynndescent
     Installing collected packages: pynndescent, umap-learn
     Successfully installed pynndescent-0.5.10 umap-learn-0.5.4
```

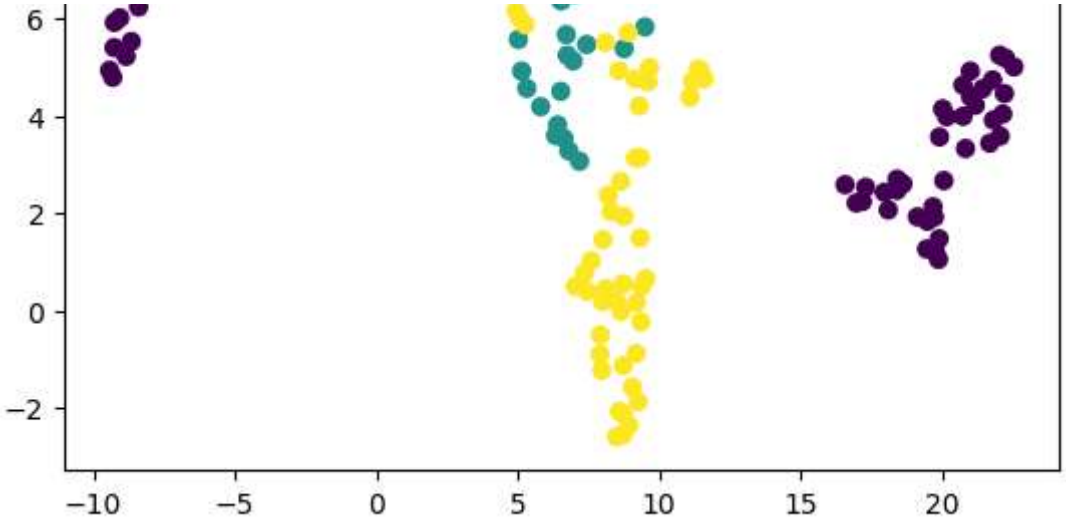
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler
import umap
import time

# Загрузка и обработка данных
dataset_url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
data = pd.read_csv(dataset_url, header=None, names=["sepal_length", "sepal_width", "petal_
data_scaled = StandardScaler().fit_transform(data.iloc[:, :-1])  # стандартизация данных

# Обучение и визуализация UMAP
neighbors = [5, 15, 30]
min_dists = [0.1, 0.3, 0.5]

for n_neighbors in neighbors:
    for min_dist in min_dists:
        start = time.time()
        reducer = umap.UMAP(n_neighbors=n_neighbors, min_dist=min_dist)
        embedding = reducer.fit_transform(data_scaled)
        end = time.time()

        plt.scatter(embedding[:, 0], embedding[:, 1], c=pd.factorize(data['class'])[0], cm
        plt.title(f'UMAP: n_neighbors = {n_neighbors}, min_dist = {min_dist}, время = {end
        plt.show()

# Обучение и визуализация t-SNE
```

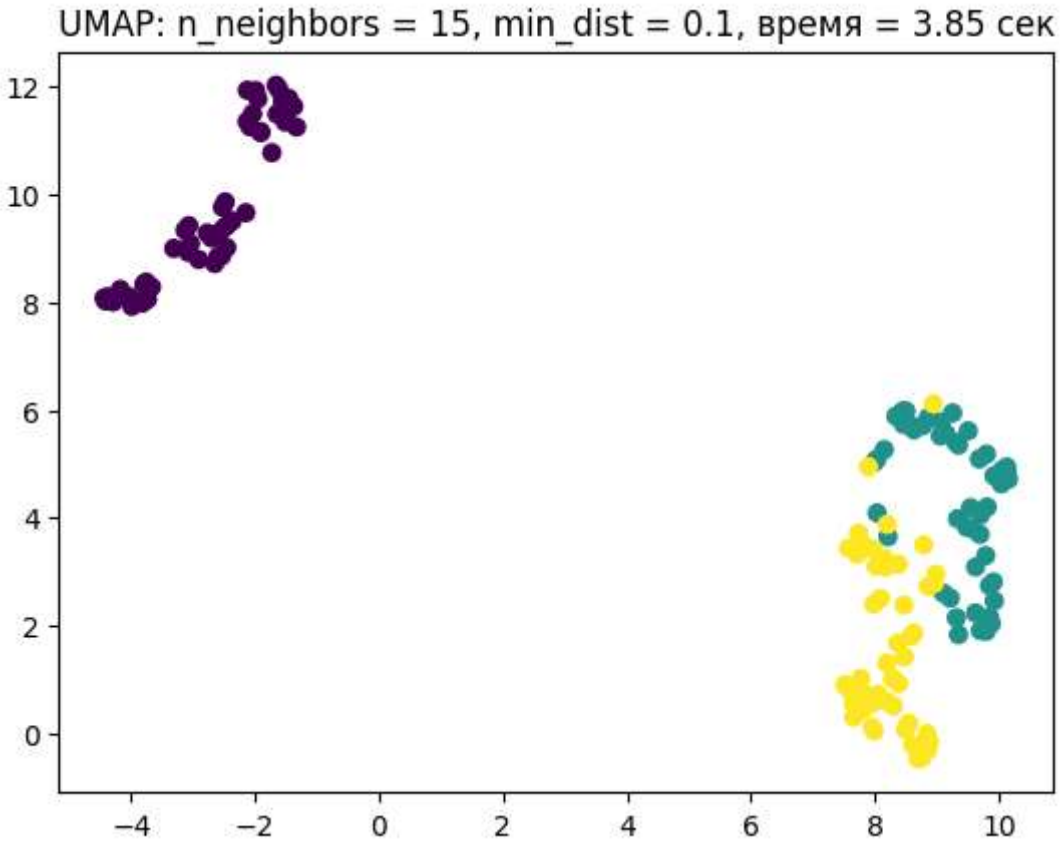
```
start = time.time()
tsne = TSNE()
tsne_embedding = tsne.fit_transform(data_scaled)
end = time.time()

plt.scatter(tsne_embedding[:, 0], tsne_embedding[:, 1], c=pd.factorize(data['class'])[0],
plt.title(f't-SNE: время = {end - start:.2f} сек')
plt.show()
```

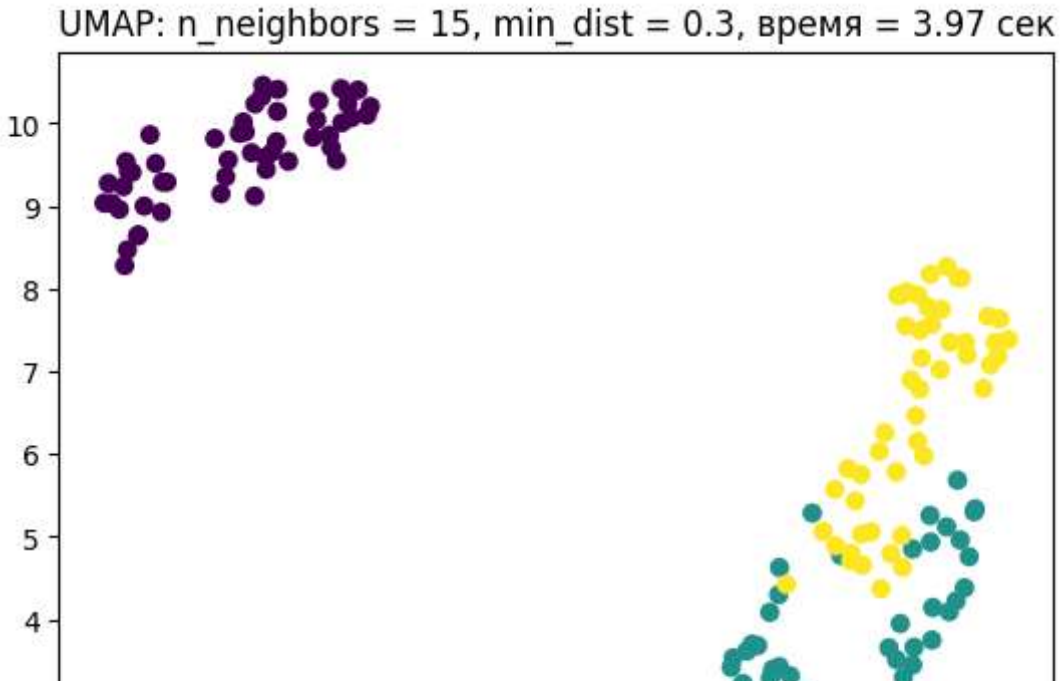

UMAP: n_neighbors = 5, min_dist = 0.1, время = 12.01 сек
12
10
8
6
4
2
0
−2
0
2
4
6
8
10
12
14
16

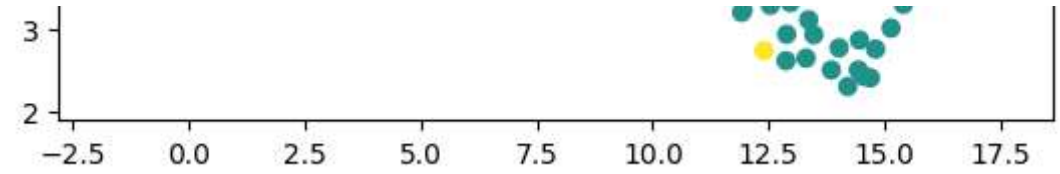

UMAP: n_neighbors = 5, min_dist = 0.3, время = 2.85 сек
8
6
4
2
0
−10
−5
0
5
10
15
20

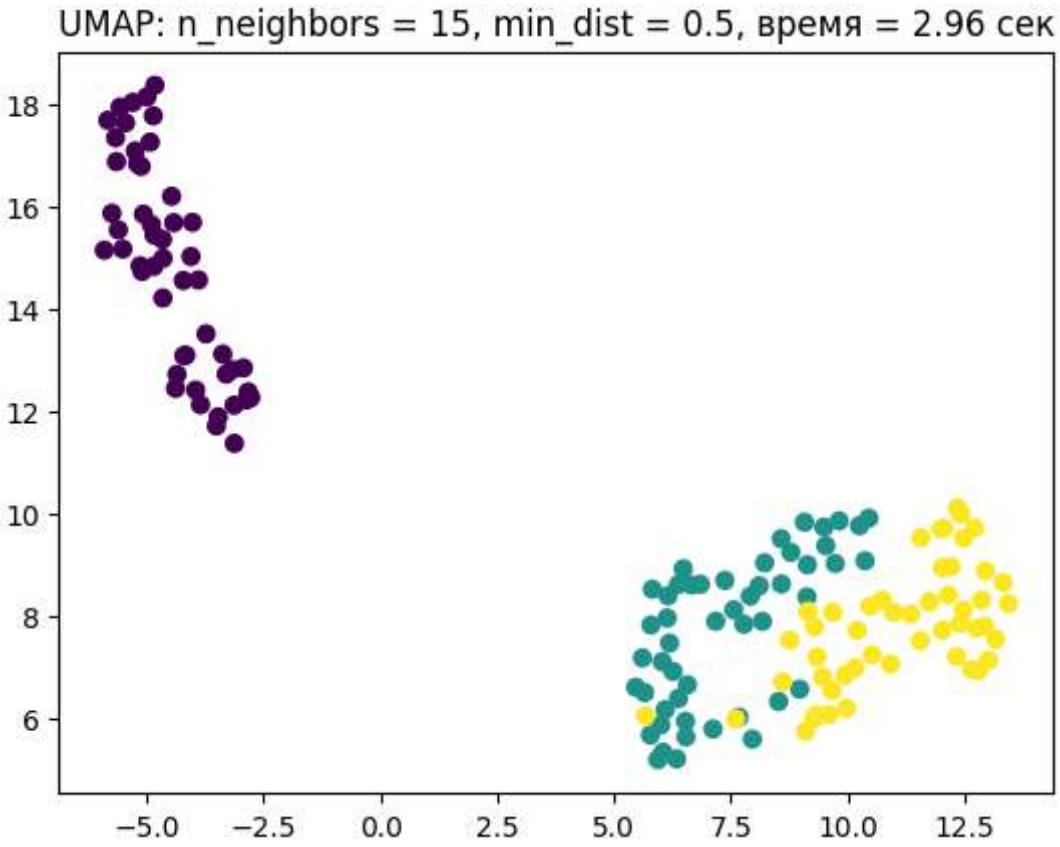

UMAP: n_neighbors = 5, min_dist = 0.5, время = 2.84 сек
10
8

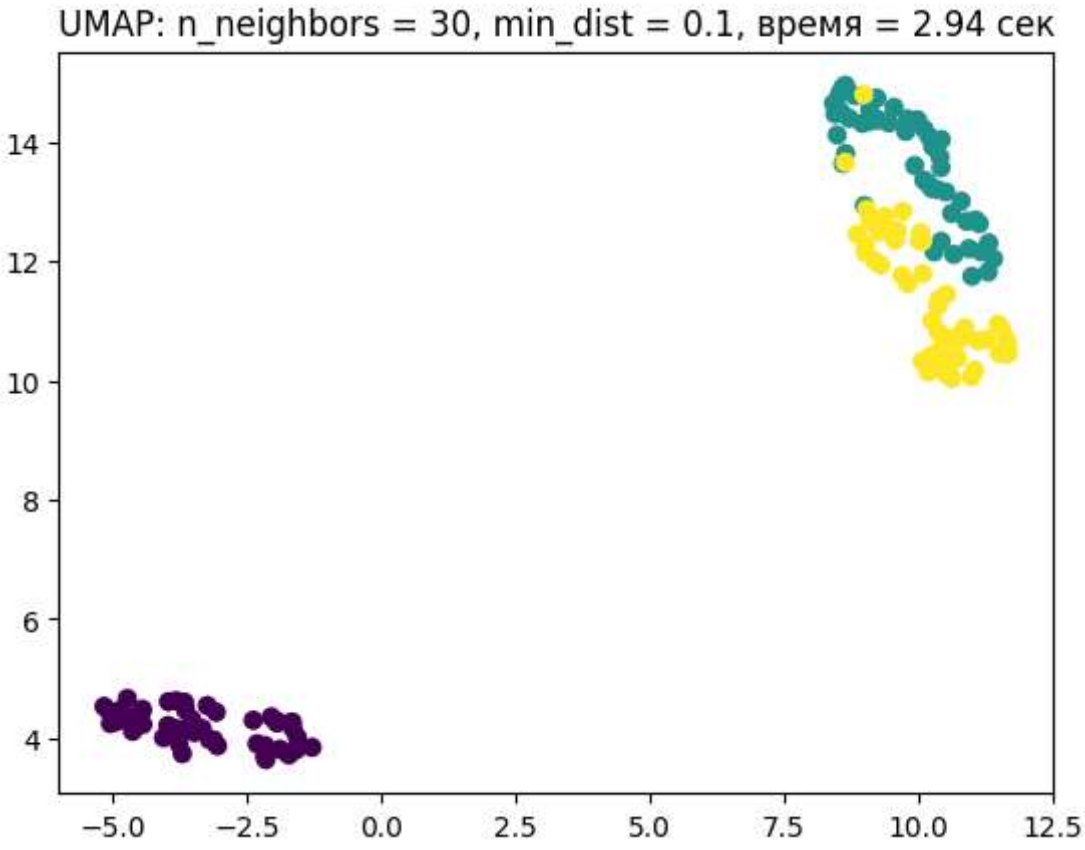

6
4
2
0
−2
−10
−5
0
5
10
15
20

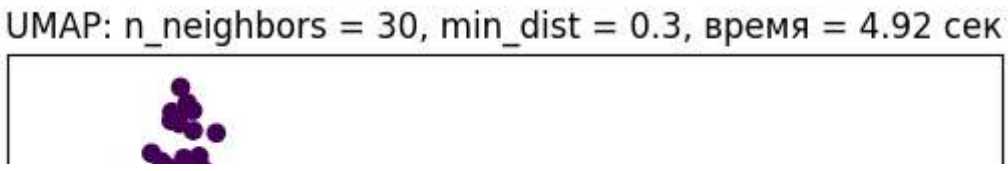

UMAP: n_neighbors = 15, min_dist = 0.1, время = 3.85 сек
12
10
8
6
4
2
0
−4
−2
0
2
4
6
8
10

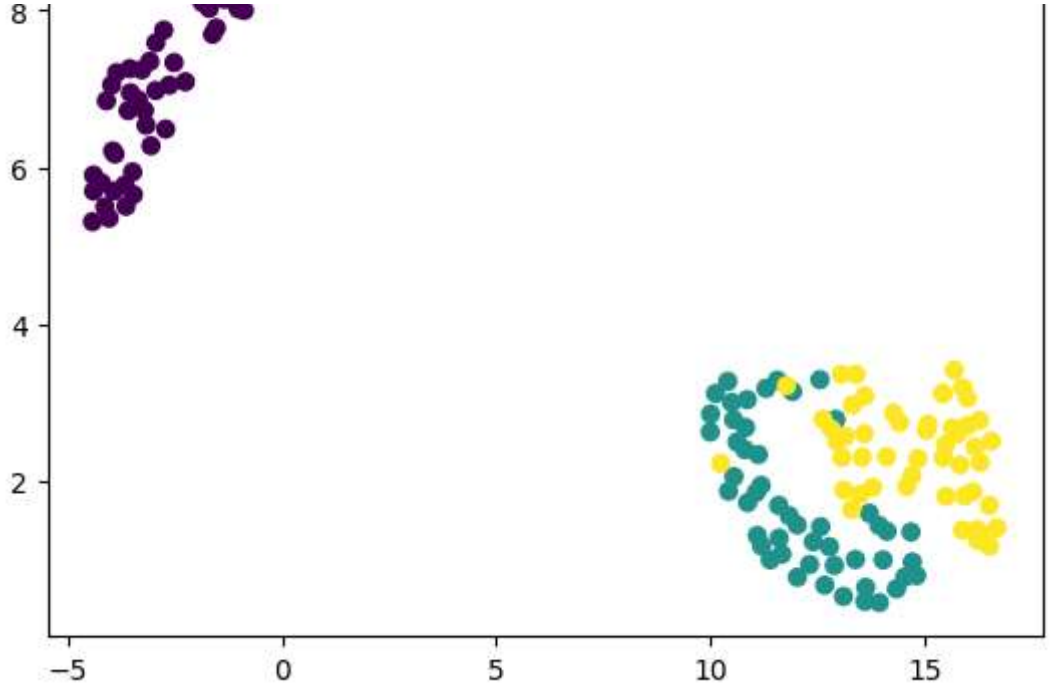

UMAP: n_neighbors = 15, min_dist = 0.3, время = 3.97 сек
10
9
8
7
6
5
4

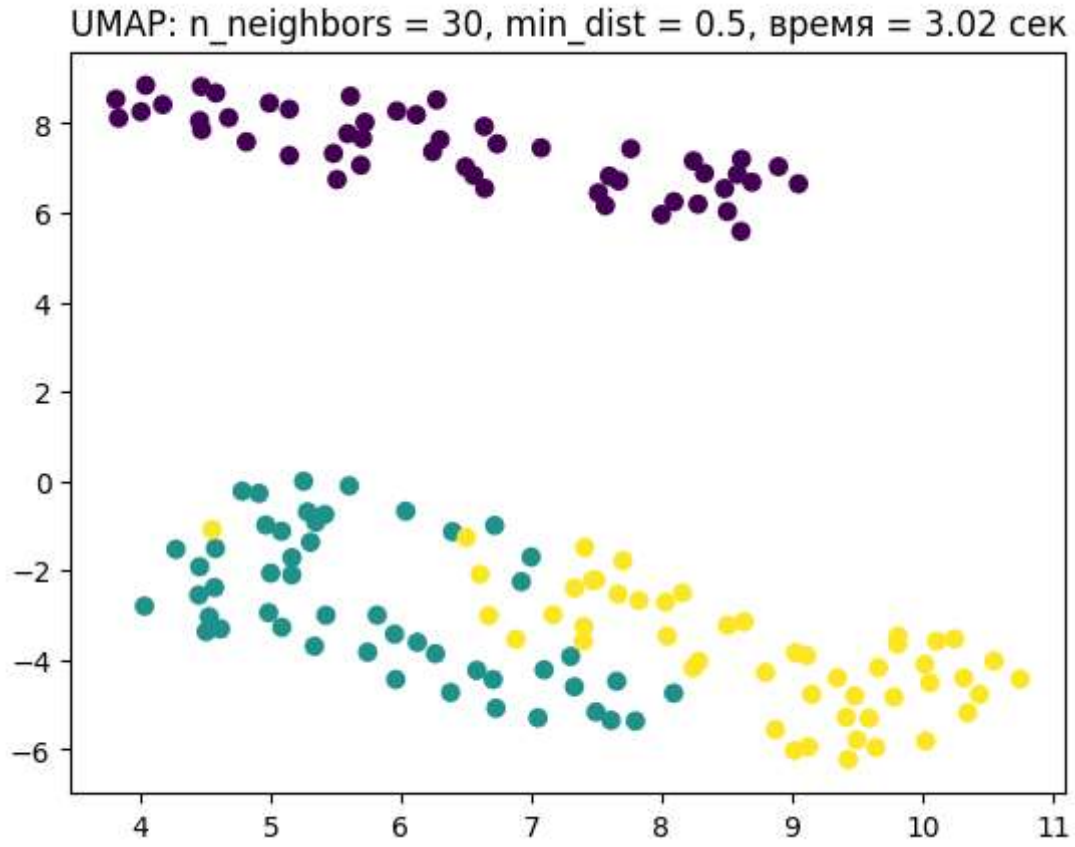

3
2
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


UMAP: n_neighbors = 15, min_dist = 0.5, время = 2.96 сек
18
16
14
12
10
8
6
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5


UMAP: n_neighbors = 30, min_dist = 0.1, время = 2.94 сек
14
12
10
8
6
4
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5


UMAP: n_neighbors = 30, min_dist = 0.3, время = 4.92 сек


8
6
4
2
−5
0
5
10
15


UMAP: n_neighbors = 30, min_dist = 0.5, время = 3.02 сек
8
6
4
2
0
−2
−4
−6
4
5
6
7
8
9
10
11