Introduction

The dataset I chose is named Spotify-2023. It is a CSV (comma-separated values) file that can be accessed through the following URL: https://www.kaggle.com/datasets/nelgiriyewithana/top-spotify-songs-2023. The dataset originated from the Spotify platform and was collected using the Spotify API by Nidula Elgiriyewithana through web scraping techniques. I utilized Python programming language through jupyter notebook to explore the dataset's structure and to get a feel of what the data is about and what the schema looks like. This helps in understanding what the dataset represents.

Structure of the dataset

The screenshot snippet below shows that this dataset is structured in a tabular format with 953 records (rows) representing the different songs and 23 columns representing the variables or attributes of the songs. The attributes are represented by numerical or categorical values. These attributes include features like tempo, energy, danceability, and loudness (numerical), as well as artist names, track titles, and genre labels (categorical), among others. Each attribute provides unique insights into the characteristics of the songs and their associated metadata.

```
In [6]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 953 entries, 0 to 952
Data columns (total 24 columns):
 #   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
```

Below is a sample of how the dataset looks like.

```
In [3]: df = pd.read_csv('spotify-2023.csv', encoding='ISO-8859-1')
        df.head()
```

Out[3]:

	track_name	artist(s)_name	artist_count	released_year	released_month	released_day	in_spotify_playlists	in_spotify_charts	streams	in_apple_playlists	...
0	Seven (feat. Latto) (Explicit Ver.)	Latto, Jung Kook	2	2023	7	14	553	147	141381703	43	...
1	LALA	Myke Towers	1	2023	3	23	1474	48	133716286	48	...
2	vampire	Olivia Rodrigo	1	2023	6	30	1397	113	140003974	94	...
3	Cruel Summer	Taylor Swift	1	2019	8	23	7858	100	800840817	116	...
4	WHERE SHE GOES	Bad Bunny	1	2023	5	18	3133	50	303236322	84	...

5 rows × 24 columns

The main feature(s) of interest in the dataset

My Question

1. Does a song with several featuring artists have a better streaming performance than one with just one artist?

   The purpose of this question is to investigate if there is a significant difference in the average number of streams for songs featuring multiple artists and those with the single featuring artist. I want to assess whether collaborations with other artists results in higher streaming numbers. This will provide insights into the potential benefits of including featuring artists to a song for its streaming success.

2. What features of music are associated with the total number of streams on Spotify?

   The purpose of the question is to understand the factors or characteristics of music tracks that influence their popularity and streaming success on the Spotify platform. I want to identify the correlation between the features and the stream numbers.

Question 1

Data Preprocessing

I first converting the 'streams' column from object data type to numeric data type using the pd.to_numeric () function from the pandas library. I then created two subsets based on the number of artists associated with each song. The first subset, 'songs_multiple_artists', comprises songs featuring more than one artist and the second subset, 'songs_single_artist', includes songs attributed to a single artist as shown in the code below.

```
In [9]: df2['streams'] = pd.to_numeric(df['streams'], errors='coerce')
        df2['streams'].dtype

Out[9]: dtype('float64')
```

```
In [10]: # create two dataframes for multiple artists and single artists
         songs_multiple_artists = df2[df2['artist_count'] > 1]
         songs_single_artist = df2[df2['artist_count'] == 1]
```

This separation enables focused analysis and comparison between songs with varying levels of artist collaboration, which will offer insights into potential differences in streaming performance and audience engagement across these categories. After separation, the next phase involved handling missing values within the dataset. Null values, were removed to maintain data integrity. After completing these data cleaning processes, the dataset was ready for analysis.

Data Analysis

I start by determining the average streaming performance for each group; those featuring multiple artists and those attributed to a single artist. This analysis helps in understanding the typical streaming performance of songs with differing levels of artist collaboration, shedding light on potential preferences or trends among listeners. The code and results are as below

```
# Calculate the average total streams for each group
avg_streams_multiple_artists = round(songs_multiple_artists['streams'].mean())
avg_streams_single_artist = round(songs_single_artist['streams'].mean())

# Print the average streaming performance for each group
print("Average streams for songs with multiple artists:", avg_streams_multiple_artists)
print("Average streams for songs with a single artist:", avg_streams_single_artist)
```

```
Average streams for songs with multiple artists: 427559548
Average streams for songs with a single artist: 568211662
```

We can see that streams with a single artist are higher compared with the streams with multiple artists but is this difference statistically significant to make the conclusion that a song with a single artist has a better streaming performance than a song with several featuring artists. To ultimately make this conclusion we need to do further test to check if the difference is statistically significant.

To decide which test to use, I first conduct variance and normality tests.

Variance

Levene's test is performed to assess the equality of variances between the streaming performance of songs with multiple artists and those with a single artist. Levene's test is a statistical method used to determine whether the variances of two or more groups are significantly different from each other (Mishra et.al., 2019). This is important because many statistical tests, such as t-tests or ANOVA, assume equal variances among groups. The code and the results are as below.

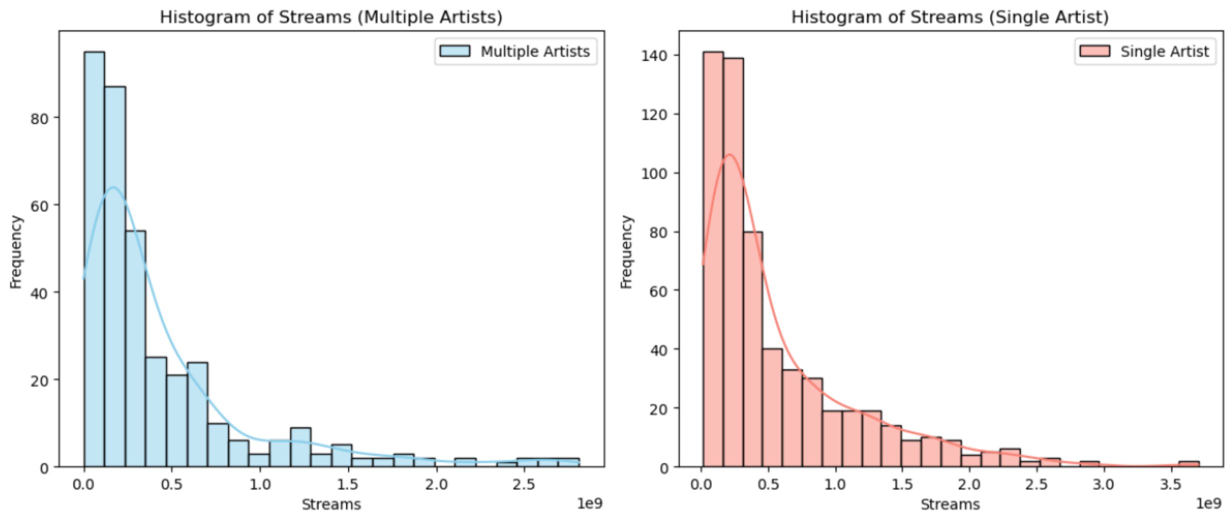
```
# Perform Levene's test
statistic, p_value = levene(streams_multiple_artists, streams_single_artist)

# Print the results
print("Levene's test statistic:", statistic)
print("p-value:", p_value)
```

```
Levene's test statistic: 8.547723062690855
p-value: 0.003541673356671995
```

With a Levene's test statistic of approximately 8.55 and a p-value of approximately 0.0035, we have evidence to reject the null hypothesis of equal variances. This indicates that the variances of the streaming data for songs with multiple artists and songs with a single artist are significantly different.

Normality

Graphical representation of the data using a histogram was plotted to assess normality and the results are shown below. Both histograms are skewed, indicating the data is not normally distributed.



To confirm I employed The Shapiro-Wilk test which is a statistical method used to determine whether a sample comes from a normally distributed population. When P > 0.05, null hypothesis accepted and data are considered as normally distributed (Mishra et.al., 2019).

```
# Statistcal Normality Tests
print("Shapiro-Wilk Test:")
_, p_value_multi = stats.shapiro(streams_multiple_artists)
_, p_value_single = stats.shapiro(streams_single_artist)
print("p-value for multiple artists:", p_value_multi)
print("p-value for single artist:", p_value_single)
```

```
Shapiro-Wilk Test:
p-value for multiple artists: 7.7163854324203325e-25
p-value for single artist: 8.110797426918251e-27
```

The extremely small p-values obtained for both groups (7.72e-25 for songs with multiple artists and 8.11e-27 for songs with a single artist) indicate strong evidence against the null hypothesis of normality. Therefore, it can be concluded that the streaming performance data for both groups are significantly non-normally distributed.

Since the assumptions of normality and equal variances are both violated, it's recommended to use Mann-Whitney/Wilcoxon Rank Sum, which is a non-parametric test, meaning it does not assume that the data are normally distributed (Fischetti, 2018). Instead, it compares the distributions of the two groups based on their ranks. The code and results are as shown below

```
# Perform Mann-Whitney U test on original data
statistic, p_value = mannwhitneyu(streams_multiple_artists, streams_single_artist)

# Print the results
print("Mann-Whitney U test statistic:", statistic)
print("p-value:", p_value)
```

```
Mann-Whitney U test statistic: 88464.0
p-value: 5.394400932557491e-06
```

Discussions and conclusions

The small p-value (less than the chosen significance level of 0.05) indicates that there is a statistically significant difference in streaming performance between the two groups. In other words, there is strong evidence to reject the null hypothesis that the mean streaming numbers are equal between the two groups. We can therefore conclude that a song with a single artist has a better streaming performance than a song with several featuring artists.

Question 2

Data Preprocessing

In preparing the dataset for analysis, a subset of the original dataset's features that I deemed relevant were chosen.

```
# create a new df with intrested features

features = df2[['streams','bpm','danceability_%','valence_%','energy_%','acousticness_%','instrumentalness_%',
              'liveness_%','speechiness_%']]
```

```
features.head()
```

	streams	bpm	danceability_%	valence_%	energy_%	acousticness_%	instrumentalness_%	liveness_%	speechiness_%
0	141381703.0	125	80	89	83	31	0	8	4
1	133716286.0	92	71	61	74	7	0	10	4
2	140003974.0	138	51	32	53	17	0	31	6
3	800840817.0	170	55	58	72	11	0	11	15
4	303236322.0	144	65	23	80	14	63	11	6

I created a new dataset from these features and the other columns were discarded. Particularly for the ‘streams' column, a data scaling technique was used to guarantee consistency and comparability among the chosen features. This was accomplished by using the scikit-learn library's MinMaxScaler. The values were normalized to a range between 0 and 1.

```
In [31]: from sklearn.preprocessing import MinMaxScaler
         scaler = MinMaxScaler()

         features['streams'] = scaler.fit_transform(features[['streams']])
         features.head()
```

Out[31]:

	streams	bpm	danceability_%	valence_%	energy_%	acousticness_%	instrumentalness_%	liveness_%	speechiness_%
0	0.038170	125	80	89	83	31	0	8	4
1	0.036101	92	71	61	74	7	0	10	4
2	0.037798	138	51	32	53	17	0	31	6
3	0.216215	170	55	58	72	11	0	11	15
4	0.081869	144	65	23	80	14	63	11	6

This preprocessing step is essential because it reduces the possibility that some characteristics may predominate simply because of their greater magnitudes. This ensures that all selected attributes are fairly and meaningfully compared.

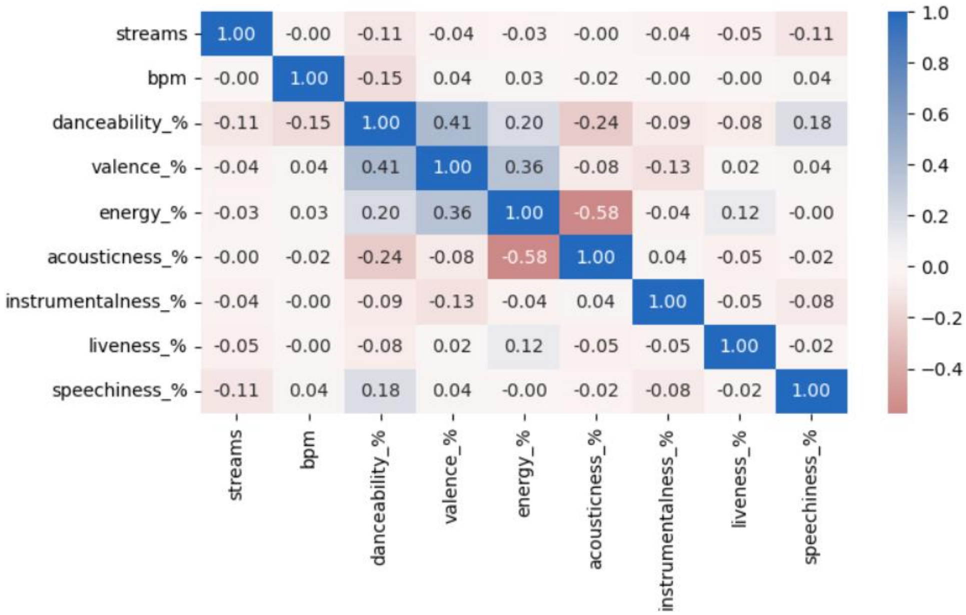
After the preprocessing steps of feature selection and data scaling, the next phase involved handling missing values within the dataset. Null values, were removed to maintain data integrity. After completing these data cleaning processes, the dataset was ready for analysis.

Data Analysis

After preparing the dataset, the next step involved conducting data analysis to gain insights into the relationships between the selected features and the target variable, in this case, the 'streams' column representing Spotify streams.

```
In [36]: plt.figure(figsize = [8, 6])
         sns.heatmap(features.corr(), annot = True, fmt = '.2f', cmap = 'vlag_r', center = 0)
         plt.show()
```

The provided code utilizes a heatmap to visualize the correlation matrix between the features. This heatmap, generated using seaborn's heatmap function, displays the correlation coefficients between each pair of features as shown below.



According to Akoglu (2018), correlation coefficients quantify the strength and direction of linear relationships between variables, ranging from -1 to 1. Values closer to 1 indicate a strong positive correlation, while values closer to -1 indicate a strong negative correlation. A value of 0 suggests no linear correlation. The matrix does not show enough to make conclusions on.

To assess better, I conducted a statistical test that evaluates the relationship between each feature and the streaming numbers. The next step in the analysis involved determining which features of music are associated with the total number of streams on Spotify. I employed a linear regression analysis to model the relationship between the selected features (independent variables) and the target variable, Spotify streams (dependent variable).

```
In [38]: # Linear Regression Analysis
         X = features.drop(columns=['streams'])
         X = sm.add_constant(X)  # Add a constant term for the intercept
         y = features['streams']

         model = sm.OLS(y, X).fit()
         print("\nLinear Regression Results:")
         print(model.summary())
```

In the provided code, the independent variables are stored in the DataFrame X, from which the 'streams' column is dropped, as it serves as the dependent variable. The dependent variable, Spotify streams, is stored in the Series y.

Results and Conclusion

```
Linear Regression Results:
                            OLS Regression Results
==============================================================================
Dep. Variable:                streams   R-squared:                       0.029
Model:                            OLS   Adj. R-squared:                  0.021
Method:                 Least Squares   F-statistic:                     3.558
Date:                Fri, 26 Apr 2024   Prob (F-statistic):           0.000450
Time:                        17:32:21   Log-Likelihood:                 450.76
No. Observations:                 952   AIC:                            -883.5
Df Residuals:                     943   BIC:                            -839.8
Df Model:                           8
Covariance Type:            nonrobust
======================================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------
const                  0.2799      0.046      6.128      0.000       0.190       0.370
bpm                 -8.31e-05      0.000     -0.465      0.642      -0.000       0.000
danceability_%        -0.0011      0.000     -2.886      0.004      -0.002      -0.000
valence_%           5.918e-05      0.000      0.235      0.814      -0.000       0.001
energy_%              -0.0003      0.000     -0.761      0.447      -0.001       0.000
acousticness_%        -0.0003      0.000     -1.253      0.211      -0.001       0.000
instrumentalness_%    -0.0012      0.001     -1.957      0.051      -0.002    3.27e-06
liveness_%            -0.0007      0.000     -1.873      0.061      -0.001    3.23e-05
speechiness_%         -0.0015      0.001     -3.044      0.002      -0.003      -0.001
==============================================================================
Omnibus:                      377.983   Durbin-Watson:                   1.521
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1334.683
Skew:                           1.944   Prob(JB):                    1.50e-290
Kurtosis:                       7.305   Cond. No.                     1.56e+03
==============================================================================
```

The results of the linear regression analysis indicate that the model has an R-squared value of 0.029, suggesting that approximately 2.9% of the variability in Spotify streams can be explained by the selected features. The adjusted R-squared value, which accounts for the number of predictors in the model, is slightly lower at 0.021.

Examining the p-values for each feature, only 'danceability_%' and 'speechiness_%' exhibit statistically significant coefficients. This shows an increase in 'danceability_%' and 'speechiness_%' is associated with a decrease in Spotify streams.

Overall, the low R-squared value indicates that the chosen characteristics would not adequately represent the variability in stream counts, even though the model offers some insights into the association between attributes and Spotify streams. To increase the predicted accuracy of the model, more research or model improvement may be required. To further improve the explanatory power of the current model, other features or elements not included in it could be taken into account.

References

Akoglu, H. (2018). User's guide to correlation coefficients. Turkish journal of emergency medicine, 18(3), 91-93.

Mishra, P., Pandey, C. M., Singh, U., Gupta, A., Sahu, C., & Keshri, A. (2019). Descriptive statistics and normality tests for statistical data. Annals of cardiac anaesthesia, 22(1), 67.

Fischetti, T. (2018). Data Analysis with R, Second Edition. Packt Publishing Ltd.