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

|  |  |
| --- | --- |
| **Отчет по второй практической работе.** | |
| **по дисциплине** | |
| **«Технологии и инструментарий анализа больших данных»** | |
|  | |
| Выполнил студент группы ИКБО- 22-20 | Никулин К.В. |
| Принял | Парамонов А.А. |

Москва 2023 г.

Оглавление

[1 ИНФОРМАЦИЯ О ДАТАСЕТЕ YOUTUBE STATISTIC 6](#_Toc146809030)

[2 СТОЛБЧАТАЯ ДИАГРАММА 8](#_Toc146809031)

[3 КРУГОВАЯ ДИАГРАММА 9](#_Toc146809032)

[4 ЛИНЕЙНЫЕ ГРАФИКИ 10](#_Toc146809033)

[5 ВИЗУАЛИЗАЦИЯ МНОГОМЕРНЫХ ДАННЫХ T-SNE 11](#_Toc146809034)

[6 ВИЗУАЛИЗАЦИЯ МНОГОМЕРНЫХ ДАННЫХ UMAP 13](#_Toc146809035)

[7 ПРИЛОЖЕНИЕ 16](#_Toc146809036)

# ИНФОРМАЦИЯ О ДАТАСЕТЕ YOUTUBE STATISTIC

Датасет Youtube statistic содержит данные о крупных каналах на сайте youtube, в нем содержится 27 параметров, по 289 записей в среднем.

Описание используемых параметров:

* subscribers – информация о количестве подписчиков;
* rank – информация о том, на каком месте в топе находится канал ютубера;
* video views – информация о общем количестве просмотров канала;
* highest yearly earnings – наибольшая сумма заработка в год.

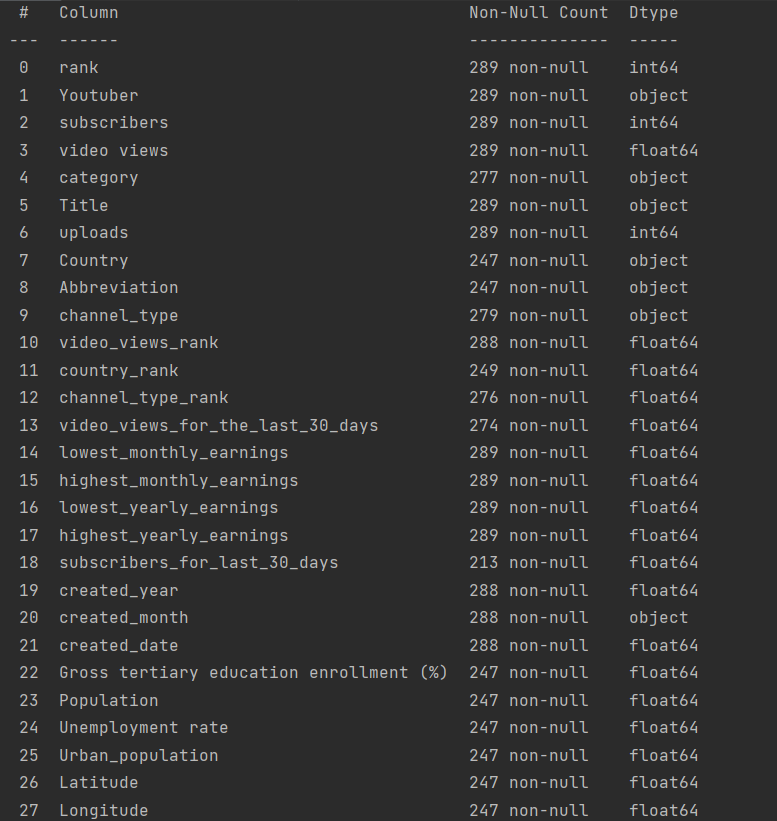


Рисунок 1 – Информация о датасете



Рисунок 2 – Заголовки датасета

# СТОЛБЧАТАЯ ДИАГРАММА

Для визуализации параметра «subscribers» сделаем столбчатую диаграмму.

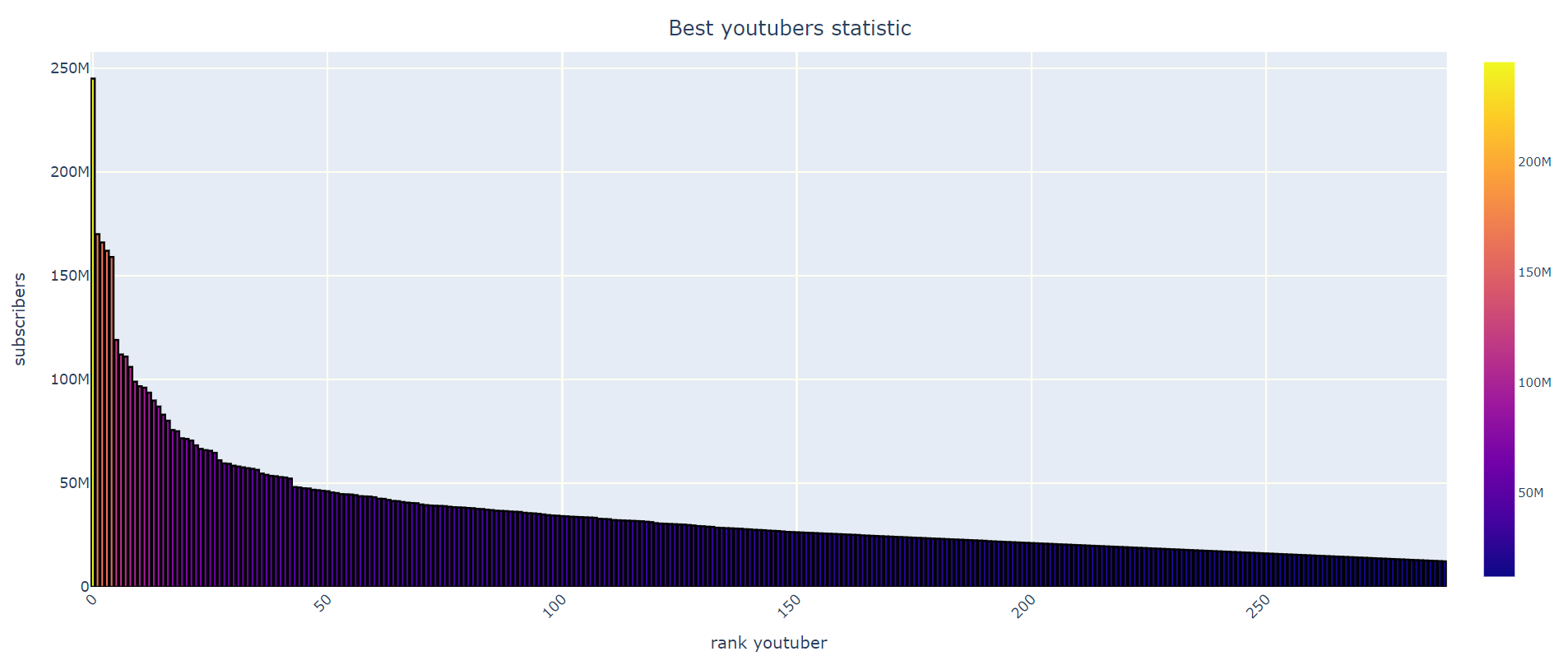


Рисунок 3 – Столбчатая диаграмма для параметра subscribers

# КРУГОВАЯ ДИАГРАММА

Для визуализации параметра «subscribers» сделаем круговую диаграмму, так-как в параметре много записей круговая диаграмма будет не читабельна, для того чтобы решить эту проблему объединим данные в следующие разделы:

* subscribers < 35 000 000
* subscribers < 80 000 000
* subscribers > 80 000 000

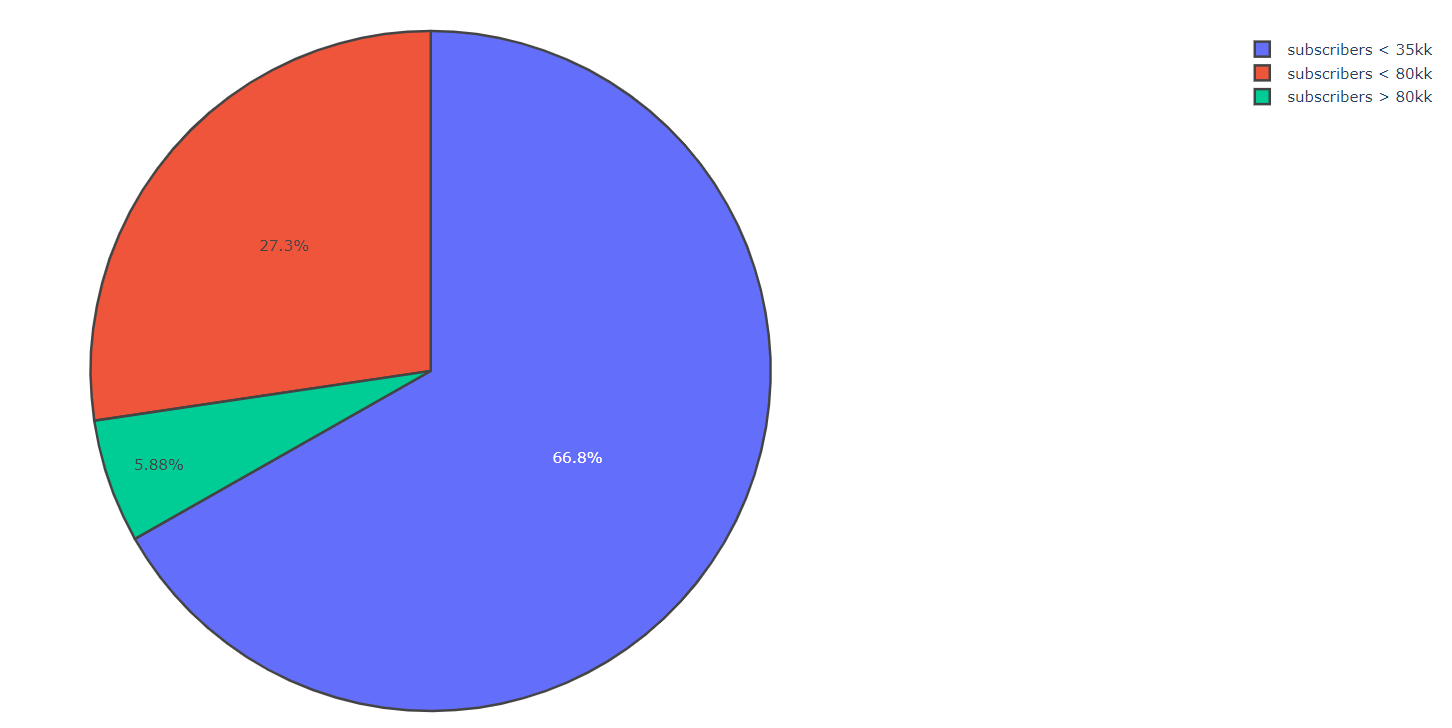


Рисунок 4 – Круговая диаграмма для параметра subscribers

# ЛИНЕЙНЫЕ ГРАФИКИ

Построим линейные графики для сравнения параметров, будем смотреть зависимость параметра subscribers, highest yearly earnings и video views от параметра rank.

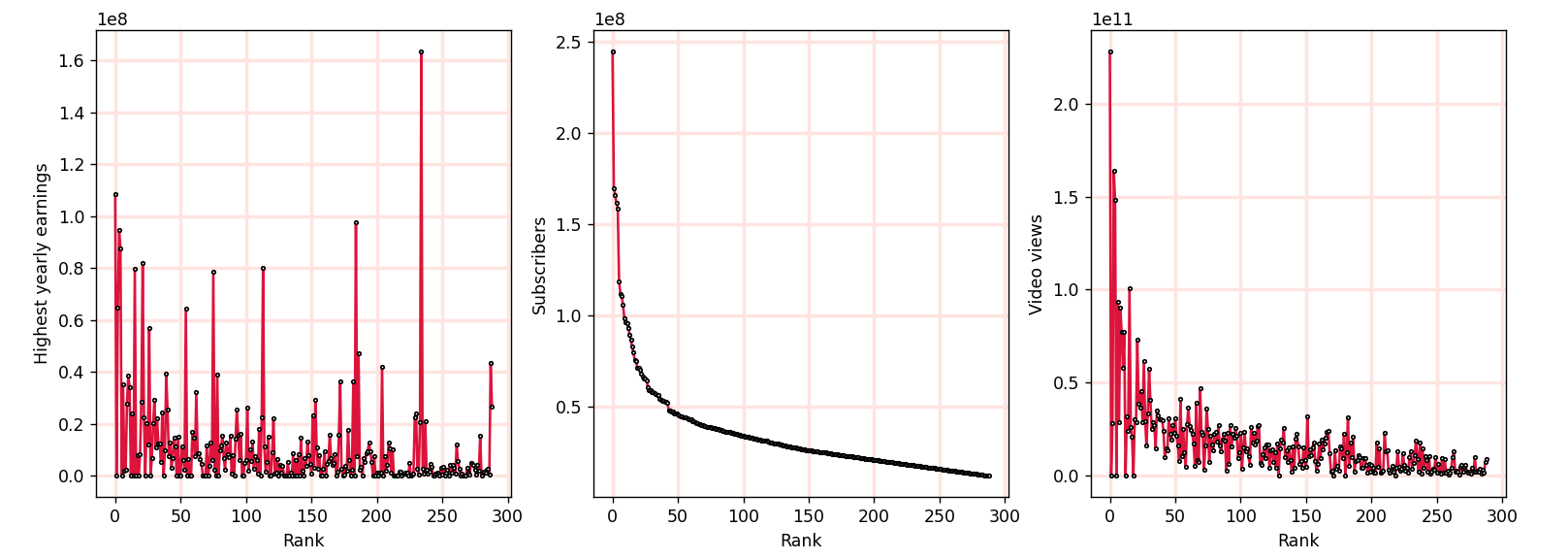


Рисунок 5 – Линейный графики для 3-ех параметров от параметра rank

# ВИЗУАЛИЗАЦИЯ МНОГОМЕРНЫХ ДАННЫХ T-SNE

Реализация визуализации многомерных данных с помощью t-SNE для датасета MNIST. Датасет MNIST содержит данные о рукописных цифрах.

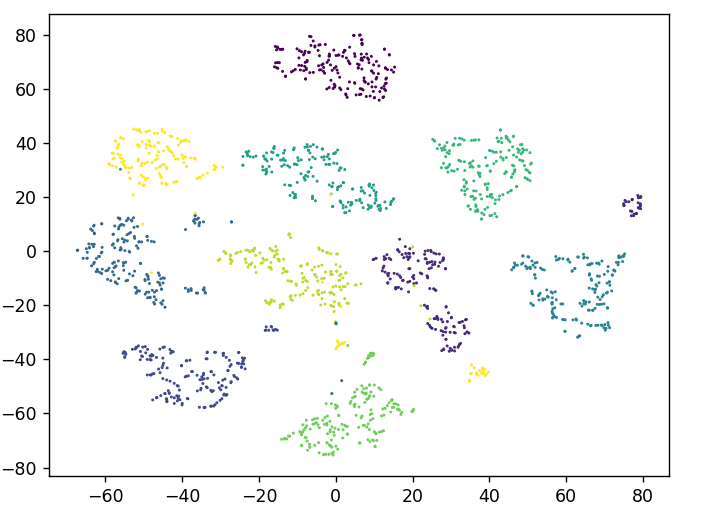


Рисунок 6 – Визуализация t-SNE с перплексией = 10

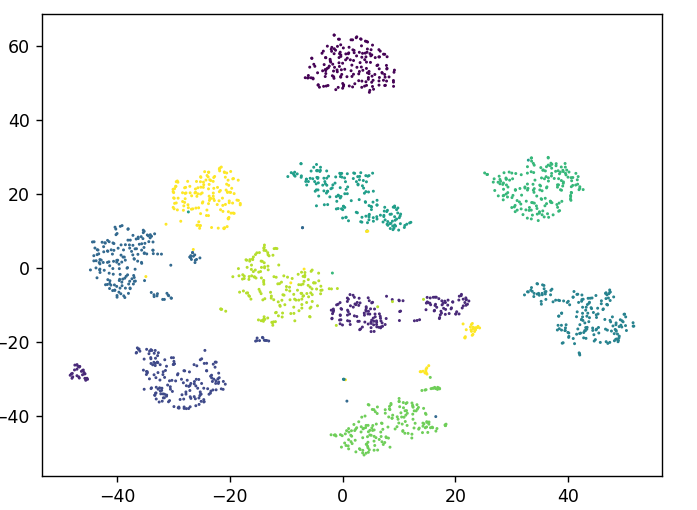


Рисунок 7 – Визуализация t-SNE с перплексией = 30

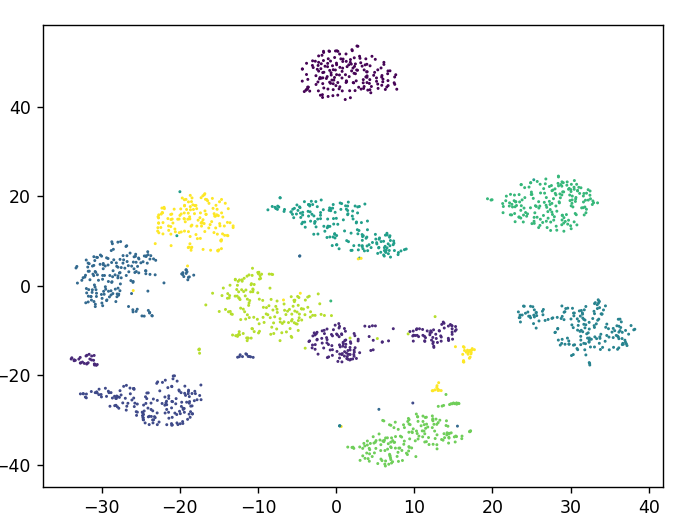


Рисунок 8 – Визуализация t-SNE с перплексией = 50

# ВИЗУАЛИЗАЦИЯ МНОГОМЕРНЫХ ДАННЫХ UMAP

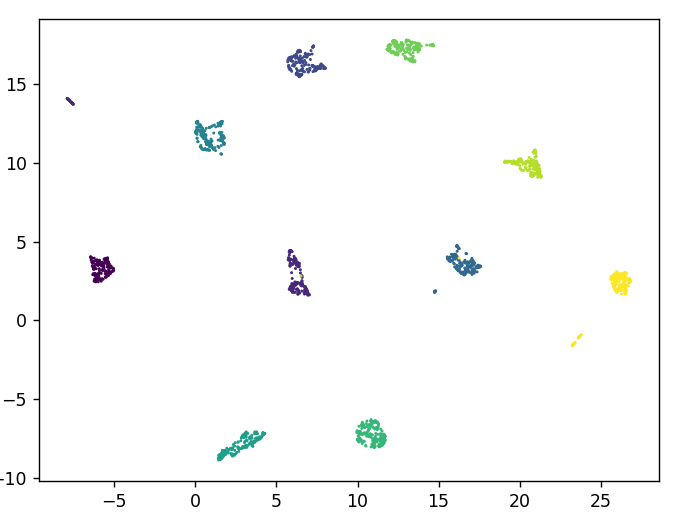
Реализация визуализации многомерных данных с помощью UMAP для датасета MNIST. Датасет MNIST содержит данные о рукописных цифрах. 

Рисунок 9 – Визуализация UMAP, n\_neighbors=10, min\_dist=0.1

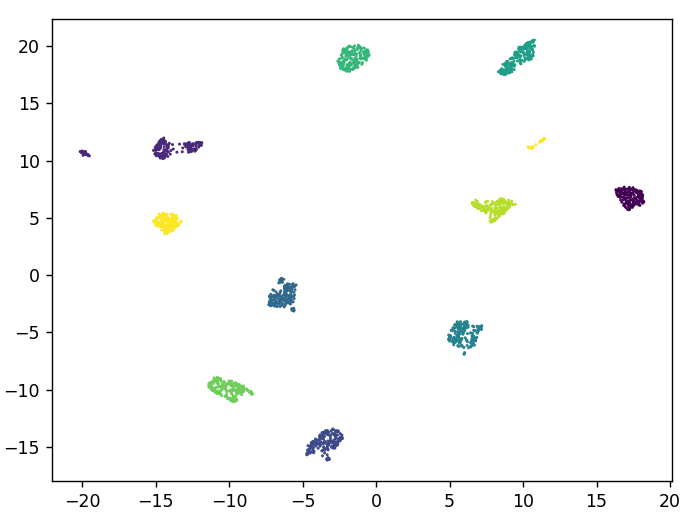


Рисунок 10 – Визуализация UMAP, n\_neighbors=25, min\_dist=0.3

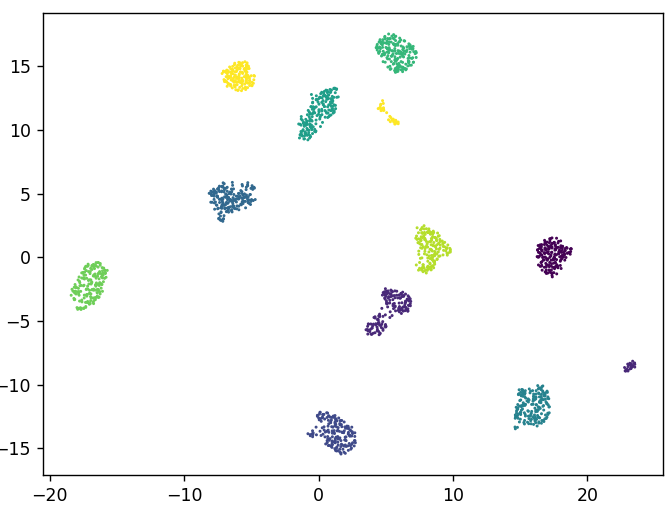


Рисунок 11 – Визуализация UMAP, n\_neighbors=50, min\_dist=0.6

Анализ времени выполнения алгоритмов t-SNE и UMAP:

Время CPU times у алгоритма UMAP значительно ниже (32s – TSNE, 8s - UMAP), это значит, что алгоритм UMAP более производительный.

Общее время выполнение зависит от заданных параметров алгоритмов.

# ПРИЛОЖЕНИЕ

Приложение 1 – Код программы

import matplotlib.pyplot as plt  
import pandas as pd  
import plotly.graph\_objs as go  
from sklearn.datasets import load\_digits  
from sklearn.manifold import TSNE  
import umap  
import time  
  
  
def bar\_grapf(df):  
 fig = go.Figure()  
 fig.add\_bar(x=[j for j in range(len(df))], y=df['subscribers'],  
 marker=dict(color=[i for i in df['subscribers']],  
 colorscale='Inferno', coloraxis="coloraxis",  
 line=dict(color='black', width=2)))  
 fig.update\_layout(title={'text': "Best youtubers statistic",  
 'y': 0.9,  
 'x': 0.5,  
 'xanchor': 'center',  
 'yanchor': 'top'},  
 title\_font\_size=20, height=700, autosize=True  
 )  
 fig.update\_xaxes(title="rank youtuber", title\_font\_size=16, tickangle=315, tickfont\_size=14,  
 showgrid=True, gridwidth=2, gridcolor='ivory')  
 fig.update\_yaxes(title="subscribers", title\_font\_size=16, tickfont\_size=14,  
 showgrid=True, gridwidth=2, gridcolor='ivory')  
 fig.show()  
 return  
  
  
def circle\_grapf(df):  
 fig = go.Figure()  
 low, mid, high = 0, 0, 0  
 new\_arr = []  
 labels = ['subscribers < 35kk', 'subscribers < 80kk', 'subscribers > 80kk']  
 for items in df["subscribers"]:  
 if items < 35\_000\_000:  
 low += 1  
 elif items < 80\_000\_000:  
 mid += 1  
 else:  
 high += 1  
 new\_arr.append(low)  
 new\_arr.append(mid)  
 new\_arr.append(high)  
 fig.add\_trace(go.Pie(values=new\_arr, labels=labels, marker\_line\_width=2))  
 fig.show()  
 return  
  
  
def line\_grapf(df):  
 fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15, 5))  
 ax1.plot([j for j in range(len(df))], df['highest\_yearly\_earnings'], marker='o',  
 color='crimson', markerfacecolor='white', markeredgecolor='black', markersize=2)  
 ax1.set\_xlabel('Rank')  
 ax1.set\_ylabel('Highest yearly earnings')  
 ax1.grid(True, color='mistyrose', linewidth=2)  
 ax2.plot([j for j in range(len(df))], df['subscribers'], marker='o',  
 color='crimson', markerfacecolor='white', markeredgecolor='black', markersize=2)  
 ax2.set\_xlabel('Rank')  
 ax2.set\_ylabel('Subscribers')  
 ax2.grid(True, color='mistyrose', linewidth=2)  
 ax3.plot([j for j in range(len(df))], df['video views'], marker='o',  
 color='crimson', markerfacecolor='white', markeredgecolor='black', markersize=2)  
 ax3.set\_xlabel('Rank')  
 ax3.set\_ylabel('Video views')  
 ax3.grid(True, color='mistyrose', linewidth=2)  
 plt.show()  
 return  
  
  
def tsne\_grapf():  
 start = time.time()  
 X, y = load\_digits(return\_X\_y=True)  
 embed = TSNE(n\_components=2, perplexity=30, random\_state=123)  
 X\_embedded = embed.fit\_transform(X)  
 end = time.time()  
 plt.scatter(X\_embedded[:, 0], X\_embedded[:, 1], c=y, s=0.5)  
 print("Время выполнения функции: ", end - start, "секунд")  
 plt.show()  
 return  
  
  
def umap\_grapf():  
 start = time.time()  
 X, y = load\_digits(return\_X\_y=True)  
 manifold = umap.UMAP(n\_neighbors=25, min\_dist=0.3, random\_state=123).fit(X, y)  
 X\_reduced = manifold.transform(X)  
 end = time.time()  
 plt.scatter(X\_reduced[:, 0], X\_reduced[:, 1], c=y, s=0.5)  
 print("Время выполнения функции: ", end - start, "секунд")  
 plt.show()  
 return  
  
  
def main():  
 df\_orig = pd.read\_csv("youtube\_stat.csv", encoding\_errors="replace")  
 df = df\_orig.drop\_duplicates(subset="subscribers")  
 print(df.info())  
 print(df.head())  
 a = input("Какую диаграмму вывести?(1-столбчатая, 2-круговая, 3-линейная, 4-t-SNE, 5-UMAP): ")  
 a = int(a)  
 if a == 1:  
 bar\_grapf(df)  
 elif a == 2:  
 circle\_grapf(df)  
 elif a == 3:  
 line\_grapf(df)  
 elif a == 4:  
 tsne\_grapf()  
 elif a == 5:  
 umap\_grapf()  
 else:  
 print("Неправильное значение")  
 return  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()