CS5811 Distributed Data Analysis

1. Data Description and Research Question:

The dataset includes a list of well-known songs over the past few years as listed on Spotify. In addition to Spotify, it provides information on the number of playlists and charts a song is in for Apple, Deezer, and Shazam. The **size** of the dataset is 106kB and contains 24 columns. The dataset has been **sourced** from Kaggle and the link to it is Most Streamed Spotify Songs 2023 (kaggle.com). Our target variable is streams. The metadata of the dataset is as follows:

Feature	Description
Track Name	Name of the song
Artist(s) Name	Name of the artist(s) of the song
Artist Count	Number of artists contributing to the song
Released Year	Year when the song was released
Released Month	Month when the song was released
Released Day	Day of the month when the song was released
In Spotify Playlists	Number of Spotify playlists the song is included in
In Spotify Charts	Presence and rank of the song on Spotify charts
Streams	Total number of streams on Spotify
In Apple Playlists	Number of Apple Music playlists the song is included in
In Apple Charts	Presence and rank of the song on Apple Music charts
In Deezer Playlists	Number of Deezer playlists the song is included in
In Deezer Charts	Presence and rank of the song on Deezer charts
In Shazam Charts	Presence and rank of the song on Shazam charts
BPM	Beats per minute, a measure of song tempo
Key	Key of the song
Mode	Mode of the song (major or minor)
Danceability %	Percentage indicating how suitable the song is for dancing
Valence %	Positivity of the song's musical content
Energy %	Perceived energy level of the song
Acousticness %	Amount of acoustic sound in the song
Instrumentalness %	Amount of instrumental content in the song
Liveness %	Presence of live performance elements
Speechiness %	Amount of spoken words in the song

Our **research question** aims to predict the number of streams for music tracks in the dataset based on their characteristics.

2. Data Preparation and Cleaning:

a) **Changed Column Names:** The original column name “artist.s._name” was changed to “artist_name” for better readability. Also, we changed “%” with “perc”, so for example, “danceability_%” to “danceability_perc”.

b) **Categorical Variables to Factors:** All the verified categorical columns were changed to factors. Additionally, integer variable, 'released_month' was transformed into string to show short forms of months, such as Jan and Feb, for better readability.

```
 $ track_name          : chr  "Seven (feat. Latto) (Explicit Ver.)" "LALA" "vampire"
"Cruel Summer" ...
 $ artist.s._name      : chr  "Latto, Jung Kook" "Myke Towers" "Olivia Rodrigo"
"Taylor Swift" ...
 $ artist_count        : int  2 1 1 1 1 2 2 1 1 2 ...
 $ released_year       : int  2023 2023 2023 2019 2023 2023 2023 2023 2023 2023 ...
 $ released_month      : int  7 3 6 8 5 6 3 7 5 3 ...
 $ released_day        : int  14 23 30 23 18 1 16 7 15 17 ...
 $ in_spotify_playlists: int  553 1474 1397 7858 3133 2186 3090 714 1096 2953 ...
 $ in_spotify_charts   : int  147 48 113 100 50 91 50 43 83 44 ...
 $ streams             : chr  "141381703" "133716286" "140003974" "800840817" ...
 $ in_apple_playlists  : int  43 48 94 116 84 67 34 25 60 49 ...
 $ in_apple_charts     : int  263 126 207 207 133 213 222 89 210 110 ...
 $ in_deezer_playlists : chr  "45" "58" "91" "125" ...
 $ in_deezer_charts    : int  10 14 14 12 15 17 13 13 11 13 ...
 $ in_shazam_charts    : chr  "826" "382" "949" "548" ...
 $ bpm                 : int  125 92 138 170 144 141 148 100 130 170 ...
 $ key                 : chr  "B" "C#" "F" "A" ...
 $ mode                : chr  "Major" "Major" "Major" "Major" ...
 $ danceability_.      : int  80 71 51 55 65 92 67 67 85 81 ...
 $ valence_.           : int  89 61 32 58 23 66 83 26 22 56 ...
 $ energy_.            : int  83 74 53 72 80 58 76 71 62 48 ...
 $ acousticness_.      : int  31 7 17 11 14 19 48 37 12 21 ...
 $ instrumentalness_.  : int  0 0 0 0 63 0 0 0 0 0 ...
 $ liveness_.          : int  8 10 31 11 11 8 8 11 28 8 ...
 $ speechiness_.       : int  4 4 6 15 6 24 3 4 9 33 ...
[1] 953  24
```

Figure 1: Before Data Cleaning

```
 $ track_name          : chr  "Seven (feat. Latto) (Explicit Ver.)" "LALA" "vampire" "Cruel
Summer" ...
 $ artist_name         : chr  "Latto, Jung Kook" "Myke Towers" "Olivia Rodrigo" "Taylor Swift"
...
 $ artist_count        : Factor w/ 8 levels "1","2","3","4",..: 2 1 1 1 1 2 2 1 1 2 ...
 $ released_year       : Factor w/ 50 levels "1930","1942",..: 50 50 50 46 50 50 50 50 50 50 ...
 $ released_month      : Ord.factor w/ 12 levels "Jan"<"Feb"<"Mar"<..: 7 3 6 8 5 6 3 7 5 3 ...
 $ released_day        : Factor w/ 31 levels "1","2","3","4",..: 14 23 30 23 18 1 16 7 15 17 ...
 $ in_spotify_playlists : int  553 1474 1397 7858 3133 2186 3090 714 1096 2953 ...
 $ in_spotify_charts   : int  147 48 113 100 50 91 50 43 83 44 ...
 $ streams             : num  1.41e+08 1.34e+08 1.40e+08 8.01e+08 3.03e+08 ...
 $ in_apple_playlists  : int  43 48 94 116 84 67 34 25 60 49 ...
 $ in_apple_charts     : int  263 126 207 207 133 213 222 89 210 110 ...
 $ in_deezer_playlists : int  45 58 91 125 87 88 43 30 48 66 ...
 $ in_deezer_charts    : int  10 14 14 12 15 17 13 13 11 13 ...
 $ in_shazam_charts    : int  826 382 949 548 425 946 418 194 953 339 ...
 $ bpm                 : int  125 92 138 170 144 141 148 100 130 170 ...
 $ key                 : Factor w/ 11 levels "A","A#","B","C#",..: 3 4 8 1 1 4 8 8 4 5 ...
 $ mode                : Factor w/ 2 levels "Major","Minor": 1 1 1 1 2 1 2 1 2 2 ...
 $ danceability_perc   : int  80 71 51 55 65 92 67 67 85 81 ...
 $ valence_perc        : int  89 61 32 58 23 66 83 26 22 56 ...
 $ energy_perc         : int  83 74 53 72 80 58 76 71 62 48 ...
 $ acousticness_perc   : int  31 7 17 11 14 19 48 37 12 21 ...
 $ instrumentalness_perc: int  0 0 0 0 63 0 0 0 0 0 ...
 $ liveness_perc       : int  8 10 31 11 11 8 8 11 28 8 ...
 $ speechiness_perc    : int  4 4 6 15 6 24 3 4 9 33 ...
```

Figure 2: After Data Cleaning

c) **Handled Missing Values:** Blank values were identified in the dataset and replaced with 'NA'. Subsequently, they were imputed using the hot-deck imputation method.

track_name	artist_name	artist_count	released_year
0	0	0	0
released_month	**released_day**	**in_spotify_playlists**	**in_spotify_charts**
0	0	0	0
streams	**in_apple_playlists**	**in_apple_charts**	**in_deezer_playlists**
1	0	0	0
in_deezer_charts	**in_shazam_charts**	**bpm**	**key**
0	50	0	95
mode	**danceability_perc**	**valence_perc**	**energy_perc**
0	0	0	0
acousticness_perc	**instrumentalness_perc**	**liveness_perc**	**speechiness_perc**
0	0	0	0

Figure 3: Missing Values in the Dataset Before Imputation

3. Exploratory Data Analysis

a) Data Distribution:

It was noted that variables such as "in_spotify_charts", "in_spotify_playlists" as well as analogous variables for Apple Music, Deezer, and Shazam, exhibited skewed distributions. Specifically, their mean and median values were observed to be around 20% of the data spread. This skewness indicates a notable concentration of instances towards the lower end of the spectrum for these variables

Figure 4: Top Streamed Artists

b) Univariate Analysis:

- It was observed that The Weeknd, Taylor Swift, and Ed Sheeran stood out as the artists with the highest streams.
- Most of the songs were released in the year 2022, followed by 2023. January and May were identified as the months with the highest number of song releases. A lot of songs were released on the first day of any month. This trend suggests that these periods may be strategically significant for artists and record labels aiming to maximize exposure and streaming potential.
- Majority of songs in the dataset have both danceability and energy percentages exceeding 50%. This indicates a trend towards songs that are both rhythmically engaging and dynamically intense, which aligns with listener preferences for more upbeat and energetic music content. Valence was found to be balanced at around 50% and instrumentalness in most of the songs is nearly non-existent.

c) Multivariate Analysis:

- Songs released between the years 2016 and 2019 accrued significantly higher streams compared to songs released in other years within the dataset. This period appears to represent a peak in streaming activity.
- While no consistent trend was observed across all months, songs released in January and September exhibited slightly higher streaming numbers compared to other months. This uptick in streaming activity during these months could potentially be attributed to factors such as post-holiday. Interestingly, December stood out with comparatively lower streaming numbers compared to other months. This observation aligns with the holiday season, where listeners may be preoccupied with festivities and may engage less with new music releases.
- Despite the absence of a clear relationship between key and mode vs streaming performance, it is notable that the top three streamed songs were all in the key of C#.
- We found a strong positive correlation between stream counts on Spotify and Apple Music, indicating synchronized popularity (Figure 6 and 7). A moderate correlation was observed with Deezer, while the correlation with Shazam was weaker.
- We found that as the percentages of danceability, valence, energy, instrumentalness, liveness, and speechiness increase, there's a consistent downward trend in streaming numbers (Figure 5).

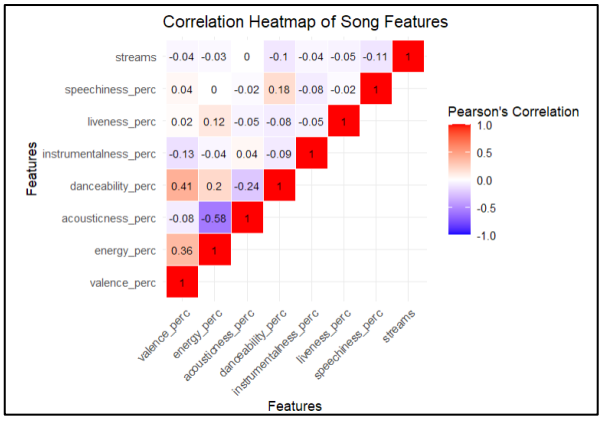

Figure 5: Heatmap of Song Features and Streams

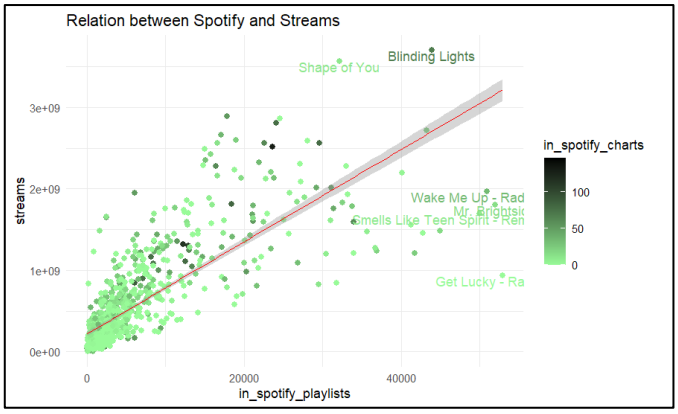

Figure 6: Spotify vs Streams

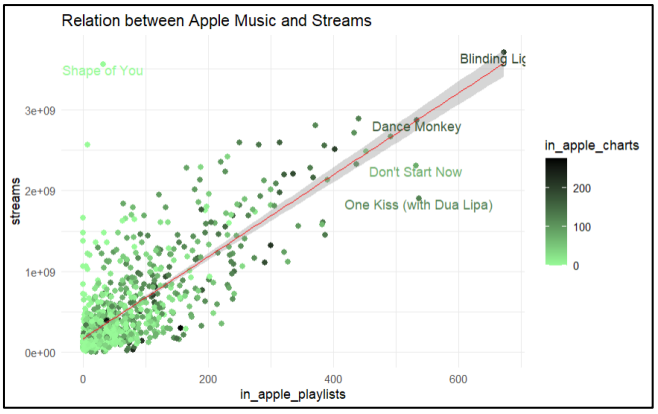

Figure 7: Apple Music vs Streams

d) Principal Component Analysis:

Principal Component Analysis (PCA) was performed to reduce the dimensionality of the dataset and uncover underlying patterns. Since it uses only numerical variables, we used "in_spotify_playlists", "in_spotify_charts", "in_apple_playlists", "in_apple_charts", "in_deezer_playlists", "in_deezer_charts", "in_shazam_charts", "bpm", "danceability_perc", "valence_perc", "energy_perc", "acousticness_perc", "instrumentalness_perc", "liveness_perc", "speechiness_perc" columns into PCA.

The first eight principal components were examined which cumulatively expalined 80% of the total variance in the data.

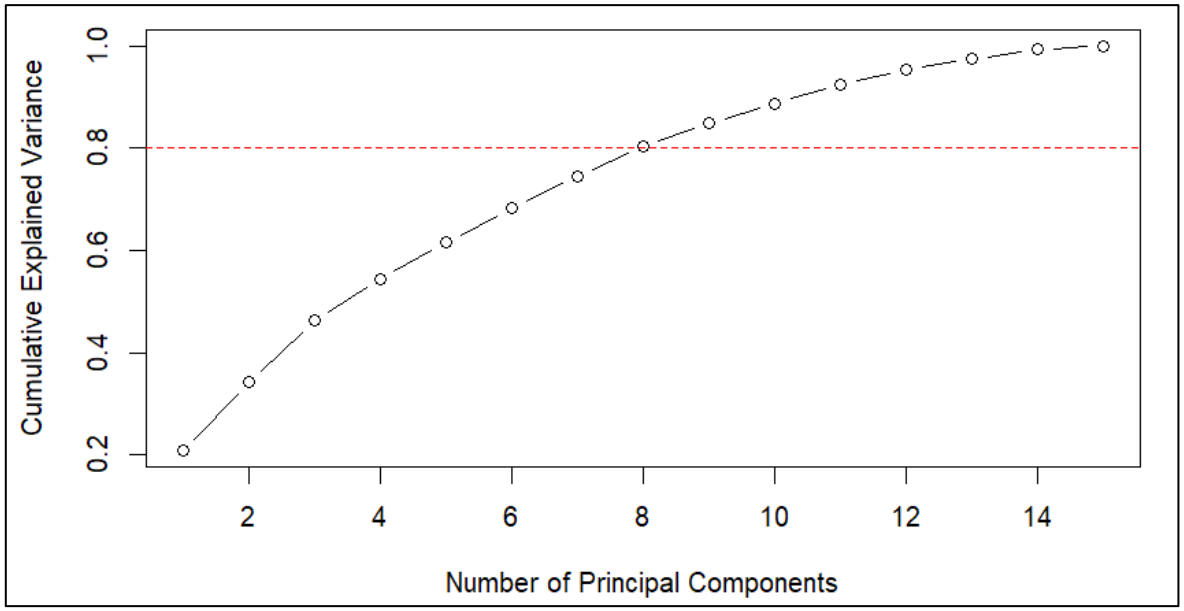

Figure 8: Scree Plot for PCA

- Principal Component 1 (PC1): This component underscores the significance of platform presence, notably within Spotify and Apple Music playlists. The strong positive associations with variables related to these platforms signify their pivotal role in determining song visibility and popularity. While musical attributes such as BPM and Energy contribute, they appear to play a secondary role compared to platform-related variables.
- Principal Component 2 (PC2): Here, the focus shifts towards musical attributes over platform-related variables. The negative association with Acousticness suggests a distinct preference for non-acoustic or less acoustic tracks within the dataset, adding an intriguing dimension to our analysis.

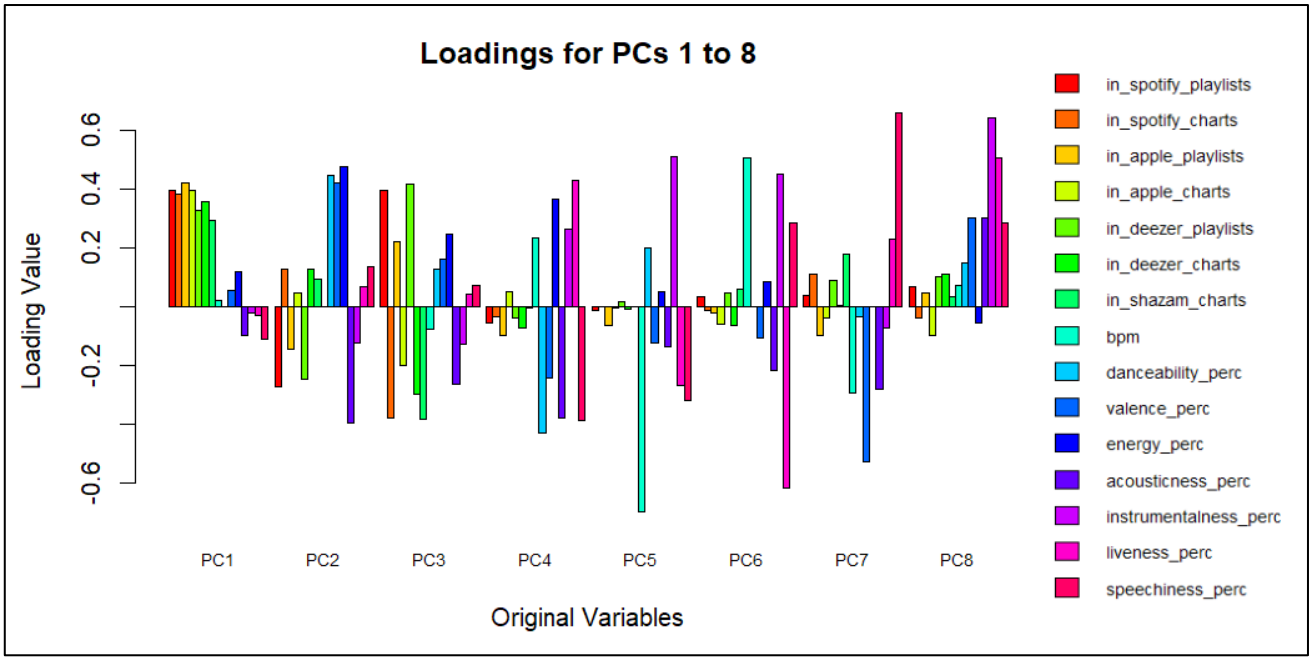

Figure 9: Principal Component Loading Values

- Principal Component 3 (PC3): PC3 primarily captures the variance related to songs' presence in playlists and charts across platforms. The emphasis on energetic and positive musical attributes hints at a prevailing trend towards high-energy compositions within our dataset.
- Principal Component 4 (PC4): Combining musical attributes with platform-related factors, PC4 stands out for its preference for high-energy, instrumental, and live performance-oriented compositions. This component offers a holistic view of the intricate interplay between musical attributes and platform-related dynamics.
- Principal Component 5 (PC5): PC5 presents a diverse range of associations, notably highlighting a positive association with Danceability. This suggests a proclivity towards highly danceable songs, adding another layer of insight to our exploration.
- Principal Component 6 (PC6): With associations related to presence in Deezer playlists and charts alongside certain musical attributes, PC6 unveils additional nuances within our dataset.
- Principal Component 7 (PC7): Focused on platform-related variables, particularly presence in Shazam charts, PC7 hints at an inclination towards more vocal-centric content, as evidenced by the positive association with Speechiness.
- Principal Component 8 (PC8): Exhibiting a balanced representation of musical attributes and platform-related factors, PC8 showcases a notable preference for instrumental elements in certain compositions.

e) K-Means Clustering:

Following the elucidation of principal components, our analysis extended to K-means clustering, a powerful unsupervised learning technique. By employing the Silhouette method, we aimed to discern the most suitable number of clusters, sidestepping arbitrary decisions.

Through the Silhouette method, we discovered that dividing our data into two clusters provided the best results. This means we can categorize our data into two main groups, each containing similar observations. Adding these clusters to our dataset allows us to see how different data points are related and grouped together based on their similarities.

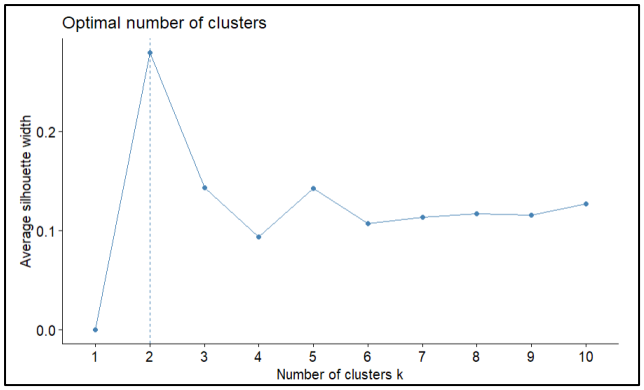

Figure 10: Optimal Clusters using Silhouette Method

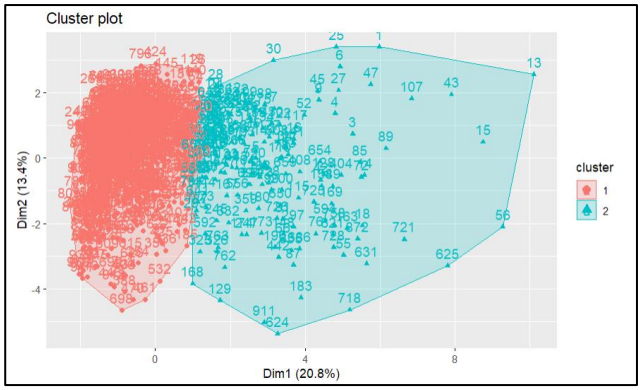

Figure 11: Cluster Representation

With these clusters in hand, we're now equipped to use them as predictive features in our models. By incorporating cluster information into our models, we aim to improve their accuracy and make more informed predictions. This way, we can better understand trends and patterns in our data, ultimately leading to more insightful decision-making.

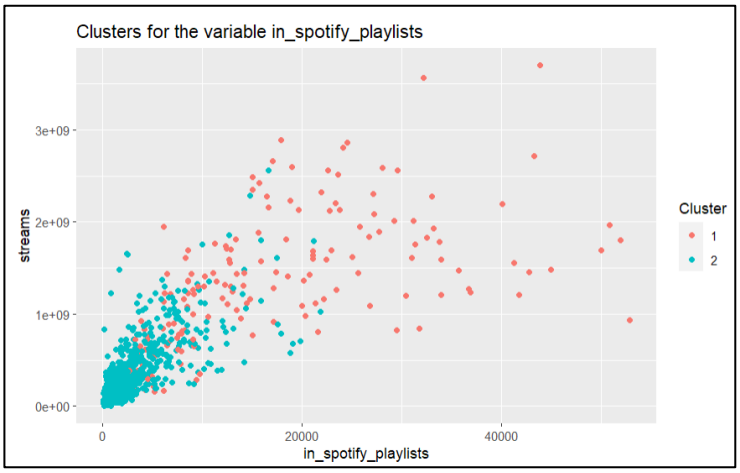

Figure 12: How the Clusters pair with streams?

4. Machine Learning Prediction:

With the completion of our exploratory data analysis (EDA) and the integration of two unsupervised learning methods (PCA and K-Means), I had amassed a wealth of insights crucial for the next phase: predicting streams. Predicting streams involves building a regression model, where I aim to forecast the number of streams based on various features present in our dataset.

Having reviewed a spectrum of available models, I opted for the random forest as it is a versatile and powerful ensemble learning technique renowned for its effectiveness in handling both classification and regression tasks.

The dataset was divided into training and testing subsets using a 70-30 split, ensuring 70% of the data for training and 30% for testing.

Model Training and Initial Evaluation:

Initially, a simple Random Forest model was trained on the training subset, yielding an initial Root Mean Squared Error (RMSE) of 294,296,502 and an R-squared value of 0.8076 on the testing subset. While this model demonstrated promising predictive performance, further optimization was pursued to enhance its accuracy.

Feature Selection Random Forest:

Building upon insights gained from EDA, feature engineering was conducted to refine the feature set and improve model performance. Non-contributing variables such as 'released_day', 'track_name', 'artist_name' and 'released_month were identified and subsequently removed from the dataset. This

refinement contributed to a further reduction in RMSE to 258,673,041 and an increase in R-squared to 0.8389, demonstrating the efficacy of feature selection in enhancing model performance.

Hyperparameter Tuning:

To optimize the model's performance, a systematic hyperparameter tuning approach was employed, focusing specifically on tuning the "mtry" parameter using a for loop. This iterative process resulted in a refined Random Forest model with an improved RMSE of 252,647,662 and an increased R-squared value of 0.8431, indicating enhanced predictive accuracy and model fit.

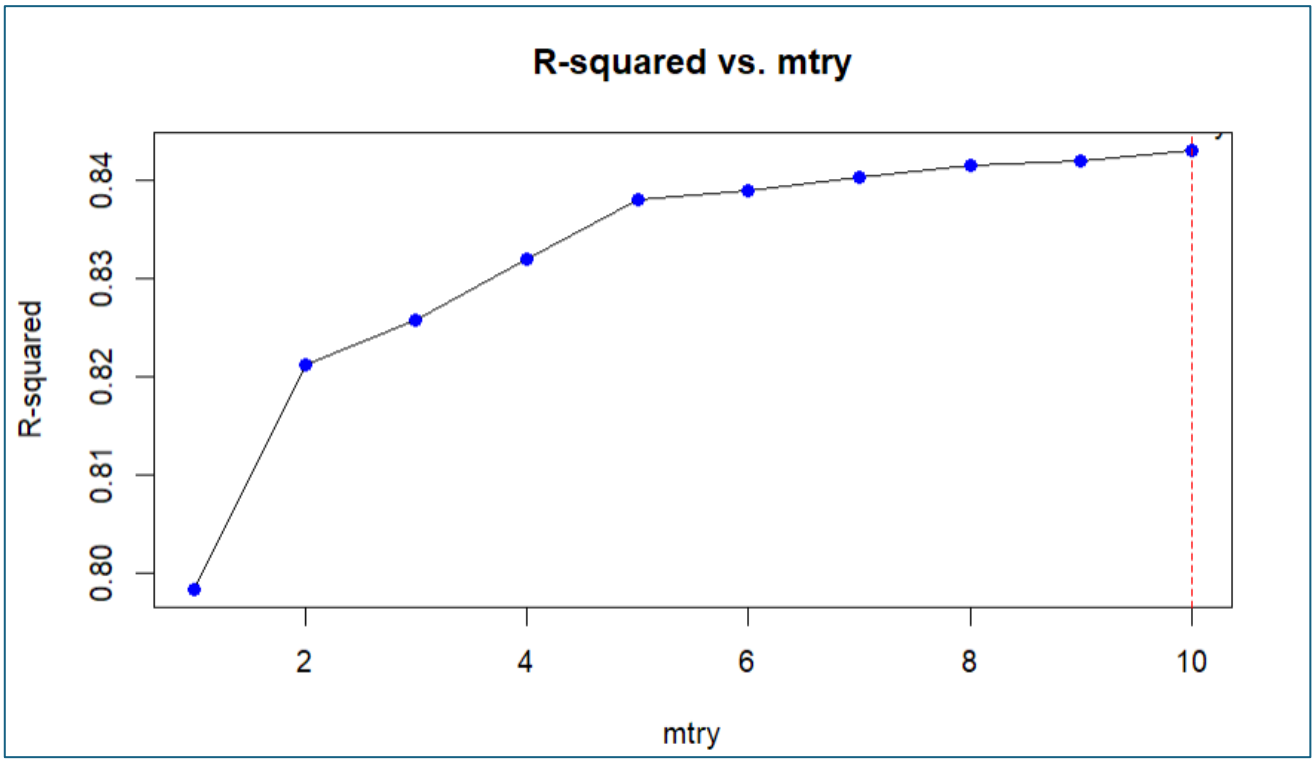

Figure 13: Graph to Determine best "mtry" value

Normalization and Data Transformation:

Additionally, I applied data normalization to ensure uniformity in the scale of features. However, this process did not yield significant improvements, as evidenced by the RMSE of 645,987,881 and an R-squared value of 0.8115.

Principal Component Analysis (PCA):

We explored the use of Principal Component Analysis (PCA) to reduce the dimensionality of our dataset. However, the model performance deteriorated with an RMSE of 337,662,612 and an R-squared value of 0.7113.

One-Hot Encoding:

Finally, we employed one-hot encoding to handle categorical variables and utilized the feature-engineered dataset for model training. This approach resulted in a notable improvement in performance, with an RMSE of 252,856,344 and an R-squared value of 0.8411.

5. High Performance Computational Implementation

Moving forward, the next step is to leverage High-Performance Computational Implementation for further data analysis. In this section, I highlight the use of PySpark in Google Colab as my HPC Implementation. I used PySpark MLlib's Gradient Boosted Tree (GBT) for predicting the number of streams for music tracks.

PySpark Setup:

Using "!pip install pyspark", PySpark was successfully installed. Necessary libraries were also installed and added to facilitate data preprocessing and model prediction tasks. After that, a SparkSession is created to serve as the entry point to PySpark. This session allows for interaction with Spark functionality.

Data Preparation:

The cleaned spotify dataset (from data cleaning) was extracted and is read as csv. Additionally, column names like "track_name" and "artist_name" are dropped from the dataset due to their potentially high cardinality as the next step is to perform One Hot Encoding.

```
categorical_cols = ["released_month", "key", "mode"]

# Perform StringIndexing for categorical columns
indexers = [StringIndexer(inputCol=col, outputCol=col+"_index") for col in categorical_cols]
indexer_pipeline = Pipeline(stages=indexers)
indexed_df = indexer_pipeline.fit(spotify_cleaned).transform(spotify_cleaned)

# Perform One-Hot Encoding
encoder = OneHotEncoder(inputCols=[col+"_index" for col in categorical_cols],
                        outputCols=[col+"_encoded" for col in categorical_cols])
encoded_df = encoder.fit(indexed_df).transform(indexed_df)

# Select relevant columns for model training
selected_cols = ["artist_count", "released_year", "released_day", "in_spotify_playlists",
                 "in_spotify_charts", "in_apple_playlists", "in_apple_charts",
                 "in_deezer_playlists", "in_deezer_charts", "in_shazam_charts",
                 "bpm", "danceability_perc", "valence_perc", "energy_perc",
                 "acousticness_perc", "instrumentalness_perc", "liveness_perc",
                 "speechiness_perc", "cluster"] + [col+"_encoded" for col in categorical_cols]

# Assemble features vector
assembler = VectorAssembler(inputCols=selected_cols, outputCol="features")
assembled_df = assembler.transform(encoded_df)
```

Figure 14: Code Snippet for One Hot Encoding

We start by identifying the categorical columns in our dataset: "released_month", "key", and "mode". These columns likely contain categorical data, such as the month of release, musical key, and mode of the song (major or minor).

For each categorical column, we create a StringIndexer transformation. For instance, let's consider the "released_month" column. The StringIndexer assigns a unique numerical index to each distinct month in the column. For example, January might be indexed as 0, February as 1, and so on.

We repeat this process for each categorical column and store the resulting indexers in the indexers list.

Once the categorical columns are indexed, we proceed with one-hot encoding. This process converts the indexed categorical columns into binary vectors. Continuing with our example, the "released_month" column, after one-hot encoding, would result in binary vectors representing each month's presence or absence in the dataset. The OneHotEncoder transformation is applied to the indexed dataset, generating encoded columns for each categorical feature.
After encoding, we select relevant columns for model training. These include numerical features such as "artist_count", "bpm", and "danceability_perc", as well as the newly encoded categorical columns. We combine these selected columns into the selected_cols list, ensuring that both numerical and encoded categorical features are included.

Finally, we use VectorAssembler to merge the selected columns into a single feature vector column named "features". This feature vector will serve as the input to our machine learning models. For instance, if we have 12 months in the "released_month" column (after one-hot encoding), along with other numerical features, VectorAssembler combines these into a unified feature vector for each data point.

Model Training and Evaluation:
The dataset was partitioned into training and testing subsets using a 70-30 split, with 70% of the data allocated for model training and 30% reserved for evaluation. After the split, two models were trained: baseline model and hyperparameter tuned model.

The baseline model as expected did not give great results. It gave an RMSE of 327,413,150 and R-squared value of 0.7119. On contrary, the hyperparameter tuned model improved the model performance a lot and produced an improved RMSE of 277,541,628 and R-squared value of 0.7930. The parameters that were tuned are “maxDepth”, “maxIter” and “stepSize”.

And finally, to conclude, some model graphs were plotted to see the feature importances and the difference between actual vs predicted streams. This will be discussed more in the upcoming section.

6. Performance Evaluation and Comparison of Methods:

In this section, we evaluate and compare the performance of three different machine learning models: Random Forest (RF), Gradient Boosting Trees (GBT), and Neural Network. Each model was trained and evaluated using the Root Mean Square Error (RMSE) as well as R-squared (R2) metric to assess its predictive capability.

Both RMSE and R2 are essential metrics in evaluating machine learning models. RMSE provides insight into the magnitude of prediction errors, while R2 offers a measure of how well the model fits the data. By considering both metrics, we can comprehensively assess the predictive capability and overall performance of each model.

Hyperparameter Tuned Random Forest Model:
The Random Forest model demonstrates robust predictive capability with an RMSE of 252,647,662 and an R-squared (R2) value of 0.8431. The model's predictions are, on average, within approximately 252,647,662 of the actual values, elucidating that approximately 84.31% of the variance in the target variable is captured.

Hyperparameter Tuned Gradient Boosting Tree Model:

Implemented via HPCI, the Gradient Boosting Trees model achieves an RMSE of 277,541,628 and an R-squared (R2) value of 0.7931. While exhibiting a marginally higher RMSE and a slightly lower R2 compared to RF, it underscores strong predictive performance, maintaining predictions within approximately 277,541,628 of actual values on average, elucidating approximately 79.31% of the variance.

Neural Network Model:

The Neural Network model, developed by group member Sofia, yields an RMSE of 342,705,706 and an R-squared (R2) value of 0.66. Despite the highest RMSE and comparatively lower R2, it offers valuable insights. However, it is noted as the least useful among the models, given its lower predictive accuracy compared to RF and GBT.

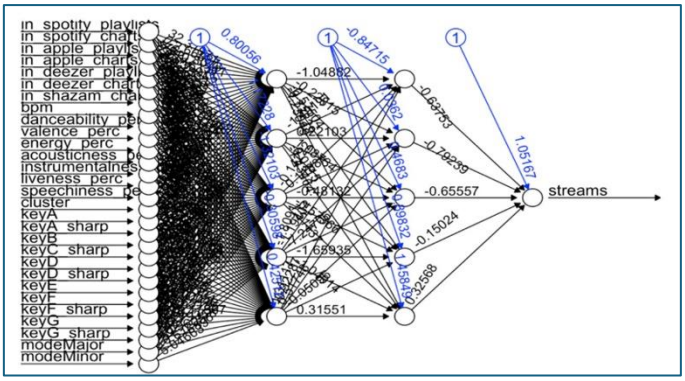

Figure15: Neural Network Model by Sofia

7. Discussion of the Findings:

As we can see from the previous section, Random Forest is the most suitable model to answer the proposed research question, closely followed by Gradient Boosting Tree while Neural Network had highest RMSE of all.

Neural Networks are highly flexible models capable of capturing complex patterns in the data. However, this flexibility can also make them more prone to overfitting, especially when the dataset is relatively small or noisy. If the Neural Network model is overly complex or not properly regularized, it may struggle to generalize well to unseen data, leading to a lower R2 score.

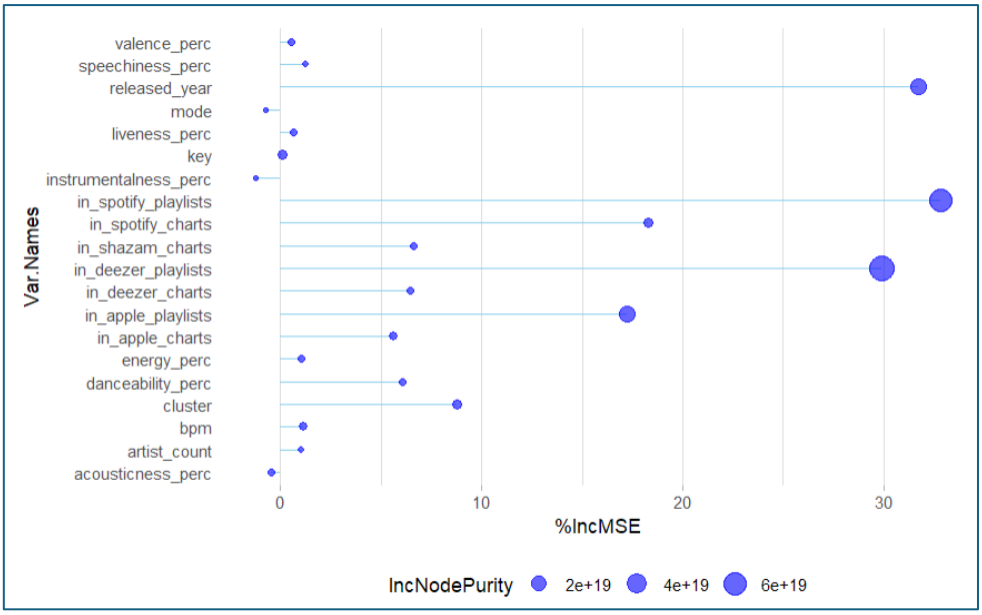

Figure 16: Feature Importance for Random Forest Model

Variables “in_deezer_playlists”, “in_spotify_playlists” and “released_year” are the most significant ones for the random forest model.

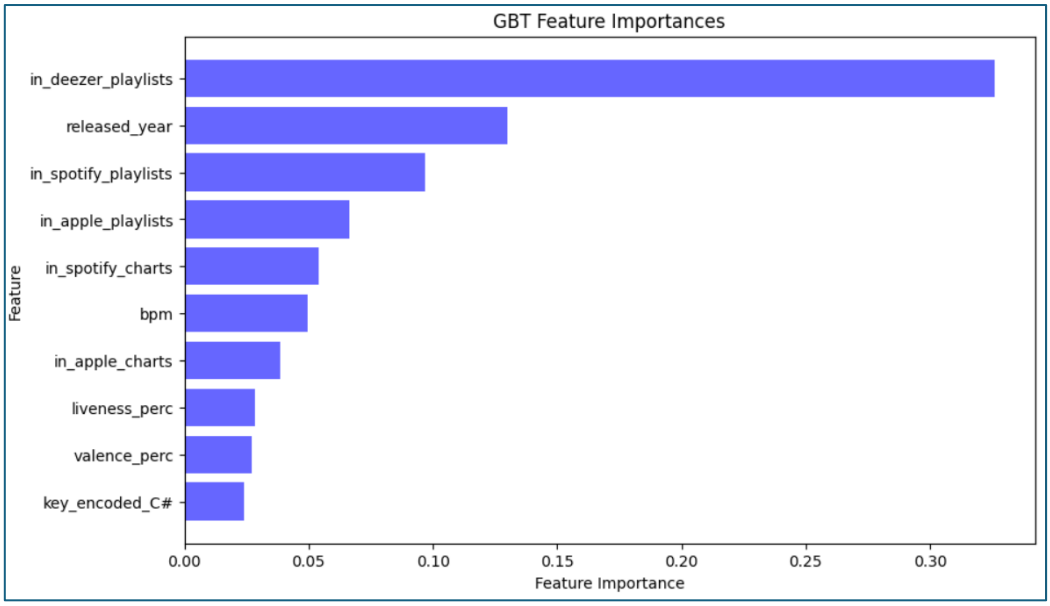

Figure 17: Feature Importance for Gradient Boosting Tree

The order of the variables might have changed but it's still the same top 3 variables. This makes Deezer and Spotify the most trending music platforms and gaining followers on these platforms will contribute towards higher song streams.

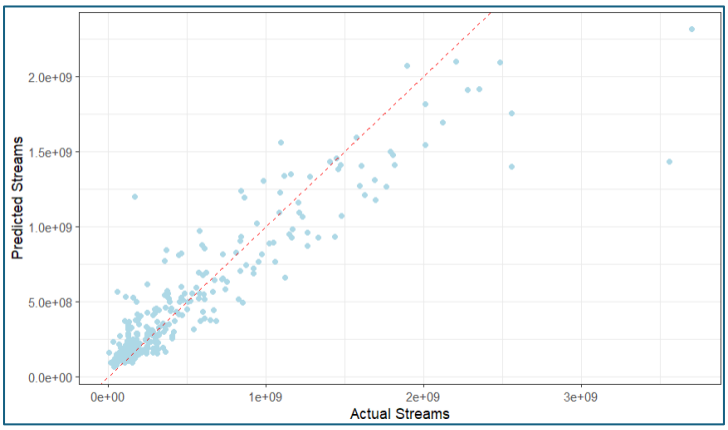

Figure18: Predicted vs Actual Streams for RF

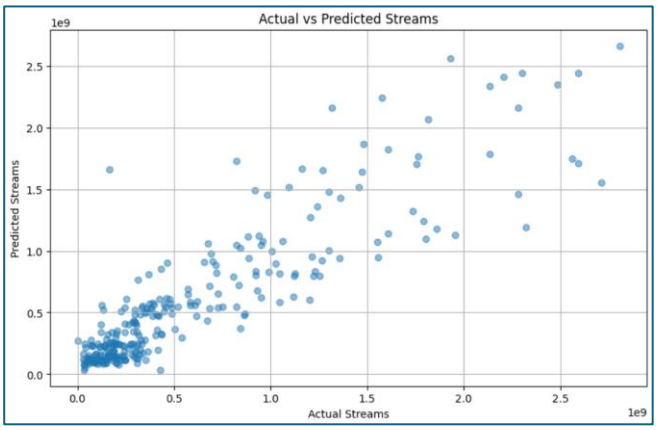

Figure19: Predicted vs Actual Streams for GBT

The Actual vs Predicted Streams plot for Random Forest and Gradient Boosting Tree are very identical. The reason for this can be as both ensemble learning methods that combine multiple decision trees to make predictions.

Limitations:

One limitation of the Random Forest (RF) and Gradient Boosting Trees (GBT) models employed in this study is the potentially higher time required for training, particularly when using high hyperparameters. For RF models, the "mtry" parameter, which determines the number of variables considered at each split, and for GBT models, parameters such as "maxDepth" and "maxIter" controlling the maximum depth of each tree and the maximum number of iterations, respectively, play crucial roles in model performance. Adjusting these hyperparameters to higher values can significantly increase the computational burden during the training process.

While efforts were made to optimize these hyperparameters within the constraints of the dataset, it's important to acknowledge that the selection process can impact both model performance and computational resources. Despite the dataset's relatively small size in this study, the potential for increased training time remains a notable consideration. Moreover, the chosen hyperparameter values might not be universally optimal and could require further refinement for different datasets or problem domains, potentially necessitating additional computational resources and time. Therefore, while RF and GBT models offer robust predictive capabilities, their higher time requirements for training under certain hyperparameter configurations represent a limitation that warrants consideration, particularly in contexts where computational resources are limited, or time constraints are stringent.

8. Data Management Plan and Author Contribution Statement:

Data Management Plan (DMP) is provided in the appendix where detailed guidelines are outlined regarding the handling, storage, and sharing of data. Additionally, the code for data analysis is included in the appendix, enabling readers to replicate and verify the results presented in the report.

Author Contribution Statement:

Sofia Sole Ferrarese (S.S.F.) and Mihir Gujarathi (M.G.) designed the data collection and performed the data cleaning section together. S.S.F. conducted exploratory data analysis (both univariate and multivariate) on the variables: artist_name, artist_count, released_year, released_month, released_day, danceability_perc, valence_perc, energy_perc, acousticness_perc, instrumentalness_perc, liveness_perc, speechiness_perc. M.G. performed exploratory data analysis (both univariate and multivariate) on the variables: in_spotify_playlists, in_spotify_charts, streams, in_apple_playlists, in_apple_charts, in_deezer_playlists, in_deezer_charts, in_shazam_charts, bpm, key, mode. S.S.F. conducted the Principal Components Analysis, while M.G. performed the Cluster Analysis. S.S.F. implemented and applied the Neural Network predictor in R and a qualitative evaluation method in Pig, while M.G. implemented and applied the Random Forest predictor in R and Gradient Boosting predictor in PySpark.