Predicting Song Popularity Using Machine Learning: An Analysis Focused on Children's Music Genre

Morris Simons
Ai and Machine Learning student
Blekinge Tekniska Högskola
Karlskrona, Sweden
Mosi21@student.bth.se

Isak Rulander
Ai and Machine Learning student
Blekinge Tekniska Högskola
Karlskrona, Sweden
Isru21@student.bth.se

***Abstract*—This study utilizes machine learning to predict song popularity within the children's music genre, focusing on the role of audio features in predicting songs success.**

***Index Terms*—song popularity, machine learning, audio features, Childrens music, feature importance**

I. Introduction

Predicting if someone is gonna like a song is difficult and it's something Spotify have been trying to do for almost 2 decades and something Spotify been famous to be doing reasonably good. One of those ways spotify analyses songs for recommending ads and new songs is the sonic profile. With this profile they can better match the feeling of the song to the feeling of the ad creating a less disruptive listing experience.

The main focus of this topic is to see if this sonic profile can be used to predict if a song is going to be popular in their own genre. If successful the model could hopefully be used as an validation layer when creating an music generating deep learning model or for an artist creating their new hit song.

The sonic profile data contains Acousticness, Danceability, Duration (ms), Energy, Instrumentalness, Key, Liveness, Loudness, Mode, Speechiness, Tempo, Time Signature, Valence. More about this in the dataset section.

II. Dataset

This report centers around a comprehensive dataset derived from Spotify, encompassing a wide array of musical genres. Despite Spotify's extensive catalog spanning over 6,000 genres, our study narrows down to a sample of 27 genres. The dataset includes 176,774 unique songs, which, due to the presence of songs classified under multiple genres, extends to 232,725 entries. This phenomenon introduces a form of "semi-duplicates" in our data, exemplified in the table below.

Genre	Artist	Song Title	Spotify ID
Pop	dvsn	Hallucinations	0UE0RhnRaEY
R&B	dvsn	Hallucinations	0UE0RhnRaEY
Soul	dvsn	Hallucinations	0UE0RhnRaEY
Movie	Henri Salvador	C'est beau de	0BRjO6ga9RK

TABLE I: Following sample illustrate the "semi duplicates"

A. Attributes

The dataset's attributes are derived from Spotify's analytical tools, which predict the characteristics of each song using deep learning models. These predictions, while highly accurate, may contain minor errors. The attributes include a variety of metrics, from basic information like genre and artist to more complex measures such as acousticness and danceability. The "popularity" attribute is of particular interest, serving as our key metric despite its potential to fluctuate with trends and listening patterns.

Attribute	**Description**
Genre	The genre of the track
Artist Name	Name of the artist who performed
Track Name	Title of the track
Track ID	Unique identifier for the track
Popularity	Popularity score of the track
Acousticness	The acoustic quality
Danceability	How suitable for dancing
Duration (ms)	Duration of the track in milliseconds
Energy	The energy level of the track
Instrumentalness	Measure of the amount of vocals
Key	The average musical key of the track.
Liveness	Presences of an live audience
Loudness	The overall loudness in decibels
Mode	The modality (major or minor)
Speechiness	The presence of spoken words
Tempo	The tempo in beats per minute(BPM)
Time Signature	The time signature of the track
Valence	The musical positiveness conveyed

TABLE II: Attributes of the Music Dataset

III. Problem Statement

Predicting song popularity poses significant challenges, particularly when relying solely on audio data. Audio features, while informative about the song's musical qualities such as rhythm, melody, and harmony, offer a limited view of a song's potential popularity. This approach overlooks external factors that can greatly influence a song's success. For instance, marketing efforts, social media presence, and cultural trends play crucial roles in shaping listener preferences and, consequently, a song's popularity. Additionally, the timing of a release can align with or against current events or trends, further impacting its reception.

IV. Methodology

A. Dataiku

In order for us to understand our data and experiment with it in an efficient way choose to use Dataiku a Data science platform. This allowed us to create different branches(flows) and in a better way work in a more collaborative way. To train and handle large workloads we deployed Dataiku on a separate more powerful server. By using dataiku in our analytics we could quickly discover outliners, misspellings, duplicates, imbalances. We could also evaluate the distribution of attribute values. Dataiku also provides Analytics tools for most ML models of variable importance. This allowed us to gain an better understanding of the impact of our data prepping solutions, the models performance and a deeper understanding of the data we are working with. Before developing our final version in Jupyter Notebook.

In hindsight we now understand that choosing a dataset with 27 genres proved to be to big because we essentially had to explore 27 different dataset(genres) and it's different characteristics. The discovery of us having to split up our genres like this was only discovered later in the prepping state as a good solution in improving the true accuracy. We still thought the idea of analyzing the multiple of genres as an interesting task to better see where our model actually might be best suitable as a tool for predicting success.

Deploying Dataiku was a huge success in terms of efficiency(time), Compute workload distribution, communication/collaboration and organizing the work and file structure of the project but required a small learning curve.

B. Success metric scope

Early models made us realize that accurately predicting a song's popularity on a 0 to 100 scale was quite challenging. This complexity likely arises from the numerous external factors that can influence a song's success, beyond just its sound qualities, and the inconsistent popularity distribution across different genres.

To tackle these issues, we opted for a binary classification approach, dividing the dataset into two halves: one labeled as "hit" and the other as "miss".

As demonstrated in Fig 7, certain genres exhibit disproportionately high accuracy in our initial models. This imbalance, where some genres almost always count as hits and barely any as misses, really shows why using the same popularity cutoff for all genres doesn't work. Looking at the graph in Fig 2, it's clear that accuracy rates for some genres are high. This happens because of the imbalance in the labeling, which doesn't give us a true picture of how accurate our predictions really are.

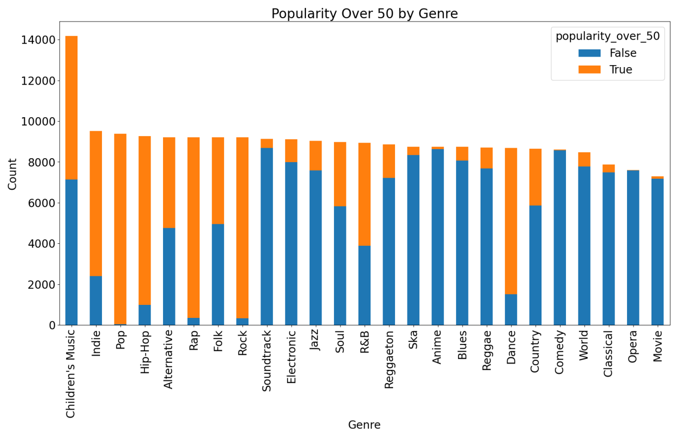

Fig. 1. Children's popularity distribution over and under 50 popularity

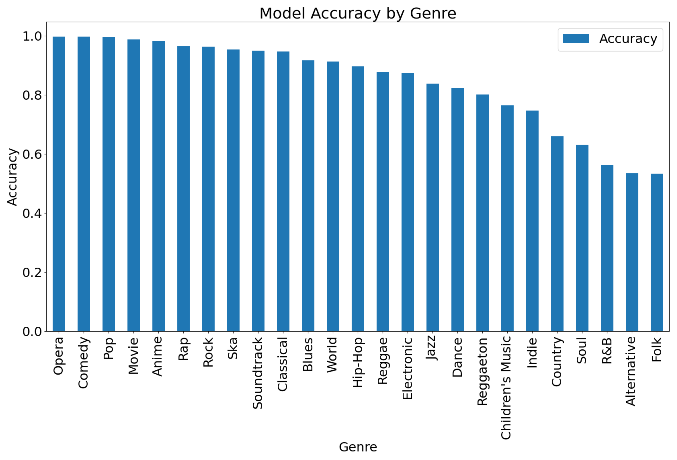

Fig. 2. Accuracy when label is split over and under 50 popularity

To make our analysis more fair, we changed our method to ensure there's an even split between "hits" and "misses" in each genre as shown in Fig 3. This step is crucial for getting rid of the bias present in our earlier classification system.

By adopting this more balanced approach, we're giving the songs in each genre their chance to be popular, it decreases accuracy as in Fig 4, but with the labels giving a more accurate result in terms of it being a good song or not.

This updated approach offers a more authentic reflection of a song's appeal across various genres. In essence this binary system simplifies the task to identifying whether a song is likely to be successful or not, rather than attempting to assign a specific popularity score. The 50/50% split gives us an balanced dataset in a fair way where the best Opera singer don't have to compete with pop musicians such as Taylor swift in popularity score and only compete with others in their own genre.

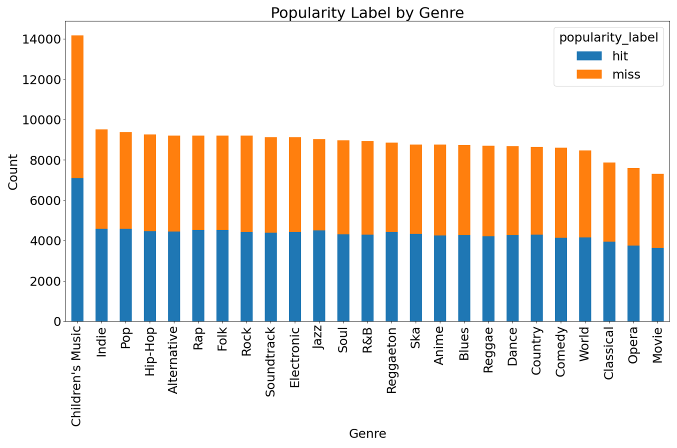

Fig. 3. Children's popularity distribution with 50/50 per genre

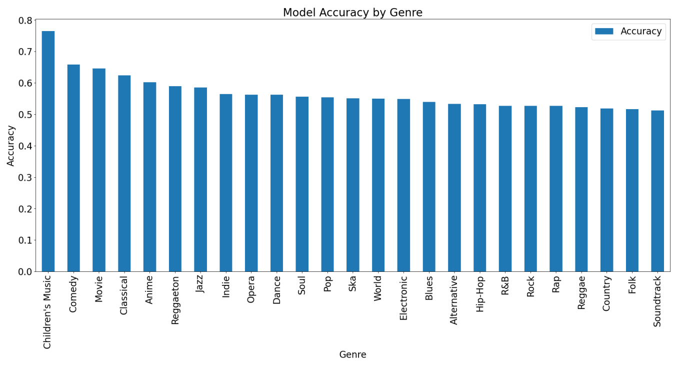

Fig. 4. Accuracy when label is split 50/50 for each genre

C. *Genre Focus*

The decision to concentrate our research on children's music was informed by two main considerations: the abundance of data available for this genre within our dataset and its average popularity score of 50. This average score indicates a broad variance in the popularity of children's music tracks, suggesting a rich field for analysis without the skew towards either high or low extremes common in other genres.

V. Data Preprocessing

Our data prepossessing strategy was essential for optimizing the dataset for machine learning analysis. This section outlines the core steps we executed, focusing on normalization, transformation, and the categorization of the popularity score, based on the actual code from our project.

we added A new sound metric called genre count as a sound metric measuring the appearance of other music genres. As we view hope this value would help our model to distinguish pure children music from mixed songs.

A. *Outliners*

Songs exceeding 500,000 milliseconds (approximately 8 minutes and 20 seconds) were considered too long, while those shorter than 60,000 milliseconds (1 minute) were deemed too short. These thresholds were set to remove tracks that fall outside the usual range of commercial music, thereby focusing our analysis on data that provides a fair representation of typical song popularity factors.

B. *Transforming labels to a numerical scale*

To adapt our dataset for the model, we converted non-integer attributes into numerical formats. The ”key” attribute was changed from strings to a 0-11 scale to match musical key representations. Major and minor modes were coded as 1 and 0, respectively. These adjustments help the model process and learn from our data more effectively.

C. *Feature Scaling*

Standard scaling is applied to enhance model performance by transforming the data to have a mean of 0 and a standard deviation of 1. This ensures that the model does not disproportionately weigh any feature over the others due to differences in scale. By standardizing the features, each one contributes equally to the model's decision process, facilitating faster convergence and improving overall predictive accuracy. It's also useful when your data follows a Gaussian distribution and when using algorithms sensitive to variance in the data, such as Support Vector Machines (SVMs) and Principal Component Analysis (PCA)

D. *Log Transformation*

To tackle the challenge of skewed distributions in some features, we applied log transformation. This method effectively reduced the skewness, rendering the distributions more symmetric and therefore better suited for the model. This improved the performance significantly with almost 10% from 76% to 86% accuracy. By just doing log transformation on liveness and speechiness. The log transformation formula is $\log(1 + x)$, with $1+ x$ to handle cases with 0.

E. *Popularity Score Treatment*

Instead of removing records with missing or outlier values, our focus shifted to the treatment of the popularity score. We explored the dataset's inherent structure and decided against a binary categorization of popularity. Recognizing the nuanced nature of song popularity, which does not neatly fit into 'Hit' and 'Miss' categories, we retained the continuous nature of the popularity score. This decision allowed us to capture the full spectrum of song popularity, facilitating a more detailed and nuanced analysis.

F. *Feature Reduction and Selection*

PCA, commonly applied in unsupervised learning, can also be used in supervised learning contexts. PCA combines attributes with high correlation, thereby reducing the dimensionality of the data while retaining as much of the variability as possible. Figure 5 Shows an confusion matix, with only 4 values above or below 0.7 witch is a pretty low correlation. This make us conclude that PCA is generally not needed but we will experiment with it anyway to measure it's impact.

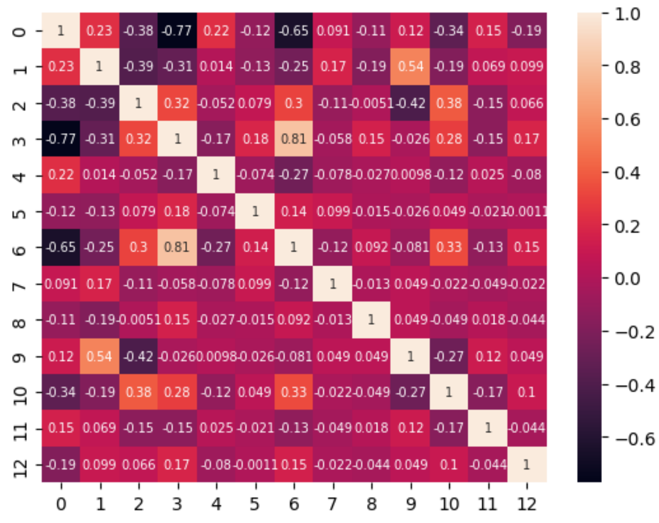

Fig. 5. Children's popularity distribution

VI. Conclusion

Through these preprocessing steps, we prepared our dataset for a detailed analysis that respects the complexity of musical popularity. By normalizing feature scales and addressing distribution skewness without discarding data prematurely, we maintained the richness of our dataset. Our treatment of the popularity score as a continuous variable reflects a commitment to capturing the nuanced dynamics of song appeal, setting the stage for insightful model development and analysis.

VII. Metric

The selected metrics for evaluating our model are accuracy and AUC-ROC. Accuracy is chosen for its simplicity, offering a direct measure of correct predictions. AUC-ROC complements this by assessing the model's ability to distinguish between classes, providing a deeper insight into its predictive performance.

VIII. Feature importance

Feature	**Importance**
Genre Count	0.413
Loudness	0.095
Acousticness	0.087
Energy	0.078
Valence	0.070
Danceability	0.064
Speechiness	0.048
Tempo	0.047
Liveness	0.043
Instrumentalness	0.039
Mode	0.012
Time Signature	0.004

TABLE III: Updated Feature Importances of childrens music

After evaluating the feature importance on an early model, we can see that Genre Count has a big impact. And is measuring the impact we intended to meet referring to meet in the data processing section.

IX. Model Development

In our analysis, we evaluated the performance of six distinct machine learning models to determine which yields the highest metric in predicting outcomes. We focused mainly on the accuracy of results, but we report the precision and recall as well. False positive predictions also gives a misleading guidance for artists.

A. *RandomForestClassifier (RFC)*

Was used as an benchmark model throughout the project the model is known for great result most of the time but also it's functions such as feature importance. The optimization includes **n_estimators**: The choice of [100, 200, 250, 300, 400, 500] for the number of trees in the forest is sensible. More trees generally improve model performance but also increase computational cost. **max_depth**: Including options like None (unlimited depth), 10, 20, and 30 provides a good range for controlling the depth of the trees. This affects both the model's ability to learn detailed patterns and its risk of overfitting. **min_samples_split:** Offering values [2, 5, 10] allows the model to explore different minimum sizes for node splitting, affecting how detailed the learned patterns are. **min_samples_leaf**: With values [1, 2, 4], this parameter helps in preventing the model from learning too specific patterns, which could lead to overfitting. **bootstrap:** Including both True and False explores the use of bootstrap samples when building trees (True) versus using the whole dataset to build each tree (False), affecting the model's variance and bias

B. *Support Vector Machines(SVM)*

Was selected for its ability to find the optimal decision boundary. Based on the data points closest to the decision boundaries, creating support vectors. The optimization includes chooseing **C** value of np.logspace(-4, 4, 9), '**linear**', **'rbf'** (**radial basis function**) and **sigmond** covering the most commonly used kernels except **poly** because of its high computation cost of longer than 8 minutes in our case. Having both '**scale**' and '**auto**' options allows the model to adjust the kernel coefficient in relation to the features' scale or use a fixed value, respectively, which can significantly affect performance.

C. *XGBoost (XGB)*

Stands for eXtreme Gradient Boosting, is an advanced and efficient implementation of gradient boosting that has gained popularity for its speed and performance. This can be used to push the workload on the **graphics processing unit (GPU)** allowing for better performance(this was used but rolled back to allow to make the code more platform friendly).The optimization includes **n_estimators**: The range [100, 120, 200, 300, 400, 500] offers a wide spectrum for the number of gradient boosted trees. This allows fine-grained control over the model's complexity and capacity to learn from the data. **learning_rate**: Including values like 0.01, 0.04, 0.05, 0.1, 0.2 provides options for both slow and more aggressive learning. This is crucial for balancing model accuracy and over-fitting. **max_depth**: The range [3, 4, 5, 6, 7, 8] allows the model to explore various levels of interaction depth. Higher values enable the model to capture more complex patterns but increase the risk of overfitting. **min_child_weight:** The options [1, 5, 10] give the model flexibility in determining the minimum sum of instance weight. This parameter is important for controlling overfitting. **gamma**: With values [0.5, 1, 1.5, 2, 5], you're allowing the model to adjust the minimum loss reduction required to make a further partition on a leaf node. This helps in pruning and managing the model's complexity. **subsample**: Including [0.6, 0.8, 1.0] lets the model experiment with the fraction of samples to be used for each tree, influencing variance and bias. **colsample_bytree**: The range [0.6, 0.8, 1.0] offers variation in the fraction of features to be used for each tree, impacting model performance and overfitting.

D. *LogisticRegression (LR)*

Logistic Regression is a popular statistical method and machine learning algorithm used for binary classification tasks, where the goal is to categorize data into one of two groups based on a set of independent variables. It's particularly well-suited for problems where you want to predict the probability of a binary outcome like in our case whit hit or miss. **C:** The range provided by np.logspace(-4, 4, 20). A wider range ensures that both very small and very large values are considered, helping to balance the bias-variance trade-off. **penalty:** Including both l1 and l2 penalties allows the search to evaluate both the Lasso (L1) and Ridge (L2) regularization techniques. This choice can significantly impact model performance, especially in the presence of irrelevant features or multicollinearity. **solver:** The choice of ['liblinear', 'saga'] is appropriate since liblinear is optimized for small datasets and saga is a good choice for large datasets and supports both l1 and l2 penalties efficiently.

max_iter: Providing options for the maximum number of iterations [100, 1000, 10000] allows the model to converge on datasets of varying complexity and size.

E. *Decision Tree Classifier (DT)*

A Decision Tree Classifier is a simple yet powerful machine learning algorithm used for both classification and regression tasks, DTs can capture non-linear relationships without needing transformations or assumptions about the distribution of data make it an interesting contribution to our project. For optimization we went whit **max_depth:** Offering a range from None (unlimited) to specific depths (10, 20, 30, 40, 50) is a good strategy. This allows the search to explore both shallow and deep trees, balancing between underfitting and overfitting.**min_samples_split:** Using np.arange(2, 11) gives a fine-grained control over the minimum number of samples required to split an internal node. This parameter helps in controlling the tree's depth and complexity. **min_samples_leaf:** Similar to min_samples_split, np.arange(1, 11) for min_samples_leaf offers nuanced control over the tree's growth, ensuring that leaf nodes have a minimum number of samples and thus reducing overfitting. **max_features**: Combining ['sqrt', 'log2', None] with a range up to the number of features (X_train_normal.shape[1] + 1) provides a broad spectrum to explore how the number of features considered at each split affects the model's performance. This can influence the diversity of the trees and help in finding the right balance between bias and variance. **criterion:** Including both gini and entropy allows the model to explore different measures of quality for splits. This choice can impact the tree's structure and ultimately the model's accuracy.

F. *KNN classifier (KNN)*

The K-Nearest Neighbors (KNN) classifier is a simple, yet powerful machine learning algorithm used for both classification and regression tasks, but it is most widely known for its application in classification. Since KNN makes no assumptions about the form of the data distribution, it is considered non-parametric. This flexibility allows it to capture complex decision boundaries.For optimization we went whit **n_neighbors:** The choice to explore a wide range from 1 to 200 for n_neighbors is comprehensive. It allows for examining the effect of both very small and very large neighborhoods on the model's performance. Smaller values make the model more sensitive to noise in the training data, while larger values tend to smooth the decision boundary, potentially leading to underfitting. **weights:** Offering both 'uniform' and 'distance' options allows the search to evaluate the impact of treating all neighbors equally versus weighting them by the inverse of their distance. This choice can significantly affect performance, especially in datasets with varied densities. **p:** Including both 1 and 2 lets you compare the Manhattan distance (L1) and the Euclidean distance (L2). This can influence how the similarity between instances is calculated, affecting the model's decision boundaries.

X. Evaluation and Validation

When finding the optimal hyper parameters we opted to use Random search, We attempted to use **Bayesian Optimization** a probabilistic model to guide the search for the best parameters. But the results proved to not as desirable as compute cost and results was not preforming well. We then tried **Grid search** an be very time-consuming, especially as the number of parameters and their possible values increase. Due to the high computial cost this to was abandoned. This led us to conclude that **Random Search** was the optimal method for finding the right parameters because of it's low compute cost. We used accuracy as the optimization scoring. **5-fold cross-validation** 5-fold cross-validation is a popular choice because it offers a good balance between computational efficiency and the reliability of the performance estimate. Using 5 folds means the data is divided into 5 parts, with each part serving as the validation set once and as part of the training set 4 times. This setup is often enough to get a reliable estimate of the model's performance without being as computationally demanding as higher folds, like 10-fold cross-validation. **Number of parameter settings sampled** Setting **n_iter** to 50 in a randomized search is an attempt to strike a balance between exploring the parameter space and keeping the computational cost within reasonable limits. While increasing n_iter would provide a more thorough search of the parameter space, it also increases the time and computational resources needed. 50 iterations allow for a broad enough exploration to identify good parameter settings without excessive computational demands. 75 candidates X 5 cross fold = 375 fits. 5 Models X 375 fits each gave 1875 + 180 fits for SVC, totaling 2055 model fits.

XI. Results

result on metric result on classifier Log tranformation benchmarks PCA benchmark final result

The best result that we got was with PCA data on a SVC model with an accuracy of 85.54% trained with k-fold. This shows that even if 4 only pairs showed an correlation greater than 0.7 and PCA was not recommended this proved to be better than any hyper parameter tuning. But with the normal data on default models without hyper tuning parameters we SVC was the winner with 86.147%.

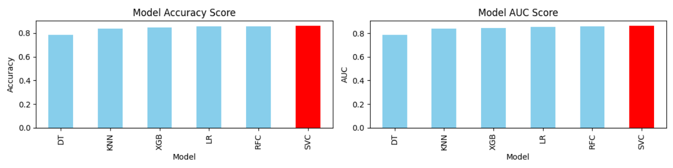

Fig. 6. Results on models without hyper parameter tuning

After hyper parameter tuning we saw a new winner RFC a model that climbed from the fifth best model to the best model with an accuracy 0.857267% showing an small increase in performance. The bench marking of model was done with 10 k fold validation this time due to the smaller amount of models needed to be fitted, 10 K fold allows for greater accuracy.

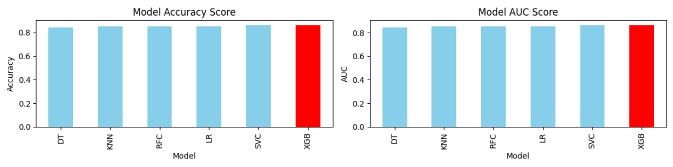

Fig. 7. Results on models

XII. Discussion

Our investigation into predicting song popularity with machine learning has illuminated the complex interplay between audio features and a song's success. While our models have shown promise, particularly with the genre-specific binary classification approach, the study also highlights the inherent challenges in such predictions. The variability in genre-specific popularity underscores the limitations of a one-size-fits-all model and points to the significant impact of external factors—like market trends and social media—on a song's popularity.

The data was also very limited to the vast amount of data that spotify, to get an more accurate prediction of an song more data is needed. Our genre was Also very influenced from outside genres as shown in our feature importance. You could argue that the metric should be removed because it prevent what we tried to do with our attempt to split up the genres to avoid comparing Mozart to Taylor swift by labeling songs that are not soly children's music as popular and music that is truly children's music as non popular. But does that mean you have to remove all songs that actually are children's music but with multiple labels of different genres.

The nuanced approach to data preprocessing and the adoption of binary classification for a fair analysis within genres have significantly improved our model's performance. However, the fluctuating nature of musical tastes and the influence of external factors beyond the dataset suggest areas for further exploration. Integrating data on marketing efforts, social media influence, and even global events could offer a more rounded understanding of what drives song popularity.

Future research could also benefit from lyrics analysis, deep learning models to analyze the Spectograms to capture melodies and sound patterns, or tools to capture what instruments are being used in the song. We did some experimenting with lyrics by with just the length of the lyrics showed to have more impact than any other value in our dataset.

In conclusion, while our study advances the understanding of song popularity prediction, it also opens up new avenues for research. The intersection of music, technology, and data science remains a fertile ground for innovation, with potential benefits extending across the music industry.

XIII. Conclusion

To sum up, our exploration into using machine learning for predicting song popularity within the Spotify dataset has revealed both the potential and challenges of this approach. By fine-tuning our data preprossessing methods and adopting a genre-specific binary classification system, we have made significant strides in enhancing model accuracy. Nonetheless, the influence of external factors on song popularity, beyond the scope of our dataset, indicates the complexity of this domain. Future work, incorporating broader datasets and exploring advanced modeling techniques, will be vital in furthering our understanding of the digital music landscape. This endeavor not only contributes to the academic field but also offers practical insights for artists, producers, and platforms aiming to navigate the ever-evolving world of music and could maybe be used as an validation layer for deep learning algorithms. You could also argue that the evaluation could have been done better by calculating the average rank of each model and use that average rank to compare what model is best similar to the work done in Assignment 2 in [**?**].

References

@miscPitchClassWiki, title = Pitch class, howpublished = https://en.wikipedia.org/wiki/Pitch_class, note = Accessed: 24-02-12

@manualSpotifyAPI, title = Spotify for Developers, organization = Spotify, howpublished = https://developer.spotify.com/documentation/web-api, note = Accessed: 24-02-12

@miscSpotifyTracksDB, author = Zaheen Hamidani, title = Ultimate Spotify Tracks DB, howpublished = https://www.kaggle.com/datasets/zaheenhamidani/ultimate-spotify-tracks-db, note = Accessed: 24-02-12

@miscBBCNewsEntertainment, title = BBC News - Entertainment & Arts, howpublished = https://www.bbc.com/news/entertainment-arts-67111517, note = Accessed: 24-02-12

@miscSpamdataset, author = Morris Simons title = Assignment 2 - DV2599 Maskininlärning code, year = 2024 howpublished = https://github.com/MorrisSimons/SCH_DV2599-Maskinlarning/tree/master/Assignment%202, note = Accessed: 24-02-12