**UiT The Arctic University of Norway**

An exam report for the course

***‘’****HEL-8048-1 24V Advanced data analysis and visualization using programming’’*

**Timofei Plotnikov**

Course coordinator: **Jaime McCutcheon**

Tromsø, April 2024

**Preface**

The current project has been created using data and resources available for free on the internet. Additionally, materials provided by Jaime McCutcheon during the course have been used in the development of this project.

This project represents an effort to implement new skills and demonstrate acquired knowledge from the course.

It has been developed solely for educational purposes and is not intended for commercialization. Upon completion of assessment and approval by the course coordinator, the project will be promptly deleted.

**Table of Contents**

**Table 14**

**Table 25**

**Table 36**

**Table 48**

**Table 510**

**Table 611**

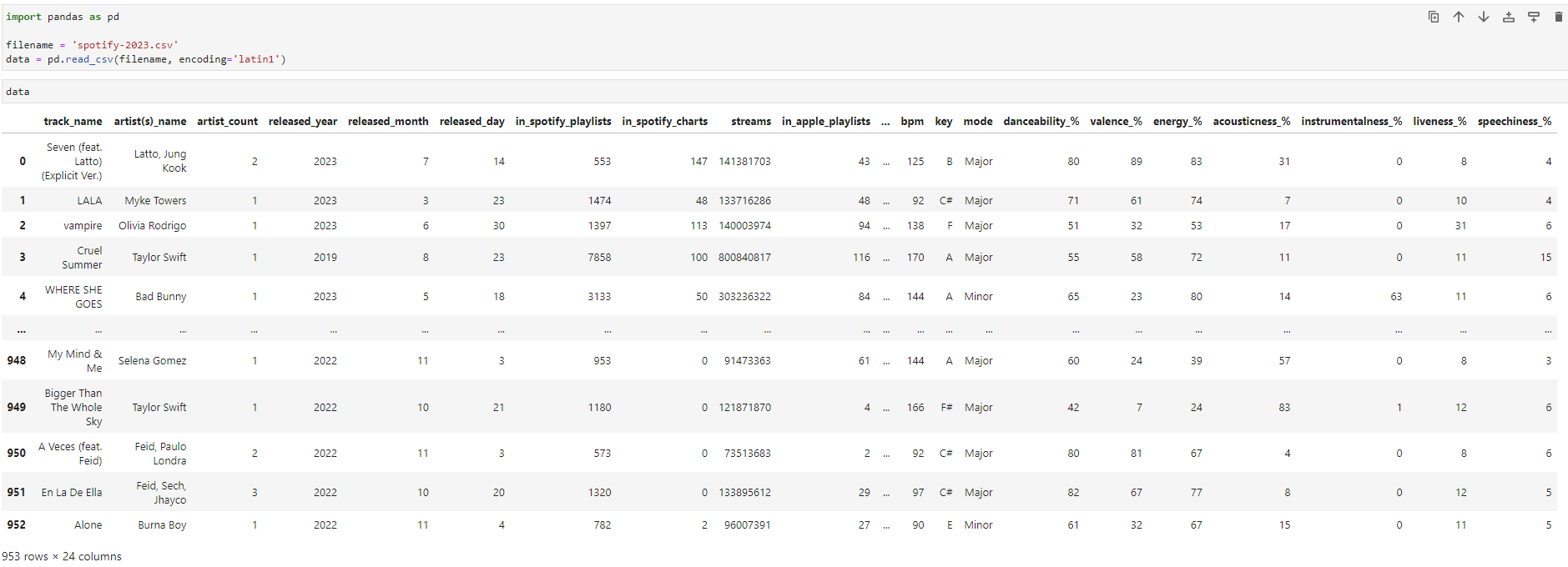
The data "Most Streamed Spotify Songs 🎧🎵", created by *Ahmed Rada*, was found on the free web-source ‘’Kaggle”, and used for further analysis.

The data was imported and visualized from the file named "spotify-2023.csv" using the following code:

import pandas as pd

filename = 'spotify-2023.csv'

data = pd.read\_csv(filename, encoding='latin1')

data

**Table 1.** *Data visualization of the whole spotify-2023.csv file*  
The file represents track names, artists' names, number of streams, danceability, energy, etc.

To carry out further data manipulations, the three biggest artists were chosen: Taylor Swift, Olivia Rodrigo, and Bad Bunny, as they have the greatest number of streams in 2023 according to the graphics.

As these artists are among the most popular, it is interesting to identify which of their songs have amassed more than 1 billion streams. Thus, the following code has been used:

# Convert 'streams' column to numeric

data['streams'] = pd.to\_numeric(data['streams'], errors='coerce')

# Filter the data for the specified artists

artists = ['Bad Bunny', 'Taylor Swift', 'Olivia Rodrigo']

filtered\_data = data[(data['artist(s)\_name'].isin(artists)) & (data['streams'] > 1000000000)]

# Select the desired columns

selected\_columns = ['track\_name', 'artist(s)\_name', 'released\_year', 'streams']

extracted\_data = filtered\_data[selected\_columns]

print(extracted\_data)

A screenshot of a computer

Description automatically generated

**Table 2**. *More than 1,000,000,0 streams*

To find out and visualize the total number of streams, the following code has been used:

import matplotlib.pyplot as plt

# Read the data

filename = 'spotify-2023.csv'

data = pd.read\_csv(filename, encoding='latin1')

# Convert 'streams' column to numeric type

data['streams'] = pd.to\_numeric(data['streams'], errors='coerce')

# Define the artists

artists = ['Taylor Swift', 'Olivia Rodrigo', 'Bad Bunny']

# Filter the data for the specified artists

filtered\_data = data[data['artist(s)\_name'].isin(artists)]

# Group the data by artist and sum the streams

grouped\_data = filtered\_data.groupby('artist(s)\_name')['streams'].sum()

# Plotting

plt.figure(figsize=(10, 6))

grouped\_data.plot(kind='bar', color=['blue', 'green', 'red'])

plt.title('Total Streams by Artist')

plt.xlabel('Artist')

plt.ylabel('Total Streams')

plt.xticks(rotation=45)

plt.show()

A graph of different colors

Description automatically generated

**Table 3**. *Total streams by artists – comparison*

The colors blue, green, and red are used in these charts because they are easy to tell apart. Even people who have trouble seeing colors can usually distinguish between them. This makes it simple for everyone to understand the information in the presented chart.

To determine the variation of specific attributes (acousticness and energy in this case), the following code is used:

import pandas as pd

import matplotlib.pyplot as plt

# Read the data

filename = 'spotify-2023.csv'

data = pd.read\_csv(filename, encoding='latin1')

# Define the artists

artists = ['Taylor Swift', 'Olivia Rodrigo', 'Bad Bunny']

# Filter the data for the specified artists

filtered\_data = data[data['artist(s)\_name'].isin(artists)]

# Plotting

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

colors = {'Taylor Swift': 'blue', 'Olivia Rodrigo': 'green', 'Bad Bunny': 'red'}

for artist, group in filtered\_data.groupby('artist(s)\_name'):

plt.scatter(group['acousticness\_%'], group['energy\_%'], color=colors[artist], alpha=0.5, edgecolors='w', s=80, label=artist)

plt.title('Scatter Plot between acousticness\_% and energy\_%')

plt.xlabel('Acousticness (%)')

plt.ylabel('Energy (%)')

plt.legend()

plt.grid(True)

plt.show()

A graph with different colored dots

Description automatically generated

**Table 4**. *Comparison of energy and acousticness between the artists*

The same colors are used in the charts for the reasons mentioned earlier. Additionally, the colors match, making it easier for others to follow along, as they will associate specific colors with specific artists.

Due to different alphabet systems, the names of artists might be translated into local languages. For example, this occurs with the Russian language due to the Cyrillic alphabet, and quite often, Latin-based names are not acceptable. Therefore, there is a need to implement an API (*Google Translate* in our case) from rapidapi.com. **Registration and free sign-up to use this API are essential for the code to function.**

Code 1:

import requests

url = "https://google-translate1.p.rapidapi.com/language/translate/v2"

payload = {

"q": "Bad Bunny, Olivia Rodrigo, Taylor Swift",

"target": "ru",

"source": "en"

}

headers = {

"content-type": "application/x-www-form-urlencoded",

"Accept-Encoding": "application/gzip",

"X-RapidAPI-Key": "00a536a0a9msh802d70bdcca163dp145d2ajsn070bd865acb1",

"X-RapidAPI-Host": "google-translate1.p.rapidapi.com"

}

response = requests.post(url, data=payload, headers=headers)

print(response.json())

Code 2:

data = {'data': {'translations': [{'translatedText': 'Плохой Банни, Оливия Родриго, Тейлор Свифт'}]}}

translated\_text = data['data']['translations'][0]['translatedText']

print(translated\_text)



**Table 5**. *API translation result*

In order to demonstrate the utilization of translation data from API manipulations, I have decided to combine two codes: 'Total Streams by Artists – Comparison' and 'API Translation Result':

import pandas as pd

import matplotlib.pyplot as plt

# Read the data

filename = 'spotify-2023.csv'

data = pd.read\_csv(filename, encoding='latin1')

# Convert 'streams' column to numeric type

data['streams'] = pd.to\_numeric(data['streams'], errors='coerce')

# Define the artists

artists = ['Taylor Swift', 'Olivia Rodrigo', 'Bad Bunny']

# Filter the data for the specified artists

filtered\_data = data[data['artist(s)\_name'].isin(artists)]

# Group the data by artist and sum the streams

grouped\_data = filtered\_data.groupby('artist(s)\_name')['streams'].sum()

# Plotting

plt.figure(figsize=(10, 6))

grouped\_data.plot(kind='bar', color=['blue', 'green', 'red'])

plt.title('Total Streams by Artist')

plt.xlabel('Artist')

plt.ylabel('Total Streams')

plt.xticks(rotation=45)

# Translation dictionary

translation\_dict = {

'Bad Bunny': 'Плохой Банни',

'Taylor Swift': 'Тейлор Свифт',

'Olivia Rodrigo': 'Оливия Родриго'

}

# Translate artist names

translated\_artists = [translation\_dict.get(artist, artist) for artist in grouped\_data.index]

# Update the xtick labels with translated names

plt.gca().set\_xticklabels(translated\_artists)

plt.show()

A graph of different colors

Description automatically generated**Table 6**. *Total streams by artists - comparison + API translation result*